DSP Animal Classifier

Efran Fernandez Fernandez, 293866, group (10/2026, 3:22 pm)

12 stycznia 2026

1 Introduction

The basic idea of this project is that there are differences in the DSP features extracted from animal sound audio files.

The objective of this project is to demonstrate that it is possible to classify animal sounds only based on some features extracted from the audio itself.

In this project a lot of concepts from the DSP class are used, like Fourier transform, resampling, time-domain analysis, spectral features and band energies.

2 Dataset and signal preprocessing

The data set used was ESC-50, a pretty popular data set for enviroment sound classification [1]. In the final project only 3 animals are being used, although the dataset provides 10 animal classes. This was done to keep things simple and to get better accuracy in the classification part of the work. The animals chosen were dog, cat, and crow, their audio files can be found in the data folder, 40 audio files for each animal, 120 clips total.

The preprocessing of each clip was done with matlab, it can be found in the ”matlab/” folder of the project with the name ”dataset generator”.

1. I load the audio with the function ”audioread” and convert the signal to mono, doing the mean between the two channels.
2. Resample the signal to 16 kHz so they are comparable to each other and I get a good running speed.
3. I normalize the amplitude, avoiding volume differences dominating the features.

3 Feature extraction

In this part of the project I take every audio clip and I change it into a small vector of numbers. These numbers are called *features*. Later the classifier in Python uses only these features, not the full audio signal.

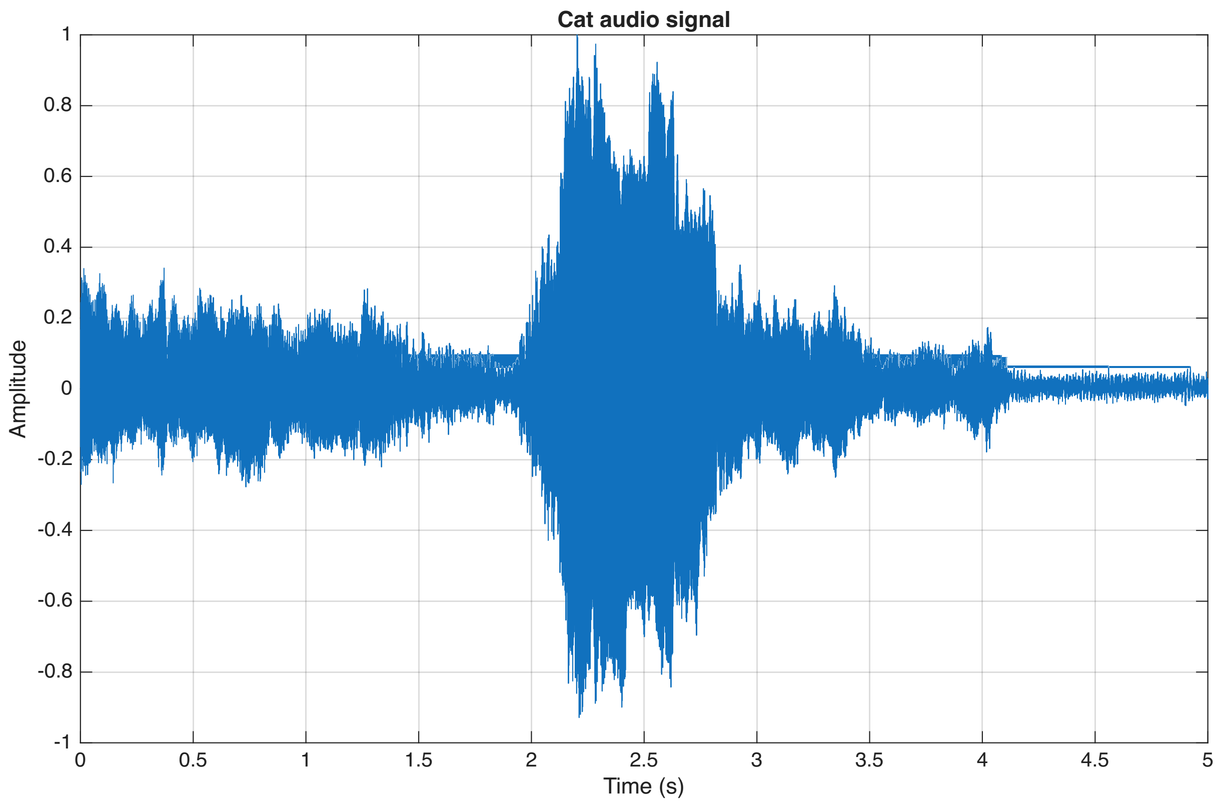

Figure 1: Time domain representation of one of the cat audio clips.

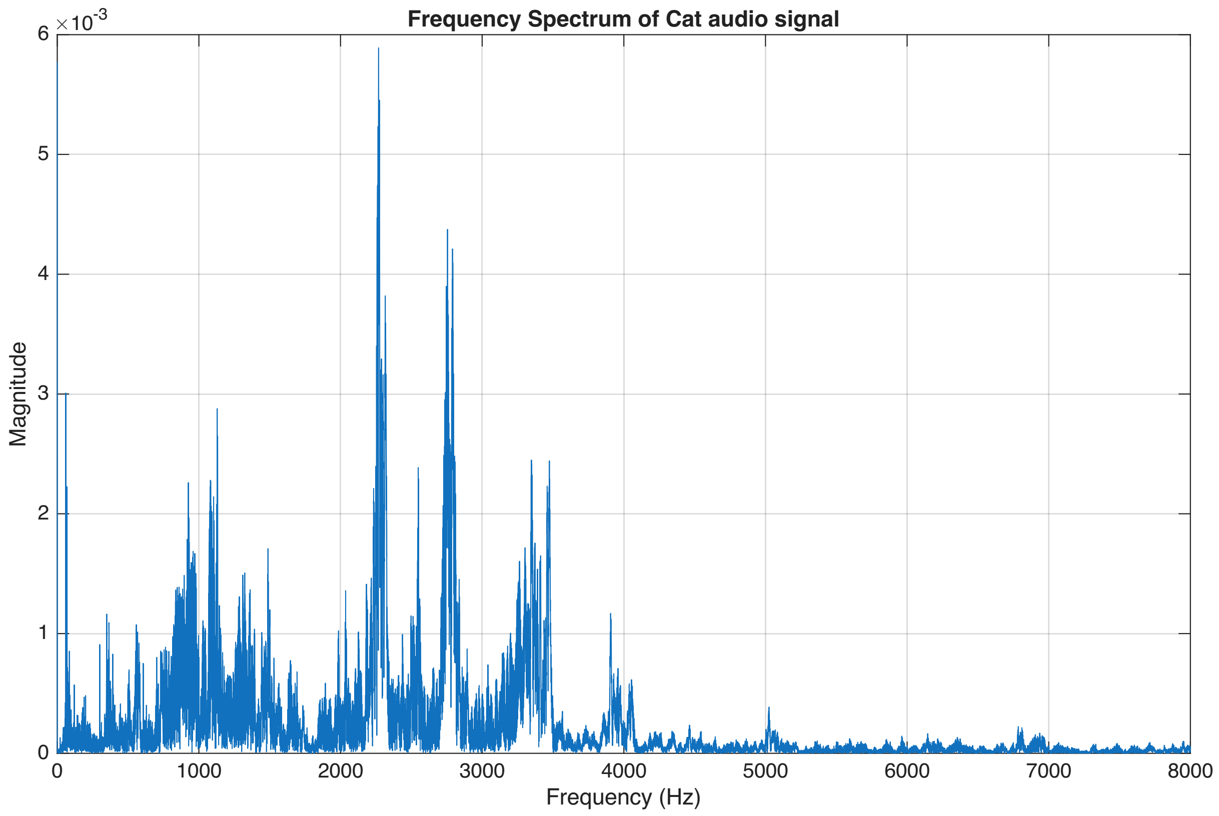

Figure 2: Frequency domain representation of one of the cat audio clips.

All clips are already mono, resampled to 16 kHz and normalised (see previous section). For each clip I use one MATLAB function called `extract_features.m`. It returns a row vector

$$\Big[\text{RMS},\ \text{ZCR},\ \text{Centroid},\ E_{\text{low}},\ E_{\text{mid}},\ E_{\text{high}}\Big].$$

So each sound is described by six numbers.

3.1 RMS energy

The first feature is the RMS (root mean square) value of the signal. For a discrete signal $x[n]$ with N samples it is

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x[n]^2}. \tag{1}$$

RMS is related to the average power of the signal. Because all clips are normalised, RMS is not about absolute loudness, but it still shows if the sound is more active or more quiet in time.

3.2 Zero-crossing rate (ZCR)

The second feature is the zero-crossing rate. It counts how many times the signal changes sign (from positive to negative or the other way) in one second.

In MATLAB I first take the sign of the signal using the function `sign(x)` and then I count the sign changes. Finally I divide the number of changes by the signal duration in seconds.

3.3 Spectral centroid

Next I look at the spectrum of the signal. I compute the FFT of $x[n]$ and I keep only the positive frequencies. Let $X[k]$ be the magnitude spectrum and $f[k]$ the frequency in Hz for bin k.

The spectral centroid is

$$\text{Centroid} = \frac{\sum_k f[k]\ |X[k]|}{\sum_k |X[k]|}. \tag{2}$$

It tells us where the "centre of mass" of the spectrum is. If the sound has more high frequencies, the centroid is higher. If it has more low frequencies, the centroid is lower. This is useful because different animals have different typical frequency ranges.

3.4 Band energies

The last three features are the energy in three simple frequency bands. After I have the magnitude spectrum $|X[k]|$, I square it to get an energy estimate and I sum it inside each band:

- low band: from 0 to 1 kHz,

- mid band: from 1 to 4 kHz,
- high band: from 4 kHz to Nyquist (8 kHz for 16 kHz sampling).

This gives three values: E_{low}, E_{mid} and E_{high}. Some animals have more energy in the low band, some in the mid or high band, so these numbers help to separate the classes.

3.5 Summary of feature vector

To sum up, for each audio clip I:

1. read the signal and normalise it,
2. compute RMS and ZCR in the time domain,
3. compute the FFT,
4. calculate the spectral centroid,
5. calculate energy in low, mid and high frequency bands.

The result is a 1×6 feature vector for each clip. All these vectors are saved to a file `features.csv` and later used in Python to train and test the classifier.

4 Classification in Python

In this part of the project I use Python to classify the animal sounds. The input to Python is the file `features.csv` that was created in MATLAB.

Each row in `features.csv` is one audio clip. There are 3 animals (cat, crow, dog) and 40 clips per animal, so in total 120 rows. For every clip I store:

- six feature columns: RMS, ZCR, spectral centroid, E_{low}, E_{mid}, E_{high},
- one label column with the animal name.

4.1 Data preparation

In Python I first load the CSV file with `pandas`. Then I split the data into training and test sets using the function `train_test_split` from `scikit-learn`. The test size is 30%, so there are 84 training clips and 36 test clips. I use *stratified* splitting, so every animal keeps the same proportion in train and test sets.

Before training the classifier I standardise the features. I use the class `StandardScaler`. On the training set it subtracts the mean and divides by the standard deviation for each feature. Then I apply the same transformation to the test set. This step makes all features have similar scale.

4.2 Classifier

As a classifier I use k-nearest neighbours (k-NN) from `scikit-learn`. I choose $k = 5$. For a new feature vector the algorithm looks at the 5 closest training points (in feature space) and takes the majority label.

I train the classifier with the standardised training data and the labels. After that I predict the labels for the test set and compare them with the true labels.

4.3 Results

For $k = 5$ I obtain the following accuracy on the test set:

$$\text{accuracy} = 63.89\%.$$

The confusion matrix (rows = true class, columns = predicted class) is:

Tabela 1: Confusion matrix for $k = 5$.

	cat	crow	dog
cat	10	2	0
crow	7	4	1
dog	1	2	9

From Table 1 we can see that:

- cat clips are classified correctly in 10 out of 12 cases,
- dog clips are also recognised quite well (9 out of 12),
- crow clips are the hardest class and are often confused with cat.

Even with only six simple DSP features, the classifier can separate the three animals noticeably better than random guessing.

5 Conclusion

In this project I built a small system that tries to recognise animals from their sounds using basic DSP tools. First I prepared the audio clips in MATLAB (mono, 16 kHz, normalised). Then I extracted six features for each clip: RMS, ZCR, spectral centroid and three band energies. These features were saved to a `.csv` file and used in Python with a k-nearest neighbours classifier.

For the three animals (cat, crow and dog) the system reached about 64% accuracy on the test set. Cat and dog were recognised better, while crow was more often confused with the other classes.

The results are not perfect, but they show that even very simple DSP features already give some useful information to separate different animal sounds. With more features or more data it should be possible to improve the classification in the future.

References

[1] Karol Piczak. Esc-50: Dataset for environmental sound classification. https://github.com/karoldvl/ESC-50. Accessed: 2026-01-10. 1