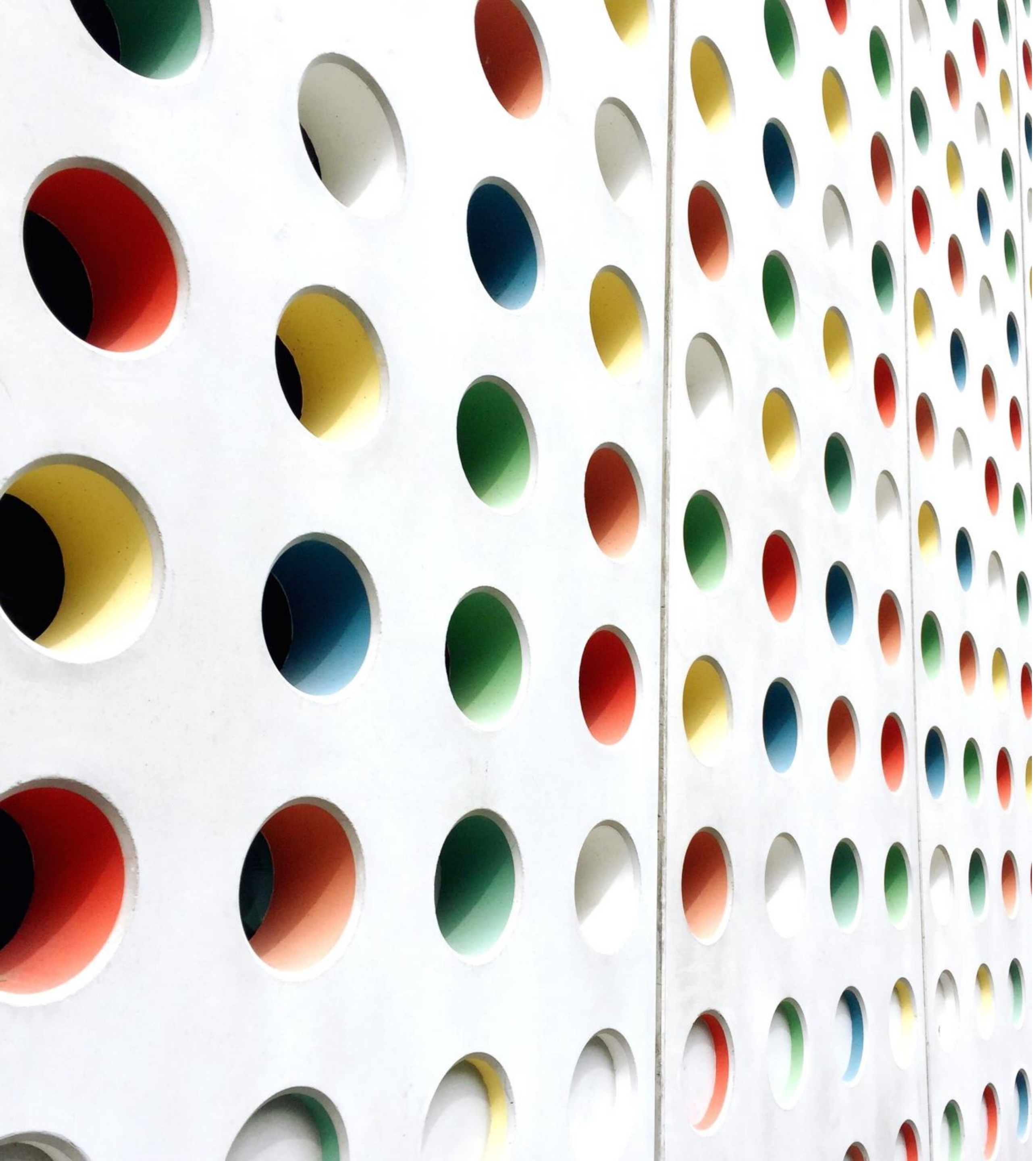


Single Class Classification with Autoencoders

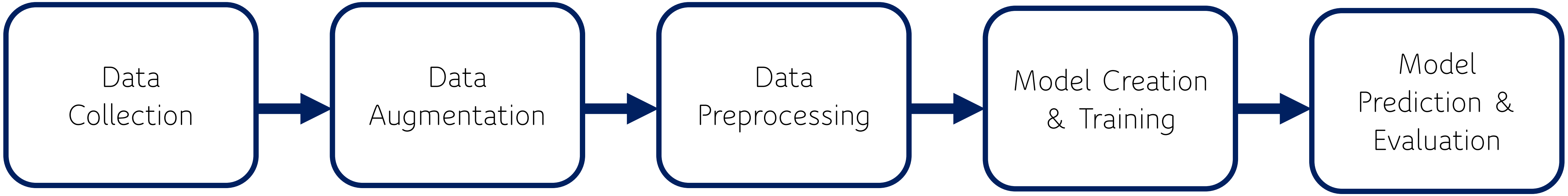
COUGH IDENTIFICATION

ABHISHEK BHATTACHARYA

Objective

To create an ANN based model which can accurately identify 'Human Cough' sounds

Approach


Data Collection
Data Augmentation
Data Preprocessing
Model Creation & Training
Model Prediction & Evaluation

Data Collection

- 250 cough audio files collected from various sources on the internet
- For the purpose of creating the model, the audio files were converted to .WAV format using https://www.online-convert.com/
- The length of the audio clips vary based on the length of the cough bout

Data Augmentation

- A dataset of 250 audio files may not be sufficient to successfully train a classification model
- Hence, the below 'Audio Data Augmentation' techniques were applied to increase the size of the dataset to 1000 :
  - Noise Injection - Add some random value into data by using NumPy
  - Shifting Time - The idea of shifting time is very simple. It just shift audio to left/right with a random second. If shifting audio to left (fast forward) with x seconds, first x seconds will mark as 0 (i.e. silence). If shifting audio to right (back forward) with x seconds, last x seconds will mark as 0 (i.e. silence)
  - Changing Speed (Stretching) - This augmentation stretches times series by a fixed rate
- A supporting class .csv file is created with the following columns:
  - Filename
  - Class Name (in this case all audio files are of the single class – 'Cough'

file_name	**class_name**
sample(1).wav	cough
sample(2).wav	cough
sample(3).wav	cough

Read:
https://medium.com/@makcedward/data-augmentation-for-audio-76912b01fdf6

Data Preprocessing

Extract and Normalize MFCCs → Extract Class Name from .csv → Join MFCCs and Class Names and Convert into Pandas Dataframe → Convert features and corresponding classification labels into NumPy arrays → Split the dataset into training and testing sets

Read:
https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd

Model Creation : Anomaly Detection for Single Class Classification

- The process of identifying outliers in a dataset is generally referred to as anomaly detection, where the outliers are “anomalies,” and the rest of the data is “normal.”
- One-Class Classification involves fitting a model on the “*normal*” data and predicting whether new data is normal or an outlier/anomaly
- A one-class classifier is fit on a training dataset that only has examples from the normal class. Once prepared, the model is used to classify new examples as either normal or not-normal, i.e. outliers or anomalies
- In our model, the cough data is the “normal” data and the intent is to predict if a new audio is Cough or ‘Not-Cough’

Read:
https://machinelearningmastery.com/one-class-classification-algorithms/

Model Creation : Autoencoders as a means for Anomaly Detection

- Autoencoders (AE) are neural networks that aim to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation
- Autoencoder neural networks are commonly used for dimensionality reduction in computer vision to natural language processing.
- Since we are trying to reproduce the input with this model, the loss function that suits the best is **'Mean Square Error' (MSE)**.
- While training the model, what it tries to do is to minimize the MSE. To minimize MSE it should try to fit the dataset much as possible i.e. it should try to reproduce many data as possible

Read:
https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f
https://towardsdatascience.com/autoencoder-neural-network-for-anomaly-detection-with-unlabeled-dataset-af9051a048

Model Creation : Autoencoders as a means for Anomaly Detection

- While training the model it learns how the features will look like for normal data and compress it into a small element and decode it back as the input with a small error. When an anomaly is sent through the model, it will fail to reproduce it, since it is trained to reproduce only normal data and will end up with a large MSE
- In our case, we would calculate the MSE of the output compared to input and predict whether the input audio is a 'Cough' or 'Not-Cough'
- The Autoencoder we would use is called a '**Sparse Autoencoder**'

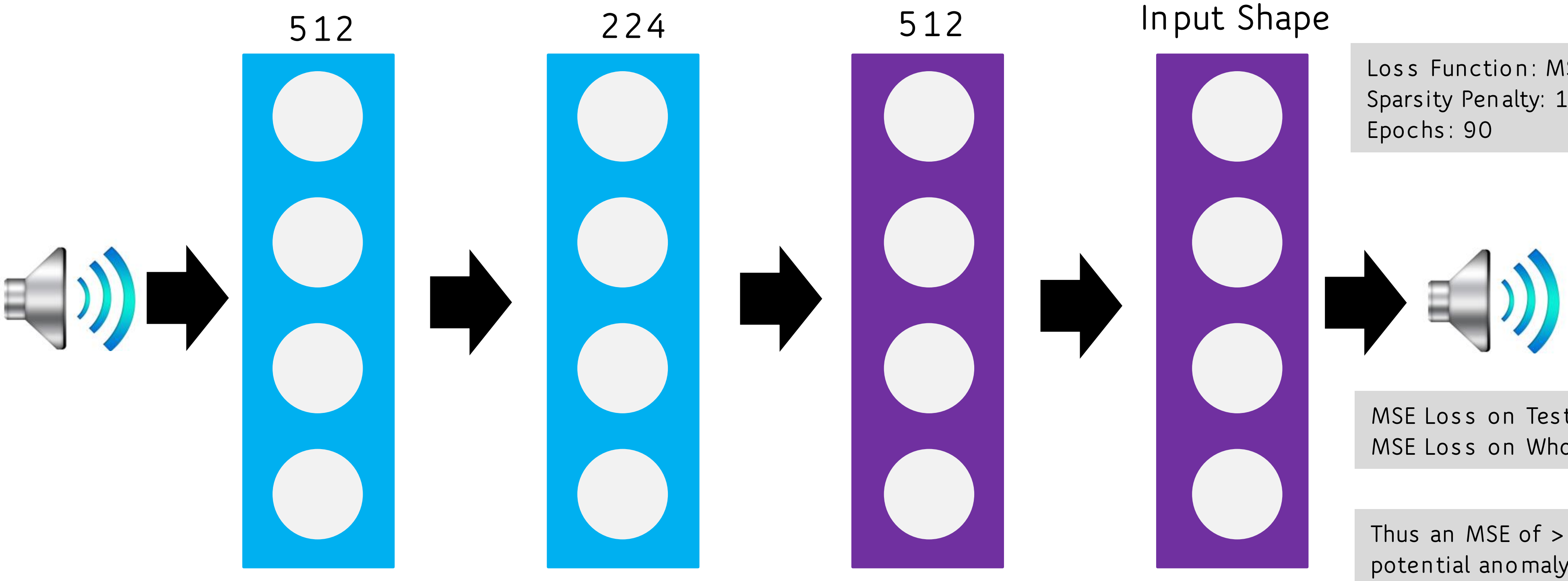
Read:
https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f
https://towardsdatascience.com/autoencoder-neural-network-for-anomaly-detection-with-unlabeled-dataset-af9051a048

Model Creation : Sparse Autoencoder

- A sparse autoencoder is simply an autoencoder whose training criterion involves a **sparsity penalty**.
- Sparsity penalty is applied on the hidden layer in addition to the reconstruction error. This prevents overfitting
- There are 2 different ways to construct our sparsity penalty:
  - **L1 regularization**
  - **KL-divergence**
- L1 regularization is often used as a method of **feature extraction**
- We will be using L1 Regularization in our model

Read:
https://medium.com/@syoya/what-happens-in-sparse-autencoder-b9a5a69da5c6
sparsity-explained-for-dummies-5b0e4be3938a

Model Creation : Model Construction & Training



Loss Function: MSE
Sparsity Penalty: 1e-5
Epochs: 90

MSE Loss on Test Dataset (out of sample): 0.61
MSE Loss on Whole Dataset (In + Out Sample) : 0.52

Thus an MSE of > 1.0 could be considered as a potential anomaly or Not-Cough class

Read:
https://medium.com/@syoya/what-happens-in-sparse-autencoder-b9a5a69da5c6
sparsity-explained-for-dummies-5b0e4be3938a
https://keras.io/api/layers/regularizers/

Prediction / Evaluation

Input Sound	MSE Score	Class Predicted
Cough Validation Sample #1	0.96	Cough
Cough Validation Sample #2	0.83	Cough
Cough Validation Sample #3	0.70	Cough
Cough Validation Sample #4	0.36	Cough
Dog Barking	1.03	Not Cough
Drilling	2.80	Not Cough
Siren	1.45	Not Cough
Gunshot	0.48	Cough

- The Model works quite well on most of the 'Non-Cough' sound classes from the Urban sound dataset
- The only class where it fails to predict the correct class is 'Gunshot'
- We believe this could be due to the limited data we used for training and can be improved if trained with a larger dataset