

REPORT

On

Classification of Sound using Machine Learning

SUBMITTED BY

Krishnakant Kumar

GUIDED BY

Mr Adarsh Sunil kumar

Scientist- B

**Naval Physical and Oceanographic Laboratory
DRDO**

CONTENTS:

1.	Abstract	
2.	Introduction	
3.	Literature Survey	
4.	Working Concept	
5.	Result	
6.	References	

ABSTRACT:

During this internship we discuss on many topics like urban sound classification, unknown classification, open set classification, autoencoder.

In urban sound classification my aim is built a model which classifies each sound into one of the unique categories. For that, I use the algorithm k-nearest neighbours for model and use MFCC as the extracted feature of each audio of the urban sound dataset.

In unknown classification we deal with the problem of unknown classes like when we train our model on classes (human, bird, and fish sound) and if I give none of these classes for testing eg. Cat sound, then the model will tell this sound belong which classes.

For deal with the problem of unknown classes I use open set recognition. These open set classifiers should handle unknown data that was not anticipated during the training phase of model.

An auto-encoder is one of unsupervised learning methods auto-encoders and their variants have used in image classification, pattern recognition, anomaly detection, and data generation. A general auto-encoder method usually contains an encoding structure and a decoding one.

INTRODUCTION:

In this report we introduced with Urban sound classification, unknown classification, open set classification, and autoencoder.

In the Urban Sound Classification, we sort sounds into classes using various classification methods. We use to differentiate of urban sounds from 10 classes: Air Conditioner, Car Horn, Children Playing, Dog bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren Street, and Music. Then we see how we can analyze the samplings in the time domain, frequency domain and how we can extract features. In the end of the Sound Analysis section In the Classification Methods section, we build model which classifies each sound into one of the unique categories. For that we use the algorithm K-nearest neighbour as our model, and we use the library librosa. For audio processing.

In unknown classification we deal with the problem of unknown classes like when we train our model on classes (human, bird, and fish sound) and if I give none of these classes for testing eg. Cat sound, then the model will tell this sound belong any of the classes. One idea is that I may create a "unknown" class and train it on all sorts of sound that are neither human, bird and fish sound, but the training set would need to be huge. That may be very difficult or impossible. One simple technique which we can use to approach learning unknown classes. That is decision function as opposed to the decision boundary or separating hyperplane used in binary classification. For identify unknown vs. known classes we use binary test, but this technique requires a data rich definition of what is unknown to allow training. For many applications this kind of method has been used. Another method to deal with the problem of unknown classes is open set recognition. These open set classifiers handle unknown data that was not anticipated during the training phase of model. During incomplete knowledge of the data exists at training time, and during the testing phase, unknown classes submitted to the algorithm. This requires the classifiers to accurately classify the seen classes and effectively deal with unseen ones. And the goal of open set classification is to develop algorithms that can distinguish between known and unknown sound data.

During this project we introduce with autoencoder and learn how birds sound train with autoencoder. We learn the autoencoder and there two parts: first encoder and second decoder. Through encoder we learn how to interpret the input and compress it to an internal representation defined by the bottleneck layer. And through decoder takes the output of the encoder the bottleneck layer and attempts to recreate the input.

Literature Survey:

1. Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm

-Eleni Tsaleraa , Andreas Papadakisb , Maria Samarakoua

In this article, They explained that how environmental noise has negative impact on human activities. In order to perform noise classification they use the KNN machine learning algorithm. In this they introduce similar work on environmental noise after they explain methodology and implementation. In methodology they explain the following steps: Initially, select the noise of interest, like sounds met in urban environments. Sound excerpts of different types of sound has been pre-processed including normalization and segmentation resulting into the training and the testing sets. Then select the sound features are extracted from the training files and after standardizing their values, they are fed into the KNN classification algorithm. The classification algorithm creates a set of classification models, enabling fine-tuning of the algorithm. Then they evaluate the result by summarised the result. For evaluation they consider true positive, true negative, false positive, false negative percentage.

2. Are open set classification methods effective on large-scale datasets?

-Ryne RoadyID, Tyler L. HayesID, Ronald Kemker, Ayesha Gonzales

In this article author introduce us with CNN (Convolutional neural networks) they explain here open set classification in CNN, here they explain two method 1) Inference methods to separate knowns from unknowns and 2) Feature representation methods n strategies to improve model robustness to novel inputs. In this paper, They performed a comprehensive comparison of OSC methods for CNNs using large-scale image classification datasets. Author demonstrated that detection performance is very dataset dependent but generally decreases as the similarity between the in-distribution and open set classes increases then still difficulty applying current state-of-the-art feature

representation strategies for OSC to large scale datasets that work in accordance with advanced inference methods.

3. An Introduction to Autoencoders

-Umberto Michelucci

In this article, Umberto Michelucci talk about autoencoders. This article covers the mathematics and the fundamental concepts of autoencoders. They explain what they are, what the limitations are, the typical use cases, and they explain some examples also. They discussed on the role of the activation function in the output layer and the loss function. After that he discuss what is the reconstruction error. They explain applications as dimensionality reduction, anomaly detection, denoising, and classification. In dimensionality reduction they explain that dimensionality reduction has one main advantage from a computational point of view that is it can deal with a very big amount of data efficiently since its training can be done with mini-batches. In Denoising autoencoders they told that it is developed to auto-correct errors in the input observations. And in application of classification they explain Classification with Latent Features. In this they told that if we want to classify our input images of the MNIST dataset. We can simply use an algorithm as KNN.

3. Autoencoders reloaded

-Herve Bourlard, Selen Hande Kabil

In this article, author Herve bourland and Selen Hande Kabil explain autoencoder. First they introduce with autoencoder and explain about encoder and decoder and MSE. Then they talk about history perspective then they explain mathematics concept after they explain architecture in which they talk about training and optimal solution during this they discuss about MSE. They use MNIST handwritten digits dataset and discuss about potential advantages of autoencoder. And explain Sparse autoencoders, Contractive autoencoders, Denoising autoencoders, Variational autoencoders, Adversarial autoencoders and then they talked about deep undercomplete and deep overcomplete and they finally told that none of those models could yield significant improvements on MSE cost and only deep undercomplete autoencoders with nonlinear space expansion of kernel-based approaches maintained improvements on the classification accuracy.

Activity:

Q1. Why you have chosen the given model?

Ans- The given model is totally based on Urban sound public dataset. Using training and test data from the UrbanSound8K public dataset, In this model use the K-Nearest Neighbors algorithm for classification. This models achieve performance between 70%and 85%.

Q2. What all features you have considered?

Ans- For classifies each sound into one of the unique categories we use K-nearest neighbour algorithm and use MFCC for extract feature of each audio and the library which is used is Librosa.

Q3. What all things you can do to improve your model?

Ans- To improve our model we should try to make the data same shape when we have transformed it. And try to maintain high degree of accuracy.

Q5. What are the limitations to the current model?

Ans- The KNN algorithm doesn't work well with high dimensional data in current model.

KNN is sensitive to noise in the dataset. I need to impute missing values and remove outliers from current model.

Q6. What all features we can consider for audio classification?

Ans- For audio classification we can consider amplitude and frequencies which is show in spectrogram, zero crossing rate, spectral centroid, MFCC.

Q7. What all models you can use for sound classification?

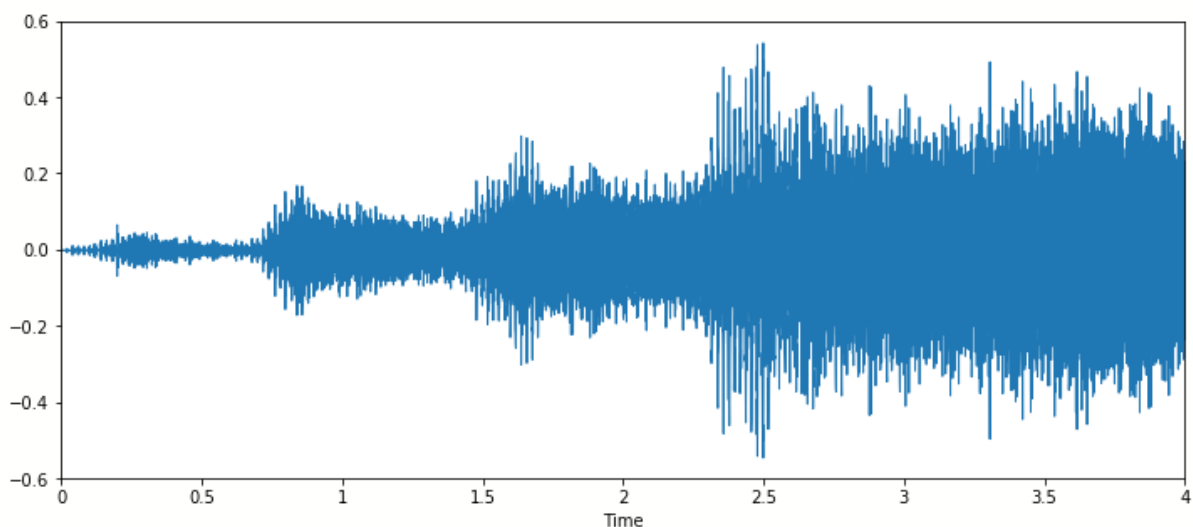
Ans- Convolutional neural network, random forest and Mel spectrogram etc.

Methodology:

Urban Sound Classification with KNN:

First we import some libraries `numpy`, `pandas`, `librosa`, `sound file`, `matplotlib`, `IPython` and `sklearn`.

Then we read the audio file, after we extract audio data and sample rate and then plot the graph.



Now we plotting the crossing rate in which we explain how many times the audio wave crossing the zero line.

Then we calculate spectral centroid and define the shape then display a spectrogram of MFCC extract feature. After extract features from our audios, we create our model.

To start we will compute the mean mfcc of each feature inside the mfcc. Reason is data that the mfcc returns is not always the same length, and since our model expects the data to be the same shape, we have transformed it.

Then we parse each audio file and compute the mean mfcc for each one. Then after we can feed our model with the data.

Now we split the data into two parts (1) train (2) test of x and y. Then we standardized our trained data by reducing noise with the help of PCA.

Then we create our model with grid search cv and applying KNN classifier then we print the score of the model and print confusion matrix for checking the accuracy of our model.

Model Score: 0.9367181751287712

Confusion Matrix:

```
[ [150  5  2  2  1  1  0  0  1  0]
[  0 67  0  1  0  0  1  0  0  1]
[  1  2 140  8  2  1  4  0  2  6]
[  0  0  3 135  0  0  2  0  2  1]
[  1  0  0  1 139  0  0  0  0  0]
[  0  2  3  0  0 152  0  0  1  0]
[  0  0  0  6  0  0  56  0  0  1]
[  0  2  0  2  0  0  0 152  0  2]
[  0  1  0  5  0  0  0  0 153  0]
[  0  3  0  0  1  0  2  2  2 129]]
```

2. Unknown class classification:

In supervised machine learning many Methods are use for the classification of multiple classes.

all classes must be known ahead of time, the classes are available as part of the training data. In many problems only a limited number of known classes and many time cannot given. This is a type of problem of unknown classes.

To approach learning unknown classes. We use "decision function" as opposed to the decision boundary or separating hyperplane used in binary classification. We define a distance metric using either linear or kernel-based projection techniques.

Then we constructed a simple binary classifier rather than defining a discrete set of decision boundaries, we used to "continuum" of decision. We projecting down from a high dimensional space onto a single dimension, but rather than compute a simple decision based on a set of known classes, we apply a function which can result in a class assignment other than those present in the trainingdata. We leaved "extra" classes latent in the decision function.

3. Open set Classification:

open-set audio recognition classification we explain methodology which can distinguish audio events. In an open-set audio recognition by using a dataset that contains audio events recorded in urban area and from the Free sound online. The recorded events have different sizes and varying noise levels. The sound event classes used are: human, dog, cat, cars, drawer, wind, birds, bikes etc

The input audio files are preprocessed with normalization, frame segmentation and windowing. Files that have different sampling frequencies are resampled to this value. Each audio file is normalized to maximum unit amplitude, to bring the gain of the entire track to its maximum without clipping.

We use the support vector machine classifier, SVM, along with peak-side-ratio, PSR. The SVM uses hyperplanes to define the decision boundaries that separate data spaces of different classes. We know that SVM is a binary classifier, so it is required to use several SVM classifiers operating in parallel to solve multiple binary classification problems. Multi-binary SVM classifiers are constructed for a multi-class problem using a One-versus-All technique.

The features of the validation set are used SVM training parameters. open set classification evaluation must keep track of incorrect multi-class classification over known categories and errors between unknown and known categories. Therefore, two types of open set experiments are conducted: reduced open-set recognition, which computes errors between known and unknown categories, and multi-class open-set recognition. This method is promising for open set audio classification.

4. Sound Classification with Autoencoder

First of all we importing the required libraries pandas, numpy as np, os, librosa, librosa.display, glob, skimage , matplotlib.pyplot, numpy , seaborn, train_test_split, tensorflow, tensorflow_addons , import keras, noisereduce, ceil, random

Then read a .csv file and load .wav file as pandas data frame. Loading the data and dividing the data into training and testing sets. The autoencoder consists of two parts that are the encoder and the decoder. The encoder learns how to interpret the input and compress it to the internal the output of the encoder representation defined by the bottleneck layer. The decoder takes and attempts to recreate the input.

Once the autoencoder is trained, the decoder is discarded and we only keep the encoder and use it to compress input to output by the bottleneck layer.

In this first autoencoder, we won't compress the input and use a bottleneck layer the same size as the input. Model learn nearly perfectly and is intended to confirm our model is implemented correctly. We plot figure of Linear-frequency power spectrogram of data.

Then defining a utility function to build the Auto-encoder neural network by Building the encoder of the Auto-encoder and building the decoder of the Auto-encoder. After that defining a utility function to build and train the Auto-encoder network.

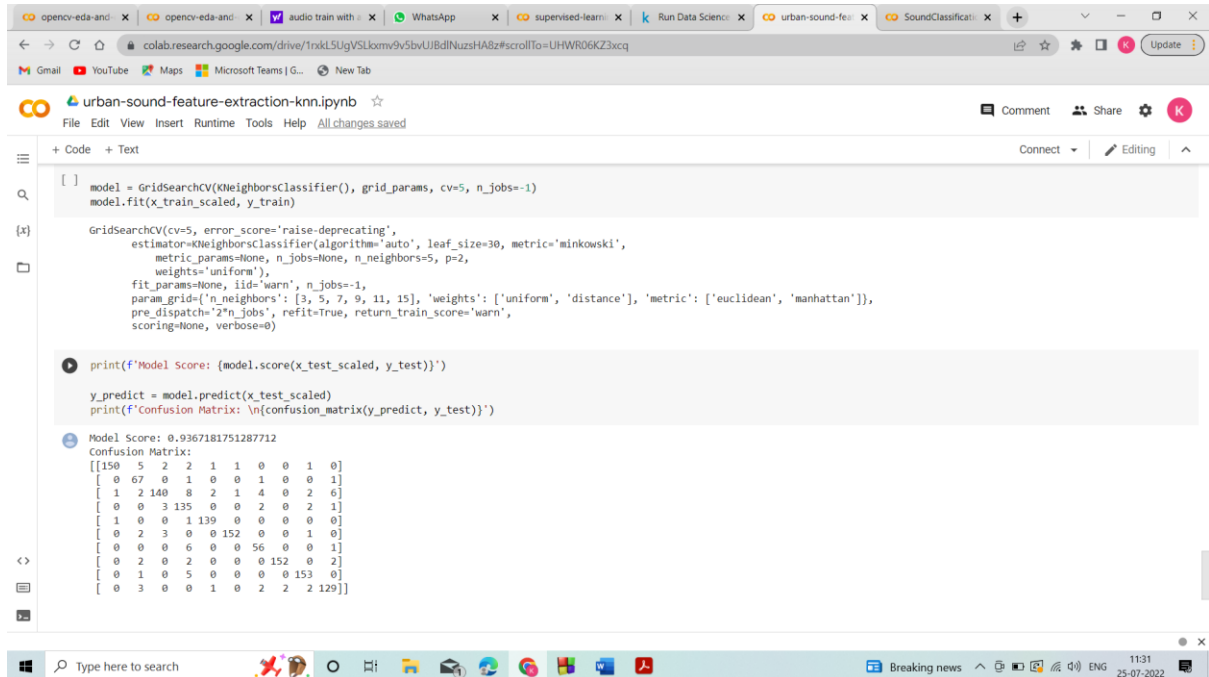
For that we defining the parameters of the Auto-encoder extract features kaiser fast is a technique used for faster extraction. We extract mfcc feature from data.

Then we configure autoencoder model by taking dimension of input signal then show dimension of encoded features, By defining epochs and batch size then setup the training and testing data of our model.

So load the data and building the network then building and training the Auto-encoder model. Then we print the result of the model.

Result:

Urban sound Dataset Result:



```
[ ] model = GridSearchCV(KNeighborsClassifier(), grid_params, cv=5, n_jobs=-1)
model.fit(x_train_scaled, y_train)

GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
             metric_params=None, n_jobs=None, n_neighbors=5, p=2,
             weights='uniform'),
             fit_params=None, iid='warn', n_jobs=-1,
             param_grid={'n_neighbors': [3, 5, 7, 9, 11, 15], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
             scoring=None, verbose=0)

print(f'Model Score: {model.score(x_test_scaled, y_test)}')

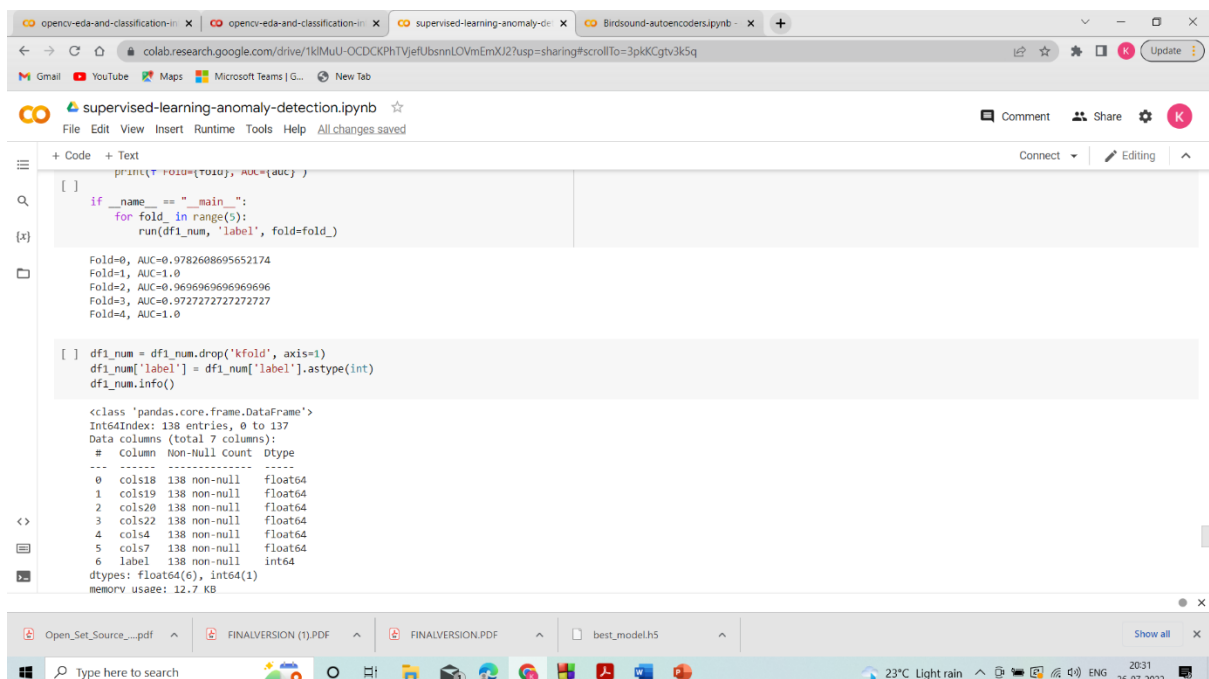
y_predict = model.predict(x_test_scaled)
print(f'Confusion Matrix: \n{confusion_matrix(y_predict, y_test)}')
```

Model Score: 0.9367181751287712

Confusion Matrix:

```
[[150  5  2  2  1  1  0  0  1  0]
 [  0 67  0  1  0  0  1  0  0  1]
 [  1  2 140  0  2  1  4  0  2  6]
 [  0  0  3 135  0  0  2  0  2  1]
 [  1  0  0  1 139  0  0  0  0  0]
 [  0  2  3  0  0 152  0  0  1  0]
 [  0  0  0  6  0  0 56  0  0  1]
 [  0  2  0  2  0  0 152  0  2  0]
 [  0  1  0  5  0  0  0 153  0  0]
 [  0  3  0  0  1  0  2  2  2 129]]
```

Supervised learning Unknown Class Result:



```
[ ] print(f'Fold={fold}, AUC={auc}')
```

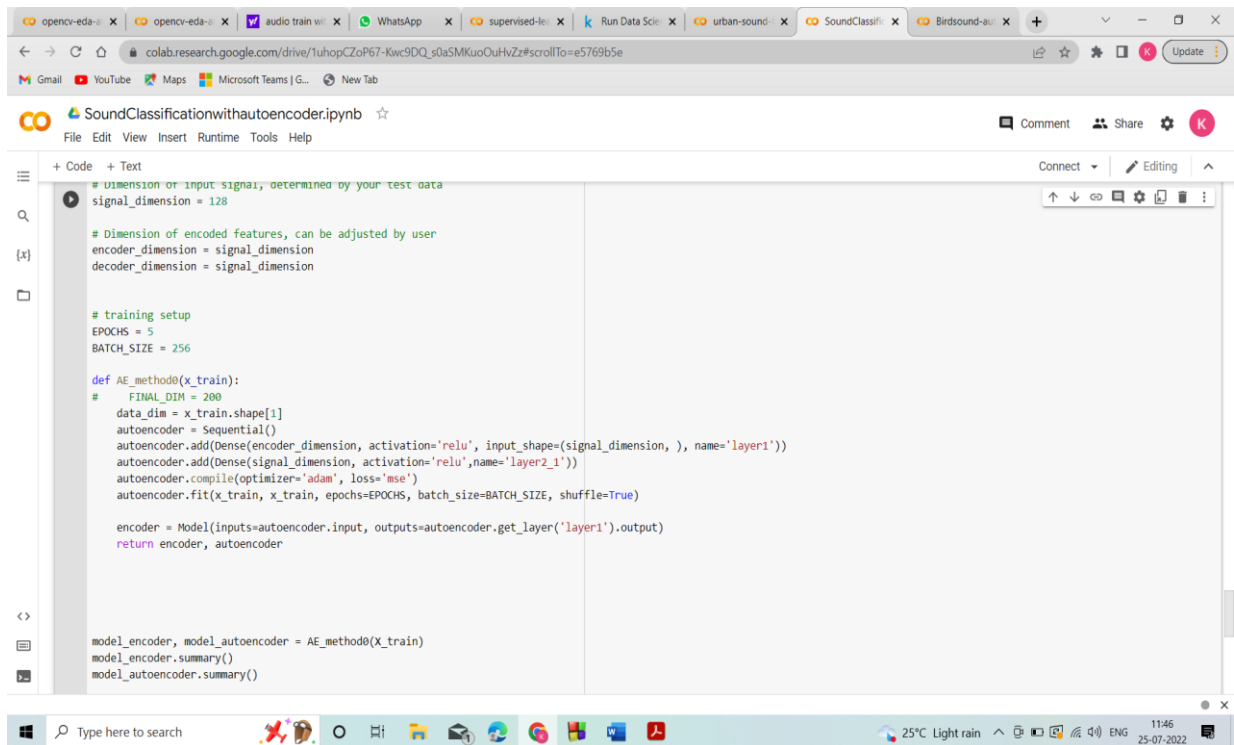
```
[ ] if __name__ == "__main__":
    for fold_in range(5):
        run(df1_num, 'label', fold=fold_)
```

Fold=0, AUC=0.9782608695652174
Fold=1, AUC=1.0
Fold=2, AUC=0.9696969696969696
Fold=3, AUC=0.9727272727272727
Fold=4, AUC=1.0

```
[ ] df1_num = df1_num.drop('kfold', axis=1)
df1_num['label'] = df1_num['label'].astype(int)
df1_num.info()
```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 138 entries, 0 to 137
Data columns (total 7 columns):
column Non-Null Count Dtype
--- ---
0 cols18 138 non-null float64
1 cols19 138 non-null float64
2 cols20 138 non-null float64
3 cols22 138 non-null float64
4 cols4 138 non-null float64
5 cols7 138 non-null float64
6 label 138 non-null int64
dtypes: float64(6), int64(1)
memory usage: 12.7 KB

Autoencoder Result:



```
# dimension of input signal, determined by your test data
signal_dimension = 128

# Dimension of encoded features, can be adjusted by user
encoder_dimension = signal_dimension
decoder_dimension = signal_dimension

# training setup
EPOCHS = 5
BATCH_SIZE = 256

def AE_method0(x_train):
    # FINAL_DIM = 200
    data_dim = x_train.shape[1]
    autoencoder = Sequential()
    autoencoder.add(Dense(encoder_dimension, activation='relu', input_shape=(signal_dimension, ), name='layer1'))
    autoencoder.add(Dense(signal_dimension, activation='relu', name='layer2_1'))
    autoencoder.compile(optimizer='adam', loss='mse')
    autoencoder.fit(x_train, x_train, epochs=EPOCHS, batch_size=BATCH_SIZE, shuffle=True)

    encoder = Model(inputs=autoencoder.input, outputs=autoencoder.get_layer('layer1').output)
    return encoder, autoencoder

model_encoder, model_autoencoder = AE_method0(X_train)
model_encoder.summary()
model_autoencoder.summary()
```

Epoch 1/5
200/200 [=====] - 14s 14ms/step - loss: 4.9231
- accuracy: 0.0417
Epoch 2/5
200/200 [=====] - 3s 14ms/step - loss: 3.8631
- accuracy: 0.0362
Epoch 3/5
200/200 [=====] - 3s 14ms/step - loss: 3.6251
- accuracy: 0.0599
Epoch 4/5
200/200 [=====] - 3s 14ms/step - loss: 3.7865
- accuracy: 0.0449
Epoch 5/5
200/200 [=====] - 3s 14ms/step - loss: 3.6565
- accuracy: 0.0997

REFERENCES:

Urban sound classification

- [Oli06] E.T. Oliphant. "A guide to NumPy". In: Trelgol Publishing (2006).
- [McK10] W. McKinney. "Data Structures for Statistical Computing in Python". In: Proceedings of the 9th Python in Science Conference (2010), pp. 51 { 56.
- [KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: Commun. ACM 60.6 (May 2012), pp. 84 { 90. issn: 0001-0782. doi: 10.1145/3065386. url: <http://doi.acm.org/10.1145/3065386>.
- [Kar13] A. Karpathy. "Convolutional Neural Networks for Visual Recognition". <http://cs231n.github.io/convolutional-networks/>. [Online; accessed 06-December-2018]. 2013.
- [Lyo13] J. Lyons. "Mel Frequency Cepstral Coefficient (MFCC) tutorial". <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>. [Online; accessed 06-December-2018]. 2013.
- [Tan13] T. Tantau. "Graph Drawing in TikZ". In: Proceedings of the 20th International Conference on Graph Drawing. GD'12. Redmond, WA: Springer-Verlag, 2013, pp. 517 { 528. isbn: 978-3-642-36762-5. doi: 10.1007/978-3-642-36763-2_46.
- [Cho+14] K. Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: CoRR abs/1406.1078 (2014). arXiv: 1406.1078. url: <http://arxiv.org/abs/1406.1078>.
- [KB14] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: CoRR abs/1412.6980 (2014). arXiv: 1412.6980. url: <http://arxiv.org/abs/1412.6980>.

Unkown class classification

- 1] B. Scholkopf, A. Smola, Learning with kernels, MIT Press, 2002.
- [2] R. Duda, P. Hart, D. Stork, Pattern Classification, Wiley-Interscience, 2001.
- [3] R. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics, (7): 179-188, 1936.
- [4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth & Brooks, 1984.
- [5] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
- [6] K. Crammer, Y. Singer, On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines, Journal of Machine Learning Research, 265-292, 2001.

Open-set classification

1. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016.
2. Hu J, Shen L, Sun G. Squeeze-and-Excitation Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018.
3. Scheirer WJ, Rocha A, Sapkota A, Boulton TE. Towards Open Set Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI). 2013; 35.
4. Kemker R, Kanan C. Fearnnet: Brain-inspired model for incremental learning. In: Proceedings of the International Conference on Learning Representations (ICLR); 2018.
5. Parisi GI, Kemker R, Part JL, Kanan C, Wermter S. Continual lifelong learning with neural networks: A review. Neural Networks. 2019. <https://doi.org/10.1016/j.neunet.2019.01.012> PMID: 30780045
6. Hayes TL, Cahill ND, Kanan C. Memory Efficient Experience Replay for Streaming Learning. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA); 2019.

Sound classification with Autoencoder

- Ashby WR (1961) An introduction to cybernetics. Chapman & Hall Ltd, New York
- Baldi P (2012) Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning. JMLR Workshop and Conference Proceedings, pp 37–49
- Baldi P, Hornik K (1989) Neural networks and principal component analysis: Learning from examples without local minima. Neural Netw 2(1):53–58
- Baldi PF, Hornik K (1995) Learning in linear neural networks: a survey. IEEE Trans Neural Netw 6(4):837–858
- Bengio Y, Lamblin P, Popovici D, Larochelle H (2007) Greedy layerwise training of deep networks. In: Advances in neural information processing systems, pp 153–160
- Ben-Hur A, Horn D, Siegelmann HT, Vapnik V (2002) Support vector clustering. J Mach Learn Res 2:125–137
- Bourlard H, Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybern 59(4):291–294

Thank You