Dynamic Speaker Identification

Kais Hasan, Ali Abbas, Santia John Jarrah

Abstract:

The reason behind this project is identifying the speaker's identity from the voice, this task becomes more and more difficult as speakers increase, diverse, and the voices was obtained from non-constrained environment.

We have used features extraction methods with CNN based model and Incremental learning.

Datasets:

We have used two datasets:

1. FSD50K [1]:

   FSD50K contains 51,197 audio clips from Freesound, totalling 108.3 hours of multi-labeled audio.

   The dataset encompasses 200 sound classes (144 leaf nodes and 56 intermediate nodes) hierarchically organized with a subset of the AudioSet Ontology.

2. Voxceleb1 [2]:
   VoxCeleb1 contains over 100,000 utterances for 1,251 celebrities, extracted from videos uploaded to YouTube.

We used both datasets in features extraction phase, but of course without the labels.

Then we used the second dataset for supervised learning.

All datasets preprocessed in the same way:

1. Converted to 16bit, 16KHz sample rate, mono audio using SoX.
2. We made the length of each audio equal to 3 seconds by either padding or cutting.
3. We converted each audio segment to mel-scaled spectrogram.
4. We had calculated the mean and standard deviation from the second dataset and used them for normalizing the two datasets, and of course any future use of an audio file with the model.

Features extraction:

We used the auto-encoder (AE) for this purpose, which works in two steps:

1. Encode the input.
2. Decode the result of first step.

   Usually, we use many techniques to ensure that the AE will not simply learn to copy the input, instead it will learn to encode the input to a set of rich useful features so that it is possible to decode it again in step 2.

In this phase, we had tried to use sparse AE but it gave us bad results so we switched to denoising auto-encoder (DAE) which gave us the required results.

DAE works by corrupting the input and trying to reconstruct the uncorrupted input from the corrupted one, which ensures that the DAE will not learn the identity function, and it will learn to extract good features.

We corrupted the spectrogram in two ways:

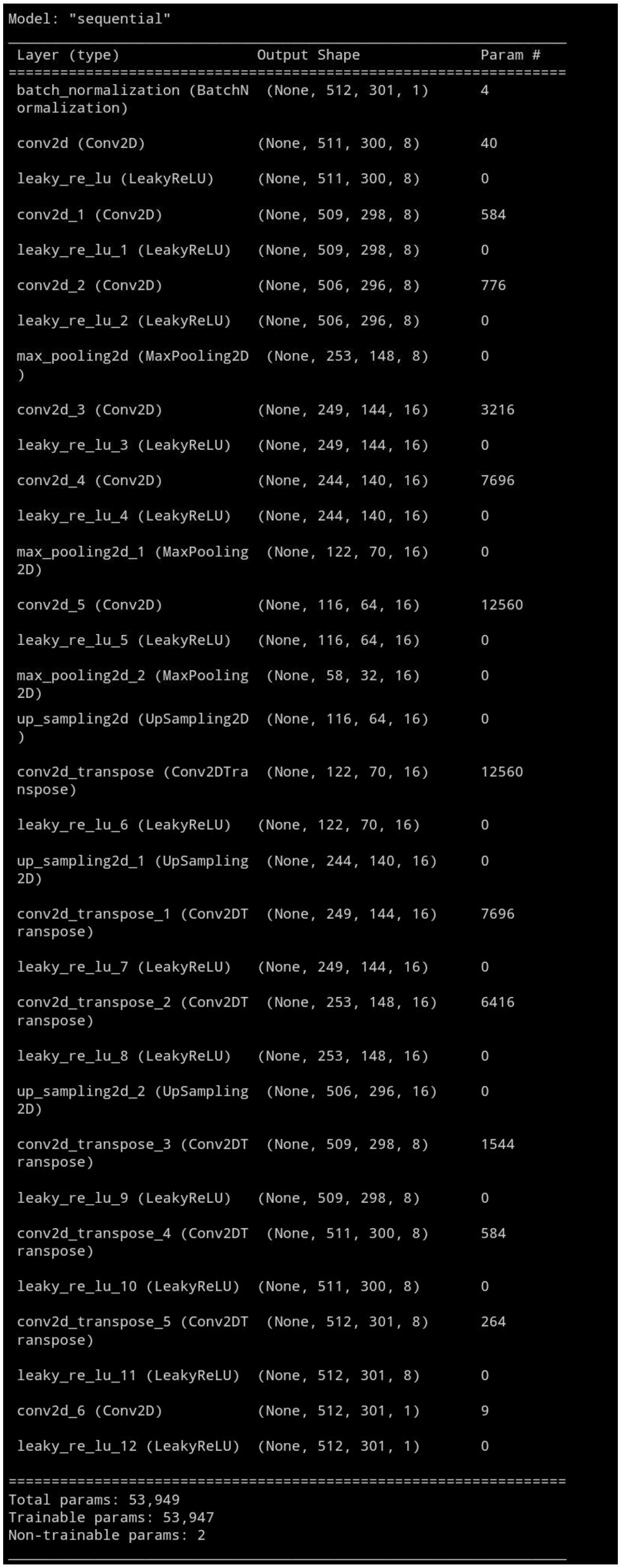
1. Adding standard gaussian noise to each example.
2. Masking the spectrogram in the frequency domain, i.e. masking a row from the spectrogram.

Using the above methods we managed to get a very good results with MSE = 0.4, and by testing the features with a simplified version of the next phase we found them as good as we wanted.

After we had trained the DAE we used it to build a new dataset consists of the extracted features.

The summary of the DAE

We have built is:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 batch_normalization (BatchN  (None, 512, 301, 1)      4
 ormalization)

 conv2d (Conv2D)             (None, 511, 300, 8)       40

 leaky_re_lu (LeakyReLU)     (None, 511, 300, 8)       0

 conv2d_1 (Conv2D)           (None, 509, 298, 8)       584

 leaky_re_lu_1 (LeakyReLU)   (None, 509, 298, 8)       0

 conv2d_2 (Conv2D)           (None, 506, 296, 8)       776

 leaky_re_lu_2 (LeakyReLU)   (None, 506, 296, 8)       0

 max_pooling2d (MaxPooling2D  (None, 253, 148, 8)      0
 )

 conv2d_3 (Conv2D)           (None, 249, 144, 16)      3216

 leaky_re_lu_3 (LeakyReLU)   (None, 249, 144, 16)      0

 conv2d_4 (Conv2D)           (None, 244, 140, 16)      7696

 leaky_re_lu_4 (LeakyReLU)   (None, 244, 140, 16)      0

 max_pooling2d_1 (MaxPooling  (None, 122, 70, 16)      0
 2D)

 conv2d_5 (Conv2D)           (None, 116, 64, 16)       12560

 leaky_re_lu_5 (LeakyReLU)   (None, 116, 64, 16)       0

 max_pooling2d_2 (MaxPooling  (None, 58, 32, 16)       0
 2D)
 up_sampling2d (UpSampling2D  (None, 116, 64, 16)      0
 )

 conv2d_transpose (Conv2DTra  (None, 122, 70, 16)      12560
 nspose)

 leaky_re_lu_6 (LeakyReLU)   (None, 122, 70, 16)       0

 up_sampling2d_1 (UpSampling  (None, 244, 140, 16)     0
 2D)

 conv2d_transpose_1 (Conv2DT  (None, 249, 144, 16)     7696
 ranspose)

 leaky_re_lu_7 (LeakyReLU)   (None, 249, 144, 16)      0

 conv2d_transpose_2 (Conv2DT  (None, 253, 148, 16)     6416
 ranspose)

 leaky_re_lu_8 (LeakyReLU)   (None, 253, 148, 16)      0

 up_sampling2d_2 (UpSampling  (None, 506, 296, 16)     0
 2D)

 conv2d_transpose_3 (Conv2DT  (None, 509, 298, 8)      1544
 ranspose)

 leaky_re_lu_9 (LeakyReLU)   (None, 509, 298, 8)       0

 conv2d_transpose_4 (Conv2DT  (None, 511, 300, 8)      584
 ranspose)

 leaky_re_lu_10 (LeakyReLU)  (None, 511, 300, 8)       0

 conv2d_transpose_5 (Conv2DT  (None, 512, 301, 8)      264
 ranspose)

 leaky_re_lu_11 (LeakyReLU)  (None, 512, 301, 8)       0

 conv2d_6 (Conv2D)           (None, 512, 301, 1)       9

 leaky_re_lu_12 (LeakyReLU)  (None, 512, 301, 1)       0

=================================================================
Total params: 53,949
Trainable params: 53,947
Non-trainable params: 2
_________________________________________________________________
```

Better data pipeline:

After we trained the DAE, we have noticed that there is a poor performance in the data pipeline caused a huge bottleneck, so we have decided to try to improve it for easier use, and for faster training.

We followed Tensorflow recommendation to use TFRecord dataset, so we converted the ‘npy’ files that contains the features extracted from the actual dataset using the DAE to tfrecords files.

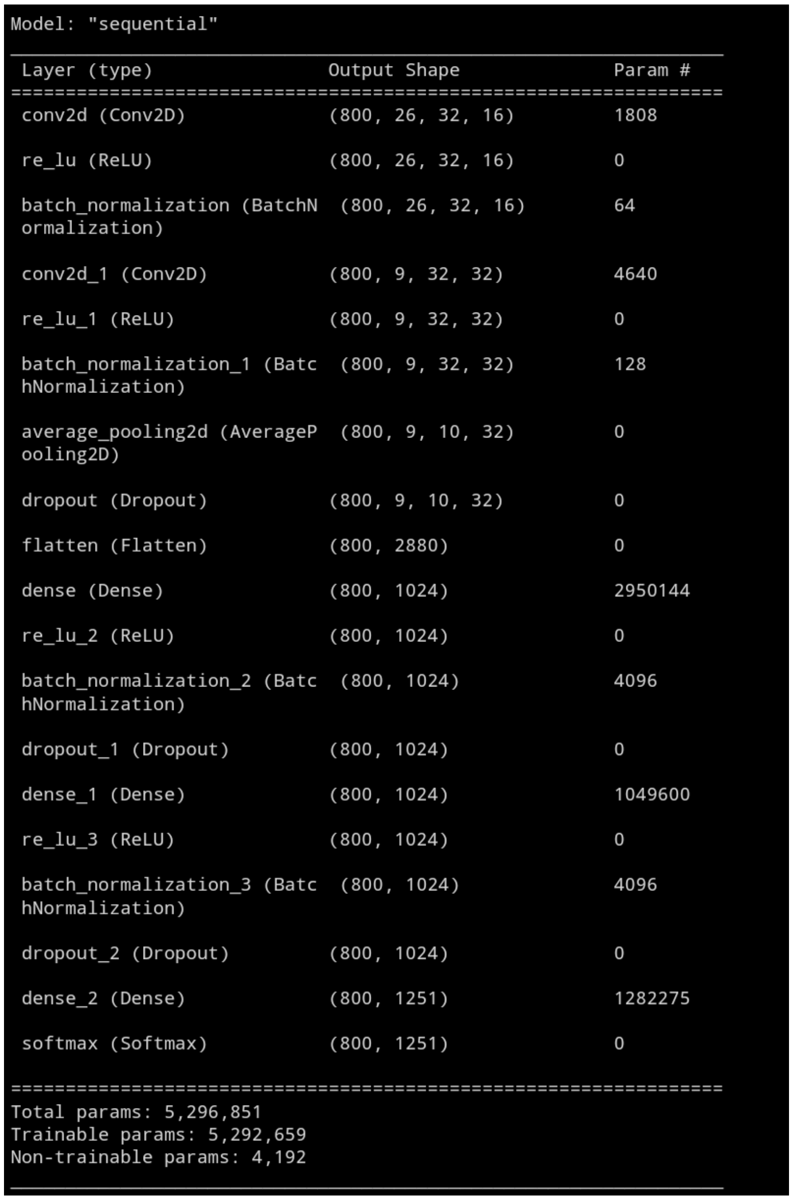
The TFRecord format is a simple format for storing a sequence of binary records.

This allowed us to take advantage of better data pipeline, which made the training process faster.

Supervised learning:

For this phase, we used CNNs with vertical kernels along with some final dense layers.

The summary of the model we have built is:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (800, 26, 32, 16)         1808

 re_lu (ReLU)                (800, 26, 32, 16)         0

 batch_normalization (BatchN  (800, 26, 32, 16)        64
 ormalization)

 conv2d_1 (Conv2D)           (800, 9, 32, 32)          4640

 re_lu_1 (ReLU)              (800, 9, 32, 32)          0

 batch_normalization_1 (Batc  (800, 9, 32, 32)         128
 hNormalization)

 average_pooling2d (AverageP  (800, 9, 10, 32)         0
 ooling2D)

 dropout (Dropout)           (800, 9, 10, 32)          0

 flatten (Flatten)           (800, 2880)               0

 dense (Dense)               (800, 1024)               2950144

 re_lu_2 (ReLU)              (800, 1024)               0

 batch_normalization_2 (Batc  (800, 1024)              4096
 hNormalization)

 dropout_1 (Dropout)         (800, 1024)               0

 dense_1 (Dense)             (800, 1024)               1049600

 re_lu_3 (ReLU)              (800, 1024)               0

 batch_normalization_3 (Batc  (800, 1024)              4096
 hNormalization)

 dropout_2 (Dropout)         (800, 1024)               0

 dense_2 (Dense)             (800, 1251)               1282275

 softmax (Softmax)           (800, 1251)               0

=================================================================
Total params: 5,296,851
Trainable params: 5,292,659
Non-trainable params: 4,192
_________________________________________________________________
```

Incremental Learning:

Since our model should allow adding more speakers after training, we must have some sort of incremental learning.

For this task, we have used the approach in Learning Without Forgetting [3] which gave us a good result.

Results and notes:

It took us about 2 days to train the DAE on NVIDIA GEFORCE GTX 1660 Ti, which corresponds to 100 epochs.

It took us about 4 days to train the final model on NVIDIA GEFORCE GTX 1660 Ti, which corresponds to 5000 epochs.

The accuracy on Voxceleb dataset is 79, but we could have improved it with a more careful hyperparameters tuning.

We have been testing the final model (with incremental learning) for 4 days in the CAPE (Central Applied Projects Exhibition) as we participated with this project representing the Artificial Intelligence department in our faculty, and it gave us a very good results, with accuracy 85.

The cause of better accuracy is that the experiment happened in a more controlled environment without too much noise.

References:

[1] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, Xavier Serra. "FSD50K: an Open Dataset of Human-Labeled Sound Events", arXiv:2010.00475, 2020.

[2] A. Nagrani*, J. S. Chung*, A. Zisserman

VoxCeleb: a large-scale speaker identification dataset INTERSPEECH, 2017.

[3] Li Z., Hoiem D. (2017) Learning Without Forgetting, arXiv:1606.09282v3.