COMP6252 COURSEWORK 2

DONG WANG
STUDENT ID: 32798881
dw6u21
dw6u21@soton.ac.uk

Abstract

The report introduces the generation and classification of audio data is achieved using LSTM and GAN pairs.

1. RNN and LSTM

The network structure is a Recurrent Neural Networks (RNN) two-layer network using LSTM, which fits the audio data of this dataset, which consists of time and frequency. Since the current output of a sequence in an RNN is also related to the previous output. The Long Short Memory network (LSTM) is a special type of RNN, which contains three gates (forget gate, input gate, output gate) and a memory unit. The LSTM unit is shown 1. The first step of the LSTM is to decide what information can pass through the cell state, a decision controlled by the "forget gate" layer via sigmoid, which generates a f_t value from 0 to 1 based on the previous output h_{t-1} and the current input x_t to decide whether to let the information learned at the previous time, C_{t-1}, pass or partially pass. This decision is controlled by the "forget gate" layer via sigmoid, which generates a f_t value from 0 to 1 based on the previous time's output h_{t-1} and the current input x_t to decide whether to let the information learned at the previous time C_{t-1} pass or partially pass. The input gate determines which values are used for updates via sigmoid, and the tanh layer is used to generate new candidate values C_t. The cell state is updated by multiplying the previous cell state by f_t forgetting the unwanted information and then add it to $i_t C_t$ to get the candidate value.The final step is to determine the output of the model, starting with a sigmoid layer to obtain an initial output, then using tanh to scale the C_t value between -1 and 1 and multiplying it pair by pair with the output obtained from sigmoid to obtain the output of the model.

$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
o_t &= \sigma\left(W_o\left[h_{t1}, x_t\right] + b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}
\tag{1}
$$

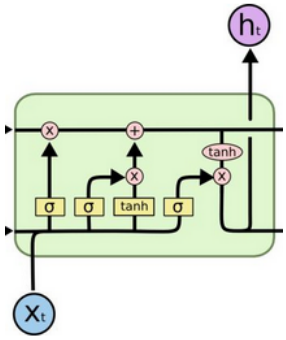

Figure 1. LSTM cell

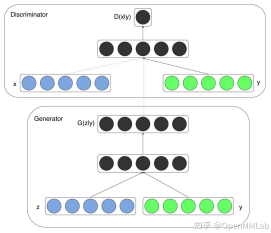

Figure 2. CGAN

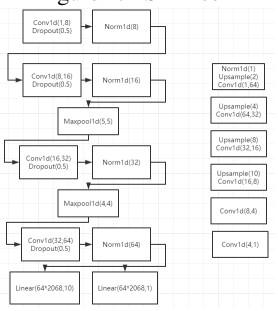

Figure 3. CNN structure

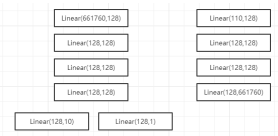

Figure 4. FCN structure

2. Data preprocessing

Firstly the dataset has 1000 audio samples, of which 'jazz00054' is corrupted. So the final dataset is 999 audio

samples. The librosa api, commonly used in Python, was used to extract features from the data when pre-processing it. A total of three types of processing were used in the dataset, firstly simply converting the amplitude of the audio signal to decibels, the data was set to a threshold, and if the audio did not reach this threshold, this was filled with zeros. However, there are drawbacks to such data processing, and there is no processing of the length of the data; the output data is longer and fits poorly in the experiment. This method is not suitable for LSTM. Secondly, using MFCC features, Mel Frequency Cepstral Coefficients are used to calculate an audio signal's Mel Frequency Cepstral Coefficients. The input to this function is the time series data of the audio signal, and the output is its corresponding MFCC feature matrix. The feature extraction is first performed on the uncut data. The obtained matrix remains large. Low accuracy and overfitting also occurred in the subsequent experiments. The third pre-processing method is to augment the dataset. The audio is sliced and added to the dataset. A threshold is set, and when the audio is greater than or equal to this threshold, it is added to the dataset, which effectively expands the dataset. Conversely overfitting improves accuracy. Also, the threshold value is set so that the dimensionality of the generated matrix is reduced.

3. GAN

When generating data, use cGAN to generate the same amount of data, because cGan can be input to the noise z of the generator network G to concat a label vector y, making the generation network to generate the data specified by the label. For the data input to the discriminator D, a label such as concat is also used to tell the discriminant network to judge whether the input is real data of this category. The structure diagram is as follows. In the experiment, cGAN uses one-dimensional convolution and fully connected network respectively. Discriminator and Generator can use linear layer [1] one-dimensional input to easily concat with label vector or label embedding. Two network structures are shown by 4 and 3. The loss function adopts BCEWithLogitsLoss because the traditional cross-entropy loss function can only input a value between zero and one, which requires the model to add the sigmoid function when outputting, but for the Generator in Gan, adding the sigmoid function will destroy the audio itself. So take this loss function. The discriminator cannot use sigmoid, which will destroy the composition of the music. This is also the loss that cannot break through 0.6059 no matter how it is trained at the beginning. The BCELOSS function cannot be used when sigmoid is cancelled, because the function only accepts values between zero and one. The separate label of blue is used to verify that the training results can get audio. During training, you should first generate a generator's fake audio, then train the discriminator for true and false, and then train the generator. All models will take 500 epochs. There are three different trainings for Gan:
1. Use CNN and do not split the data
2. Use FCN and do not split the data
3. Use FCN and split the data into 4 seconds
The reason for cutting out 4 seconds is due to the lack of computing power of the computer. After splitting the data, the fully connected layer can be added, and the batch size can be increased at the same time. At the same time, in order to verify whether it will affect the generation of audio, the Gan of FCN is used to train the blue type of music separately. The results show that the audio effect generated by a separate label is better. A multi-category model may have fewer samples per category, so the generation will be noisy. However, because cgan combines labels, it is more suitable for the generation of multiple label data. During the training process, the loss of the generator is gradually increased, and the loss of the discriminator is gradually decreased. The loss of the Generator of the above three models can reach more than 10. This proves that the model fit is relatively perfect.

4. Result evaluation

Data evaluation As the table shows, when no expansion of the data was performed and the initial length was maintained, both accuracy and loss at 50 epochs were not as good as the results when the amount of data was increased by reducing the length. This is due to the nature of the LSTM for data reading, where the LSTM forgets or remembers the data from the previous text. But data that is too long is often not a good fit. This is due to underfitting caused by the excessive length of the text. Also, to solve the underfitting problem that occurs in training, increasing the amount of data is the most effective way. The model that worked best without the addition of the data generated by Gan was the one that used Adam as the optimizer after the expanded dataset, achieving an accuracy of 0.753 in the validation set. 5 shows the accuracy of the normal unsegmented data, 6,7 shows the accuracy and loss after segmentation. 8 is the accuracy rate after adding gan data. Due to the lack of data, Gan could not generate enough high-quality audio, so the performance after adding LSTM did not get a high accuracy rate. Accuracy is best in GANs using fully connected layers, but it suffers from overfitting. As can be seen from the figure, using the clipped data is not ideal. After experiments, the size of the Zdim noise has little effect on the audio generation. But during the experiment, the generated audio with sigmoid is more helpful for LSTM classification. The accuracy rate is similar to the first phase, but there is an overfitting situation.

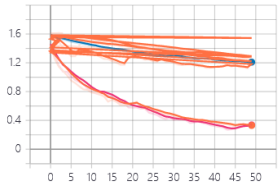

Figure 5. LSTM loss

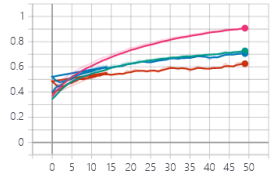

Figure 6. LSTM accuracy

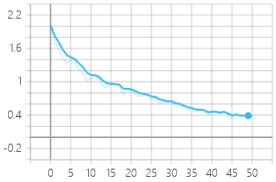

Figure 7. LOSS

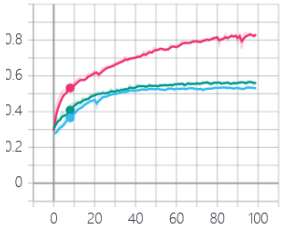

Figure 8. CGAN

Model	Loss	Accuracy	Epoachs	Vail
LSTM_SGD_50__no_segment	2.0408	0.380	50	0.359
LSTM_ADAM_50_no_segment	2.0317	0.3733	50	0.3199
LSTM_ADAM_50_segment	0.8796	0.881	50	0.509
LSTM_ADAM_50_segment	0.8265	0.915	50	0.753
LSTM_CNN_GAN_segment	1.381	0.572	100	0.471
LSTM_FCN_GAN_segment	0.8621	0.837	100	0.480
LSTM2_FCN_4S_GAN_segment	1.2787	0.550	100	0.454

Figure 9. table

5. conclusion

LSTM is picky about the data. Although LSTM is suitable for temporal data, the length of the data, still needs to be selected to the right value. The amount of data can also improve accuracy. cGAN is a big improvement for audio generation and classification with the addition of labels. In the future, you can try to enter each individual label for the number of times corresponding to this number of samples. This results in a corresponding amount of generated audio. Since the data has 999 different audios, ordinary computers can no longer complete this task with limited time.

6. Appendix

References

[1] Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.