Efficacy of augmentation of speech datasets

Kavisha Jayathunge

April 2021

1 Introduction

When gathering training data for a deep learning application, it is necessary to have a sufficiently large dataset - if it is too small, it might cause the model to over-fit to the data when training. Augmentation is the practice of artificially increasing the size of a dataset by adding new, meaningful data points to it. In the case of audio, this can be done by in transforming either the raw waveform or a latent representation of a given piece of audio. This report will describe an audio data augmentation method and its application to the LJSpeech dataset[1], which will be used to train the Glow-TTS[2] Text To Speech (TTS) model. It is expected that using an augmented dataset would result in a faster model convergence rate than using the base dataset.

2 The Mel spectrogram

Applying a Fourier transform to a time-varying signal decomposes it into its constituent frequencies over its entire time domain. This means that we lose information about how frequencies change over the run-time of the signal. To solve this problem, Short-Time Fourier Transforms (STFT) are used. These work by dividing the signal into time frames and computing the Fast Fourier Transform of each frame in turn.

Given a signal with sampled at a particular time $x(n)$ and a window function $w(n)$, its STFT S at a particular time frame m and frequency bin index λ can be calculated according to the equation

$$S(m, \lambda) = \sum_{n=0}^{N-1} x(n + m \cdot H) \cdot w(n) \cdot e^{-i2\pi n \frac{\lambda}{N}} \tag{1}$$

where $w(n)$ is the window function, N is the length of the window and H is the hop length (the distance the window is moved to get to the next time frame). To get the frequency ϕ corresponding to a particular λ

$$\phi(\lambda) = \frac{\lambda \cdot s_r}{N} \tag{2}$$

where s_r is the sampling frequency of the audio signal. Equation 1 shows that the parameters m and H determine the starting point of the summing process within the signal. The parameter N determines how many samples from this starting point are to be summed. This summation is done for each frequency index in the range $[0, \frac{N}{2}]$. Then, m is incremented by 1, "sliding" the window function over to the next time frame and repeating. By the end of this process, we are left with τ time frames and F Fourier coefficients for each time frame[1], giving us an idea of how the frequencies change over time. The spectrogram matrix $\mathbf{S}$ is in the form

[1]Here, $\tau = \frac{\Gamma - N}{H} + 1$ and $F = \frac{N}{2}$, where Γ is the total number of samples in the signal x.

$$\mathbf{S} = \begin{bmatrix} s_{0,0} & \cdots & s_{0,\tau} \\ & \vdots & \\ s_{F,0} & \cdots & s_{F,\tau} \end{bmatrix} \tag{3}$$

The window function used in the current implementation was a Hann, s_r was 22050, N was 1024 and H was 256.

Mel-scale spectrograms are an extension of this idea. Frequencies measured in Hz do not accurately represent the subjective perception of sound by humans. Where the Hz scale is linear, the Mel scale is logarithmic: distances between lower frequencies (which humans are good at telling apart) are greater than distances between higher frequencies (which humans are poor at telling apart). Hz spectrograms have many more frequency bins than necessary for this reason, and Mel spectrograms can reduce this dimension, while still preserving features that are relevant to human perception.

To convert the Hz spectrogram $\mathbf{S}$ to its Mel equivalent, a Mel filter bank must be constructed using the minimum Hz frequency f_{min}, the maximum Hz frequency f_{max} and the desired number of Mel channels C. The Mel filter bank matrix $\mathbf{J}$ has the form

$$\mathbf{J} = \begin{bmatrix} j_{0,0} & \cdots & j_{0,F} \\ & \vdots & \\ j_{C,0} & \cdots & j_{C,F} \end{bmatrix} \tag{4}$$

Please refer to Figure 1 for a visual representation of this. The final Mel spectrogram $\mathbf{Y}$ is calculated by applying the filter bank to the Hz spectrogram

$$\mathbf{Y} = \mathbf{J} \cdot \mathbf{S} = \begin{bmatrix} y_{0,0} & \cdots & y_{0,\tau} \\ & \vdots & \\ y_{C,0} & \cdots & y_{C,\tau} \end{bmatrix} \tag{5}$$

Therefore, the Mel spectrogram has dimension $C \times \tau$. In this implementation, f_{min} was 0, f_{max} was 8000, and C was 80.

3 Augmentation and training

The Mel spectrogram $\mathbf{Y}$ was subjected to the following augmentation policies, which were adapted from work done by Park et al[3]. and Hwang et al[4]. Each audio clip in the LJSpeech dataset (which consists of 13,100 .wav files) was converted into a Mel spectrogram and saved to disk twice - once without any augmentation and once with the policies described below. The augmented dataset therefore contained 26,200 Mel spectrograms.

3.1 Frequency and time masking

The following policies were applied to each audio clip in the LJSpeech dataset. Note that f_0 is defined in such a way to avoid picking frequency bins that are out of range for the spectrogram's dimensions. In this implementation, Φ was 3, N_f was 2, T was 4 and N_t was 2.

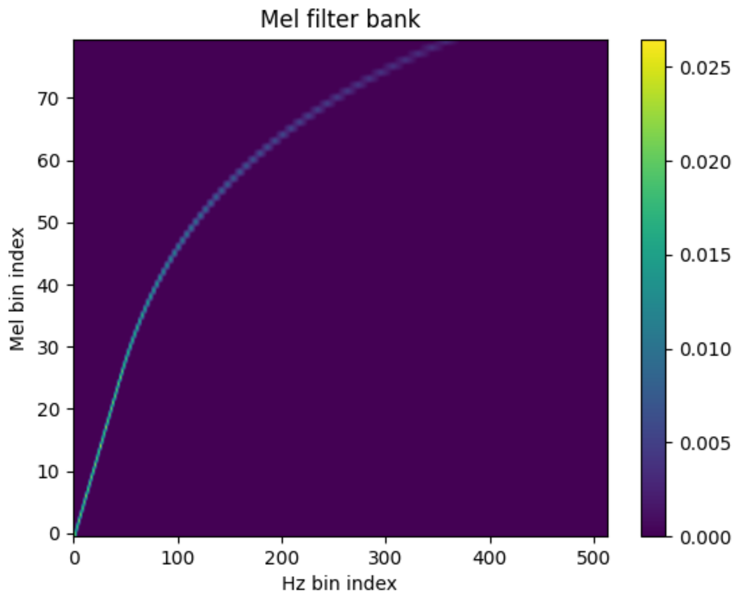

Figure 1: A visualisation of the filter bank used in this implementation

Algorithm 1: Frequency masking

inputs : Mel spectrogram $\mathbf{Y}$ of dimension $C \times \tau$, frequency masking parameter Φ, number of iterations N_f

output: Frequency masked Mel spectrogram $\mathbf{Y}$

```
1  $iteration \leftarrow 0$;
2  $\mu \leftarrow \texttt{minValue}(\mathbf{Y})$;
3  while $iteration < N_f$ do
4      $f \leftarrow \texttt{randomInt}(0, \Phi)$;
5      $f_0 \leftarrow \texttt{randomInt}(0, C - f)$;
6      for $i \leftarrow f_0$ to $f + f_0$ do
7          for $j \leftarrow 0$ to $\tau$ do
8              $\mathbf{Y}[i][j] \leftarrow \mu$
9          end
10     end
11     $iteration \leftarrow iteration + 1$ ;
12 end
```

Algorithm 2: Time masking

```
inputs : Mel spectrogram $\mathbf{Y}$ of dimension $C \times \tau$, time masking parameter $T$, number of
         iterations $N_t$
output:  Time masked Mel spectrogram $\mathbf{Y}$

1  $iteration \leftarrow 0$;
2  $\mu \leftarrow$ minValue($\mathbf{Y}$);
3  while $iteration < N_t$ do
4      $t \leftarrow$ randomInt($0, T$);
5      $t_0 \leftarrow$ randomInt($0, \tau - t$);
6      for $i \leftarrow 0$ to $C$ do
7          for $j \leftarrow 0$ to $\tau$ do
8              if $t_0 \leq j \leq t_0 + t$ then
9                  $\mathbf{Y}[i][j] \leftarrow \mu$
10             end
11         end
12     end
13     $iteration \leftarrow iteration + 1$ ;
14 end
```

3.2 Training

The Glow-TTS model was trained in several runs with sucessively smaller dataset sizes. This was done to determine the size of dataset for which augmentation resulted in better convergence rates than the unaugmented. This introduces the parameter γ, which is the fraction of the original dataset used to train a particular version of the model. Synthesiser models were trained with $\gamma = 1$, 0.25 and 0.125 for augmented and unaugmented databases. The model was first trained on each dataset for 112 epochs. For $\gamma = 1$, this translates to a training time of 14 hours for the base model and 26 hours for the augmented one. All other hyperparameters as seen in the original Glow-TTS paper[2] were left unchanged.

4 Results

Figure 2 shows that using a dataset augmented using the policies described in Section 3 produces no significant effect on the convergence of the model. This is also seen in subjective terms - each model was used to generate audio for a number of sentences[2], and while there are subtle differences in each case, the augmented model doesn't seem to be better than the base version.

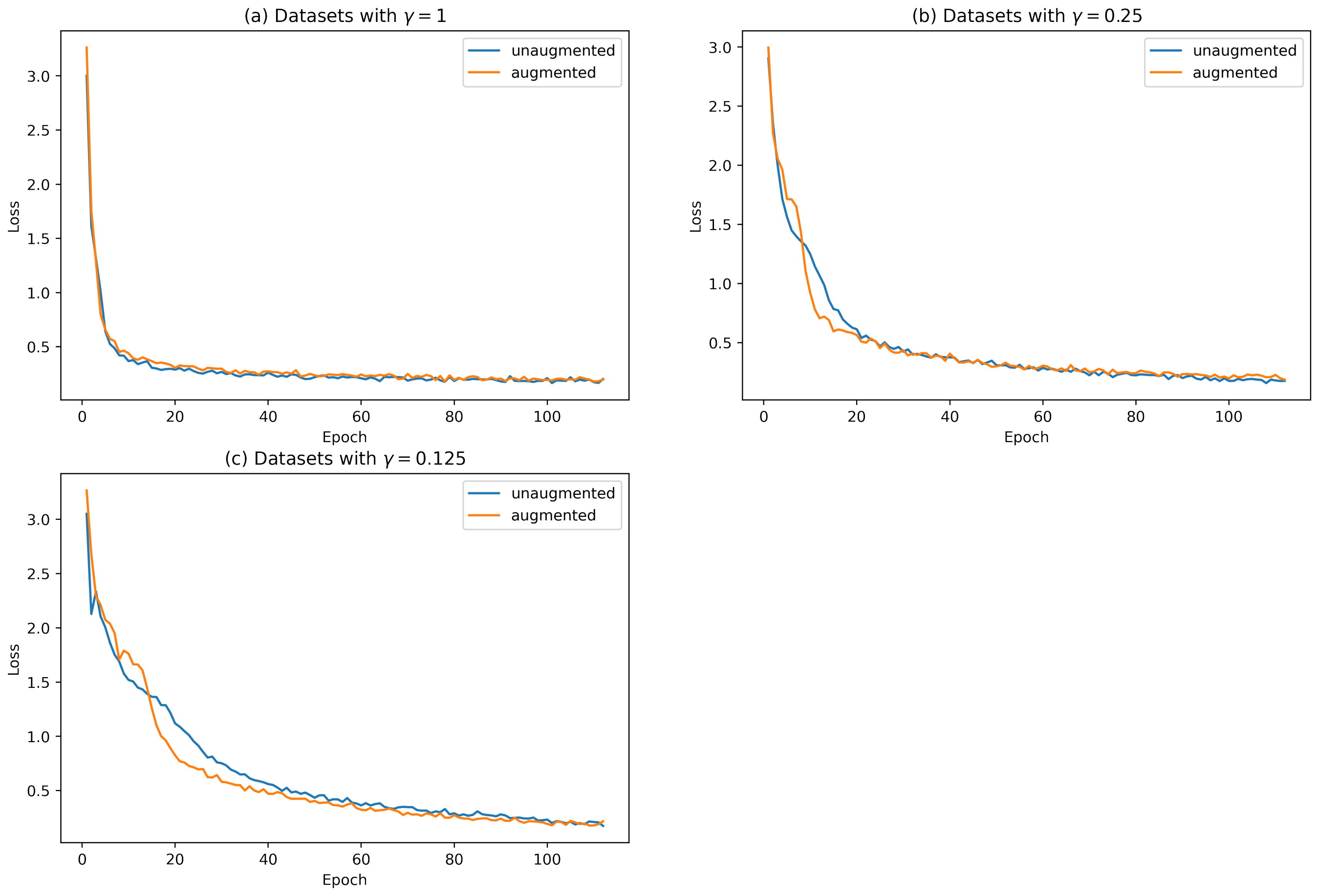

Figure 2: Training losses across augmented and unaugmented datasets of various sizes.

5 Conclusion

Audio augmentation is a method used to increase the size of a speech dataset in various applications. Augmentation policies consisting of frequency and time step masking have seen success in the domains of Automatic Speech Recognition (ASR) and Voice Conversion, but show no noticeable gains in TTS.

[2]These clips can be found at https://github.com/Intel-PA/audio-augmentation/tree/main/generated_wavs

Audio augmentation in ASR for example is motivated by the need of the model to be robust to partially garbled, missing, or otherwise 'non-perfect' speech. However, using such augmentations in training a TTS model might not be so useful since we're interested in producing speech, not interpreting it. This could be the reason that the simple techniques of frequency and time masking did not show any gain in the TTS domain, and suggests that another as of yet unexplored augmentation method might be more effective.

References

[1] K. Ito and L. Johnson, "The LJ Speech Dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[2] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," Oct. 2020. [Online]. Available: https://arxiv.org/pdf/2005.11129.pdf

[3] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech 2019*. ISCA, Sep. 2019. [Online]. Available: https://doi.org/10.21437/interspeech.2019-2680

[4] Y. Hwang, H. Cho, H. Yang, D.-O. Won, I. Oh, and S.-W. Lee, "Mel-spectrogram augmentation for sequence to sequence voice conversion," Jun. 2020. [Online]. Available: https://arxiv.org/pdf/2001.01401.pdf