VRIJE
UNIVERSITEIT
BRUSSEL

Master thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science In de Ingenieurswetenschappen: Computerwetenschappen

VARIATIONAL GREEDY INFOMAX

Towards independent and interpretable representations

Fabian Denoodt

2022-2023

Promotor(s): Prof. Dr. Bart de Boer
Science and Bio-Engineering Sciences

VRIJE
UNIVERSITEIT
BRUSSEL

Proefschrift ingediend met het oog op het behalen van de graad van Master of Science In de Ingenieurswetenschappen: Computerwetenschappen

[DUTCH] VARIATIONAL GREEDY INFOMAX

[Dutch] Towards independent and interpretable representations

Fabian Denoodt

2022-2023

Promotor(s): Prof. Dr. Bart de Boer

Wetenschappen en Bio-ingenieurswetenschappen

Contents

Chapter 1

Experiments

In this chapter, we evaluate the effectiveness of the latent representations obtained from Variational Greedy InfoMax (V-GIM) in comparison to its non-variational counterpart, GIM, by training both models on sequential data from the audio domain. To assess the quality of the representations, we project them into a two-dimensional space using t-SNE and analyse the emergence of potential clusters. Moreover, we employ a linear classifier that takes these representations as input, and evaluate its accuracy to gain insights into these models their performance. To further examine the efficacy of these representations, we investigate their potential for downstream task generalisation by training the linear classifier on smaller subsets of the dataset and assessing the amount of labelled data required to achieve satisfactory performance. Finally, we delve into the underlying structure of V-GIM's representations by training a decoder on top of each of V-GIM's modules, providing insights into their composition.

1.1 Experimental details GIM and V-GIM

GIM and V-GIM are trained on raw speech signals of fixed length sampled at 16kHz. The dataset consists of 851 audio files and is randomly shuffled and split up into 80% training data (680 files) and 20% test data (171 files). Each audio file consists of a single spoken sound consisting of three consonants and three vowels, where the consonants and vowels alternate each other. Some examples are the sounds "gi-ga-bu" and "ba-bi-gu". The words consists of three syllables from the following list: ba, bi, bu, da, di, du, ga, gi and gu. All the sounds are spoken by the same person, at a constant and peaceful tone. We crop the files to a fixed length of 10240 samples, or 640 milliseconds, which is slightly longer than half a second.

GIM and V-GIM are trained on the same architecture, consisting of two encoder modules $g^1_{enc}(\cdot)$ and $g^2_{enc}(\cdot)$, no autoregressor module $g_{ar}(\cdot)$ is used. Modules are trained in parallel. Each module consists of solely convolution and pooling layers. Since the architecture contains no fully connected layers, the length of the audio files can therefore be variable during inference. The architecture details for the two modules are presented in tables 1.1 and 1.2. ReLu non-linearity is applied after each convolution, except in the last layer in each module. No batch normalisation is used. The number of hidden channels produced by each convolution and pooling layer remains constant, at 32 channels. In each module, data flows synchronously from one layer to the successive layer, except in the final two layers. These run in parallel and both take the same input from the preceding layer. This architecture choice is related to the parametrisation trick we discussed in section **??**. One layer generates $\boldsymbol{\mu}$ and the other $\boldsymbol{\sigma}$ such that $\mathbf{z}^m_t$ can be computed via the reparametrisation trick. The architecture is visualised in figure 1.1. An input

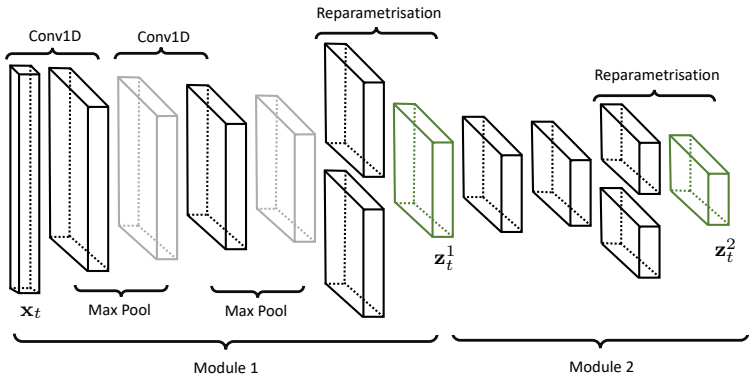


Figure 1.1: Visualisation of V-GIM's architecture.

signal of length 1×10240 is downsampled to 32×52 by the first module and to 32×13 by the second module. Due to overlap, each latent representation $\mathbf{z}_t^M = \mathbf{z}_t^2$ captures 64ms of speech.

Layer	**Kernel Size**	**Stride**	**Padding**
Conv1D	10	4	2
Max Pool	8	4	0
Conv1D	8	3	2
Max Pool	8	4	0
In parallel:			
Conv1D	3	1	2
Conv1D	3	1	2

Table 1.1: Module 1 architecture.

Layer	**Kernel Size**	**Stride**	**Padding**
Conv1D	6	2	2
Conv1D	6	2	2
In parallel:			
Conv1D	3	1	1
Conv1D	3	1	1

Table 1.2: Module 2 architecture.

Importantly, the reparametrisation trick is included in our implementation of GIM's architecture as well. However, by enforcing no constraints on the latent space, which is achieved by assigning $\beta = 0$ in $\mathcal{L}_{\text{V-NCE}}$, we observed that $\boldsymbol{\sigma}$ moves towards $\mathbf{0}$ such that all the randomness from sampling from a Gaussian distribution is removed. As such, our implementation of GIM is equivalent to GIM introduced in [1] but with an altered architecture.

Both GIM and V-GIM are trained using the Adam optimiser for 800 epochs with an exponential learning rate scheduler [2], with initial learning rate $lr_{init} = 0.01$ and $\gamma = 0.995$, such that lr is updated as follows:

$$lr_{epoch} = \gamma * lr_{epoch-1}$$

The batch size is set to 171, which is exactly the size of the test set. The number of patches to predict in the future k, is set to 12. Implementation details with regards to drawing negative samples for $f_k^m(\cdot)$ remain identical to the experiments from [3] and [1]. The regularisation importance term β is set to 0 for GIM and 0.0035 for V-GIM, which is the largest value we could obtain without causing posterior collapse.

After training V-GIM and GIM for 800 epochs, we train linear multi-class classifiers for 100 epochs on their respective latent representations for every module, evaluated using Cross-Entropy [4]. We set $lr = 0.001$ and use the Adam optimiser without scheduler. The classifier is tasked to predict the syllable corresponding to its latent representation. Details on the algorithm used

to split up audio files by individual syllables is provided in appendix A. Speech waves are zero-padded at both the beginning and the end to achieve a fixed length between syllables. The corresponding latent representations thus all consist of a fixed number of time frames $\mathcal{T}$ and 32 channels. Here, $\mathcal{T}$ is different depending on the module. The classifier consists of a single connected layer with $32 \times \mathcal{T}$ input nodes and 9 output nodes, conforming the number of input features and number of classes respectively. We reshuffle the dataset and use 80% for training and 20% for testing.

1.2 Results GIM and V-GIM

1.2.1 Training error

Figure 1.2 displays the loss curves corresponding to GIM ($\beta = 0$) and V-GIM ($\beta = 0.0035$) for both modules. In the first module, the test loss for GIM is lower compared to V-GIM. This is as expected because V-GIM includes an additional regularisation term in its loss function. In the second module, both GIM and V-GIM have loss curves that converge to lower values than their previous module. This suggests that the task becomes easier for the subsequent modules, indicating that the preceding module does indeed make simplified representations for the speech data.

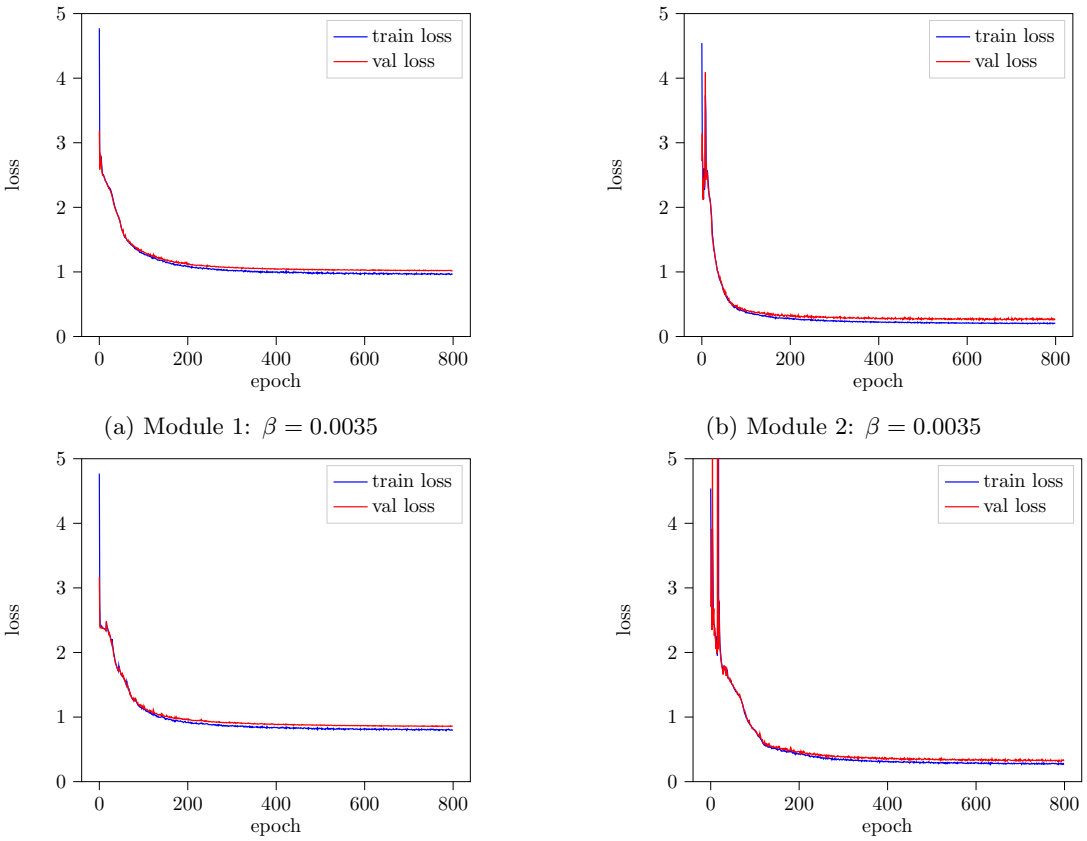
Interestingly, in the second module, V-GIM has a lower loss compared to GIM, even though V-GIM includes the regularisation term in its loss function. We argue that this is related to internal covariate shift, which we discussed in section **??**. Using a relatively large learning rate for GIM and V-GIM leads to a significant "drift" in the distribution of activations during training, making it difficult for GIM's successive modules to keep up with these changes. V-GIM is less susceptible to this issue due to the internal batch-normalisation mechanism that is built into the regularisation term. Consequently, V-GIM's modules can train with a larger learning rate without affecting the performance of successive modules, resulting in faster convergence.

1.2.2 t-SNE

We project the latent representations obtained from each module to a two-dimensional plane using t-SNE [5]. This enables us to observe potential clusters, as similar data points are mapped close together, while dissimilar points remain far away. Similar to the classifier discussed in section 1.1, t-SNE is trained on flattened representations that are split into individual syllables, without performing any pooling. We run t-SNE with random initialisation, perplexity of 30 and a learning rate of 200 for 1000 iterations.

The graphs for GIM and V-GIM are displayed in figure 1.3. T-SNE was not provided any information about the class labels. Syllables containing the vowel "a" are represented with a red tint, "u" with green, and "i" with blue. We observe similar results in the first module of both GIM and V-GIM. T-SNE identifies two clusters, with one cluster corresponding to syllables containing an "a" and the other cluster to containing "u" or "i". Within the "u/i" cluster, there is still a grouping of "u" and "i" data points. However, distinguishing between "b", "d" or "g" is more challenging as data points within the clusters are entangled.

In the second module, we observe more significant differences between GIM and V-GIM. In V-GIM's second module, there is clearer separation between the "i" and "u" syllables, indicating that the second module contributes to improved latent representations. However, distinguishing between the pronounced consonant remains difficult. On the other hand, GIM's second module does not appear to have converged well, as t-SNE struggles to separate the representations into



(a) Module 1: $\beta = 0.0035$

(b) Module 2: $\beta = 0.0035$

(c) Module 1: $\beta = 0$

(d) Module 2: $\beta = 0$

Figure 1.2: Training and validation loss (GIM: $\beta = 0$, V-GIM: $\beta = 0.0035$).

meaningful clusters. Consequently, syllables are mixed together without much structure. This further emphasises how GIM is affected by internal covariate shift, while V-GIM is not.

1.2.3 Classification performance

Table 1.3 presents the accuracies of the linear classifier, which aims to predict the syllable corresponding to the latent representation. There are a total of 9 different syllables, such that a random model would obtain an accuracy of 11%. While the t-SNE plots in figure 1.3 have demonstrated that vowels can be distinguished relatively easy, differentiating consonants poses a greater challenge. As a result, both GIM and V-GIM show mediocre test accuracies, ranging between 53% and 54% in the first module, and further decreasing in the second module. These accuracy values, coupled with the t-SNE plots, indicate that information regarding the consonants may no longer be adequately captured in GIM and V-GIM's latent representations. Moreover, the performance continues to decline when considering accuracies obtained from the second modules.

We believe that these mediocre accuracies can be attributed to the limited mutual information between temporally nearby patches. GIM and V-GIM assume data that adheres to the slowly changing features assumption, which is necessary to maximise the mutual information between

Method	Accuracy Module 1 (%)	Accuracy Module 2 (%)
GIM	54.39	28.07
V-GIM	52.92	41.15

Table 1.3: Accuracies obtained on the test dataset.

the latent representations of temporally nearby data patches. Consequently, abrupt changes in the patches are discarded from the latent representations. However, in general, the words spoken in the dataset consist of longer durations for vowels compared to consonants. For instance, in the syllable "ba", the phoneme "b" is usually pronounced for a shorter duration than the "a". This can cause a problem when the phone "b" is not captured over multiple patches, as only the mutual information between the patches is kept. Additionally, latent representations are optimised to maximise the mutual information with the future k patches. Since k remains fixed over the different modules and deeper modules capture longer time windows, the latent representations of deeper modules must capture more information, while the number of channels remains relatively small, at 32.

1.2.4 Distributions

Figures 1.4, 1.5, 1.6 and 1.6 depict the distributions of activations corresponding to an individual dimension. Each sub-figure depicts the distributions for a single module. We show distributions for the first 6 dimensions. We observe that V-GIM does indeed learn to constrain the latent space to the standard normal for most dimensions. Meanwhile in GIM's first module, we observe high and thin peaks, indicating that the same value is regularly predicted with little noise due to $\boldsymbol{\sigma} \approx 0$. The peaks are less dominant in the second module, which may be attributed to suboptimal convergence.

1.3 Generalisation study

In contrast to GIM, V-GIM's latent representations are samples from a distribution, resulting in a single patch of data to have multiple latent representations. In this section we examine whether a linear classifier with little annotated training data can benefit from this representation variability introduced by V-GIM. We train multiple multi-class classifiers with the same experimental details as discussed in section 1.1 but with a modification to the annotated training set. Each classifier is trained on a subset of the dataset, varying between 1 data point per class, all the way up to 128 data points. The batch size is set to the size of the subset. Training details for GIM and V-GIM remain unchanged, including the size of training set, which does not require any labels.

The test accuracies are shown in figure 1.8. Overall, we observe no performance benefit from V-GIM's representation variance. Performance in GIM and V-GIM's first modules remains consistent, regardless of the subset size. Meanwhile, differences in the second module are more prominent. However, this is related to the internal batch normalisation mechanism, discussed in section 1.2.1, resulting in faster convergence of V-GIM's modules.

1.4 V-GIM's Interpretability analysis

In the following sections we delve deeper into V-GIM's latent representations, aiming to gain understanding in the captured information and understand underlying structures. This is achieved

by employing a decoder on top of each of V-GIM's modules. By altering a representation's component values and observing the effect through the decoder, we can analyse the contained information in each individual dimension. As we argued in section **??**, this is only possible because V-GIM's latent space is optimised to be approximate to the standard normal. The decoder can then generalise to the altered representations as long as the representations are close to the origin.

1.4.1 Decoders for V-GIM

We employ two decoders, one for each module, which can be represented as follows: $D(\mathbf{z}_t^1) = \widetilde{\mathbf{x}}_t$ for the intermediate decoder and $D(\mathbf{z}_t^2) = \widetilde{\mathbf{x}}_t$ for the final decoder. This allows us to assess the information in both the final and the intermediate representations. Architecture details for the two decoders are provided in tables 1.4 and 1.5. An intermediate representation $\mathbf{z}_t^1$ with a shape of 32×1, capturing a single time step, is transformed into a speech signal $\widetilde{\mathbf{x}}_t$ with a shape of 1×448 (or 28ms). Similarly, the final representation $\mathbf{z}_t^2$ is transformed into a shape of 1×1024 (or 64ms).

Layer	**Kernel Size**	**Stride**	**Padding**
ConvTrans	3	1	1
ConvTrans	8	4	0
ConvTrans	8	3	2
ConvTrans	8	4	0
ConvTrans	10	4	2

Table 1.4: Decoder architecture for intermediate latent representations.

Layer	**Kernel Size**	**Stride**	**Padding**
ConvTrans	3	1	1
ConvTrans	8	3	2
ConvTrans	8	3	2

Table 1.5: Decoder architecture for final latent representations:

While the decoders can be optimised by minimising the Mean Squared Error (MSE) of the speech waves, this metric may not reflect well with the natural biases in human hearing. Humans perceive certain frequencies to be louder than others [6, 7]. To account for this, we instead minimise the MSE on the mel-frequency spectrograms, which are an adaptation of linear spectrograms that emphasise frequency bins based on perceptual hearing biases [8]. Additionally, we employ a logarithmic transformation to account for humans' logarithmic perceptual hearing [9]. Figure 1.9 illustrates an example of a log mel-spectrogram.

The loss function we use is the following:

$$\mathcal{L}_{\text{decoder}} = \frac{1}{n} \sum_{i=1}^{n} \left(\log(\text{MEL}(\mathbf{x}_t^{(i)})) - \log(\text{MEL}(\widetilde{\mathbf{x}}_t^{(i)})) \right)^2$$

In this equation, $\text{MEL}(\mathbf{x}_t^{(i)})$ represents the mel-spectrogram of the original signal $\mathbf{x}_t^{(i)}$, computed using the following parameters: the number of FFT bins set to 4096, the length of the hop between STFT windows set to 2048, and the number of filter banks set to 128. Other parameters are kept at their default values in PyTorch's MelSpectrogram implementation [10]. Logarithms in the equation are computed in base 10.

1.4.2 V-GIM representation analysis through decoder

Decoding arbitrary latent representations allows us to perceive the remaining information in the representation. It is worth mentioning that the reliability of this analysis is dependent on

the performance of the decoder. If it is not able to reconstruct certain features, this does not necessarily mean the information is not contained in the latent representation.

Training and validation loss curves for both decoders are shown in figure 1.10. Overall, in both decoders we observed that audio files could be reconstructed and the pronounced syllables were recognisable when listening to them. However, reconstructed sounds seemed noisy and information about the speaker's identity was unrecognisable, suggesting that this information is either no longer contained in the latent representations, or the decoders are not able to replicate it. Additionally, while we could vowels were easily identifiable, we observed that the decoder occasionally made mistakes with consonants. For instance decoding the latent representation for "gu" could occasionally result in an incorrect "bu" or "du". These observations are in line with the t-SNE plots classification performance we discussed in the previous section, further suggesting that consonant information may no longer be present in the latent representations.

Figures 1.11, 1.12 and 1.13 in Appendix C show decoded latent representations corresponding obtained from the *intermediate* decoder. In each figure, values from an individual dimension is manipulated. As such, changes to the speech wave give indications on what information is encoded in that dimension. Each representation starts as the zero-vector $\mathbf{0}$ but a single dimension is being modified, ranging between -2.68 and 2.68. The reconstructed audio wave is shown both in frequency and time domain.

We classify the dimensions and their contained information into one of three categories. Figures 1.11, 1.12 and 1.13 show an example of each category. The first category contains information about frequencies for a specific frequency bin. Altering this dimension will have strong influence on the magnitude of one or two frequency bins while other bins remain relatively unchanged. This is shown in figure 1.13, where changing the value of the 12th dimension towards -2.68 causes a peek around the 100 Hz frequency bin. The peek disappears when approaching 0. Meanwhile, when approaching +2.68, the magnitude of the 150 Hz frequency bin increases. Overall, we observed this kind of behaviour for roughly half of the 32 dimensions. The majority of dimensions were sensitive in the 100 Hz and 150 Hz frequency bins, while a few in the 175 Hz bin. These ranges are to be expected as the pitch range for men's voices is between 60 and 180 Hz [11].

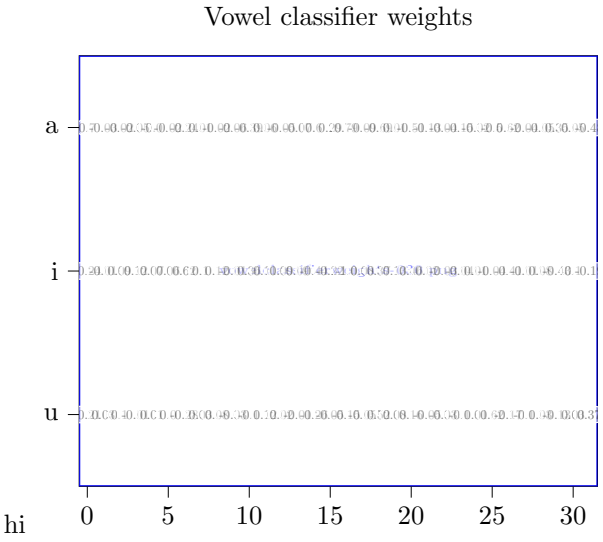
Dimensions that belong to the second category, do not only influence magnitude of frequencies in a specific bin, but also to the nearby bins. In this category, changes to a dimension will cause magnitude of a specific frequency to increase as well, but it will also take along its neighbouring frequencies, resulting in a smooth envelop with a single peak which gradually decreases over the neighbouring frequencies. This behaviour is shown in figure 1.11.

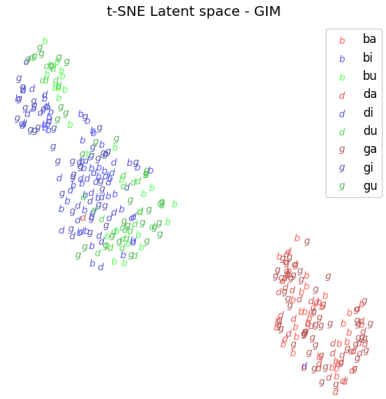
The final category of dimensions are those that do not seem to contribute anything to the reconstructed signal. Changes to the values in these dimensions cause very little change to the resulted speech wave both in time domain and frequency domain. These are either useless dimensions or the decoder was not able to capture their contribution. An example is shown in figure 1.12.

Final decoder...

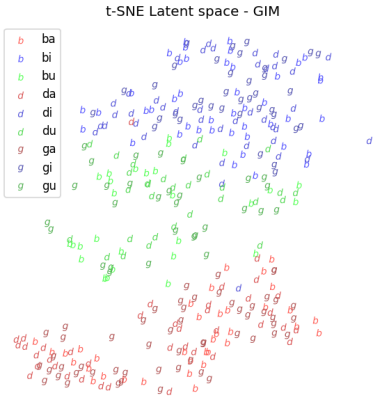
Graphs related to the interpolation experiment are shown in figures 1.11, 1.12, 1.13. We categorise the dimensions in V-GIM in three sections, the graphs present an example of each category.

1.4.3 Consonants

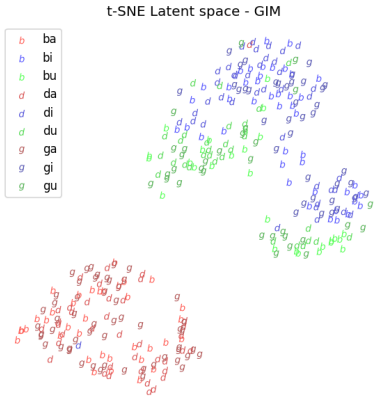




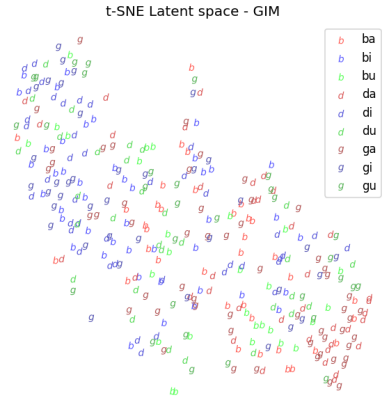
(a) Module 1: $\beta = 0.0035$



(b) Module 2: $\beta = 0.0035$



(c) Module 1: $\beta = 0$



(d) Module 2: $\beta = 0.0035$

Figure 1.3: t-SNE plots (GIM: $\beta = 0$, V-GIM: $\beta = 0.0035$).

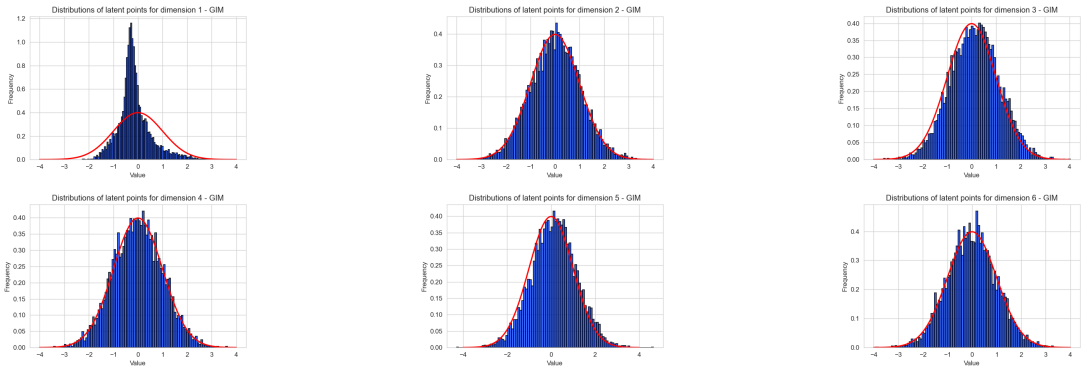


Figure 1.4: Module 1: $\beta = 0.0035$

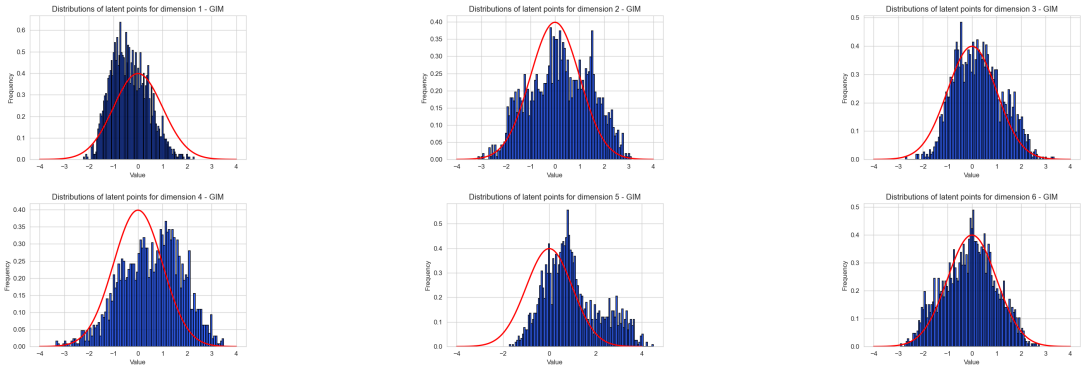


Figure 1.5: Module 2: $\beta = 0.0035$

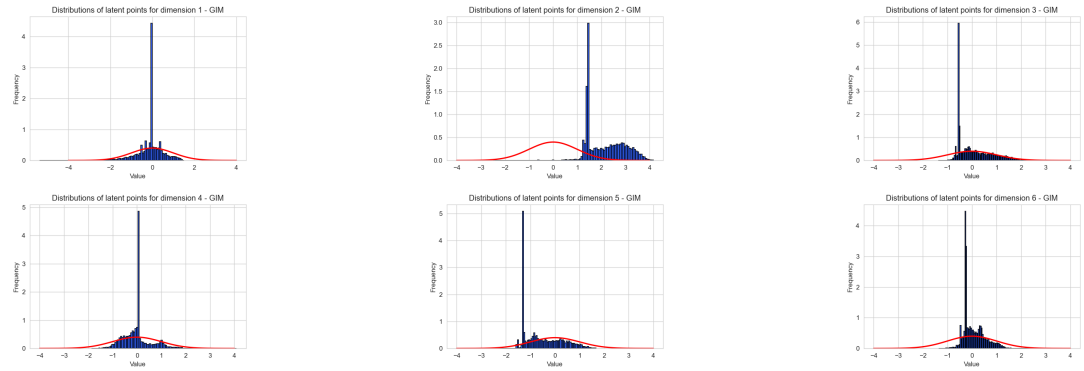


Figure 1.6: Module 1: $\beta = 0$

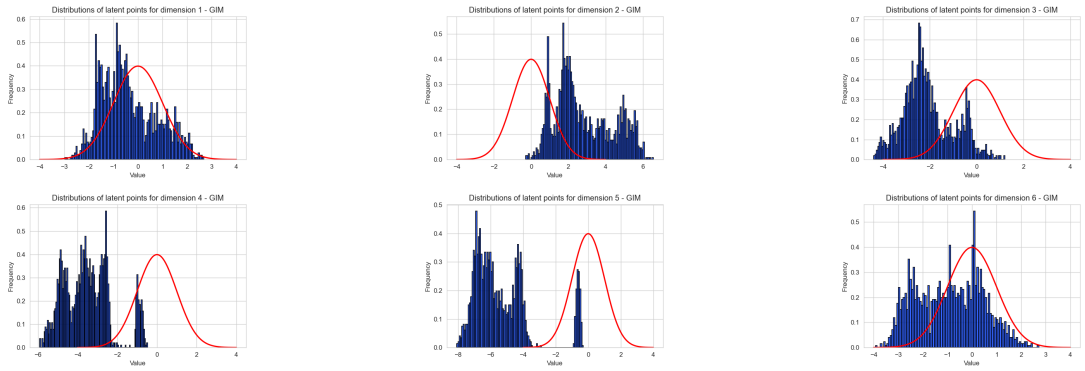


Figure 1.7: Module 2: $\beta = 0$

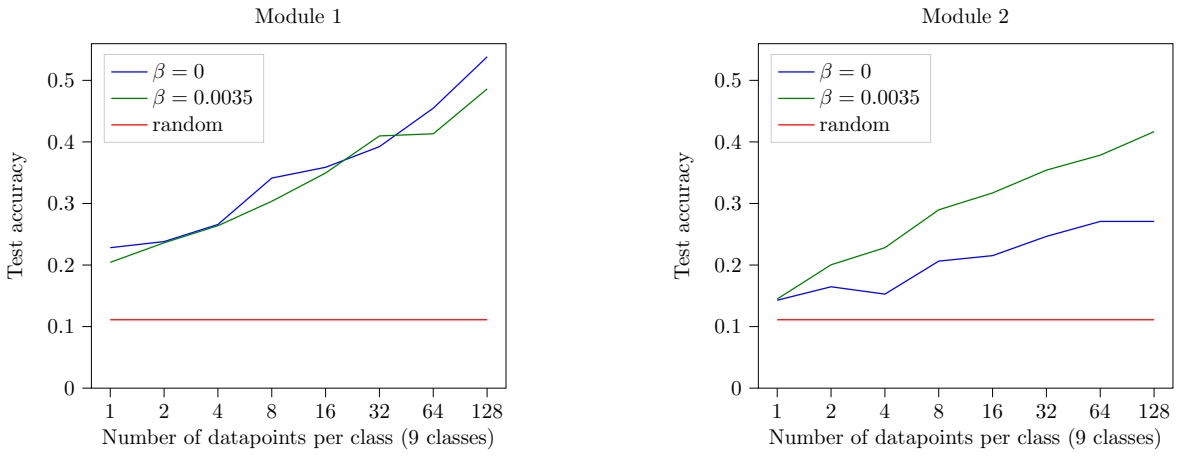


Figure 1.8: Test accuracy for different subset sizes

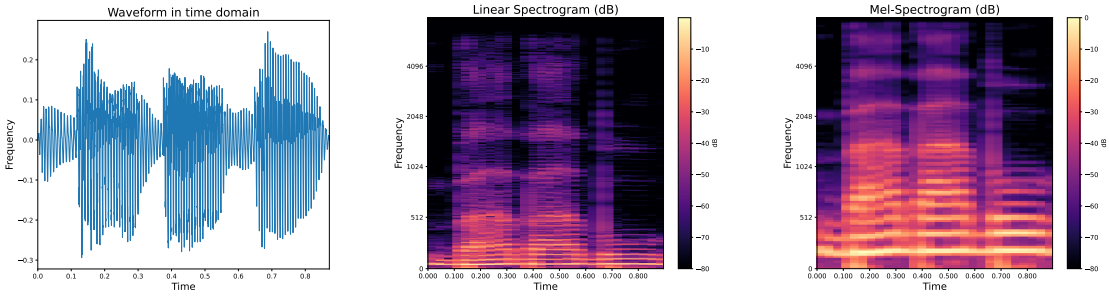


Figure 1.9: Example waveform and spectrogram for "bababu".

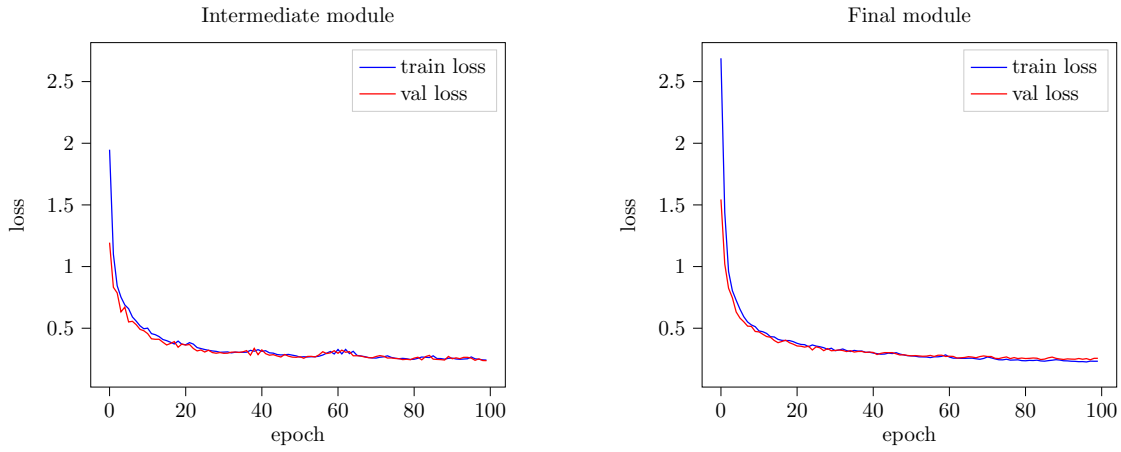


Figure 1.10: Training and validation loss for the two decoders, trained with V-GIM's representations.

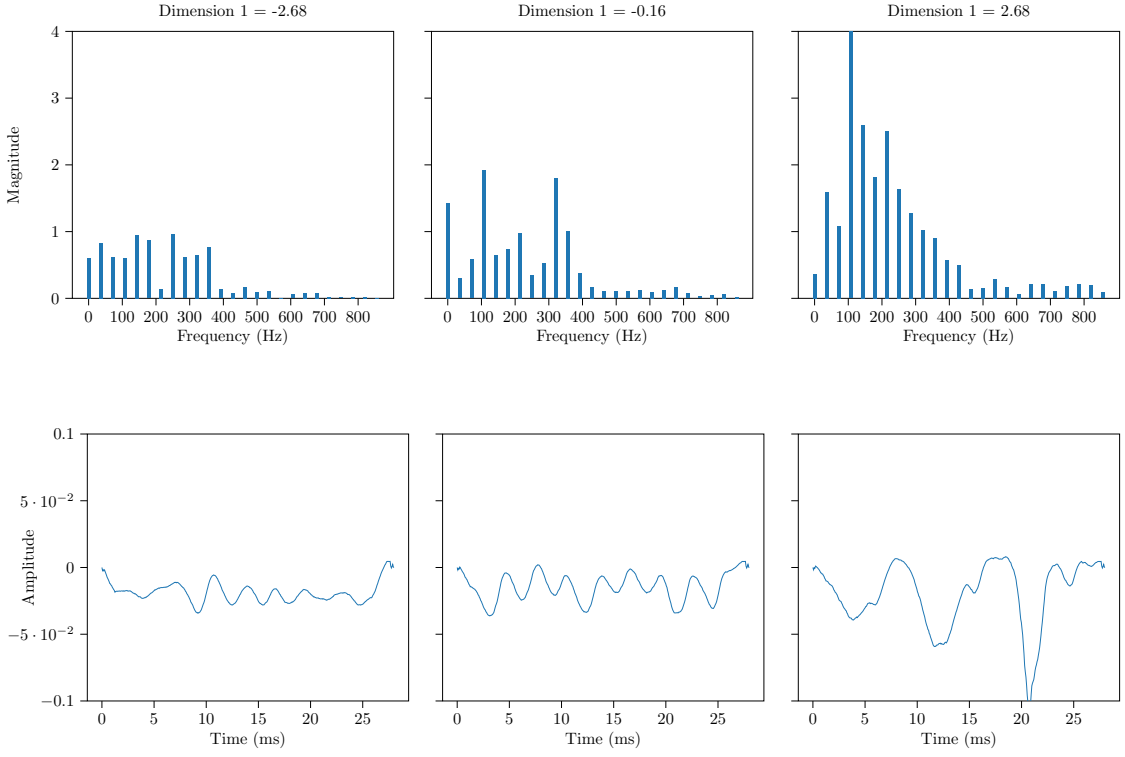


Figure 1.11: The first dimension is being modified, while other dimensions are fixed at 0. The dimension has influence on the 150Hz frequency band, but also neighbouring ranges.

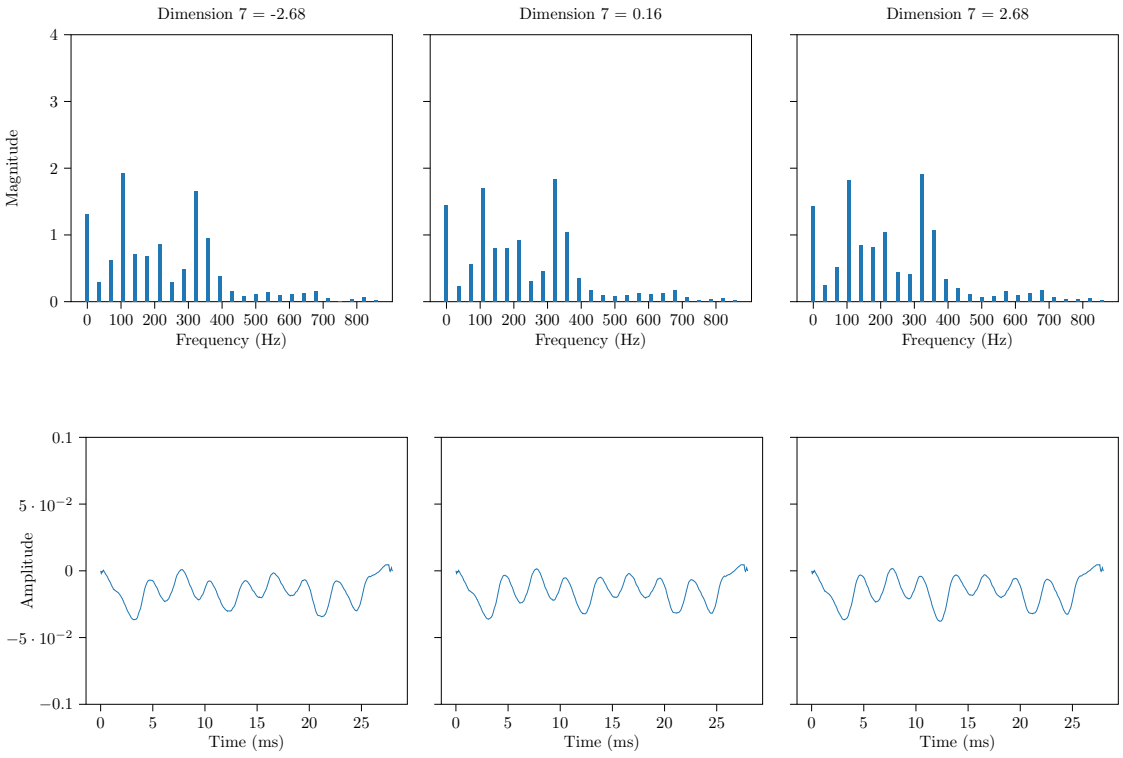


Figure 1.12: The seventh dimension is being modified, while other dimensions are fixed at 0. We observe no significant differences when adjusting, indicating that the dimension may not capture any information at all.

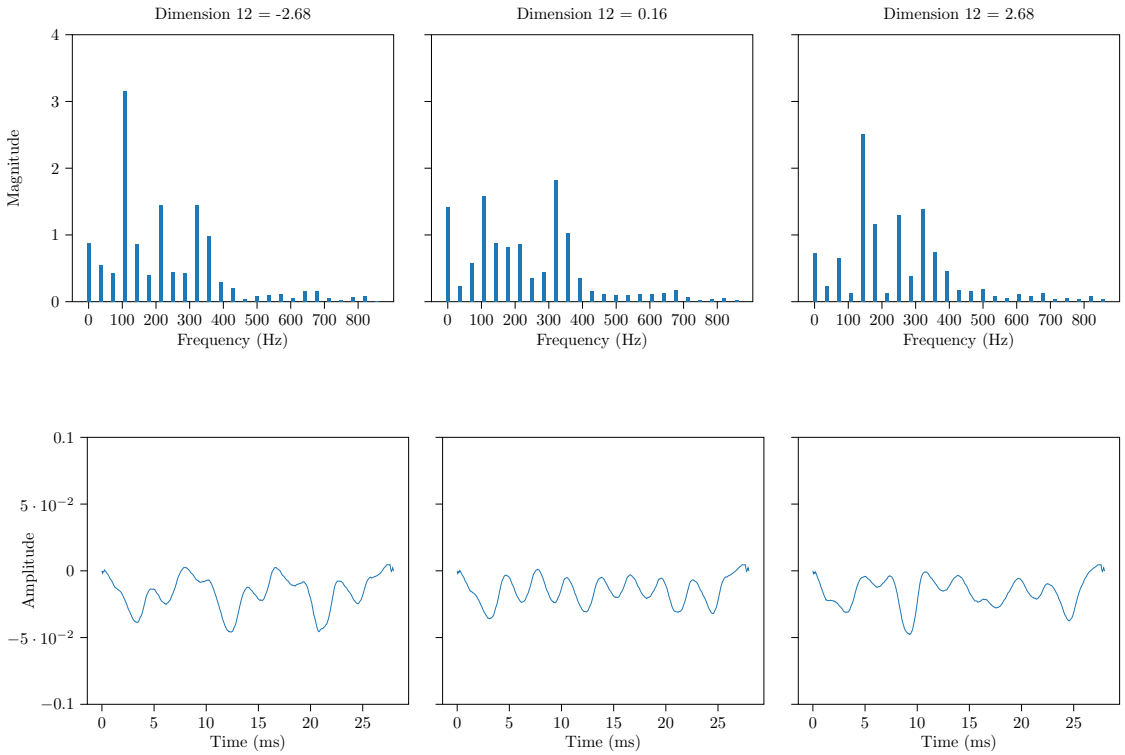


Figure 1.13: The twelfth dimension is being modified, while other dimensions are fixed at 0. Negative values cause a large spike around 100Hz while positive values fully remove the 100Hz and influences the 150Hz range instead.

Chapter 2

Discussion

– decaying learing rate: we train using decaying lr, because models must first learn distributions and goes too slow if lr is too small. and a learning rate scheduler ExponentialLR decay rate 0.995

— batch norm: - sindy didn't have issues of batch norm, but believe this is because each module consisted of a single layer, ours contain a number of layers. potentially: outputs from first module change too fast for second module to catch up.

while GIM argues to resolve memory constraints, not entirely true. In fact we even countered the opposite as containing multiple neural networks, each with their own personal loss function (the loss function is based on fk which contains parameters that must be learned), and thus for early layers where the sequence is still long, a lot of memory is required. We went for a compromise on GIM by splitting up the architecture in merely two modules, significantly reducing the memory constraints.

— The second module in GIM clearly doesn't have as much effect. This can be explained because there may not be as much common information anymore between the patches. There may be a source that says that cpc learns low level features, but the second module is supposed to learn more high level features, which cpc may have trouble with? —

Future work: - Related work in VAE shows that gradually increasing regularisation term, results in better disentengledment, while avoiding posterior collapse. could have a kldweight scheduler.

- not constrained solely to InfoNCE loss, the GIM architecture could work for other losses too that allow for greedy optimisation.

- I didn't add an autoregressor as i didn't find a performance benefit. Potentially, with larger architecture could further improve performance.

—- Towards production setting: encodings are thus optimised to be close the standard normal. When in a production environment and new data is given, could in fact have an idea of how well generalisation to the production data: eg via anomaly detection if encodings are too far away from center. = gives automated way of verifying generalisation.

can then maybe see to which data that doesn't generalise well via outliers.

—-

future work: - disentanglement should do more investations

— GIM: Modular training could incrementally increase numb of modules and observe performance increase for downstream tasks. based on this, could find smallest gim architecture depth which satisfies required accuracies.

—- interpretability: most dimensions sensitive around 75 to 150 hz. this is as expected as the adult man speaks around 80 to 180 hz.

— interpretability is only as good as the decoder. if a shitty decoder doesn't construct well, doesn't necessarily give correct conclusions about V-GIM.

- Explainability of latents is dependent on the performance of the decoder.
- Intermediate loss function with kld resulted in similar behaviour as batch normalisation. Resulting in faster convergence than without kld.
- We observed no quality loss in the learned representations. Data was equally easily separable.

Bibliography

[1] S. Löwe, P. O'Connor, and B. S. Veeling. Putting An End to End-to-End: Gradient-Isolated Learning of Representations. [Online]. Available: http://arxiv.org/abs/1905.11786

[2] Bhargav Lad. Guide to Pytorch Learning Rate Scheduling. [Online]. Available: https://kaggle.com/code/isbhargav/guide-to-pytorch-learning-rate-scheduling

[3] p. d. u. family=Oord, given=Aaron, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. [Online]. Available: http://arxiv.org/abs/1807.03748

[4] Y. Ho and S. Wookey, "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling," vol. 8, pp. 4806–4813.

[5] p. d. u. family=Maaten, given=Laurens and G. Hinton, "Visualizing Data using t-SNE," vol. 9, no. 86, pp. 2579–2605. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[6] E. Radkoff. Loss Functions in Audio ML. Sounds and Words. [Online]. Available: https://soundsandwords.io//audio-loss-functions/

[7] A. Li, R. Peng, C. Zheng, and X. Li, "A Supervised Speech Enhancement Approach with Residual Noise Control for Voice Communication," vol. 10, no. 8, p. 2894. [Online]. Available: https://www.mdpi.com/2076-3417/10/8/2894

[8] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779–4783.

[9] S. Braun and I. Tashev. A consolidated view of loss functions for supervised deep learning-based speech enhancement. [Online]. Available: http://arxiv.org/abs/2009.12286

[10] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch." [Online]. Available: https://openreview.net/forum?id=BJJsrmfCZ

[11] D. E. Re, J. J. M. O'Connor, P. J. Bennett, and D. R. Feinberg, "Preferences for Very Low and Very High Voice Pitch in Humans," vol. 7, no. 3, p. e32719. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3293852/

Appendices

Appendix A

Syllable classification through histogram segmentation

probabily bs: During inference, (in this context obtaining the latent representations for our input signals), depending on the length of the input signal, the length of the output latent representation will differ. If we wish to look at how separable latent representations are for syllables, the length can be variable. Some input sounds could be 6,600 samples, while others 8,800 samples. We therefore pad the syllables with zeroes in front and end of the signal, to obtain fixed length of equal to that of the longest syllable; 8,800 samples.

Training happens on longer data samples, and every **X** epochs t-SNE visualisations are made to observe evolutional of dis-entanglement.

Since the recordings are very consistent in loudness and are noise free, we can split up the files per syllable, obtaining three files per original sound (one for each syllable).

A sliding window is used of size 0.02 seconds. With a sample rate of 22050, this corresponds to roughly 500 samples per window. The maximum is computed for each window. Speech signals can then be split up when a severe dip happens in the signal. Regions where the amplitude is greater than 0.2 are considered **klinkers**, the regions with with lower values are considered **medeklinkers**. Apart from a few edge cases, this technique worked well enough for this purpose. In those cases, the splitting points closest to the one-third and two-third splitting points were considered.

ERR: OUDE AFBEELDINGEN ZIJN WEG

note: we do need a hard threshold which is based on the signal's intentisity level. One could consider the alternative approach of looking at the gradient at each point and selecting the points with largest negative gradient. This will work in many cases, however, not for temporal envelops which gradually move towards zero, s.t: A.1. Instead we use a dynamic threshold. This threshold is computed by creating transforming the signal into bins of 90'th percentile, creating a histogram of the single signal and applying otsu's image segmentation algorithm to obtain the threshold of that single audio sample. We also tried directly applying otsu to the moving average and maximum of the bins. This either gave a threshold that was too small or too large. the 90th percent resulted in an acceptable compromise.

Example where the explained strategy does not work:

Reference images for in the text:

audio padded to maximum length. (added zeros in front and back)

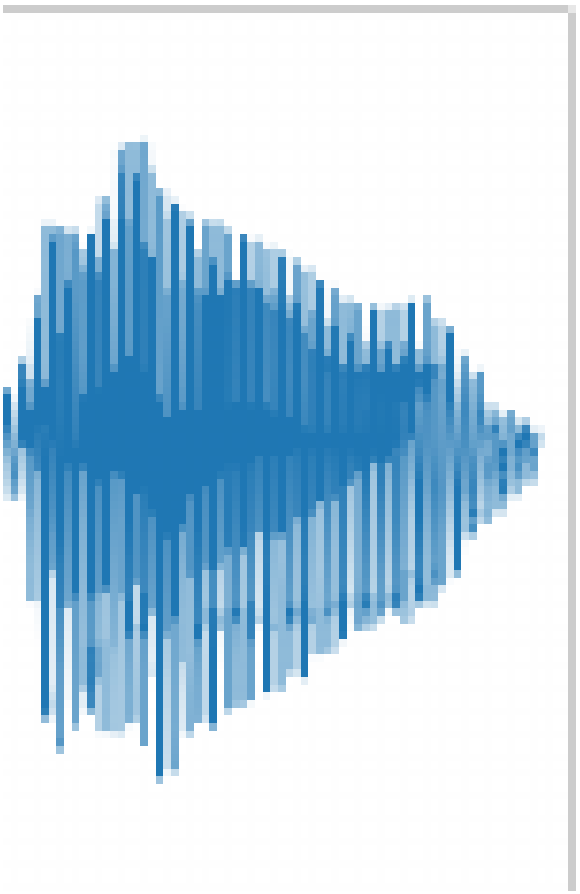


Figure A.1

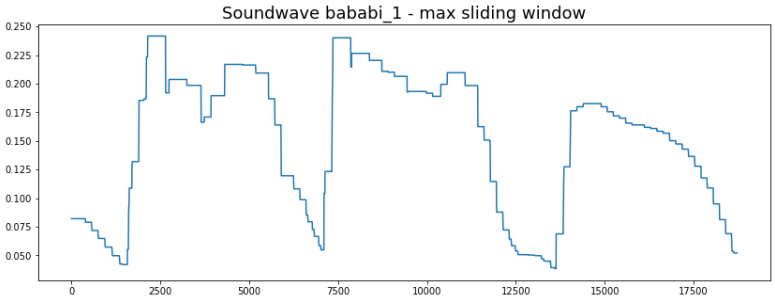


Figure A.2

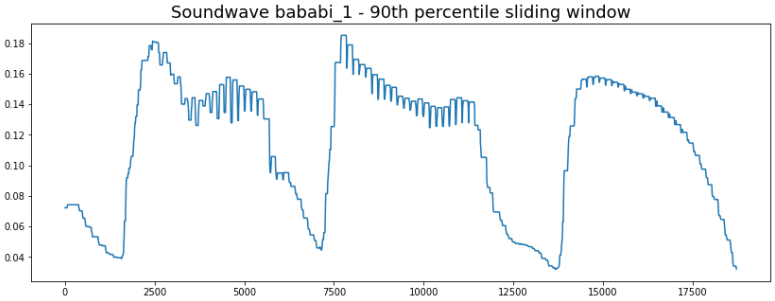


Figure A.3

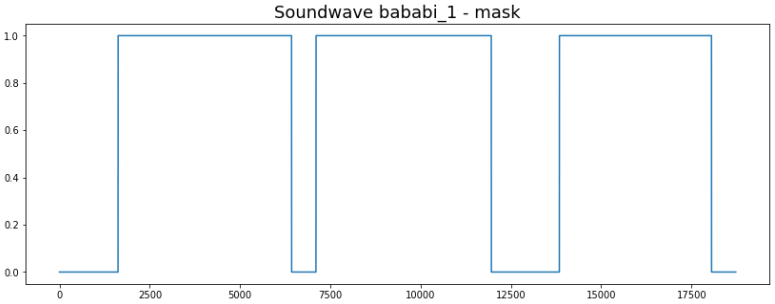


Figure A.4

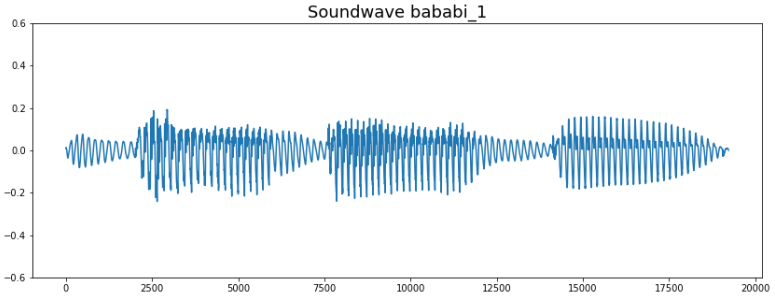


Figure A.5

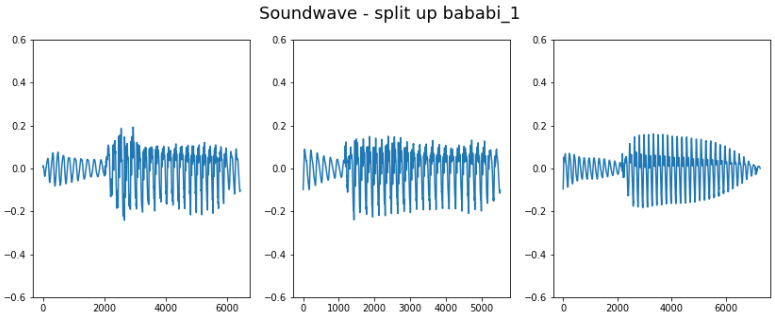


Figure A.6

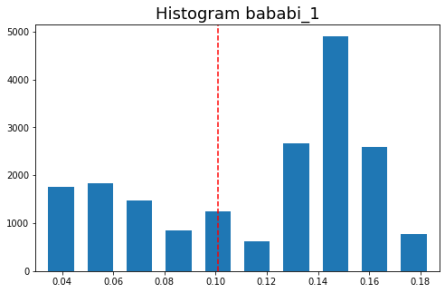

Histogram bababi_1
5000
4000
3000
2000
1000
0
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18

Figure A.7

Appendix B

Some more appendix

B.1 GIM: Activations visualisations

thought for later: its actually weird i was able to play enc as audio as enc is 512 x something so huh? that means that a lot of info is already in first channel? what do other 511 channels then contain? """ Observations: First layer decoded still contains the same sound, but with some added noise (could be because decoder hasn't trained very). However, the encoded first layer, still contains the exact sound as the original sound. It is however downsampled a lot -¿ from 16khz to 3khz """ thought for later: its actually weird i was able to play enc as audio as enc is 512 x something so huh? that means that a lot of info is already in first channel? what do other 511 channels then contain?

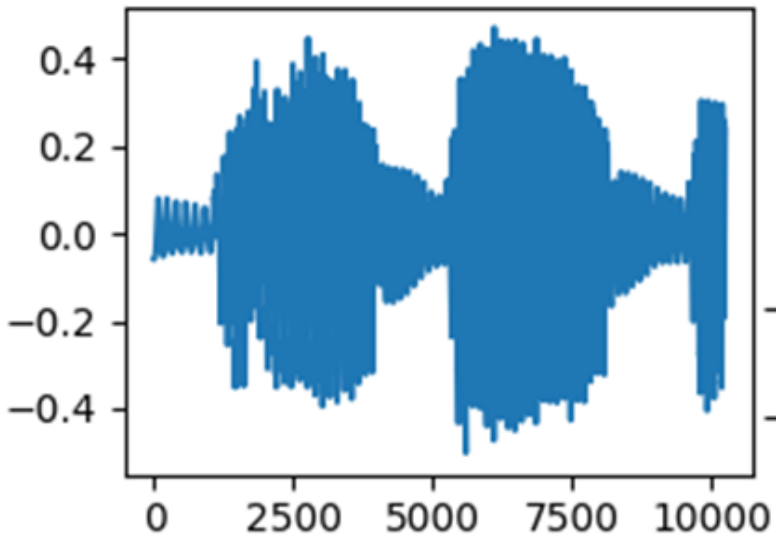


Figure B.1: "BA-BA-BA" time domain

No batch normalisation, so although channels appear to have larger activations than other channels, size of activation does not really say anything about information. eg activations 0.01 could still contain more information than 3.0 activation.

Figure B.2: Activations of the sound "BA-BA-BA" through GIM

Since the activations from convolutional neural networks, the order is still maintained. Hence, can align activations with original signal.

Observations in latent representations:

Layer 1: The activations of the first decoder still contain a lot of similarity with the original signal, in terms of structure. There is a lot of redundant data within the representation. Eg: the one channel could be replied

Layer 2

Layer 3:

Layer 4: Still notices multiple channels which have high activations when signal is has high amplitudes and small activations when amplitude is low.

Also activations which are high when volume is low. –¿ indicates that certain kernel weights are sensitive for **"klinkers"** and other kernels for **medeklinkers**. see B.3.

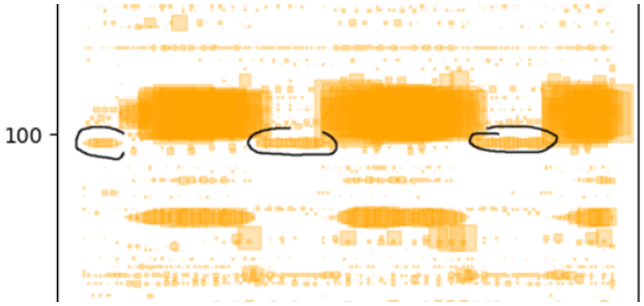


Figure B.3: zoomed in

Observe that activations happen in clusters/sequences. So it is usually a patch of signal samples that cause high activations. This could for instance indicate that both kernels are sensitive for the **medeklinker** "b", but sensitive for different features. eg the letter B has spoken sound "buh". so maybe one is sensitive for "b" and other for "uh".

Figure B.4 also nicely shows how different channels have clusters of activations at slightly different times.

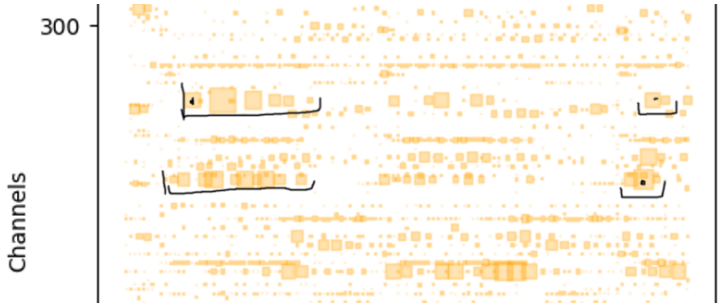


Figure B.4: Zoomed in

Appendix C

Interpolation

TODO REMOVE