

Inter-Voice Morphing using a Neural Network based Dimensionality Reduction Technique

A preliminary experiment

Alwin de Rooij & Tommaso Villanova

Seminar Speech Recognition 2009, LIACS, Leiden University

Abstract

In this paper we present a novel approach to voice synthesis and morphing. Based on experimental research in non-linear dimensionality reduction or auto-encoding we have set up an experiment to enable inter-voice morphing. This done by training a feed forward neural network with the auto association method and introducing an encoding layer, bottleneck layer and decoding layer. The aim of this project is encode phonemes in the auto-encoder through a training phase with phoneme sounds and with these auto-encoders construct a voice synthesis program. This paper describes a first set of experiments done and the choices made about the technologies used.

Introduction

Currently the state of the art in voice synthesis and voice morphing employs a variety of technologies. Some of the most used and researched techniques at this point is linear predictive coding (LPC). A technique where sounds are decomposed into a quasi-excitation signal and quasi-stationary filter coefficients which can be used for voice morphing [1]. Other often used techniques use spectral envelopes for synthesis and morphing, the spectral envelopes are generated based on a wide variety of concepts [2, 3, 4]. Also physical models of the vocal tract are employed [5, 6]. However many of these voice synthesis techniques lack a realistic quality. In this project we introduce a non-linear dimensionality reduction technique or auto-encoding method as a fundamentally novel way to synthesize voice sounds and hope to take the first steps in creating a novel voice synthesis method that may yield more realistic results, which in the future may prove to be a valuable addition to the research field of voice and speech synthesis. In this project we will limit the experimentation to the synthesis and auto-encoding of phonemes.

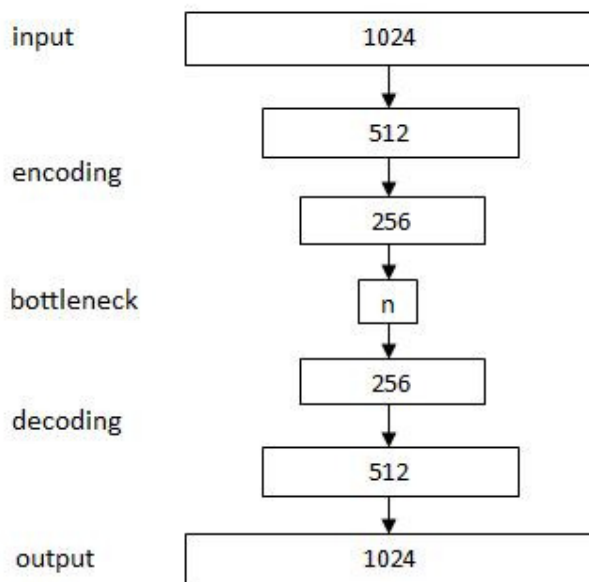
auto-encoding is a special niche in the research field of machine learning and more specific artificial neural networks. Early work was already done in 1988 by Baldi & Hornik [7]. They showed that dimensionality reduction using feed forward networks could extract principal components from data. Short thereafter it was shown that these networks could be used as a method extract features from images and develop compact encodings [8]. In his paper about non-linear dimensionality reduction DeMers showed that the networks in combination with principal component analysis minimize the amount of features and can reduce a very high dimensional data space to a small set of points on a reduced n-dimensional manifold describing that particular type of data. In one of his experiments he trained the auto-encoder with a dataset containing pictures of faces which were all well centered. The input space was 64x64, reduced to 50 principal components and after training, all faces could be reduced to only 5 points in space. What made his experiments interesting for this project is that he was able to use the trained neural network to synthesize the pictures of the faces in the dataset but also non existing faces [9]. More recently Hinton continued similar work in auto-encoding by devising a suitable pre-training

method using restricted Boltzmann Machines and after pretraining fine tune the neural net with a back propagation algorithm [10, 11].

When successful there are some interesting applications for this technique that can be considered. Most of them in the field of TTS. Projects such as MBROLA could benefit from this project [12].

Approach

To define an auto-encoder. In the universal approximation theorem it is proven that a universal approximator can be constructed with only one hidden layer with a sigmoid activation function and an output layer with a linear activation function. Universal approximation is possible as long as you can use the right finite amount of nodes [13]. An auto-encoder exploits this trait and is in fact a special type of feedforward neural networks. Its architecture differs from other feedforward neural networks in that it has a large amount of hidden layers that are significantly smaller than its input and output layers. The most prominent feature of an auto-encoder is the middle hidden layer, which is called the bottleneck layer. The amount of perceptrons in the bottleneck are minimized and after the training phase the data is encoded in the network. The perceptrons in the bottleneck layer can represent features or the near minimal amount of parameters necessary to describe the



whole of the dataset it is being trained with. In practice this means that the hidden layers before the bottleneck layer function as an encoding scheme and the hidden layers after the bottleneck as a decoding scheme. Auto-encoding is done using auto-association in the training phase. This means that the input data for the neural network is the same as the target or output data. An interesting advantage of this type of neural network is that it cannot only be used for encoding, but also that the parameters of the bottleneck layer can be tweaked to synthesize information from the dataset and also information that was not in the dataset but could have been there [9].

Fig 1: An example of an auto-encoder as used in this project.

Choices about the architecture of the network are made based on current literature and experimentation. In principle each perceptron is fully connected to all perceptrons in the previous layer. The number of hidden layers and their size or proportions were taken from the research done by Hinton where it has determined by trial and error that the amount of hidden layers was set to five and the amount of nodes per layer are half the size of the previous layer in the encoding part and mirrored in the decoding part of the auto-encoder [10, fig 1]. The amount of bottleneck nodes that are needed for proper resynthesis are the subject of research in this project.

The training method used to train the auto-encoder is conjugate gradient backpropagation. Conjugate gradient is a local algorithm for an objective function whose gradient can be computed, belonging for that reason to the class of first order methods. According to its behavior, it can be described as a deterministic method. The conjugate gradient method can be regarded as being somewhat intermediate between the method of gradient descent and Newton's method. It is motivated by the desire to accelerate the typically slow convergence associated with gradient descent while avoiding the information requirements associated with the evaluation, storage, and inversion of the Hessian matrix as required by the Newton's method. In the conjugate gradient algorithm search is performed along conjugate directions, which produces generally faster convergence than steepest descent directions [14]. Both for auto-encoders and other supervised learning architectures the conjugate gradient method has been shown to outperform other standard training algorithms [15]. The mean squared is used as the error function, the advantage of this function over other error standards its value does not grow with the size of the input-target data set. This advantage is key to a successful training with conjugate gradient backpropagation and therefore it is used in the experiments described below.

The training set used is extracted from the TIMIT database. This database includes precisely phonetically annotated speech sounds including speech from a wide variety of accents. Voice sounds of both males and females are included. This project uses this database to automatically extract all sound data available from a selected phoneme. With this data a dataset is generated that in turn is used to train the auto-encoder. To avoid the usage of a hard to interpret digital sound signal we use the discrete Fourier transform to transform the sound bytes (phonemes extracted from the TIMIT database) to the frequency domain. To reduce the amount of data the imaginary part of the Fourier transform was removed and reset to zero for resynthesis. This has two main advantages. One is that

without the imaginary data the timbre quality remains similar. The second advantage is that the amount of data needed to train the auto-encoder is halved, increasing the chance of a correct convergence. For practical reasons a half complex real Fourier transform was used.

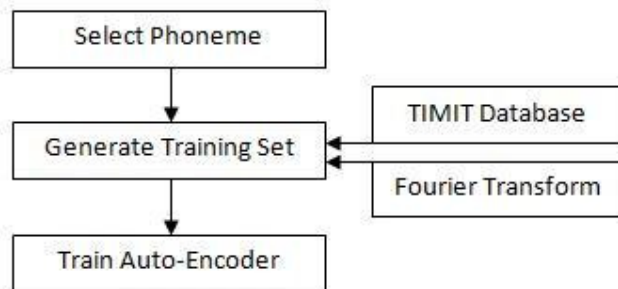


Fig 2: Generation scheme for the auto-encoder.

After the auto-encoder is trained with the spectral phoneme data it can be used to construct a voice synthesis application. In practice this means that the input layer and the hidden layers that are used for encoding are removed from the neural network and a new neural network is constructed with the bottleneck layer as the input layer, the decoding layer and the output layer. Inserting values in the newly constructed input layer generates new sound spectra encoded in the neural network. On the resulting output the inverse Fourier transform is performed, a new sound is generated (Fig. 3).

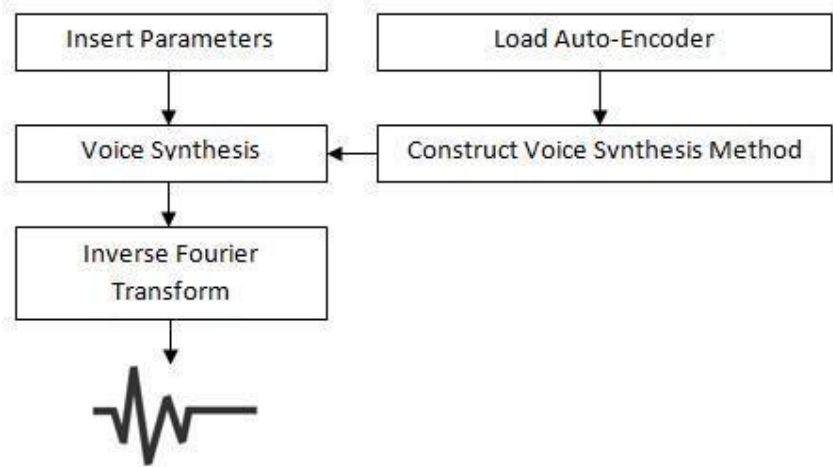


Fig 3: Voice Synthesis Scheme

Experimental Results

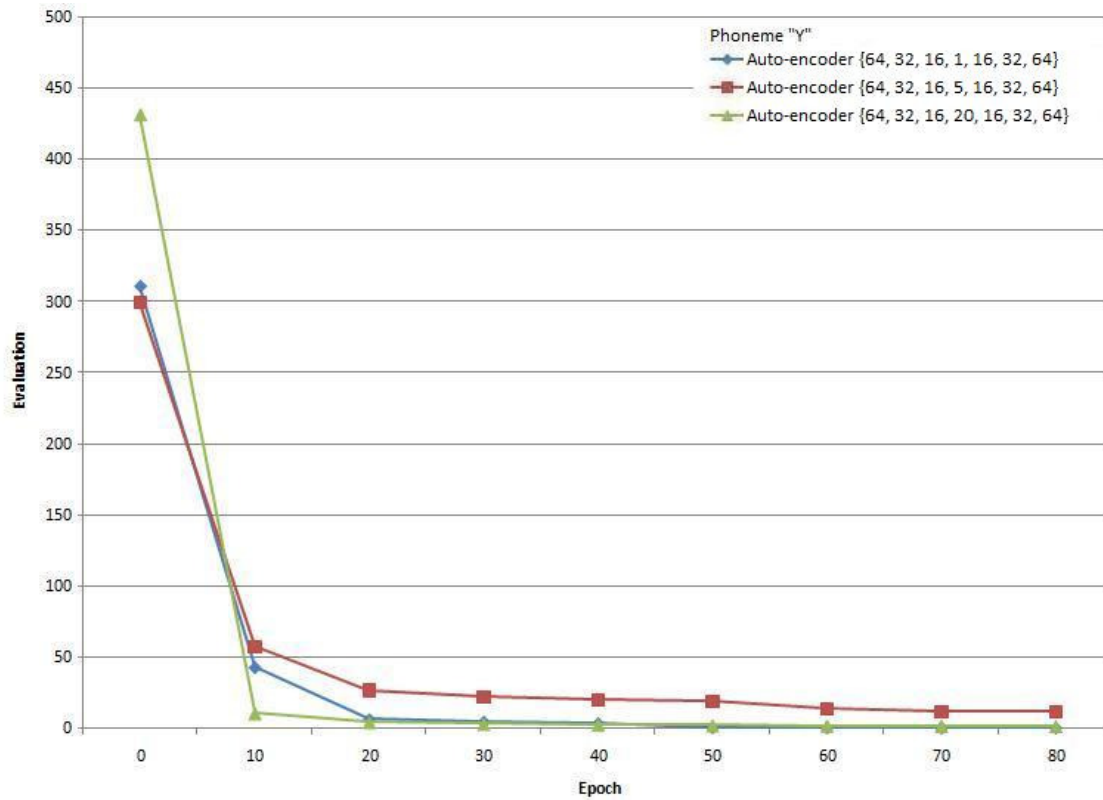


Fig 4: Evaluation of the convergence of the auto-encode.

In figure 4 you can see the convergence of the auto encoder over a training session of 80 epochs. The dataset consists of 2371 sound bites of the phoneme "y", the fft size is 128 resulting in a network architecture of {64, 32, 16, n, 16, 32, 64}. Nothing definitive can be said about the convergence of the auto-encoder during training. With a bottleneck size of one the network converged very close to the target maximum error. However, with a

bottleneck size of 5 the convergence went very stuck and it got stuck in a local optimum. With a bottleneck size of 20 the results were again better, but not as good as with a bottleneck size of 1. Other runs give similar results in the sense that independent of the size of the bottleneck layer the network sometimes converges to the target error but mostly not.

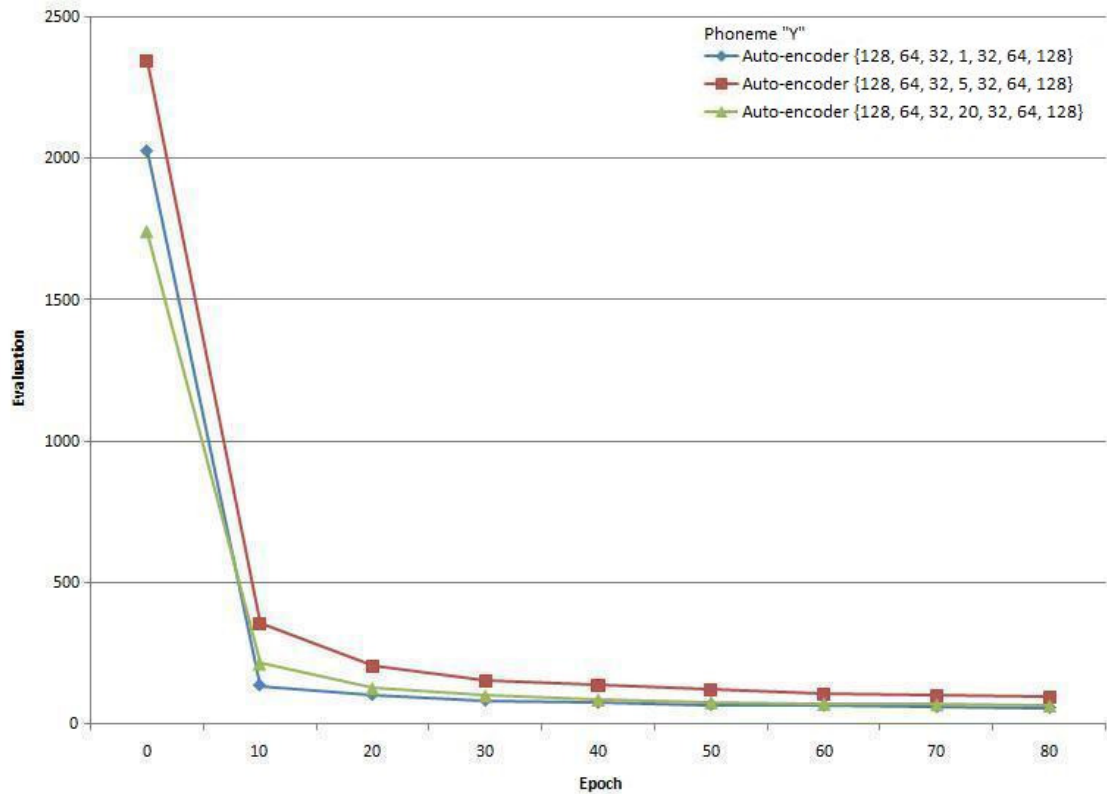


Fig 5: Evaluation of the convergence of the auto-encoder.

From figure 4 and 5 it is obvious that when the size of the neural network is larger due to a larger fft size, only increased to 256, it is harder to convergence towards the goal evaluation. The mean of the dataset is trained due to that the network gets stuck in some local optimum. This is in relation to the literature where it is stated that when the size of a neural network increases and even more the amount of hidden layers increase the network has trouble to converge, and often tends to converge to the average of the dataset. This is due to the fact that once the errors get backpropagated to the first few layers, they are minuscule and therefore quite ineffective. Resulting in slow training and poor solutions [16]. Another fundamental problem arises with this fact. And this is that, related to Heisenberg's uncertainty principle, it is not possible determine a detailed spectrum at a precise location and vice versa. Zero padding can do this, but this does not solve any problems in our case, also due to resynthesis. One would like to have a small spectrum that can regenerate a larger sound. Due to the these issues we chose to limit the resolution for the auto-encoder to stay trainable. What is unknown at this point is whether there is enough information available when the resolution of the Fourier transform is decreased.

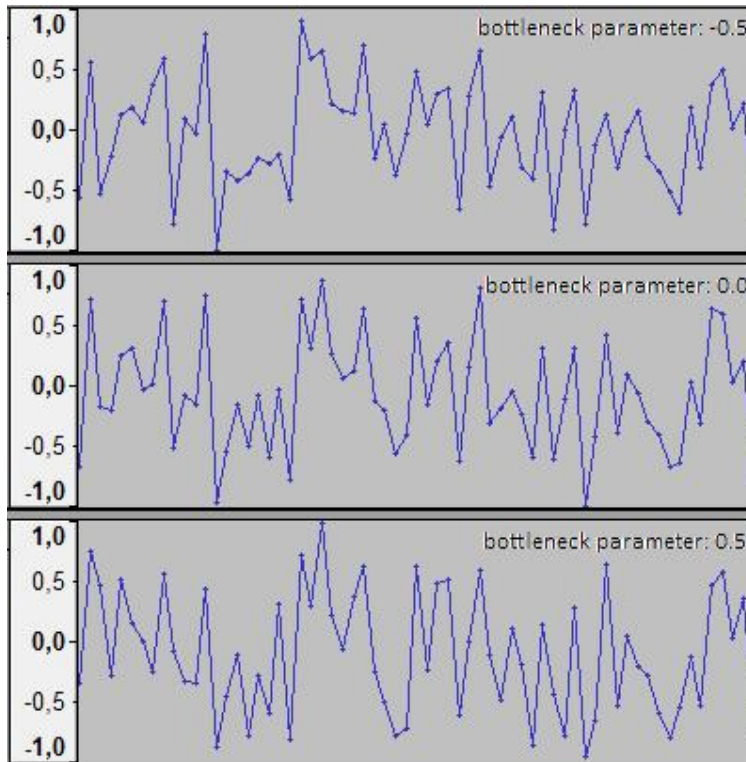


Fig 6: Synthesized sounds with the bottleneck parameters -0.5, 0.0, 0.5.

Figure 6 shows some sounds that were synthesized with auto-encoder {64, 32, 16, 1, 16, 32, 64} from figure 4. You can see that certain patterns are pseudo stable when synthesized and other patterns shift more when changing the parameter of the input.

Conclusion & Future Research

The results of our research indicate that it is possible to auto-encode spectral qualities of voice sounds and more specifically phonemes. This method does however only work sporadically and only on sound bites that are too small to be of any practical use as this point. Often the auto-encoder converges to the average of the dataset rendering it useless for voice synthesis. From the experiments we were not able to conclude anything about the right amount of bottleneck nodes that can be used for synthesis. Also a fundamental problem of using the sound spectrum arose in the development of this technique concerning the problem of network size and the resolution of the sound spectrum, this remains an open problem. Although the first small and modest success is here, clearly more research is needed to make this technique a promising one.

Recent work on image encoding and image synthesis indicates that a pre training method with restricted Boltzmann machines yields significantly better results [10, 11]. Also Hebbian learning and competitive methods can be an interesting option to explore. These should be highly considered as a next step in the continuation of our research.

References

- [1] Pfitzinger, H.R. Unsupervised Speech Morphing between Utterances of any Speakers. In Proc. of the 10th Australian Int. Conf. on Speech Science and Technology (SST 2004), pp. 545-550. Sydney. Dec., 2004.
- [2] Masanobu Abe „Speech morphing by gradually changing spectrum parameter and fundamental frequency“
- [3] T.Ezzat, E.Meyers, J.Glass, and T.Poggio. Morphing Spectral Envelopes using Audio Flow. In Proc. ICASSP, Lisbon, Portugal, September, 2005.
- [4] Yizhar Lavner and Gidon Porat “VoiceMorphing using 3DWaveform Interpolation Surfaces and Lossless Tube Area Functions”
- [5] Peter Birkholz and Dietmar Jackel (2005) “A three-dimensional model of the vocal tract for speech Synthesis” 15th ICPhS Barcelona
- [6] Xiao B Lu, Peter J Bier and C William Thorpe (2006) “A time-varying three-dimensional model of the vocal tract” Proceedings of the 11th Australian International Conference on Speech Science & Technology,
- [7] Pierre Baldi and Kurt Hornik (1988) “Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima”, *Neural Networks* 2, 53–58.
- [8] G W Cottrell and P Munro (1988) “Principal components analysis of images via back propagation” *In Proceedings of the Society of Photo-Optical Instrumentation Engineers*
- [9] David DeMers and Garrison Cottrell (1993) “Non-Linear Dimensionality Reduction”, *Advances in Neural Information Processing Systems* 5
- [10] Hinton, G. E. and Salakhutdinov, R. R. (2006) Reducing the dimensionality of data with neural networks. *Science*, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.
- [11] Hinton, G. E. (2007) To recognize shapes, first learn to generate images In P. Cisek, T. Drew and J. Kalaska (Eds.) *Computational Neuroscience: Theoretical Insights into Brain Function*. Elsevier.
- [12] <http://tcts.fpms.ac.be/synthesis/mbrola.html>, 05-05-2009
- [13] Kurt Hornik: Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, vol. 4, 1991.
- [14] Roberto Lopez Flood 2 2008 An Open Source Neural Networks C++ Library. www.cimne.com/flood
- [15] Martin F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.
- [16] <http://en.wikipedia.org/wiki/Autoencoder>, 05-05-2009