A Survey on Voice Conversion using Deep Learning

Benjamin Meier
Report for module VT2, Master of Science in Engineering
Zurich University of Applied Sciences, Winterthur, Switzerland
meierbe8@students.zhaw.ch

***Abstract*—Voice conversion covers several topics and there are many approaches available to partially solve the problem. This survey explains the required fundamentals, presents different methods to solve this task and contains more material about similar problems. It focuses on end-to-end deep learning methods.**

The objective of this work is to give an overview over the state-of-the-art of voice conversion methods that are based on deep learning.

***Keywords*-speech synthesis, voice, conversion, audio classification, style transfer, audio style transfer**

I. Introduction

Voice conversion describes the problem of changing a speaker's voice in a given audio sample to another voice, leaving the content unchanged. A perfect system would be able to take an audio input and a target voice input and do the voice conversion in real-time. The generated audio should sound natural and the content should not be changed. This survey explains different models and methods that can help solve this problem. It focuses on methods that are based on neural networks.

Neural networks already proved their potential for computer vision [1] [2] [3], but already many audio tasks also exist for which neural networks are more effective than any other previously used method [4]. Especially, they work great for speech recognition tasks [5]. The created models are often much simpler than previous approaches and contain far fewer handcrafted features.

In many tasks, neural networks have the advantage that much less pre-processing is required and therefore the models are often more general. On the other side, the training requires much labelled data. Sometimes it is very expensive to get these labelled records. One approach to face this problem is to use active learning [6]: The system gives feedback about what type of labelled record is required for an optimal training. This idea works sometimes very well (e.g. for captcha images [7]), but this may depend on the specific training task.

A task with just a few records is, e.g., a speech recognition system for rare languages. It is hard to get new data records for these languages, because often there are not many speakers available who can create huge amounts of new records in the required quality. Fortunately, it is often a much simpler problem to create a speech synthesis that produces well-understandable audio for a given text in a given language. The audio then does not sound extremely natural, but at least it is well understandable. An idea is now to use this simple speech synthesis system to produce endless amounts of training data. A drawback is that the speech recognition system then just learns to recognize the synthesized speech of the second system. Because of this, one could train a neural network that performs a voice conversion: The used voice of an audio stream is changed to another given voice. This neural network could then be used as pre-processing for the speech recognition system for rare languages. It standardizes the input voice and therefore the speech recognition system just has to learn one single voice. Therefore, the generated speech and the real dataset may be both used as training data, because the input for the speech recognition system is always converted to the same voice.

A well-working voice conversion system also has other use cases like voice restauration, speech-to-speech translation and security-related applications [8]. It is also assumed that not only the resulting system is helpful, but also the developed technologies and models could be used for similar problems.

The idea of voice conversion can also be generalized to the idea of audio style transfer. This is an adaption of image style transfer [9]. The idea behind it is to take the style of some audio and transfer it to another piece of audio. An example would be to convert the voice of a speaker to another speaker's voice or to change the used instrument in a piece of audio. The idea behind (audio) style transfer is very general.

If a voice conversion system is given that produces good and natural sounding outputs, it should also be possible to use the synthesizer for a text-to-speech (TTS) model. There are already many use cases for this subsystem, e.g. using TTS for devices of blind people or to generate spoken text in computer games in real-time.

In this paper, we give an overview over the state-of-the-art of voice conversion with deep learning. We focus on the general methods and on methods based on audio style transfer. This paper contains surveys for all required main topics. Speech synthesis is an essential part of voice conversion; therefore, some papers and methods are introduced in Section III. Section IV covers several methods to do the voice conversion itself. Section V and Section VI focus on the methods based on audio style transfer that may be used for the voice conversion task. Finally Section VIII presents the conclusions.

II. Fundamentals

Many of the explained approaches are based on convolutional neural networks (CNNs) [10], because audio data fits this type of network quite well [11]. This section explains the

basic data types and formats that are required for audio data processing. Unlike in computer vision, it is not always that simple to create good visualizations of audio data (e.g. it is very hard to "read" the spoken audio content or to identify the used voice). To make this section simple, it is assumed that the audio data has only one audio stream.

A single audio stream is just a one-dimensional list of floating point numbers where the time delta between all numbers is fixed/known. The x axis usually describes the time and the y axis the sound pressure in the SI unit "pascal". This form of the audio data is raw data. The only parameter of the raw data is the sample rate. This rate defines the samples/values per second. Often, this data is processed to get another representation of the audio data that is easier to use for further processing. The result is at most times some kind of spectrogram. This spectrogram can be created with a Fourier transform, a constant-Q transform or any other suitable transform. The idea behind these transforms is very similar, therefore, and because most people use the Fourier transform, only the Fourier transform spectrogram is explained in more detail.

The Fourier transform decomposes a signal over the time into the frequencies that make it up. Because of the form of our data, we use the discrete-time Fourier transform (DTFT).

The DTFT expects a periodic signal, but audio signals are usually not periodic; therefore, a small window is used to calculate the DTFT. To calculate the DTFT we assume the wave in this window is always repeated. Then this window is moved by a given step size, which is usually smaller than the window size, and the DTFT is calculated again. The result is a list of frequency intensities for each window. This so-called spectrogram has about $T/\text{size}_{\text{step}}$ entries and each entry has an intensity value (amplitude) for each frequency. The phase is no longer used, because for many tasks it is not that important.

The frequency axis (used unit: "hertz") of the spectrogram is often converted to the mel scale. The mel scale has the advantage of an equal distance for pitches. The scale is based on data that was produced by listeners who had to judge the distance from one pitch to another [12]. A popular formula to convert f hertz into m mels is given by the following expression [13]:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

It is important to see that this formula cannot be inverted without additional errors if discrete values are used. In general, it is doable, but the quality decreases.

To reconstruct audio from a spectrogram, the phase has to be recovered [14]. The spectrogram contains implicitly some phase information if the windows used for the Fourier transform overlap. This information allows one to recover the phase to some degree. Some neural networks that produce audio prefer spectrogram outputs and therefore it is important to convert a spectrogram back to audio.

III. Speech Synthesis

Speech synthesis describes the problem of generating spoken audio with computers. The source could be a text, but also some other data. The main objective of speech synthesis is to generate a natural-sounding voice. There are different approaches to solve this problem, but nowadays mainly neural networks are used for this difficult task. The problem can be solved for one specific synthetic target voice or also more dynamically, so that it may produce audio for any given voice.

On the technical side, there are different approaches to solve the problem. Traditionally, hidden Markov models (HMM) were used for speech synthesis tasks, but nowadays mostly neural networks are used because of their flexibility. This survey does not focus on HMM-based models, but for the interested reader J. Yamagishi [15] gives a great introduction.

Even if the base technique is chosen, more details are needed about the used data. The input data may be task-dependent (e.g. text), but the output should be audio. Often a mel-spectrogram or raw waveform [4] [16] is used for the output. Some systems may use other data structures, but generally, data structures that do not require much post-processing are preferred. The neural network then must choose an own encoding and abstraction of the data. A reason for this so-called "feature learning" [17] is that handcrafted features often require expert knowledge and do not generalize well. Neural networks may learn better features that generalize better and do not need any expert knowledge.

Different approaches based on neural networks may be trained end-to-end [4] [16] or the different networks parts are trained independently [18] and are fitted together at the end. Some approaches [19] combine these two ideas: First the different network parts are trained independently and then the full network is trained end-to-end. In general, end-to-end training is preferred, not only because it is often simpler, but also because then the neural network, having learned the full task, has the chance to see the big picture and use some non-obvious improvements.

The evaluation of a speech synthesis system is quite hard, because currently no known measure is able to replace the human ear. Especially, it is hard to measure the naturalness of generated speech with an algorithm. For this reason, a mean opinion score (MOS) is often used as a final measurement to decide how good the quality of the generated voice is. Unfortunately, such an MOS is expensive: Many people are required to get a statistically useful value. It is also very costly to have the people hear all samples and decide how good they are.

Voice conversion always contains a synthesizer. Therefore, it is helpful to understand different concepts of speech synthesizers and how they work. Speech synthesis also contains the part of including a given target voice, which is also important for the voice conversion task. This voice may be fixed or can be dynamically chosen, depending on the used speech synthesis system.

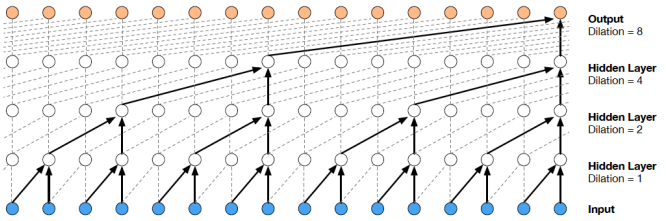

Fig. 1: Stacked dilated causal convolutions.
Source: [4]

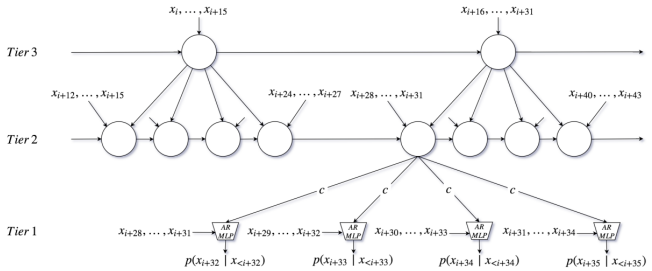

Fig. 2: A SampleRNN module and its receptive field.
Source: [16]

A. WaveNet: A Generative Model for Raw Audio

WaveNet [4] is a deep neural network architecture proposed by van den Oord et al. that generates raw audio output. The authors show that, as per ratings by humans concerning naturalness, the text-to-speech tasks outperform currently existing parametric solutions. WaveNet may also be used to generate music. The generated fragments are often realistic and novel. The used architecture directly produces audio waveform and therefore does not require any post-processing.

WaveNet is based on the idea of PixelCNN [20] and is adapted to audio data. The network uses dilated causal convolutions to deal with temporal dependencies. These convolutions are preferred over recurrent neural networks (RNNs) [21] because they are faster and easier to train. Stacked dilated causal convolutions allow the network to have a very large receptive field, which is highly required for audio data. Figure (1) visualizes a stacked dilated causal convolution.

The network generates in every timestamp the next value for the raw output. This value then may be used as the next input for the network. The output is trained with a sample rate of 16 kHz and 256 different possible discrete values for each sample. The discretization allows it to use "softmax" and solve this problem as a classification problem.

Dilated causal convolutions have a dilation factor that defines which elements should be processed from the previous layer. If the factor is 2, then every second element is processed, if it is 8, every 8th element is processed. A dilated causal convolution also has a window length, like the classical convolutions. This type of convolution allows an exponential receptive field width according to the count of the dilated convolution layers.

Additional inputs may be used to produce audio with specific characteristics, e.g., to generate audio with a specific voice, an input with the speaker identity may be added.

WaveNet allows the generation of raw audio that sounds very natural. The flexibility of the network allows using it for several audio-generating tasks, not only for speech generation. A drawback of the network is that it requires much time to generate raw audio (about 90 min for 1 s of audio). This makes it currently not useable for real-time tasks like real-time voice conversion. Even for many offline calculation use cases, it may be not fast (and energy efficient) enough. Nevertheless, WaveNet highly improved the state-of-the-art of generated audio.

B. SampleRNN: An Unconditional End-to-End Neural Audio Generation Model

SampleRNN [16] is a network architecture for raw audio generation based on RNNs. The hierarchical and stateful architecture allows handling long-time dependencies. The naturalness of the network is very good according to humans and it is also very fast (especially compared to WaveNet [4]).

RNNs are used in a hierarchical architecture. The reason for this is that audio correlates in different time scales, not only to the neighboring samples, but also to samples that are a few thousand steps away. Every such time scale is handled by a so-called module. The lowest module directly works with samples and higher modules with more abstract versions of the audio data. Such a module hierarchy is shown in Figure (2). The complete hierarchy may be trained end-to-end with backpropagation.

To implement the idea, several tricks are required, e.g., to discretize the network output. Only 256 discrete output values are used. This allows the use of "softmax" to view the task as a classification problem. For tasks that have longer time dependencies, it is sometimes very hard to train RNNs. For audio data, this is also the case and to avoid this problem, truncated backpropagation through time [22] is used. Only short snippets of the original sequence are fed into the network and used for the training.

The audio quality is not as high as the WaveNet output, but SampleRNN requires much less time for the generation of the audio. Another advantage compared to WaveNet is that the researchers published their code. This makes it much easier to reimplement the architecture for third persons.

This network is a good candidate for audio generation. The quality is quite good and it does not require very much processing time. For example, Char2Wav [19] uses this raw audio generator network to produce audio from text.

C. Deep Voice: Real-Time Neural Text-to-Speech

Deep Voice [18] is a real-time text-to-speech synthesizer model that is completely based on neural networks. The synthesizer is based on a smaller version of the WaveNet [4] model. Deep Voice is inspired by traditional text-to-speech architectures and adopts the different steps. Each of these steps is replaced by a neural network.

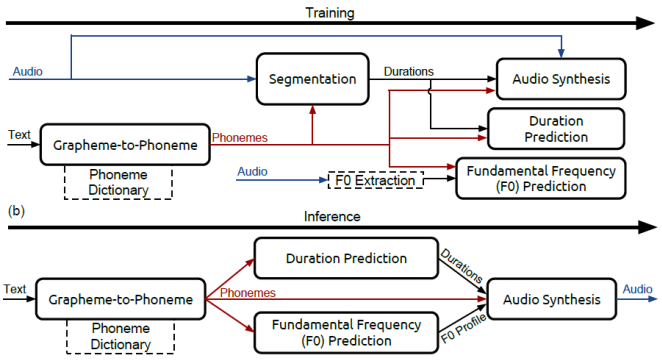

Fig. 3: The Deep Voice model.
Source: [18]

Traditional text-to-speech systems contain five components (grapheme-to-phoneme, segmentation, phoneme duration, fundamental frequency, audio synthesis). The main idea is to use a neural network for each of these components.

Every single component is trained and the final combined network may be used for the text-to-speech synthesis. All these components are shown in Figure (3).

The grapheme-to-phoneme network is based on a bidirectional RNN that uses GRU [23] cells and builds an encoder-decoder architecture. The segmentation model uses an RNN to detect the alignment between utterances. Also, the fundamental frequency network uses RNNs in combination with fully connected layers to predict the F_0-frequency. The audio synthesis model is a variant of WaveNet [4] with the difference that the first layers are bidirectional RNN, because according to the authors this performs better.

The different networks are trained independently of one another; therefore, it is not trivial to change the system for an end-to-end training.

The network architecture is more complex than SampleRNN, but as mentioned by the authors, it requires 4-5 times less computational resources.

The described architecture and the training are not very simple, but the network works well. The architecture and the required details are well described in the paper.

D. TTS Synthesis with Bidirectional LSTM based Recurrent Neural Networks

The described text-to-speech model [24] that produces vocoder features is based on a deep recurrent neural network (RNN) with LSTM [25] cells. The model requires preprocessing to do feature extraction and produces vocoder features (compared to raw waveforms [4] [16]).

The idea is to use a bidirectional LSTM network to handle time dependencies. To make the training easier, the data is preprocessed and post-processed. This allows creating a simple network architecture and training the network's output features directly with a given ground truth.

The architecture works well with standard backpropagation through time (BPTT) and does not require additional tricks like truncated backpropagation through time. On the other side, there are handcrafted features in the text analysis, the input feature extraction and in the vocoder. The used neural network has a clean and simple design, because it only uses stacked bidirectional LSTM-layers.

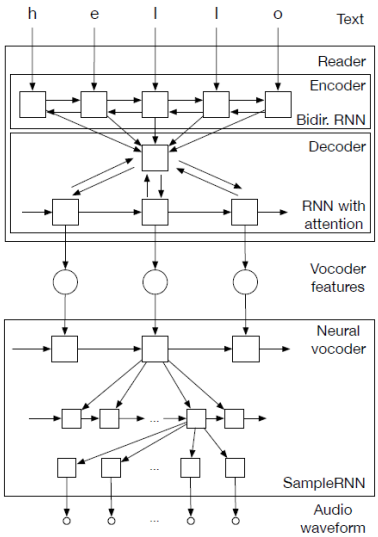

Fig. 4: The Char2Wav encoder/decoder architecture.
Source: [19]

The paper focuses on the transformation of the input features to the features for the vocoder. It is then compared to an HMM-based model. The system works well, but it contains many handcrafted features and has also a not very flexible structure that is given by the pre- and post-processing.

It is probably not the preferred solution for audio generation. Models that use more neural networks and fewer fixed features should be used instead, because they are more flexible for different tasks and therefore can be reused much better.

E. Char2Wav: End-to-End Speech Synthesis

Char2Wav [19] describes a two-stage neural network architecture that has a text input and outputs waveforms. The network is based on an RNN and uses SampleRNN [16]. With traditional speech synthesis systems, expert linguistic knowledge is required to define the linguistic features. This approach does not require them because they are trained by the system.

The two-stage network contains a reader component and a vocoder component. The reader uses an RNN with attention mechanism [27] to produce vocoder features. These features are then processed by the neural vocoder that is based on SampleRNN [16]. The network architecture is visualized in Figure (4).

The reader and the neural vocoder are pre-trained in a first step. WORLD vocoder features [28] are used as target for this step. Finally, the complete network is trained end-to-end. The neural vocoder is mainly based on SampleRNN [16], which is also a non-trivial network.

TABLE I: A comparison of all speech synthesis models.

Paper	Method	Availability (code)	End-to-End	Training Time	Real-Time	Elegant
WaveNet: A Generative Model for Raw Audio [4]	Deep learning with dilated convolutions	yes (third-party)	yes	data dependent/long	no	yes
SampleRNN: An Unconditional End-to-End Neural Audio Generation Model [16]	Hierarchical RNNs	yes	yes	1 day	yes	yes
Deep Voice: Real-Time Neural Text-to-Speech [18]	Uses for each classical TTS component a neural network	no	no	unknown	yes	complex
TTS Synthesis with Bidirectional LSTM based Recurrent Neural Networks [24]	Stacked bidirectional LSTM-cells: Input and output are abstract features	no	yes (given the abstract features)	unknown	yes	no (many handcrafted features)
Char2Wav: End-to-End Speech Synthesis [19]	Train vocoder features; based on SampleRNN	Alpha version	Pre-training, then end-to-end	unknown	yes	yes (except for the pre-training)
Tacotron: Towards End-to-End Speech Synthesis [26]	Encoder-decoder networks that produce a spectrogram	yes (third-party)	yes	multiple days	yes	yes

There is some code available on GitHub, but unfortunately it does not run and it is not documented. As described in the "issues" the used hyperparameters are not yet available. The authors of the paper already created a voice conversion software [29], but the used technology is not published. Probably, it is based on Char2Wav, but this is not known. Therefore, a modification of this network should also work for the voice conversion task.

As the authors have written on GitHub, they have submitted a more detailed paper, but it is not yet published. Probably one should wait for this paper before the system is used/implemented, because currently some relevant details are missing (e.g. hyperparameters).

F. Tacotron: Towards End-to-End Speech Synthesis

Tacotron [26] is a neural network architecture for the text-to-speech task that can be trained end-to-end. Unlike Char2Wav [19] it does not require that the encoder and the decoder are pre-trained before the end-to-end training is done. The network requires only text/audio-pairs for the training.

The input for the network is a character embedding. The encoder network creates a stable sequential representation of the content. This representation is then transformed in the decoder network to an 80-band mel scale spectrogram. This spectrogram is finally converted to a waveform with the Griffin-Lim reconstruction algorithm [30] with 50 iterations.

The complete model, including the Griffin-Lim reconstruction algorithm (it has no trainable weights), is implemented as a single graph. It then can be trained end-to-end. The authors used TensorFlow [31] to implement this model.

As experiments show, the synthesis works quite well. Compared to Char2Wav [19], the training of the Tacotron network is much easier, because it does not use vocoders and can be trained end-to-end in one step.

G. Comparison

All described papers about speech synthesis are compared in Table (I) to give an overview over the described methods.

IV. Voice Conversion

Voice conversion may be seen as an extension of the speech synthesis problem, but it is generally more difficult to solve the voice conversion problem. The objective is to create a function $f_{a\to b}(x_a) = x_b$ that converts an audio sample x_a that is spoken with the voice a to an audio sample x_b that is spoken with the voice b. The content of the audio sample must not change. The easiest version of the problem is if the source voice and target voice are fixed and just one single conversion is possible. The most general version would be if the audio input could be spoken by any voice and it would be possible on-the-fly to define the target voice (e.g. by an audio sample). A big problem for the general case may be that the voice extraction and the training have to be done with only a little bit of data. The problem becomes much easier if the source and target voice are fixed and there are many data records for both voices. In the best case, the data records are evenly paired: For both speakers/voices there are audio samples with the same content and with the specific speaker voice.

In the past years many approaches based on vector quantization (VQ) [32] [33], hidden Markov models (HMMs) [34] [35] and Gaussian mixture models (GMMs) [36] [37] were used for these tasks. Currently, there are more and more neural network based approaches available. Some of them use deep belief networks (DBN) [38] and Boltzmann machines [39]. None of the currently available systems are able to generate extremely natural and correct sounding samples, but the current state-of-the-art is already performing quite well [40]. In this work, we focus on the neural network-based approaches.

For speech synthesis (see Section III), there is the problem that it is hard to automatically evaluate the results. Because voice conversion is generally an even harder task, the same problem presents itself, especially because no good measure exists for the naturalness of speech. The voice conversion challenge 2016 (VCC2016) [40] uses humans for the evaluation. They have to choose between five values: from 1 for completely unnatural to 5 for completely natural. Of course, this measure is not well-defined mathematically and does not

scale, but it is currently the best available measure for this task.

There are still some other measures available that at least give a good hint if the generated output has a good quality. Huang et al. [41] use an automatic evaluation based on perceptual background noise distortion and a speaker similarity score. Sun et al. [42] calculate the Euclidean distance between the mel-cepstral (mel-cepstral Distortion). This measure is used to see how close the converted speech is to the target speech.

Another approach from Chen et al. [43] uses the logspectral distortion (LSD) to measure the quality of the voice conversion; this approach is especially useful to compare the content. Unfortunately, the only measure that works very reliably for this task is the human ear.

Sotelo et al., the authors of [19], implemented a non-publicly available voice conversion algorithm and have an API [29] published. It is not explained how the voice conversion algorithm works, but it may be assumed that it is based on the ideas of Char2Wav [19].

A. Phone-Aware LSTM-RNN for Voice Conversion

Lai et al. [44] describe a voice conversion model based on an LSTM-RNN. They not only use a mel-cepstral as input, but also monophone features that are generated by a speech recognition system.

It is important for the training to have aligned data for the source and target speaker. This is not always given and therefore a dynamic time warping (DTW) algorithm is introduced in the paper. The neural network itself is relatively simple: The inputs are the $\log(F_0)$ frequency, the mel-cepstral and the monophone features. Just a basic LSTM-architecture is used.

To convert the generated output of the network, combined with the $\log(F_0)$, back to audio, the STRAIGHT [45] method is used. In general, there is much pre- and post-processing required.

The network produces good results, as shown in the VCC2016 [40]; so it is a good candidate for an implementation. The network is tested with the VCC2016 data; therefore, all results should be reproducible for anyone. On the other side, the network may not be trained end-to-end, because of the pre- and post-processing. A speech recognition system is also required, which sometimes may be a problematic point.

B. Voice Conversion using Deep Bidirectional Long Short-Term Memory based Recurrent Neural Networks

Speech is highly correlated over time. This requires a model that is able to handle this. Sun et al. [42] use for this reason a deep bidirectional long short-term memory-based recurrent neural network (DBLSTM-RNN). A simple architecture with pre-processed audio features as input is used.

The source audio is pre-processed. Features like the F_0-frequency and the aperiodicity are extracted and not processed by the neural network. The neural network gets only the mel-cepstrum as input. Several bidirectional LSTM layers are used to process this input and to produce a new mel-cepstrum for the output. This output is then combined with the other extracted features of the input to create the final converted speech.

The STRAIGHT [45] method is used to extract parameters like the F_0-frequency. The neural network itself is trained with standard BPTT. The synthesis at the end is also done with the STRAIGHT method. This includes a smoothening of the parameters to create a better output.

The network has a clean and simple structure and is therefore an easy-to-use solution. Unfortunately, much pre-processing and post-processing is required. In general, the architecture of the network is similar to [24]. The network is a good example that it is possible to use the mel-cepstrum to analyse and generate speech.

C. Multi-Output RNN-LSTM for Multiple Speaker Speech Synthesis and Adaptation

Pascual et al. [46] propose a multi-layer neural network that may be used for speech synthesis and for voice conversion. The architecture uses an output branch for each available speaker.

The core network may be trained with some speakers and can then be frozen and reused for other speakers. This highly improves the training time for new speakers.

Standard backpropagation may be used to train the complete model. There are good comparisons available about the model quality, whether it is trained with only one output speaker or multiple output speakers. The authors found out that the quality improves if the network is trained with multiple outputs.

The network input contains a set of phonetic features and the output is also a set of features. The output features include 40 mel-cepstral coefficients, maximum voiced frequency, the $\log(F_0)$ value and a voiced/unvoiced flag. A vocoder [49] is used to convert these features to a waveform.

An adavantage is that the architecture is simple. On the other side, a disadvantage is that the network does not output a waveform but a set of features. There are no detailed benchmarks about the quality available, which makes it hard to get a feeling for the effective network quality.

D. Voice Conversion using Convolutional Neural Networks

Mobine et al. [48] try to solve the voice conversion task by not only transforming the pitch, but also the timbre. A CNN is used for this task. The background is that, e.g., a human can differentiate between a piano and a trumpet that play the same pitch because they have a different timbre.

The main idea is to create a relationship like $f(d) - f(c) \sim f(b) - f(a)$. Given a, b, c it should be possible to predict d. To learn this $f(x)$ operator, a CNN is used. A GAN-like [50] network is used to produce an output that should sound like a real voice.

The input and the output for the network are pre-processed with the constant-Q wavelet transform (CQT). This transform makes it easier to process the data. Unfortunately, the paper does not go into details about the network architecture.

TABLE II: A comparison of all voice conversion models.

Paper	Method	Availability (code)	End-to-End	Input Data Type	Output Data Type	Elegant
Phone-Aware LSTM-RNN for Voice Conversion [44]	A LSTM-RNN is used with different input features (mel-cepstral, $\log(F_0)$, monophones) and produces a mel-cepstral	no	no	mel-cepstral, $\log(F_0)$, monophone features generated by a speech recognition system	mel-cepstral	The neural network is simple, but much pre- and post-processing is required
Voice Conversion using Deep Bidirectional Long Short-Term Memory based Recurrent Neural Networks [42]	Using a BDLSTM that takes a mel-cepstral as input and produces a mel-cepstral; the cepstral is combined with several handcrafted features to produce audio	no	no	mel-cepstral	mel-cepstral	The neural network is quite elegant, but there are many handcrafted features involved.
Multi-Output RNN-LSTM for Multiple Speaker Speech Synthesis and Adaptation [46]	Using a core model that has multiple output branches: one for each speaker	no	no	Several features extracted by [47]	mel-cepstral, $\log(F_0)$, etc.	yes
Voice Conversion Using Convolutional Neural Networks [48]	Using a GAN to produce audio	yes (alpha)	yes	CQT spectrogram	CQT spectrogram	yes

The used dataset is very small and there are no detailed tests available. The authors do not write much about how the network is implemented, but there is some code available. It does not seem that there is any progress. Nevertheless, the idea to use a discriminator network to refine the produced output is good. In general, it could be a good idea to use a GAN-architecture. The method is not described in much detail and therefore it is probably hard to use it for real tasks.

E. A Survey on the Evolution of Various Voice Conversion Techniques

Sathiarekha et al. [51] explain and compare voice conversion models that are based on vector quantization (VQ) [32] [33], HMMs [34], GMMs and neural networks [36] [37]. Only a few approaches of each method are compared, but the paper gives a good overview over the methods and the state-of-the-art. It also explains the history of voice conversion.

Neural networks that are based on deep belief networks (DBN) and Boltzmann machines are presented and briefly explained. A final table shows the advantages and disadvantages of the different voice conversion methods. Not specific models are compared, but classes like HMM-based models and neural network based models. Neural networks have the advantage that they can be used to fully do the learning and synthesis process. They are also able to produce good signals, even in noisy environments. A disadvantage is that the synthesis accuracy is only high on deeper architectures and therefore more complex architectures are required.

The paper gives some information about the different ways to implement voice conversion (GMMs, HMMs, etc.). None of these ways is based on audio style transfer, but on more specific models. Sathiarekha et al. [51] write that it is a disadvantage that solutions based on neural networks require complex architectures. We think there is still the advantage that often almost no handcrafted features are required and this is a good trade-off, because the neural network itself is often general and may be used for similar tasks. This is often not true for other models (HMMs, GMMs, etc.).

F. Comparison

All described papers about voice conversion are compared in Table (II) to give an overview over the different methods. These methods do not use audio style transfer, but solve the more specific problem of voice conversion directly. These algorithms do solve the complete problem, but many of them use handcrafted features and the results still have potential for improvements. Still, new ideas are required to solve this problem with neural networks.

V. Audio Classification for Voice Conversion using Audio Style Transfer

Audio classification describes the task where a given audio sample has to be classified. Classes may be, e.g., different audio genres [52] or speakers [11]. In general, the audio input size is fixed, but it would also be possible to create a system with dynamic input sizes.

There exist approaches based on HMMs [53] [54] [55] and neural networks [52] [11]. As shown, the solutions based on neural networks use simple models and are already very accurate. In this work, we focus on the neural network-based solutions.

There are several possible input data types for classification networks: Raw audio, a spectrogram as 1D image, where each frequency is an input feature, or a spectrogram as 2D image (frequencies $\times$ time). Raw data has the advantage that no pre-processing is required, on the other side, spectrograms also produce very accurate results. All these data types may be used for CNNs, which are powerful for hierarchical feature

TABLE III: A comparison of all audio classification models.

Paper	Method	Availability (code)	End-to-End	Input Data Format	Training Time	Elegant
End-to-End Learning for Music Audio [52]	Using a CNN to classify audio	no	yes	Raw data or spectrogram (1D: F features, or 2D: $F \times T$)	unknown	yes
Convolutional Neural Network-based continuous Speech Recognition using Raw Speech Signal [5]	Using a CNN to classify audio	no	yes	Raw data	unknown	yes
Speaker Identification and Clustering using Convolutional Neural Networks [11]	Using a CNN to train a classification process and then use an embedding for clustering	no	yes	Spectrogram (2D, $F \times T$)	9 h	yes

extraction. As already shown by Krizhevsky et al. [1], CNNs work extremely well on image data, even for complex object datasets like [56]. Several approaches like [11] show that CNNs also work great if a spectrogram is interpreted as a 2D image.

The evaluation of this task is quite easy. It is possible to calculate the accuracy and therefore it is known how well an audio classifier works.

Classification networks are an important part of some style transfer models (e.g. [9]) and therefore for audio style classification-based voice conversion. These networks also have to handle audio input data, which is also an important point for any voice conversion model. Because of these reasons, some audio classification networks are explained in this section.

A. End-to-End Learning for Music Audio

Dieleman et al. [52] use a CNN to classify music. They also test if raw input works as well as a spectrogram input.

CNNs are used for image classification tasks and work well there. For this reason, the authors use also a CNN for the audio classification task. The used CNN is only 1D (compared to images where usually 2D CNNs are used). The authors try to avoid handcrafted features and compare a CNN with raw input to a CNN with a spectrogram as input. A third test is done with an extended network with raw input. The authors are able to show that raw input works as well as a spectrogram input.

The used CNN has a simple and standard architecture. The strided convolution for the raw input uses parameters that are similar to a Fourier transform (window length=200, stride=200).

The paper shows that CNNs are able to perform audio classification. It is also impressive that also raw audio input works as well as a spectrogram input. This means the neural network is able to train the otherwise required pre-processing.

For the style transfer as described in [9] it is important to have a CNN that is able to extract good features. In general, it is also preferred if the data has not to be pre-processed and the network is able to learn this step.

B. Convolutional Neural Networks-based continuous Speech Recognition using Raw Speech Signal

Palaz et al. [5] present a neural network that uses raw input and does classification. They also show that the learned filters in the initial convolutional layers may be reused for other audio classification networks.

A small CNN is used to do the audio classification. The initial convolution is used to extract frequency features. These features are then processed in two dense layers. It can be shown that the first convolution learns filters for specific frequencies. This is similar to a Fourier transform.

This paper shows that the filters on the raw input learn something like a Fourier transform and that the filters of this layer are very general and may be reused. It is important to see that a neural network is able to learn how to deal with raw audio input.

Especially the information that the first layer learns something general may be used for transfer learning and therefore provide shorter learning times for new (audio) networks. The classification with the proposed architecture works very well.

C. Speaker Identification and Clustering using Convolutional Neural Networks

Lukic et al. [11] describe a network architecture that may be used for audio/speaker classification. They also use a network embedding to do clustering. The network uses a spectrogram as input and outputs a distribution for the classification.

The described architecture is simple and shows that a 2D CNN is able to learn useful features to identify speakers (or more specific: to do audio classification) and to do speaker clustering. The idea to interpret and process audio as a 2D-image may be helpful, especially for audio style transfer tasks.

D. Comparison

The described papers about audio classification are compared in Table (III) to give an overview of the different methods. All three described architectures are simple and work well. For the audio style transfer only CNN based solutions are required, if any classification network must be used to solve the style transfer task. Such pre-trained networks are only required for some style transfer approaches, but not for all (see Section VI).

VI. Style Transfer methods to do Voice Conversion using Audio Style Transfer

Style transfer describes the idea of taking the style of one input and transferring it to another input to create a new output. This may be very general and is not limited to one specific data type (e.g. images [9] [57], text [58], ...). Style transfer could be done with many models, but we focus on neural networks, because they currently offer the most powerful and general models to do style transfer.

Style transfer is still a general expression. There are still different types of styles transfers for a single data type, e.g. images: Some models transfer the drawing style [9] and some can also transfer the style of a product on an image to another product type on an image [57] (e.g. create an image of a shoe that has the same style as a bag on a given input image). It may be task-dependent what exactly the style transfer is.

The required training data may be very different for some approaches. Some just require a pre-trained good CNN-based feature extraction network [9], others two unpaired sets of images for a source domain and the target domain [59] [57] and others require two sets of paired images for the given domains [60]. Sometimes for applications it may be hard to get a large dataset of paired data records, so the required training data may be critical for custom tasks.

All described style transfer networks are trained on images. In general, it is possible to use the architecture for any data type, but of course it may be more difficult to train the style transfer for other data types.

Style transfer could be used for spoken audio where the style should be the used voice. As described in Section II there are different audio encodings available. For example, raw data may be used as input. For the CNN-based approach, a 1D-CNN may be used. Another approach is to use a spectrogram as input and output. It is possible to do 1D-convolutions (over the time) or also to interpret the spectrogram as a 2D image and use this as data type, then 2D-convolutions are used. Especially the image-based approaches may be preferred for the given network architecture, because they already work well for images. One difference may be that the style is encoded in another way. It is hoped that the end-to-end trained neural networks are capable of generalizing for this task.

A voice conversion model could be created with a style transfer like the one described by Gatys et al. [9] combined with the classification network of Lukic et al. [11]. Some tests we have done show that this combination does not produce good results. It seems that audio classification networks do not use the same features to detect a voice/speaker as the human ear does. This explains why it did not work. Other possibilities would be to use another style transfer architecture (like [57] or [60]) or another classification or feature extraction network.

A. A Neural Algorithm of Artistic Style

The proposed neural style transfer by Gatys et al. [9] allows copying the style of an image and reusing it on another image. For example, the style could be extracted from an art painting and can be adapted to another picture. This picture will then look like if it was drawn by the artist of the original painting.

CNNs and neural networks generally allow reconstructing the input from a given representation at some layer. Deeper layers normally contain less information about specific inputs/pixels, which is the reason why lower layers produce finer reconstructions; this is especially true for CNNs. This information is all about the content. For the style transfer itself, a style loss is required. Style information can be extracted from every convolutional layer with the help of the Gram matrix [61].

To do a style transfer, the style features of the original painting/image are extracted at several convolutional layers and the content features of the new image are extracted at one fixed convolutional layer. A new image that only contains white noise is used as network input and the sum of the differences between the new content features and the original content features and the new style features and the original style features is minimized. To minimize this sum, the input is changed. This is done by an iterative algorithm.

The main novelty is in the use of the Gram matrix. For a convolutional layer, a matrix M can be created, where each column is a feature of a specific convolutional layer and each row a position of such a feature (therefore, the 2D map of each feature is unrolled to a long vector). The Gram matrix then is $M^T M$. It contains the correlations between different filters/features and, in the diagonal values, the absolute occurrences of a specific feature. An important note is that the style loss is taken as a sum over several layers and not only one layer (unlike the content loss).

To use this algorithm, a trained CNN must be given. For image data VGG-16 [2] works fine. As shown, simple architectures like VGG-16 work much better than complex architectures like GoogLeNet [62].

This idea of a style transfer could be used for audio. The input could be an image (e.g. a spectrogram) or also raw data. In both cases, a CNN has to be used. It could be trained for classification (like VGG-16 [2]) and be used for the style transfer. A big advantage of the proposed architecture is that since it does not require any training for the style transfer, only a classification network has to be trained (this is usually much easier). Unfortunately, currently no big pre-trained audio classification network (like VGG-16 for images) is available. Another disadvantage is that the iterative algorithm is slow, even on high-end GPUs.

B. Perceptual Losses for Real-Time Style Transfer and Super-Resolution

Neural style transfer [9] is slow. This is because the proposed algorithm is iterative. The proposed technique by Johnson et al. [63] uses an architecture that is based on a single feed-forward network and therefore works much faster. The proposed architecture may also be used for the image super-resolution task.

An image transformation network is trained to do the style transfer. There are several possible architectures for this

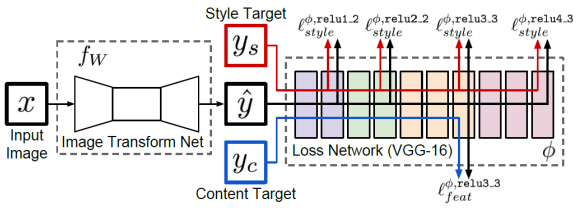

Fig. 5: The loss function for the fast style transfer.
Source: [63]

transformation network. The style loss in combination with the content loss, as described in [9], is used as loss function for this image transformation network. The loss function is therefore another neural network that is fixed during the training. The training is then done for one specific style. In Figure (5), the coarse architecture and especially the loss function are visualized.

The image transformation network is a residual network, but other architectures are possible. Convolutions and transposed convolutions are used to downsample and upsample the image representation. As described in [9], the loss network already has to be available and it must be pre-trained. A good choice for images is the VGG-16 [2] network.

This architecture is important and useful if the style transfer of Gatys et al. [9] is implemented for real-time applications, because otherwise the style transfer is slow. Like [9], the architecture could also be used for audio data.

C. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks

The proposed network architecture by Kim et al. [57] is designed to identify relations between different domains, e.g. handbags and shoes. It is the able to transform a shoe into a related handbag. The architecture does not even require labels for the training of the relations, but only image datasets of different domains. The architecture is based on generative adversarial networks (GANs) [50].

A function/generator G_{AB} exists to map an object from the domain A to the domain B. There is also a function/generator G_{BA} that maps an object from the domain B to the domain A. There are two discriminator networks: D_A for the domain A and D_B for the domain B.

A GAN [50] architecture is used to build the network. A difference is that the generators (G_{AB} and G_{BA}) do not use noise as input, but an image/object from another domain. These generators are therefore some kind of transformation networks.

Given the mapping functions, the objectives of the network are:

- $G_{BA}(G_{AB}(x_a)) = x_a, x_a \in A$
- $G_{AB}(G_{BA}(x_b)) = x_b, x_b \in B$
- Train the discriminator D_A
- Train the discriminator D_B

The objectives are visualized in Figure (6).

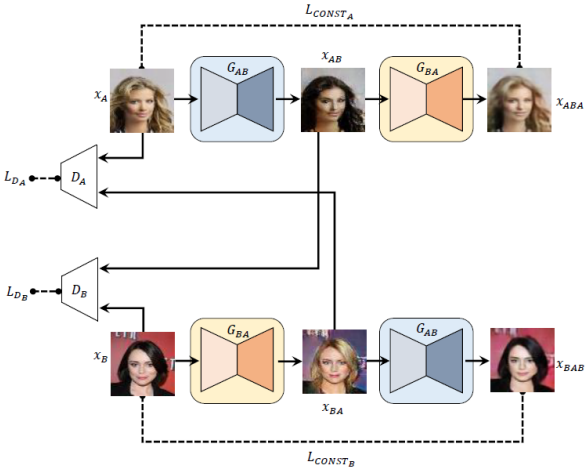

Fig. 6: The GAN-based architecture proposed by Kim et al. [57] to do domain transfer.
Source: [57]

This architecture could be used for audio data. Examples show that the architecture is powerful and able to detect complex relations between domains. If audio is used, the input could be raw data or also a spectrogram. Different domains then relate to different speakers/voices.

D. Image-to-Image Translation with Conditional Adversarial Networks

Isola et al. [60] describe a method to do image-to-image translation. They use a GAN [50] to implement a network that is able to translate images from one domain to another.

There is a dataset of paired images required, e.g., satellite pictures and a street map for those images. The discriminator has then to learn and to decide if such a pair is real or generated. The generator is trained to translate one image of such a pair into the other. Therefore a generated pair contains a real image and a second image that is translated by the generator with the first image. The network can just be trained like any other GAN [50].

A detail of the network is the generator: Often simple encoder-decoder networks with a bottleneck are used, but the chosen architecture in this case has skip-connection from the layer i to the layer $n - i$, where n is the total layer count. This network architecture is based on U-Net [64].

The network architecture could be used for voice conversion. Pairs are then just two audio samples with the same sentence spoken by two different people. One drawback is that, depending on the task, relatively many such paired samples have to exist and the network must be trained for each speaker pair. The input could be a spectrogram or just raw data.

E. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

The proposed network architecture by Zhu et al. [59] is able to do a domain transfer on images. For example, it could

TABLE IV: A comparison of all style transfer models.

Paper	Method	Availability (code)	End-to-End	Training Time	Real-Time	Required Training Data	Elegant
A Neural Algorithm of Artistic Style [9]	Using the Gram matrix and an already existing CNN to extract and transfer a style with an iterative algorithm	yes (third-party)	no (iterative optimization)	No training required, assuming the CNN is given	no	None, assuming the CNN is given	yes
Perceptual Losses for Real-Time Style Transfer and Super-Resolution [63]	Improving [9] to allow real-time style transfer; an already existing CNN is required	yes	yes	unknown	yes	Original images and style transferred version (may be generated with [9])	yes (especially the loss function)
Learning to Discover Cross-Domain Relations with Generative Adversarial Networks [57]	Create two functions to convert objects from two different domains; using GANs	yes	yes	unknown	yes	Two datasets with images of the two domains (unpaired)	yes
Image-to-Image Translation with Conditional Adversarial Networks [60]	A GAN is used to check if a pair of datarecords from two domains is real or generated	yes	yes	unknown	yes	Paired data from two different domains	yes
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks [59]	Create two functions to convert objects from two different domains; using GANs	yes	yes	unknown	yes	Two datasets with images of the two domains (unpaired)	yes

change a horse to a zebra on an image. The training does not require any image pairs, but only two datasets of images.

The architecture is similar to the architecture proposed by Kim et al. [57]. Two mapping functions F and G are defined, which map from domain A to B and from domain B to A. The networks are trained that $F(G(x_b)) = x_b$ and $G(F(x_a)) = x_a$. There is also a discriminator loss that forces $F(x_a)$ to produce images from the domain B and $G(x_b)$ to produce images from the domain A. The network can also be seen as two autoencoders that are trained at the same time. The style transfer is done via the domain transfer.

A GAN-architecture [50] is used to implement the network. For the discriminator 70×70 sized convolutional PatchGANs are chosen for the implementation. These GANs only penalize structure at the scale of image patches.

One big difference to [57] is the used loss function. The proposed CycleGAN [59] uses the mean squared error, where the hinge loss is preferred for [57].

Compared to [60], the network is more complex, but on the other side, image pairs are no longer required. This makes the network much more useful, because in reality such pairs are often expensive. The network could be trained on audio data for two different speakers. The audio could be there as raw data or as a spectrogram. Depending on the task, many samples have to be available for good training results.

F. Comparison

All described papers about style transfer are compared in Table (IV) to give an overview over the different methods. For audio style transfer, [9] is an easy-to-implement choice, assuming a CNN is given that extracts good features. If no such CNN is given, but a paired dataset (audio files with the same content spoken by different voices), then [60] might be a good choice. If only two unpaired sets of data are available, then [57] or [59] are possible solutions for audio style transfer. If the audio style transfer for voices works, then the more specific task of voice conversion is also solved.

VII. Discussion

As described in Section IV, there are currently no perfect solutions based on neural networks for the voice conversion problem available. The only currently available candidate with good results is [29], but it is proprietary and it is not even known how it is implemented. Most solutions with neural networks contain much pre- and post-processing, but end-to-end solutions would be preferred. Audio style transfer-based solutions do not work well for our tests, because the classification networks do not focus on the same audio features as humans; therefore, the voice that is generated based on these features does not sound natural and almost no voice conversion is done. We did not test methods based on CycleGAN [59] and [60]; therefore, they could still work for voice conversion. It may also be the case that the used CNN for the audio style transfer could be trained somehow else to perform better on the style transfer task. Maybe it is easier to use an architecture without style transfer to solve voice conversion.

VIII. Conclusions

More research has to be done to solve the voice conversion task. Impressive results for images with style transfer show that neural networks are even able to solve complex transfer/conversion tasks. Also for the voice conversion itself, there are some networks available, but unfortunately there are no end-to-end solutions. Therefore, it should also be possible to solve the voice conversion with an end-to-end learning network architecture. This survey summarizes currently available methods that may help to solve this task, or at least to give an introduction into this topic.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR abs/1609.03499*, 2016.

[5] D. Palaz, M. M. Doss, and R. Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4295–4299.

[6] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, no. 55-66, p. 11, 2010.

[7] F. Stark, C. Hazırbas, R. Triebel, and D. Cremers, "Captcha recognition with active deep learning," in *GCPR Workshop on New Challenges in Neural Computation*, 2015.

[8] J. Nurminen, H. Silén, V. Popa, E. Helander, and M. Gabbouj, "Voice conversion," in *Speech enhancement, modeling and recognition-algorithms and applications*. InTech, 2012.

[9] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.

[12] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[13] D. O'Shaughnessy, *Speech communication: human and machine*. Addison-Wesley Publishing Company, 1987.

[14] E. (https://dsp.stackexchange.com/users/7390/edouard), "Reconstruction of audio signal from spectrogram," Digital Signal Processing. [Online]. Available: https://dsp.stackexchange.com/a/13401/26784

[15] J. Yamagishi, "An introduction to hmm-based speech synthesis," *Technical Report*, 2006.

[16] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[17] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[18] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, J. Raiman, S. Sengupta *et al.*, "Deep voice: Real-time neural text-to-speech," *arXiv preprint arXiv:1702.07825*, 2017.

[19] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," 2017.

[20] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[21] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[22] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*. GMD-Forschungszentrum Informationstechnik, 2002, vol. 5.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[24] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks." in *Interspeech*, 2014, pp. 1964–1968.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[26] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech syn," *arXiv preprint arXiv:1703.10135*, 2017.

[27] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949.

[28] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[29] A. de Brébisson, J. Sotelo, K. Kumar, A. Auvolat, É. Simon, R. Kumar, and T. Szymkowiak, "Lyrebird, an api for speech synthesis," https://lyrebird.ai/. [Online]. Available: https://lyrebird.ai/

[30] N. Perraudin, P. Balazs, and P. L. Sondergaard, "A fast griffin-lim algorithm," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, 2013, pp. 1–4.

[31] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[32] K. Shikano, K.-F. Lee, and R. Reddy, "Speaker adaptation through vector quantization," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, vol. 11. IEEE, 1986, pp. 2643–2646.

[33] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, "Voice conversion through vector quantization," *Journal of the Acoustical Society of Japan (E)*, vol. 11, no. 2, pp. 71–76, 1990.

[34] K. Tokuda, T. Kobayashi, and S. Imai, "Speech parameter generation from hmm using dynamic features," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1. IEEE, 1995, pp. 660–663.

[35] E.-K. Kim, S. Lee, and Y.-H. Oh, "Hidden markov model based voice conversion using dynamic characteristics of speaker." in *EUROSPEECH*, 1997.

[36] A. Kain and M. W. Macon, "Spectral voice conversion for text-to-

speech synthesis," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 1. IEEE, 1998, pp. 285–288.

[37] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE Transactions on speech and audio processing*, vol. 6, no. 2, pp. 131–142, 1998.

[38] T. Nakashika, R. Takashima, T. Takiguchi, and Y. Ariki, "Voice conversion in high-order eigen space using deep belief nets." in *Interspeech*, 2013, pp. 369–372.

[39] T. Nakashika, T. Takiguchi, and Y. Ariki, "Voice conversion in time-invariant speaker-independent space," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7889–7893.

[40] D. Saito, F. Villavicencio, J. Yamagishi, T. Tomoki, M. Wester, Z. Wu, L.-H. Chen *et al.*, "The voice conversion challenge 2016," 2016.

[41] D. Huang, L. Xie, Y. Lee, J. Wu, H. Ming, X. Tian, S. Zhang, C. Ding, M. Li, Q. H. Nguyen *et al.*, "An automatic voice conversion evaluation strategy based on perceptual background noise distortion and speaker similarity," in *9th ISCA Speech Synthesis Workshop (SSW9)*, 2016.

[42] L. Sun, S. Kang, K. Li, and H. Meng, "Voice conversion using deep bidirectional long short-term memory based recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4869–4873.

[43] L.-H. Chen, Z.-H. Ling, L.-J. Liu, and L.-R. Dai, "Voice conversion using deep neural networks with layer-wise generative training," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 12, pp. 1859–1872, 2014.

[44] J. Lai, B. Chen, T. Tan, S. Tong, and K. Yu, "Phone-aware lstm-rnn for voice conversion," in *Signal Processing (ICSP), 2016 IEEE 13th International Conference on*. IEEE, 2016, pp. 177–182.

[45] H. Banno, H. Hata, M. Morise, T. Takahashi, T. Irino, and H. Kawahara, "Implementation of realtime straight speech manipulation system: Report on its first implementation," *Acoustical science and technology*, vol. 28, no. 3, pp. 140–146, 2007.

[46] S. Pascual and A. Bonafonte, "Multi-output rnn-lstm for multiple speaker speech synthesis with α-interpolation model," *way*, vol. 1000, p. 2.

[47] A. Bonafonte, P. D. Agüero, J. Adell, J. Pérez, and A. Moreno, "Ogmios: The upc text-to-speech synthesis system for spoken translation," in *TC-STAR Workshop on Speech-to-Speech Translation*, 2006, pp. 199–204.

[48] S. Mobin and J. Bruna, "Voice conversion using convolutional neural networks," *arXiv preprint arXiv:1610.08927*, 2016.

[49] D. Erro, I. Sainz, E. Navas, and I. Hernáez, "Improved hnm-based vocoder for statistical synthesizers." in *INTERSPEECH*, 2011, pp. 1809–1812.

[50] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[51] K. Sathiarekha and S. Kumaresan, "A survey on the evolution of various voice conversion techniques," in *Advanced Computing and Communication Systems (ICACCS), 2016 3rd International Conference on*, vol. 1. IEEE, 2016, pp. 1–5.

[52] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6964–6968.

[53] I. Karpov and D. Subramanian, "Hidden markov classification for musical genres," *Course Project*, 2002.

[54] K. M. Chit and K. Z. Lin, "Audio-based action scene classification using hmm-svm algorithm," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, no. 4, pp. pp–1347, 2013.

[55] E. Batlle and P. Cano, "Automatic segmentation for music classification using competitive hidden markov models," 2000.

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[57] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017.

[58] W. Xu, A. Ritter, W. B. Dolan, R. Grishman, and C. Cherry, "Paraphrasing for style," in *24th International Conference on Computational Linguistics, COLING 2012*, 2012.

[59] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[60] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[61] E. W. Weisstein, "Gram matrix. From MathWorld—A Wolfram Web Resource," http://mathworld.wolfram.com/GramMatrix.html. [Online]. Available: http://mathworld.wolfram.com/GramMatrix.html

[62] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[63] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[64] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[65] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," *Linguistic data consortium*, vol. 10, no. 5, p. 0, 1993.

[66] D. Ulyanov and V. Lebedev, "Audio texture synthesis and style transfer." [Online]. Available: https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/

[67] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[68] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[69] K. Park, "A (heavily documented) tensorflow implementation of tacotron." [Online]. Available: https://github.com/Kyubyong/tacotron

[70] J. Sotelo and K. Kumar, "Speech synthesis using recurrent neural networks." [Online]. Available: https://github.com/sotelo/parrot

APPENDIX

A. Preliminary Experiments

During this work many experiments have been conducted. Some of these experiments and the most important results are documented in this section.

The main idea is to use audio style transfer to do the voice conversion. It should work similar like the style transfer for images. Our implemented style transfer is based on the method described by Gatys et al. [9], because it already works very well for images and the architecture is relatively simple. To implement this architecture, a classification network for audio data that is based on a CNN is required. To train this network, we use different datasets: A proprietary speech dataset, TIMIT [65] and a YouTube dataset that contains about $1'000$ speakers with each 1 h of audio. Other possible large datasets of spoken text would be audio books.

For our experiments, we mostly use the YouTube dataset, because it is the largest of the three datasets. The TIMIT dataset has the highest quality, but it is quite small. The proprietary dataset is required for the final task, but it is also very small; therefore, it is not the preferred choice for the development of a network architecture.

For each network architecture, we test different input formats: Raw audio (1D stream) [5] [52], mel-cepstral as 1D image [52] [66] and mel-cepstral as 2D image [11]. The trained network is a classification network that is used to do the style transfer described by Gatys et al. [9]. Networks with 1D convolutions have the advantage that often much less training time is required, because the processing of the data requires fewer computations. On the other side, 2D convolutions may be more powerful for some tasks, especially [11] shows that 2D convolutions also work very well for audio data. For audio data it is not obvious if 1D convolutions or 2D convolutions should be preferred, because audio data is in general 1D, but spectrograms may be seen as a 2D image.

The network described by Lukic et al. [11] is not able to generate high quality voice conversion results with the given style transfer. An example of this style transfer with a mel-cepstral that is handled as a 2D image is shown in Figure (7). The input is a female voice of the TIMIT dataset and it should be converted to a male voice. This conversion does not work and the output sounds similar to the input, except for some noise that is added. The reason why this architecture does not work is probably, because the network uses different features to classify a voice than humans do. The network is not very deep and therefore it can be assumed that the generated features are less abstract than, e.g., the features that are used in the style transfer described by Gatys et al. [9].

For image data, there is the public available VGG-16 network [2]. All pre-trained weights may be downloaded for free in the internet. This deep network architecture is based on a simple CNN and performs very well for the style transfer task for images [9]. We use a network that has a similar architecture, but that includes batch normalization [67] and uses audio data as input. The classification quality for the

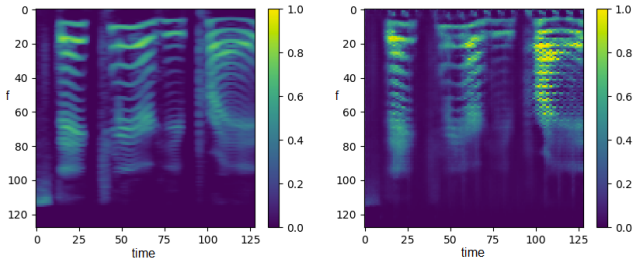

Fig. 7: An example of an audio input (left) and audio output (right) of a style transfer with the architecture described by Lukic et al. [11]. A female voice from the TIMIT [65] dataset should be converted to a male voice. The x axis describes the time and the y axis the mel-scale. The values are normalized to the range $[0, 1]$.

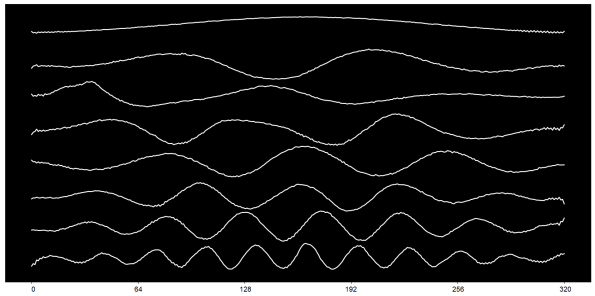

Fig. 8: Some trained weights for an initial 1D convolution with the length 320 and a stride of 160.

YouTube dataset is good, but the style transfer based on [9] does not work. At this point it must be assumed that neural networks use other features to classify a voice than humans do. A possible solution to this problem could be if a larger dataset with not only voices, but many more different sound classes is used for the training. This would force the network to include more different features in the classification process. We still assume that the style transfer described by Gatys et al. [9] works also well for audio data, but it depends a lot on a good and very general trained classification network. This seems to be a much harder issue for audio data than for image data.

In general, our experiments are done with fixed-size inputs, because this makes the development easier and the training faster. Both described architectures are tested with different input data types. They are compared in Table (V). Both approaches are based on a simple CNN. The main difference between these two architectures is that the architecture proposed by Lukic et al. [11] is much smaller than the VGG-16 [2] based model. Therefore, it requires less training time. The quality, at least for the tested speaker classification task, is very similar and both architectures reach high accurate results. All tests are conducted on a NVIDIA TITAN Xp. For the audio style transfer / voice conversion task none of these networks works well. The outputs are always similar to the inputs, except for some additional noise. It does not seem that

TABLE V: Different trained classification networks for the style transfer described by Gatys et al. [9] are compared to each other. For all networks a YouTube dataset with $1'000$ speakers is used for the training. The voice conversion task does not work for any of these networks in a minimum required quality; therefore, there are no more details about the voice conversion in this table.

Architecture	Input Format	Classification works	Training Time
Based on [11]	2D mel-cepstral	yes	7 h
	1D mel-cepstral	yes	3 h
	2D Fourier spectrogram	ok, but not very well	$>$ 10 h, slow convergence
	1D Fourier spectrogram	yes	3 h
	raw audio / 1D after initial layer	yes	5 h
	raw audio / 2D after initial layer	no	no convergence
Based on VGG-16 [2]	2D mel-cepstral	yes	22 h
	1D mel-cepstral	yes	10 h
	2D Fourier spectrogram	ok, but not very well	$>$ 25 h, slow convergence
	1D Fourier spectrogram	yes	10 h
	raw audio / 1D after initial layer	yes	12 h
	raw audio / 2D after initial layer	no	no convergence

any style or voice transfer is done.

It is notable that the raw input forces the network architecture to learn something like a Fourier transform in the initial convolutional layer. To obtain this result we use a convolution with the length 320 and a stride of 160. The used non-linearity is ReLU [68] and we do not use a bias. If the resulting lists of feature weights are ordered by their lowest frequency and are plotted, it might be seen that they look like sine-curves with different frequencies. This is visualized in Figure (8). This fact is already described by Palaz et al. [5]. It allows to do "feature learning" [17] and the network then no longer depends on a specific transform, e.g. the Fourier transform, and therefore does not need any pre-processing. This initial trained layer may be frozen and reused for other networks. Unfortunately, this learned transform is only powerful if the following layers are also 1D convolutions and not 2D convolutions. The main reason for this is, because of the initial 1D convolution there is a random order for the frequencies of the trained transform. This is not a problem for 1D convolutions, but for 2D convolutions the order of the frequencies is very important. The training does not converge to a good solution, if directly after the initial 1D convolution 2D convolutions are used.

Ulyanov et al. [66] describe a simple network that is able to do audio style transfer based on [9]. They use a spectrogram as input and then one 1D convolutional layer for their network. All weights are initialized randomly and therefore no classification network has to be trained. For some sounds this architecture works well, but for voices the style transfer does not work. Our tests include different filter sizes and layer counts, but in the end it does not seem to be possible to implement voice transfer with randomly initialized weights. It is still impressive that this architecture already works a bit for some simple sounds.

Additional tests could be done with the CycleGAN [59] network. This architecture directly may be used if a fixed-size spectrogram is used as a 2D input image. It also might be modified to take a fixed-size raw input or a spectrogram as a 1D image with a feature for each frequency. Probably much data is required for the training, therefore the audio dataset should be large. A big advantage of the architecture is that it does not require a paired dataset. On the other side, the architecture only allows to train a conversion between two speakers, therefore the training has to be done for each possible speaker pair. Nevertheless, this approach is already very powerful for a general domain transfer on images and could also work for audio data.

If a paired dataset is available, where each spoken text is present for two speakers, the pix2pix [60] architecture may be used. The training is more efficient and it creates often better results than CycleGAN as shown in [59]. On the other side, often there is no paired dataset available. Especially for real use cases this might be a problem.

The audio synthesizer is a very important part of the final neural network, therefore we also try to analyse Tacotron [26] and Char2Wav [19] . The available third-party Tacotron implementation [69] still does not produce good results after one week of training and the public Char2Wav implementation [70] is not complete and therefore the training does not produce any useable results. Wavenet is not tested, because it takes too much time for the training and for the runtime. Especially Tacotron would be a good solution, because it is trained end-to-end and therefore could be included easily in a custom neural network architecture.

Our experiments do not produce any audio style transfer or voice conversion, but we are able to describe why our network architectures do not work well in combination with the style transfer described in [9]. We also show that for 2D CNNs the mel-cepstral works much better as an input than a plain Fourier spectrogram. Another interesting fact that we can show, is that a neural network learns a transform that is similar to the Fourier transform, if the input of the network is raw data and the initial layer is a convolutional layer.