# Real Non-Volume Preserving Voice Conversion

Santiago Pascual, Joan Serrà, Antonio Bonafonte

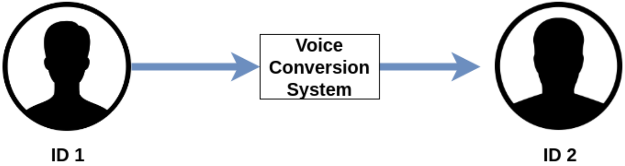
December 8, 2018

Universitat Politècnica de Catalunya, Barcelona, Spain
Telefónica Research, Barcelona, Spain

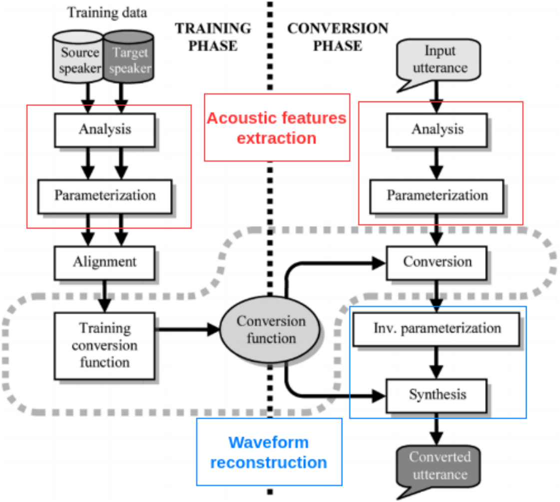
# Table of contents

# Introduction

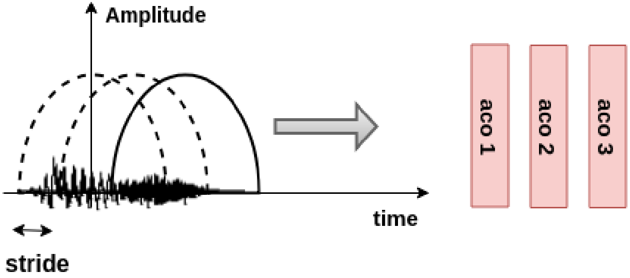
- Voice conversion binds a transformation between two speakers.
- The contents uttered by a source speaker are transferred to a target speaking style and identity.



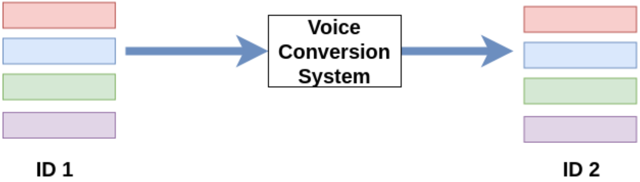
# Introduction: Voice Conversion Pipeline



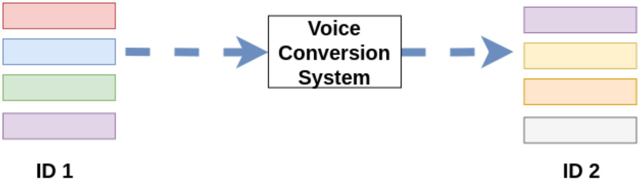
- Classical conversion pipelines work in an acoustic domain after signal framing.
- We use a vocoder (Ahocoder) to make aco. frames $x_n \in \mathbb{R}^{43}$: 40 MFCC, 1 logF0, 1 voiced/unvoiced flag, 1 max. voiced freq.



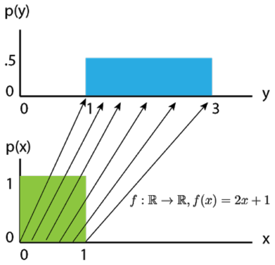
- Supervised training of the conversion function $f$: we have matching frames b/w speakers, they say the same.



- Challenging: Unsupervised, no labeled conversions to targets: speakers differ in contents and/or language!



Fundamentals of NF: learn invertible, volume-tracking transformations of distributions that we can manipulate easily [1].




Green square: Uniform(0, 1). Blue square: $Y = f(X) = 2X + 1$. Y is thus a simple affine (scale and shift) transformation of the underlying source distribution X.

[1] https://blog.evjang.com/2018/01/nf1.html

Preserve total probability: change of $p(x)$ along $dx$ must be equivalent to change of $p(y)$ along $dy$:

$$p(x)dx = p(y)dy$$

Only care about the amount of change in $y$ and not its direction:

$$p(y) = p(x)|\frac{dx}{dy}|$$

$$\log p(y) = \log p(x) + \log|\frac{dx}{dy}|$$

Only care about the amount of change in $y$ and not its direction:

$$p(y) = p(x)|\frac{dx}{dy}|$$

$$\log p(y) = \log p(x) + \log|\frac{dx}{dy}|$$

Going N-dimensional: volume change is the transformation matrix determinant.

$$y = f(x)$$

$$p(y) = p(x) \cdot |detJ(x)|$$

$$\log p(y) = \log p(x) + \log|detJ(x)|$$

Going N-dimensional: volume change is the transformation matrix determinant.
**Additionally enforce function $f$ to have inverse $f^{-1}$:**

$$y = f(x)$$

$$p(y) = p(f^{-1}(y)) \cdot |detJ(f^{-1}(y))|$$

$$\log p(y) = \log p(f^{-1}(y)) + \log |detJ(f^{-1}(y))|$$

NFs are based on the concept of bijective transformations (bijectors). A bijector will implement:

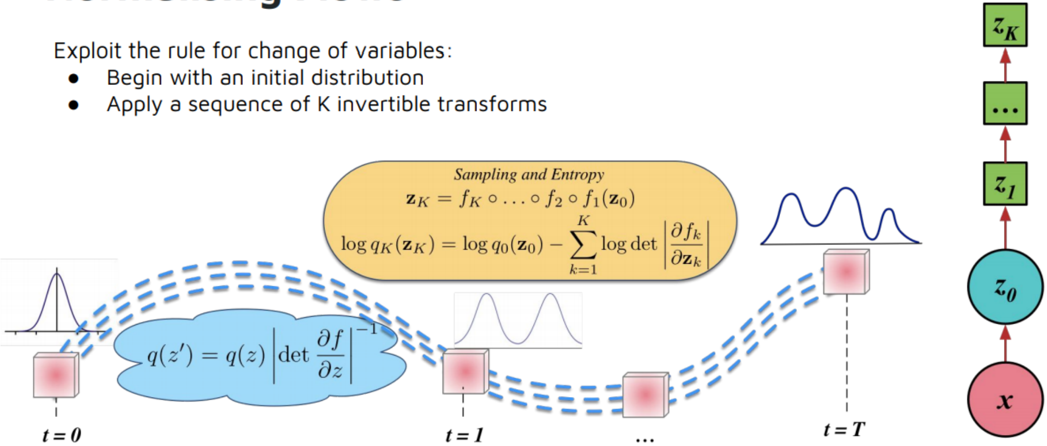
- A forward transformation $y = f(x)$ where $f : \mathbb{R}^d \to \mathbb{R}^d$.
- its inverse transformation $x = f^{-1}(y)$.
- the inverse log determinant of the Jacobian $\log |detJ(f^{-1}(y))|$ (ILDJ).

If bijector has tunable parameters $\rightarrow$ can be learned to transform a base distribution $\mathcal{X}$ to suit an arbitrary density $\mathcal{Z}$, and go back!

# Normalising Flows

Exploit the rule for change of variables:

- Begin with an initial distribution
- Apply a sequence of K invertible transforms




**Distribution flows through a sequence of invertible transforms**

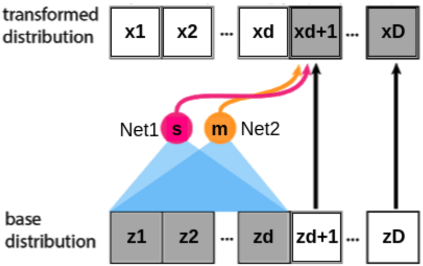
*Rezende and Mohamed, 2015*

# Introduction: Real Non-Volume Preserving Flows (Dinh et al. 2016)

Let $1 < d < D$, $\odot$ element-wise multiplication and $m$, $s$ two mappings $\mathbb{R}^d \to \mathbb{R}^{D-d}$. R-NVPs are defined as [2]:

$$\boldsymbol{x}_{1:d} = \boldsymbol{x}_{1:d},$$

$$\boldsymbol{x}_{d+1:D} = \boldsymbol{z}_{d+1:D} \odot \exp(s(\boldsymbol{z}_{1:d})) + m(\boldsymbol{z}_{1:d})$$




[2] http://akosiorek.github.io/ml/2018/04/03/norm_flows.html#simple_flows

Forward transformation (sampling):

- Copy first part of dimensions.
- Scale and shift the other part by learnable parameters.

Fully parallelizable!. Inverse transformation (inference):

$$\boldsymbol{z}_{1:d} = \boldsymbol{x}_{1:d}$$

$$\boldsymbol{z}_{d+1:D} = (\boldsymbol{x}_{d+1:D} - m(\boldsymbol{x}_{1:d}))/\exp(s(\boldsymbol{x}_{1:d}))$$

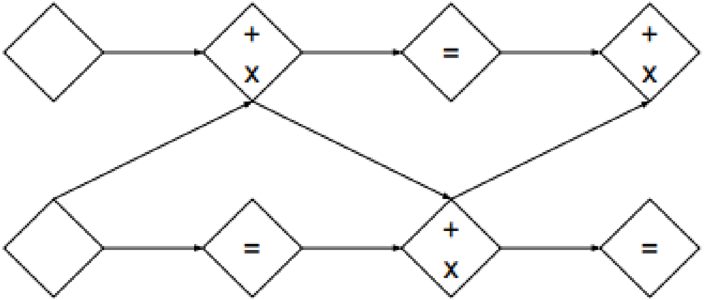
This operation is the affine coupling layer.

The determinant of this layer is as simple as:

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \operatorname{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}$$

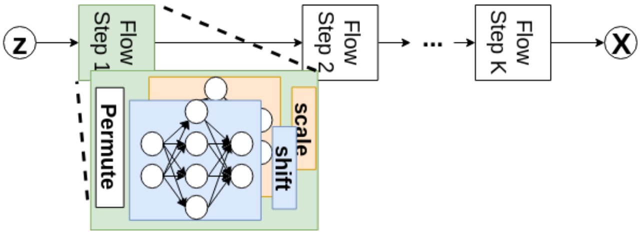
Where $s(x_{1:d})$ is the predicted scale vector $\rightarrow$ Not necessary to compute $s$ or $m$ Jacobians; $s$ and $m$ can be arbitrarily complex (e.g. MLPs).

- Certain dimensions being just copied and forwarded.
- Permute the intermediate vectors and concatenate many affine coupling flows.
- After enough levels everything is transformed.



# Real Non-Volume Preserving Voice Conversion

- Use $K = 6$ RNVP-like affine blocks.
- Each block is an MLP w/ 3 layers of sizes: $h_1 = 256$, $h_2 = 256$, and $h_3 = 43$ and LeakyReLUs.



- Project $x$ frames from any speaker to $z$ w/ reverse flow $f$.
- Compute likelihood of $z$ belonging to an isotropic Gaussian distribution.
- Completely unsupervised task with all our pool of speakers.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} - \log p_{\theta}(x_i)$$

## Mean Shift Conversion Method

Once RNVP-VC is trained on the mapping $z = f(x)$:

- Infer $z$ samples for all training frames of source spk and target spk, storing vectors $z^S_{\text{mean}}$ and $z^T_{\text{mean}}$ .
- Conversion: source speaker frames $x^S_n \in \mathbb{R}^{43}$ are transformed into latent space features $z^S_n \in \mathbb{R}^{43}$.
- Shift $z^S_n$ to $z^T_n$ like: $z^T_n = z^S_n + \alpha(z^T_{\text{mean}} - z^S_{\text{mean}})$ , with hyperparameter $\alpha$ controlling trade-off "distortion vs id change".

# Initial Results

We train RNVP-VC with 2 speakers from CMU Arctic datset [3]: awb (male) and slt (female). We post some initial conversion results b/w these speakers online: http://veu.talp.cat/rnvpvc .

[3] http://festvox.org/cmu_arctic/

# Conclusions

## Conclusions

- An unsupervised approach to voice conversion has been shown with the use of normalizing flows.
- A stack of RNVP-like blocks acts as a density transformation from acoustic space $\mathcal{X}$ to latent space $\mathcal{Z}$.
- The mean-shift operation can be used to transform identity in $\mathcal{Z}$ space.
- Preliminary results show potential of this generative approach for unaligned voice conversion, leaving room for further improvement.

**Thanks!**