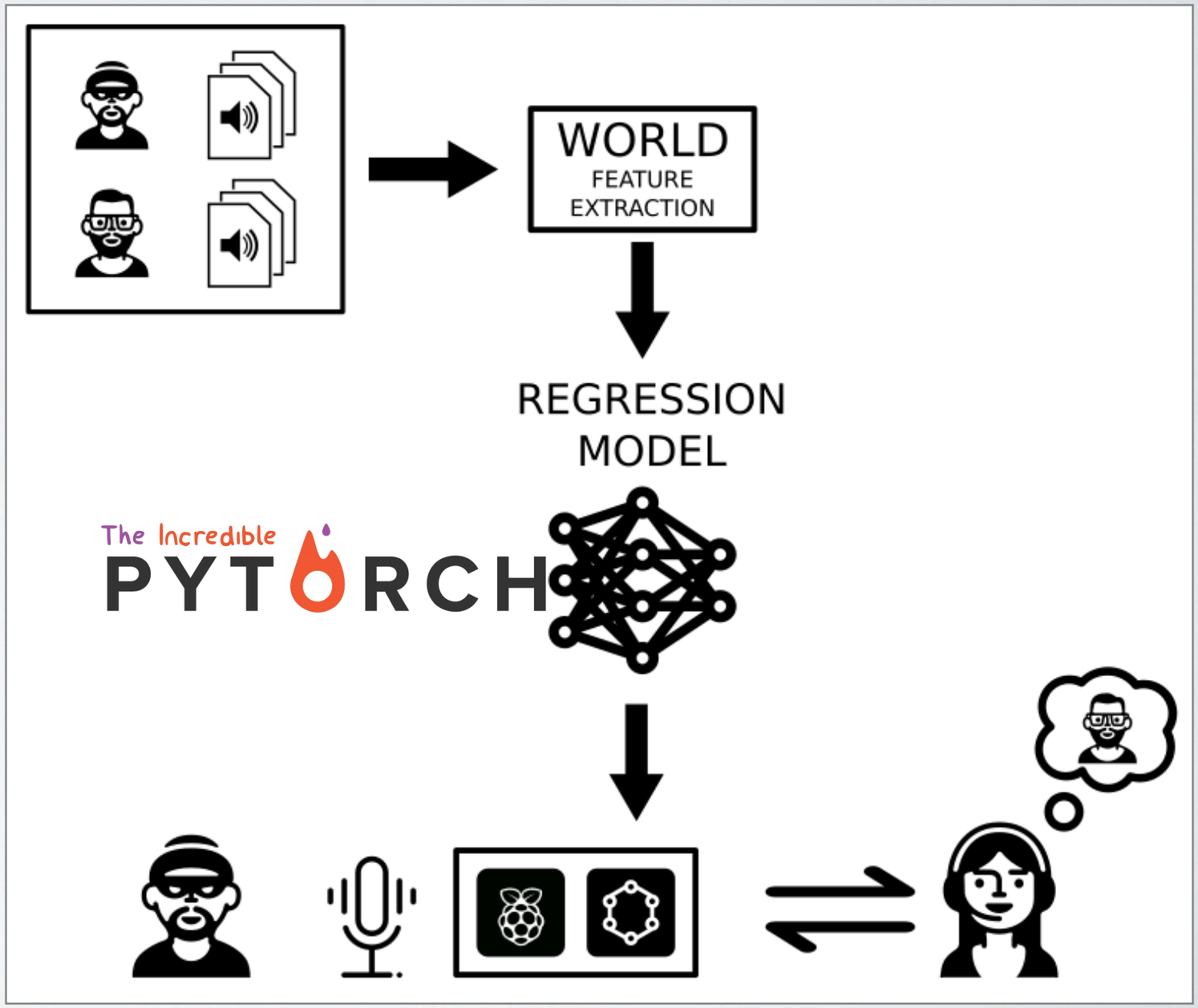
NEURAL PARAMETRIC VOICE CONVERSION

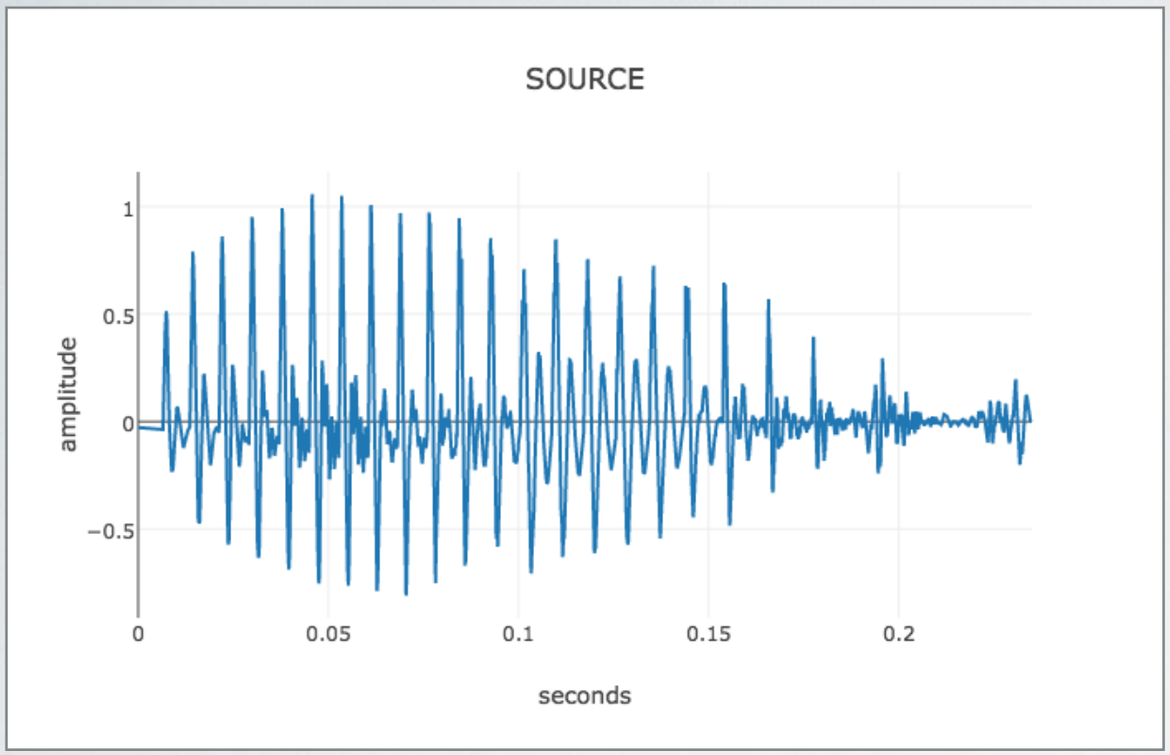
Alberto Mur
Javier Antorán

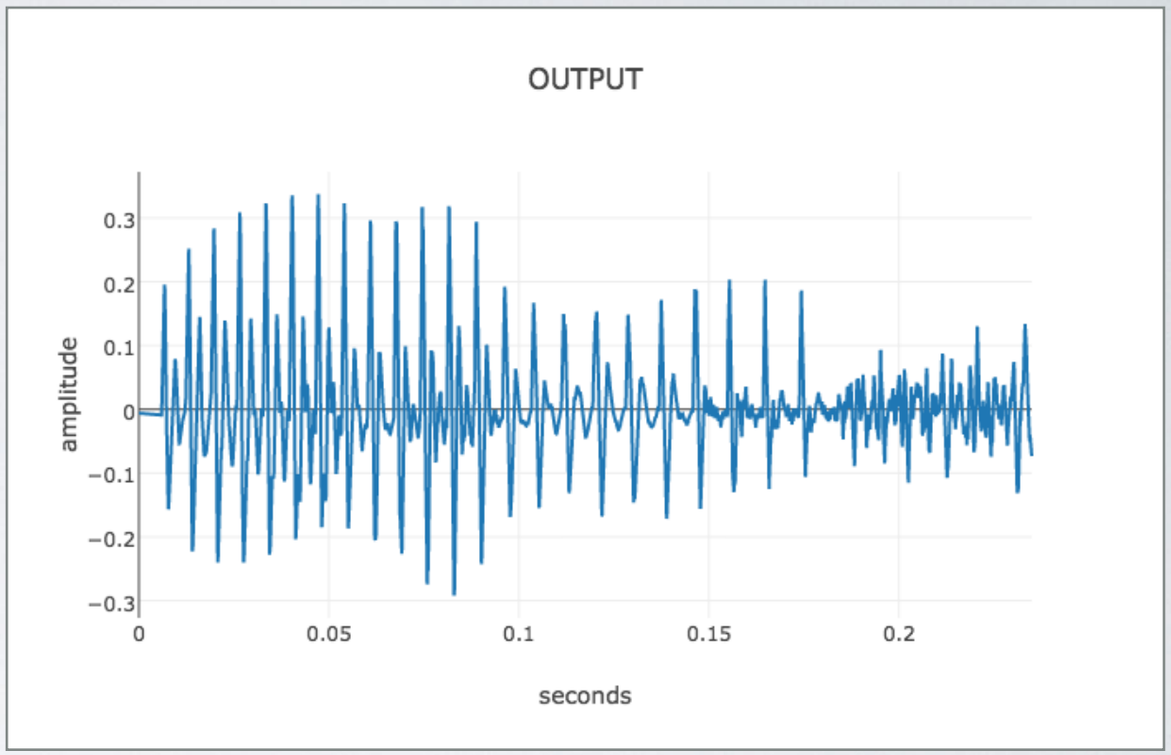
ARCHITECTURE

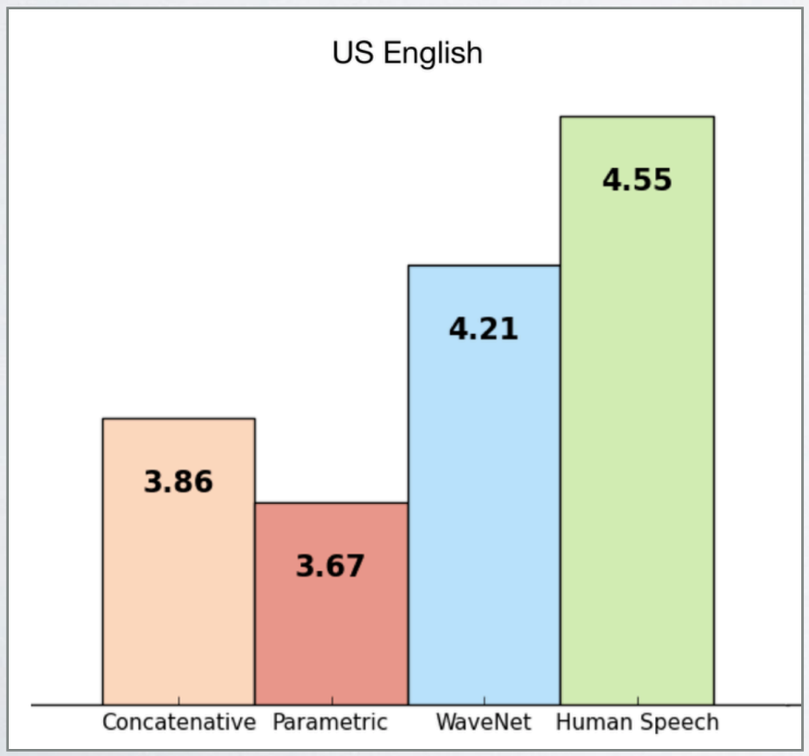
- Raspberry Pi
- Google Collab
- Python + Numpy
- WORLD
- SPTK
- PyTorch

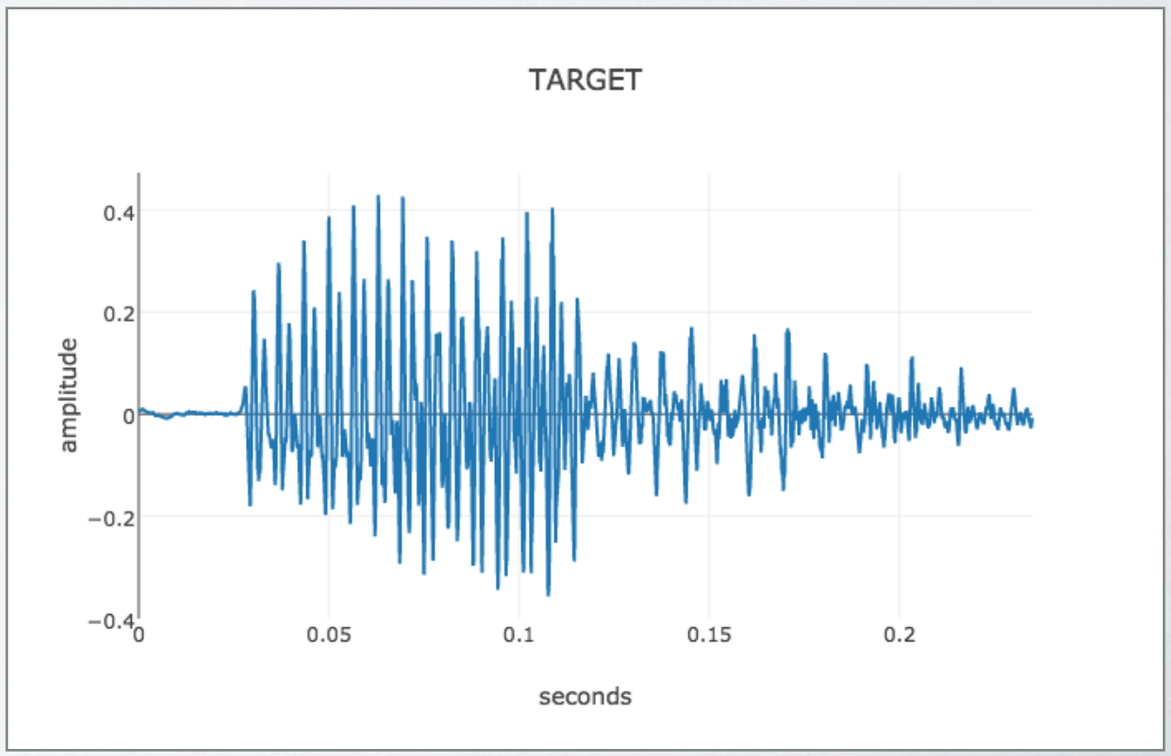


VOICE CONVERSION

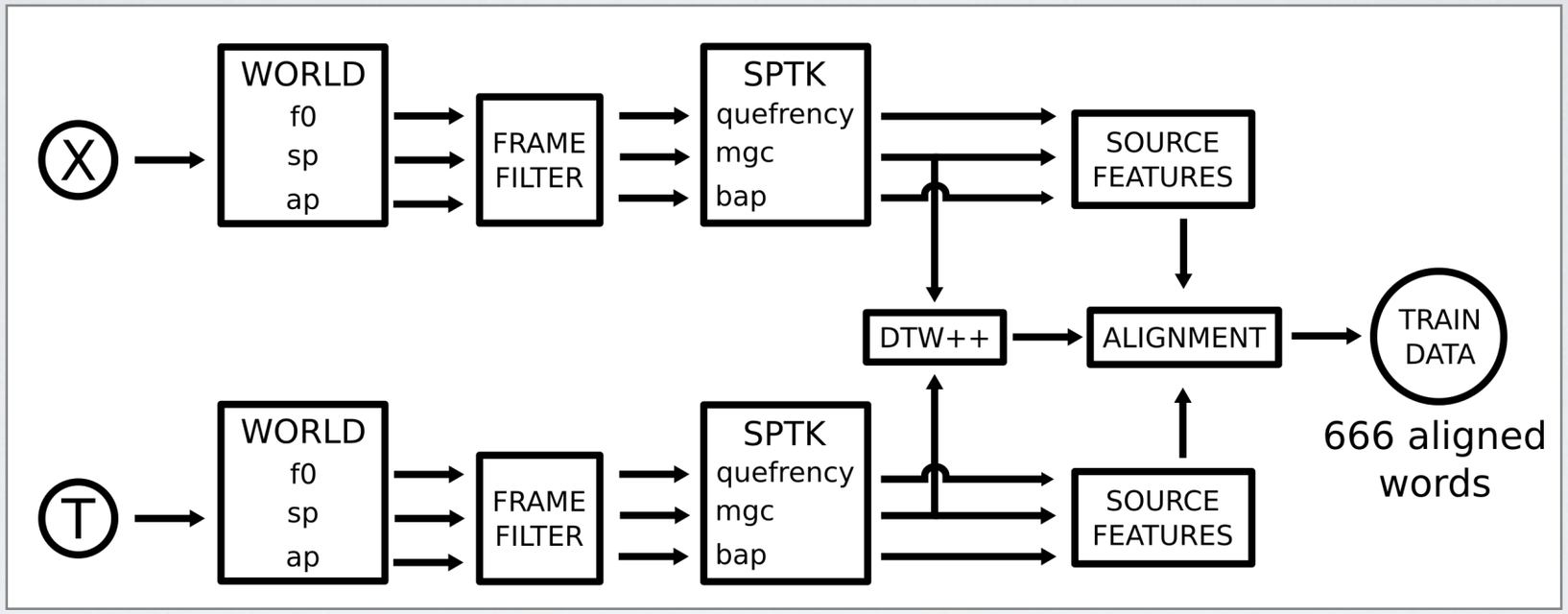


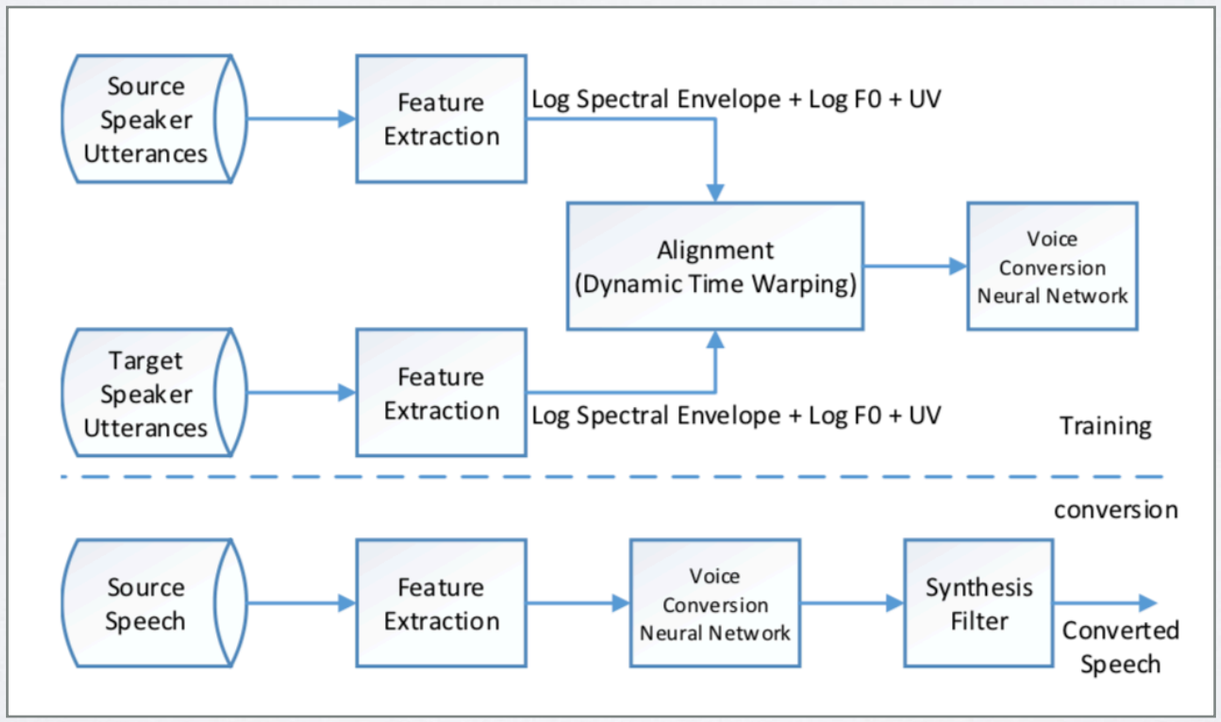




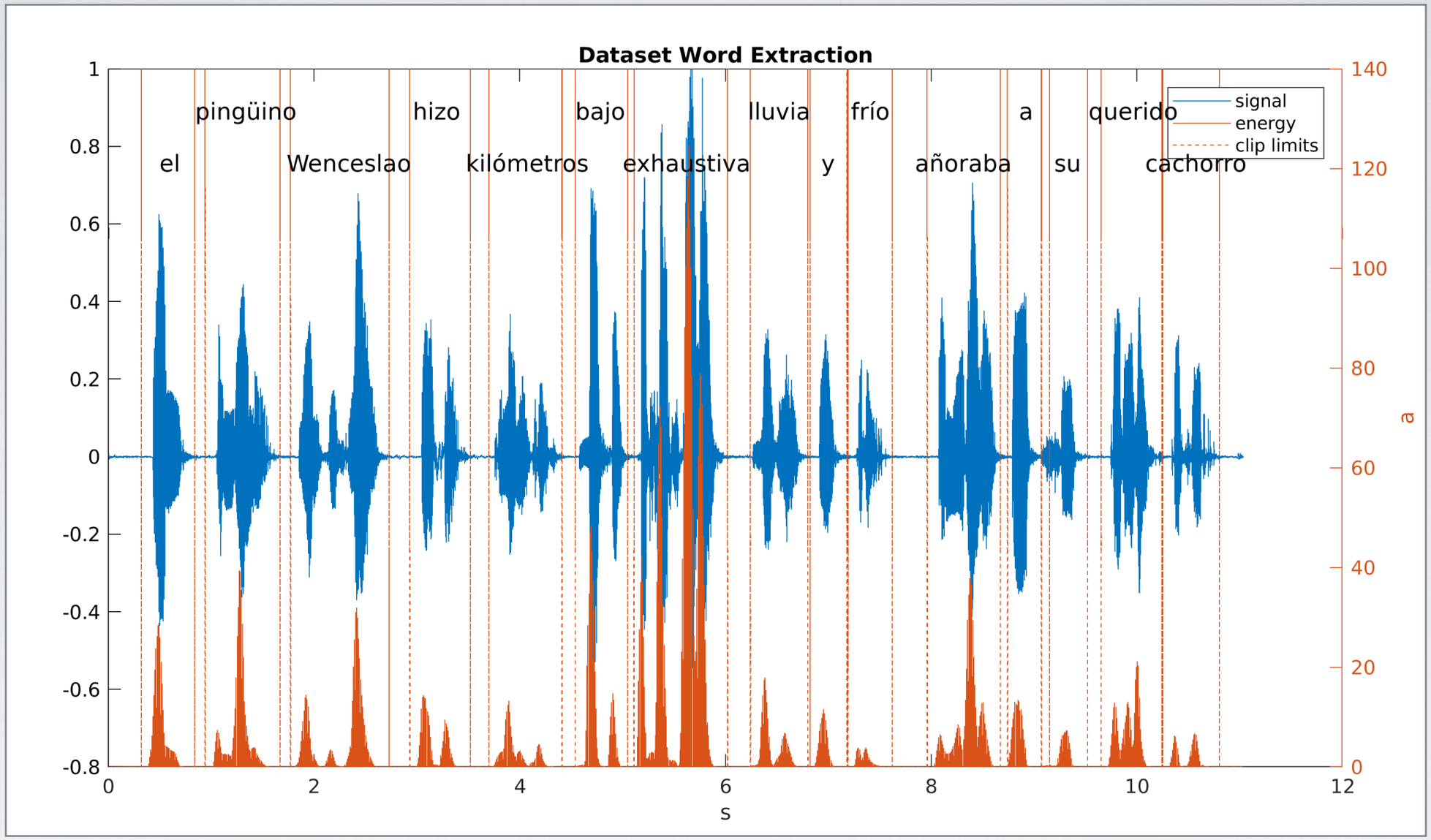


PROCESSING PIPELINE



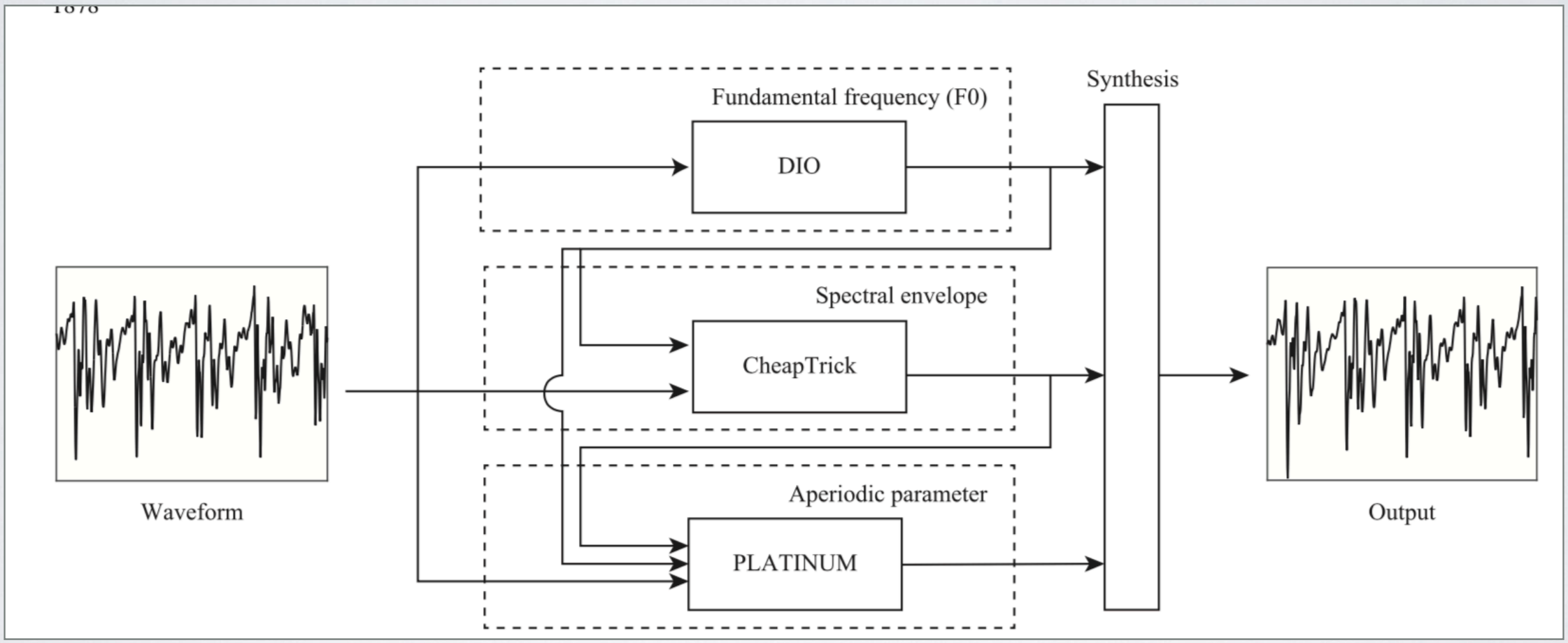


DATA PROCESSING



- Silence filter + Frame filter + Separator

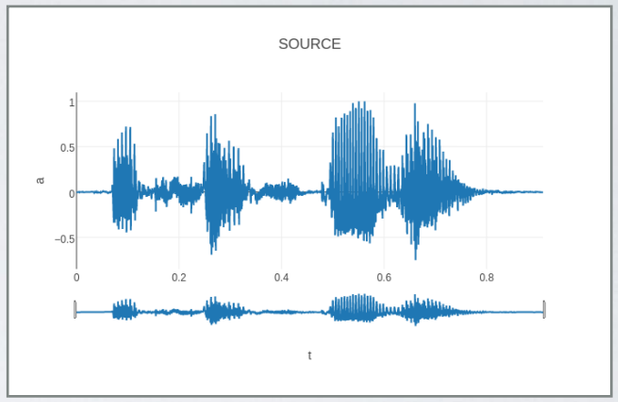
WORLD VOCODER

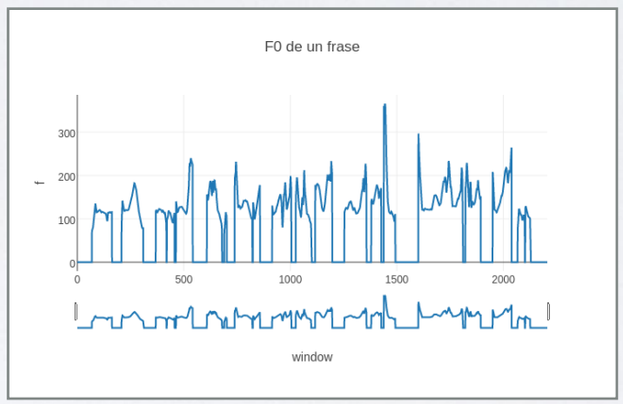


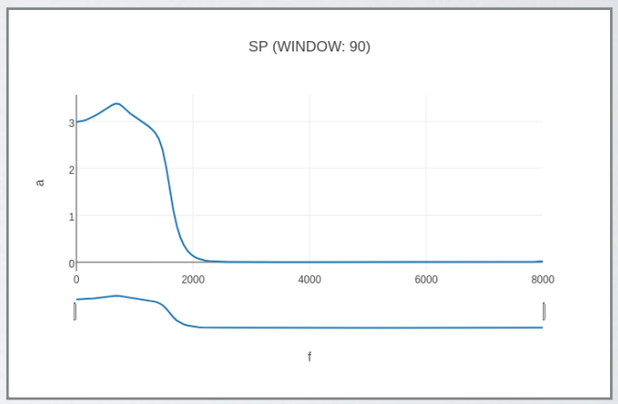
FEATURE EXTRACTION:

- WORLD
  - F0, UV
  - SP
  - AP
- SPTK
  - MGCC
  - logF0

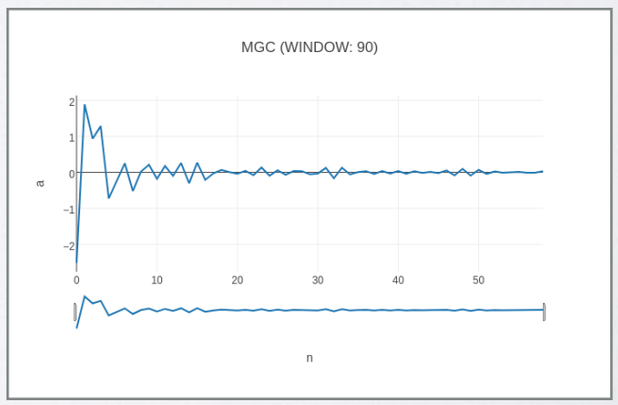
WORLD





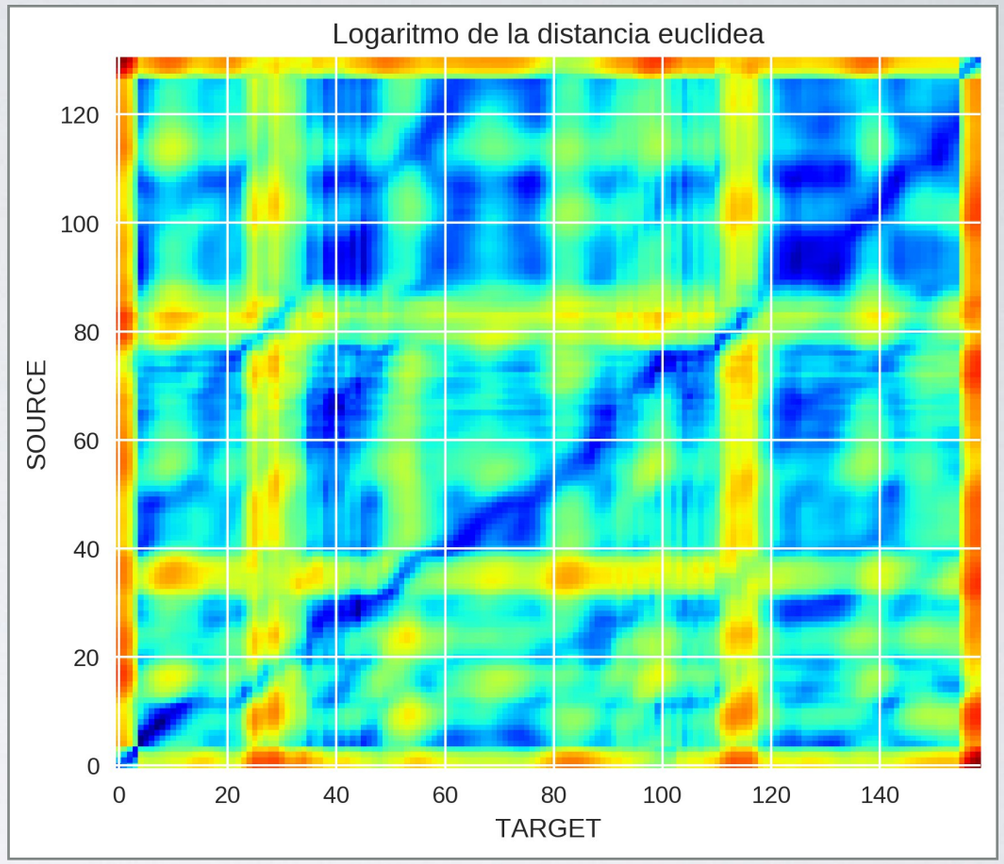


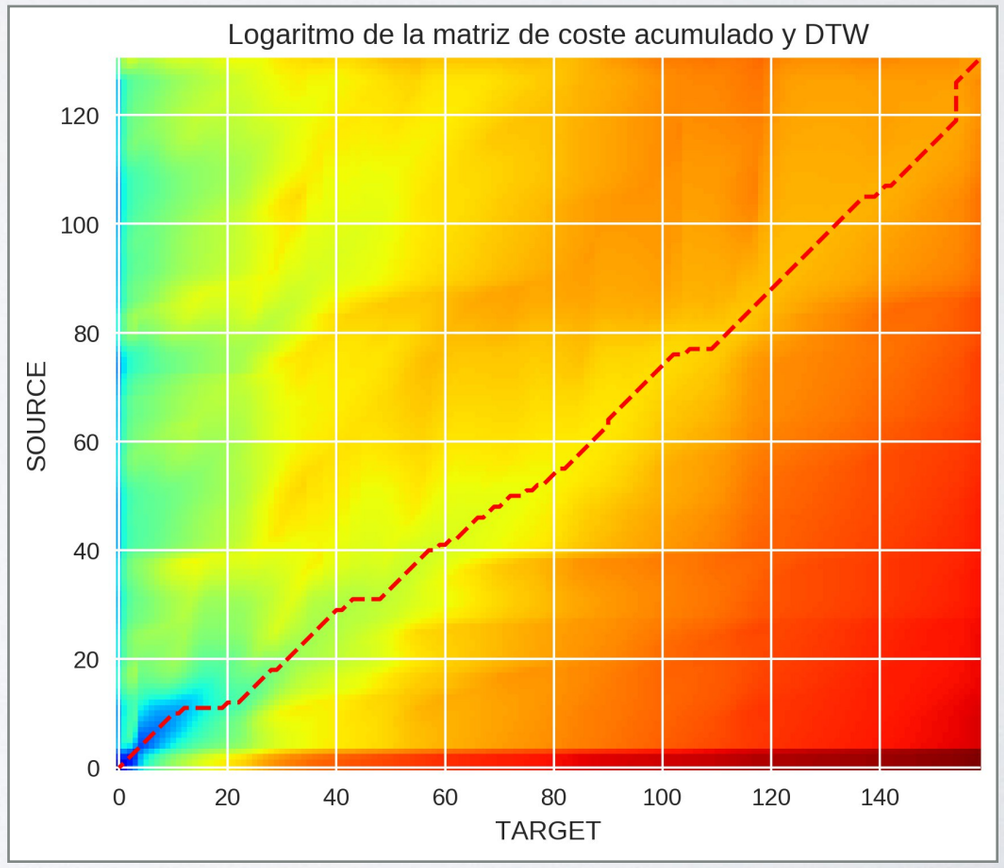
SPTK



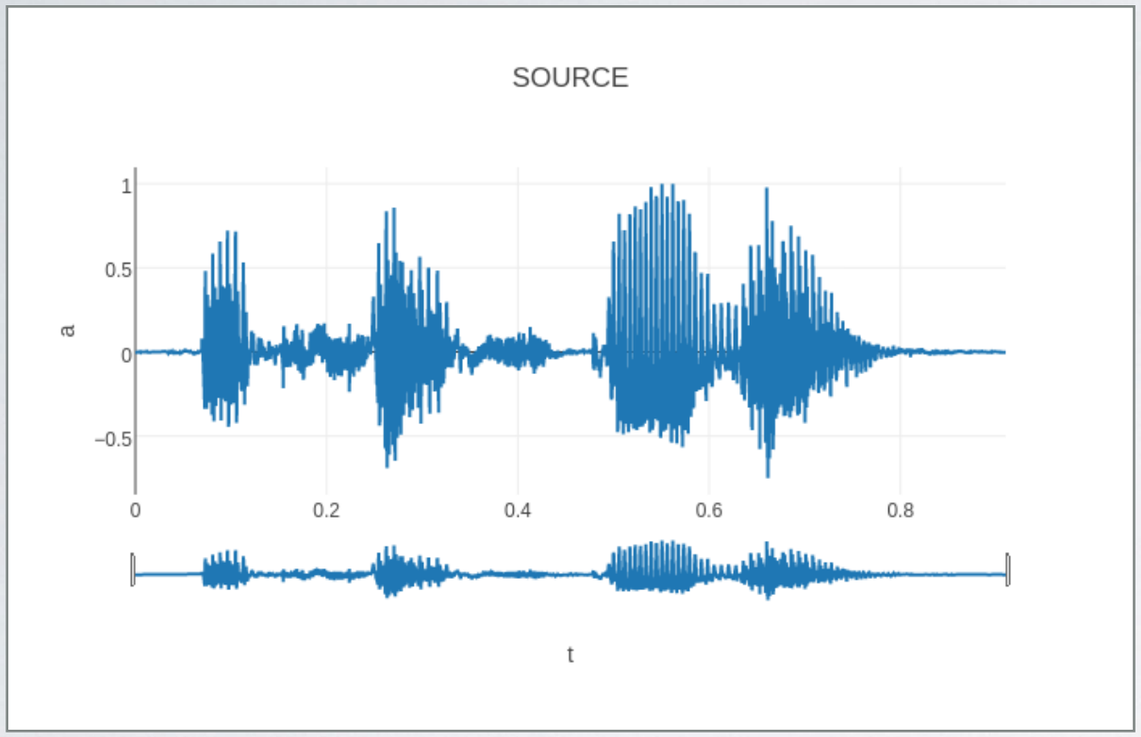
DTW ++

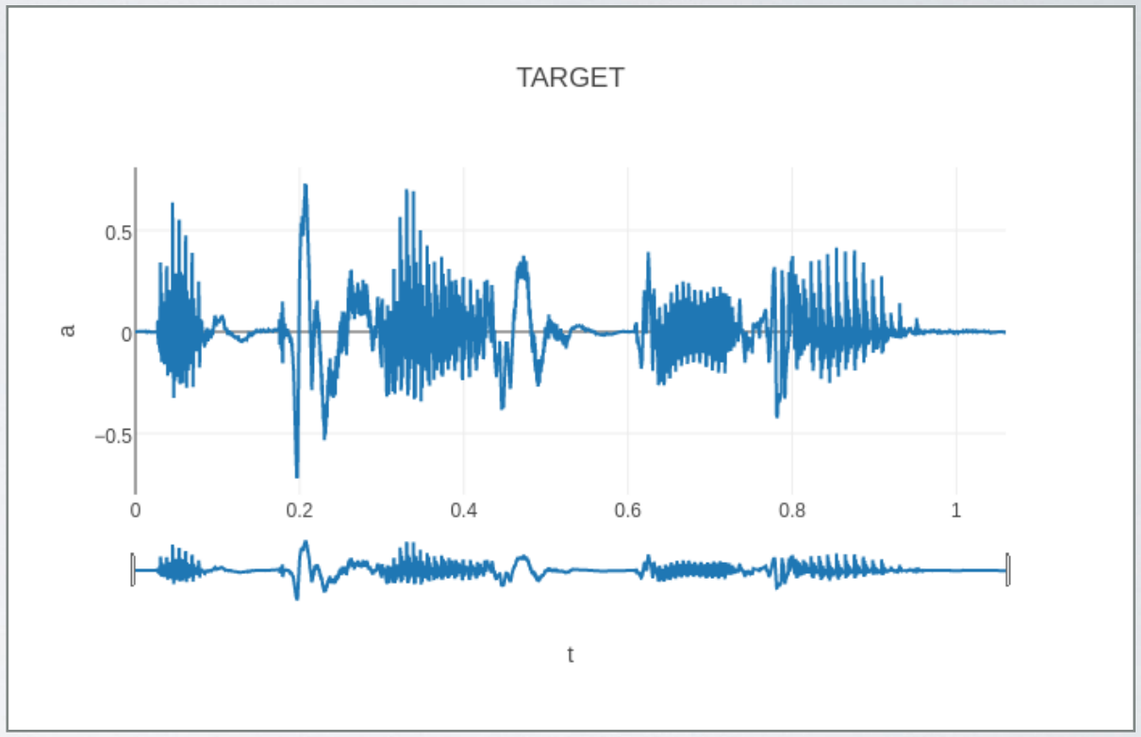
- Time derivatives
- Horizontal step cost
- Upper left corner cost
- Starting window

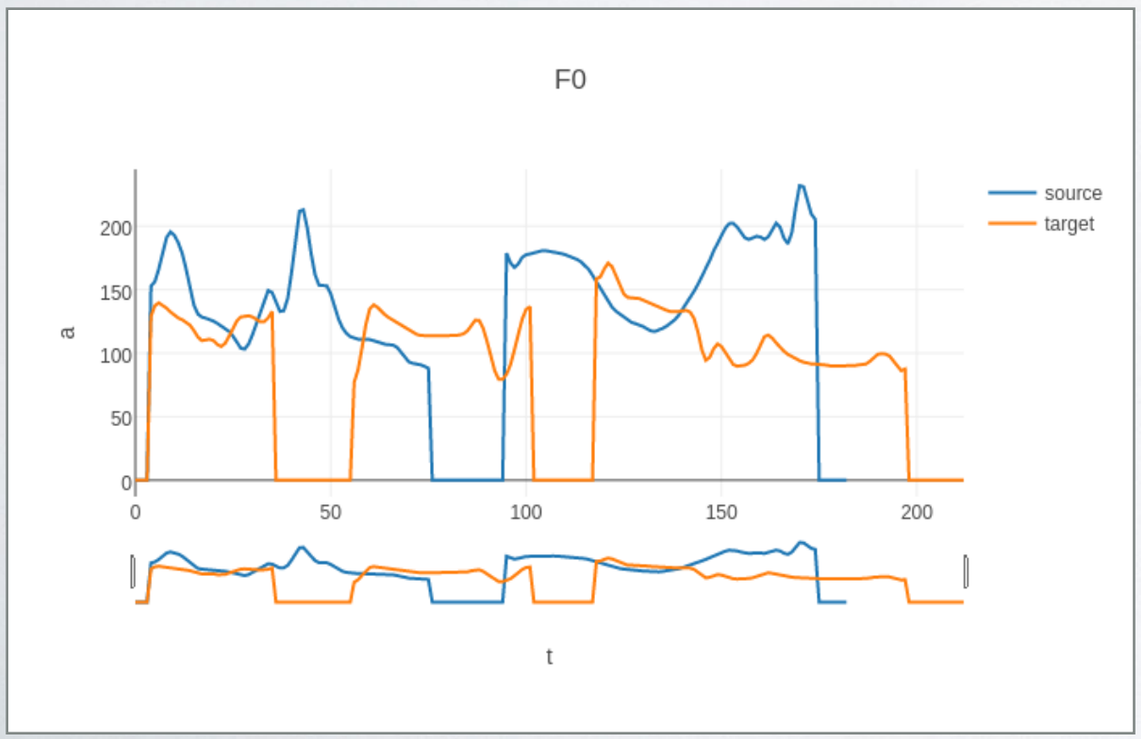


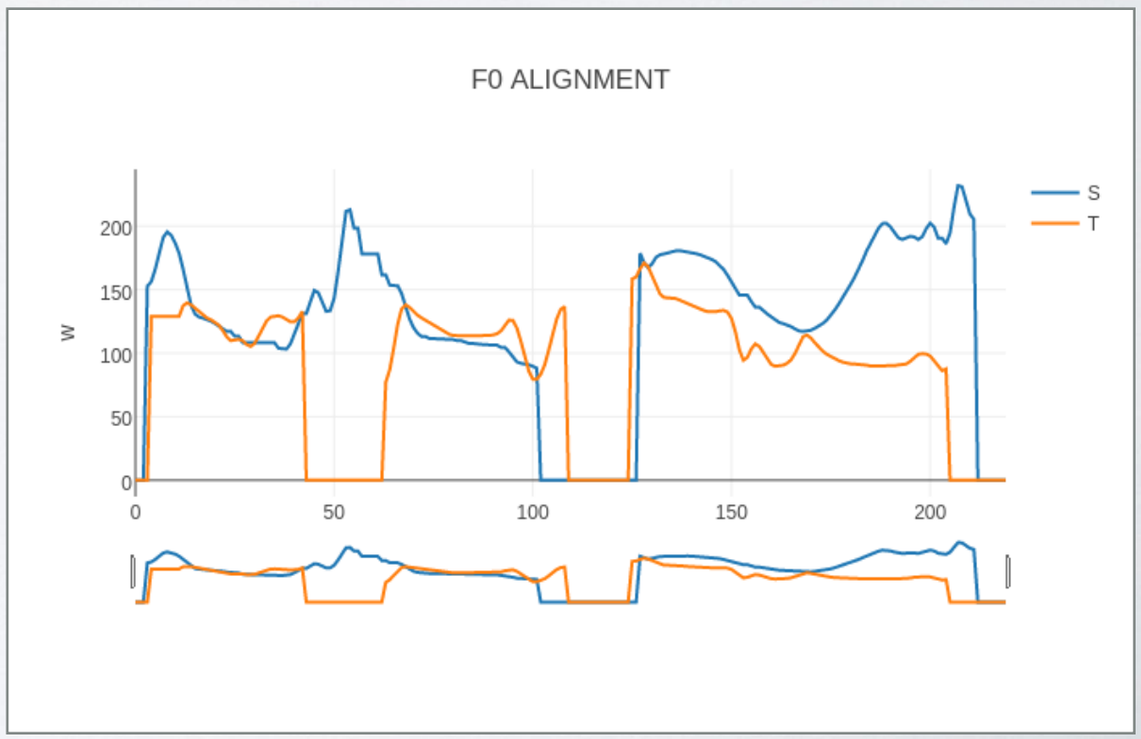


WAVEFORM ALIGNMENT



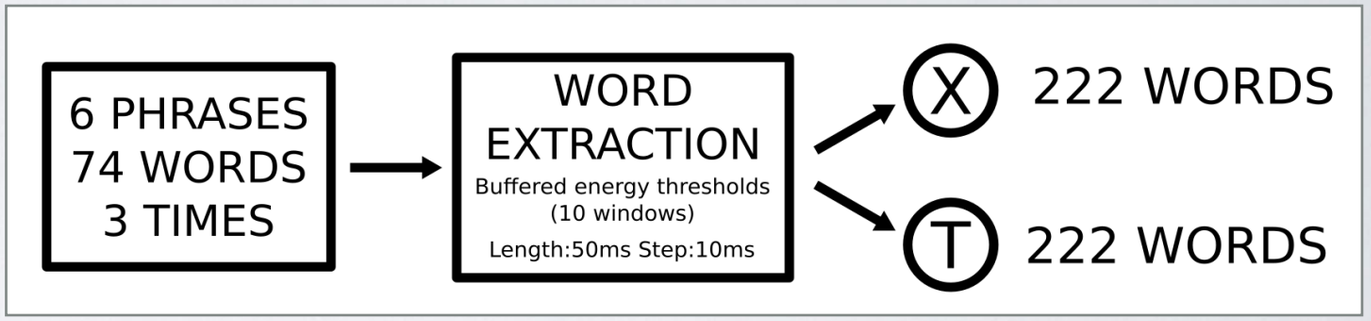


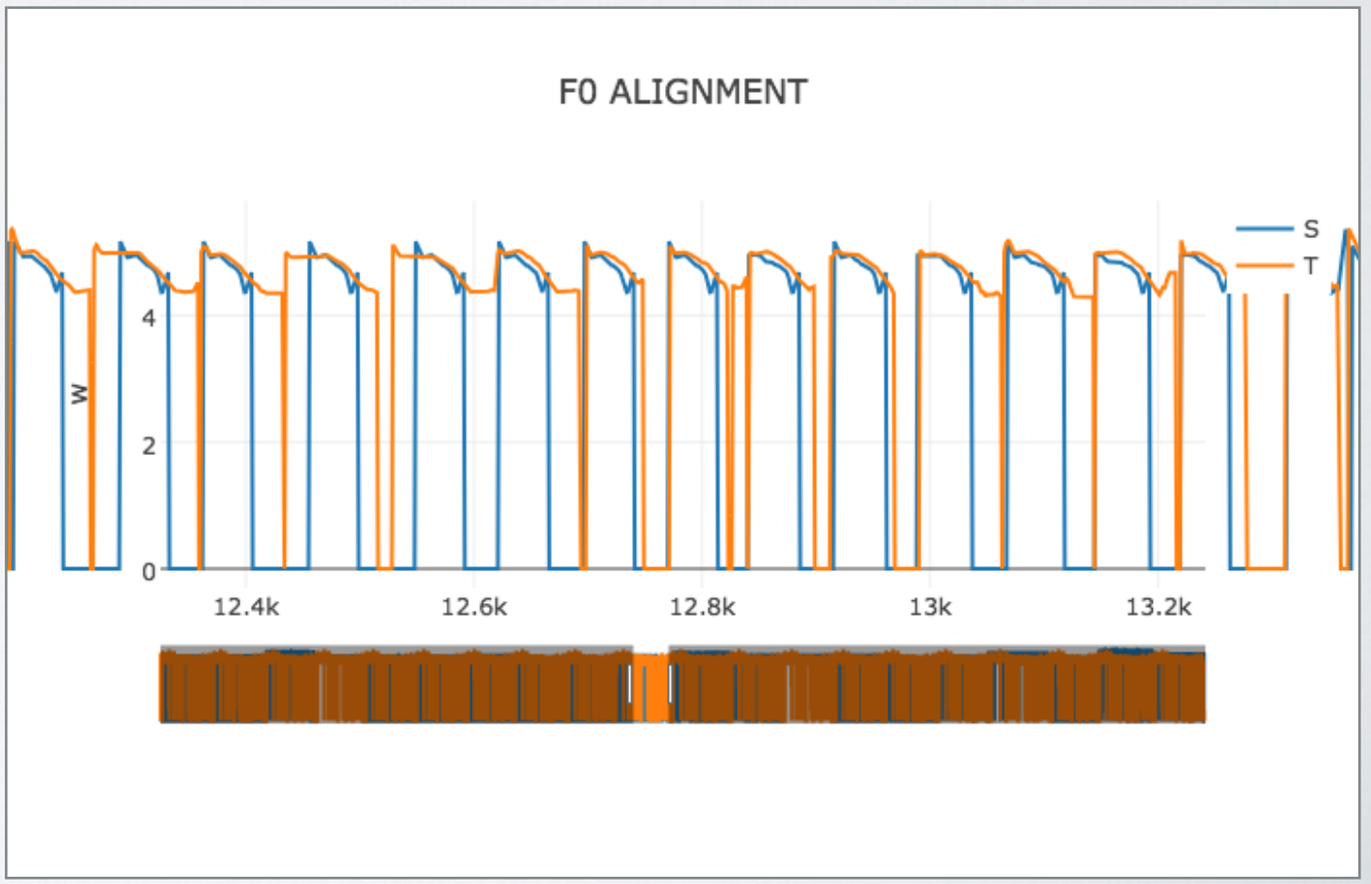




MORE DATASET PREPROCESSING

- Match all individual words
- Audio vectors of 61 Features
- UV enforcing on output
- 95% train set, 5% validation
- Random shuffle

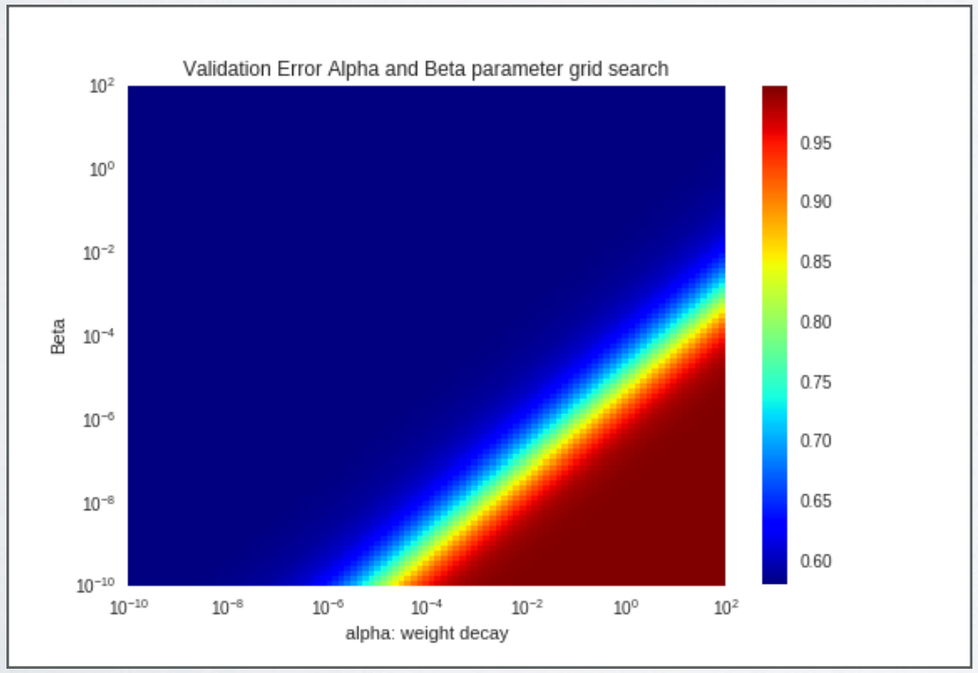


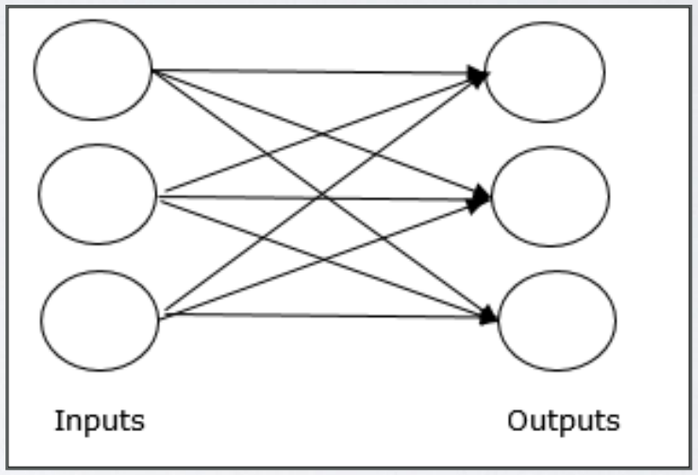


REGRESSION: DENSE LINEAR

$$
\begin{aligned}
p(\mathbf{w}|\alpha,\beta,\mathcal{D}) &= \frac{p(\mathcal{D}|\mathbf{w},\beta)p(\mathbf{w}|\alpha)}{p(\mathcal{D}|\alpha,\beta)} \\
&\propto \left(\sqrt{\frac{\beta}{2\pi}}\right)^N \exp\left(-\frac{\beta}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^\top \mathbf{x}_n)^2\right)\left(\sqrt{\frac{\alpha}{2\pi}}\right)^d \exp\left(-\frac{\alpha}{2}||\mathbf{w}||^2\right)
\end{aligned}
$$

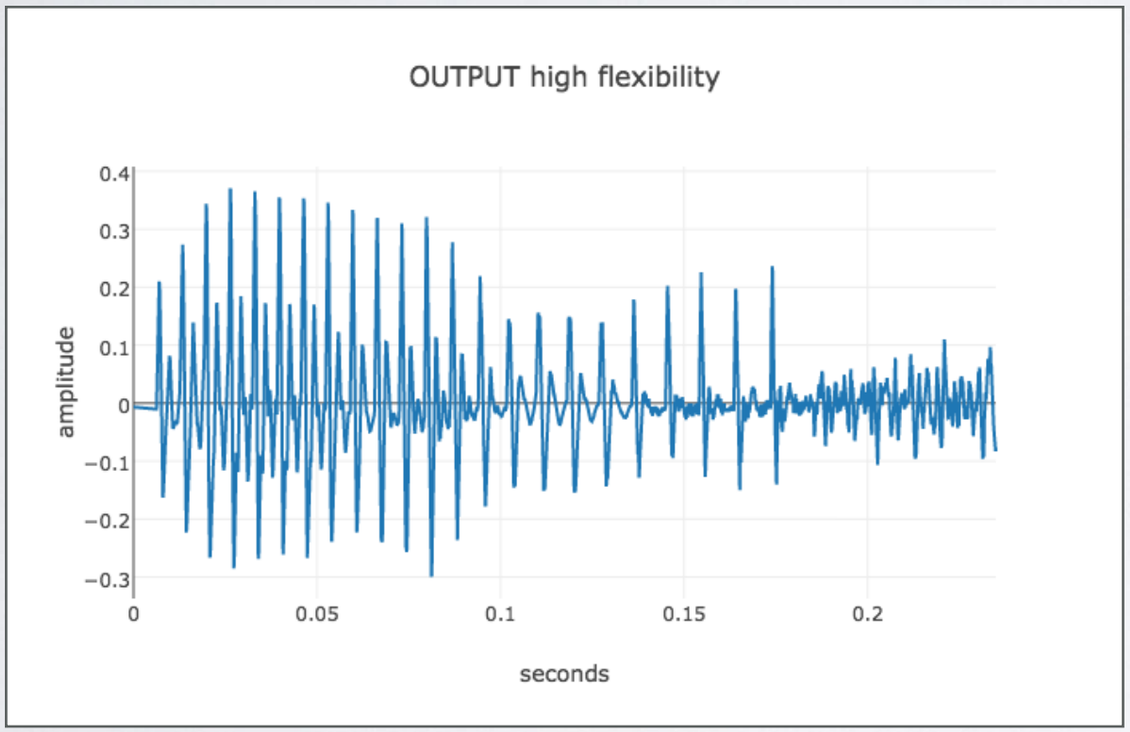
$$
\begin{aligned}
\boldsymbol{\mu}_p &\equiv \left(\alpha\mathbf{I} + \beta\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top\right)^{-1} \beta\sum_{n=1}^{N}\mathbf{x}_n t_n \\
\boldsymbol{\Sigma}_p &\equiv \left(\alpha\mathbf{I} + \beta\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top\right)^{-1} .
\end{aligned}
$$

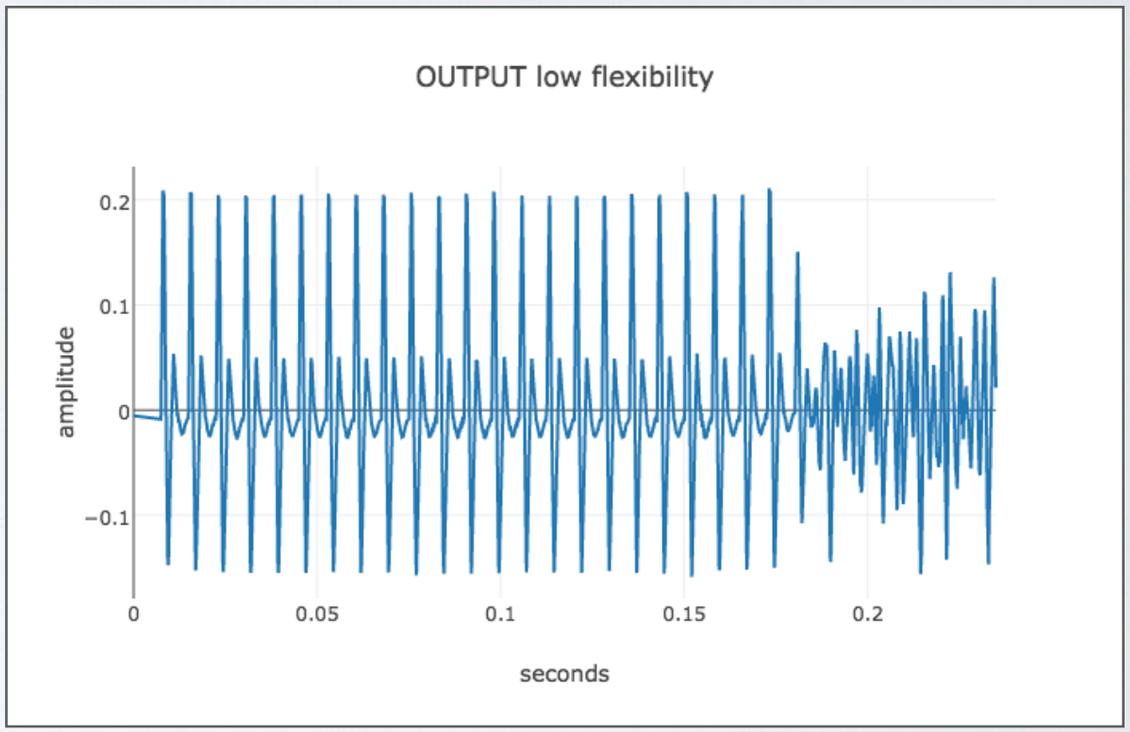




- t = (wx + e) ~ N(y, β^-1)
- w ~ N(0, α^-1)
- Works well for single words
- Need a more flexible model

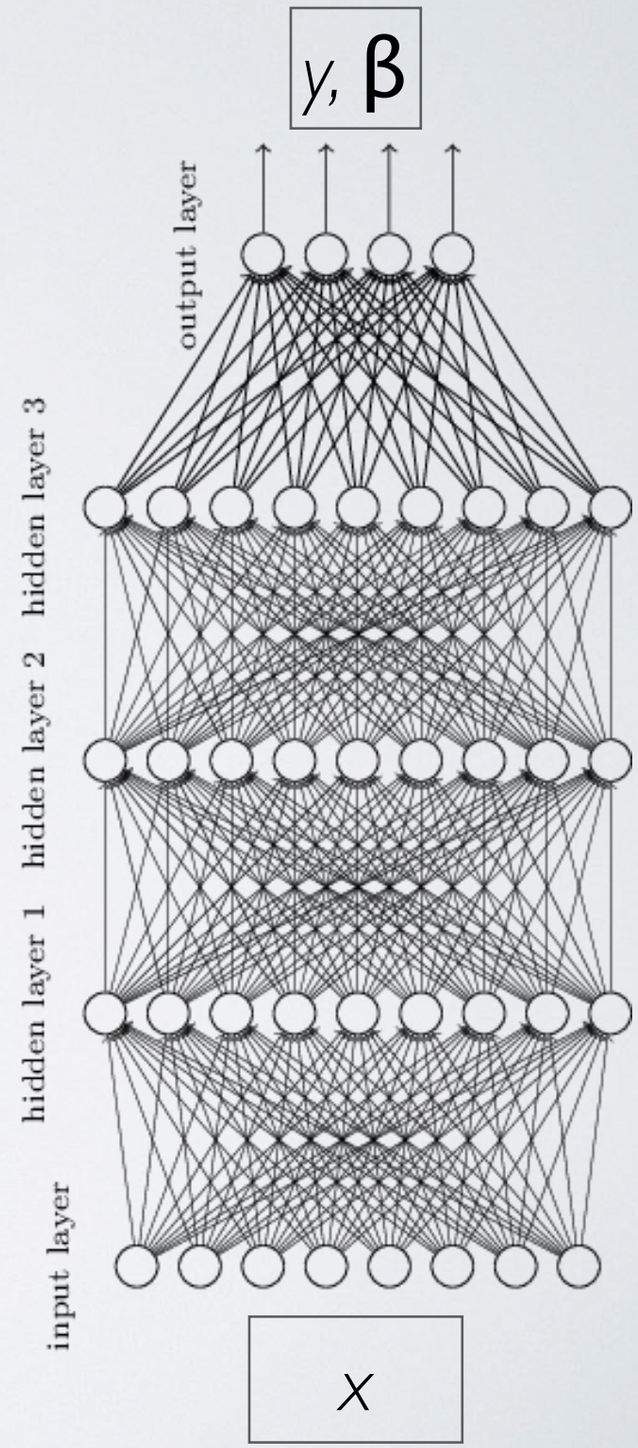
DENSE LINEAR: ONE WORD



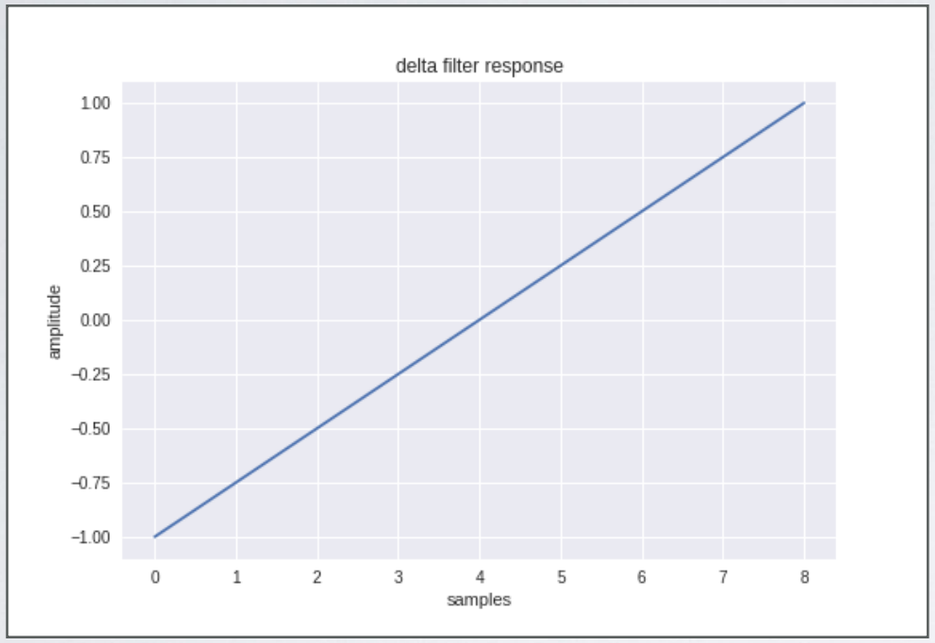


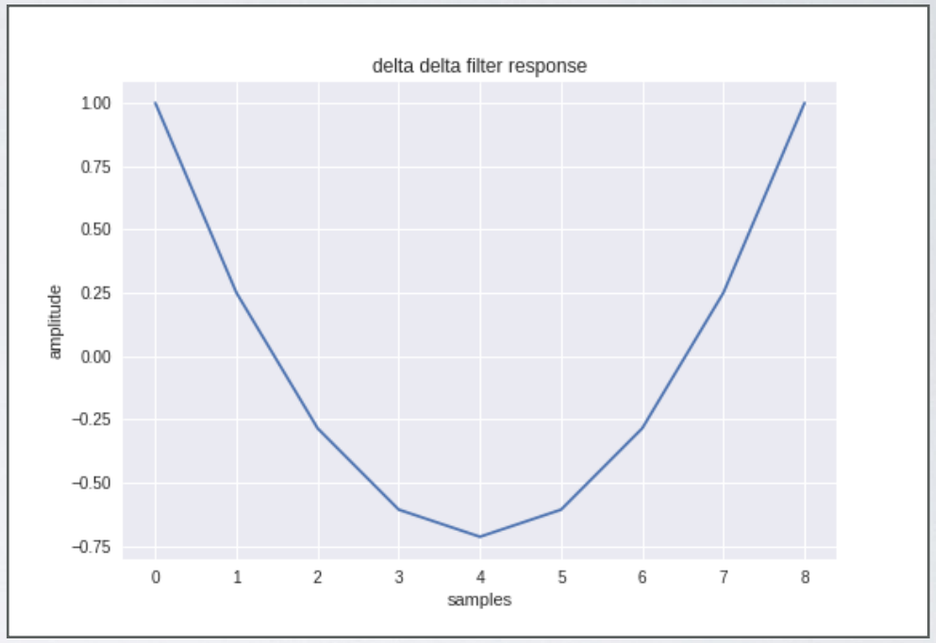
REGRESSION: NN

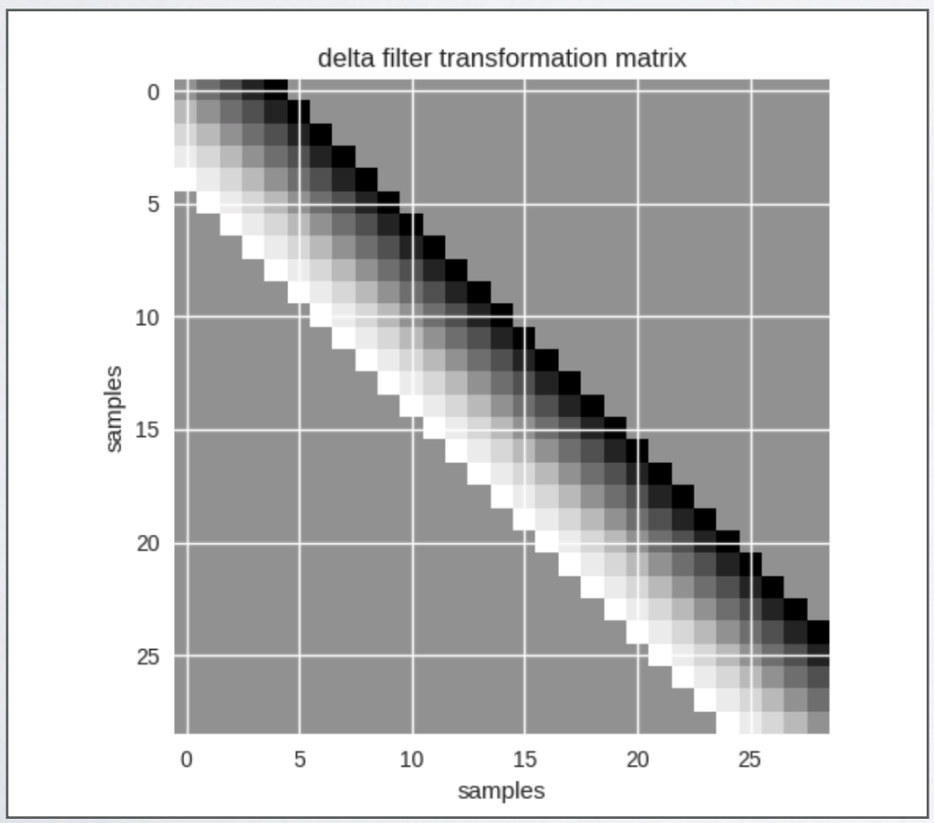
- 4 layer FC network, 0.94M params
- ReLU activation
- 3 batch norm layers
- ADAM optimizer
- Loss = -log $N(y; \mu=t, \beta^{-1})$

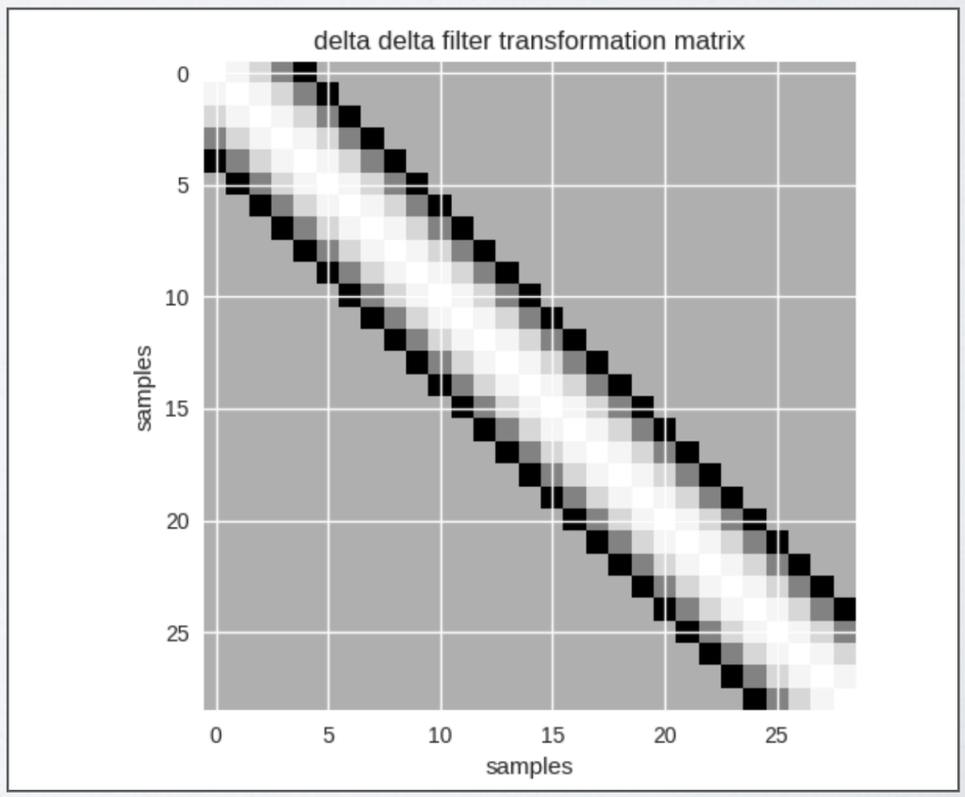


DELTA FEATURES



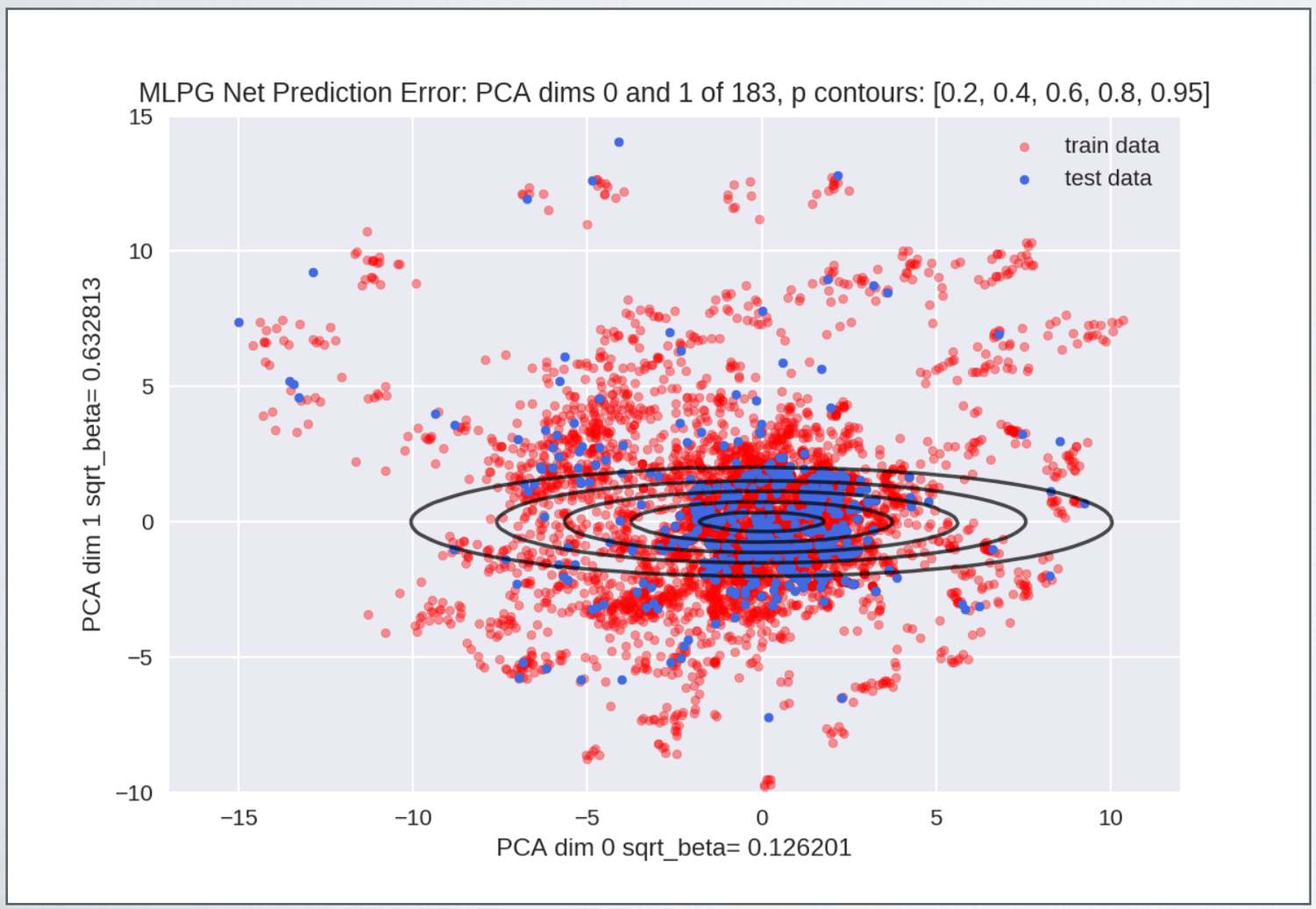


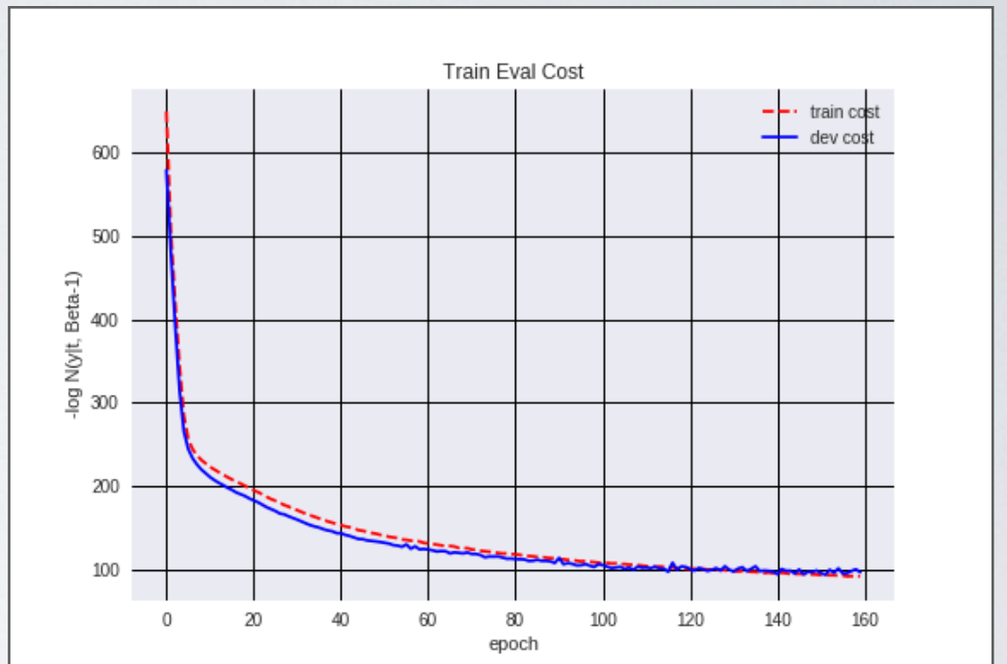


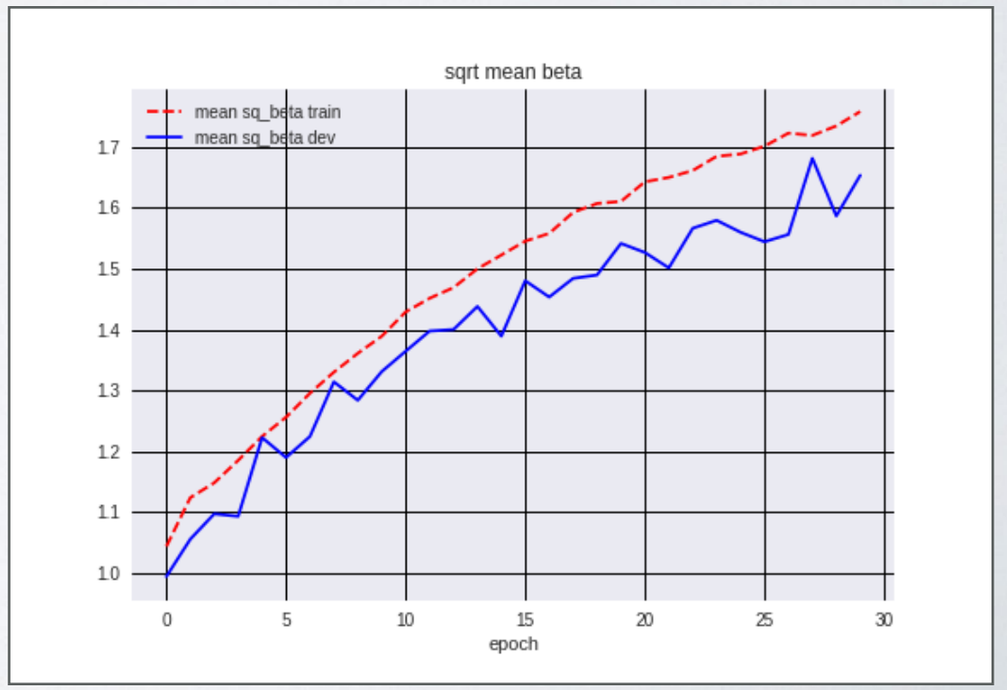


REGRESSION: μ, β

- N(y; μt(y|x), βt(y|x))
- Growing β: Overconfidence, Muffling







MLPG: SMOOTHING

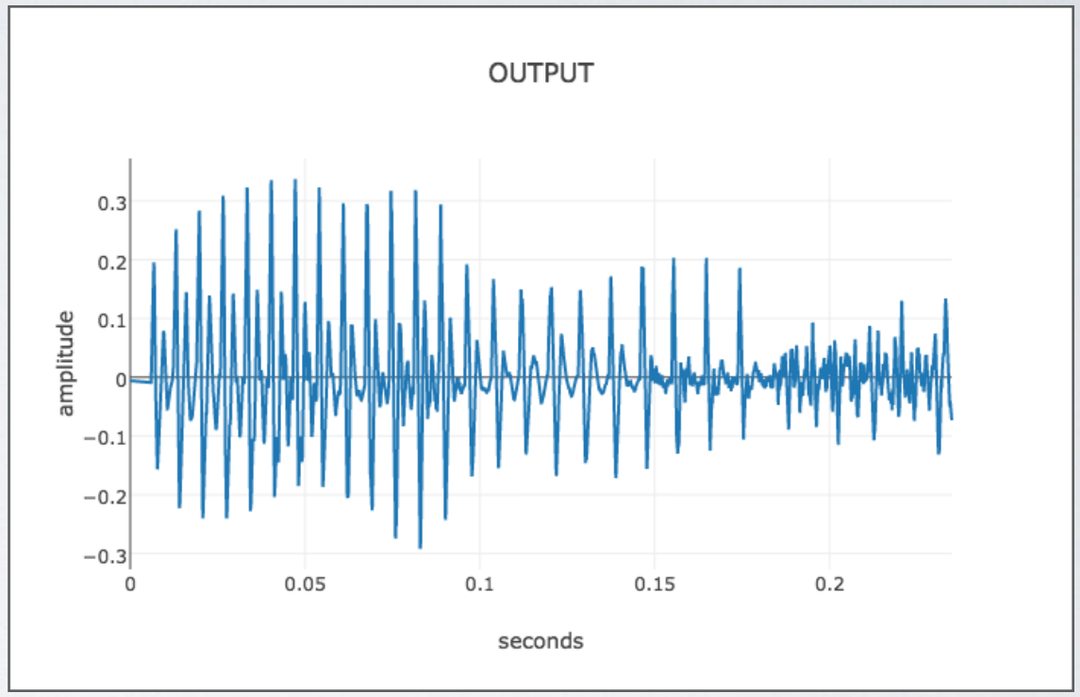
$$\boldsymbol{Y}_t = \left[\boldsymbol{y}_t^\top, \Delta\boldsymbol{y}_t^\top\right]^\top$$

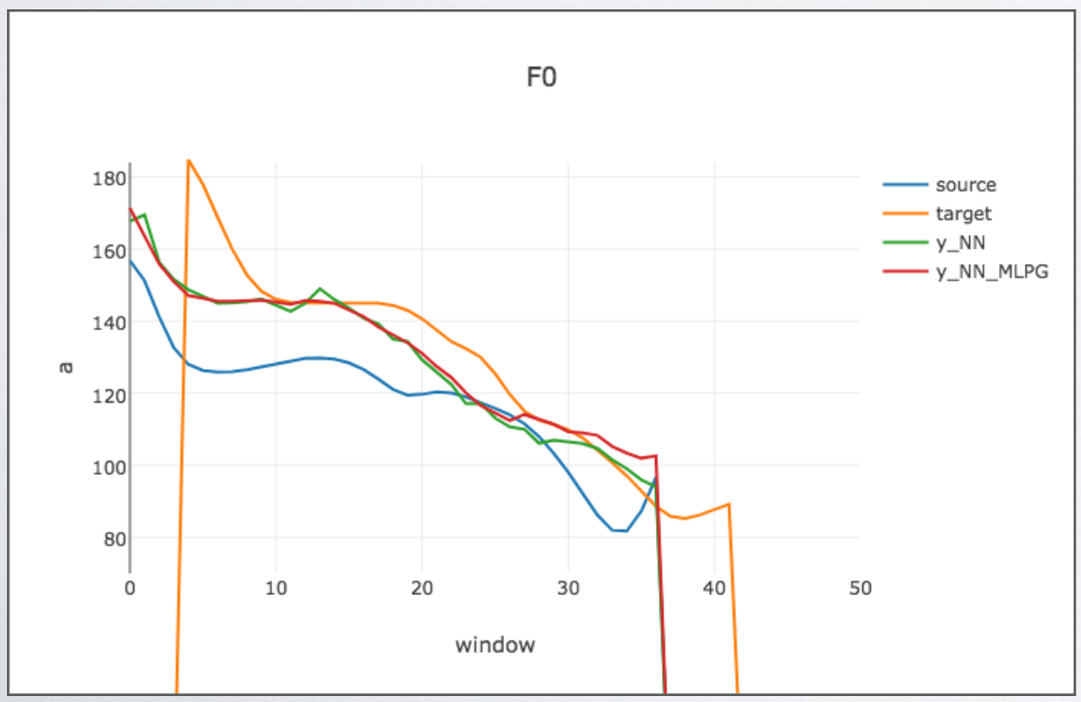
$$\boldsymbol{X}_t = \left[\boldsymbol{x}_t^\top, \Delta\boldsymbol{x}_t^\top\right]^\top$$

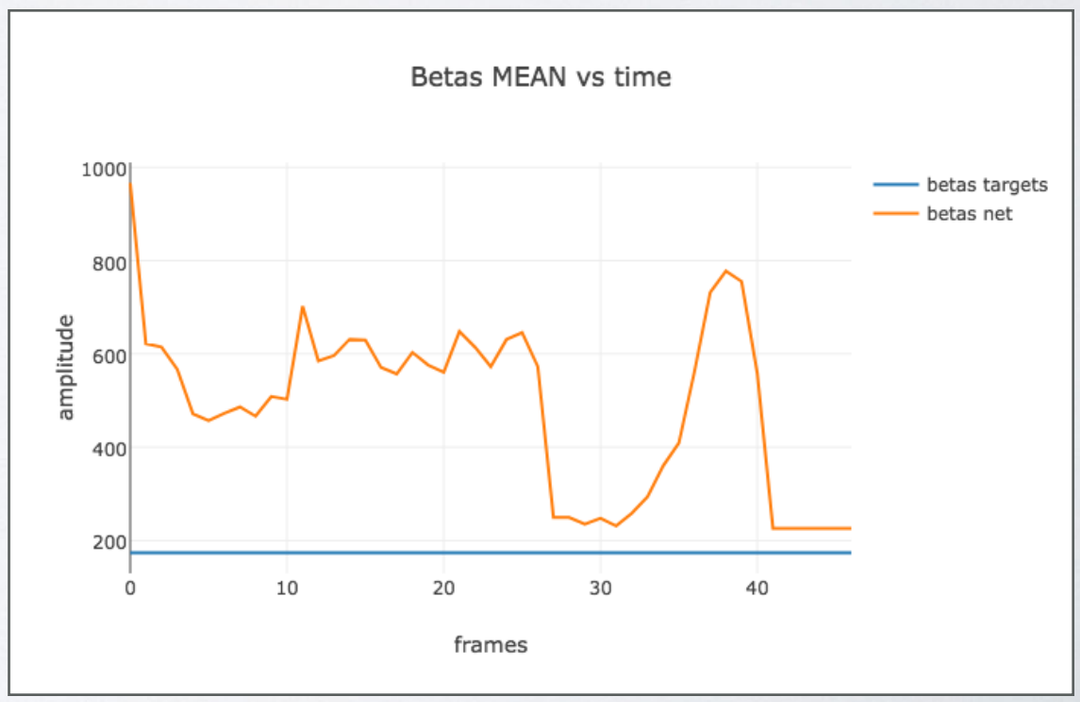
$$\hat{\mathbf{y}} = \arg\max P(\mathbf{Y} \mid \mathbf{X}, \lambda^{(\mathbf{Y}|\mathbf{X})}), \text{ s.t. } \mathbf{Y} = \mathbf{M}\mathbf{y}$$

$$\hat{\mathbf{y}} = (\mathbf{M}^\mathrm{T}\mathbf{U}^{(\mathbf{Y}|\mathbf{X})}\mathbf{M})^{-1}\mathbf{M}^\mathrm{T}\mathbf{U}^{(\mathbf{Y}|\mathbf{X})}\mathbf{E}^{(\mathbf{Y}|\mathbf{X})}$$

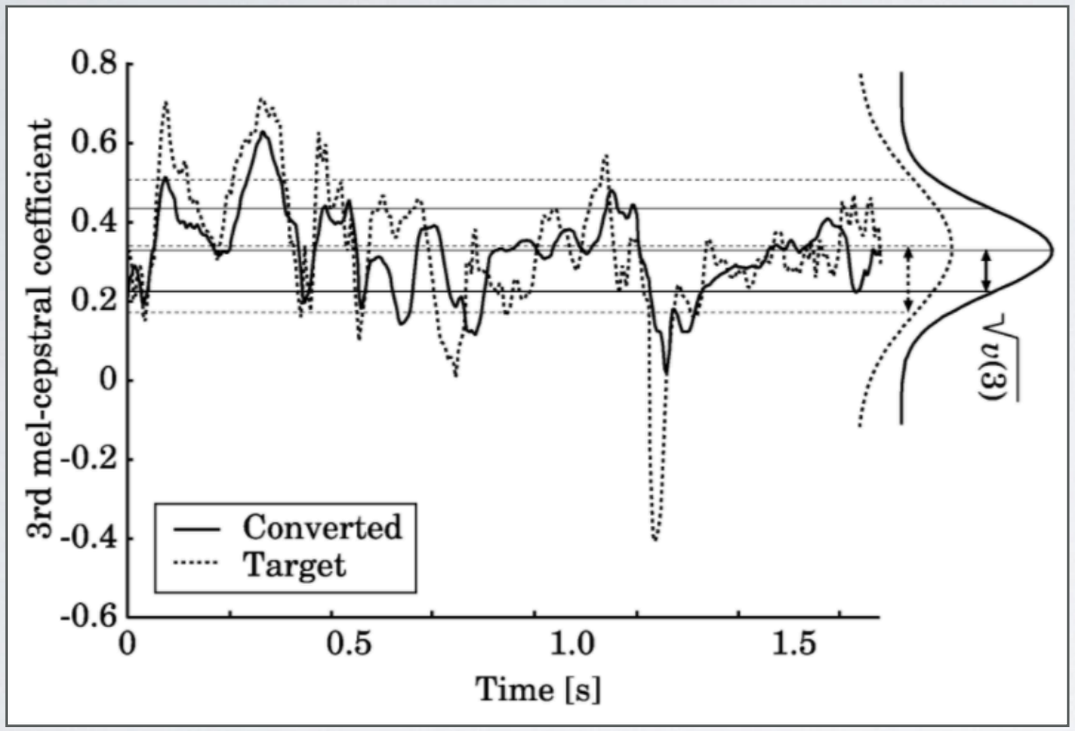
$$\mathbf{E}^{(\mathbf{Y}|\mathbf{X})} = [(\boldsymbol{\mu}_1^{(\mathbf{Y}|\mathbf{X})})^\mathrm{T}, \cdots, (\boldsymbol{\mu}_t^{(\mathbf{Y}|\mathbf{X})})^\mathrm{T}, \cdots, (\boldsymbol{\mu}_T^{(\mathbf{Y}|\mathbf{X})})^\mathrm{T}]^\mathrm{T},$$
$$\mathbf{U}^{(\mathbf{Y}|\mathbf{X})} = \mathrm{daig}[\boldsymbol{\Lambda}_1^{(\mathbf{Y}|\mathbf{X})}, \cdots, \boldsymbol{\Lambda}_t^{(\mathbf{Y}|\mathbf{X})}, \cdots, \boldsymbol{\Lambda}_T^{(\mathbf{Y}|\mathbf{X})}].$$

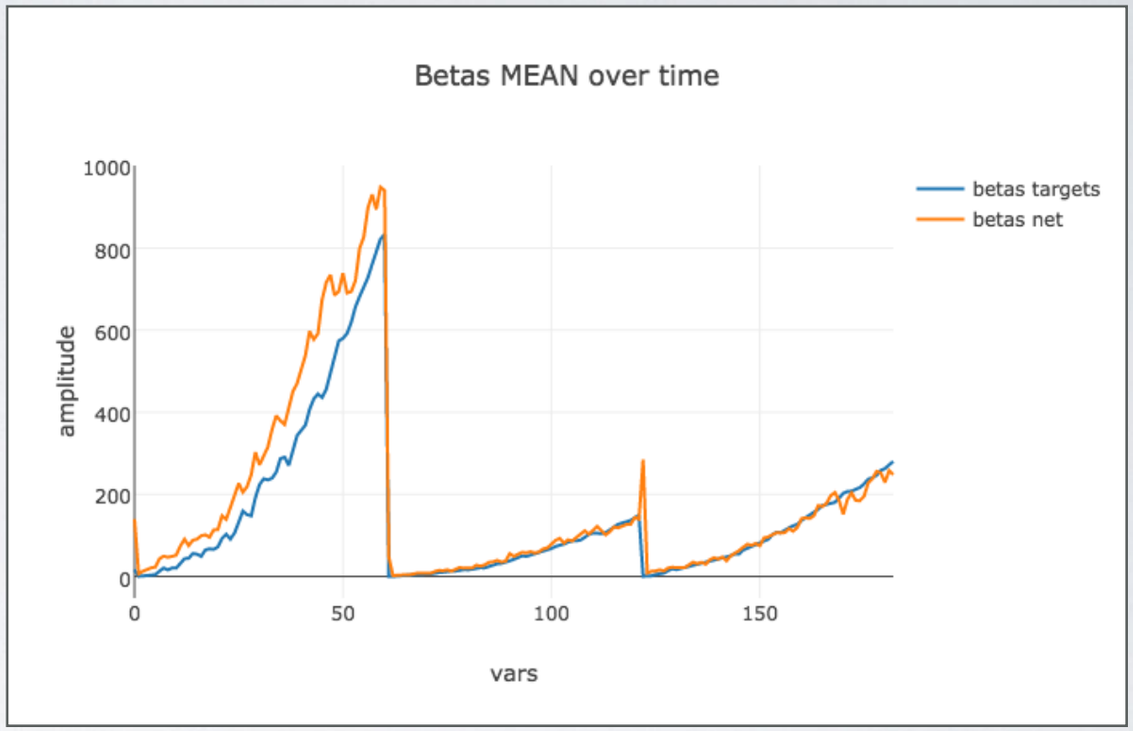






MLPG + GV





$$L_{(MLGV)} = \omega \cdot \log P(\mathbf{Y} \mid \mathbf{X}, \lambda^{(\mathbf{Y}|\mathbf{X})}) + \log P(v(\mathbf{y}) \mid \lambda^{(v)}),$$

$$P(v(\mathbf{y}) \mid \lambda^{(v)}) = N(v(\mathbf{y}); \boldsymbol{\mu}_v, \boldsymbol{\Sigma}^{(vv)}).$$

- v is a parameter's variance in an utterance.
- mu_v, covar_v are mean and covariance of parameters' variance over whole dataset.
- Enforce Global Variance: Eliminate Muffling. (Need large dataset)

LIMITED HARDWARE

- MLPG -> Large diagonal MTX mults
- Bandmat library!
- Real time with Delta feat but without MLPG

```
javi@raspberrypi:~/tiger-costume $ python test_run_mlpg.py

Data:
  x_test: 300 vectors of audio of dim: 61

Net:
    Total params: 0.94M
Reading model_saves/theta_best_mlpg.dat

  restoring epoch: 29, lr: 0.000200
net input_dim: 183
net output_dim: 366

Net loaded,
 Evaluating:
/home/javi/tiger-costume/src/regression_utils.py:46: UserWarnin
  v = Variable(v, volatile=volatile)
output feature shape before mlpg: (300, 183)
sq_Betas feature shape: (300, 183)
net done: time: 0.339912 seconds

out features: (300, 61)
mlpg done: time: 10.851585 seconds
```

DEMO +

Sources + References: → https://github.com/JavierAntoran/tiger-costume