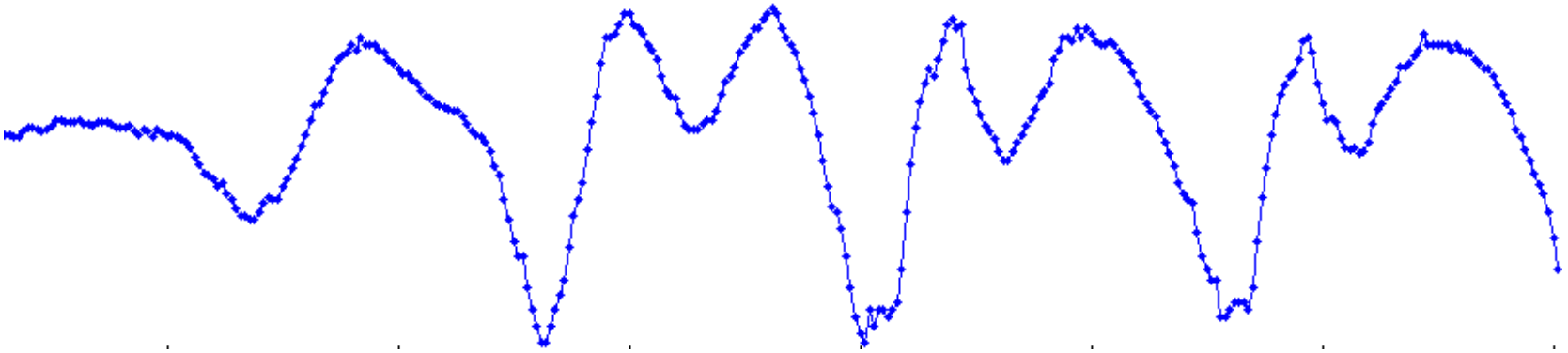
An implementation of WaveNet

March 2023

Vassilis Tsiaras
Computer Science Department
University of Crete

Introduction

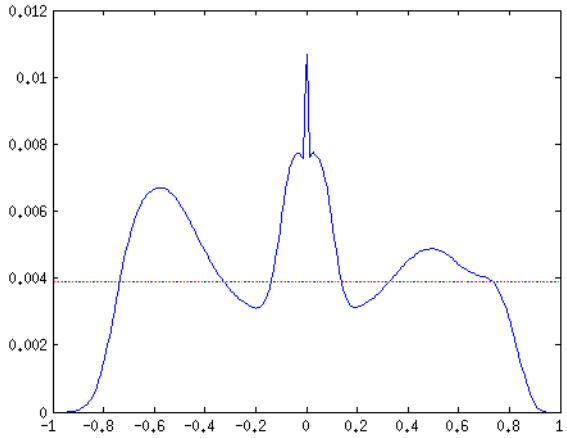
- In September 2016, DeepMind presented WaveNet.
- Wavenet is an ARM with dilated, causal, 1d convolutions.
- Wavenet out-performed the best TTS systems (parametric and concatenative) in Mean Opinion Scores (MOS).
- Before wavenet, all Statistical Parametric Speech Synthesis (SPSS) methods modelled parameters of speech, such as cepstra, F0, etc.
- WaveNet revolutionized our approach to SPSS by directly modelling the raw waveform of the audio signal.



- DeepMind published a paper about WaveNet, but it did not reveal all the details of the network.
- Here an implementation of WaveNet is presented, which fills some of the missing details.

Probability of speech segments

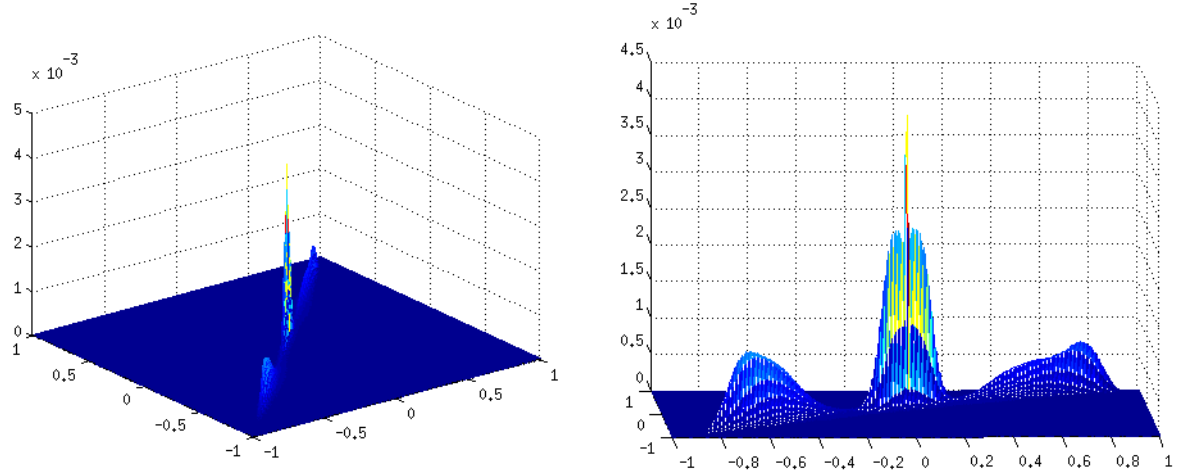
- Let Ω_T denote the set of all possible sequences of length T over $\{0,1,\ldots,d-1\}$.
- Let $P:\Omega_T \rightarrow [0,1]$ be a probability distribution which achieves higher values for speech sequences than for other sequences.
- Knowledge of the distribution $P:\Omega_T \rightarrow [0,1]$, allow us to test whether a sequence $x_1 x_2 \cdots x_T$ is speech or not.
- Also, using random sampling methods, it allows us to generate sequences that with high probability look like speech.
- The estimation of P is easy for very small values of T (e.g., $T = 1,2$).



Estimation of $P(x_1)$

Green: Random samples

Blue: Speech samples. Value 0 corresponds to silence



Different views of $P(x_1 x_2)$, which was estimated from speech samples from the arctic database.

Probability of speech segments

- The estimation of P for very small values of T is easy but it is not very useful since the interdependence of speech samples, whose time indices differ more than T, is ignored.
- In order to be useful for practical applications, the distribution P should be estimated for large values of T.
- However, the estimation of P becomes very challenging as T grows, due to sparsity of data and to the extremely low values of P.
- In order to robustly estimate P, we take the following actions.

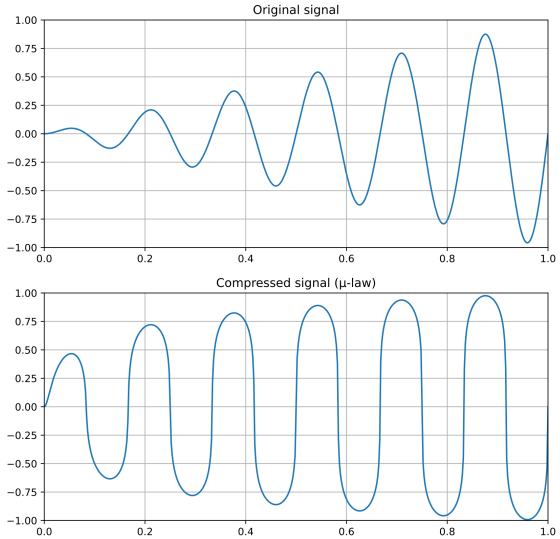
1. The dynamic range of speech is reduced within the interval [-1,1] and then the speech is quantized into a number of bins (usually $d = 256$).
2. Based on the factorization $P(x_1, \ldots, x_t) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1})$, we calculate the conditional probabilities $P(x_t \mid x_1, \ldots, x_{t-1})$ instead of $P(x_1, \ldots, x_t)$.
    - The conditional probability $P(x_t \mid x_1, \ldots, x_{t-1}) = \frac{P(x_1, \ldots, x_t)}{P(x_1, \ldots, x_{t-1})}$ is numerically more manageable than $P(x_1, \ldots, x_t)$.

Dynamic range compression and Quantization

- Raw audio, $y_1 \ldots y_t \ldots y_T$, is first transformed into $x_1 \ldots x_t \ldots x_T$, where $-1 < x_t < 1$, for $t \in \{1, \ldots, T\}$ using an μ-law transformation

$$x_t = sign(y_t) \frac{\ln\left(1 + \mu \left| y_t \right|\right)}{\ln\left(1 + \mu\right)}$$

where $\mu = 255$



- Then x_t is quantized into 256 values.
- Finally, x_t is encoded to one-hot vectors.

- **Toy example:**

$-2.2, \ -1.43, \ -0.77, \ -1.13, \ -0.58, \ -0.43, \ -0.67, \ldots$ ***signal*** ⇒ $-0.7, \ -0.3, \ 0.2, \ -0.1, \ 0.4, \ 0.6, \ 0.3, \ldots$ ***μ-law transformed***

⇒ $0, 1, 2, 1, 2, 3, 2, \ldots$ ***quantized into 4 bins*** ⇒

bin 0	1	0	0	0	0	0	0
bin 1	0	1	0	1	0	0	0
bin 2	0	0	1	0	1	0	1
bin 3	0	0	0	0	0	1	0

one-hot vectors ... ⇒ ***Input to WaveNet***

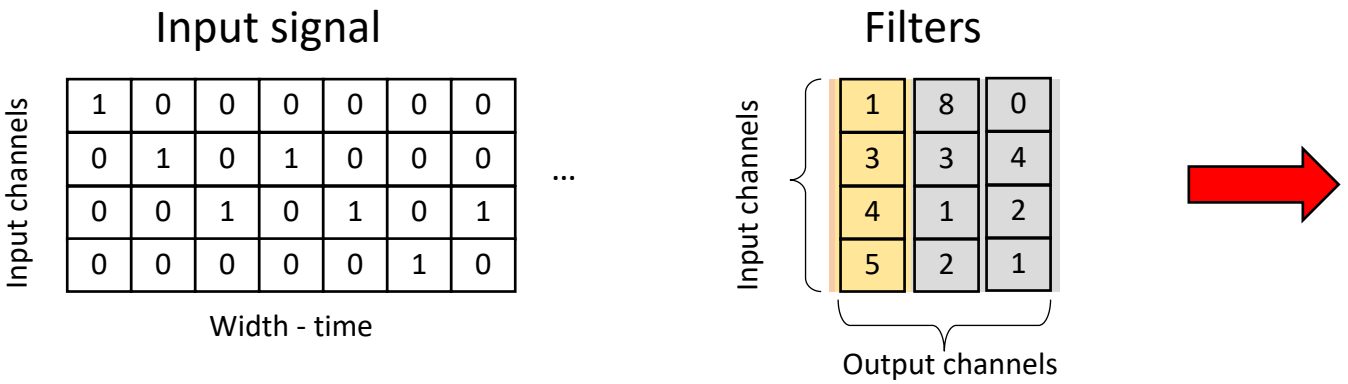
The conditional probability

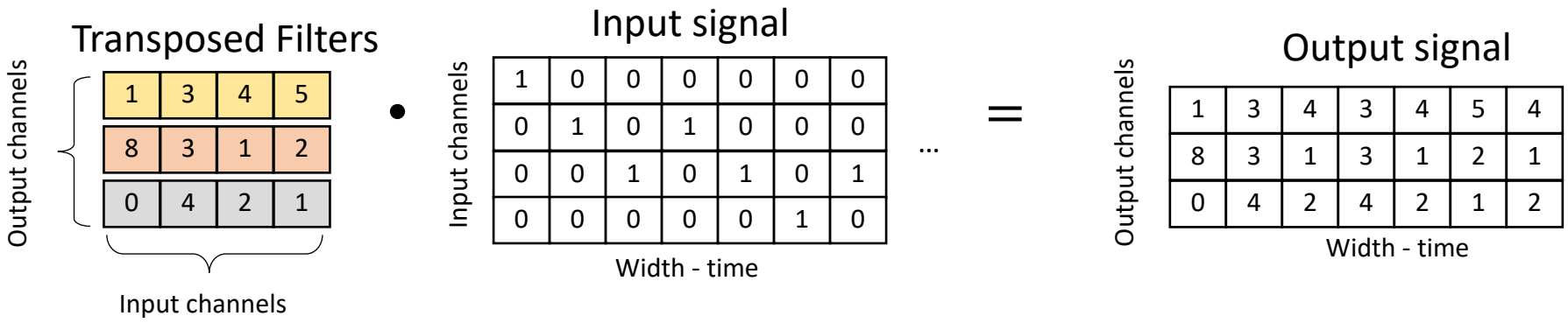
- The conditional probability $P\left(x_t \middle| x_1, \ldots, x_{t-1}\right)$ is modelled with a categorical distribution where x_t falls into one of a number of bins (usually 256).
- The tabular representation of $P\left(x_t \middle| x_1, \ldots, x_{t-1}\right)$ is infeasible, since it requires space proportional to 256^t.
- Instead, function approximation of P is used.
- Well known function approximators are the neural networks.
- The recurrent and the convolutional neural networks model the interdependence of the samples in a sequence and are ideal candidates to represent $P\left(x_t \middle| x_1, \ldots, x_{t-1}\right)$.
- The recurrent neural networks usually work better than the convolutional neural networks but their computation cannot be parallelized across time.
- WaveNet, uses one-dimensional causal convolutional neural networks to represent $P\left(x_t \middle| x_1, \ldots, x_{t-1}\right)$.

WaveNet architecture – 1×1 Convolutions

- 1×1 convolutions are used to change the number of channels. They do not operate in time dimension.
- They can be written as matrix multiplications.
- **Example** of a 1×1 convolution with 4 input channels, and 3 output channels

1x1 convolution

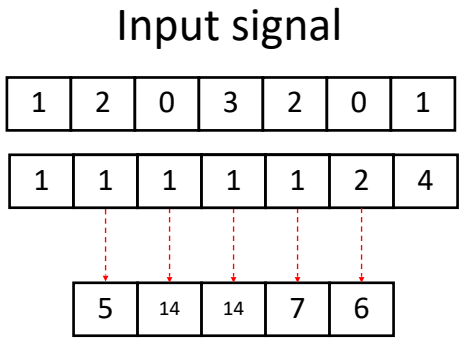


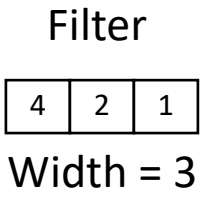


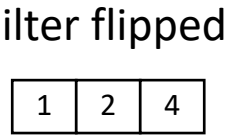
$$out[c_{out}, t] = \sum_{c_{in}=0}^{3} in[c_{in}, t] \bullet filter[c_{out}, c_{in}]$$

Causal convolutions

- Example of a convolution

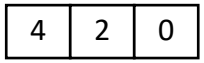




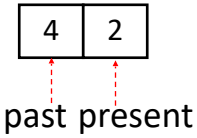


- Many machine learning libraries avoid the filter flipping.
- For simplicity, we will also avoid the filter flipping.
- Causal convolutions do not consider future samples. Therefore all values of the filter kernel that correspond to future samples are zero.

Filter of a causal convolution

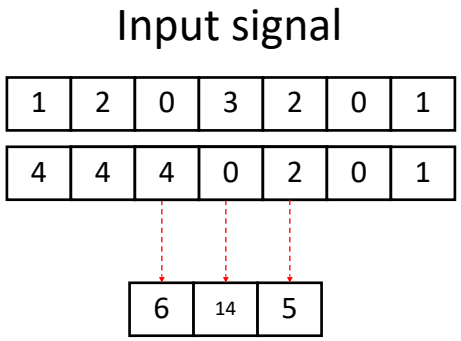


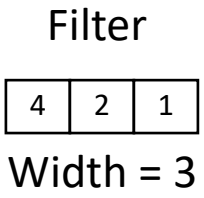
- Filters of width 2 are causal

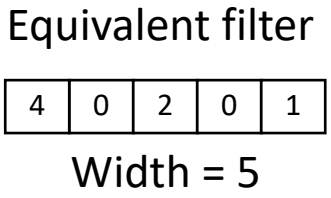


Dilated convolutions

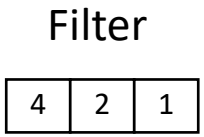
- Example of a dilated convolution, with dilation=2

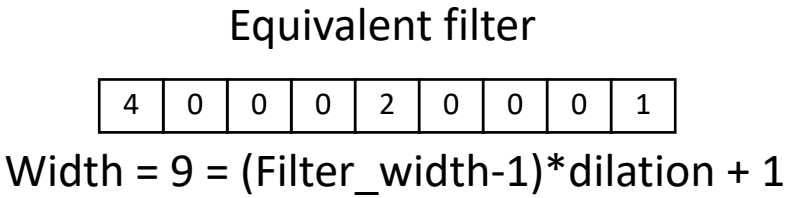






- Equivalent filter of a dilated convolution, with dilation=4

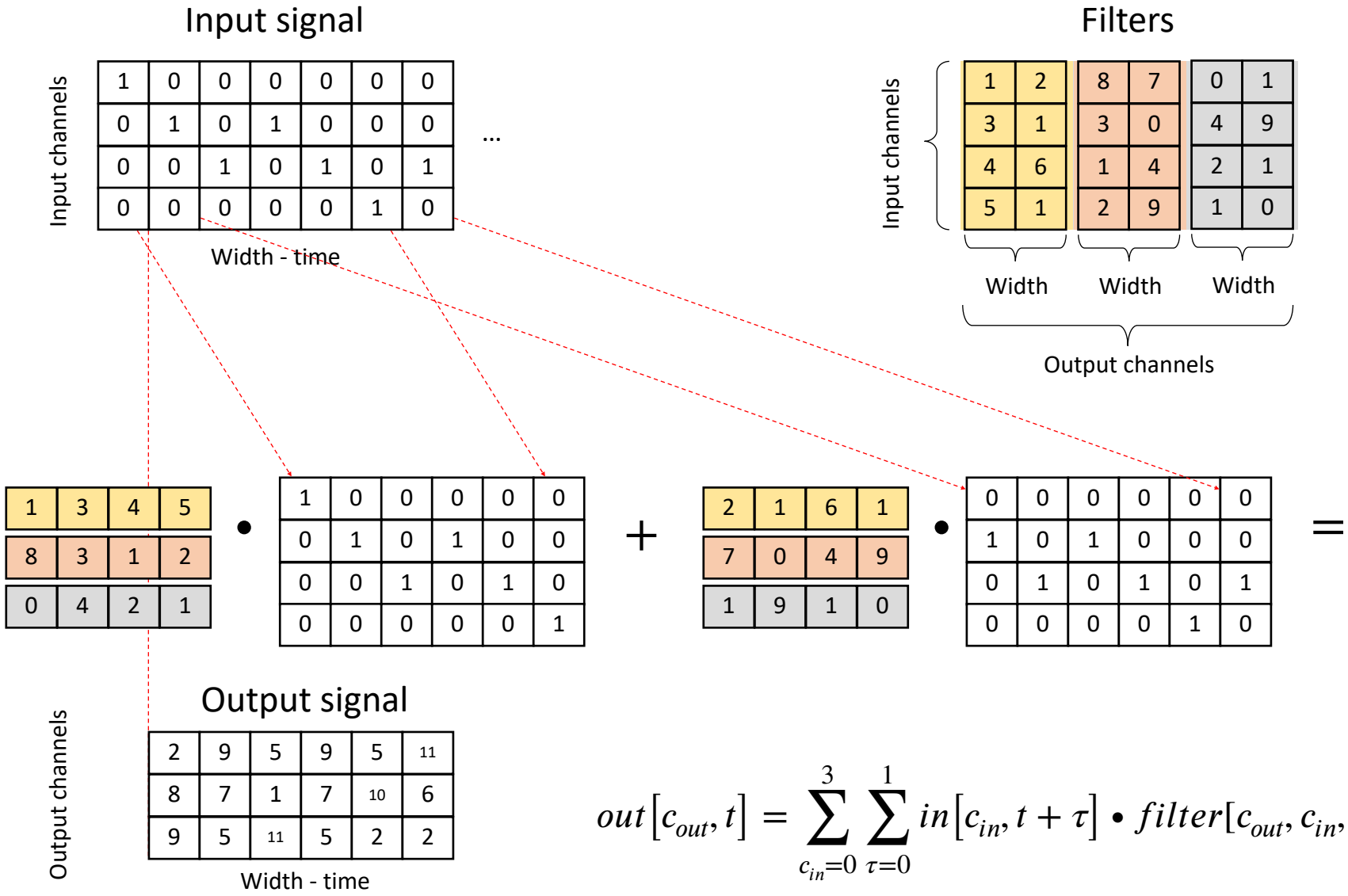




- Dilated convolutions have longer receptive fields.
- Efficient implementations of dilated convolutions do not consider the equivalent filter with the filled zeros.

Causal convolutions - Matrix multiplications

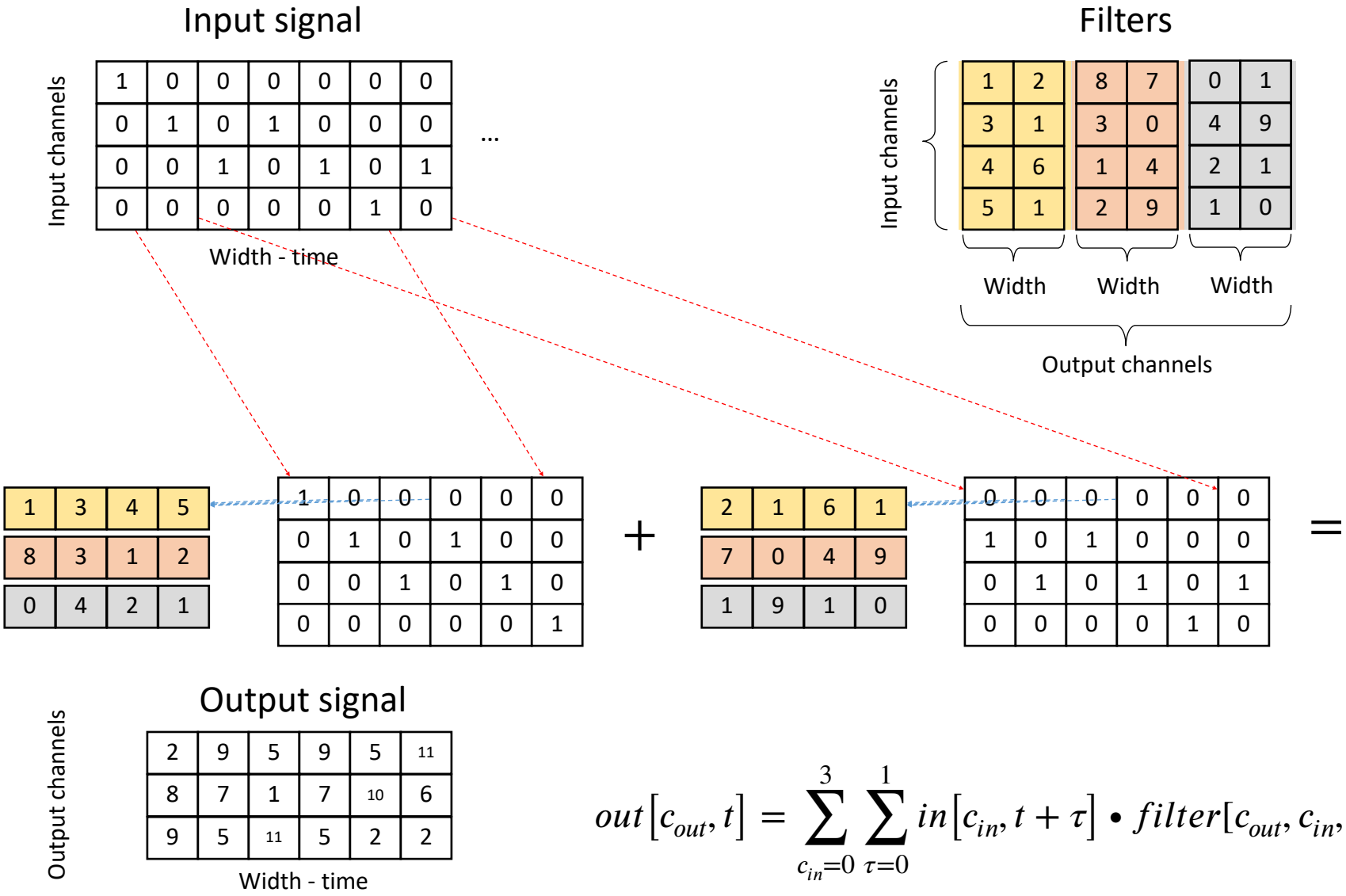
- Example of a **causal** convolution of **width 2**, **4 input channels**, and **3 output channels**



$$out[c_{out}, t] = \sum_{c_{in}=0}^{3} \sum_{\tau=0}^{1} in[c_{in}, t+\tau] \bullet filter[c_{out}, c_{in}, \tau]$$

Causal convolutions - Embedding

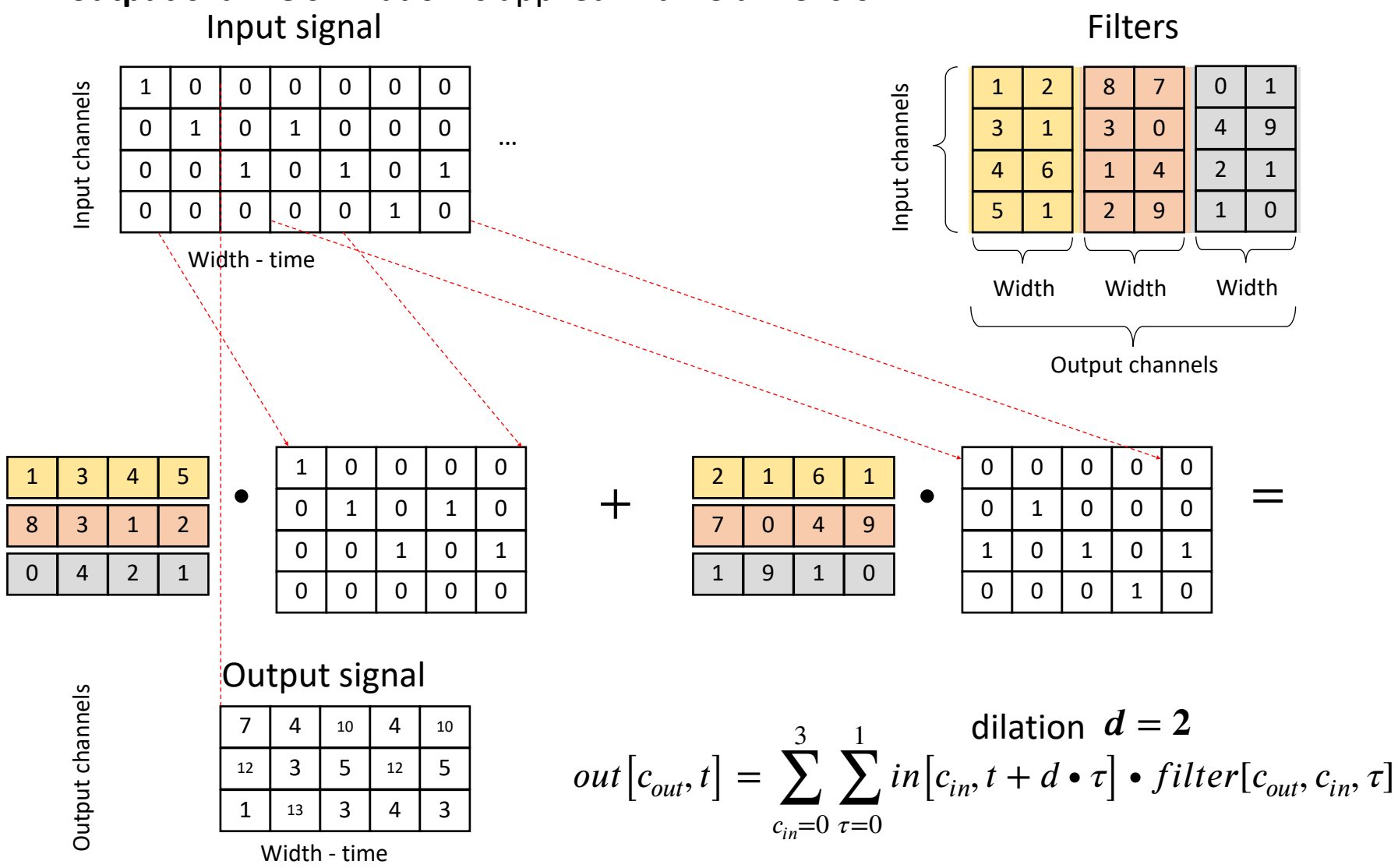
- Example of a **causal** convolution of **width 2**, **4 input channels**, and **3 output channels**



$$out[c_{out}, t] = \sum_{c_{in}=0}^{3} \sum_{\tau=0}^{1} in[c_{in}, t+\tau] \bullet filter[c_{out}, c_{in}, \tau]$$

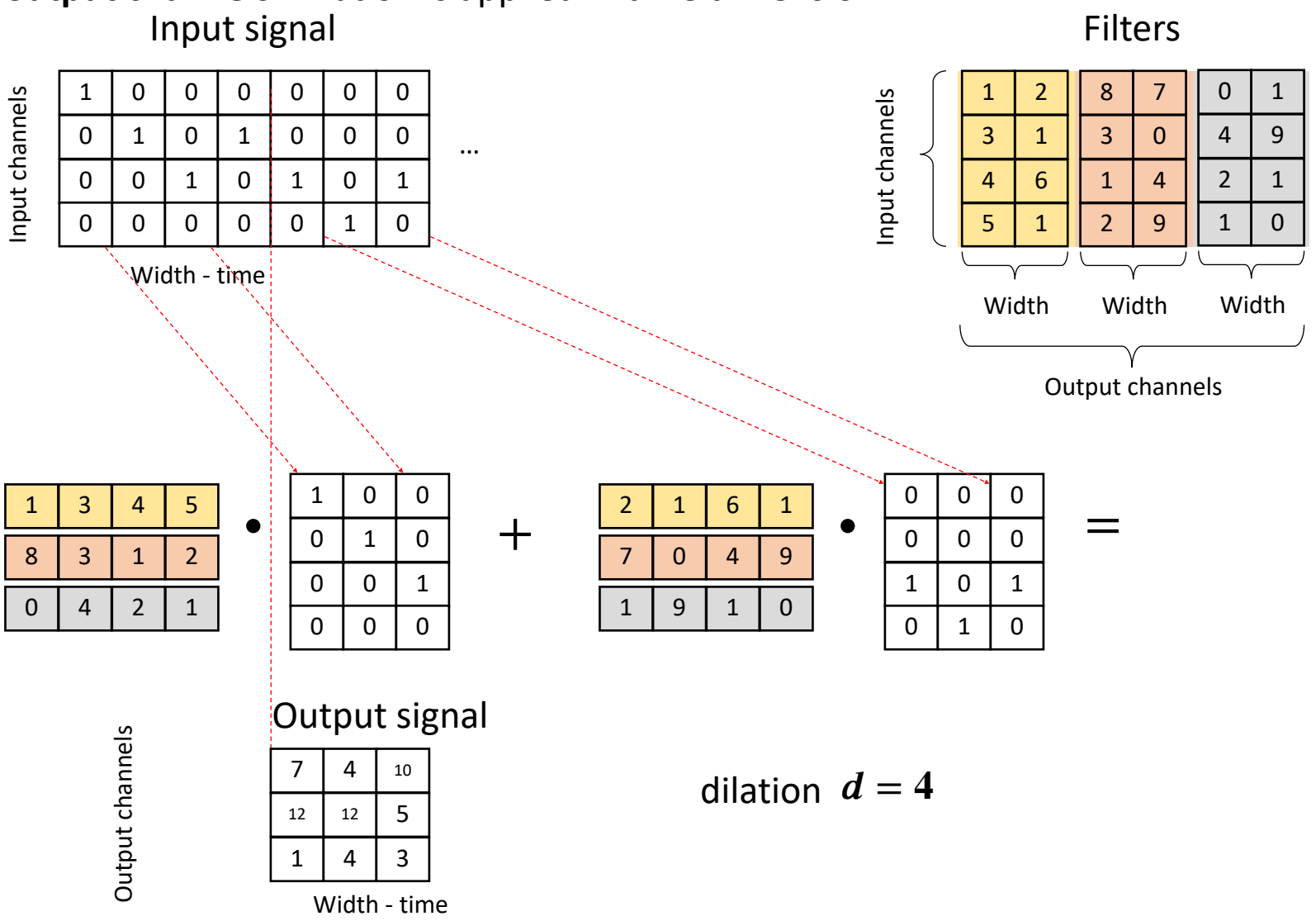
Dilated convolutions – Matrix Multiplications

- Example of a **causal dilated** convolution of **width 2**, **dilation 2**, **4 input channels**, and **3 output channels**. Dilation is applied in time dimension



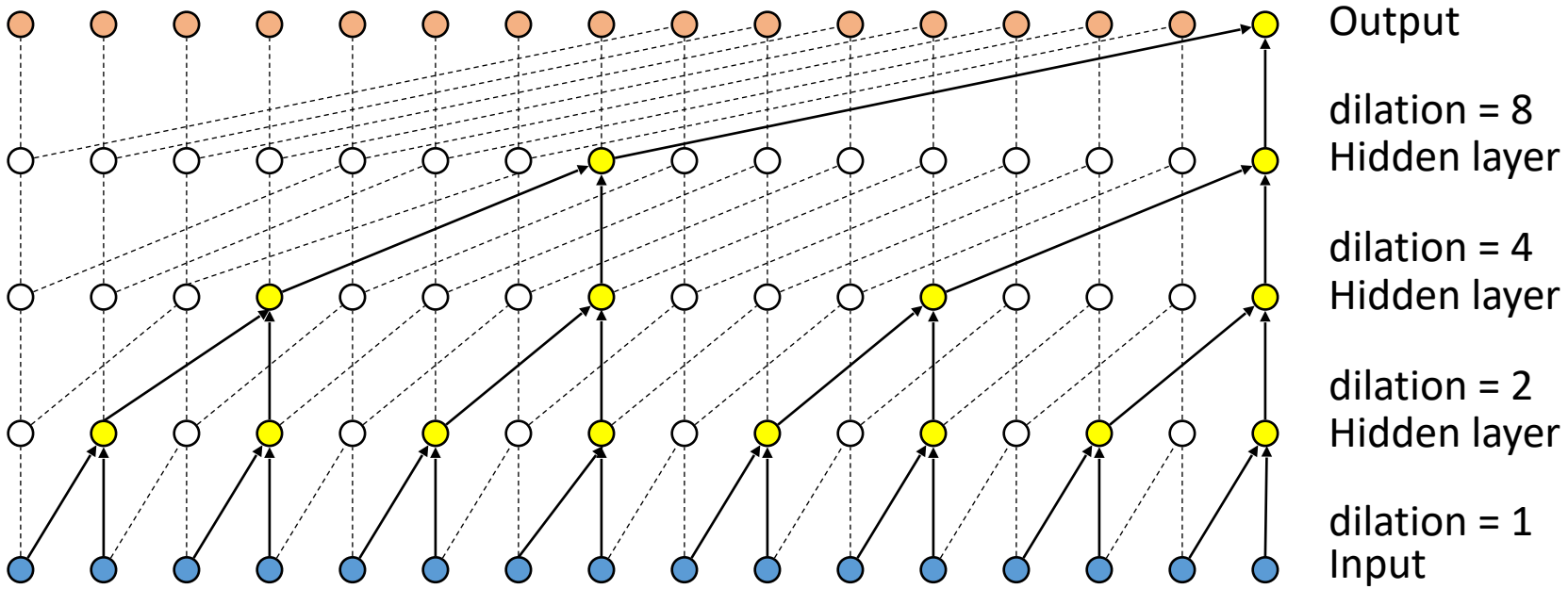
Dilated convolutions – Matrix Multiplications

- Example of a **causal dilated** convolution of **width 2**, **dilation 4**, **4 input channels**, and **3 output channels**. Dilation is applied in time dimension



WaveNet architecture – Dilated convolutions

- WaveNet models the conditional probability distribution $p(x_t \mid x_1, \ldots, x_{t-1})$ with a stack of dilated causal convolutions.

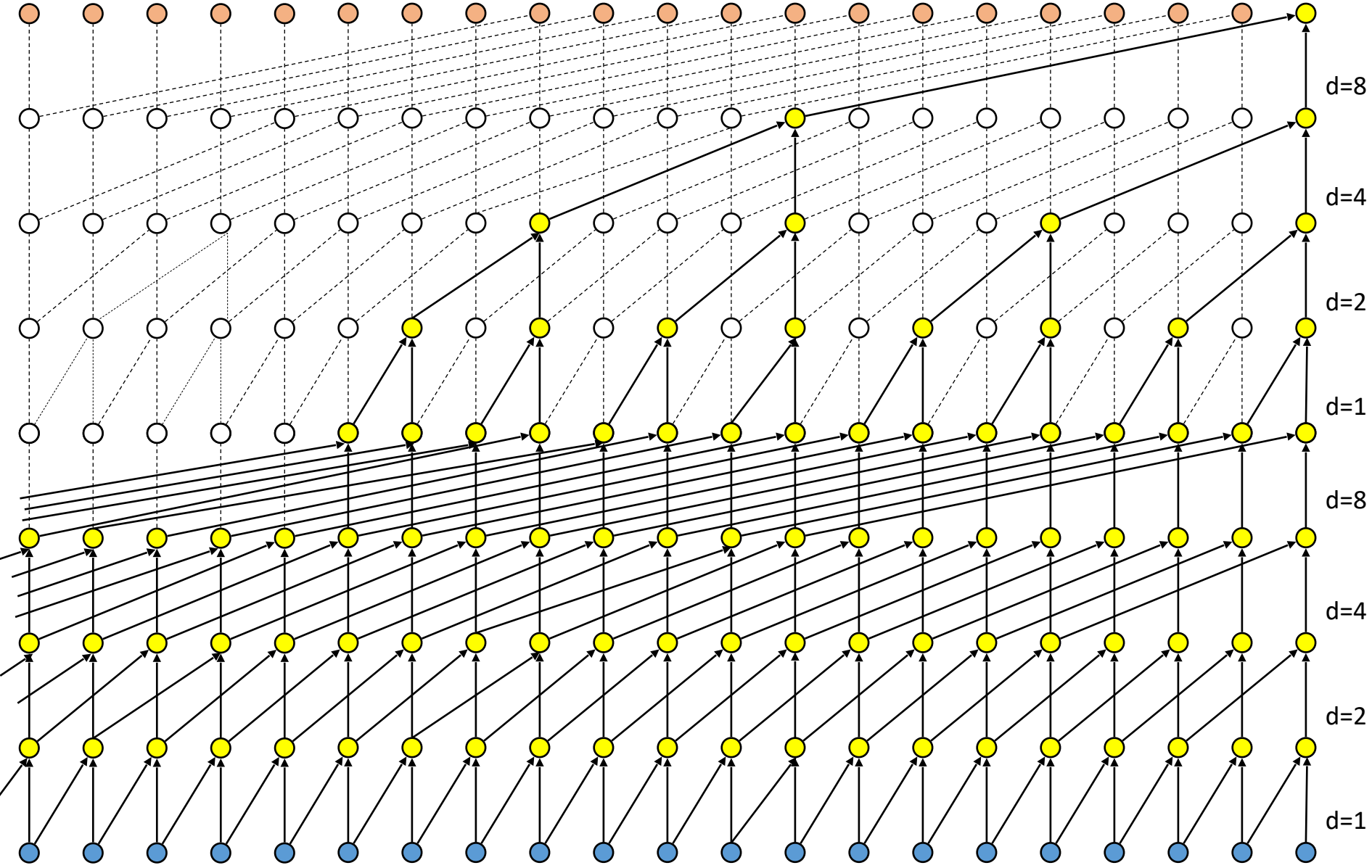


Visualization of a stack of dilated causal convolutional layers

- Stacked dilated convolutions enable very large **receptive fields** with just a few layers.
- The receptive field of the above example is (8+4+2+1) + 1 = 16
- In WaveNet, the dilation is doubled for every layer up to a certain point and then repeated: 1, 2, 4, ..., 512, 1, 2, 4, ..., 512, 1, 2, 4, ..., 512, 1, 2, 4, ..., 512, 1, 2, 4, ..., 512

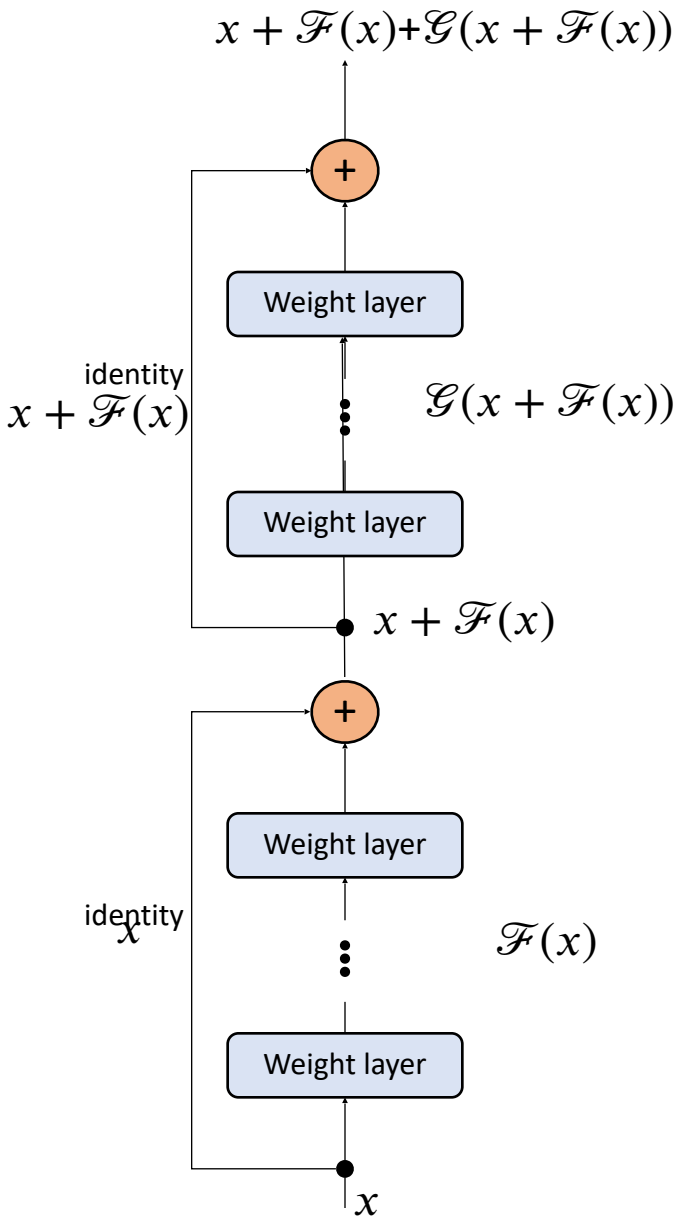
WaveNet architecture – Dilated convolutions

- Example with dilations 1,2,4,8,1,2,4,8



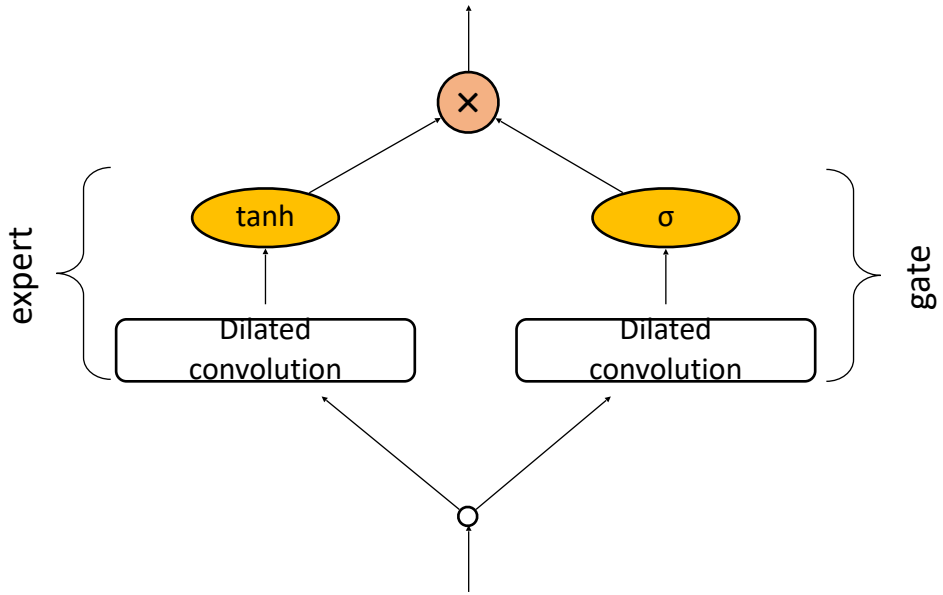
WaveNet architecture – Residual connections

- In order to train a WaveNet with more than 30 layers, residual connections are used.
- Residual networks were developed by researchers from Microsoft Research.
- They reformulated the mapping function, $x \rightarrow f(x)$, between layers from $f(x) = \mathcal{F}(x)$ to $f(x) = x + \mathcal{F}(x)$.
- The residual networks have identity mappings, x, as skip connections and inter-block activations $\mathcal{F}(x)$.
- Benefits
  - The residual $\mathcal{F}(x)$ can be more easily learned by the optimization algorithms.
  - The forward and backward signals can be directly propagated from one block to any other block.
  - The vanishing gradient problem is not a concern.



WaveNet architecture – Experts & Gates

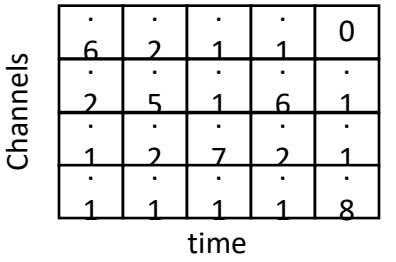
- WaveNet uses gated networks.
- For each output channel an expert is defined.
  - Experts may specialize in different parts of the input space
- The contribution of each expert is controlled by a corresponding *gate* network.
- The components of the output vector are mixed in higher layers, creating mixture of experts.



WaveNet architecture – Output

- WaveNet assigns to an input vector x_t a probability distribution using the softmax function.

$$h(z)_j = \frac{e^{z_j}}{\sum_{c=1}^{256} e^{z_c}}, \quad j = 1, \ldots, 256$$



WaveNet output:
probabilities from softmax

- **Example** with receptive field = 3

Speech: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$

Input: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9$

Output: $p_4, p_5, p_6, p_7, p_8, p_9, p_{10}$

target: $x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$

where $p_4 = P(x_4 \mid x_1, x_2, x_3), \; p_5 = P(x_5 \mid x_2, x_3, x_4), \; \ldots.$

WaveNet architecture – Loss function

- **Example** with receptive field = 3 and filter width = 2 and $T = 10$ speech samples

  Speech: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$

  Input: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9$ # drop the last sample

  Output: $p_4, p_5, p_6, p_7, p_8, p_9, p_{10}$ # After shift right "filter_width"-1 samples

  target: $x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$ # crop the first "receptive_field" samples

  where $p_4 = P\left(x_4 \middle| x_1, x_2, x_3\right),\ p_5 = P\left(x_5 \middle| x_2, x_3, x_4\right),\ \ldots.$

- During training the estimation of the probability distribution $p_t = P\left(x_t \middle| x_{t-1}, \ldots, x_{t-R}\right)$ is compared with the one-hot encoding of x_t.
- The difference between these two probability distributions is measured with the mean (across time) cross entropy:

$$H\left(x_4, \ldots, x_T, p_4, \ldots, p_T\right) = -\frac{1}{T-3}\sum_{t=4}^{T} x_t^{\intercal} \bullet \log\left(p_t\right) = -\frac{1}{T-3}\sum_{t=4}^{T}\sum_{c=1}^{256} x_t(c)\log(p_t(c))$$

WaveNet – Audio generation

- After training, the network is sampled to generate synthetic utterances.
- At each step during sampling a value is drawn from the probability distribution computed by the network.
- This value is then fed back into the input and a new prediction for the next step is made.
- **Example** with receptive field 3 and 4 quantization channels

Input: x_1, x_2, x_3

Output: $p_4 = Wavenet(x_1, x_2, x_3) = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$ Probability distribution over the symbols 0,1,2,3

sample: $x_4 = 1$

Input: x_2, x_3, x_4

Output: $p_5 = Wavenet(x_2, x_3, x_4) = \begin{bmatrix} 0.7 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$

sample: $x_5 = 0$

WaveNet – Audio generation

- Sampling methods
  - **Direct sampling**: Sample randomly from $P(x)$
  - **Temperature sampling**: Sample randomly from a distribution adjusted by a temperature θ, $\tilde{P}_\theta(x) = \frac{1}{Z} P(x)^{1/\theta}$, where Z is a normalizing constant.
  - **Mode**: Take the most likely sample, $\underset{x}{\operatorname{argmax}} P(x)$
  - **Mean**: Take the mean of the distribution, $E_p[x]$
  - **Top k**: Sample from an adjusted distribution that only permits the top k samples

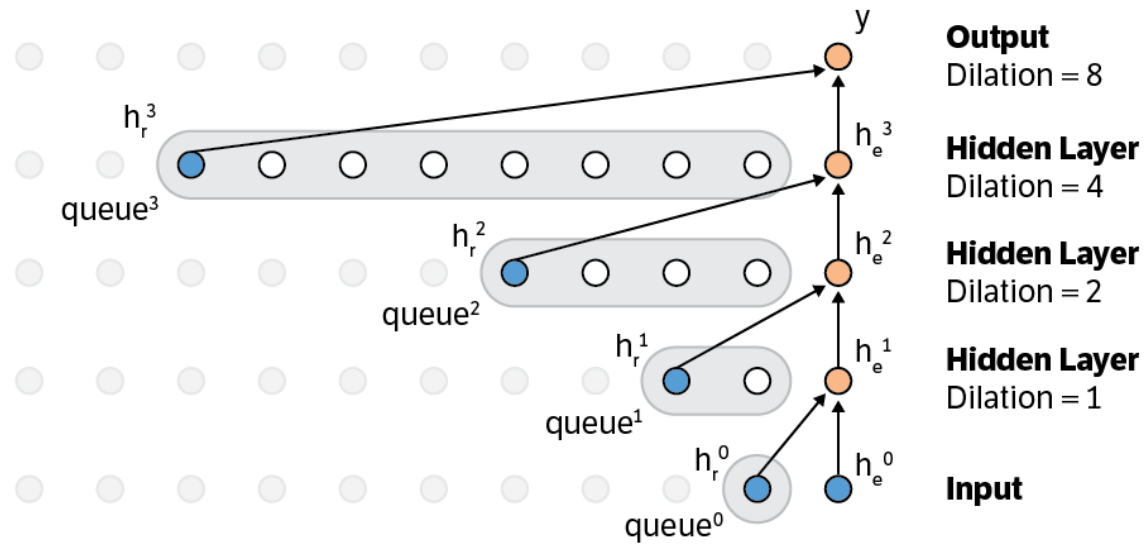
- The generated samples, x_t, are scaled back to speech with the inverse μ-law transformation.

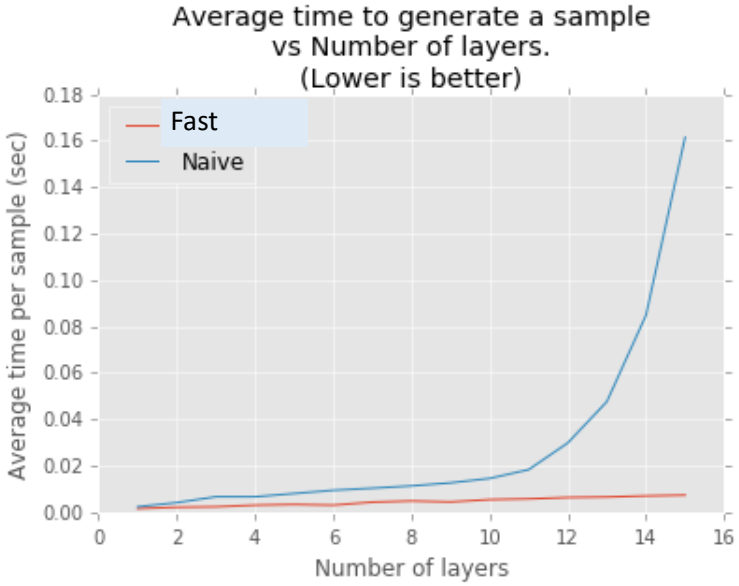
$$u = 2\frac{x}{\mu} - 1$$ Convert from $x \in \{0,1,2,\ldots,255\}$ to $u \in [-1,1]$

$$\text{speech} = \frac{sign(u)}{\mu}\left((1+\mu)^{u} - 1\right)$$ Inverse μ-law transform

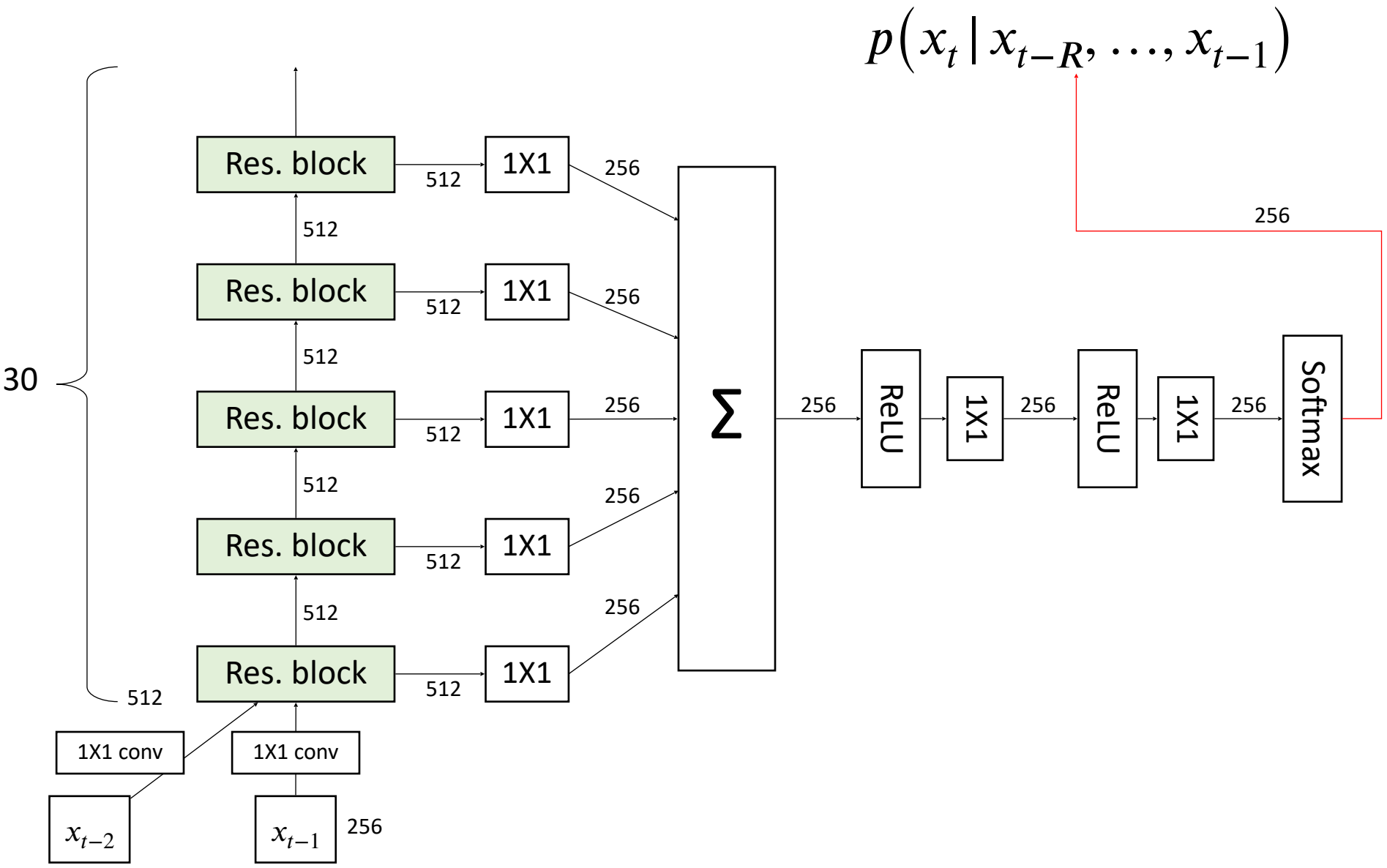
Fast WaveNet – Audio generation

- A naïve implementation of WaveNet generation requires time $O\left(2^L\right)$, where L is the number of layers.
- Recently, Tom Le Paine et al. have published their code for fast generation of sequences from trained WaveNets.
- Their algorithm uses queues to avoid redundant calculations of convolutions.
- This implementation requires time $O(L)$.

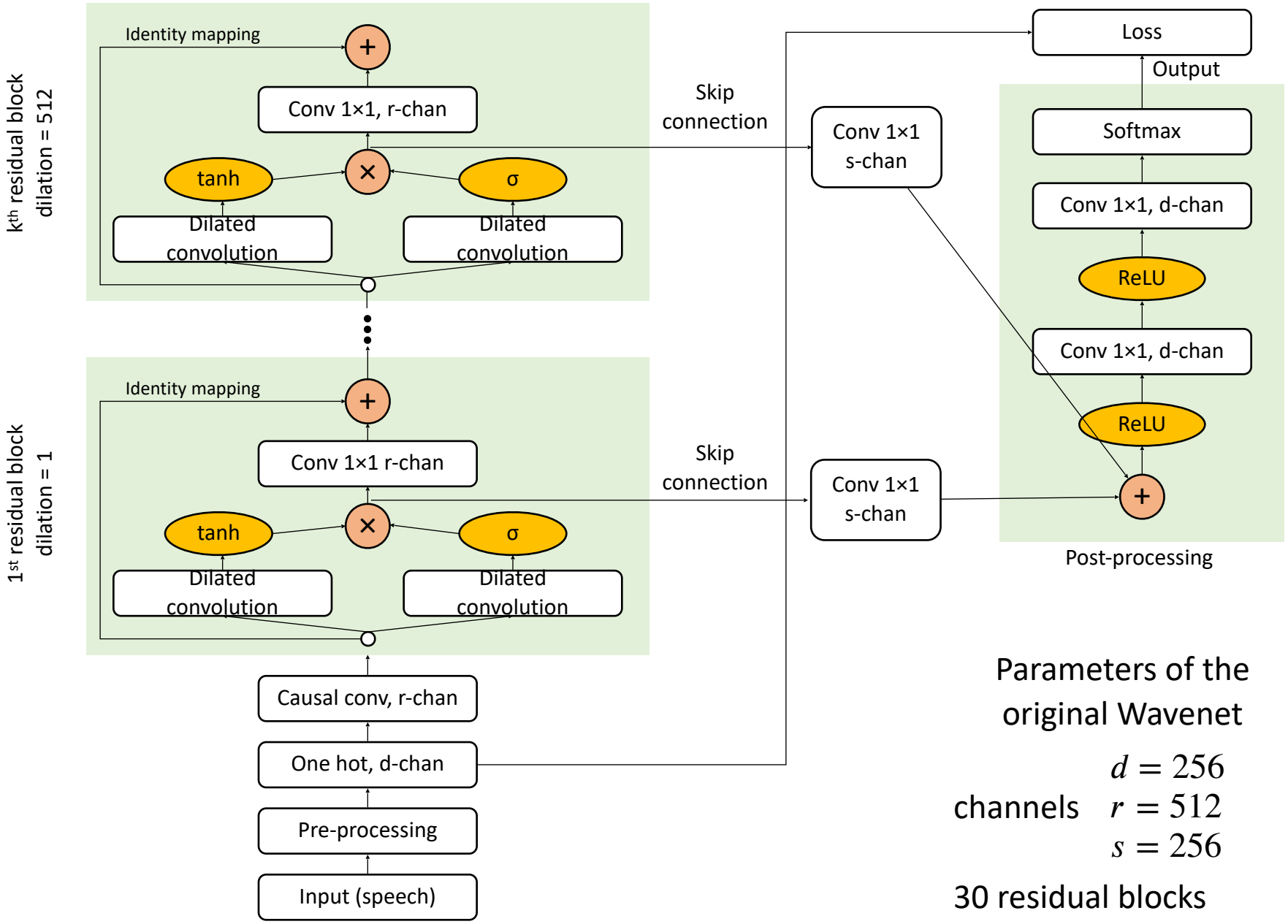




Basic WaveNet architecture



Basic WaveNet architecture



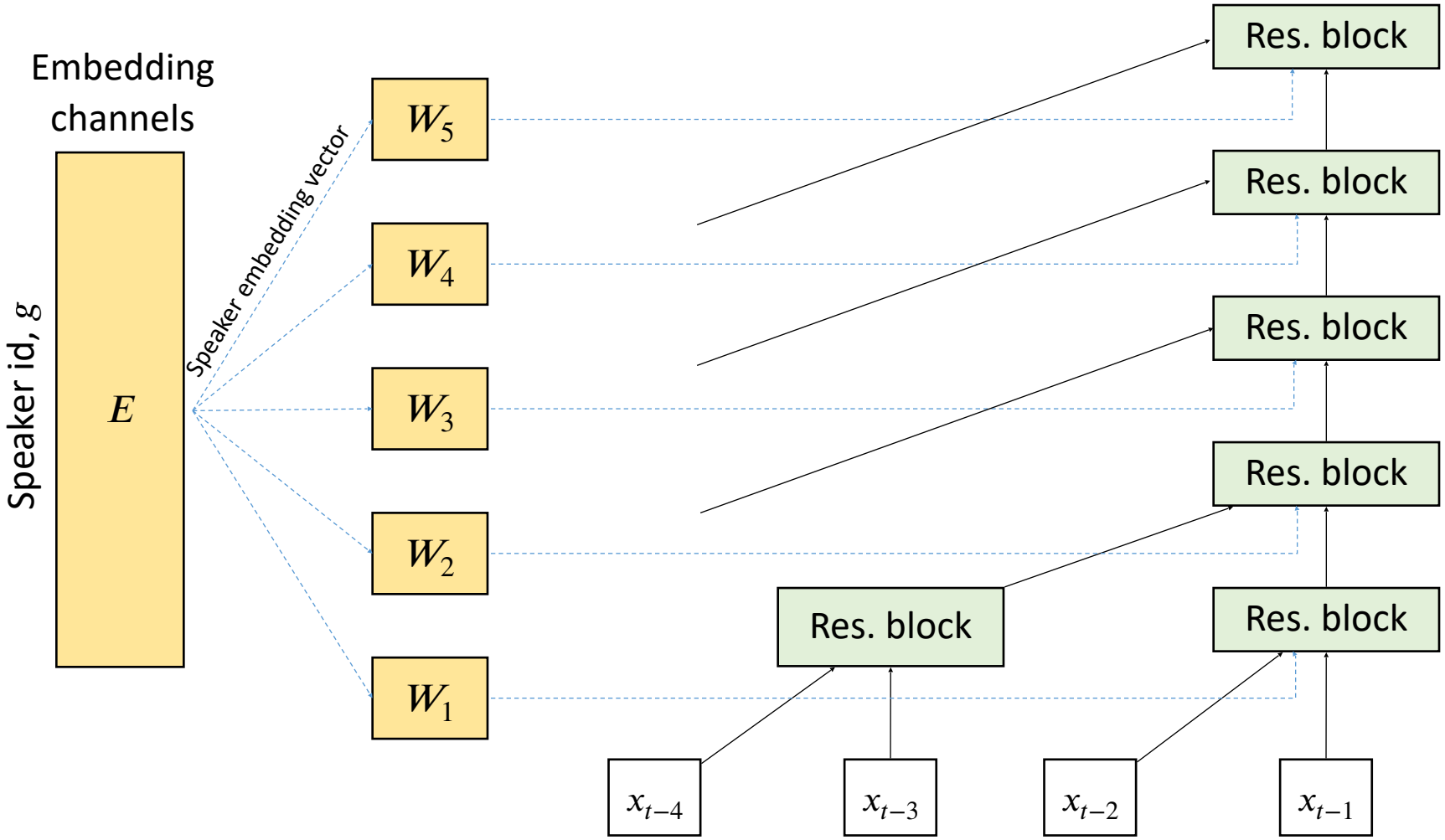
Parameters of the original Wavenet

channels $d = 256$, $r = 512$, $s = 256$

30 residual blocks

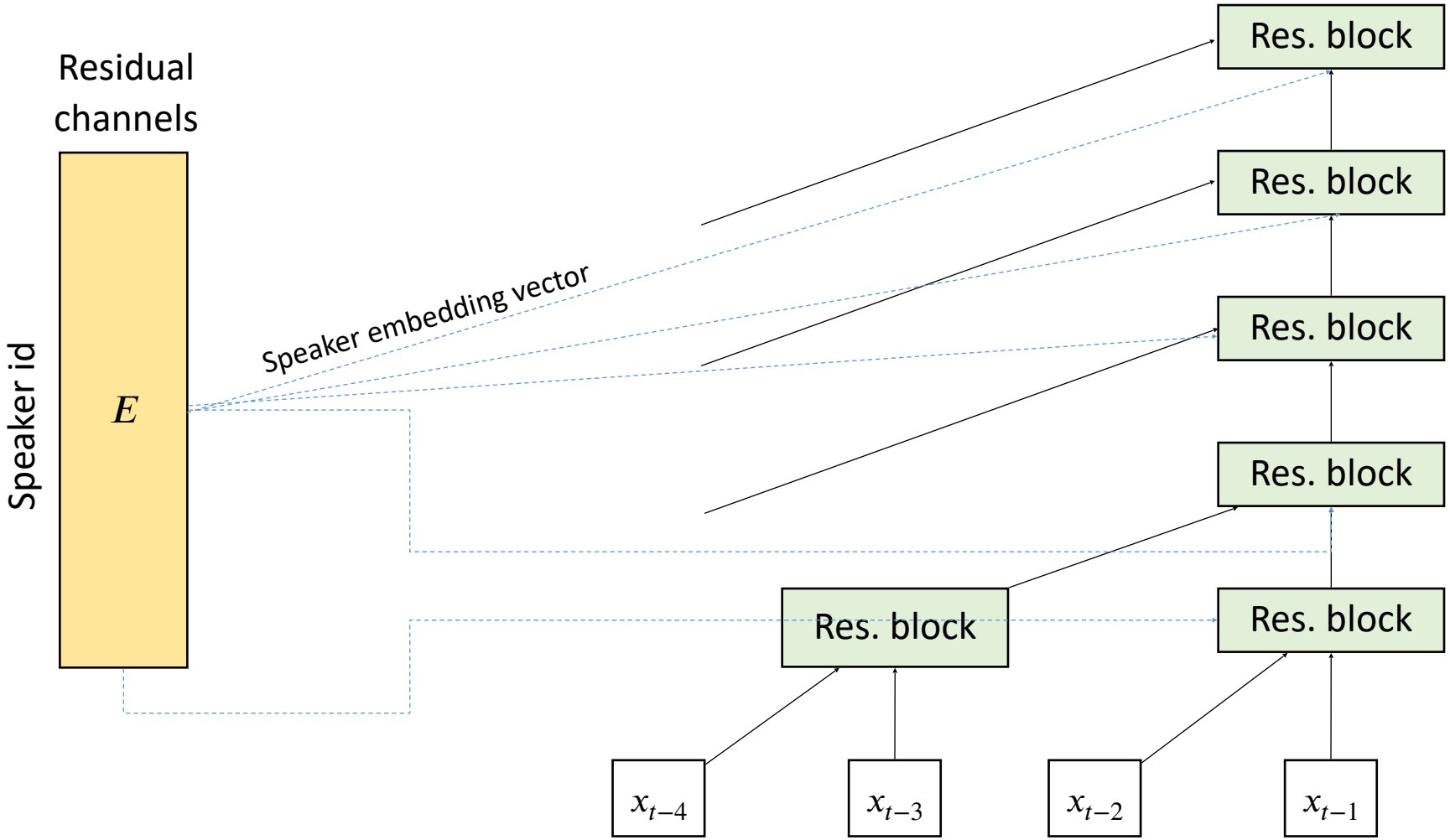
WaveNet architecture – Global conditioning

$$p\left(x_t \mid x_{t-R}, \ldots, x_{t-1}, g\right)$$



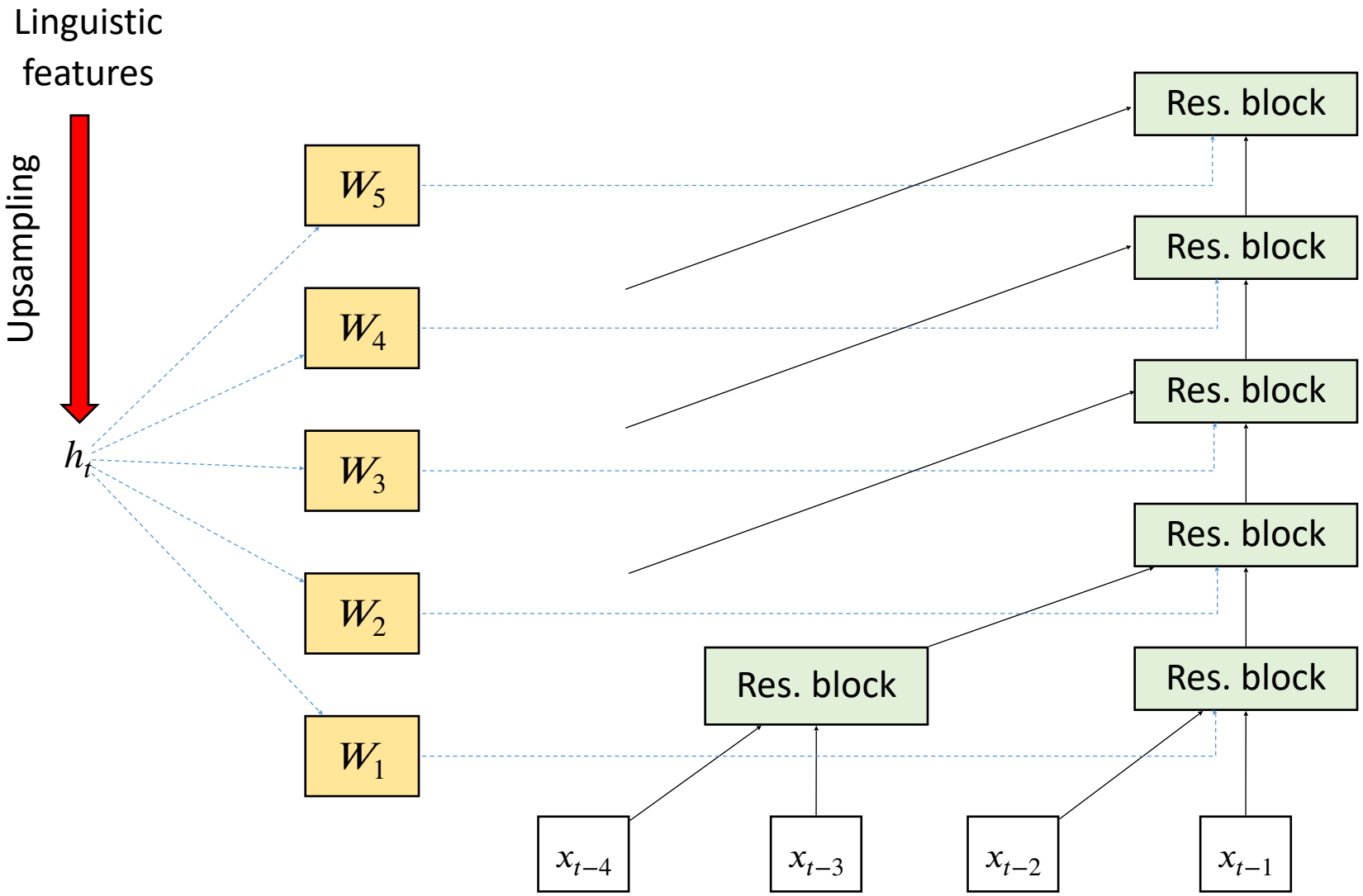
WaveNet architecture – Global conditioning

$$p\left(x_t \mid x_{t-R}, \ldots, x_{t-1}, g\right)$$



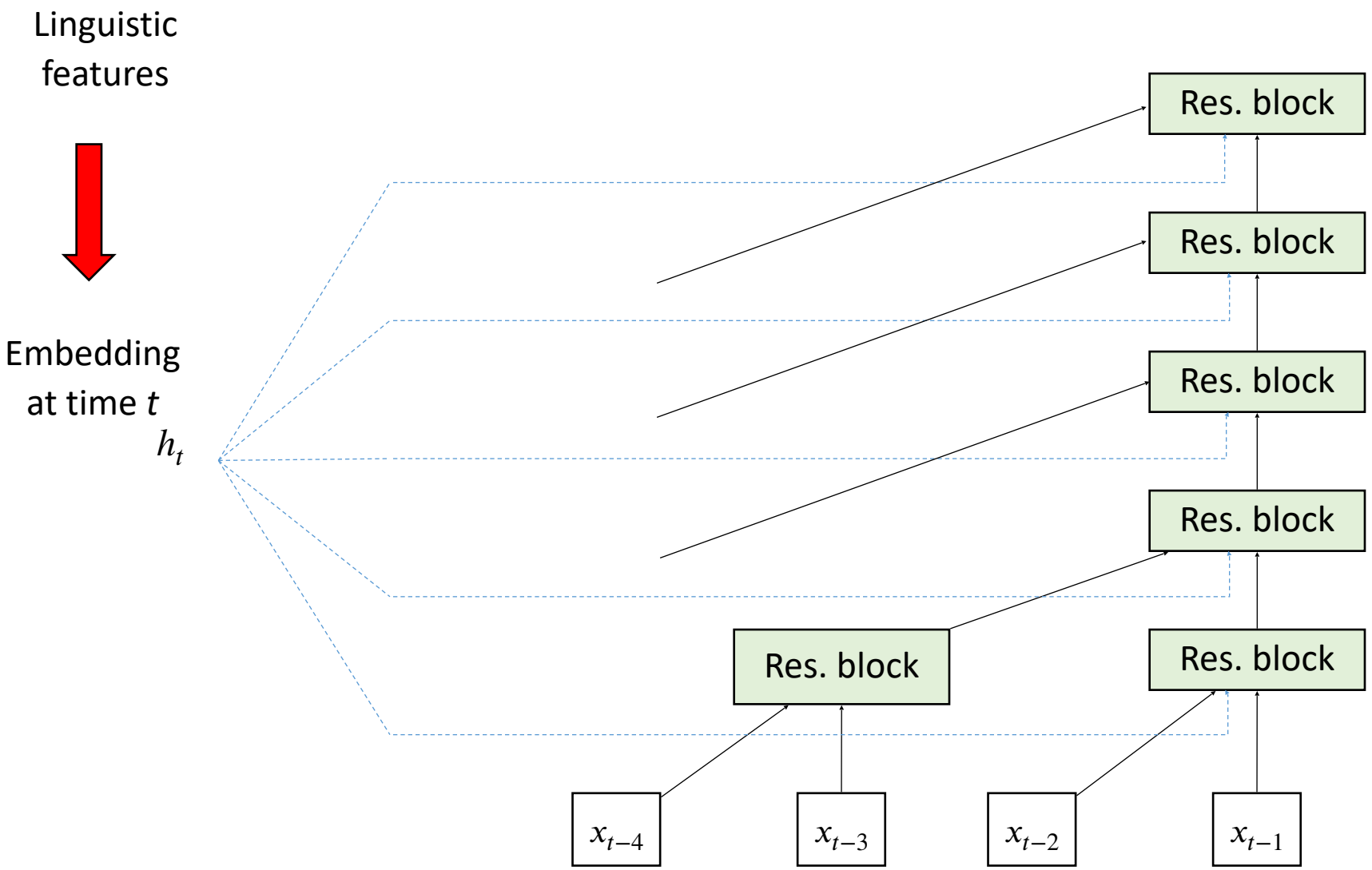
WaveNet architecture – Local conditioning

$$p(x_t \mid x_{t-R}, \ldots, x_{t-1}, h_t)$$



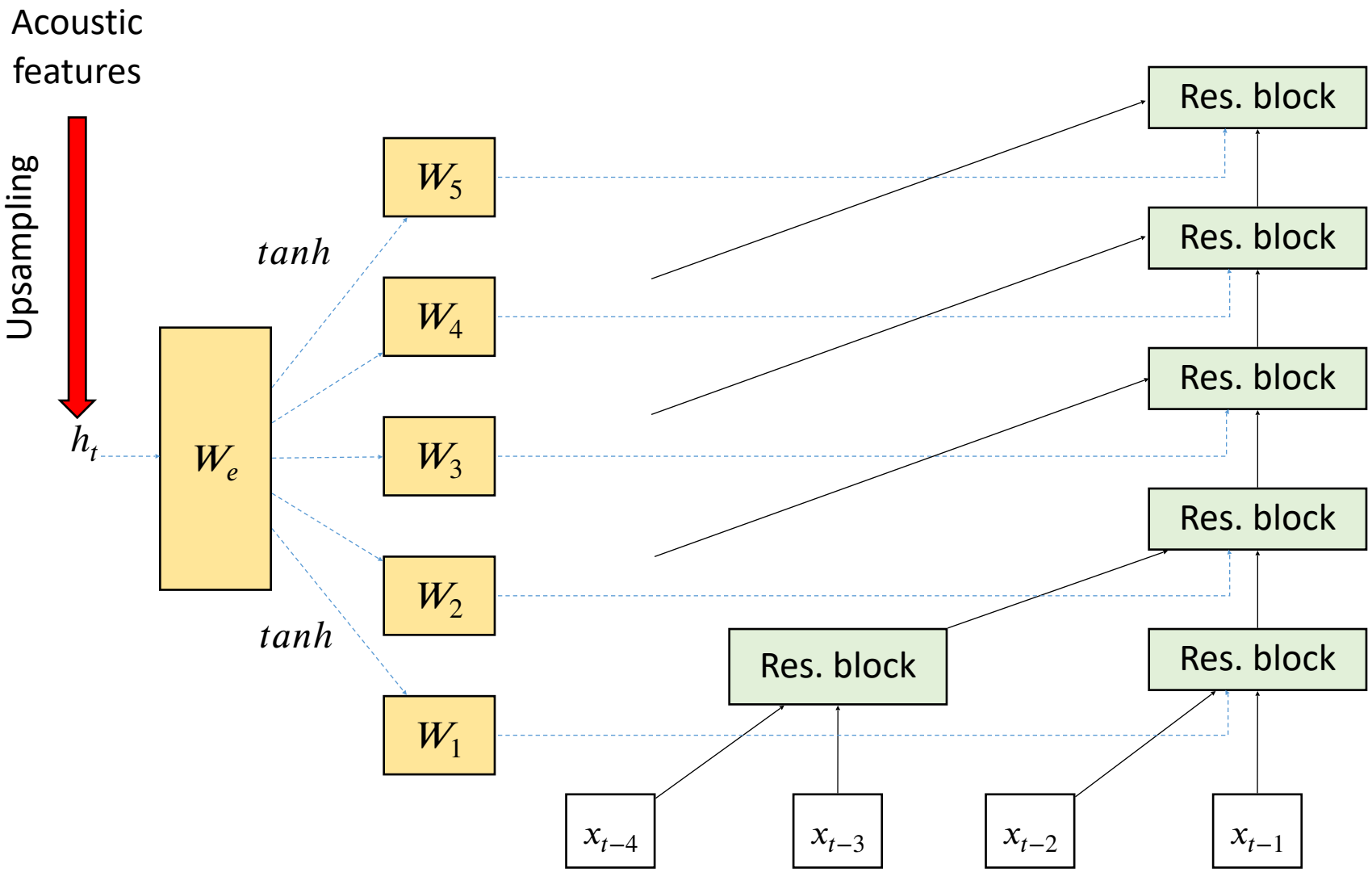
WaveNet architecture – Local conditioning

$$p(x_t \mid x_{t-R}, \ldots, x_{t-1}, h_t)$$



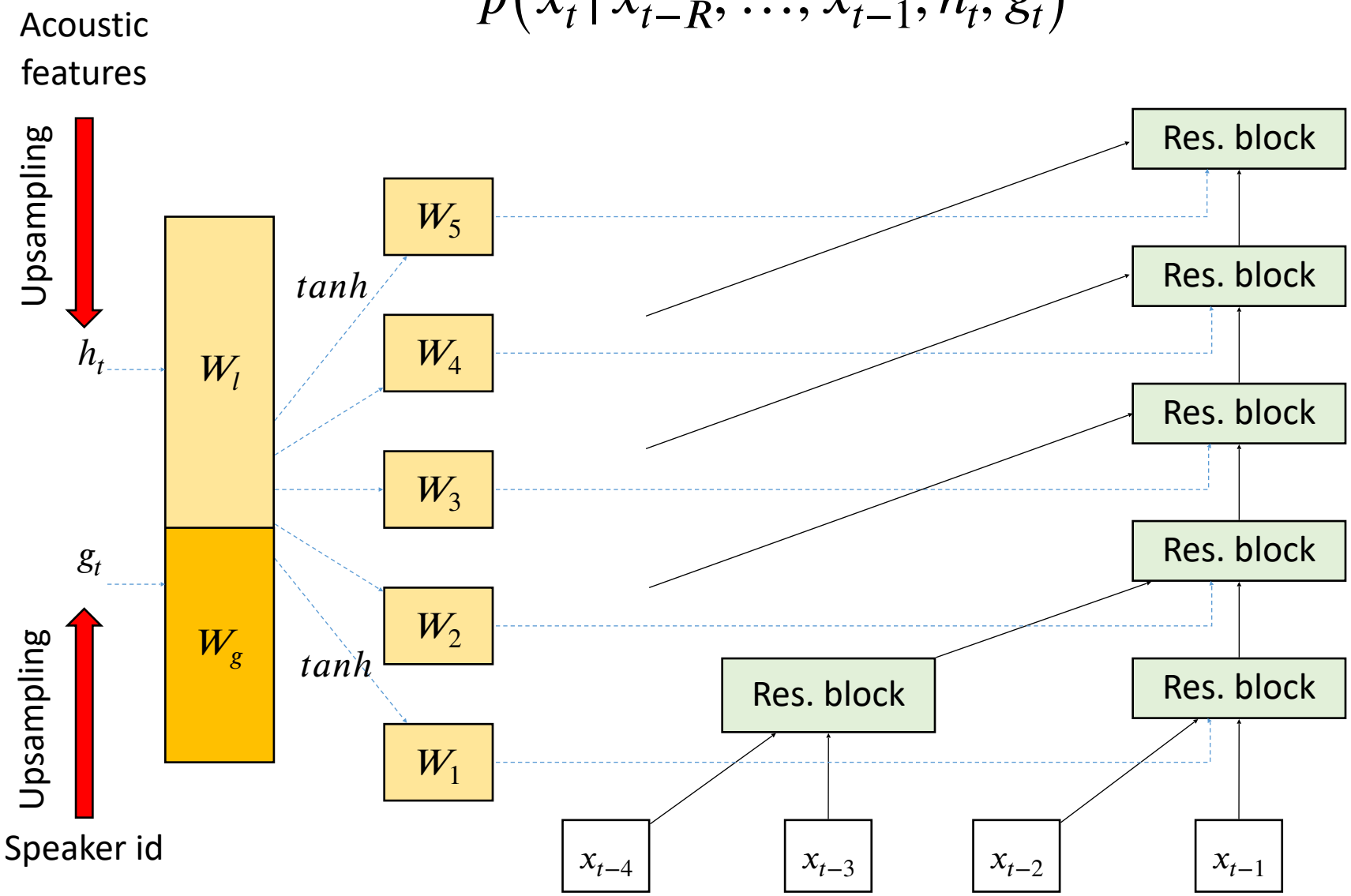
WaveNet architecture – Local conditioning

$$p(x_t \mid x_{t-R}, \ldots, x_{t-1}, h_t)$$

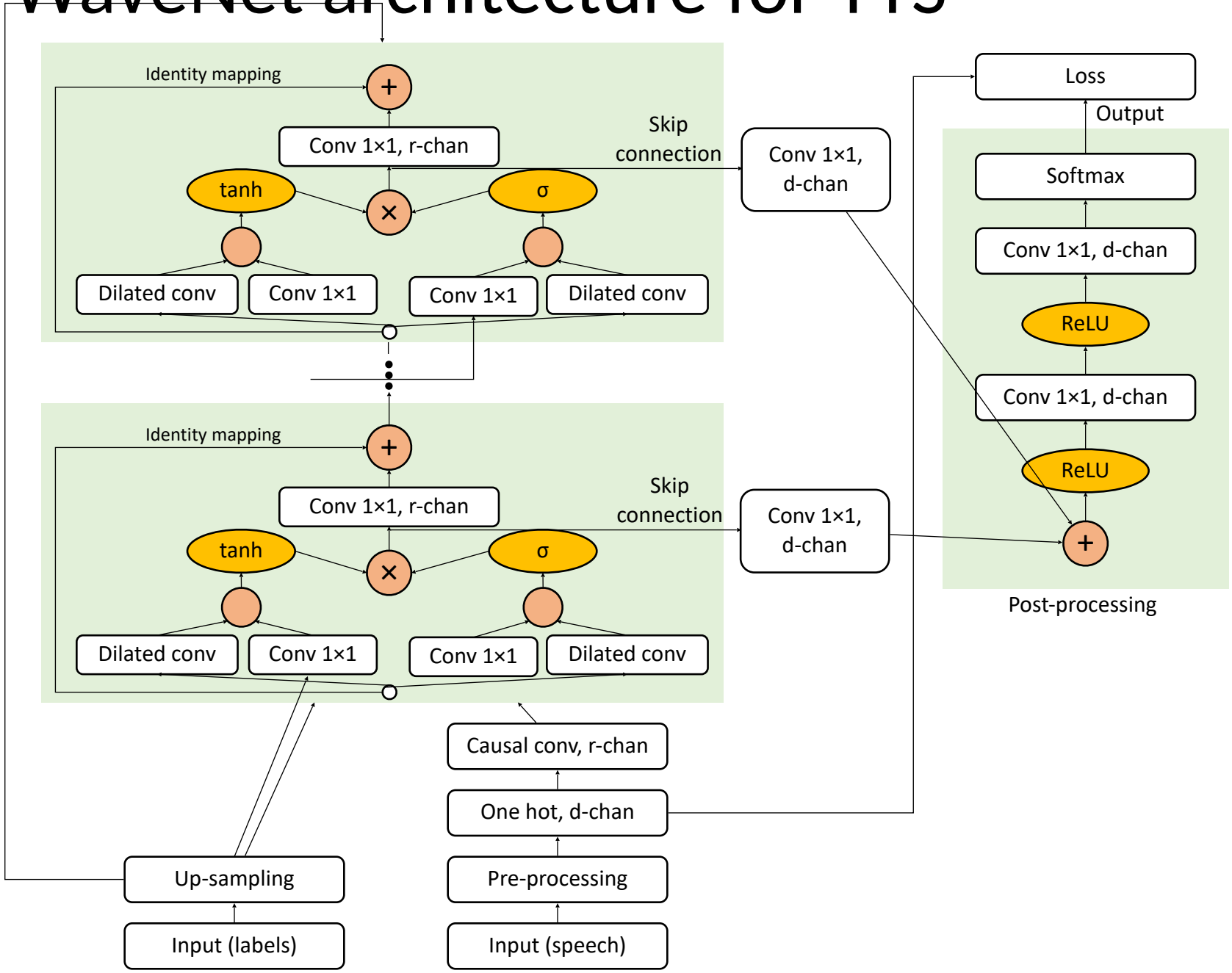


WaveNet architecture – Local and global conditioning

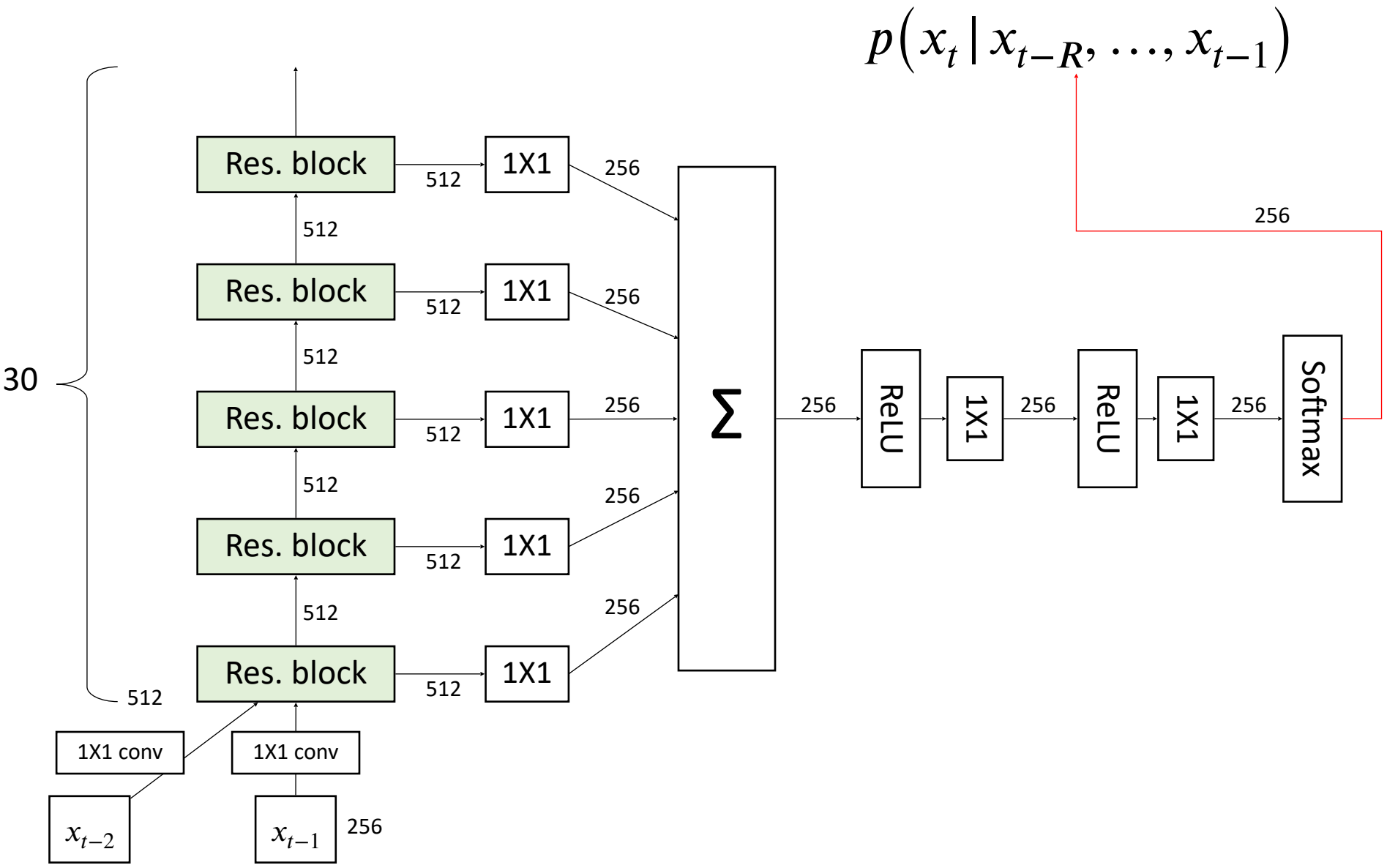
$$p(x_t \mid x_{t-R}, \ldots, x_{t-1}, h_t, g_t)$$



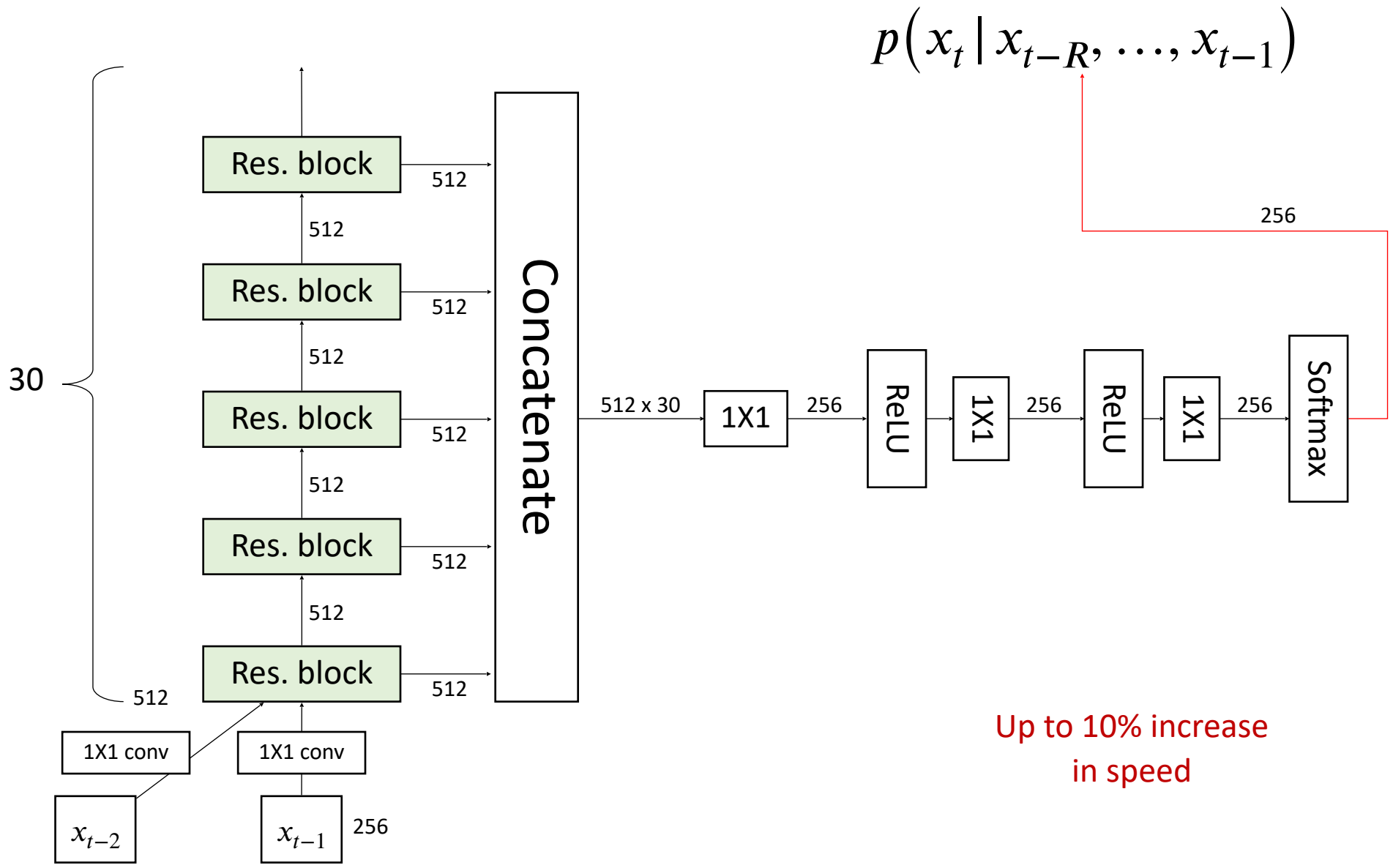
WaveNet architecture for TTS



WaveNet architecture -Improvements

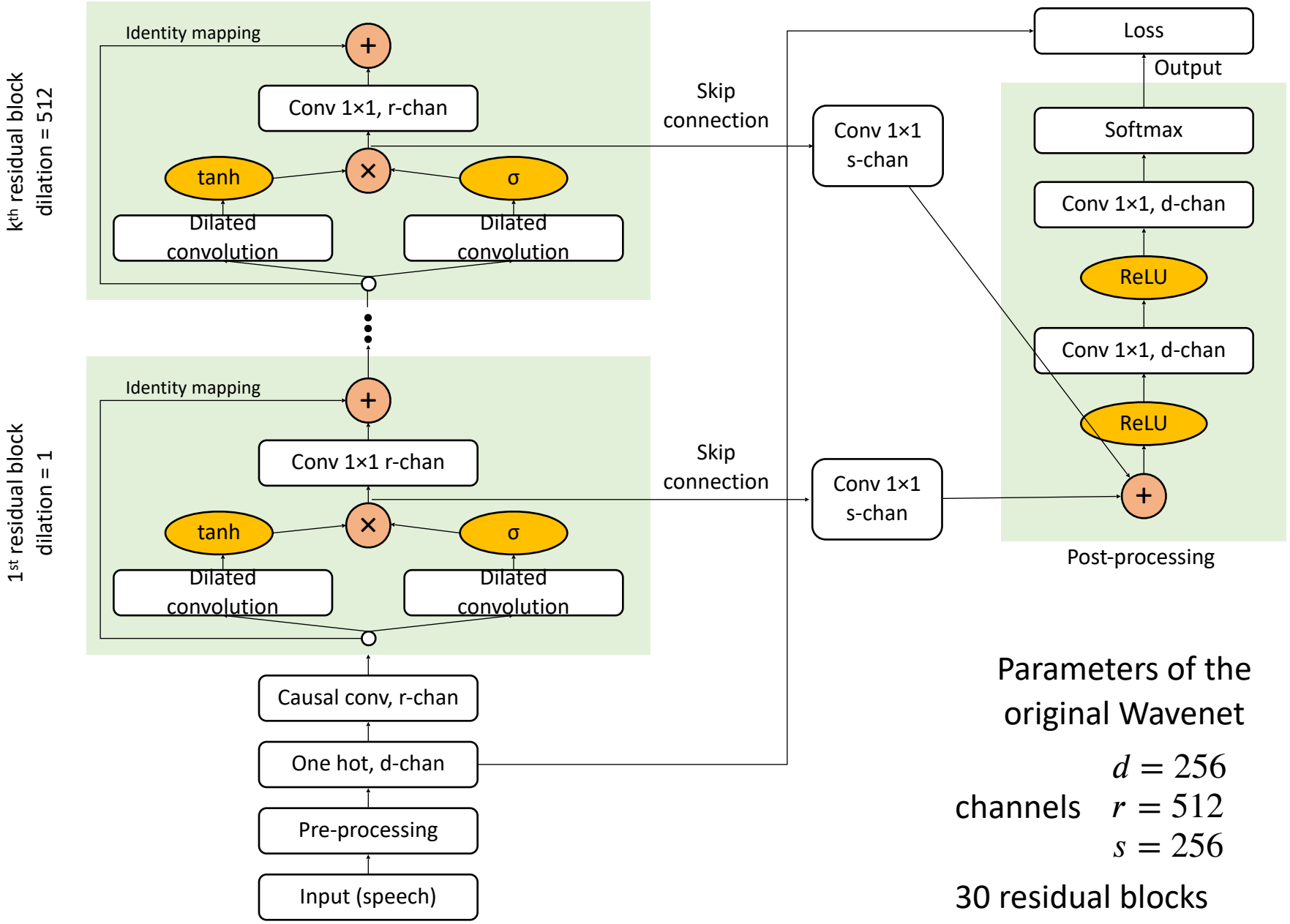


WaveNet architecture - Improvements



Up to 10% increase in speed

WaveNet architecture - Improvements

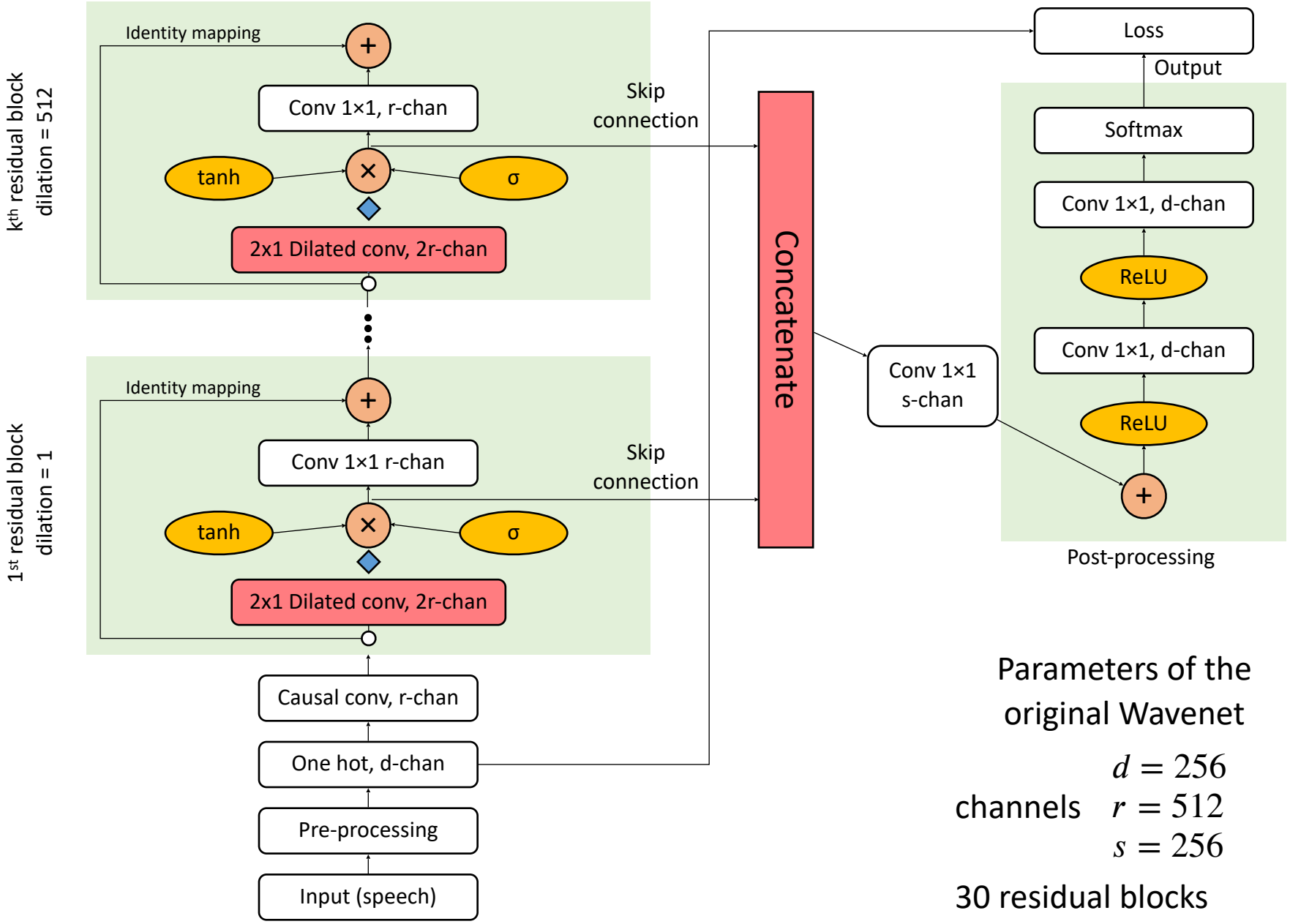


Parameters of the original Wavenet

channels $d = 256$, $r = 512$, $s = 256$

30 residual blocks

WaveNet architecture - Improvements

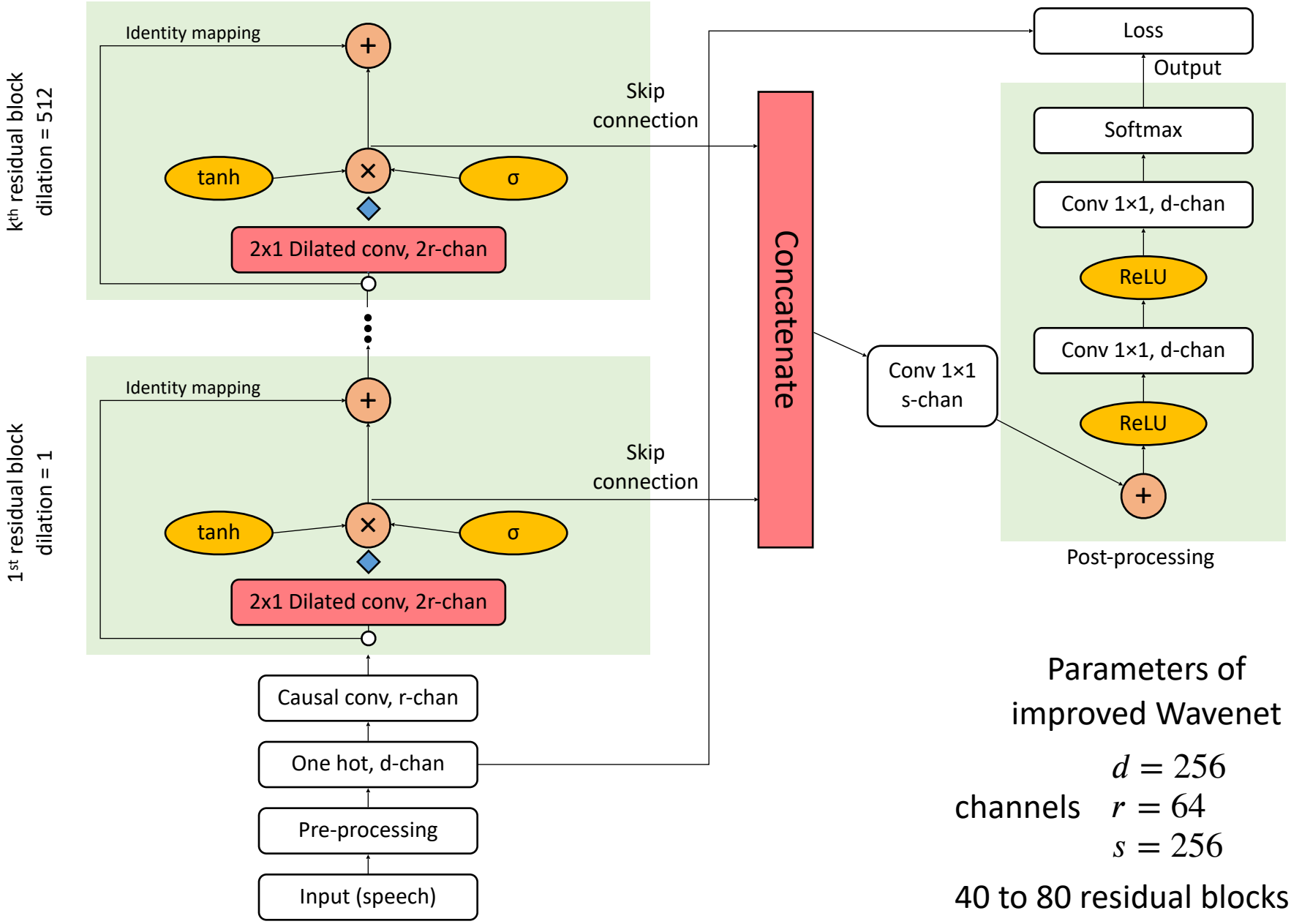


Parameters of the original Wavenet

channels $d = 256$, $r = 512$, $s = 256$

30 residual blocks

WaveNet architecture - Improvements



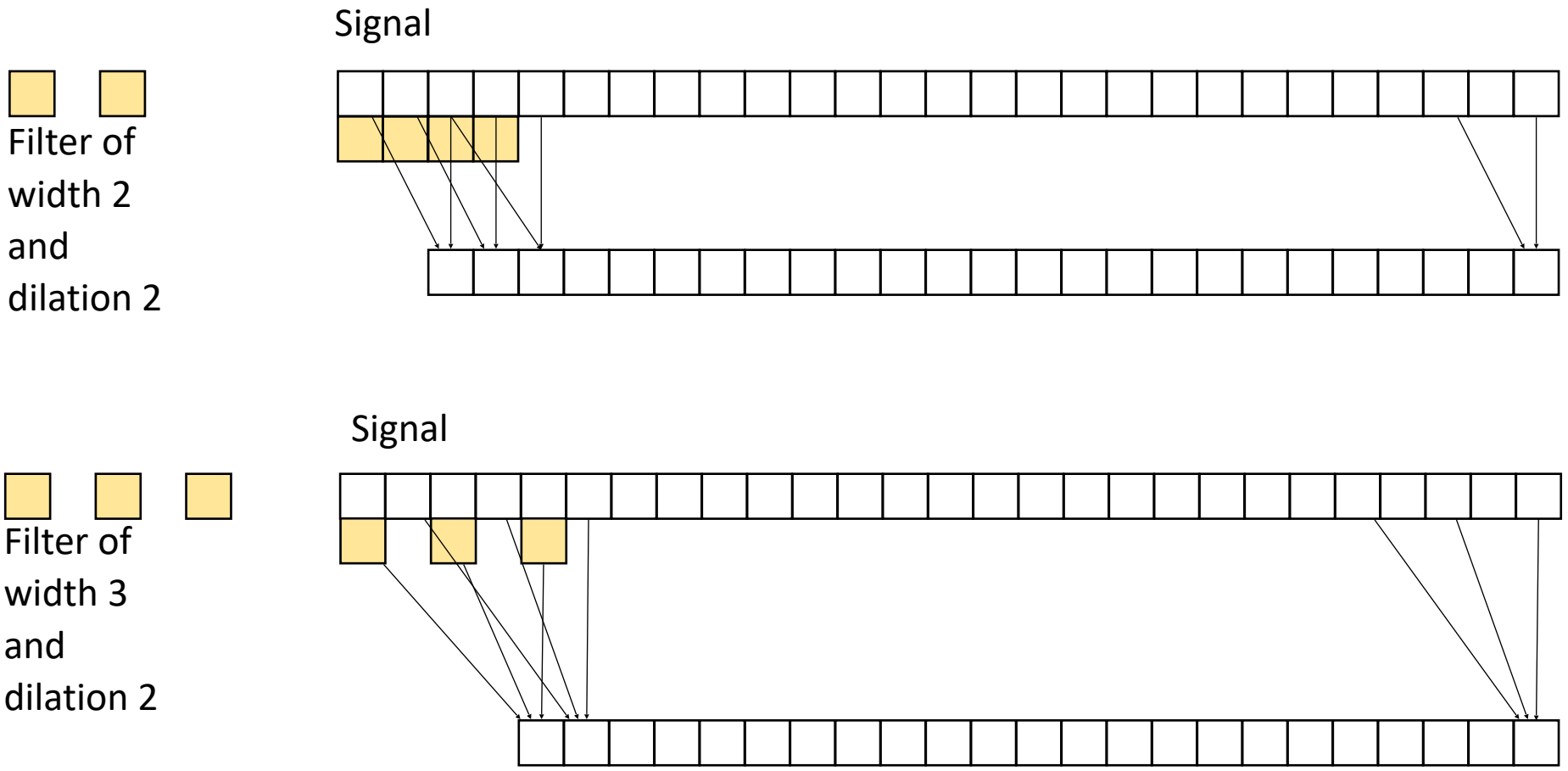
Parameters of improved Wavenet

channels $d = 256$, $r = 64$, $s = 256$

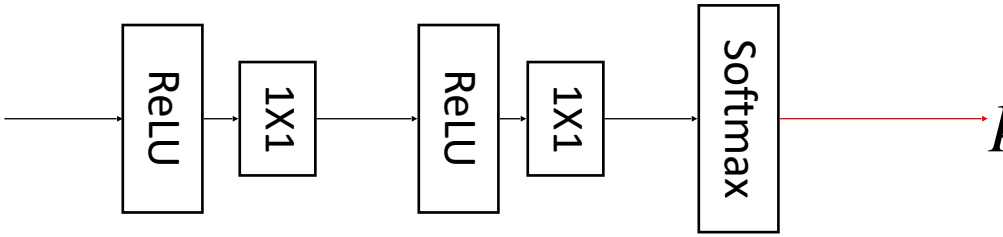
40 to 80 residual blocks

WaveNet architecture -Improvements

- In order to increase the receptive field of WaveNet, we may use filter width 3 instead of 2 in dilated causal convolutions.



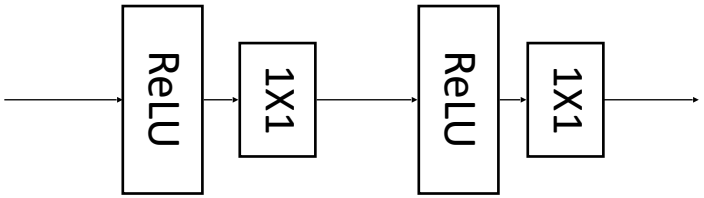
WaveNet architecture -Improvements



$$p\left(x_t \mid x_{t-R}, \ldots, x_{t-1}\right)$$

Post-processing in the original WaveNet

8-bit quantization



Parameters of mixture of logistics

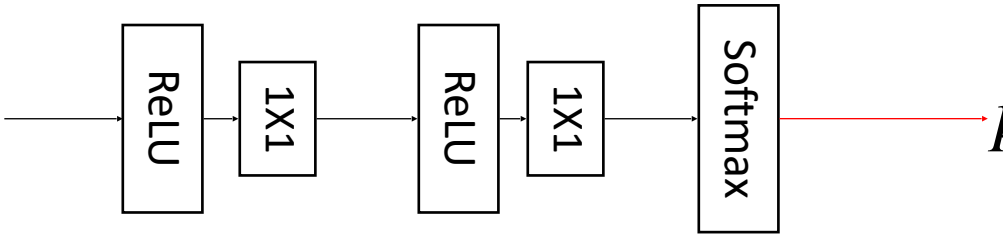
Post-processing in new Wavenet

16-bit quantization

$$P\left(x \,\middle|\, \pi, \mu, s\right) = \sum_{i=1}^{K} \pi_i \frac{1}{s_i} \sigma\left(\frac{x-\mu_i}{s_i}\right)\left(1-\sigma\left(\frac{x-\mu_i}{s_i}\right)\right)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

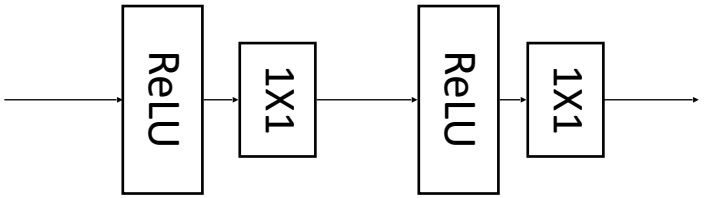
WaveNet architecture -Improvements



$$p\left(x_t \mid x_{t-R}, \ldots, x_{t-1}\right)$$

Post-processing in the original WaveNet

8-bit quantization



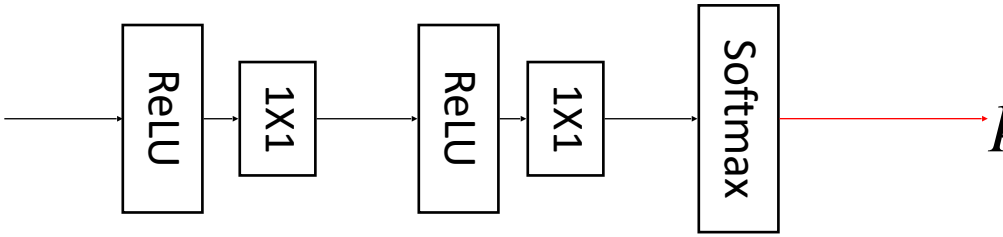
Parameters of mixture of Gaussians

Post-processing in new Wavenet

16-bit quantization

$$P\left(x \,\middle|\, \pi, \mu, s\right) = \sum_{i=1}^{K} \pi_i \frac{1}{\sqrt{2\pi}\, s_i} \exp\left(-\frac{\left(x-\mu_i\right)^2}{2 s_i^2}\right)$$

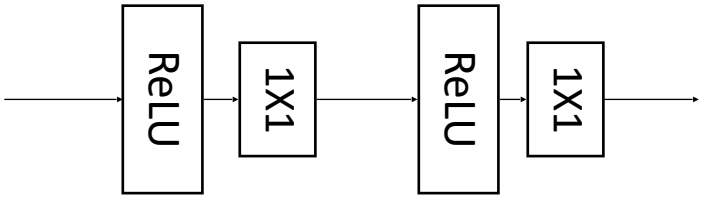
WaveNet architecture -Improvements



$$p\left(x_t \mid x_{t-R}, \ldots, x_{t-1}\right)$$

Post-processing in the original WaveNet

8-bit quantization



Parameters of mixture of logistics

Post-processing in 16bit WaveNet (high fidelity WaveNet)

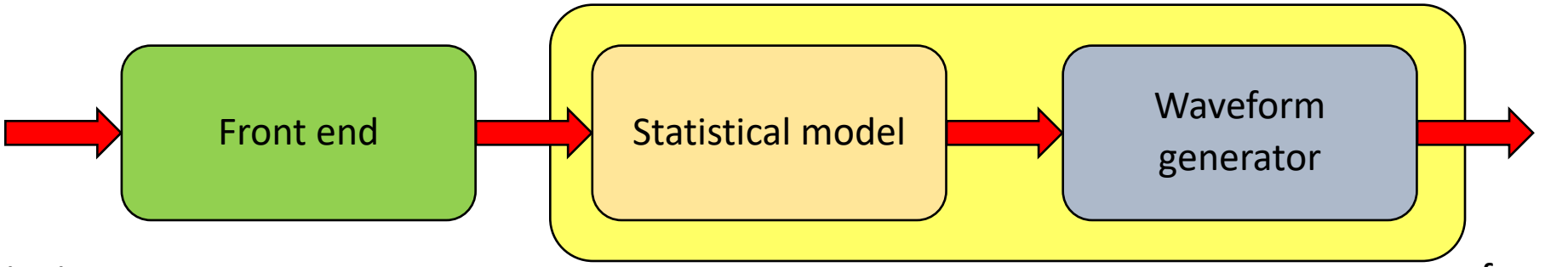
16-bit quantization

- The original WaveNet maximizes the cross-entropy between the desired distribution $\left[0,\ldots,0,1,0,\ldots,0\right]$ and the network prediction $p\left(x_t \mid x_{t-R}, \ldots, x_{t-1}\right) = \left[y_1, y_2, \ldots, y_{256}\right]$
- The 16bit WaveNet maximizes the log-likelihood $\frac{1}{T-R}\sum_{n=R+1}^{T} \log p(x_t \mid \pi, \mu, s)$

Generated audio samples for the original WaveNet

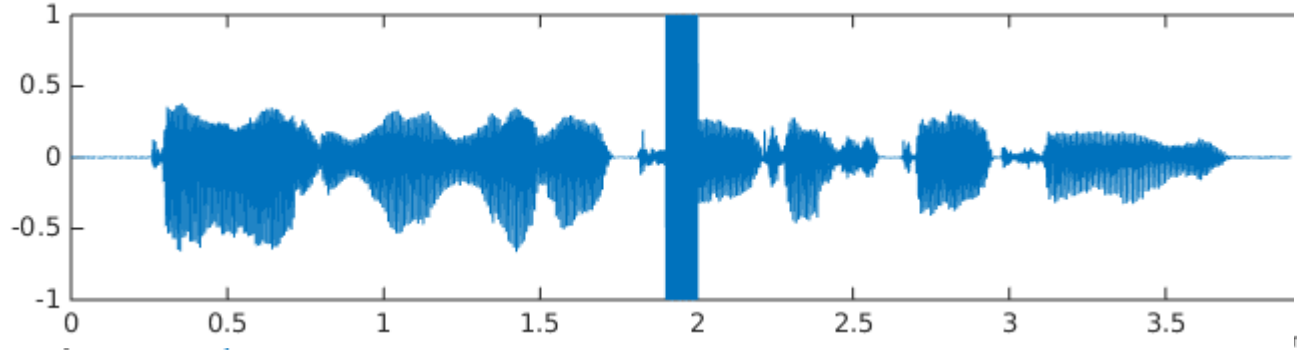
- Basic WaveNet Model trained with the cmu_us_slt_arctic-0.95-release.zip database (~40 min, 16000 Hz)

- WaveNet model with local conditioning TTS linguistic labels, trained with an American English female speaker database (~5 hours, 16000 Hz).

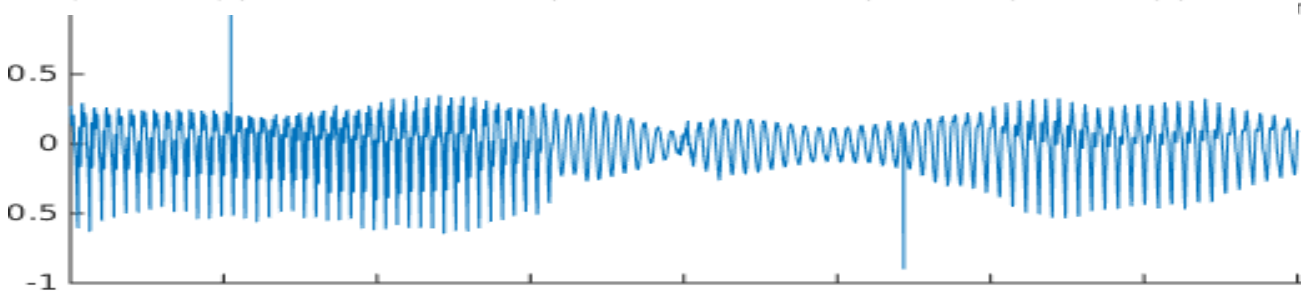


WaveNet – Known Problems

- WaveNet sometimes produces type I and type II artifacts (Wu et al., Collapsed speech segment detection and suppression for WaveNet vocoder, 2018.).
- Type II artifacts are common in the original WaveNet, but very rare in the 16-bit WaveNet.
  - They are caused by the sampling during synthesis.
- Type I artifacts are rare in the original WaveNet but occur sometimes in the 16-bit WaveNet
  - They are instabilities which can be detected even at training time, by plotting the parameters of the predicted distribution $p(x_t \mid x_{<t}, l_t)$.
  - During synthesis they are triggered by mis-aligned or by unseen labels or even by silence segments.



Type I artifats



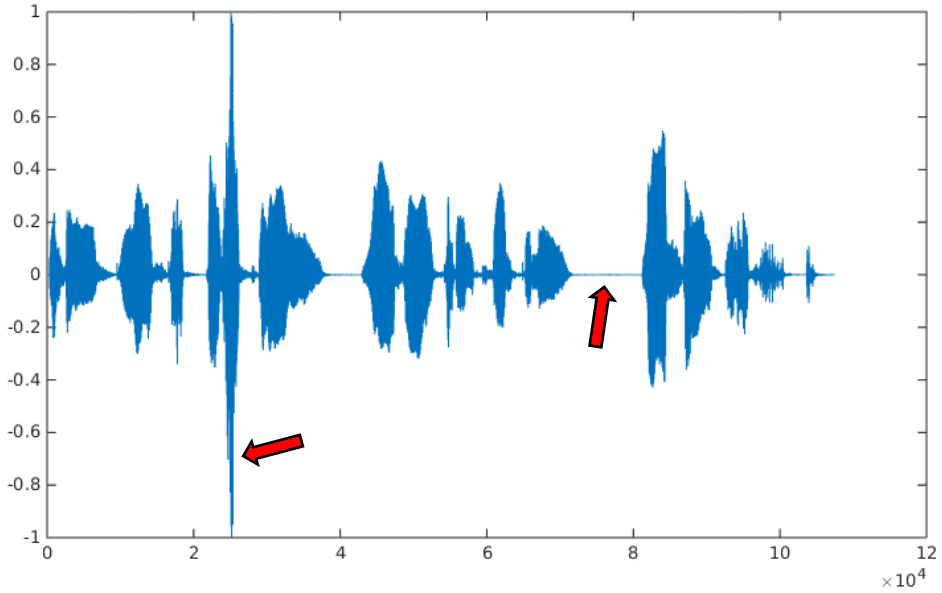
Type II artifats

The source of type I artifacts

- The **type I artifacts** is a characteristic of a trained model which predicts out of range values for the mean and scale of the output distribution.
- They are not caused by the sampling algorithm.
- They are instabilities of the model which are related with,
  1. the form of the loss function in combination with
  2. audio segments with small variations around the mean trajectory, where the mean trajectory is computed across the whole database.
  3. In silence segments, a WaveNet model learns mean $\boldsymbol{\mu \cong 0}$, and standard deviation $\boldsymbol{0 < \sigma \approx 0}$.

Cross-entropy loss vs negative log-likelihood loss

- We will compare the stability of the two loss functions using the LJ Speech Dataset 1.1, created by Keith Ito.
- This dataset contains **audio clippings** and **long silence segments** between words.



- The network architecture is similar in both tests. Only the output layers and the losses differ.
  a) The output layer is a 256 softmax and the loss is cross-entropy
  b) The ouput layer predicts the two parameters σ and μ of a Gaussian distribution with loss the negative log-likelihood

Cross-entropy loss vs negative log-likelihood loss

- The following images show the utterance LJ001-0001 synthesized by a model trained with cross-entropy and a model trained with negative log-likelihood.

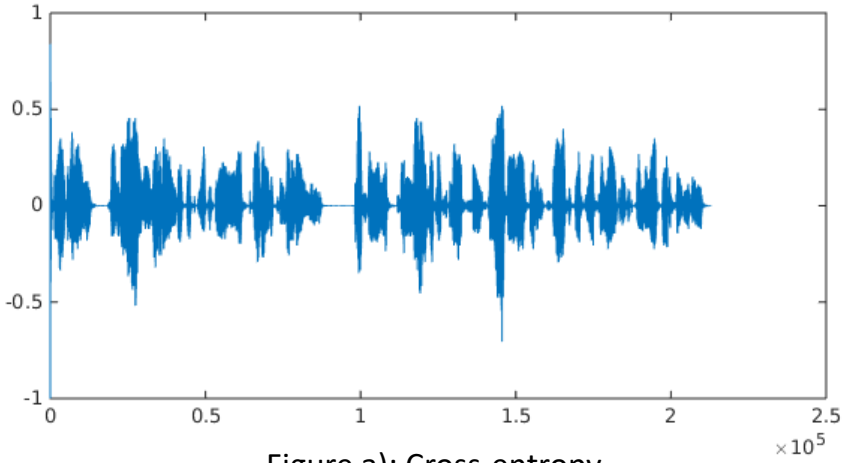


Figure a): Cross-entropy

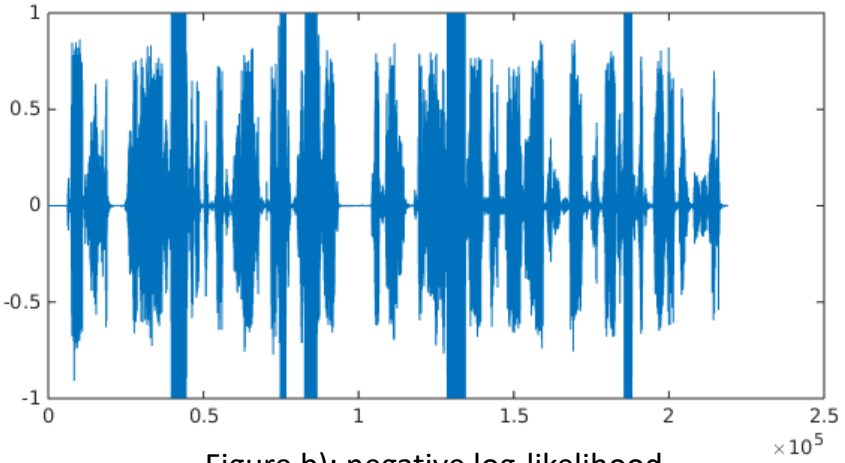


Figure b): negative log-likelihood

Heuristic solutions to avoid type I artifacts

1. *Using bigger and cleaner databases*
   - This is the classic machine learning approach, which, as expected, it also reduces the type I artifacts.
   - However, depending on the application, having a bigger or a cleaner database is not always feasible.
2. *Reduce the silence regions*
   - All publicly available implementations of WaveNet remove the silence regions at the beginning and at the end of each utterance. And this simple mechanism greatly improves their stability.
   - However, not all silence regions can be removed (e.g., the silence regions between words).
3. *Adding small noise to the data*
   - This is effective to avoid type I artifacts in WaveNet, but the noise during training affects the quality of the synthesized speech.
   - Also, this technique does not significantly helps with Parallel WaveNet.
4. *Using dropout*
   - Dropout helps the convergence of a model, especially when the training database is small or has noise.
   - Dropout, is absolutely necessary in speech recognition.
   - However, dropout may slightly affect the quality of the synthesized speech.
   - Also, dropout reduces but it does not completely eliminate the type I artifacts.

Heuristic solutions to avoid type I artifacts

5. *Clipping the gradients* (e.g., tf.clip_by_value(gradient, min_value, max_value)).
   - This mechanism does not help significantly when there are outliers with extremely large values (e.g., $\log loss_t = 10^{12}$).
     - Due to memory and time constraints, the gradient clipping is applied after the averaging across time (tf.reduce_mean(loss)).
   - For example, in Figure a) a few outliers affect the mean loss of Wavenet.
6. *Clipping the values of the outliers in the loss function* (e.g. loss = tf.maximum(loss, max_loss_value)). This is shown in Figure b) . Then average the loss across time (tf.reduce_mean(loss)).
   - The influence of the outliers is reduced and kept under control.

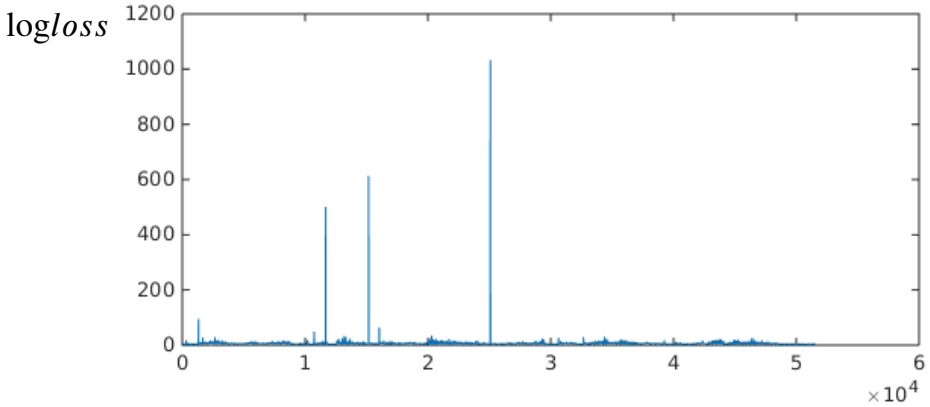


Figure a)

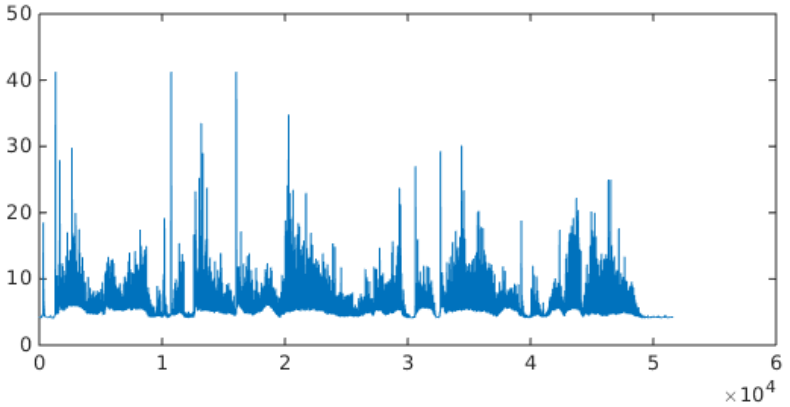


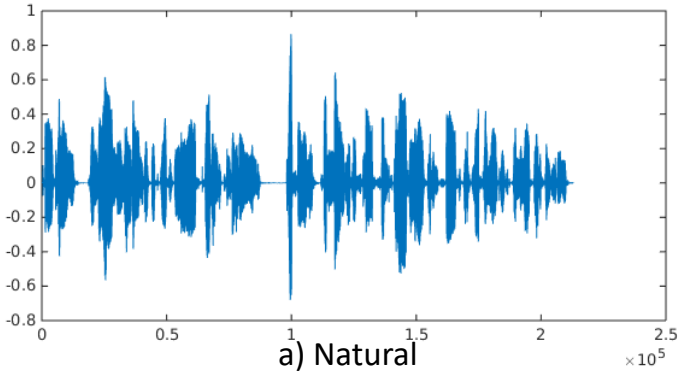
Figure b)

Heuristic solutions to avoid type I artifacts

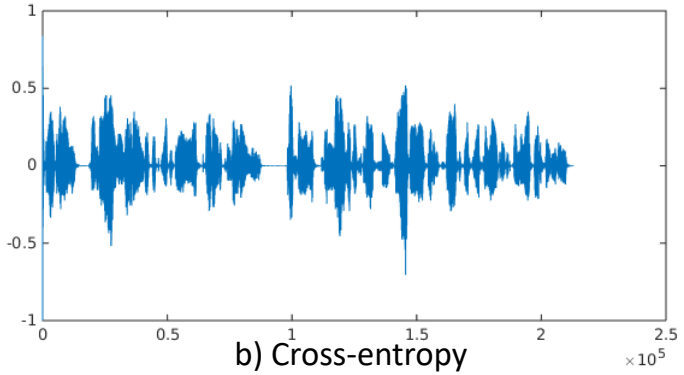
7. *Clipping the variance parameter* predicted from the network to values above a threshold (e.g., log_scale = tf.maximum(log_scale, log_scale_min).
   - This mechanism is used in all publicly available implementations of WaveNet and it works.
   - However, it does not prevent the network to keep producing very small variance parameters, since no feedback signal that is propagated back to the network.
   - Unfortunately, very small variances are also associated with out of range mean values.
8. Adding a term to the loss function that *penalizes the small variances*.
9. *Using weight normalization* (Tim Salimans, Diederik P. Kingma, Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks, 2016).
10. *Use large batch_size*.

Heuristic solutions to avoid type I artifacts

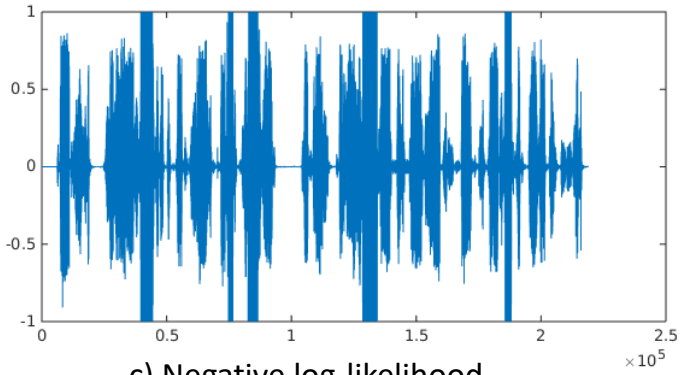
Example from the LJ Speech dataset:



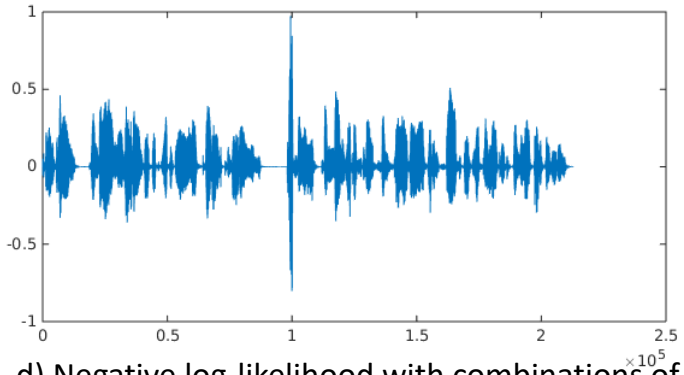
a) Natural



b) Cross-entropy



c) Negative log-likelihood



d) Negative log-likelihood with combinations of the heuristics

References

1. van den Oord, Aaron; Dieleman, Sander; Zen, Heiga; Simonyan, Karen; Vinyals, Oriol; Graves, Alex; Kalchbrenner, Nal; Senior, Andrew; Kavukcuoglu, Koray, WaveNet: A Generative Model for Raw Audio, arXiv:1609.03499
2. Arik, Sercan; Chrzanowski, Mike; Coates, Adam; Diamos, Gregory; Gibiansky, Andrew; Kang, Yongguo; Li, Xian; Miller, John; Ng, Andrew; Raiman, Jonathan; Sengupta, Shubho; Shoeybi, Mohammad, Deep Voice: Real-time Neural Text-to-Speech, eprint arXiv:1702.07825
3. Arik, Sercan; Diamos, Gregory; Gibiansky, Andrew; Miller, John; Peng, Kainan; Ping, Wei; Raiman, Jonathan; Zhou, Yanqi, Deep Voice 2: Multi-Speaker Neural Text-to-Speech, eprint arXiv:1705.08947
4. Le Paine, Tom; Khorrami, Pooya; Chang, Shiyu; Zhang, Yang; Ramachandran, Prajit; Hasegawa-Johnson, Mark A.; Huang, Thomas S., Fast Wavenet Generation Algorithm, eprint arXiv:1611.09482
5. Ramachandran, Prajit; Le Paine, Tom; Khorrami, Pooya; Babaeizadeh, Mohammad; Chang, Shiyu; Zhang, Yang; Hasegawa-Johnson, Mark A.; Campbell, Roy H.; Huang, Thomas S., Fast Generation for Convolutional Autoregressive Models, eprint arXiv:1704.06001
6. Wavenet implementation in Tensorflow. Found in https://travis-ci.org/ibab/tensorflow-wavenet (Author: Igor Babuschkin et al.).
7. Fast Wavenet implementation in Tensorflow. Found in https://github.com/tomlepaine/fast-wavenet (Authors: Tom Le Paine, Pooya Khorrami, Prajit Ramachandran and Shiyu Chang)
8. Aäron van den Oord et al., Parallel WaveNet: Fast High-Fidelity Speech Synthesis, arXiv:1711.10433
9. Tim Salimans, Andrej Karpathy, Xi Chen, Diederik P. Kingma, PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications