Imperial College London

Sequence Modelling

Deep Learning Course

Kevin Webster, Pierre Harvey Richemond

Course outline

1. Introduction to deep learning
2. Neural networks optimisation
3. Convolutional neural networks
4. Introduction to reinforcement learning - part 1
5. Introduction to reinforcement learning - part 2
6. Sequence models
7. Generative adversarial networks (GANs)
8. Variational autoencoders (VAEs)
9. Normalising flows
10. Theories of deep learning

Outline

Outline

Example sequence modelling problems

- **Speech recognition**

⇒ "These aren't the droids you're looking for"

- **Machine translation**

"I want forty kilograms of persimmons" ⇒ "Ich will vierzig Kilogramm Persimonen"

- **Question answering**

"What sport is this?"

⇒ "Baseball"

Example sequence modelling problems

- **Sentiment analysis**

⇒

- **Anomaly detection**

⇒ "Robbery"

- **Music generation**

Outline

Sequence modelling problems

Recurrent Neural Networks

- Backpropagation through time
- Vanishing and exploding gradients
- LSTM and GRU networks
- Bidirectional RNNs

Attention

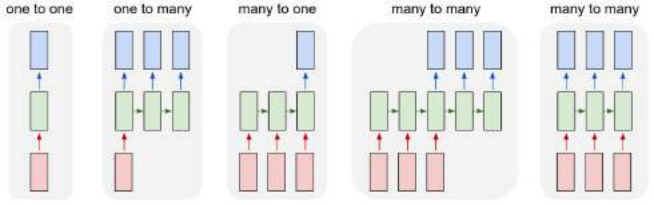
- Machine translation example

Autoregressive models

- WaveNet

Recurrent Neural Networks

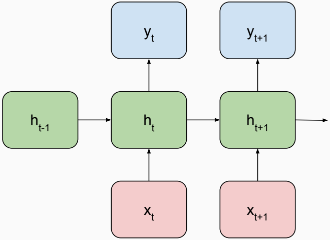
- Feedforward / MLP networks are constrained to a fixed size input and output, and fixed number of hidden layers
- Recurrent Neural Networks (RNNs) are designed to handle sequential data
- They allow flexibility in the lengths of inputs and outputs
- Similar to ConvNets, they also use weight sharing for learning features across the sequence



Flexibility of RNN architectures. Red: inputs, green: hidden states, blue: outputs.

Recurrent Neural Networks

Basic RNN computation for inputs $x_t \in \mathbb{R}^{n_{in}}$ and outputs $y_t \in \mathbb{R}^{n_{out}}$:



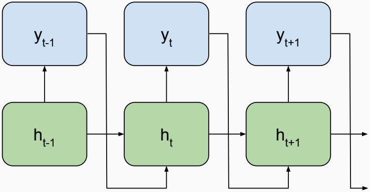
$$
\begin{aligned}
h_t &= \sigma(W^{(hh)} h_{t-1} + W^{(xh)} x_t + b_h), \\
y_t &= W^{(hy)} h_t + b_y,
\end{aligned}
$$

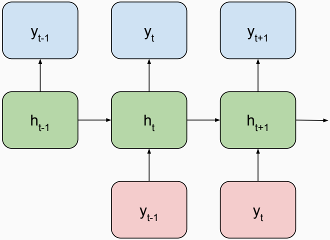
where σ is an activation function, $h_t \in \mathbb{R}^{n_h}$ is the hidden state, $W^{(hh)} \in \mathbb{R}^{n_h \times n_h}$, $W^{(xh)} \in \mathbb{R}^{n_h \times n_{in}}$, $W^{(hy)} \in \mathbb{R}^{n_{out} \times n_h}$, $b_h \in \mathbb{R}^{n_h}$ and $b_y \in \mathbb{R}^{n_{out}}$.

The output could also be passed through e.g. a softmax layer.

Recurrent Neural Networks

We can also wire the RNN to send the output back as the input to the next time step:





This type of architecture can be used for unsupervised sequence modelling (e.g. music generation, language models).

To train this type of RNN, we shift the sequence by one to obtain the target sequence.

RNN properties

- In theory, RNNs model sequences using all information from the past (cf. Markov models)
- RNNs use a distributed hidden state that allows them to store a lot of information (cf. HMMs)
- Nonlinear transformations allow them to update the hidden state in complicated ways
- The internal dynamics of the RNN is deterministic (although the output can be made to be stochastic)
- Models can be made more powerful by stacking extra hidden layers
- Note that there is the issue of the initial hidden state: in practice the initial state can either be learned in the same way as the weights, or simply set to the zero vector

Backpropagation through time

- We can think of the RNN as a layered, feedforward network with shared weights
- Training RNNs also uses the backpropagation algorithm
- In the case of RNNs, we can think of the forward and backward passes stepping through time
- After the backward pass we add the derivatives at all the different times for each weight to preserve weight sharing:

$$\begin{aligned} \text{To constrain:} \quad & w_1 = w_2 \\ \text{We need:} \quad & \Delta w_1 = \Delta w_2 \\ \text{Compute:} \quad & \frac{\partial L}{\partial w_1} \text{ and } \frac{\partial L}{\partial w_2} \\ \text{Use:} \quad & \frac{\partial L}{\partial w_1} + \frac{\partial L}{\partial w_2} \text{ for both } w_1 \text{ and } w_2 \end{aligned}$$

Backpropagation through time

- In practice, training sequences may be very long and/or be of different lengths
- It may be necessary to truncate the sequence lengths used for training (truncated backpropagation through time)
  - Longer sequences are split into shorter subsequences for training
  - The internal RNN state can be carried over between subsequences of a full sequence
  - It is also possible to separate the number of steps in the forward and backward pass
- Shorter sequences could be zero padded to fit into a minibatch tensor for training

Vanishing and exploding gradients

- Recall from the backpropagation calculation that gradients can vanish or explode backwards through the layers[1]:

$$\delta_i = \left(\prod_{k=i}^{N-1} \Sigma'(\hat{h}_k)(W^{(k)})^T\right) \Sigma'(\hat{h}_N) \nabla_{h_N=\mathbf{y}} L.$$

- This problem is especially bad in recurrent networks that are trained on long sequences (e.g. 100 time steps)
- Good weight initialisation can mitigate this to an extent
- In general, RNNs struggle with long-range dependencies

[1] [Hochreiter, 1991]

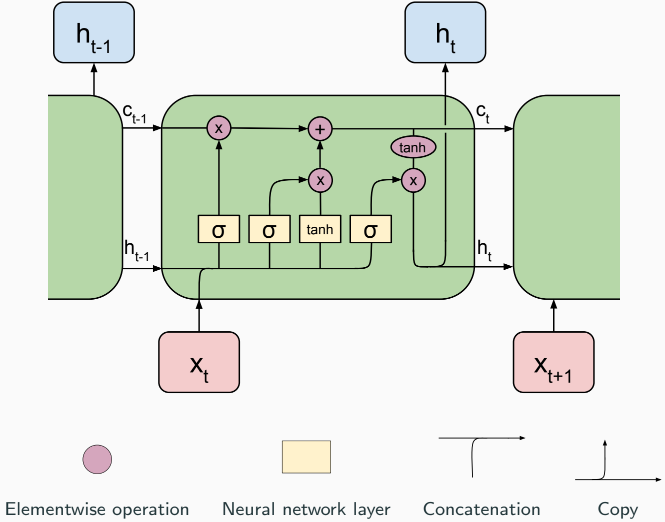
Long Short Term Memory

- The Long Short Term Memory (LSTM) network was introduced[2] to treat the problem of vanishing gradients and enable the network to remember things for a long time
- The LSTM cell has inputs x_t and h_{t-1} and calculates h_t as before
- However, it also includes an internal cell state c_t that allows the unit to store and retain information
- It uses a gating mechanism consisting of logistic and linear units with multiplicative interactions:
    - Information is allowed into the cell state when the 'write' gate is on
    - Information stays in the cell state when the 'keep' gate is on
    - Information can be read from the cell state when the 'read' gate is on

[2] [Hochreiter and Schmidhuber, 1997]

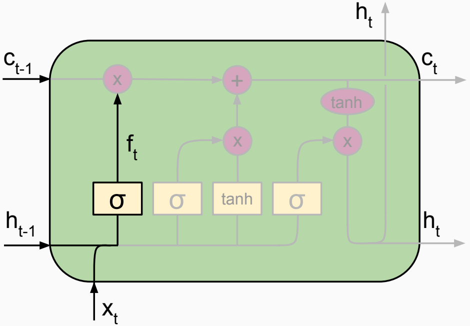
Long Short Term Memory

Schematic diagram for the LSTM unit:



Long Short Term Memory

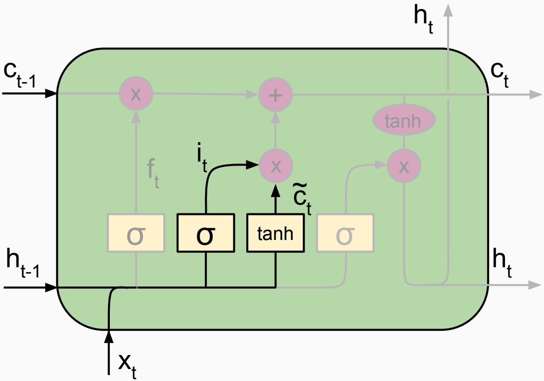
- The LSTM unit can be understood as a combination of gating mechanisms



- The *forget* gate determines what should be erased from the cell state:

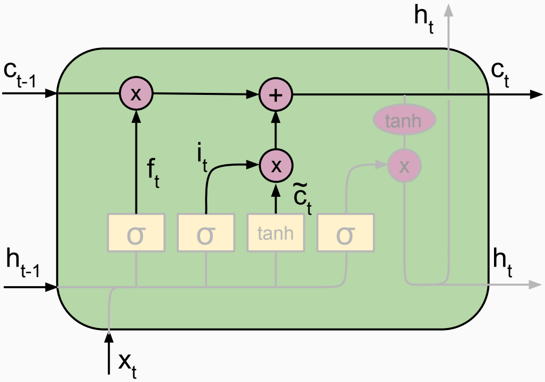
$$f_t = \sigma(W^{(f)}.[x_t, h_{t-1}] + b_f)$$

Long Short Term Memory



- The *input* gate decides which values to update in the cell state, with the candidate content given by $\tilde{c}_t$:

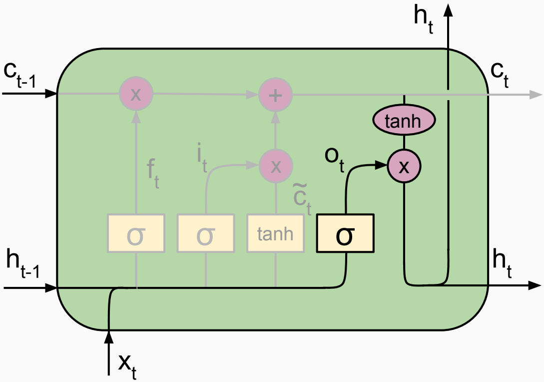
$$
\begin{aligned}
i_t &= \sigma(W^{(i)}.[x_t, h_{t-1}] + b_i) \\
\tilde{c}_t &= \tanh(W^{(c)}.[x_t, h_{t-1}] + b_c)
\end{aligned}
$$



- The cell state is then updated using the output of the forget, input and content gates:

$$c_t = c_{t-1} \odot f_t + i_t \odot \tilde{c}_t$$

Long Short Term Memory



- Finally, the *output* gate decides which cell state values should be output in the hidden state:

$$
\begin{aligned}
o_t &= \sigma(W^{(o)}.[x_t, h_{t-1}] + b_o) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

LSTM Shakespeare

```
PANDARUS:
Alas, I think he shall be come approached and the day When little
srain would be attain'd into being never fed, And who is but a
chain and subjects of his death, I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul, Breaking and
strongly should be buried, when I perish The earth and thoughts
of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and my fair nues begun
out of the fact, to be conveyed, Whose noble souls I'll have the
heart of the wars.
```

LSTM algebraic geometry

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

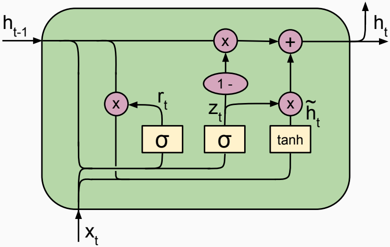
Proof. We will use the property we see that $\mathfrak{p}$ is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Gated Recurrent Unit

The Gated Recurrent Unit (GRU)[3] is a variation on the same idea. It combines the forget and input gates into a single 'update gate'. It also merges the cell state and hidden state.



$$
\begin{aligned}
z_t &= \sigma(W^{(z)}.[x_t, h_{t-1}] + b_z) \\
r_t &= \sigma(W^{(r)}.[x_t, h_{t-1}] + b_r) \\
\tilde{h}_t &= \tanh(W.[x_t, r_t \odot h_{t-1}] + b) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
$$

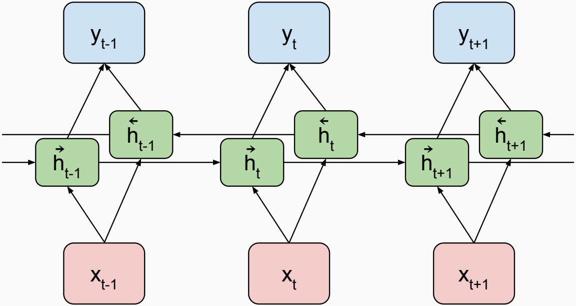
[3] [Cho et al., 2014]

Bidirectional Recurrent Networks

- Standard RNN architectures are uni-directional, and only use information from the past to make predictions
- For certain tasks it makes sense to provide information from the future as well, e.g. in a language model:

`Hello, how ____ you?`

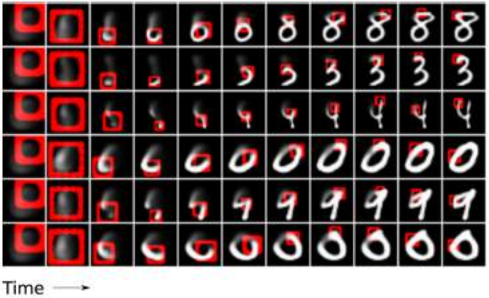
- Bidirectional RNNs consist of forward and backward RNNs whose states are combined to make predictions



Outline

Attention

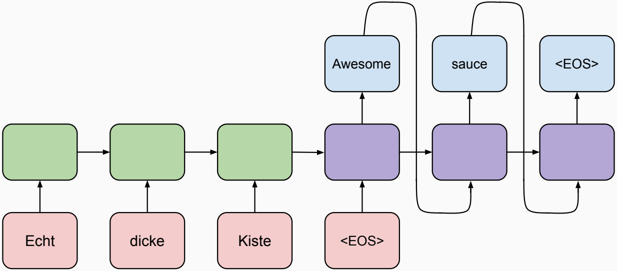
- Attention mechanisms allow neural networks to choose where they focus in the data in order to accomplish certain tasks
- Generally, attention mechanisms can be thought of as using *query, key* and *value* vectors
- These vectors serve to generate a *context* vector that can be fed to the network to help with making predictions



The DRAW network [Gregor et al., 2015] generating MNIST digits. The red rectangle shows the area attended to by the network

Machine translation

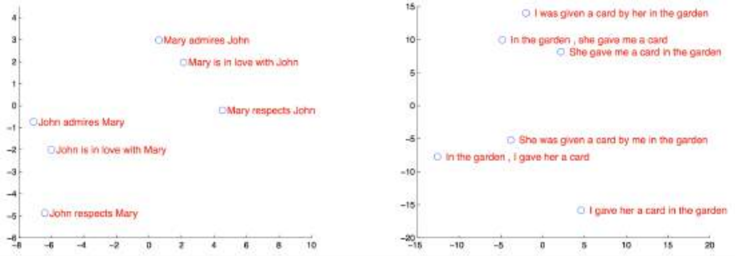
- As an example, consider machine translation. A typical architecture for machine translation is the RNN encoder-decoder:



- The encoder network generates a sequence of hidden states from a source sentence
- The decoder network needs to translate the sentence based on the final hidden state of the encoder

Machine translation

- The encoder RNN needs to encode the entire source sentence into the final hidden state
- This hidden state can be thought of as a sentence *embedding*:



2D PCA projections of sentence embeddings [Sutskever et al., 2014]

- For long sentences, it is difficult for the encoder to store everything in the final hidden state

Attention mechanism

- An attention mechanism allows the decoder to attend to different parts of the source sentence at each step of the translation
- The model can learn what to attend to
- At each step i of the decoder translation, an alignment model a computes scores $e_{ij} \in \mathbb{R}$ between the most recent decoder hidden state s_{i-1} and each encoder hidden state h_j:

  $$e_{ij} = a(s_{i-1}, h_j), \qquad j = 1, \ldots, T_x,$$

  where T_x is the length of the source sentence
- The alignment model a could be an inner product, or a (learned) neural network

Attention mechanism

- The alignment scores e_{ij} $(j = 1, \ldots, T_x)$ are then used to compute a distribution over the encoder hidden states using a softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

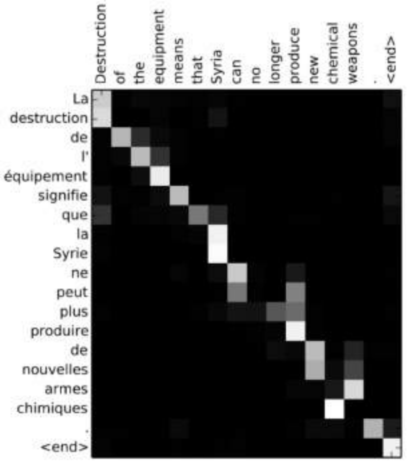
- The current context is then a weighted sum of the hidden states h_j:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

- c_i is provided to the decoder at step i to help make its prediction

Attention - interpretation

- One advantage of attention is that it allows us to inspect and interpret the model behaviour:



Visualisation of attention weights α_{ij} in a French to English translation task

Outline

Autoregressive models

- Autoregressive models represent another general deep learning approach to sequence modelling
- Instead of modelling temporal dynamics through an internal hidden state, the approach is to cast the joint distribution as a product of conditional distributions

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

- Autoregressive models are highly expressive networks aimed at modelling the highly nonlinear and long-range correlations in the data

Autoregressive models

Some advantages of autoregressive generative models:

- They provide a direct way to calculate exact likelihood. This is in contrast to other classes of generative models such as GANs or VAEs.
- The training is stable. The GAN objective in particular is defined as a minimax game, and models are prone to training problems.
- AR models can be applied to both discrete and continuous data.

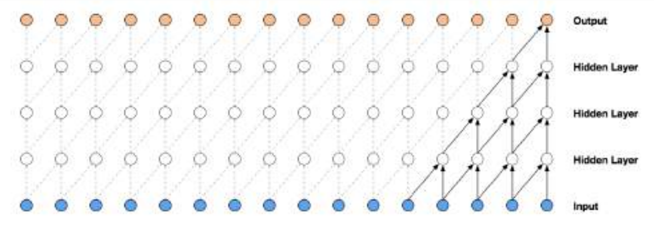
WaveNet

- We will look at at autoregressive model that has become very popular in audio modelling called **WaveNet**[4]
- The architecture is inspired by earlier work on PixelRNN and PixelCNN[5]
- It is a generative model of raw audio signals
- WaveNet can be thought of as a 1D version of PixelCNN (which was developed as a generative model of images)
- The model makes use of **masked/causal** convolutions to make sure the autoregressive property is respected
- The convolutions are **dilated** to increase the receptive field

[4] [van den Oord et al., 2016a]
[5] [van den Oord et al., 2016b]

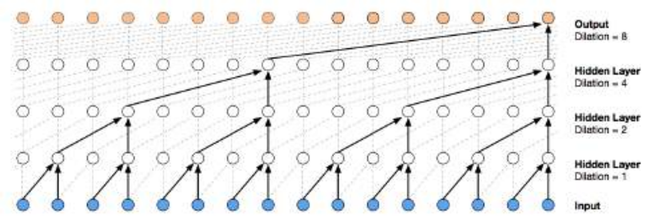
WaveNet - causal convolutions

- WaveNet does not include any pooling layers, and the output has the same size as the input (in the time dimension)
- The model outputs a categorical distribution over the (quantised) next value x_t with a softmax
- Causal (or masked) convolutions enforce the autoregressive property
- Note that at training time, predictions can be made for all timesteps in parallel. At generation time, predictions are sequential



WaveNet - dilated convolutions

- Note there are no recurrent connections - this makes the model faster to train than an RNN, especially on long sequences
- However, many layers or large filters are required to increase the receptive field (NB the audio has a sample rate of 16kHz)
- Dilated convolutions increase the receptive field by orders of magnitude
- In WaveNet, dilations are doubled every layer up to a point and then repeated: e.g. $1, 2, 4, \ldots, 512, 1, 2, 4, \ldots, 512, 1, 2, 4, \ldots, 512$

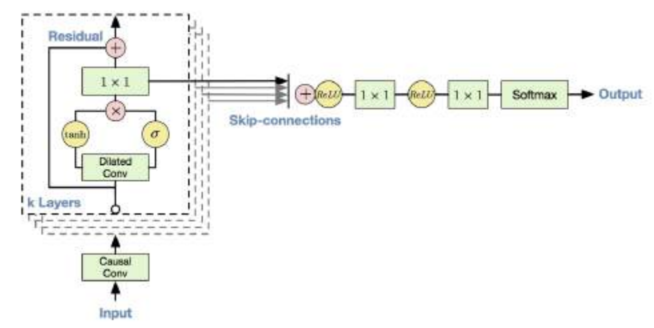


WaveNet - residual and skip connections

- Each layer of the architecture consists of a residual block with a skip connection
- Within this block there is also a gated activation unit:

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x}) \odot \sigma(W_{g,k} * \mathbf{x})$$

- These blocks are stacked many times in the network



WaveNet - conditioning variables

- WaveNet can be conditioned to produce audio with required characteristics (e.g. speaker identity or text input)
- Global conditioning variables $\mathbf{h}$ are implemented in the gated activation unit as linear projections $V_{\cdot,k}$ that are broadcast over the time dimension:

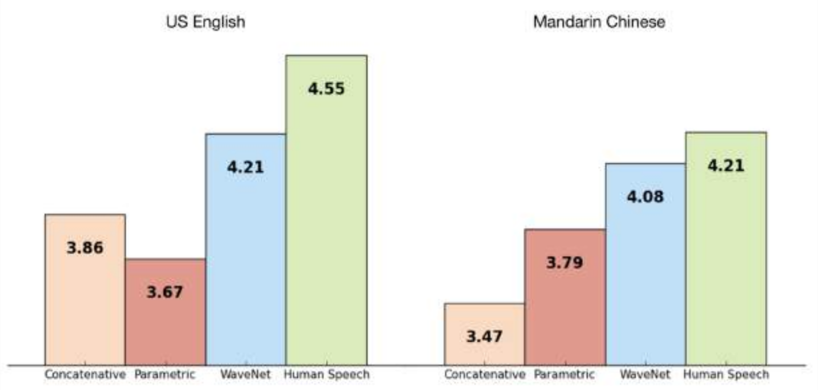
$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k}^T \mathbf{h}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k}^T \mathbf{h})$$

- Similarly, local conditioning variables h_t are first upsampled with transposed convolutions to a new time series $\mathbf{y} = f(\mathbf{h})$ with the same resolution as the original signal and used in the activation unit as:

$$\mathbf{z} = \tanh(W_{f,k} * \mathbf{x} + V_{f,k} * \mathbf{y}) \odot \sigma(W_{g,k} * \mathbf{x} + V_{g,k} * \mathbf{y}),$$

where $V_{\cdot,k} * \mathbf{y}$ is now a 1×1 convolution.

WaveNet - speech synthesis evaluation



Mean opinion scores (MOS) obtained in blind tests with human subjects show the quality of WaveNets on a scale from 1 to 5, compared with previous state-of-the-art text-to-speech (TTS) systems

Bahdanau, D., Cho, K., and Bengio, Y. (2014).
Neural machine translation by jointly learning to align and translate.
arXiv e-prints, abs/1409.0473.

Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014).
Learning phrase representations using rnn encoder–decoder for statistical machine translation.
In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015).
DRAW: A recurrent neural network for image generation.
In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1462–1471.

Hochreiter, S. (1991).
Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural Comput., 9(8):1735–1780.

- Karpathy, A. (2015).
  The unreasonable effectiveness of recurrent neural networks.
  http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

- Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014).
  A clockwork rnn.
  In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1863–1871, Bejing, China. PMLR.

- Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., and Bengio, Y. (2017).
  Samplernn: An unconditional end-to-end neural audio generation model.

Olah, C. (2015).
Understanding lstm networks.
http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Schuster, M. and Paliwal, K. (1997).
Bidirectional recurrent neural networks.
Trans. Sig. Proc., 45(11):2673–2681.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
Sequence to sequence learning with neural networks.
In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

References v

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016a).
Wavenet: A generative model for raw audio.
CoRR, abs/1609.03499.

van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016b).
Conditional image generation with pixelcnn decoders.
CoRR, abs/1606.05328.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017).
Attention is all you need.
In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.