Machine Learning tutorial

Speak! 20th February 2019

Zack Hodari

Overview

1-a	Neural networks	**Part 1**
1-b	Recurrent neural networks	
2-a	Convolutional neural networks	**Part 2**
2-b	Autoregressive models	
3-a	Attention	**Part 3**
3-b	Self-attention	

Part 1

Overview

1-a	**Neural networks**	**Part 1**
1-b	Recurrent neural networks	
2-a	Convolutional neural networks	Part 2
2-b	Autoregressive models	
3-a	Attention	Part 3
3-b	Self-attention	

Some linear algebra

What is a neural network?

Learns some task given data

Requires labels that indicate the true output corresponding to the input

Predicting plant species

Let's use a typical machine learning classification task to detail how TTS is different

Features	Plant 1	Plant 2	...	Plant N
Petal length (cm)	0.87	0.34	...	0.54
Petal width (cm)	2.02	1.87		2.23
...				

↓

MACHINE LEARNING MODEL

↓

Species	Setosa	Versicolour	...	Versicolour

But what is actually happening

species =

weight_1 * petal length +

weight_2 * petal width +

...

weight_D * colour

But what is actually happening

species =

$\quad weight_1$ * petal length +

$\quad weight_2$ * petal width +

$\quad$...

$\quad weight_D$ * colour

species = **w** * [petal length , petal width , ... , colour]

where **w** = [$weight_1$, $weight_2$, ..., $weight_D$]

But what is actually happening

species =

$\quad weight_1$ * petal length +

$\quad weight_2$ * petal width +

$\quad$ …

$\quad weight_D$ * colour

species = **w** * **x**

where **w** = [$weight_1$, $weight_2$, …, $weight_D$]

where **x** = [petal length , petal width , …, colour]

But what is actually happening

$\mathbf{y}$ = [setosa , versicolour] =

$[\ \text{weight}_{1,1}\ ,\ \text{weight}_{1,2}\ ]$ * petal length +

$[\ \text{weight}_{2,1}\ ,\ \text{weight}_{2,2}\ ]$ * petal width +

…

$[\ \text{weight}_{D,1}\ ,\ \text{weight}_{D,2}\ ]$ * colour

$\mathbf{y} = \mathbf{W} * \mathbf{x}$

where $\mathbf{W} = [\ \text{weight}_{1,1}\ ,\ \text{weight}_{1,2}\ , \dots,\ \text{weight}_{D,2}\ ]$

where $\mathbf{x}$ = [petal length , petal width , …, colour]

But what is actually happening

y = **W** * **x**

where **W** = [$weight_{1,1}$, $weight_{1,2}$, ..., $weight_{D,2}$]
where **x** = [petal length , petal width , ..., colour]

Feature vector **x** has D values

Weight vector **W** has D * 2 values

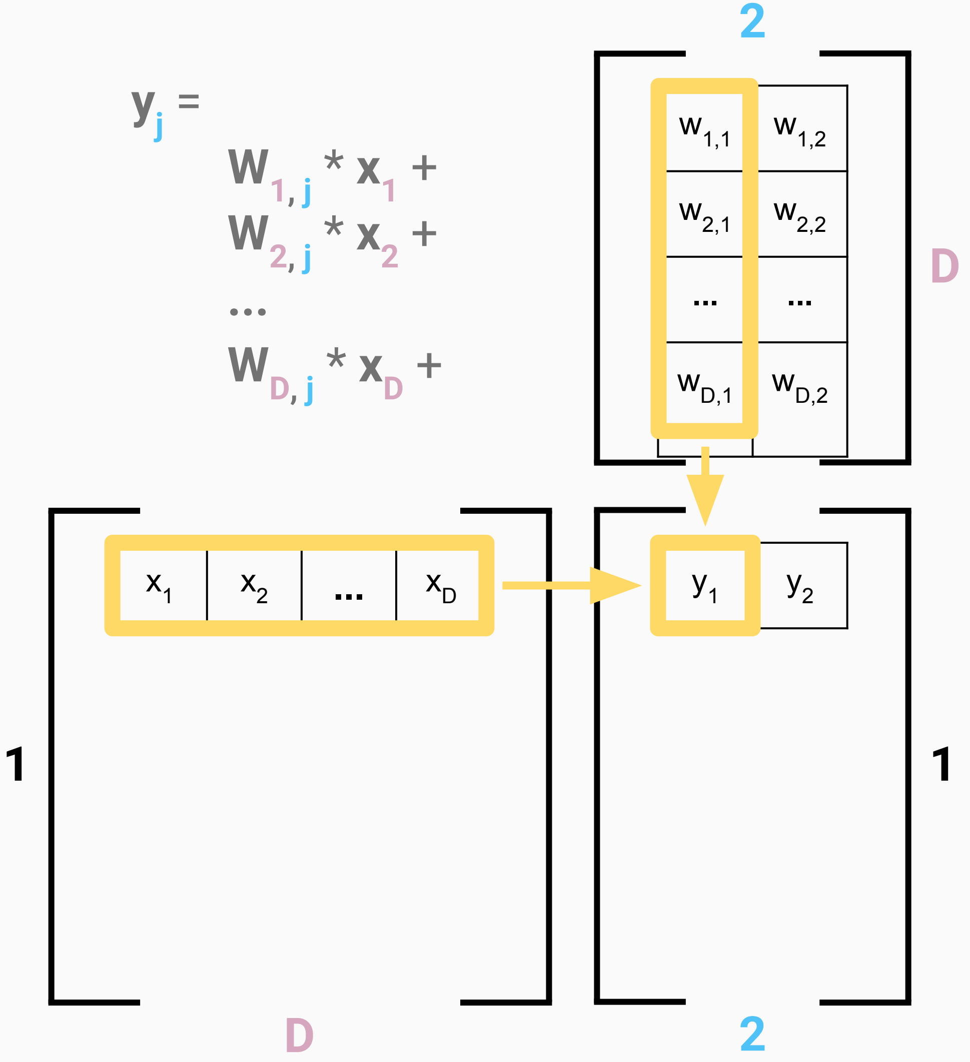
Prediction vector **y** has 2 values

Some linear algebra

$$x \quad . \quad W \quad = \quad y$$

$$(D) \quad . \; (D\,,2) = \quad (2)$$

$$y_j = W_{1,j} * x_1 + W_{2,j} * x_2 + \dots W_{D,j} * x_D +$$



Some linear algebra

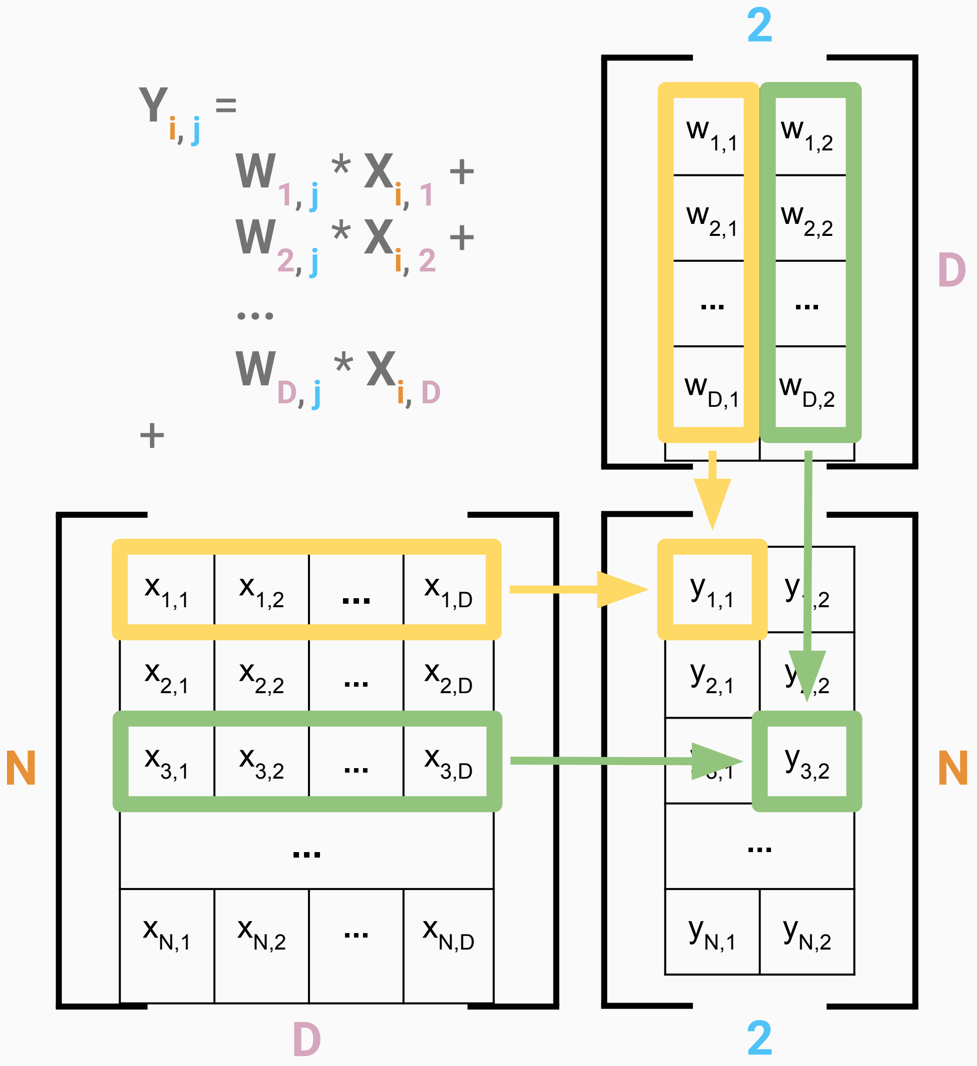
$$X \;.\; W = Y$$

$$(N, D) \;.\; (D, 2) = (N, 2)$$

This is matrix multiplication

The operation is on the rows in X and the columns in W, but it is more important to ***remember how shapes cancel***

$$Y_{i,j} = W_{1,j} * X_{i,1} + W_{2,j} * X_{i,2} + \dots W_{D,j} * X_{i,D} +$$



Some linear algebra

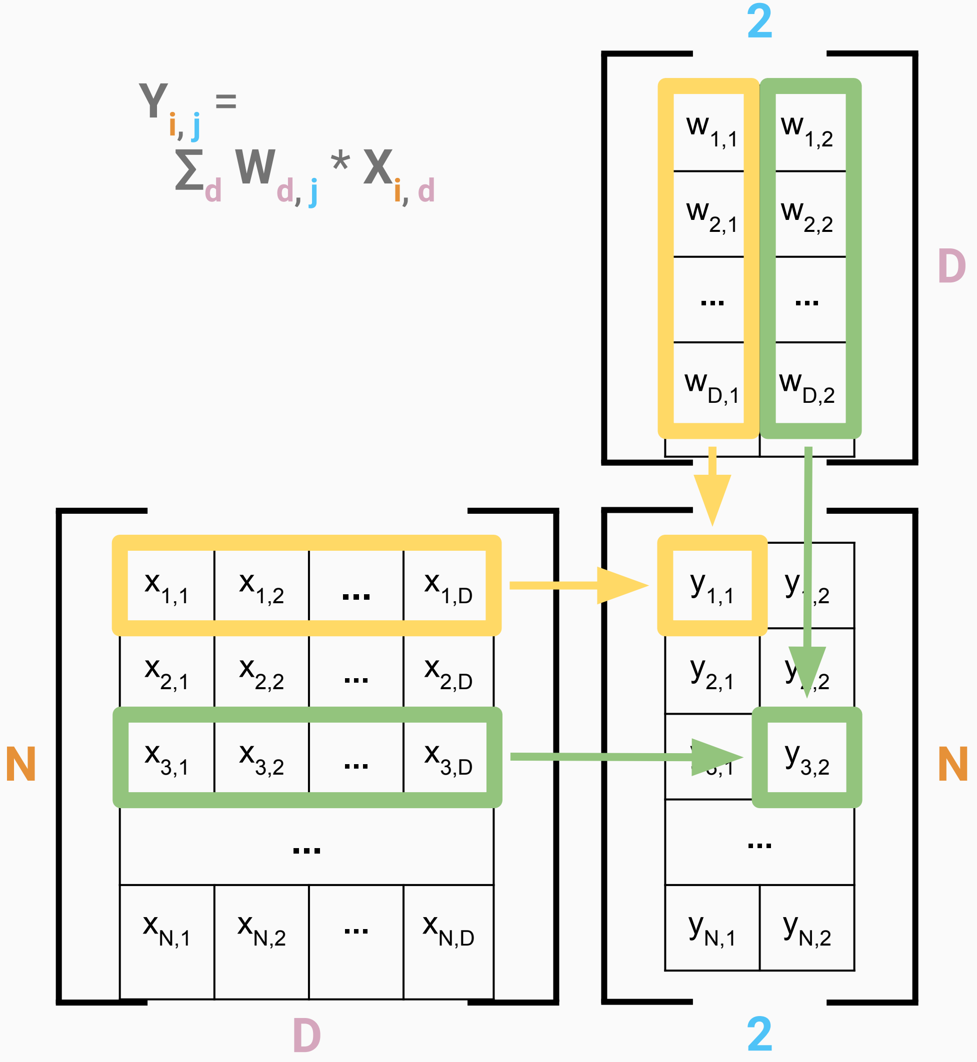
X . W = Y

(N , D) . (D , 2) = (N , 2)

This is matrix multiplication

The operation is on the rows in X and the columns in W, but it is more important to ***remember how shapes cancel***

$$Y_{i,j} = \sum_d W_{d,j} * X_{i,d}$$



Acoustic model

Can we just plug in our speech?

For linguistic features **X** of shape (F, L)

For a weight vector **W** of shape (L, A)

For acoustic features **Y** of shape (F, A)

F is number of frames in the sentence
L is the dimensionality of the linguistic labels
A is the dimensionality of the acoustic frames

Prediction for one acoustic frame:

X . **W** = **Y**

(F , L) . (L , A) = (F , A)

We can!

There are multiple acoustic frames in a sentence

Each frame is equivalent to the feature descriptor of a single flower example

Use feedforward neural networks repeatedly on each frame of speech

We perform each frame prediction independently

Overview

1-a	Neural networks	**Part 1**
1-b	**Recurrent neural networks**	
2-a	Convolutional neural networks	Part 2
2-b	Autoregressive models	
3-a	Attention	Part 3
3-b	Self-attention	

Recurrent cells

But acoustic frames are not independent

Ideally we want a model that can take into account this dependence

RNNs learn a *state* that aims to track relevant information

Explanation of RNNs, GRU cells, and LSTM cells:

colah.github.io/posts/2015-08-Understanding-LSTMs/

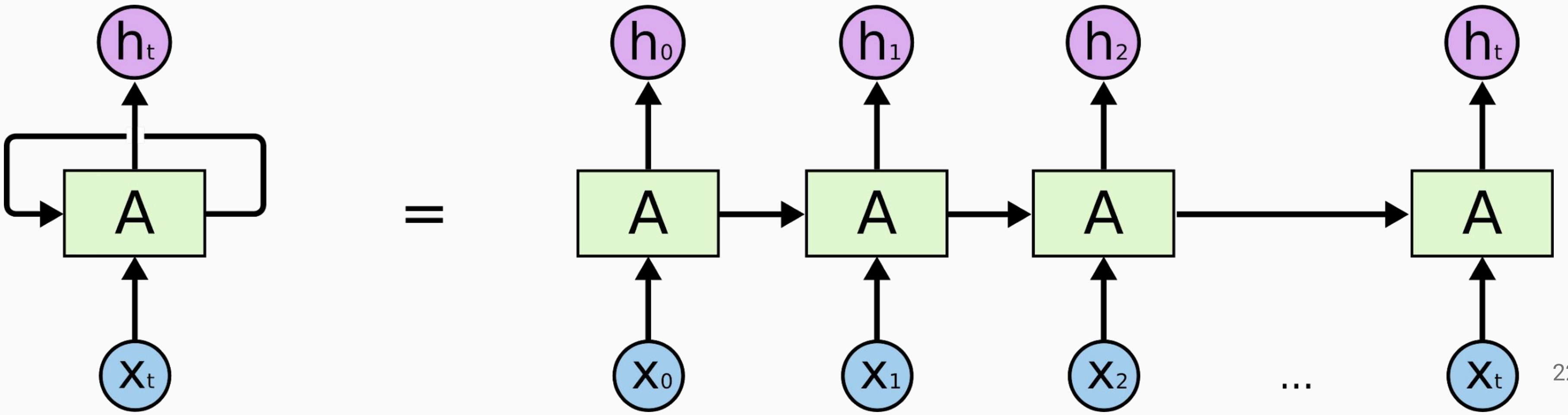
Examples of RNNs for simple tasks and demonstration code:

karpathy.github.io/2015/05/21/rnn-effectiveness/

But acoustic frames are not independent

Ideally we want a model that can take into account this dependence

RNNs learn a *state* that aims to track relevant information



“Designed to forget”

RNNs try to remember everything incrementally

This is done by adding to a single state vector

For LSTMs we have a forget gate which allows us to free up “space” in our vector

Leads to new architectures:

- all-convolutional model, self-attention, encoder-decoder with attention

Part 2

Overview

1-a	Neural networks	Part 1
1-b	Recurrent neural networks	
2-a	**Convolutional neural networks**	**Part 2**
2-b	Autoregressive models	
3-a	Attention	Part 3
3-b	Self-attention	

Image processing

Image processing background

Convolutions are used in image processing

Useful for removing noise, extracting edges, and much more!

A kernel is defined, this is what performs our desired operation

Edge extraction

The Laplacian filter is a classic preprocessing step for edge extraction

We define the following kernel

$$W = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

But how do we convolve?

$$\boxed{\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \boxed{\Sigma} \begin{bmatrix} 0 & 0 & 0 \\ 1 & -4 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

But how do we convolve?

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -2 \end{bmatrix}$$

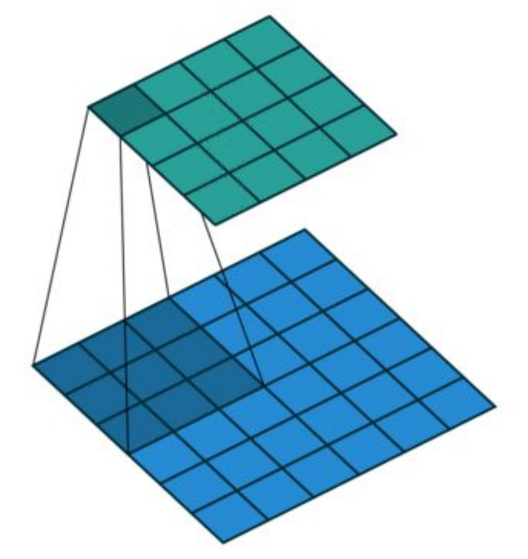
But how do we convolve?

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
=
\begin{bmatrix}
-2 & -2 & -2 & -1 \\
1 & 1 & 3 & -3 \\
0 & 2 & -4 & 2 \\
2 & -4 & 2 & 0
\end{bmatrix}
$$

But how do we convolve?

Excellent visualisations!
github.com/vdumoulin/conv_arithmetic

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} =$$



extra – Convolution as matrix multiplication

We are calculating the sum of products

X	*	K	=	Y
(3 x 3)	*	(3 x 3)	=	(1)

X	.	K	=	Y
(9)	.	(9)	=	(1)

extra – Convolution as matrix multiplication

We are calculating the sum of products

This can be formulated as a matrix multiplication if we reshape

X	*	K	=	Y	im2col(X)	.	K	=	Y
(6 x 6)	*	(3 x 3)	=	(4 x 4)	(4 x 4 x 9)	.	(9)	=	(4 x 4)

Learning the convolution kernel

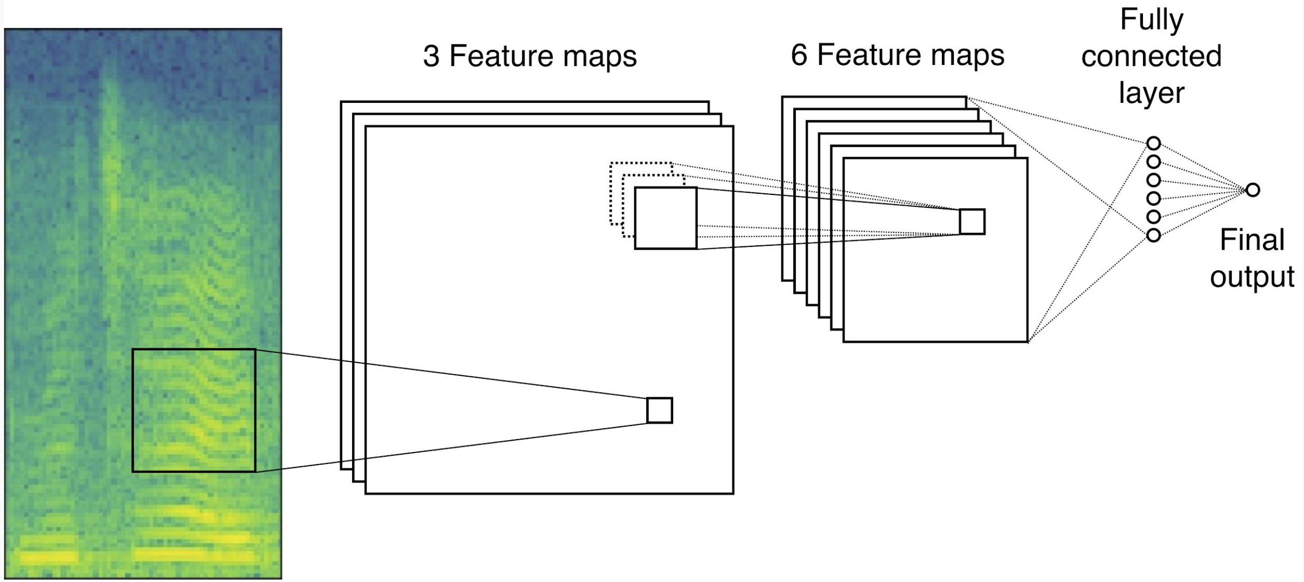
i.e. CNNs

Why don’t we learn the operation?

Hand-crafting kernels to extract features is not easy

A convolutional neural network (layer) learns its kernel(s)

Why don't we learn the operation?



Learn information from a spectrogram

Later features maps can represent higher level information

From this information we can classify something like phoneme identity

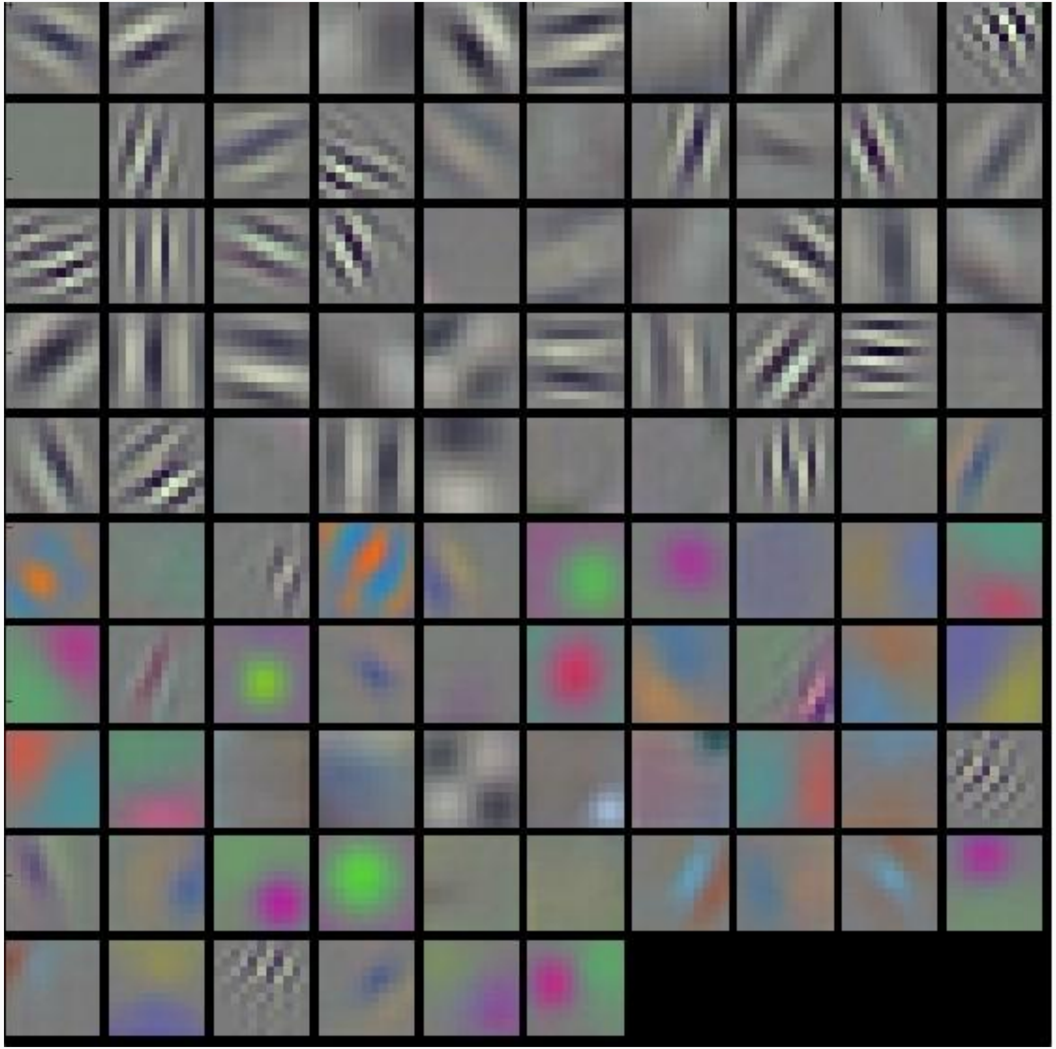
Examples of learned kernels

Kernels extract features such as

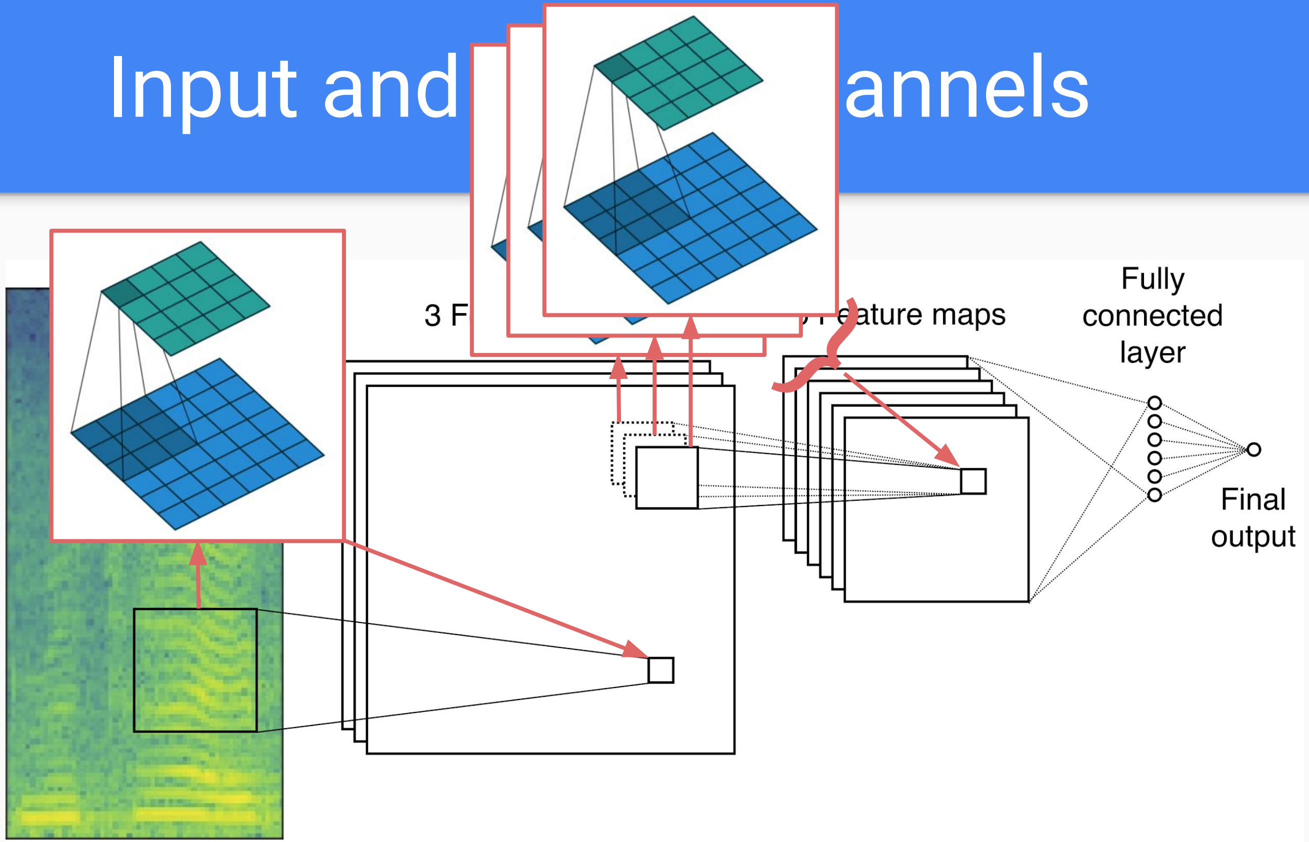
- edges
- patterns
- shapes
- objects?

cs231n.github.io/convolutional-networks/

colah.github.io/posts/2014-07-Understanding-Convolutions/



Input and annels



If we have **N** input channels (or feature map), then instead of one kernel we have **N** kernels

Each of these **N** kernels are convolved with their respective input channel, and their result is summed to create one output channel

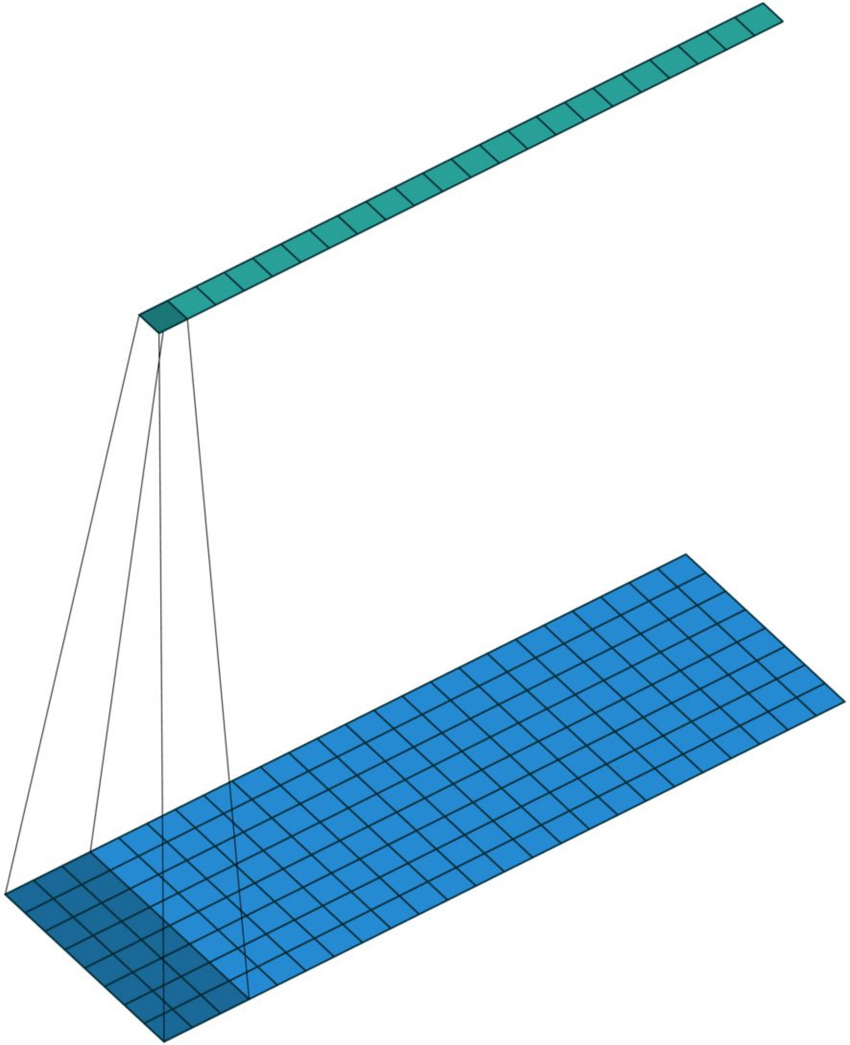
Convolution variants

e.g. 1-d, 1x1, dilated, causal, transposed

1-dimensional convolutions

Refers to the shape of the input

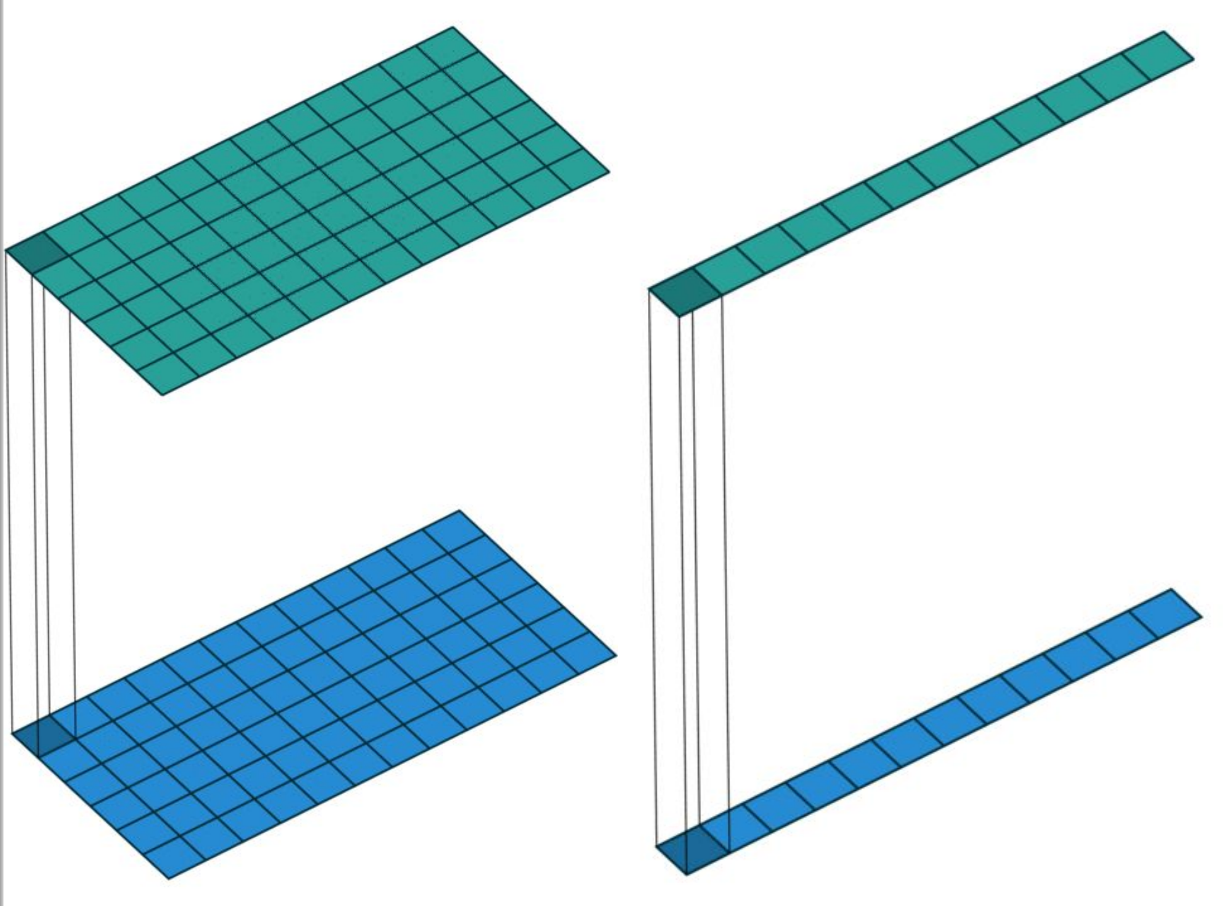
This is what we refer to when we use CNNs on sequence data



1x1 convolutions

Refers to the shape of the kernel (developed for dimensionality reduction)

This is equivalent to a feedforward layer applied to each item



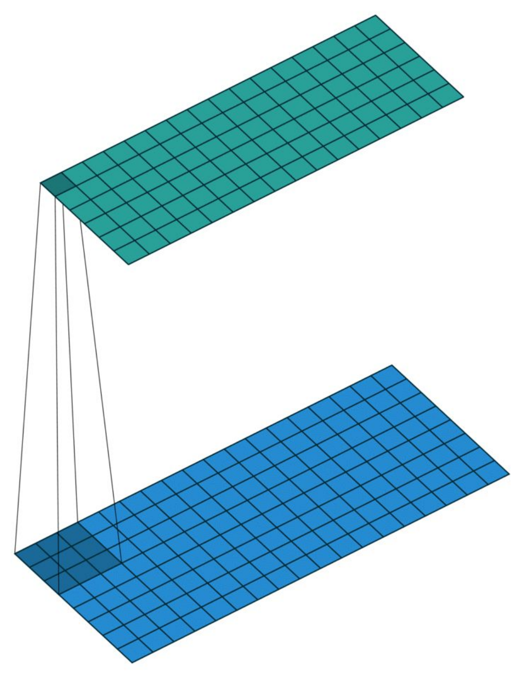
extra – Reducing the number of parameters

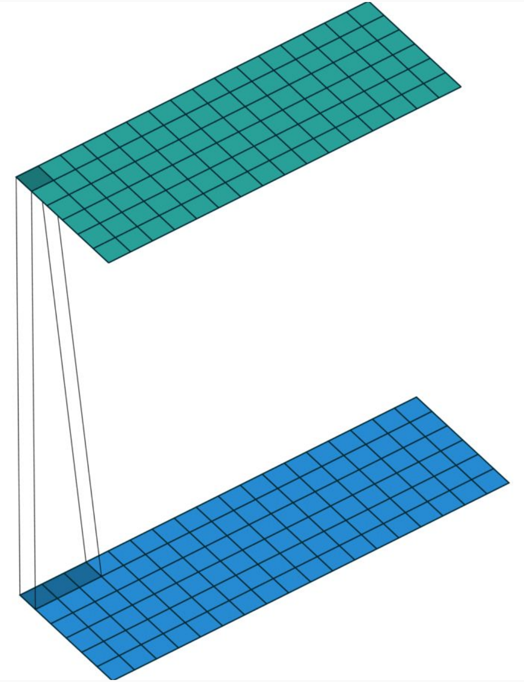
A 3x3 kernel can be replaced with two CNN layers, the first with a 3x1 kernel then a 1x3 kernel. This new architecture contains less parameters: $(3 * 1 + 1 * 3) < 3 * 3$

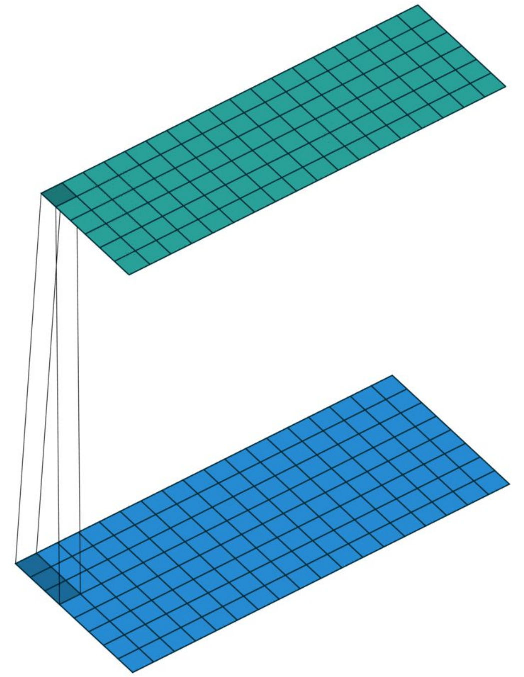
Note that these can be tricky to train, and don't always help

- cs231n.stanford.edu/slides/2016/winter1516_lecture11.pdf (slides 60-62)
- towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728
- towardsdatascience.com/speeding-up-convolutional-neural-networks-240beac5e30f
- keras.io/layers/convolutional/#separableconv1d

extra – Reducing the number of parameters

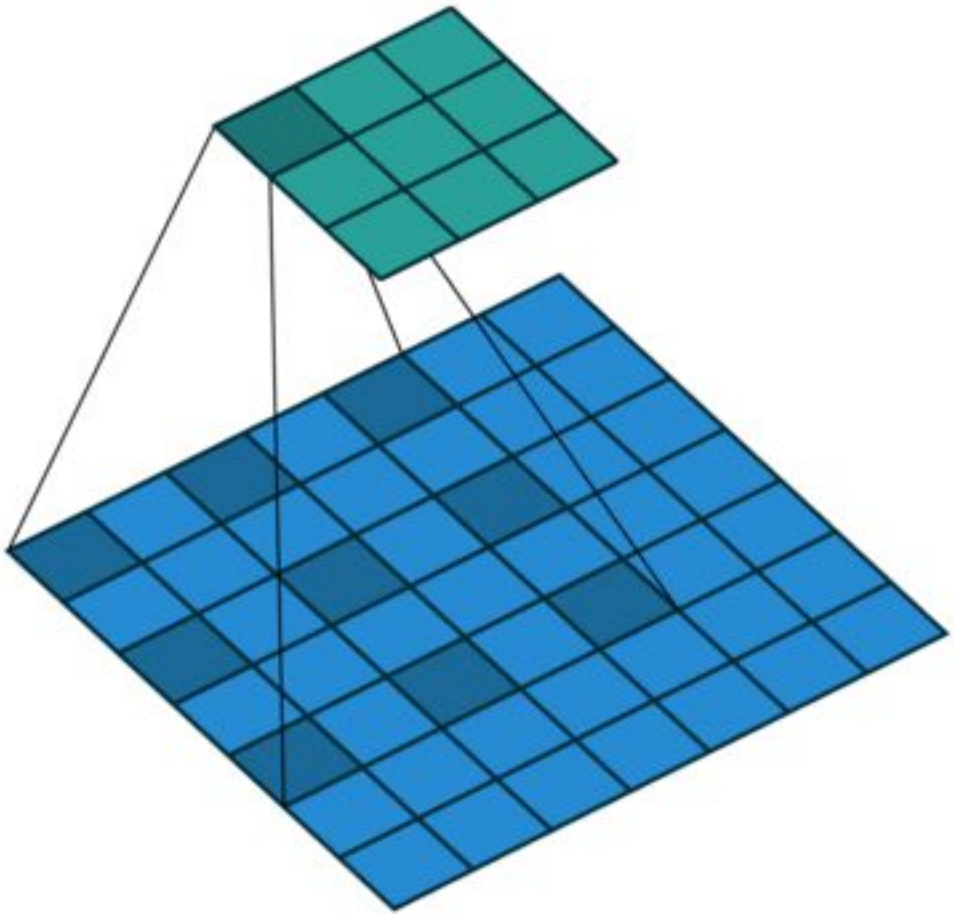




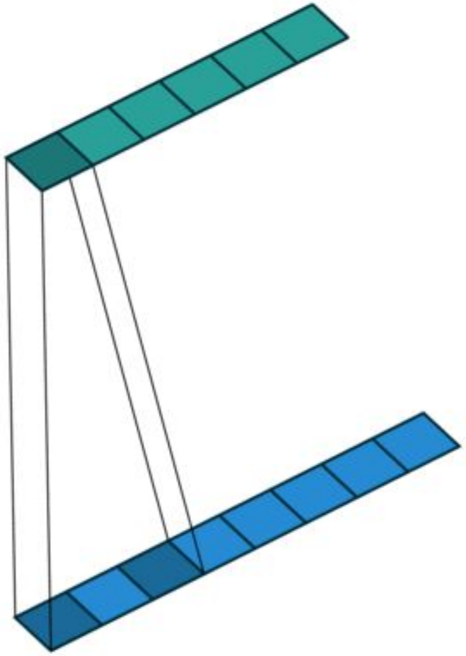


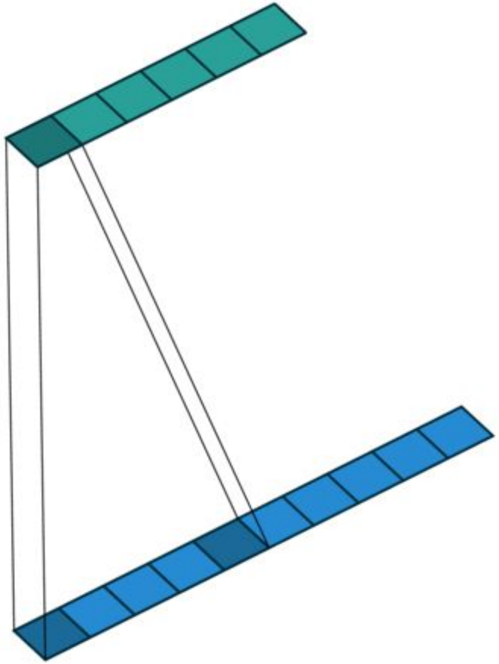
Dilated convolutions

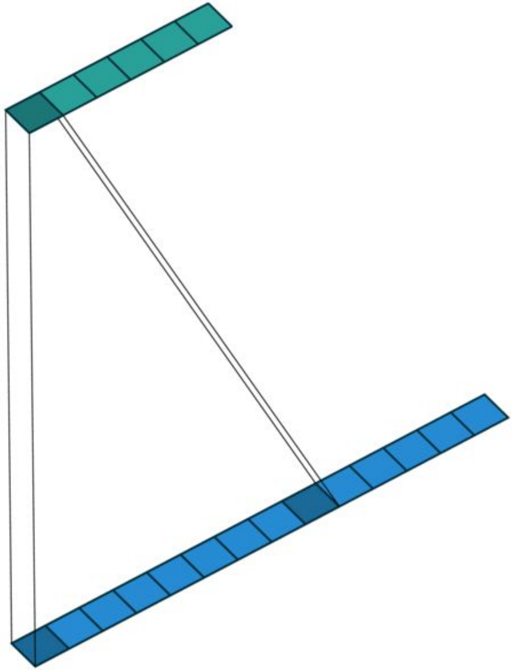
Convolve over a region in the input that is spread out



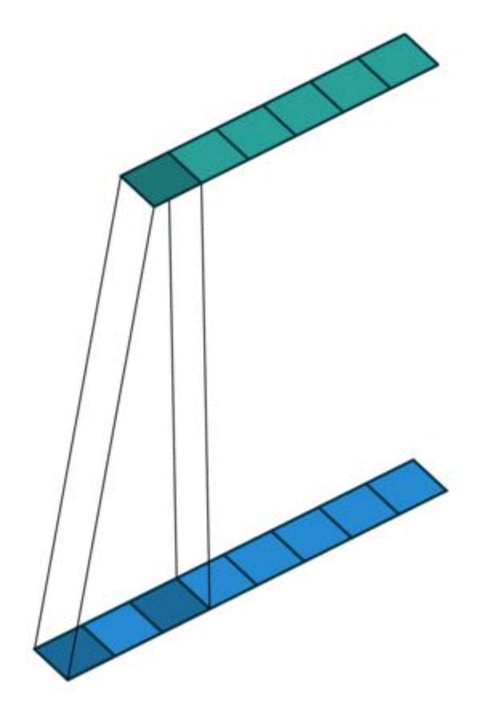
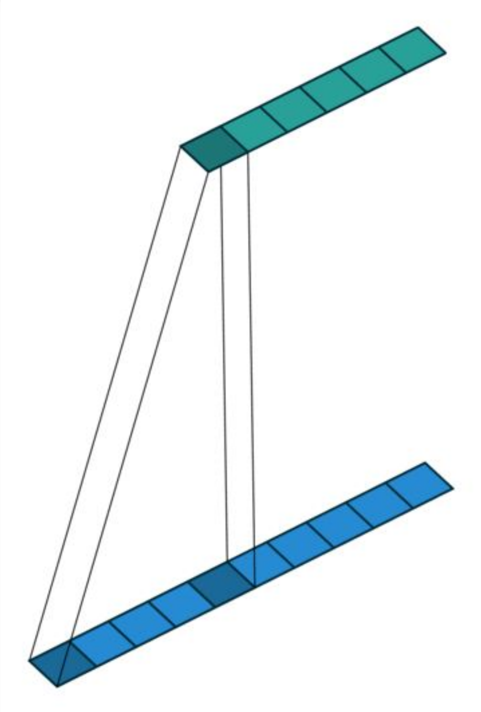
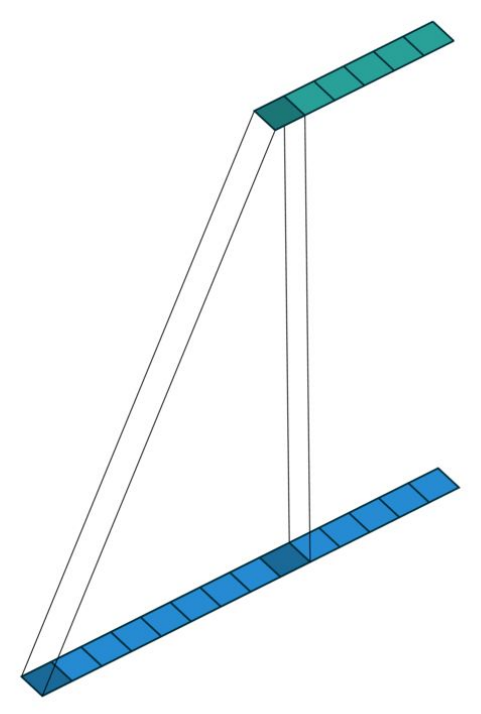
Dilated convolutions



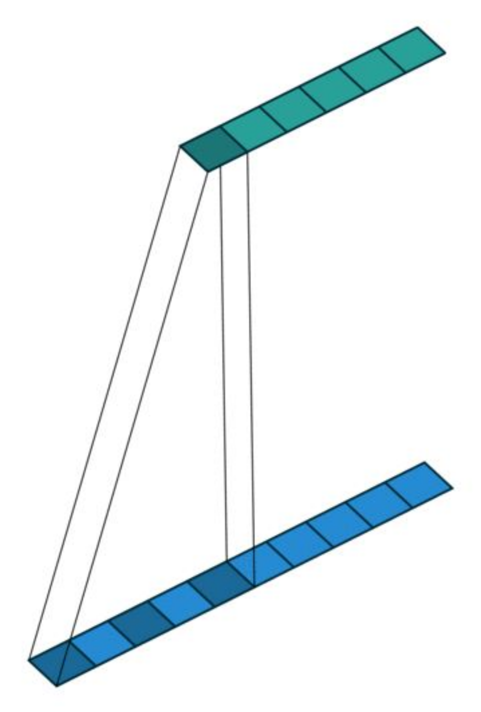


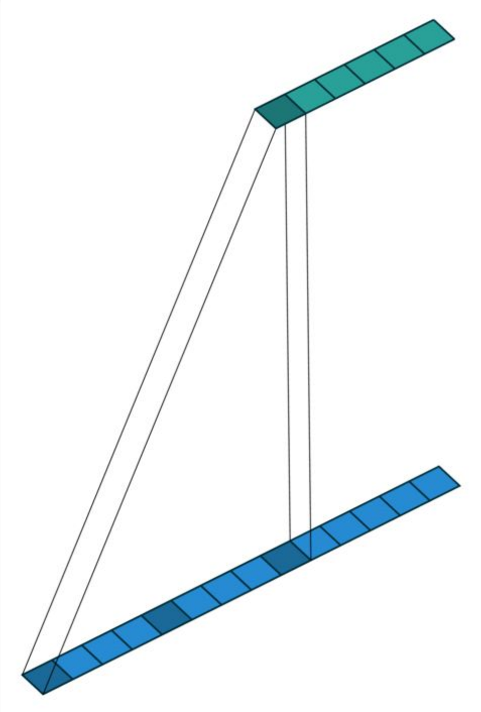


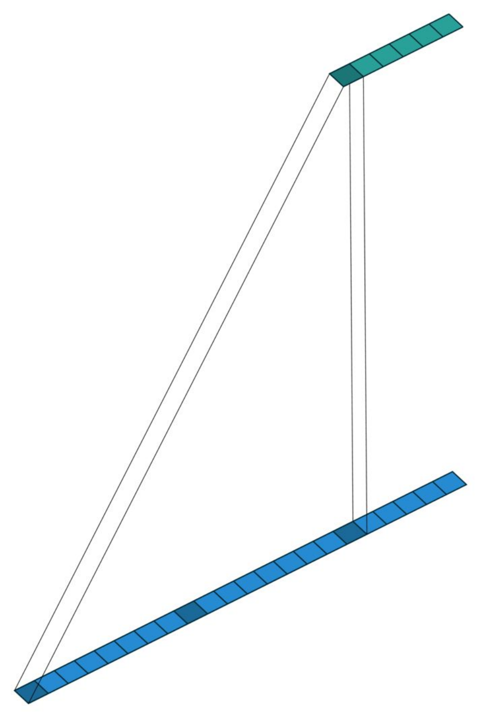
Causal dilated convolutions

Causal dilated convolutions



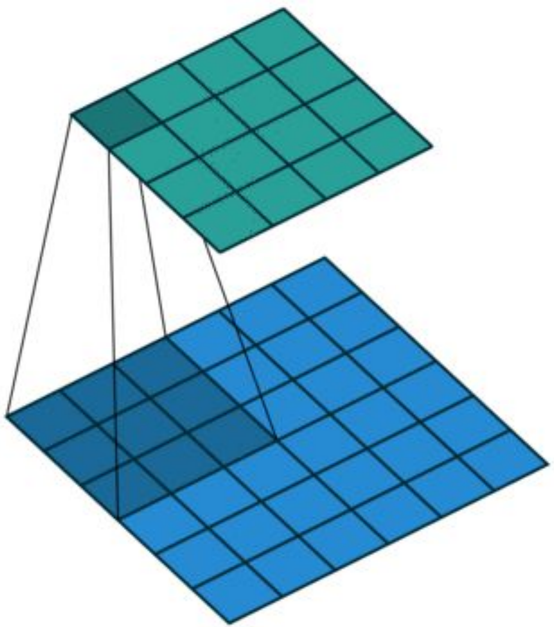


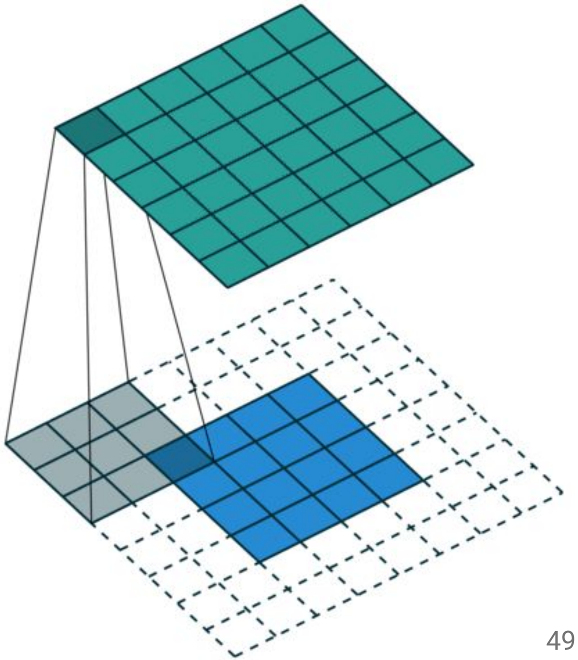


Transposed convolutions

Performs the inverse operation of a normal convolution

Excellent tutorial:
deeplearning.net/software/theano/tutorial/conv_arithmetic.html#transposed-convolution-arithmetic



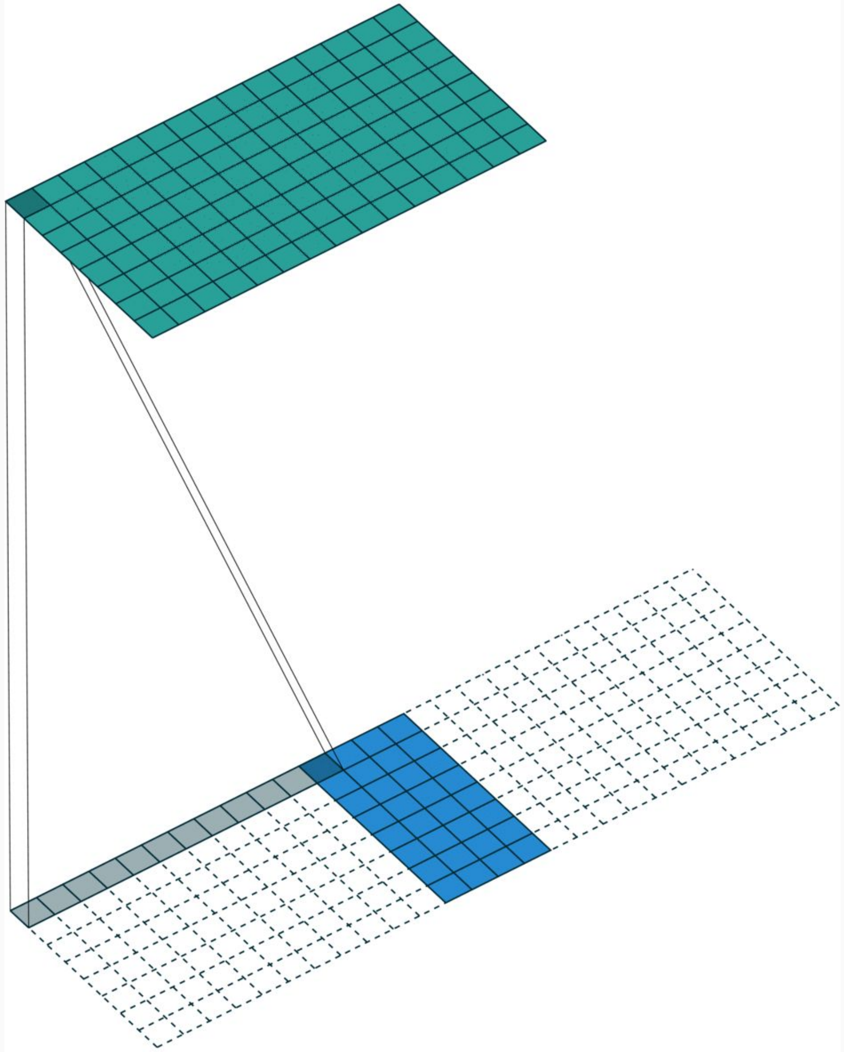


Transposed convolutions

Use as a method to learn to upsample the input

Given a spectrogram at some frame-rate, we can learn to upsample it to some sample-rate

Checkerboard pattern issue:
distill.pub/2016/deconv-checkerboard/



Overview

Population modelling

Autoregressive models

Originally developed to model time-varying processes

e.g. Population modelling of animals

Population modelling

The future population is a function of the current population

$$population_{next\_year} = scalar * population_{this\_year}$$

Population modelling

The future population is a function of the current population

$$p_{t+1} = scalar * p_t$$

$$\boxed{p_{t+1} = f(p_t)}$$

Autoregressive neural networks

Uses the idea of predicting based on the previous value

$$p_{t+1} = f(p_t)$$

To get the new prediction we pass the previous one through our model

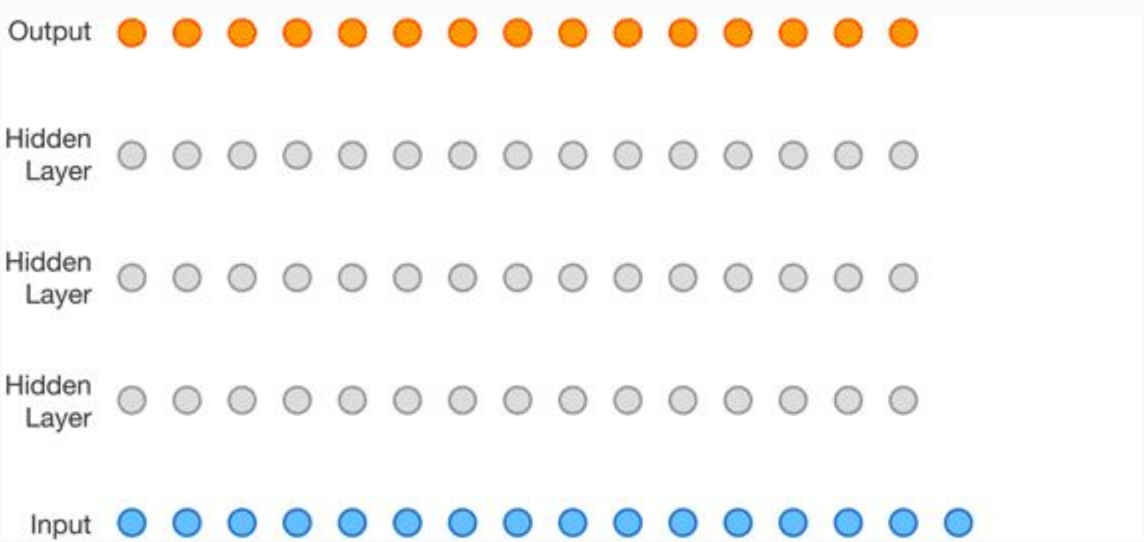
WaveNet

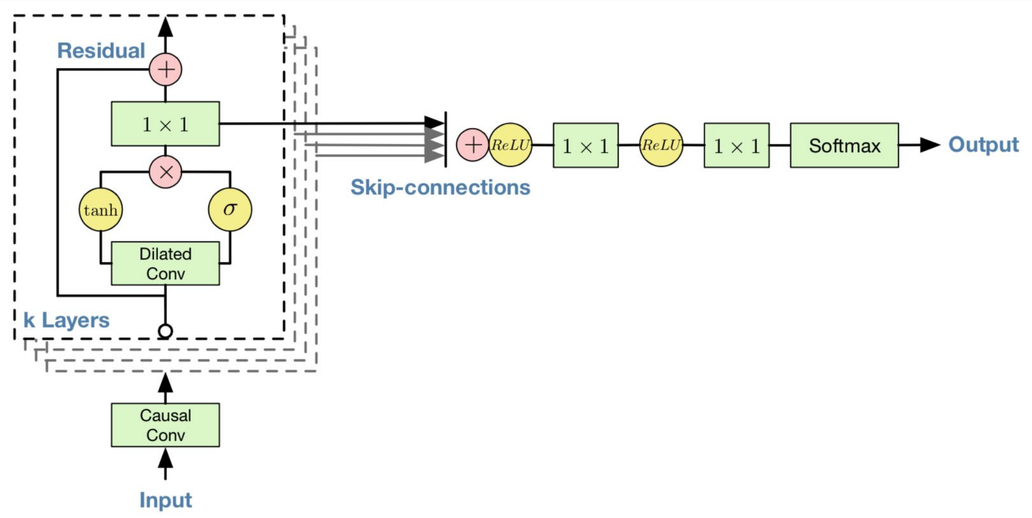
Finally!

WaveNet

You may have seen these before, but try to forget them

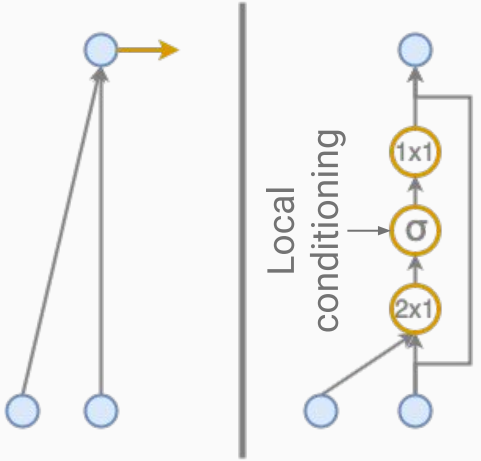
The following diagrams will use similar notation, but will reorganise much of the structure

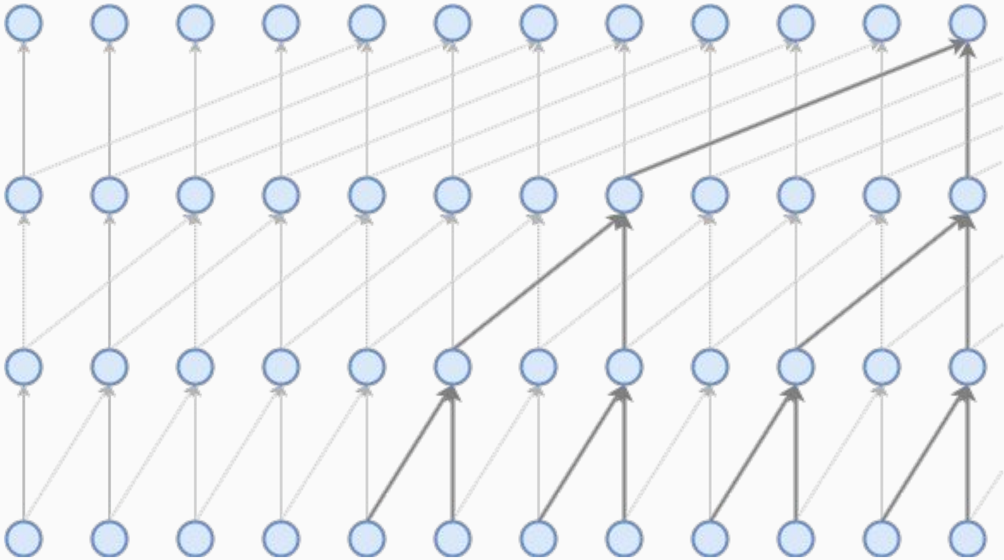




WaveNet

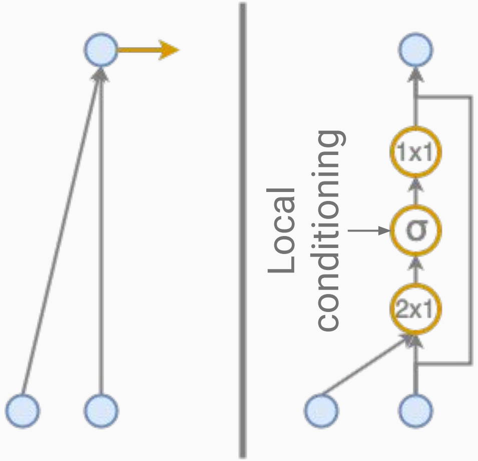
Sequence of dilated convolutions, with some extra 1x1 convolutions and residuals hidden in the detail

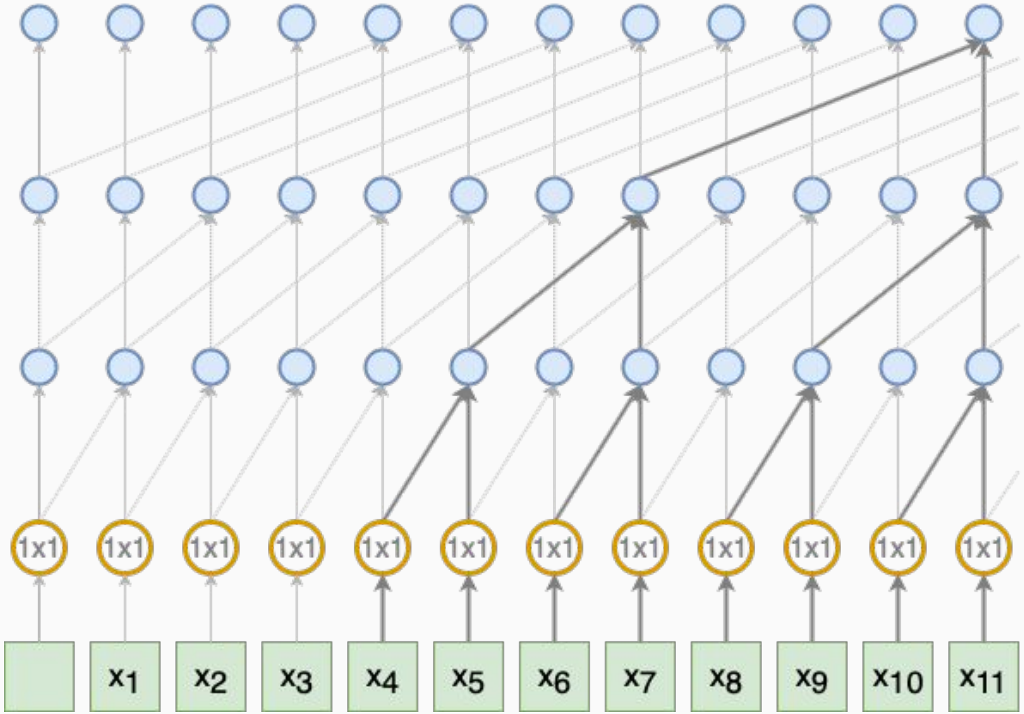




WaveNet

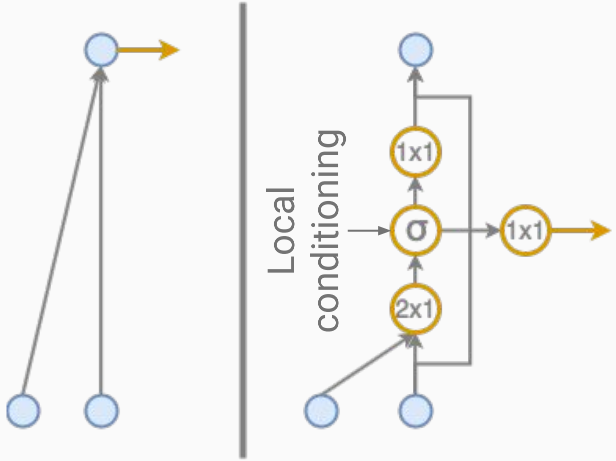
Our inputs are first projected with a 1x1 convolution.
Same as a feedforward NN

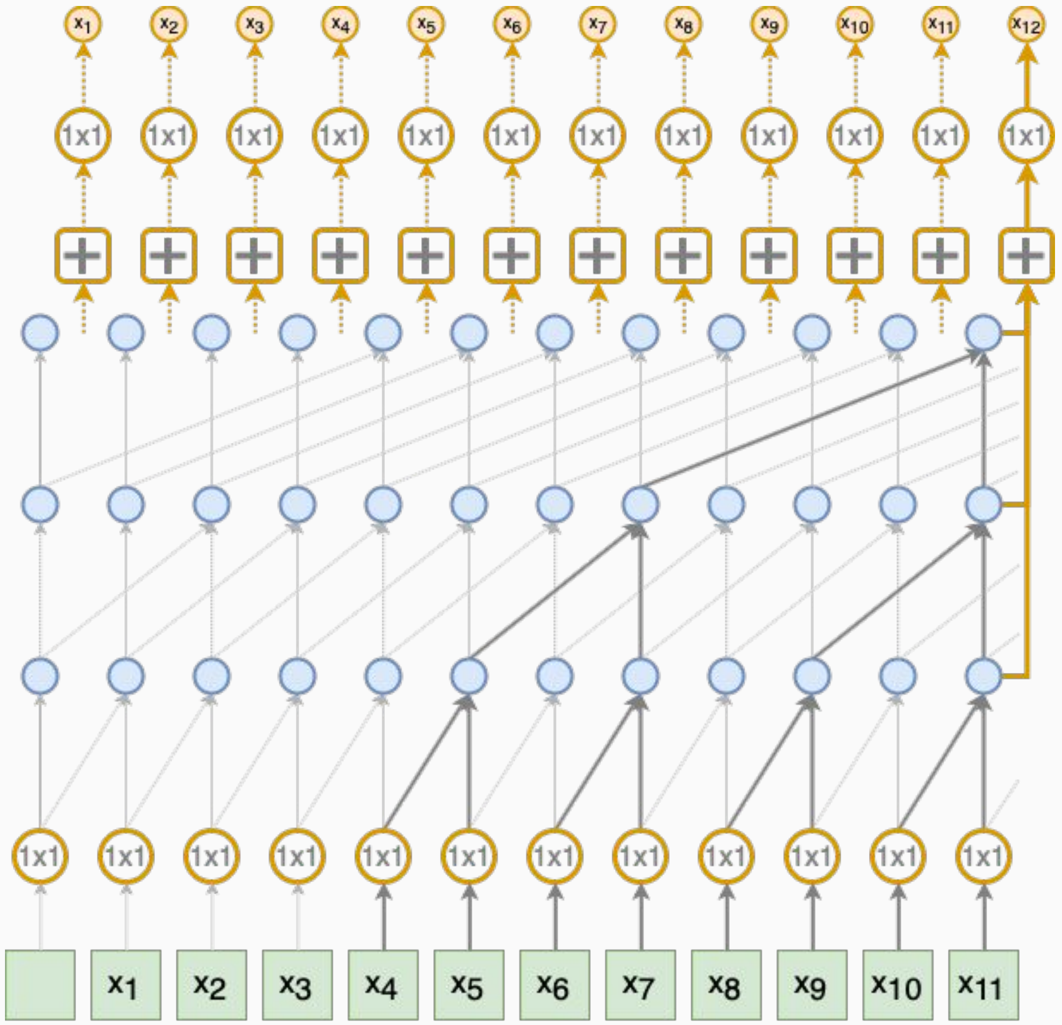




WaveNet

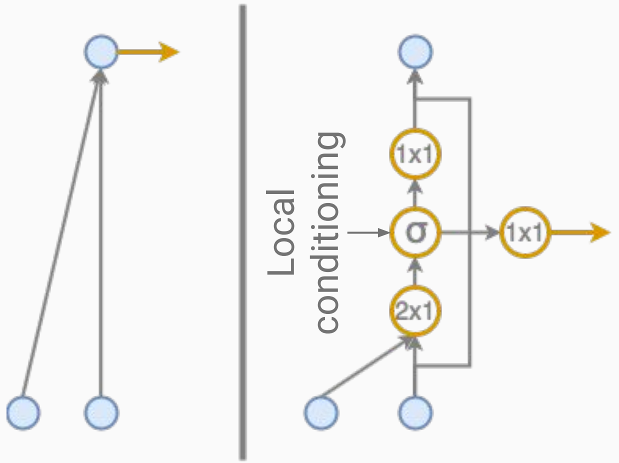
Output actually comes from a sum of additional "skip" outputs

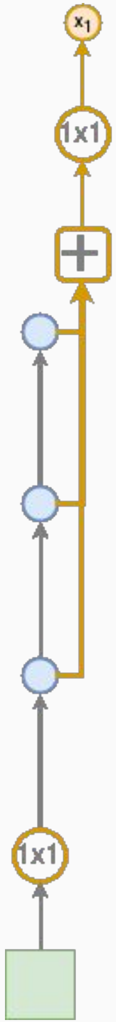




WaveNet

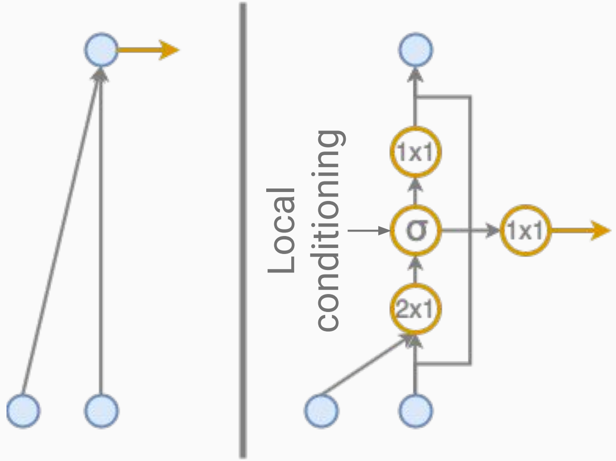
To generate $\mathbf{x}_{12}$ we first need to generate all previous samples autoregressively

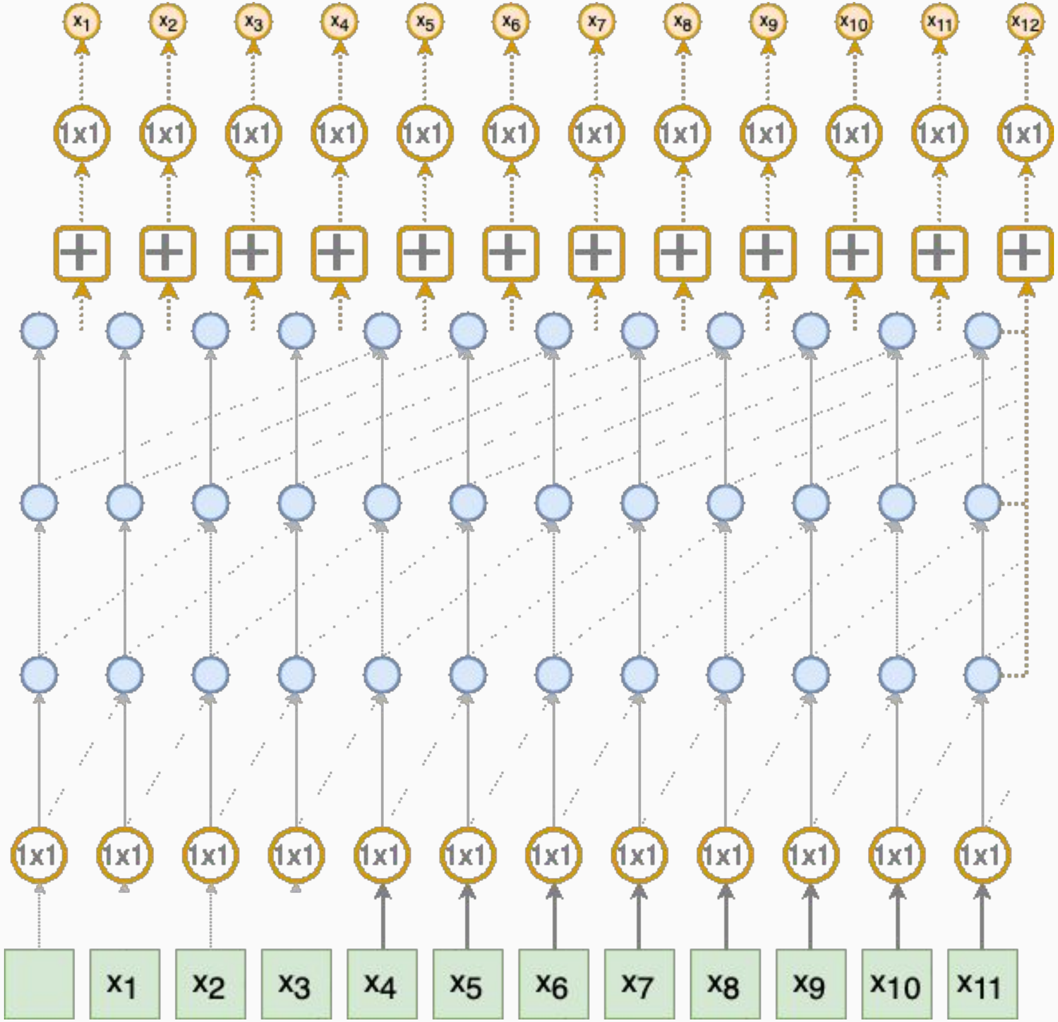




WaveNet

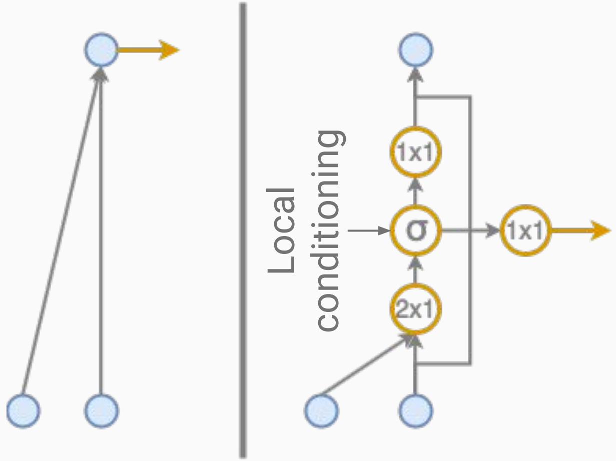
To generate $\mathbf{x}_{12}$ we need to use the following convolution outputs

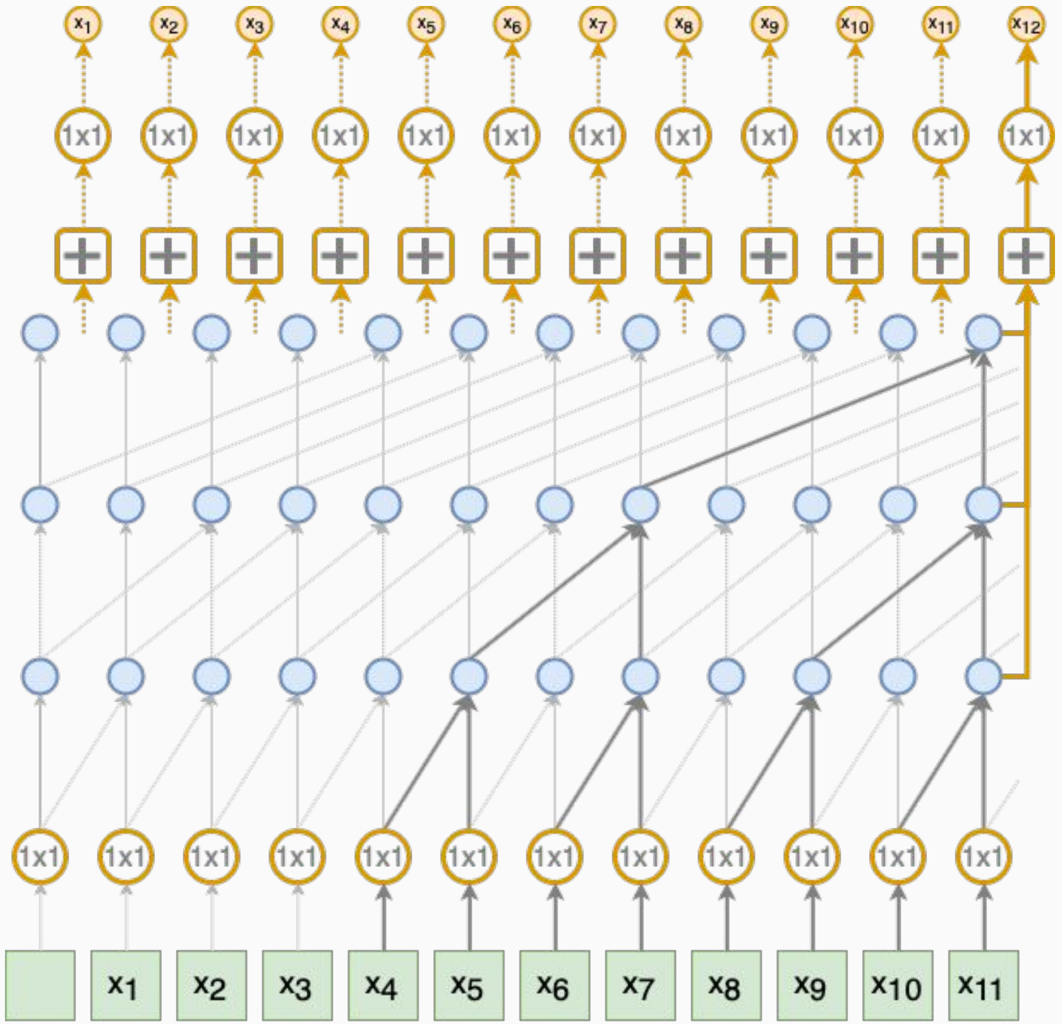




WaveNet

Output actually comes from a sum of additional “skip” outputs



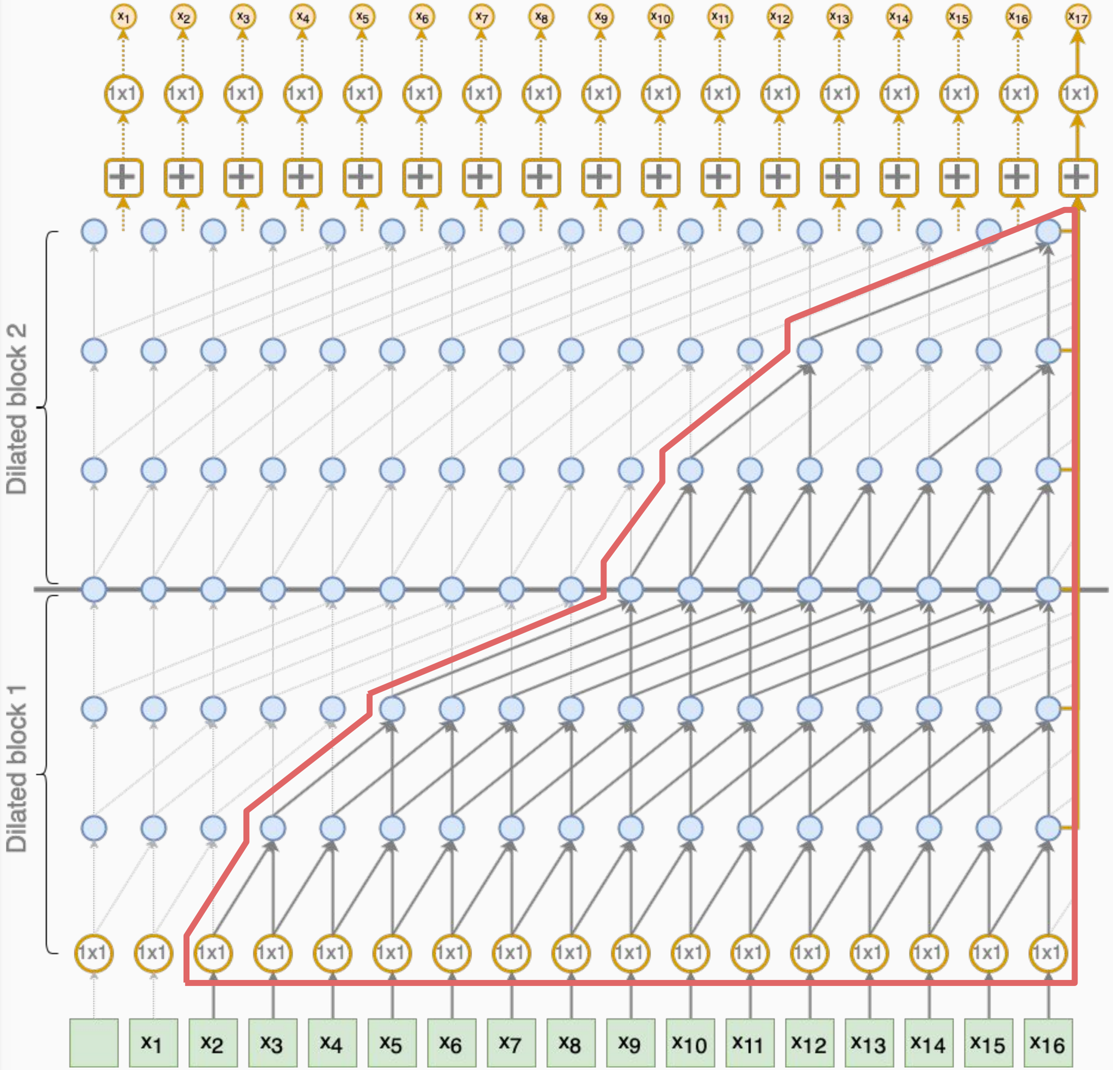


WaveNet

We can stack multiple dilation blocks to create larger models

Note the context size for 3 stacked dilated convolutions is 8

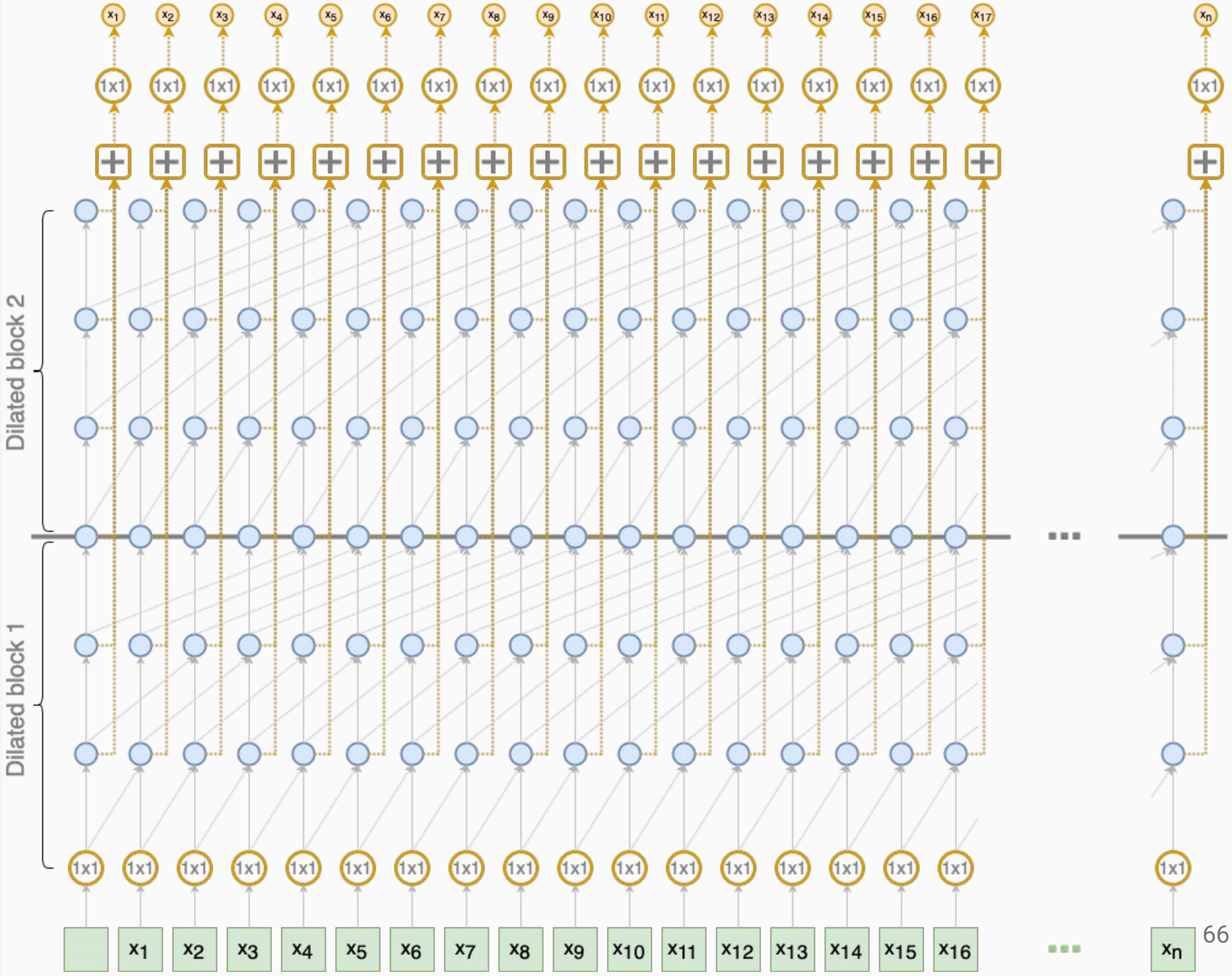
But 2 consecutive blocks of the same architecture has a context of 16 (-1)



WaveNet

Training

We can run all steps at once during training as we have the input for all columns



What did we miss?

Padding

Probabilistic modelling interpretation (and flow based models – future tutorial?)

Loss criterion – negative log-likelihood

Attention – Jason F will cover this in a future Speak session

Part 3

Overview

1-a	Neural networks	Part 1
1-b	Recurrent neural networks	
2-a	Convolutional neural networks	Part 2
2-b	Autoregressive models	
3-a	**Attention**	**Part 3**
3-b	Self-attention	

Learning to align

Tacotron

Overview

1-a	Neural networks	Part 1
1-b	Recurrent neural networks	
2-a	Convolutional neural networks	Part 2
2-b	Autoregressive models	
3-a	Attention	**Part 3**
3-b	**Self-attention**	

Self-attention explained

Transformer

Thanks