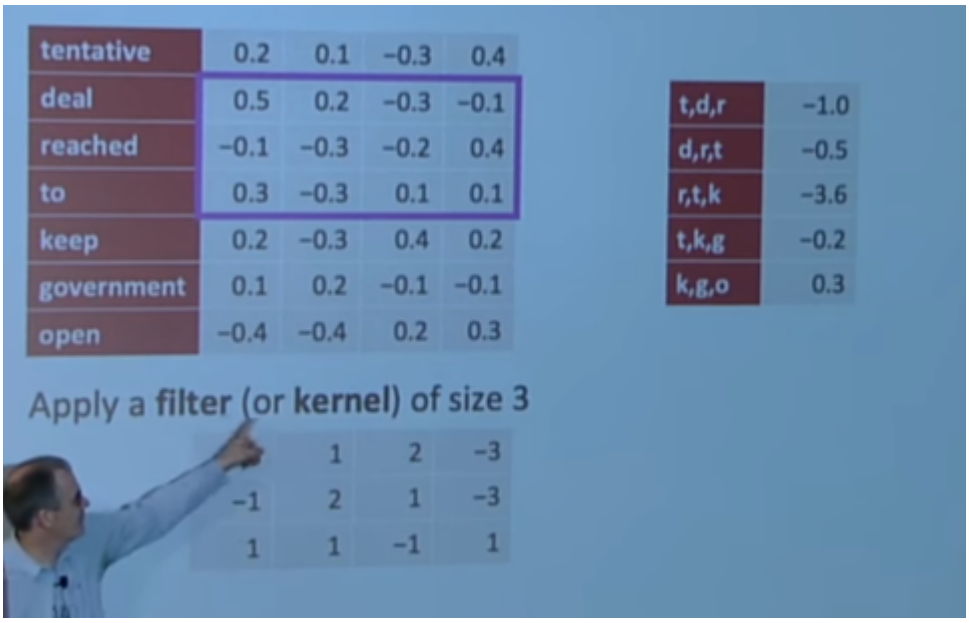
Convolutional NeuralNetworks in NLP:

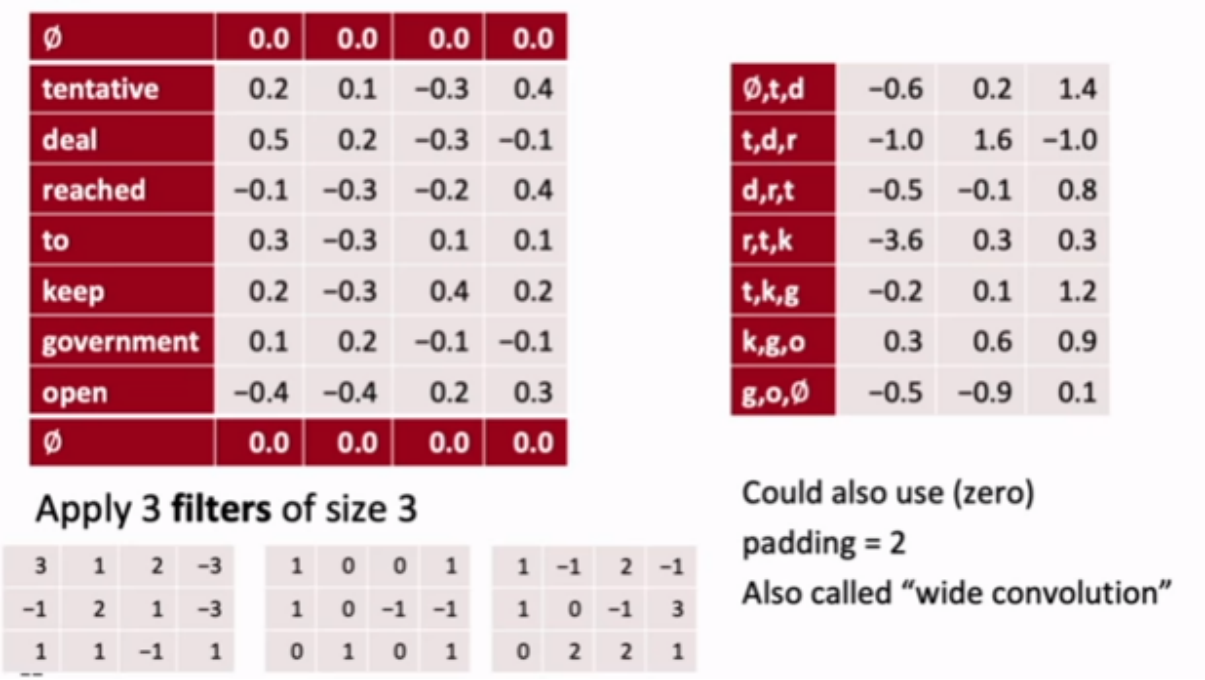
We use CNNs because the RNNs don't have any representation for a part of a sentence or phrase. They also have disadvantages like giving more weight to the final words.

CNNs work by taking a particular part of an input and processing it (the filter is called kernel). CNNs do it across all the input by moving the window.



Sometimes we add padding at the end in order to get the same number of processed output as the actual data.

In the examples above we have applied one filter/kernel which is a matrix which is element wise multiplied with the moving window and the sum of terms of resultant matrix is taken. We could also do this by taking many different filters/kernels with each kernel concentrating on different aspects of the data.



Max Pooling:

Sometimes when we want to judge the tone of an input we don’t have to care about what the average word tells us, Because if we notice most of the time even a very negative review about a movie does not contain a lot of negative words. So what we should do is calculate the maximum negativity expressed in the sentence, hence we take the max-pooling.

Example: Max pooling done on a processed output with three kernels:

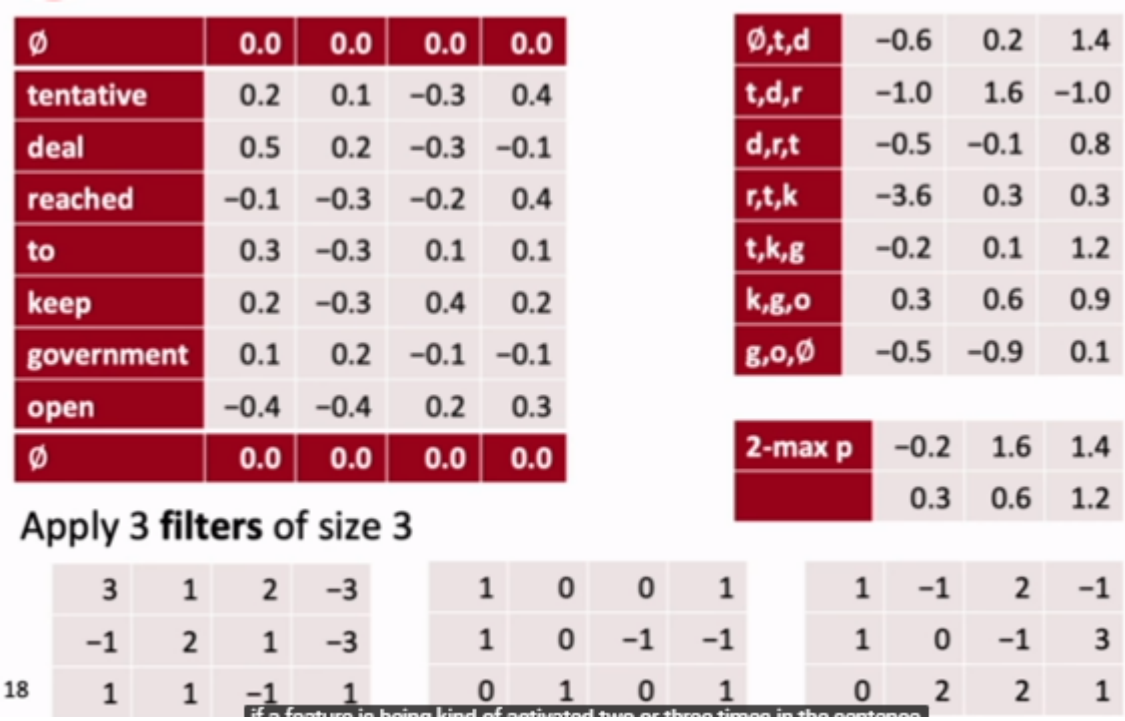
Ø,t,d	−0.6	0.2	1.4
t,d,r	−1.0	1.6	−1.0
d,r,t	−0.5	−0.1	0.8
r,t,k	−3.6	0.3	0.3
t,k,g	−0.2	0.1	1.2
k,g,o	0.3	0.6	0.9
g,o,Ø	−0.5	−0.9	0.1

max p	0.3	1.6	1.4

Now these 3 max p values quantify the 3 characteristics the 3 kernels want to express

K-max pooling:

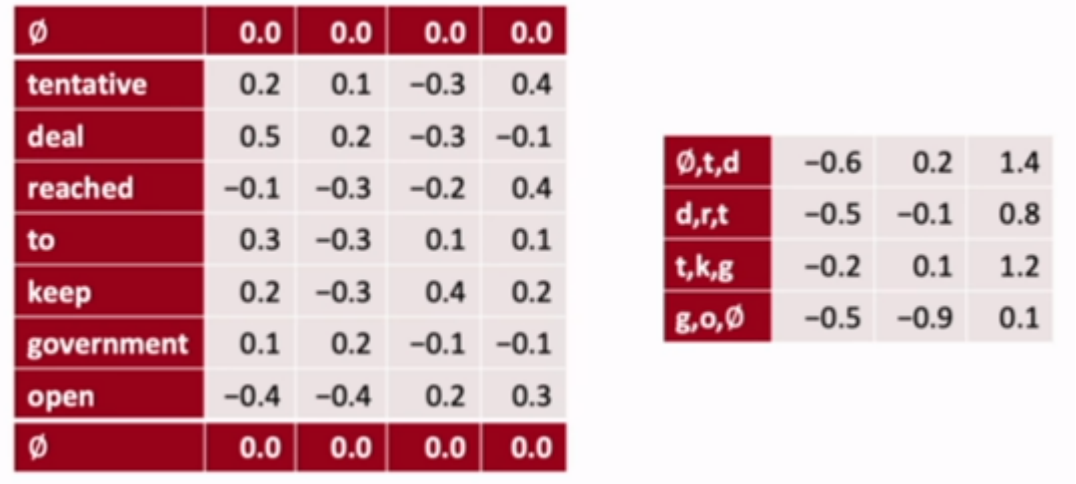
In this the best k max values are taken instead of the one maximum value as this will represent the input better and will not be skewed by one case.



Note: you put the two maximum values not in ascending/descending order but in the order in which they appear.

Stride:

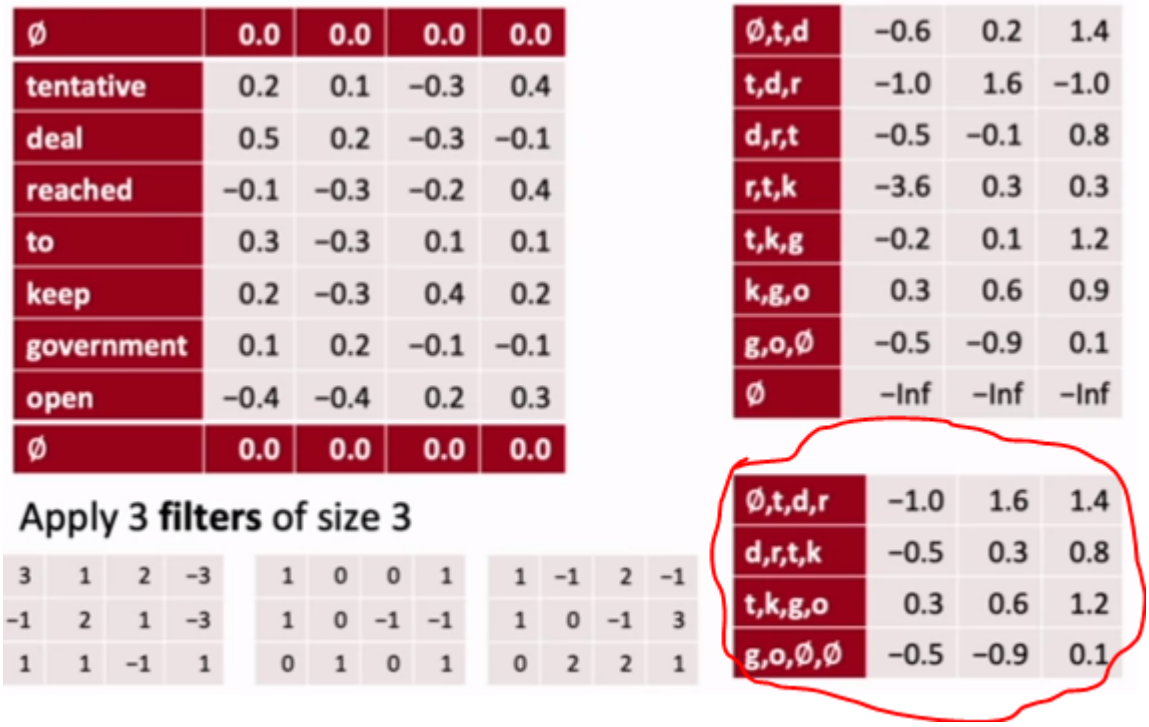
By default when the window moves on the input data it moves by one word, This is a stride of 1. We can also increase the stride to be more than one (while keeping it less than the size of the window) to capture all the words while having to deal with less processing.



The above example has stride of 2 and 3 filters of size 3.

Local Pooling:

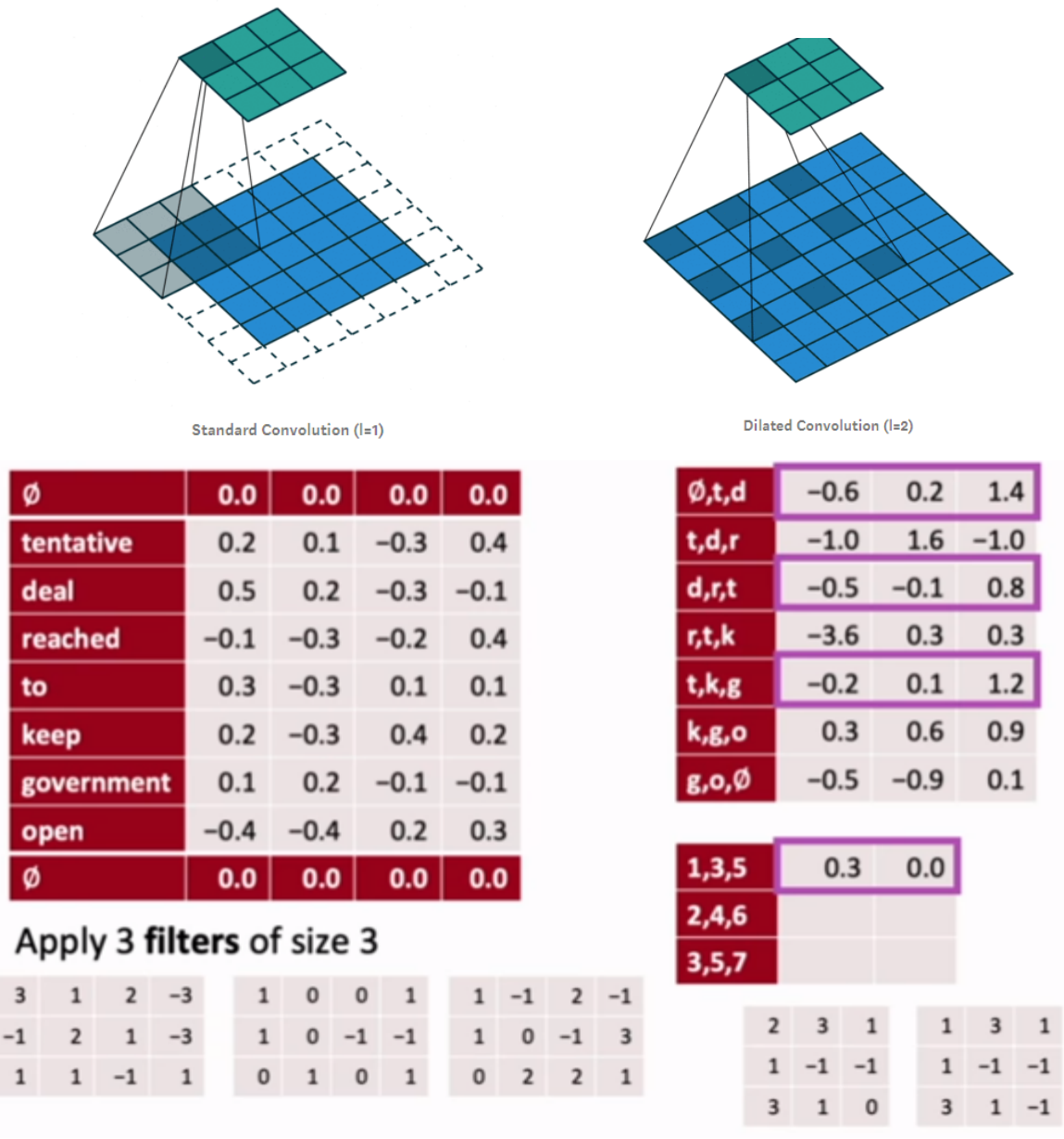
We first do a normal convolution and then, if we want a local pooling of stride 2, what we do is calculate the max pooling / average pooling of pairs of two rows in the convolution data.



The circled table is the max pooled data.

Dilation:

Diagramatic explanation:

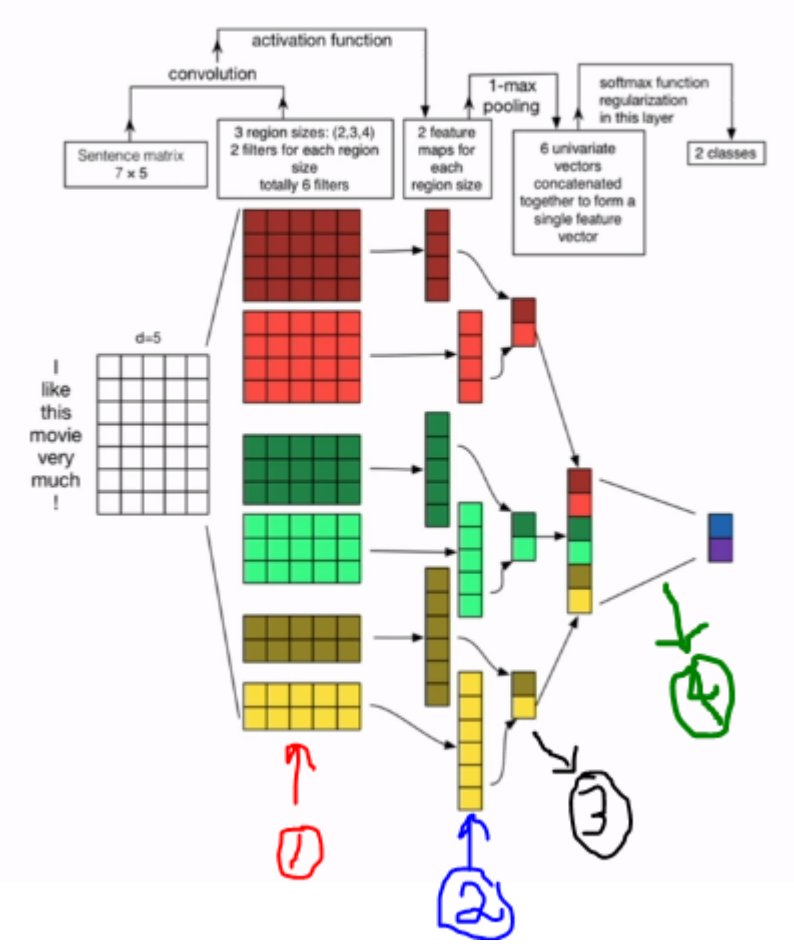


Dilated convolution can also be viewed as a convolution over a convolution used to keep the matrix small while being able to represent much of the data.

Multi channel Input data:

A brilliant idea will be to double the word vectors and keep one of them static used to represent the word's meaning absolutely and another set can be backpropagated into in order to tweak precisely for this scenario. The final word representation will be a concatenation of both of these.

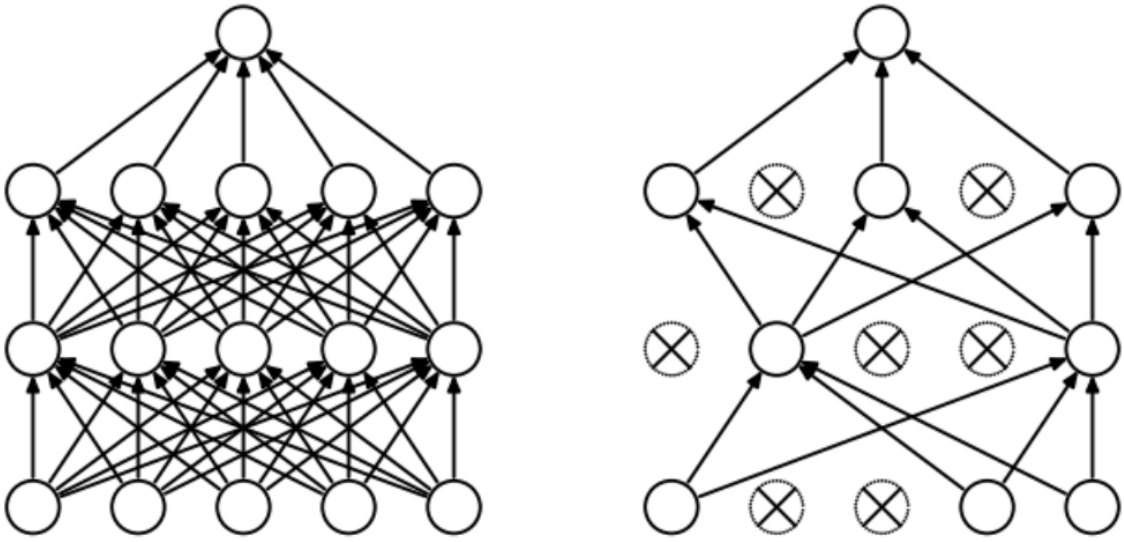
Example of a CNN:



1: The 6 different kernels used, they have different sizes in order for variety.
2: The outputs 6 of using the 6 kernels. (Note the smaller kernels / filters have more outputs as they have more windows as there is no padding)
3: The Max pooling result
4: Pass the concatenated result through softmax in order to get the final result like sentiment expressed.

Regularization:

Regularization in RNN is done using dropouts.(Dropping neural network nodes so that they have lesser links and don't overfit the training data).



(a) Standard Neural Net (b) After applying dropout.

The usual dropout ratio used is 0.5. (Half of the nodes are dropped)
Dropout usually gives a 2 to 4 % improvement.

4. Model comparison: Our growing toolkit

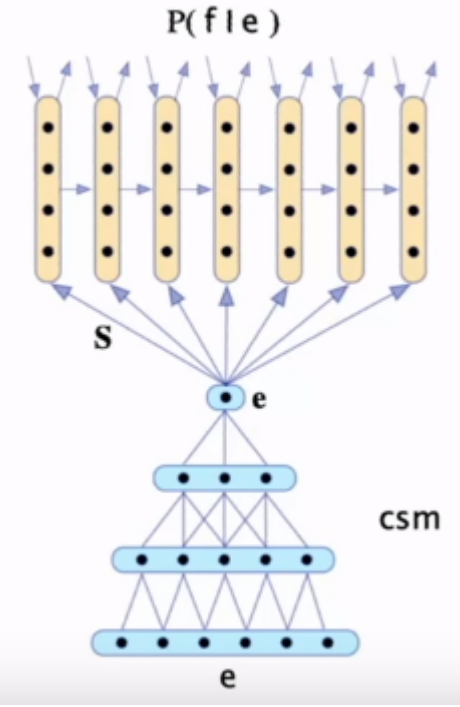
- **Bag of Vectors**: Surprisingly good baseline for simple classification problems. Especially if followed by a few ReLU layers! (See paper: Deep Averaging Networks)
- **Window Model**: Good for single word classification for problems that do not need wide context. E.g., POS, NER.
- **CNNs:** good for classification, need zero padding for shorter phrases, hard to interpret, **easy to parallelize on GPUs.** Efficient and versatile
- **Recurrent Neural Networks**: Cognitively plausible (reading from left to right), not best for classification (if just use last state), much slower than CNNs, good for sequence tagging and classification, great for language models, can be amazing with attention mechanisms

1x1 convolutions (Network-in-Network(NiN)):

Initially it looks like it doesn't make any sense but what is does is it takes the word vectors and converts it into a number. This can be used to reduce the dimensions to n (by using n filters/kernels) or something used to bring about non-linearity.

CNN for encoding:

One of the first Neural Language Processing models had a multilayer CNN that kept on decreasing the size of the input to generate the encoding and an RNN was then used to decode it. Diagrammatic Representation:



CNNs can also be used to move across characters and analyze it to create word embeddings.
CNNs are also used in very Deep models which work in the character level.

Quasi-RNN (Q-RNN):

In this model the work of LSTMs is done by a CNN. It can be used to get results that are sometimes better than LSTMs but this model is definitely faster as we can introduce more parallelism than an LSTM.