conv nets for language classification

- CNNs can be naively parallelized, unlike RNNs. This is because of data depedency between time steps in RNNs. need to compute first step before moving onto next. in CNN all filters at a given layer can be computed in parallel on a GPU.
- just like we do CNNs for image processing, we can do CNNs to capture n-gram phrases (a series of n consecutive words)
- we will convolve over the word vectors.
- if we are using a k-gram convolution and the word vectors are d dimensional, our convolution will be k by d.
- The convolution filter is flattened so as to form one big vector. Then it is just a dot product

$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$

- a common application of this is convolutional neural nets for sentence classification → **sentiment classification** or document classification etc.
- we can then use a max pooling layer. this max pooling layer allows us to have variable length sentences and yet generate a fixed size output from this variable sized sentence.
- with optimization, we can tune this filter to be maximum for the phrase that matters most
- if we have m convolutional filters, then output of max pooling will be a m dimensional vector. $z = [c_1, c_2 ... c_m]$
- Note that there is a different type of dropout: instead of dropping out activations as we usually do, we could drop out links. this means the activation still stays but its contribution to some neurons of the next layer is 0-weighted (contributes nothing to some neurons of the next layer). which works better? the jury is still out - try both for your task.
- note that if what you are trying to extract from the document is too distributed, and you need a lot of different phrases to make up your decide the classification, then you might need many different conv filters.