Convolut tion

Convolutional Networks

- A convolutional network is designed to identify indicative local indicators in a large structure, and combine t
  fixed size vector representation of the st
  pooling function, capturing the local aspect
  informative
- A convolutio
  feedforward network is.
- It has been tremendously successful in im
  computer vision), where the input is the raw pixels of an image
- In NLP, it has been shown to be effective in sentence classification, etc.

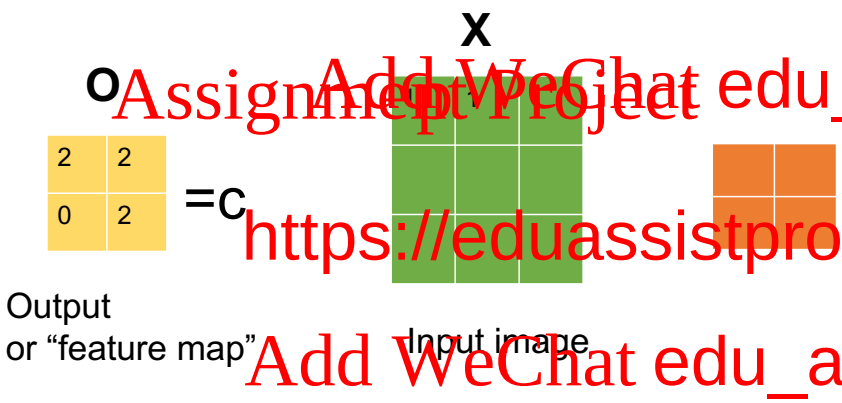
Why it has been so effective in image processing

Image pixels

Four operations in a convolutional network

- Convolution
- Non-linear a
- Pooling or sub
- Classification with a fully connected lay

Image convolution



Kernel size = (2,2), strides = 1

Image convolution

conv(X, U)

Kernel size = (2,2), strides = 1

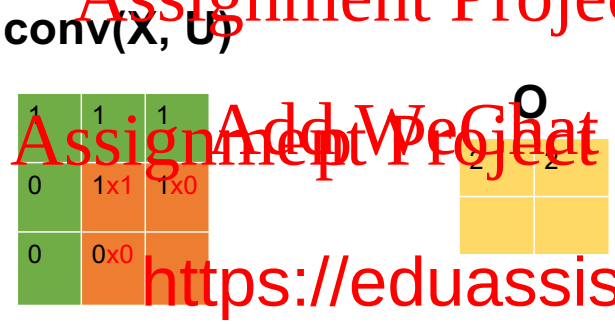
Image convolution

Kernel size = (2,2), strides = 1

Image convolution

Kernel size = (2,2), strides = 1

Image convolution



Kernel size = (2,2), strides = 1

Forward computation

$o_{11} = x_{11}u_{11} + x_{12}u_{12} + x$

o_1 $_3u_{22}$

o_2 $_2u_{22}$

$o_{22} = x_{22}u_{11} + x_{23}u_{12}$ $_{22}$

ReLU

- Nonlinear transformation with ReLU

$$(0, input)$$

- As we know, Re t does not change the dimension of the feat

- ReLU replaces all negative pixel values in t 0

Image ReLU

ReLU activation and Max pooling

- ReLU activation is a component-wise function and does not change the dimension of the input

- Max pooling does change the dimension
  specify the pool size and strides

$$[2] = Max\left(\begin{bmatrix} 2 & 2 \\ 0 & 2 \end{bmatrix}\right)$$

pool size $= (2, 2)$, strides $= 2$

Training a CNN

- Loss functions: Cross entropy, hinge, square
- What are the parameters of a CNN?
  - The filte ard networ ses
- Computing the gradient for the convolu t from a feedforward neural network...

Computing the gradient on U

$$\frac{\partial E}{\partial u_{11}} = \frac{\partial E}{\partial o_{11}}\frac{\partial o_{11}}{\partial u_{11}} + \frac{\partial E}{\partial o_{12}}\frac{\partial o_{12}}{\partial u_{11}} + \frac{\partial E}{\partial o_{21}}\frac{\partial o_{21}}{\partial u_{11}} + \frac{\partial E}{\partial o_{22}}\frac{\partial o_{22}}{\partial u_{11}}$$
$$= \frac{\partial E}{\partial o_{11}}x_{11} + \frac{\partial E}{\partial o_{12}}x_{12} + \frac{\partial E}{\partial o_{21}}x_{21} + \frac{\partial E}{\partial o_{22}}x_{22}$$

$$\frac{\partial E}{\partial u_{12}} = \frac{\partial E}{\partial o_{11}}\frac{\partial o_{11}}{\partial u_{12}} + \frac{\partial E}{\partial o_{12}}\frac{\partial o_{12}}{\partial u_{12}} + \frac{\partial E}{\partial o_{22}}\frac{\partial o_{22}}{\partial u_{12}}$$

$$\frac{\partial E}{\partial u_{21}} = \frac{\partial E}{\partial o_{11}}\frac{\partial o_{11}}{\partial u_{21}} + \frac{\partial E}{\partial o_{12}}\frac{\partial o_{12}}{\partial u_{21}} + \frac{\partial E}{\partial o_{22}}\frac{\partial o_{22}}{\partial u_{21}}$$
$$= \frac{\partial E}{\partial o_{11}}x_{21} + \frac{\partial E}{\partial o_{12}}x_{22} + \frac{\partial E}{\partial o_{21}}x_{31} + \frac{\partial E}{\partial o_{22}}x_{32}$$

$$\frac{\partial E}{\partial u_{22}} = \frac{\partial E}{\partial o_{11}}\frac{\partial o_{11}}{\partial u_{22}} + \frac{\partial E}{\partial o_{12}}\frac{\partial o_{12}}{\partial u_{22}} + \frac{\partial E}{\partial o_{21}}\frac{\partial o_{21}}{\partial u_{22}} + \frac{\partial E}{\partial o_{22}}\frac{\partial o_{22}}{\partial u_{22}}$$
$$= \frac{\partial E}{\partial o_{22}}x_{11} + \frac{\partial E}{\partial o_{12}}x_{23} + \frac{\partial E}{\partial o_{21}}x_{32} + \frac{\partial E}{\partial o_{22}}x_{33}$$

Summing up errors from all outputs that the filter component has contributed to.

Reverse Convolution

The computation of the gradient on the filter can be viewed as a reverse convolution:

$$\begin{bmatrix} \frac{\partial E}{\partial u_{11}} & \frac{\partial E}{\partial u_{12}} \\ \frac{\partial E}{\partial u_{21}} & \frac{\partial E}{\partial u_{22}} \end{bmatrix} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} * \begin{bmatrix} \frac{\partial E}{\partial o_{11}} & \frac{\partial E}{\partial o_{12}} \\ \frac{\partial E}{\partial o_{21}} & \frac{\partial E}{\partial o_{22}} \end{bmatrix}$$

Computing the gradient on X (if this is not the input layer)

$$\frac{\partial E}{\partial x_{11}} = \frac{\partial E}{\partial o_{11}} u_{11} + \frac{\partial E}{\partial o_{12}} 0 + \frac{\partial E}{\partial o_{21}} 0 + \frac{\partial E}{\partial o_{22}} 0$$

$$\frac{\partial E}{\partial x_{12}} = \frac{\partial E}{\partial o_{11}} u_{12} + \frac{\partial E}{\partial o_{12}} u_{11} + \frac{\partial E}{\partial o_{21}} 0 + \frac{\partial E}{\partial o_{22}} 0$$

$$\frac{\partial E}{\partial x_{13}} = \frac{\partial E}{\partial o_{11}} 0 + \frac{\partial E}{\partial o_{12}} u_{12} + \frac{\partial E}{\partial o_{21}} + \frac{\partial E}{\partial o_2}$$

$$\frac{\partial E}{\partial x_2} \quad \frac{\partial E}{\partial o_{11}} \quad \frac{\partial E}{\partial o_{12}} \quad \frac{\partial E}{\partial o_{21}} \quad \frac{\partial E}{\partial o_{22}} 0$$

$$\frac{\partial E}{\partial x_{22}} \quad \frac{\partial E}{\partial o_{11}} \quad \frac{\partial E}{\partial o_{12}} \quad \frac{\partial E}{\partial o_{21}} \quad \frac{\partial E}{\partial o_{22}} u_{11}$$

$$\frac{\partial E}{\partial x_{23}} = \frac{\partial E}{\partial o_{11}} 0 + \frac{\partial E}{\partial o_{12}} u_{22} + \frac{\partial E}{\partial o_{21}} + \frac{\partial E}{\partial o_{22}}$$

$$\frac{\partial E}{\partial x_{31}} = \frac{\partial E}{\partial o_{11}} 0 + \frac{\partial E}{\partial o_{12}} 0 + \frac{\partial E}{\partial o_{21}} u21 + \frac{\partial E}{\partial o_{22}} 0$$

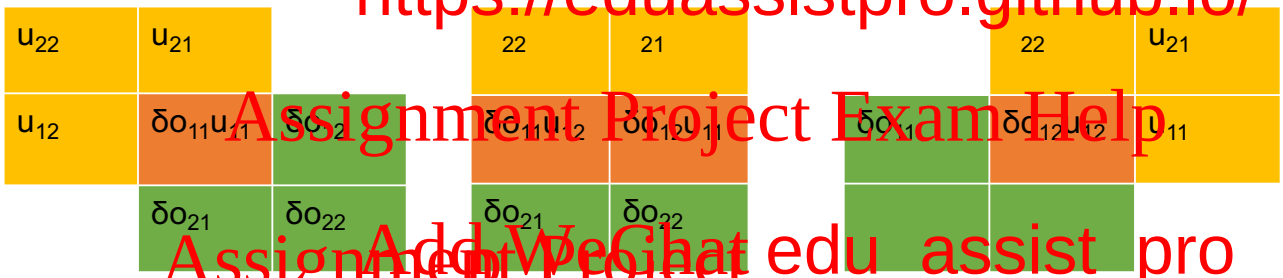
$$\frac{\partial E}{\partial x_{32}} = \frac{\partial E}{\partial o_{11}} 0 + \frac{\partial E}{\partial o_{12}} 0 + \frac{\partial E}{\partial o_{21}} u_{22} + \frac{\partial E}{\partial o_{22}} u_{21}$$

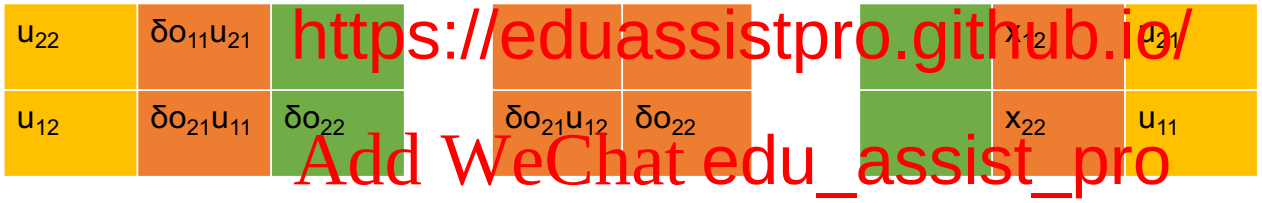
$$\frac{\partial E}{\partial x_{12}} = \frac{\partial E}{\partial o_{11}} 0 + \frac{\partial E}{\partial o_{12}} 0 + \frac{\partial E}{\partial o_{21}} 0 + \frac{\partial E}{\partial o_{22}} u_{22}$$

Full convolution

$$\left[\begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \frac{\partial E}{\partial x_{12}} \\ \frac{\partial E}{\partial x_{21}} & \frac{\partial E}{\partial x_{22}} \end{bmatrix}, \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix}\right)$$

Gradient on X if it is not the inputs





Why convolutational networks for NLP?

- Even though bag-of-words models are simpl
  some text classification tasks, they don't acc
  where multip
  "not interest
- The analogy with image processing is if th
  as separate features. (The analogy migh

Alternative text representations

- When using feed-forward networks for text classification, the input text is represented as a sparse vector that encodes bag-of-words features.
- Alternatively, a text can be represented as a se tokens $w_1, w_2, w_3, \ldots, w_M$. Th is useful for models such as **Co** r **ConvNets**, which proces
- Each word token w_m is represent $\boldsymbol{e}_{w_m}$, with dimension V. The complete do represented by the horizontal concatenation of these one-hot vectors: $\boldsymbol{W} = [\boldsymbol{e}_{w_1}, \boldsymbol{e}_{w_2}, \cdots, \boldsymbol{e}_{w_m}] \in R^{V \times M}$.
- To show that this is equivalent to the bag-of-words model, we can recover the word count from the matrix-vector product $\boldsymbol{W}[1, 1, \cdots, 1]^\top \in R^V$.

Input to a convolutional network in a text classification task

When using conv nets for text classification, it is com
"look up" the pretrained word embedding (e.g., th
produced by Word2Vec or GLOVE[1]) f ord in the sequence:

$$\tilde{\boldsymbol{x}}^{(0)} = \boldsymbol{\Theta}\,[\,w_1, \quad \in \mathbb{R}^{K_e \times M}$$

where $\boldsymbol{e}_{w_m}$ is a column vector of zeros, with a 1 at position w_m, K_e is the size of embeddings

[1] https://nlp.stanford.edu/projects/glove

"Convolve" the input with a set of filters (kernels)

- A *filter* is a weight matrix of dimension $\boldsymbol{C}^{(k)} \in \mathbb{R}^{K_e \times h}$ where $\boldsymbol{C}^{(k)}$ is the kth filter. Note the first dimension of the filter is the same as the size of the embedding.
  - Unlike image processing, the filter doesn't full widt
- To merge adj by sliding a set of filters across it:

$$\boldsymbol{X}^{(1)} = f(\boldsymbol{b}$$

where f is an activation function (e.g., tanh, ReLU), b is a vector of bias terms, and $*$ is the convolution operator.

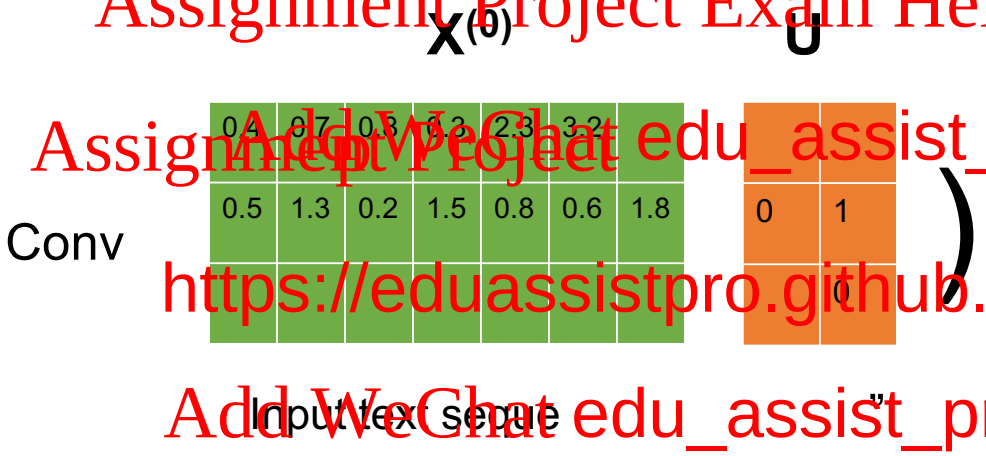
Computing the convolution

- At each position m, compute the element-wise product of the kth filter and the sequence of words of window size h (think of it as an ngram of length h) starting a m and take its $\ldots (k) \odot \boldsymbol{X}^{(0)}_{m:m+h-1}$
- The value of the kth position with the $\ldots$ e computed as

$$x^{(1)}_{mk} = f\left(b_k + \sum_{k'=1}^{K_e} \ldots_{n-1}\right)$$

- When we finish the convolution step, if we have K_f filters of dimension $\mathbb{R}^{K_e \times h}$, then $\boldsymbol{X}^{(1)} \in \mathbb{R}^{K_f \times M-h+1}$
- In practice, filters of different sizes are often used to captured ngrams of different lengths, so $\boldsymbol{X}^{(1)}$ will be K_f vectors of variable lengths, and we can write the size of each vector of h_k

Convolution step when processing text



=	2.9	3.1	3.4	?	?	?

$X^{(1)}$

Padding

- To deal with the beginning and end of the input, the matrix is often padded with $h - 1$ columns at the beginning and end of the input. This is known as **wide convolution**.
- If no padding is applied, the output of the convolutional layer will be $h - 1$ units smaller than the input. This is known as **narrow convolution**.

Pooling

- After D convolutional layers, assuming filters have identical lengths, we have a representation of the docu $\boldsymbol{X}^{(D)} \in \mathbb{R}^{K_z \times M}$.

- It is very likely th , so we need to tur before feeding them to a feedward ne classification.

- This can done by **pooling** across times (over the sequence of words)

Prediction and training with CNN

- The CNN needs to be fed into a feedforward network to make a prediction $\hat{y}$ and compute the loss $\ell^{(i)}$ in training.
- Parameters of a CNN includes the weight ma feedforward network and the filters s well as the biases.
- The paramet , which may involve computing the gradient for t function.

$$\frac{\partial z_k}{\partial x_{k,m}^{(D)}} = \begin{cases} 1, & x_{k,m}^{(D)} = \max\left(x_{k,1}^{(D)}, x_{k,2}^{(D)}, \cdots, x_{k,M}^{(D)}\right) \\ 0, & \text{Otherwise} \end{cases}$$

Different pooling methods

- Max pooling

$$z_k = \max\left(x_{k,1}^{(b)}, x_{k,2}^{(b)}, \ldots, x_{k,M}^{(b)}\right)$$

- Average pool

$$z_k = \frac{1}{M}\sum_{m=1}^{M} x_{k,m}$$

A graphic representation of a CNN

Figure 1: Caption

Using Conv Nets in PyTorch