Deep Learning Techniques

DL 3. Convolutional Neural Networks (CNN)

Outline

- Motivation
- CNN Operations
- Gradient-based Optimization
- LM with word-level and character-level CNNs

Comparison with RNN-based models

- RNN
  - Long-term dependency
  - Sequential computation (slow), prone to gradient vanishing
- CNN
  - N-gram-like models focusing on local dependency
  - Computation is faster (easier to be parallelized)
- RNN and CNN can be jointly used in combination!

What is Convolution?

Dictionary

Search for a word

con·vo·lu·tion

/ˌkänvəˈlo͞oSHən/

noun

1. a thing that is complex and difficult to follow.
   "the convolutions of farm policy"
   synonyms: complexity, intricacy, complication, twist, turn, entanglement, contortion; More

2. a coil or twist, especially one of many.
   "crosses adorned with elaborate convolutions"
   synonyms: twist, turn, coil, spiral, twirl, curl, helix, whorl, loop, curlicue, kink, sinuosity; More

What is Convolution?

- Wikipedia https://en.wikipedia.org/wiki/Convolution
  - **convolution** is a mathematical operation on two functions (*f* and *g*) to produce a third function that expresses how the shape of one is modified by the other.

$$\boxed{(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau.}$$

An equivalent definition is (see commutativity):

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(t-\tau) g(\tau)\, d\tau.$$

  - Essentially, it is an operation over f and g by taking a weighted sum of f (or g) using g (or f) as the weights.

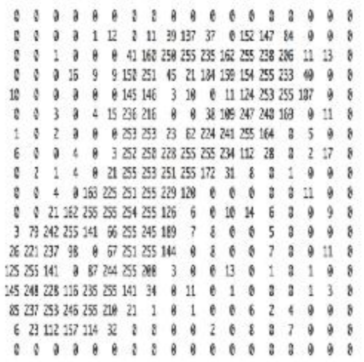
Convolutional Neural Networks (CNNs)

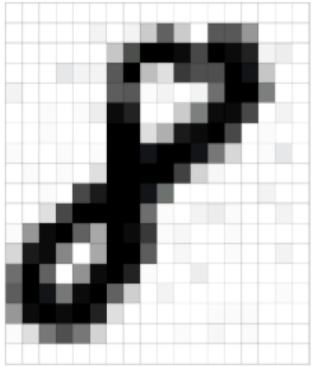
- A type of neural networks
- Popular in image/video recognition, text mining, recommendation, etc.
- Inspired by biological processes
  - The *receptive (activated) field* of cortical neurons can be approximated mathematically by a *convolution operation.*

Convolution Operations over an Image

Ujjwal Karn https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

- Every **image** is an input matrix of pixel values (corresponding to function f)

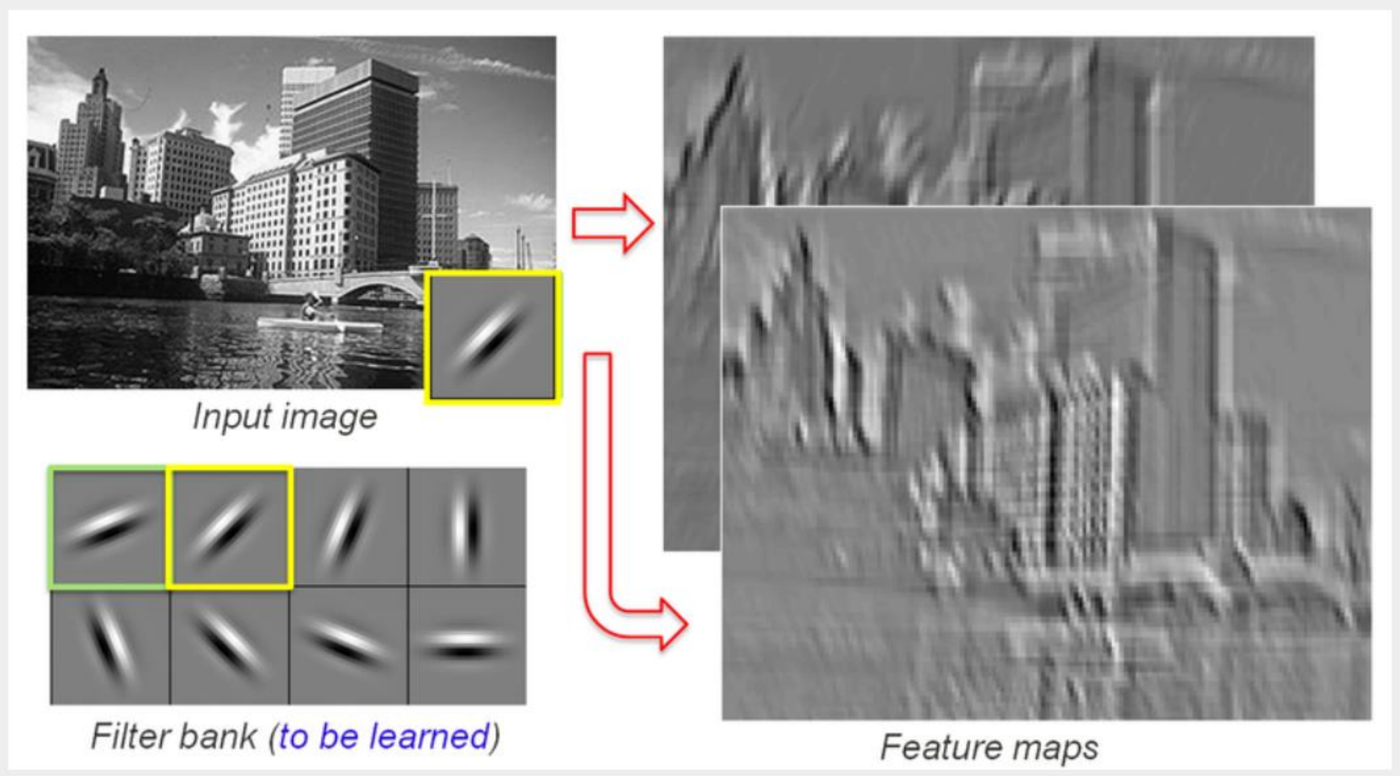




- Each **filter** (corresponding to function g) applies to a local region over the entire input.

Convolution Operations over an Image (cont'd)

Ujjwal Karn https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/



Convolution Operations over an Image (cont'd)

AI Shack http://aishack.in/tutorials/image-convolution-examples/

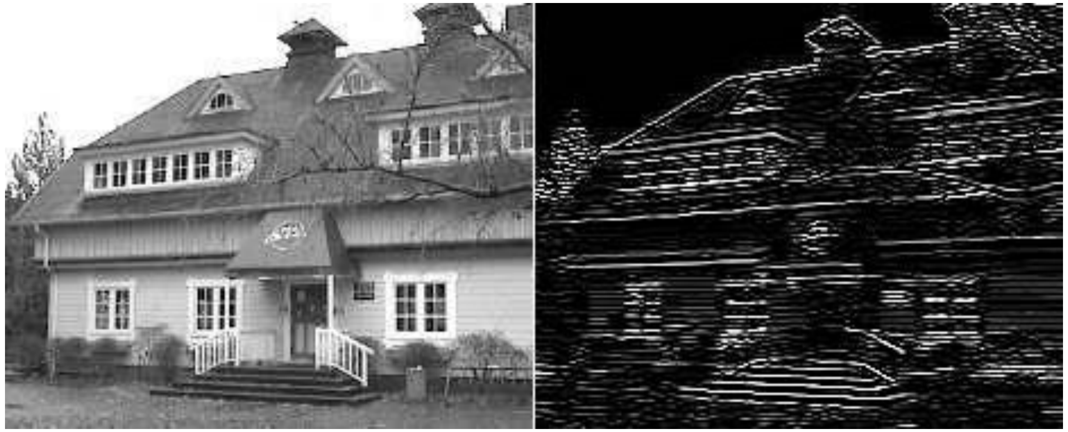
Blur Filter

(local average)

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal
Line Filter



Convolution Operations over an Image (cont'd)

AI Shack http://aishack.in/tutorials/image-convolution-examples/

-1	-1	-1
-1	8	-1
-1	-1	-1

Edge Filter

Below result I got with edge detection:

Terminology & Notation

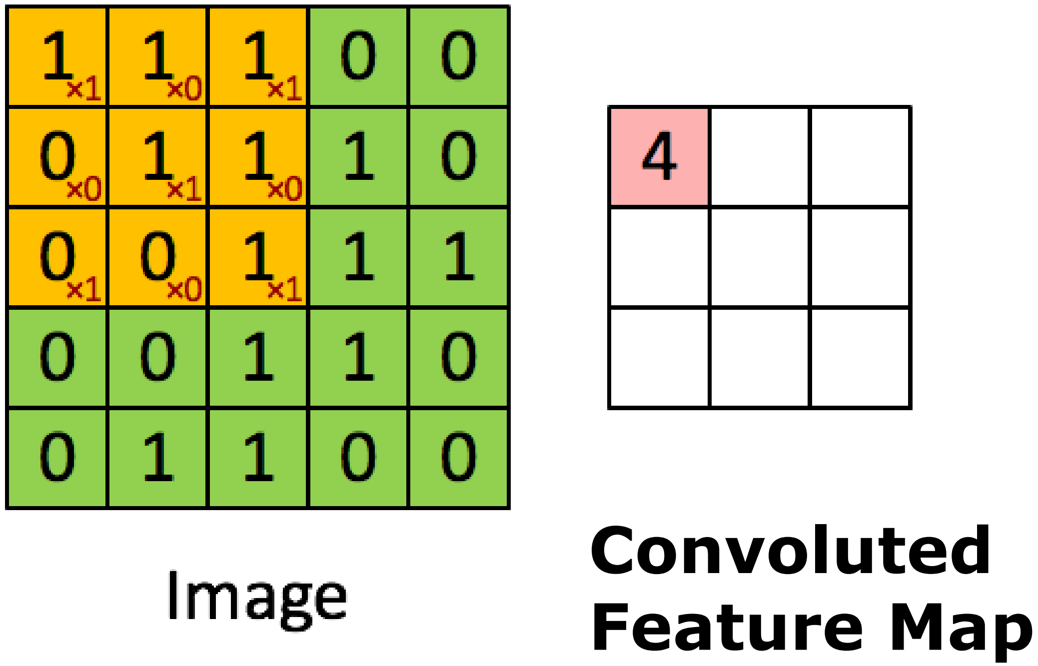
- Input X (or function f in our introduction of convolution)
  - Numerical representation of an image, a sentence, a time series, etc.
- Filter (kernel) W (or function g in our introduction of convolution)
- Receptive Field: the local region that the filter is applied to.
- Feature Map: the output of convolution

CNN Operations

- ✓ Convolution
- Striding
- Padding
- Pooling

Image Convolution (a toy example)

Animation: *http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution*
More references [here]()

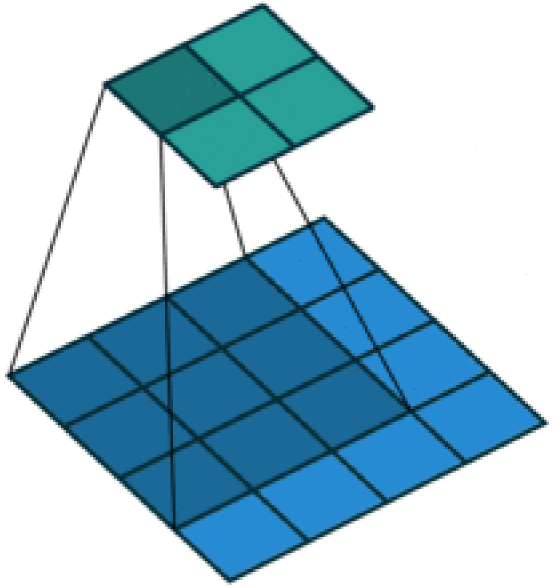


Learnable Parameters in the CNN filter

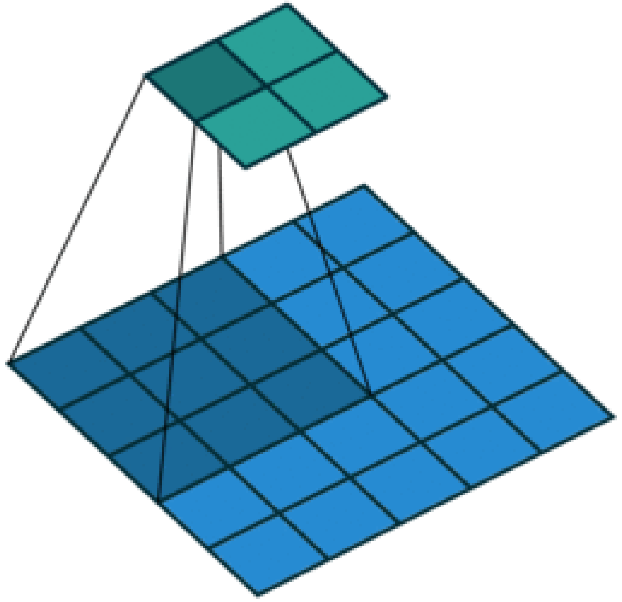
- Filter size is $k \times k$ (learnable model parameters)
  - Given an $n \times n$ input, the produced feature map is $m \times m$ whrere $m = n - k$ +1.
- Compared to a fully connected network (perceptron) for a $R^{n \times n} \rightarrow R^{m \times m}$ linear mapping, its number of model parameter is $n^2 \times m^2$ (much larger than k^2).
- Therefore, CNN is computationally more feasible for extracting lower dimensional features from input data and is less prone to overfitting.

Striding Operation in CNN

- Striding size (hyperparameter): the number of units to shift step-by-step



Striding of size 1 (“no striding”)

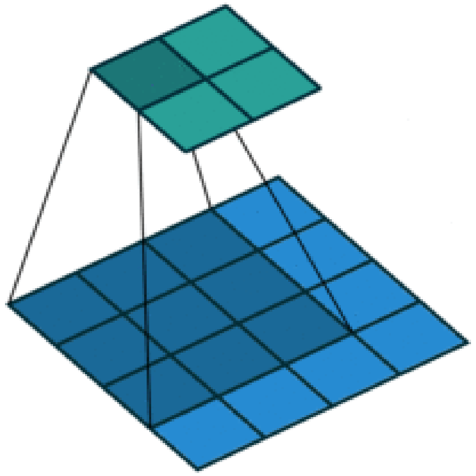


Striding of size 2

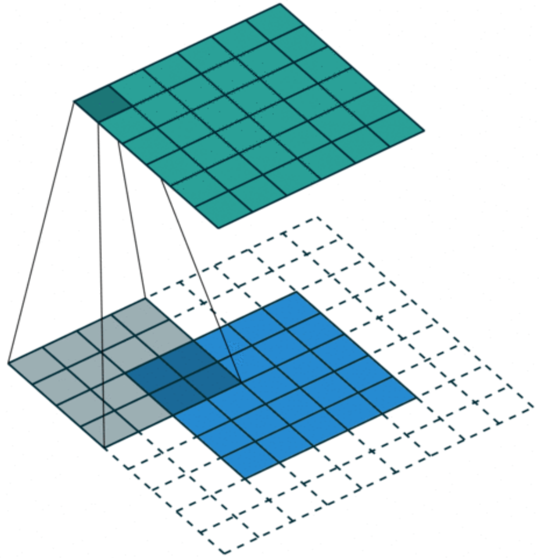
Animation: https://github.com/vdumoulin/conv_arithmetic

Padding in CNN

- Adding zeros around the input image (for desirable output size, or focusing on edges)



No padding

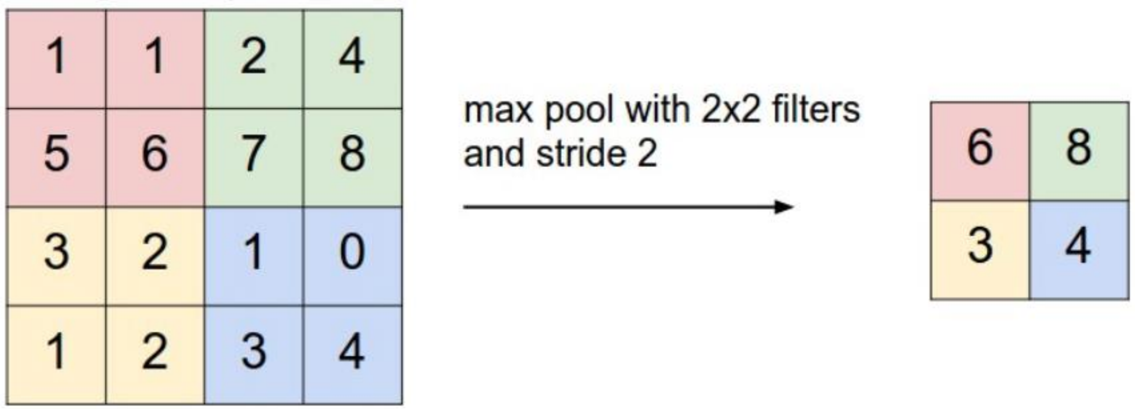


Padding

Source: https://github.com/vdumoulin/conv_arithmetic

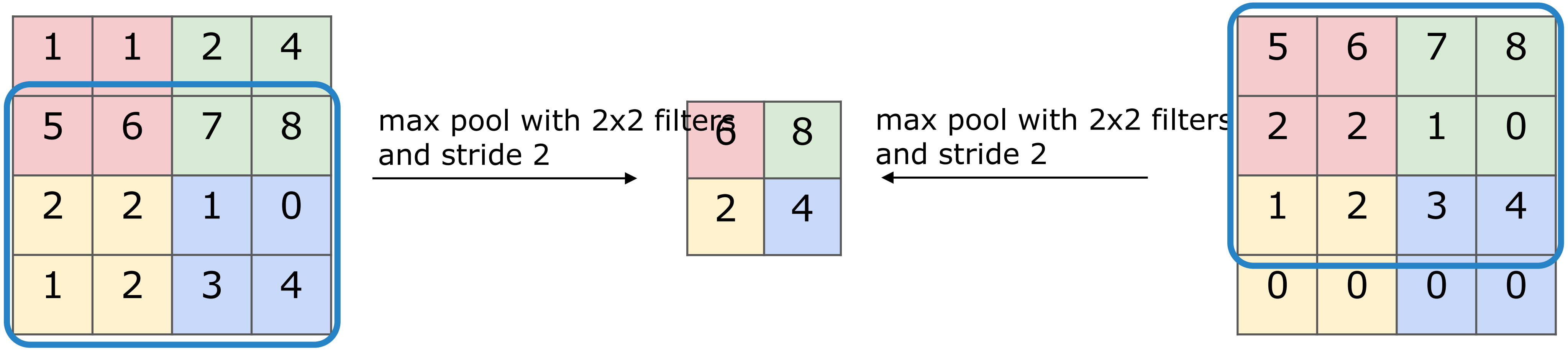
Pooling in CNN

- Fixed operation (average or max) over each local region for further dimensionality reduction



Max pooling with 2×2 Filter.
Source: http://cs231n.github.io/convolutional-networks/#pool

Image Pooling: Capturing the Local Invariance



Max pooling with 2×2 Filter.
Source: http://cs231n.github.io/convolutional-networks/#pool

Comparing the images on the left and the right: The local pattern (orange box) is shifted upwards by 1 unit on the right, but the output of pooling remain unchanged.

CNN for image object detection

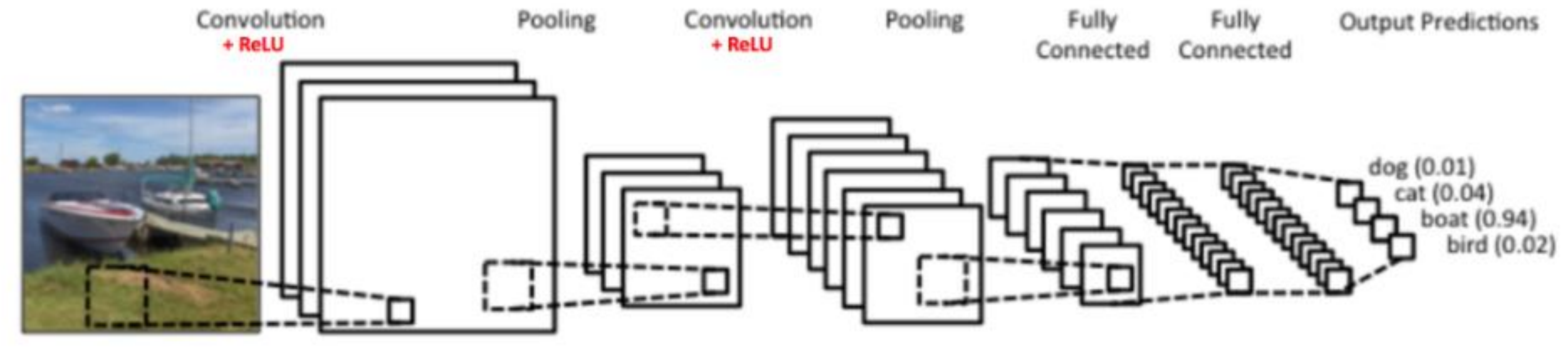
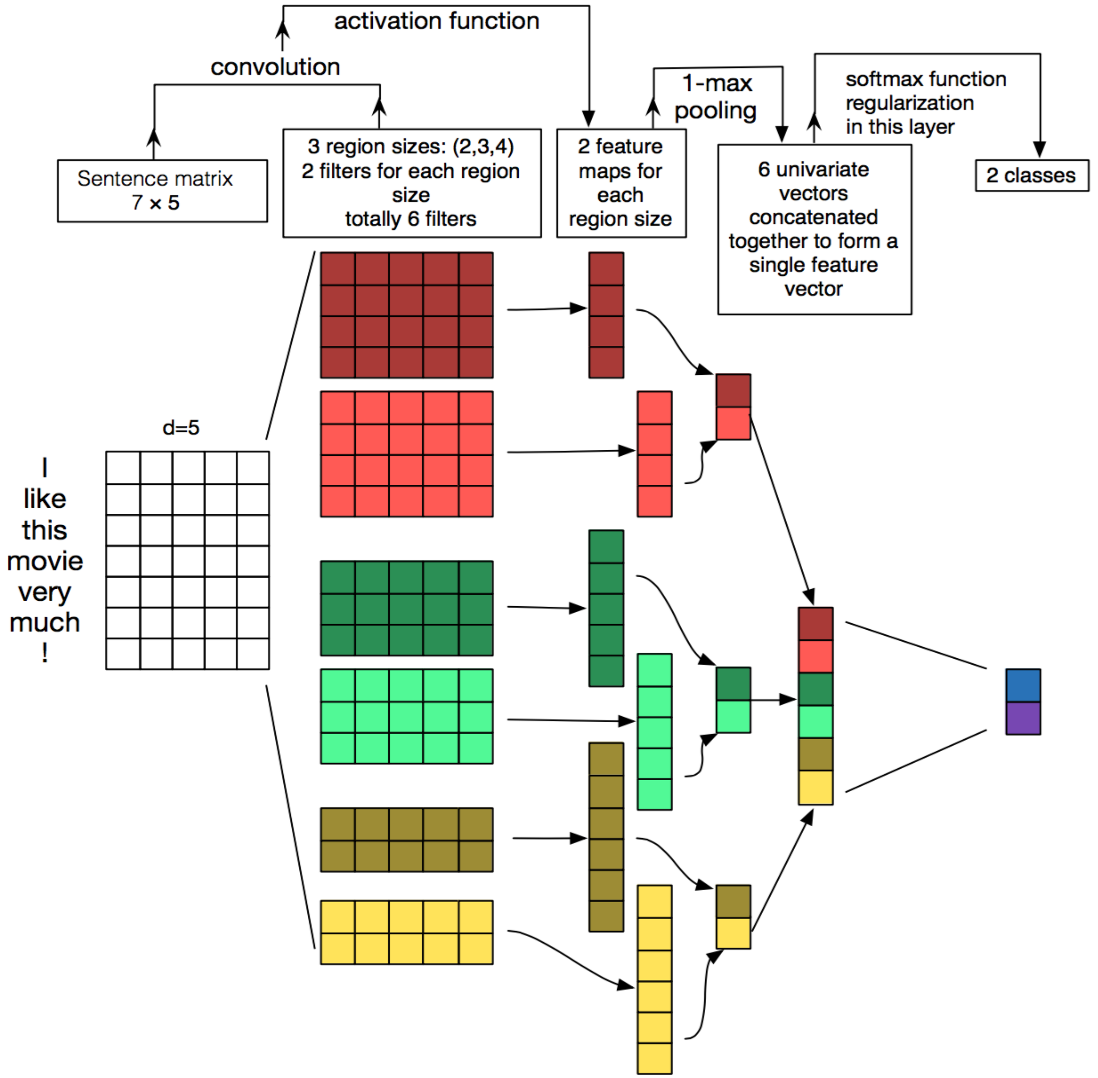


Figure 3: A simple ConvNet. Source [5]

- Automatically learn the filters (kernels) based on labeled training data
- Extracting local and lower-dimensional features from input data
- Computationally more efficient than MLPs or RNNs
- Why do we need multiple filters in parallel?

CNNs for Text

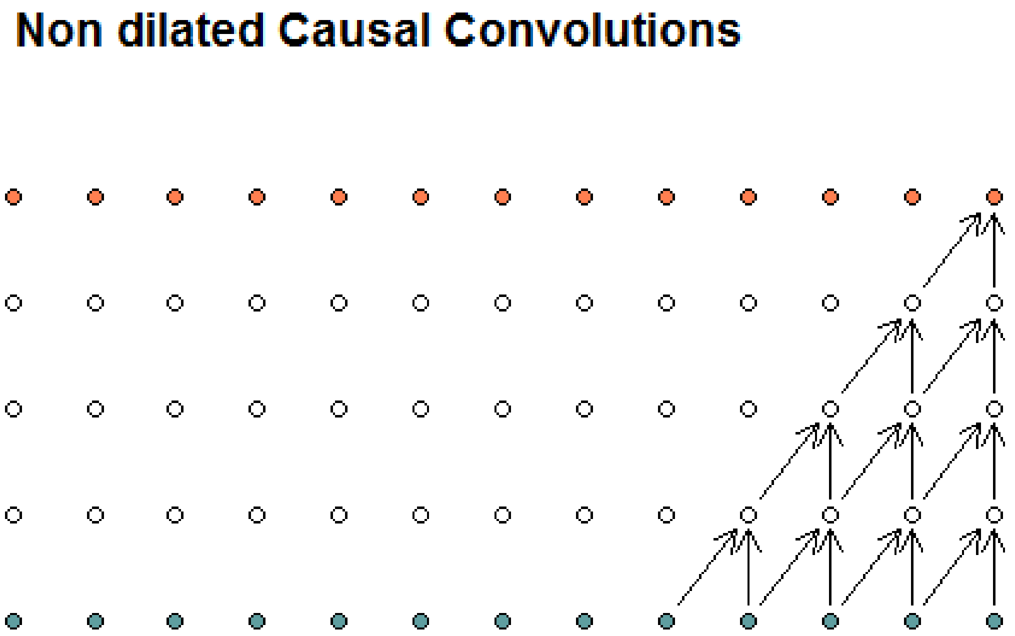
- Use word embedding to obtain the input "image" (1-D)
- Set the sizes of convolution filters to be $\boldsymbol{m} \times \boldsymbol{d}$ for $m = 2, 3, 4, \cdots$
  - $\boldsymbol{m}$ is the number of words a filter takes into account (usually 1-5, like n-gram)
  - $\boldsymbol{d}$ is the size of word embedding
- Use multiple filters for each size (why?)



Typical CNN structure for text classification [Kim 2014]

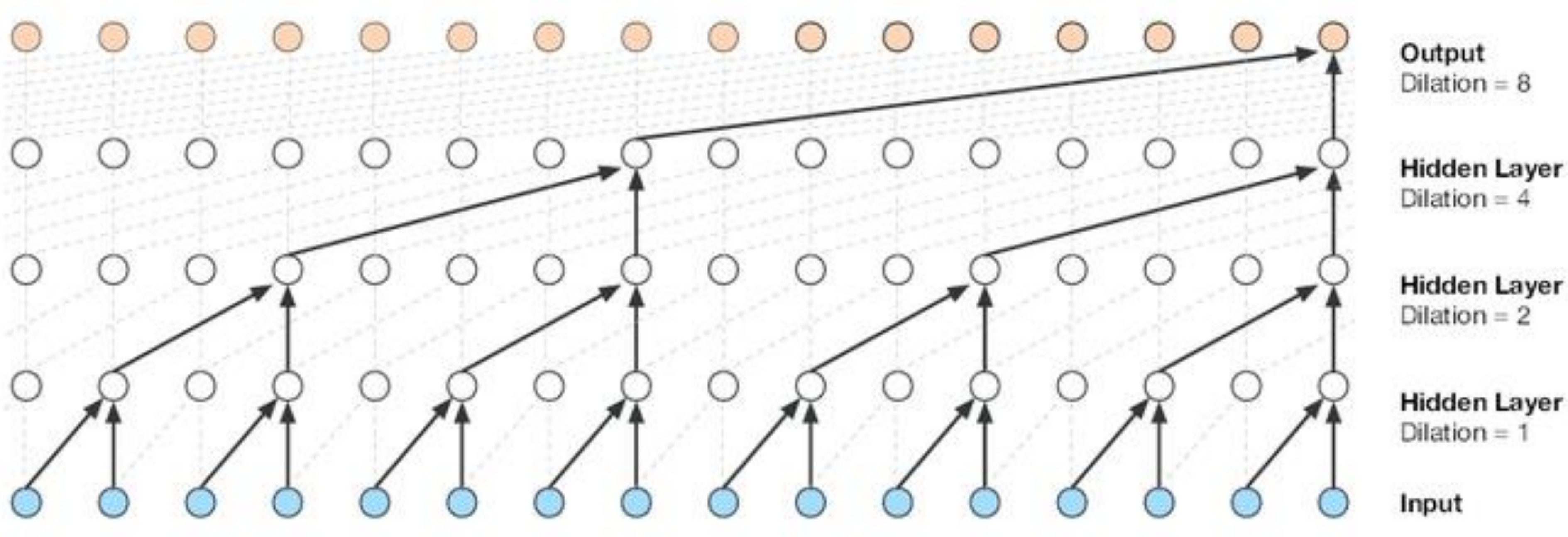
Conventional CNN with Stacked Layers

https://stats.stackexchange.com/questions/287774/wavenet-is-not-really-a-dilated-convolution-is-it



- Enlarging the receptive-field size which grows linearly in the numbers of the layers
- Each layer uses multiple filters (instead of one), for the expressiveness of the model.
- For a desired receptive-field size of n, the total umber of model parameters is $O(kmn)$ where k is the filter size and m is the number of filters per layer in the above architecture.

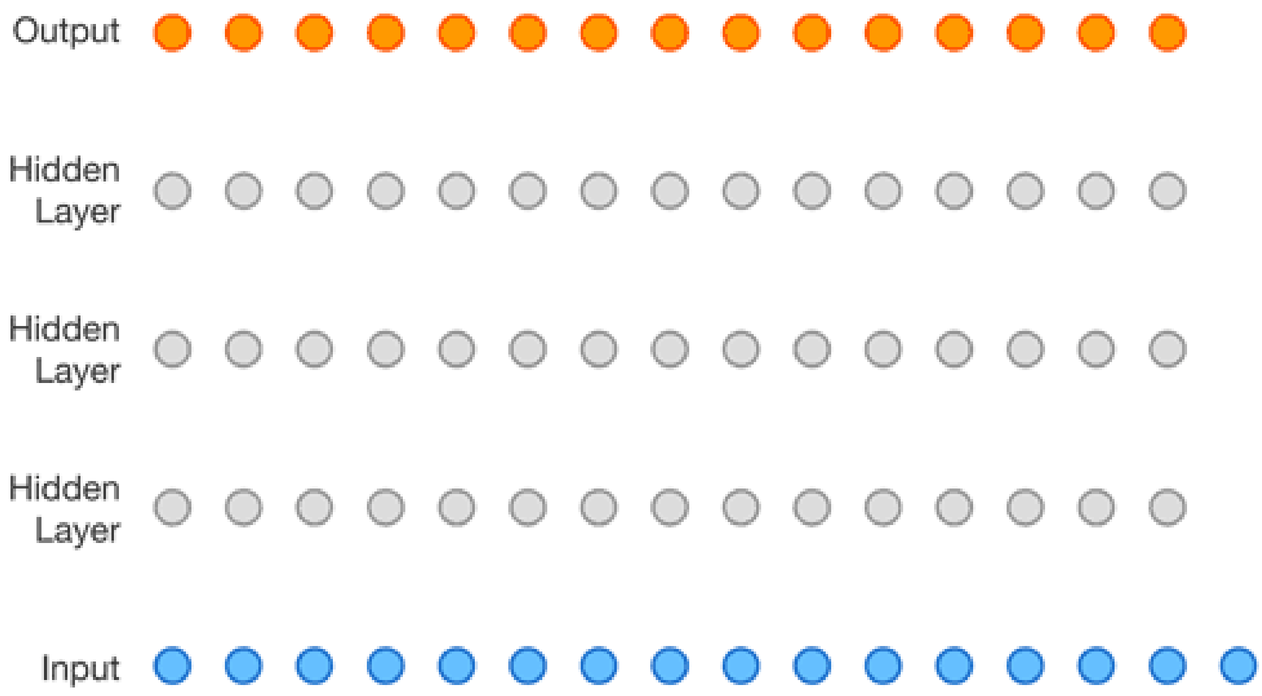
Dilated CNN



- Skipping certain numbers of nodes at each layer (e.g., skipping 1, 3, 5, …in Dilation 1)
- The receptive-field size grows exponentially in the number of layers ($\log n$).
- For a desired receptive-field size of n, the total number of model parameters is $O(km \log n)$.

Animation of Dilated CNN

https://stats.stackexchange.com/questions/287774/wavenet-is-not-really-a-dilated-convolution-is-it



For the same receptive-filter size, dilated CNN is more compact than conventional stacked CNN.

Outline

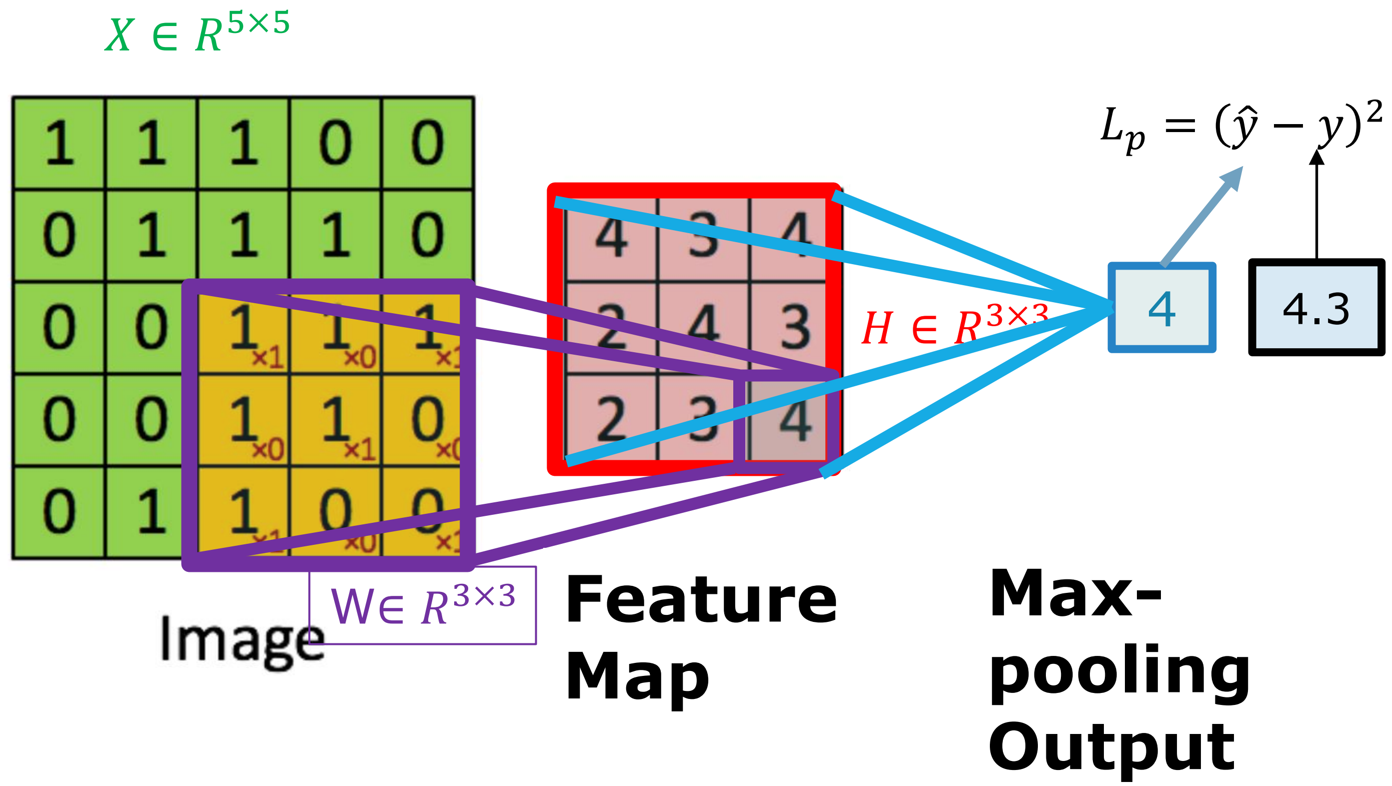
- ✓ Motivation
- ✓ CNN Operations
- Gradient-based Optimization
- LM with word-level and character-level CNNs

Gradient-based Optimization in CNN

- Denode the loss function as $L(y, \hat{y}_W(x))$
  - $\hat{y}_W$ depends on the steps of (padded) convolution and pooling;
  - W consists of the matrix of the learnable filter weights.
- Model Training
  - Optimizing W through backpropagation
  - Differentiable $\frac{\partial \hat{y}_W(x)}{\partial W}$ in both the convolution and pooling steps
  - Leveraging parallel computing on GPU

Optimization in CNN: A Toy Example (1D output)

- $X \in R^{5\times5}$: the input matrix
- $W \in R^{3\times3}$: the kernel matrix
  - Indexed by (k, l)
- $H \in R^{3\times3}$: the feature map;
  - Indexed by (i, j)
- $\hat{Y} \in R$: the system's output
- $Y \in R$: the ground truth

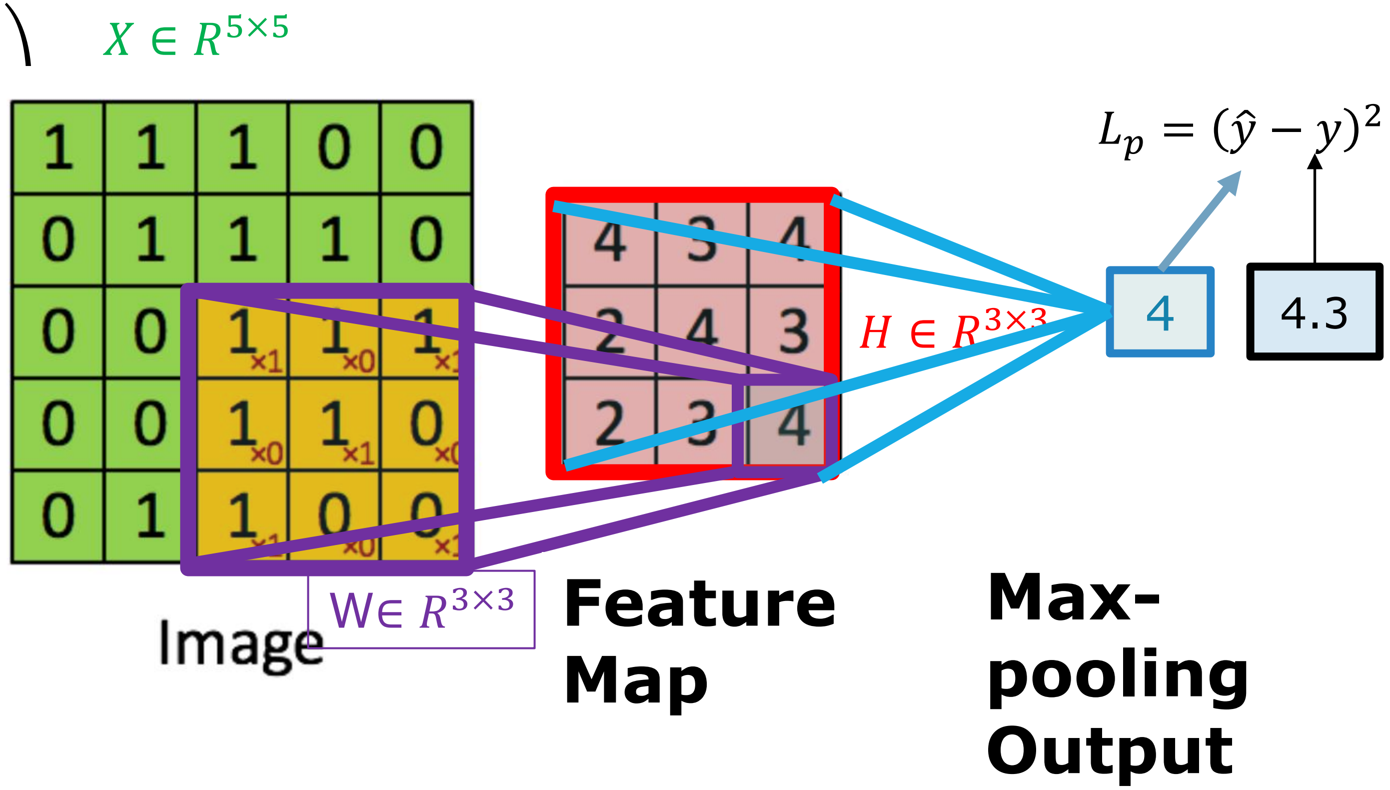


Optimization via stochastic gradient descent

$$W_{kl}{}^{(new)} := W_{kl}{}^{(old)} - \eta_k \nabla\left(\frac{1}{|B|}\sum_{p\in B} I\right)$$

- L_p: loss (on a training pair)
- B: mini-batch of training pairs
- η_k: learning rate at step k

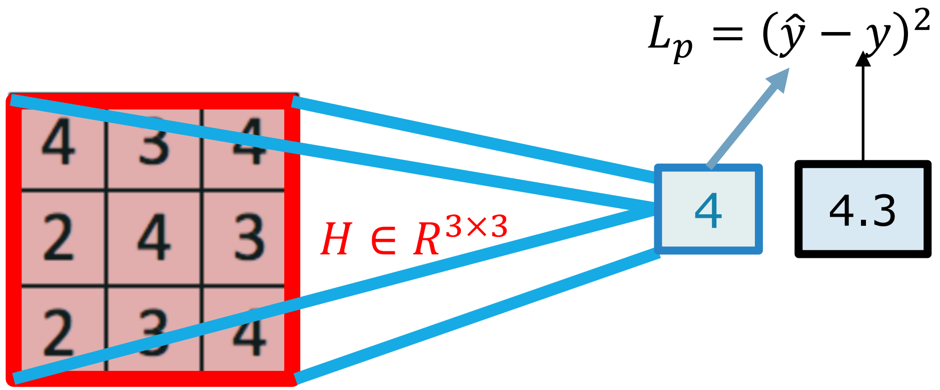
$$\frac{\partial L_p}{\partial W_{kl}} = \frac{\partial L_p}{\partial \hat{y}} \sum_{i,j\in\{1,2,3\}} \frac{\partial \hat{y}}{\partial H_{ij}} \frac{\partial H_{ij}}{\partial W_{kl}}$$



Backpropagation for Max-pooling

- $\frac{\partial L_p}{\partial \hat{y}}$ is trivial, $\frac{\partial L_p}{\partial \hat{y}} = 2(\hat{y} - y)$
- $\frac{\partial \hat{y}}{\partial H_{11}} = 1$
- $\frac{\partial \hat{y}}{\partial H_{12}} = 0$
- $\frac{\partial \hat{y}}{\partial H_{21}} = 0$
- $\frac{\partial \hat{y}}{\partial H_{22}} = 1$
- …

We need to remember the index of the maximum in H



Feature Map

Max-pooling Output

Backpropagation for Convolution

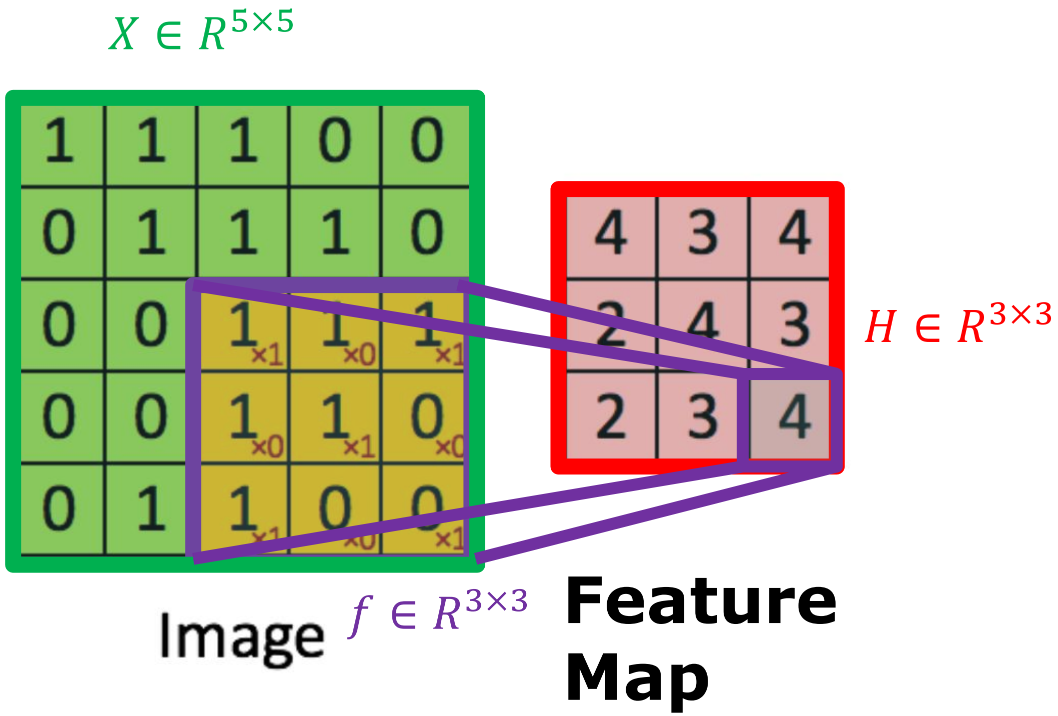
- Now we need the gradient

$$\frac{\partial \hat{y}}{\partial W_{kl}} = \boxed{\sum_{i,j\in\{1,2,3\}}} \frac{\partial \hat{y}}{\partial H_{ij}} \boxed{\frac{\partial H_{ij}}{\partial W_{kl}}}$$

$$= \sum_{i,j\in\{1,2,3\}} \frac{\partial \hat{y}}{\partial H_{ij}} X_{i+k-1,j+l-1}$$

Act like a filter

Over the cells of X



Another convolutional operation (a weighted sum of the input (which has no vanishing gradient issue)

Text classification results in error (smaller is better)

[Xiang Zhang, Junbo Zhao, Yann LeCun, NIPS 2015]

Table 4: Testing errors of all the models. Numbers are in percentage. “Lg” stands for “large” and “Sm” stands for “small”. “w2v” is an abbreviation for “word2vec”, and “Lk” for “lookup table”. “Th” stands for thesaurus. ConvNets labeled “Full” are those that distinguish between lower and upper letters

Non-neural models

Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Red: worst performance on each dataset

Parameters/hyperparameters have a large influence!

Blue: best performance on each dataset

CNN vs. non-neural methods

- Non-neural methods (conventional classifiers)
  - Bag-of-word (BOW) features of input text (with binary or TF-IDF term weighting) assumes independency among words.
  - Bag-of-nGram features of input text (with binary or TF-IDF term weighting) captures local dependencies without dimensionality reduction
- CNN Models
  - Convolution filters and pooling capture local dependencies with dimension-reduced features (the convoluted ones in the feature map) in an efficient way.

Parameters and Other Factors

- Hyperparameter **matters**!
- Hyperparameters include
  - Convolution/pooling size
  - Striding/padding size
  - Dilation scope
- Optimization Algorithms
  - SGD/Ada-grad
  - Mini-batch size in SGE
- In text classification
  - Embedding methods

Embedding Strategies

- Pre-training vs. No pre-training
  - If an unlabeled large corpus is available, pretraining of word embedding can be used for faster fine-tuning and better generalization ability
- Word-level/Character-level embedding
  - Word embedding can be initialized with pretrained embedding, missing words are randomly initiated
  - If too many words are unknown in pretraining, character-level embedding may be used

Outline

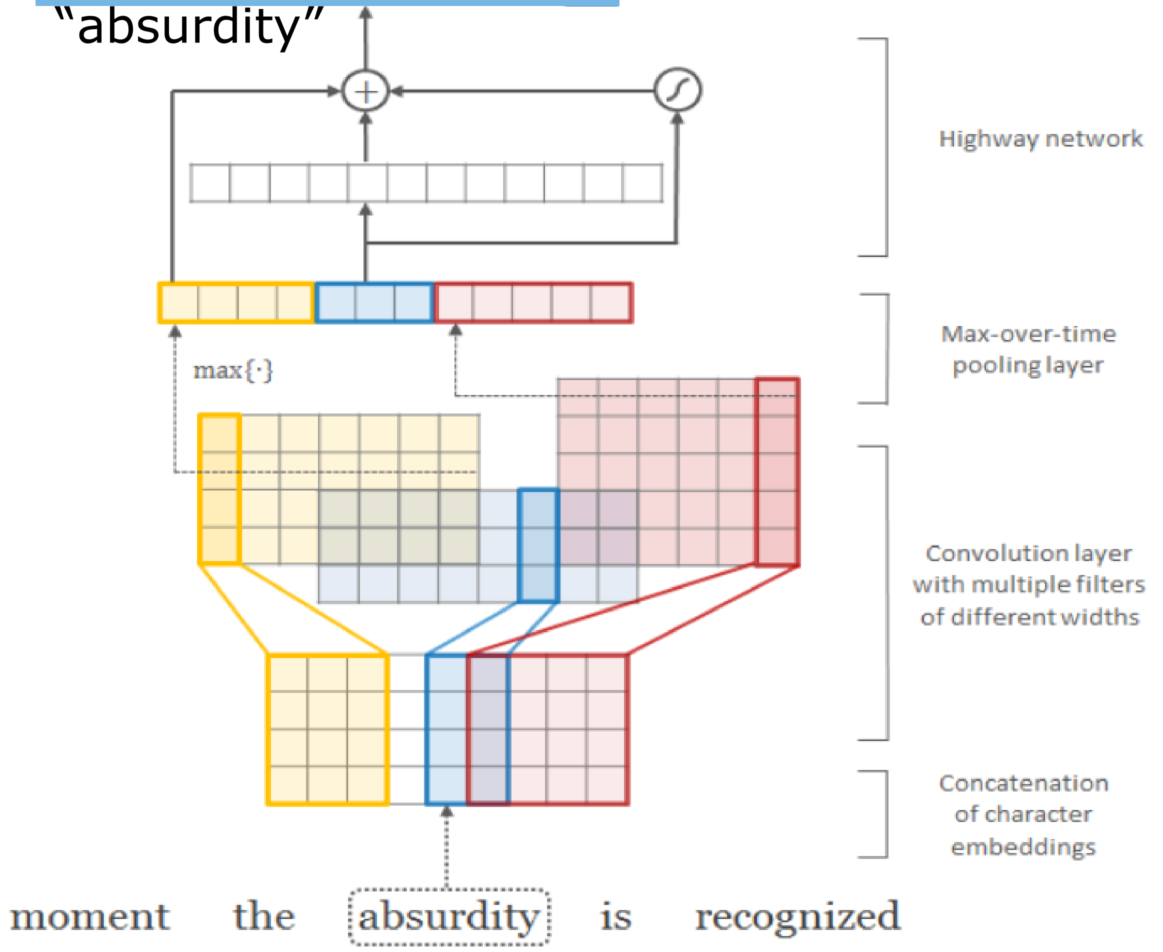
- ✓ Motivation
- ✓ CNN Operations
- ✓ Gradient-based Optimization
- LM with word-level and character-level CNNs

Language modeling with character-level CNN

[Yoon Kim et al. AAAI 2016]

- Input
  - character-level embedding
- Output
  - Word-level embedding
- Aimed Advantage
  - Capture OOV (Out-Of-Vocabulary) words

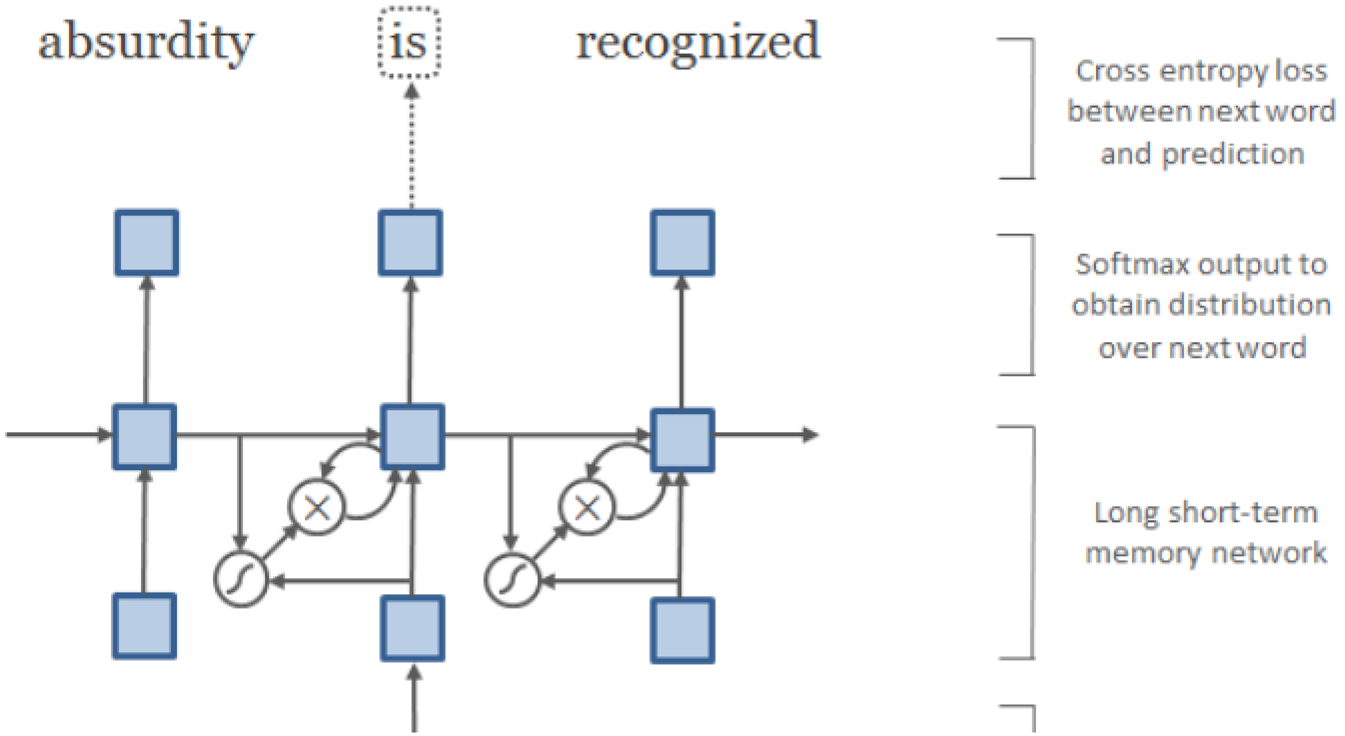
CNN-generated embedding for "absurdity"



Language Modelling with CNN+LSTM

[Yoon Kim et al. AAAI 2016]

- The CNN-generated embedding is then fed into a LSTM for predicting the current word given the past context.



CNN-generated embedding of "absurdity"

CNN-generated embedding

[Yoon Kim et al. AAAI 2016]

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
LSTM-Word	*although* *letting* *though* *minute*	*your* *her* *my* *their*	*conservatives* *we* *guys* *i*	*jonathan* *robert* *neil* *nancy*	*advertised* *advertising* *turnover* *turnover*	– – – –	– – – –	– – – –
LSTM-Char (before highway)	*chile* *whole* *meanwhile* *white*	*this* *hhs* *is* *has*	*your* *young* *four* *youth*	*hard* *rich* *richer* *richter*	*heading* *training* *reading* *leading*	*computer-guided* *computerized* *disk-drive* *computer*	*informed* *performed* *transformed* *inform*	*look* *cook* *looks* *shook*
LSTM-Char (after highway)	*meanwhile* *whole* *though* *nevertheless*	*hhs* *this* *their* *your*	*we* *your* *doug* *i*	*eduard* *gerard* *edward* *carl*	*trade* *training* *traded* *trader*	*computer-guided* *computer-driven* *computerized* *computer*	*informed* *performed* *outperformed* *transformed*	*look* *looks* *looked* *looking*

Table 6: Nearest neighbor words (based on cosine similarity) of word representations from the large word-level and character-level (before and after highway layers) models trained on the PTB. Last three words are OOV words, and therefore they do not have representations in the word-level model.

Performa ng

[Yoon Kim et al. AAAI 2016]

	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Table 3: Performance of our model versus other neural language models on the English Penn Treebank test set. PPL refers to perplexity (lower is better) and size refers to the approximate number of parameters in the model. KN-5 is a Kneser-Ney 5-gram language model which serves as a non-neural baseline. †For these models the authors did not explicitly state the number of parameters, and hence sizes shown here are estimates based on our understanding of their papers or private correspondence with the respective authors.

Comparison with RNN-based models

- RNN
  - Long-term dependency
  - Sequential computation (slow), prone to gradient vanishing
- CNN
  - N-gram-like models focusing on local dependency
  - Computation is faster (easier to be parallelized)
- RNN and CNN can be jointly used in combination!

References

- An Intuitive Explanation of Convolutional Neural Networks, ujjwalkarn, 2016 https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
- AI Shack http://aishack.in/tutorials/image-convolution-examples/
- Character-Aware Neural Language Models. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush., AAAI 2016
- Character-level Convolutional Networks for Text Classification, Xiang Zhang, Junbo Zhao, Yann LeCun. NIPS 2015. http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf