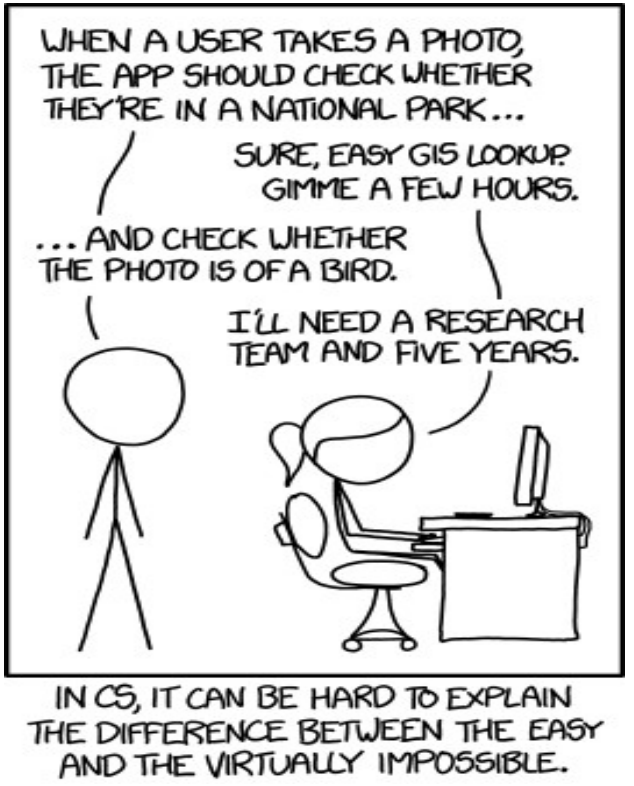
Natural Language and Speech Processing

Lecture 7: Convolutional and Recurrent Neural Networks for Text

Tanel Alumäe

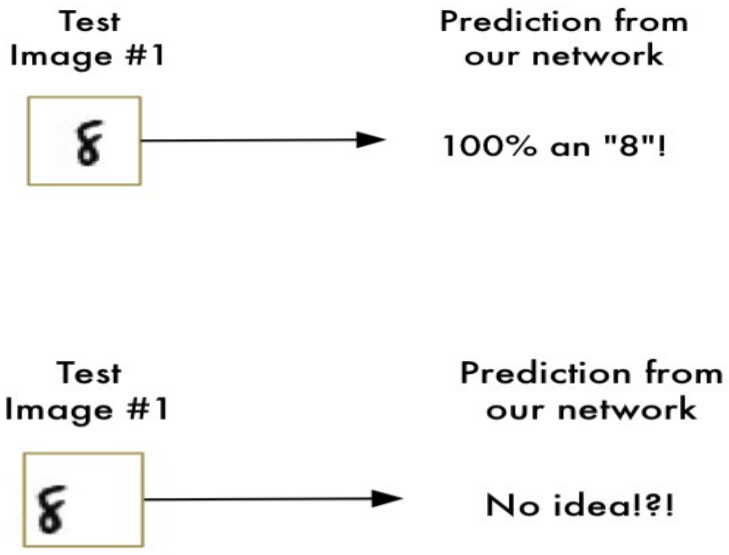
Recognizing objects in images

- Recognizing objects in images is a very active research field
- Most modern systems use convolutional networks
- Available in many end-user systems
  - e.g. search for „cat” or „table” in Google Photos



Why convolutions?

- Example: digit recognition from images
- Works relatively well when the digit is in the centre of the image
- But fails if it’s not in the centre

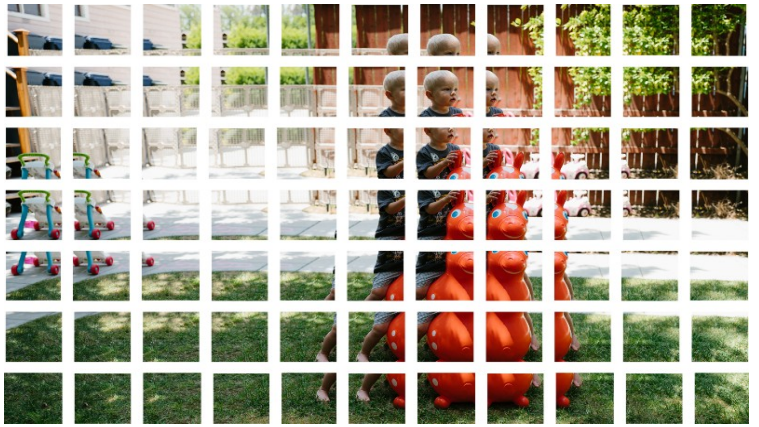


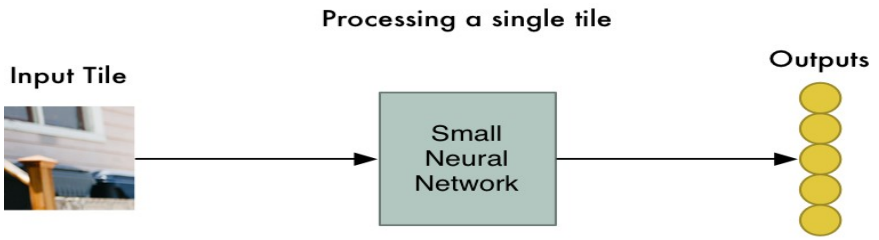
Why convolution?

- Humans recognize instantly that there is a child on the picture
- We recognize the idea of a child no matter what surface the child is on
- We need to give our neural network understanding of translation invariance—an “8” is an “8” no matter where in the picture it shows up

Idea behind convolution

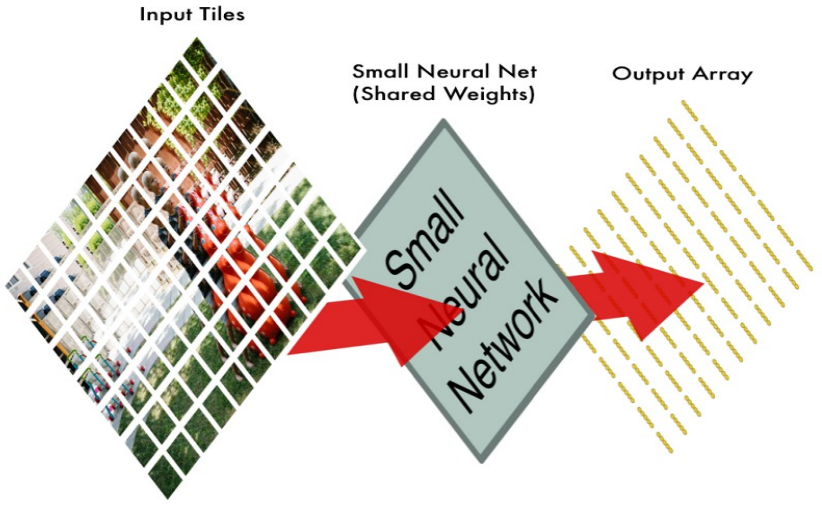
- Break the image into overlapping image tiles
- Feed each image into a small neural network
- The outputs of the small neural network can be trated as high-level features
  - e.g. is there a child's face in the tile?
- We'll keep the same neural network weights for every single tile
  - i.e., the weights are **shared**





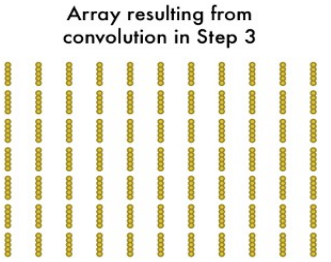
Idea behind convolution, cont.

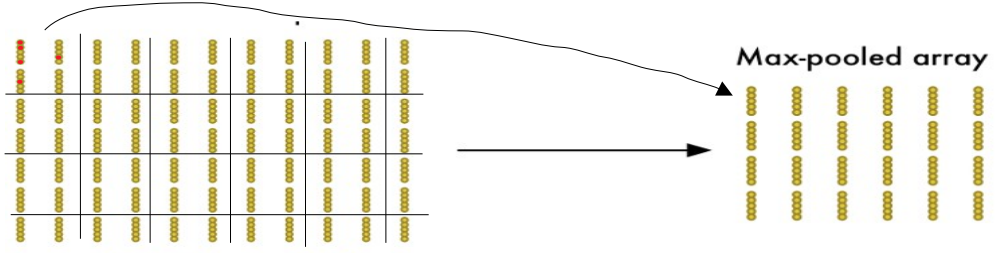
- Save the outputs of the small neural network into a new array
- We’ve started with a large image and we ended with a slightly smaller array that records which sections of our original image were the most „interesting”



Downsampling

- The result from 1st convolution is still pretty big
- To reduce the size of the array, we downsample it using an algorithm called max pooling
- We'll just look at each 2x2 square of the array and keep the biggest number
- Or average pooling





Convolutions for text

- The main idea of convolution: move a “cat identifier” across the image, and apply it in the same way, regardless of the position in the image
- Can we also apply the same idea for text?
  - At first glance, this is not so straightforward: we can not move phrases easily - the meaning will change or we will get something that does not make much sense
- Yes: for text classification, we don’t care much where a very informative phrase exactly is

An absolutely great movie! I watched the premiere with my friends.

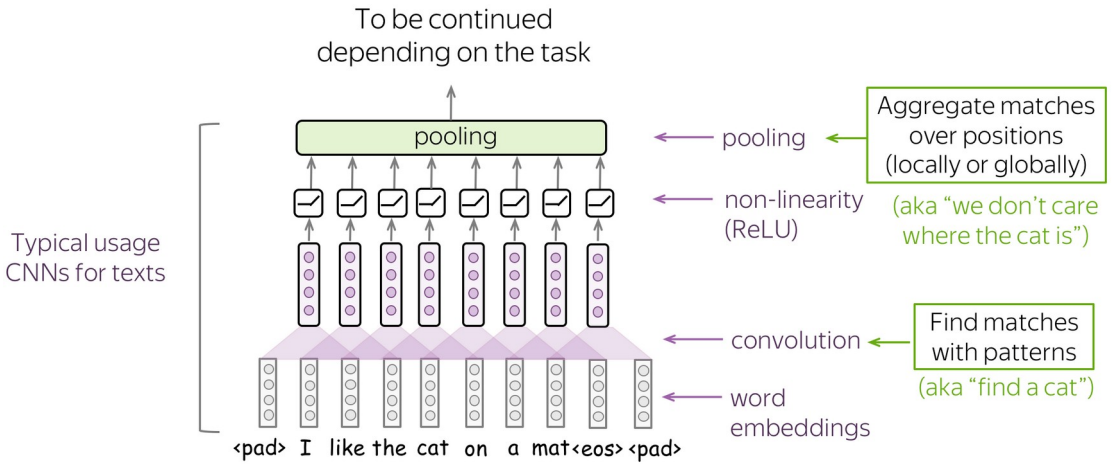
The movie about cats was absolutely great, and the cats were cute.

The movie is about cats running around, and it is absolutely great.

If a clue is very informative, maybe we don’t care much where in a text it appears?

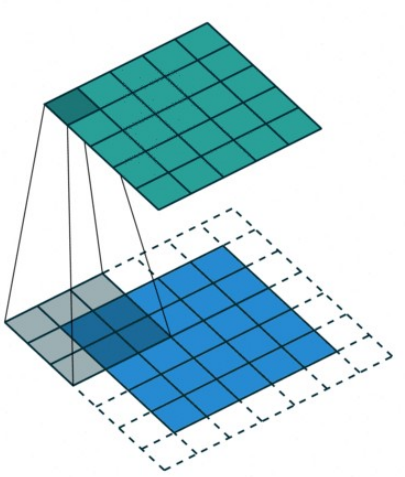
A Typical Model: Convolution+Pooling Blocks

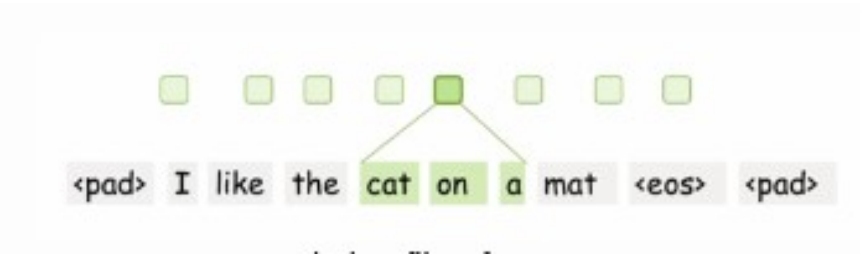
- Two main components:
  - convolution: finds matches with patterns
  - pooling: aggregates these matches over positions (either locally or globally)
- Usually, a convolutional layer is applied to word embedding, which is followed by a non-linearity (usually ReLU) and a pooling operation



Building Blocks: Convolution

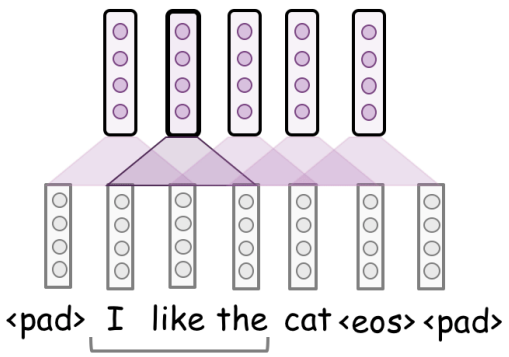
- Convolutions in computer vision go over an image with a sliding window and apply the same operation, convolution filter, to each window
- Texts have only one dimension
  - Therefore, a convolution here is one-dimensional

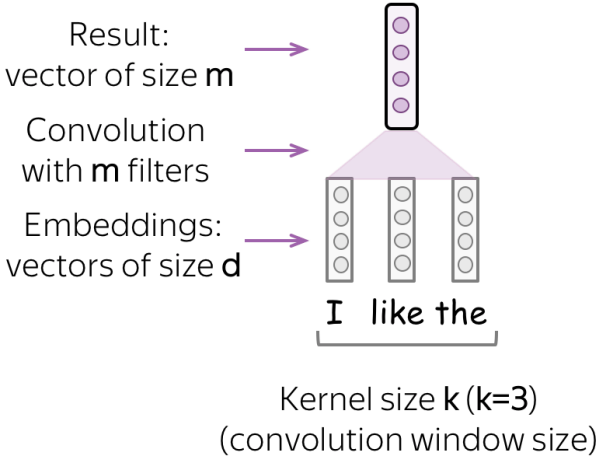


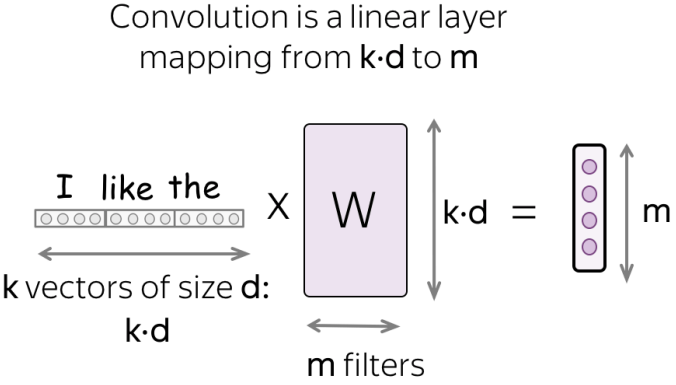


Convolution is a Linear Operation Applied to Each Window

- A convolution goes over an input with a sliding window and applies the same linear transformation to each window

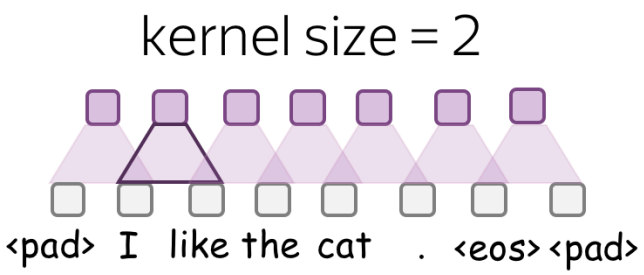


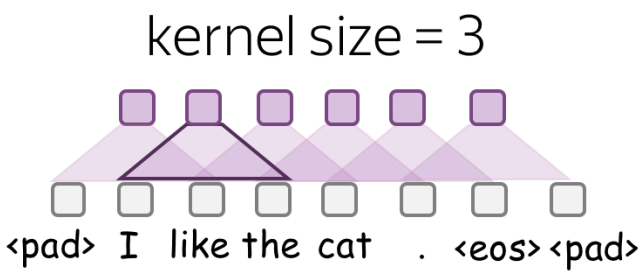


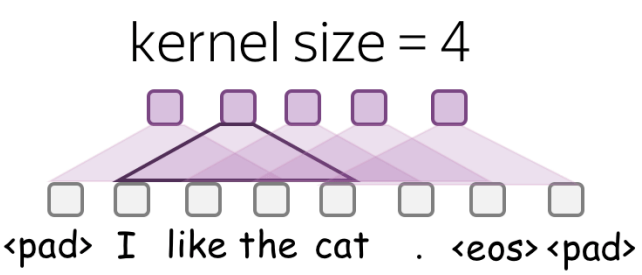


Kernel size

- Kernel size: How far to look
- Kernel size is the number of input elements (tokens) a convolution looks at each step
- For text, typical values are 2-5

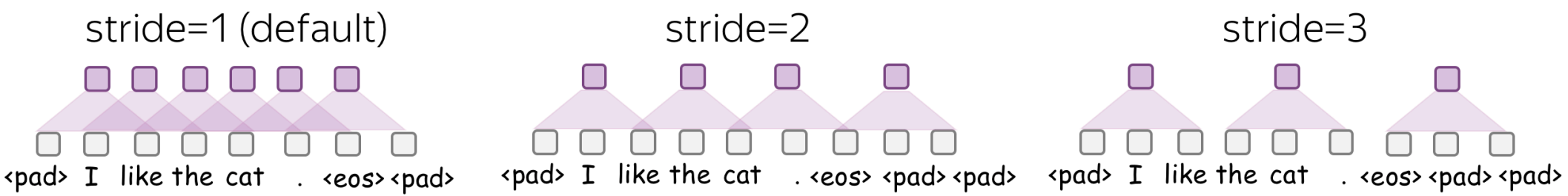






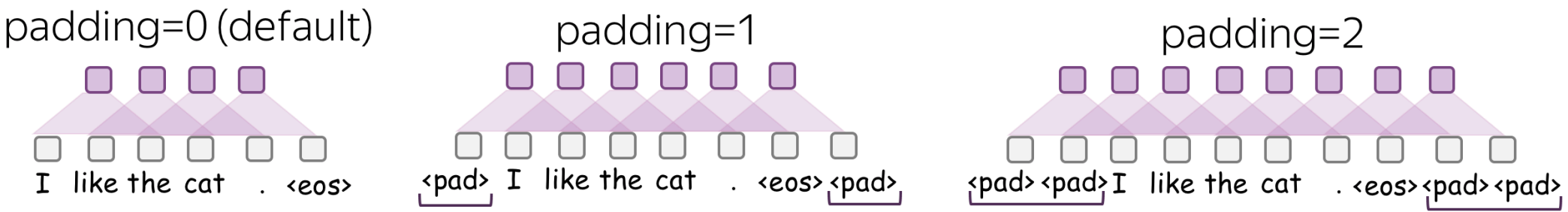
Stride: How much move a filter at each step

- Stride: How much move a filter at each step
- Stride tells how much to move filter at each step
  - For example, stride equal to 1 means that we move the filter by 1 input element (pixel for images, token for texts) at each step.



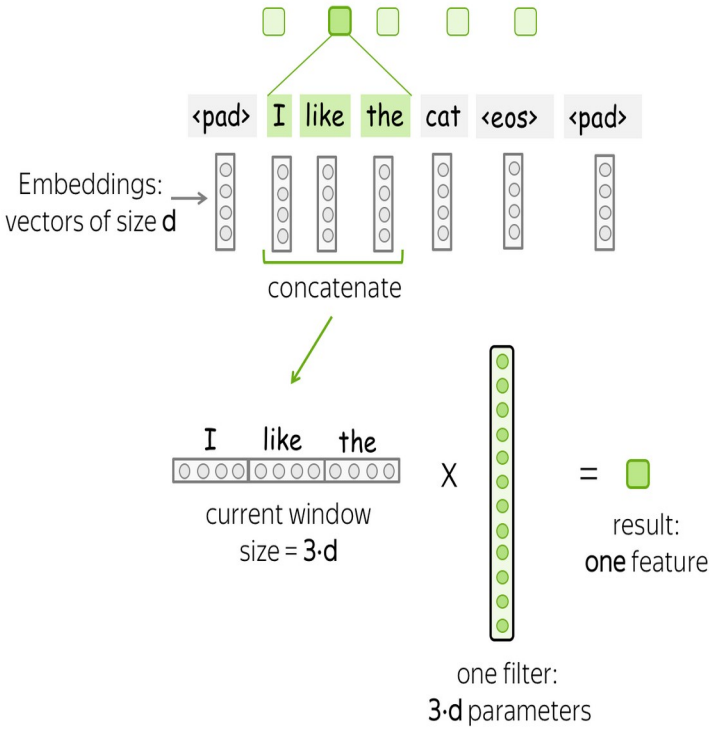
Padding: Add zero vectors to both sides

- Padding: Add zero vectors to both sides
- Padding adds zero vectors to both sides of an input
- If you are using stride>1, you may need padding - be careful!



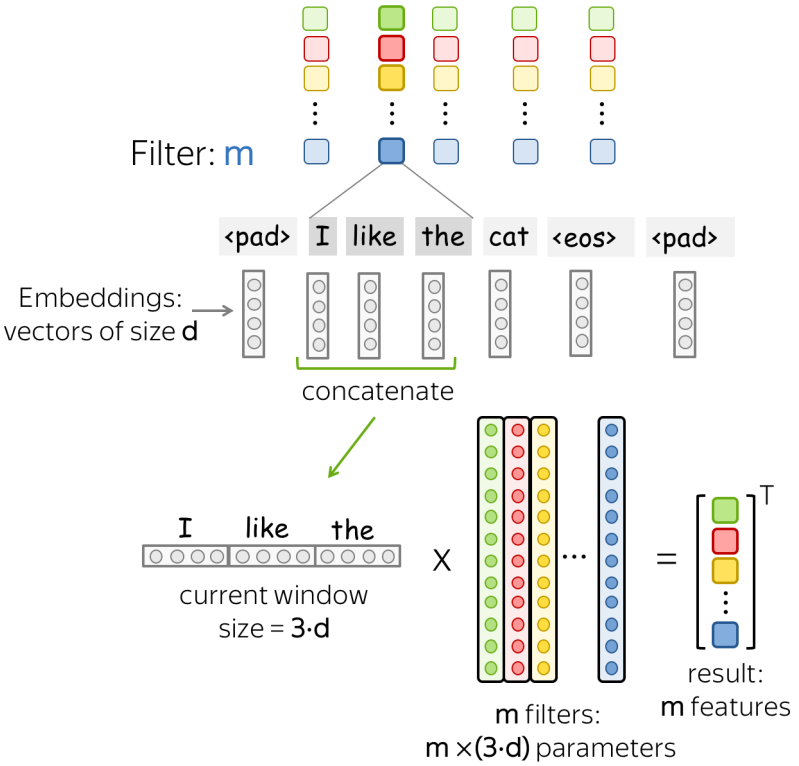
Intuition: Each Filter Extracts a Feature

- One filter - one feature extractor
- A filter takes vector representations in a current window and transforms them linearly into a **single** feature



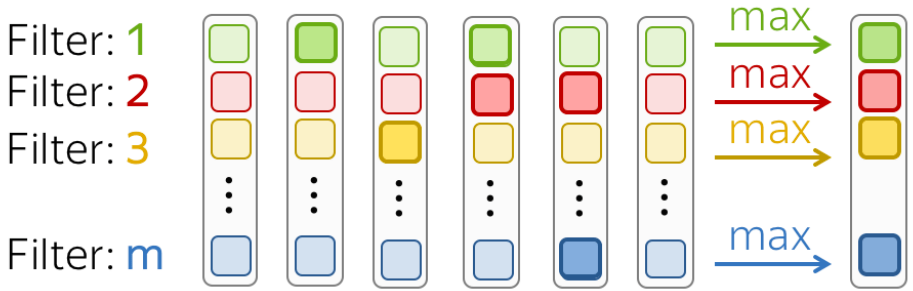
Several filters = several features

- Usually, we want many features
  - For this, we have to take several filters
  - Each filter reads an input text and extracts a different feature
  - The number of filters is the number of output features you want to get
  - Each feature acts as a feature indicator with a local field of view: e.g. “is this word 3-gram a positive phrase?”

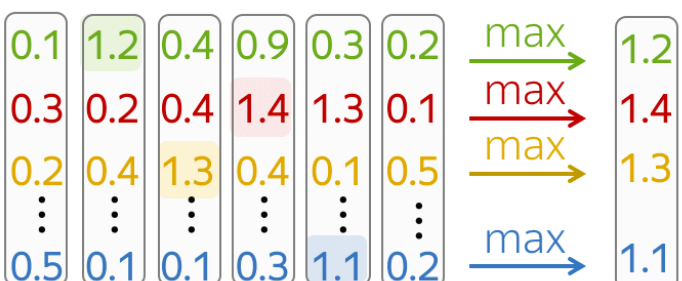


Max and average pooling

- After a convolution extracted features from each window, a pooling layer summarizes the features in some region
- Pooling layers are used to reduce the input dimension, and, therefore, to reduce the number of parameters used by the network
- The most popular is max-pooling
  - It takes maximum over each dimension, i.e. takes the maximum value of each feature
  - Intuitively, each feature "fires" when it sees some text pattern
  - After a pooling operation, we have a vector saying which of these patterns occurred in the input

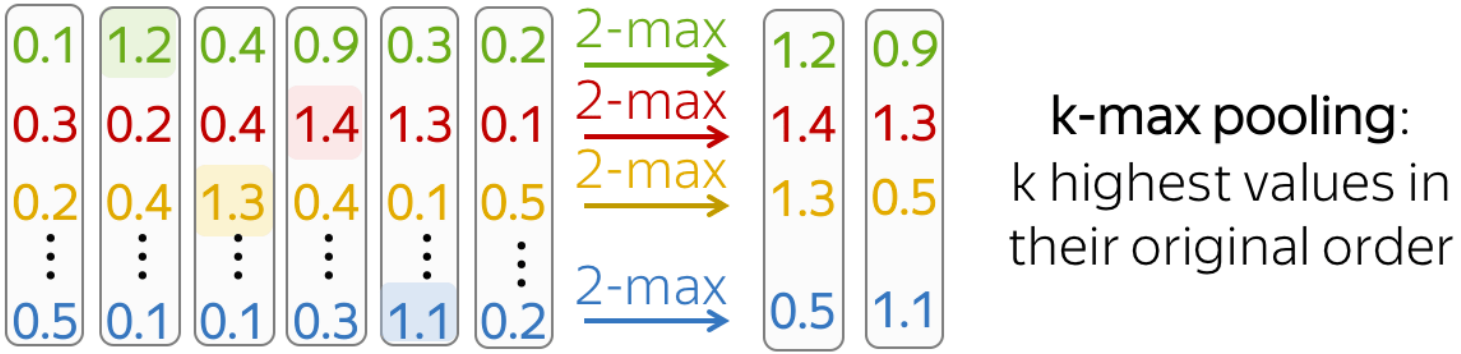


Max pooling: maximum for each dimension (feature)



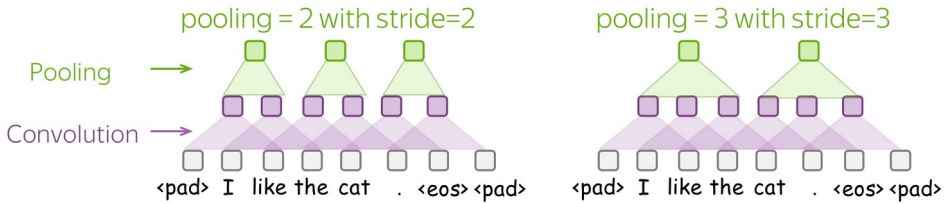
k-max pooling

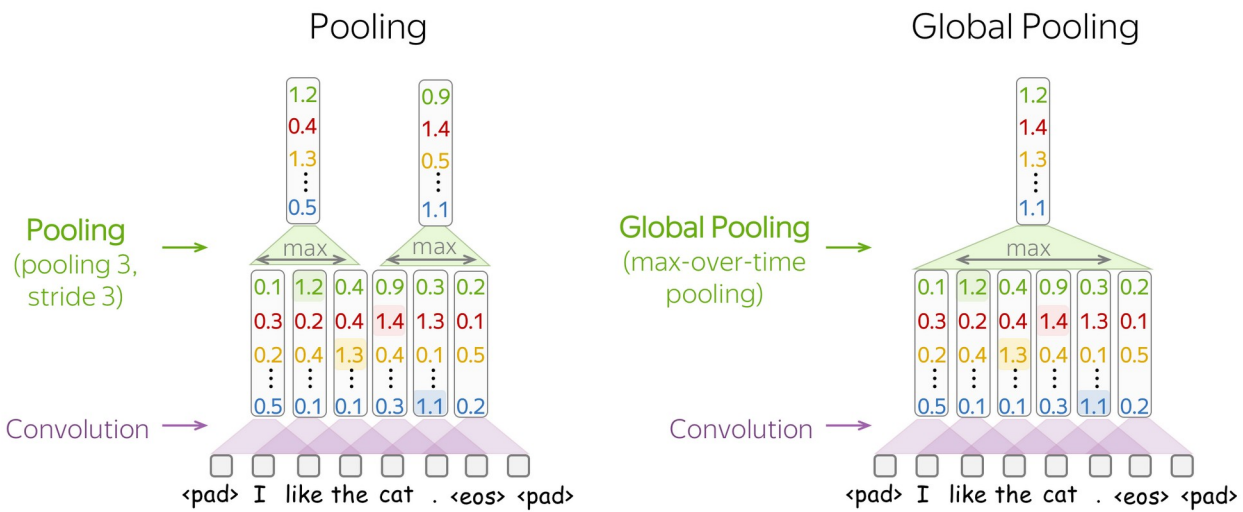
- k-max pooling is a generalization of max-pooling
- Instead of finding one maximum feature, it selects k features with the highest values
- The order of these features is preserved



Pooling and global pooling

- Pooling also has the stride parameter, and the most common approach is to use pooling with non-overlapping windows
  - For this, you have to set the stride parameter the same as the pool size
- The difference between pooling and global pooling is that pooling is applied over features in each window independently, while global pooling performs over the whole input
- Global pooling is often used to get a single vector representing the whole text



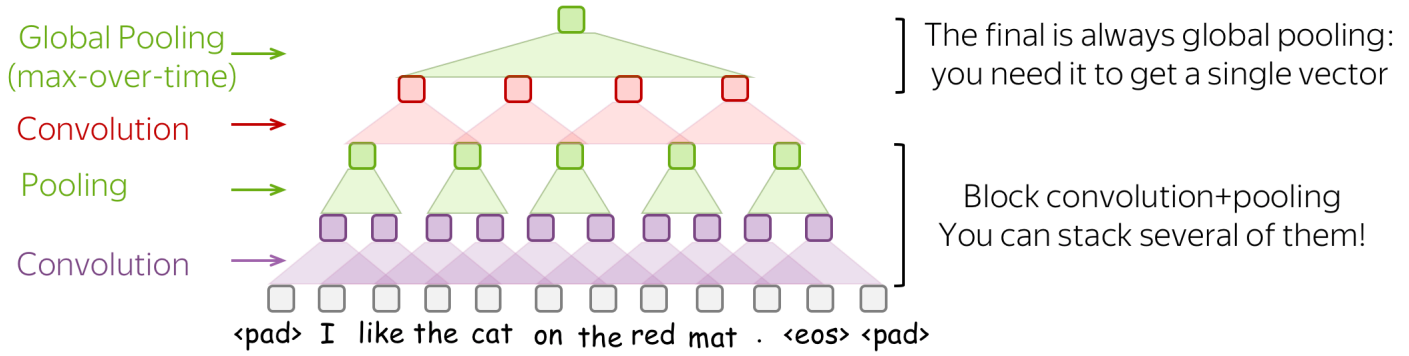


Dynamic pooling

- Max-pooling doesn’t retain positional information
- Sometimes positional information can be important
- Solution: dynamic pooling
- E.g., for document topic classification:
  - Apply one max-pooing pooling for the title
  - Another max-pooling for the 1st paragraph
  - Another for the rest of the document

Stack Several Blocks Convolution+Pooling

- Instead of one layer, you can stack several blocks convolution+pooling on top of each other
- More later convolutions layers operate not on the word embeddings, but on the extracted phrase features



Understanding convolutions

- Convolving filters are used as ngram detectors
  - Each filter specializes in one or several families of closely-related ngrams
- The simplest way to understand what a network captures is to look which patterns activate its neurons
  - For convolutions, we pick a filter and find those n-grams which activate this filter **most**
  - You can see that the n-grams that activate one filter the most have a very similar meaning

filter	Top n-gram	Score
1	poorly designed junk	7.31
2	simply would not	5.75
3	a minor drawback	6.11
4	still working perfect	6.42
5	absolutely gorgeous .	5.36
6	one little hitch	5.72
7	utterly useless .	6.33
8	deserves four stars	5.56
9	a mediocre product	6.91

Top n-grams for filter 4		Score
1	still working perfect	6.42
2	works - perfect	5.78
3	isolation proves invaluable	5.61
4	still near perfect	5.6
5	still working great	5.45
6	works as good	5.44
7	still holding strong	5.37

A filter activates for a family of n-grams with similar meaning

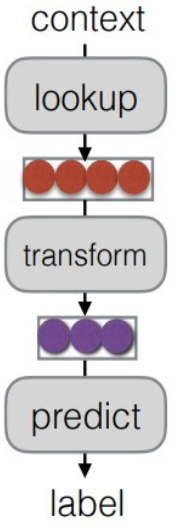
Recurrent neural networks

Long-distance dependencies

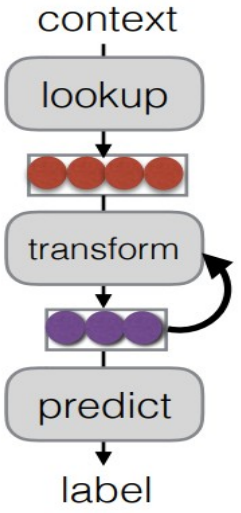
- Agreement in number, gender, etc
  - ***He*** *does not have very much confidence in* ***himself****.*
  - ***She*** *does not have very much confidence in* ***herself****.*
- Selectional preference
  - *The* ***reign*** *has lasted as long as the life of the* ***queen****.*
  - *The* ***rain*** *has lasted as long as the life of the* ***clouds****.*

Recurrent neural networks

- Feed-forward neural network

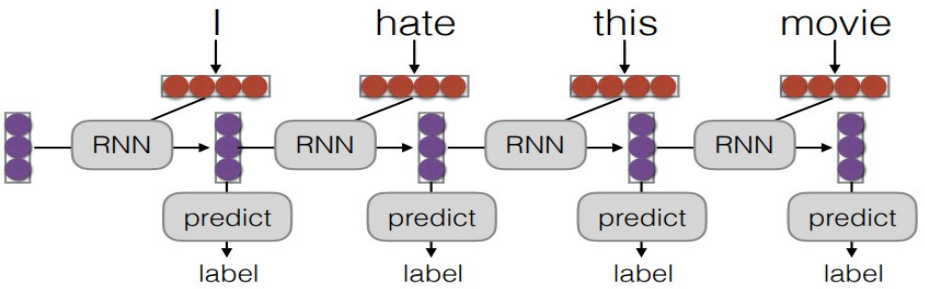


- Recurrent neural network



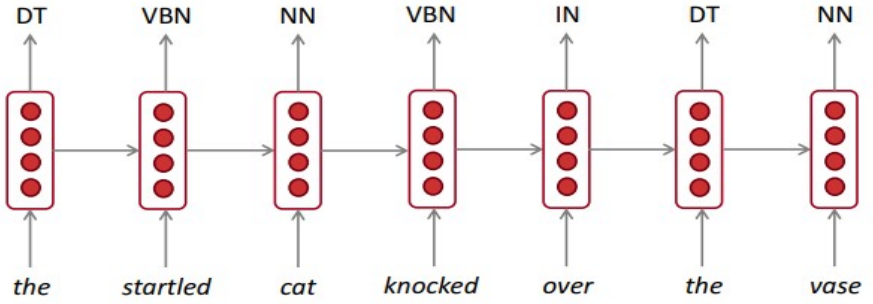
Unrolling a neural network in time

- Processing a sequence (e.g. a sentence) with a neural network
- The current output of RNN depends on the hidden state at the prior state (a **vector**) and current input (another **vector**)
- In NLP terms, the prediction for the current output word depends on the current input word and the hidden layer output of the previous word
- This is the reason RNNs are considered to have **memory**



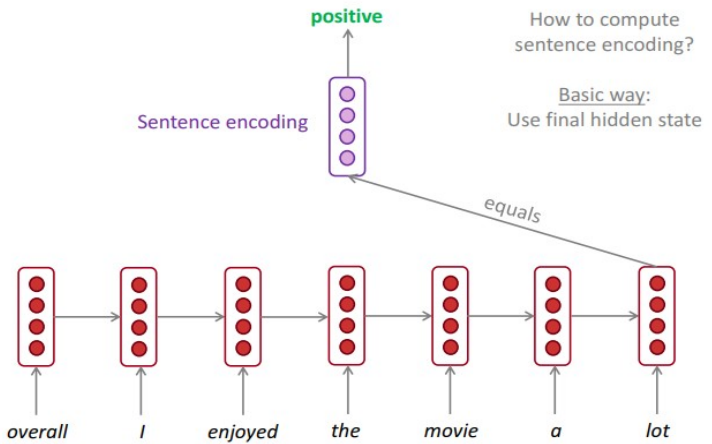
RNNs for word labeling

- RNNs are very flexible and can be customized for different tasks
- Below: RNN for word labeling
  - i.e., classify every element of a sequence



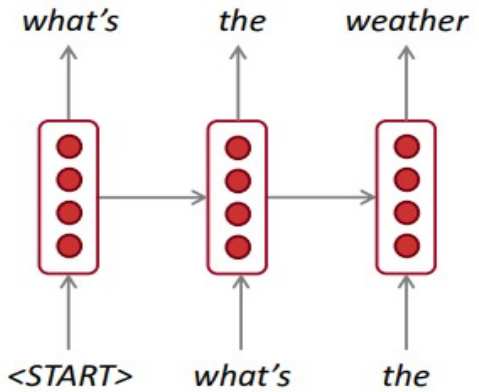
RNN for sequence labeling

- RNNs can be used for labeling the whole sequence
  - E.g., for classifying sentences/documents
- Simplest: just use the final hidden state for classification
  - The RNN should learn to propagate the most important information through the sentence



RNN for generating text

- RNN can be used for generaing text
  - Start with some random initial state, use a pseudo-word <START> as the initial input
  - Generate a word (or character)
    - **Sample** it from the output distribution (**sample** = pick a random word, with the given probability)
  - Feed it back to the same RNN, and generate another word



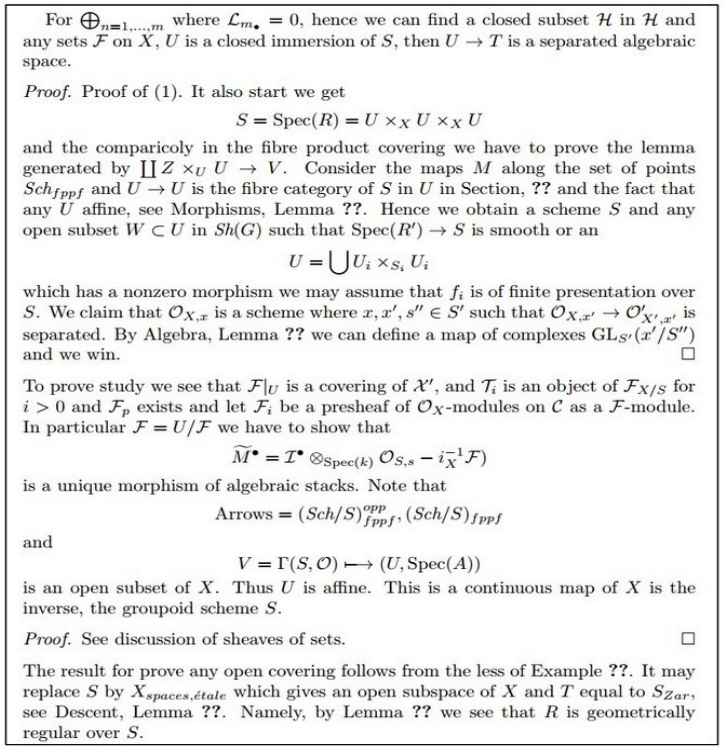
RNN text generation samples I

- **Shakespeare**:
  ```
  KING LEAR:
  O, if you were a feeble sight, the courtesy of your law,
  Your sight and several breath, will wear the gods
  With his heads, and my hands are wonder'd at the deeds,
  So drop upon your lordship's head, and your opinion
  Shall be against your honour.
  ```

- **Wikipedia**:
  ```
  Naturalism and decision for the majority of Arab countries' capitalide
  was grounded by the Irish language by [[John Clair]], [[An Imperial
  Japanese Revolt]], associated with Guangzham's sovereignty. His
  generals were the powerful ruler of the Portugal in the [[Protestant
  Immineners]], which could be said to be directly in Cantonese
  Communication, which followed a ceremony and set inspired prison,
  training.
  ```

RNN generated text, II

- PDF compiled from generated LaTeX source (with some minor syntactic mistakes hand-fixed)



For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariconly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition* **??** *(without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

RNN generated text, III

- Generated baby names, based on 8000 actual names (only generated names not in the training data are listed): *Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacrie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa Wallon Martine Merus Jelen Candica Wallin Tel Rachene Tarine Ozila Ketia Shanne Arnande Karella Roselina Alessia Chasty Deland Berther Geamar Jackein Mellisand Sagdy Nenc Lessie Rasemy Guen Gavi Milea Anneda Margoris Janin Rodelin Zeanna Elyne Janah Ferzina Susta Pey Castina*
- More on this: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN for conditional text generation

- As before, but the initial state encodes some input (e.g. an image)
  - E.g., use a convolutional neural network to reduce the image to a fixed length feature vector
- Generate text as before
- Can be used for image captioning
- Application: help blind people to understand images/scenes

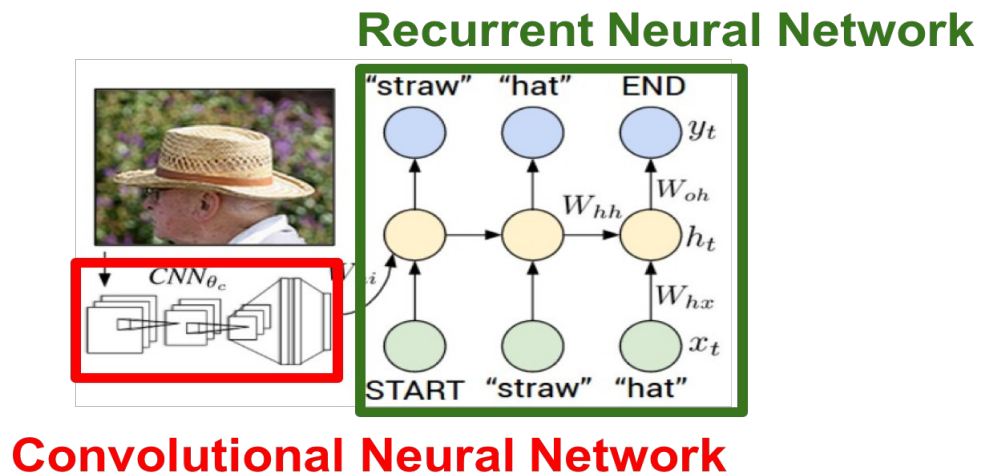
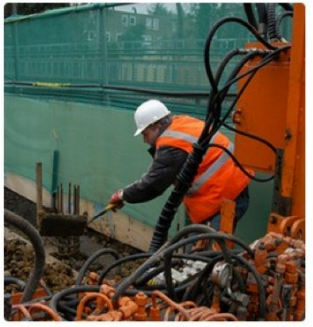


Image captioning examples

"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

"a young boy is holding a baseball bat."

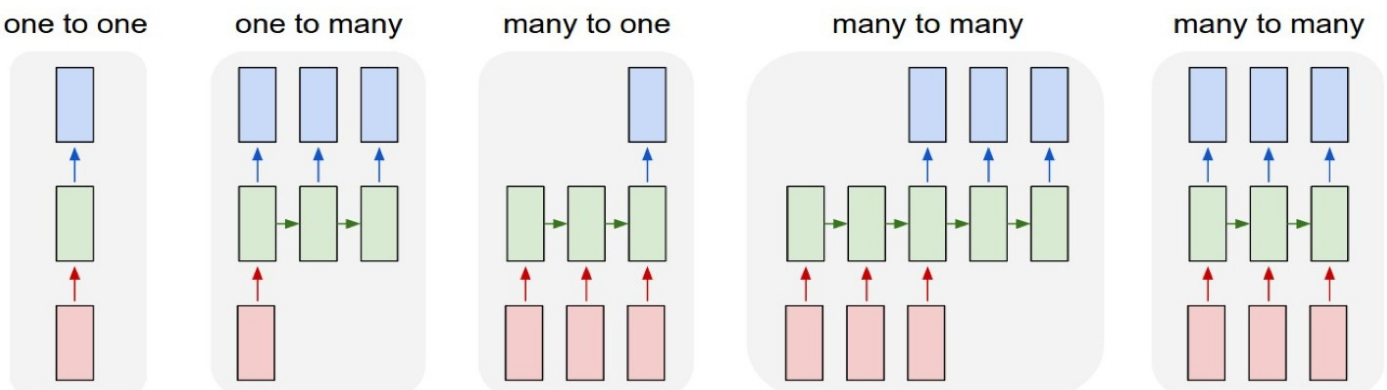
"a cat is sitting on a couch with a remote control."

"a woman holding a teddy bear in front of a mirror."

"a horse is standing in the middle of a road."

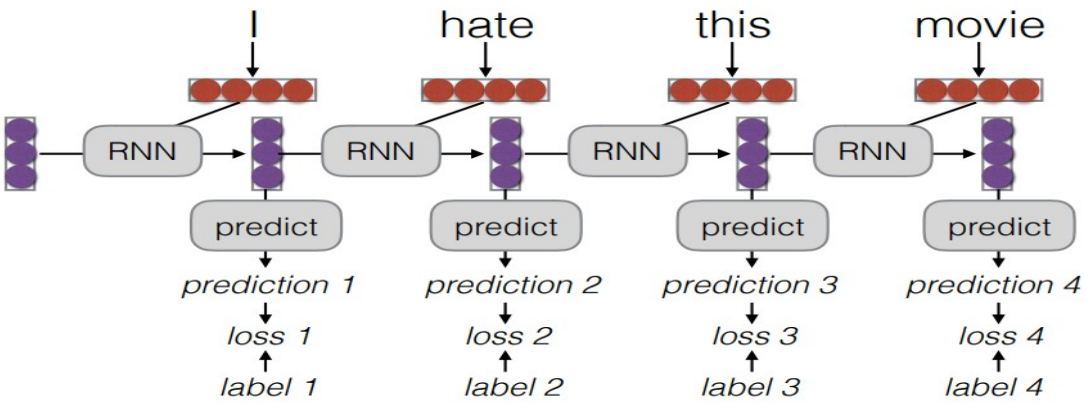
Summary of neural network architectures

- One-to-one: feed-forward DNN we have studied previously (e.g. document classification with CNN, named entity recognition)
- One-to-many: image captioning
- Many-to-one: document classification using RNN
- Many-to-many: machine translation
- Many-to-many (2): word classification using RNN



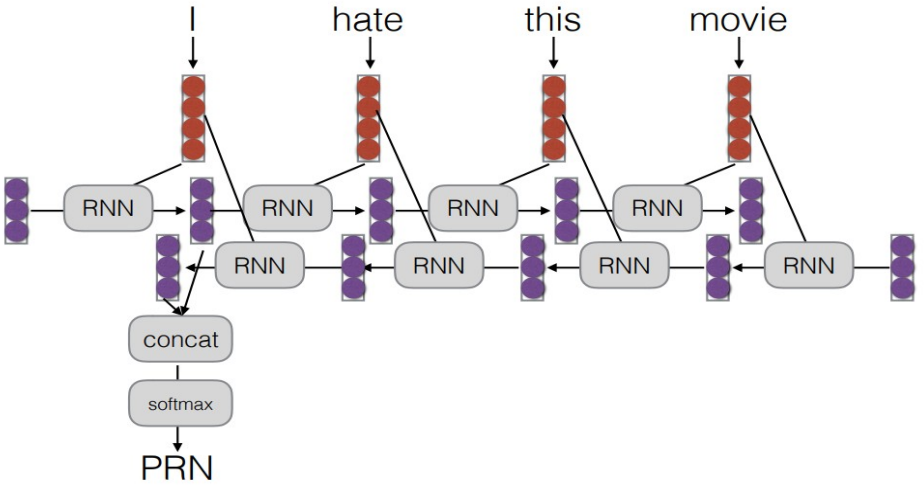
Training RNN

- Training RNN can be done using backpropagation, as with feed-forward NNs
- As before, we (or our toolkit) need to have a computational graph representing the RNN and its input and ouputs
- Training can be better understood using an **unrolled RNN**



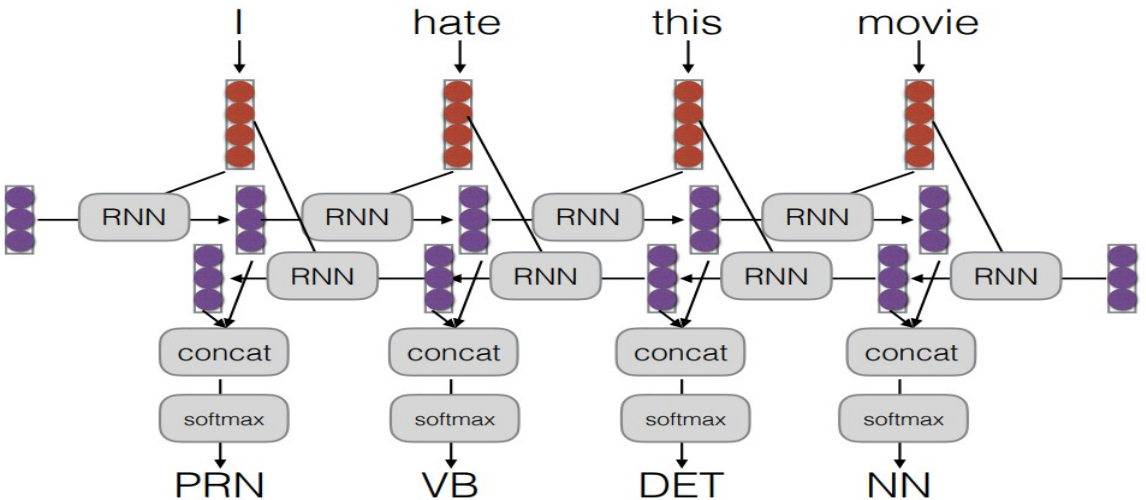
Bidirectional RNNs

- Simple extension: have two RNNs, running in both directions
- The hidden states from the two RNNs are simply concatenated



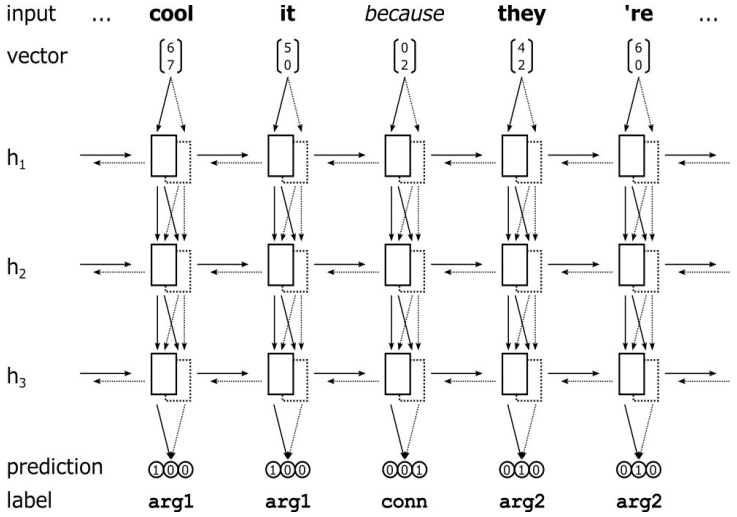
Bidirectional RNNs

- Why?
  - Word classification: enables to use information from both past and future words
  - Alternative: use **label delay** – predict the label of the previous word when consuming the current word



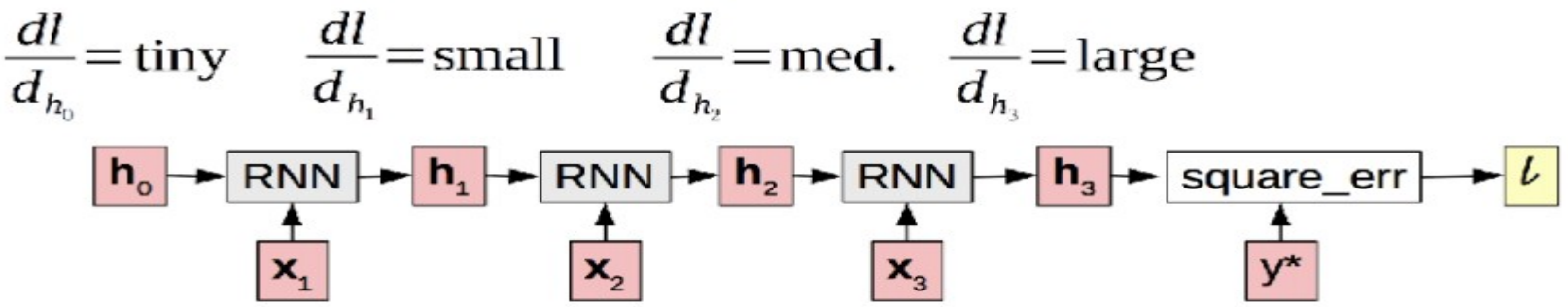
Deep RNNs

- We can stack many recurrent layers
- Each subsequent layer uses the output of the previous recurrent layer as input



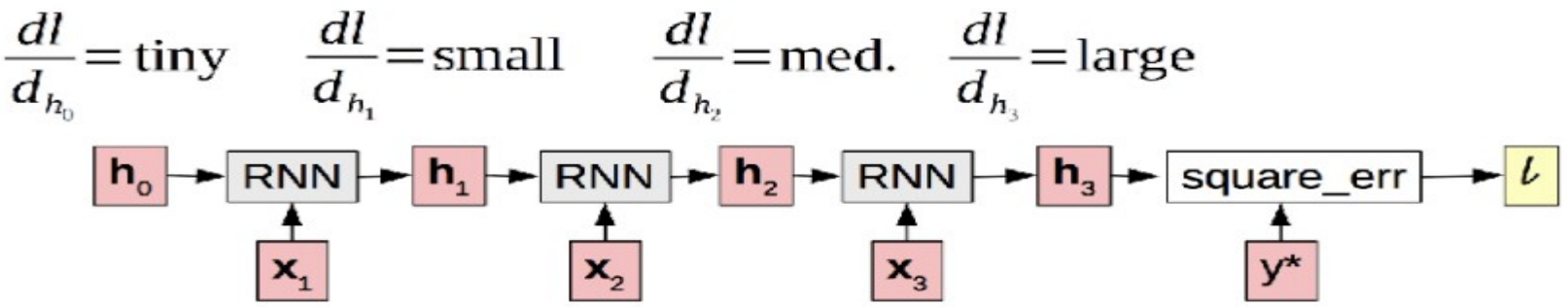
Vanishing Gradients

- Result: the RNN doesn’t learn to use information that is **many time steps in the past**
- Or, (in the exploding case), the parameters of the RNN get quickly very large and the RNN starts producing nonsense



Vanishing Gradients

- Gradients decrease (vanish) or increase (explode) when they get pushed back through the RNN
- They get „squashed” through non-linearities and small (or large) weight matrices

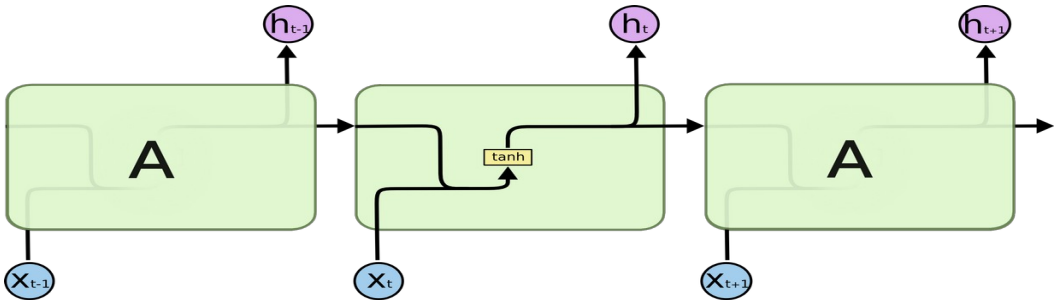


Solution to vanishing gradients: gated units

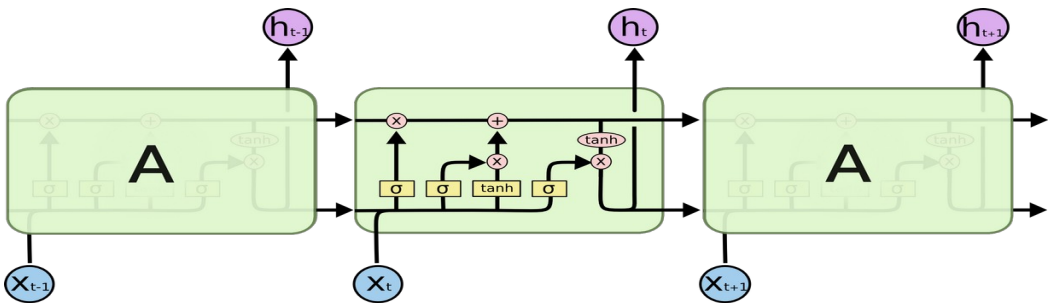
- The main solution to the vanishing gradient problem is to use a more complex hidden unit in the recurrent cell
- Gated Recurrent Unit (**GRU**) and Long Short-Term Memory (**LSTM**)
- Main ideas:
  - Make additive connections between time steps (not multiplicative)
    - Addition does not modify the gradient -- no vanishing
  - Allow gradients to flow through the recurrent cell, depending on the input

LSTM

- Standard RNN



- LSTM: excellent explanation:
  https://colah.github.io/posts/2015-08-Understanding-LSTMs/



LSTM vs RNN

- When we talk about remarkable results achieved with RNNs, we really usually talk about LSTMs
- They really work better than raw RNNs for most things
- GRU is similar to LSTM, but a bit simpler
- GRU might work better than LSTM if your training data is rather limited (and it’s faster)
- LSTM formulas seem intimidating
  - But don’t worry – all NN toolkits have them implemented for you as basic units
  - It’s still good to have an idea what LSTM is doing

RNN strengths and weaknesses

- RNNs, particularly LSTMs have much more modeling power than feed-forward NNs
- However, they can take time to tune
  - Often, initial experiments produce disappointing results
  - Sensitive to learning rate, model initialization, scale of the features, etc
  - Tuning them is even deeper black magic than for feed-forward NNs
- They also require a lot of training data

Pre-training/Transfer

- Idea:
  - First train the RNN for the simpler task that has a lot of training data
  - Then, use the pretrained RNN as part of a new model for the more complex task
- Example: LM -> Sentence Classifier
  - First train a RNN to do language modeling (predicting the next word) – lots of data available
  - Then use the trained RNN as a starting point for training a sentence classification model

- Questions?