Text Classification

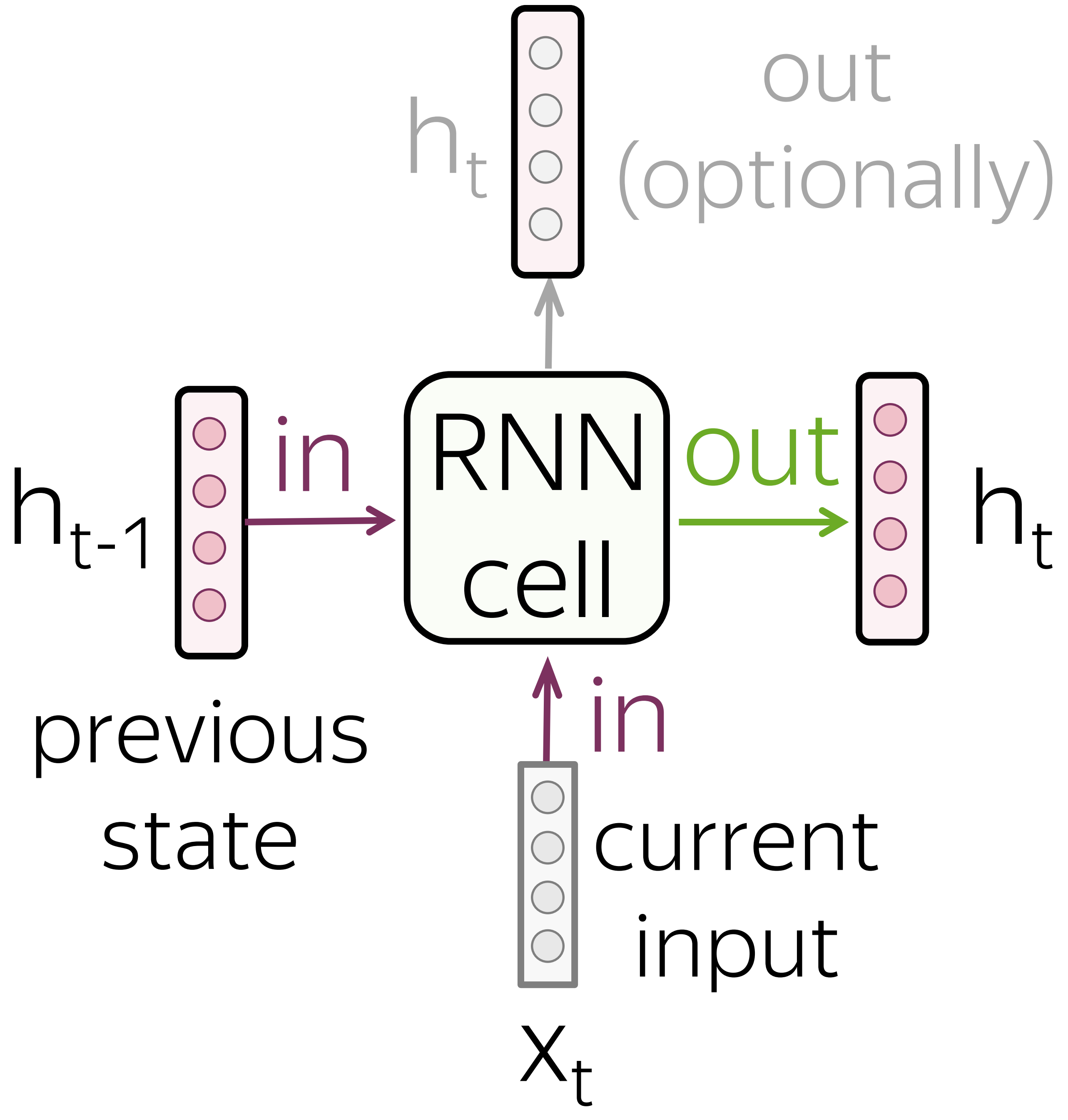
Lena Voita

What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
  - High-Level View
  - Training: Cross-Entropy
  - Models: (Weighted) BOW
  - Models: Convolutional
  - Models: Recurrent
- Multi-Label Classification
- Practical Tips
- Analysis and Interpretability

Basics: Recurrent Neural Networks

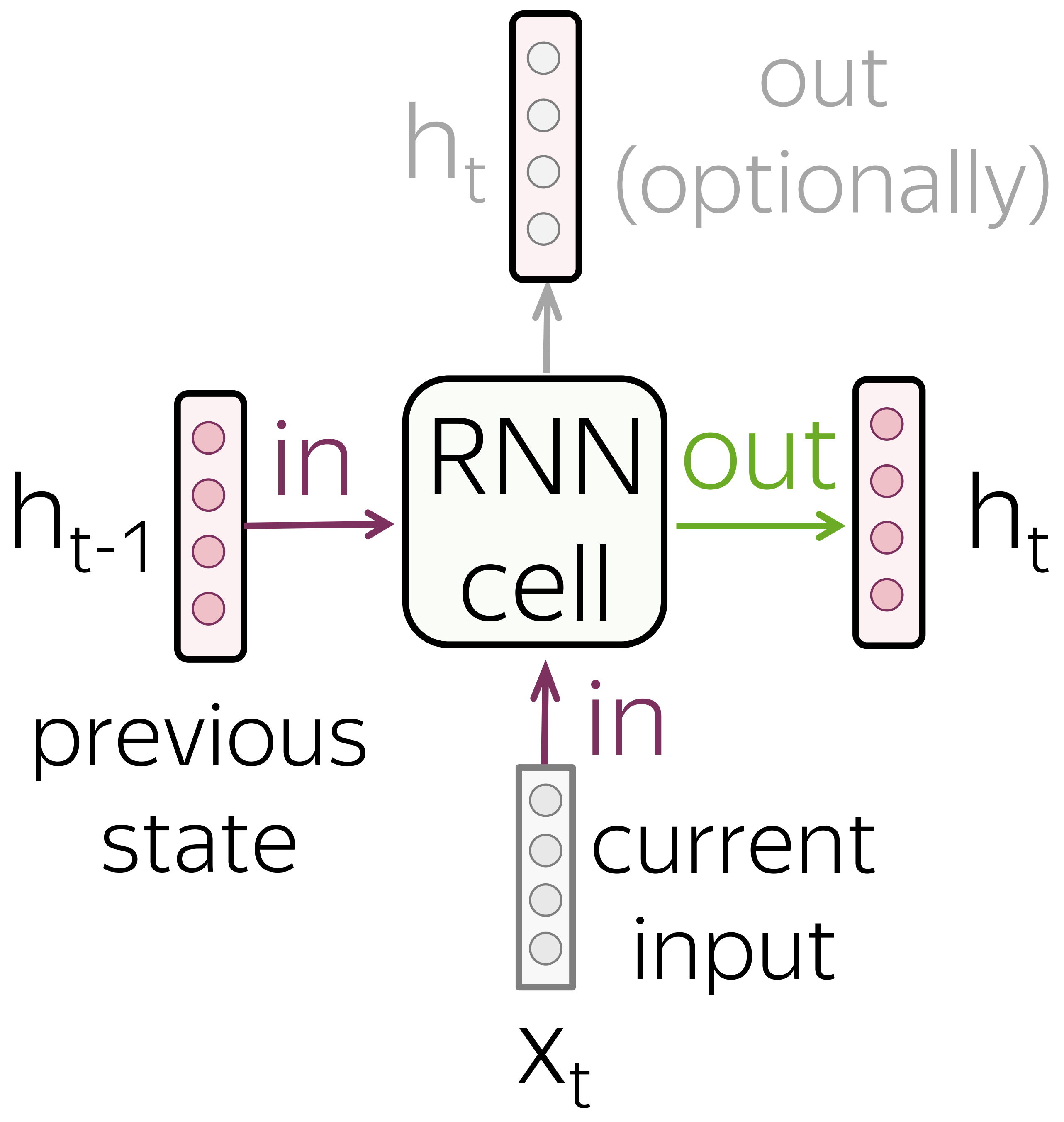
- recurrent cell



For more details of RNN basics, look at the Colah's blog post.

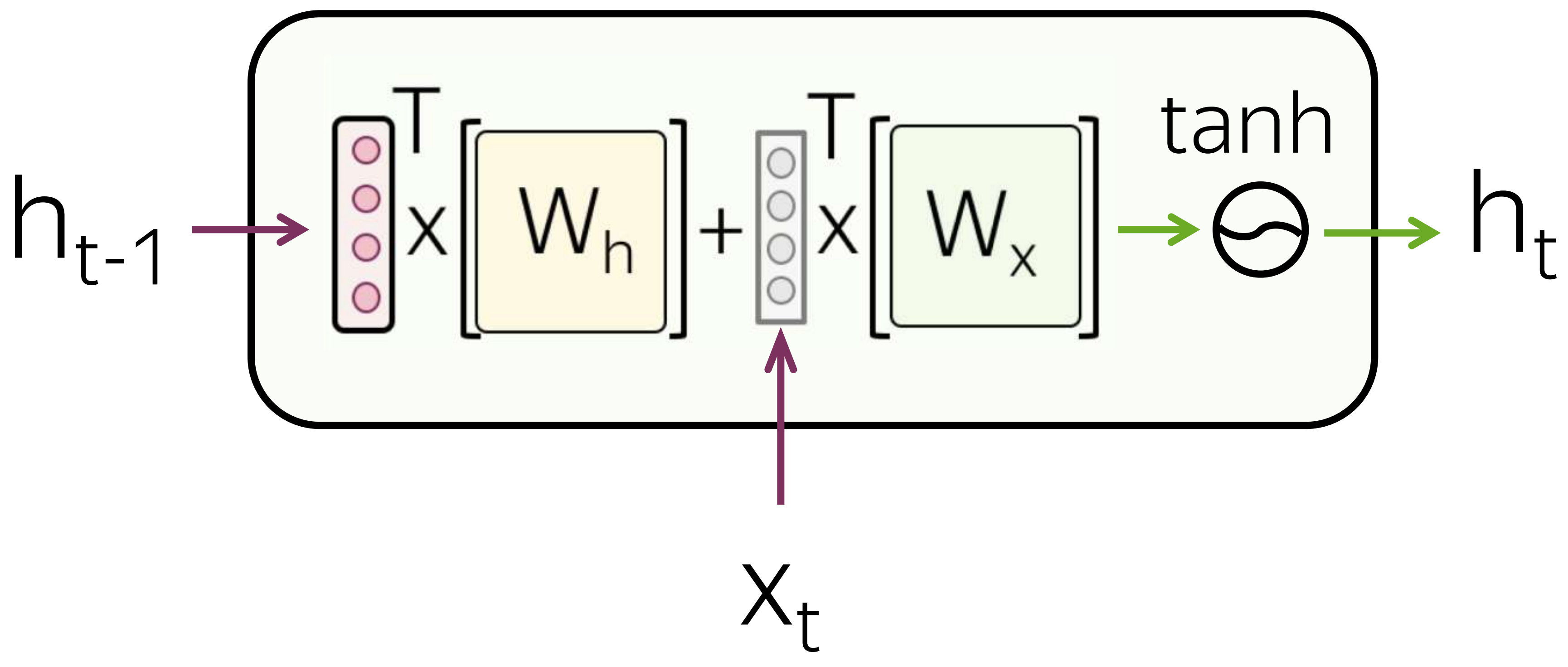
Basics: Recurrent Neural Networks

- recurrent cell



- vanilla RNN

$$h_t = \tanh(h_{t-1}W_h + x_t W_x)$$

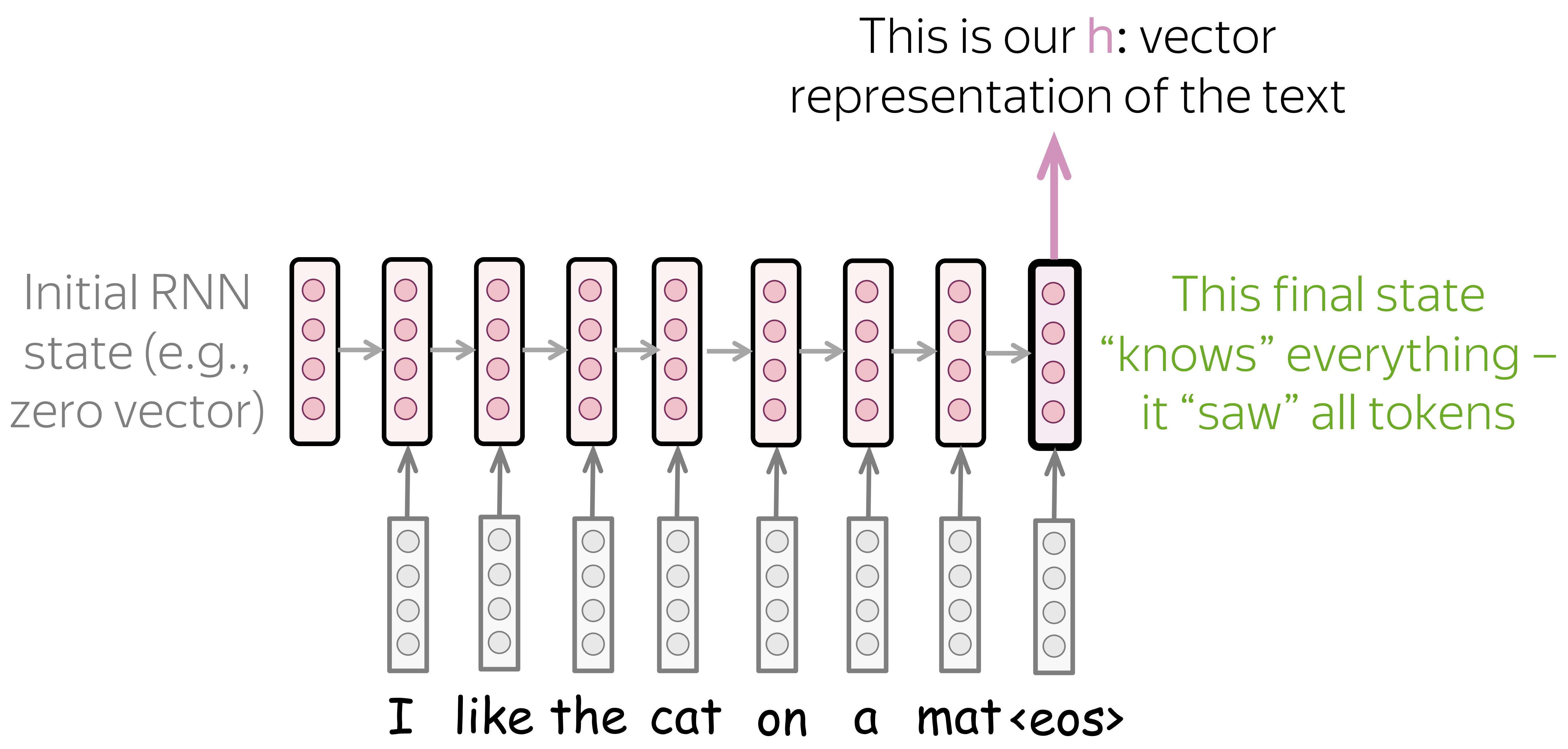


For more details of RNN basics, look at the Colah's blog post.

Recurrent Models for Text Classification

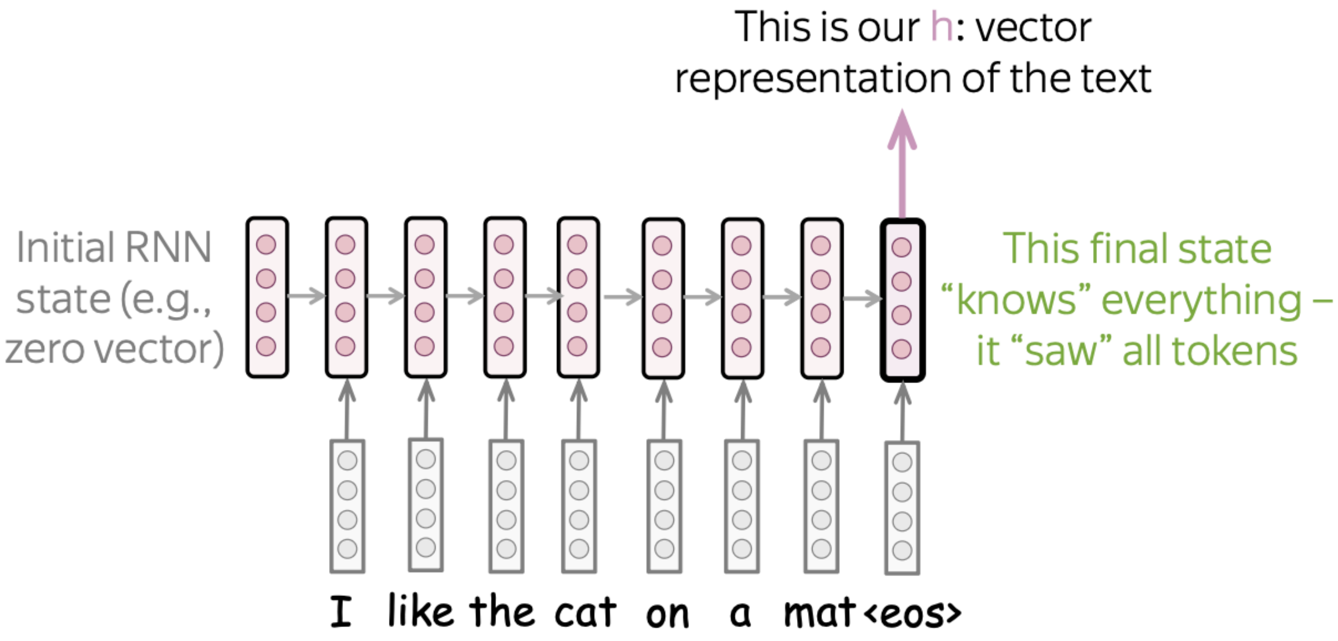
> We need a model that can produce a **fixed-sized** vector for inputs of **different** lengths.

- **simple**: read a text, take the final state



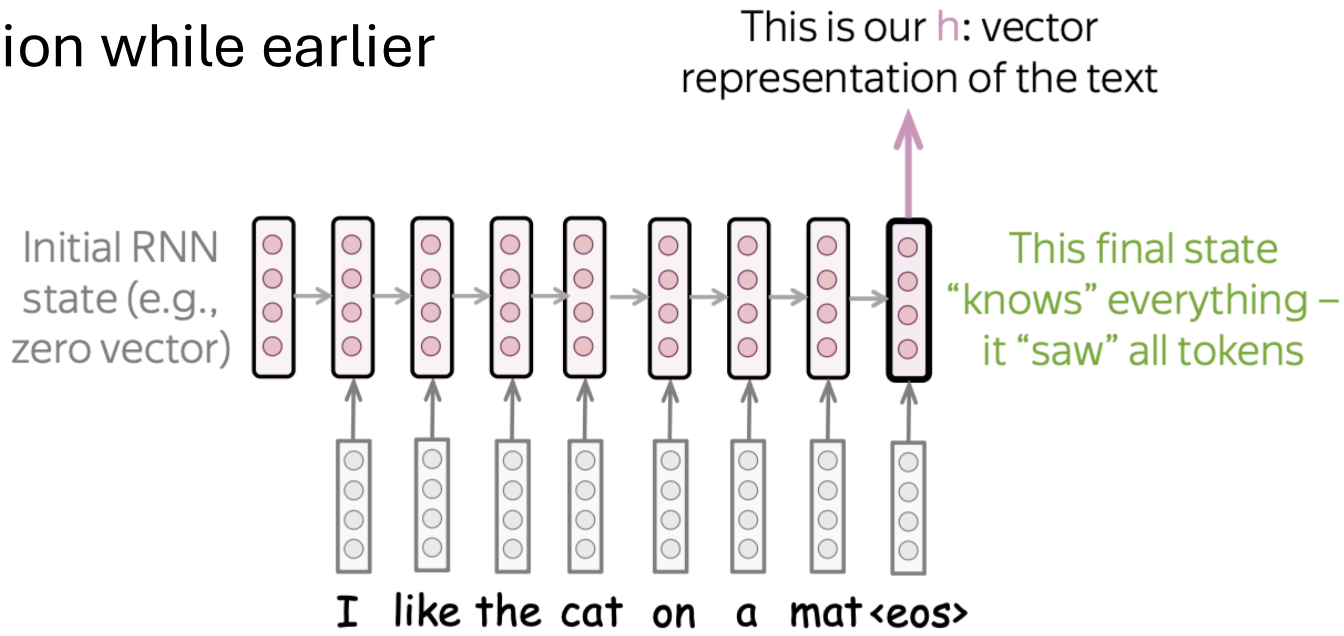
The Problems with Recurrence

- Vanishing and Exploding Gradients
  - Lots of matrix multiplications either blow up or diminish the activity going through the network



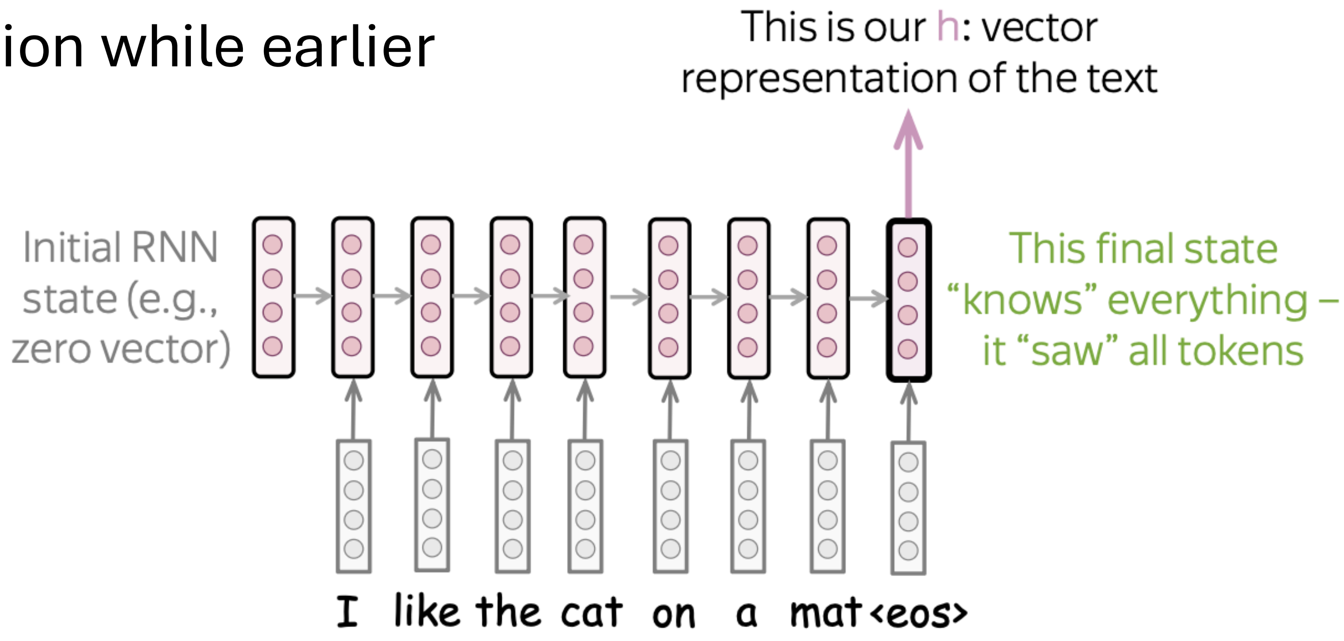
The Problems with Recurrence

- Vanishing and Exploding Gradients
  - Lots of matrix multiplications either blow up or diminish the activity going through the network
- Credit assignment
  - While it is possible for the network to be sensitive to when a token is seen it is very subtle and position information is often lost in the final vector
  - There is also a recency bias – recent information dominates the combined vector representation while earlier information is forgotten



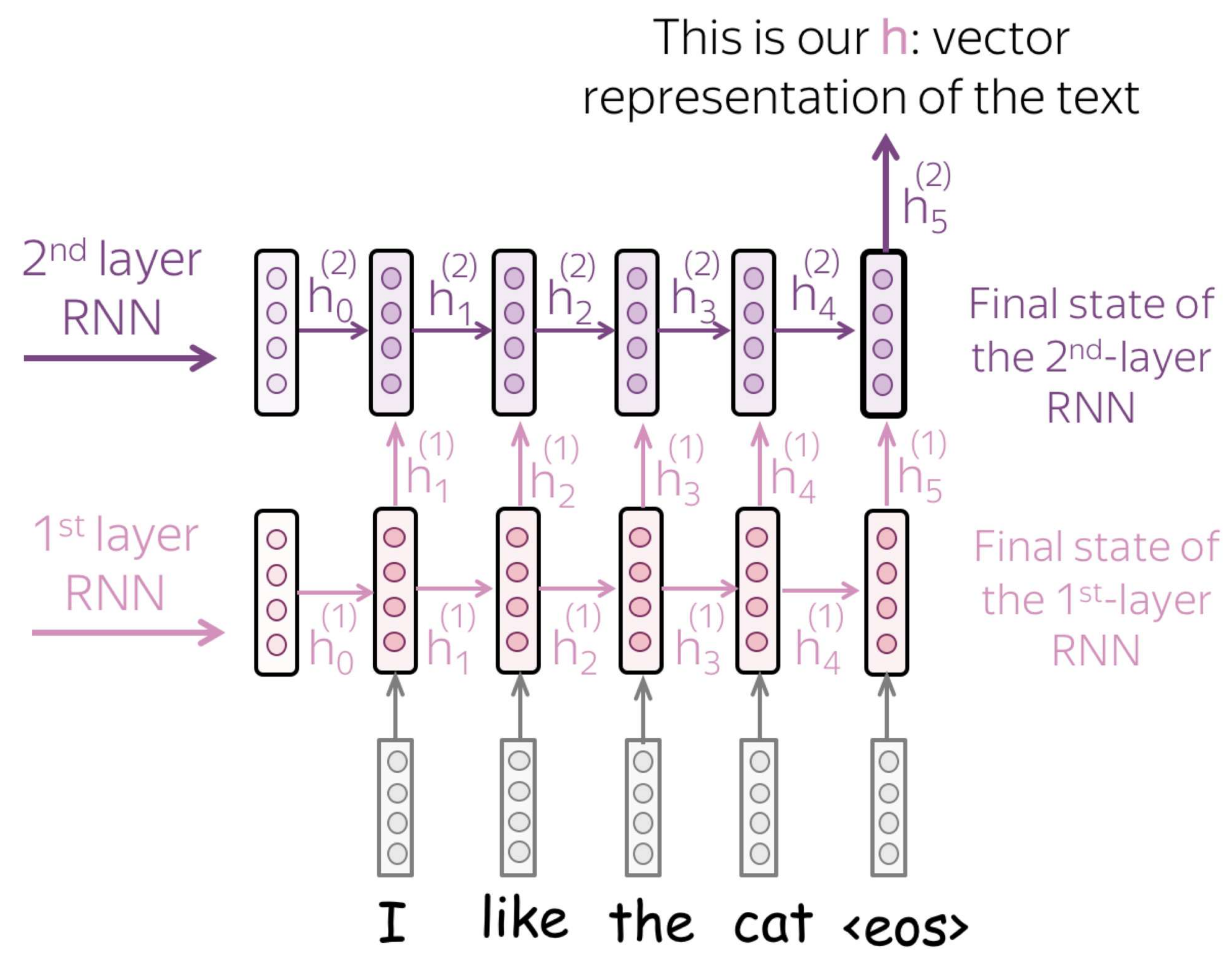
The Problems with Recurrence

- Vanishing and Exploding Gradients
  - Lots of matrix multiplications either blow up or diminish the activity going through the network
- Credit assignment
  - While it is possible for the network to be sensitive to when a token is seen it is very subtle and position information is often lost in the final vector
  - There is also a recency bias – recent information dominates the combined vector representation while earlier information is forgotten
- Fixed vector size:
  - The size of the combined vector does not adjust to the length of the sentence



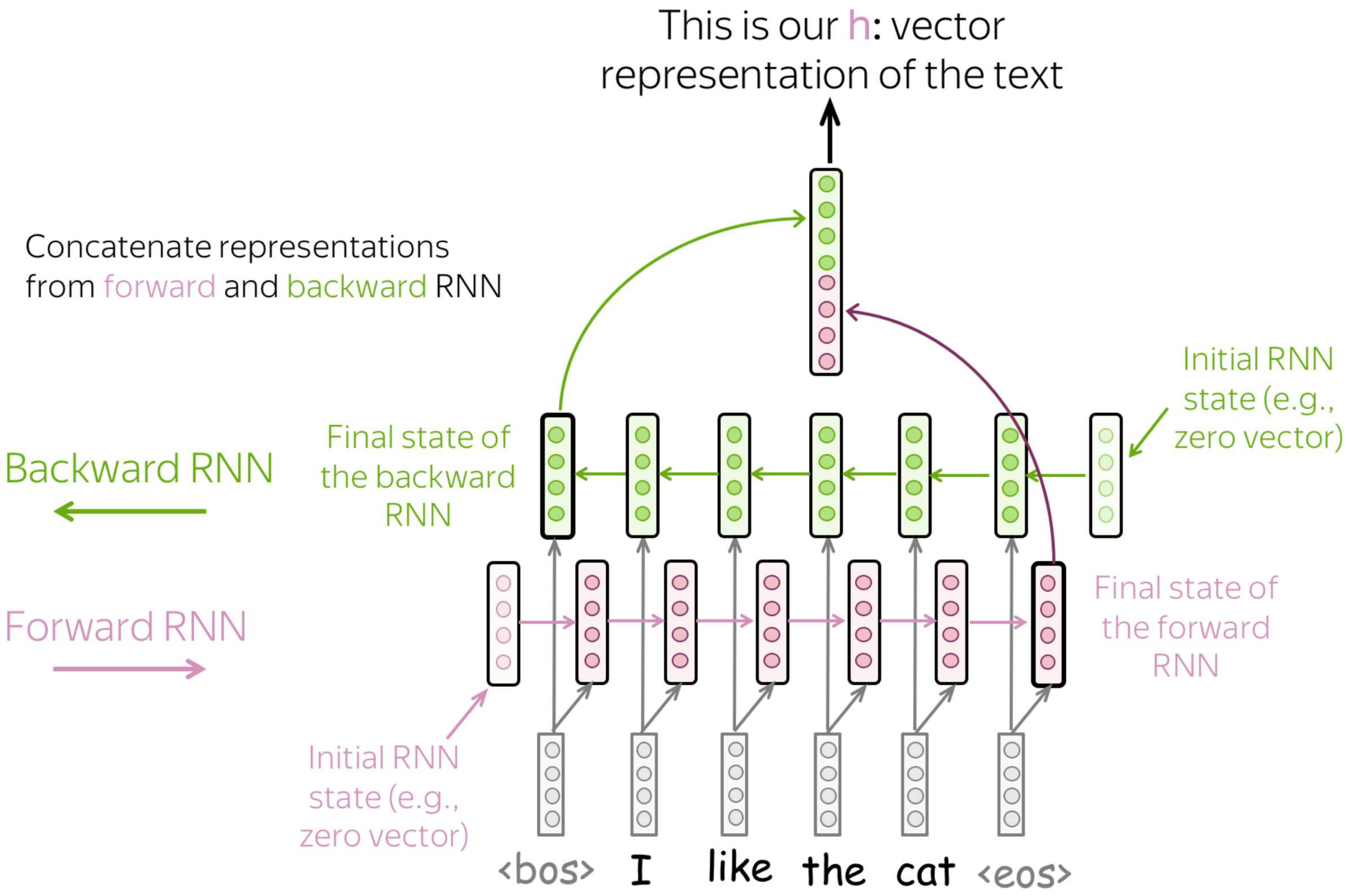
Recurrent Models for Text Classification

- **Multiple layers**: feed the states from one RNN to the next



Recurrent Models for Text Classification

- **bidirectional**: use final states from forward and backward RNNs



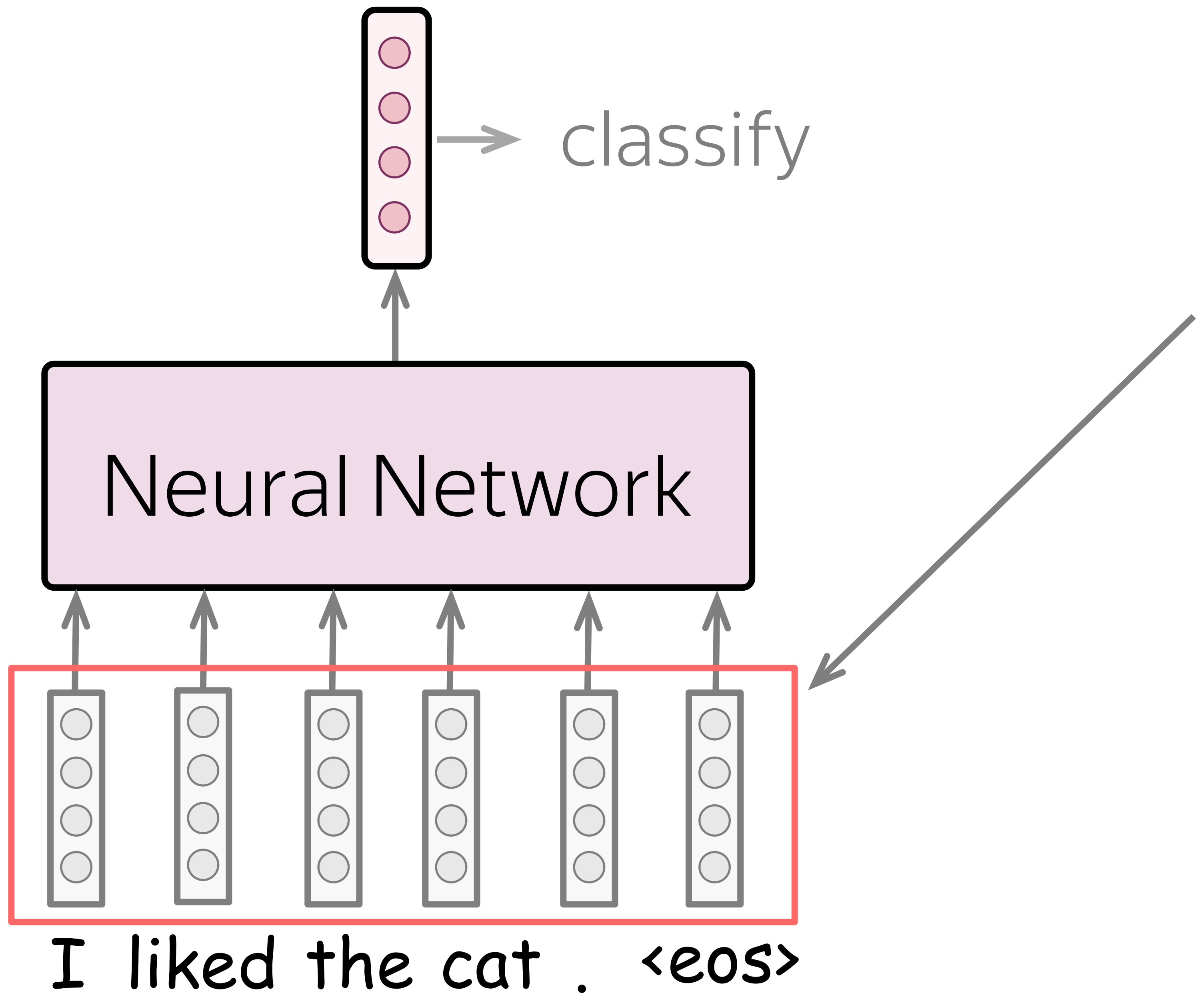
What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
  - High-Level View
  - Training: Cross-Entropy
  - Models: (Weighted) BOW
  - Models: Convolutional
  - Models: Recurrent
- Multi-Label Classification
- Practical Tips
- Analysis and Interpretability

What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
- Multi-Label Classification
- Practical Tips
    - Word Embeddings
    - Data Augmentation
- Analysis and Interpretability

Word Embeddings: How to Deal with them?

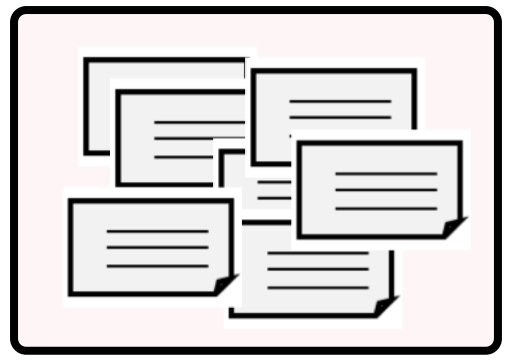


Input word embeddings:

- Train from scratch
- Take pretrained (Word2Vec, GloVe)
- Initialize with pretrained, then fine-tune

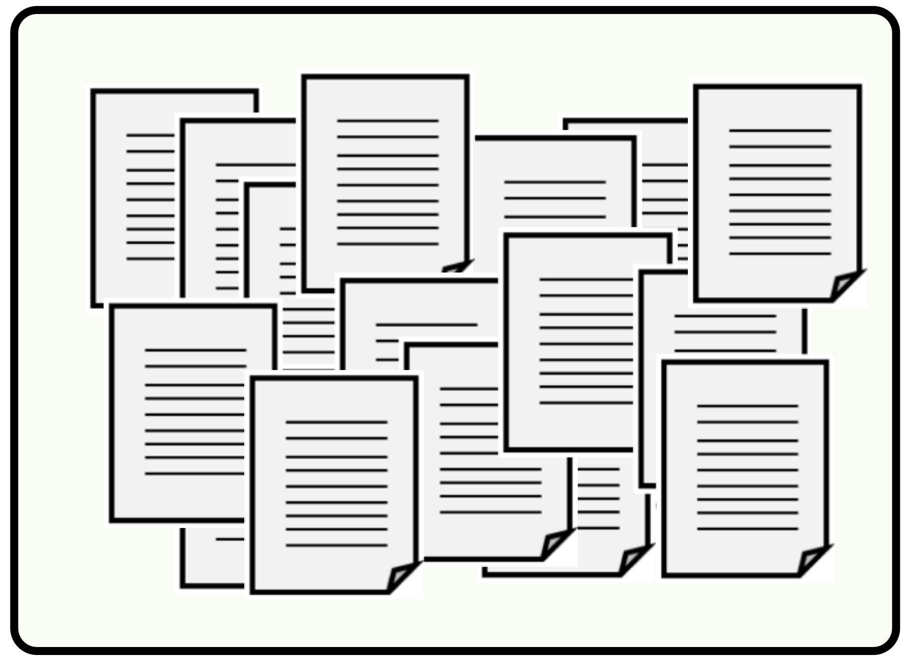
Which data do we have?

Training data for text classification (labeled)



- Not huge, or not diverse, or both
- Domain: task-specific

Training data for word embeddings (unlabeled)

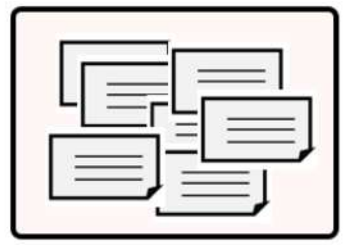


- Huge diverse corpus (e.g., Wikipedia)
- Domain: general

Word Embeddings: How to Deal with them?

- Train from scratch

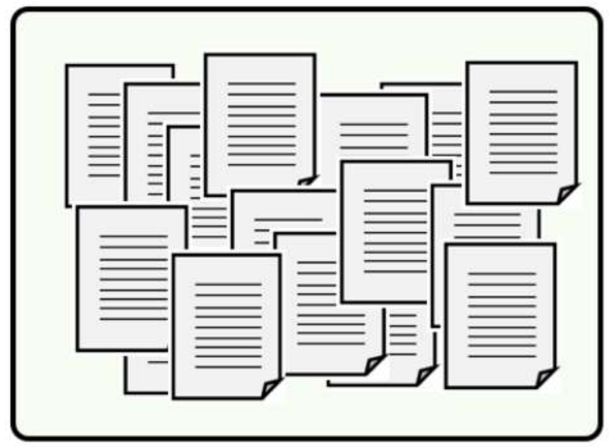
What they will know:



May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

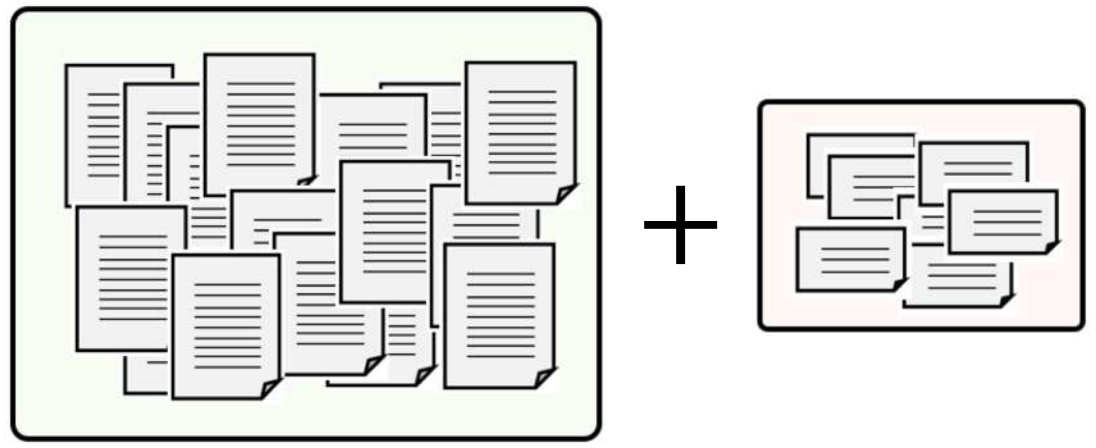
What they will know:



Know relationships between words, but are **not** specific to the task

- **Initialize with pretrained, then fine-tune**

What they will know:

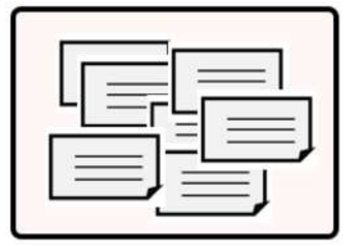


Know relationships between words and adapted for the task

Word Embeddings: How to Deal with them?

- Train from scratch

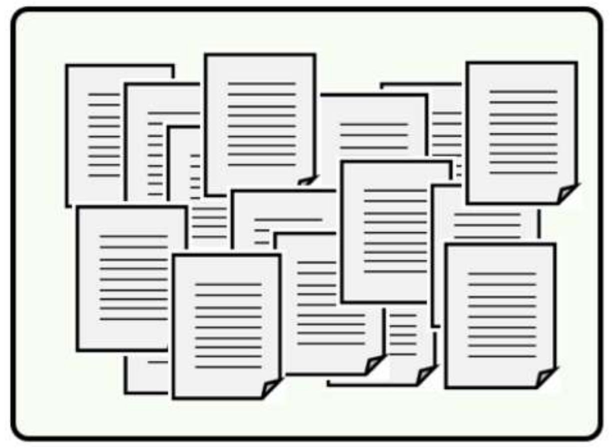
What they will know:



May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

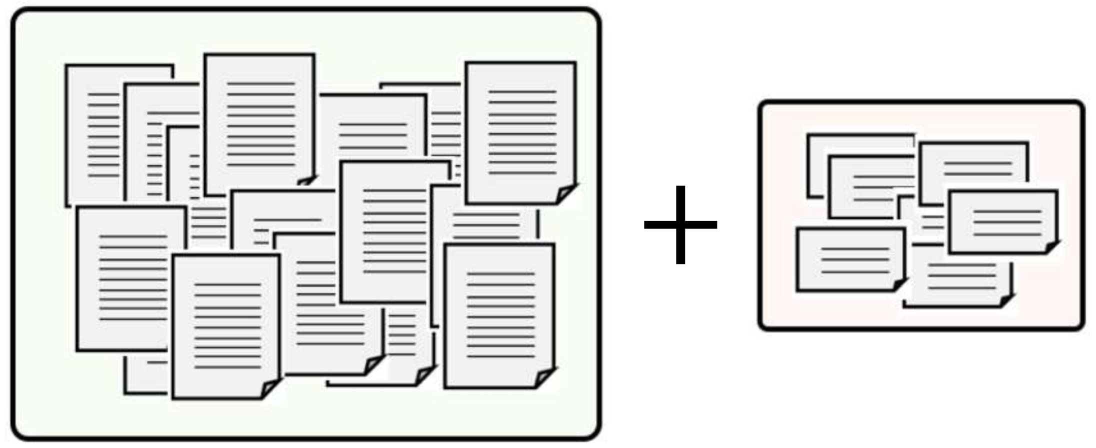
What they will know:



Know relationships between words, but are **not** specific to the task

- **Initialize with pretrained, then fine-tune**

What they will know:



Know relationships between words and adapted for the task

"**Transfer**" knowledge from a huge unlabeled corpus to your task-specific model

We'll learn more about this later in the course!

What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
- Multi-Label Classification
- Practical Tips
    - Word Embeddings
    - Data Augmentation
- Analysis and Interpretability

What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
- Multi-Label Classification
- Practical Tips
    - Word Embeddings
    - Data Augmentation
- Analysis and Interpretability

Data Augmentation: Get More Data for Free

Data augmentation alters your dataset in different ways to get alternative versions of the same training example.

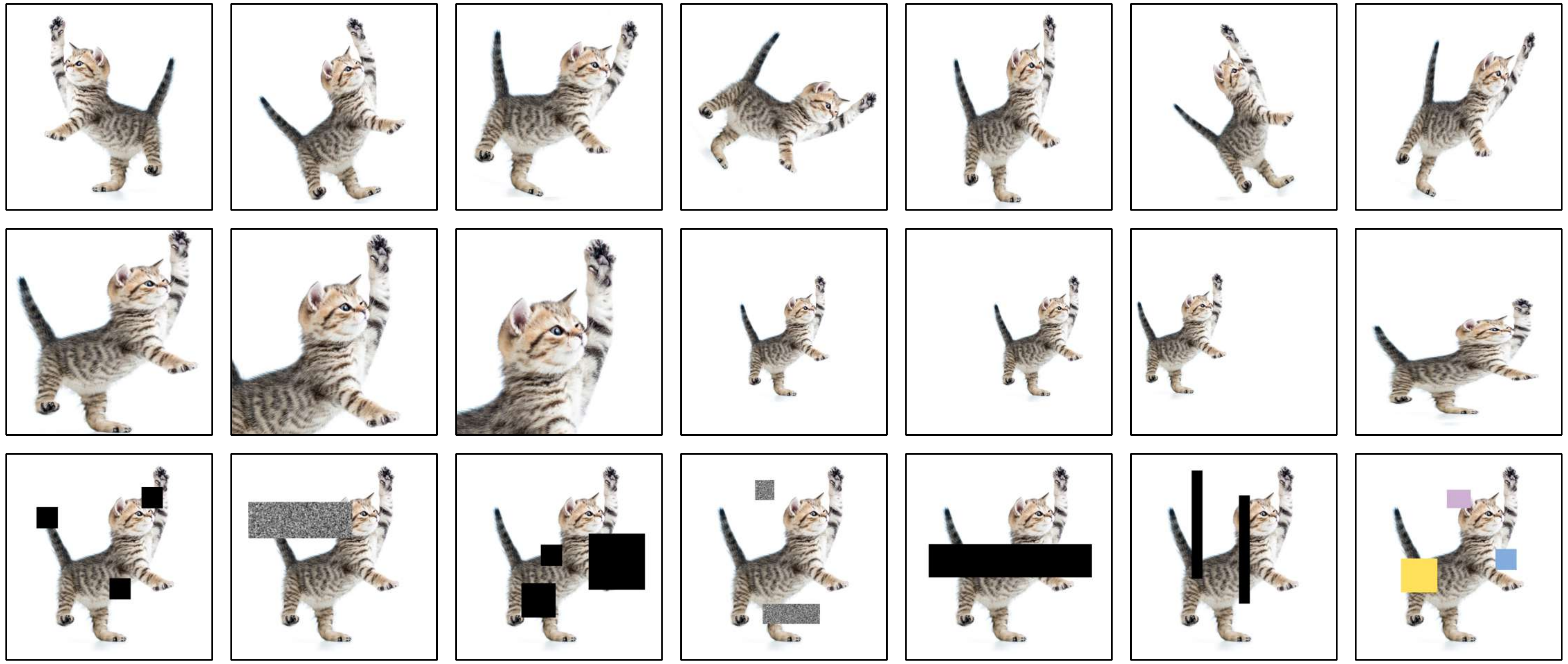
It can increase:

- the amount of data
- diversity of data

Data Augmentation for Images

- Flipping an image
- Geometrical transformations (rotation, stretching, zoom in/out)
- Covering some parts with patches

original image



Data Augmentation for Text

- **word dropout** - the most simple and popular

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

pick several words randomly

replace with UNK

The movie UNK cats was absolutely UNK, and the UNK were cute.

replace with random words

The movie mejorate cats was absolutely fellows, and the mak were cute.

Data Augmentation for Text

- **word dropout** - the most simple and popular

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

pick several words randomly

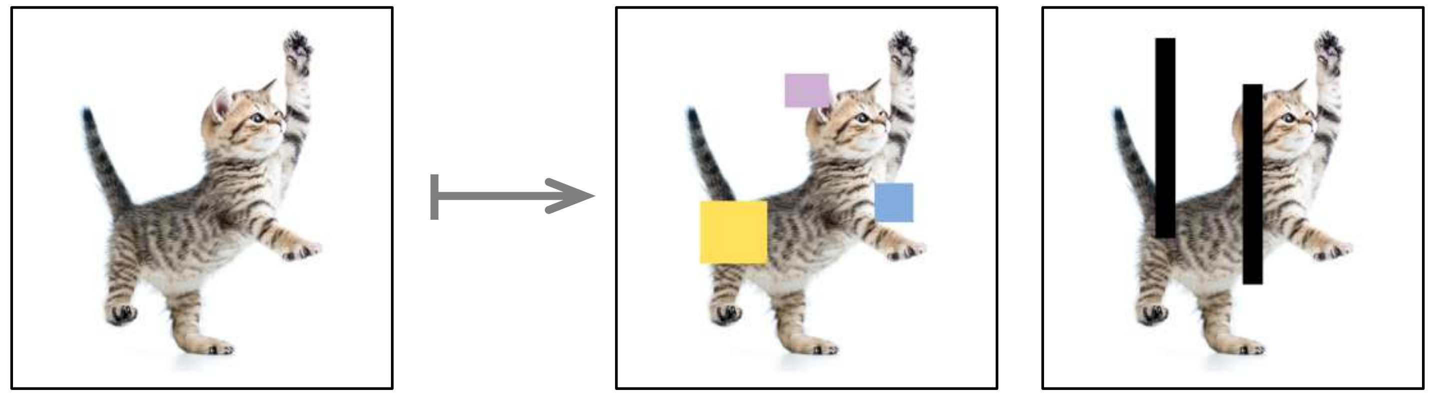
replace with UNK

The movie UNK cats was absolutely UNK, and the UNK were cute.

replace with random words

The movie mejorate cats was absolutely fellows, and the mak were cute.

For images, this is similar to covering some parts: these parts are “dropped” from an image



Data Augmentation for Text

- use external resource (e.g., thesaurus) - a bit more complicated

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

pick words where you have synonyms and use thesaurus

The film about cats was certainly great, and the cats were nice.

The video about cats was completely great, and the cats were charming.

Data Augmentation for Text

- **use separate models** for paraphrasing - if you really care

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

En-Ru ↓ Yandex Translate

Фильм о кошках был замечательный,
и кошки были милые.

Ru-En ↓ Yandex Translate

The cat movie was just great,
and the cats were cute.

En-Zh ↓ Google Translate

關於貓的電影絕對很棒,
而且貓很可愛。

Zh-En ↓ Google Translate

Movies about cats are absolutely
great, and cats are cute.

En-Ja ↓ Google Translate

猫の映画は本当に素晴らしく、
猫はかわいい。

Ja-En ↓ Yandex Translate

The Cat movie is really nice and
the cat is cute.

Data Augmentation for Text

- **use separate models** for paraphrasing - if you really care

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

En-Ru ↓ Yandex Translate

Фильм о кошках был замечательный,
и кошки были милые.

Ru-En ↓ Yandex Translate

The cat movie was just great,
and the cats were cute.

En-Zh ↓ Google Translate

關於貓的電影絕對很棒,
而且貓很可愛。

Zh-En ↓ Google Translate

Movies about cats are absolutely
great, and cats are cute.

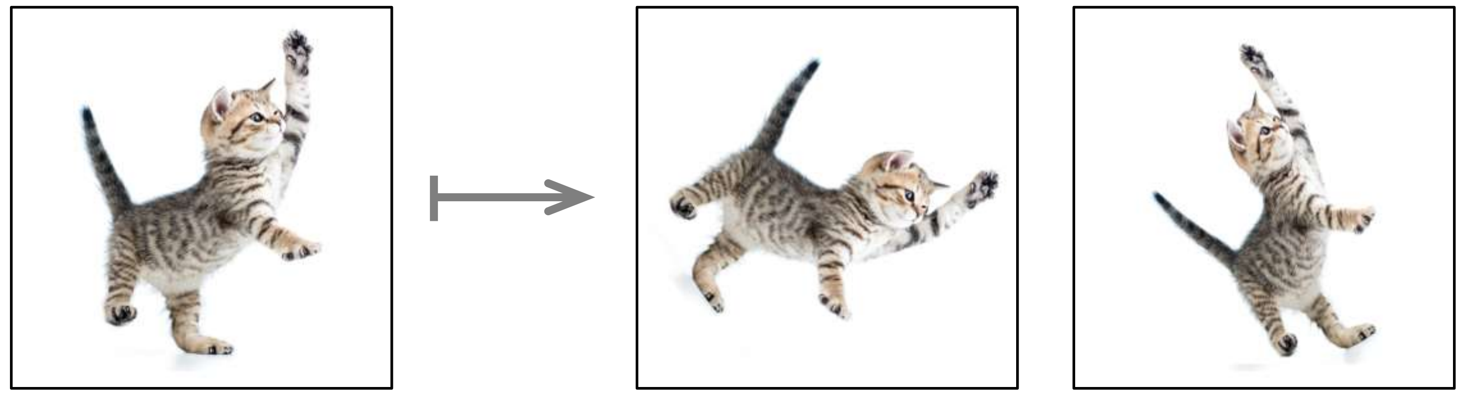
En-Ja ↓ Google Translate

猫の映画は本当に素晴らしく、
猫はかわいい。

Ja-En ↓ Yandex Translate

The Cat movie is really nice and
the cat is cute.

For images, this is similar to geometric transformations: we change text, but want to preserve all meaning.



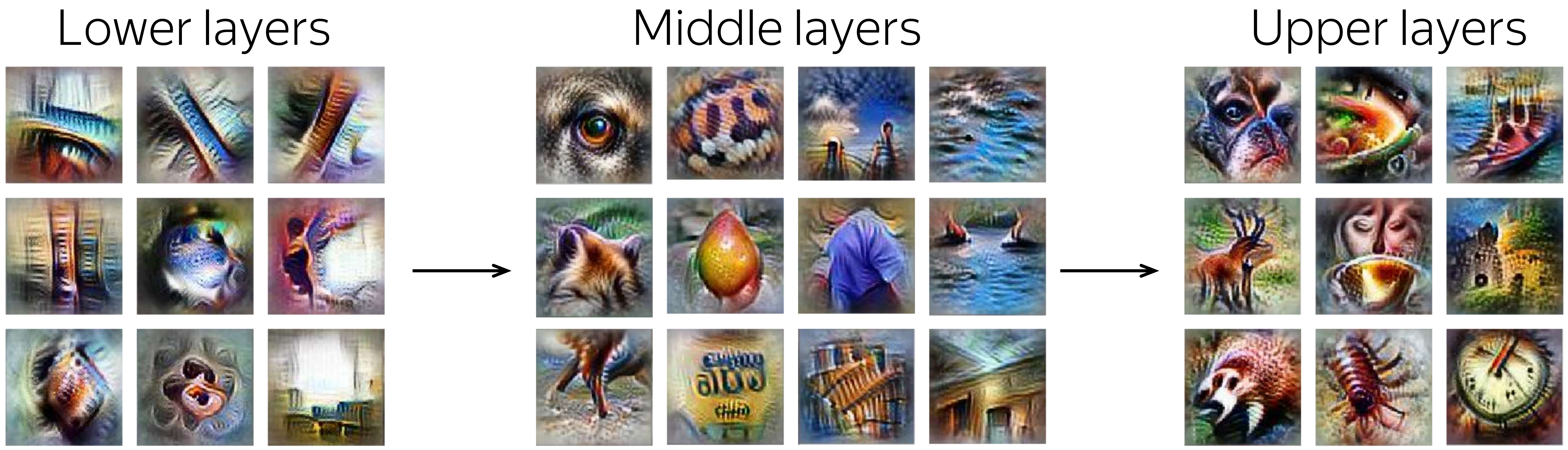
What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
- Multi-Label Classification
- Practical Tips
- Analysis and Interpretability

What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
- Multi-Label Classification
- Practical Tips
- Analysis and Interpretability

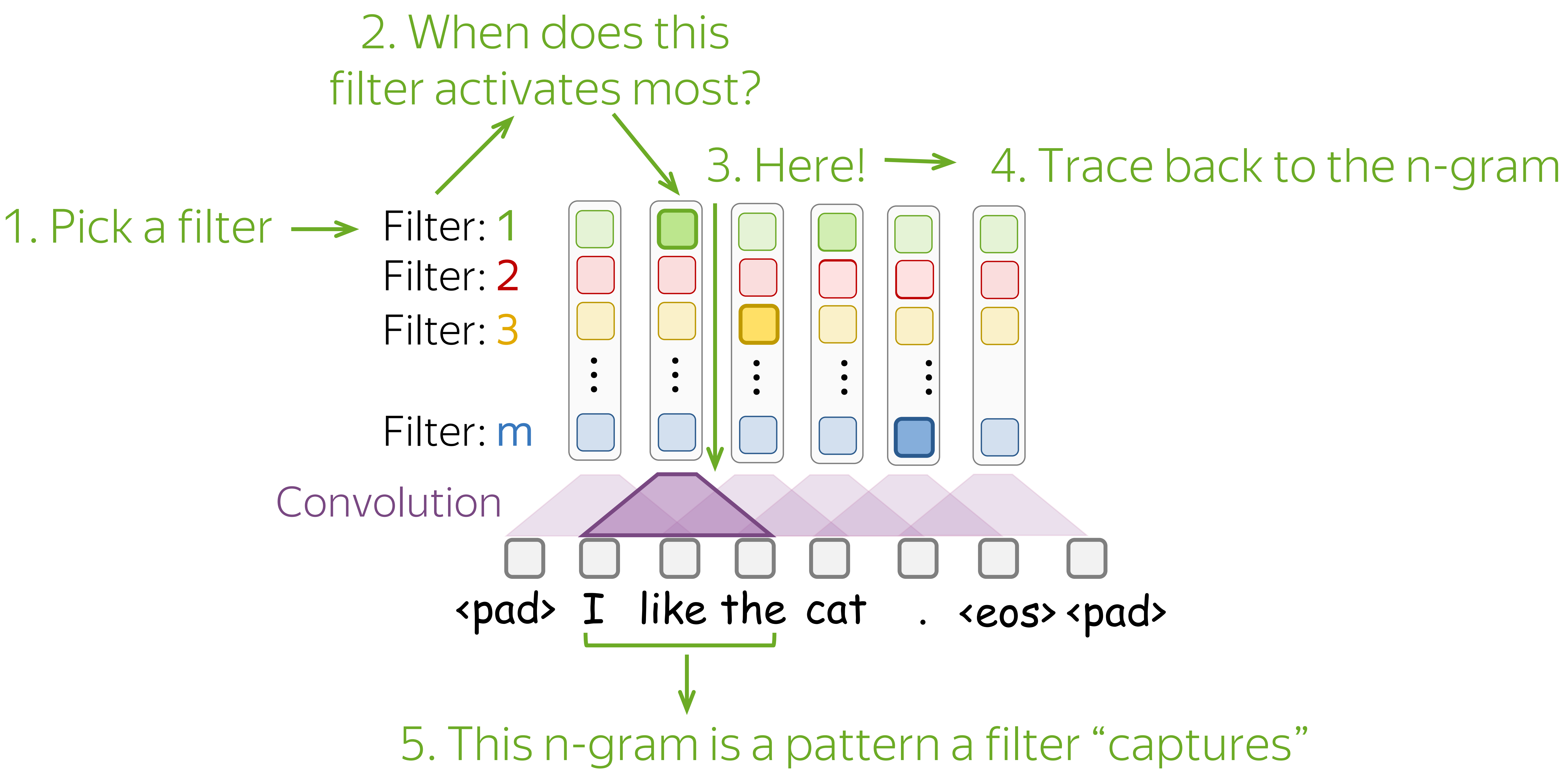
Analyzing Convolutional Filters



Examples of patterns captured by convolution filters for images.
The examples are from Activation Atlas from distill.pub.

Analyzing Convolutional Filters

- Find which patterns activate neurons



Analyzing Convolutional Filters

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

The example is from the paper Understanding Convolutional Neural Networks for Text Classification.

What is going to happen:

- Examples of classification tasks
- General View: Features + Classifier
- Models: Generative vs Discriminative
- Classical Methods
- Neural Methods
- Multi-Label Classification
- Practical Tips
- Analysis and Interpretability

Moving Beyond Words

- Splitting text by whitespace works well in many cases.
- But this means that models have a predefined vocabulary.
- Any words not seen in the training set will be replaced by the semantically meaningless “UNK” token.
- This means we can only process a fixed number of words.

Moving Beyond Words

- What is a good strategy when you encounter a new word though?
- Consider whether you have seen pieces of the word before!
- Remember: morphemes are the smallest units of meaning in language.
- Solution: represent rare or unknown words by their subwords

Moving Beyond Words

Tokenization

Word-level

- fixed vocabulary
- can process only a fixed number of words

Subword-level

- open vocabulary
- rare and unknown tokens are encoded as sequences of subword units

Moving Beyond Words

- This is exactly what the Byte Pair Encoding tokenization strategy does (BPE is an older algorithm, we use an adaption of it).

- Frequent words are found and kept intact while rare words are split.

- See the paper Neural Machine Translation of Rare Words with Subword Units by Rico Sennrich, Barry Haddow and Alexandra Birch

Byte Pair Encoding

- Main Idea: Search through the current list of symbols and find the most common pairings. Then merge the pairing into a new symbol.

- There are two parts:
  - Training: find which pairs of symbols to merge
  - Inference: apply the learned merges to a new piece of text

- Let’s consider each part separately.

Byte Pair Encoding: Training

- Input: a corpus of text to train the tokenizer – not necessarily the same data we use to train our model.
- Output: a merge table and vocabulary of tokens:
  - Merge table: says which characters to combine
  - Vocabulary: which tokens we can identify – tokens are a combination of words, subwords and characters depending on the data (it's adaptive!)

- To begin we split all of the text in our corpus into their characters

Byte Pair Encoding: Training

- With the characters split we **iteratively** apply the steps:
  - count pairs of symbols: how many times each pair occurs together in the training data;
  - find the most frequent pair of symbols;
  - merge this pair - add a merge to the merge table, and the new token to the vocabulary

- Note: we only count character pairs within words (so we can begin by counting the words to make the process more efficient)

Byte Pair Encoding: Training - Example

- Assume we have a corpus with the following words and their number of occurances:
  - Cat x4
  - Mat x5
  - Mats x2
  - Mate x3
  - Ate x3
  - Eat x2

Byte Pair Encoding: Training - Example

Words (split chars)	Count
c a t	4
m a t	5
m a t s	2
m a t e	3
a t e	3
e a t	2

Merge Table
** Start Empty**

- From these counts “a” and “t” appear together the most – 19 times
- “m” and “a” appear together 10 times
- Add “at” to our merge table!

Byte Pair Encoding: Training - Example

Words (split chars)	Count
c β	4
m β	5
m β s	2
m β e	3
β e	3
e β	2

Merge Table
at -> β

- “m” and “β” now appear together the most with 10 occurrences.
- Second is “β” and “e” with 6.
- Add “mβ” to our merge table

Byte Pair Encoding: Training - Example

Words (split chars)	Count
c β	4
α	5
α s	2
α e	3
β e	3
e β	2

Merge Table
at -> β
mβ -> α

- Finally, we can merge “c” and “β” since this is the largest combination.

Byte Pair Encoding: Training - Example

Words (split chars)	Count
μ	4
α	5
α s	2
α e	3
β e	3
e β	2

Merge Table
at -> β
mβ -> α
cβ -> μ

- Finally, we can merge “c” and “β” since this is the largest combination.

Byte Pair Encoding: Training - Example

Start Words (split chars)	End Words (split chars)
c a t	μ
m a t	α
m a t s	α s
m a t e	α e
a t e	β e
e a t	e β

- Note how the words “cat” and “mat” are common and kept together (represented with one symbol).
- Things like plurals are kept separate longer (like “mats”)
- Implementation detail: we usually add “@@” to tokens which are whole words. So “mat@@” and “mat s” can be treated differently.

Byte Pair Encoding: Inference

- After learning BPE rules, you have a merge table – you can use it to segment new text.
- To start, split words into a sequence of characters. Then iteratively over the merge table and apply the merges wherever they occur in the text.
- Note that the merge table is ordered - the merges that are higher in the table were more frequent in the data and given priority.
- BPE is deterministic.