CSE8803/CX4803

Machine Learning in Computational Biology

Lecture 6: Deep learning for sequence data

Yunan Luo

CSE8803/CX4803 in-class survey

Please complete this 1-min survey

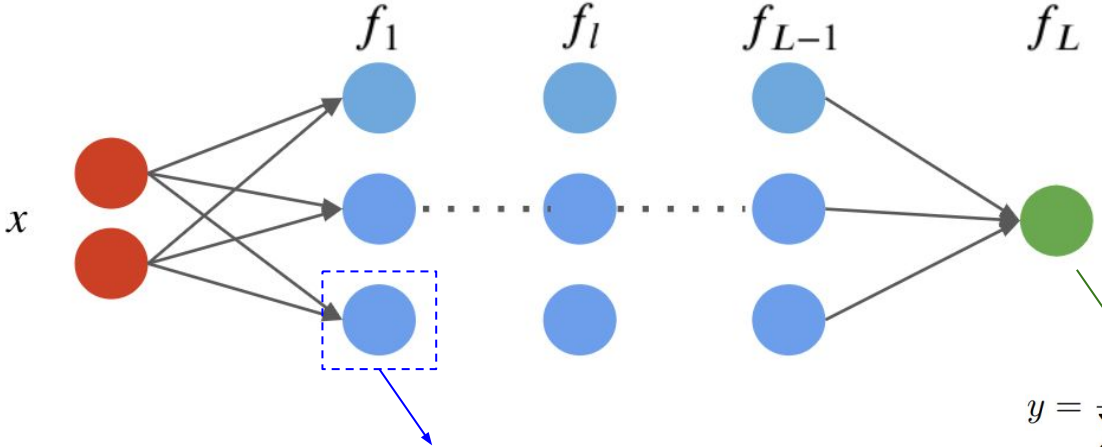
https://tinyurl.com/cse8803DLsurvey

Today’s plan

- A short primer on deep learning
- Biological sequences
- Deep learning for sequence data in biomedicine
  - Supervised learning
    - CNN, RNN, LSTM, Transformer
  - Unsupervised learning
    - Protein language modeling

Recap: neural network

A (fully connected) neural network is a computational model that consists of a composition of multiple neural network layers



$$y = \frac{\exp(\theta_y^T h)}{\sum_{c \in class} \exp(\theta_c^T h)}$$

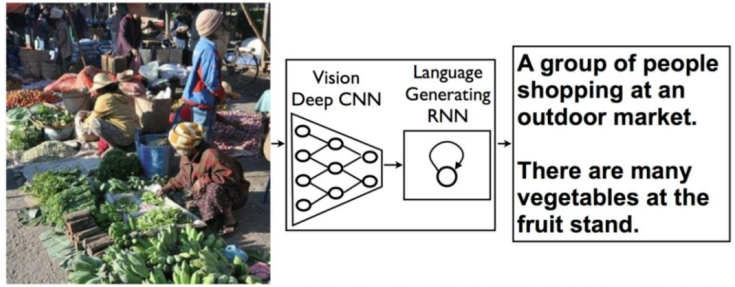
$$f(x) = \sigma(\theta^\top x) = \frac{1}{1 + \exp(-\theta^\top x)}$$

https://github.com/kuleshov/cornell-cs5785-2021-applied-ml/blob/main/notebooks/lecture19-deep-learning.ipynb

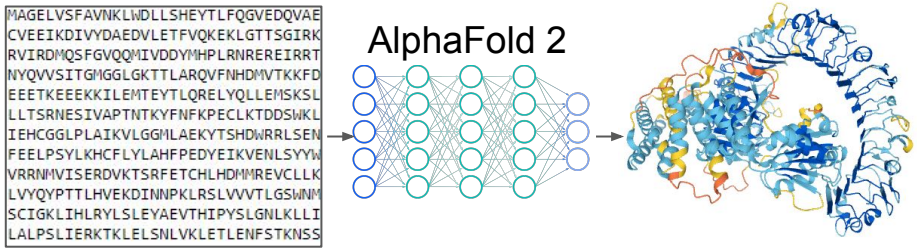
Deep learning

A class of machine learning methods that emphasizes:

- *Deep* neural networks
- Large unstructured datasets, especially images, text, and audio
- Modern computational resources, like GPUs

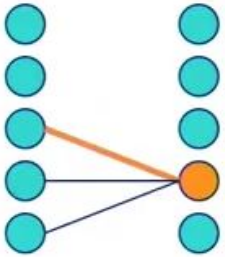


Ref: Learning CNN-LSTM Architectures for Image Caption Generation: https://cs224d.stanford.edu/reports/msoh.pdf



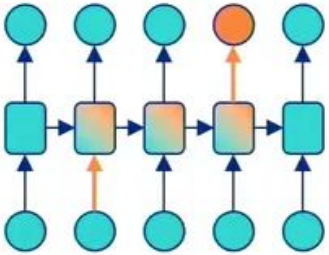
https://github.com/kuleshov/cornell-cs5785-2021-applied-ml/blob/main/notebooks/lecture19-deep-learning.ipynb

Deep learning models and implementations



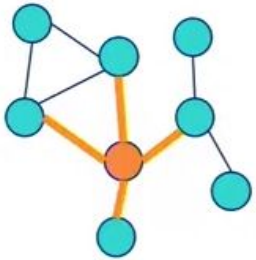
Convolutional Networks
(e.g. computer vision)

- data in regular grid
- information flow to local neighbours
- AlphaFold 1



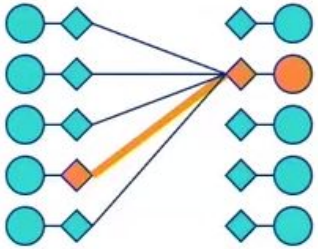
Recurrent Networks
(e.g. language)

- data in ordered sequence
- information flow sequentially



Graph Networks (e.g. recommender systems or molecules)

- data in fixed graph structure
- information flow along fixed edges



Attention Module (e.g. language)

- data in unordered set
- information flow dynamically controlled by the network (via keys and queries)

PyTorch TensorFlow JAX Keras Google colab

Image from John Jumper's talk "Highly accurate protein structure prediction with AlphaFold"

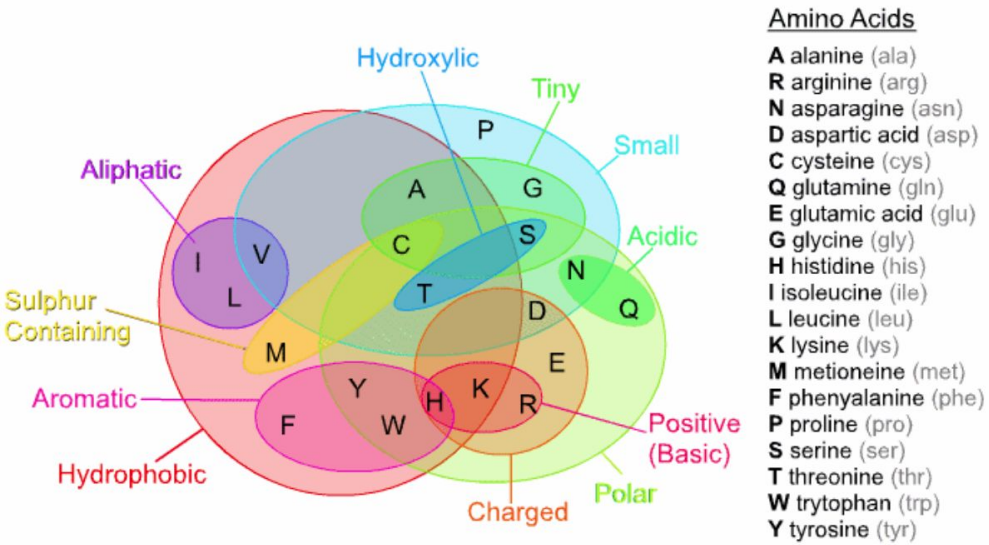
Biological sequences

Slides credit: Jian Peng, CS466 & CS598, UIUC

Central dogma

Protein and amino acid

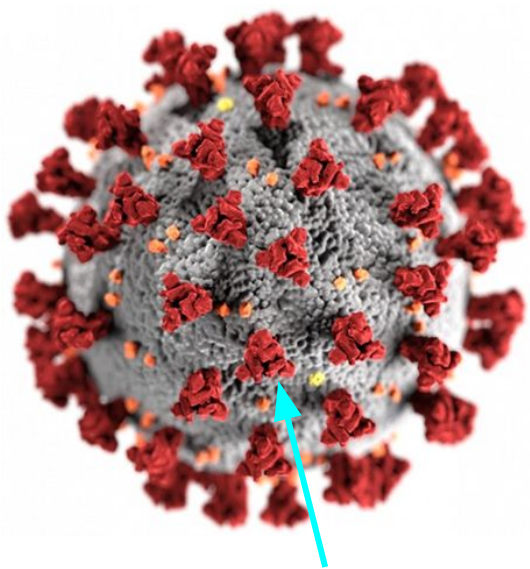
In CS language: protein sequence is a string ***s*** of length ***L*** over the alphabet ***Σ*** of 20 characters



Alphabet ***Σ***: amino acids

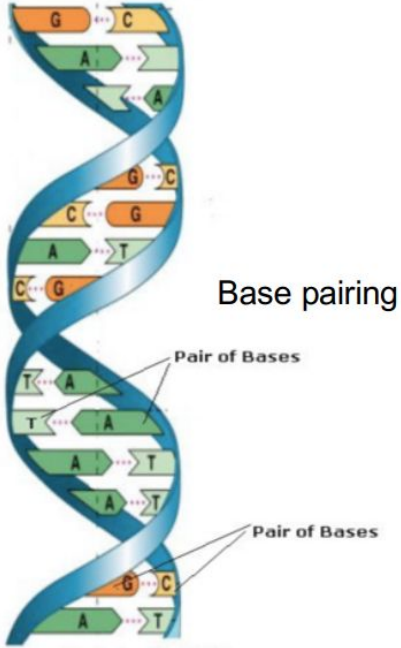
Protein sequence

```
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSSVLHSTQDLFLPFFS
NVTWFHAIHVSGTNGTKRFDNPVLPFNDGVYFASTEKSNIIRGWIFGTTLDSKTQSLLIV
NNATNVVIKVCEFQFCNDPFLGVYYHKNNKSWMESEFRVYSSANNCTFEYVSQPFLMDLE
GKQGNFKNLREFVFKNIDGYFKIYSKHTPINLVRDLPQGFSALEPLVDLPIGINITRFQT
LLALHRSYLTPGDSSSGWTAGAAAYYVGYLQPRTFLLKYNENGTITDAVDCALDPLSETK
CTLKSFTVEKGIYQTSNFRVQPTESIVRFPNITNLCPFGEVFNATRFASVYAWNRKRISN
CVADYSVLYNSASFSTFKCYGVSPTKLNDLCFTNVYADSFVIRGDEVRQIAPGQTGKIAD
YNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLYRLFRKSNLKPFERDISTEIYQAGSTPC
NGVEGFNCYFPLQSYGFQPTNGVGYQPYRVVVLSFELLHAPATVCGPKKSTNLVKNKCVN
FNFNGLTGTGVLTESNKKFLPFQQFGRDIADTTDAVRDPQTLEILDITPCSFGGVSVITP
GTNTSNQVAVLYQDVNCTEVPVAIHADQLTPTWRVYSTGSNVFQTRAGCLIGAEHVNNSY
ECDIPIGAGICASYQTQTNSPRRARSVASQSIIAYTMSLGAENSVAYSNNSIAIPTNFTI
SVTTEILPVSMTKTSVDCTMYICGDSTECSNLLLQYGSFCTQLNRALTGIAVEQDKNTQE
VFAQVKQIYKTPPIKDFGGFNFSQILPDPSKPSKRSFIEDLLFNKVTLADAGFIKQYGDC
LGDIAARDLICAQKFNGLTVLPPLLTDEMIAQYTSALLAGTITSGWTFGAGAALQIPFAM
QMAYRFNGIGVTQNVLYENQKLIANQFNSAIGKIQDSLSSTASALGKLQDVVNQNAQALN
TLVKQLSSNFGAISSVLNDILSRLDKVEAEVQIDRLITGRLQSLQTYVTQQLIRAAEIRA
SANLAATKMSECVLGQSKRVDFCGKGYHLMSFPQSAPHGVVFLHVTYVPAQEKNFTTAPA
ICHDGKAHFPREGVFVSNGTHWFVTQRNFYEPQIITTDNTFVSGNCDVVIGIVNNTVYDP
LQPELDSFKEELDKYFKNHTSPDVDLGDISGINASVVNIQKEIDRLNEVAKNLNESLIDL
QELGKYEQYIKWPWYIWLGFIAGLIAIVMVTIMLCCMTSCCSCLKGCCSCGSCCKFDEDD
```



SARS-CoV-2 spike protein

DNA



Base Pairing in DNA Double Helix

Base pairing property

=

The DNA Molecule

5’

G -- C
A -- T
T -- A
G -- C
C -- G
G -- C
T -- A
G -- C
T -- A
T -- A
A -- T
A -- T
C -- G
T -- A

3’

DNA sequence

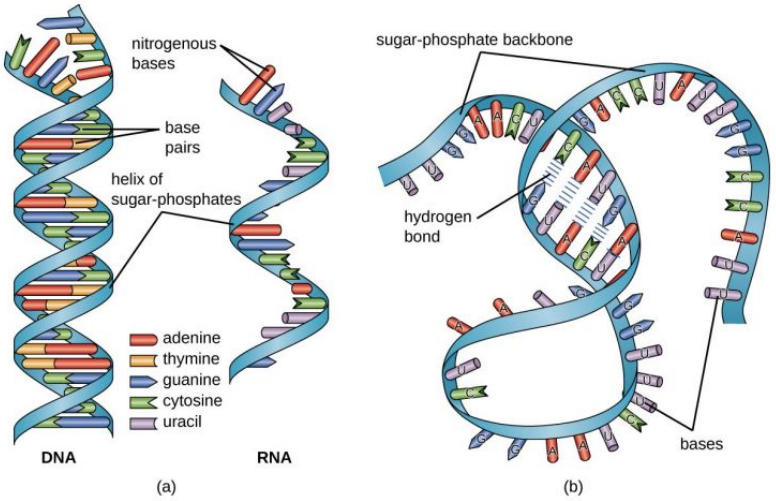
A string ***s*** of length ***L*** over the alphabet ***Σ***={A,C,G,T}

agcactctctcacttctggccagggaacgtggaaggcgcaccgacagggatccggccagggagggcgagtgaaagaaggaaatcagaaaggaaggga
gttaacaaaataataaaaacagcctgagccacggctggagagaccgagacccggcgcaagagagcgcagccttagtaggagaggaacgcgagacgcg
gcagagcgcgttcagcactgacttttgctgctgcttctgctttttttttcttagaaacaagaaggcgccagcggcagcctcacacgcgagcgccacgcgaggctc
ccgaagccaacccgcgaagggaggagggggagggaggaggaggcggcgtgcagggaggagaaaaagcattttcactttttttgctcccactctaagaagtct
cccggggattttgtatatatttttaacttccgtcagggctcccgcttcatatttccttttctttccctctctgttcctgcacccaagttctctctgtgtccccctcgcgggcccc
gcacctcgcgtcccggatcgctctgattccgcgactccttggccgccgctgcgcatggaaagctctgccaagatggagagcggcggcgccggccagcagccc
cagccgcagccccagcagcccttcctgccgcccgcagcctgtttctttgccacggccgcagccgcggcggccgcagccgccgcagcggcagcgcagagcg
cgcagcagcagcagcagcagcagcagcagcagcagcaggcgccgcagctgagaccggcggccgacggccagccctcagggggcggtcacaagtcag
cgcccaagcaagtcaagcgacagcgctcgtcttcgcccgaactgatgcgctgcaaacgccggctcaacttcagcggctttggctacagcctgccgcagcagc
agccggccgccgtggcgcgccgcaacgagcgcgagcgcaaccgcgtcaagttggtcaacctgggctttgccacccttcgggagcacgtccccaacggcgc
ggccaacaagaagatgagtaaggtggagacactgcgctcggcggtcgagtacatccgcgcgctgcagcagctgctggacgagcatgacgcggtgagcgc
cgccttccaggcaggcgtcctgtcgcccaccatctcccccaactactccaacgacttgaactccatggccggctcgccggtctcatcctactcgtcggacgaggg
ctcttacgacccgctcagccccgaggagcaggagcttctcgacttcaccaactggttctgaggggctcggcctggtcaggccctggtgcgaatggactttggaa
gcaggtaggttgcattttggggtgggcagggggtattcttgccttcgtcctccctctgagtgtctgtggaagtggggatgtctccaaggagataaggggatttttattt
aaagaatttgtgaaagttggtcgatttcaagtcctagtttgttagtttcagcactggcctctgaaaatggccttgcccaggtctccaaggagtgaagggtagtagtga
ggtgcagagatactggtgaaccgaatactgggacatgttaaaagagatgtctacctgacagactctttccccagacctccatctccctctaccactagcctacac
gttcaaattaacctctcctgttcttttccttatgttatagggtgatcgcacaacctgcatctttagtgctttcttgtcagtggcgttgggagggggagaaaaggaaaaga
aaaaaaaaagaagaagaagaagaaaagagaagaagaaaaaaacgaaaacagtcaaccaaccccatcgccaactaagcgaggcatgcctgagaga
catggctttcagaaaacgggaagcgctcagaacagtatctttgcactccaatcattcacggagatatgaagagcaactgggacctgagtcaatgcgcaaaatg
cagcttgtgtgcaaaagcagtgggctcctggcagaagggagcagcacacgcgttatagtaactcccatcacctctaacacgcacagctgaaagttcttgctcgg
gtcccttcacctcctcgcccttttcttaaagtgcagttcttagccctctagaaacgagttggtgtctttcgtctcagtagcccccaccccaataagctgtagacattggttt
acagtgaaactatgctattctcagcccttttgaaactctgcttctcctccagggcccgattcccaaaccccatggcttccctcacactgtcttttctaccattttcattatag
aatgcttccaatcttttgtgaattttttattataaaaaatctatttgtatctatcctaaccagttcggggatatattaagatatttttgtacataagagagaaagagagaga
aaaatttatagaagttttgtacaaatggtttaaaatgtgtatatcttgatactttaacatgtaatgctattacctctgcatattttagatgtgtagttcaccttacaactgcaat
tttccctatgtggttttgtaaagaactctcctcataggtgagatcaagaggccaccagttgtacttcagcaccaatgtgtcttactttatagaaatgttgttaatgtattaat
gatgttattaaatactgttcaagaagaacaaagtttatgcagctactgtccaaactcaaagtggcagccagttggttttgataggttgccttttggagatttctattactg
cctttttttttcttactgttttattacaaacttacaaaaatatgtataaccctgttttatacaaactagtttcgtaataaaactttttcctttttttaaaatg

RNA

RNA = ribonucleic acid

- “U” instead of “T”
- Usually single stranded
- Has base-pairing capability
    - Can form simple non-linear structures
- Life may have started with RNA

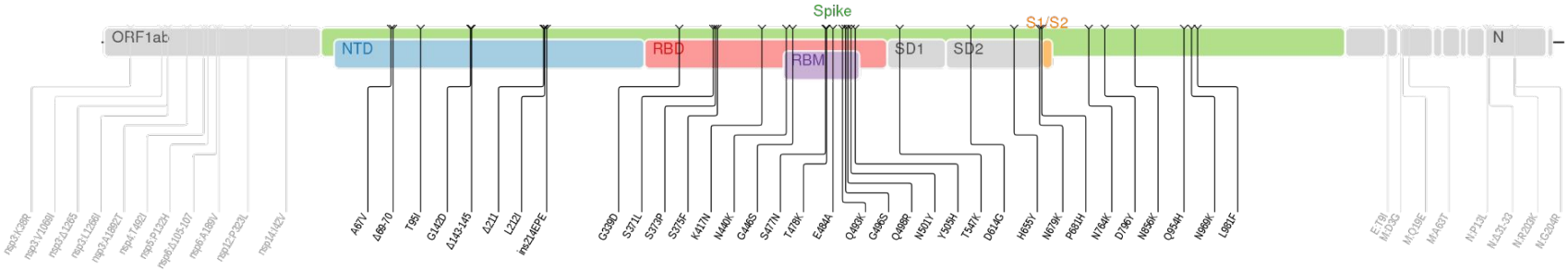


Summary: biological sequences

- DNA = nucleotide sequence
  - Alphabet size = 4 (A,C,G,T)
- RNA (single stranded)
  - Alphabet size = 4 (A,C,G,U)
- Protein sequence
  - Alphabet size = 20

Why study biological sequences

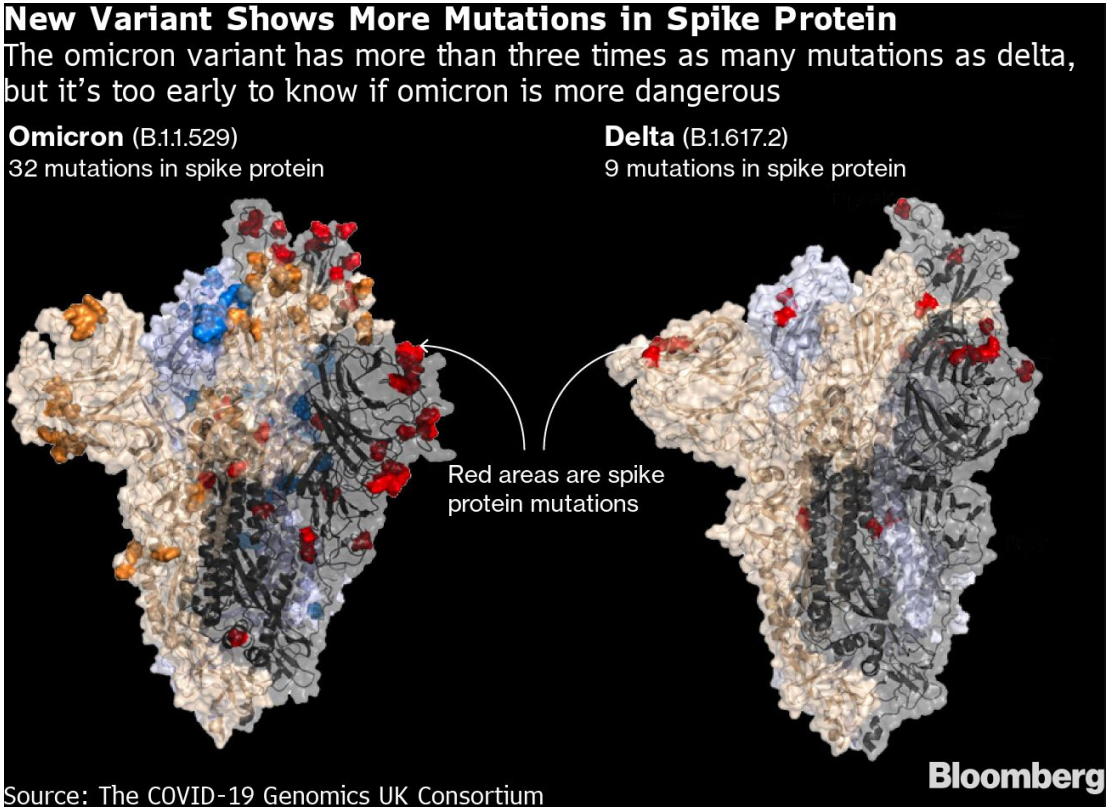
- **DNA** stores the genetic information
- **RNA** is the intermediate molecule for protein synthesis
- **Proteins** carry out the biochemical functions in cell



Example: genomic sequence SARS-CoV-2 Omicron

Image from wiki

Protein sequence-structure-function



Deep learning for sequence data

Problem formulation

Regression/Classification

Example: protein stability prediction

Input	Output
…DNGVDGEWTYDDATKTFTVTE	1.0
…DNGCDGEWTYDDATKTFTVTE	-0.2
…DNGVWGEWTYDDATKTFTVTE	3.9
…DNGVWGEWTYDDATKTFTFTE	5.4
…DNGVMGEWTYDDATKTFTDTE	-0.1

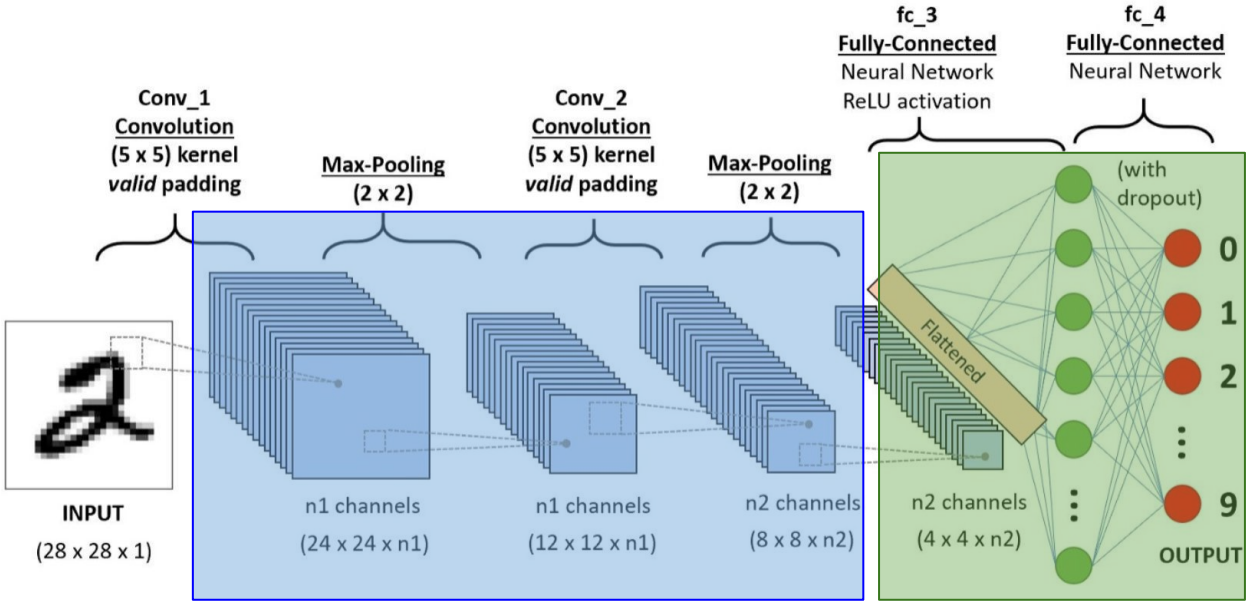
Sequence encoding

- one-hot encoding

	C	G	A	T	A	A	C	C	G	A	T	A	T
A	0	0	1	0	1	1	0	0	0	1	0	1	0
C	1	0	0	0	0	0	1	1	0	0	0	0	0
G	0	1	0	0	0	0	0	0	1	0	0	0	0
T	0	0	0	1	0	0	0	0	0	0	1	0	1

- contextual embedding (language models)
  - Rives, Alexander, et al. "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences." *Proceedings of the National Academy of Sciences* 118.15 (2021).

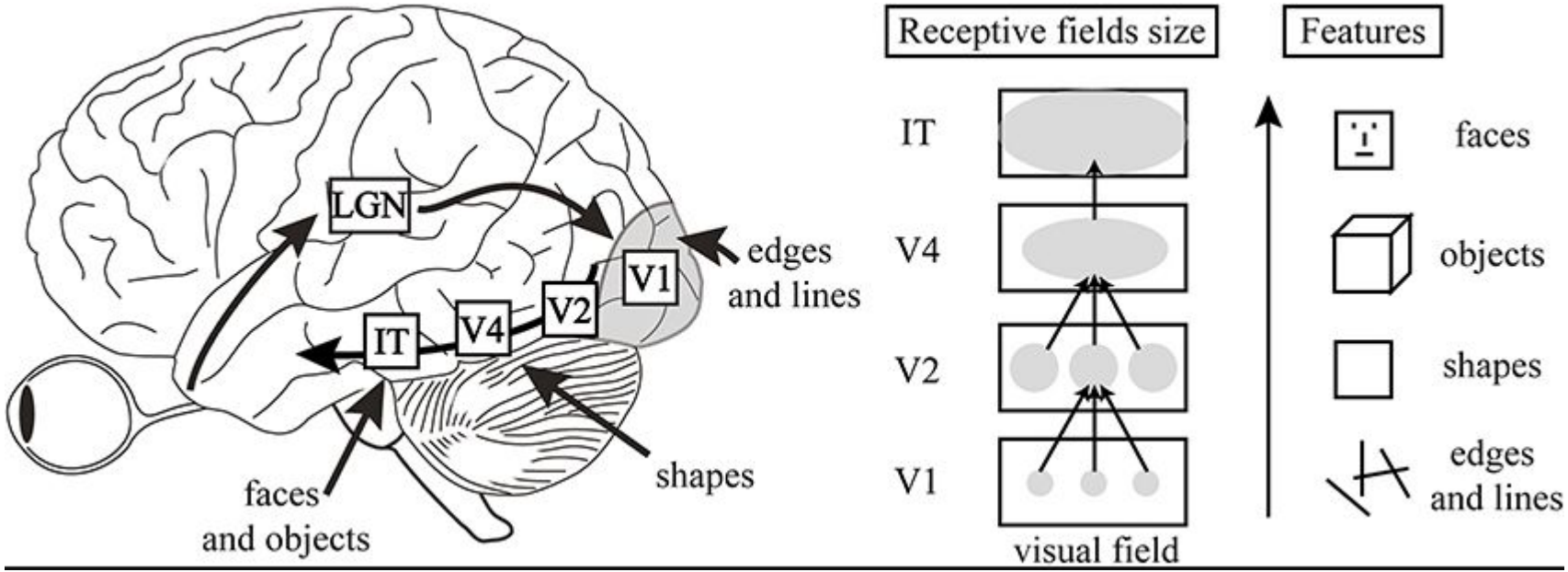
Model #1: Convolutional Neural Network (CNN)



Feature extraction
Hierarchical representation learning

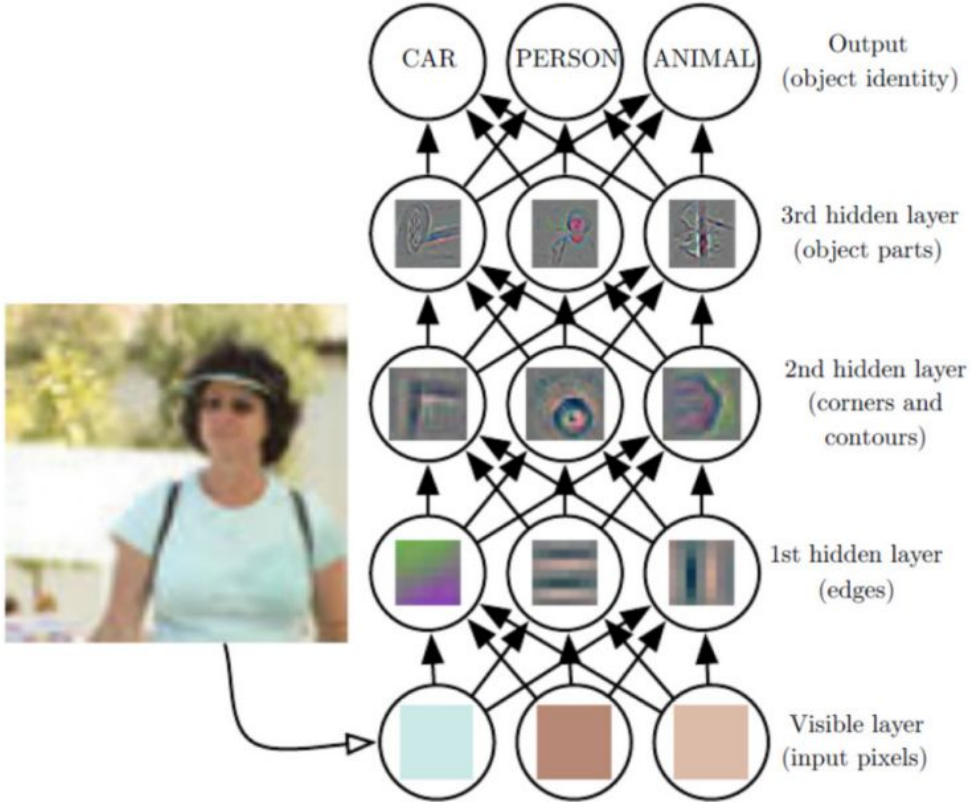
Classification
Fully-connected neural network

Inspired by human vision



https://www.frontiersin.org/articles/10.3389/fncom.2014.00135/full

Hierarchical feature extraction in CNN

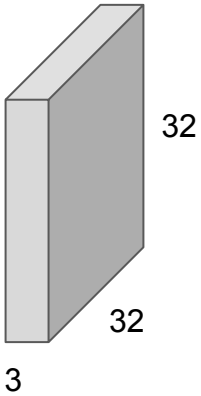


Goodfellow, 2016

Convolution layer

Input image

32 x 32 x 3



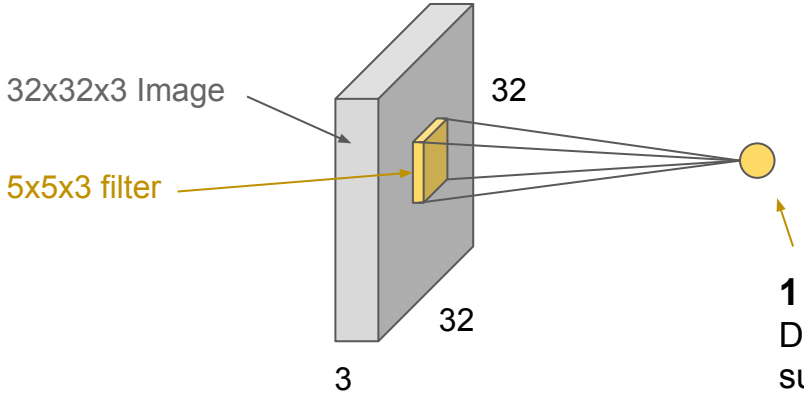
Filter/kernel

5 x 5 x 3

Convolution: slide the filter over the image spatially and compute the dot products

Adapted from CS231N, Stanford

Convolution layer

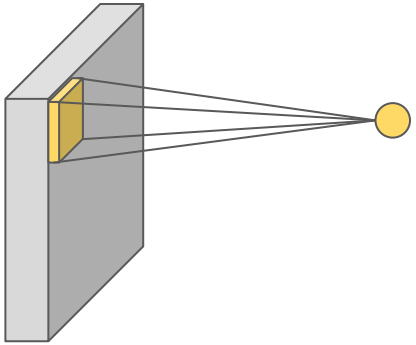


1 number:
Dot product between the filter and a small 5x5x3 sub-image (i.e., 5x5x3 = 75-dim dot product + bias)

Convolution: slide the filter over the image spatially and compute the dot products

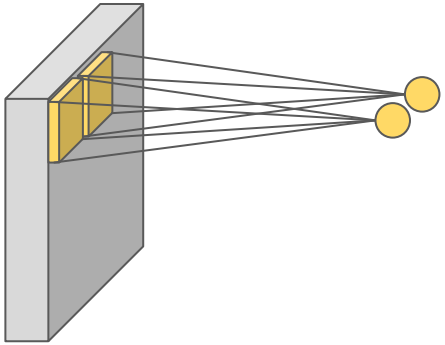
Adapted from CS231N, Stanford

Convolution layer



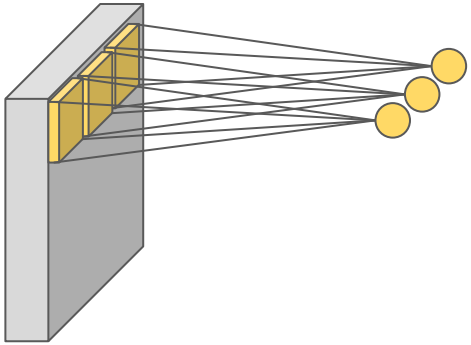
Adapted from CS231N, Stanford

Convolution layer



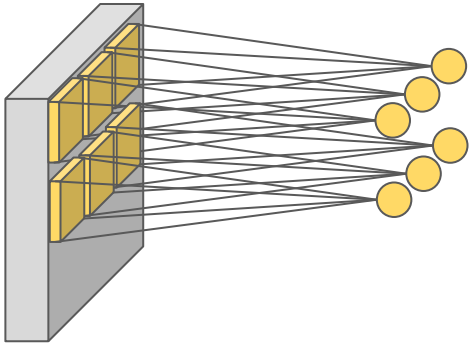
Adapted from CS231N, Stanford

Convolution layer



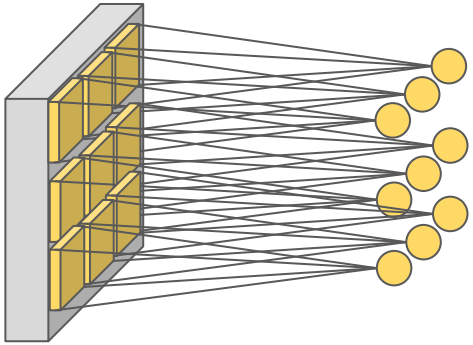
Adapted from CS231N, Stanford

Convolution layer



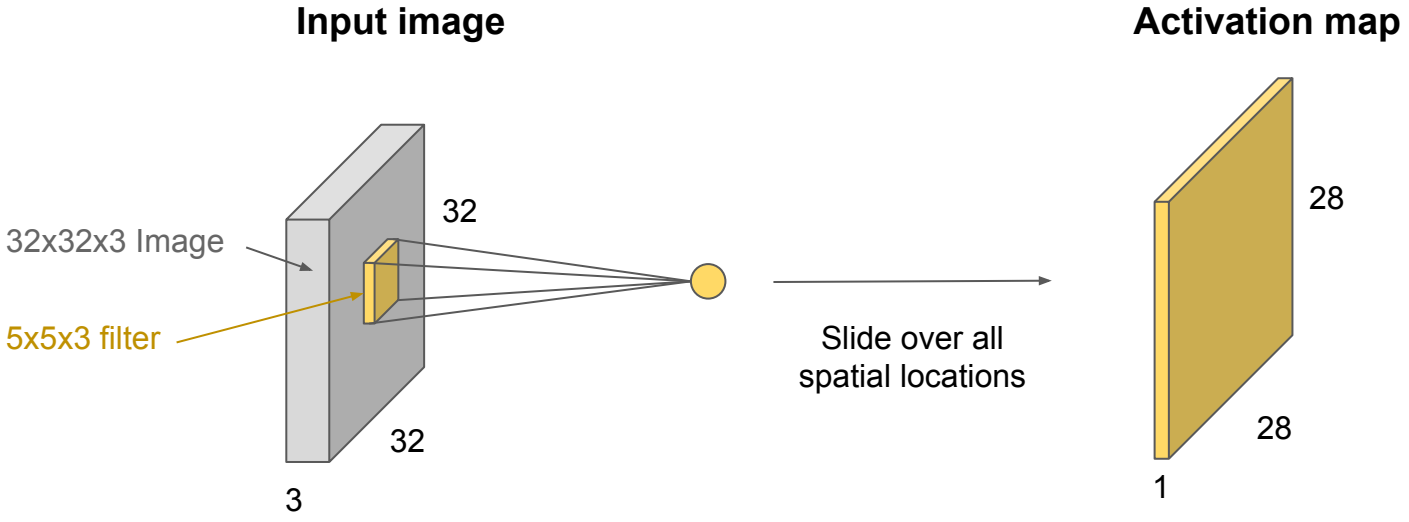
Adapted from CS231N, Stanford

Convolution layer



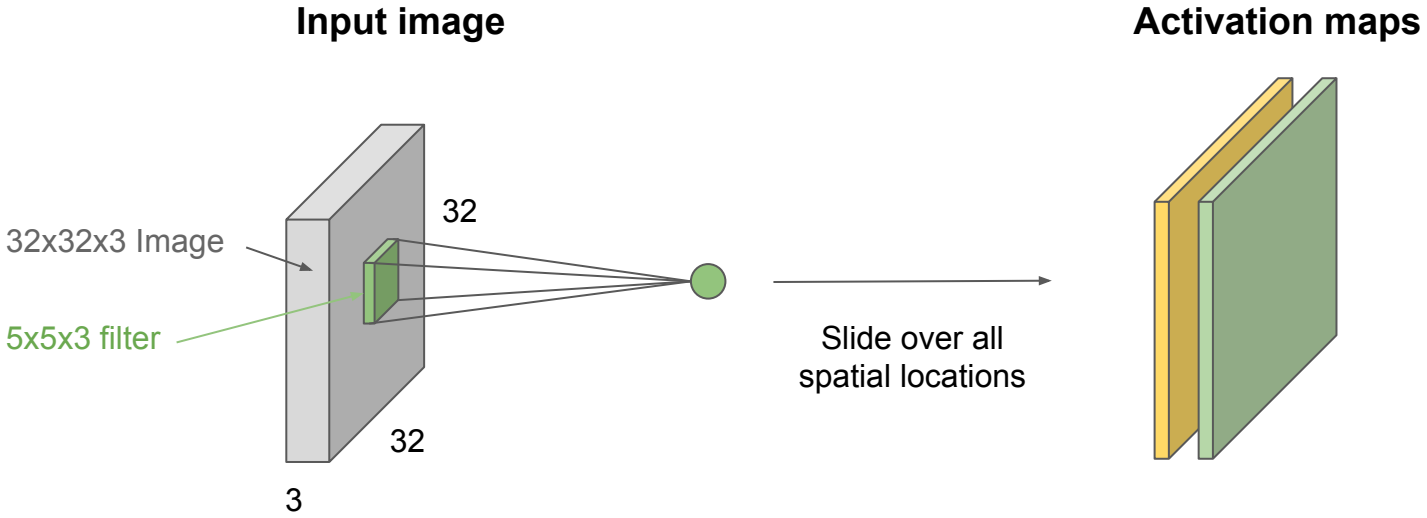
Adapted from CS231N, Stanford

Convolution layer



Adapted from CS231N, Stanford

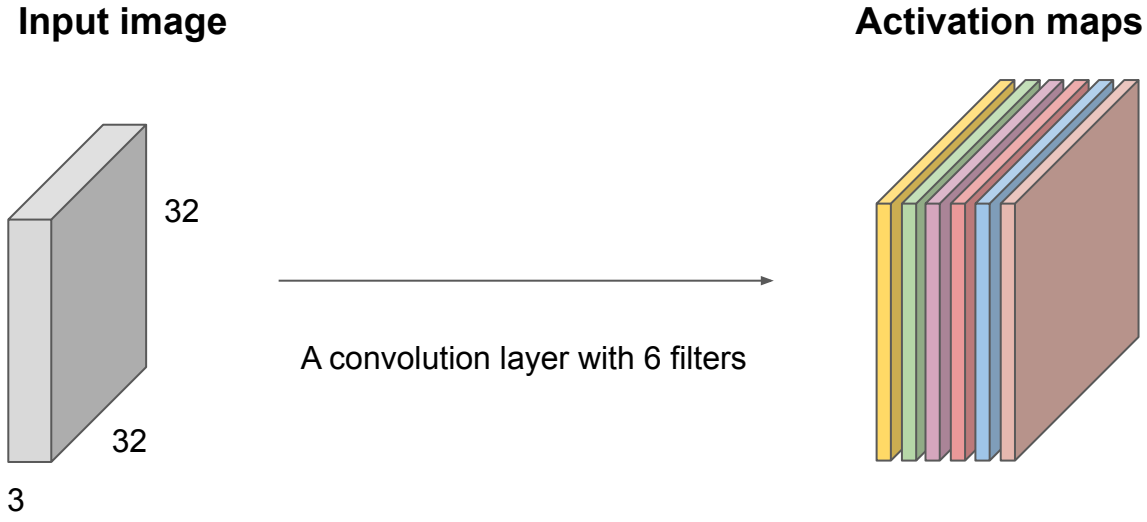
Convolution layer



Add a second filter

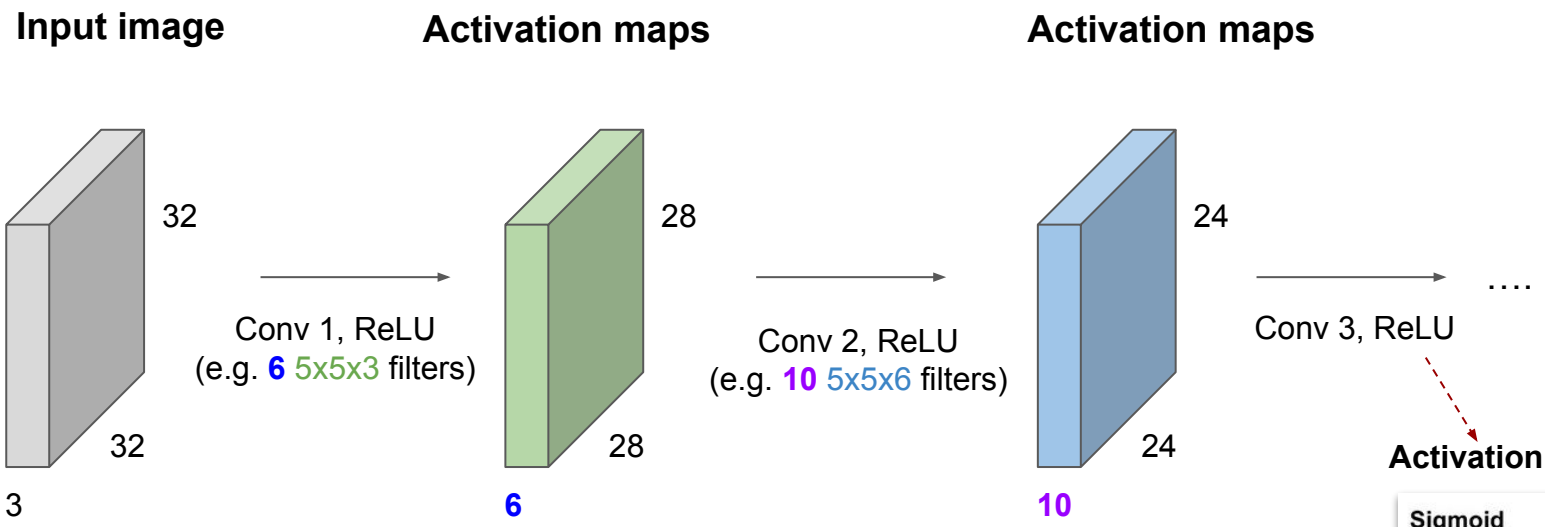
Adapted from CS231N, Stanford

Convolution layer

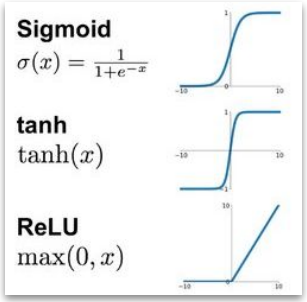


Adapted from CS231N, Stanford

CNN is a sequence of convolution layers, interspersed with activation functions

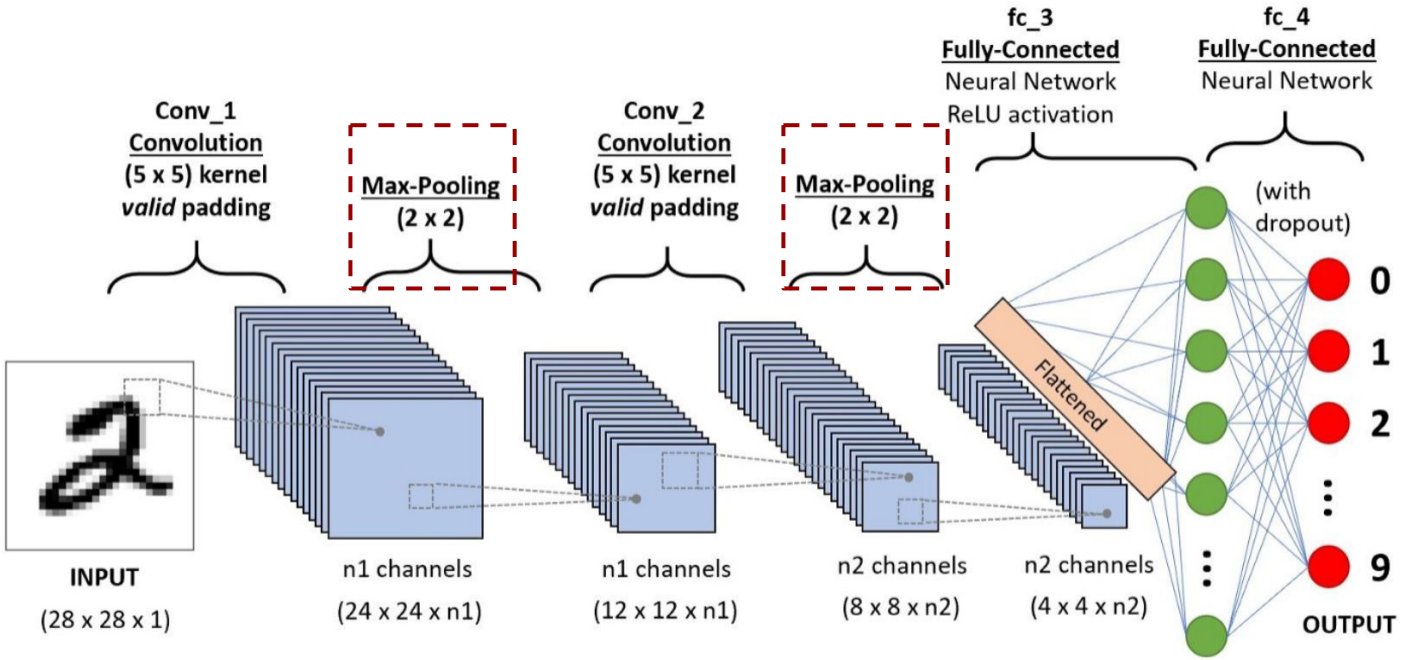


Activation functions

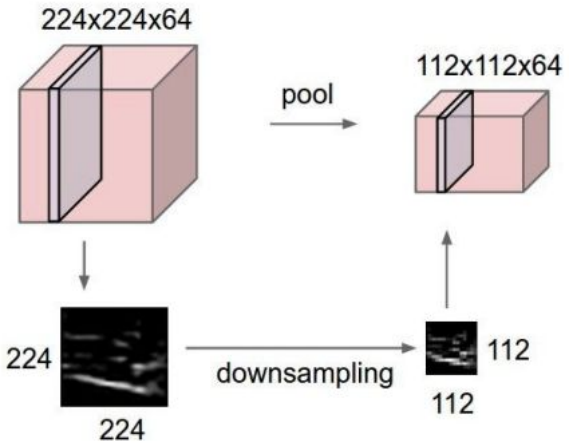


Adapted from CS231N, Stanford

Pooling layer

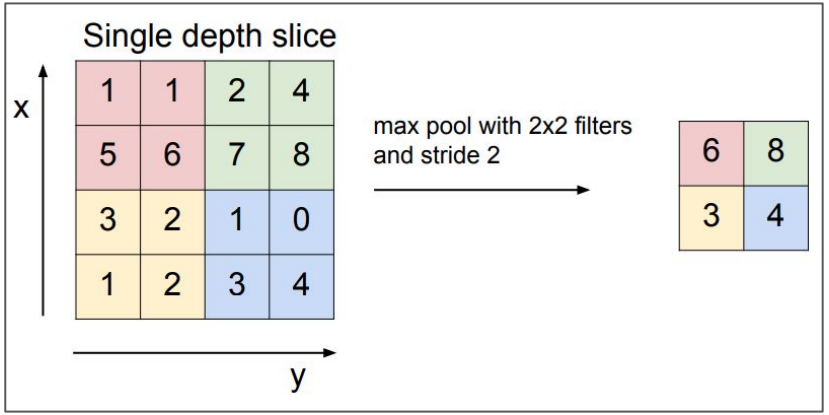


Pooling layer



Pooling layer makes the activation map smaller and more manageable

Max pooling

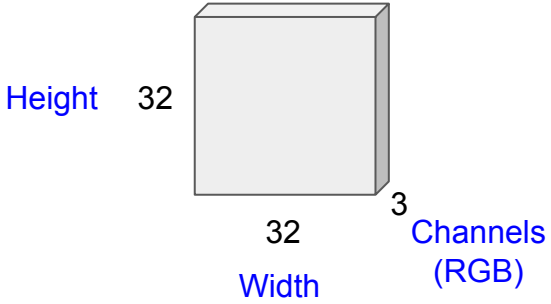


Adapted from CS231N, Stanford

CNN for biological sequences

Imge (2D CNN)

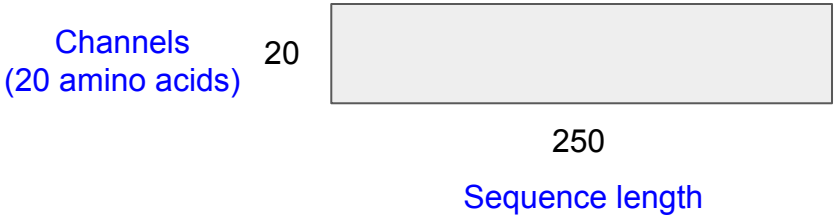
Input dimension: e.g., 32x32x3



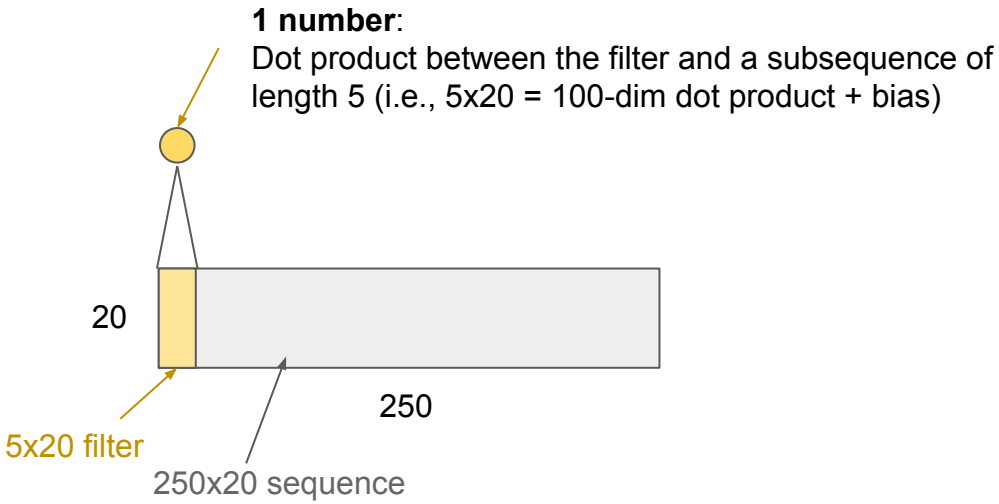
Sequence (1D CNN)

Input dimension: e.g., 250x20

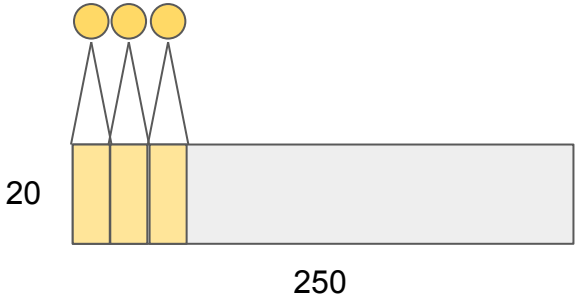
Alphabet size:
4 for DNA/RNA
20 for protein



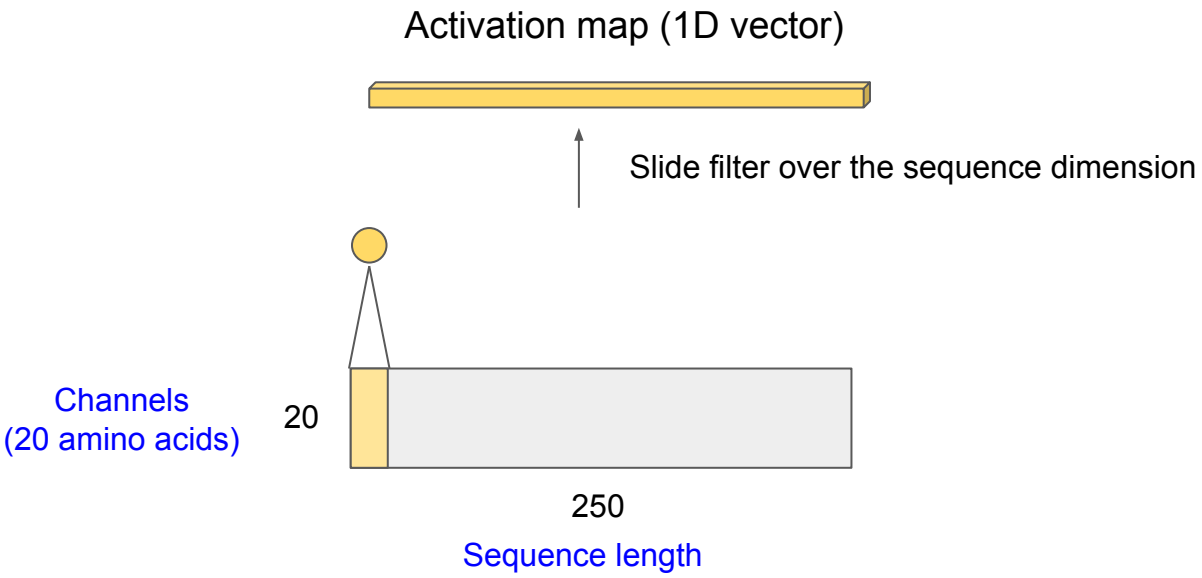
1D convolution



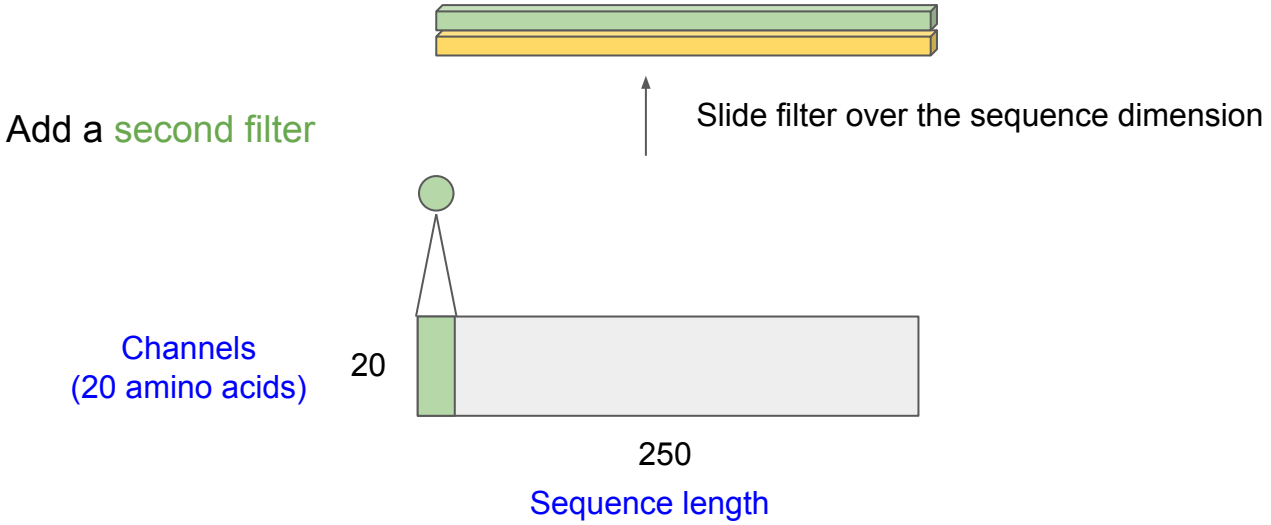
1D convolution



1D convolution



1D convolution



Convolution layers in PyTorch

PyTorch

Convolution Layers

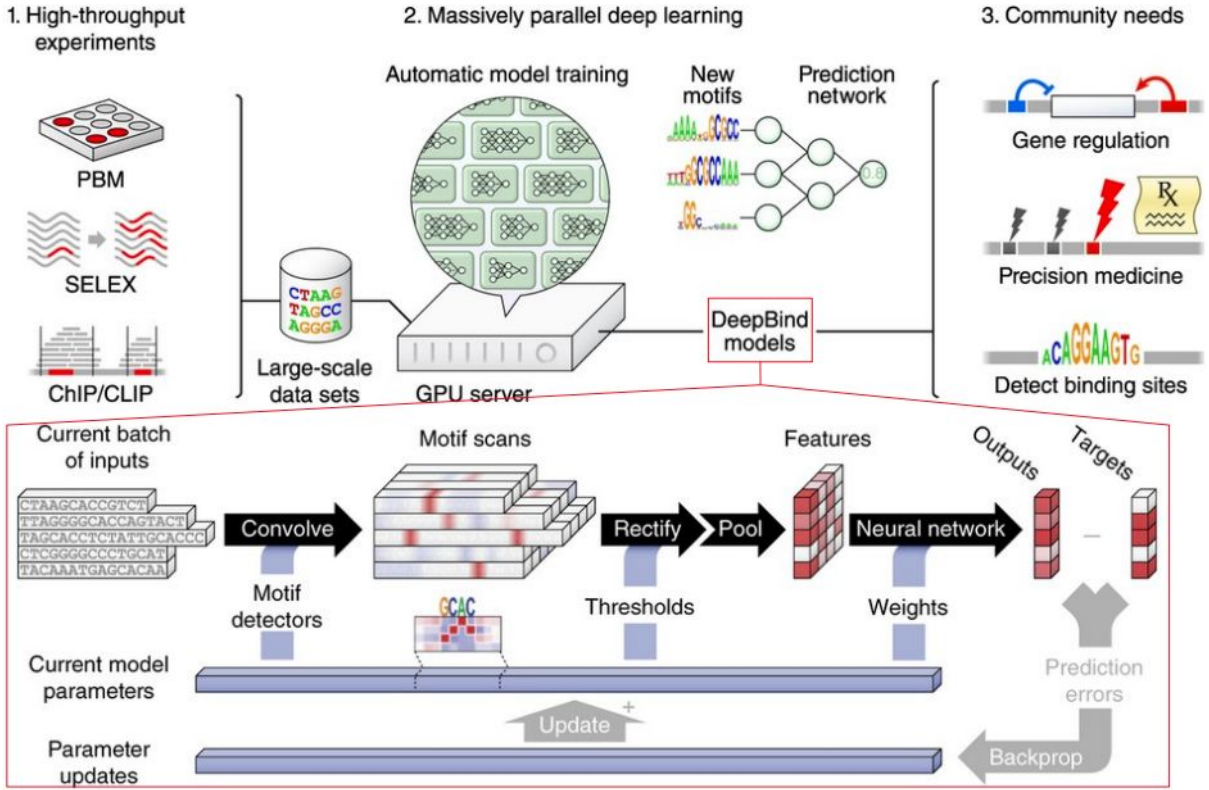
`nn.Conv1d`	Applies a 1D convolution over an input signal composed of several input planes.
`nn.Conv2d`	Applies a 2D convolution over an input signal composed of several input planes.
`nn.Conv3d`	Applies a 3D convolution over an input signal composed of several input planes.

An example of 2-layer 2D CNN for RGB image

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

Example: predicting DNA/RNA-protein binding

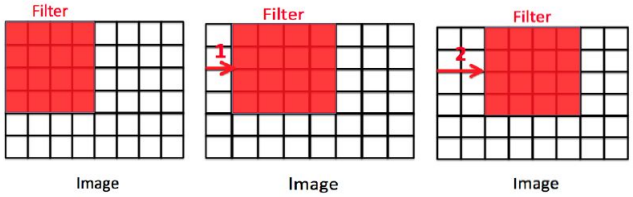


Alipanahi, Babak, et al. "Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning." *Nature biotechnology* 33.8 (2015): 831-838.

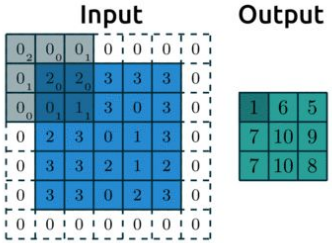
Further readings of CNN

Other important concepts of CNN:

- *Deep Learning*, Chapter 5



Stride



Padding

Image from Google

Summary: key ideas of CNNs

- Use a set of filters to extract **local** features
- Use **multiple** filters to extract different features
- Spatially **share** parameters of each filter

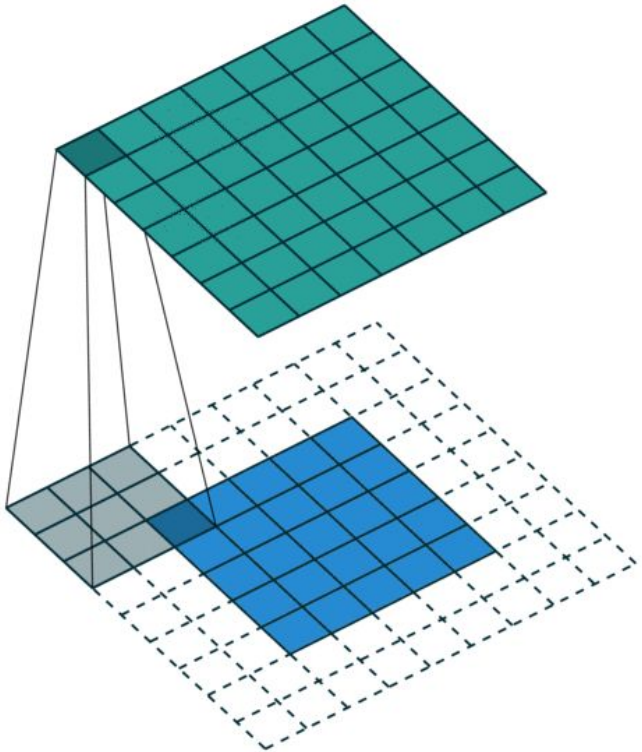


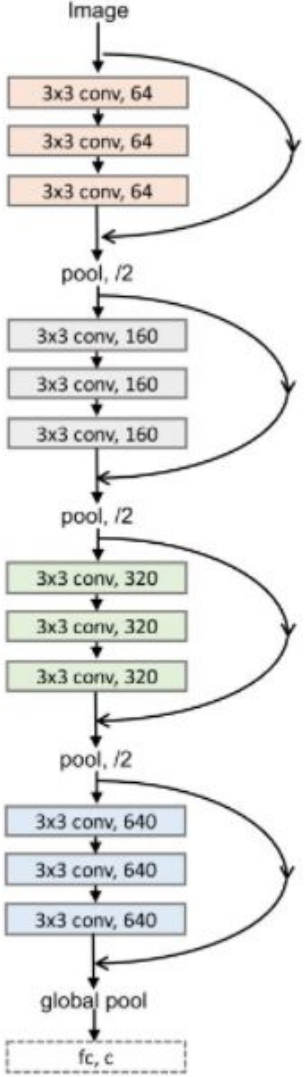
Image from Google

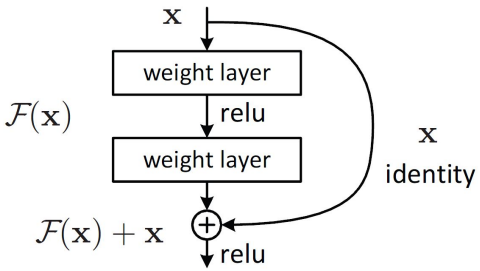
ResNet

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Deep residual learning for **image recognition**
K He, X Zhang, S Ren, J Sun - ... and pattern **recognition**, 2016 - openaccess.thecvf.com
Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer ...
Save Cite Cited by 103017 Related articles All 63 versions





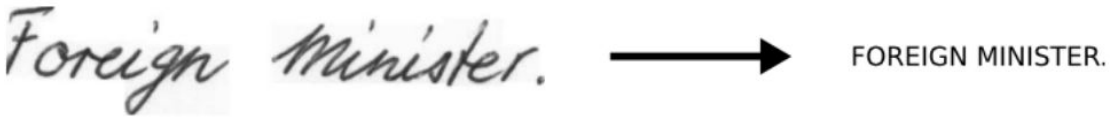
Model #2: Recurrent Neural Network (RNN)

Example: sequence labeling problems

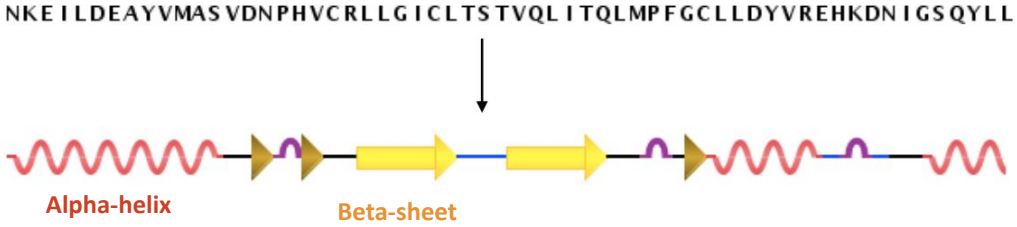
- Part of speech

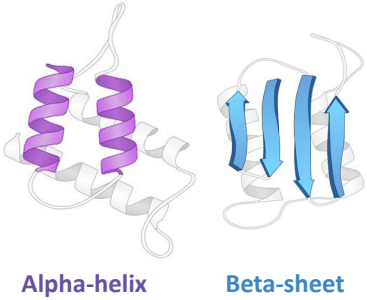
Article	Noun	Verb	Preposition	Article	Noun
The	cat	sat	on	the	mat.

- Handwriting recognition

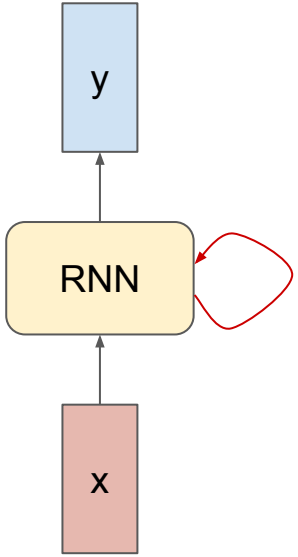


- Protein secondary structure prediction



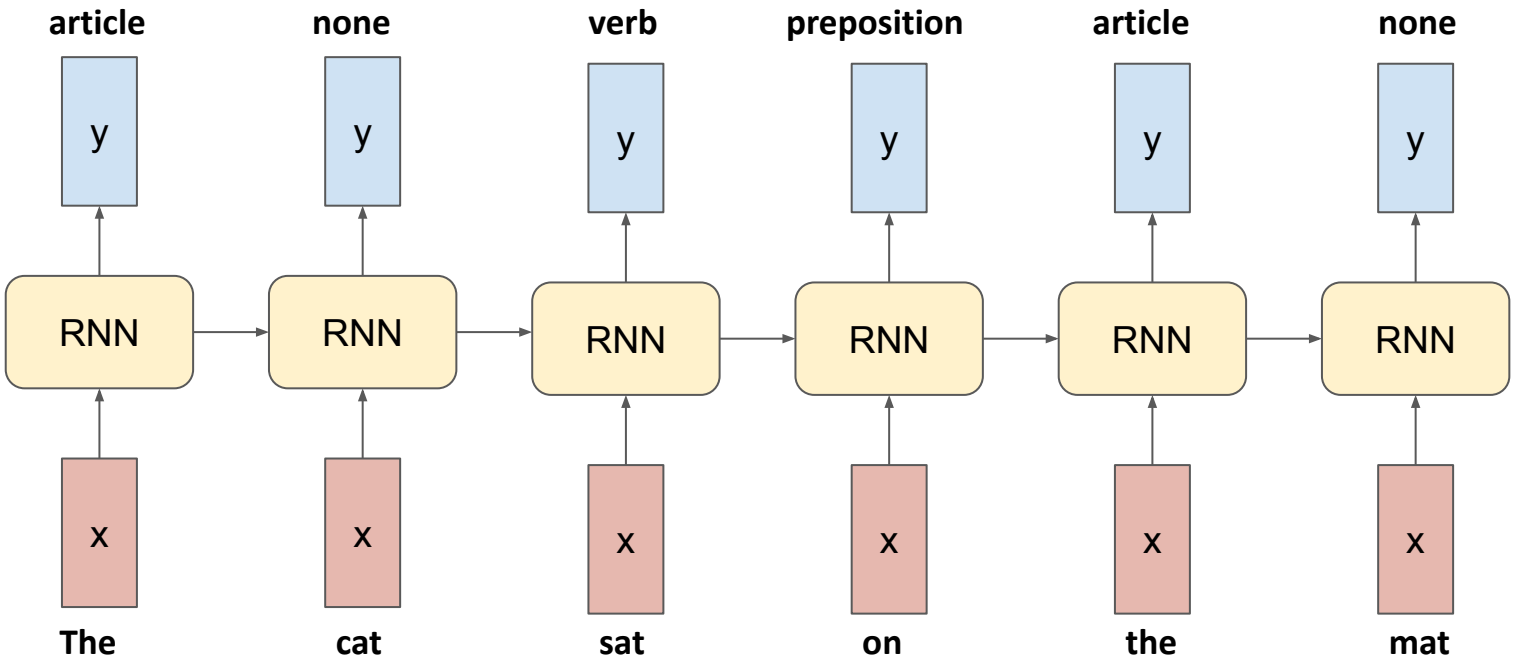


Recurrent Neural Network (RNN)

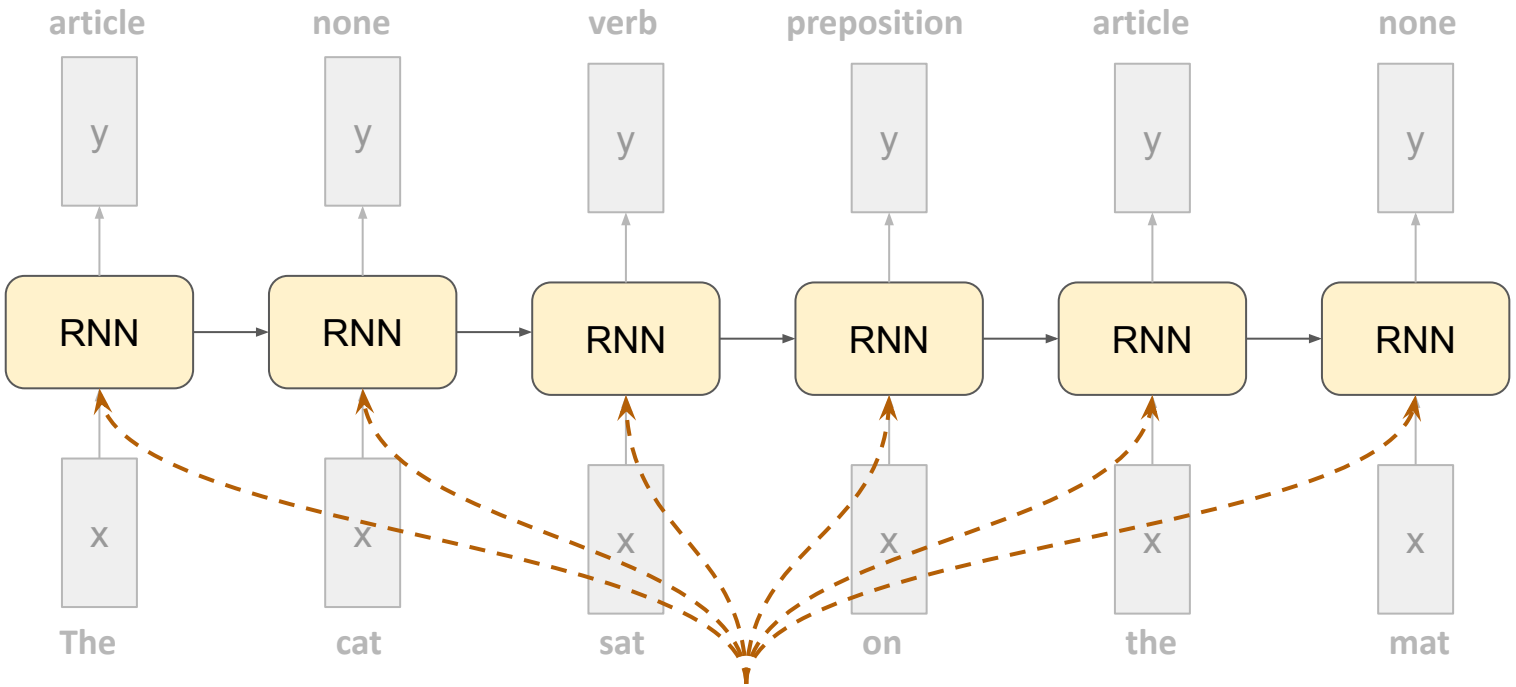


Key idea: RNNs have an “internal state” that is updated as the input sequence is processed

Unrolled RNN



Unrolled RNN

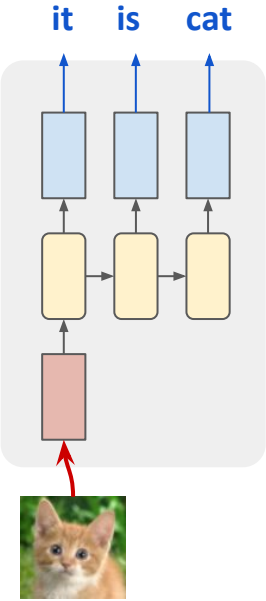


The same set of function and the same set of parameters are used at every time step

RNNs for sequence processing

one to many

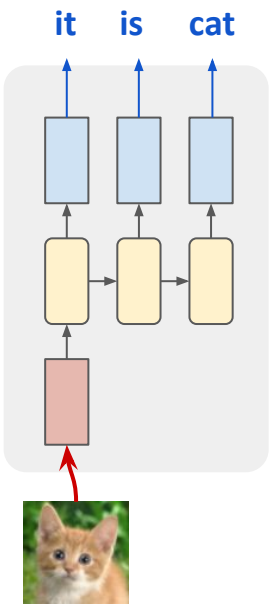
(e.g., image captioning)



RNNs for sequence processing

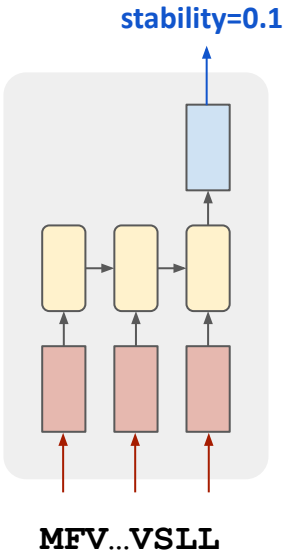
one to many

(e.g., image captioning)



many to one

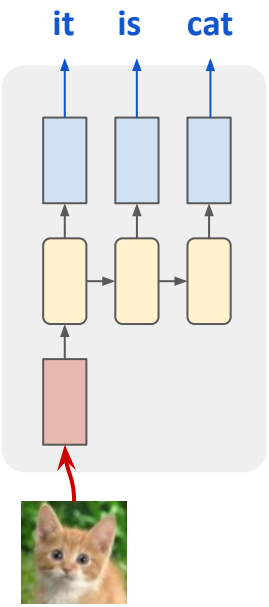
(e.g., protein function prediction)



RNNs for sequence processing

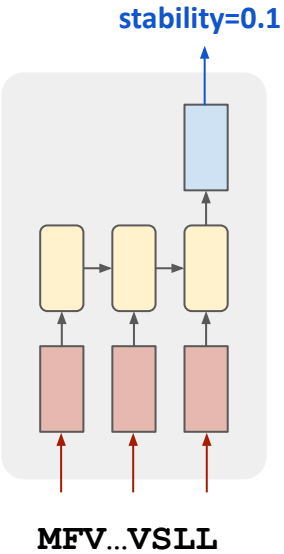
one to many

(e.g., image captioning)



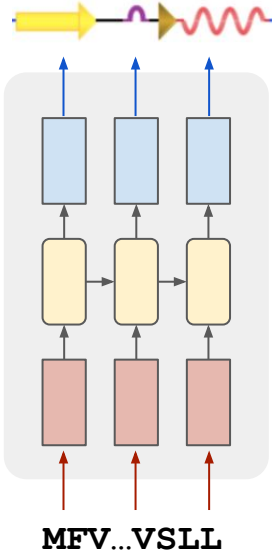
many to one

(e.g., protein function prediction)



many to many

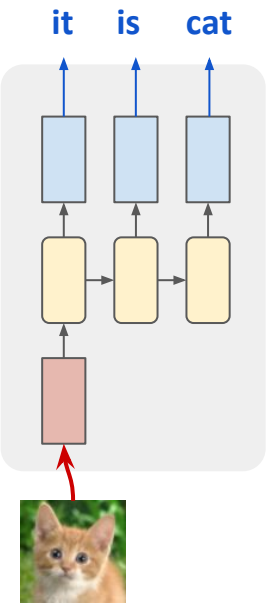
(e.g., protein structure prediction)



RNNs for sequence processing

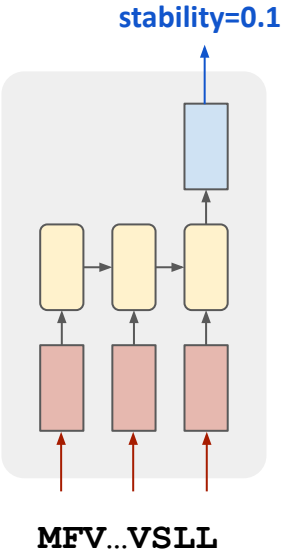
one to many

(e.g., image captioning)



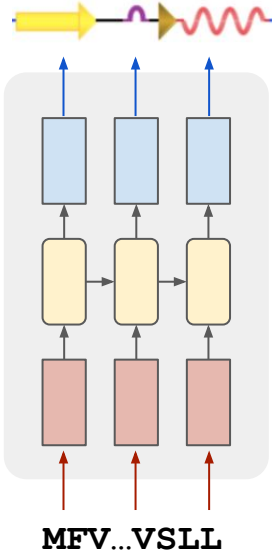
many to one

(e.g., protein function prediction)



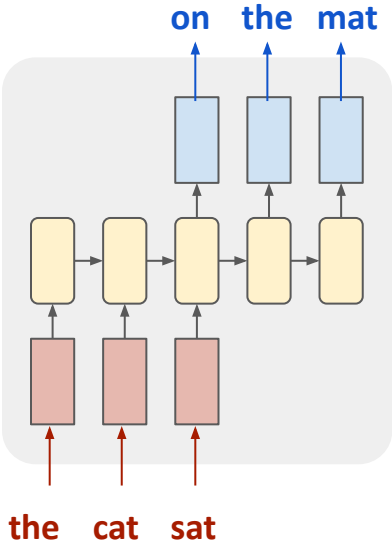
many to many

(e.g., protein structure prediction)

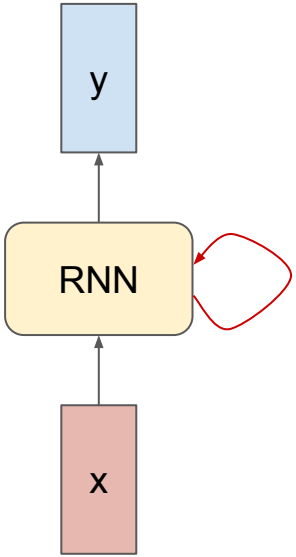


many to many

(e.g., auto completion)



RNN hidden state update



At every step t, the hidden state is updated based on the previous state and the current input

function with parameter *W*

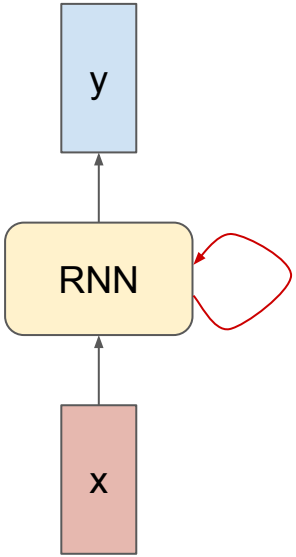
$$h_t = f_W(h_{t-1}, x_t)$$

new state

previous state

Input vector at time step *t*

RNN output



At every step t, the output is generated based on the current state

Another function with parameter *O*

$$\boxed{y_t} = \boxed{f_o}(\boxed{h_t})$$

new state

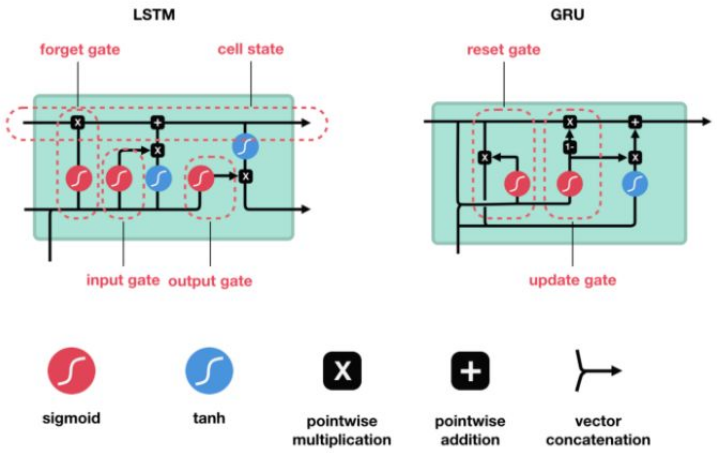
new state

Further readings of RNN

- *Deep Learning*, Chapter 6
- What is function *f*()?
  - f() is usually called “unit” in RNN
  - It defines a “computational graph” the produces h_t based on h_{t-1} and x_t
- Popular RNN variants
  - Long short-term memory (LSTM)
  - Gated recurrent unit (GRU)

function with parameter *W*

$$h_t = \boxed{f_W}(h_{t-1}, x_t)$$



(image source)

Recurrent layers in PyTorch

PyTorch

Recurrent Layers

`nn.RNNBase`	
`nn.RNN`	Applies a multi-layer Elman RNN with $\tanh$ or ReLU non-linearity to an input sequence.
`nn.LSTM`	Applies a multi-layer long short-term memory (LSTM) RNN to an input sequence.
`nn.GRU`	Applies a multi-layer gated recurrent unit (GRU) RNN to an input sequence.
`nn.RNNCell`	An Elman RNN cell with tanh or ReLU non-linearity.
`nn.LSTMCell`	A long short-term memory (LSTM) cell.
`nn.GRUCell`	A gated recurrent unit (GRU) cell

Summary: key ideas of RNNs

- Process sequence data with **variable lengths**
  - DNA/RNA/protein sequences, text, audio, time series data
- Capture **sequential** (temporal) information/dependencies in the data
- Parameters **shared** over time steps
- Common to use LSTM or GRU

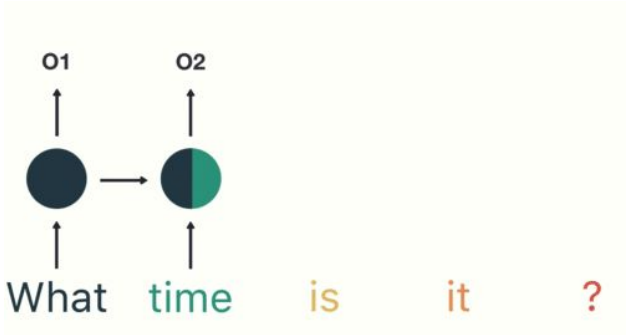
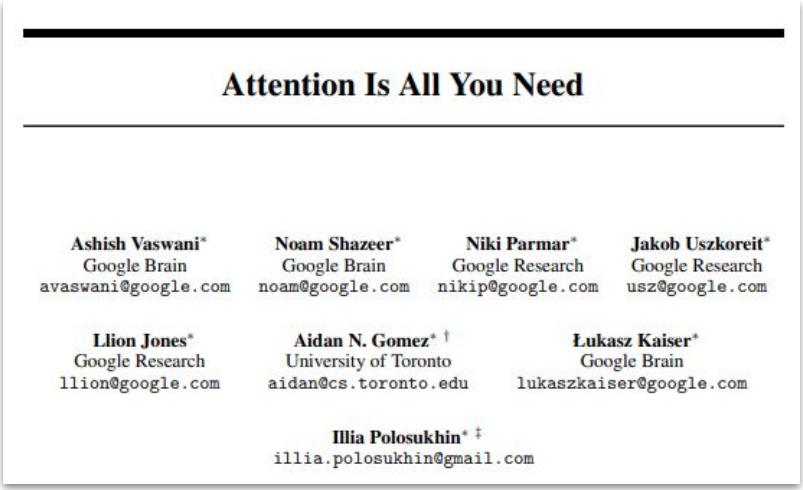


Image from Google

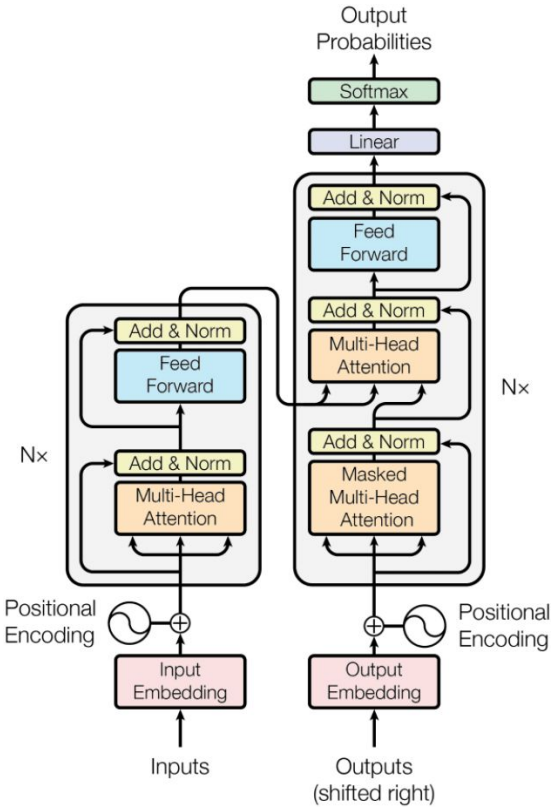
Model #3: Transformers



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

(NeurIPS 2017)

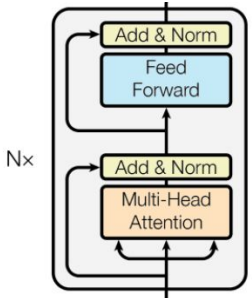


- Encoder-Decoder
- Sequence-to-sequence
- Transforms one sequence into another sequence, using full context of each

Building blocks of Transformer

N blocks, each has

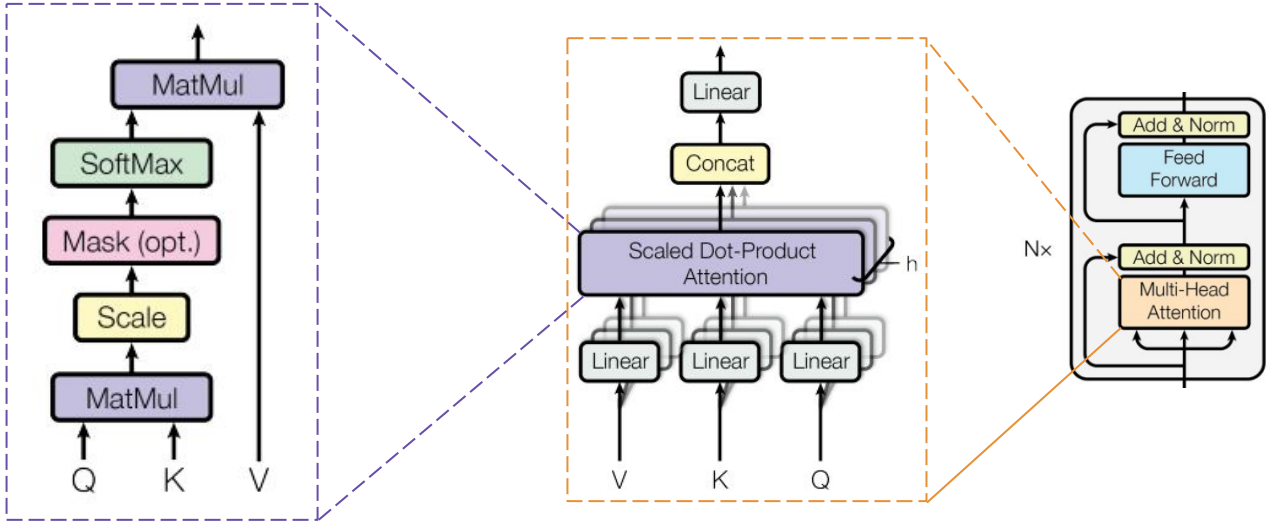
- Multi-head self-attention layer
- Two-layer feed-forward neural nets



Residual connection and layer normalization are used

- Reading: LayerNorm (https://arxiv.org/pdf/1607.06450.pdf)

Building blocks of Transformer



Key ideas: self-attention layer

- **Attention layer**: a layer to learn the dependency between words in the input. The dependency is quantified using “attention weights”
- For each word, a new representation is computed by weight-averaging the old representations of all words, where the weight is the learned attention weight

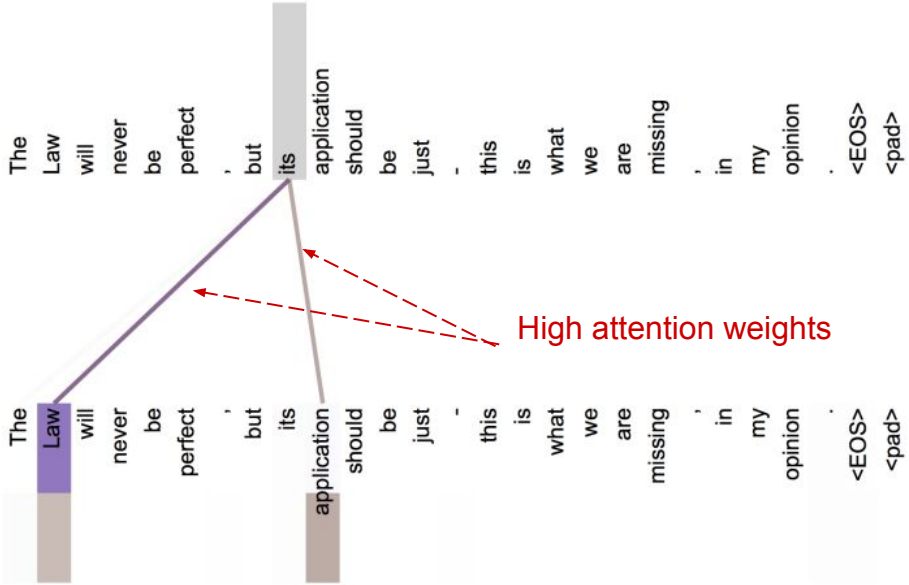


Image credit: CS224N, Stanford

Key ideas: self-attention layer

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Q: “query” matrix, a vector representation for each word
K: “key” matrix, a vector representation for each word
V: “value” matrix, a vector representation for each word

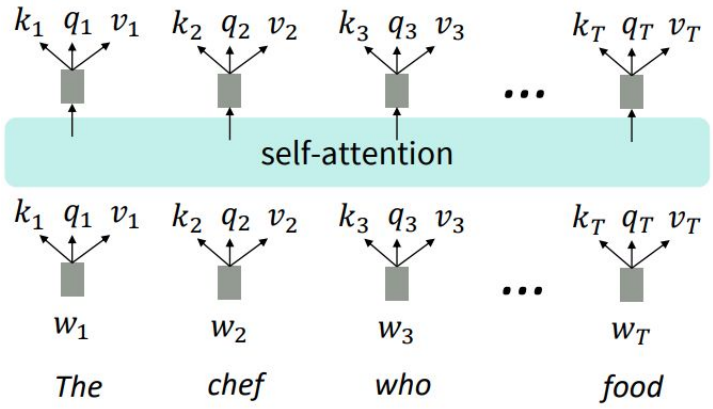
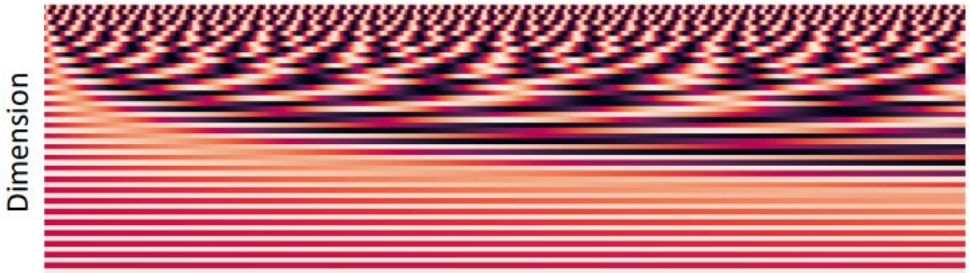


Image credit: CS224N, Stanford

Key ideas: positional encoding

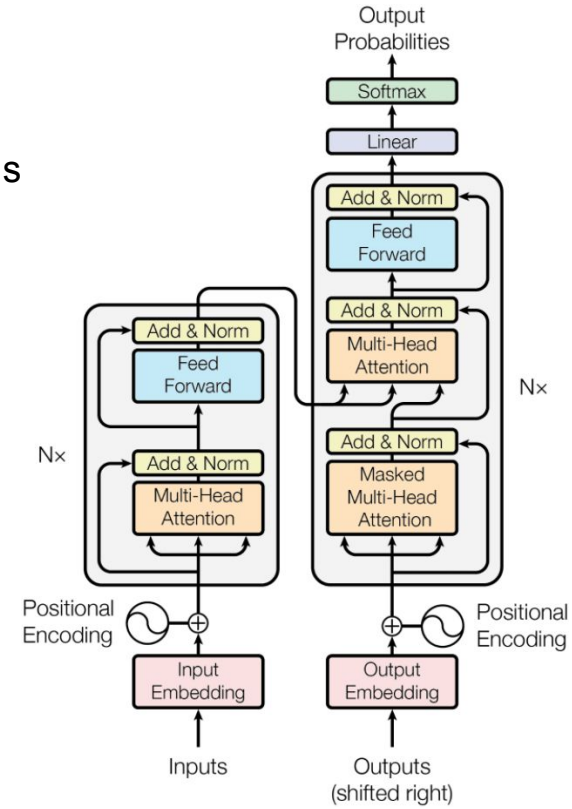
- Self-attention does not know the order of input words
- Positional encodings are added to the word representations, so same words at different locations have different overall representations

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Summary of Transformer

- Learning temporal relationships without unrolling and without RNNs
- Encoder/Decoder framework, multi-head self-attention modules
- Widely used in state-of-the-art NLP models
- Readings:
  - “Attention is all you need” (https://arxiv.org/abs/1706.03762)
  - PyTorch implementation and tutorial of Transformer

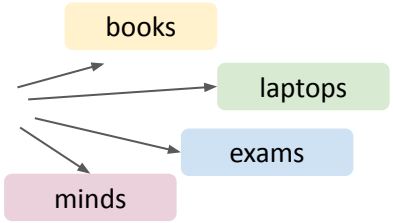


Language modeling of protein sequence

Language modeling in natural language

- **Language modeling** is the task of predicting what word comes next

"The students opened their _____"



- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V=\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$.

- A system that does this is called a **language model (LM)**

Adapted from CS224N, Stanford

Large LMs

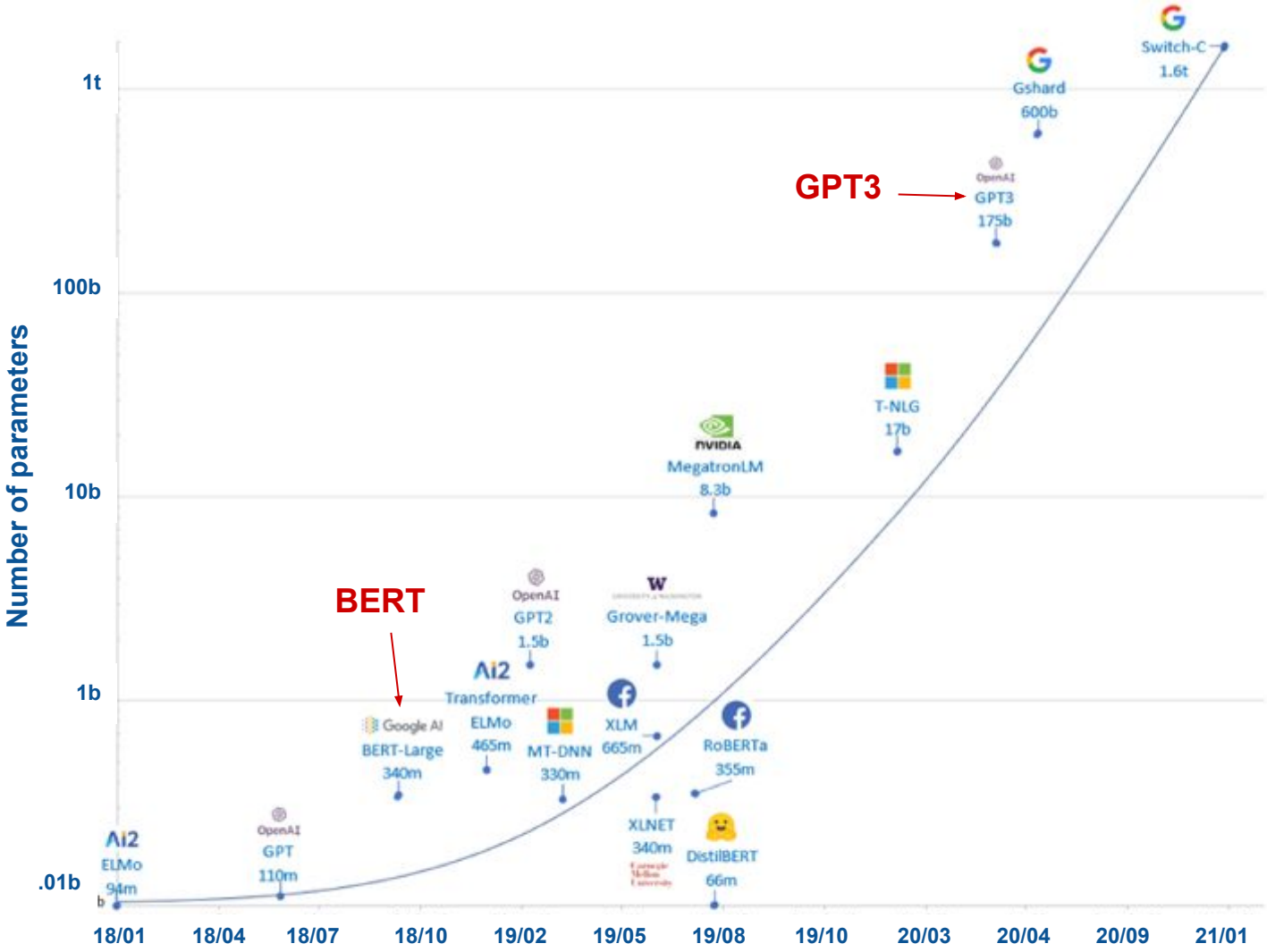
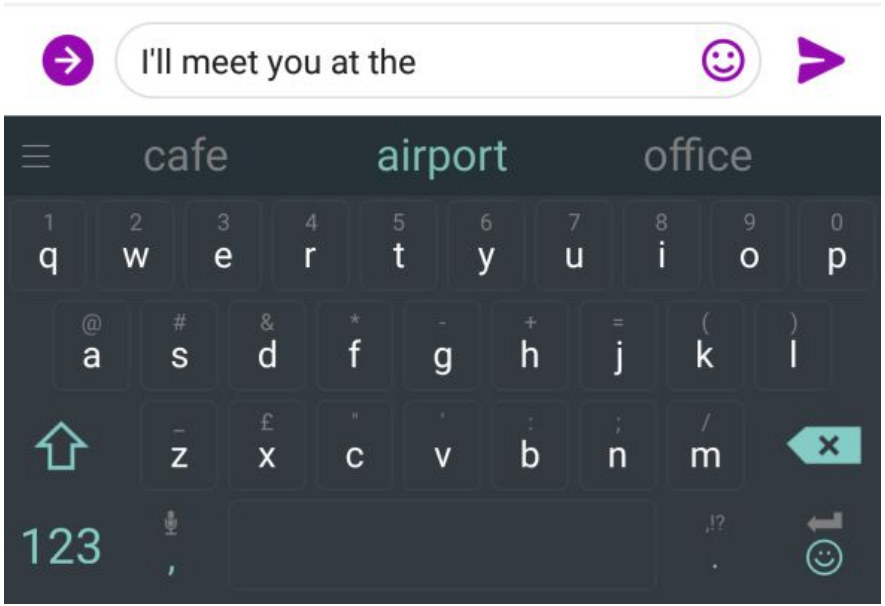


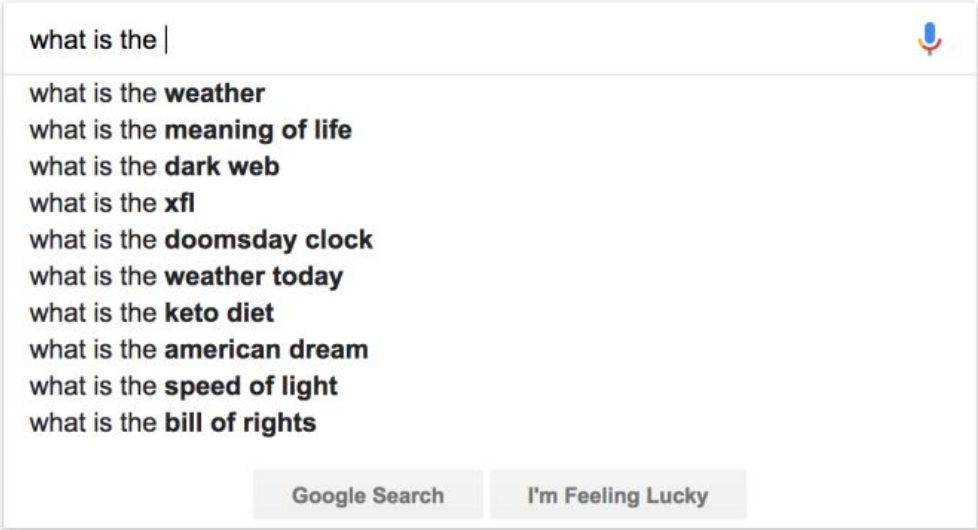
Image source

You use LM every day!



Adapted from CS224N, Stanford

You use LM every day!



Adapted from CS224N, Stanford

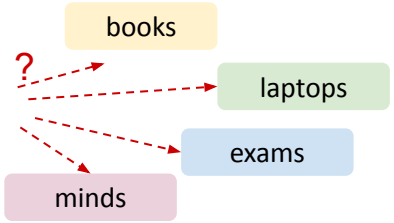
You use LM every day!

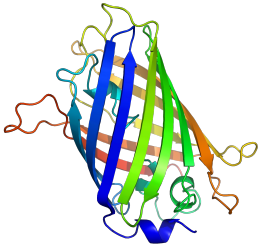
GitHub Copilot

https://copilot.github.com/

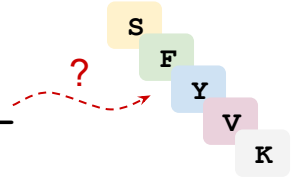
Learning the language of proteins

The students opened their _____

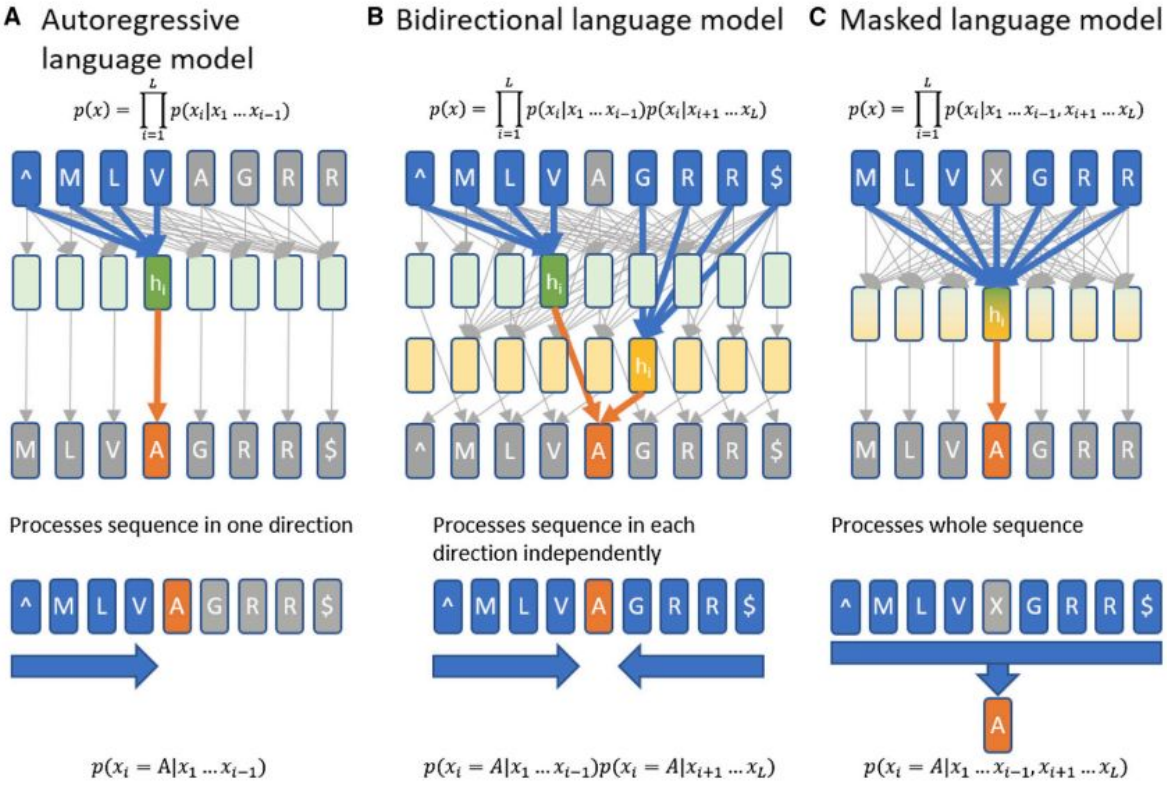




MSKGEELFTGVVPILVELDGDVNGHKFSV_

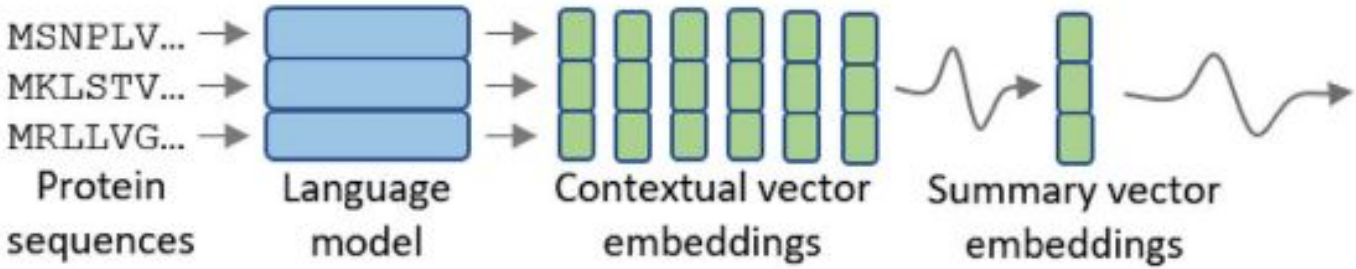


Different language modeling approaches



Bepler and Berger, *Cell Systems*, 2021

Protein language models

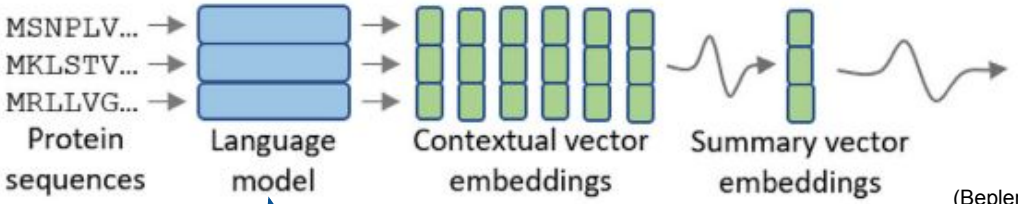


Downstream tasks

Pre-trained on millions or more protein sequences

Bepler and Berger, *Cell Systems*, 2021

Application: improving downstream prediction tasks

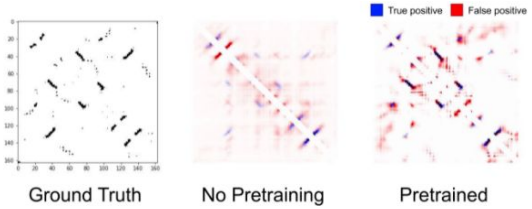


Downstream tasks

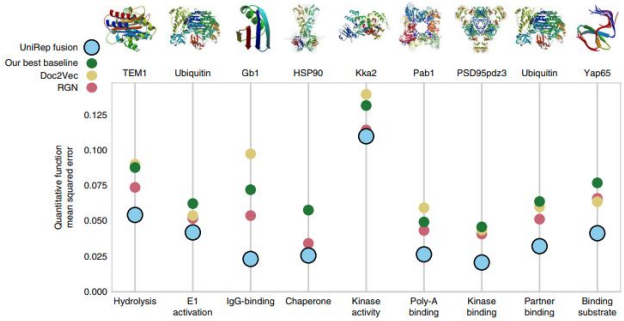
(Bepler and Berger, *Cell Systems*, 2021)

Pre-trained on millions or more protein sequences

Protein contact prediction
(Rao et al., NeurIPS, 2019)



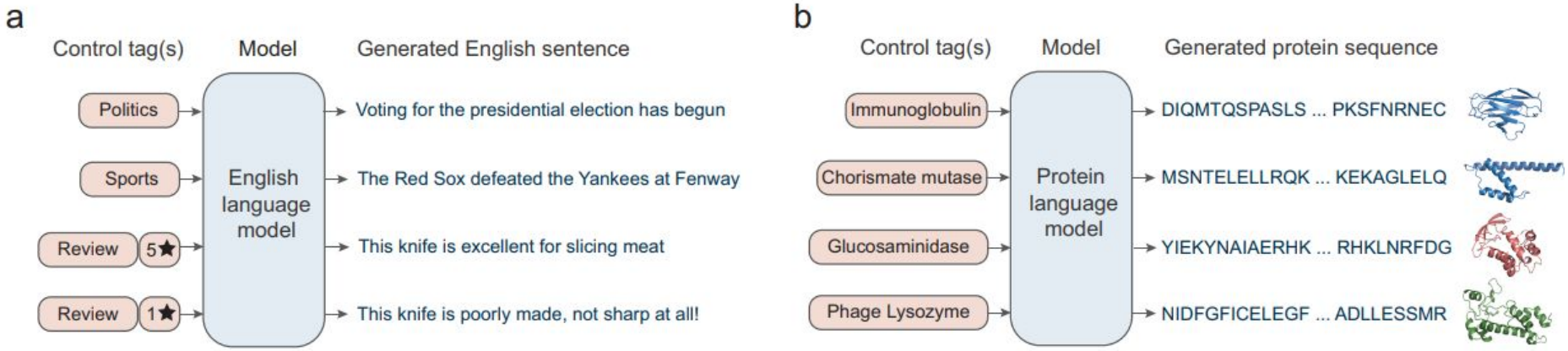
Protein function prediction
(Alley et al., Nature Methods, 2019)



Remote homology detection
(Rives et al., PNAS, 2021)

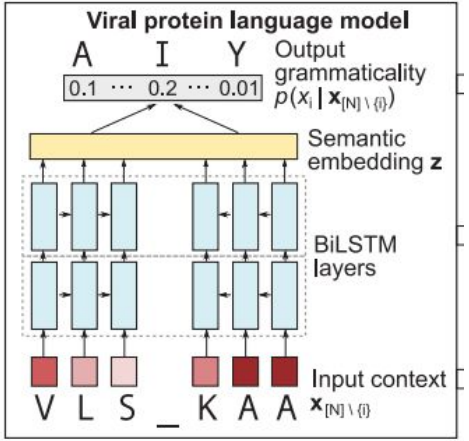
		Hit-10		AUC	
	Pretraining	Fold	SF	Fold	SF
HHblits*		0.584	0.965	0.831	0.951
LSTM (S)	UR50/S	0.558	0.760	0.801	0.863
LSTM (L)	UR50/S	0.574	0.813	0.805	0.880
Transformer-6	UR50/S	0.653	0.878	0.768	0.901
Transformer-12	UR50/S	0.639	0.915	0.778	0.942
Transformer-34	(None)	0.481	0.527	0.755	0.807
Transformer-34	UR100	0.599	0.841	0.753	0.876
Transformer-34	UR50/D	0.617	0.932	0.822	0.932
Transformer-34	UR50/S	0.639	0.931	0.825	0.933
ESM-1b	UR50/S	0.532	0.913	0.770	0.880

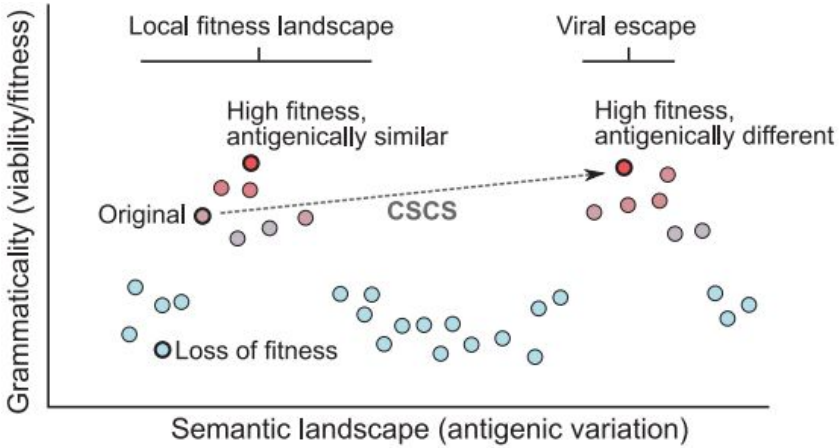
Application: generating novel functional protein sequences



Controllable generation of protein sequences
(Madani et al., bioRxiv, 2021)

Application: unsupervised prediction of viral escape





LM predicts viral escape
(Hie et al., Science, 2021)

Summary of today

- Biological sequences
  - DNA, RNA, protein
- CNNs
  - Extract spatial features using convolution filters
- RNNs
  - Capture temporal dependency
  - LSTM, GRU
- Transformers
  - Self-attention layer
  - Widely used in recent state-of-the-art NLP models
- Example: language model of protein sequences
  - LM: predicting a word given context
  - Protein LM used for supervised predictions, sequence generation, unsupervised prediction (e.g., viral escape)