10-423/623: Generative AI Lecture 5 – Vision Transformers

Henry Chai & Matt Gormley

9/11/24

Front Matter

- Announcements:
  - HW1 released 9/9, due 9/23 at 11:59 PM
    - Recitation on 9/13 (this Friday) will be on HW1 topics

Convolutional Neural Networks

- Neural networks are frequently applied to inputs with some inherent spatial structure, e.g., images
- Idea: use the first few layers to identify relevant macro-features, e.g., edges
- Insight: for spatially-structured inputs, many useful macro-features are shift or location-invariant, e.g., an edge in the upper left corner of a picture looks like an edge in the center
- Strategy: learn a *filter* for macro-feature detection in a small window and apply it over the entire image

Convolutional Filters

- Images can be represented as matrices: each element corresponds to a pixel and its value is the intensity
- A filter is just a small matrix that is *convolved* with same-sized sections of the image matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Convolutional Filters

- Images can be represented as matrices: each element corresponds to a pixel and its value is the intensity
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}
*
\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}
=
\begin{bmatrix} 0 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}
$$

$$
(0*0)+(0*1)+(0*0)+(0*1)+(1*-4) \\ +(2*1)+(0*0)+(2*1)+(4*0)=0
$$

Convolutional Filters

- Images can be represented as matrices: each element corresponds to a pixel and its value is the intensity
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 2 & 1 & 0 \\
0 & 2 & 4 & 4 & 2 & 0 \\
0 & 1 & 3 & 3 & 1 & 0 \\
0 & 1 & 2 & 3 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0
\end{bmatrix}
*
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
=
\begin{bmatrix}
0 & -1 & & \\
 & & & \\
 & & & \\
 & & &
\end{bmatrix}
$$

$$
(0 * 0) + (0 * 1) + (0 * 0) + (1 * 1) + (2 * -4) \\
+ (2 * 1) + (2 * 0) + (4 * 1) + (4 * 0) = -1
$$

Convolutional Filters

- Images can be represented as matrices: each element corresponds to a pixel and its value is the intensity
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

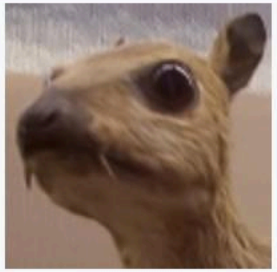
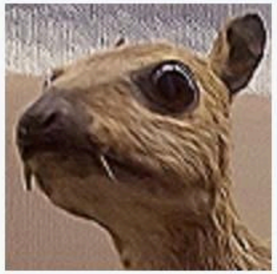
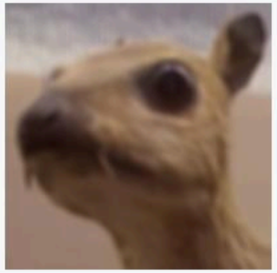
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 & 0 \\ -2 & -5 & -5 & -2 \\ 2 & -2 & -1 & 3 \\ -1 & 0 & -5 & 0 \end{bmatrix}$$

Convolutional Filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

More Filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Source: https://en.wikipedia.org/wiki/Kernel_(image_processing)

Convolutional Filters

- Images can be represented as matrices: each element corresponds to a pixel and its value is the intensity
- A filter is just a small matrix that is convolved with same-sized sections of the image matrix

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

*

0	1	0
1	-4	1
0	1	0

=

0	-1	-1	0
-2	-5	-5	-2
2	-2	-1	3
-1	0	-5	0

Convolutional Filters

- Convolutions can be represented by a feed forward neural network where:
  1. Nodes in the input layer are only connected to some nodes in the next layer but not all nodes.
  2. Many of the weights have the same value.

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

0	-1	-1	0
-2	-5	-5	-2
2	-2	-1	3
-1	0	-5	0

- Many fewer weights than a fully connected layer!
- **Convolution weights are learned using gradient descent/ backpropagation, not prespecified**

Convolutional Filters: Padding

- What if relevant features exist at the border of our image?
- Add zeros around the image to allow for the filter to be applied "everywhere" e.g. a *padding* of 1 with a 3x3 filter preserves image size and allows every pixel to be the center

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	2	2	1	0	0
0	0	2	4	4	2	0	0
0	0	1	3	3	1	0	0
0	0	1	2	3	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

$*$

0	1	0
1	-4	1
0	1	0

$=$

0	1	2	2	1	0
1	0	-1	-1	0	1
2	-2	-5	-5	-2	2
1	2	-2	-1	3	1
1	-1	0	-5	0	1
0	2	-1	0	2	0

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

$$\begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 2 & 2 & 1 & 0 \\ \hline 0 & 2 & 4 & 4 & 2 & 0 \\ \hline 0 & 1 & 3 & 3 & 1 & 0 \\ \hline 0 & 1 & 2 & 3 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & -2 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline -2 & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1
1	-2

$=$

-2	-2	

Downsampling: Stride

- Only apply the convolution to some subset of the image
  e.g., every other column and row = a *stride* of 2

0	0	0	0	0	0
0	1	2	2	1	0
0	2	4	4	2	0
0	1	3	3	1	0
0	1	2	3	1	0
0	0	1	1	0	0

$*$

0	1
1	-2

$=$

-2	-2	1

Downsampling: Stride

- Only apply the convolution to some subset of the image

  e.g., every other column and row = a *stride* of 2

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 1 \\ 0 & & \\ & & \end{bmatrix}$$

Downsampling: Stride

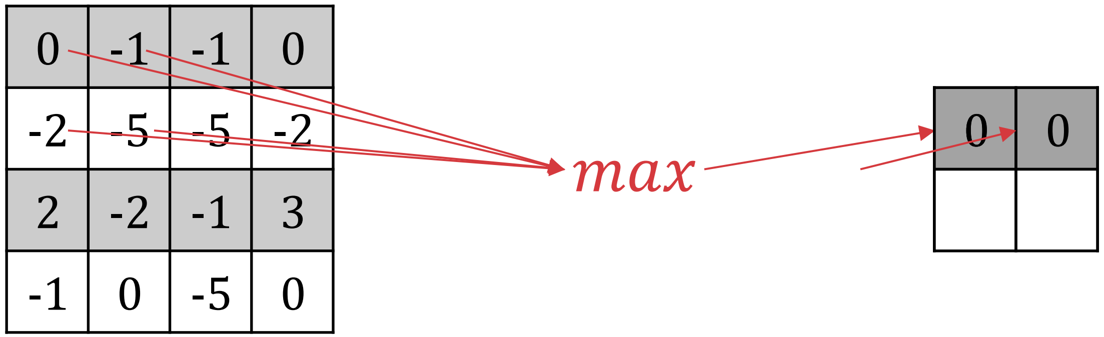
- Only apply the convolution to some subset of the image e.g., every other column and row = a *stride* of 2

$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 1 & 0 \\ 0 & 2 & 4 & 4 & 2 & 0 \\ 0 & 1 & 3 & 3 & 1 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 1 \\ 0 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix}
$$

- Reduces the dimensionality of the input to subsequent layers and thus, the number of weights to be learned
- Many relevant macro-features will tend to span large portions of the image, so taking strides with the convolution tends not to miss out on too much

Downsampling: Pooling

- Combine multiple adjacent nodes into a single node



Downsampling: Pooling

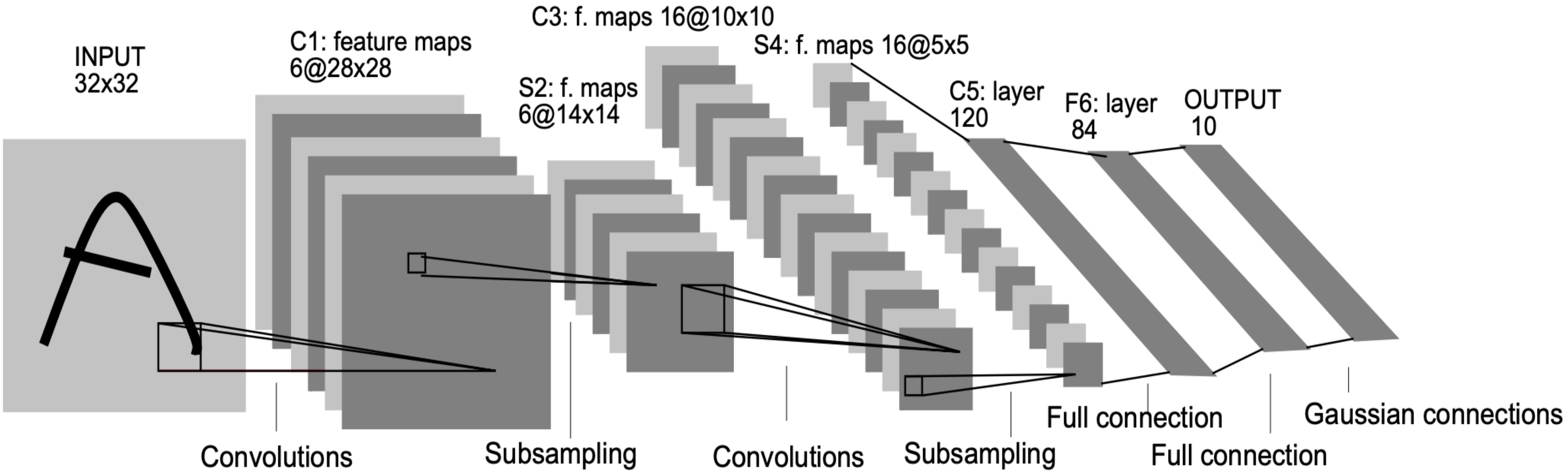
- Combine multiple adjacent nodes into a single node

0	-1	-1	0
-2	-5	-5	-2
2	-2	-1	3
-1	0	-5	0

→ *max pooling* →

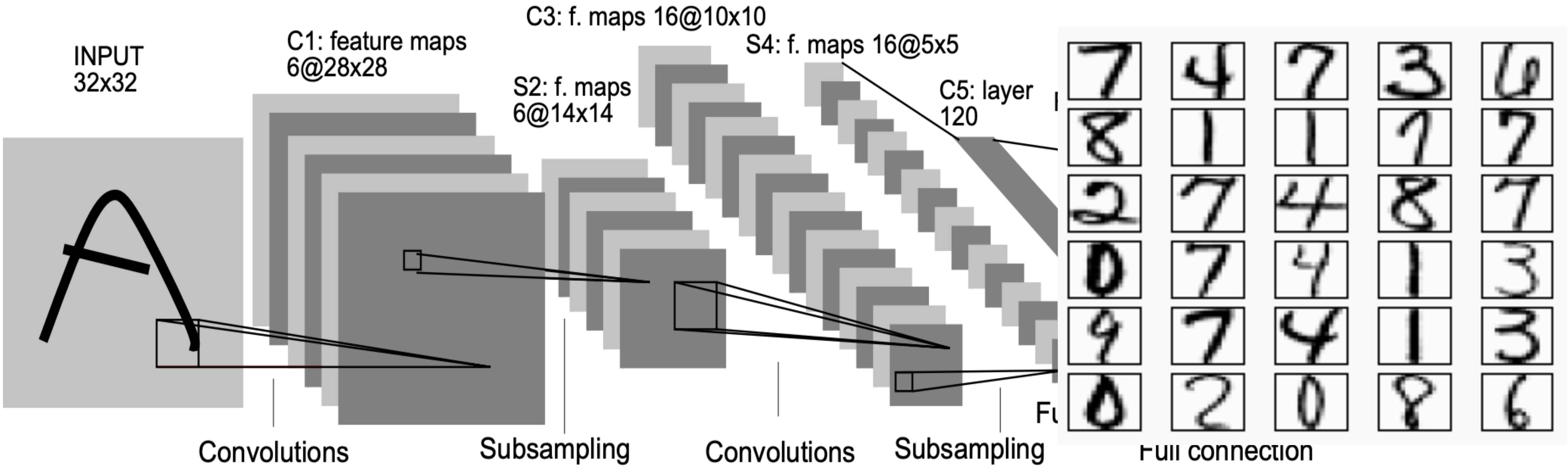
0	0
2	3

- Reduces the dimensionality of the input to subsequent layers and thus, the number of weights to be learned
  - Protects the network from (slightly) noisy inputs



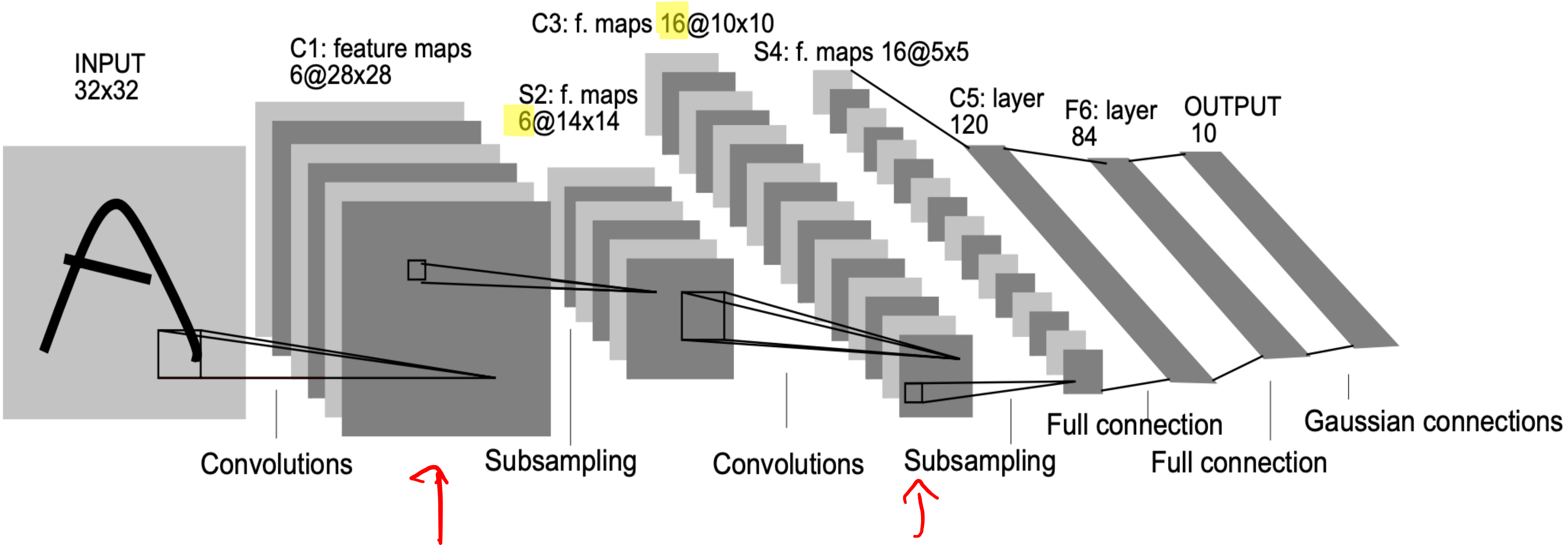
LeNet (LeCun et al., 1998)

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



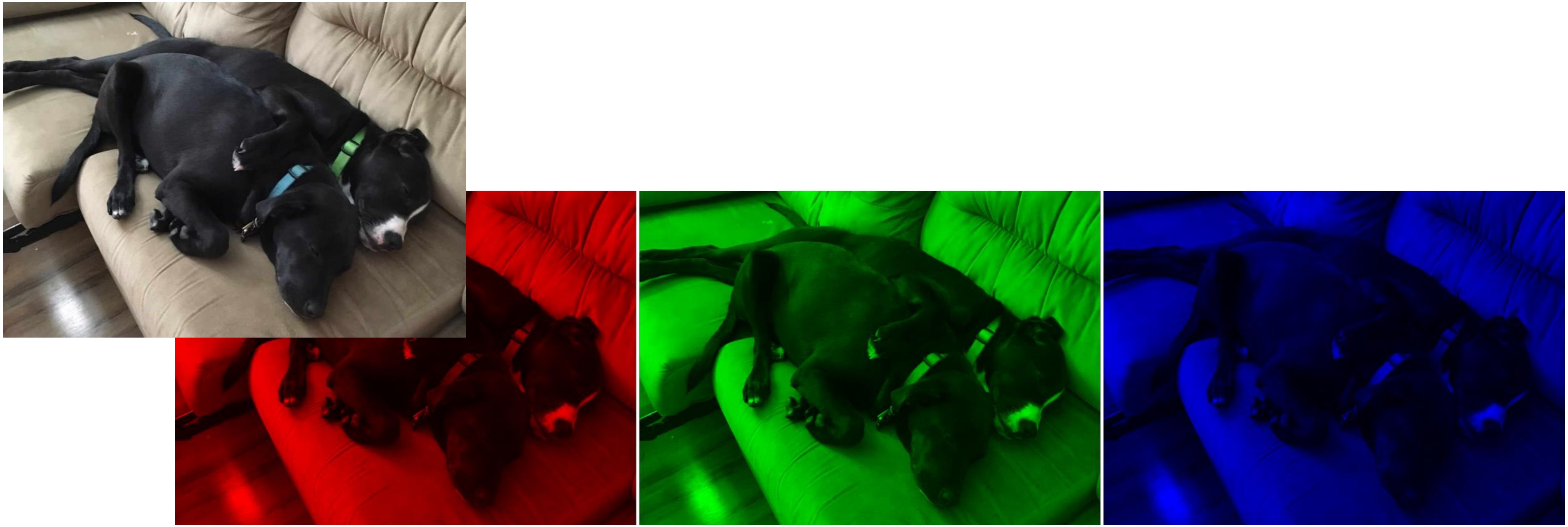
- One of the earliest, most famous deep learning models – achieved remarkable performance at handwritten digit recognition (< 1% test error rate on MNIST)
- Used sigmoid (or logistic) activation functions between layers and mean-pooling, both of which are pretty uncommon in modern architectures

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

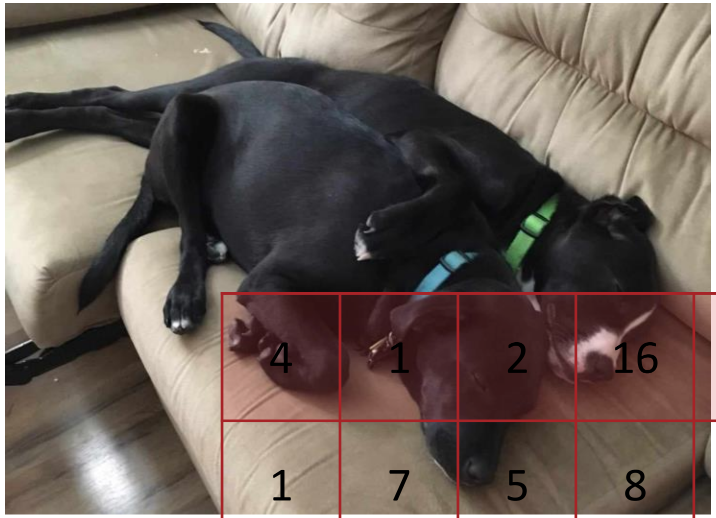


Wait how did we go from 6 to 16?

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



Channels



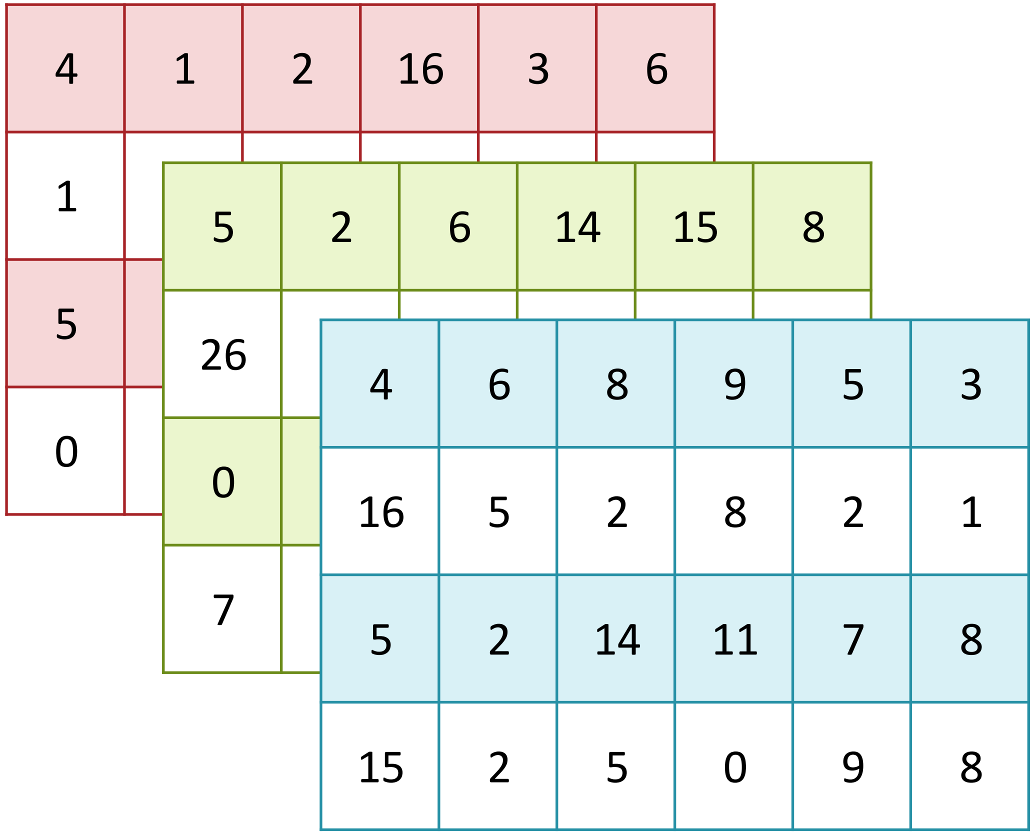
4	1	2	16	3	6
1	7	5	8	19	27
5	2	5	12	17	8
0	4	9	9	6	11

5	2	6	14	15	8
26	3	6	8	4	9
0	15	24	6	1	8
7	4	9	5	24	17

4	6	8	9	5	3
16	5	2	8	2	1
5	2	14	11	7	8
15	2	5	0	9	8

- An image can be represented as the sum of red, green and blue pixel intensities
- Each color corresponds to a *channel*

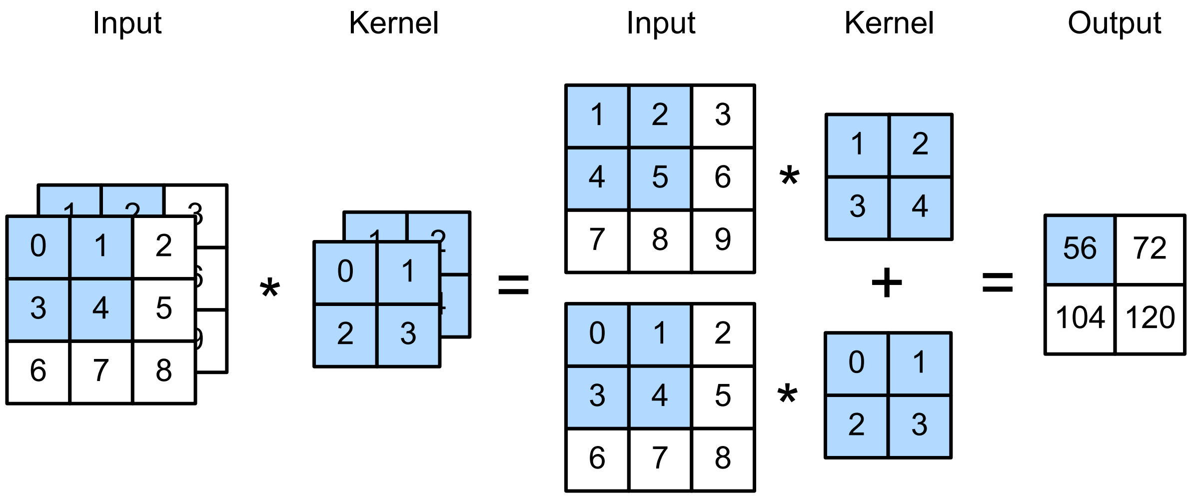
Example: $3 \times 4 \times 6$ tensor



- An image can be represented as the sum of red, green and blue pixel intensities
- Each color corresponds to a *channel*

Convolutions on Multiple Input Channels

- Given multiple input channels, we can specify a filter for each one and sum the results to get a 2-D output tensor

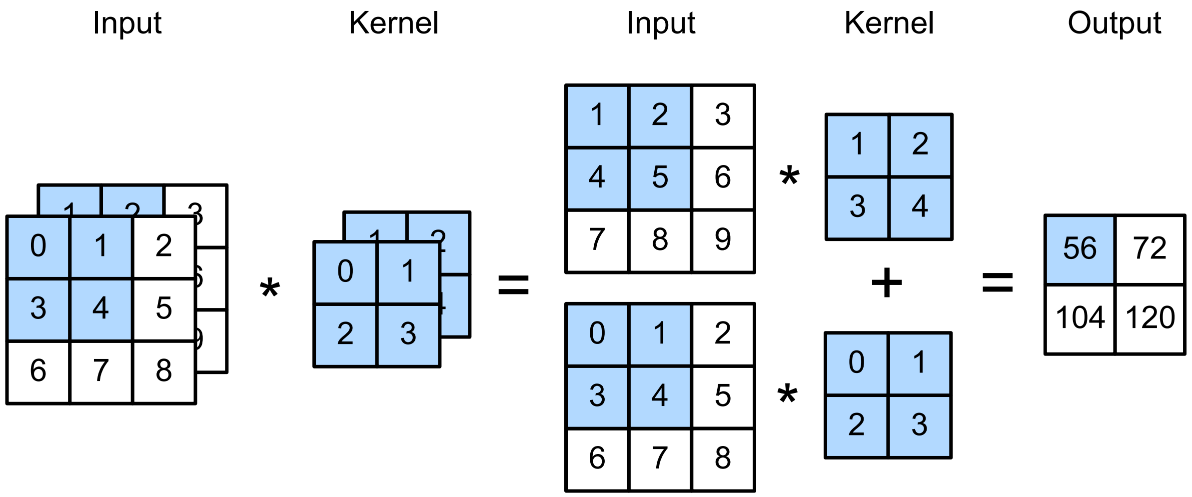


- For c channels and $h \times w$ filters, we have $chw + c$ learnable parameters (each filter has a bias term)

Source: http://preview.d2l.ai/d2l-en/master/chapter_convolutional-neural-networks/channels.html

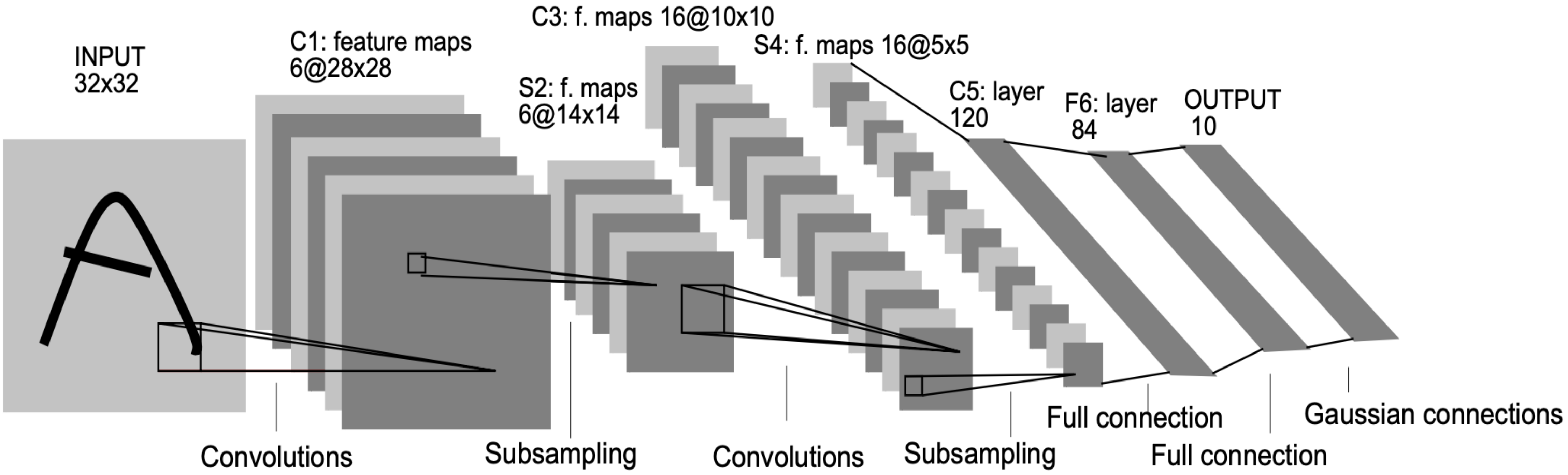
Convolutions on Multiple Input Channels

- Given multiple input channels, we can specify a filter for each one and sum the results to get a 2-D output tensor



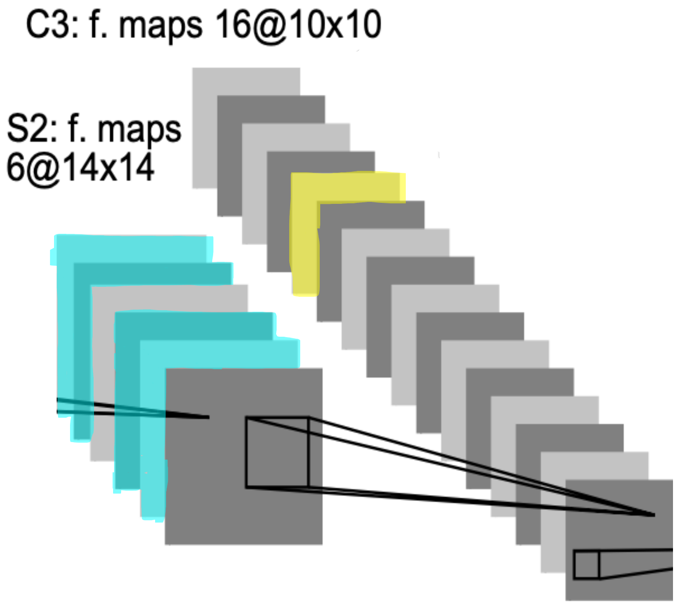
- Questions:
  1. Why might we want a different filter for each input?
  2. Why do we combine them together into a single output channel?

Source: http://preview.d2l.ai/d2l-en/master/chapter_convolutional-neural-networks/channels.html



- Channels in hidden layers correspond to different macro-features, which we might want to manipulate differently → one filter per channel

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



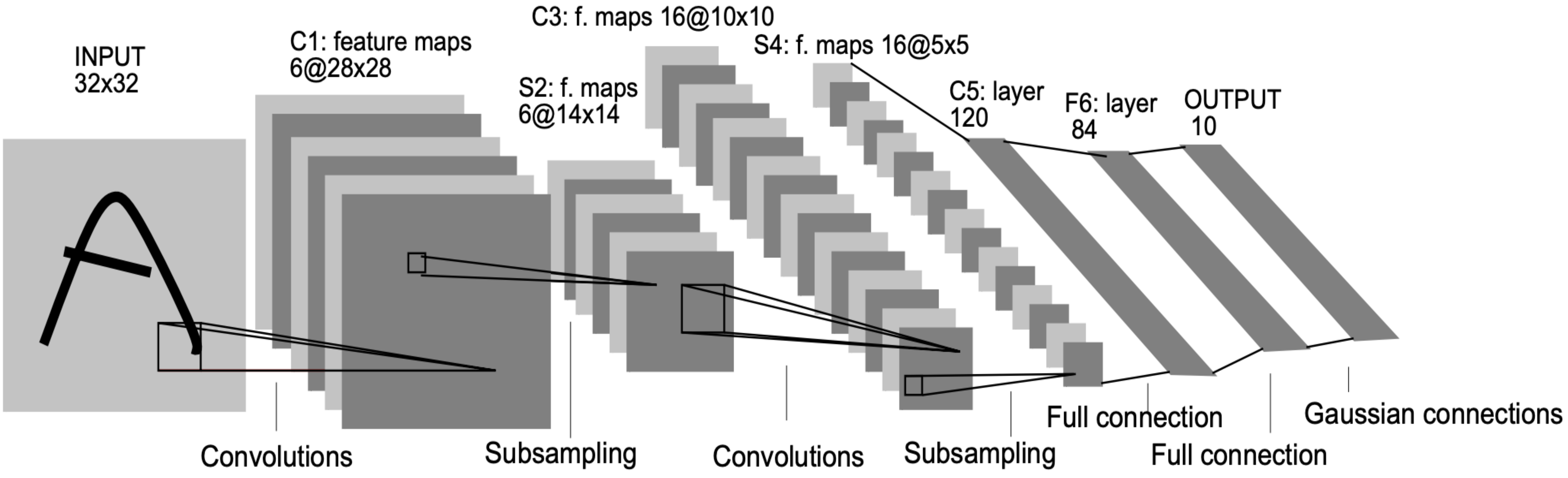
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

- We can combine these macro-features into a new, interesting, “higher-level” feature
  - But we don’t always need to combine all of them!
  - Different combinations → multiple output channels
  - Common architecture: more output channels and smaller outputs in deeper layers

Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf



Alright, so what kind of stuff can we actually do with this thing?

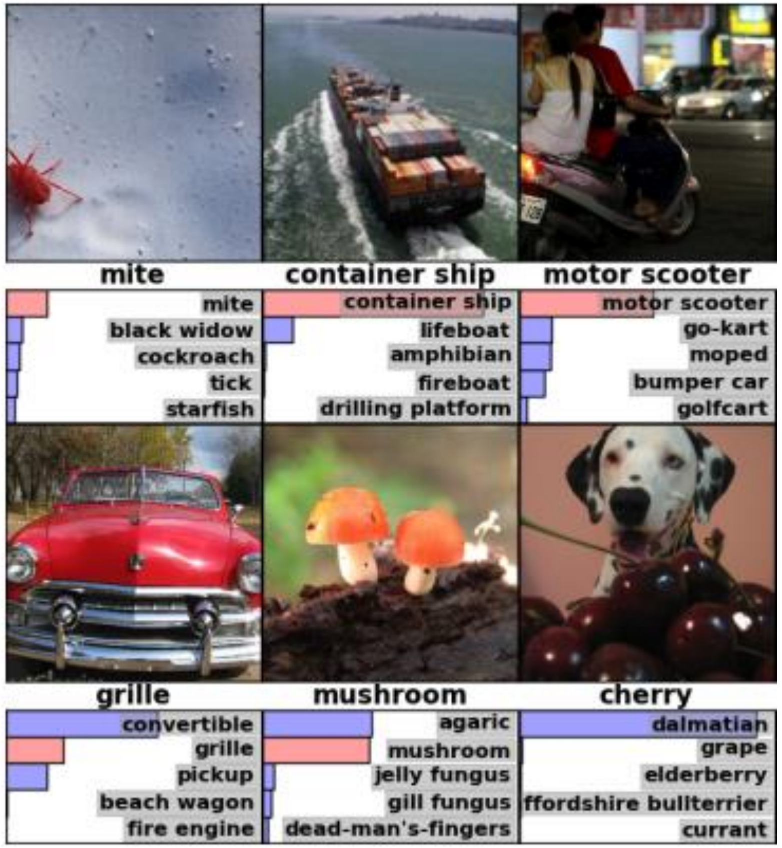
Source: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Image Captioning
- Image Generation

Common Tasks in Computer Vision

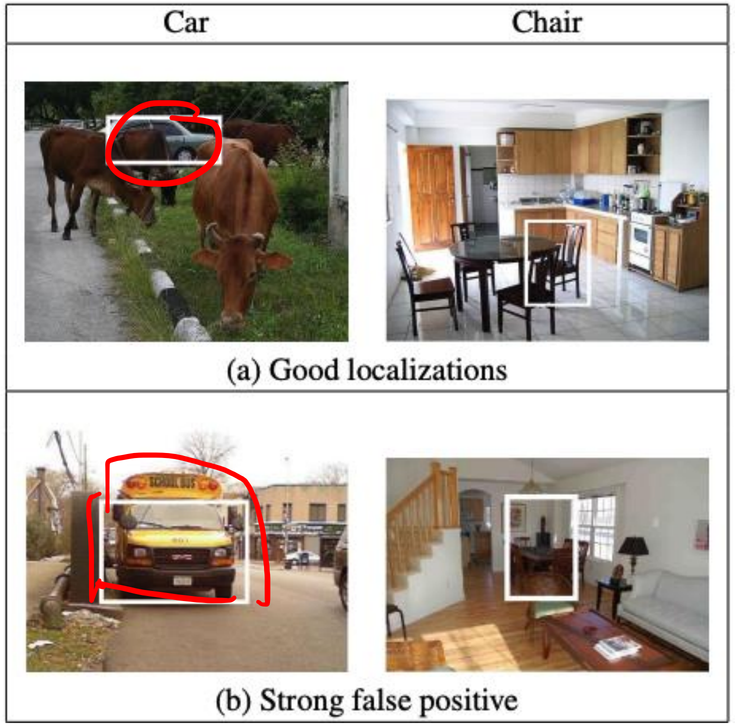
- **Image Classification**
- Object Localization
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Image Captioning
- Image Generation



Source: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Common Tasks in Computer Vision

- Image Classification
- **Object Localization**
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Image Captioning
- Image Generation



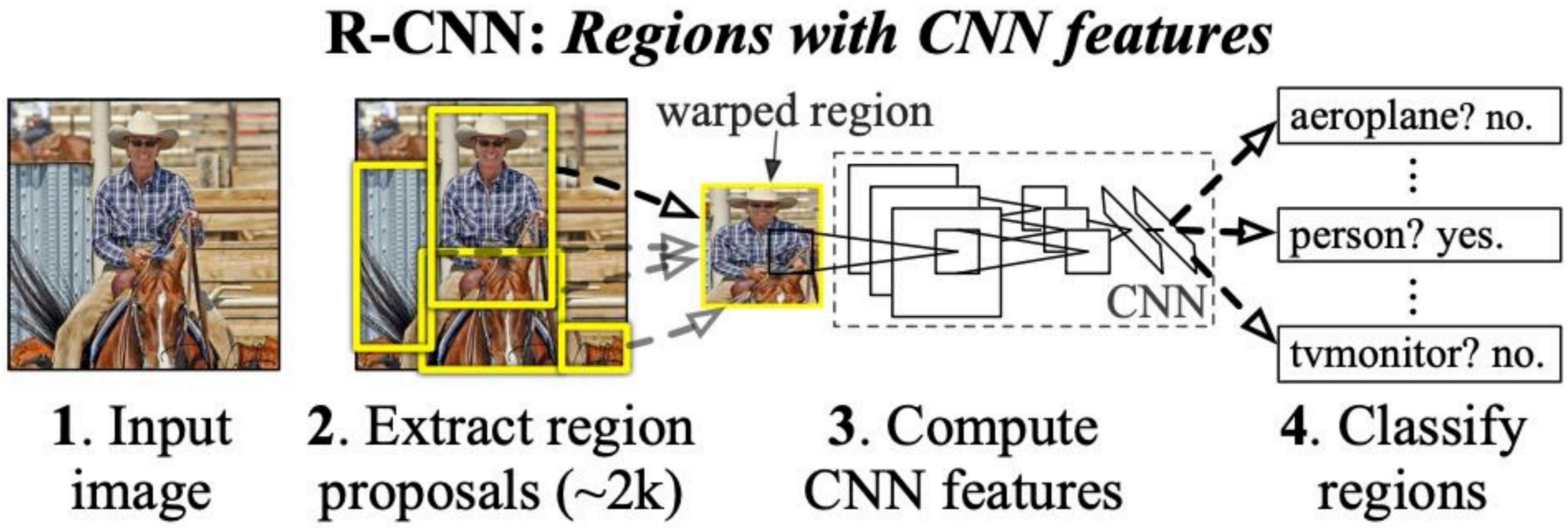
- Given an image, predict a single label and a bounding box, represented as position (x, y) and height/width (h, w).

Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5459257

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- **Object Detection**

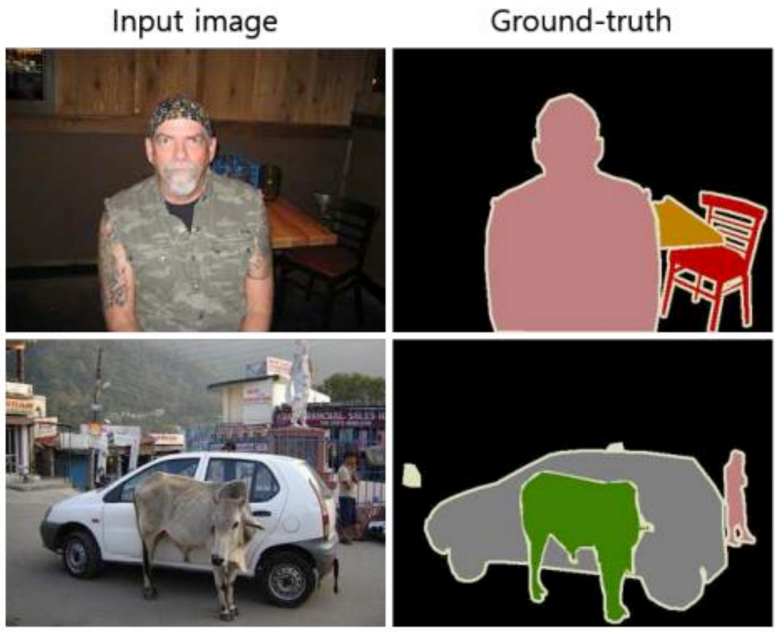
- Given an image, for each object predict a bounding box and a label, l: (x, y, w, h, l)



Source: https://openaccess.thecvf.com/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- **Semantic Segmentation**
- Instance Segmentation
- Image Captioning
- Image Generation

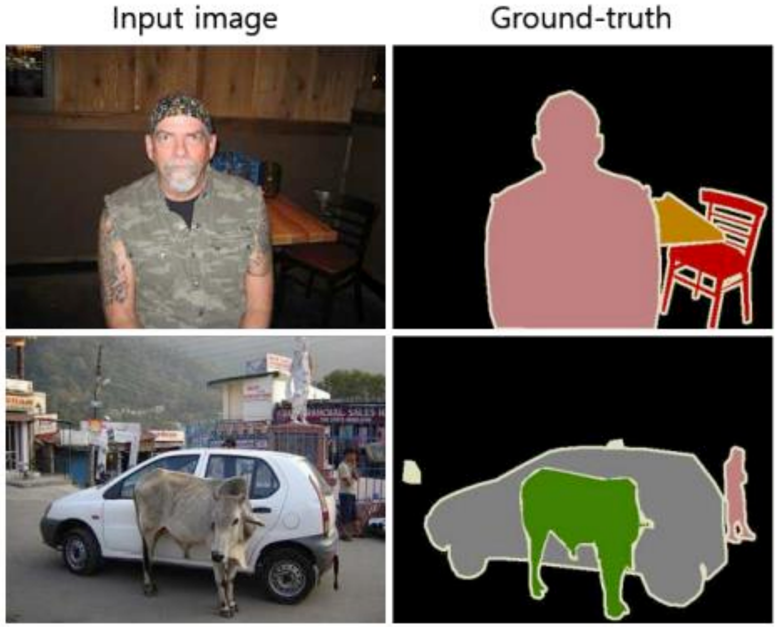


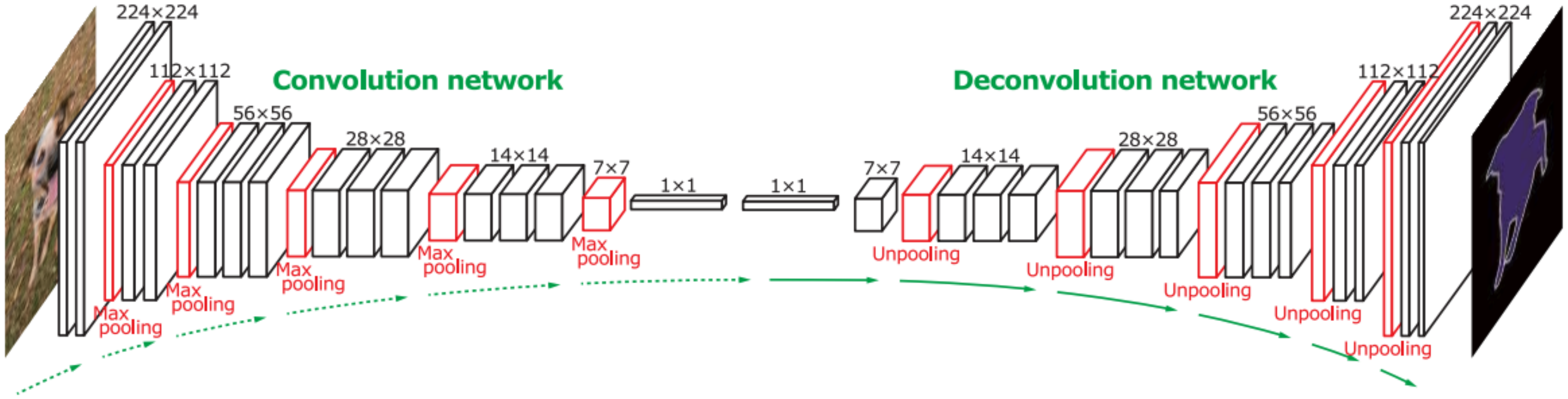
- Given an image, predict a label for every pixel in the image

Source: https://openaccess.thecvf.com/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- **Semantic Segmentation**

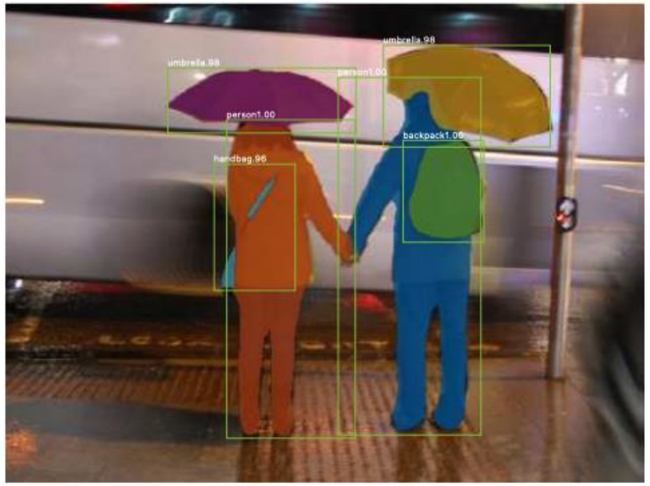




Source: https://openaccess.thecvf.com/content_iccv_2015/papers/Noh_Learning_Deconvolution_Network_ICCV_2015_paper.pdf

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- Semantic Segmentation
- **Instance Segmentation**
- Image Captioning
- Image Generation

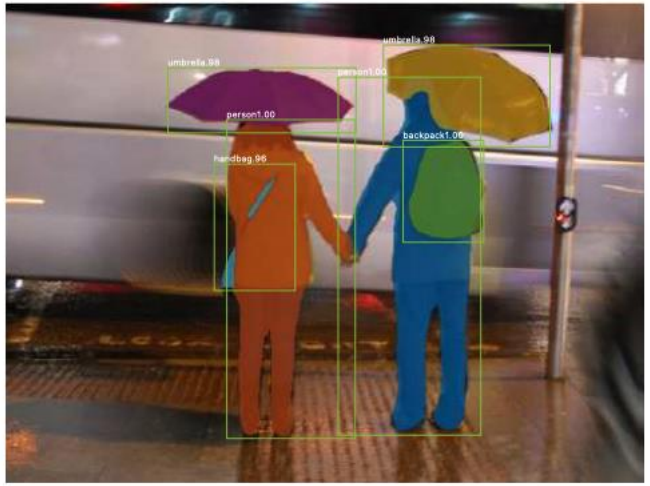


- Predict per-pixel labels as in semantic segmentation, but differentiate between different instances of the same label e.g., given two people, one should be labeled **person-1** and one should be labeled **person-2**

Source: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- Semantic Segmentation
- **Instance Segmentation**
- Image Captioning
- Image Generation



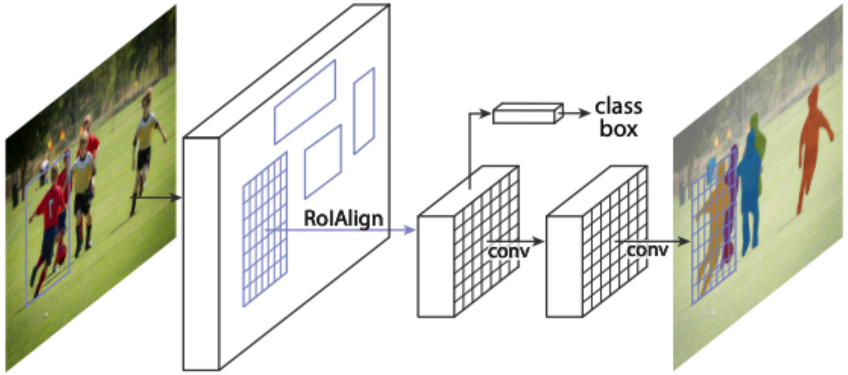
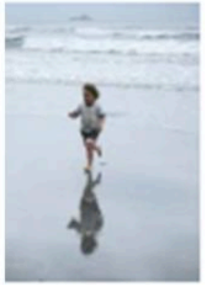


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Source: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- **Image Captioning**
- Image Generation



Ground Truth Caption: A little boy runs away from the approaching waves of the ocean.

Generated Caption: A young boy is running on the beach.

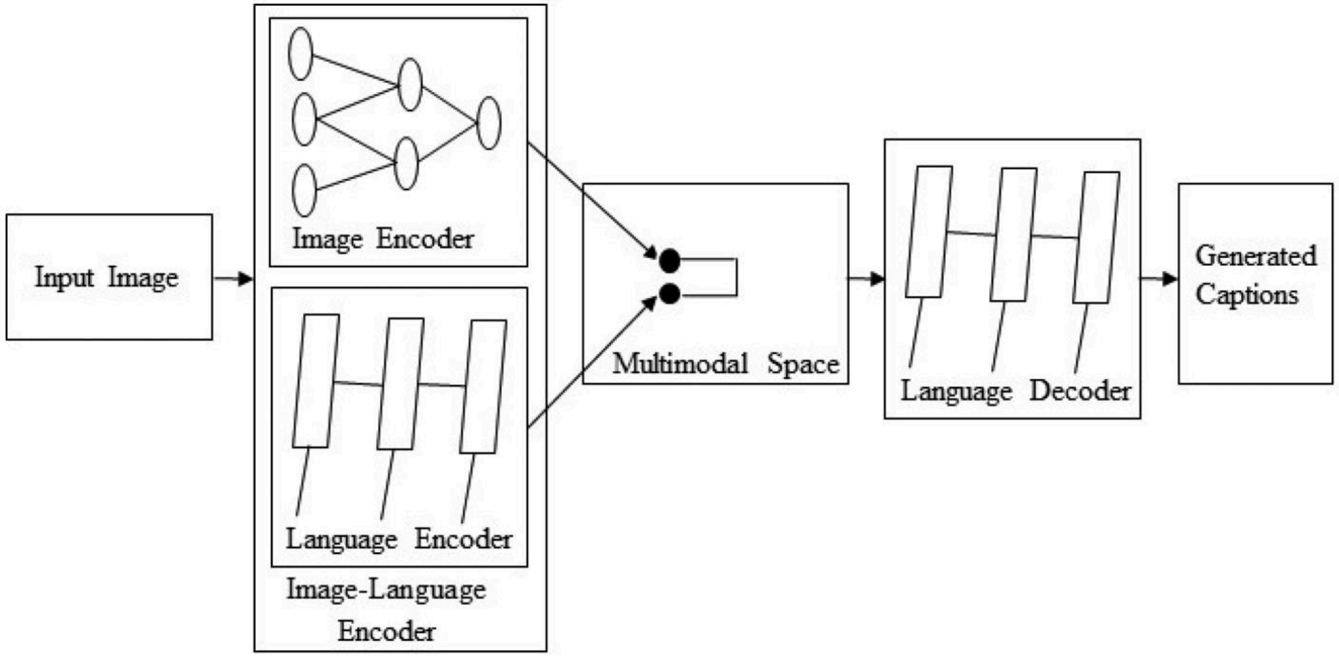
Ground Truth Caption: A brunette girl wearing sunglasses and a yellow shirt.

Generated Caption: A woman in a black shirt and sunglasses smiles.

- Take an image as input, and generate a sentence describing it as output
  - *Dense captioning* generates one description per bounding box

Source: https://dl.acm.org/doi/pdf/10.1145/3295748

Common Tasks in Computer Vision



- Instance Segmentation
- **Image Captioning**
- Image Generation

- Typical architectures will combine a CNN and an RNN-like language model

Source: https://dl.acm.org/doi/pdf/10.1145/3295748

Common Tasks in Computer Vision

- Instance Segmentation
- **Image Captioning**
- Image Generation

Table 1. An Overview of the Deep-Learning-Based Approaches for Image Captioning

Reference	Image Encoder	Language Model	Category
Kiros et al. 2014 [69]	AlexNet	LBL	MS, SL, WS, EDA
Kiros et al. 2014 [70]	AlexNet, VGGNet	1. LSTM 2. SC-NLM	MS, SL, WS, EDA
Mao et al. 2014 [95]	AlexNet	RNN	MS, SL, WS
Karpathy et al. 2014 [66]	AlexNet	DTR	MS, SL, WS, EDA
Mao et al. 2015 [94]	AlexNet, VGGNet	RNN	MS, SL, WS
Chen et al. 2015 [23]	VGGNet	RNN	VS, SL, WS, EDA
Fang et al. 2015 [33]	AlexNet, VGGNet	MELM	VS, SL, WS, CA
Jia et al. 2015 [59]	VGGNet	LSTM	VS, SL, WS, EDA
Karpathy et al. 2015 [65]	VGGNet	RNN	MS, SL, WS, EDA
Vinyals et al. 2015 [142]	GoogLeNet	LSTM	VS, SL, WS, EDA
Xu et al. 2015 [152]	AlexNet	LSTM	VS, SL, WS, EDA, AB
Jin et al. 2015 [61]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Wu et al. 2016 [151]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Sugano et at. 2016 [129]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Mathews et al. 2016 [97]	GoogLeNet	LSTM	VS, SL, WS, EDA, SC
Wang et al. 2016 [144]	AlexNet, VGGNet	LSTM	VS, SL, WS, EDA
Johnson et al. 2016 [62]	VGGNet	LSTM	VS, SL, DC, EDA
Mao et al. 2016 [92]	VGGNet	LSTM	VS, SL, WS, EDA
Wang et al. 2016 [146]	VGGNet	LSTM	VS, SL, WS, CA
Tran et al. 2016 [135]	ResNet	MELM	VS, SL, WS, CA
Ma et al. 2016 [90]	AlexNet	LSTM	VS, SL, WS, CA
You et al. 2016 [156]	GoogLeNet	RNN	VS, SL, WS, EDA, SCB
Yang et al. 2016 [153]	VGGNet	LSTM	VS, SL, DC, EDA
Anne et al. 2016 [6]	VGGNet	LSTM	VS, SL, WS, CA, NOB
Yao et al. 2017 [155]	GoogLeNet	LSTM	VS, SL, WS, EDA, SCB
Lu et al. 2017 [88]	ResNet	LSTM	VS, SL, WS, EDA, AB
Chen et al. 2017 [21]	VGGNet, ResNet	LSTM	VS, SL, WS, EDA, AB
Gan et al. 2017 [41]	ResNet	LSTM	VS, SL, WS, CA, SCB
Pedersoli et al. 2017 [112]	VGGNet	RNN	VS, SL, WS, EDA, AB
Ren et al. 2017 [119]	VGGNet	LSTM	VS, ODL, WS, EDA
Park et al. 2017 [111]	ResNet	LSTM	VS, SL, WS, EDA, AB
Wang et al. 2017 [148]	ResNet	LSTM	VS, SL, WS, EDA
Tavakoli et al. 2017 [134]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Liu et al. 2017 [84]	VGGNet	LSTM	VS, SL, WS, EDA, AB
Gan et al. 2017 [39]	ResNet	LSTM	VS, SL, WS, EDA, SC
Dai et al. 2017 [26]	VGGNet	LSTM	VS, ODL, WS, EDA
Shetty et al. 2017 [126]	GoogLeNet	LSTM	VS, ODL, WS, EDA
Liu et al. 2017 [85]	Inception-V3	LSTM	VS, ODL, WS, EDA
Gu et al. 2017 [51]	VGGNet	1. Language CNN 2. LSTM	VS, SL, WS, EDA
Yao et al. 2017 [154]	VGGNet	LSTM	VS, SL, WS, CA, NOB

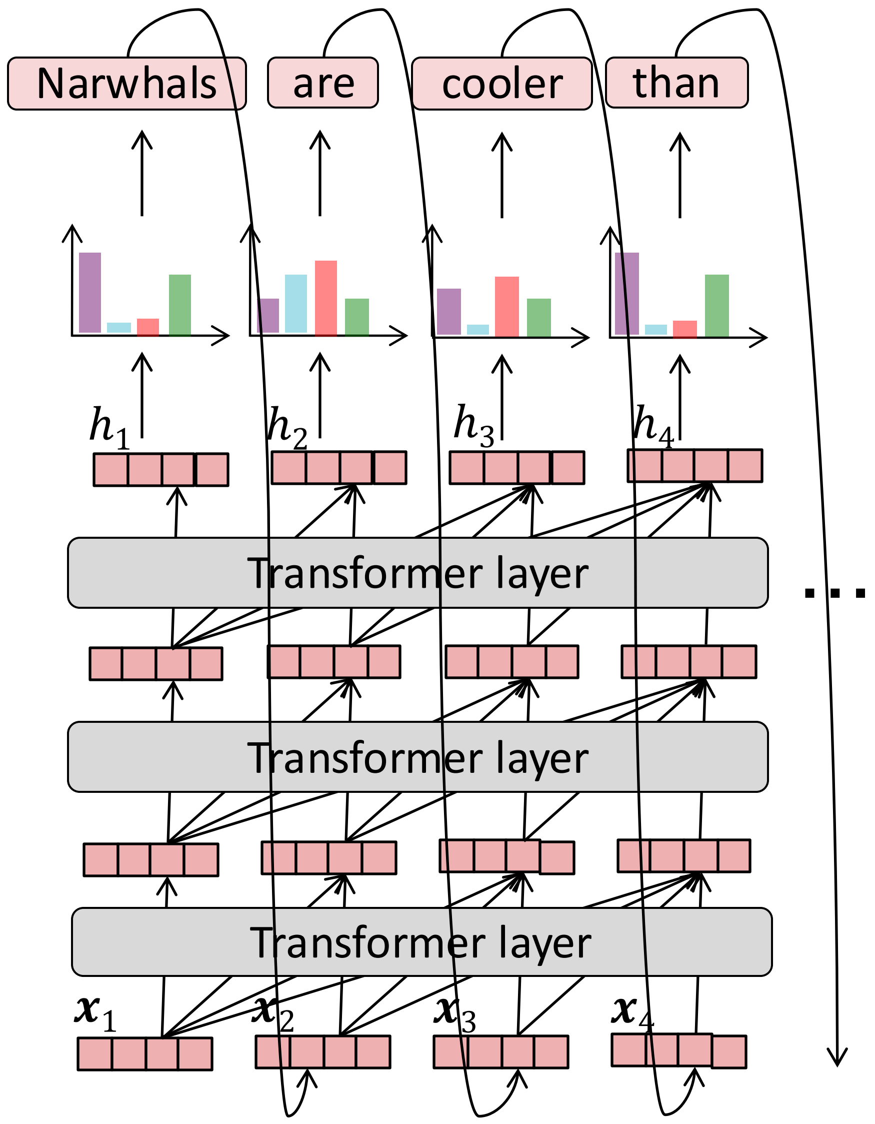
(Continued)

Source: https://dl.acm.org/doi/pdf/10.1145/3295748

Common Tasks in Computer Vision

- Image Classification
- Object Localization
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Image Captioning
- Image Generation?

Recall: Transformer Language Model

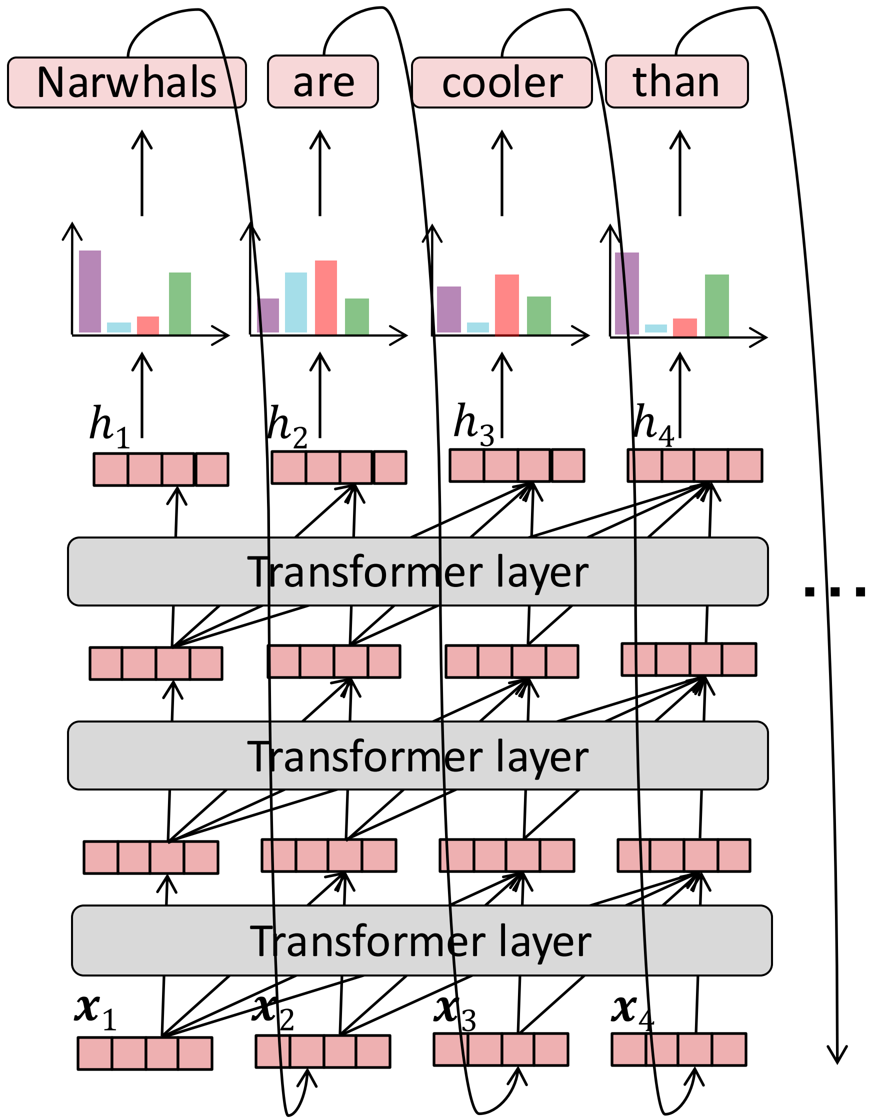


Each layer of a Transformer LM consists of:

1. causal attention
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of the current and previous timesteps in the previous layer.

Recall: Transformer Language Model a.k.a. Decoder-only Transformer



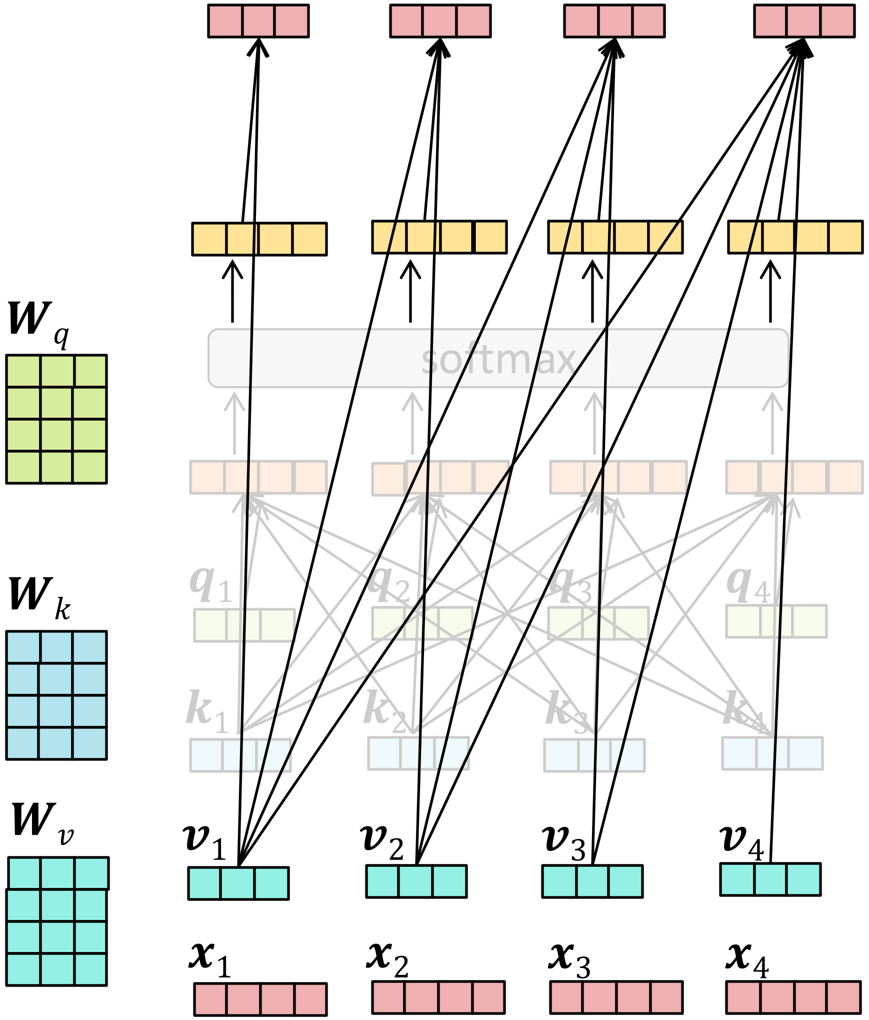
Each layer of a Transformer LM consists of:

1. **causal attention**
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of the **current and previous timesteps in the previous layer.**

Recall: Causal Attention

Idea: we can effectively delete or "mask" some of these arrows by selectively setting attention weights to 0



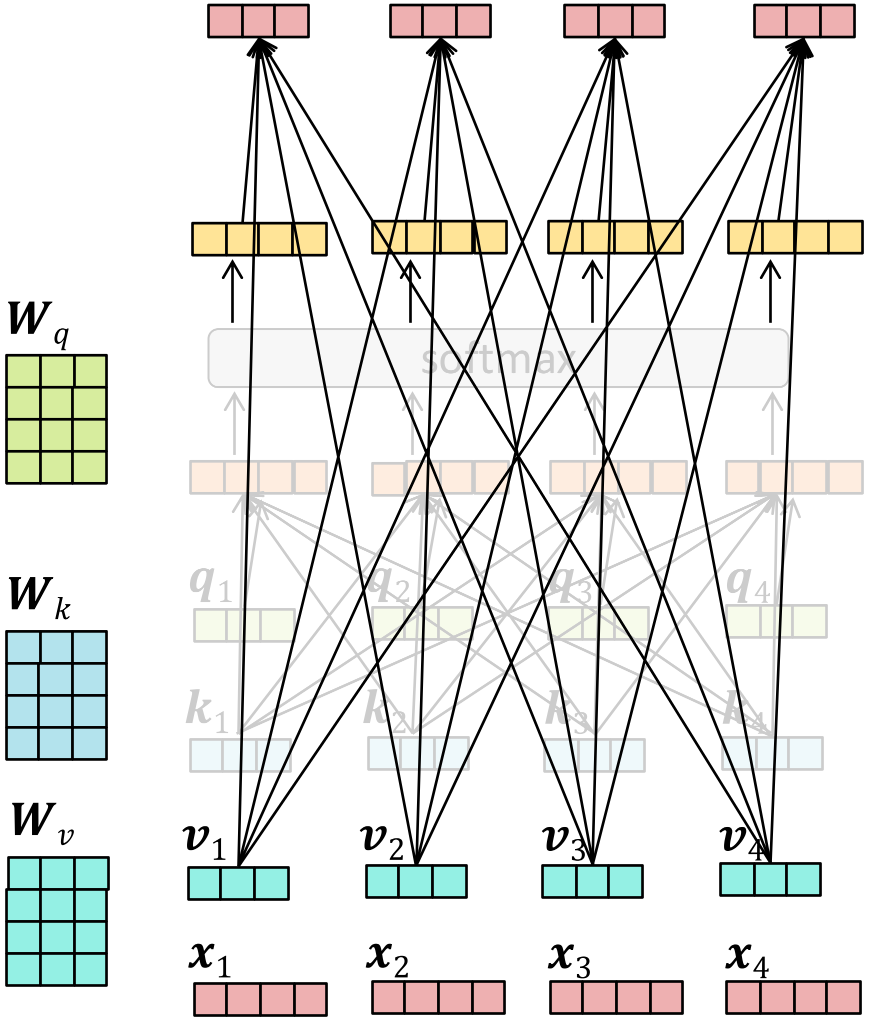
$$X' = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

$$A_{mask} = \text{softmax}(S + M)$$

$$M = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Holy cow, that's a lot of new arrows… do we *always* want/need all of those?

No…



$$X' = AV = \boxed{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

$$A = \text{softmax}(S)$$

$$S = \frac{QK^T}{\sqrt{d_k}}$$

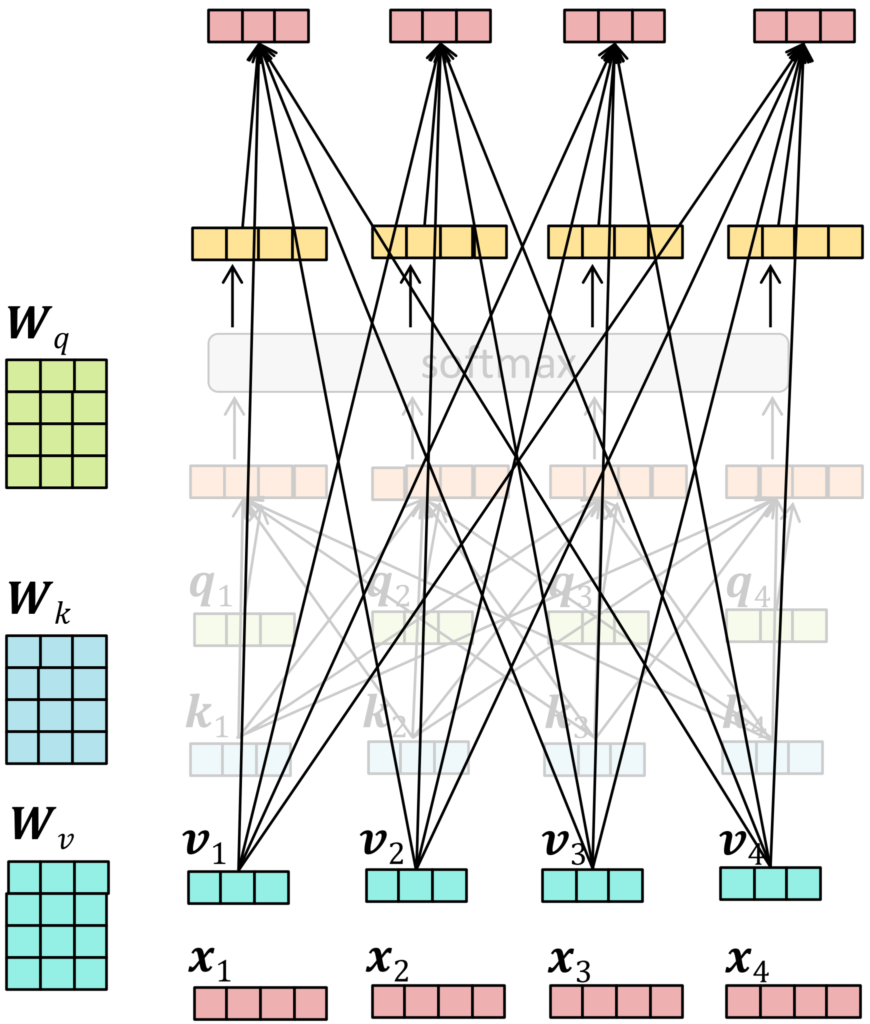
$$Q = X\boldsymbol{W}_q$$

$$K = X\boldsymbol{W}_k$$

$$V = X\boldsymbol{W}_v$$

Holy cow, that's a lot of new arrows… do we *sometimes* want/need all of those?

Yes!



$$X' = AV = \boxed{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

$$A = \text{softmax}(S)$$

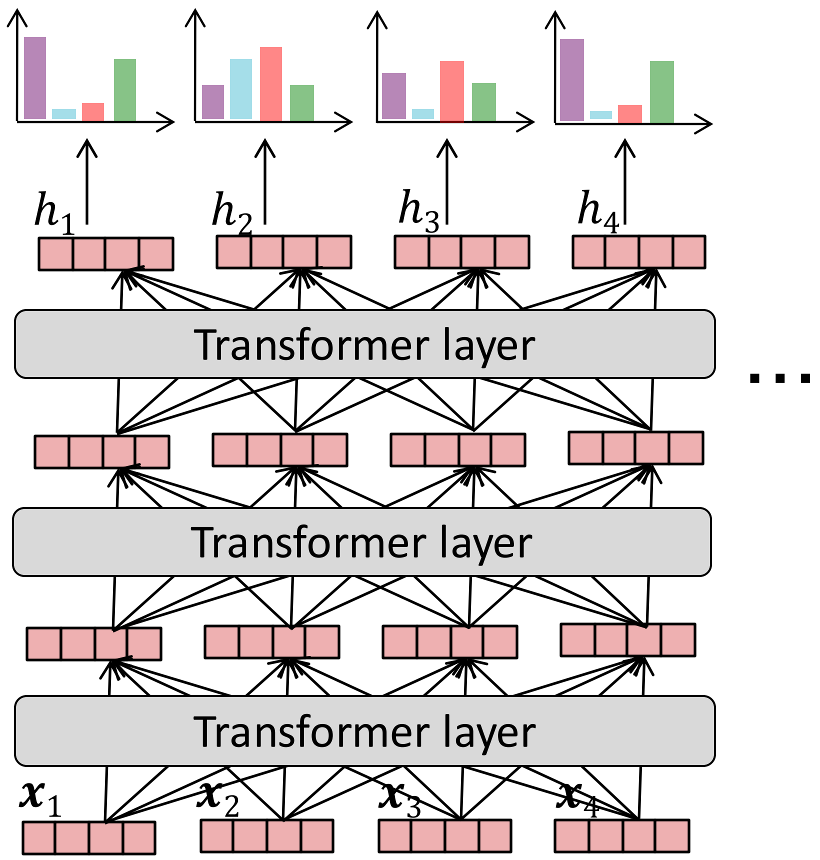
$$S = \frac{QK^T}{\sqrt{d_k}}$$

$$Q = X\boldsymbol{W}_q$$

$$K = X\boldsymbol{W}_k$$

$$V = X\boldsymbol{W}_v$$

Encoder-only Transformer

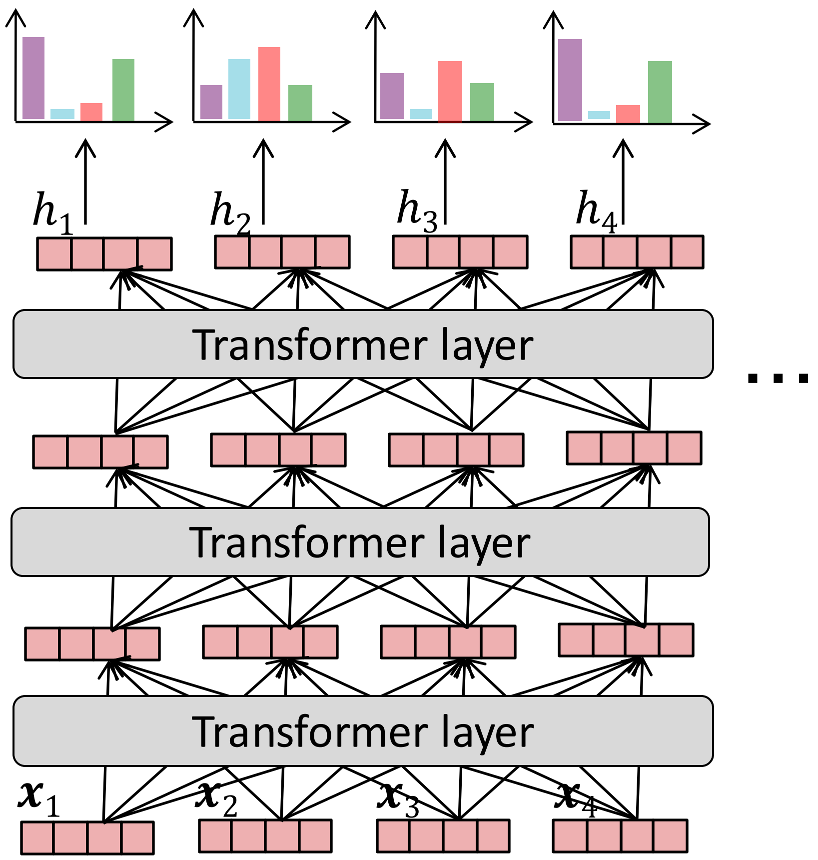


Each layer of a Transformer LM consists of:

1. **non-causal attention**
2. feed-forward neural network
3. layer normalization
4. residual connections

Each hidden vector looks back at the hidden vectors of **all timesteps in the previous layer.**

Okay, but how would we train one of these things?

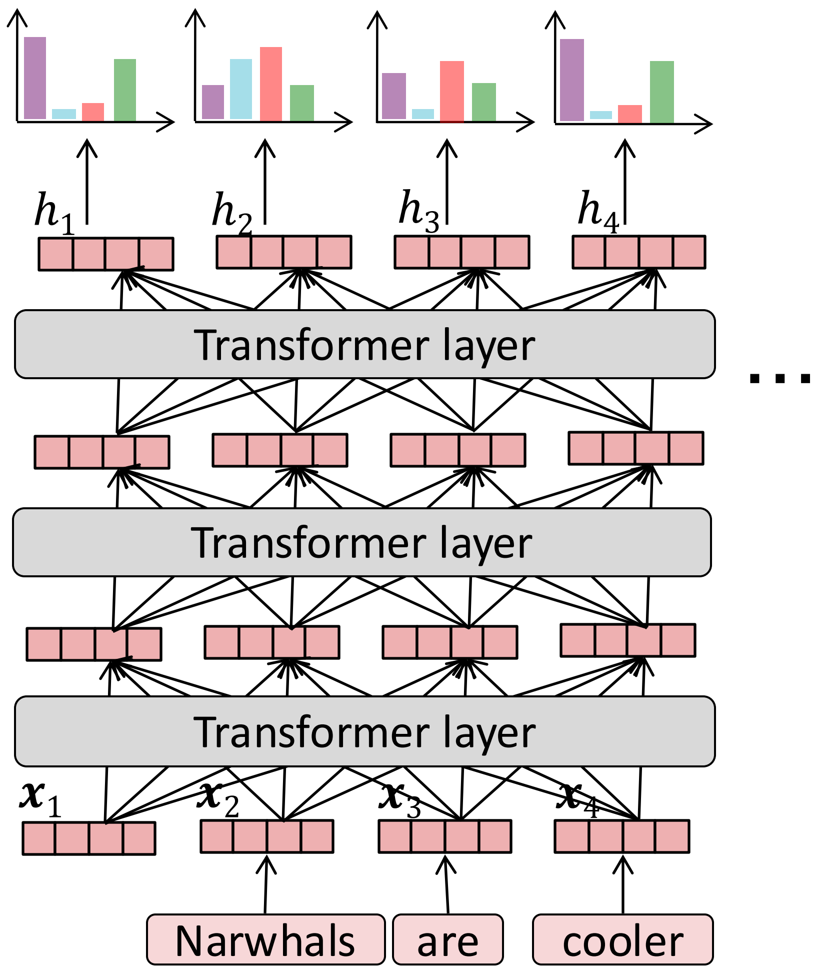


Each layer of a Transformer LM consists of:

1. **non-causal attention**
2. feed-forward neural network
3. layer normalization
4. residual connections

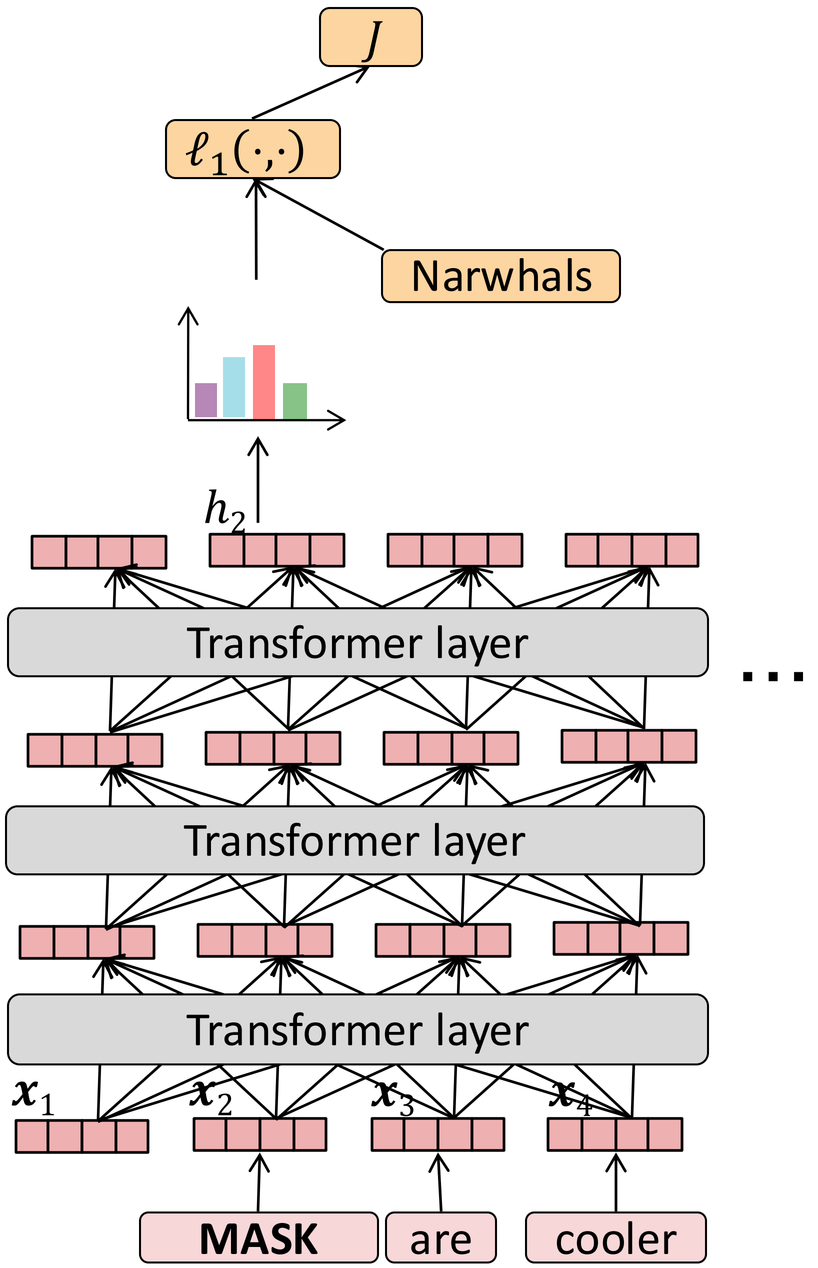
Each hidden vector looks back at the hidden vectors of **all timesteps in the previous layer.**

Masked Language Model Pre-training



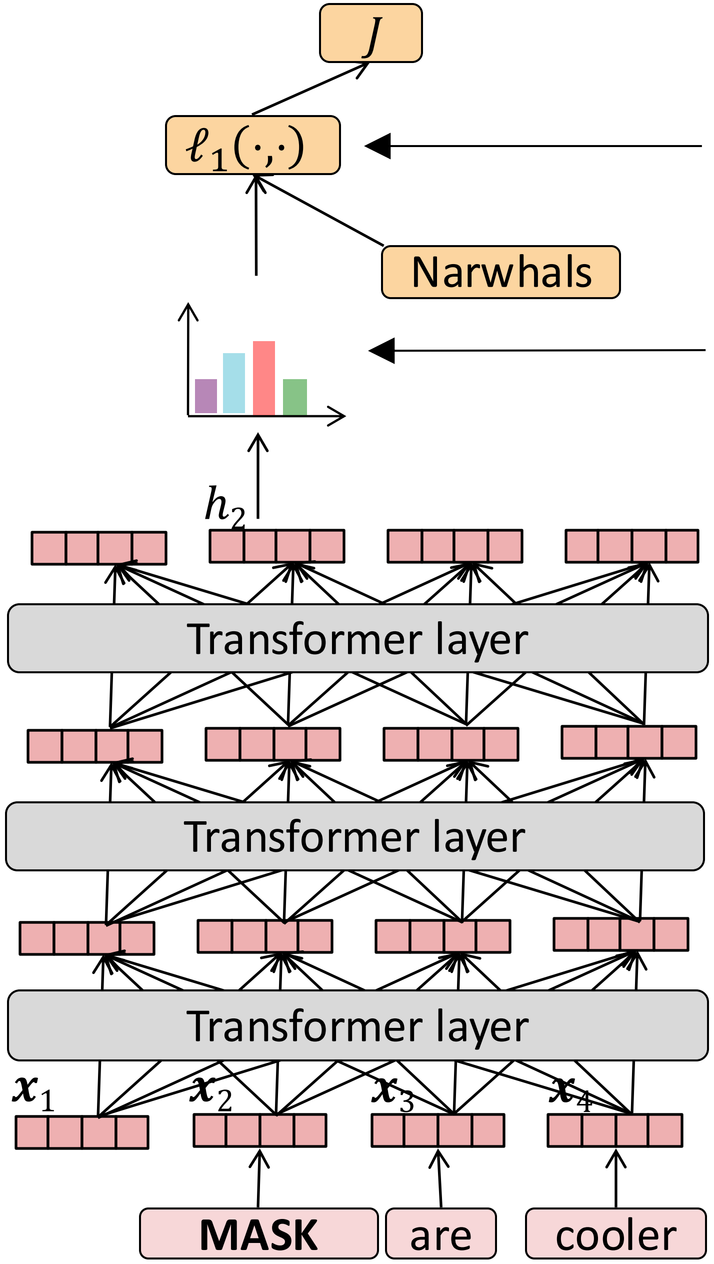
Rather than trying to predict the next token, *mask* out a few tokens in the sequence and train the model to predict the masked tokens.

Masked Language Model Pre-training



Rather than trying to predict the next token, *mask* out a few tokens in the sequence and train the model to predict the masked tokens.

Masked Language Model Pre-training



What is this loss?

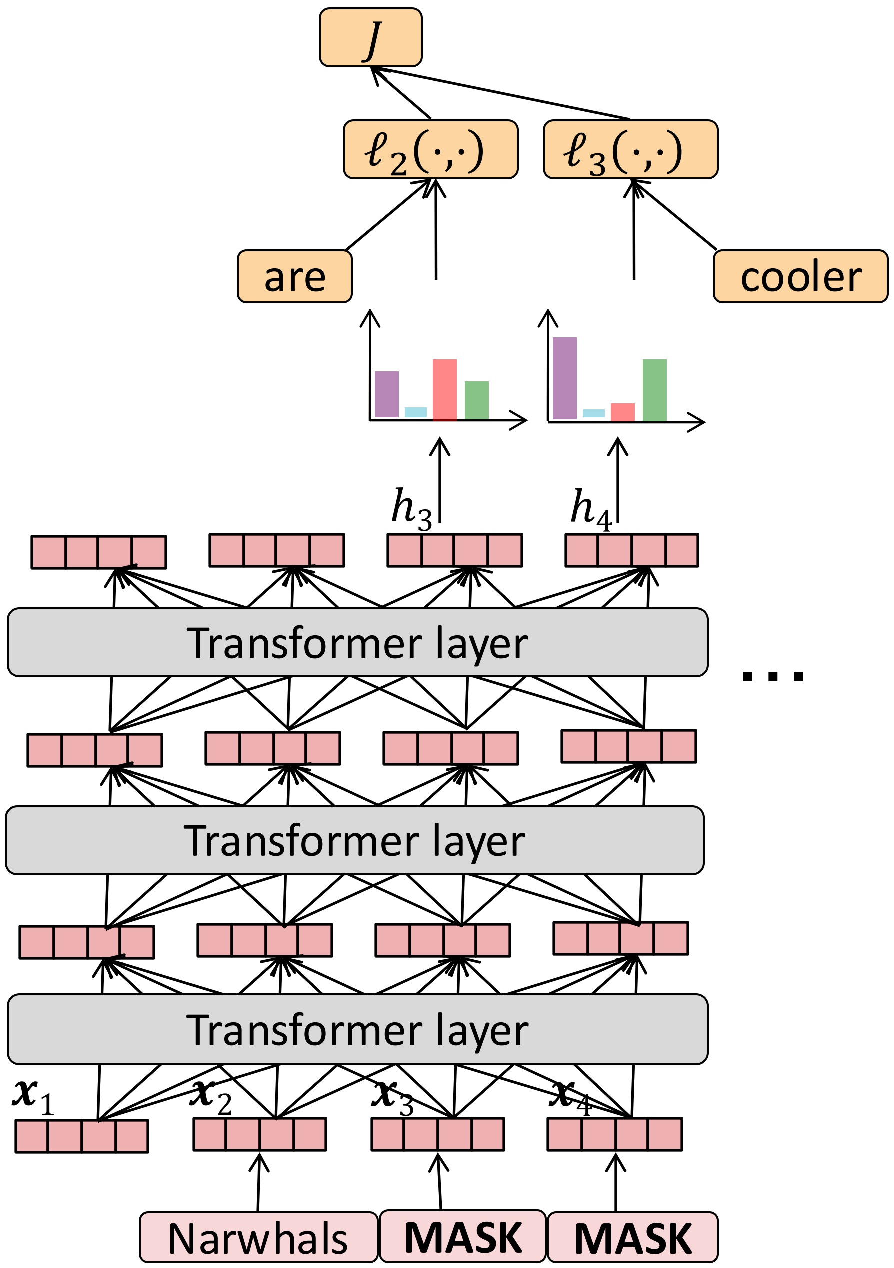
$-\log p(w_1 = \text{Narwhals} \mid w_2, w_3)$

What is this probability distribution?

$P(w_1 \mid w_2, w_3)$

…

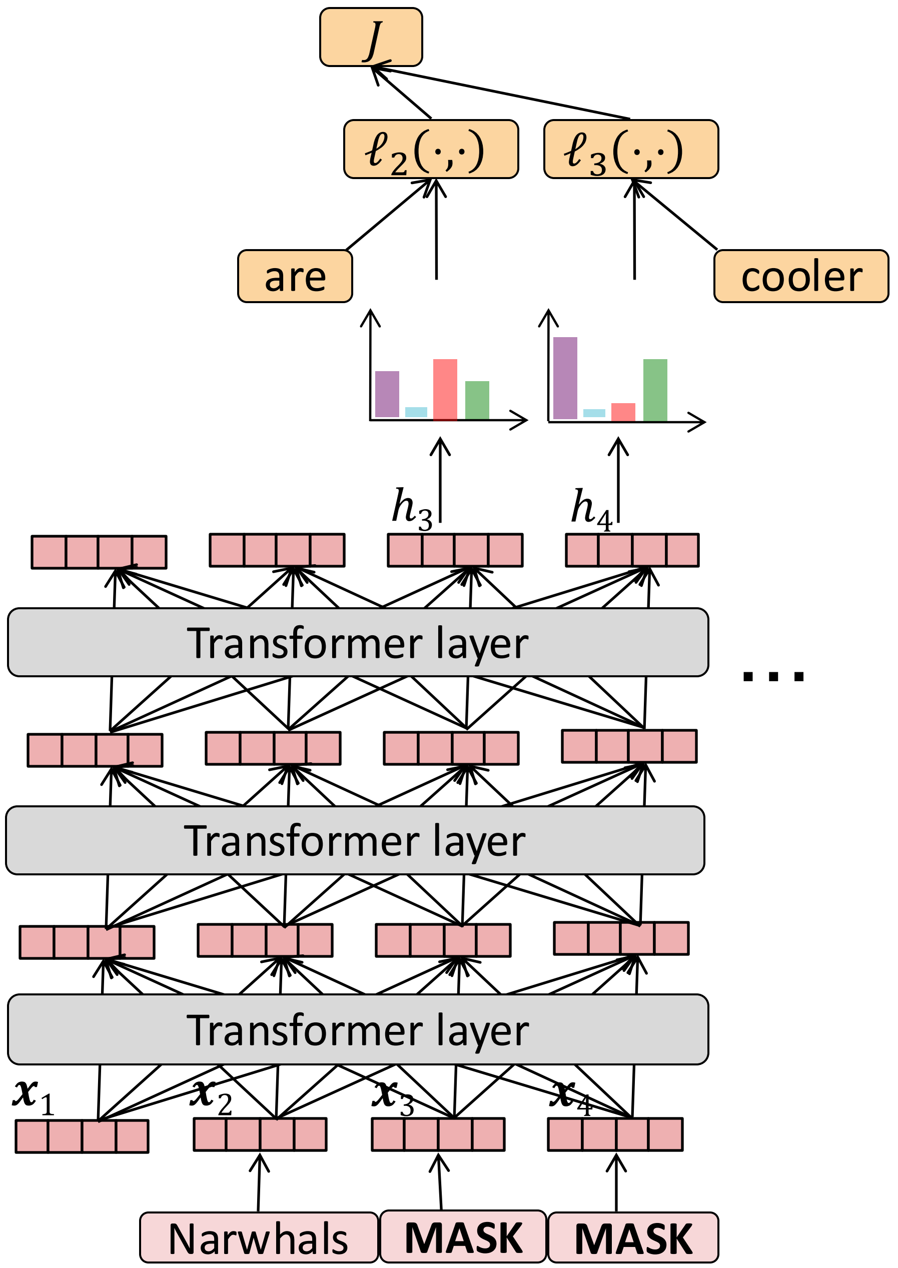
Masked Language Model Pre-training



Rather than trying to predict the next token, *mask* out a few tokens in the sequence and train the model to predict the masked tokens.

This kind of pre-training was popularized by the BERT language model

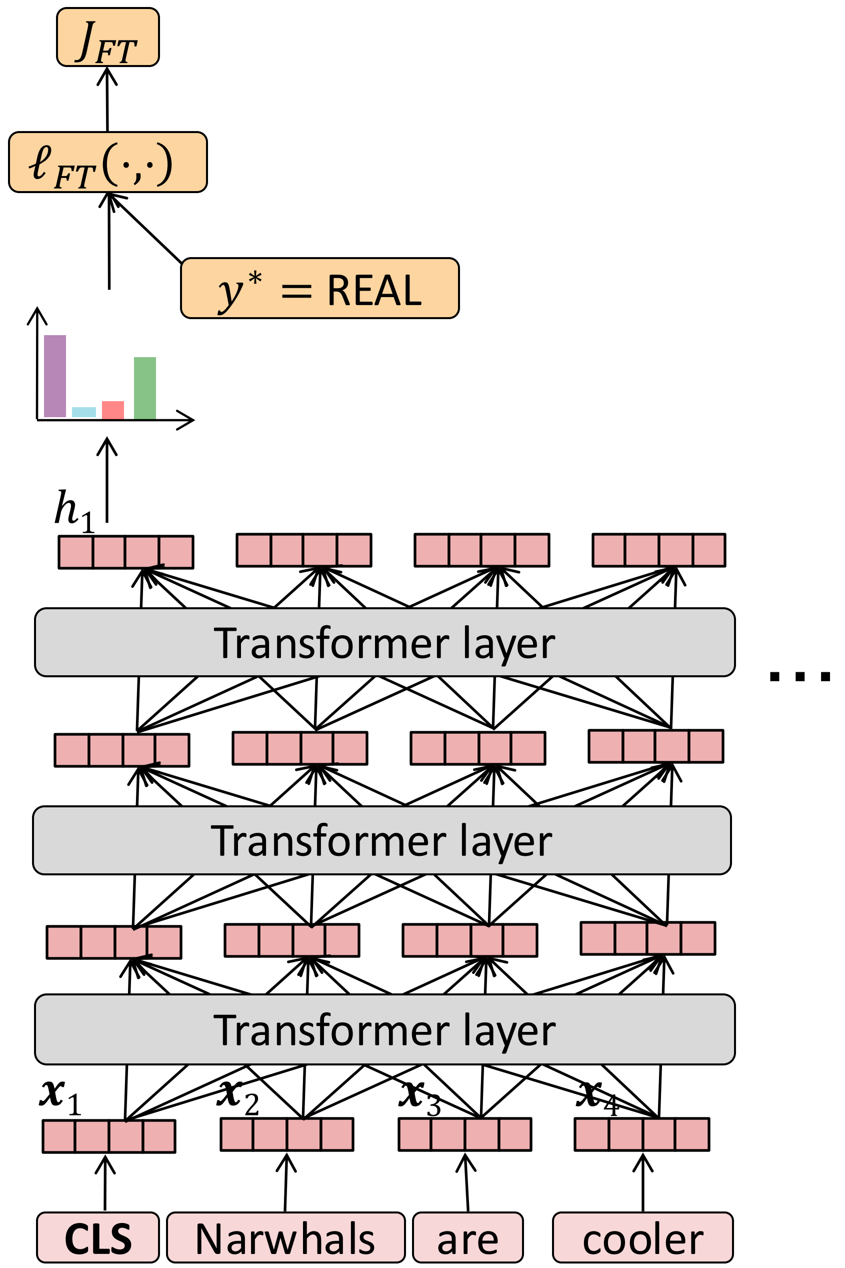
Masked Language Model Pre-training



Rather than trying to predict the next token, *mask* out a few tokens in the sequence and train the model to predict the masked tokens.

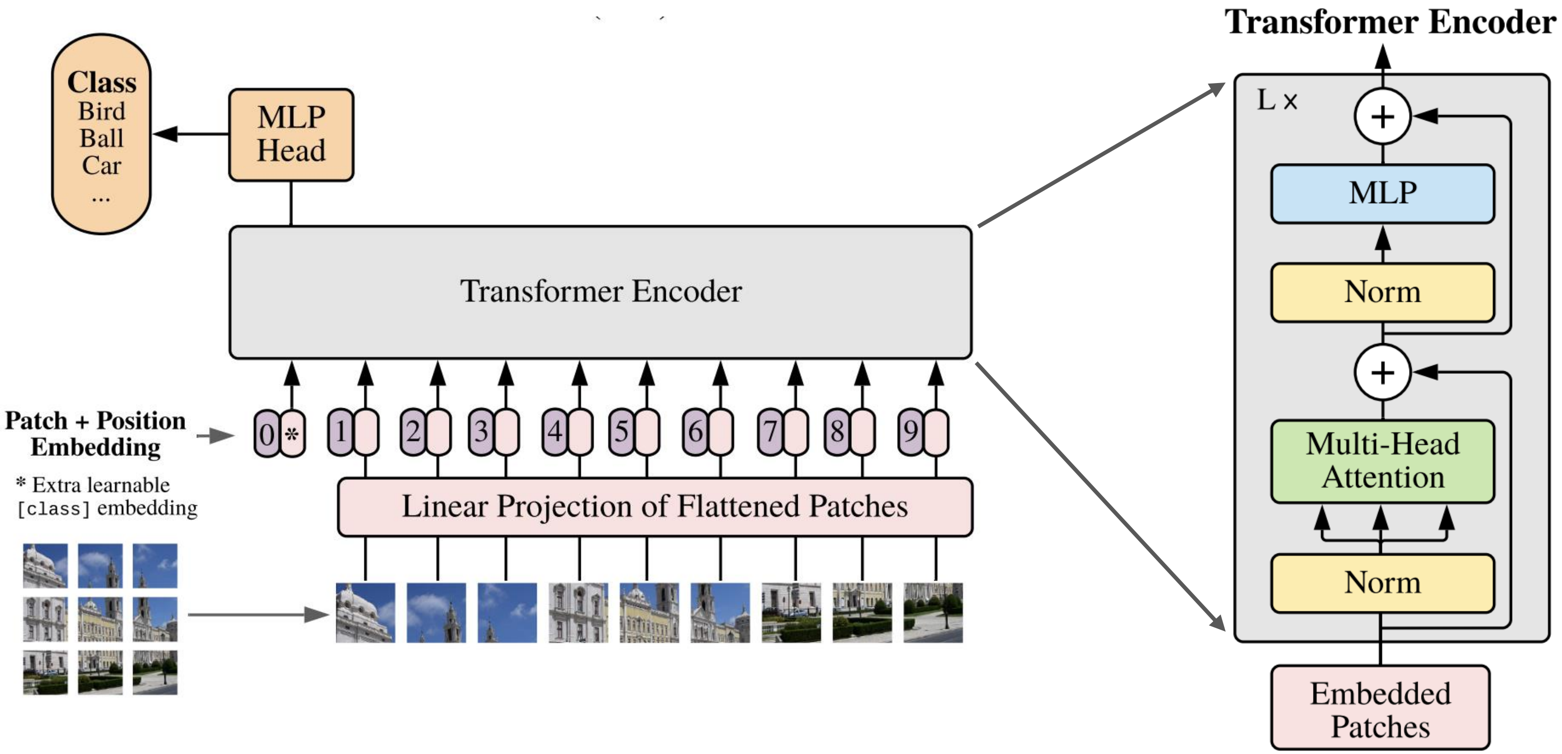
This kind of **pre-training** was popularized by the BERT language model

Supervised Fine-tuning



Prepend a special class token and *fine-tune* the (pre-trained) model to predict the label for each sequence

This model is not generative but has been shown to be a highly effective discriminator on a variety of tasks

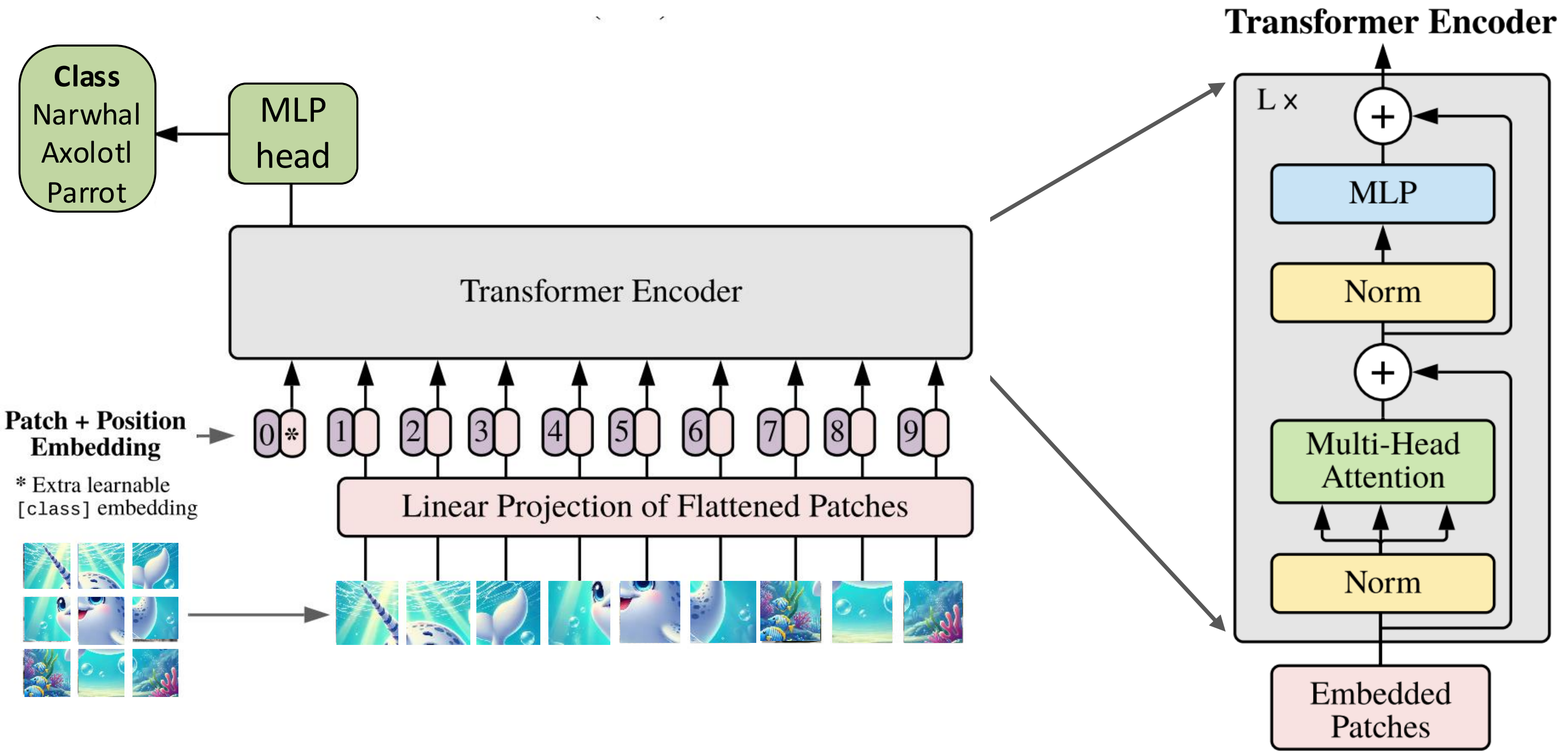


Vision Transformer (ViT)

- Instead of words as input, the inputs are $P \times P$ pixel *patches*
- Each patch is embedded linearly into a vector of size 1024
- Uses 1D positional embeddings
- Pre-trained on a large, supervised dataset (e.g., ImageNet 21K, JFT-300M)

Source: https://arxiv.org/pdf/2010.11929

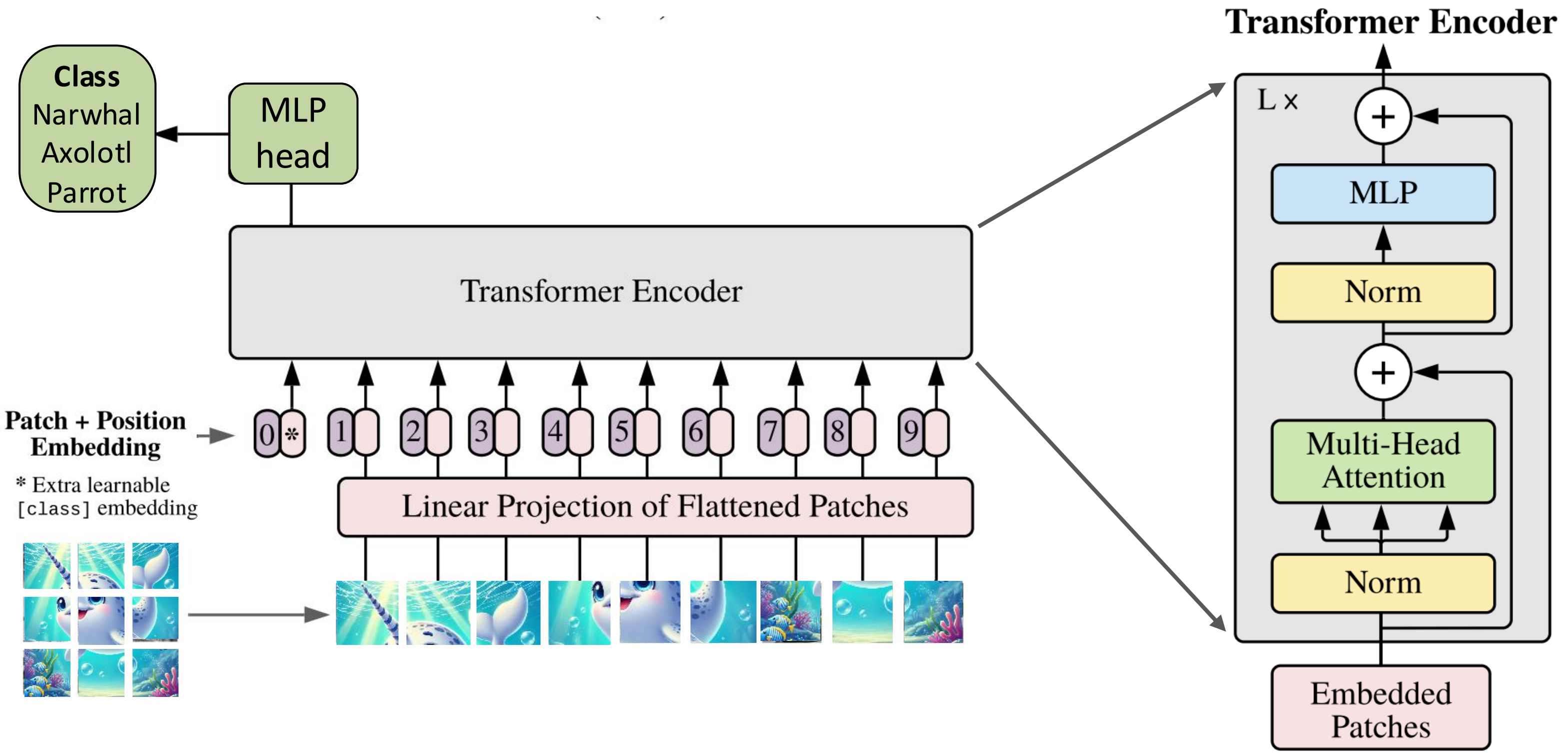
Vision Transformer (ViT)



- Instead of words as input, the inputs are $P \times P$ pixel *patches*
- Each patch is embedded linearly into a vector of size 1024
- Uses 1D positional embeddings
- Can be fine-tuned by learning a new classification head on some (small) target dataset (e.g., CIFAR-100)

Source: https://arxiv.org/pdf/2010.11929

How can a ViT learn 2D positional information from a 1D positional embedding?



- Instead of words as input, the inputs are $P \times P$ pixel *patches*
- Each patch is embedded linearly into a vector of size 1024
- **Uses 1D positional embeddings**
- Can be fine-tuned by learning a new classification head on some (small) target dataset (e.g., CIFAR-100)

Source: https://arxiv.org/pdf/2010.11929

Language Modelling: Timeline

2000
n-grams

2010
RNN LMs

2017
Transformer LMs

2018
ELMO
BERT
GPT

2019
GPT-2
RoBERTa

2020
GPT-3

2021
InstructGPT
LaMBDA

2022
Palm
ChatGPT
BLOOM

2023
Llama
GPT-4
Falcon
Mistral

Language Modelling: Timeline

- **2000**: n-grams
- **2010**: RNN LMs
- **2017**: Transformer LMs
- **2018**: ELMO, **BERT**, **GPT**
- **2019**: **GPT-2**, **RoBERTa**
- **2020**: **GPT-3**
- **2021**: **InstructGPT**, **LaMDA**
- **2022**: **Palm**, **ChatGPT**, **BLOOM**
- **2023**: **Llama**, **GPT-4**, **Falcon**, **Mistral**

Language Modelling: Timeline

- **1998**: LeNet
- **2009**: ImageNet
- **2012**: AlexNet
- **2013**: VAEs
- **2014**: VGG, R-CNN, GANs
- **2015**: Diffusion models, ResNet
- **2017**: Transformer
- **2020**: DDPM
- **2021**: **ViT**, **Dall-E**, **CLIP**
- **2022**: Dall-E 2, Imagen, Stable diffusion
- **2023**: SDXL, SDXL Turbo

Why did Transformers take so long to gain traction in computer vision?

- **1998**: LeNet
- **2009**: ImageNet
- **2012**: AlexNet
- **2013**: VAEs
- **2014**: VGG, R-CNN, GANs
- **2015**: Diffusion models, ResNet
- **2017**: Transformer
- **2020**: DDPM
- **2021**: **ViT**, **Dall-E**, **CLIP**
- **2022**: Dall-E 2, Imagen, Stable diffusion
- **2023**: SDXL, SDXL Turbo

Why did Transformers take so long to gain traction in computer vision?

AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]
[*]equal technical contribution, [†]equal advising
Google Research, Brain Team
{adosovitskiy, neilhoulsby}@google.com

When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.

However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias. Our Vision Transformer (ViT) attains excellent results when pre-trained at sufficient scale and transferred to tasks with fewer datapoints. When pre-trained on the public ImageNet-21k dataset or the in-house JFT-300M dataset, ViT approaches or beats state of the art on multiple image recognition benchmarks. In particular, the best model reaches the accuracy of 88.55% on ImageNet, 90.72% on ImageNet-ReaL, 94.55% on CIFAR-100, and 77.63% on the VTAB suite of 19 tasks.

Source: https://arxiv.org/pdf/2010.11929

Why did Transformers take so long to gain traction in computer vision?

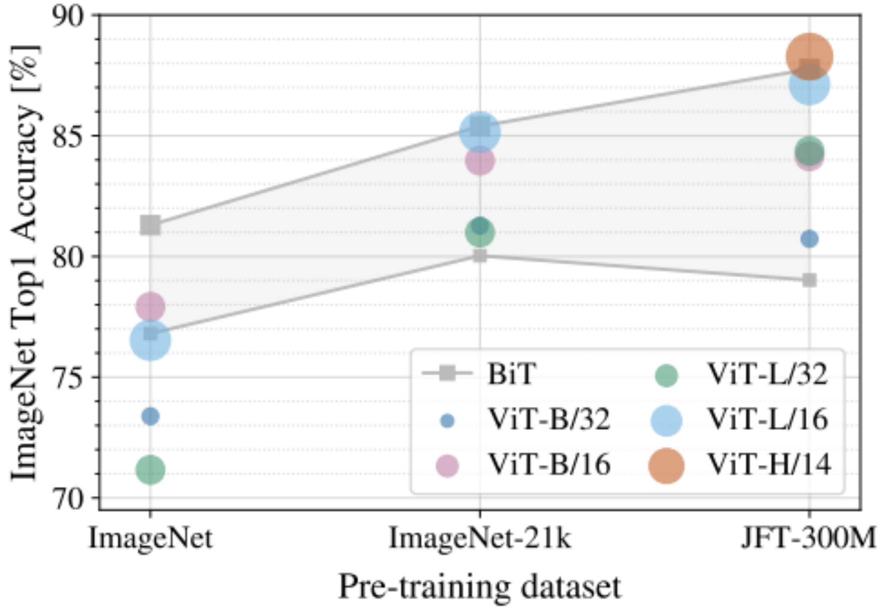


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

Source: https://arxiv.org/pdf/2010.11929

Wait, hang on: is this even a generative model?

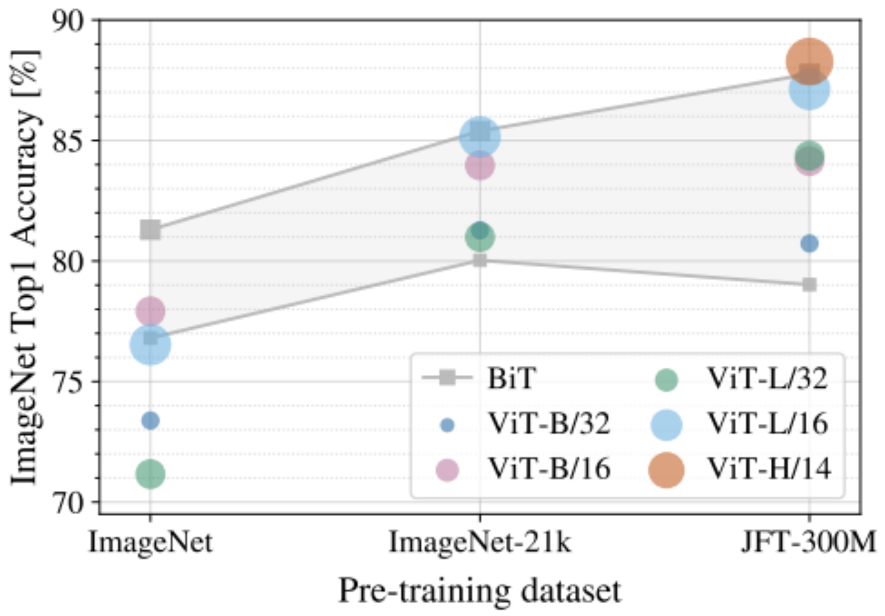


Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

Source: https://arxiv.org/pdf/2010.11929