CMB Workshop

Intro to AI and Machine Learning

Anastasios Kyrillidis
Rice CS

Overview

- Introduction to ML/AI

Overview

- Introduction to ML/AI
- Introduction to neural networks

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)

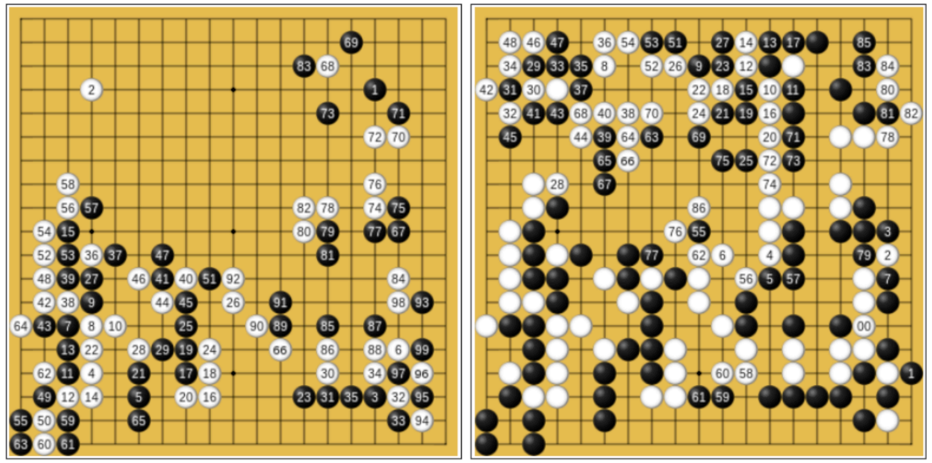
Where are we today?

GPT-3 Example

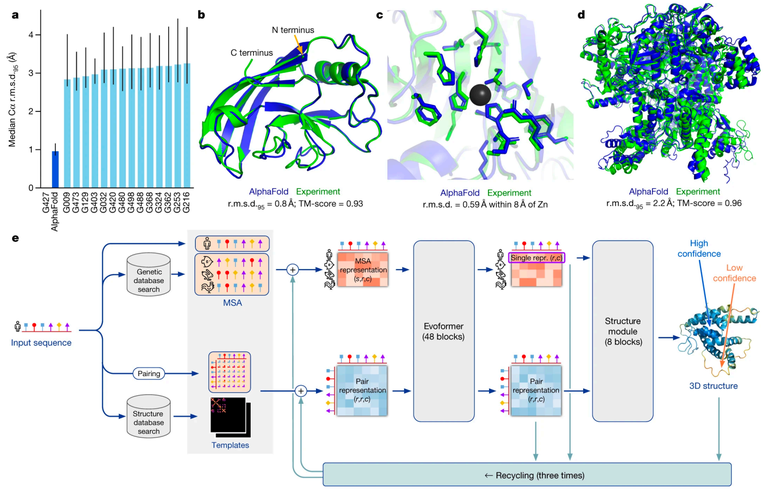
Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Example: GANS

Game Playing



Structural Biology



Underlying principles in ML: Supervised Learning

Foundation of modern ML: supervised learning

- One of the most fundamental problems in data science
  - Given a bunch of (x, y) pairs
  - Goal: learn how to predict value of y from x

Underlying principles in ML: Supervised Learning

Foundation of modern ML: supervised learning

- One of the most fundamental problems in data science
  - Given a bunch of (x, y) pairs
  - Goal: learn how to predict value of y from x
  - Classically, x is a feature vector
  - Example: x is $\langle \texttt{age}, \texttt{income}, \texttt{gender} \rangle$
  - Called "supervised" because have examples of correct labeling

Underlying principles in ML: Supervised Learning

Foundation of modern ML: supervised learning

- One of the most fundamental problems in data science
  - Given a bunch of (x, y) pairs
  - Goal: learn how to predict value of y from x
  - Classically, x is a feature vector
  - Example: x is $\langle \texttt{age}, \texttt{income}, \texttt{gender} \rangle$
  - Called "supervised" because have examples of correct labeling

Any task where we want to mimic existing labeling:

- Given an image, tell if has a cat or not
- Given a text EMR, label "breast cancer" or not
- Given a document (email) in a court case, figure which subjects relevant to

Underlying principles in ML: Supervised Learning

Foundation of modern ML: supervised learning

- One of the most fundamental problems in data science
  - Given a bunch of (x, y) pairs
  - Goal: learn how to predict value of y from x
  - Classically, x is a feature vector
  - Example: x is $\langle \texttt{age}, \texttt{income}, \texttt{gender} \rangle$
  - Called "supervised" because have examples of correct labeling

Any task where we want to mimic existing labeling:

- Given an image, tell if has a cat or not
- Given a text EMR, label "breast cancer" or not
- Given a document (email) in a court case, figure which subjects relevant to
- Given information about a patient surgery, predict death
- Given head trauma patient info, predict ICP crisis
- Given an set of surgical vital signs, label "good surgery" or not
- Many others!

Underlying principles in ML: Type of outputs

- Classification:
  - Outcome to predict is in $\{+1, -1\}$ ("yes" or "no")
  - Ex: Given a text EMR, label "breast cancer" or not

(e.g., Logistic regression)

Underlying principles in ML: Type of outputs

- Classification:
  - Outcome to predict is in $\{+1, -1\}$ ("yes" or "no")
  - Ex: Given a text EMR, label "breast cancer" or not

(e.g., Logistic regression)

- Regression:
  - Outcome to predict is a real number
  - Ex: Given an ad, predict number of clickthrus per hour

(e.g., Linear regression)

Underlying principles in ML: Training vs Testing

- Proper methodology
  - Break data into three subsets:

Underlying principles in ML: Training vs Testing

- Proper methodology
  - Break data into three subsets:
  - Training, validation, testing

Underlying principles in ML: Training vs Testing

- Proper methodology
  - Break data into three subsets:
  - Training, validation, testing
  - Training: used to learn the model (min the loss)

Underlying principles in ML: Training vs Testing

- Proper methodology
  - Break data into three subsets:
  - Training, validation, testing
  - Training: used to learn the model (min the loss)
  - Validation: used to see if the model is OK

Underlying principles in ML: Training vs Testing

- Proper methodology
  - Break data into three subsets:
  - Training, validation, testing
  - Training: used to learn the model (min the loss)
  - Validation: used to see if the model is OK
  - Maybe params are wrong... or bad features, or wrong model
  - Use testing to predict accuracy in deployment
  - You often find test accuracy much lower than validation (over-fitting)

Underlying principles in ML: Model design

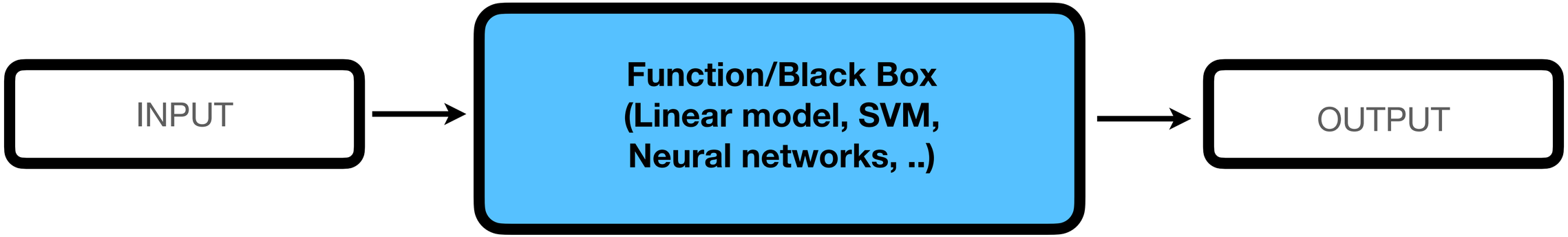
- Often more art than science
- Choices are based on the application and prior information about data

Underlying principles in ML: Model design

- Often more art than science
- Choices are based on the application and prior information about data
- This is what has led to the huge amount of models in deep learning
  - Old news: Linear regression model, logistic regression model, SVMs..
  - "New" news: MLPs, CNNs, ResNets, RNNs, LSTMs, Transformers, ..

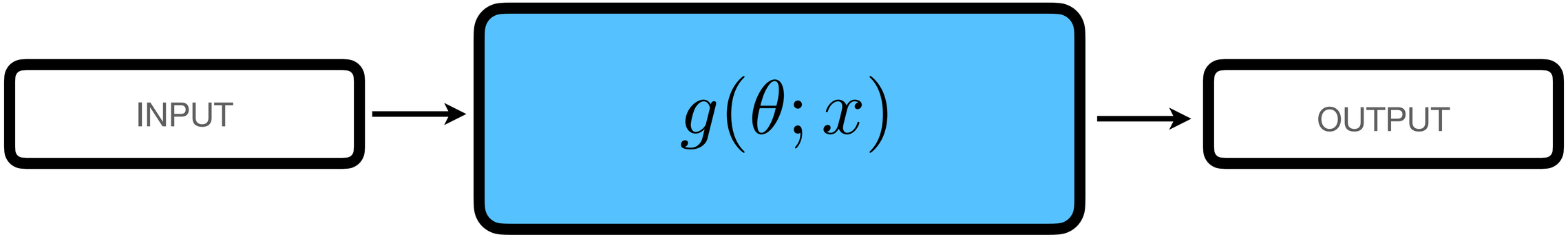
Underlying principles in ML: Model design

- Often more art than science
- Choices are based on the application and prior information about data
- This is what has led to the huge amount of models in deep learning
  - Old news: Linear regression model, logistic regression model, SVMs..
  - "New" news: MLPs, CNNs, ResNets, RNNs, LSTMs, Transformers, ..
- Boils down to designing a "black box":



Underlying principles in ML: Model design

- Often more art than science
- Choices are based on the application and prior information about data
- This is what has led to the huge amount of models in deep learning
  - Old news: Linear regression model, logistic regression model, SVMs..
  - "New" news: MLPs, CNNs, ResNets, RNNs, LSTMs, Transformers, ..
- Boils down to designing a "black box":



Underlying principles in ML: Loss function

- The loss function connects all the above
- I.e., given a trained model $g(\theta; x)$, the loss function will measure how far the model is from reality, during the testing phase.

Underlying principles in ML: Loss function

- The loss function connects all the above
- I.e., given a trained model $g(\theta; x)$, the loss function will measure how far the model is from reality, during the testing phase.
- I.e., given a untrained model $g(\theta; x)$, the loss function will measure how far the **current** model is from reality (training data), during the training phase.

Underlying principles in ML: Loss function

- The loss function connects all the above
- I.e., given a trained model $g(\theta; x)$, the loss function will measure how far the model is from reality, during the testing phase.
- I.e., given a untrained model $g(\theta; x)$, the loss function will measure how far the **current** model is from reality (training data), during the training phase.
- Thus the loss function connects:
  - Training/testing data (input/output)
  - Model $g(\theta; x)$
  - Defines the type of "distance" to measure how good of a model we get

Underlying principles in ML: Loss function

- The loss function connects all the above
- I.e., given a trained model $g(\theta; x)$, the loss function will measure how far the model is from reality, during the testing phase.
- I.e., given a untrained model $g(\theta; x)$, the loss function will measure how far the **current** model is from reality (training data), during the training phase.
- Thus the loss function connects:
  - Training/testing data (input/output)
  - Model $g(\theta; x)$
  - Defines the type of "distance" to measure how good of a model we get
- Finally, the loss function defines..

Underlying principles in ML: Optimization

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; \{x_i, y_i\}) \right\}$$

Underlying principles in ML: Optimization

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; \{x_i, y_i\}) \right\}$$

Regression
(e.g., image reconstruction)

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \| g(\theta; x_i) - y_i \|_2^2 \right\}$$

Underlying principles in ML: Optimization

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; \{x_i, y_i\}) \right\}$$

Regression
(e.g., image reconstruction)

Classification
(e.g., image classification)

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \| g(\theta; x_i) - y_i \|_2^2 \right\}$$

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \texttt{x-entropy}(g(\theta; x_i), y_i) \right\}$$

Underlying principles in ML: Optimization

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; \{x_i, y_i\}) \right\}$$

Regression
(e.g., image reconstruction)

Classification
(e.g., image classification)

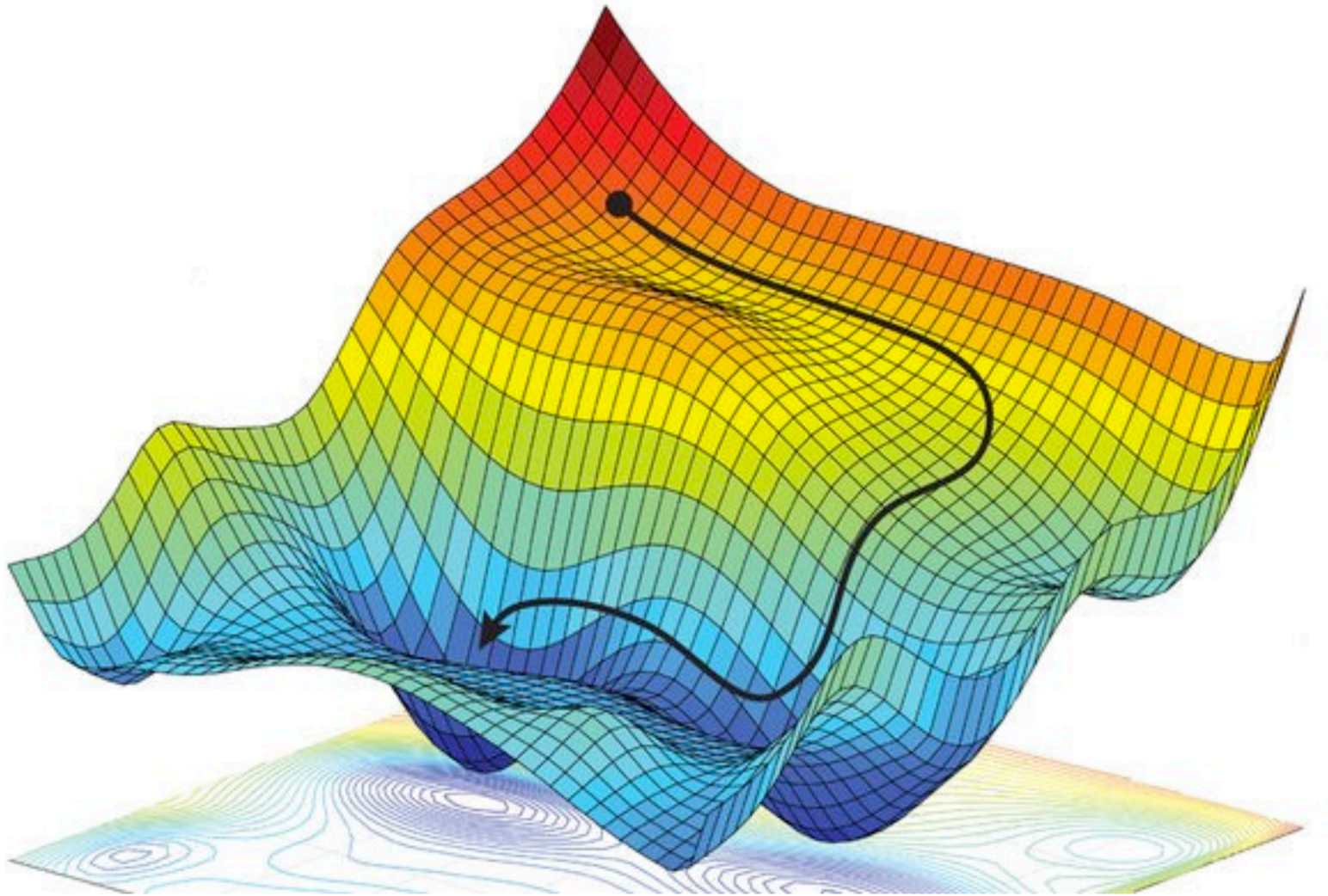
$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \| g(\theta; x_i) - y_i \|_2^2 \right\}$$

$$\arg\min \left\{ \mathcal{L}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \texttt{x-entropy}(g(\theta; x_i), y_i) \right\}$$

- How we optimize? Gradient descent (or backpropagation)!

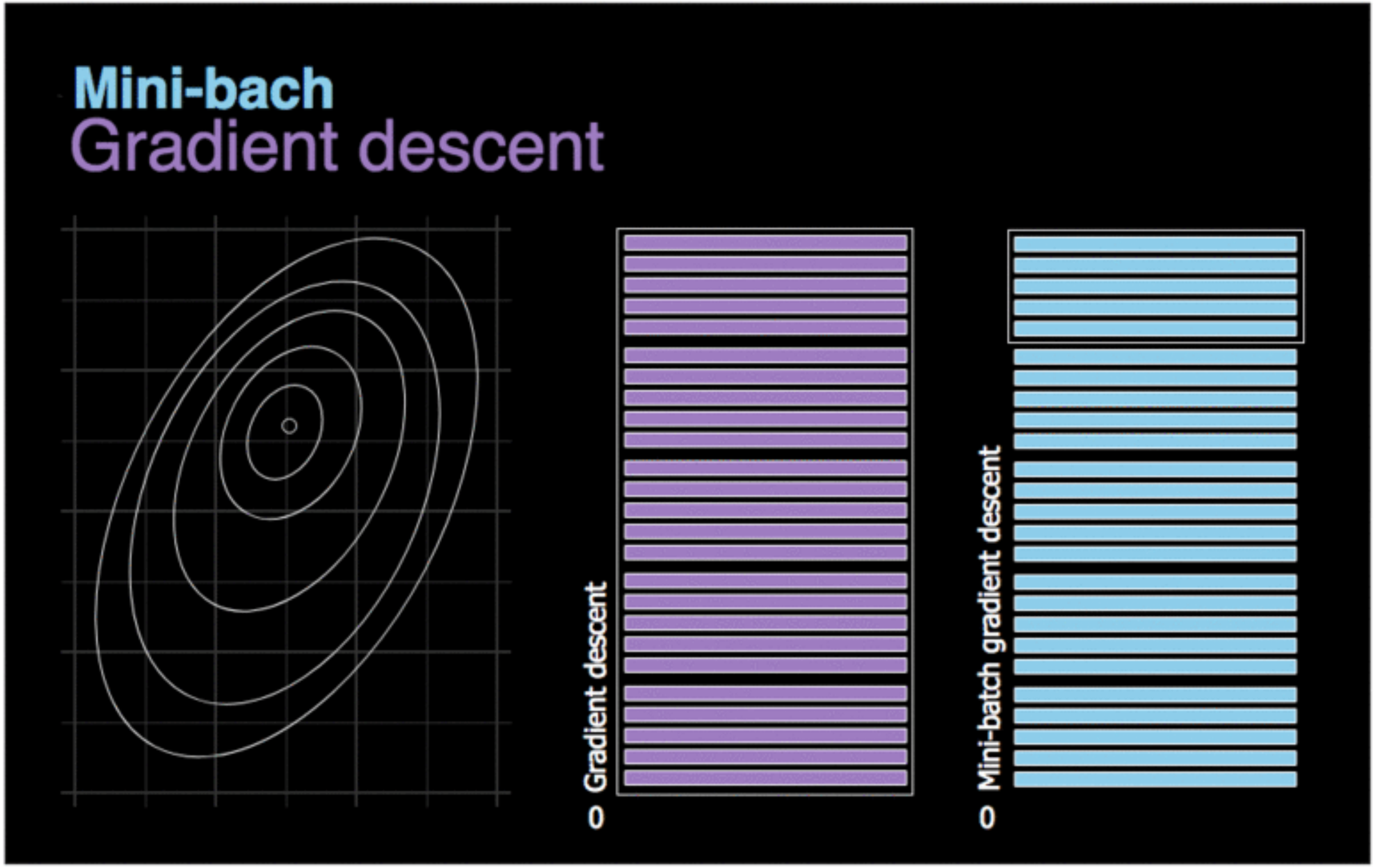
$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}_{i_t}(\theta_t)$$

Underlying principles in ML: Optimization



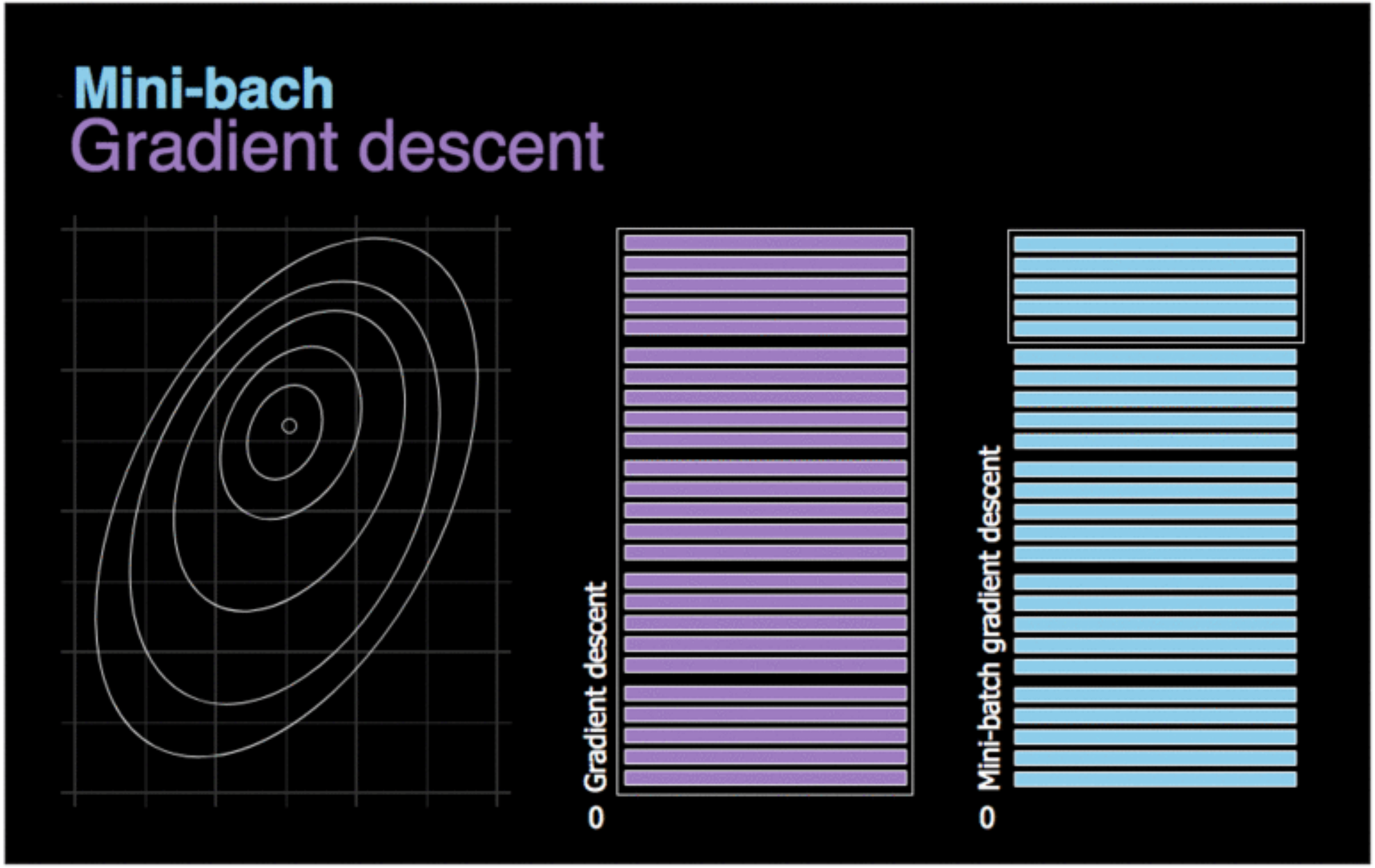
- In practice, we use variants of GD: Adam, AdaGrad, AdamW, ..

Stochastic gradient descent



Tribute to
Piotr Skalski

Stochastic gradient descent



Tribute to
Piotr Skalski

Overview

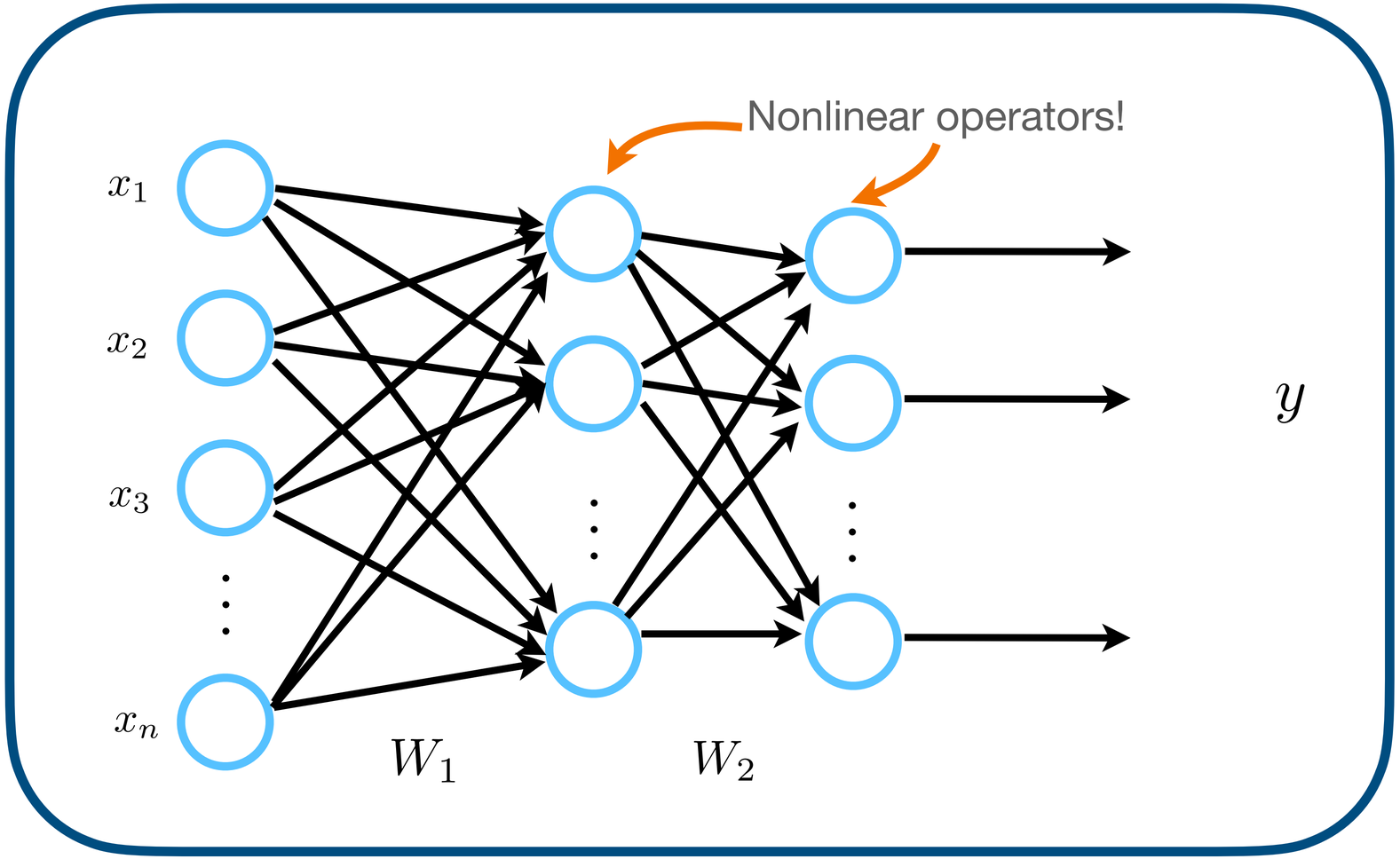
- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)

Overview

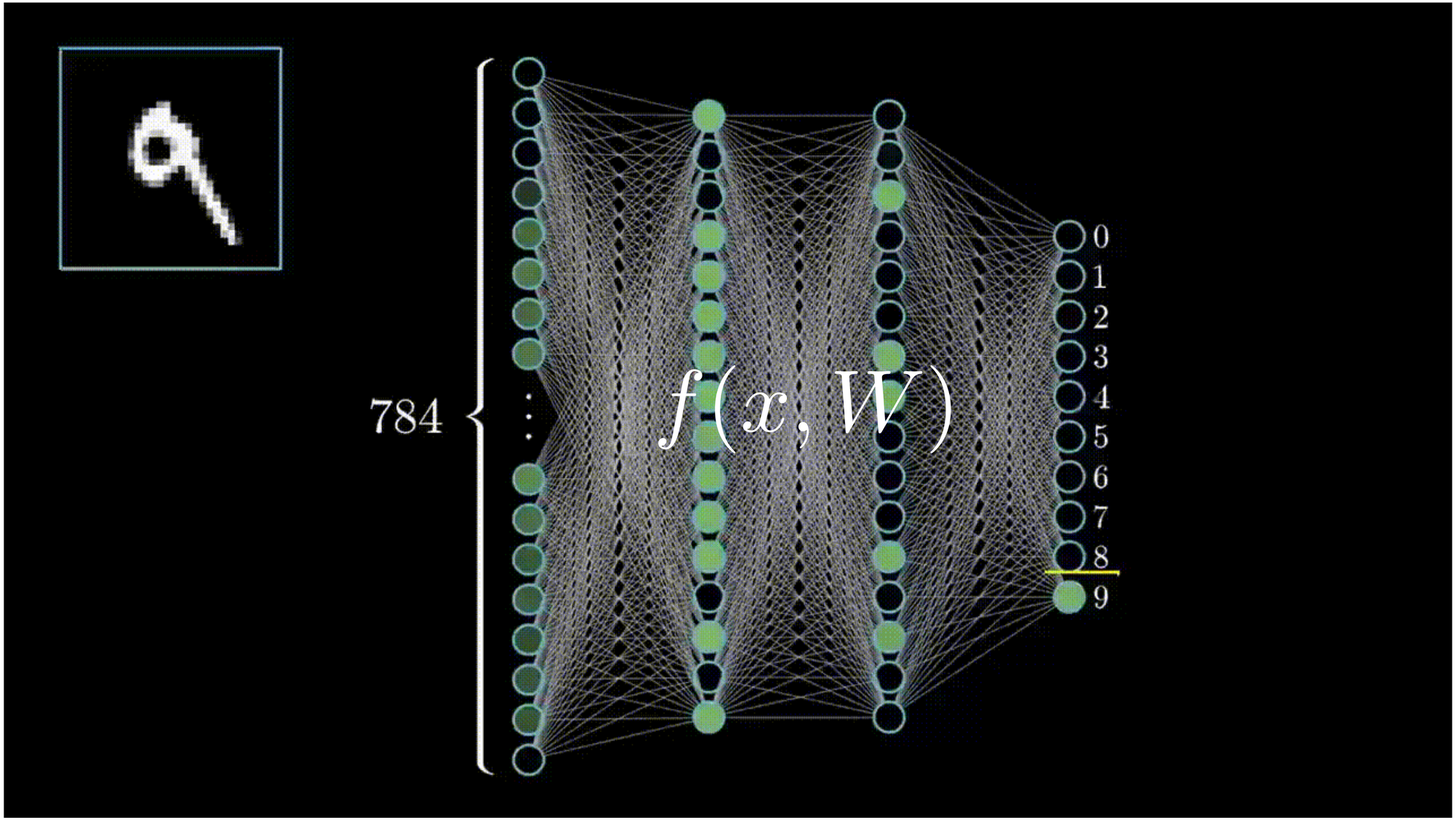
- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)

Multi-Layer Perceptrons or MLPs

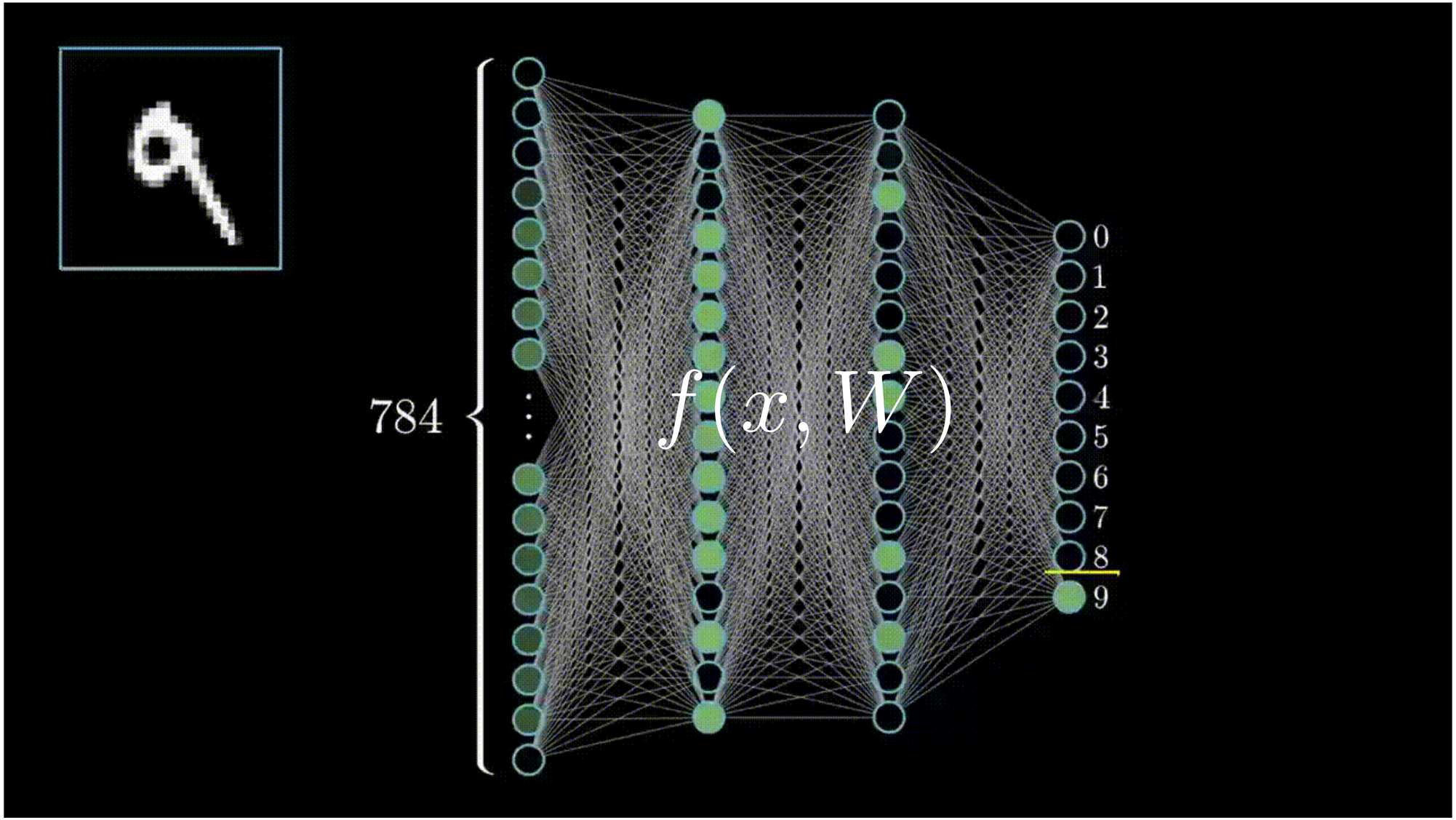
Feedforward/fully connected neural network



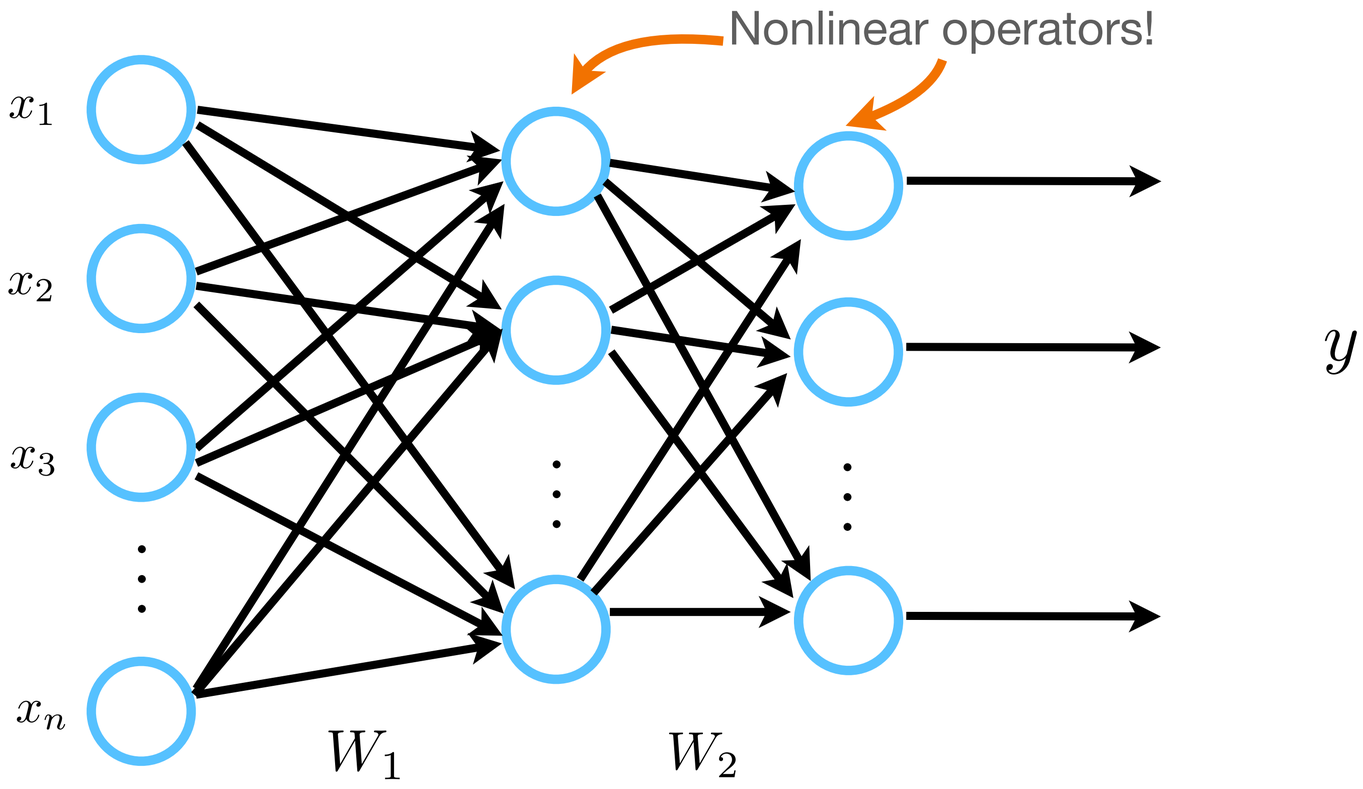
Fully connected neural networks or MLPs



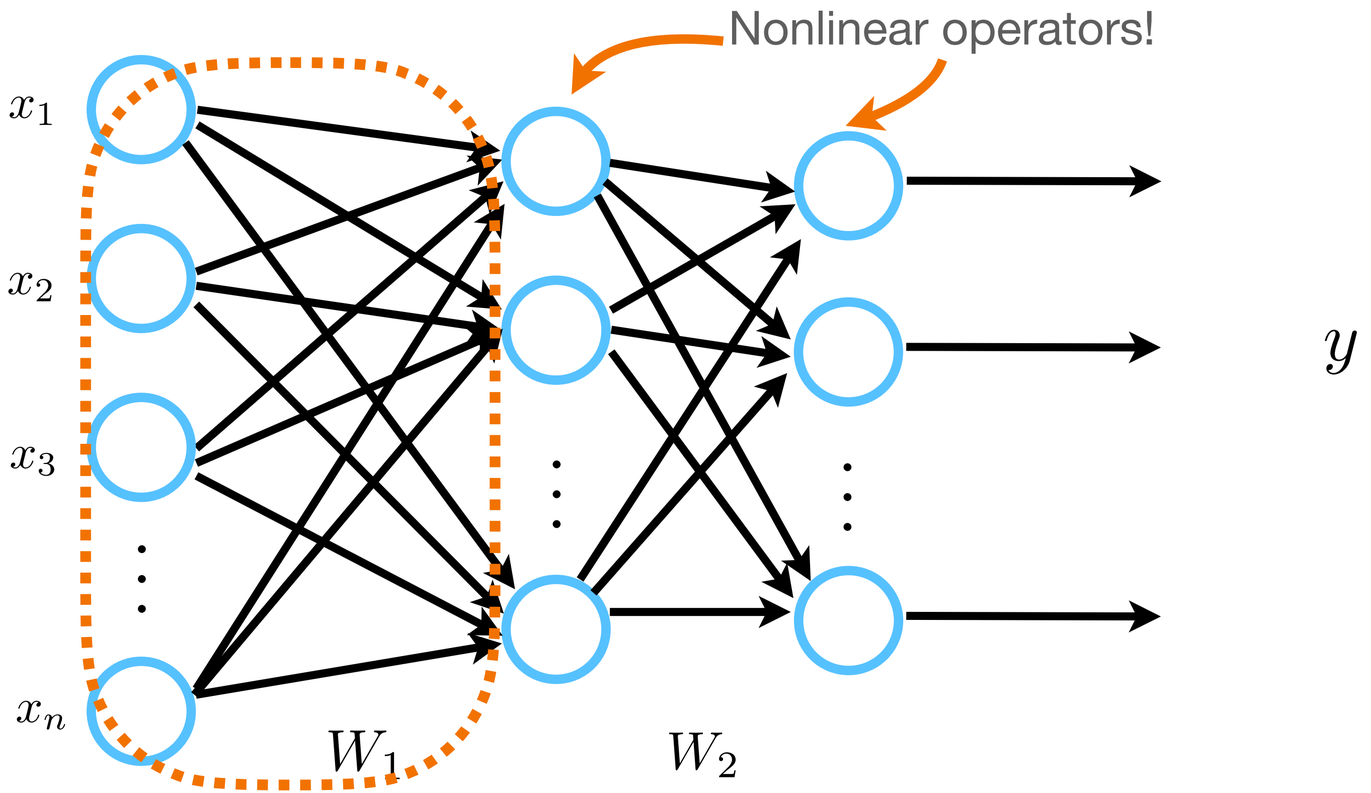
Fully connected neural networks or MLPs



Fully connected neural networks or MLPs

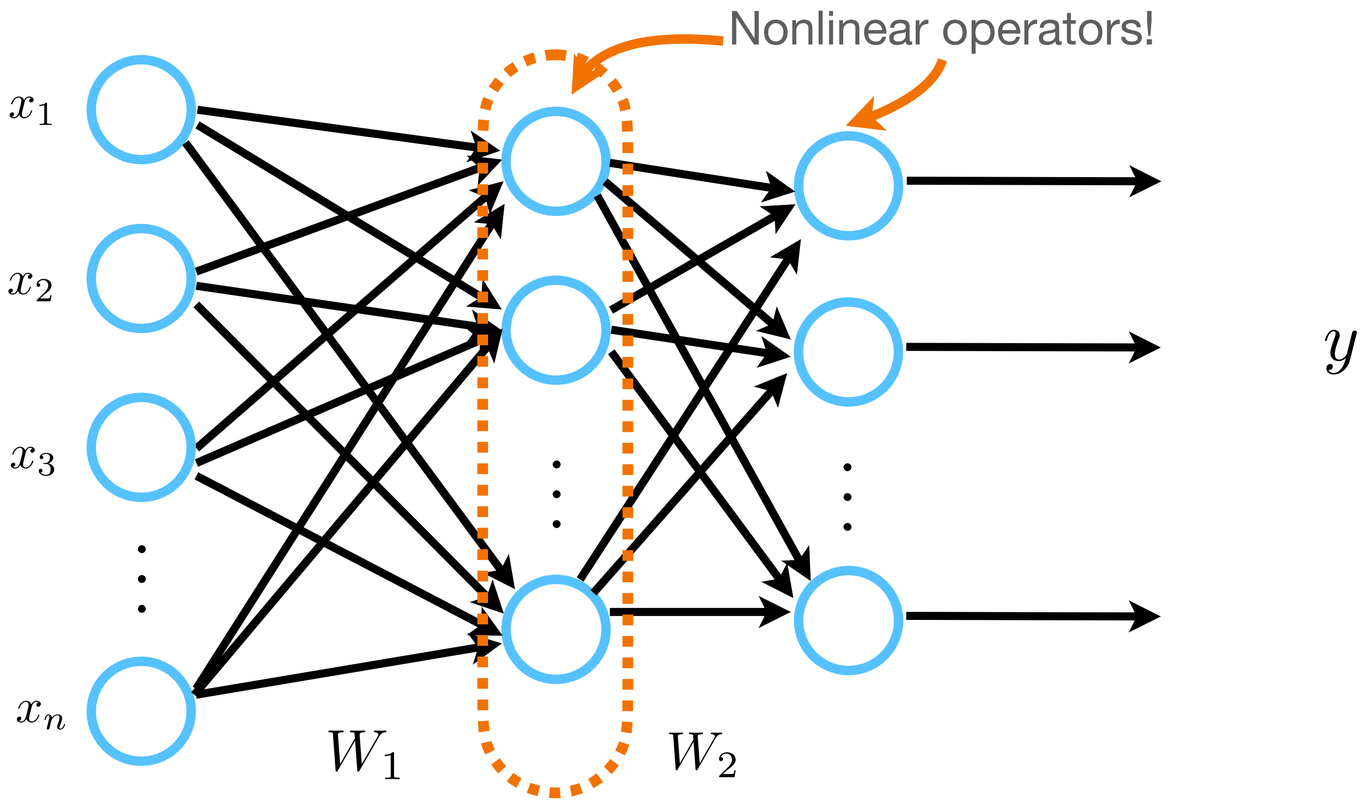


Fully connected neural networks or MLPs



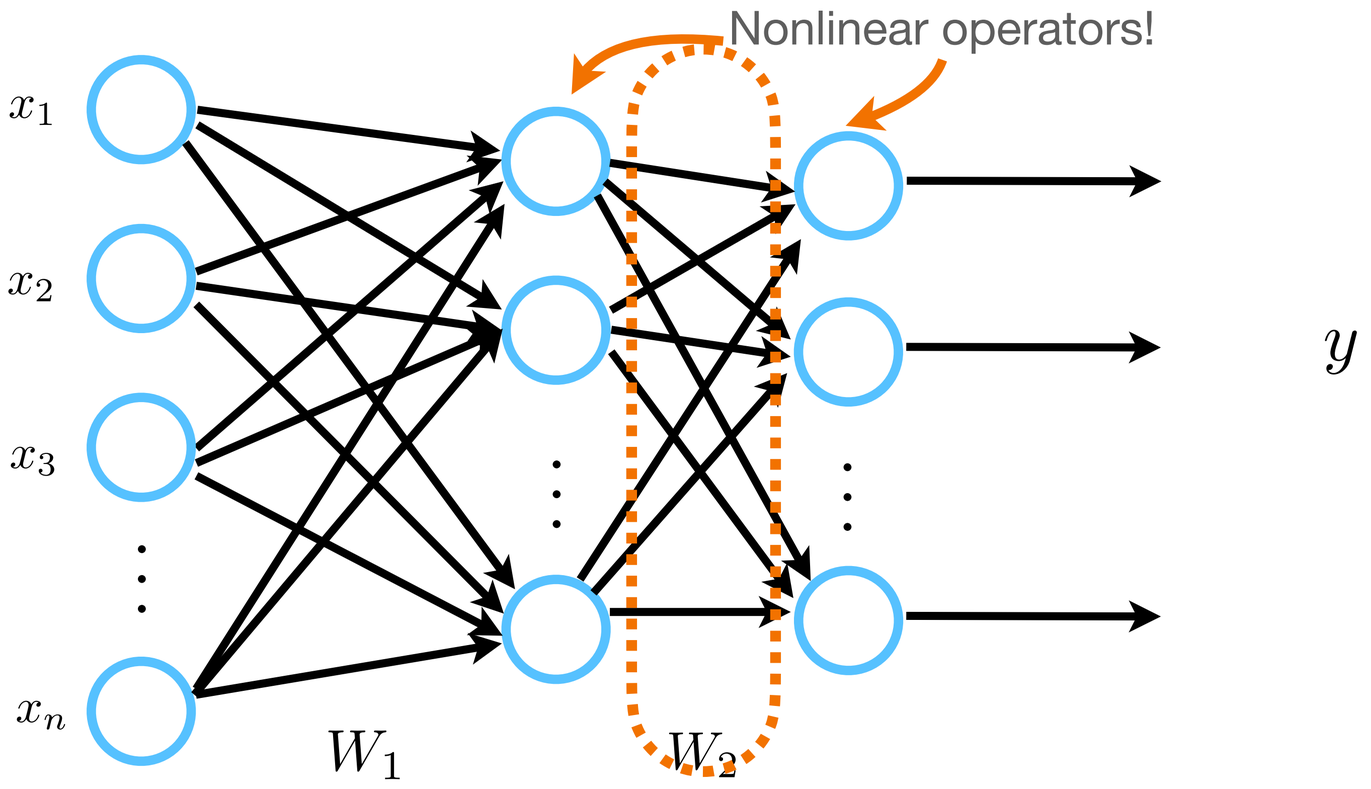
1. $z_1 = W_1 \cdot x$

Fully connected neural networks or MLPs



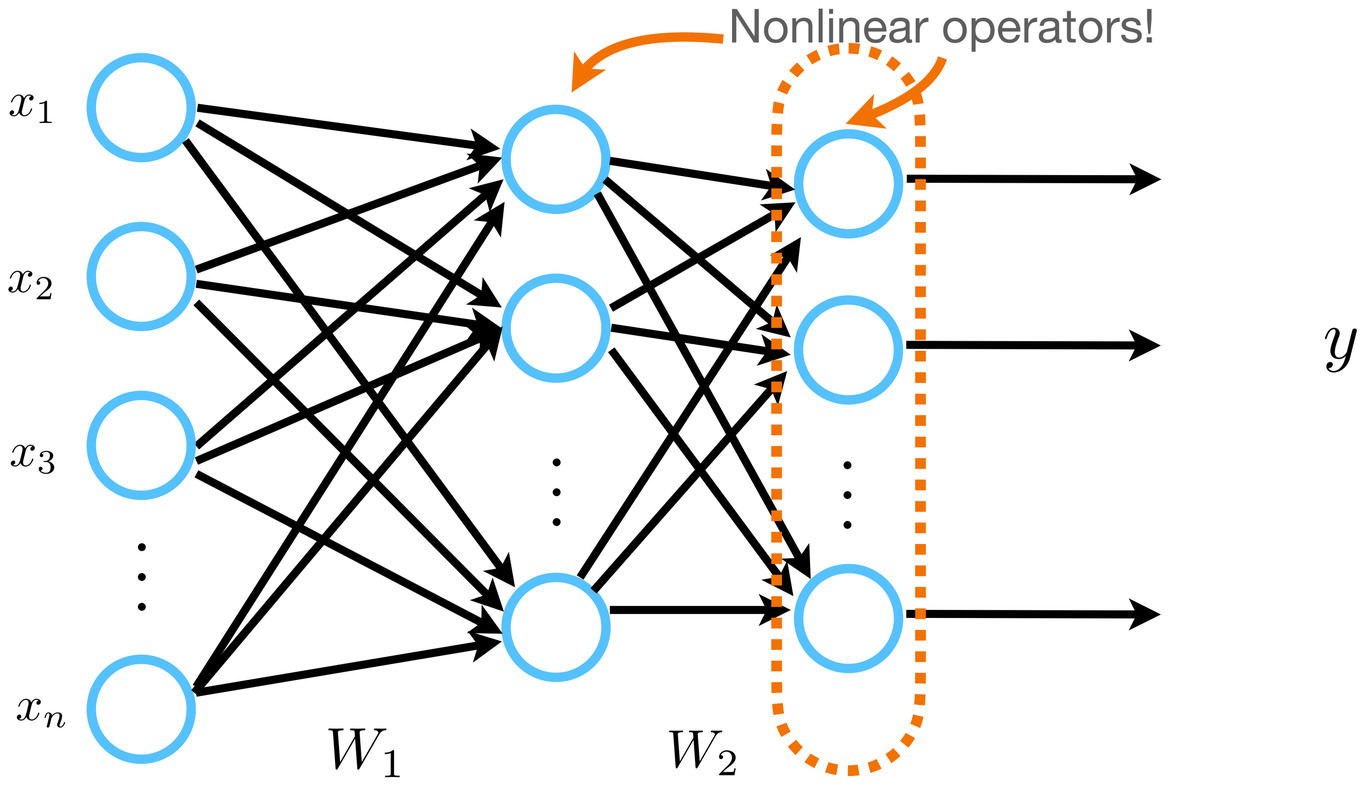
1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$

Fully connected neural networks or MLPs



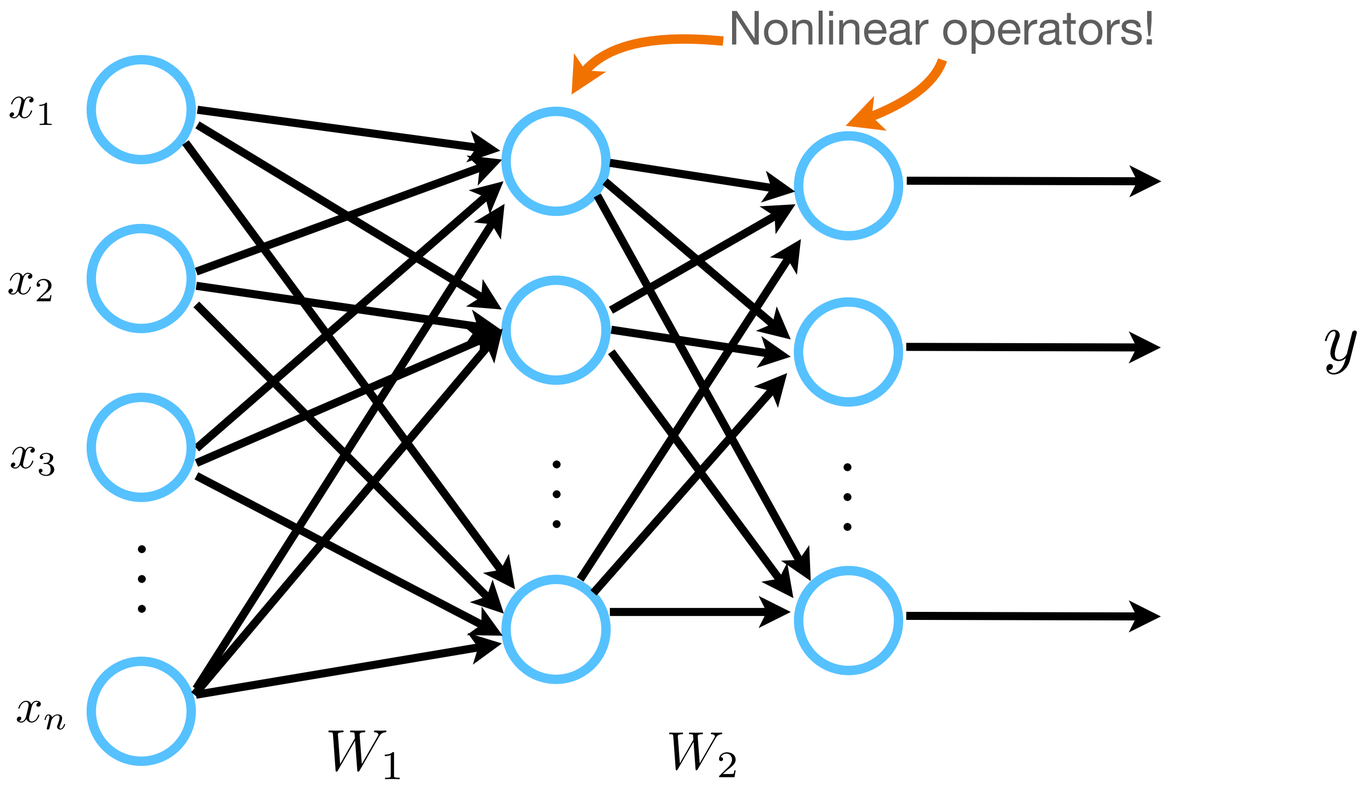
1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$

Fully connected neural networks or MLPs



1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$
4. $\texttt{softmax}(z_2) = \texttt{softmax}(W_2 \cdot \sigma(W_1 \cdot x))$

Fully connected neural networks or MLPs



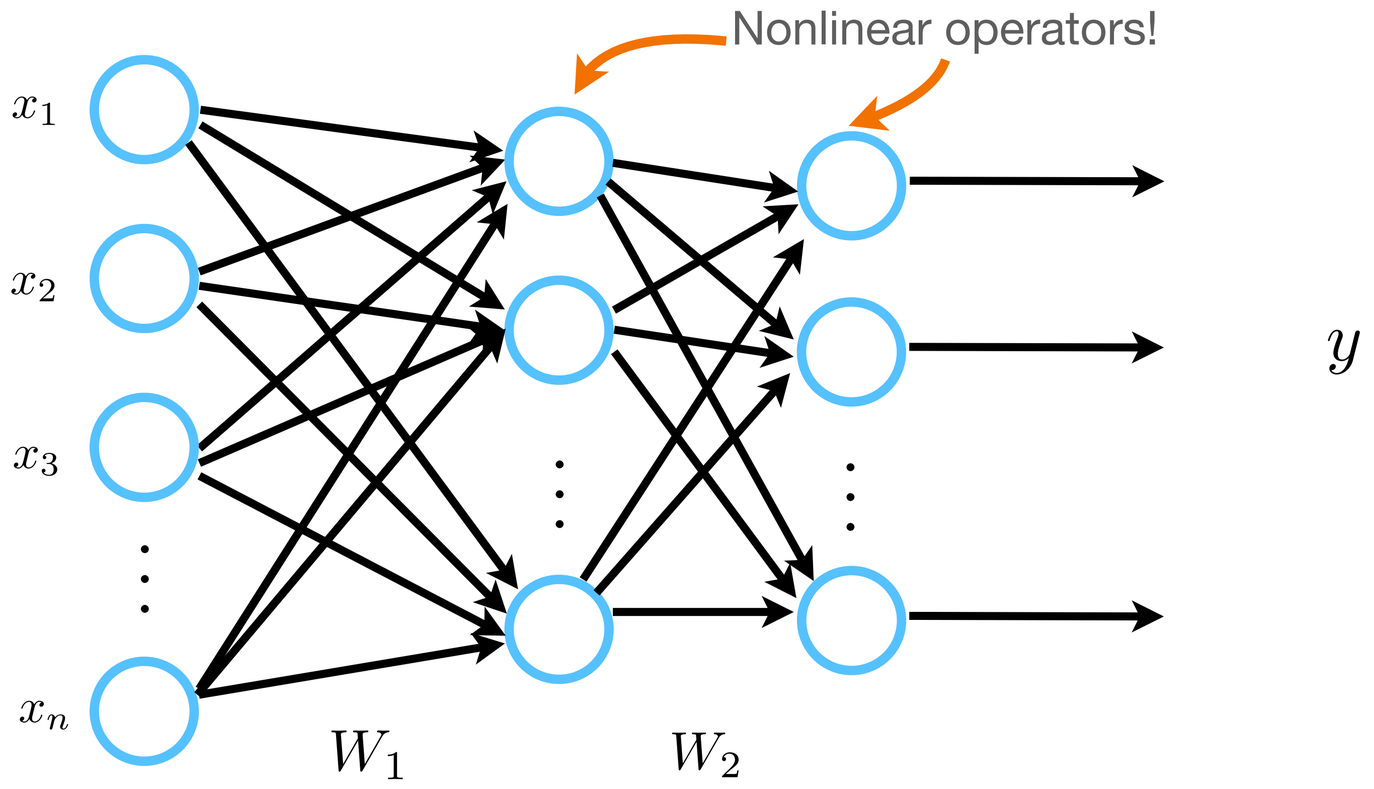
1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$
4. $\texttt{softmax}(z_2) = \texttt{softmax}(W_2 \cdot \sigma(W_1 \cdot x))$

- Nothing more complicated
- Boils down to what modules will be used
- Famous architecture due to simplicity and theoretical guarantees

Fully connected neural networks or MLPs

This sequence of operations is also known as the **forward pass on the neural network**

You can think of forward pass as function evaluation



1. $z_1 = W_1 \cdot x$
2. $\sigma(z_1) = \sigma(W_1 \cdot x)$
3. $z_2 = W_2 \cdot \sigma(z_1) = W_2 \cdot \sigma(W_1 \cdot x)$
4. $\texttt{softmax}(z_2) = \texttt{softmax}(W_2 \cdot \sigma(W_1 \cdot x))$

- Nothing more complicated
- Boils down to what modules will be used
- Famous architecture due to simplicity and theoretical guarantees

What differentiates one neural network from another

What differentiates one neural network from another

- Operations involved: Feedforward layers only

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps

What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps
- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.

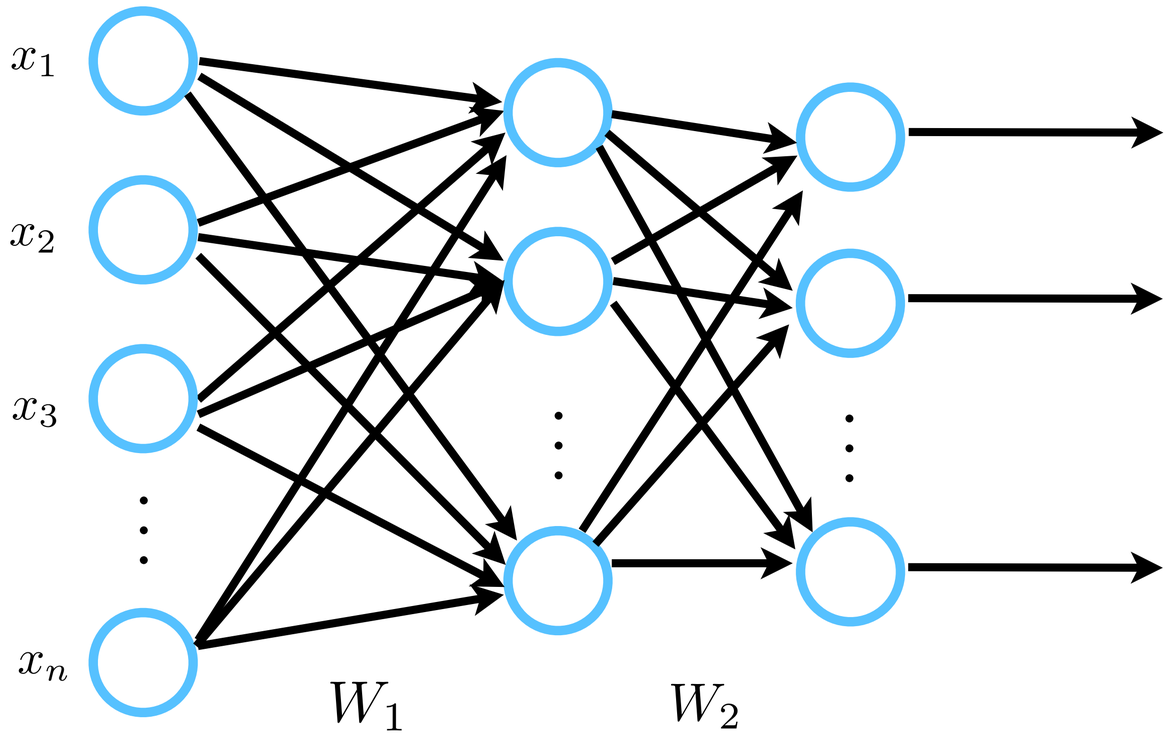
What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps
- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.
- Deep vs. shallow, wide vs. narrow: Although shallow wide NNs work well in theory, deep nets are more efficient, and generalize better.

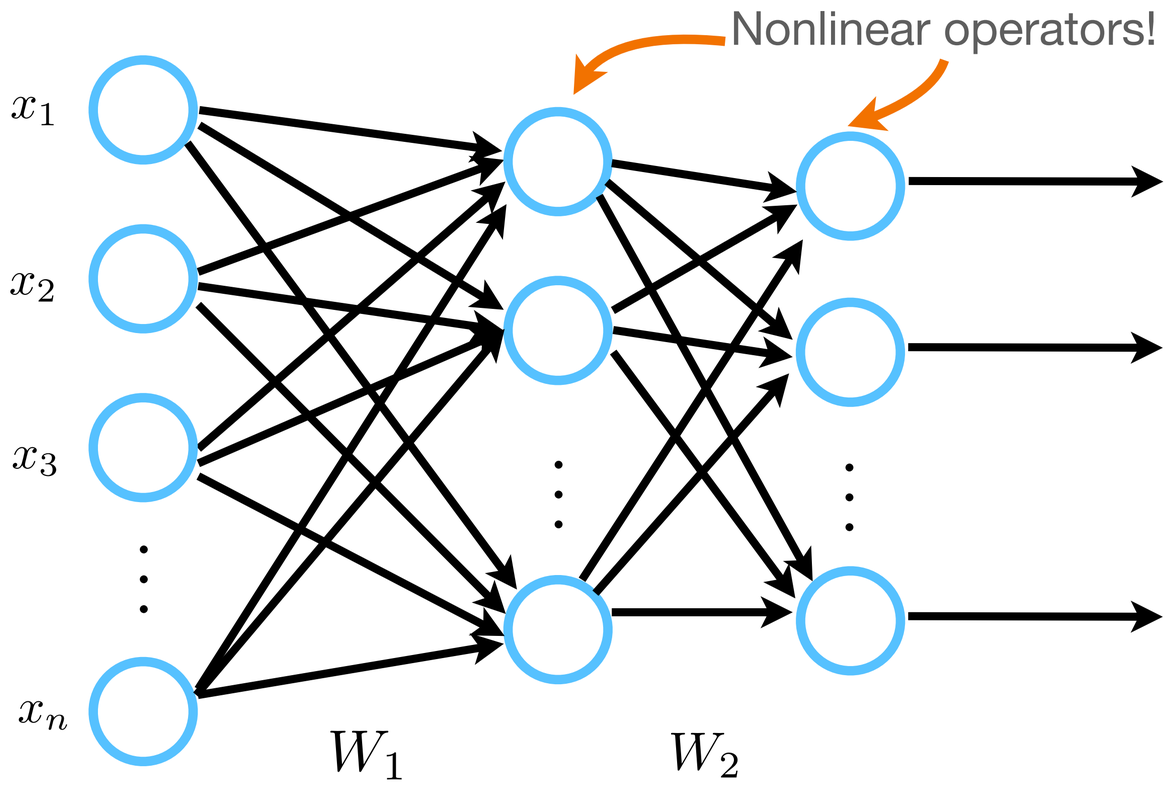
What differentiates one neural network from another

- Operations involved: Feedforward layers only
  Convolution layers
  Pooling operations
  Regularization techniques (dropout, batch normalization)
  Residual steps
  Recurrent steps
- Non-linear functions used: ReLU, leaky ReLUs, tanh, sigmoid, etc.
- Deep vs. shallow, wide vs. narrow: Although shallow wide NNs work well in theory, deep nets are more efficient, and generalize better.
- Objective functions: euclidean norm (regression), cross entropy (classification) only input data (autoencoders), min-max (GANs), etc.
  (or type of learning)

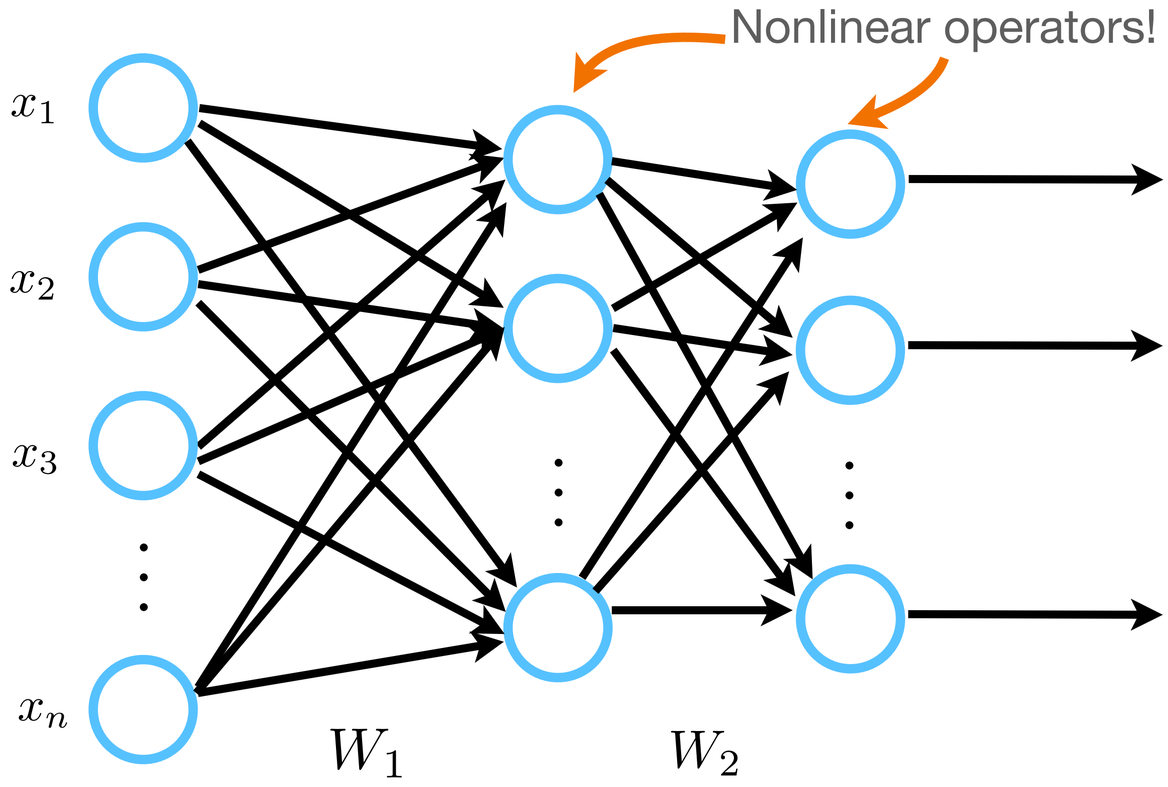
What are our options here wrt modules?



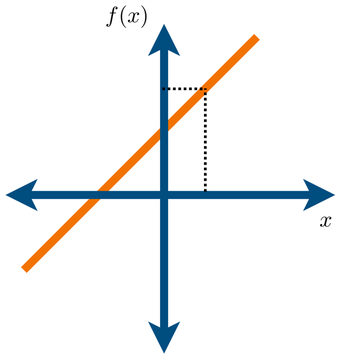
What are our options here wrt modules?

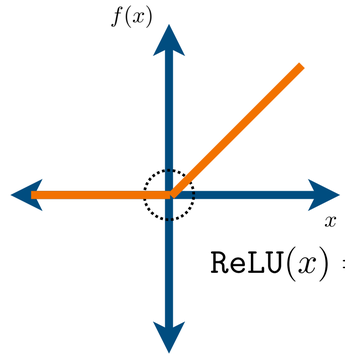


What are our options here wrt modules?



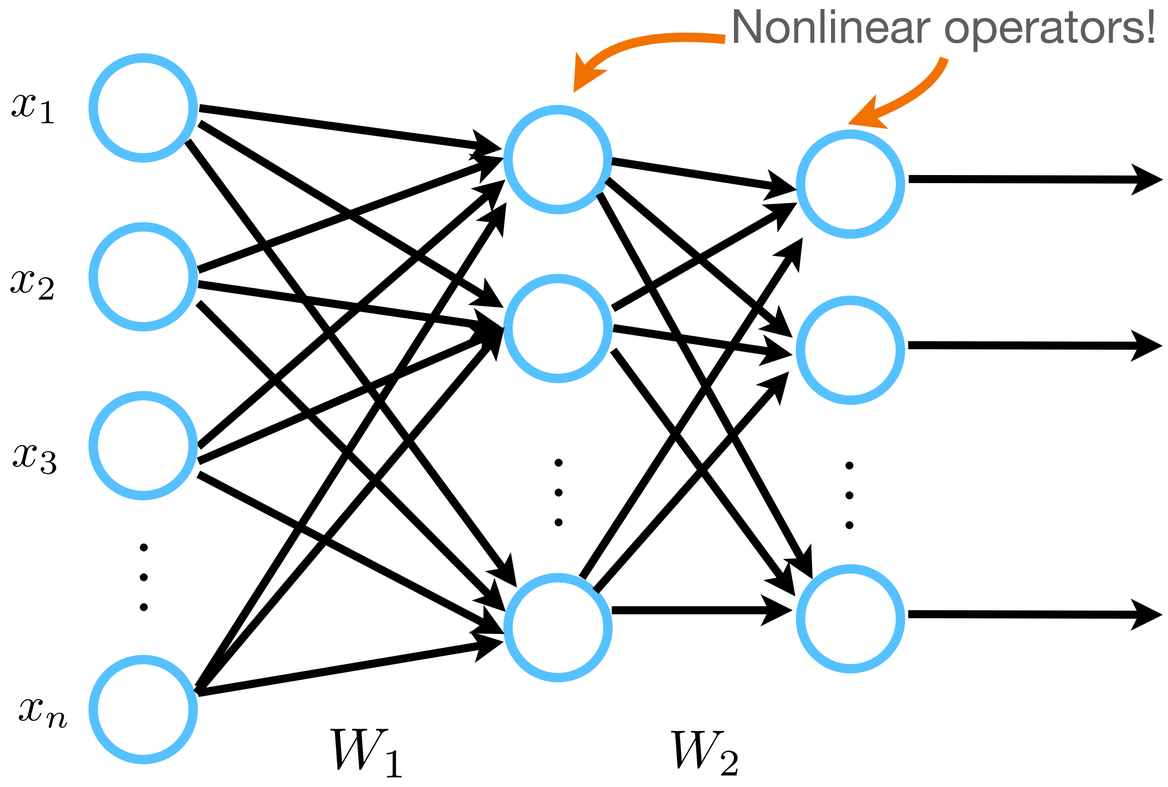
- We saw linear and ReLU



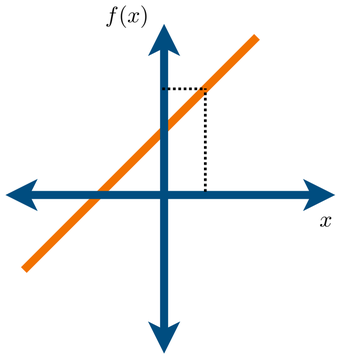


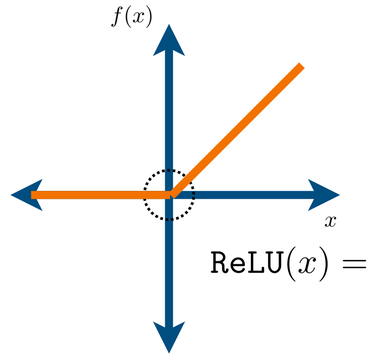
$$\texttt{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

What are our options here wrt modules?



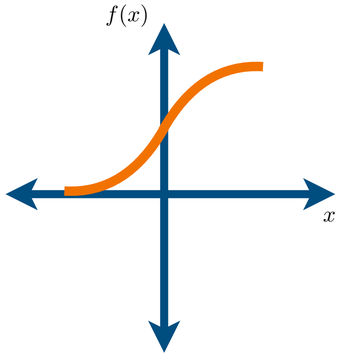
- We saw linear and ReLU



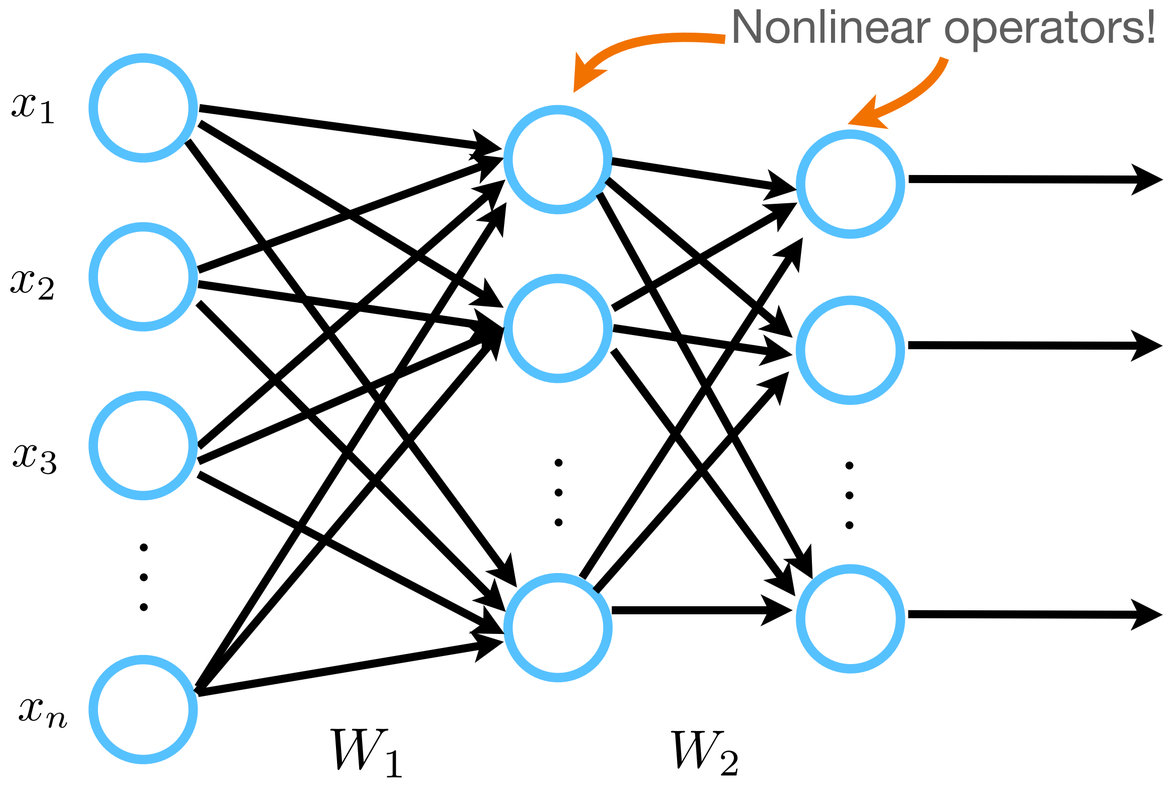


$$\texttt{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

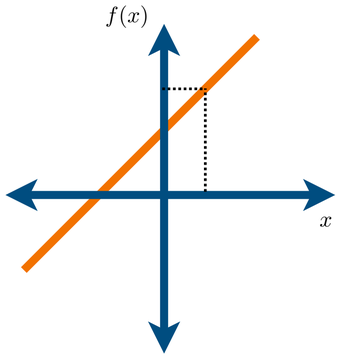
- There is also sigmoid..

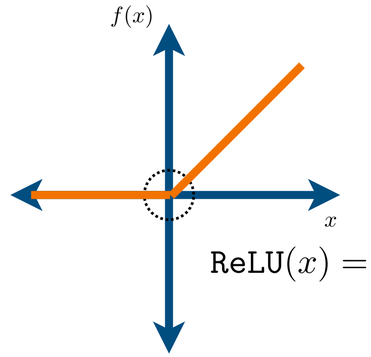


What are our options here wrt modules?



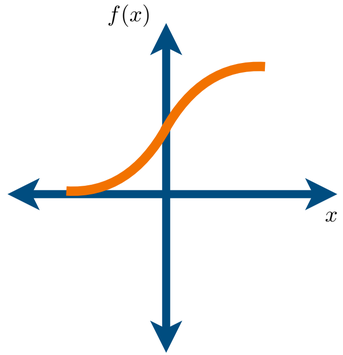
- We saw linear and ReLU

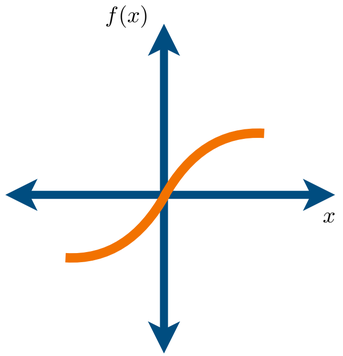




$$\texttt{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

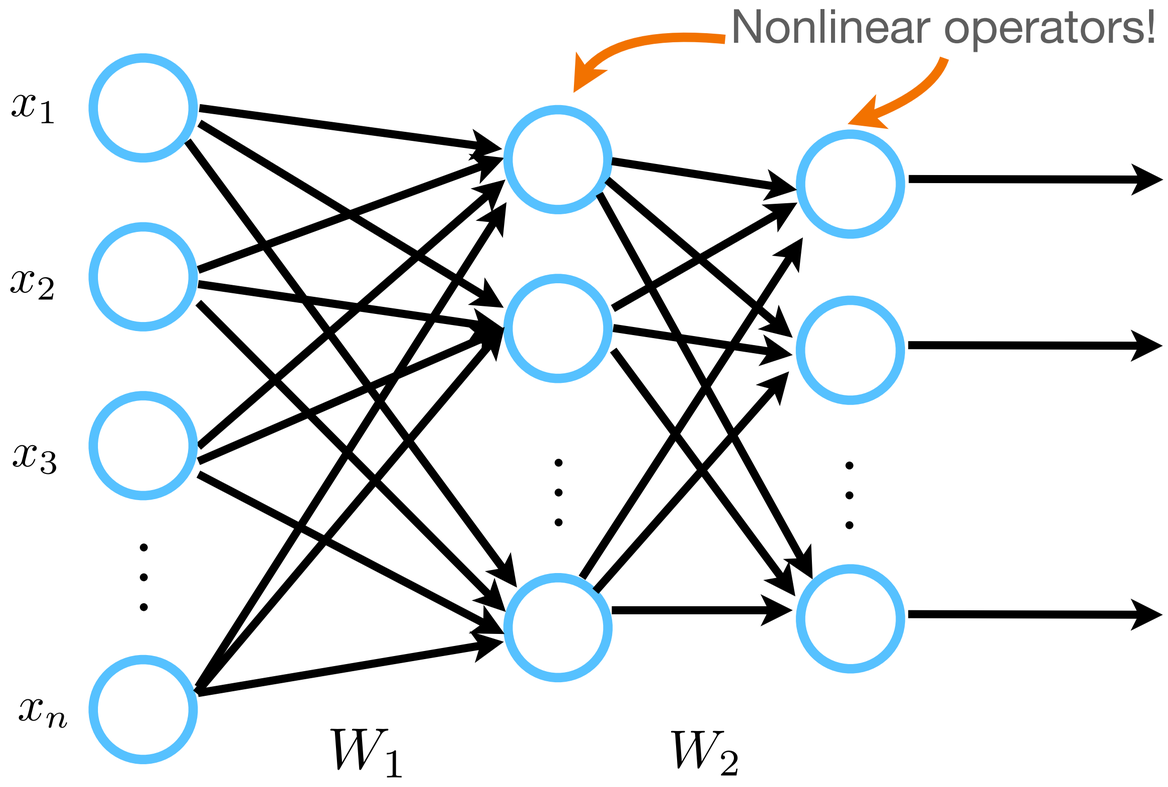
- There is also sigmoid..



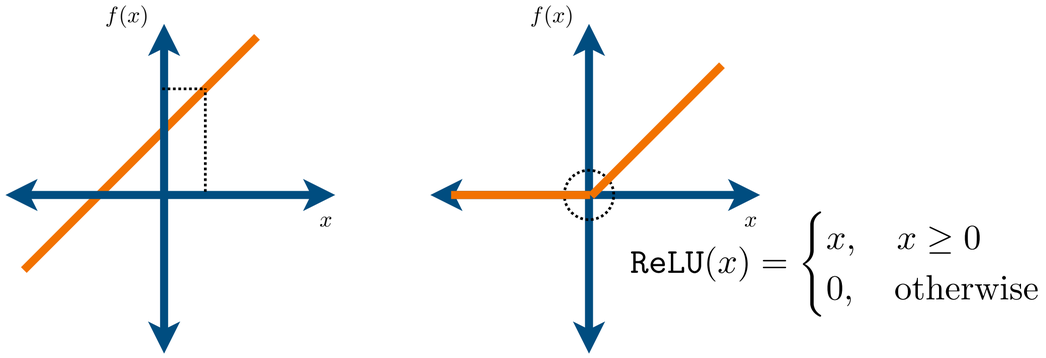


- ..and tanh functions.

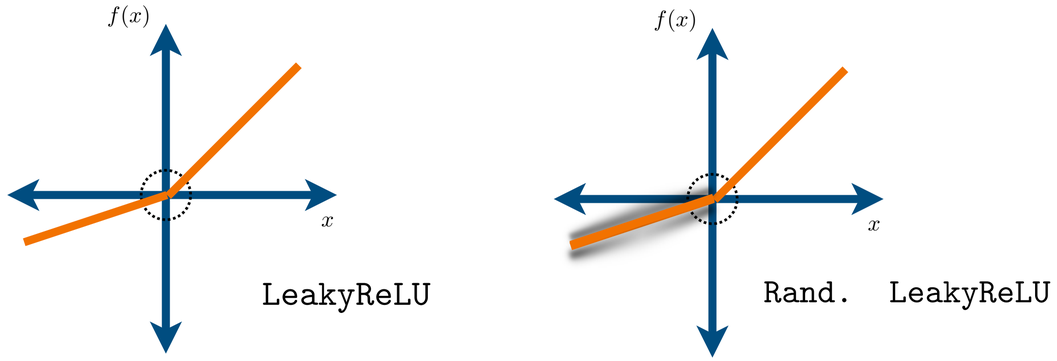
What are our options here wrt modules?



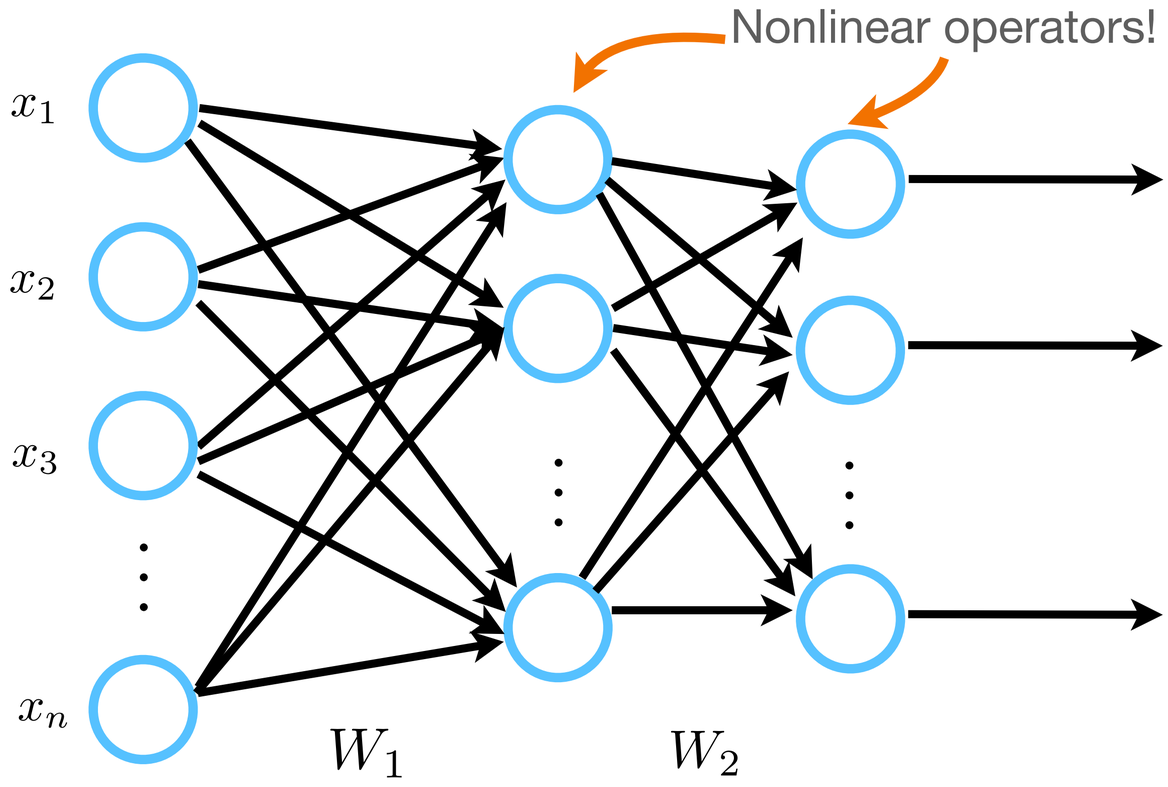
- We saw linear and ReLU



- Zoo of ReLU functions

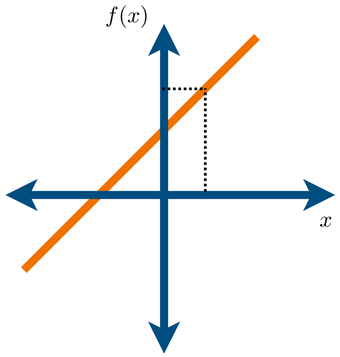


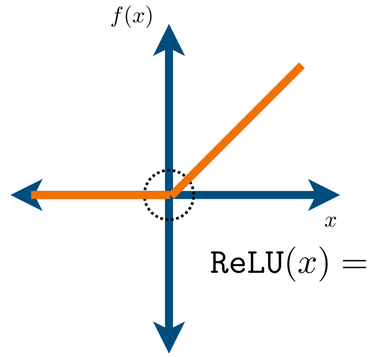
What are our options here wrt modules?



- What are the differences?

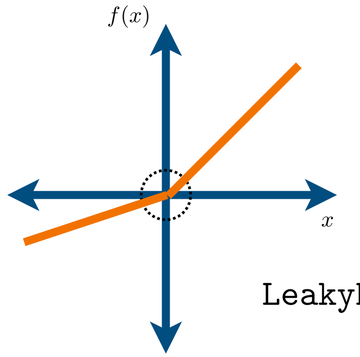
- We saw linear and ReLU



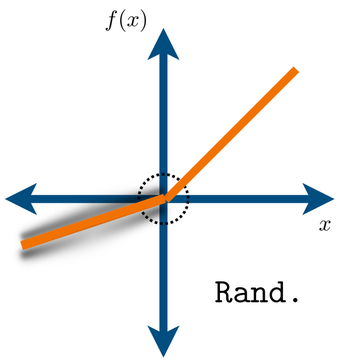


$$\texttt{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

- Zoo of ReLU functions

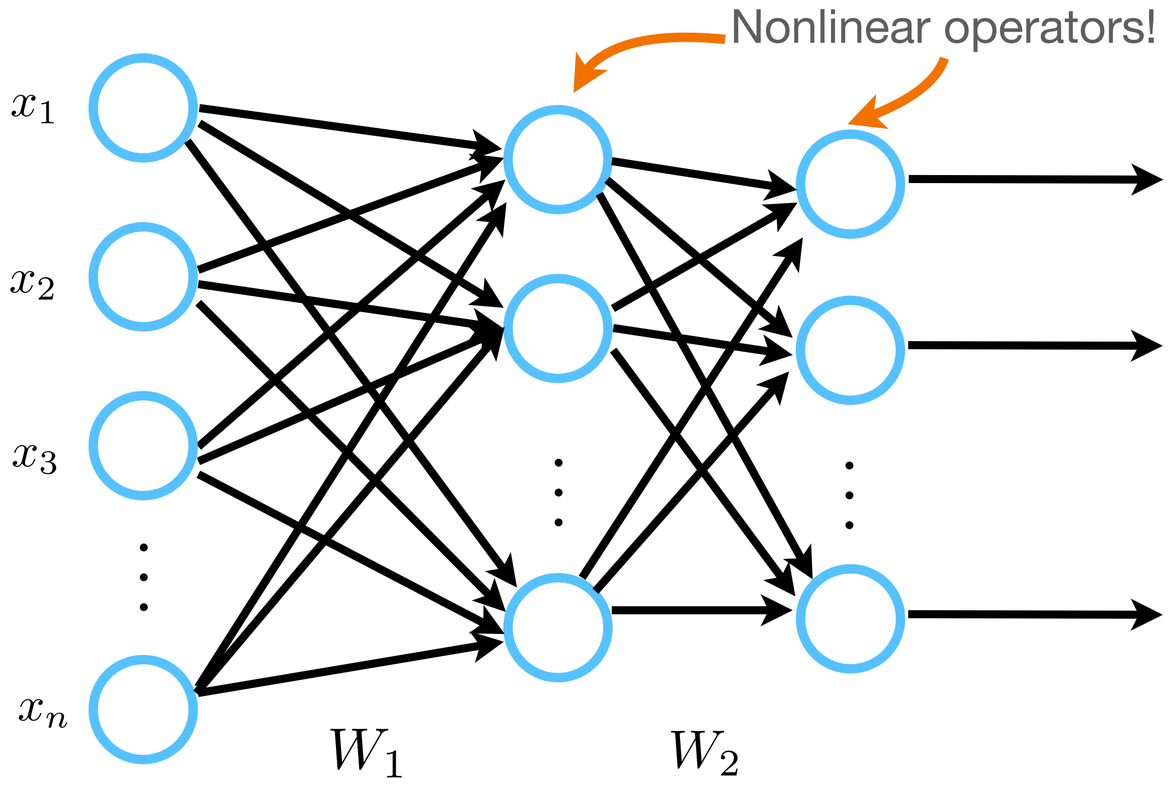


`LeakyReLU`



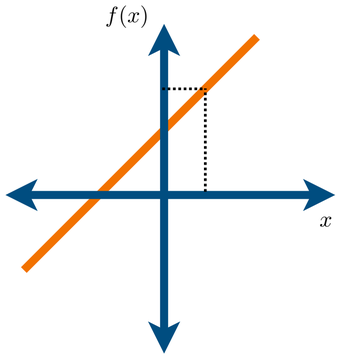
`Rand.` `LeakyReLU`

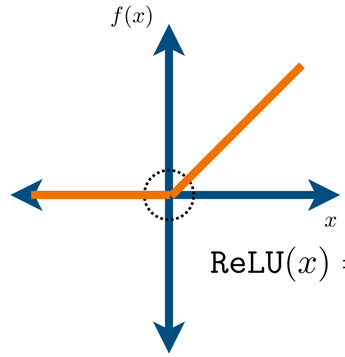
What are our options here wrt modules?



- What are the differences?
  - Some modules have simpler derivates than others (affects optimization)

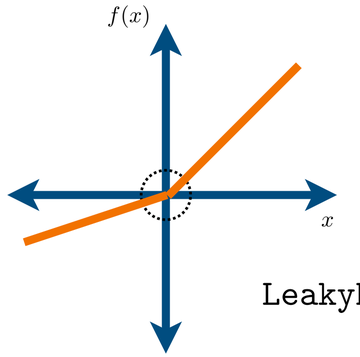
- We saw linear and ReLU



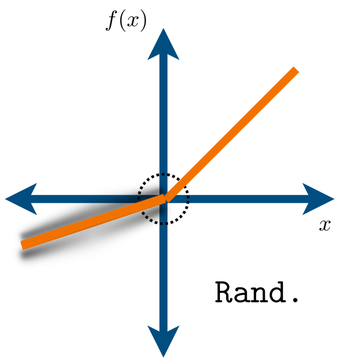


$$\texttt{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

- Zoo of ReLU functions

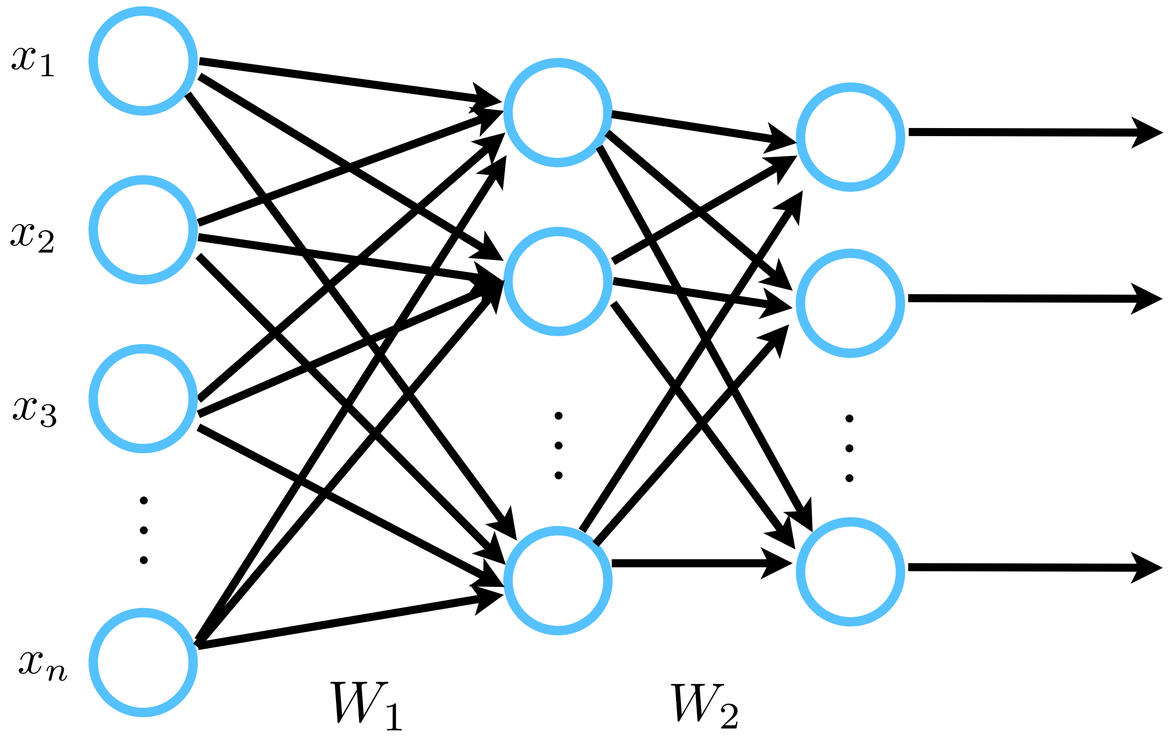


`LeakyReLU`

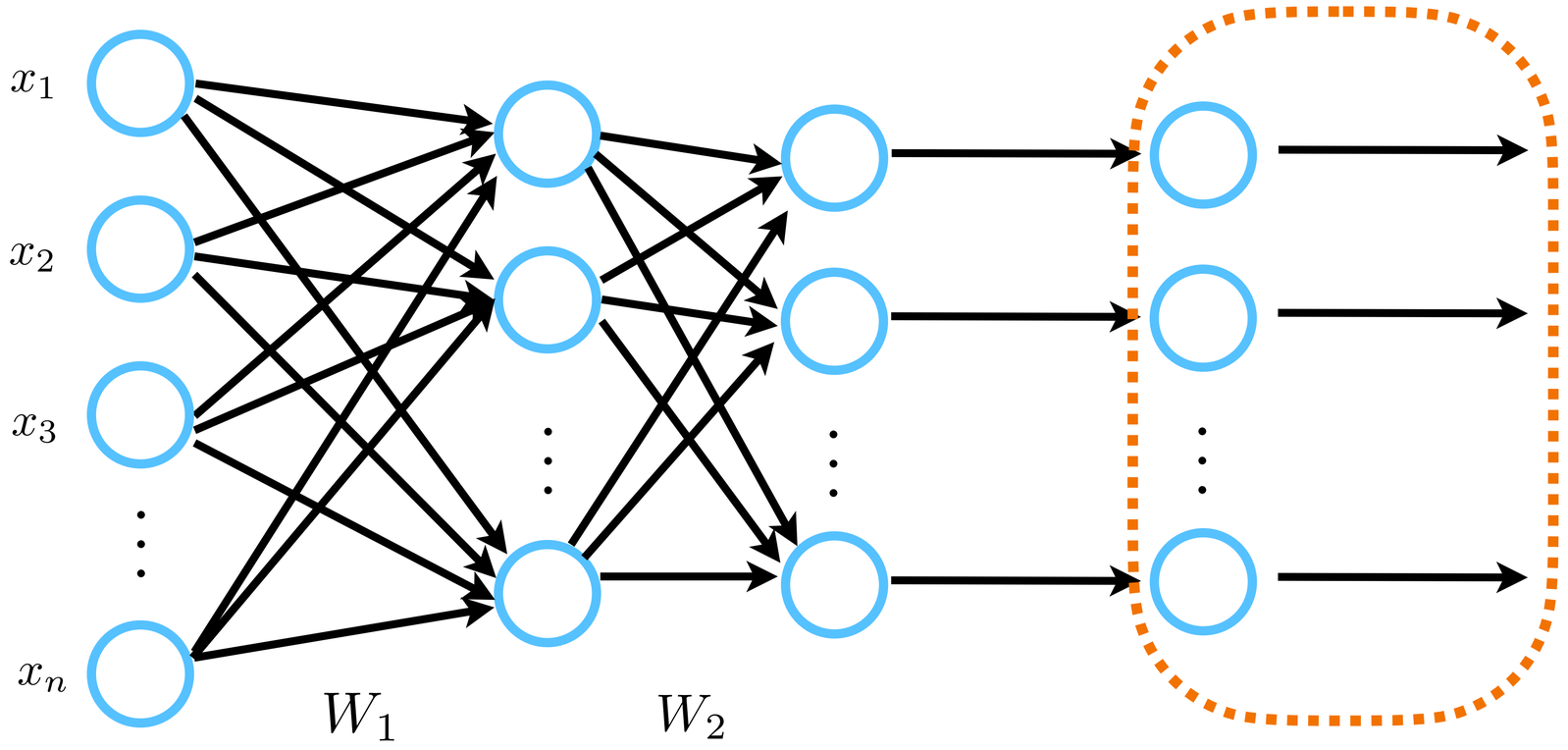


`Rand.` `LeakyReLU`

What are our options here wrt modules?

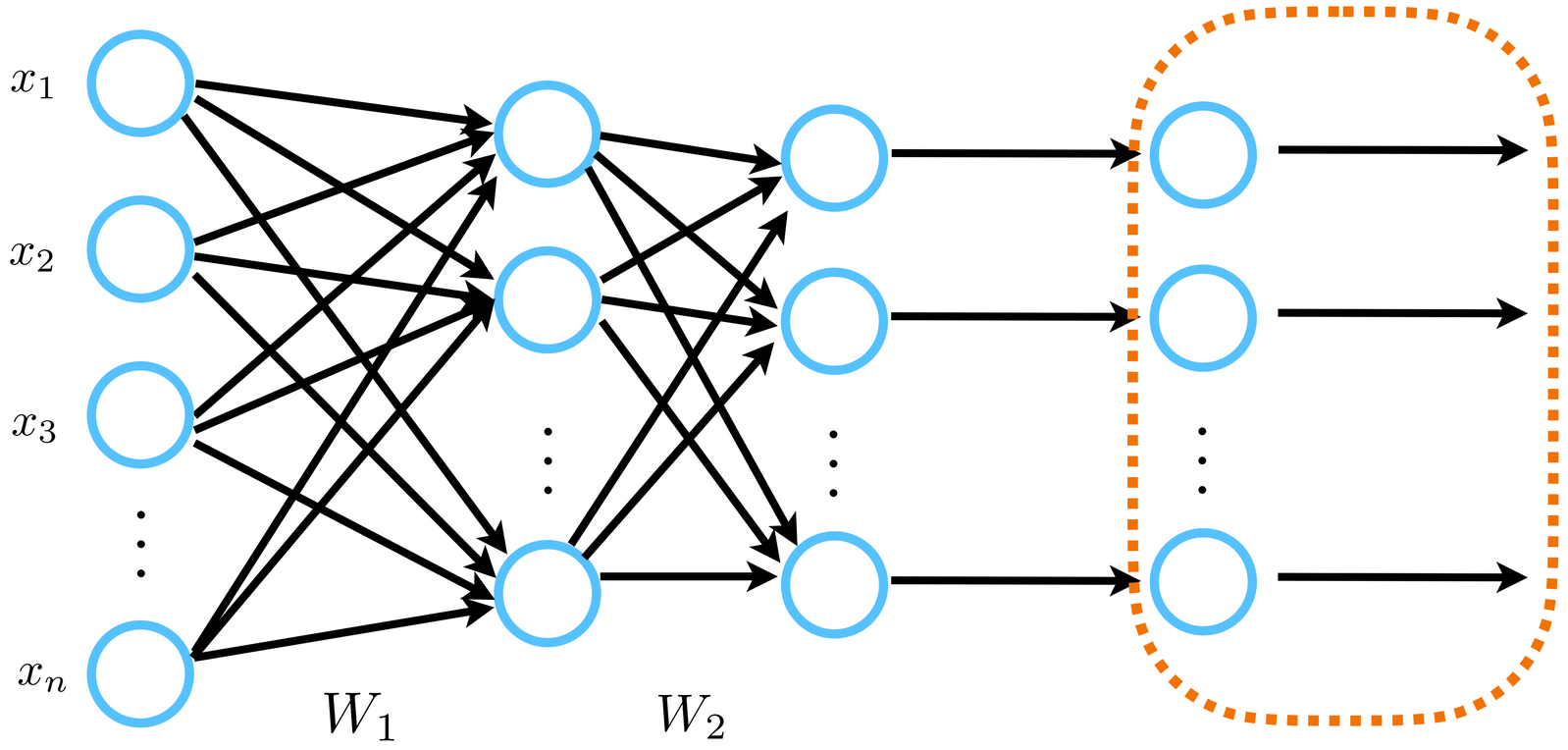


What are our options here wrt modules?



- In many cases, we use an additional module: $\texttt{softmax} : \mathbb{R}^k \to \mathbb{R}^k$

What are our options here wrt modules?

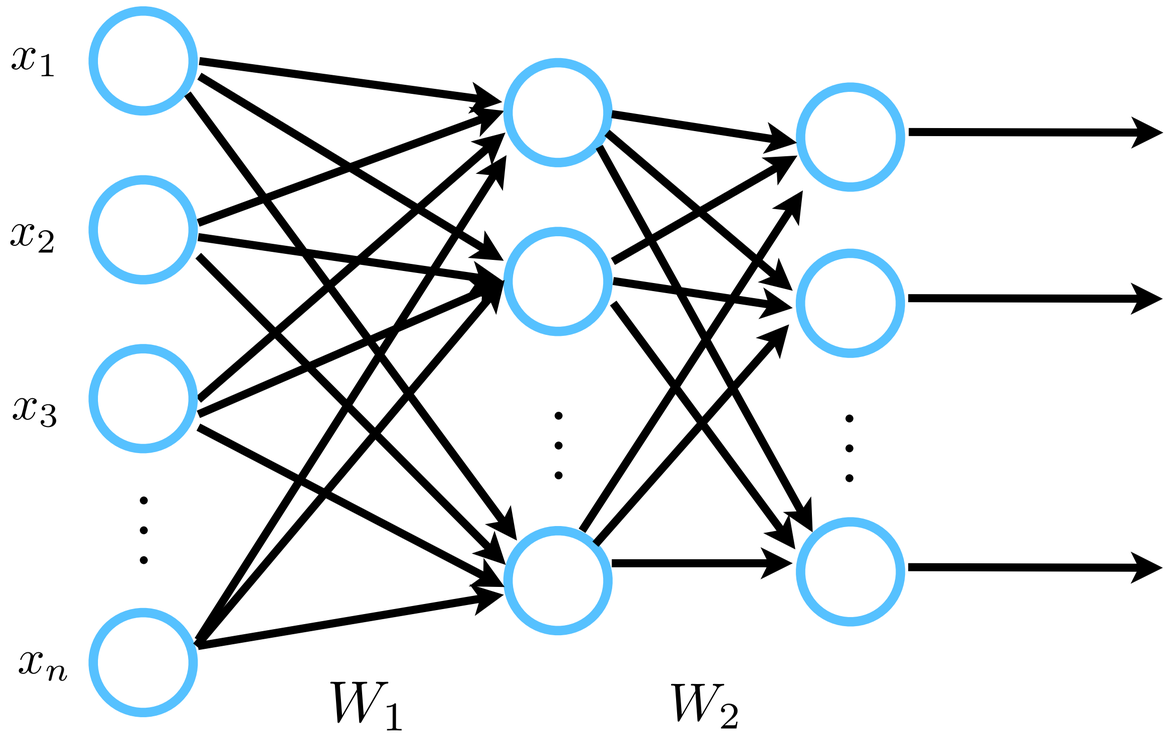


- In many cases, we use an additional module: $\texttt{softmax} : \mathbb{R}^k \to \mathbb{R}^k$

$$\texttt{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

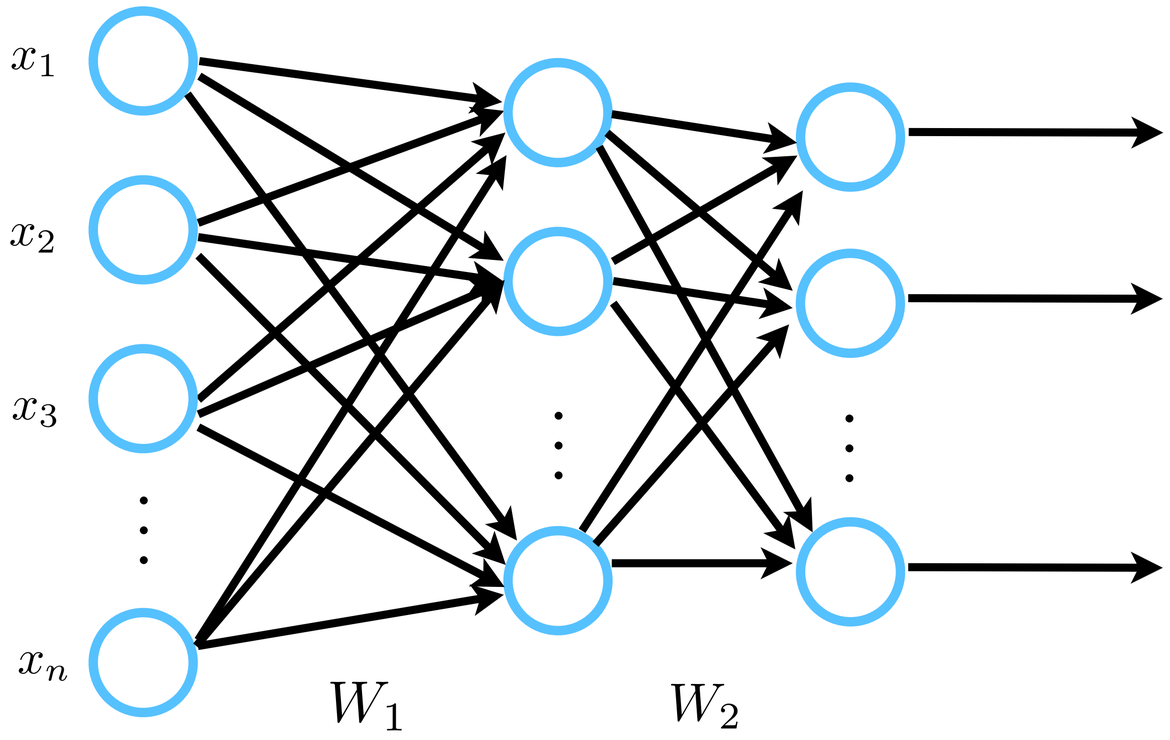
- Even for this module, there are alternatives: sparsemax, sparse projections onto the simplex, ..
- Out of scope

What are our options here wrt modules?



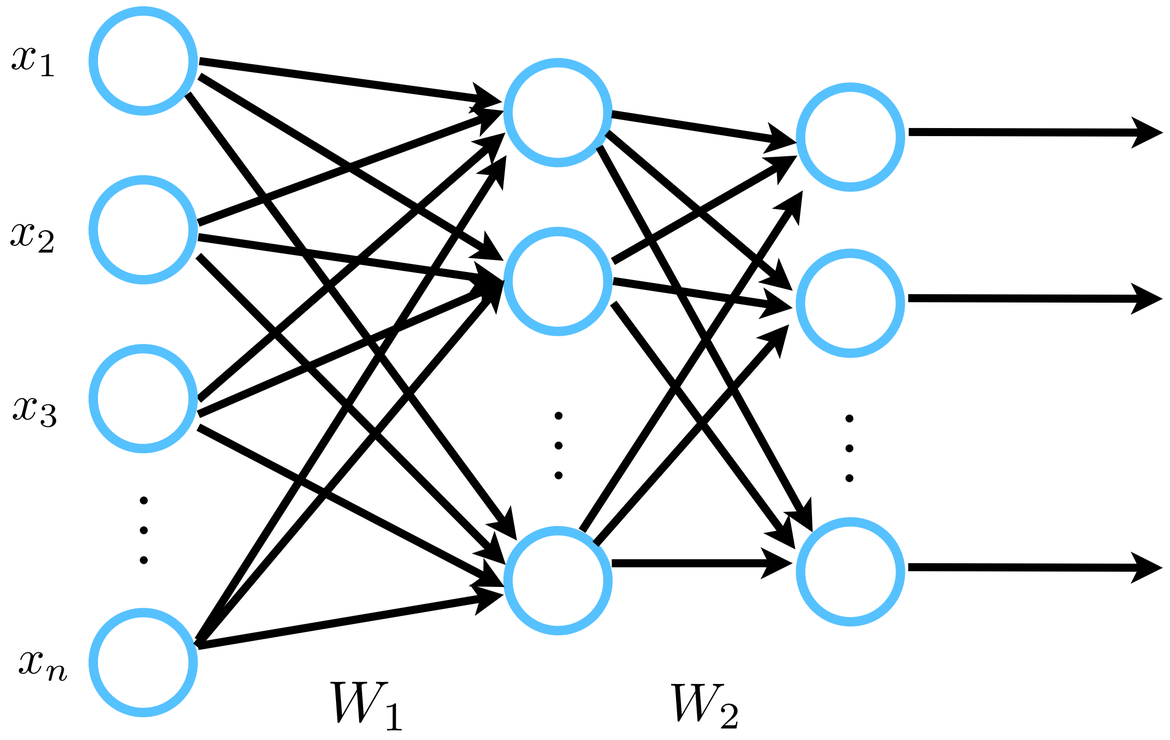
- Regularization modules: dropout and batch normalization
  - Dropout resembles to ensemble training: trains for subset of iterations a subset of the network – this creates diversity in decisions at the end, increasing accuracy

What are our options here wrt modules?



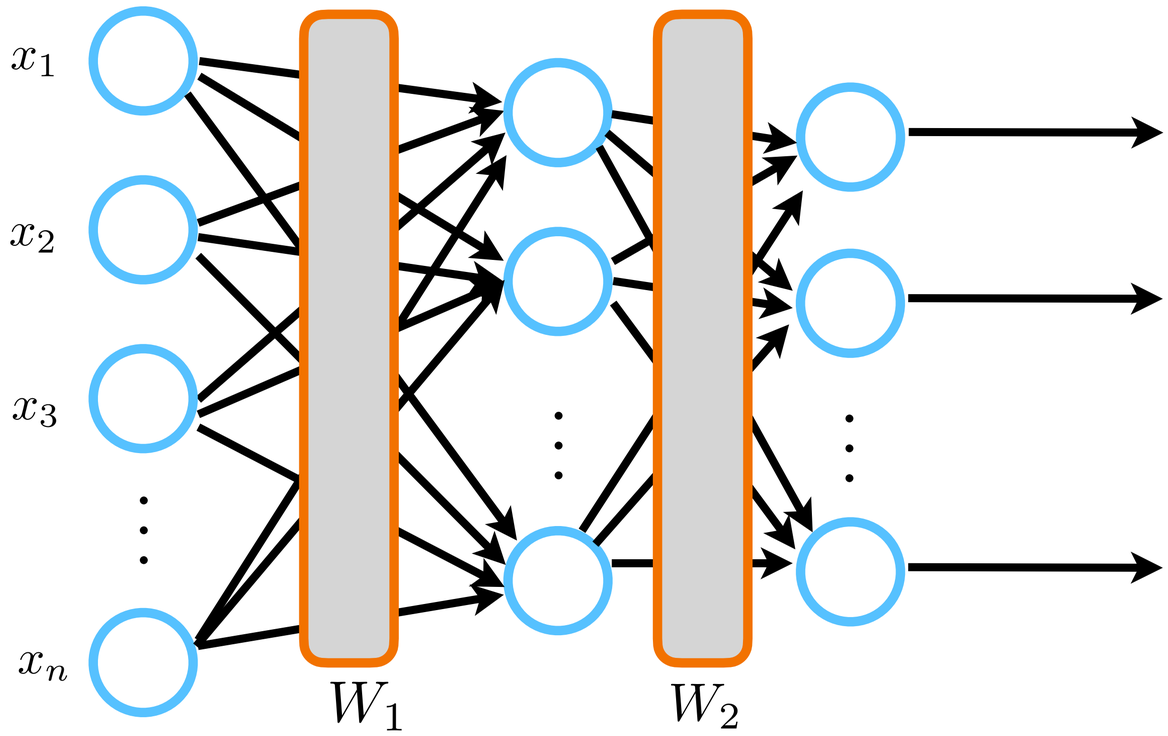
- Regularization modules: dropout and batch normalization
  - Dropout resembles to ensemble training: trains for subset of iterations a subset of the network – this creates diversity in decisions at the end, increasing accuracy

What are our options here wrt modules?



- Regularization modules: dropout and batch normalization
  - Dropout resembles to ensemble training: trains for subset of iterations a subset of the network – this creates diversity in decisions at the end, increasing accuracy

What are our options here wrt modules?



$$\mu_B = \frac{1}{B}\sum_{i=1}^{B} x_i$$

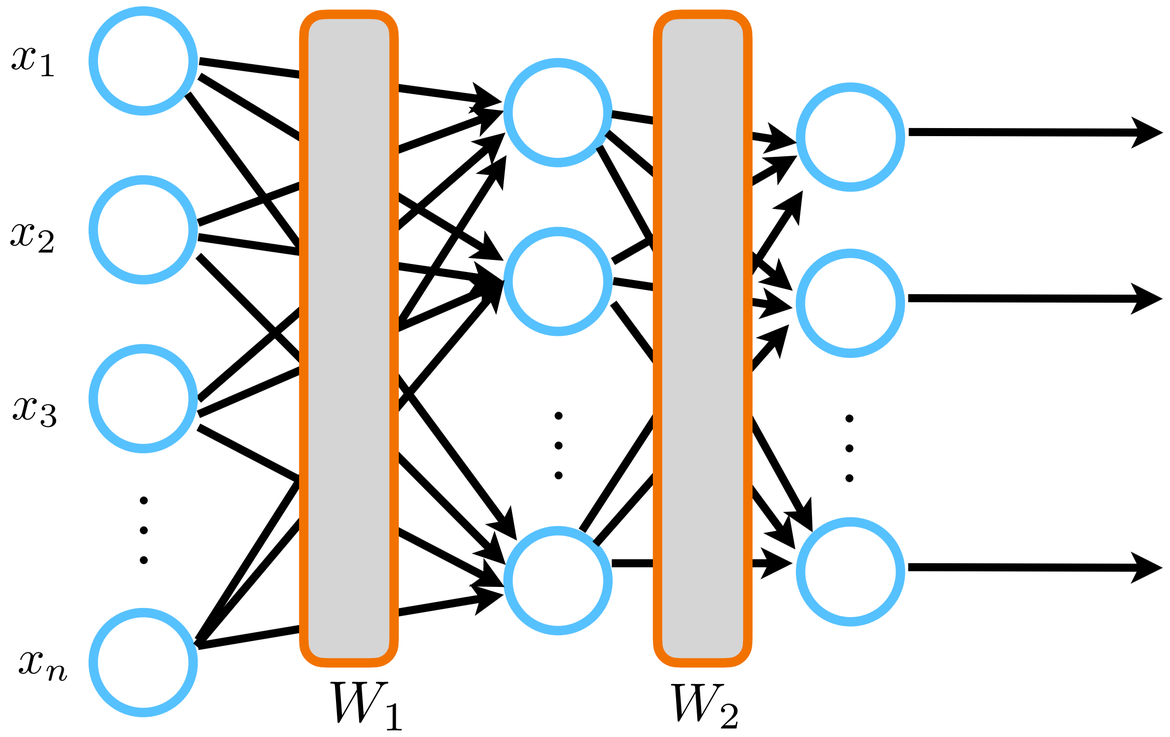
$$\sigma_B = \frac{1}{B}\sum_{i=1}^{} (x_i - \mu_B)^2$$

$$\widehat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$\widehat{y}_i = \gamma \cdot \widehat{x}_i + \beta$$

- Regularization modules: dropout and batch normalization
  - Dropout resembles to ensemble training: trains for subset of iterations a subset of the network – this creates diversity in decisions at the end, increasing accuracy
  - Batch normalization makes sure that inputs to layers behave like “whitened data”.

What are our options here wrt modules?



$$\mu_B = \frac{1}{B}\sum_{i=1}^{B} x_i$$

$$\sigma_B = \frac{1}{B}\sum_{i=1}(x_i - \mu_B)^2$$

$$\widehat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

$$\widehat{y}_i = \gamma \cdot \widehat{x}_i + \beta$$

Both techniques increase significantly the performance of a neural network (e.g., BN allows larger step sizes)

- Regularization modules: dropout and batch normalization
  - Dropout resembles to ensemble training: trains for subset of iterations a subset of the network – this creates diversity in decisions at the end, increasing accuracy
  - Batch normalization makes sure that inputs to layers behave like “whitened data”.

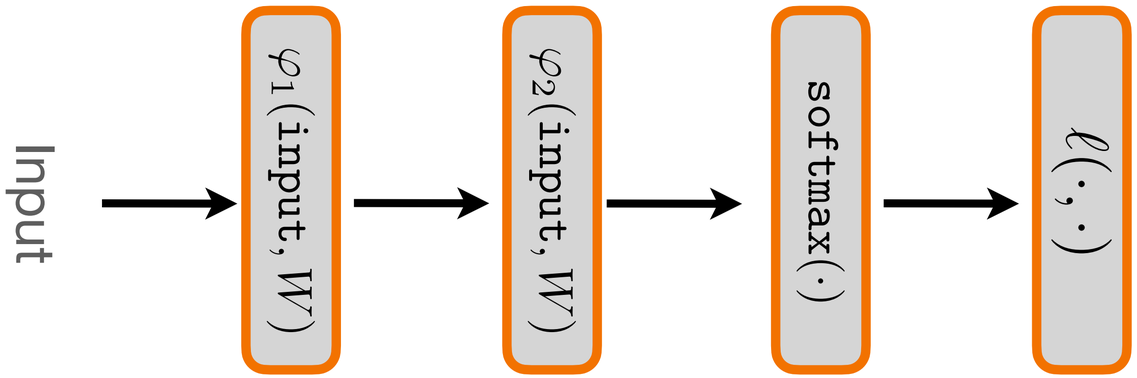
“But how do we train a neural network?”

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n} \sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$
$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$



$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial \varphi(\cdot, W_2)} \cdot \frac{\partial \varphi(\cdot, W_2)}{\partial W} \\
&= \dots
\end{aligned}$$

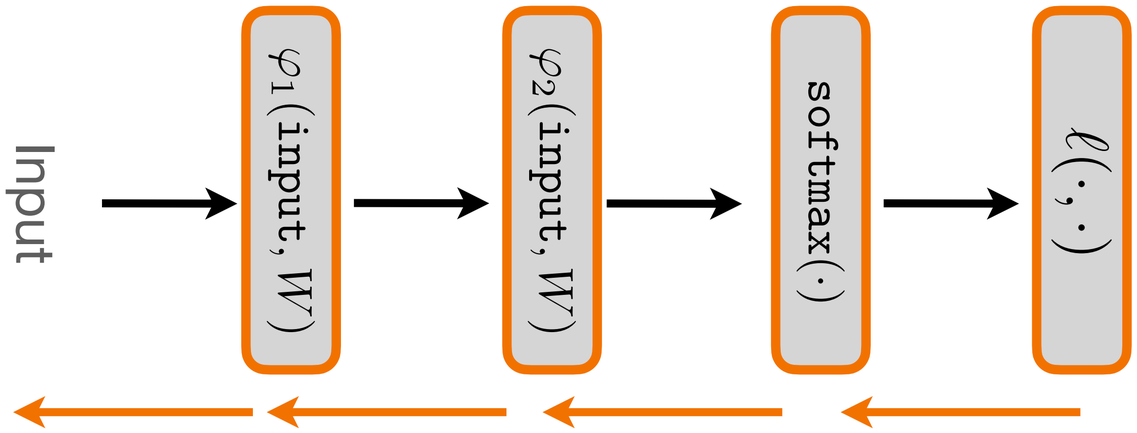
(chain rule of derivatives)

Motivation: Gradient descent for MLPs

$$\min_{W_i} f(W_1, W_2) := \frac{1}{n} \sum_{i=1}^{n} \ell(\widehat{y}_i,\ y_i) \quad \text{where} \quad \widehat{y}_i = \texttt{softmax}\left(\sigma\left(W_2 \cdot \sigma\left(W_1 \cdot x_i\right)\right)\right)$$

– Using modules:

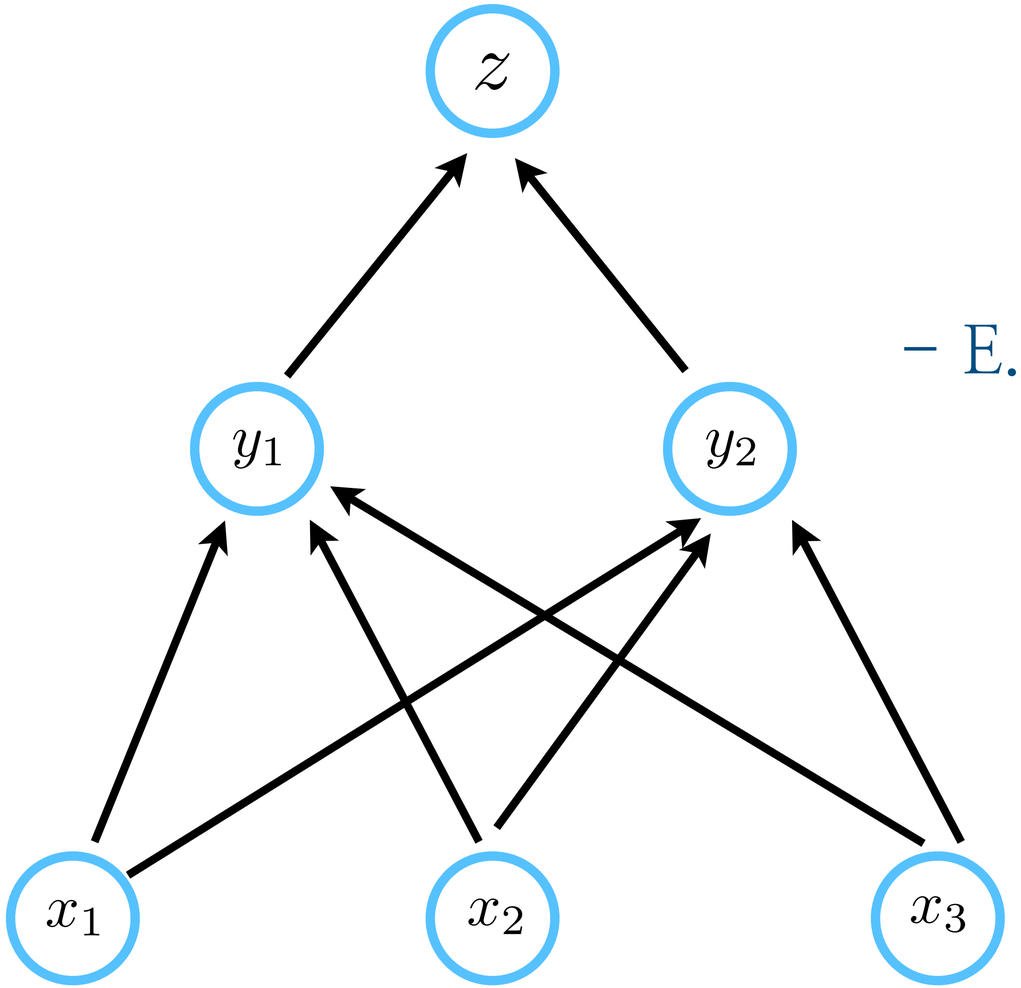
$\varphi_1(\texttt{input}, W) = \sigma(W_1 \cdot x_i)$
$\varphi_2(\texttt{input}, W) = \sigma(W_2 \cdot \varphi_2(\cdot))$



(backward pass on modules!)

$$\begin{aligned}
\nabla f_i(x) &= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial W} \\
&= \frac{\partial \ell(y_i, \widehat{y}_i)}{\partial \texttt{softmax}(\cdot)} \cdot \frac{\partial \texttt{softmax}(\cdot)}{\partial \varphi(\cdot, W_2)} \cdot \frac{\partial \varphi(\cdot, W_2)}{\partial W} \\
&= \cdots
\end{aligned}$$

(chain rule of derivatives)



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

– E.g.,

$$\frac{\partial z}{\partial x_3} = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_3} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_3}$$

(Follow all relevant paths)

Backpropagation = Gradient descent

(Just done efficiently on graphs, without redoing calculations)

Overview

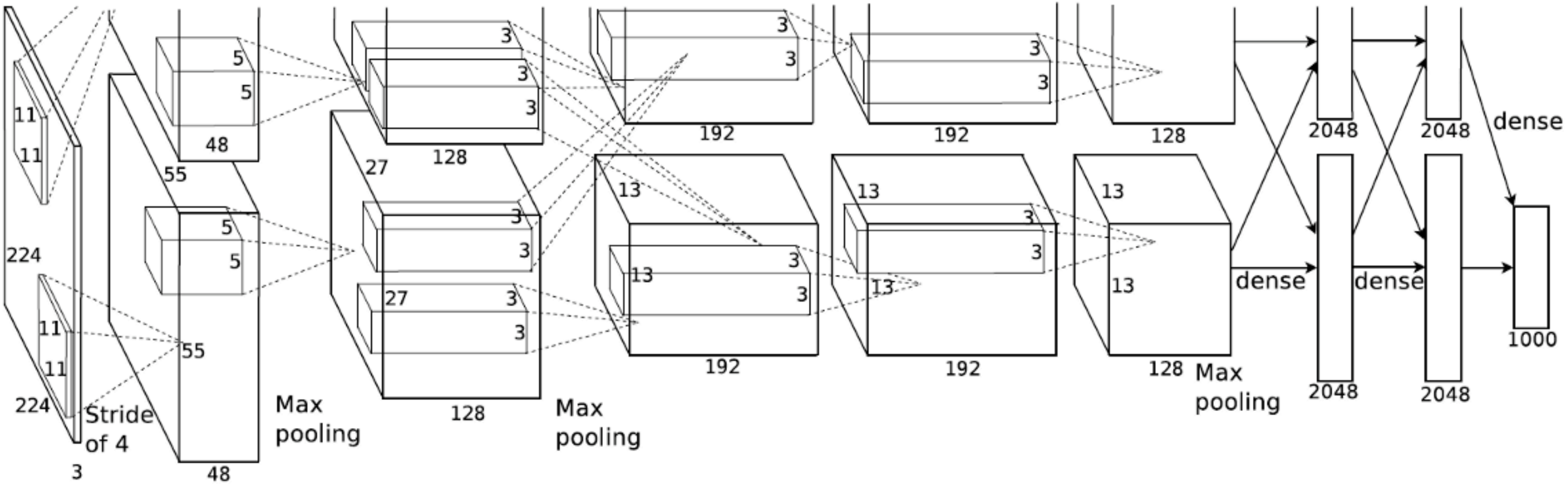
- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)

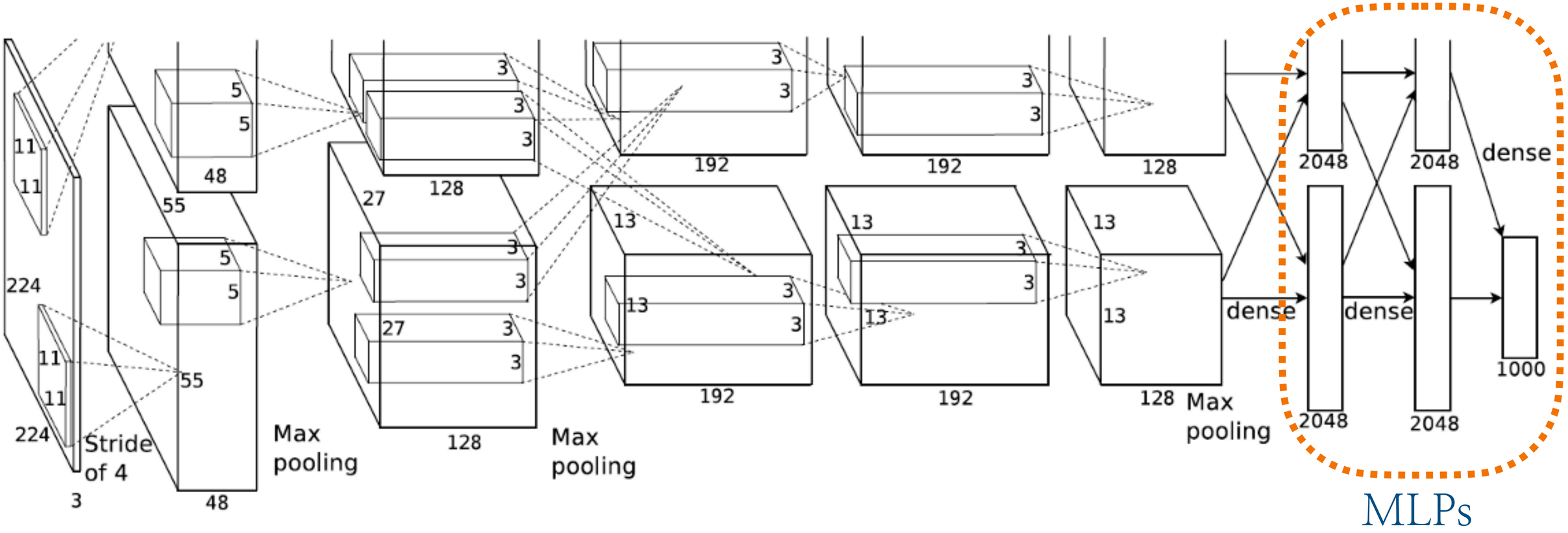
Spoiler alert

AlexNet – Winner of ImageNet 2012 competition



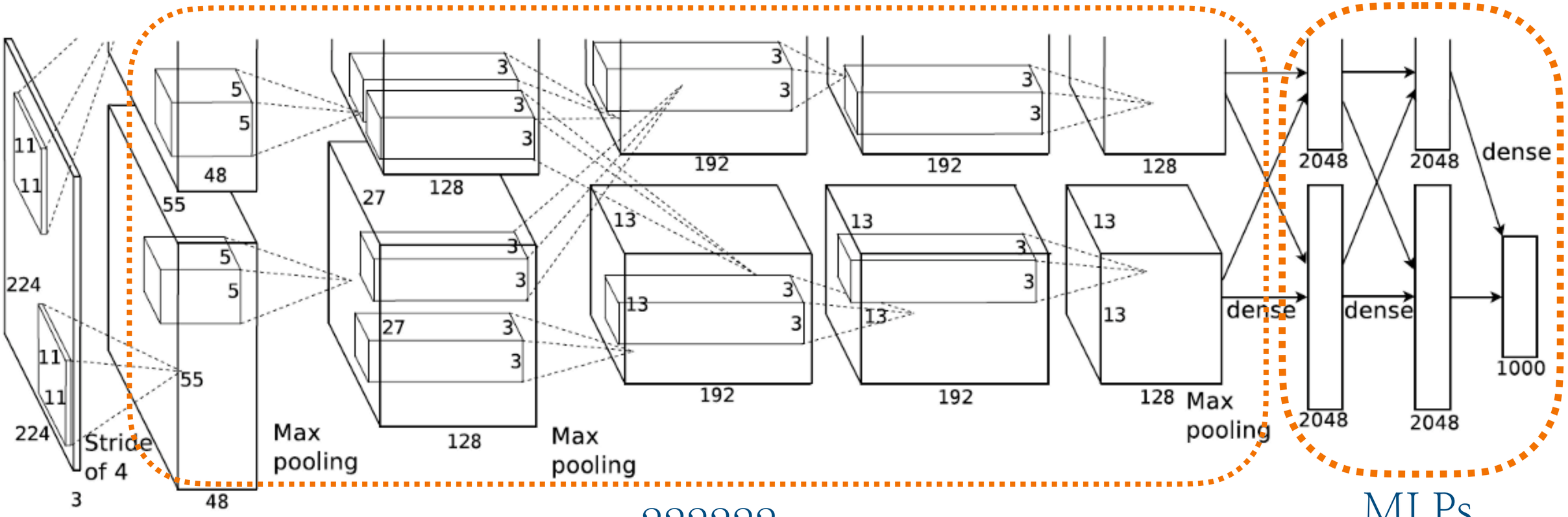
Spoiler alert

AlexNet – Winner of ImageNet 2012 competition



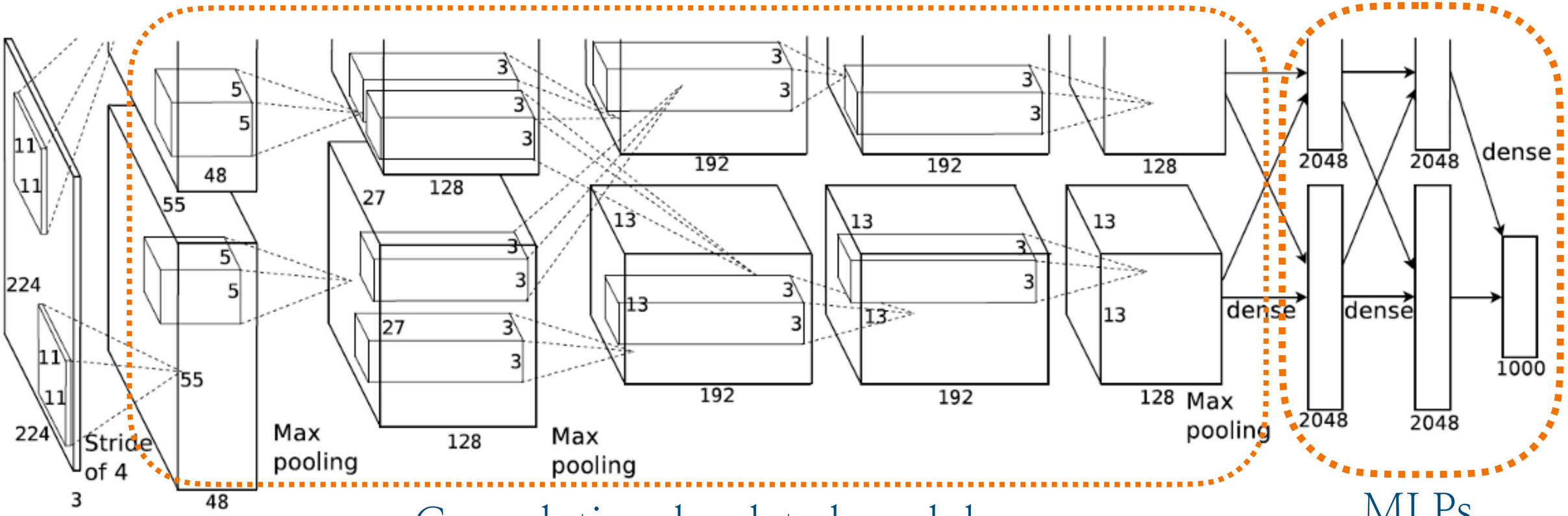
Spoiler alert

AlexNet – Winner of ImageNet 2012 competition



Spoiler alert

AlexNet – Winner of ImageNet 2012 competition



Convolutional-related modules

MLPs

Convolutional neural networks (CNNs)

- Another type of neural networks

Convolutional neural networks (CNNs)

- Another type of neural networks
- Basic module: convolution

Convolutional neural networks (CNNs)

- Another type of neural networks
- Basic module: convolution
- Basic application: image-related/computer vision

Convolutional neural networks (CNNs)

- Another type of neural networks
- Basic module: convolution
- Basic application: image-related/computer vision
- Basic property: images (as an array of pixels) have strong spatial properties

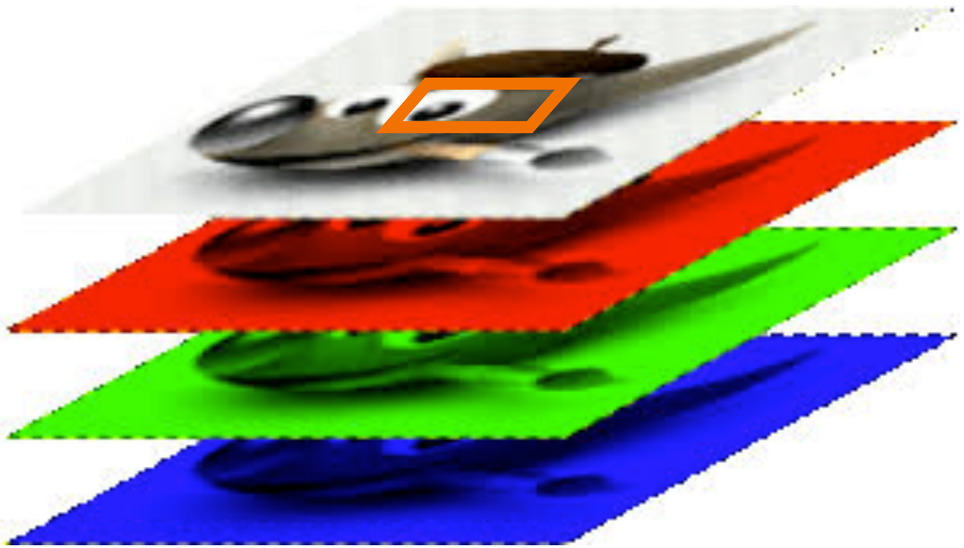
  Convolutions preserve such spatial structure and create relevant features

Convolutional neural networks (CNNs)

- Another type of neural networks
- Basic module: convolution
- Basic application: image-related/computer vision
- Basic property: images (as an array of pixels) have strong spatial properties

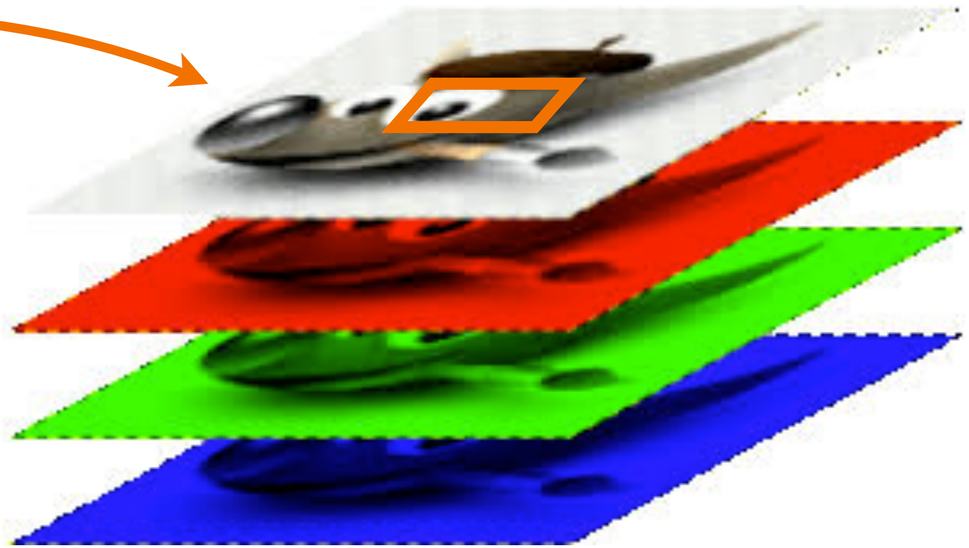
  Convolutions preserve such spatial structure and create relevant features
- Secondary but important module: pooling – robustifies against local variances

Images and their spatial properties



Images and their spatial properties

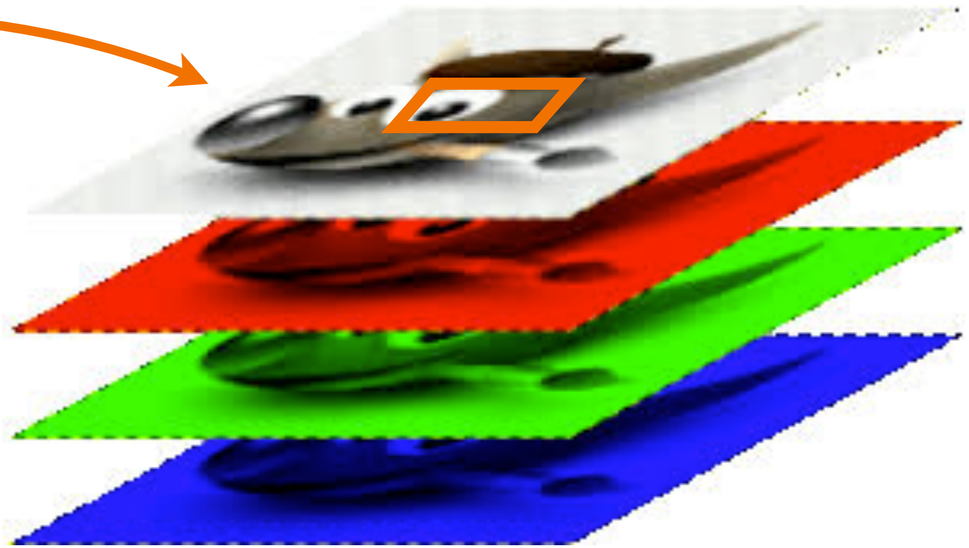
Original image



Images and their spatial properties

Original image

Decomposition
into 3 channels

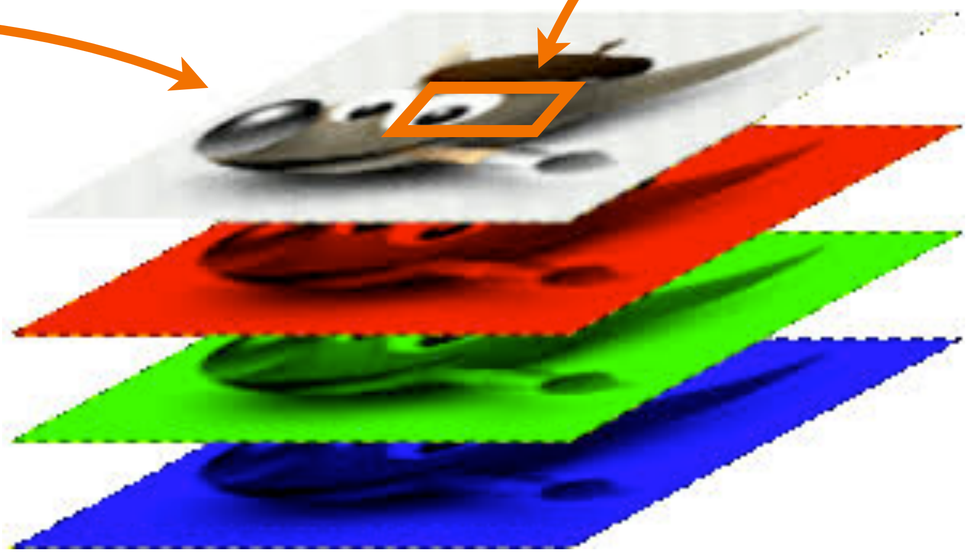


Images and their spatial properties

Neighboring pixels
are locally correlated

Original image

Decomposition
into 3 channels



Filters/Kernels

- Fairly old concept of extracting information from images

Filters/Kernels

- Fairly old concept of extracting information from images
- Example: Sobel filters

Filters/Kernels

- Fairly old concept of extracting information from images
- Example: Sobel filters

Horizontal

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Vertical

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Filters/Kernels

- Fairly old concept of extracting information from images
- Example: Sobel filters

Horizontal

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Vertical

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Applying Sobel kernels on original image = edge detection

Filters/Kernels

- Fairly old concept of extracting information from images
- Example: Sobel filters

Horizontal

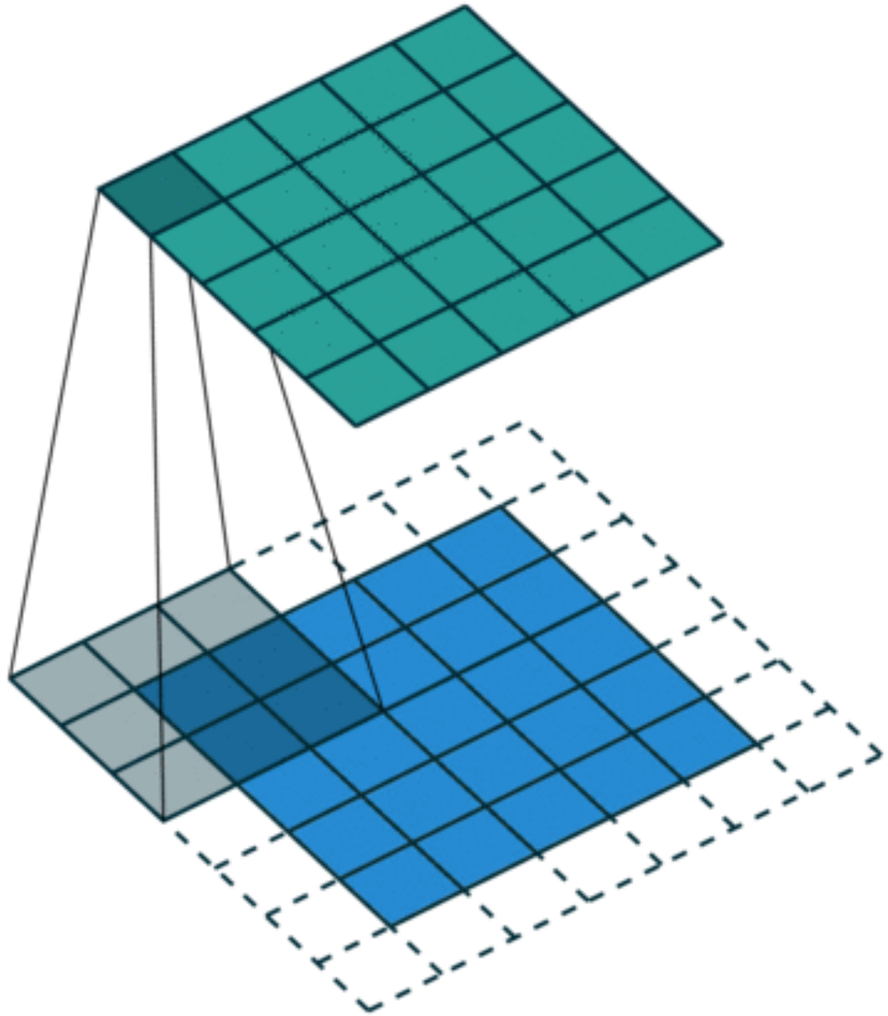
$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Vertical

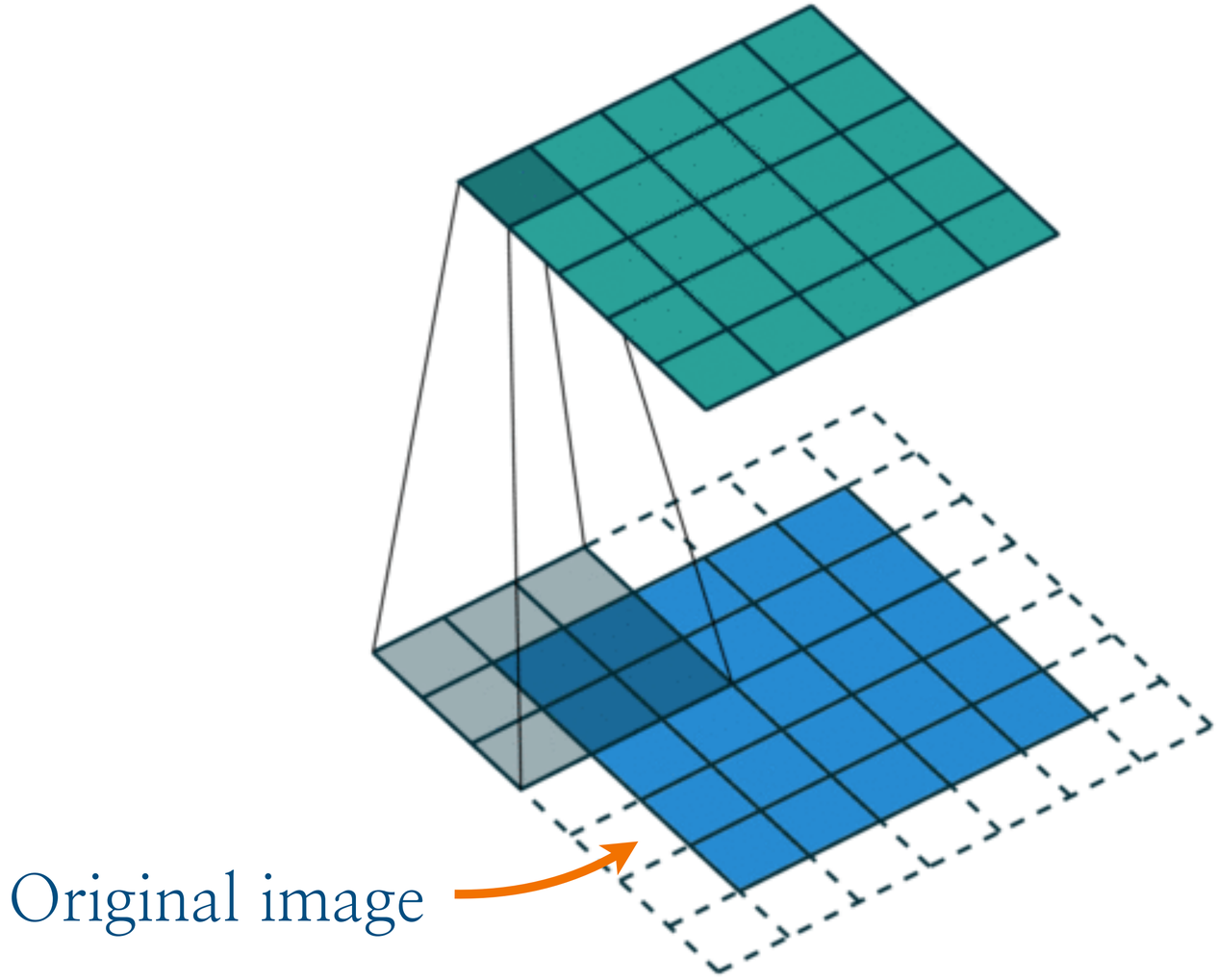
$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Applying Sobel kernels on original image = edge detection

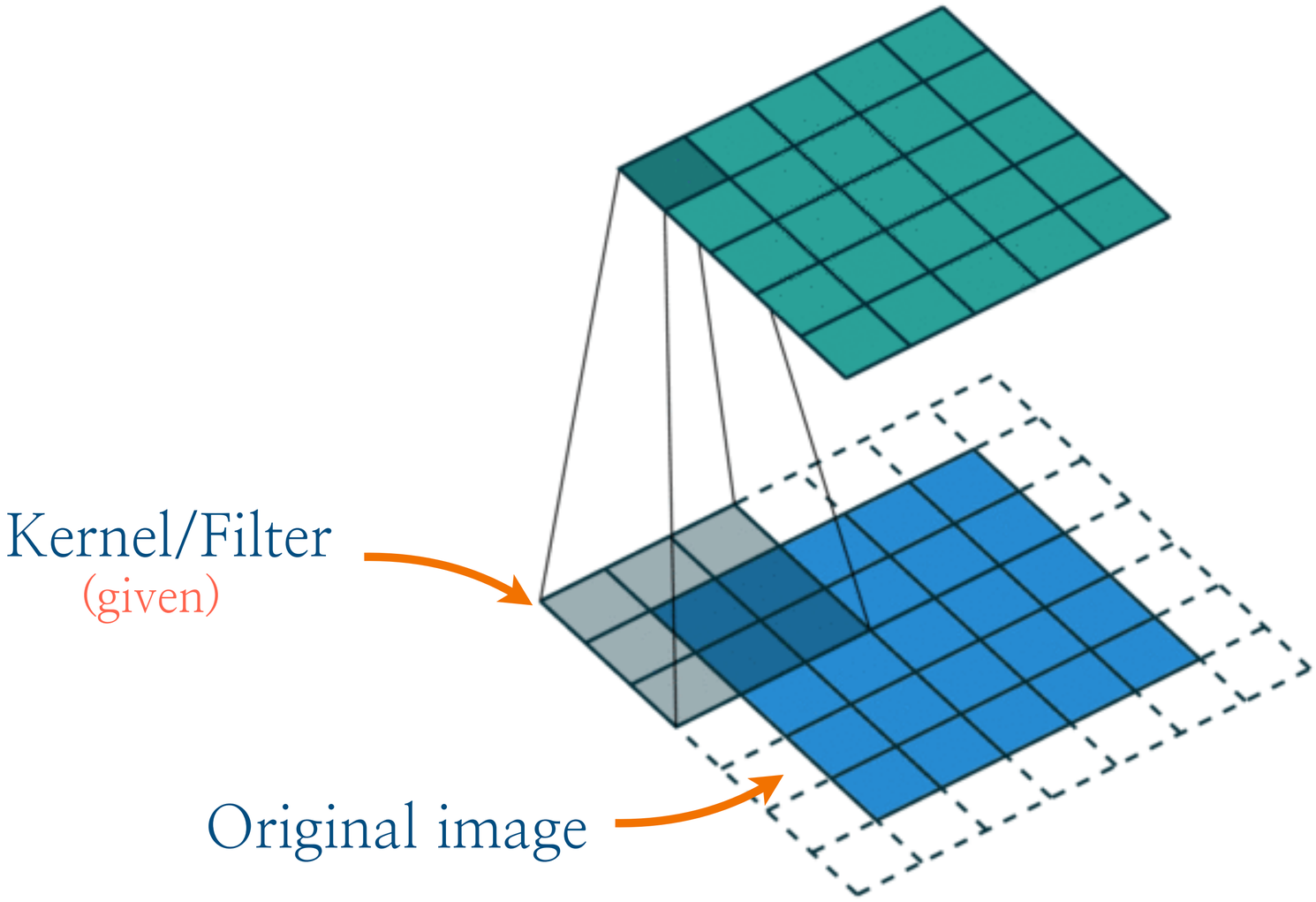
Convolution operation: Basic operation



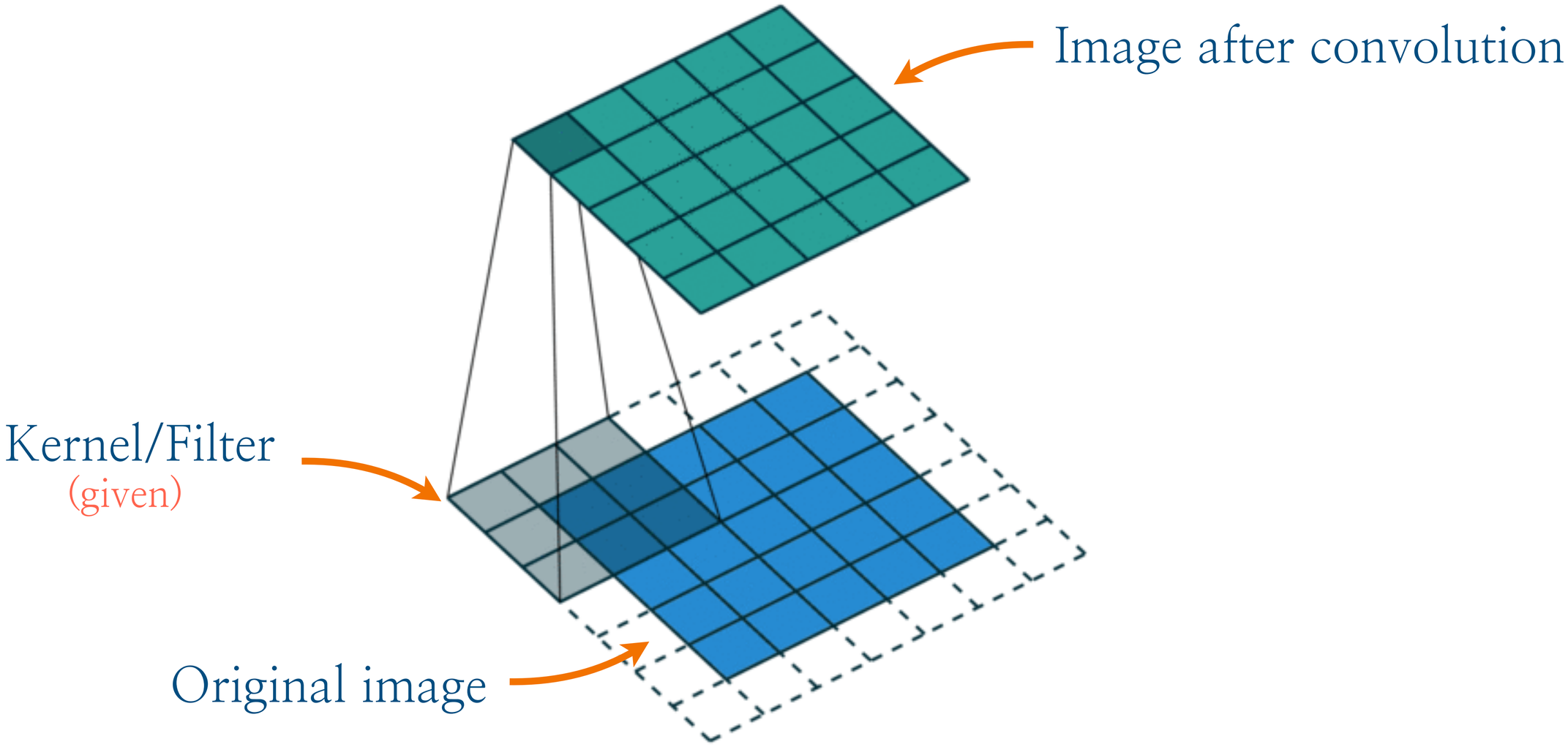
Convolution operation: Basic operation



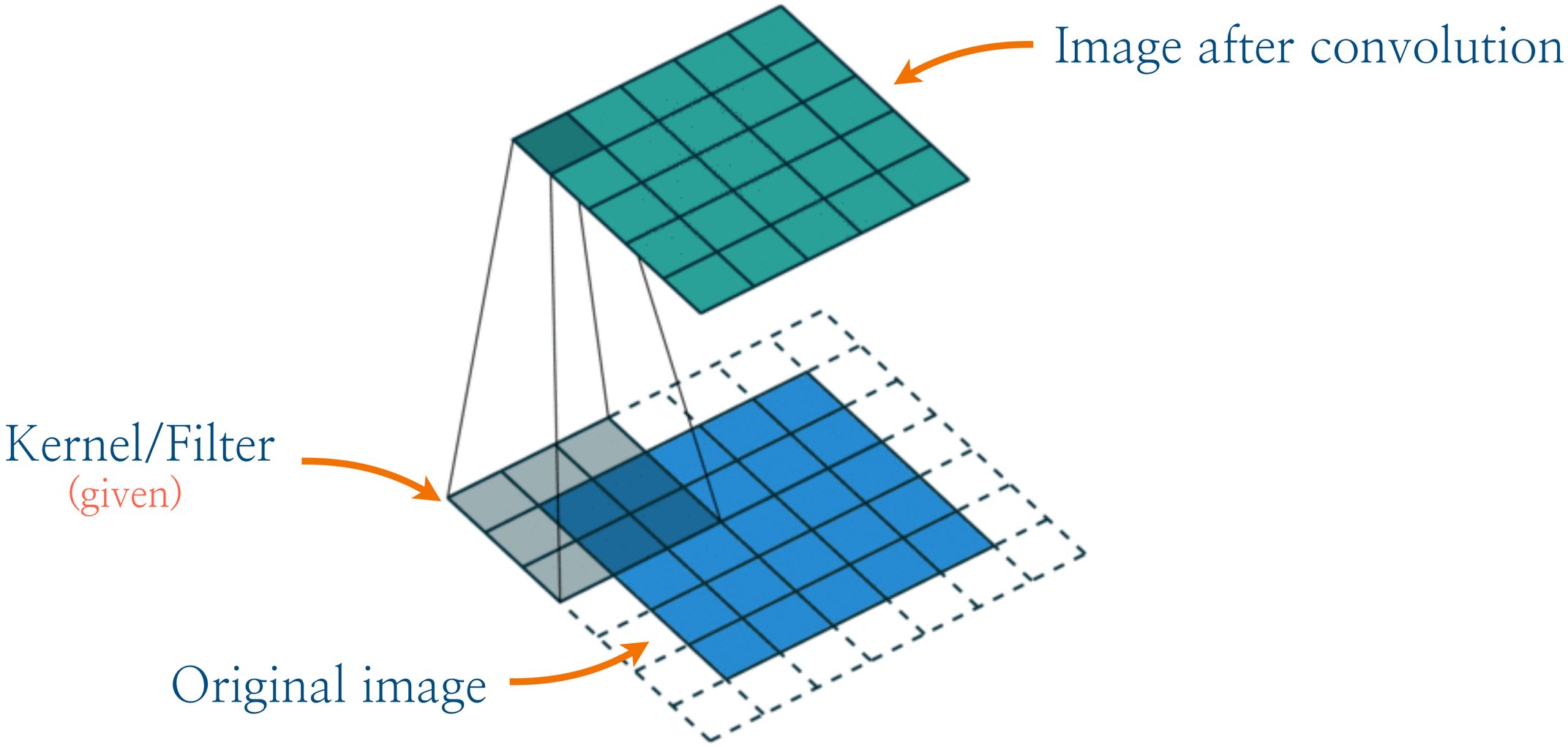
Convolution operation: Basic operation



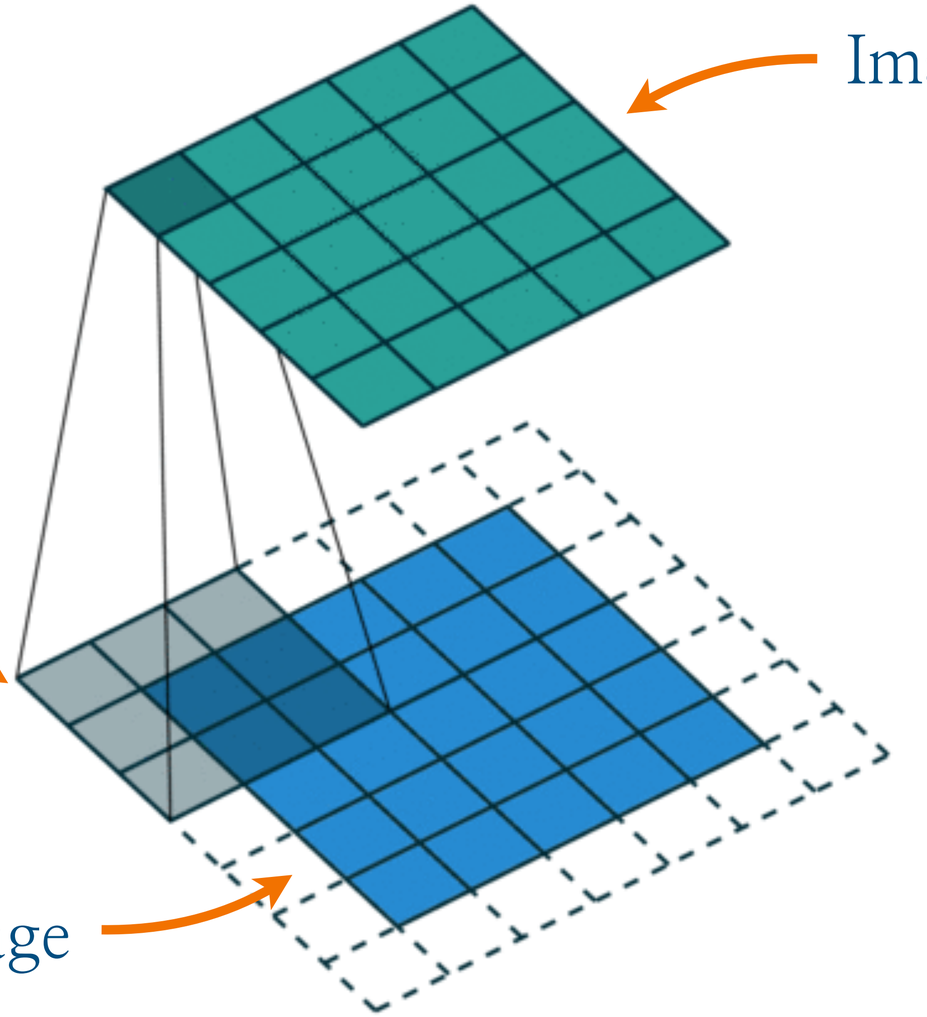
Convolution operation: Basic operation



Convolution operation: Basic operation

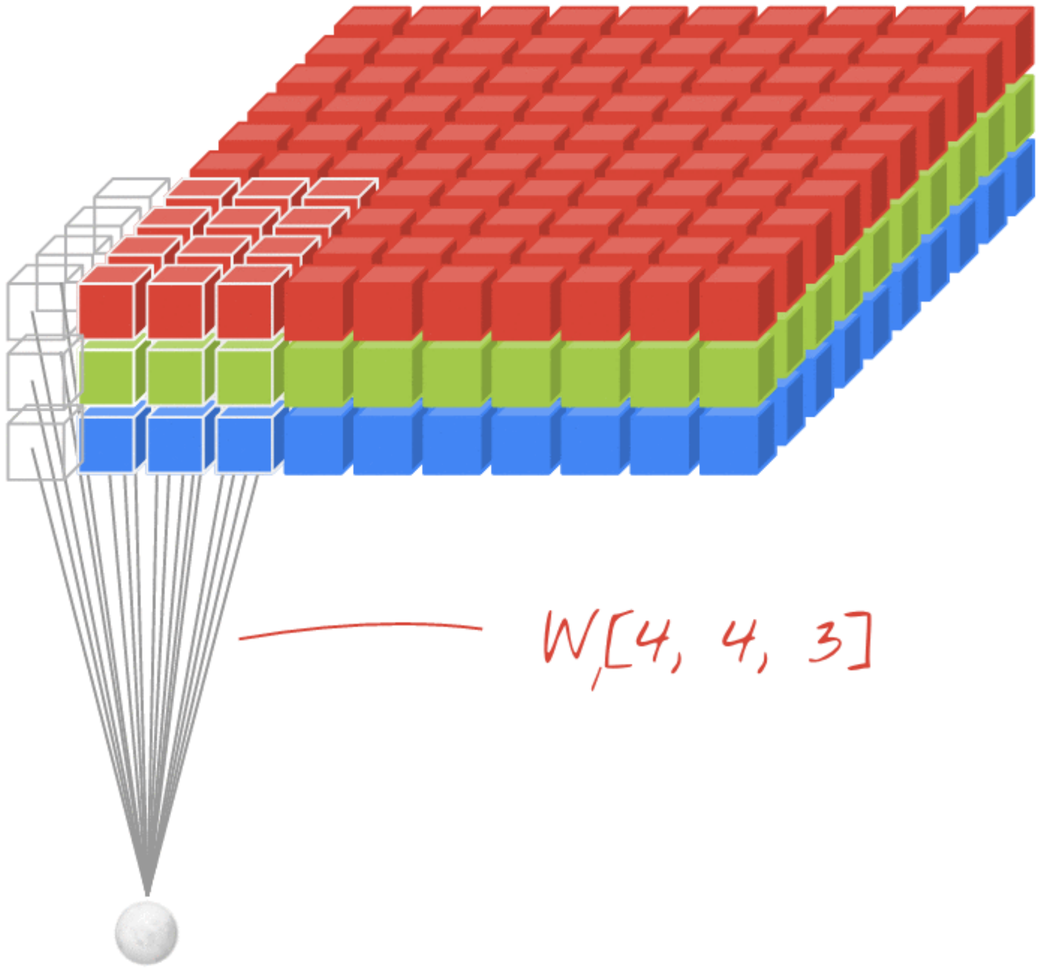


Convolution operation: Basic operation

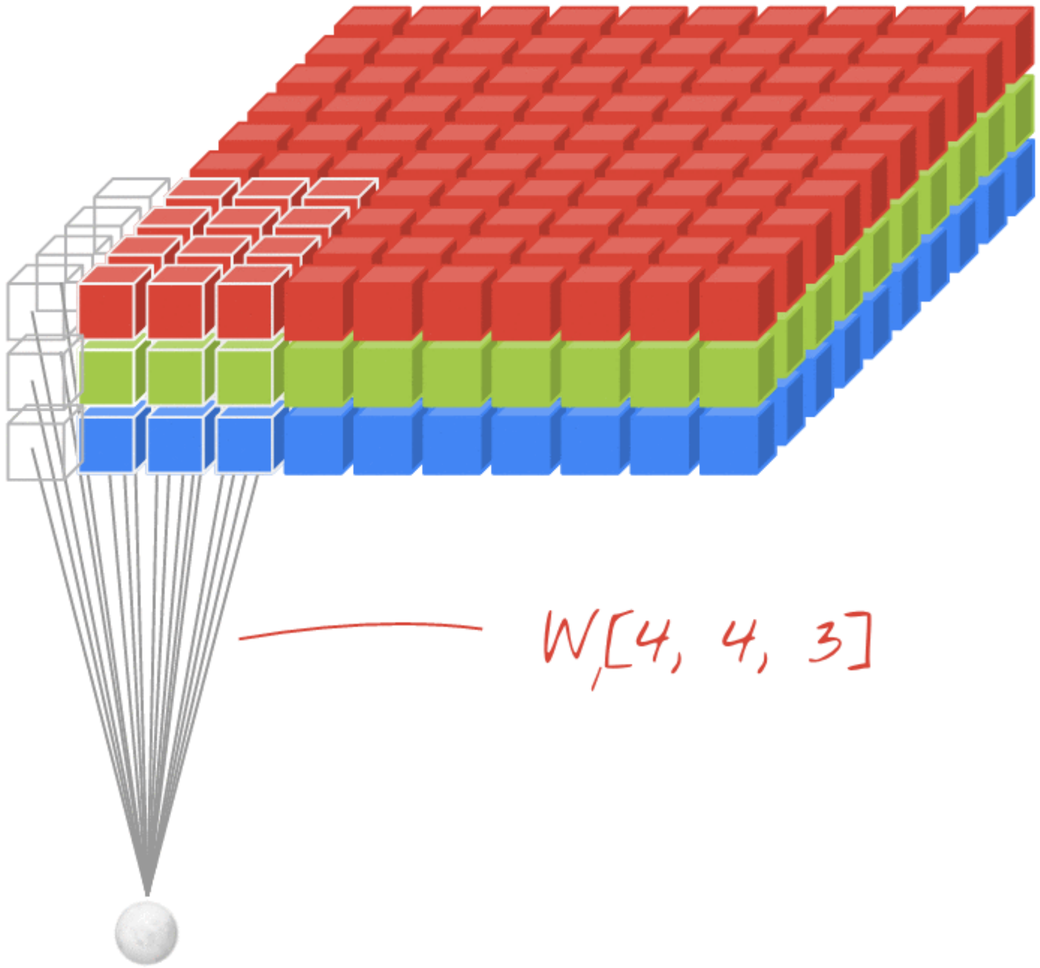


Input				Kernel			Output	
0	1	2	*	0	1	=	19	25
3	4	5		2	3		37	43
6	7	8						

Convolution operation: Channels and multiple filters



Convolution operation: Channels and multiple filters



Convolution operation

- Fairly old concept of extracting information from images
- Example: Sobel filters

Horizontal

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Vertical

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Applying Sobel kernels on original image = edge detection

Convolution operation

"Handcrafted filters/kernels"
(Sobel, Gaussian, smoothness, Gabor, etc.)

- Fairly old concept of extracting information from images
- Example: Sobel filters

Horizontal

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Vertical

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Applying Sobel kernels on original image = edge detection

Convolution operation

"Handcrafted filters/kernels"
(Sobel, Gaussian, smoothness, Gabor, etc.)

- Fairly old concept of extracting information from images
- Example: Sobel filters

Are they optimal for all computer vision applications?

Horizontal

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Vertical

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Applying Sobel kernels on original image = edge detection

Learnable kernels

Learnable kernels

- Instead of handcrafted kernels..

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Learnable kernels

- Instead of handcrafted kernels..

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

.. we try to learn one:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

..based on our application.

Learnable kernels

- Instead of handcrafted kernels..

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

.. we try to learn one:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

..based on our application.

- Convolution modules are fully differentiable!

Learnable kernels

- Instead of handcrafted kernels..

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

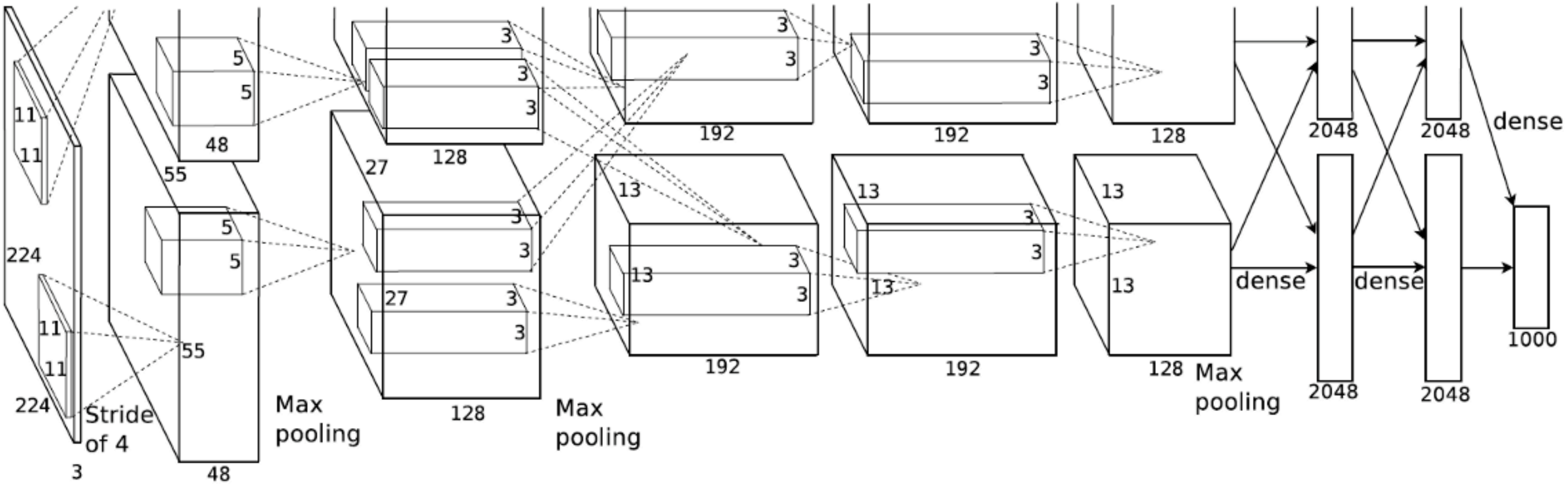
.. we try to learn one:

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

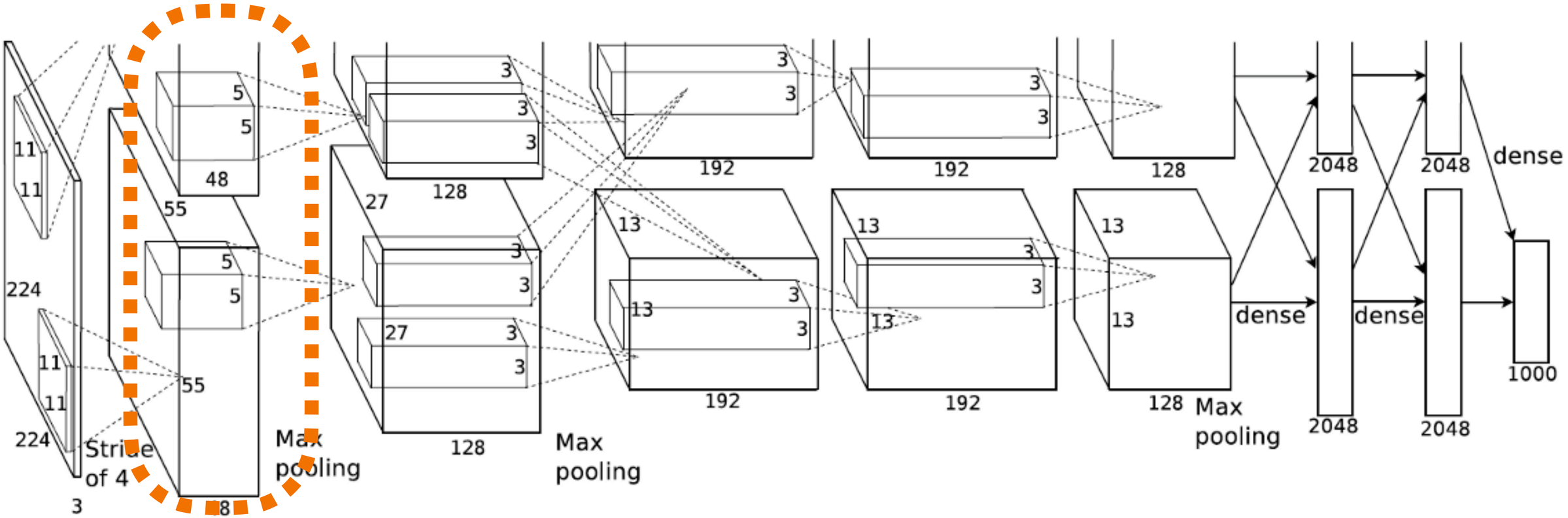
..based on our application.

- Convolution modules are fully differentiable!
- Good practices: Stride 1, 3x3 or 5x5 kernels work well

AlexNet

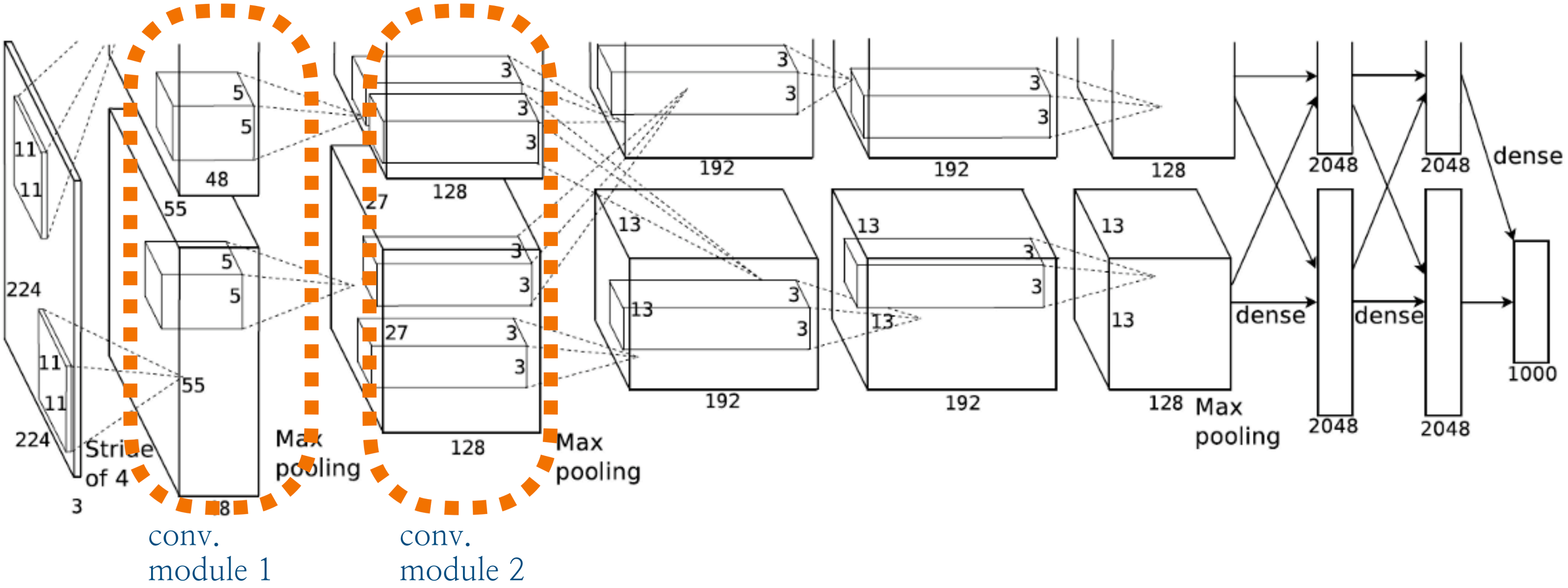

11
11
224
224
3
Stride
of 4
5
5
48
55
48
5
5
55
Max
pooling
3
3
3
128
27
128
3
3
3
27
128
Max
pooling
3
3
192
3
3
3
192
128
13
13
3
3
192
13
13
3
3
192
13
13
128
Max
pooling
dense
dense
2048
2048
2048
2048
dense
1000

AlexNet

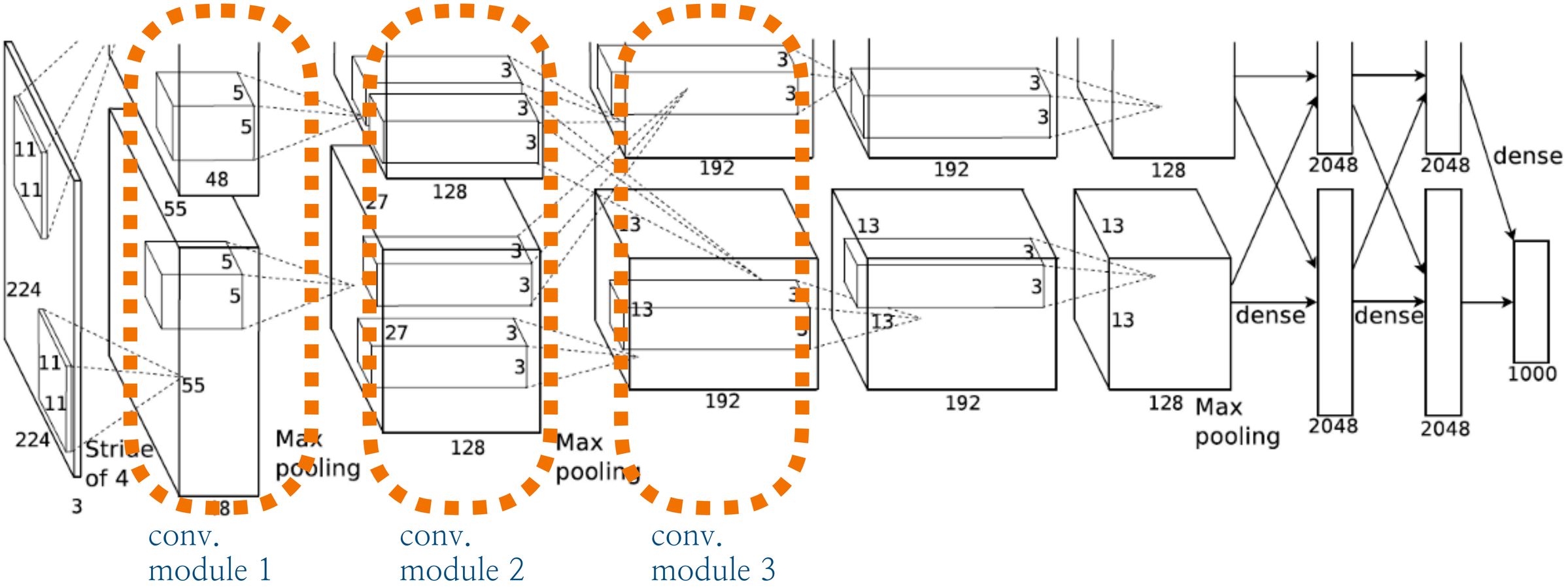


conv. module 1

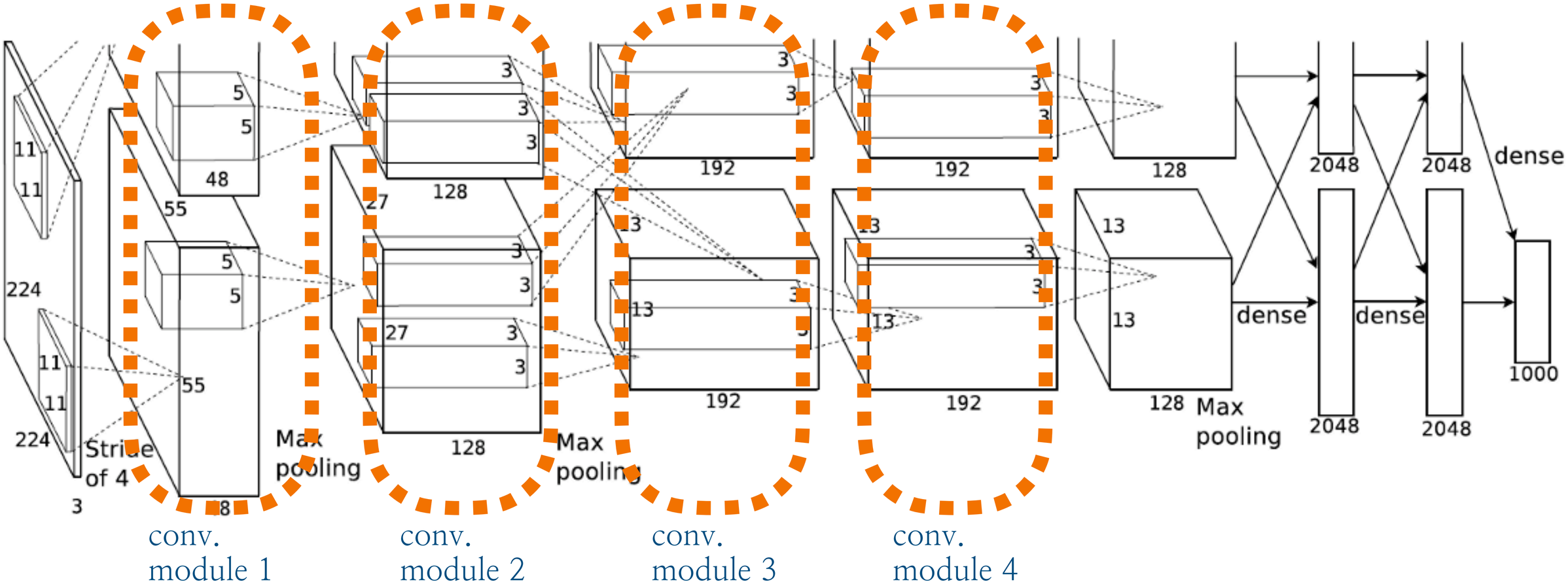
AlexNet


11
11
224
224
3
Stride of 4
55
48
5
5
55
5
5
27
128
3
3
3
27
128
3
3
3
Max pooling
Max pooling
192
192
128
3
3
3
3
3
3
13
13
13
13
13
13
192
192
128
3
3
3
3
3
3
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000
conv. module 1
conv. module 2

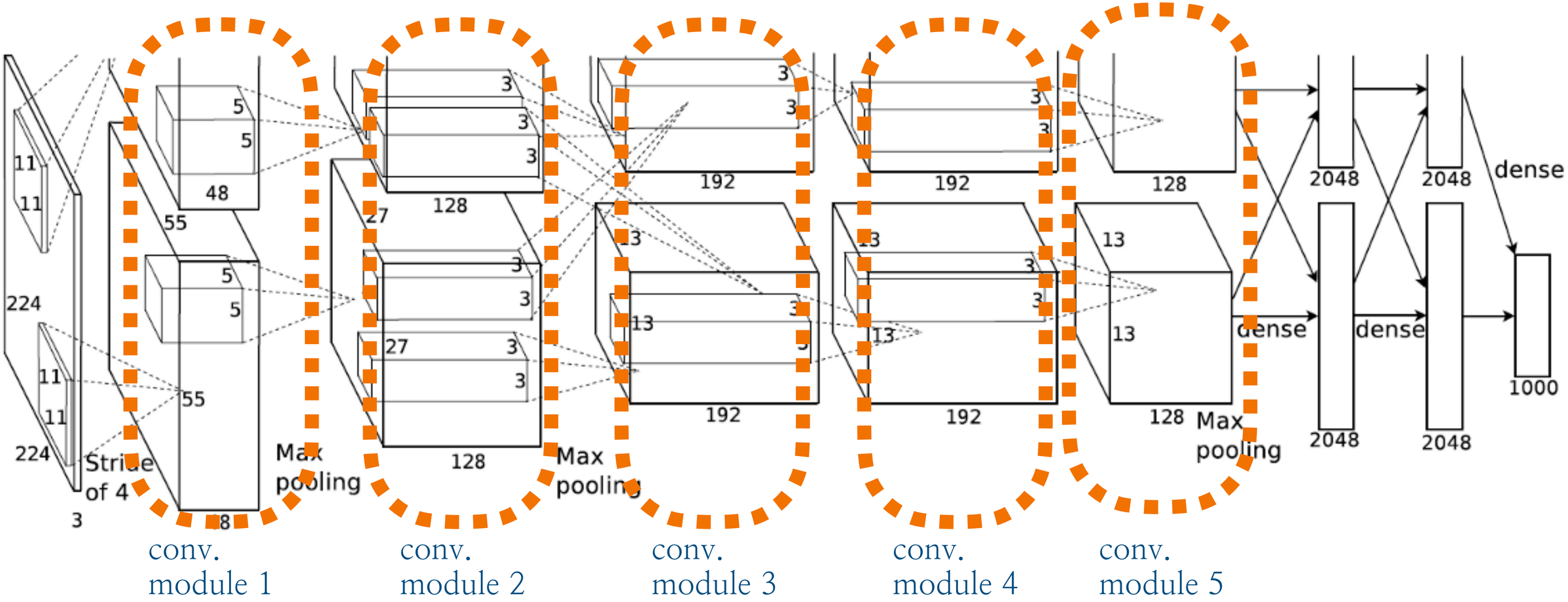
AlexNet


11
11
224
224
3
Stride of 4
55
48
5
5
55
48
5
5
Max pooling
27
128
3
3
3
27
128
3
3
3
Max pooling
13
192
3
3
3
13
192
3
3
3
192
3
3
3
192
3
3
3
128
13
13
Max pooling
dense
2048
2048
2048
2048
dense
dense
1000
conv. module 1
conv. module 2
conv. module 3

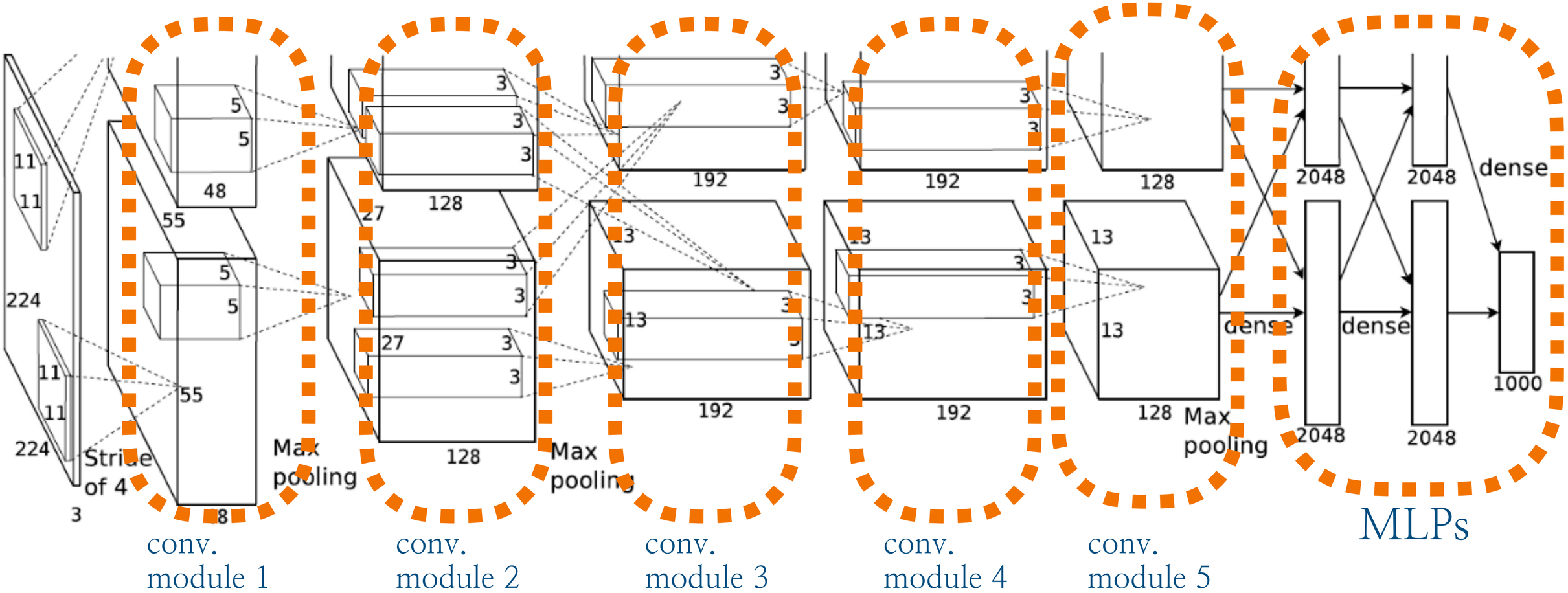
AlexNet


11
11
224
224
3
Stride of 4
5
5
48
55
5
5
55
8
Max pooling
3
3
3
128
27
3
3
27
3
3
128
Max pooling
192
192
13
13
192
192
128
13
13
128
Max pooling
dense
dense
2048
2048
2048
2048
dense
1000
conv. module 1
conv. module 2
conv. module 3
conv. module 4

AlexNet


11
11
224
224
3
Stride of 4
5
5
48
55
5
5
55
48
Max pooling
3
3
3
128
27
3
3
27
3
3
128
Max pooling
3
3
192
13
3
3
13
192
3
3
192
13
3
3
13
192
128
13
13
128
Max pooling
dense
dense
2048
2048
2048
2048
dense
1000
conv. module 1
conv. module 2
conv. module 3
conv. module 4
conv. module 5

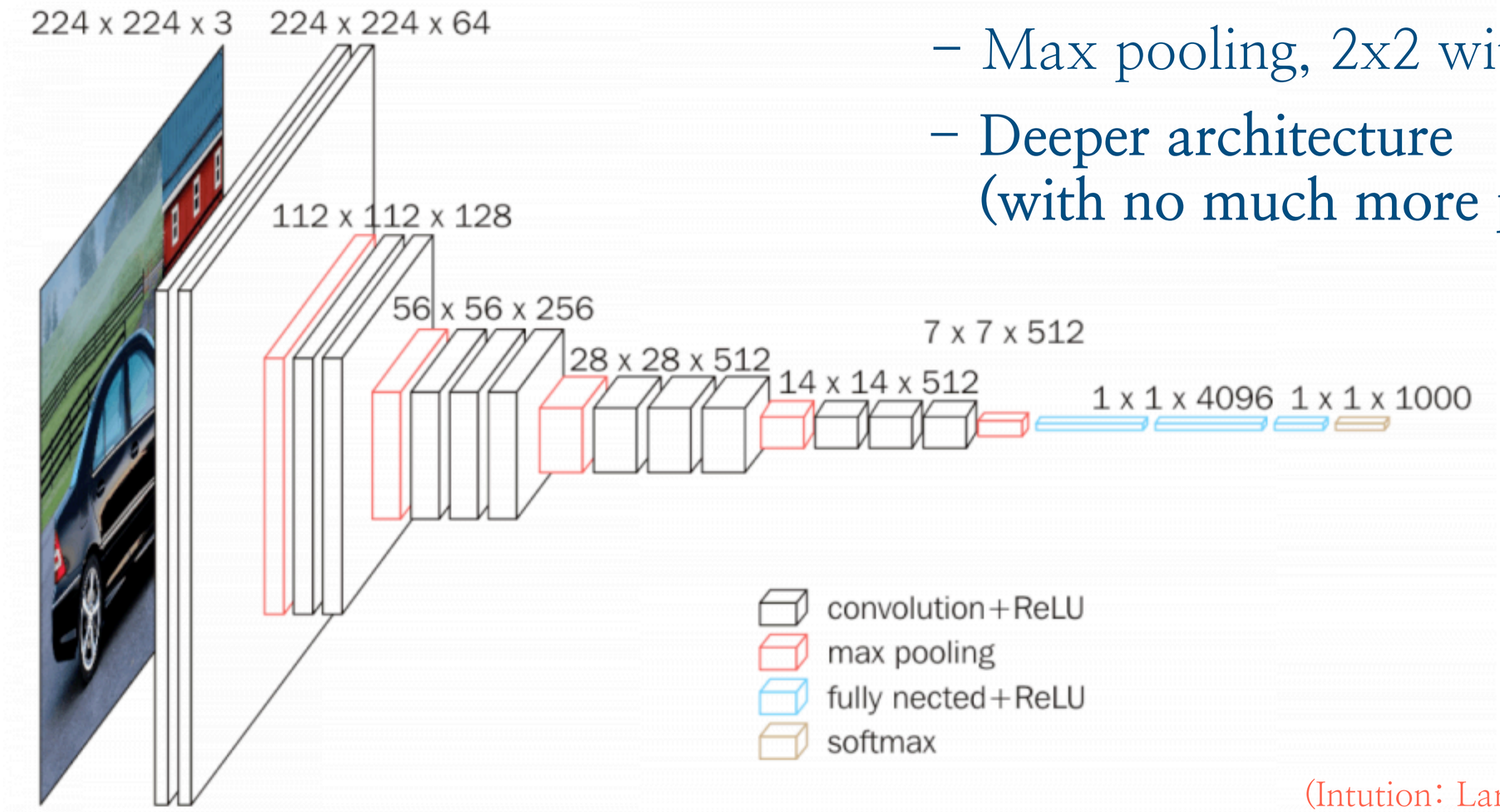
AlexNet


11
11
224
224
3
Stride of 4
5
5
48
55
5
5
55
Max pooling
3
3
3
128
27
3
3
27
3
3
128
Max pooling
3
3
192
13
3
3
13
192
3
3
192
13
3
3
13
192
128
13
13
128
Max pooling
dense
dense
2048
2048
2048
2048
dense
1000
conv. module 1
conv. module 2
conv. module 3
conv. module 4
conv. module 5
MLPs

VGG architecture

VGG16

- Only 3x3 filters, Stride of 1
- Max pooling, 2x2 with stride of 2
- Deeper architecture
  (with no much more parameters)

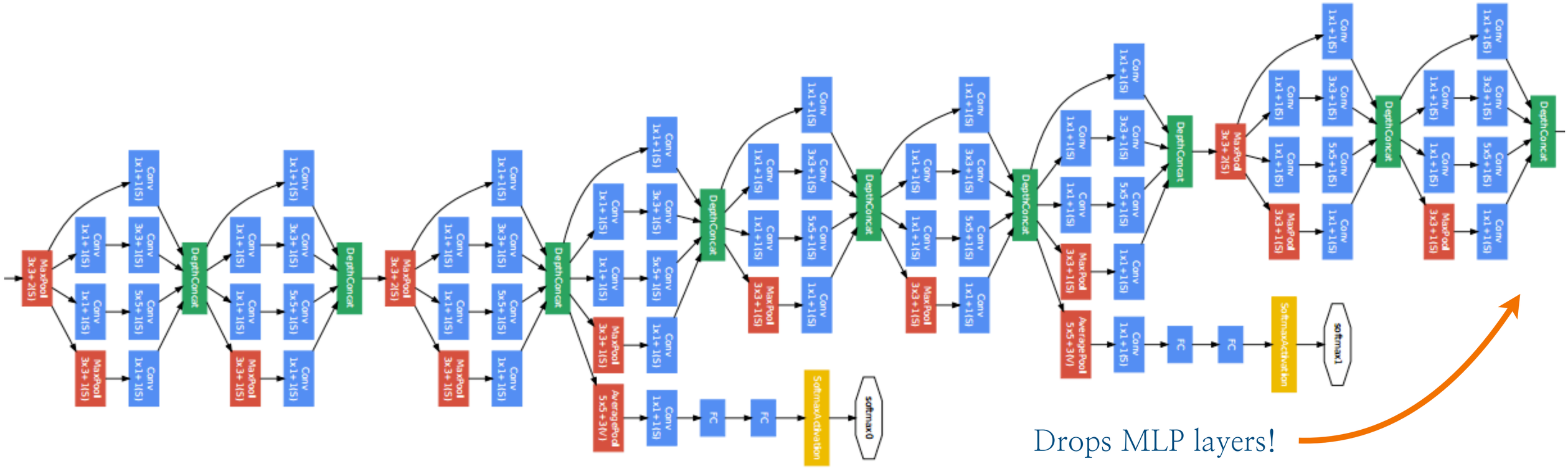


(Intution: Large filters = deep stack of smaller filters: this provides more "power")

Inception

Google's CV architecture

- ImageNet: 93.3% accuracy – compare with 81.8% AlexNet



Residual Networks (ResNets)

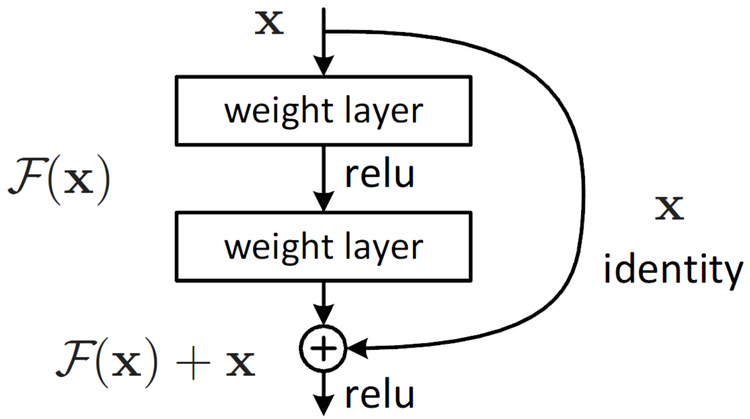
- Architecture that allows truly deep networks via skip connections

Residual Networks (ResNets)

- Architecture that allows truly deep networks via skip connections
- Skip connections provide the option to learn the identity map from layers to layers (could tackle vanishing gradients issue)

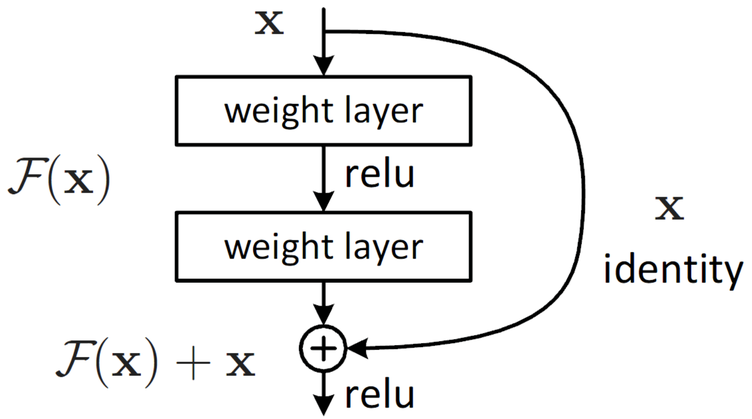
Residual Networks (ResNets)

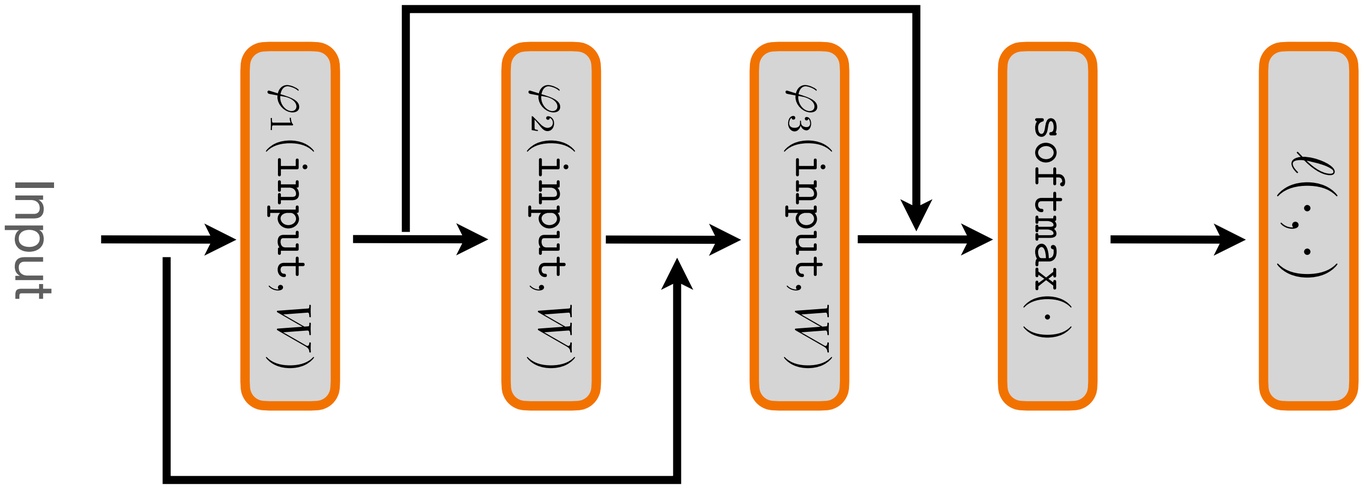
- Architecture that allows truly deep networks via skip connections
- Skip connections provide the option to learn the identity map from layers to layers (could tackle vanishing gradients issue)



Residual Networks (ResNets)

- Architecture that allows truly deep networks via skip connections
- Skip connections provide the option to learn the identity map from layers to layers (could tackle vanishing gradients issue)





Residual Networks (ResNets)

- Remarks
  - ImageNet accuracy: 96.4%!
  - Winning architecture in many other applications (default module/ strategy in many current architectures)

Residual Networks (ResNets)

- Remarks
  - ImageNet accuracy: 96.4%!
  - Winning architecture in many other applications (default module/ strategy in many current architectures)
    - Nowadays, not exactly an architecture, but a module you can use in your neural networks
    - Several extensions (e.g., ResNext, HighwayNet, DenseNet)

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)
- Some trends in deep learning

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)
- Some trends in deep learning

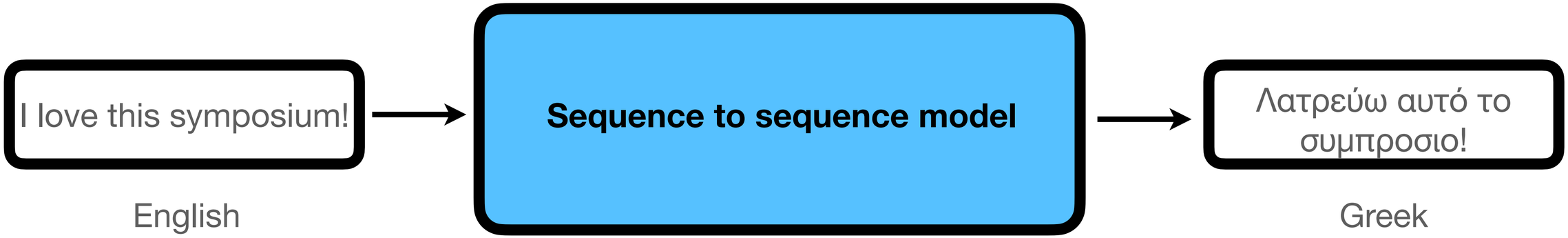
Transformers everywhere

The notion of self-attention

- Consider Sequence-to-Sequence (seq2seq) models
  (Machine translation, text summarization, image captioning..)

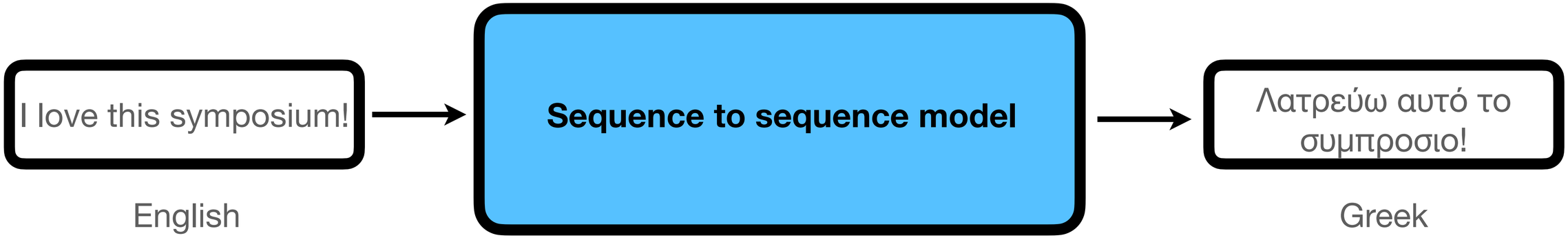
The notion of self-attention

- Consider Sequence-to-Sequence (seq2seq) models
  (Machine translation, text summarization, image captioning..)

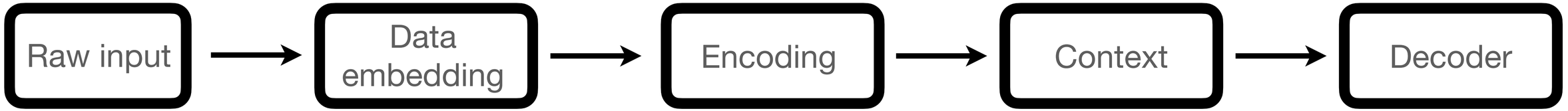


The notion of self-attention

- Consider Sequence-to-Sequence (seq2seq) models
  (Machine translation, text summarization, image captioning..)



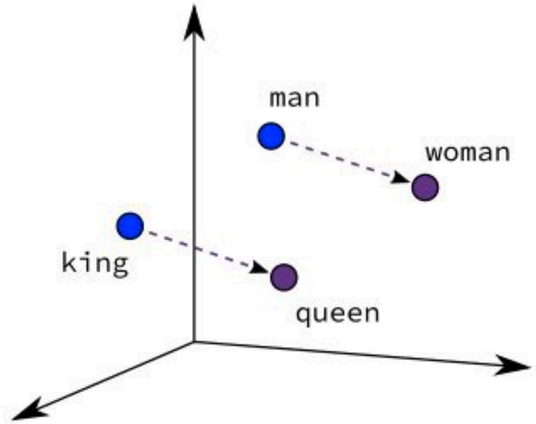
- What happens under the "hood":



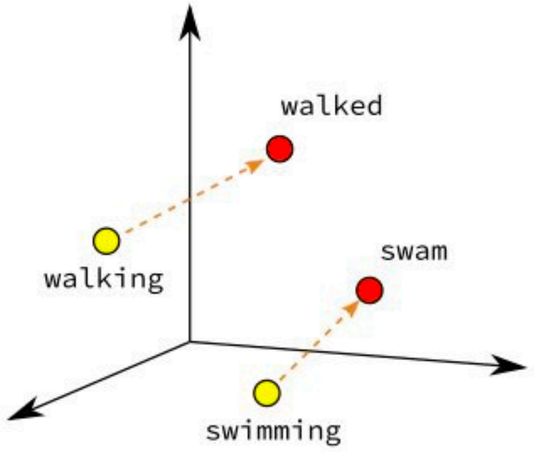
Data representation and word embeddings

Data representation and word embeddings

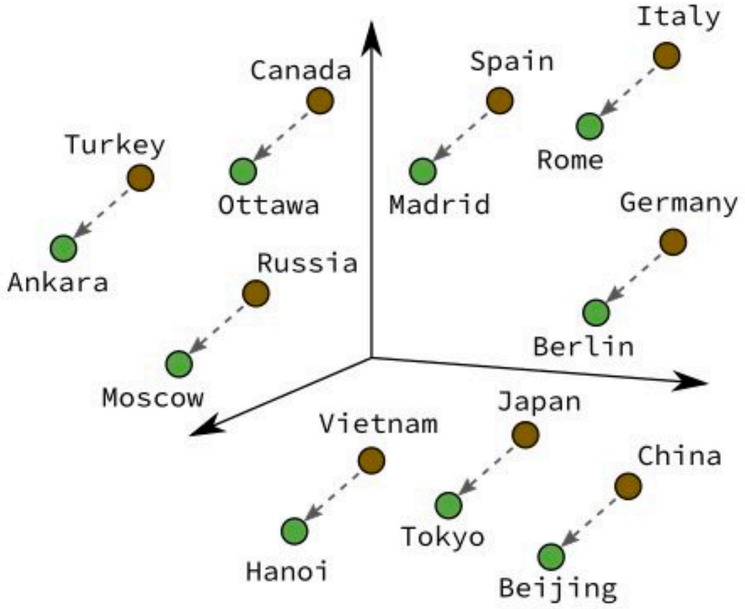
- Mechanisms that turn raw text pieces (words, sentences, etc) into vectors



Male-Female



Verb Tense

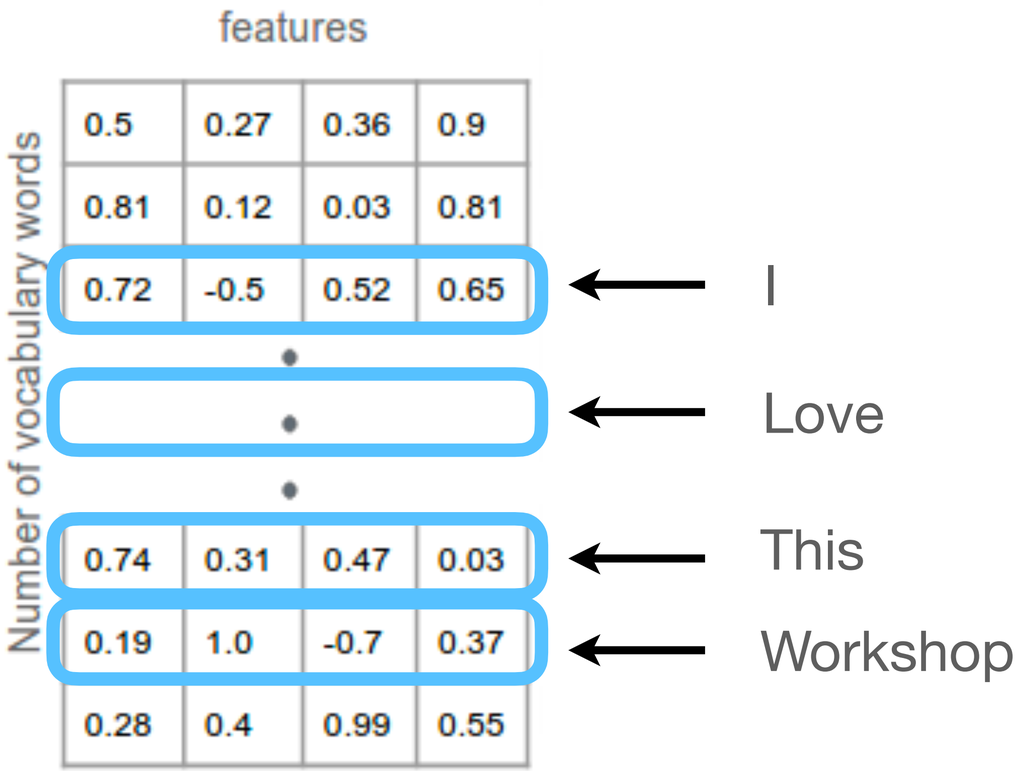


Country-Capital

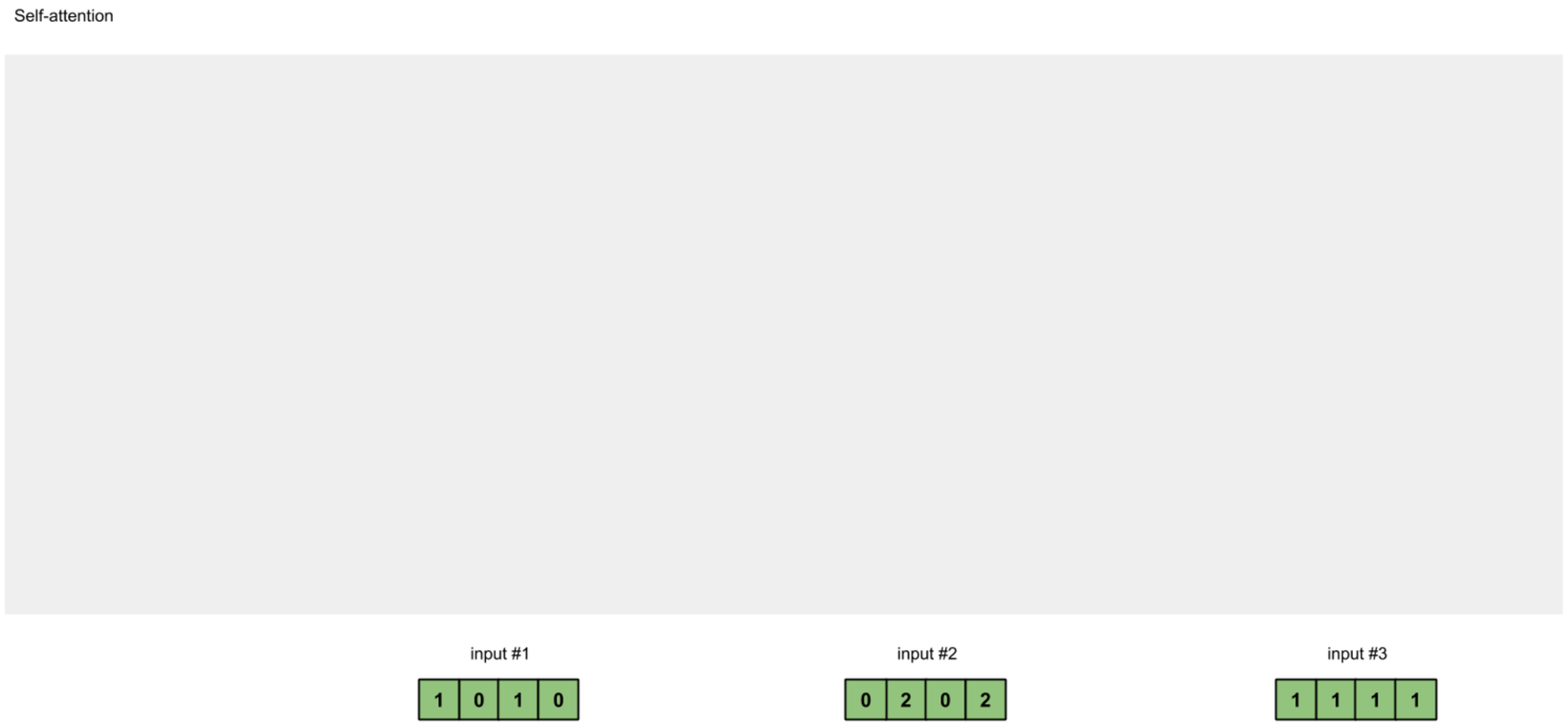
Tribute to
Google Developers

Data representation and word embeddings

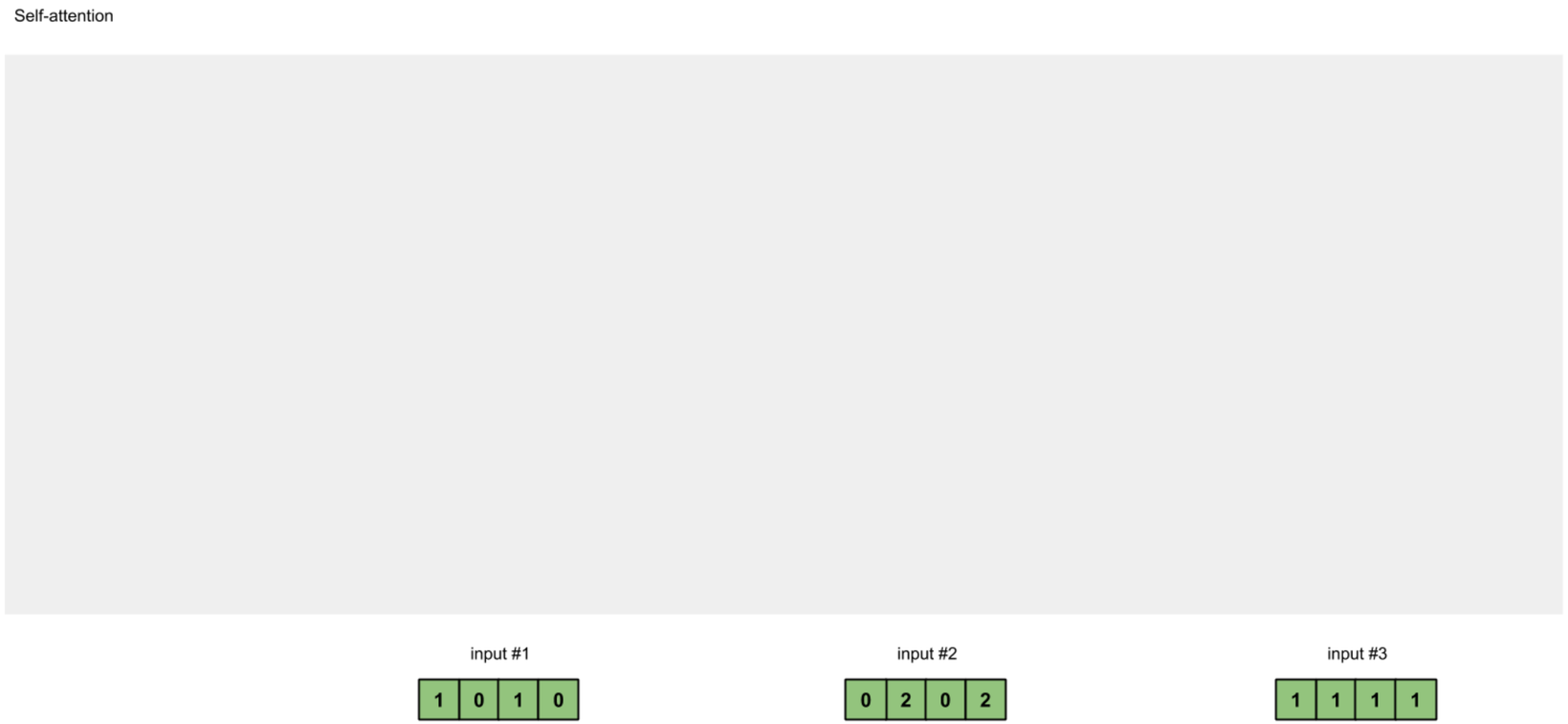
- Mechanisms that turn raw text pieces (words, sentences, etc) into vectors



What is self-attention?

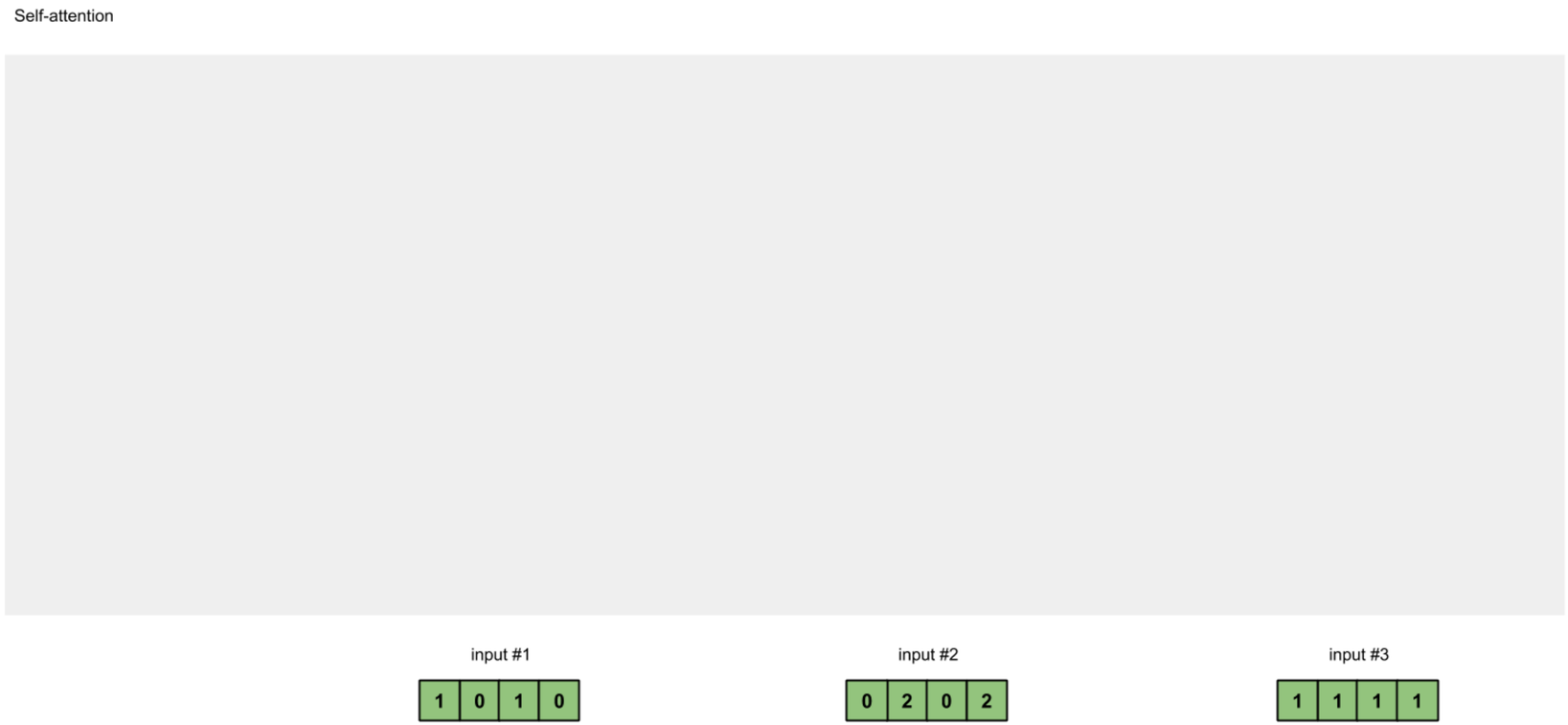


What is self-attention?



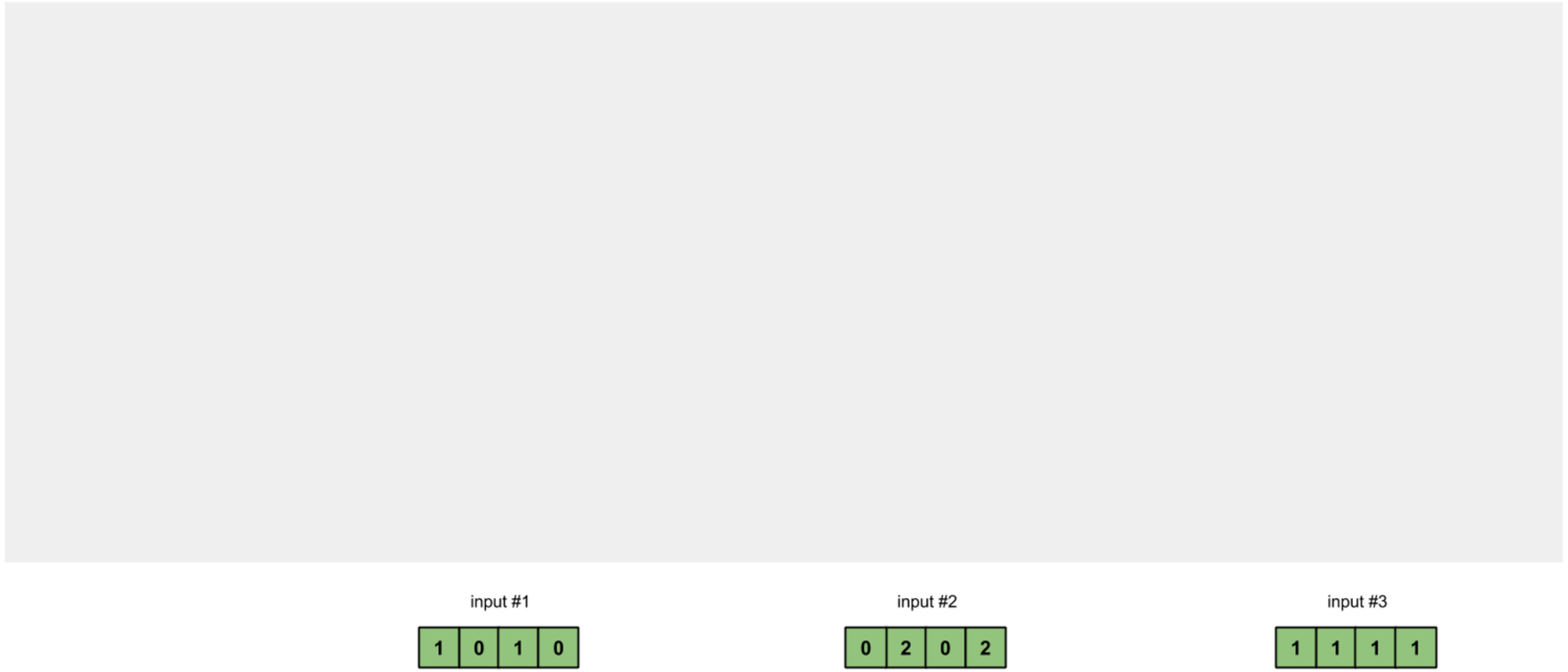
What is self-attention?

- Self-attention breaks the sequential nature of some NNs:
  i.e., the input is a long streak of words, and we compute correlations among ther



What is self-attention?

- Self-attention breaks the sequential nature of some NNs:
  i.e., the input is a long streak of words, and we compute correlations among ther
- Self-attention simultaneously compare each part of the input
  with other parts of the input



Transformers = multiple self-attention + FC layers

- Work by Google on 2017

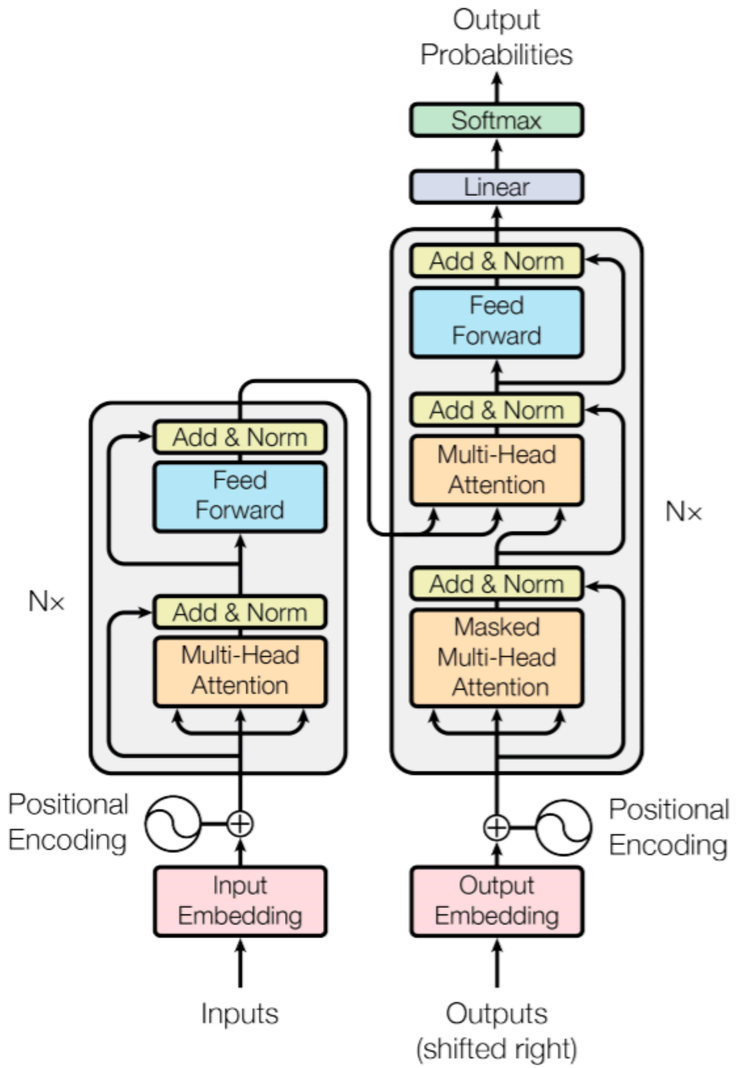


Figure 1: The Transformer - model architecture.

Transformers = multiple self-attention + FC layers

- Work by Google on 2017
- Applications: machine translation
  document summarization
  document generation
  bio sequence analysis
  video understanding
  playing games
  …

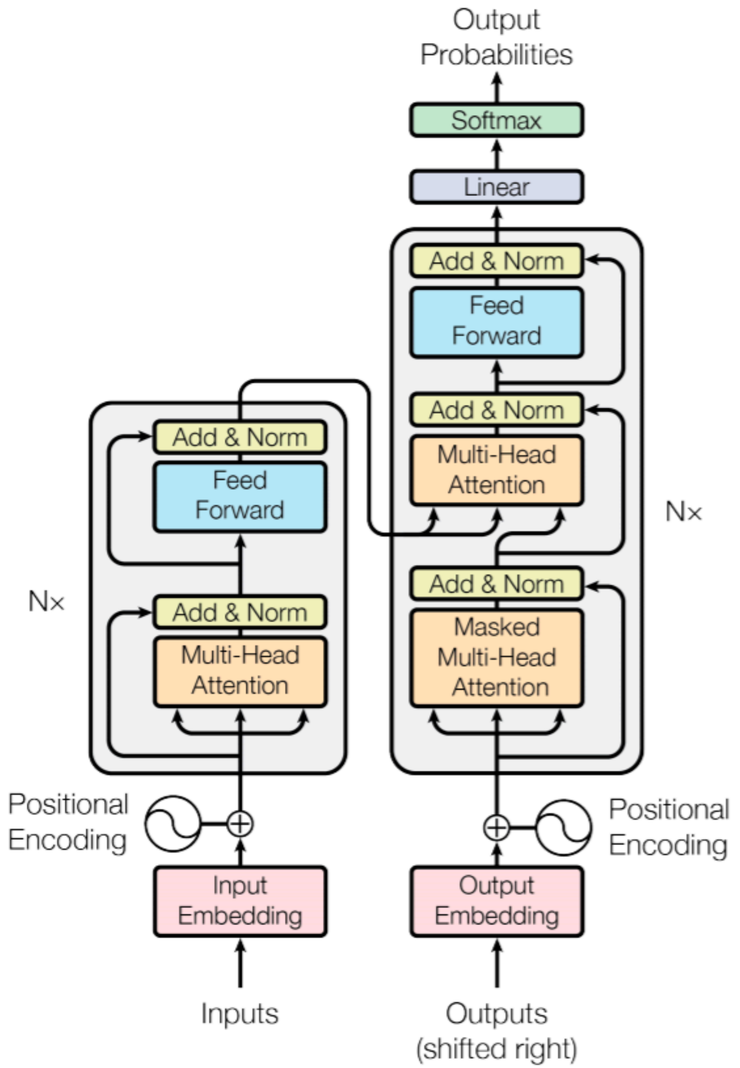


Figure 1: The Transformer - model architecture.

Transformers = multiple self-attention + FC layers

- Work by Google on 2017
- Applications: machine translation
  document summarization
  document generation
  bio sequence analysis
  video understanding
  playing games
  …
- Known large architectures: GPT-2 (1.5b)
  GPT-3 (175b)
  Wu Dao (1.75tr)
- Large models have raised many environmental questions

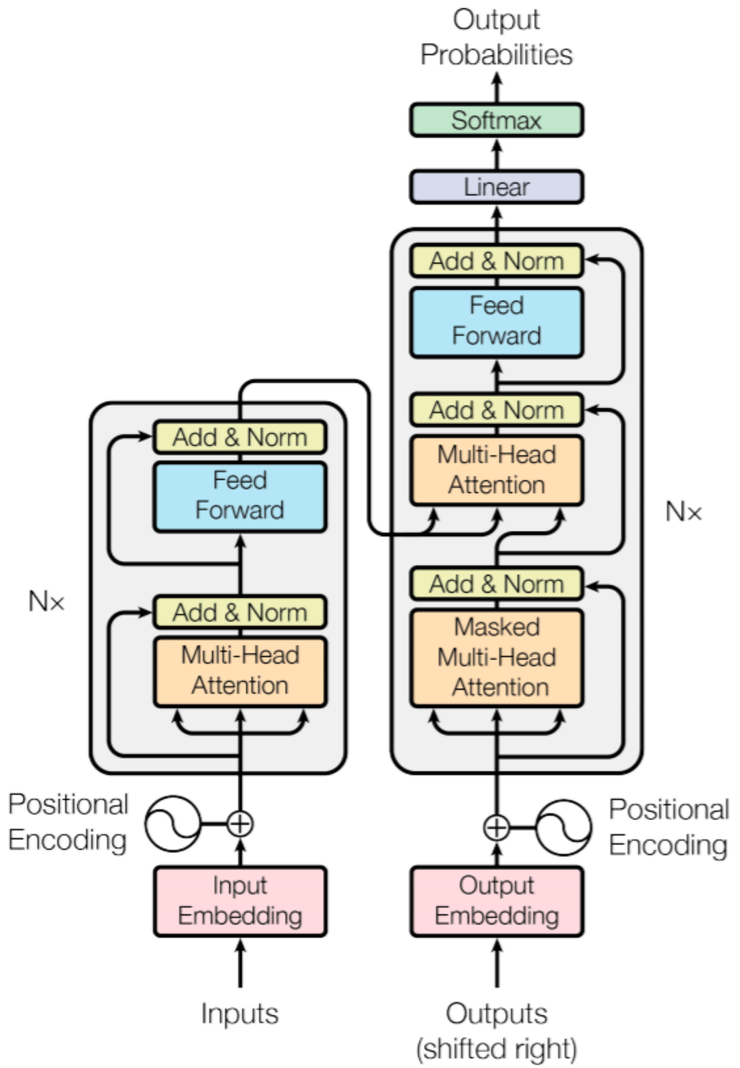


Figure 1: The Transformer - model architecture.

Overview

- Introduction to ML/AI
- Introduction to neural networks
- Feedforward / Fully-connected neural networks or MLPs
- Convolutional neural networks (CNNs)
- Some trends in deep learning

Thank you!