Spring 2024

CMPE 258-01 Deep Learning

Dr. Kaikai Liu, Ph.D. Associate Professor

Department of Computer Engineering

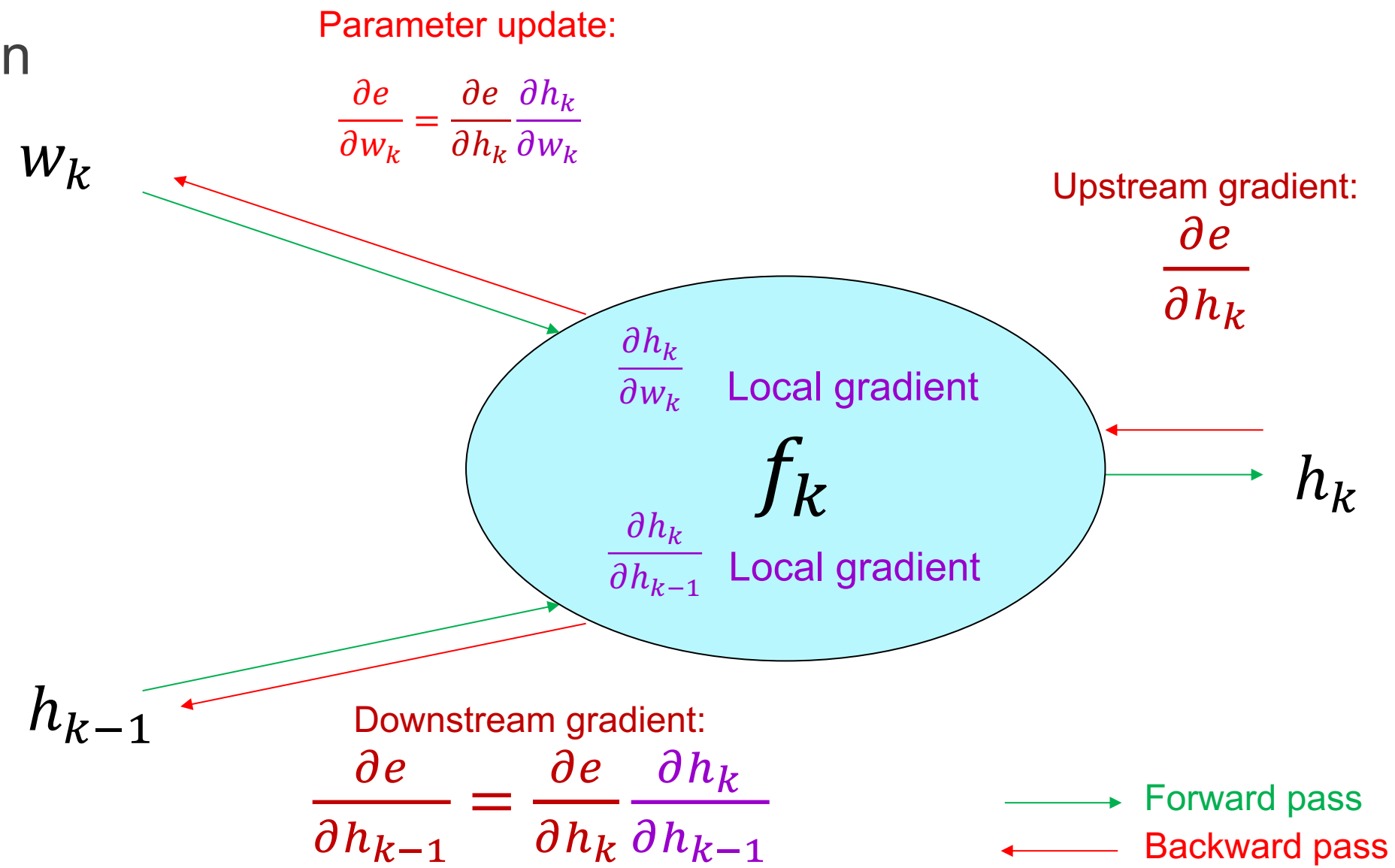
San Jose State University

Email: kaikai.liu@sjsu.edu

Website: https://www.sjsu.edu/cmpe/faculty/tenure-line/kaikai-liu.php

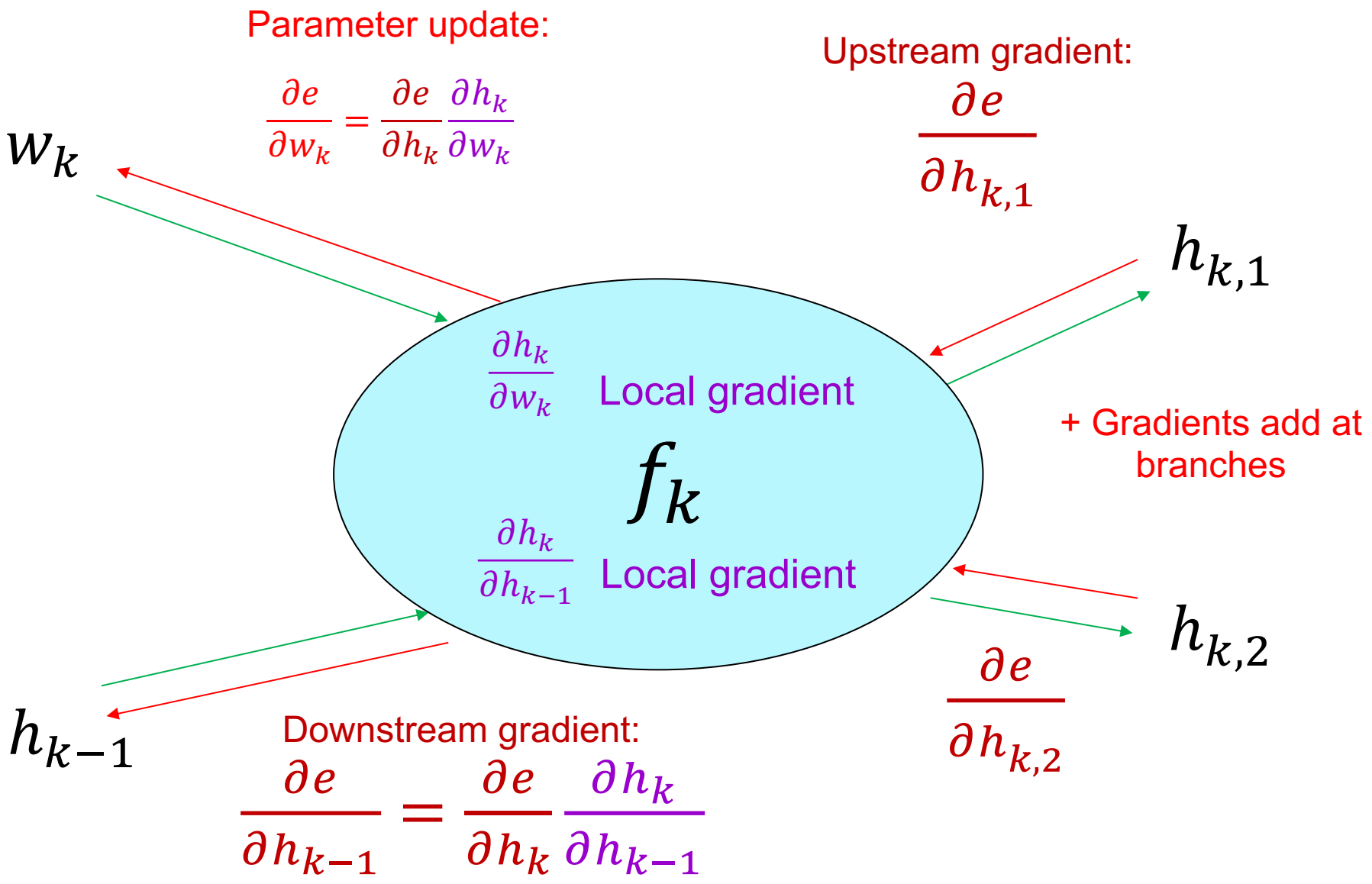
Neural Network Training

- Backpropagation



Neural Network Training

- With branches
  - E.g., Resnet

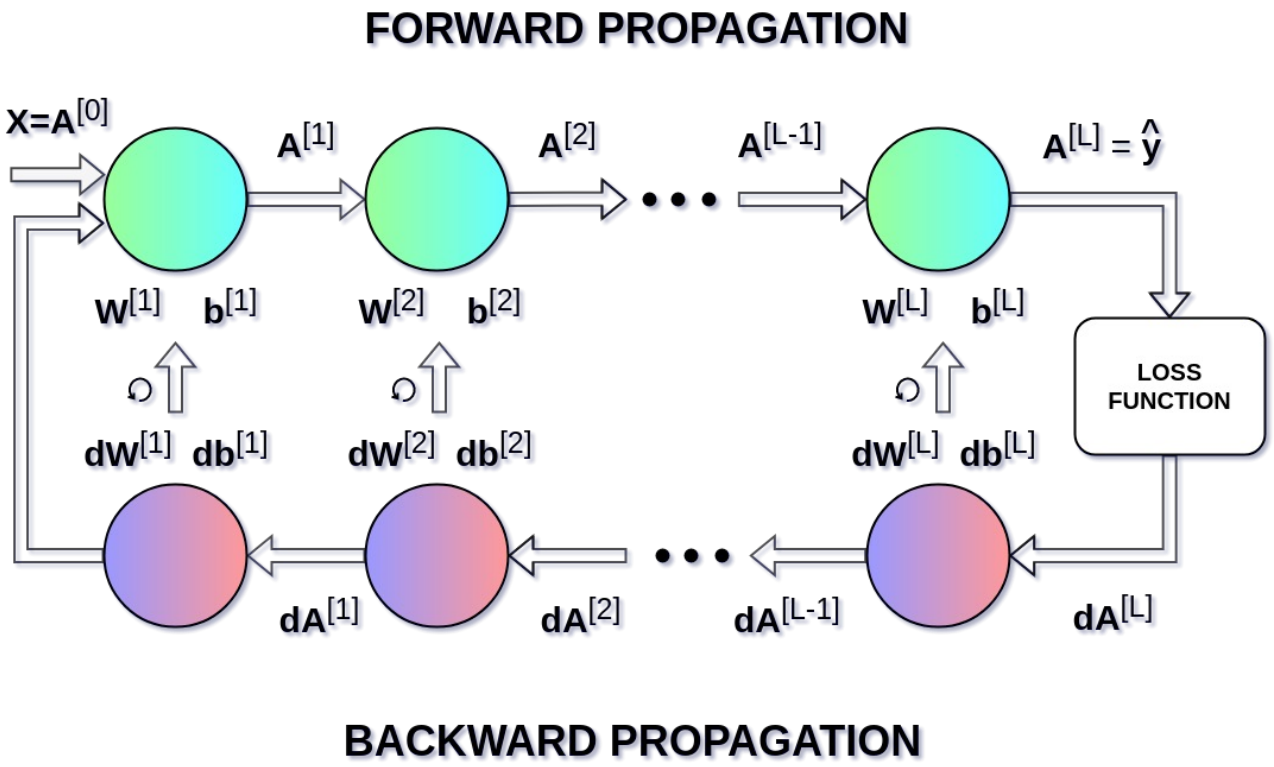


Backpropagation

- Backpropagation is an algorithm that allows us to calculate a very complicated gradient, like the one we need.
  - The parameters of the neural network are adjusted according to the following formulae.

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \mathbf{dW}^{[l]}$$
$$\mathbf{b}^{[l]} = \mathbf{b}^{[l]} - \alpha \mathbf{db}^{[l]}$$

  - α represents learning rate - a hyperparameter which allows you to control the value of performed adjustment.



Backpropagation

- **Simple example**

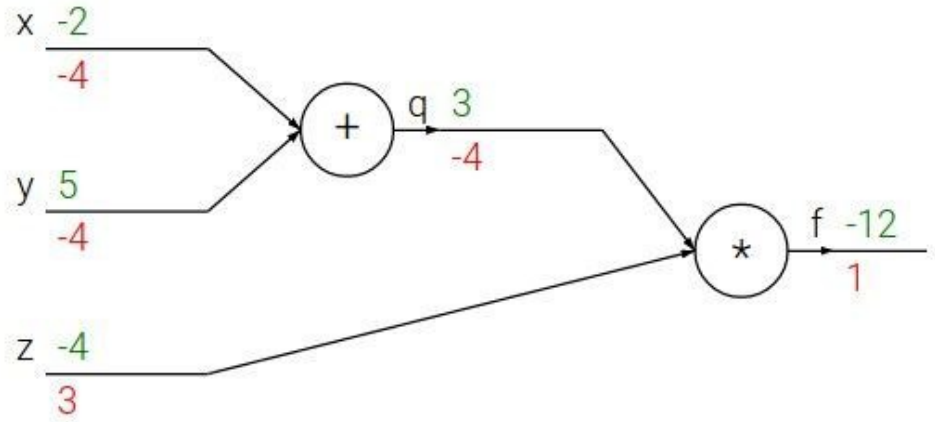
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

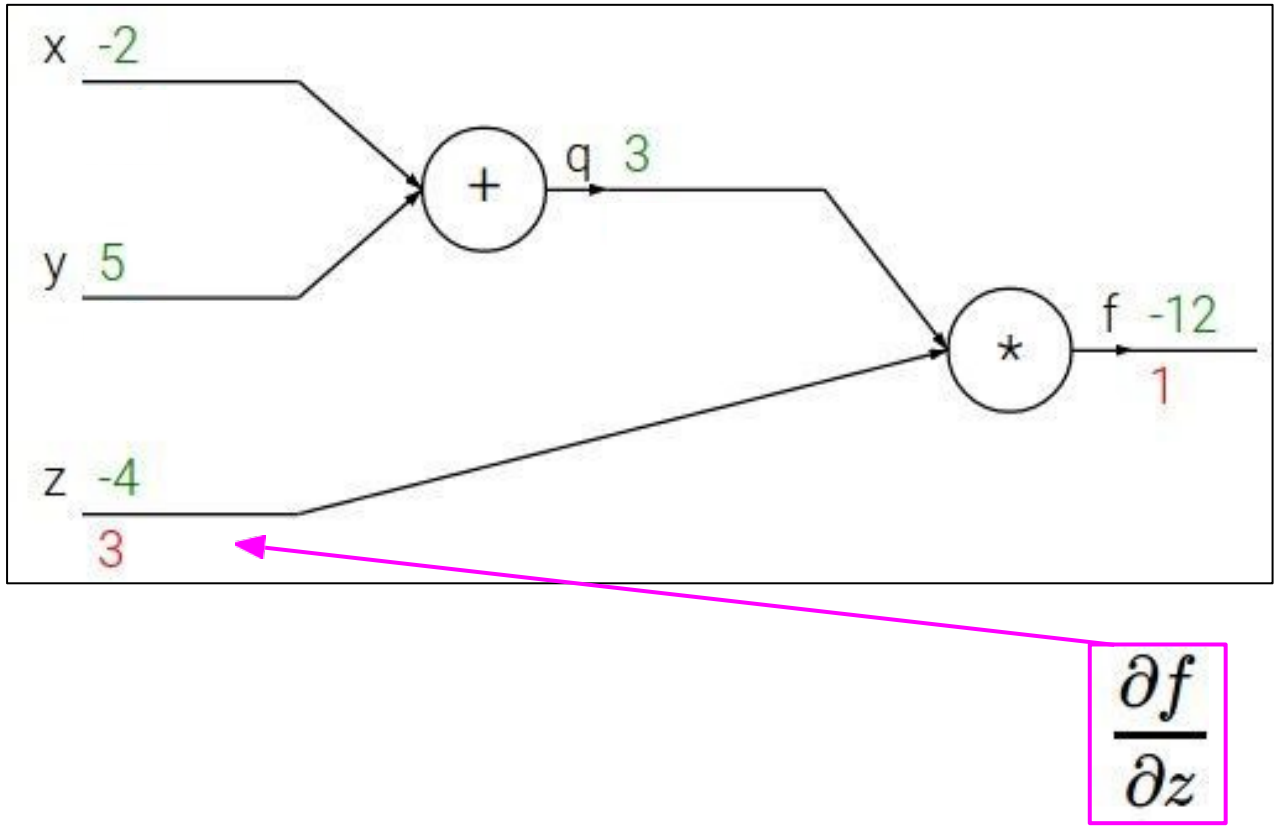
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

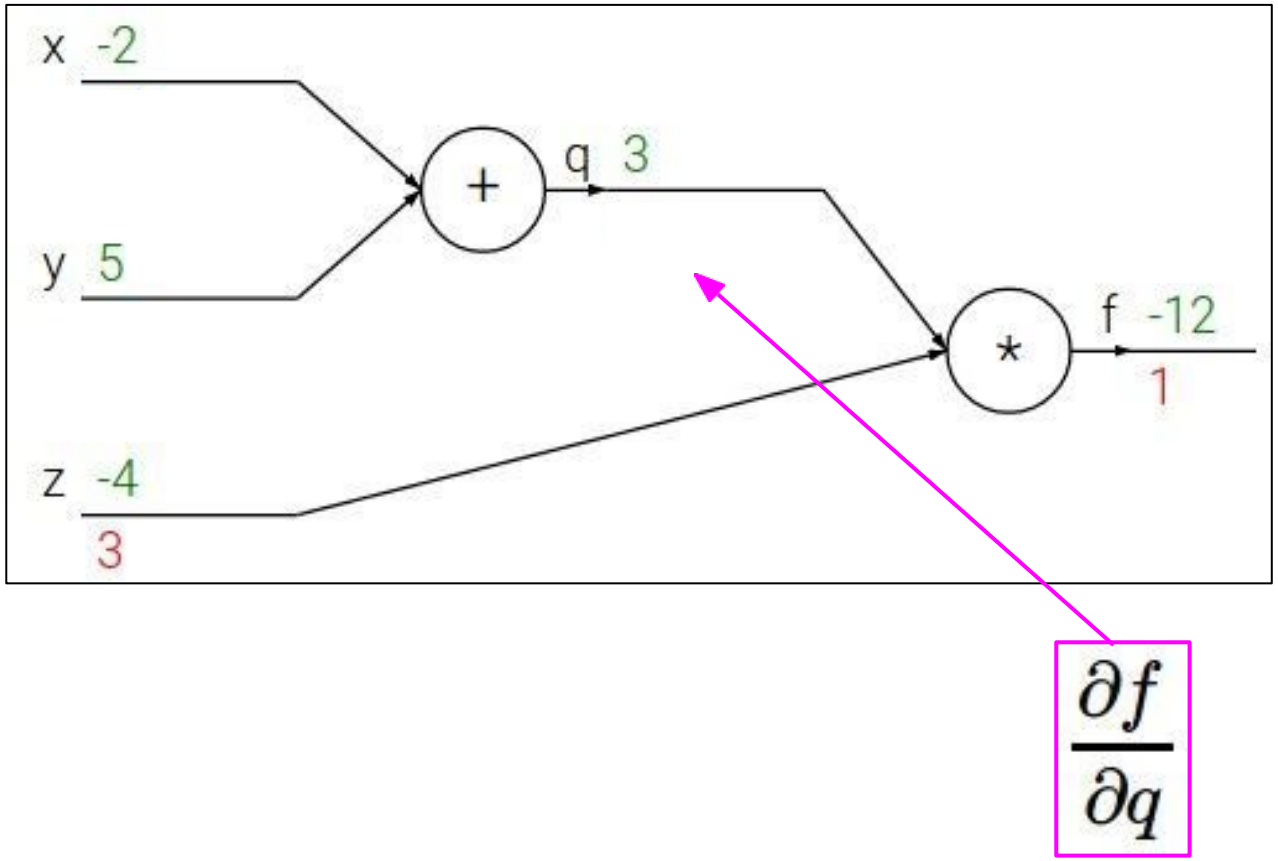
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

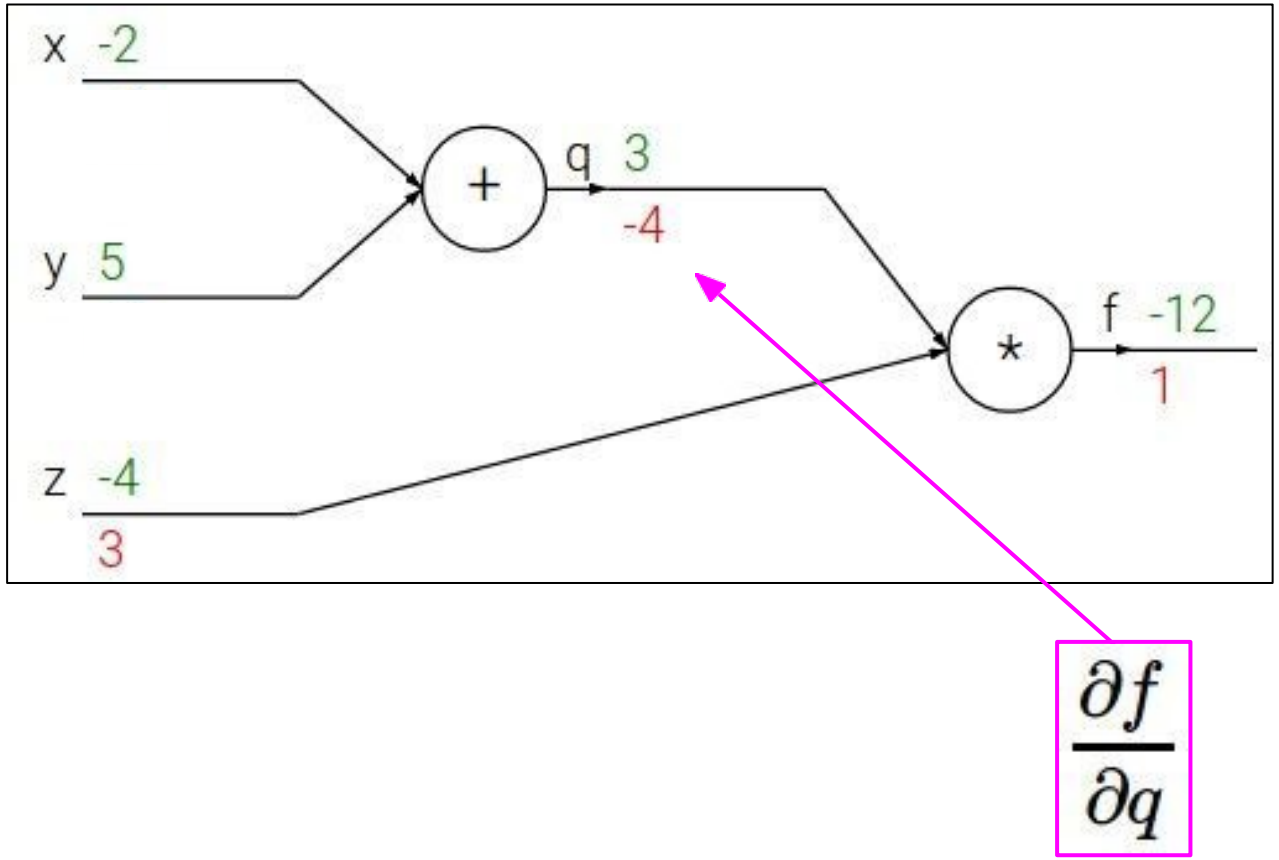
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Backpropagation: a simple example

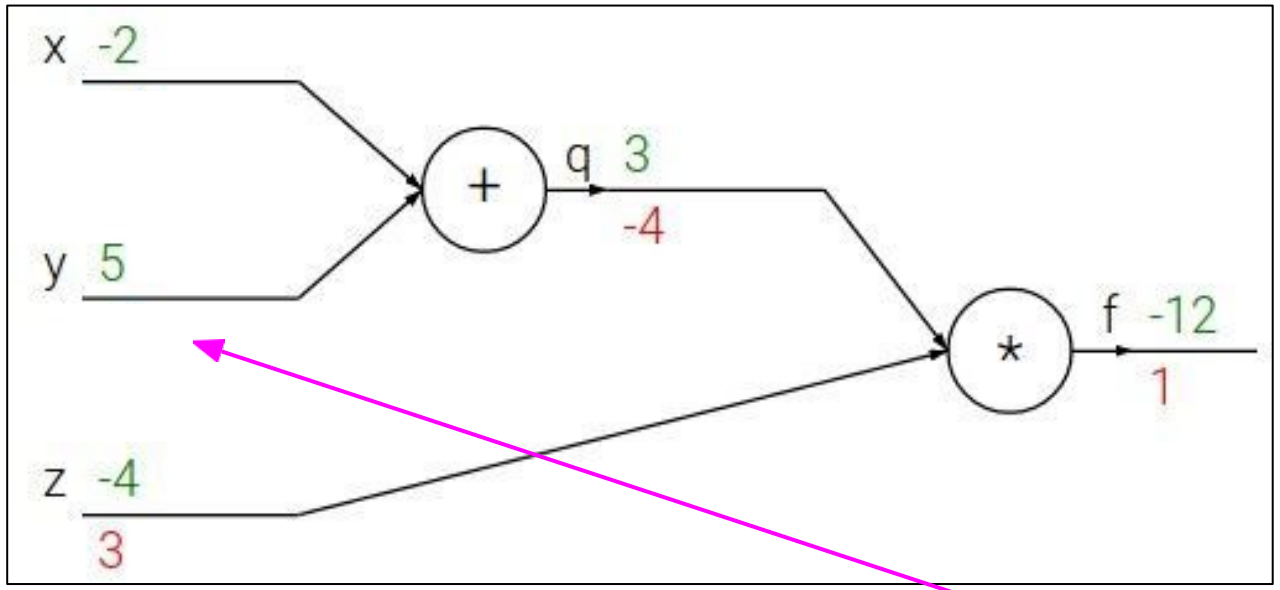
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial y}$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Upstream gradient

Local gradient

Backpropagation: a simple example

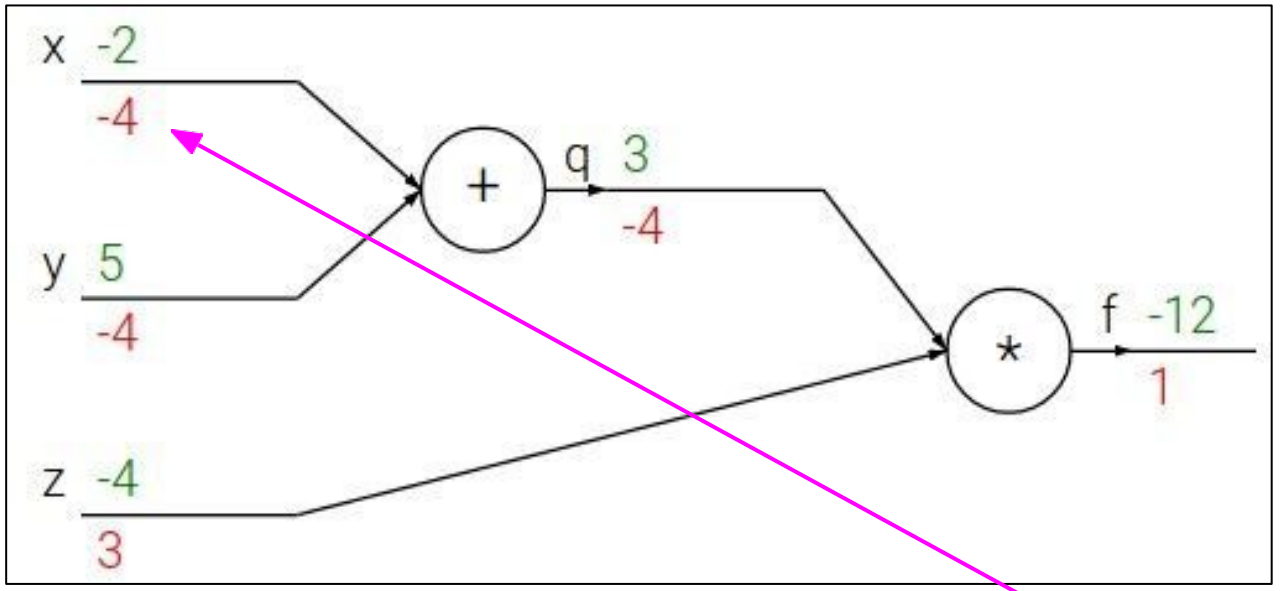
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial x}$

Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Upstream gradient

Local gradient

Jacobian Matrix

- In vector calculus, the Jacobian matrix of a vector-valued function of several variables is the matrix of all its first-order partial derivatives
- The Jacobian matrix collects all first-order partial derivatives of a multivariate function that can be used for backpropagation.

Suppose $\mathbf{f} : \mathbf{R}^n \to \mathbf{R}^m$ is a function such that each of its first-order partial derivatives exist on $\mathbf{R}^n$. This function takes a point $\mathbf{x} \in \mathbf{R}^n$ as input and produces the vector $\mathbf{f}(\mathbf{x}) \in \mathbf{R}^m$ as output. Then the Jacobian matrix of $\mathbf{f}$ is defined to be an $m \times n$ matrix, denoted by $\mathbf{J}$, whose (i,j)th entry is $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$, or explicitly

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

where $\nabla^{\mathrm{T}} f_i$ is the transpose (row vector) of the gradient of the i-th component.

Neural Network Training

- Gradient

$$y = f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$$

$$\frac{\partial y}{\partial \mathbf{x}} = \nabla_{\mathbf{x}} f(\mathbf{x}) = \left[\frac{\partial f}{\partial \mathbf{x}_1}, \ldots, \frac{\partial f}{\partial \mathbf{x}_d}\right]$$

- Dealing with vectors: Jacobian

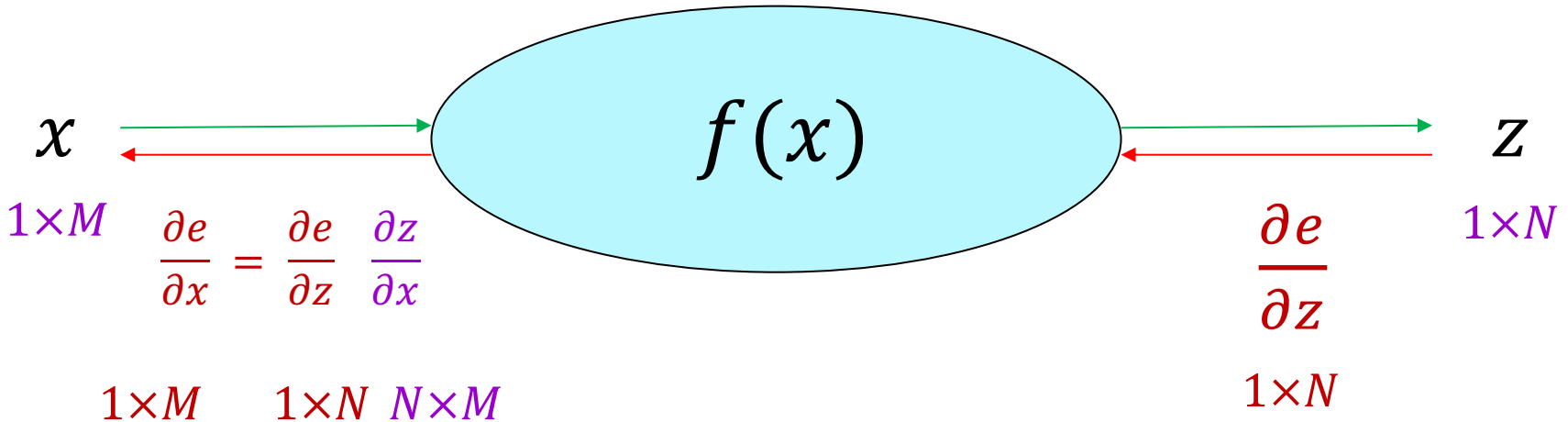
$$\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^k$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{J}_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_1}{\partial \mathbf{x}_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_k}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_k}{\partial \mathbf{x}_d} \end{bmatrix}$$

Jacobian: row indices correspond to outputs, column indices correspond to inputs. The i, jth element of the Jacobian is the partial derivative of the ith output w.r.t. jth input

$$\frac{\partial z}{\partial x} = \begin{pmatrix} \frac{\partial z^{(1)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(1)}}{\partial x^{(M)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial z^{(N)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(N)}}{\partial x^{(M)}} \end{pmatrix}$$

$N \times M$ Jacobian



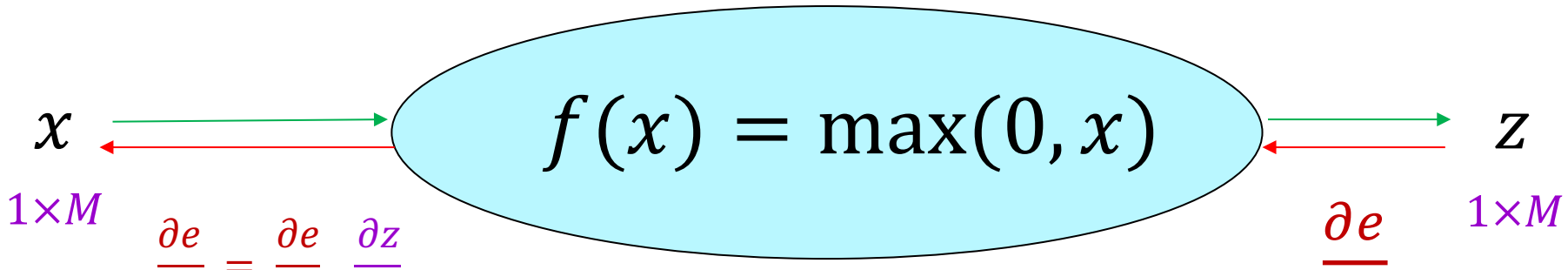
Neural Network Training

- Simple case: Elementwise operation (ReLU layer)

$$\frac{\partial z}{\partial x} = \begin{pmatrix} \frac{\partial z^{(1)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(1)}}{\partial x^{(M)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial z^{(M)}}{\partial x^{(1)}} & \cdots & \frac{\partial z^{(M)}}{\partial x^{(M)}} \end{pmatrix}$$

$M \times M$ Jacobian

What does the Jacobian for an elementwise function look like?



$$\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x}$$

$1 \times M \quad 1 \times M \quad M \times M$

$$\frac{\partial e}{\partial z}$$

$1 \times M$

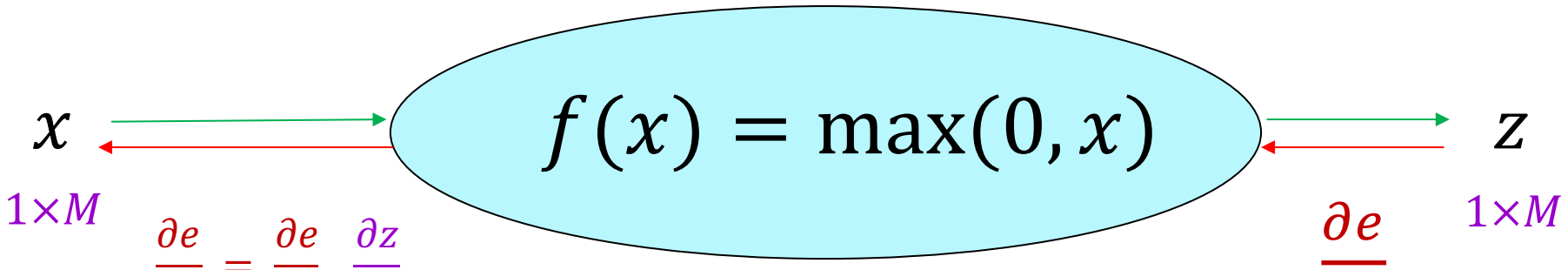
Neural Network Training

- Simple case: Elementwise operation (ReLU layer)

$$\frac{\partial z}{\partial x} = \begin{pmatrix} \frac{\partial z^{(1)}}{\partial x^{(1)}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\partial z^{(M)}}{\partial x^{(M)}} \end{pmatrix}$$

$M \times M$ Jacobian

What does the Jacobian for an elementwise function look like?



PyTorch

- PyTorch
  - originally developed by Meta AI and now part of the Linux Foundation umbrella
  - It is free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.
  - Meta operates both PyTorch and Convolutional Architecture for Fast Feature Embedding (Caffe2), but models defined by the two frameworks were mutually incompatible. The Open Neural Network Exchange (ONNX) project was created by Meta and Microsoft in September 2017 for converting models between frameworks. Caffe2 was merged into PyTorch at the end of March 2018. In September 2022, Meta announced that PyTorch would be governed by PyTorch Foundation, a newly created independent organization – a subsidiary of Linux Foundation.
- PyTorch is a Python package that provides two high-level features:
  - Tensor computation (like NumPy) with strong GPU acceleration
  - Deep neural networks built on a tape-based autograd system
- You can reuse your favorite Python packages such as NumPy, SciPy, and Cython to extend PyTorch when needed.
  - A replacement for NumPy to use the power of GPUs.
  - A deep learning research platform that provides maximum flexibility and speed.

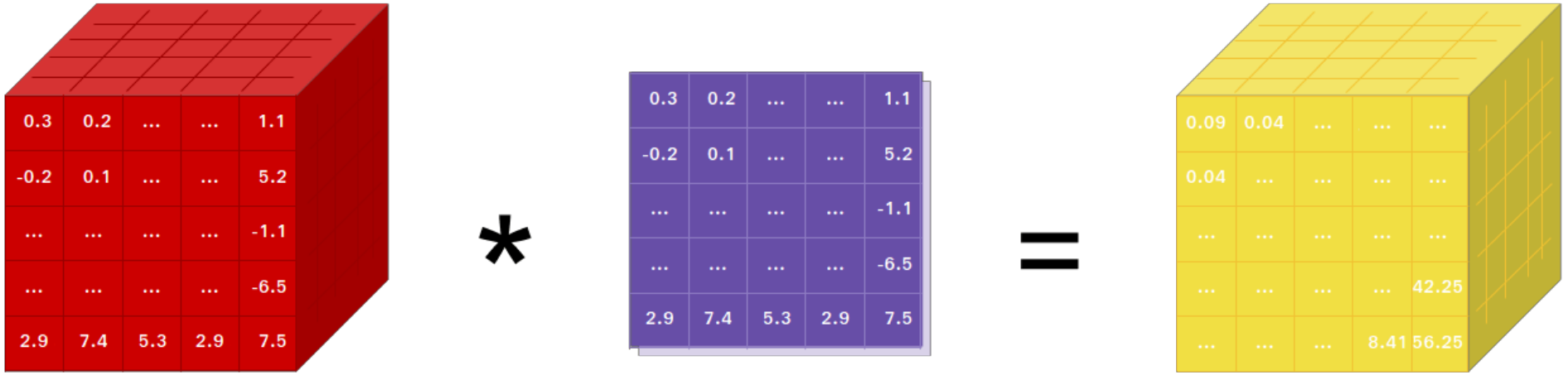
PyTorch

- PyTorch is a library that consists of the following components:

Component	Description
torch	A Tensor library like NumPy, with strong GPU support
torch.autograd	A tape-based automatic differentiation library that supports all differentiable Tensor operations in torch
torch.jit	A compilation stack (TorchScript) to create serializable and optimizable models from PyTorch code
torch.nn	A neural networks library deeply integrated with autograd designed for maximum flexibility
torch.multiprocessing	Python multiprocessing, but with magical memory sharing of torch Tensors across processes. Useful for data loading and Hogwild training
torch.utils	DataLoader and other utility functions for convenience

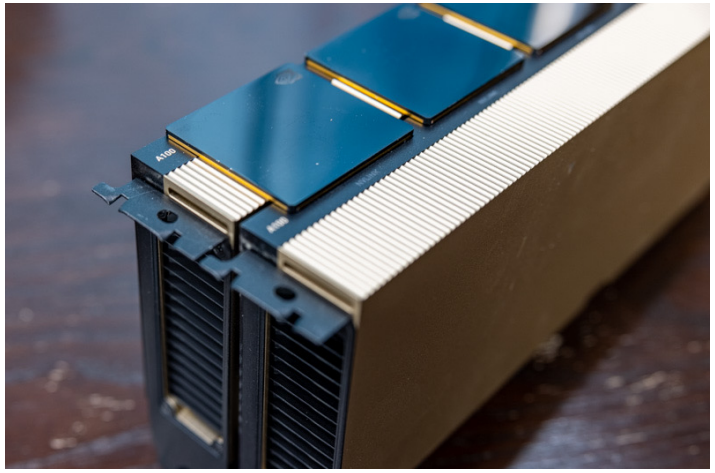
PyTorch

- PyTorch
  - A GPU-Ready Tensor Library: If you use NumPy, then you have used Tensors (a.k.a. ndarray).
  - PyTorch provides Tensors that can live either on the CPU or the GPU and accelerates the computation by a huge amount. A wide variety of tensor routines to accelerate and fit your scientific computation needs such as slicing, indexing, mathematical operations, linear algebra, reductions.
  - Integrate acceleration libraries such as Intel MKL and NVIDIA (cuDNN, NCCL) to maximize speed. At the core, its CPU and GPU Tensor and neural network backends are mature and have been tested for years.



NVIDIA AI Chips

- The NVIDIA A100 (based on NVIDIA Ampere architecture) is specifically tuned toward AI and high-performance computation instead of rendering 3D frames quickly for gaming.
  - Most 8x NVIDIA A100 systems, especially at larger cloud service providers, use a special NVIDIA-only form factor called SXM4
  - Each of these SXM4 A100's is not sold as a single unit. Instead, they are sold in either 4 or 8 GPU subsystems

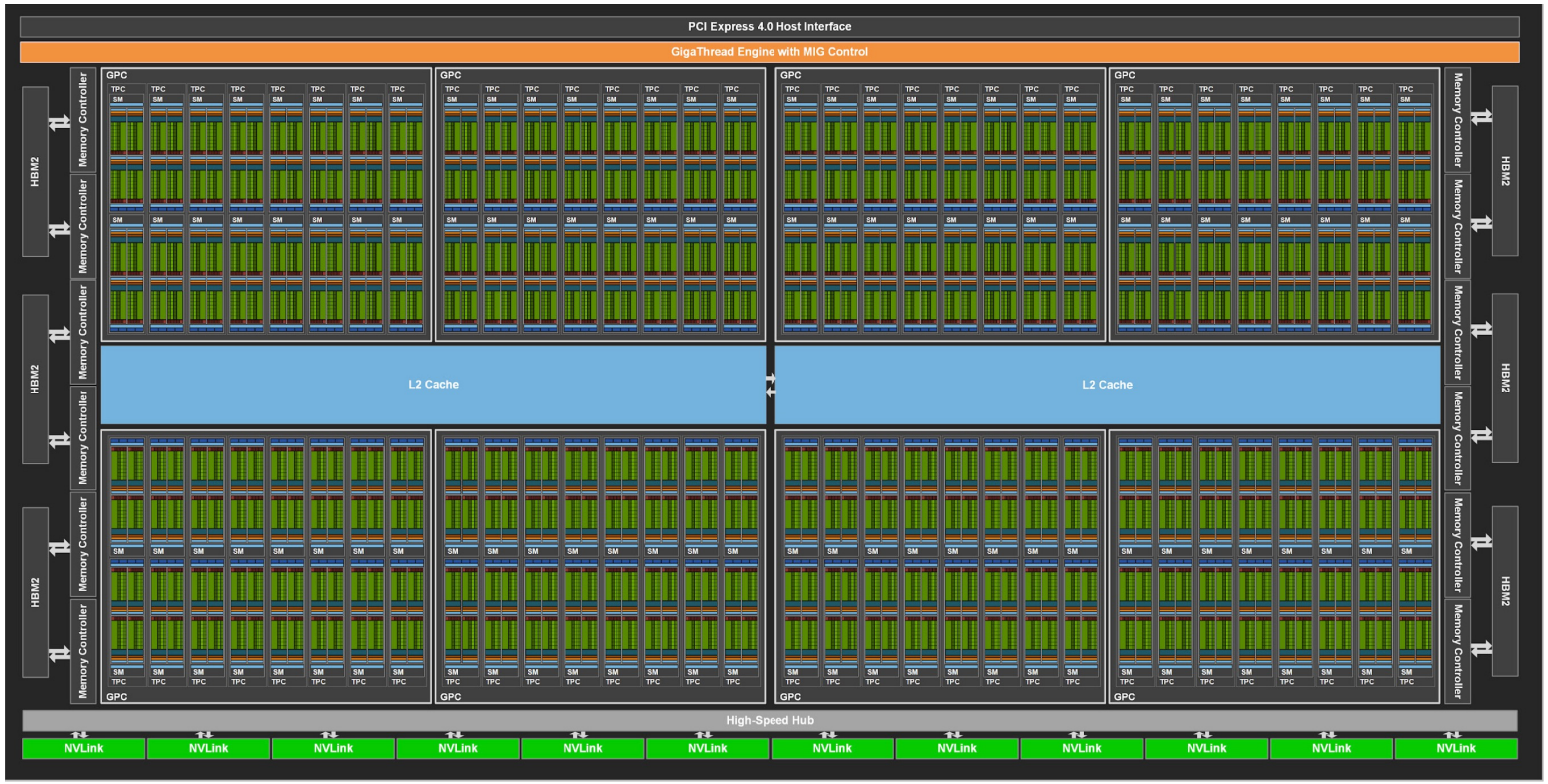


2x NVIDIA A100 PCIe With NVLink Bridges Installed

NVIDIA HGX A100 8 GPU Assembly

NVIDIA AI Chips

- The A100 is based on GA100 and has 108 SMs.
  - The NVIDIA GA100 GPU is composed of multiple GPU Processing Clusters (GPCs), Texture Processing Clusters (TPCs), Streaming Multiprocessors (SMs), and HBM2 memory controllers.



NVIDIA AI Chips

- NVIDIA H100
  - The NVIDIA H100 GPU with SXM5 board form-factor includes the following units:
    - 8 GPCs, 66 TPCs, 2 SMs/TPC, 132 SMs per GPU
    - 128 FP32 CUDA Cores per SM, 16896 FP32 CUDA Cores per GPU
    - 4 fourth-generation Tensor Cores per SM, 528 per GPU
    - 80 GB HBM3, 5 HBM3 stacks, 10 512-bit memory controllers
    - Fourth-generation NVLink and PCIe Gen 5

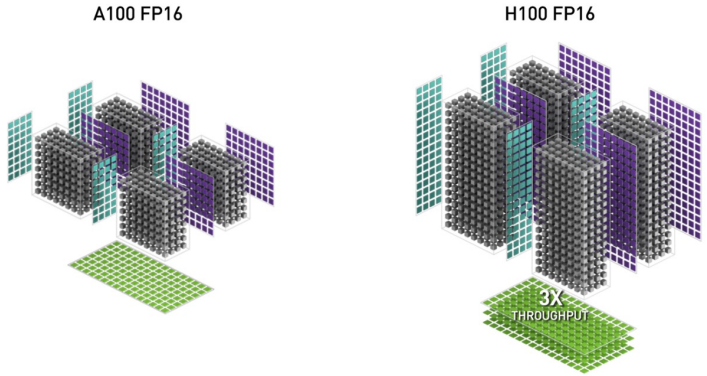
Nvidia Makes Nearly 1000% Profit on H100 GPUs: Report

Nvidia is raking in nearly 1,000% (about 823%) in profit percentage for each H100 GPU accelerator it sells, according to estimates made in a...

A $40000 Nvidia chip has become the world's most sought-after hardware

Companies and governments want to deploy generative AI—but first they need access to Nvidia's H100 chips.

Nvidia plans to triple production of its $40,000 chips as it races to meet huge demand from AI companies, report says



The New York Times

Nvidia Revenue Doubles on Demand for A.I. Chips, and Could Go Higher

BBC

Artificial intelligence chip giant Nvidia sees sales more than double

The Verge

Nvidia just made $6 billion in pure profit over the AI boom

PyTorch

- PyTorch Introduction in https://github.com/lkk688/DeepDataMiningLearning
  - Pytorch Introductions, Tensors, and Autograd: colablink
  - Pytorch Regression and Logistic Regression: colablink
  - Pytorch Simple Neural Networks: colab

PyTorch

- Dynamic Neural Networks: Tape-Based Autograd
  - PyTorch has a unique way of building neural networks: using and replaying a tape recorder.
  - Most frameworks such as TensorFlow, Theano, Caffe, and CNTK have a static view of the world. One has to build a neural network and reuse the same structure again and again. Changing the way the network behaves means that one has to start from scratch.
  - With PyTorch, we use a technique called reverse-mode auto-differentiation, which allows you to change the way your network behaves arbitrarily with zero lag or overhead.

A graph is created on the fly


```
W_h = torch.randn(20, 20, requires_grad=True)
W_x = torch.randn(20, 10, requires_grad=True)
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)
```

While this technique is not unique to PyTorch, it's one of the fastest implementations of it to date.

Deep Learning Frameworks

- Tensorflow: https://www.tensorflow.org
  - It was developed by the Google Brain team for Google's internal use in research and production
  - As TensorFlow's market share among research papers was declining to the advantage of PyTorch, the TensorFlow Team announced a release of a new major version of the library in September 2019. TensorFlow 2.0 introduced many changes, the most significant being TensorFlow eager, which changed the automatic differentiation scheme from the static computational graph, to the "Define-by-Run" scheme originally made popular by Chainer and later PyTorch
  - Keras abstractions.
- Jax: Composable transformations of Python+NumPy programs: differentiate, vectorize, JIT to GPU/TPU, and more
  - https://github.com/google/jax
  - use Flax (https://github.com/google/flax) on top of JAX, which is a neural network library developed by Google. It contains many ready-to-use deep learning modules, layers, functions, and operations
  - Flax is a neural network library for JAX that is designed for flexibility.
- Apple MLX: An array framework for Apple silicon
  - https://github.com/ml-explore/mlx
  - MLX has a Python API that closely follows NumPy. MLX also has a fully featured C++ API, which closely mirrors the Python API. MLX has higher-level packages like mlx.nn and mlx.optimizers with APIs that closely follow PyTorch to simplify building more complex models.

Thank You

Address:
ENG257, SJSU

Email Address:
Kaikai.liu@sjsu.edu