

# **Эволюция DL, Backpropagation**

# Глубинное обучение: приложения



# Глубинное обучение: приложения

≡ Google Translate ⋮

Text Documents

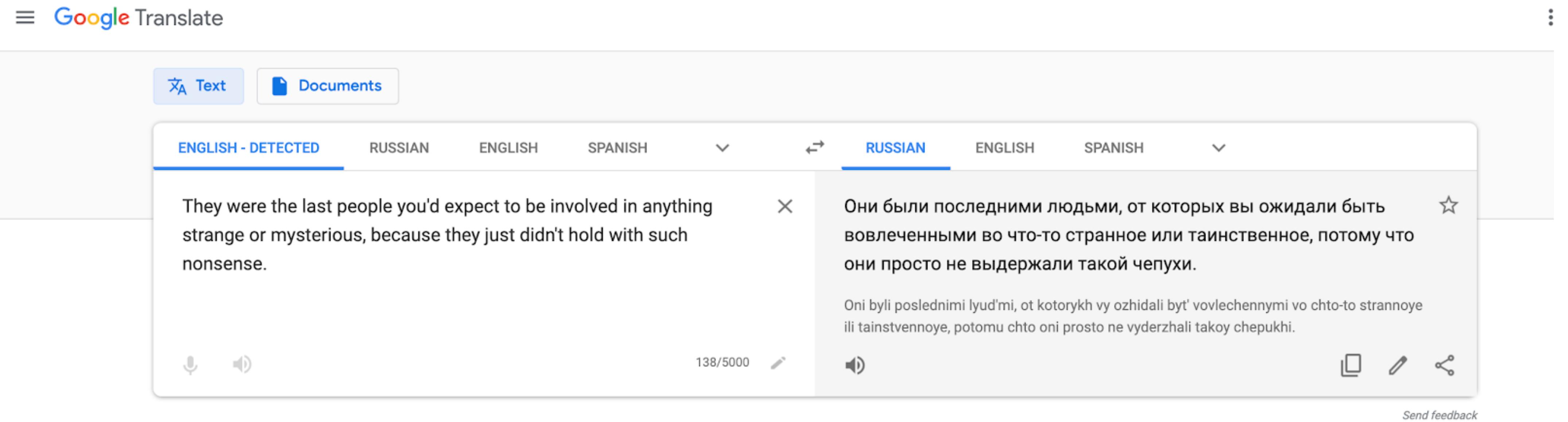
ENGLISH - DETECTED RUSSIAN ENGLISH SPANISH RUSSIAN ENGLISH SPANISH

They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.

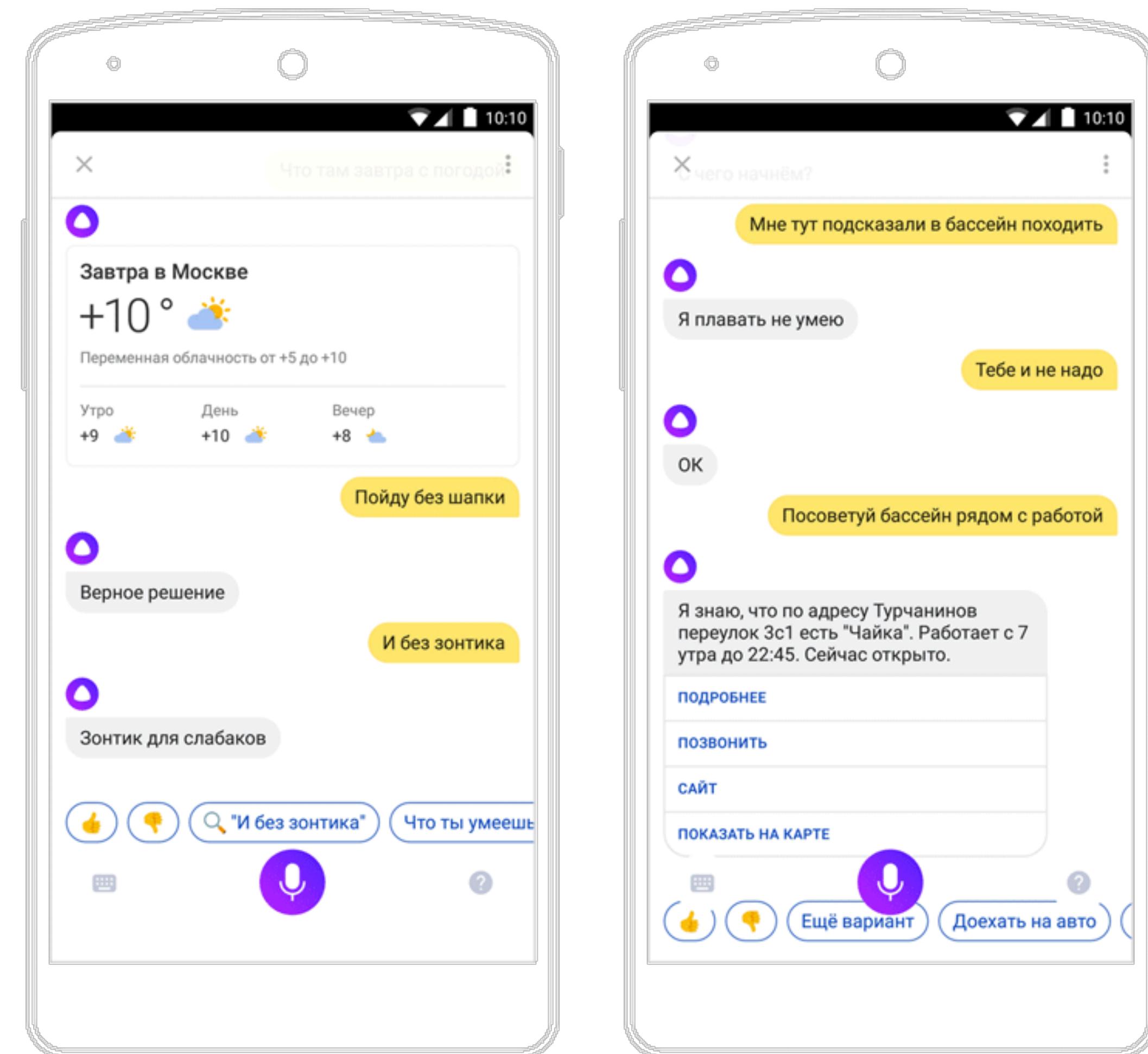
Oни были последними людьми, от которых вы ожидали быть вовлечеными во что-то странное или таинственное, потому что они просто не выдержали такой чепухи.

Oni byli poslednimi lyud'mi, ot kotorikh vy ozhidali byt' vovlechennymi vo chto-to strannoye ili tainstvennoye, potomu chto oni prosto ne vyderzhali takoy chepukhi.

Send feedback



# Глубинное обучение: приложения



# Глубинное обучение: о курсе

- Начнем с самых основ и дойдем до современных SOTA моделей
  - Как обучать нейросети, когда и какие архитектуры использовать
  - Практическое применение - CV, NLP

- Нейронные сети были давно, но обучать глубокие сети не умели.
- А теперь — научились.
- И это перевернуло весь мир машинного обучения!

# ЧЕЛОВЕЧЕСКИЙ МОЗГ И ИСТОРИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

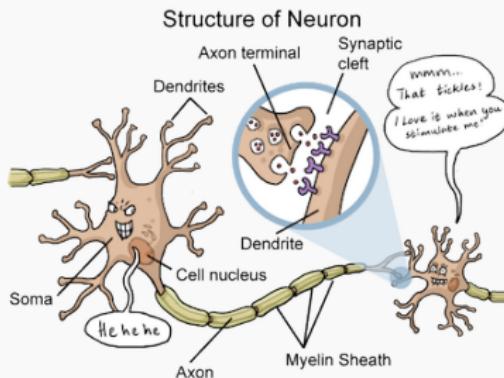
---

# Компьютер у нас в голове

- Компьютер считает быстрее человека, но гораздо хуже
  - понимает естественный язык,
  - распознаёт изображения, узнаёт людей (это уже не совсем так),
  - обучается в широком смысле этого слова и т.д...
- Почему так? Как человек всего этого добивается?

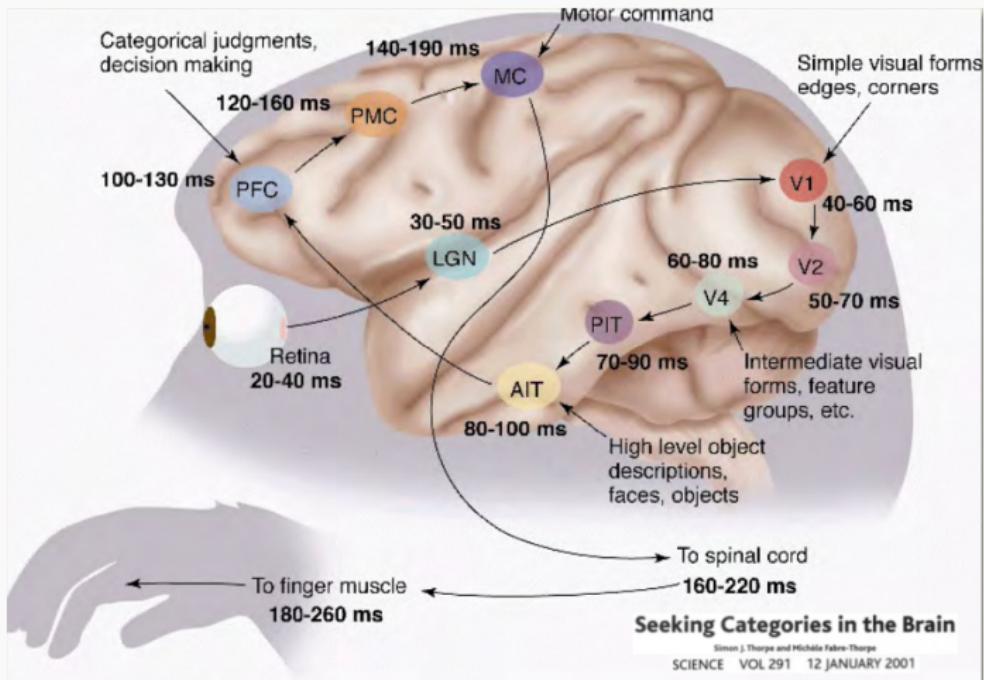
# Компьютер у нас в голове

- В мозге много нейронов; каждый нейрон:
  - через дендриты получает сигнал от других нейронов;
  - время от времени запускает сигнал по аксону;
  - через синапсы сигнал аксона доходит до дендритов других нейронов.



# КОМПЬЮТЕР У НАС В ГОЛОВЕ

- Вот как мы обрабатываем картинку:



## Обучение признаков

- Другая сторона вопроса – обучение признаков.
- Мозг очень хорошо умеет обучаться на очень-очень маленькой выборке данных.
- И может адаптироваться к новым источникам информации.
- Как он это делает? Можем ли мы сделать так же?

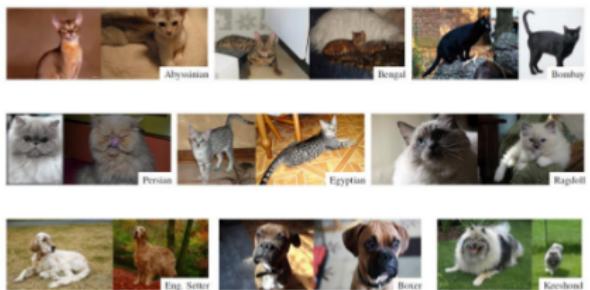
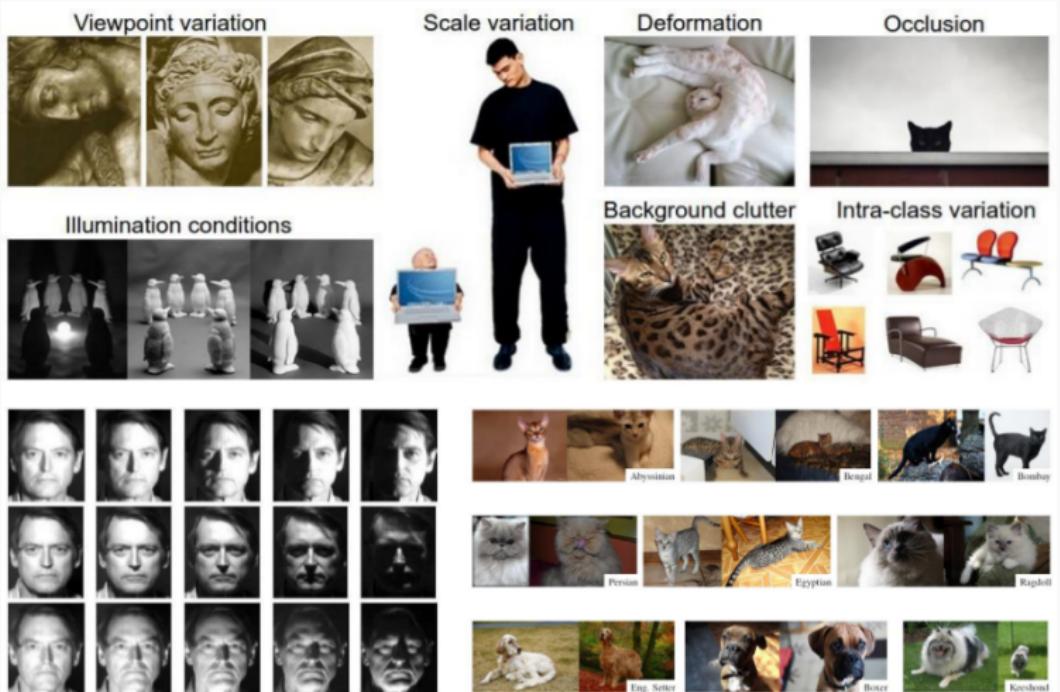
# Обучение признаков

- Системы обработки неструктурированной информации выглядят обычно так:

вход → признаки → классификатор

- Люди много десятилетий пытались придумать хорошие признаки:
  - MFCC для распознавания речи;
  - SIFT для обработки изображений;
  - ...
- Задача feature engineering: как сделать такие признаки?
- Но, может быть, можно найти признаки автоматически? Мозг ведь это как-то делает...
- Но это сложно!

# ПОЧЕМУ ЭТО СЛОЖНО: РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ

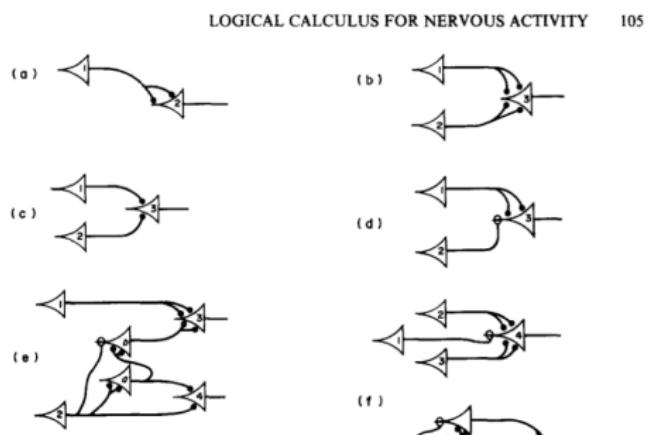
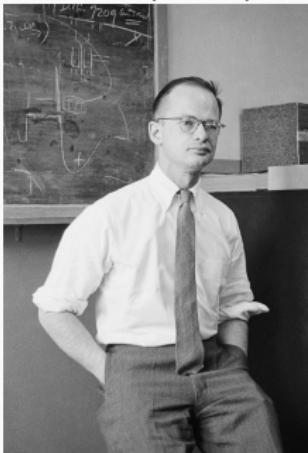


## Суть происходящего

- Вообще говоря, всё, что мы делаем в машинном обучении, — это аппроксимация и оптимизация функций.
- Например:
  - есть функция из картинок в то, что на них изображено;
  - ещё проще: бинарная функция из пикселей картинки в «котик или не котик»;
  - функция, мягко скажем, довольно сложная; задана в виде датасета; как нам её реализовать?
- Обычный подход в машинном обучении:
  - строим какой-то класс моделей, обычно параметрический;
  - и подгоняем параметры так, чтобы модель хорошо описывала данные;
  - то есть оптимизируем какую-то функцию ошибки или функцию качества (правдоподобие, апостериорную вероятность).
- Нейронные сети — это очень мощный и гибкий класс таких моделей.
- Сложность в том, как же обучить их параметры.

# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

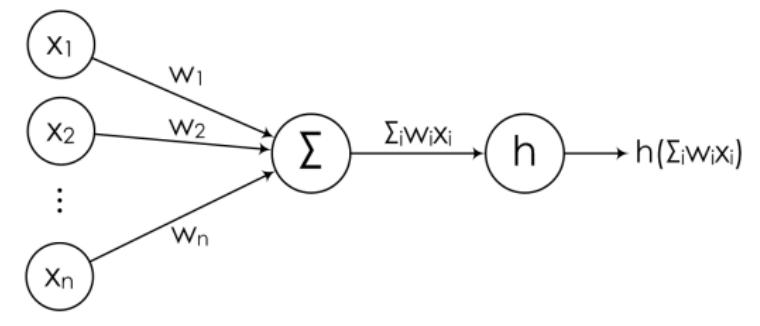
- Давайте попробуем как в природе: есть связанные между собой нейроны, которые передают друг другу сигналы.
- McCulloch, Pitts, 1943: идея.



- Идею искусственных сетей, похожих на современные (даже с несколькими уровнями), предлагал ещё Алан Тьюринг (1948).
- Rosenblatt, 1958: перцептрон (один искусственный нейрон).
- Тогда же появился алгоритм обучения градиентным спуском.

# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Один нейрон обычно моделируется вот так:



- Линейная комбинация входов, потом нелинейность:

$$y = h(w^\top x) = h\left(\sum_i w_i x_i\right).$$

- Как обучить перцептрон, и каков будет результат?

# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Обучение градиентным спуском.
- Для исходного перцептрана Розенблатта ( $h = \text{id}$ ):

$$E_P(\mathbf{w}) = - \sum_{x \in M} y(\mathbf{x}) (\mathbf{w}^\top \mathbf{x}),$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla_{\mathbf{w}} E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta t_n \mathbf{x}_n.$$

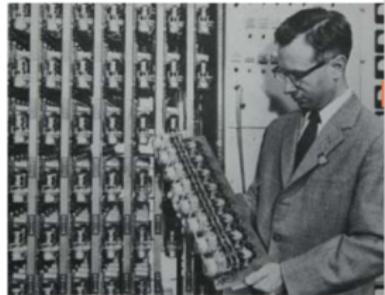
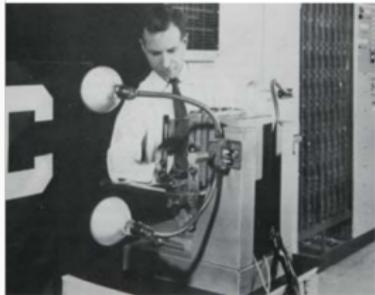
- Или, к примеру, можно делать классификацию, подставляя в качестве функции активации логистический сигмоид  $h(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ :

$$E(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N (y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1 - y_i) \log (1 - \sigma(\mathbf{w}^\top \mathbf{x}_i))).$$

- По сути это просто логистическая регрессия.

# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Первый перцептрон (Rosenblatt, 1958) распознавал цифры.



# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- И люди в него верили:

## NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's \$2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human be-

ings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

### Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

1958 New York Times...

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

### Learns by Doing

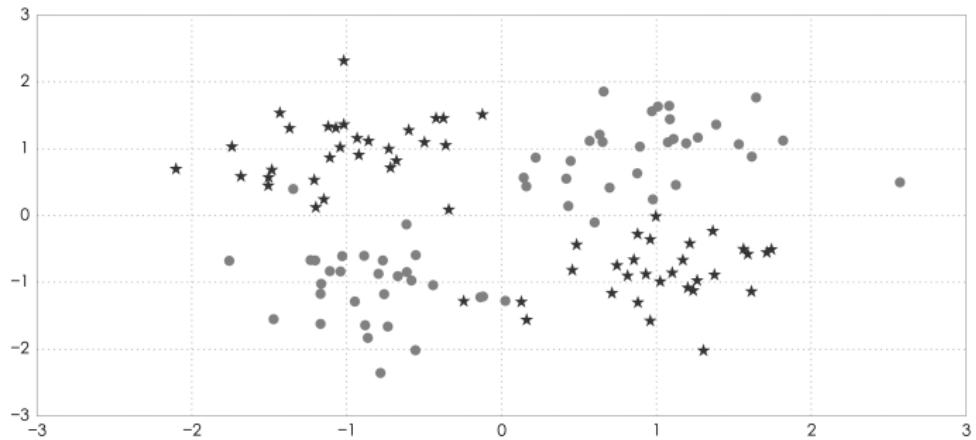
In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

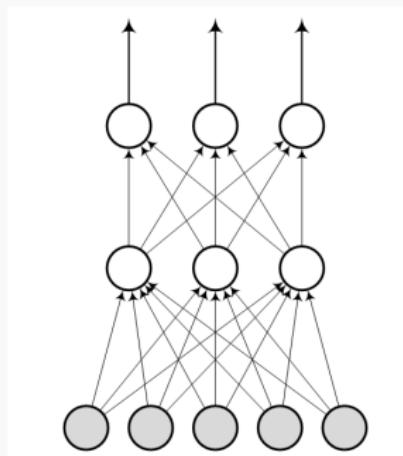
- 1960-е годы: изучали перцептроны.
- (Minsky, Papert, 1969): даже с нелинейностью один перцептрон задаёт только линейную разделяющую поверхность, XOR нельзя моделировать перцептроном...



- Это почему-то восприняли как большую проблему, которая ставит крест на нейронных сетях.

# ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- ...хм, ну конечно! Для нелинейных функций нужна сеть перцептронов:



## ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- (Brison, Hoback, 1969): предложили алгоритм обратного распространения ошибки (backpropagation).
- (Hinton, 1974): переоткрыл backpropagation, с тех пор он стал популярным.
- Во второй половине 1970-х появились многоуровневые ANN, была разработана современная теория.
- Глубокие модели появились в первой половине 1980-х гг.! Это вообще не очень хитрая идея сама по себе.

## ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Но к началу 1990-х решили, что нейронные сети использовать смысла нет, и началась «вторая зима».
- Это было потому, что тогда ни математически, ни вычислительно не могли нормально обучить большие модели.
- Нужно было обучать сети неделями, а качество, например, на распознавании речи проигрывало HMM, в других задачах проигрывало другим методам.
- John Denker, 1994: «neural networks are the second best way of doing just about anything».
- К концу 90-х на нейросетях остались только обработка изображений (Yann LeCun) и несколько других групп (Geoffrey Hinton, Yoshua Bengio).

## ИСТОРИЯ ANN И ПЕРЦЕПТРОН

- Революция deep learning начинается в 2006: в группе Джейфри Хинтона изобрели Deep Belief Networks (DBN).
- Основная идея: научились делать обучение без учителя, выделять признаки (при помощи машин Больцмана, RBM; появились в 1983), использовать это как предобучение.
- Компьютеры стали мощнее, появились способы передать вычисления на GPU, и вычислительно мы смогли обрабатывать гораздо более объёмные датасеты, которые как раз стали доступными.
- А потом появились новые методы регуляризации (дропаут, затем batch normalization), и RBM с предобучением тоже стали не очень нужны.
- Вот основные компоненты революции глубокого обучения.
- Осталось только понять, что всё это значит. :)

# ML recap

# Постановка задачи: обучение с учителем

Объекты       $x_1, \dots, x_n$

Ответы       $y_1, \dots, y_n$

# Постановка задачи: обучение с учителем

Объекты       $x_1, \dots, x_n$

Ответы       $y_1, \dots, y_n$

Функция потерь     $L(\mathbf{y}, \hat{\mathbf{y}})$

# Постановка задачи: обучение с учителем

Объекты  $x_1, \dots, x_n$

Ответы  $y_1, \dots, y_n$

Алгоритм предсказания  $f(x, \theta)$

Функция потерь  $L(y, \hat{y})$

**Задача:** настроить  $\theta$

# Постановка задачи: обучение с учителем

Объекты  $x_1, \dots, x_n$

Ответы  $y_1, \dots, y_n$

Алгоритм предсказания  $f(x, \theta)$

Функция потерь  $L(\mathbf{y}, \hat{\mathbf{y}})$

**Задача:** настроить  $\theta$

$$\mathbb{E}_{(x,y)} L(\mathbf{y}, f(x, \theta)) \rightarrow \min_{\theta}$$

# Постановка задачи: обучение с учителем

Объекты  $x_1, \dots, x_n$

Ответы  $y_1, \dots, y_n$

Алгоритм предсказания  $f(x, \theta)$

Функция потерь  $L(\mathbf{y}, \hat{\mathbf{y}})$

**Задача:** настроить  $\theta$

$$\mathbb{E}_{(x,y)} L(\mathbf{y}, f(x, \theta)) \rightarrow \min_{\theta}$$

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

**эмпирический риск**

# Пример: бинарная классификация

Объекты  $x_1, \dots, x_n$

Алгоритм предсказания  $f(x, \theta)$

Функция потерь  $L(y, \hat{y})$

Ответы  $y_1, \dots, y_n \in [0,1]$



**Задача:** настроить  $\theta$

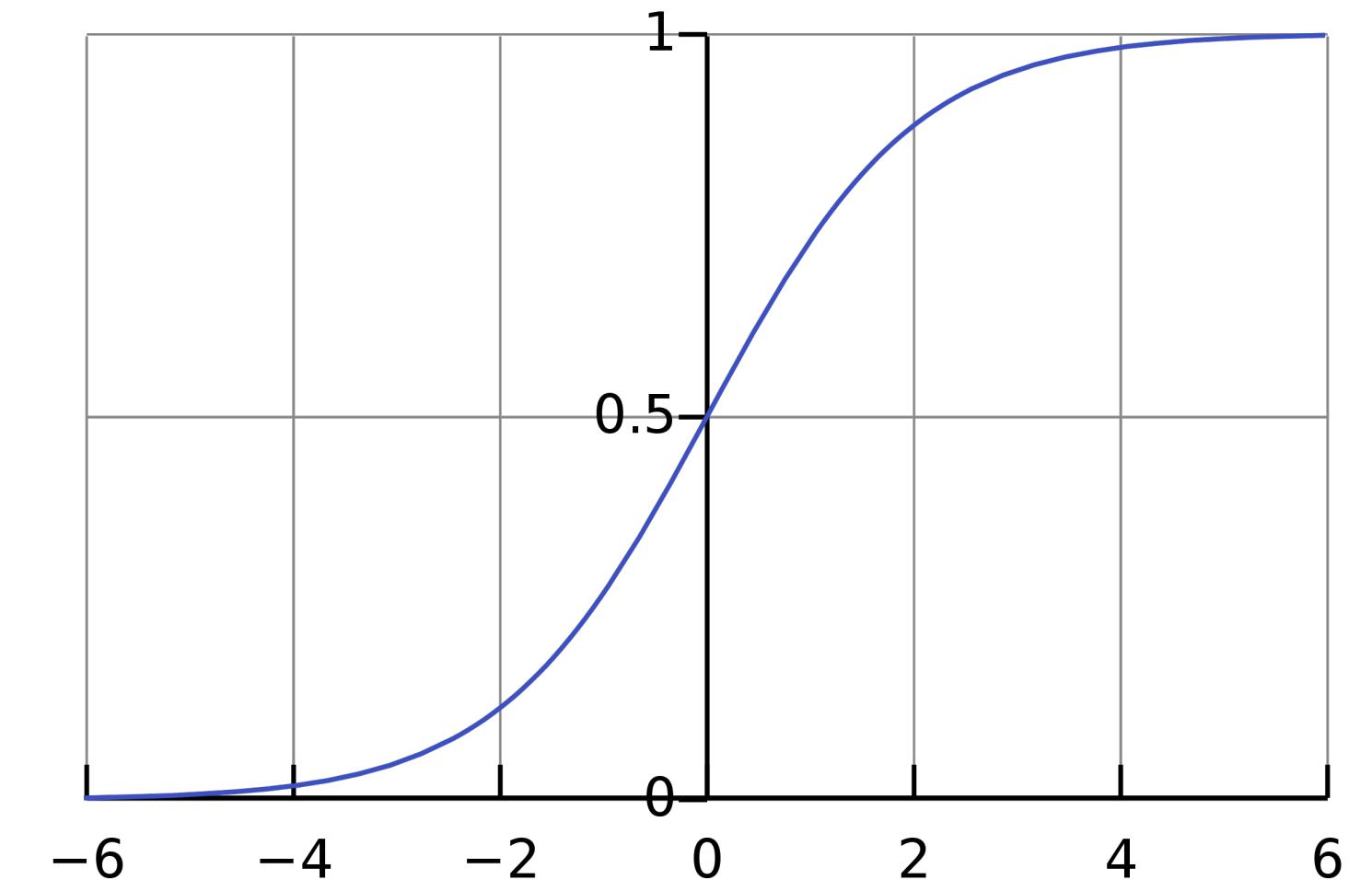
# Логистическая регрессия

Объекты  $x_1, \dots, x_n$

Ответы  $y_1, \dots, y_n \in [0,1]$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$

$$P(y = 1 | x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$



# Логистическая регрессия

Объекты  $x_1, \dots, x_n$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$

Ответы  $y_1, \dots, y_n \in [0,1]$

$$P(y = 1 | x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Функция потерь

$$L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_i [y_i \log P(y = 1 | X, \theta) + (1 - y_i) \log(1 - P(y = 1 | X, \theta))]$$



# Логистическая регрессия

	какие признаки?	
Объекты	$x_1, \dots, x_n$	
Ответы	$y_1, \dots, y_n \in [0,1]$	
Алгоритм предсказания	$f(x, \theta) = P(y = 1   x, \theta) > \frac{1}{2}$	
		
	$P(y = 1   x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$	
Функция потерь		

$$L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_i [y_i \log P(y = 1 | X, \theta) + (1 - y_i) \log(1 - P(y = 1 | X, \theta))]$$

# Логистическая регрессия

Объекты  $x_1, \dots, x_n$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$

Ответы  $y_1, \dots, y_n \in [0,1]$



$$P(y = 1 | x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Функция потерь

$$L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_i [y_i \log P(y = 1 | X, \theta) + (1 - y_i) \log(1 - P(y = 1 | X, \theta))]_i$$

# Логистическая регрессия

Объекты  $x_1, \dots, x_n$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$

Ответы  $y_1, \dots, y_n \in [0,1]$



$$P(y = 1 | x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Функция потерь

$$L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_i [y_i \log P(y = 1 | X, \theta) + (1 - y_i) \log(1 - P(y = 1 | X, \theta))]$$

для линейно разделяемых классов

# Логистическая регрессия

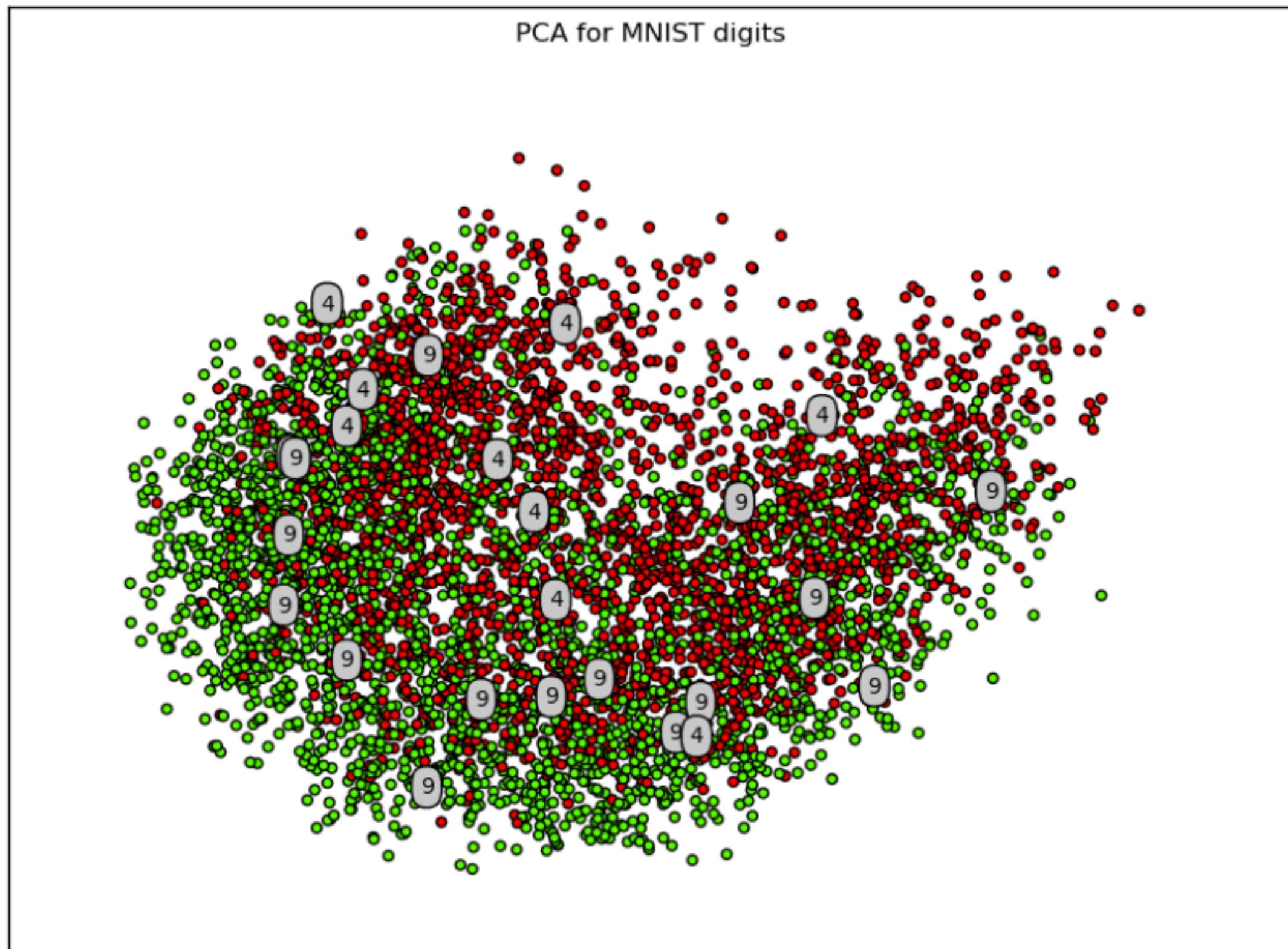


Image credit



# Логистическая регрессия

Объекты  $x_1, \dots, x_n$

Ответы  $y_1, \dots, y_n \in [0,1]$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$



$$P(y = 1 | x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Последовательное применение операций:

$$x \longrightarrow \theta^T x \longrightarrow \sigma(\theta^T x) \longrightarrow P(y = 1 | x, \theta)$$

# Логистическая регрессия

Объекты  $x_1, \dots, x_n$

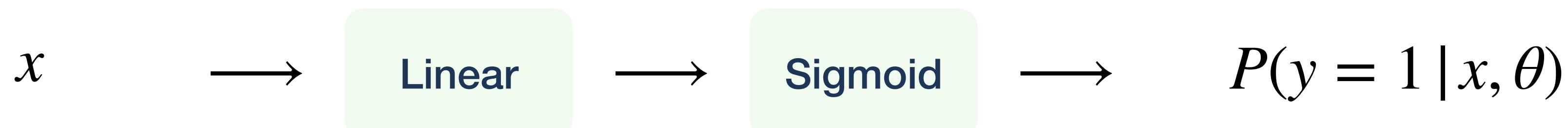
Ответы  $y_1, \dots, y_n \in [0,1]$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$



$$P(y = 1 | x, \theta) = \sigma(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Последовательное применение операций:



# Простейшая нейросеть

Объекты  $x_1, \dots, x_n$

Ответы  $y_1, \dots, y_n \in [0,1]$

Алгоритм предсказания  $f(x, \theta) = P(y = 1 | x, \theta) > \frac{1}{2}$

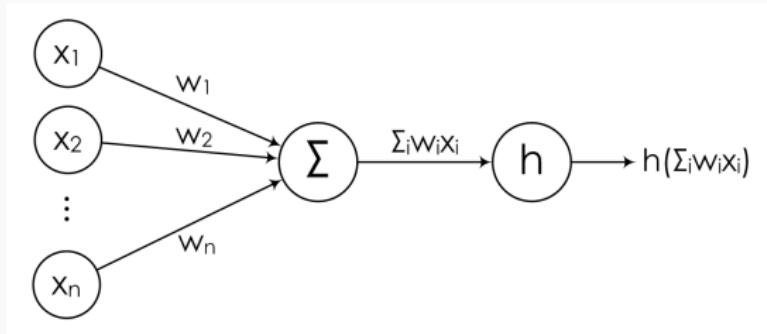


Последовательное применение операций:



**Нейросеть - это сложная  
функция**

- Основной компонент нейронной сети – *перцептрон*:



- Линейная комбинация входов, потом нелинейность:

$$y = h(w^\top x) = h \left( \sum_i w_i x_i \right).$$

- Обучение градиентным спуском.
- Для исходного перцептрана Розенблатта ( $h = \text{id}$ ):

$$E_P(\mathbf{w}) = - \sum_{x \in M} y(\mathbf{x}) (\mathbf{w}^\top \mathbf{x}),$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla_{\mathbf{w}} E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta t_n \mathbf{x}_n.$$

- Или, к примеру, можно делать классификацию, подставляя в качестве функции активации логистический сигмоид  $h(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ :

$$E(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N (y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1 - y_i) \log (1 - \sigma(\mathbf{w}^\top \mathbf{x}_i))).$$

- По сути это просто логистическая регрессия.

# Простейшая нейросеть



# Простейшая нейросеть



Нейросеть - это сложная функция

Можно не придумывать сложные признаки (feature engineering)

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

Блок - слой нейросети, вычисляющий определенную функцию

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

Блок - слой нейросети, вычисляющий определенную функцию

Linear

линейный слой

$$x_{output} = W \cdot x_{input}$$

параметры:  $W$  - матрица размера  $d1 \times d2$  - веса линейного слоя

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

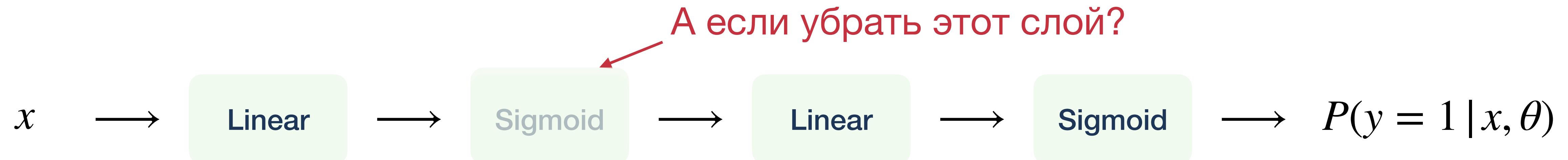
Блок - слой нейросети, вычисляющий определенную функцию

Sigmoid

нелинейность или функция активации (сигмоида)

$$x_{output} = \sigma(x_{input}) = \frac{1}{1 + e^{-x_{input}}}$$

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

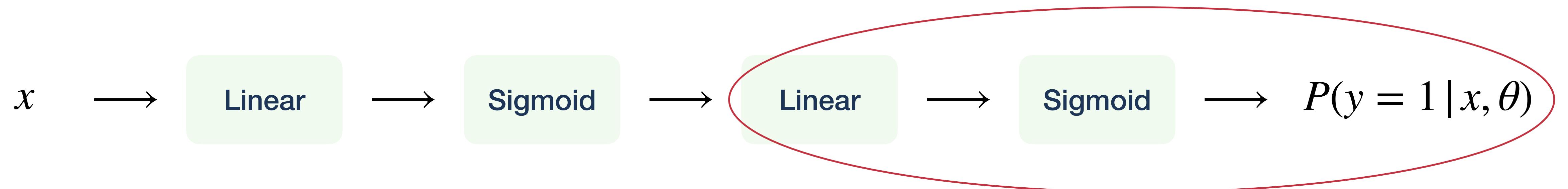
Блок - слой нейросети, вычисляющий определенную функцию

Sigmoid

нелинейность или функция активации (сигмоида)

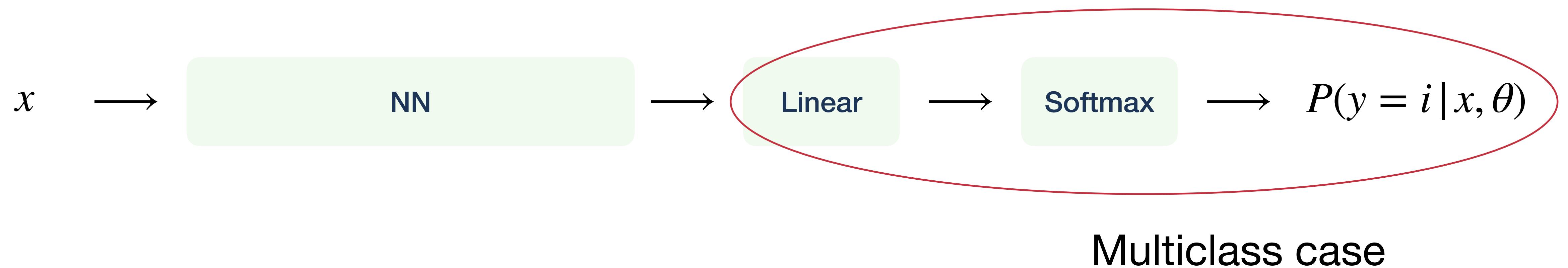
$$x_{output} = \sigma(x_{input}) = \frac{1}{1 + e^{-x_{input}}}$$

# Простейшая нейросеть



Эта часть есть в почти всех современных нейросетях!

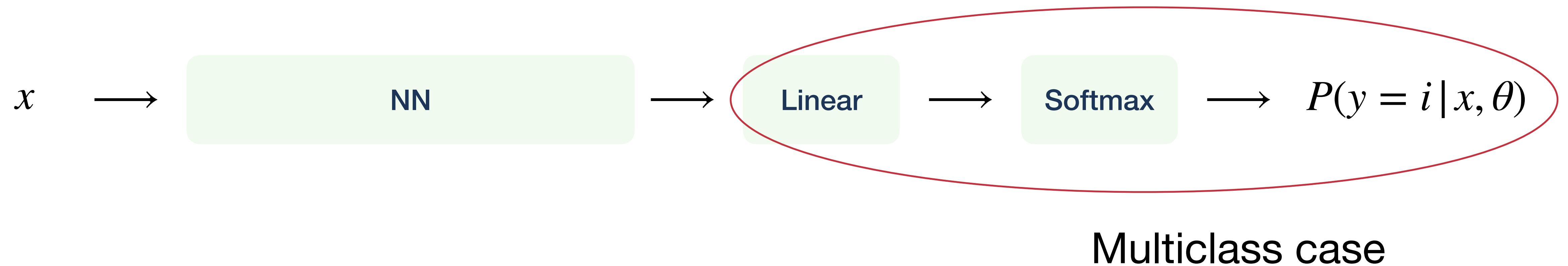
# Простейшая нейросеть



Softmax

$$x_{output_i} = \frac{e^{x_{input_i}}}{\sum_j e^{x_{input_j}}}$$

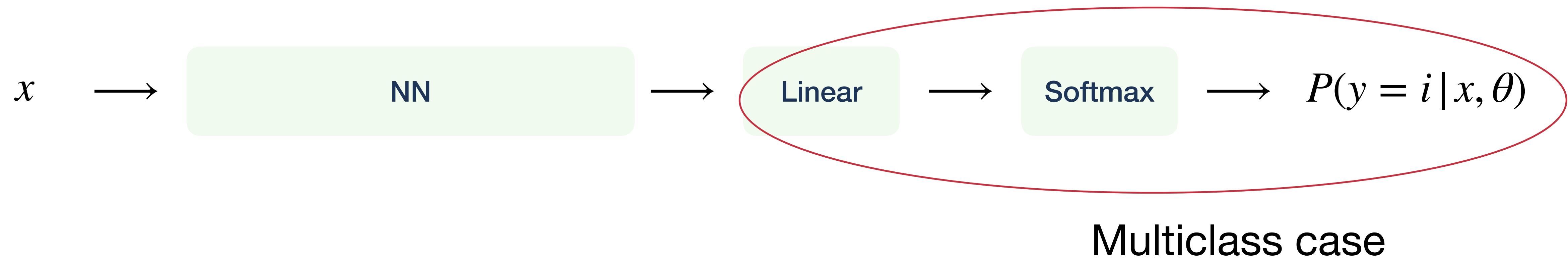
# Простейшая нейросеть



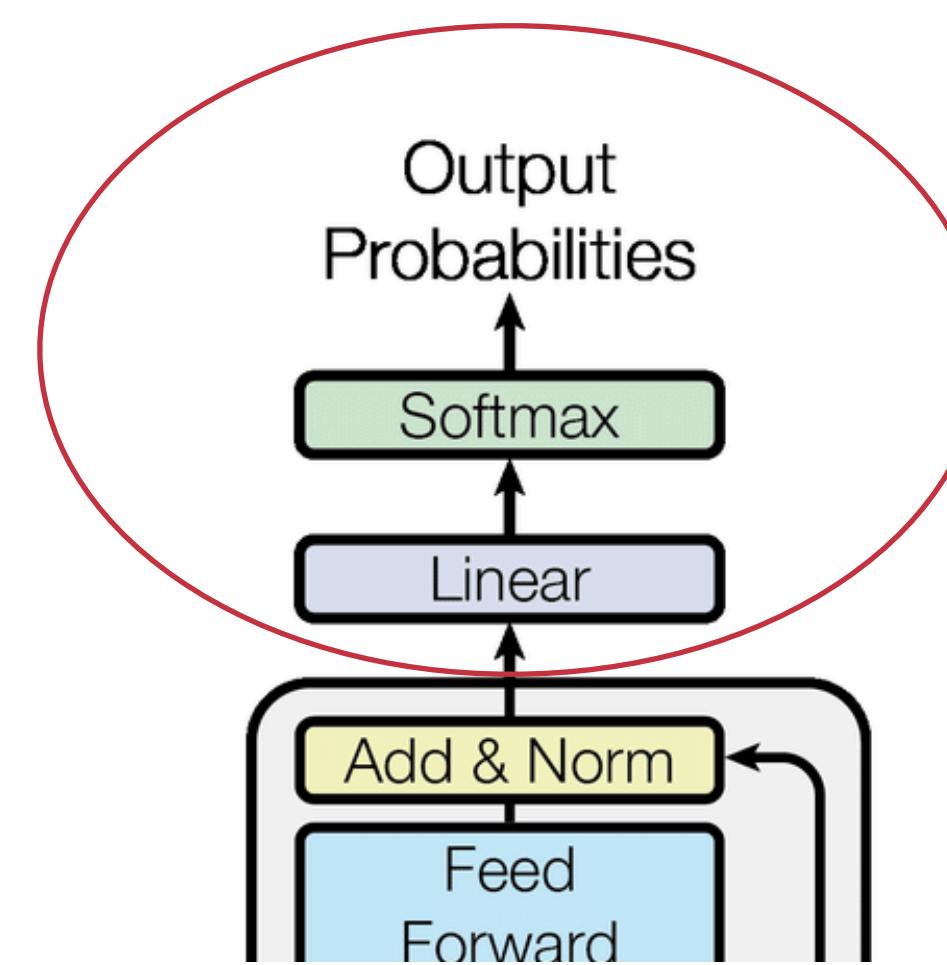
Linear

параметры:  $W$  - матрица размера  $d \times n\_classes$

# Простейшая нейросеть



Transformer last layers



# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

Блок - слой нейросети, вычисляющий определенную функцию

Как обучать?

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

Блок - слой нейросети, вычисляющий определенную функцию

Как обучать?

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

задача оптимизации

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

Блок - слой нейросети, вычисляющий определенную функцию

Как обучать?

Градиентный спуск:

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$
$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

задача оптимизации

методы оптимизации  
1го порядка

## КЛЮЧЕВЫЕ АСПЕКТЫ ОБУЧЕНИЯ

**Инициализация весов:** В самом начале обучения параметры персептрона (веса нейронов) устанавливаются случайным образом. Это необходимо для того, чтобы сеть могла начать процесс настройки и адаптации к входным данным.

**Алгоритм обратного распространения ошибки (Backpropagation):** Одним из основополагающих методов является механизм обратного распространения ошибки. Он позволяет корректировать веса на основе разницы между ожидаемым и полученным результатами, сокращая ошибку сети.

**Функция активации:** Функции активации преобразуют входные сигналы в выходные, обеспечивая переход от линейных к нелинейным преобразованиям. Выбор функции активации может значительно влиять на скорость и качество обучения.

**Нормализация данных:** Подготовка данных важных для обучения шаг включает нормализацию входных данных. Это помогает улучшить сходимость алгоритма и повысить точность результатов.

# Простейшая нейросеть



Нейросеть - это сложная функция

Можно представить в виде вычислительного графа:

Блок - слой нейросети, вычисляющий определенную функцию

Как обучать?

Градиентный спуск:

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$
$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

Как посчитать градиенты?

# Обучение нейросети

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

Численные методы:

$$\frac{df(x, \theta)}{d\theta} = \frac{f(x, \theta + eps) - f(x, \theta - eps)}{2eps}$$

Градиентный спуск:

$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

- конечные разности

# Обучение нейросети

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

Численные методы:

$$\frac{df(x, \theta)}{d\theta} = \frac{f(x, \theta + eps) - f(x, \theta - eps)}{2eps}$$

Градиентный спуск:

$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

- конечные разности  
хороший unit test

# Обучение нейросети

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

Градиентный спуск:

$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

Обычно блоки дифференцируемые - можно посчитать “на бумаге”

# Обучение нейросети

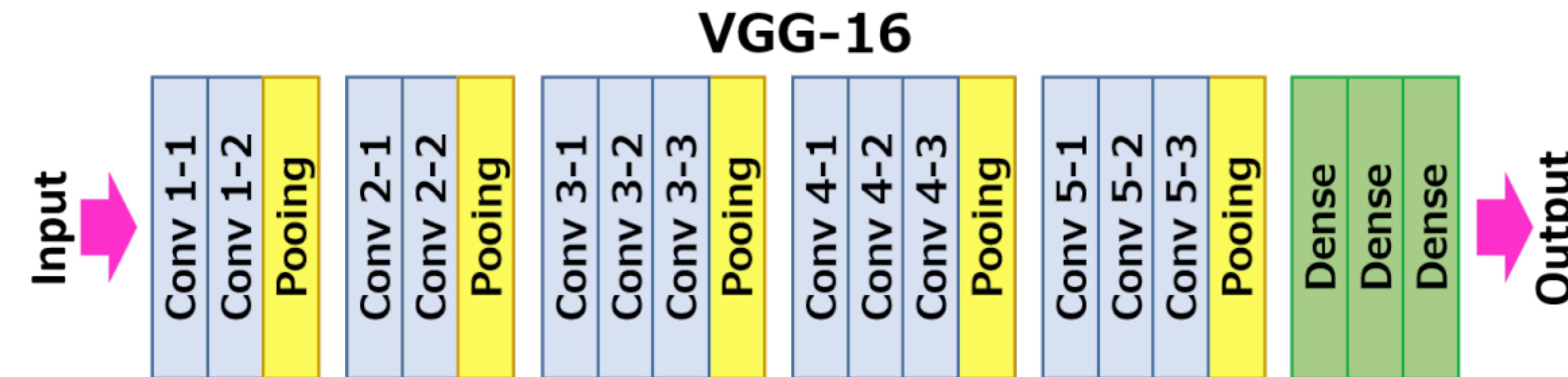
$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

Градиентный спуск:

$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

Обычно блоки дифференцируемые - можно посчитать “на бумаге”

БЛОКОВ МОЖЕТ БЫТЬ ОЧЕНЬ МНОГО



# Backpropagation

$$\frac{1}{n} \sum_i L(y_i, f(x_i, \theta)) \rightarrow \min_{\theta}$$

Градиентный спуск:

$$\theta_{t+1} = \theta_t - \alpha \frac{dL}{d\theta}$$

Chain rule (дифференцирование сложной функции):  $\frac{dL}{d\theta} = \frac{dL}{dz} \frac{dz}{d\theta}$

# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

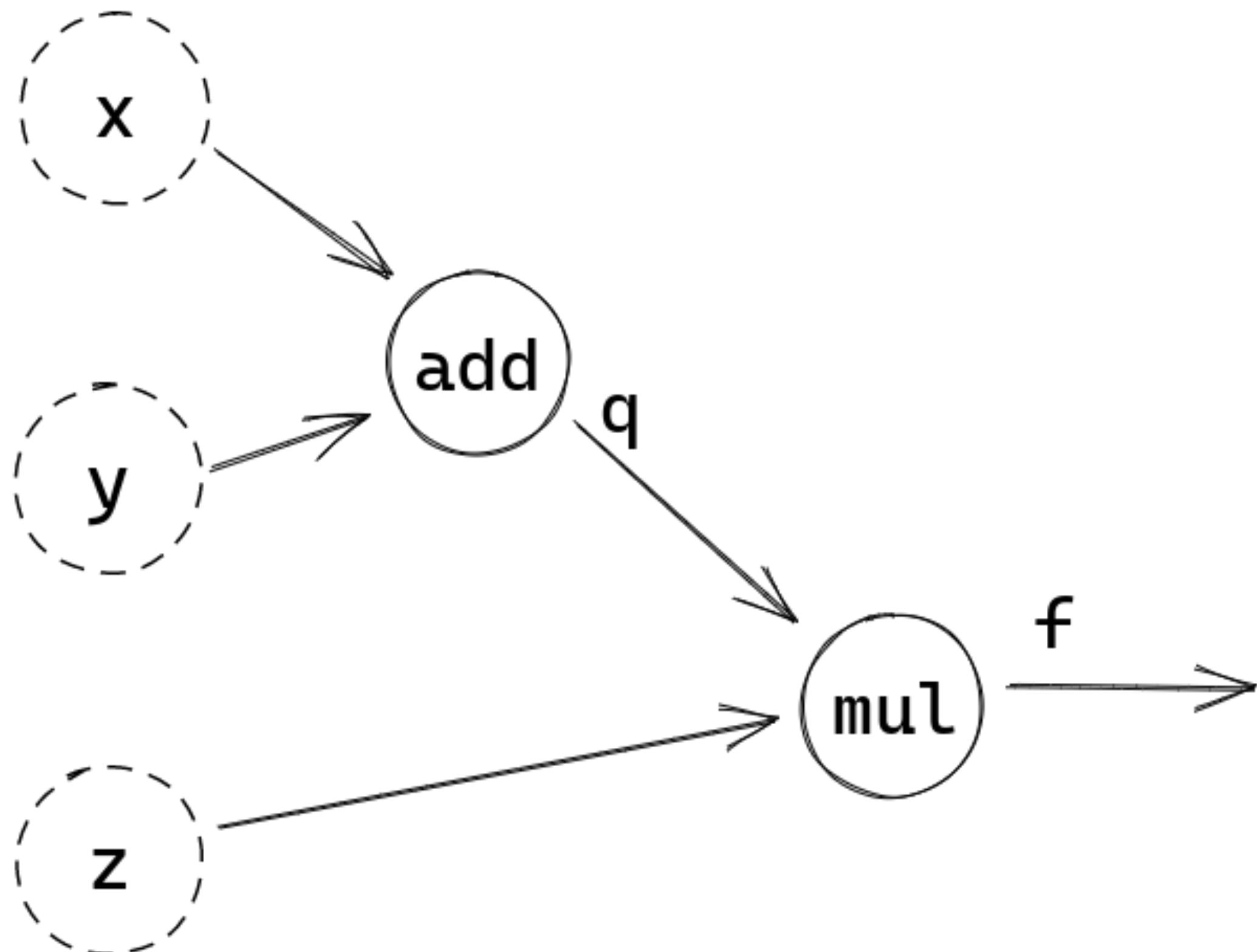
# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y$$

$$f = qz$$



# Backpropagation: простой пример

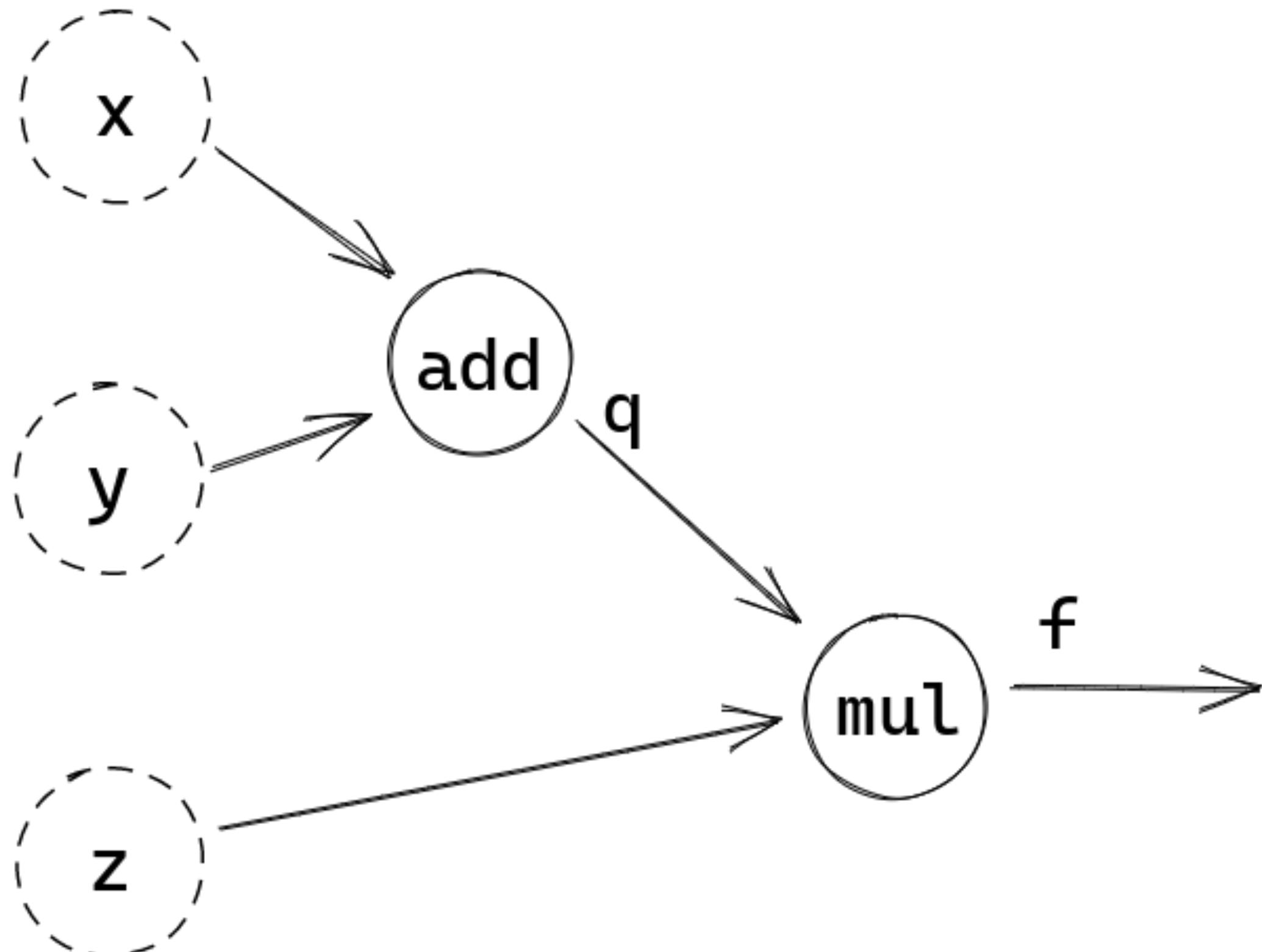
Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y$$

$$f = qz$$

$$x = 1, y = -2, z = 3$$



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

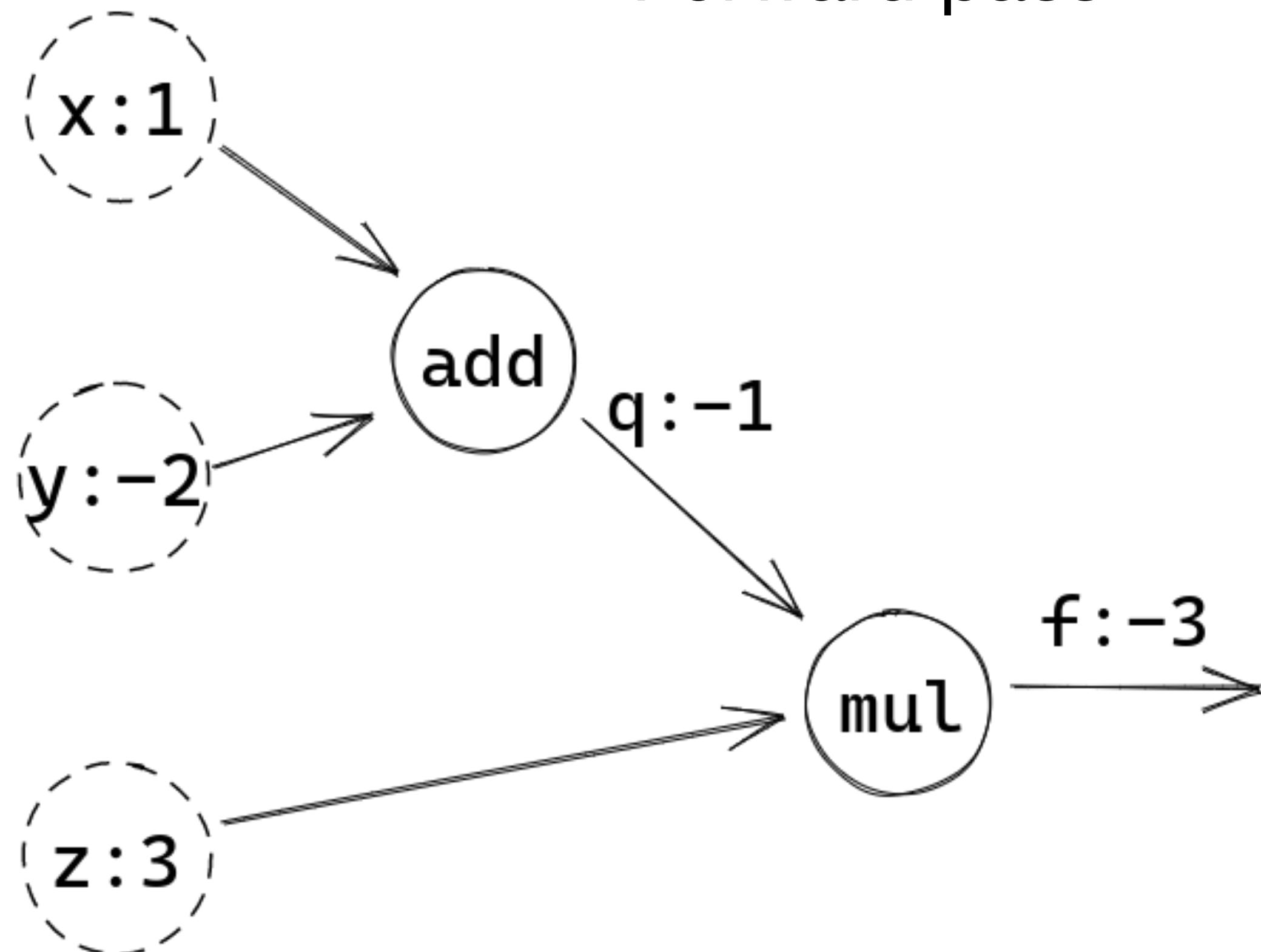
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Forward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

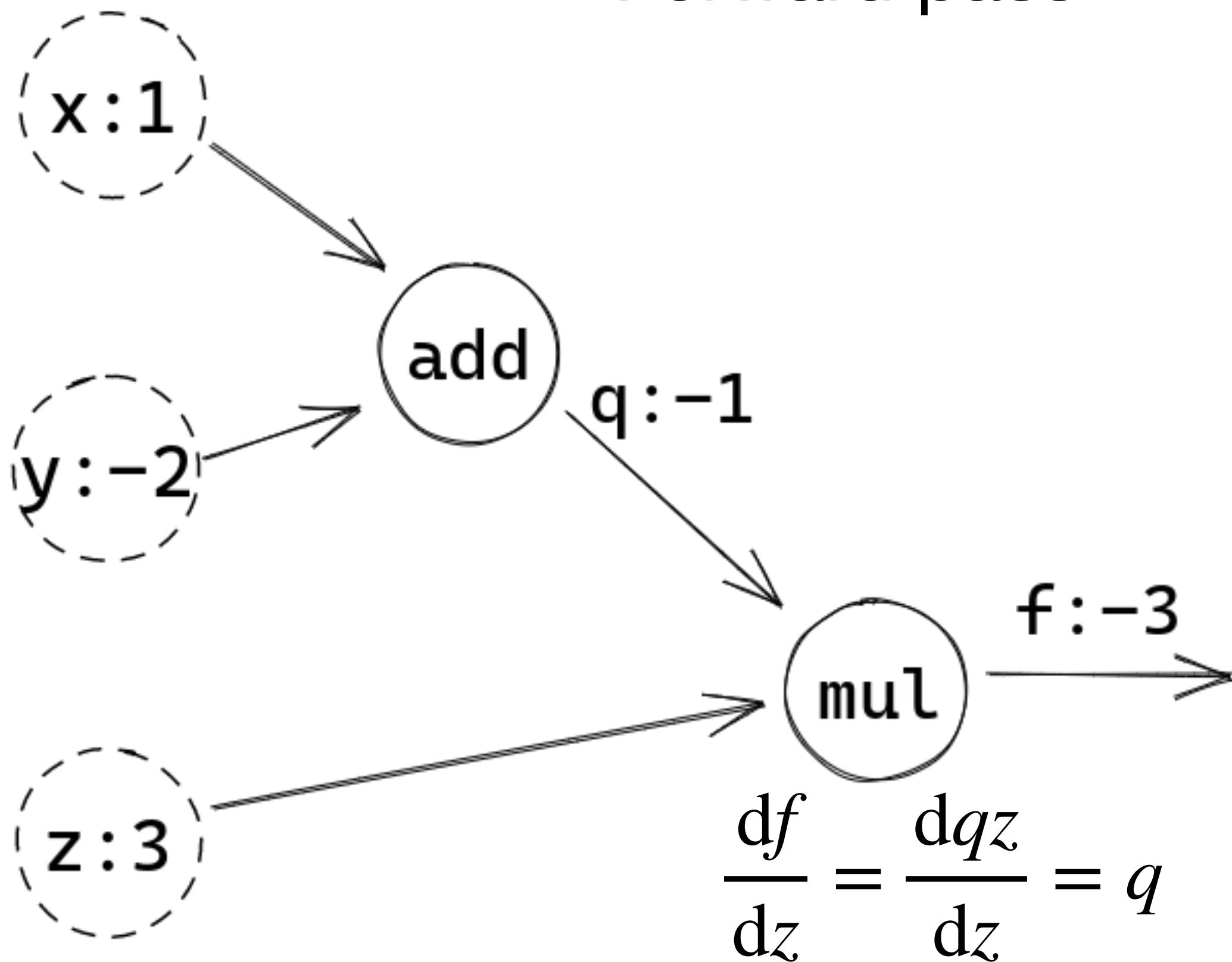
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Forward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

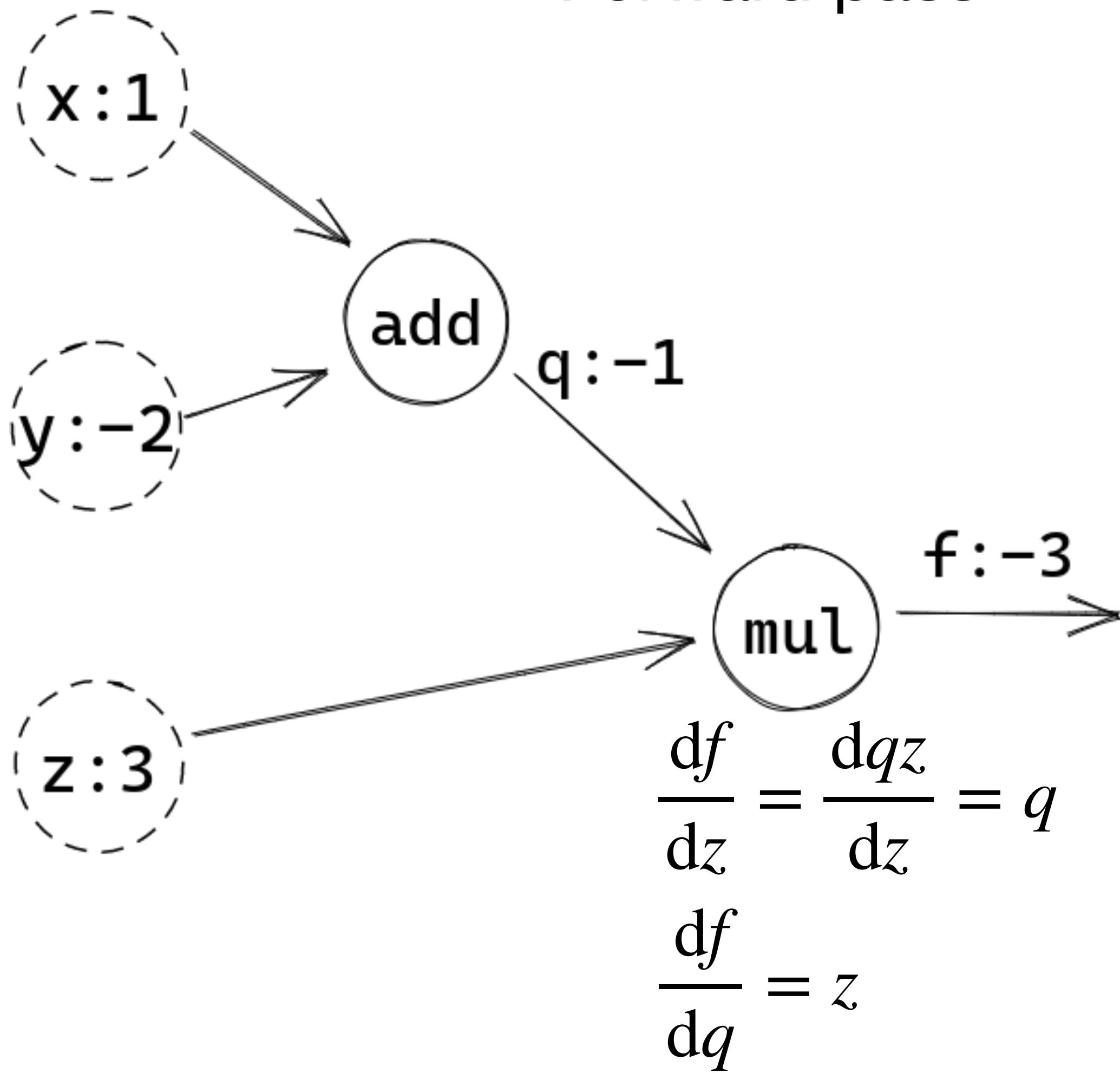
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Forward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

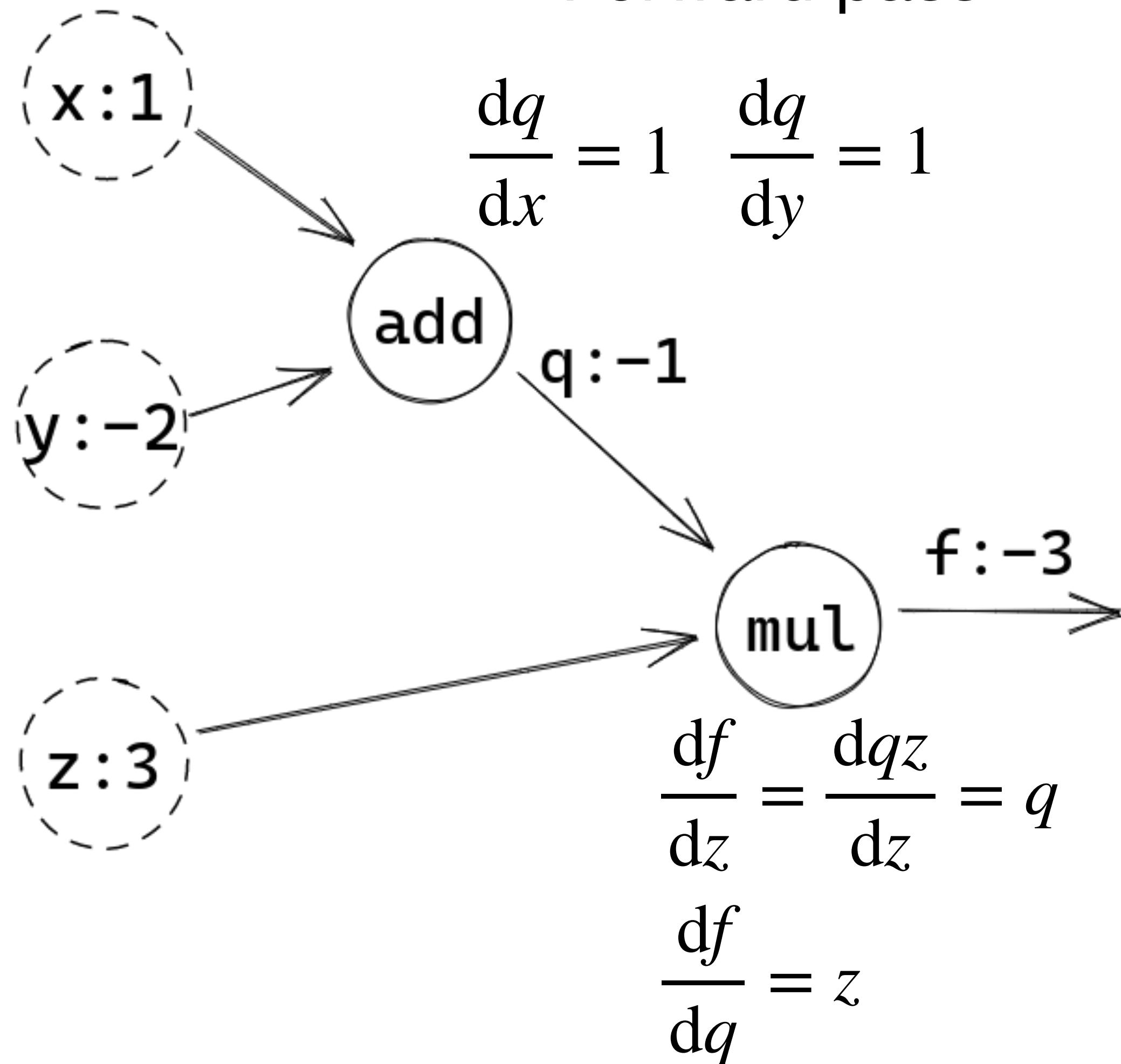
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Forward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

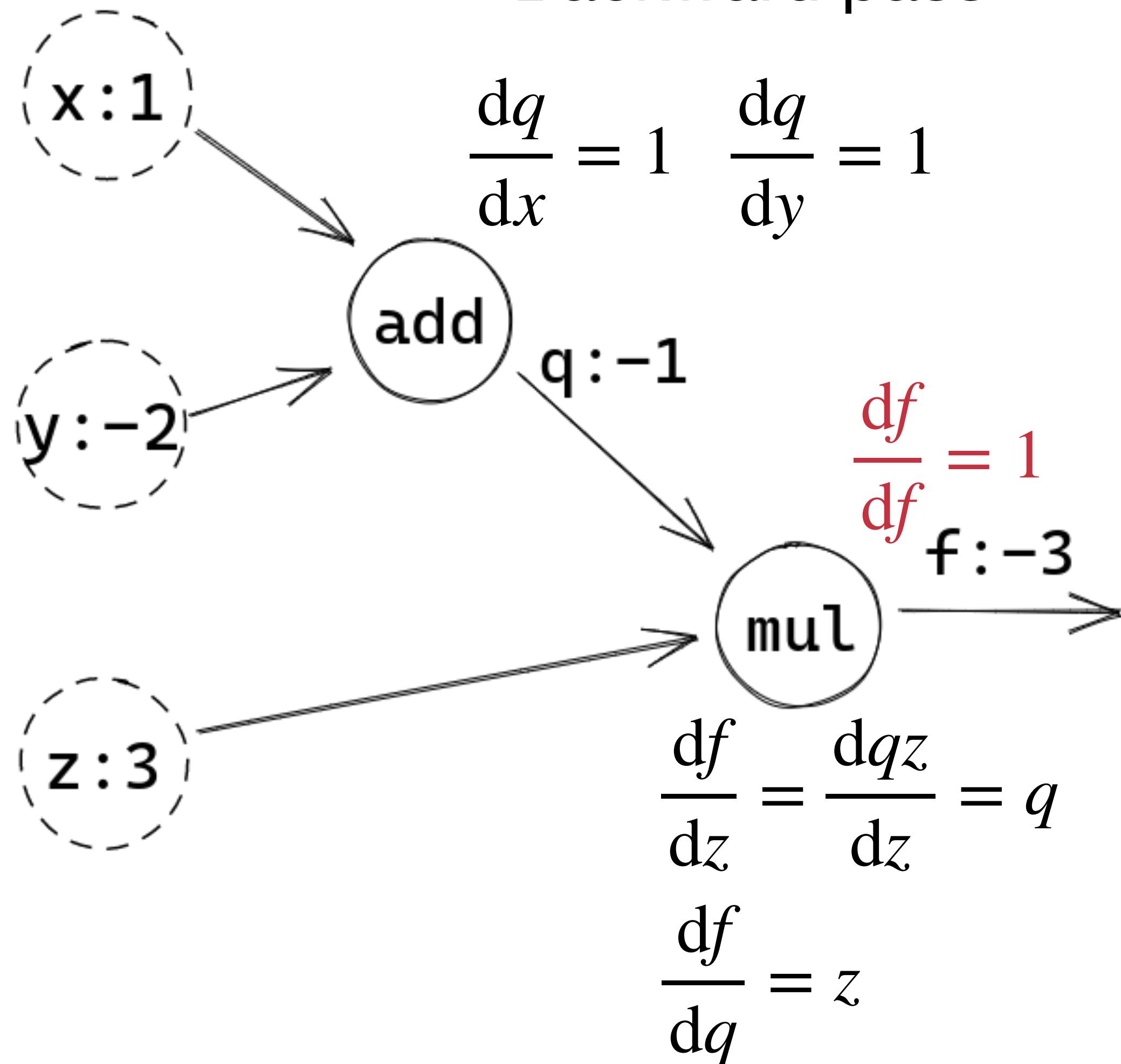
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Backward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

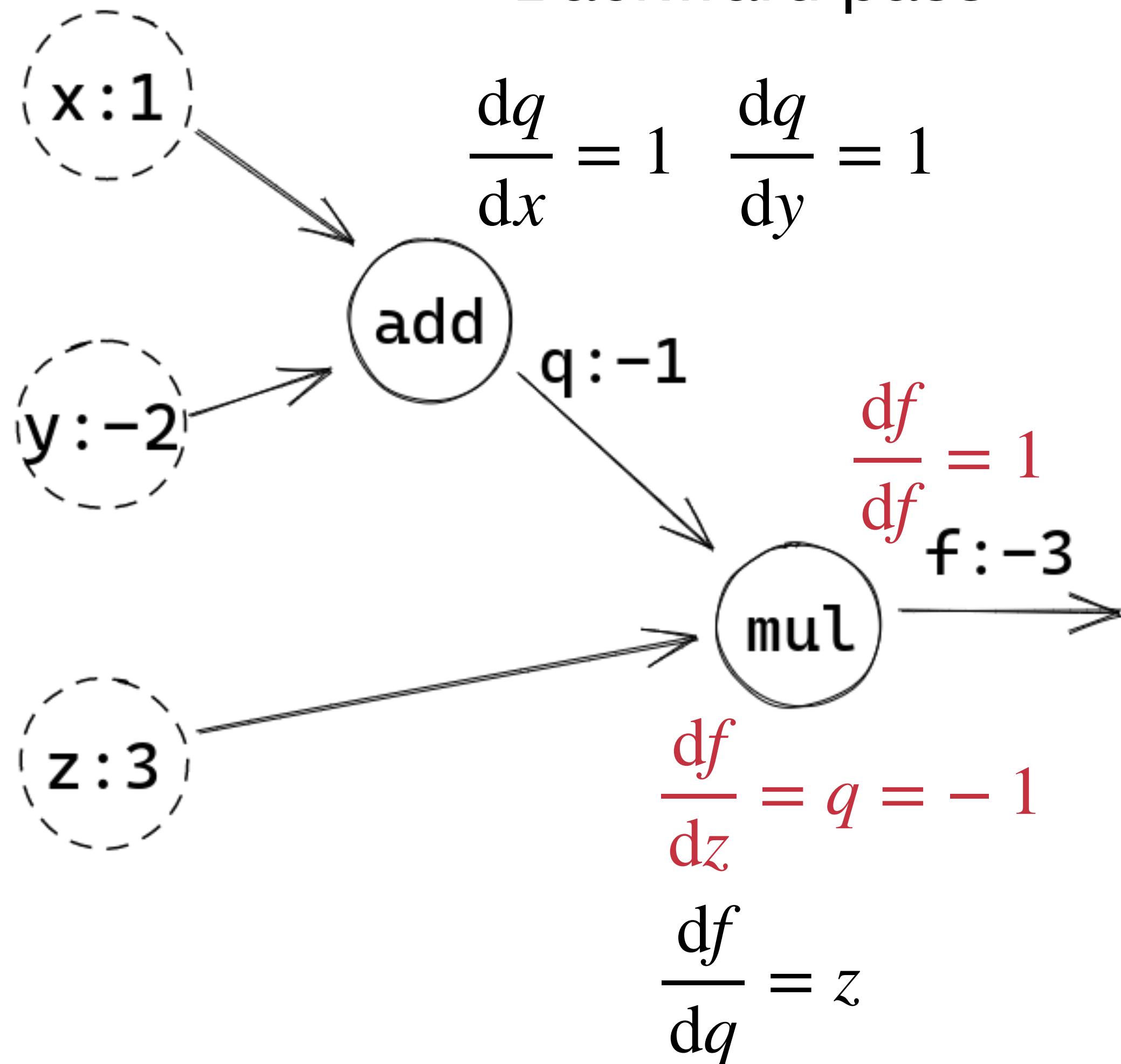
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Backward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

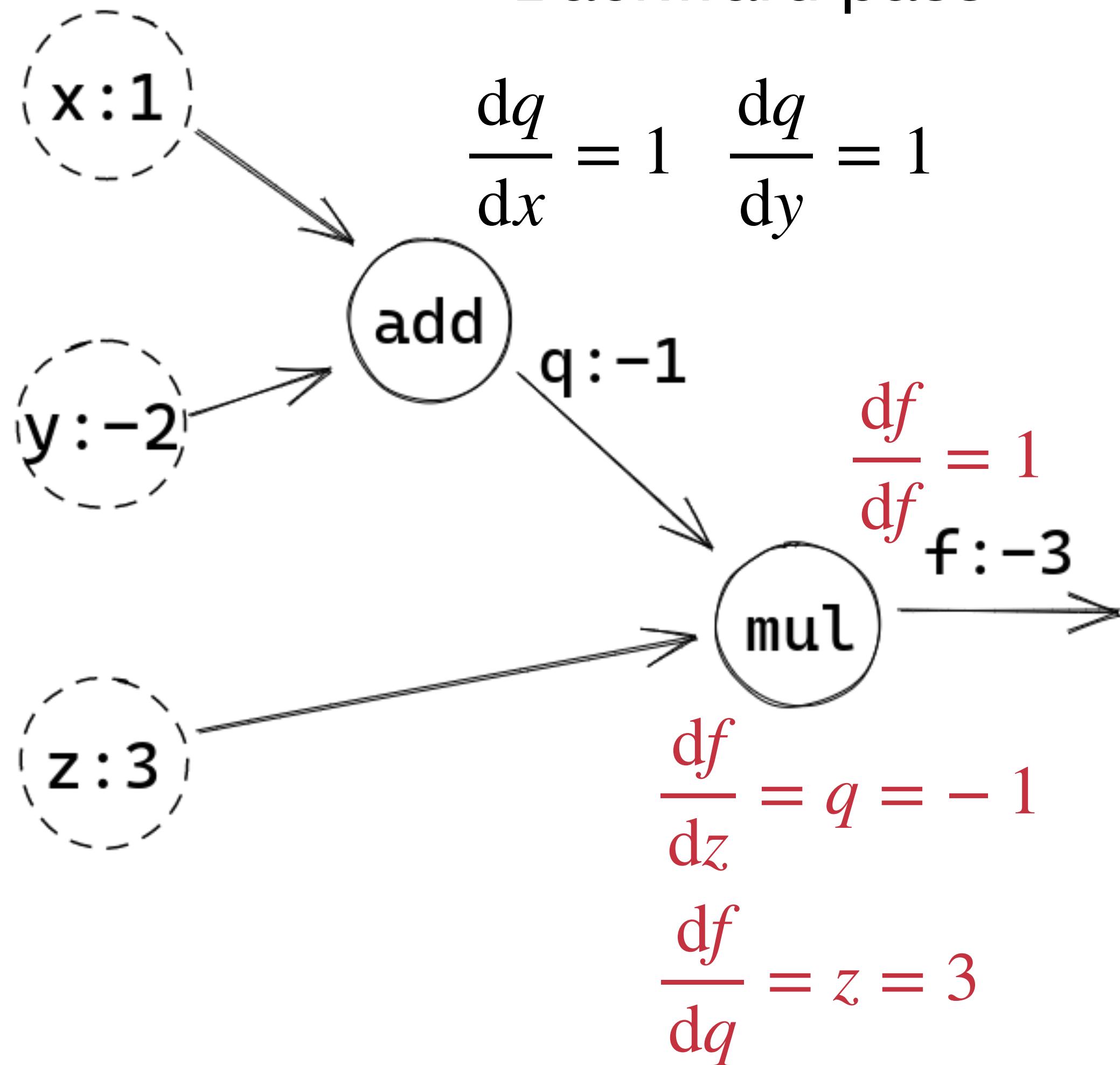
Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

Backward pass



# Backpropagation: простой пример

Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

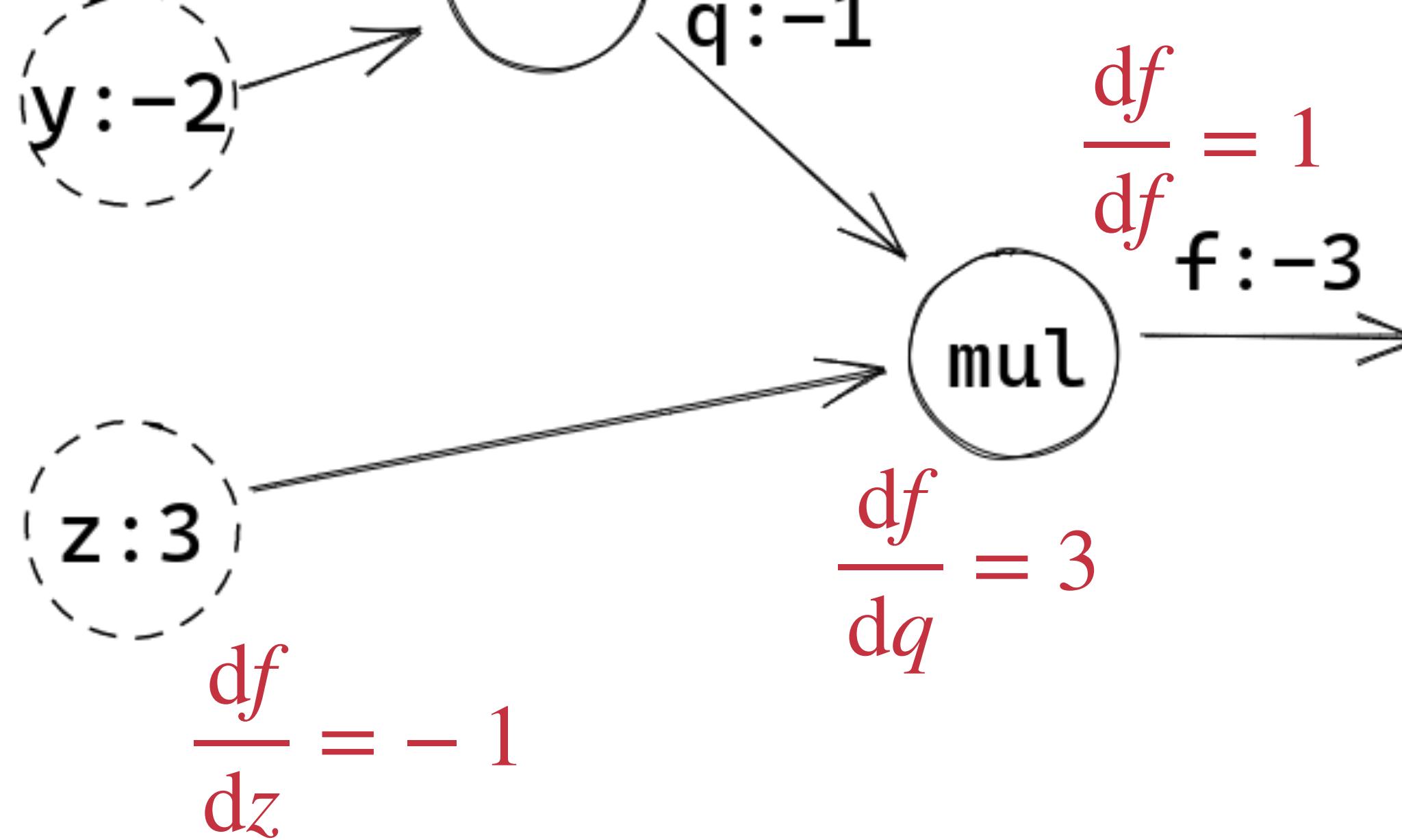
$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$

$$\frac{df}{dx} = \frac{df}{dq} \frac{dq}{dx}$$

Backward pass

$$\frac{dq}{dx} = 1 \quad \frac{dq}{dy} = 1$$



# Backpropagation: простой пример

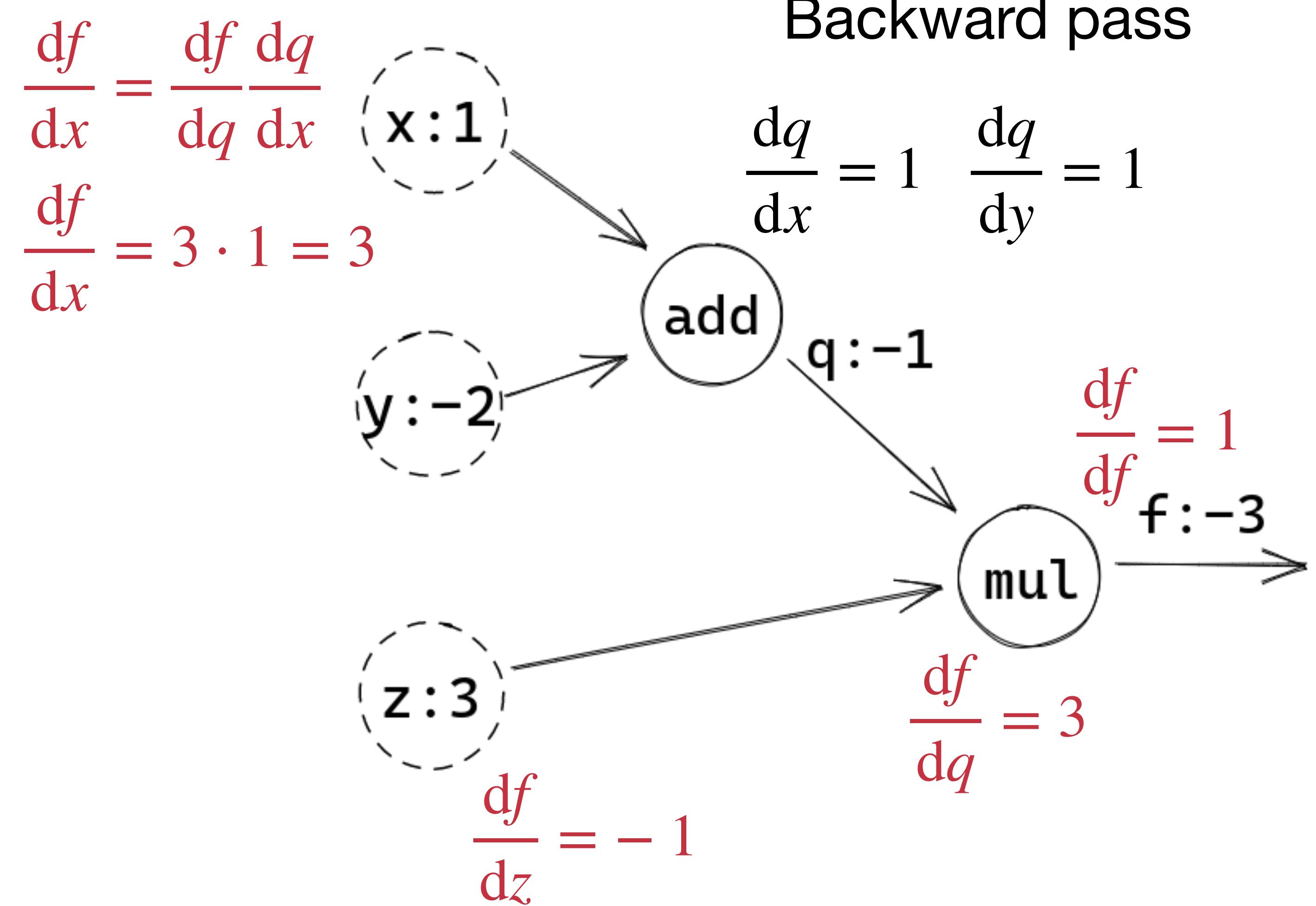
Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$



# Backpropagation: простой пример

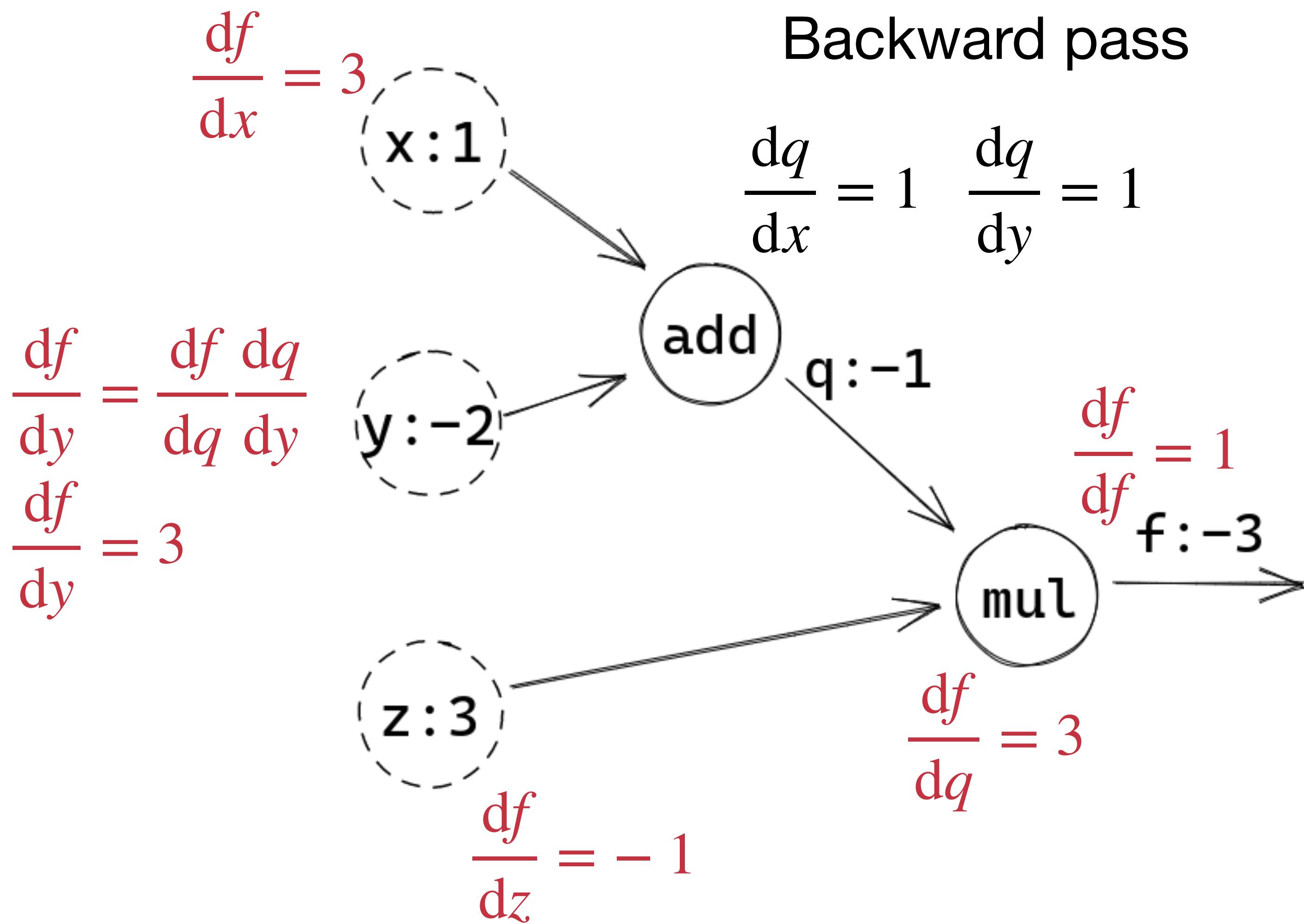
Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$



# Backpropagation: простой пример

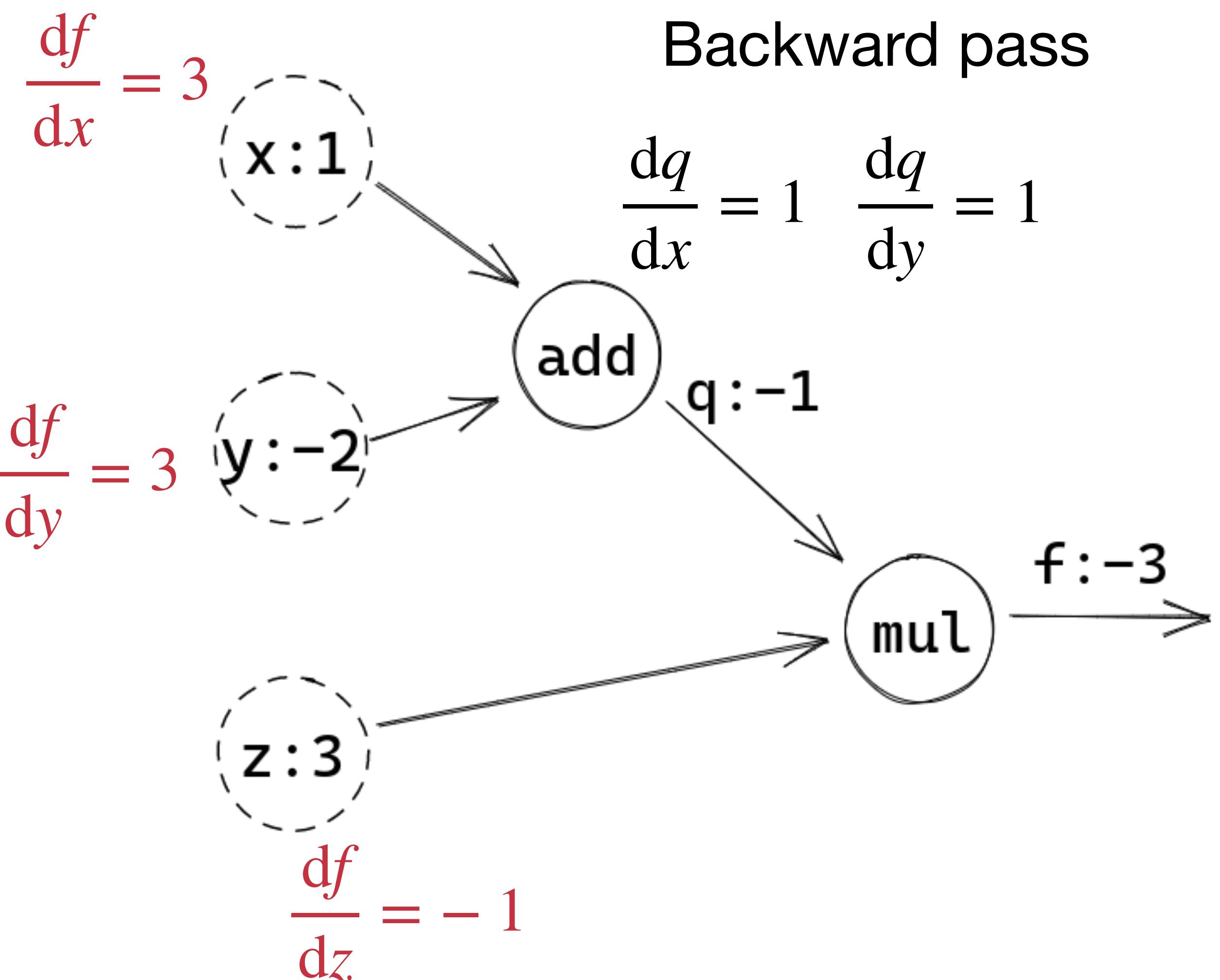
Дано:  $f(x, y, z) = (x + y)z$

Вычислить:  $\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}$

$$q = x + y = 1 - 2 = -1$$

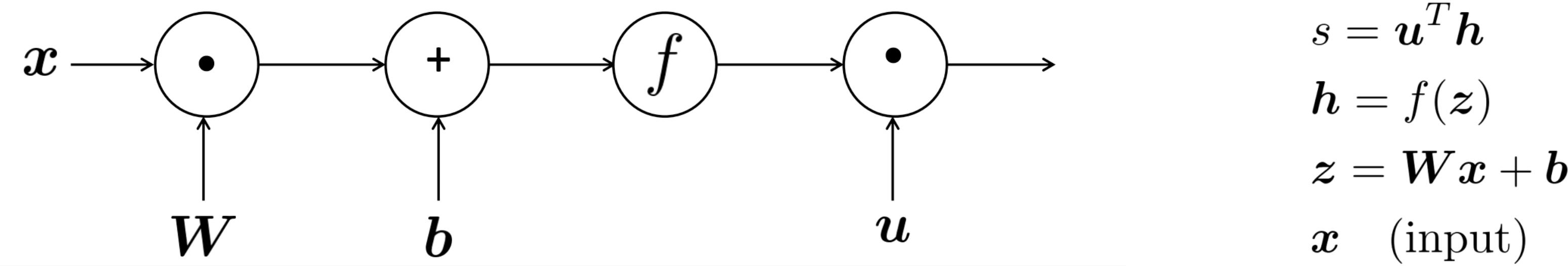
$$f = qz = -1 \cdot 3 = -3$$

$$x = 1, y = -2, z = 3$$



# Backpropagation в нейросетях

Представление нейросети в виде вычислительного графа

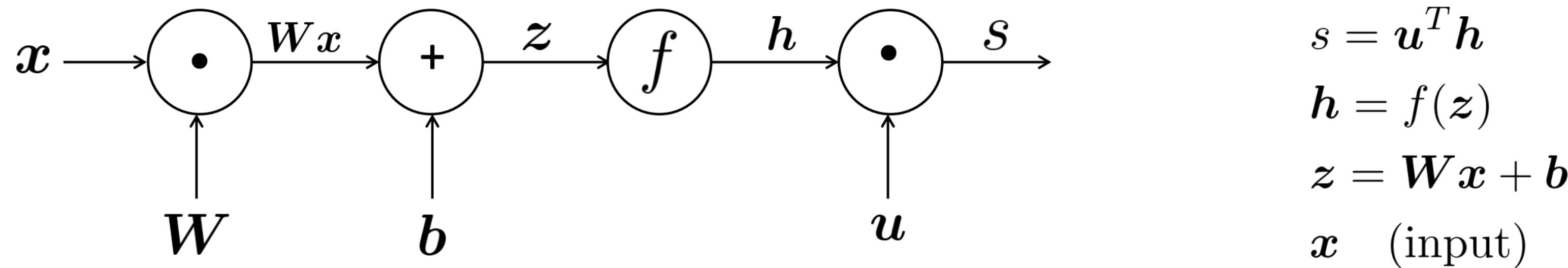


[Image credit](#)

# Backpropagation в нейросетях

Представление нейросети в виде вычислительного графа

**Forward pass:** вычисление результатов всех операций



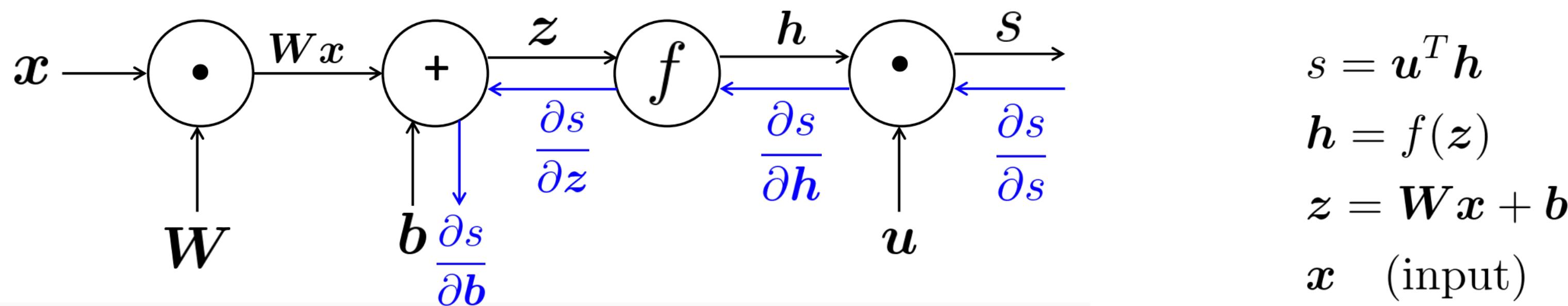
[Image credit](#)

# Backpropagation в нейросетях

Представление нейросети в виде вычислительного графа

**Forward pass:** вычисление результатов всех операций

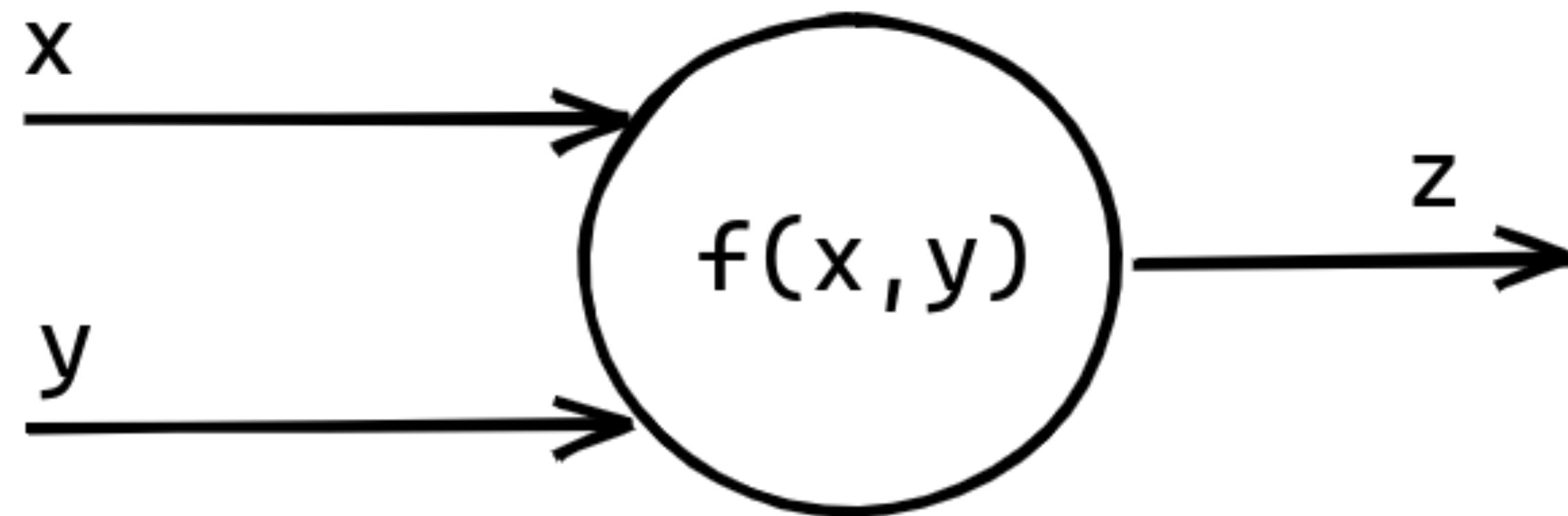
**Backward pass:** вычисление всех градиентов



[Image credit](#)

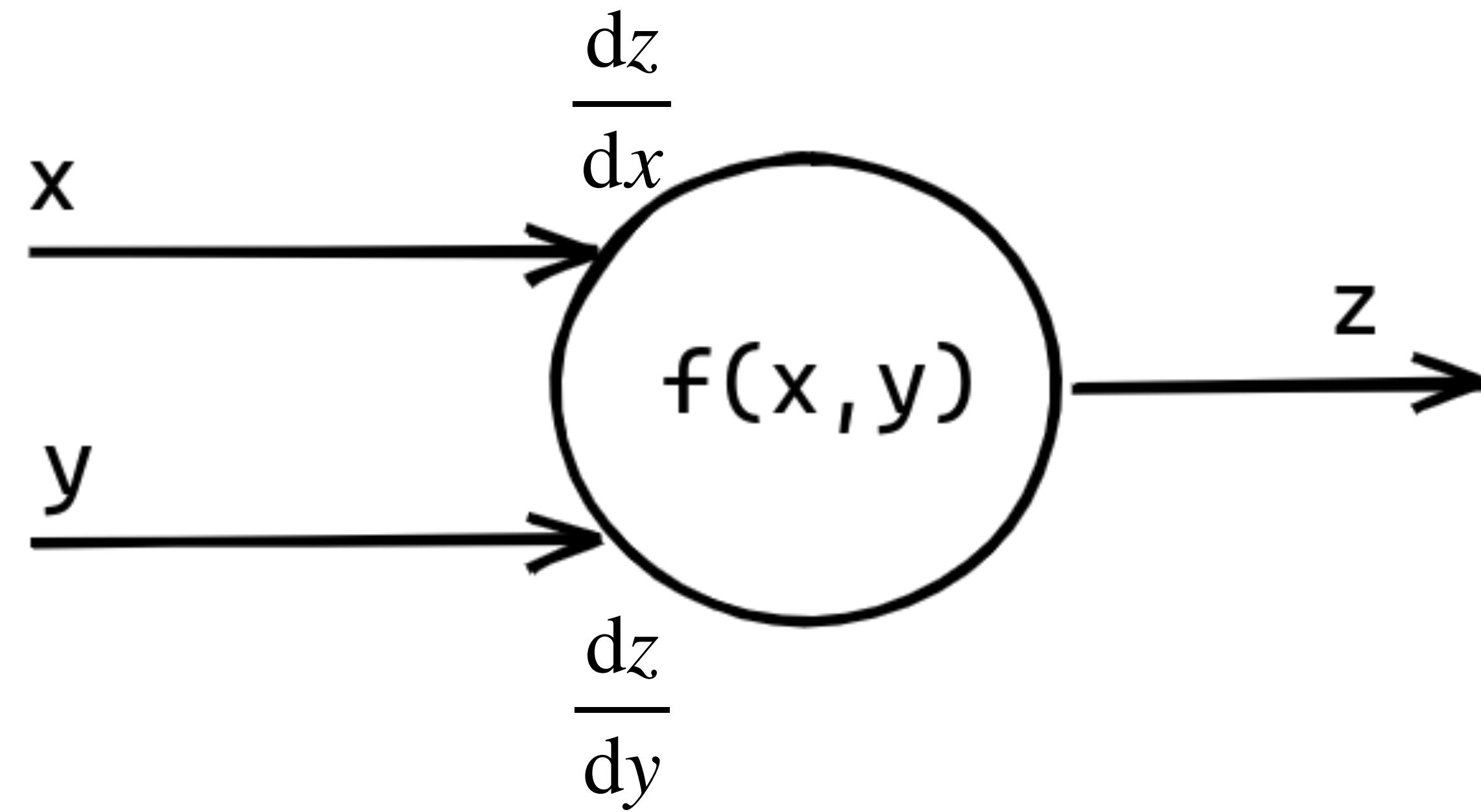
# Backpropagation: одна вершина

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



# Backpropagation: одна вершина

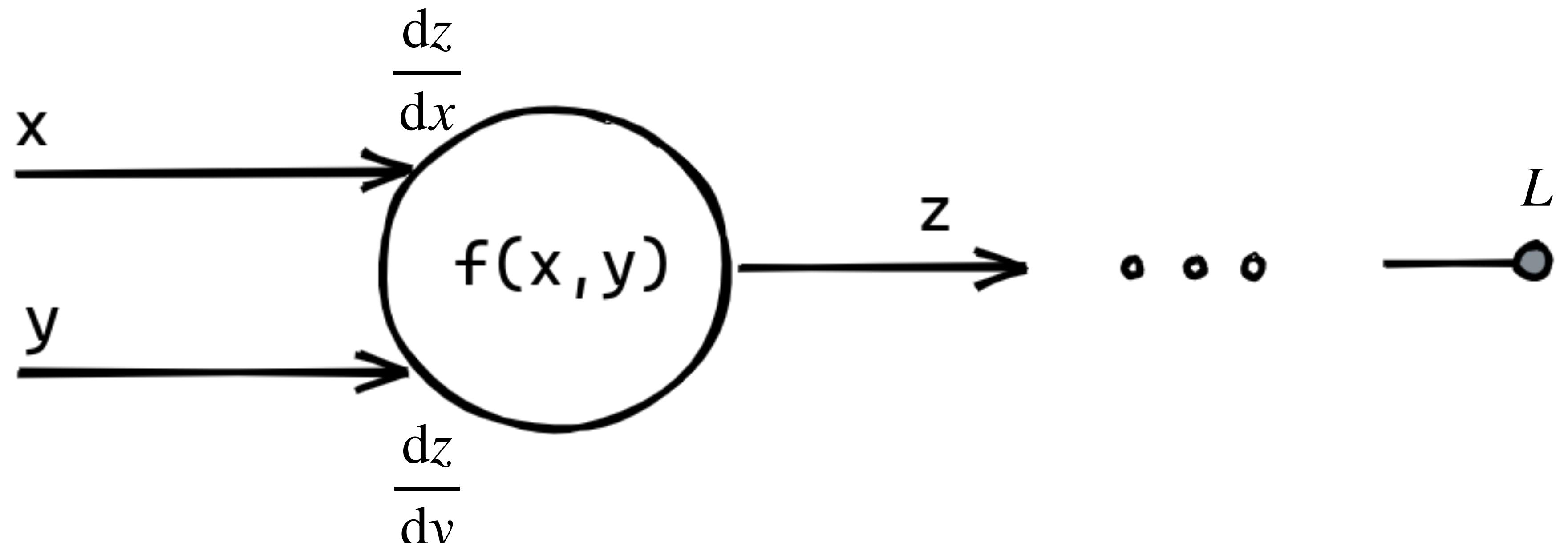
$$z = f(x, y)$$



**Forward pass:** вычисление  $z$ , сохраняем локальные градиенты

# Backpropagation: одна вершина

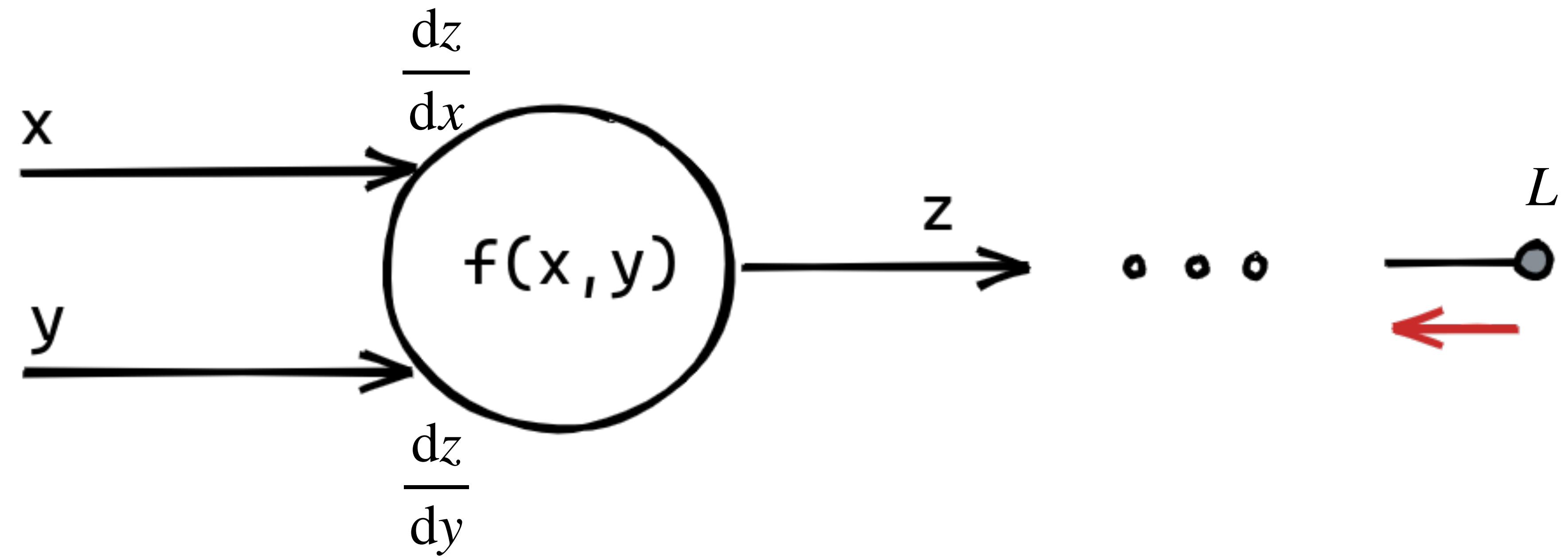
$$z = f(x, y)$$



...посчитался лосс

# Backpropagation: одна вершина

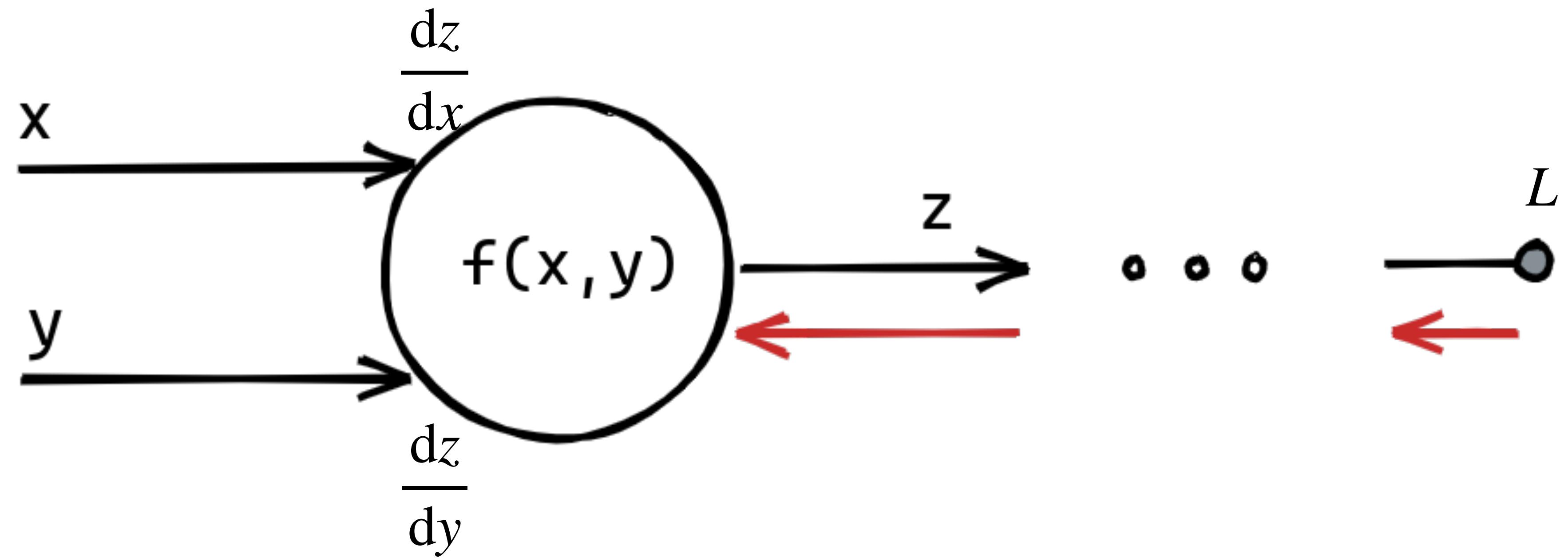
$$z = f(x, y)$$



**Backward pass:** итеративно вычисляем глобальные градиенты

# Backpropagation: одна вершина

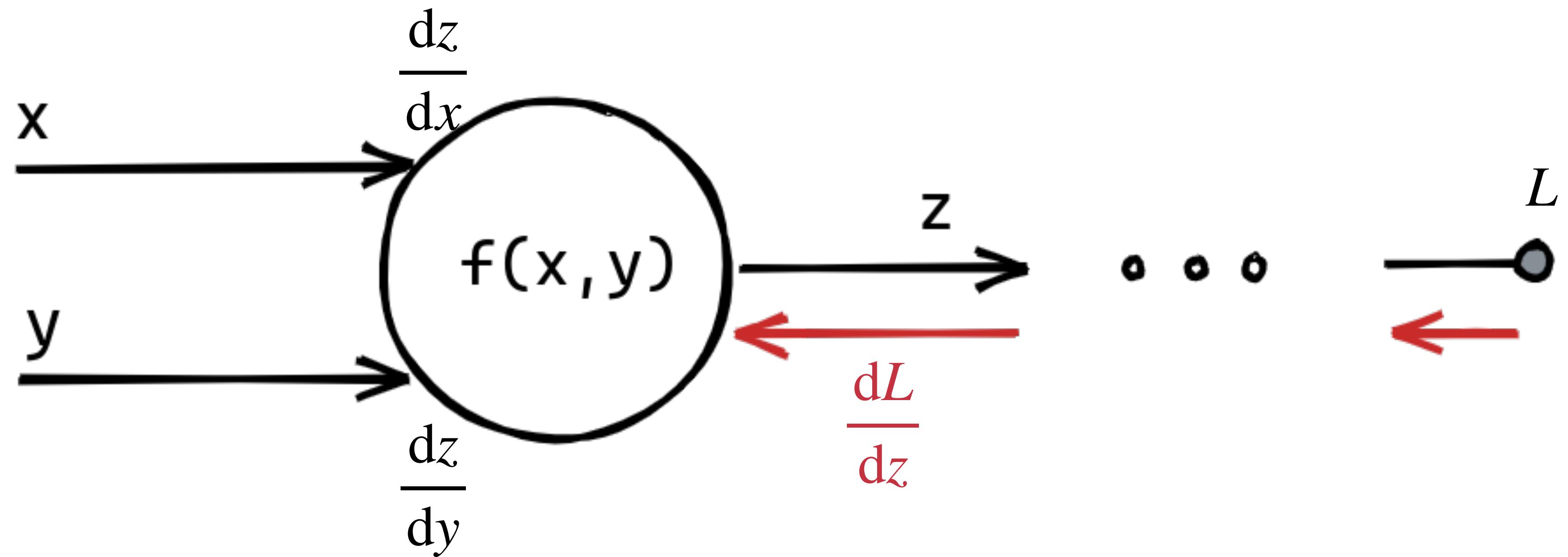
$$z = f(x, y)$$



**Backward pass:** итеративно вычисляем глобальные градиенты

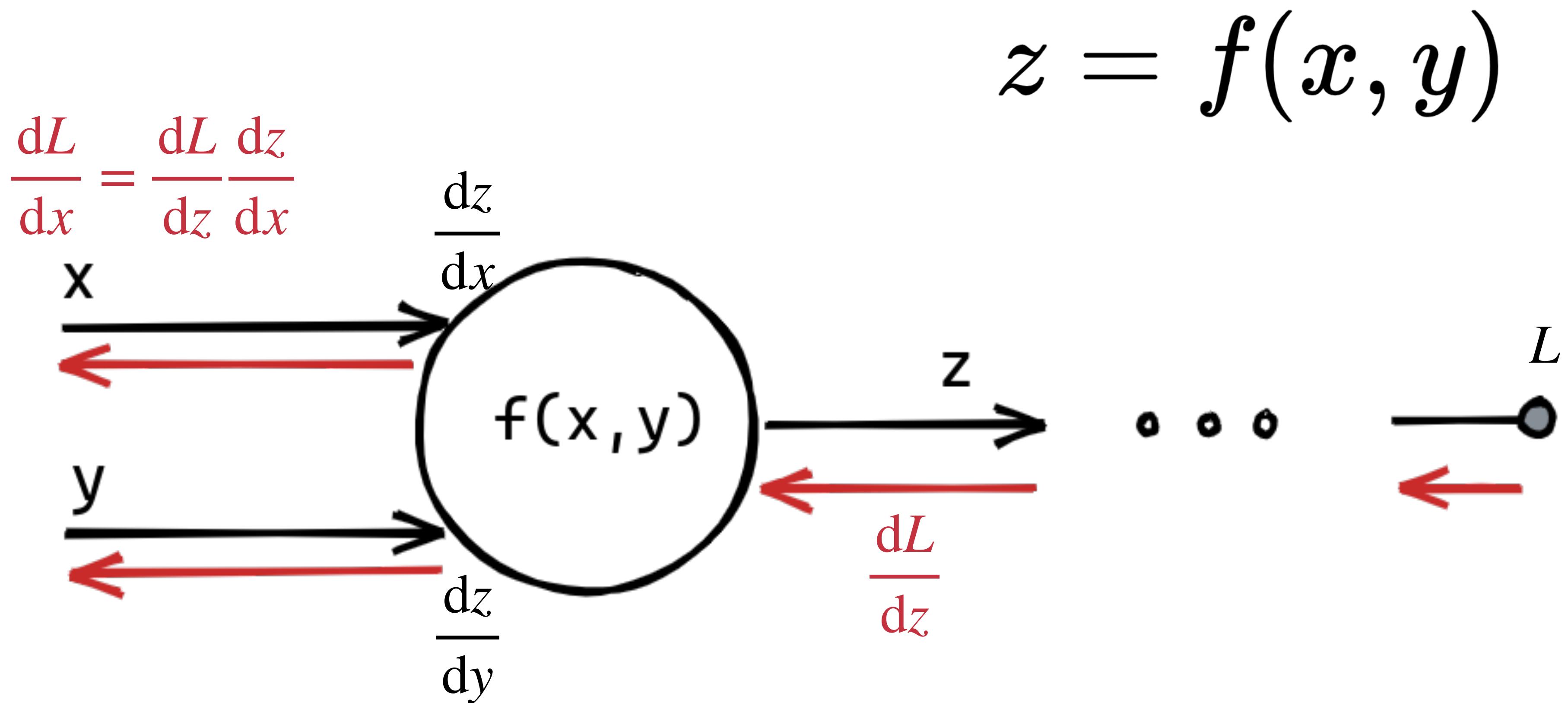
# Backpropagation: одна вершина

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



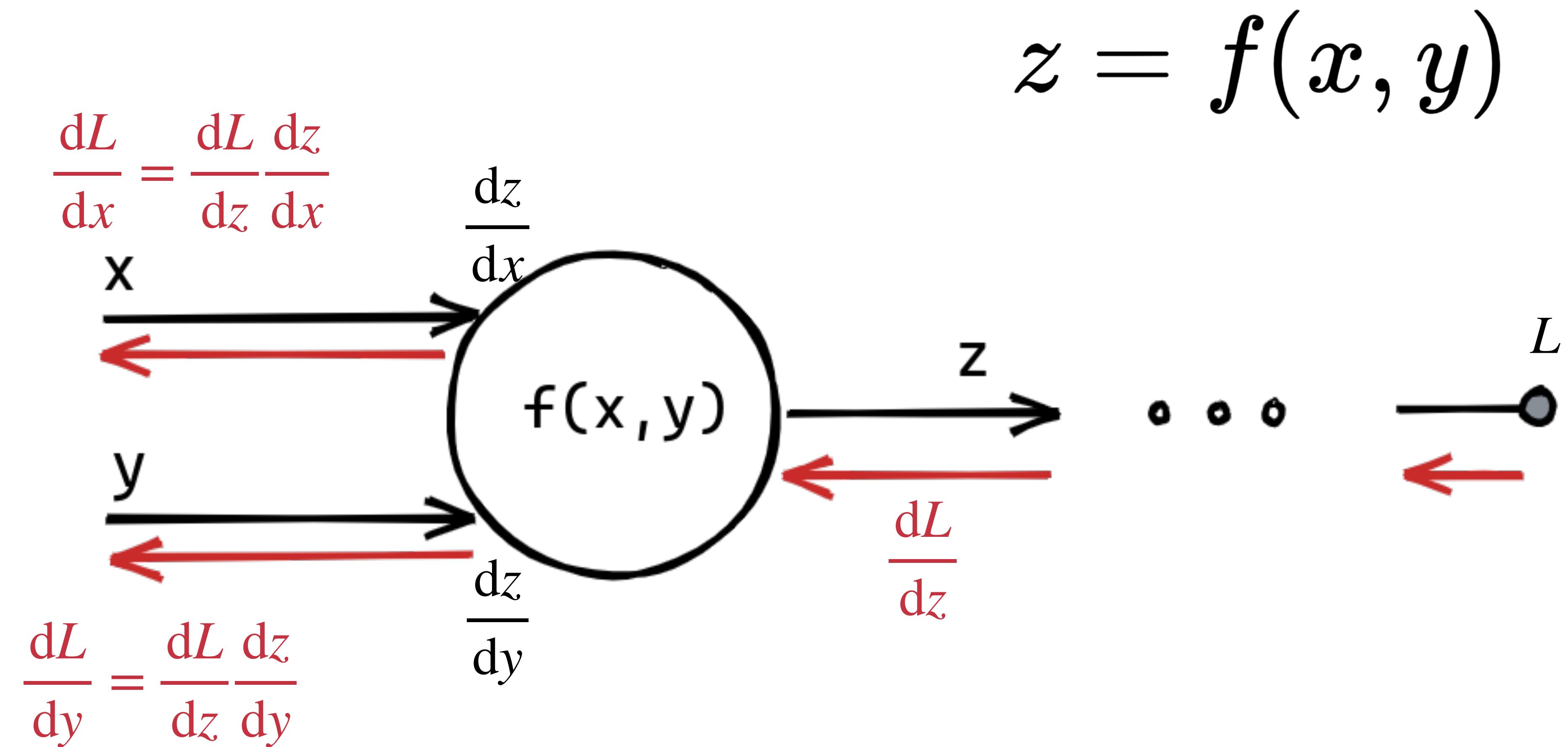
**Backward pass:** итеративно вычисляем глобальные градиенты

# Backpropagation: одна вершина



**Backward pass:** итеративно вычисляем глобальные градиенты

# Backpropagation: одна вершина



**Backward pass:** итеративно вычисляем глобальные градиенты

# Backpropagation: как считать?

Можно оперировать векторной формой

$$x \in \mathbb{R}, y \in \mathbb{R}$$

$$\frac{dy}{dx} \in \mathbb{R}$$

Производная

$$x \in \mathbb{R}^n, y \in \mathbb{R}$$

$$\frac{dy}{dx} \in \mathbb{R}^n$$

$$\left( \frac{dy}{dx} \right)_i = \frac{dy}{dx_i}$$

Градиент

$$x \in \mathbb{R}^n, y \in \mathbb{R}^m$$

$$\frac{dy}{dx} \in \mathbb{R}^{n \times m}$$

$$\left( \frac{dy}{dx} \right)_{i,j} = \frac{dy_j}{dx_i}$$

Якобиан

# Backpropagation: как считать?

Можно оперировать векторной формой

$$x \in \mathbb{R}, y \in \mathbb{R}$$

$$\frac{dy}{dx} \in \mathbb{R}$$

Производная

$$x \in \mathbb{R}^n, y \in \mathbb{R}$$

$$\frac{dy}{dx} \in \mathbb{R}^n$$

$$\left( \frac{dy}{dx} \right)_i = \frac{dy}{dx_i}$$

Градиент

$$x \in \mathbb{R}^n, y \in \mathbb{R}^m$$

$$\frac{dy}{dx} \in \mathbb{R}^{n \times m}$$

$$\left( \frac{dy}{dx} \right)_{i,j} = \frac{dy_j}{dx_i}$$

Якобиан

# DL frameworks

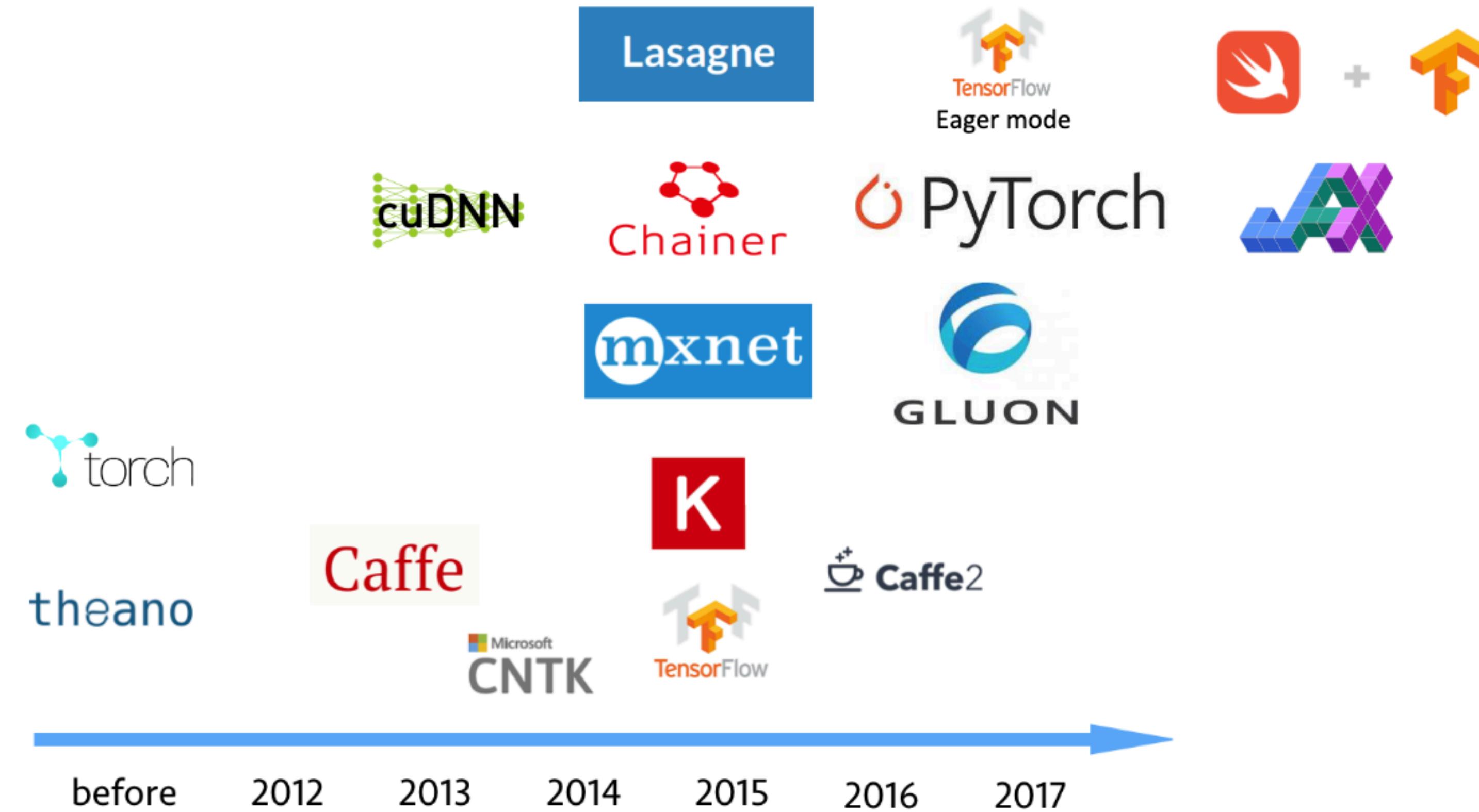


Image credit

# DL frameworks



**Josh Tobin**  
@josh\_tobin\_

Why do people always ask what ML framework to use? It's easy:

- jax is for researchers
- pytorch is for engineers
- tensorflow is for boomers

...



**Josh Tobin** @josh\_tobin\_ · 12 map.  
Keras is for infants

...



**Josh Tobin** @josh\_tobin\_ · 12 map.  
mxnet is for no one

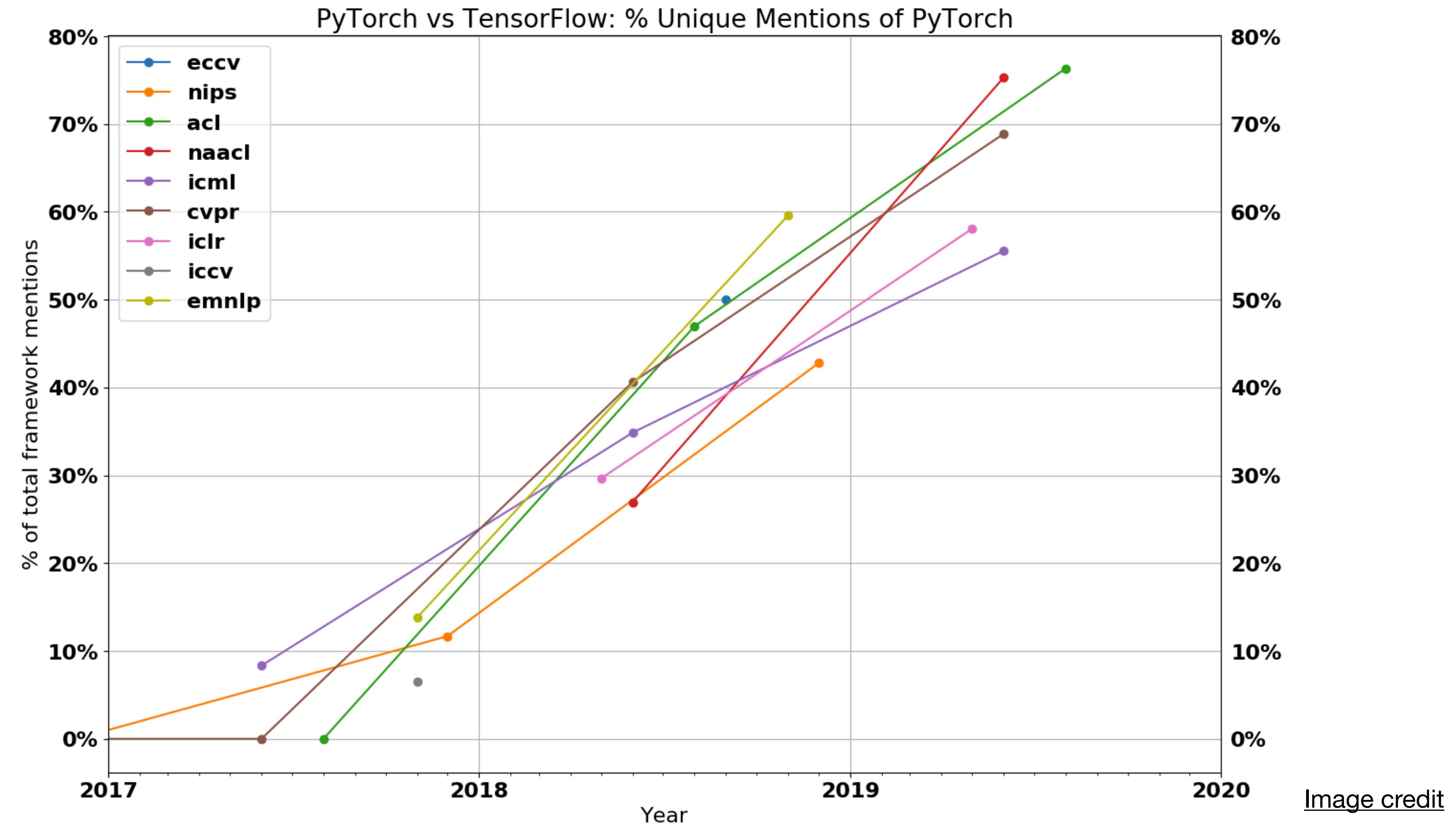
...



**Josh Tobin** @josh\_tobin\_ · 12 map.  
matlab is for professors

...

# DL frameworks



# Итог

- Нейросеть - сложная функция
- Обучение - методы оптимизации 1го порядка
  - Градиентный спуск
- Backpropagation - способ подсчета градиентов в нейросетях
- Linear + Softmax = вероятности классов на выходе нейросети