KARPOV.COURSES >>>
КОНСПЕКТ

# > Конспект > 1 урок > Введение. Полносвязные слои. Функции активации

## > Оглавление



## >Задачи, решаемые с помощью глубинного обучения

Глубинное обучение и нейронные сети могут применяться для работы с информацией практически любого вида - тексты, изображения, видео, музыка,  а также решения разного рода практических задач, например:

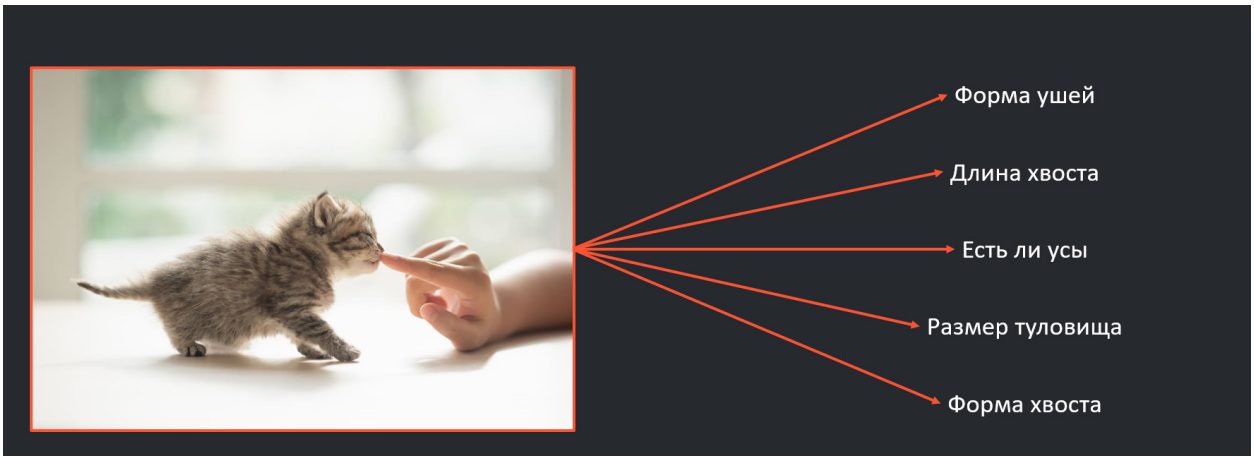
- Классификация человека на изображении (сотрудник/не сотрудник)
- Определение, какие продукты покупатель берет с полок супермаркета

- Классификация изображений на фото и многие другие.

До этого мы решали задачи, в которых можно было выделить конкретные признаки, как-то обработать их и обучать модель. Можем ли мы использовать МЛ подход для вышеперечисленных задач?

Давайте разберем на примере классификации изображений. Будем отличать кошек от собак.

- Выделим некоторые признаки (размер, форма ушей, хвоста, есть ли усы и тд)
- Обучим на этих признаках классификатор



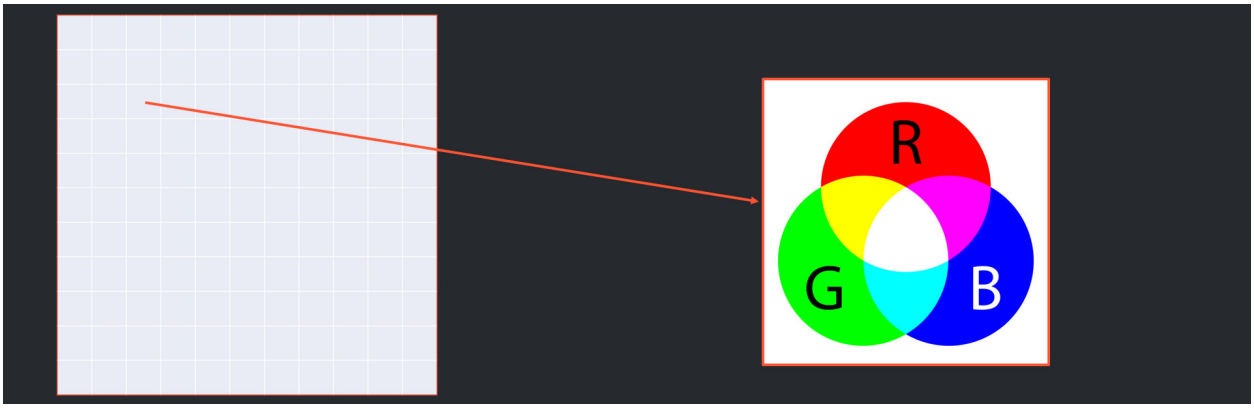


При этом сразу же возникает очевидная трудность - придумывать огромное количество признаков для каждой новой задачи кажется не самым оптимальным решением. Можем ли мы выделять нужные  признаки автоматически?

## >Представление изображений

Разберемся, как представляются изображения в памяти компьютера.

Каждое изображение состоит из отдельных пикселей. Каждый пиксель имеет определенное значение. Если изображение черно-белое, то каждый пиксель содержит информацию об интенсивности белого цвета в диапазоне [0,255], где белому цвету соответствует максимум.

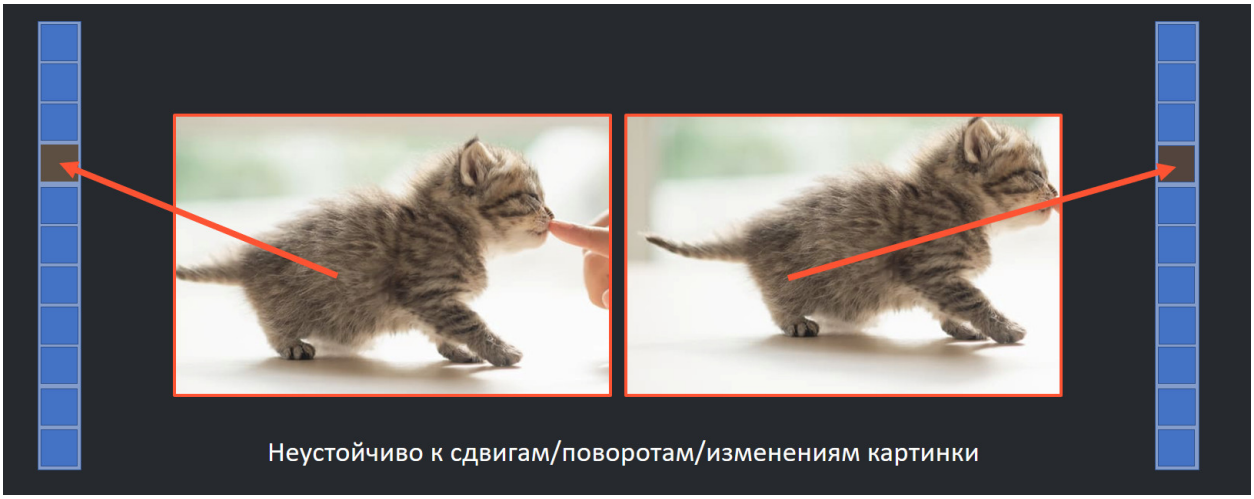




Цветное изображение кодируется сходным образом, только канал не один, как в случае с ч/б, а сразу три - красный, зеленый, синий (RGB) - любой цвет можно представить как комбинацию этих трех цветов.

## >Пиксели как признаки

Вернемся к нашему изображению и  будем работать с пикселями как с признаками! Для этого нужно привести изображение к некоторому стандартному размеру, а затем вытянуть в вектор для предсказания.

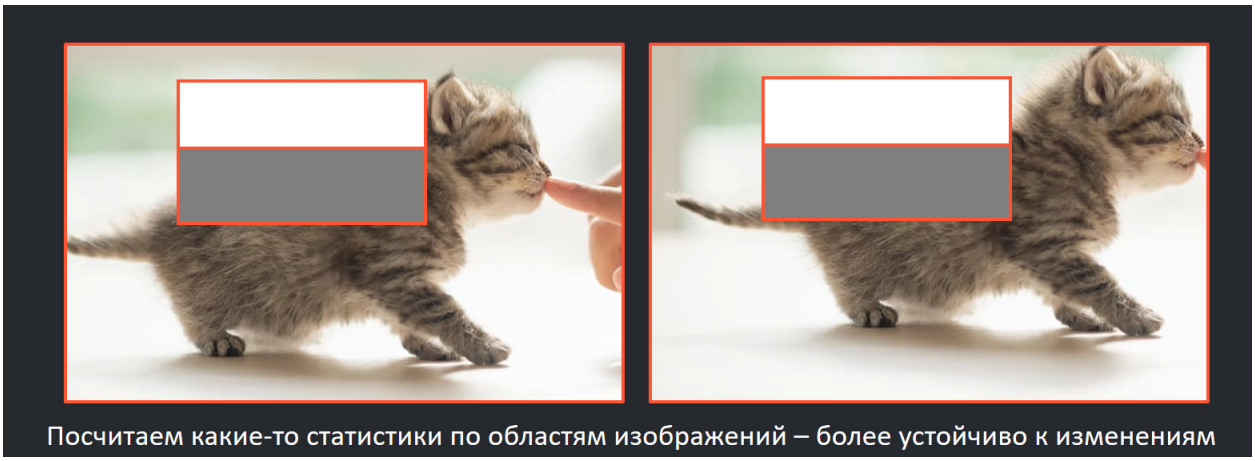
Однако тут снова возникает проблема - предположим у нас две идентичные фотографии и на одной из них есть небольшой сдвиг:




Тогда все значения пикселей поменяются и обученный классификатор уже не сможет распознать нашего кота.

Таким образом данный подход абсолютно неустойчив к любым изменениям картинки (сдвиг, изменение контраста и т.п.)

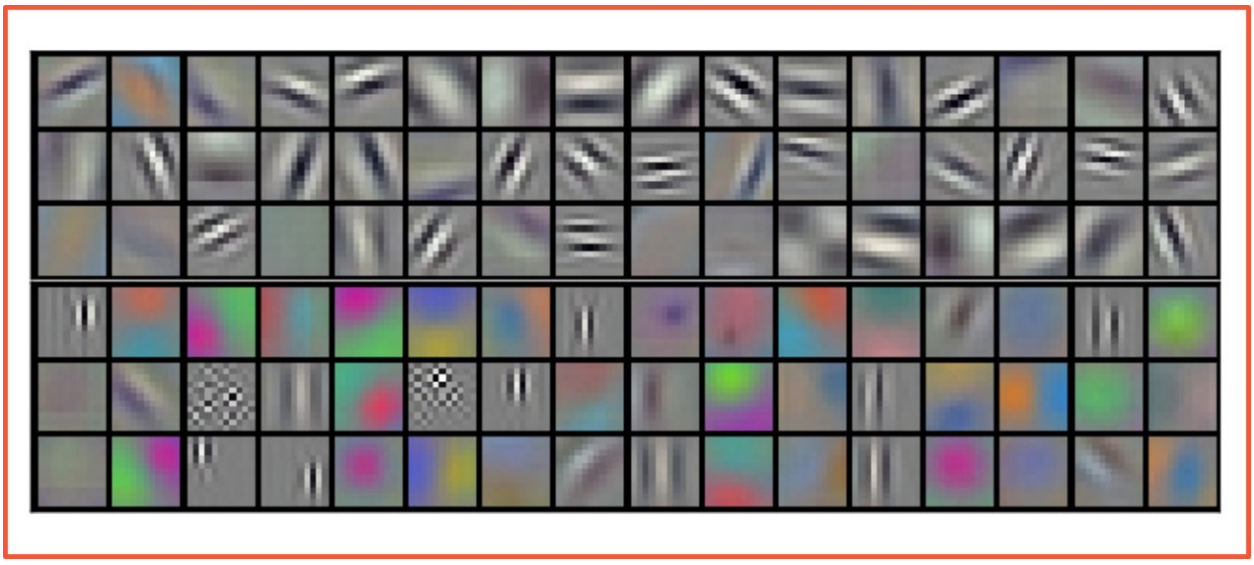
Еще одна идея - объединим пиксели в группы и посчитаем по ним некоторые статистики, например усредним их значения:



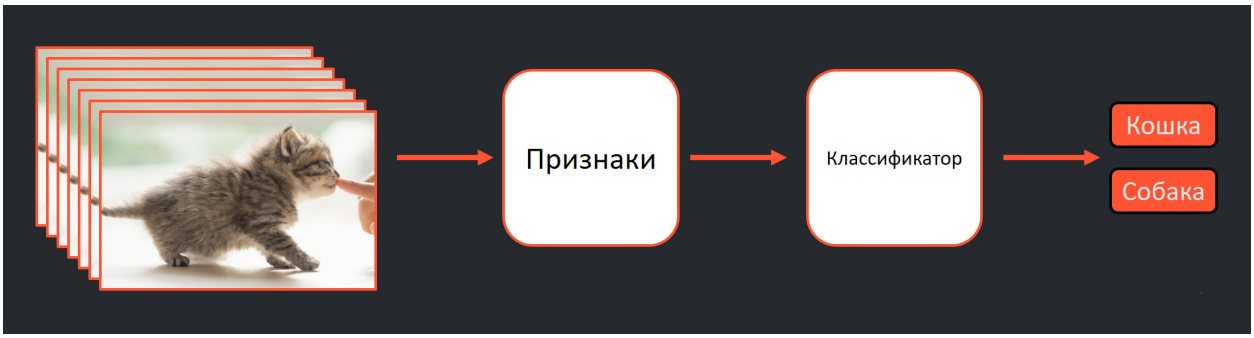

В данном случае снова возникает узкое место - не понятно, сколько и какие области выбирать, какого размера и в каком месте делать области?

## >Глубинное обучение в задачах классификации изображений

Идея глубинного обучения - сделать выделение признаков из данных обучаемым. Выделяются мелкие паттерны/шаблоны, хотим найти такие, которые минимизируют функцию ошибки.

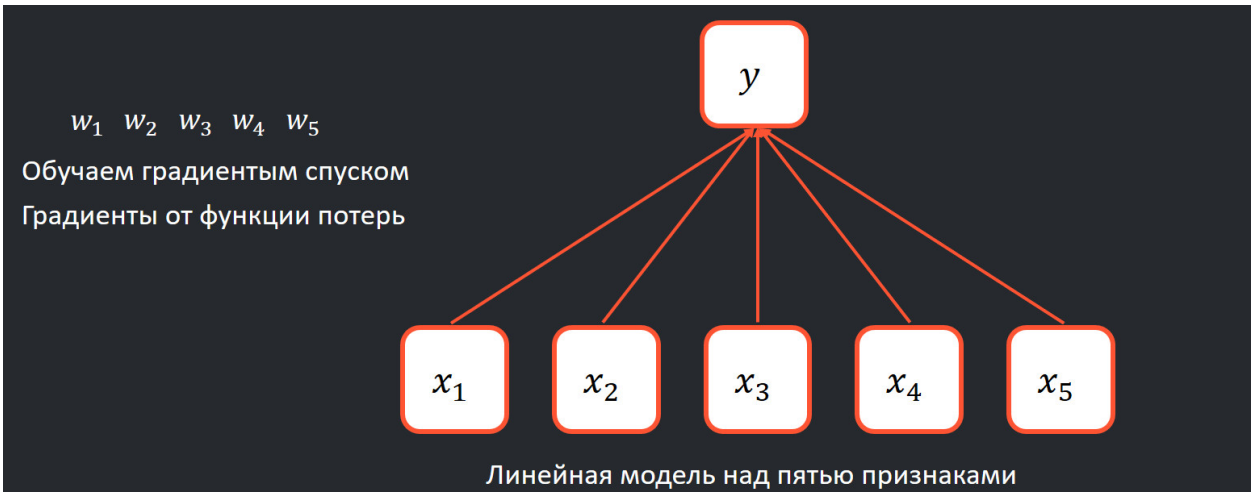


На основе данных паттернов строится классификатор. Получаем end-to end модель, где на вход подаются сырые данные, из них автоматически выделяются признаки, подаются в классификатор, на выходе получаем готовый ответ.

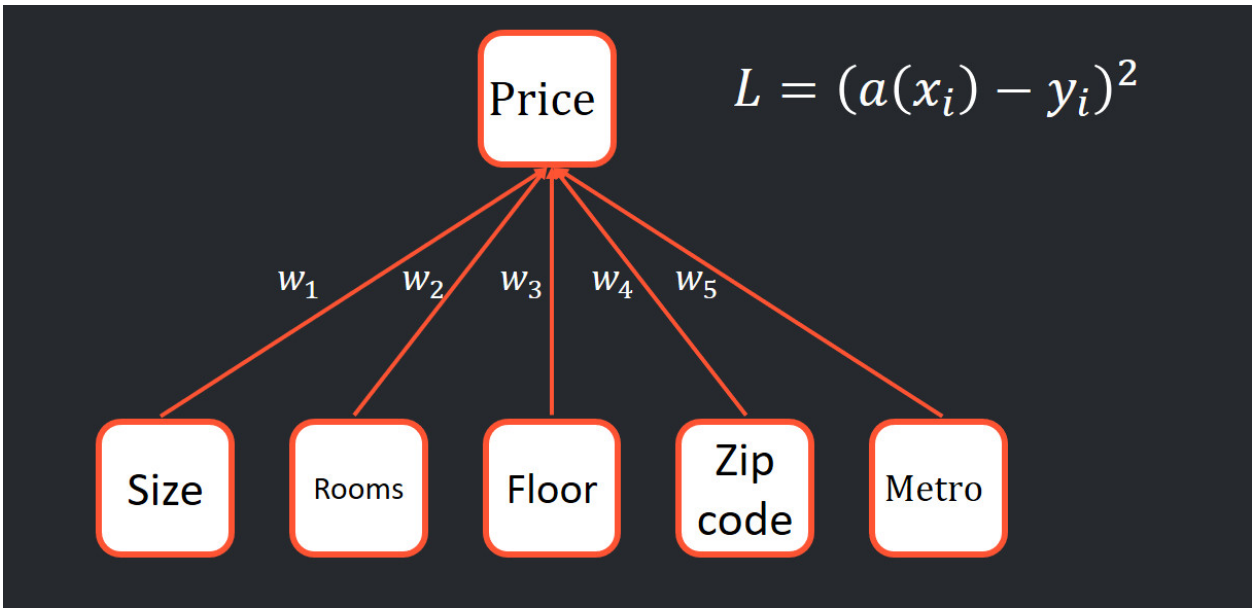



# >Линейные модели в глубинном обучении

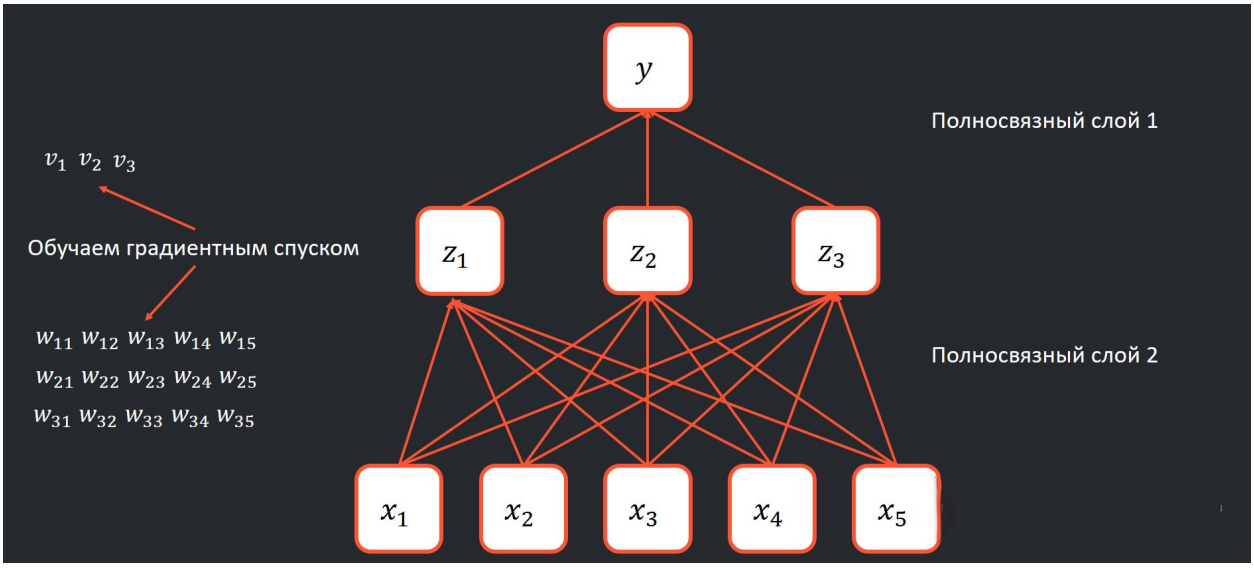
Вернемся к уже знакомой с предыдущего блока теме линейных моделей. Рассмотрим модель с пятью признаками:




Допустим решаем задачу предсказания стоимости квартиры:



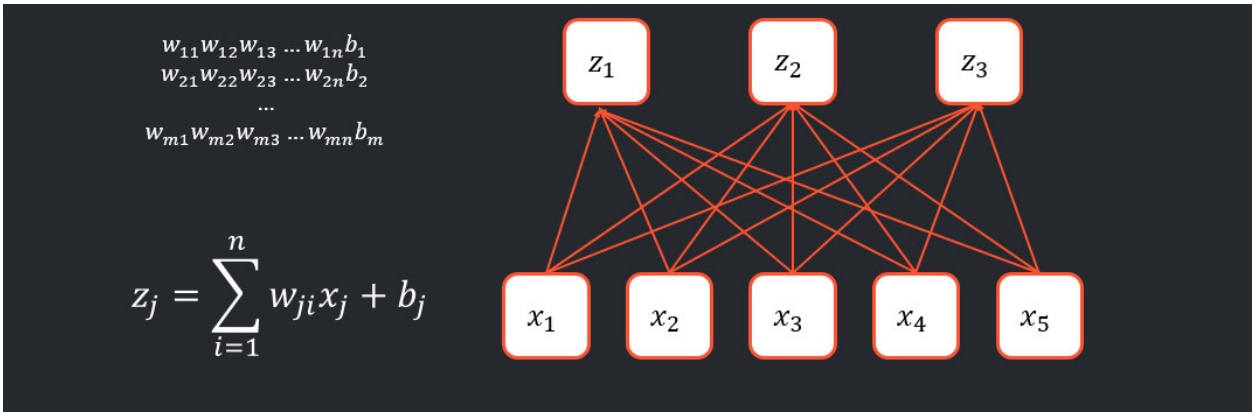

Представим, что хотим обучить на этих пяти признаках несколько линейных моделей - $z_1, z_2, z_3$, поверх этих трех моделей обучим еще одну линейную модель у с весами $v_1, v_2, v_3$ соответственно:




Мы получили два полносвязных слоя!

# >Полносвязный слой

Полносвязный слой (Fully connected) — слой, в котором каждый нейрон соединен со всеми нейронами на предыдущем уровне.




Один полносвязный слой

- на входе n чисел, на выходе m чисел
- $x_1, x_2 \dots x_n$ — входы
- $z_1, z_2, \dots z_m$ — выходы
- m линейных моделей, в каждой $(n + 1)$ параметров ($n$ весов + 1 свободный коэффициент)
- каждый выход — применение линейной модели над входами

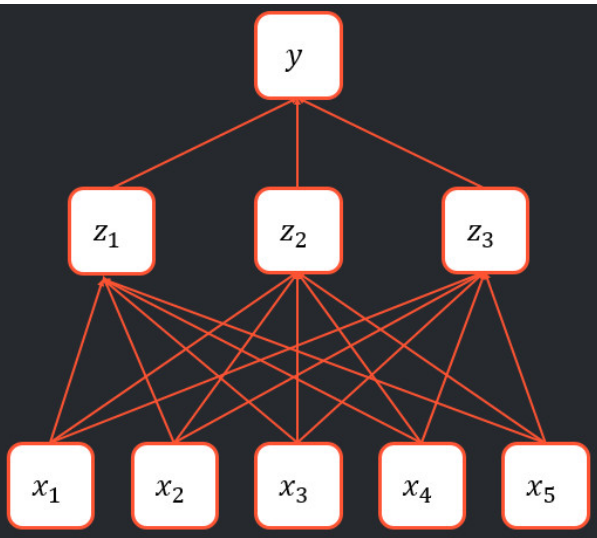
# >Функция активации

Распишем уравнение для получения y, подставив $x$ вместо $z$:

$$y = v_1 z_1 + v_2 z_2 + v_3 z_3$$

$$z_1 = w_{11} x_1 + w_{12} x_2 + w_{13} x_3 + w_{14} x_4 + w_{15} x_5$$

$$y = v_1 z_1 + \cdots = \beta_1 (w_{11} x_1 + w_{12} x_2 + w_{13} x_3 + w_{14} x_4 + w_{15} x_5) + \cdots = c_1 x_1 + c_2 x_2 + c_3 x_3 + \ \ldots$$




Два полносвязных слоя

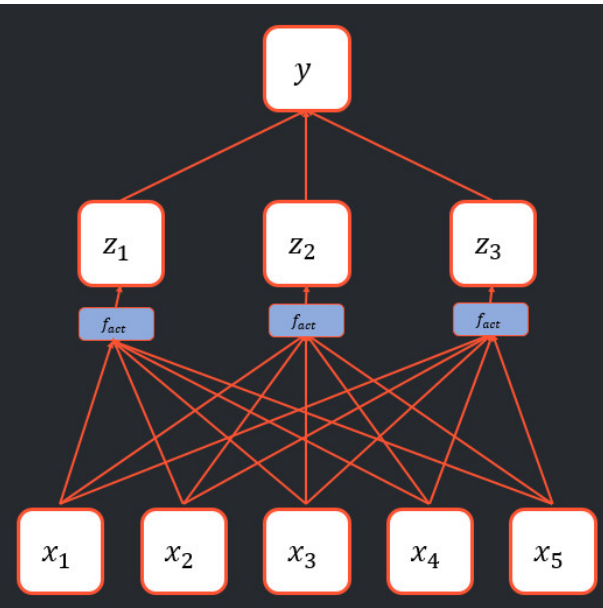
Упростив данное выражение можно заметить, что все наши полносвязные слои преобразовались в одну линейную модель  и  получается, что смысла в добавлении промежуточных слоев нет.  Такая ситуация справедлива для сколь угодно большого количества слоев и называется вырождением слоя.

Решить ее можно добавлением между полносвязными слоями нелинейной функции - функцию активации.

$$y = v_1 z_1 + v_2 z_2 + v_3 z_3$$

$$z_1 = f_{\text{act}}(w_{11} x_1 + w_{12} x_2 + w_{13} x_3 + w_{14} x_4 + w_{15} x_5)$$

$$y = v_1 z_1 + \cdots = \beta_1 (f_{\text{act}}(w_{11} x_1 + w_{12} x_2 + w_{13} x_3 + w_{14} x_4 + w_{15} x_5)) + \cdots$$



Два полносвязных слоя + функция активации

Рассмотрим более подробно на примере:

$$\begin{aligned} y &= 5z_1 + 3z_2 \\ z_1 &= -3x_1 + 2x_2 \\ z_2 &= 4x_1 + 2x_2 \end{aligned} \longrightarrow \begin{aligned} y &= -15x_1 + 10x_2 + 12x_1 + 6x_2 \\ &= -3x_1 + 16x_2 \end{aligned}$$

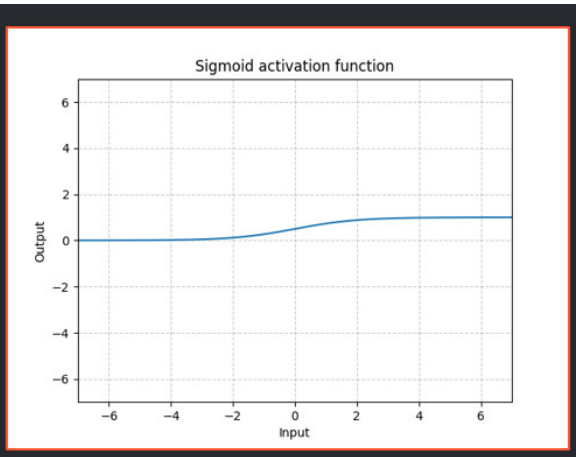
Добавим нелинейность в виде функции синуса:

$$\begin{aligned} y &= 5z_1 + 3z_2 \\ z_1 &= \sin(-3x_1 + 2x_2) \\ z_2 &= \sin(4x_1 + 2x_2) \end{aligned} \longrightarrow y = \sin(-3x_1 + 2x_2) + \sin(4x_1 + 2x_2)$$

Функция активации — некоторая нелинейная функция, которая определяет выходное значение нейрона, применяется поэлементно.

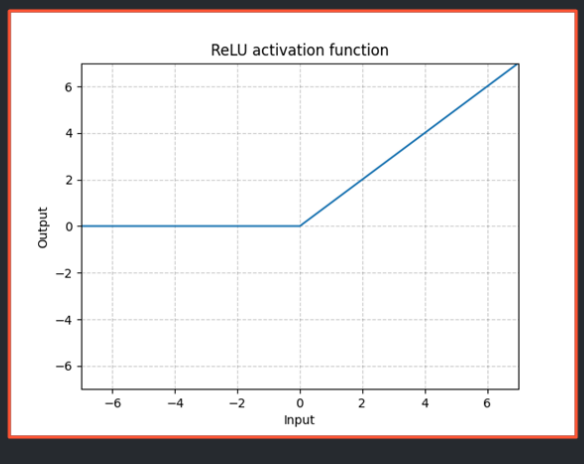
Наиболее известные функции активации — сигмоида и ReLU:

Сигмоида $f(x) = \frac{1}{1 + e^{-x}}$




https://pytorch.org/docs/stable/generated/torch.nn.Sigmoid.html Select an Image

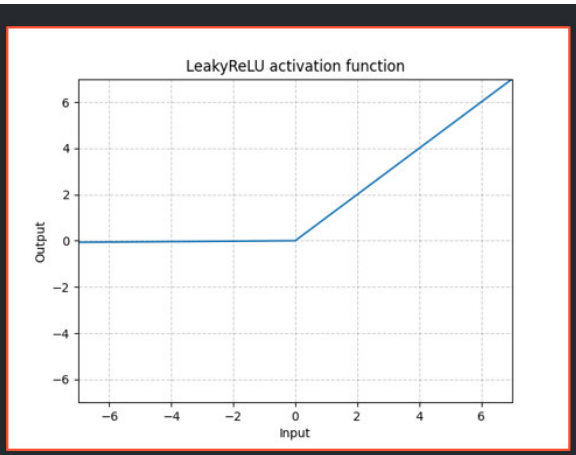
ReLU $y = \max(0, x)$




https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html Select an Image

Leaky ReLU

$y = \max(0, x) + \alpha \min(0, x)$

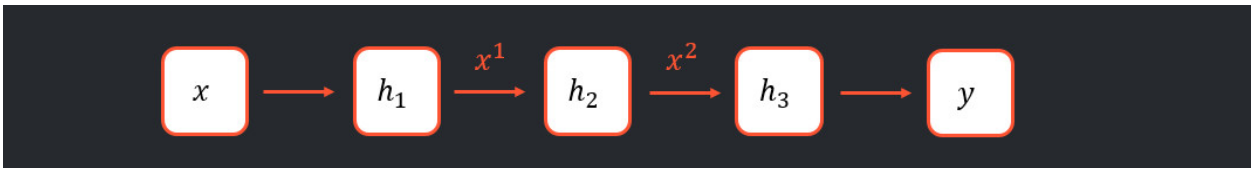



https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html Select an Image

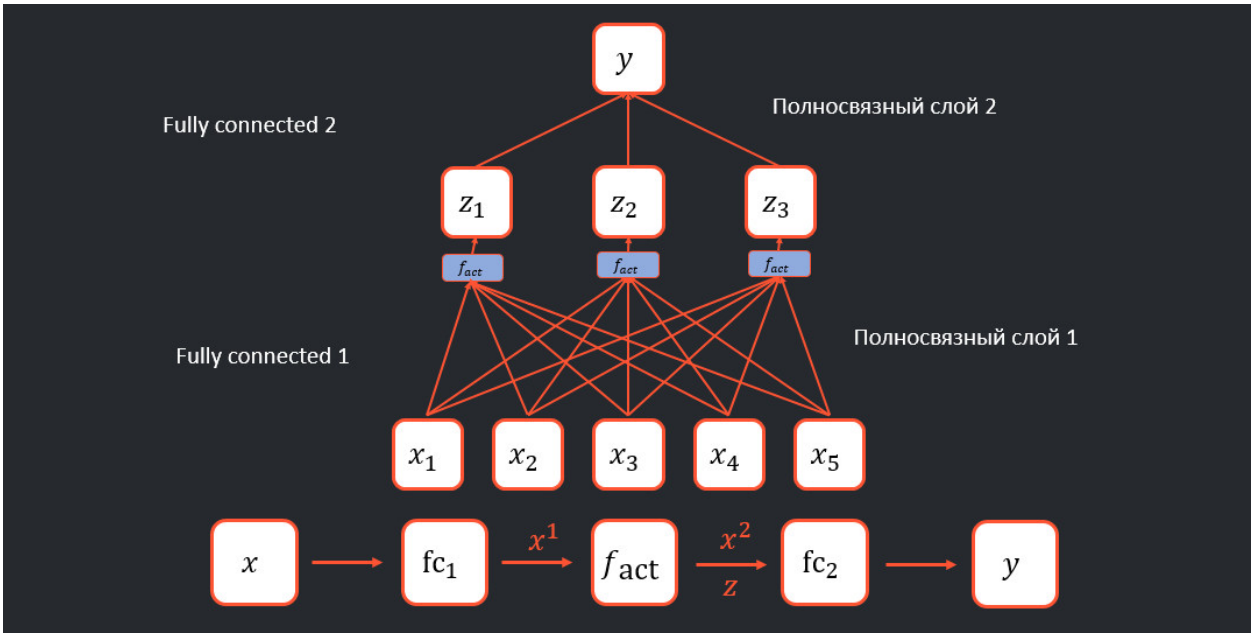
## >Граф вычислений

В общем виде нейросети представляют собой граф вычислений (computational graph):

- есть некие признаки объекта $(x)$, которые подаются на вход
- h_1 - преобразование внутри нейросети (слой или функция активации)
- $x^1, x^2 ... x^n$ - результаты после применения $1, 2, ...n$ слоёв соответственно

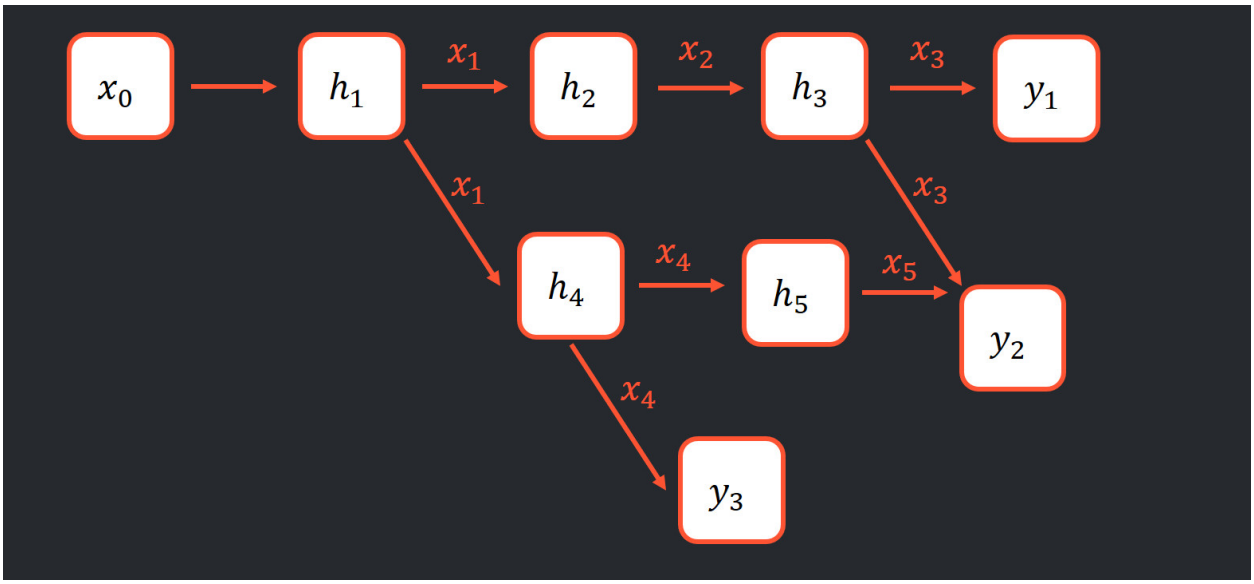



Рассмотрим как выглядит граф предыдущего примера нейронной сети с двумя полносвязными слоями:



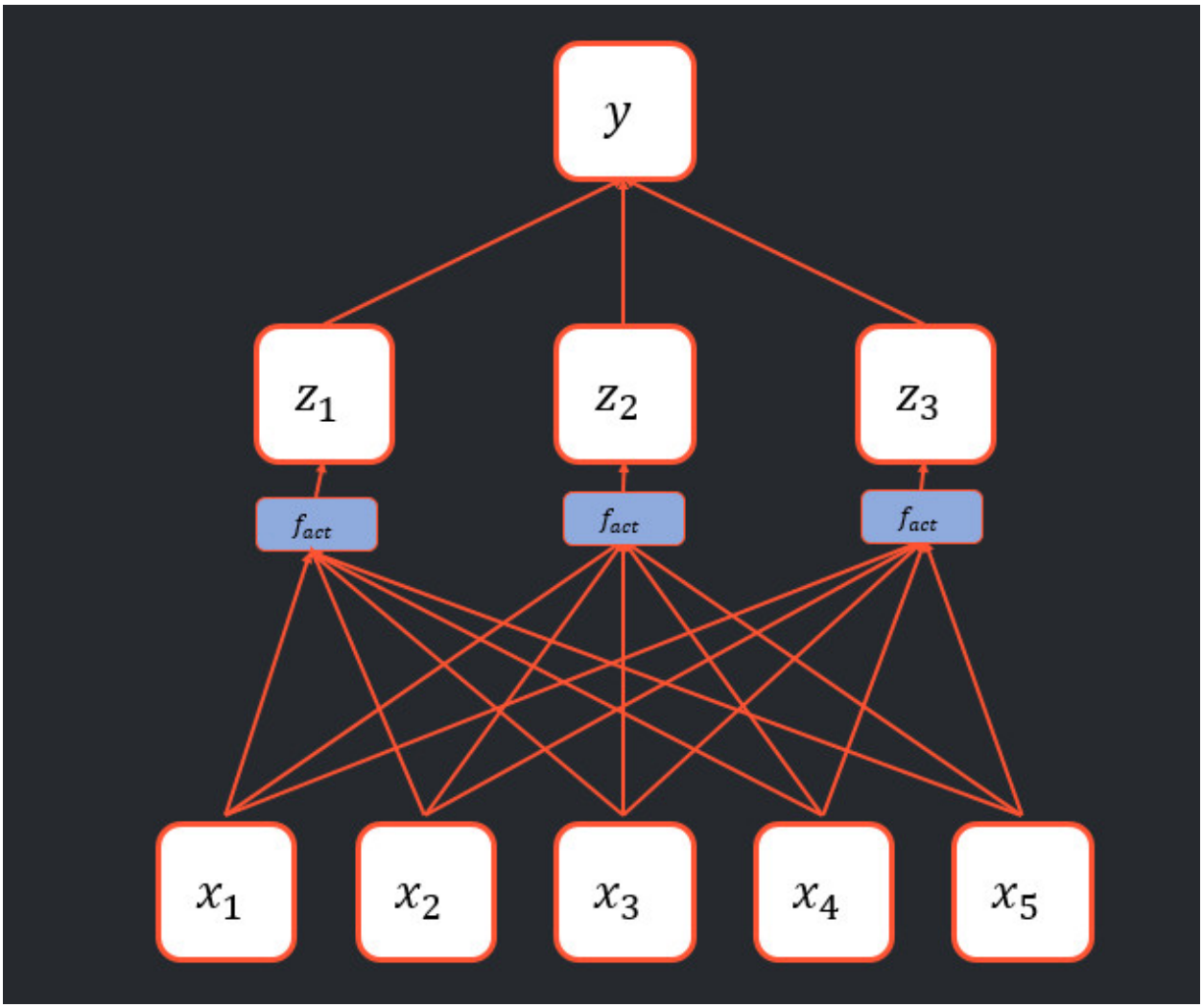

Граф вычислений со множественным входом и одним выходом

Граф вычислений может иметь от 1 до $n$ входов и выходов.




Граф вычислений с одним входом и множественным выходом

## >Простейшая архитектура нейронной сети




Архитектура нейронной сети — определяет конкретные используемые слои и их порядок (в примере выше два полносвязных и нелинейность между ними). Конкретная нейронная сеть задается с помощью архитектуры и размеров слоев.

Согласно теореме Цыбенко, с помощью такой нейронной сети можно аппроксимировать любую функцию, надо лишь задать желаемую точность и множество, на котором это нужно сделать. Однако, чем больше размер множества и чем точнее мы хотим приближать функцию, тем больше будет размер нейронной сети.

Глубинное обучение (Deep Learning) — это вид машинного обучения, включающий алгоритмы, обрабатывающие входные данные путём последовательного

применения к ним большого количества линейных и нелинейных преобразований (слоев).

- Результат одного слоя подается на вход последующему;
- Главное условие к слоям — они должны быть дифференцируемы;
- Обучение осуществляется методом градиентного спуска для параметров каждого слоя;
- Градиенты вычисляются методом «обратного распространения ошибки», мы подробнее рассмотрим эту темы во втором занятии.