

Лекция 2. Нейронные сети. Архитектуры, активации

Денис Деркач, Дмитрий Тарасов

Слайды от А. Маевского, М. Гущина, А. Кленицкого, М Борисяка

23 января 2026 года

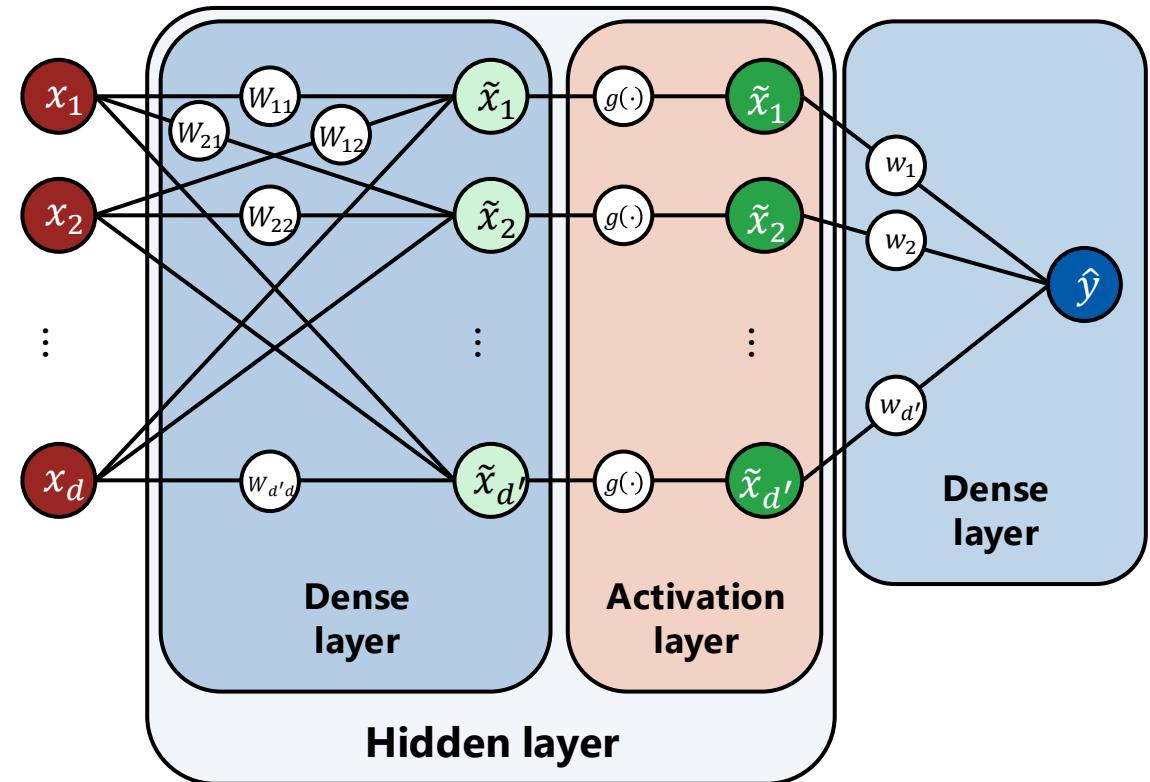


Универсальный аппроксиматор



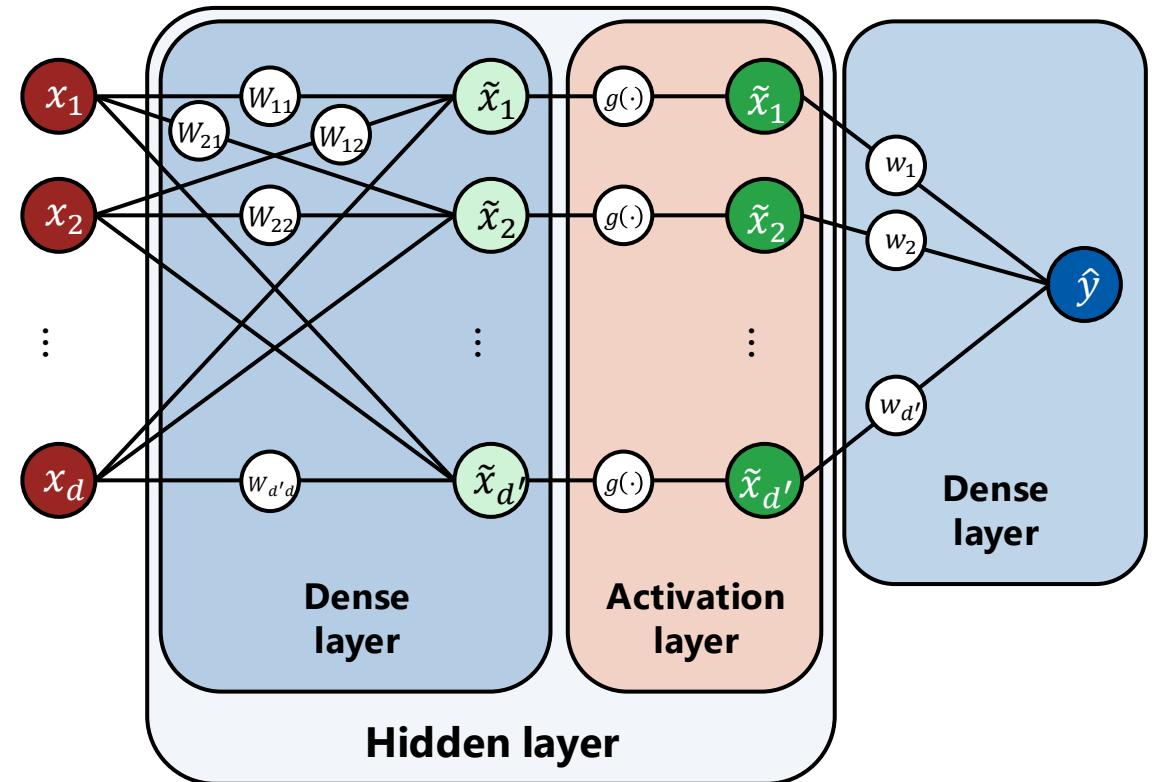
Universal approximator

- ▶ Всего один скрытый слой с нелинейностью делает эту модель универсальным аппроксиматором



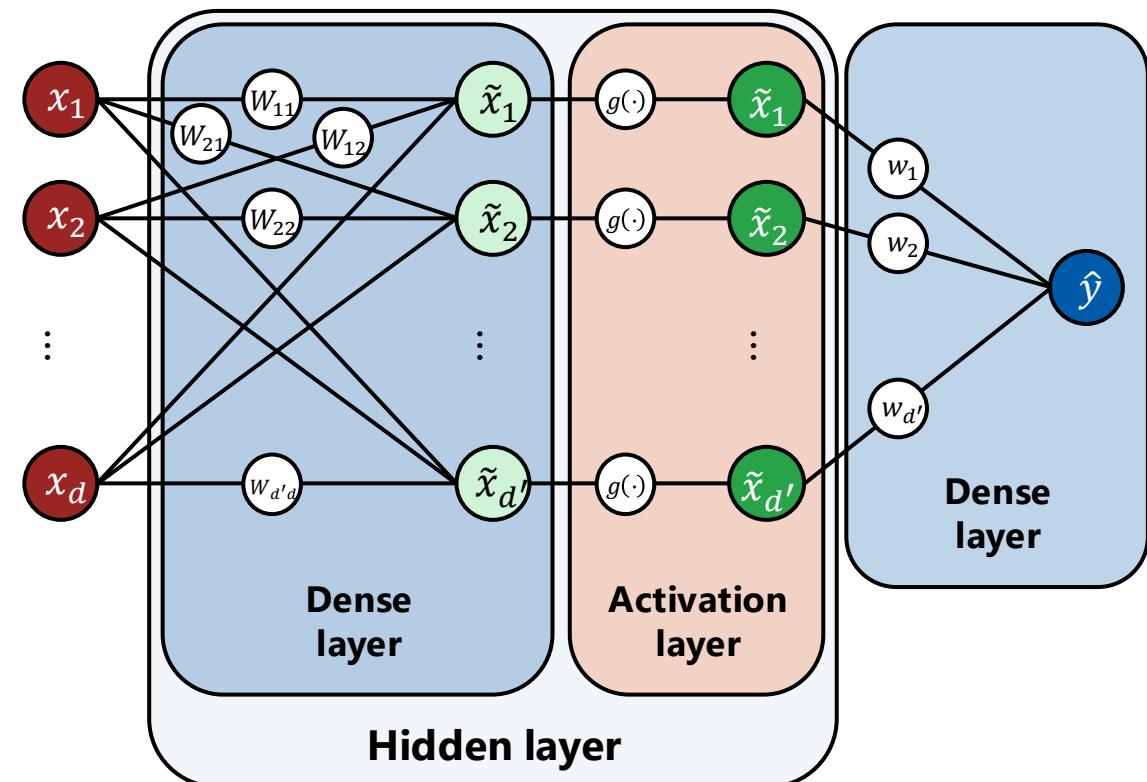
Universal approximator

- ▶ Всего один скрытый слой с нелинейностью делает эту модель универсальным аппроксиматором
 - любая функция может быть аппроксимирована с произвольной точностью при достаточно широком скрытом слое (достаточно большом d')



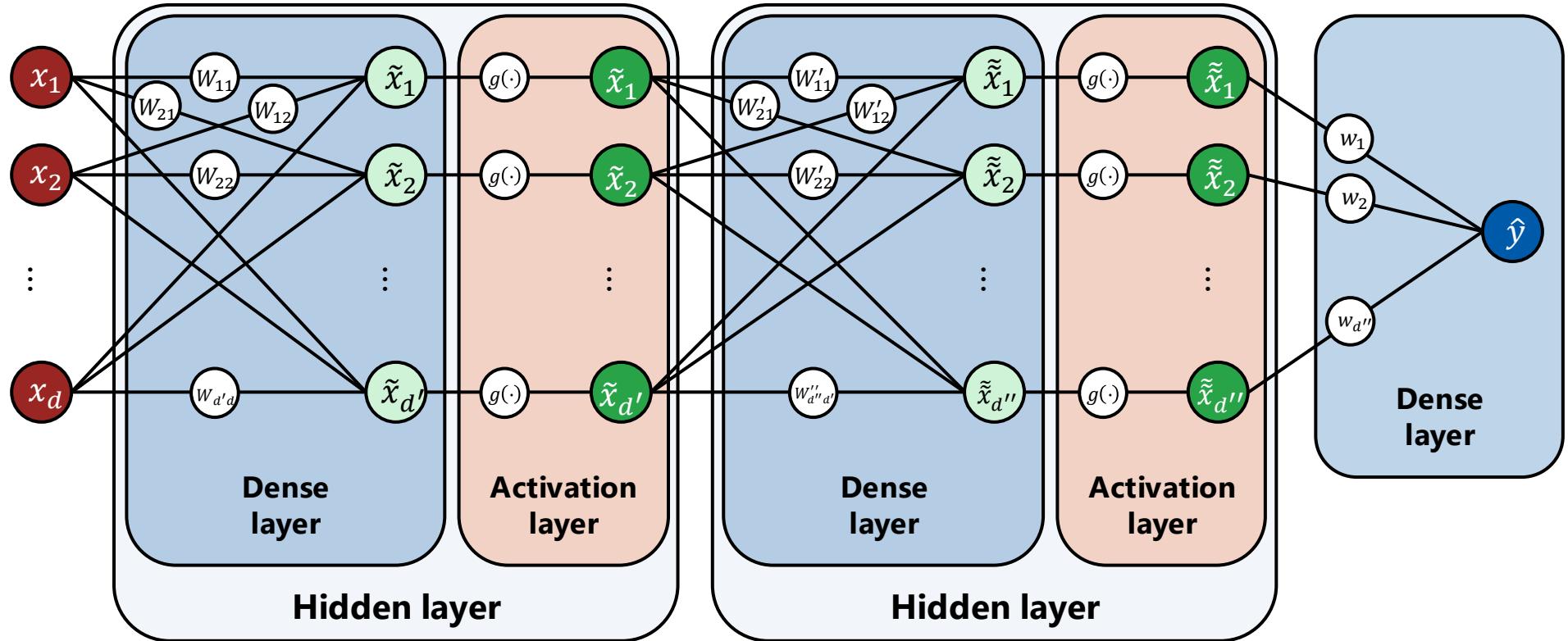
Universal approximator

- ▶ Всего один скрытый слой с нелинейностью делает эту модель универсальным аппроксиматором
 - любая функция может быть аппроксимирована с произвольной точностью при достаточно широком скрытом слое (достаточно большом d')
 - Примечание: на практике мы можем не найти такого приближения
 - например, из-за сильно невыпуклой функции потерь, невыполнимо большого d' , чрезмерной подгонки



Глубокие сети

'multilayer perceptron'



- ▶ На практике стэкинг большего числа скрытых слоев часто уменьшает количество нейронов, необходимых для представления заданной функции

Теорема о бесплатных обедах No free lunch (немного философии)



Тест IQ

Если следовать схеме, показанной в последовательности чисел ниже, какое число отсутствует?

1, 8, 27, ?, 125, 216

- ▶ 36;
- ▶ 45;
- ▶ 46;
- ▶ 64;
- ▶ 99.

Тест IQ (ML edition)

Если следовать схеме, показанной в последовательности чисел ниже, какое число отсутствует?

X_{train}	1	2	3	5	6
y_{train}	1	8	27	125	216

$$X_{\text{test}} = (4,)$$

Тест IQ (ML edition): решений

Предлагаю решение:

$$f(x) = \frac{1}{12}(91x^5 - 1519x^4 + 9449x^3 - 26705x^2 + 33588x - 14940);$$

$$f(1) = 1;$$

$$f(2) = 8;$$

$$f(3) = 27;$$

$$f(4) = 99;$$

$$f(5) = 125;$$

$$f(6) = 216.$$

Определения

A supervised learning algorithm A :

- ▶ is defined on sample set \mathcal{X} and target set \mathcal{Y} ;
- ▶ receives training dataset $D_{\text{train}} = (X, y) \subseteq \mathcal{X} \times \mathcal{Y}$;
- ▶ does ~~magic~~ learning;
- ▶ returns prediction function f :

$$f = A(D_{\text{train}});$$

$$f: \mathcal{X} \mapsto \mathcal{Y}.$$

By this definition, decision trees are a family of algorithms.

Теорема

All supervised learning algorithms are equally **useless**:

- ▶ for all binary datasets D_{train} of size n with m features:
 - $\mathcal{X} = \{0, 1\}^m, \mathcal{Y} = \{0, 1\}$;
- ▶ averaged by all test datasets D_{test} : $X_{\text{test}} = \mathcal{X} \setminus X_{\text{train}}$;

$$\text{gP}(A, D_{\text{train}}) = \frac{1}{|D_{\text{test}}|} \sum_{x, y \in D_{\text{test}}} \mathbb{I}[A(D_{\text{train}})(x) = y] - \frac{1}{2};$$

$$\boxed{\sum_{D_{\text{train}}} \text{gP}(A, D_{\text{train}}) = 0;}$$

Some algorithms are better at memorization.

Шокирующее следствие

Тесты IQ измеряют вашу способность думать так же, как их авторы... возможно, с поправкой на вашу скорость.

Ещё следствие

Чтобы решить IQ-тест, нужно думать, как автор теста.

Совсем шокирующее следствие

Чтобы решить ML задачу, нужно мыслить как... природа.

Критика теоремы

- Не все задачи одинаково вероятны:
 - непрерывность;
 - человеческая предвзятость: предварительный отбор признаков;
 - предварительное знание рассматриваемой проблемы;
- запоминание также важно:
 - особенно в сочетании с непрерывностью.

Утверждение остаётся в любом случае:

Чтобы улучшить результаты в решении одного класса задач, нужно пожертвовать эффективностью в решении других.

Идеальный алгоритм/семейство

Для одной задачи:

- максимизирует производительность на задачах, соответствующих предыдущим знаниям;
- жертвует производительностью в других местах.

Для класса задач:

- максимизирует производительность на классе проблем;
- легко модифицировать/ограничить.

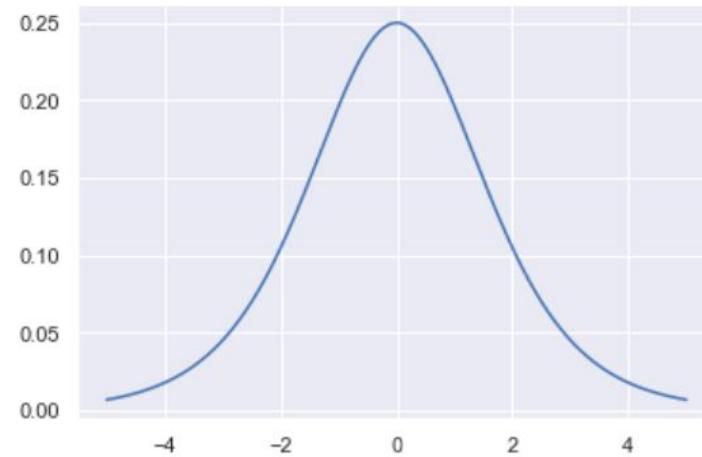
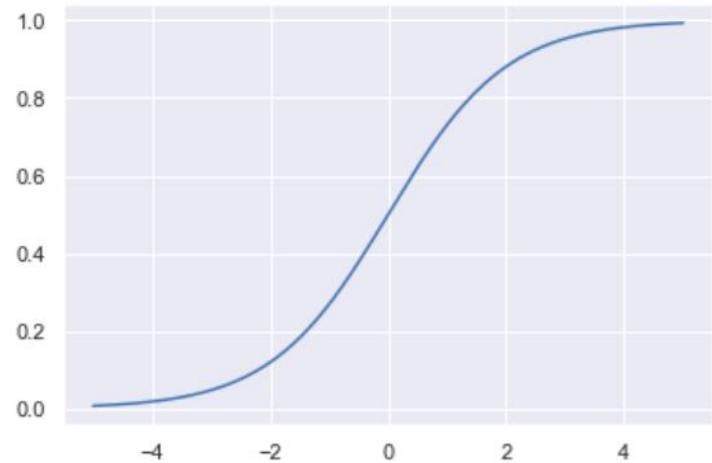
Функции активации



Сигмоида

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

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$



- Насыщение на краях, градиент близок к нулю
- Максимальное значение производной 0,25
- Выход не центрирован вокруг нуля

Сигмоида - использование

По-прежнему используется в отдельных слоях, например:

- На выходе модели для бинарной классификации
- В гейтах (например, LSTM/GRU)

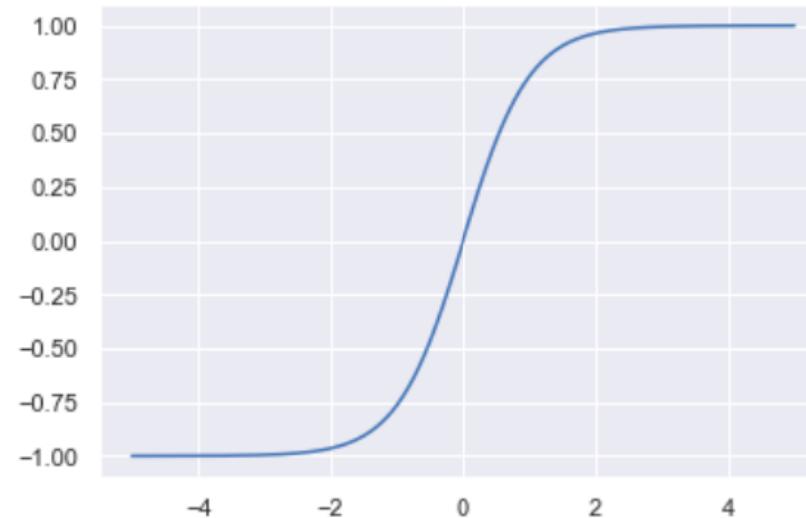
$$h_t = \sigma * h_{t-1} + (1 - \sigma) * h'_t$$

Гиперболический тангенс

Гиперболический тангенс

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\tanh(x) = 2\sigma(2x) - 1$$

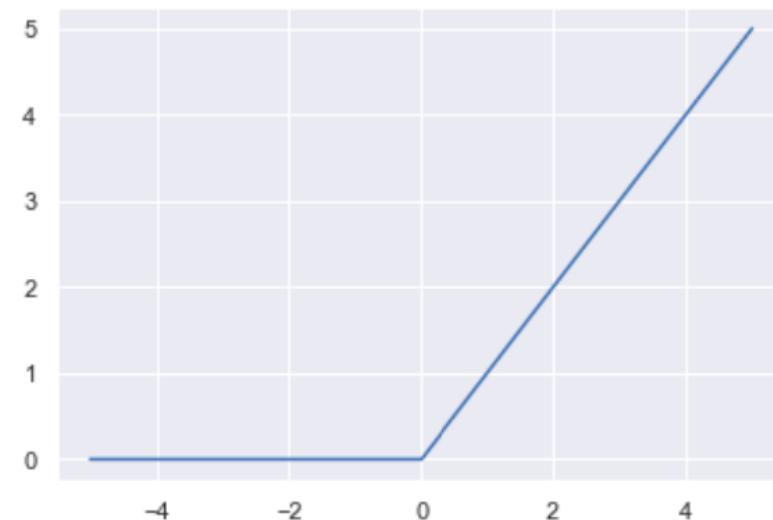


- Максимальное значение производной 1
- Выход центрирован, в нуле значение 0
- По-прежнему насыщение на краях

Rectified Linear Unit

$$f(x) = \max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



Плюсы

- Нет насыщения, ненулевые градиенты
- Вычислительно очень дешево

Минус

- “Dead neurons” - нейроны, для которых всегда $x < 0$
- Проблема может усугубляться при большом learning rate

Модификации ReLU

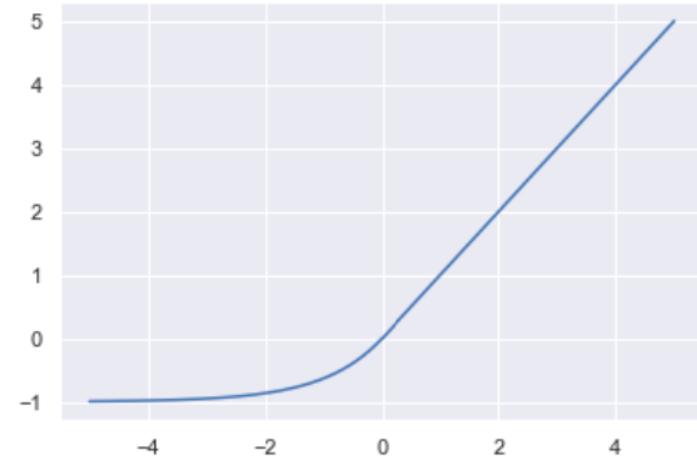
Leaky ReLU, PReLU

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$



ELU

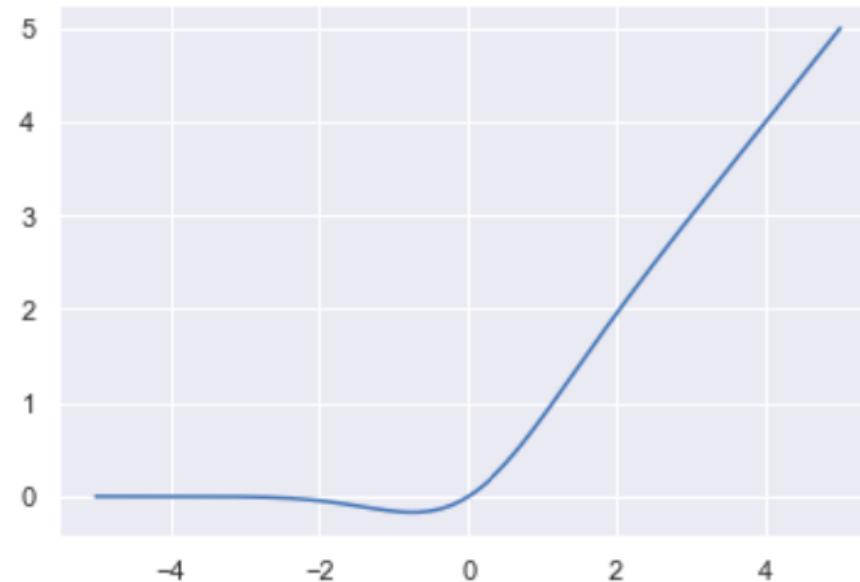
$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$$



Решаем проблему “Dead neurons”

Gaussian Error Linear Unit

$$f(x) = xP(X \leq x), \\ X \sim N(0, 1)$$



$$f(x) \approx \begin{cases} 0.5x + \frac{1}{2\pi}x^2, & |x| \ll 1 \\ ReLU(x), & |x| \rightarrow \infty \end{cases}$$

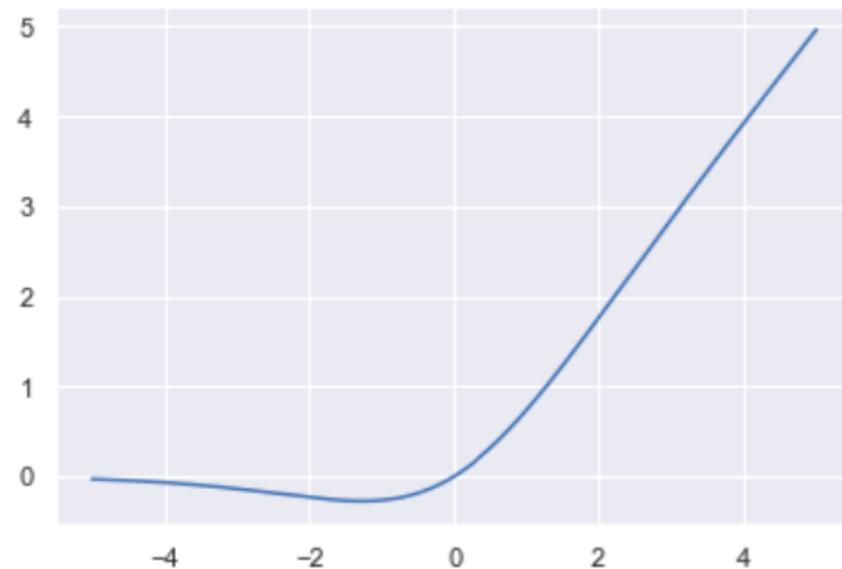
Обычно используется в трансформерах

Swish

Swish, он же SiLU (Sigmoid Linear Unit)

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

$$f'(x) = f(x) + \sigma(x)(1 - f(x))$$



Влияние активации на нейросети

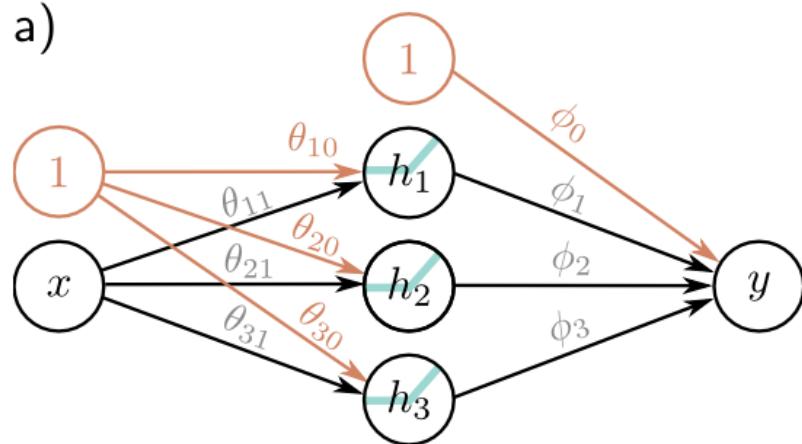
$$h_1 = a(\theta_{10} + \theta_{11}x)$$

$$h_2 = a(\theta_{20} + \theta_{21}x)$$

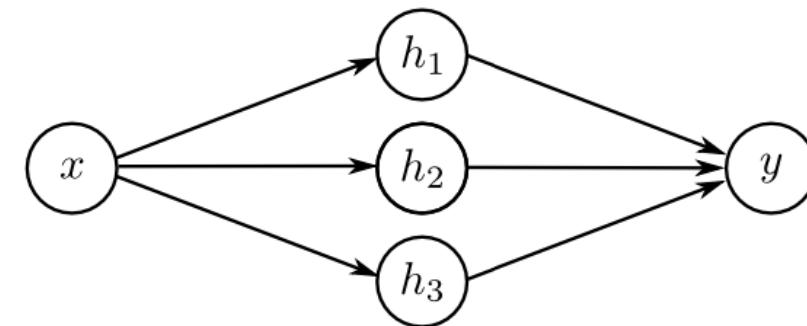
$$h_3 = a(\theta_{30} + \theta_{31}x)$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

a)

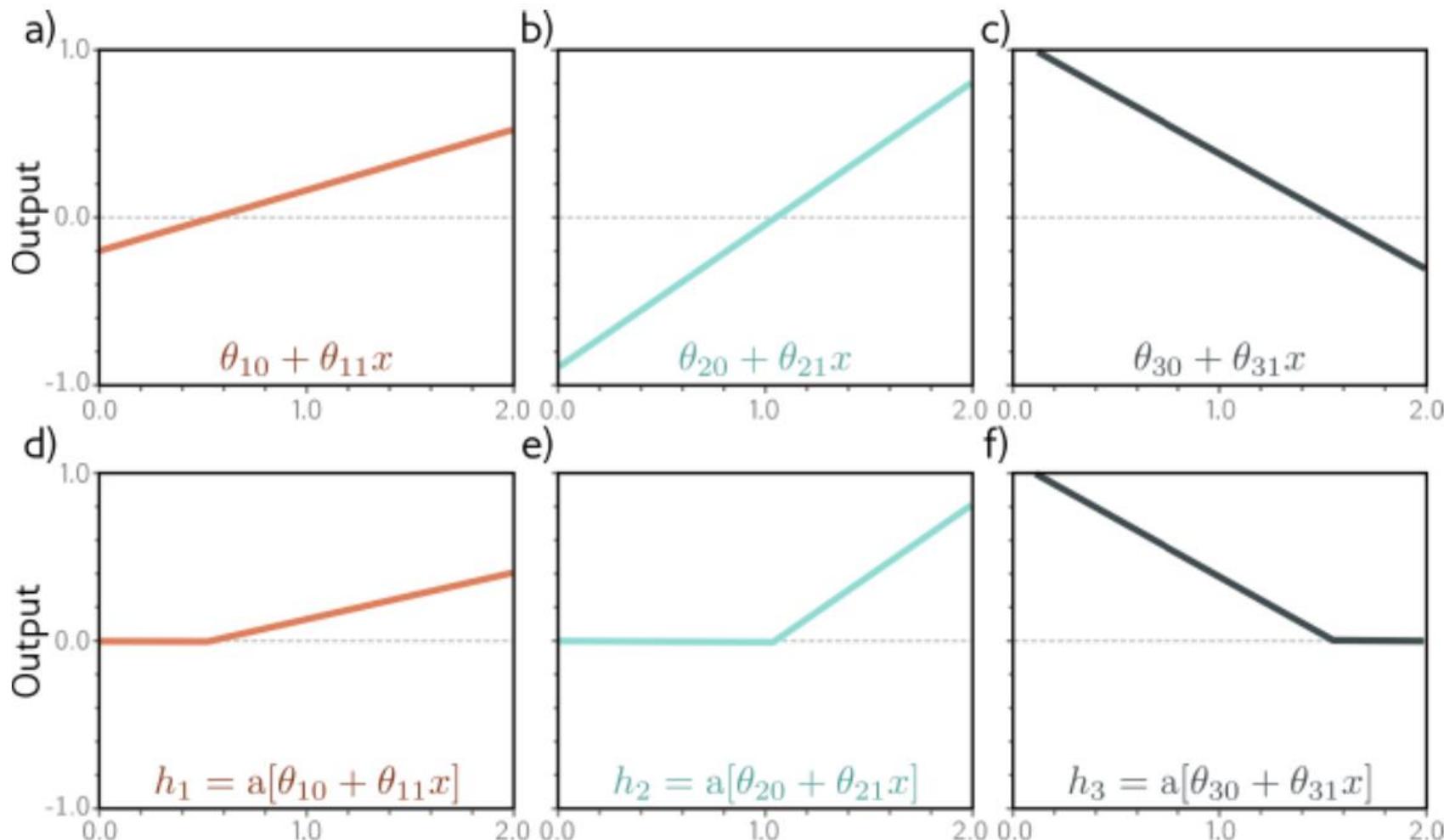
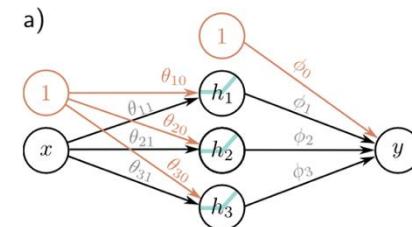


b)



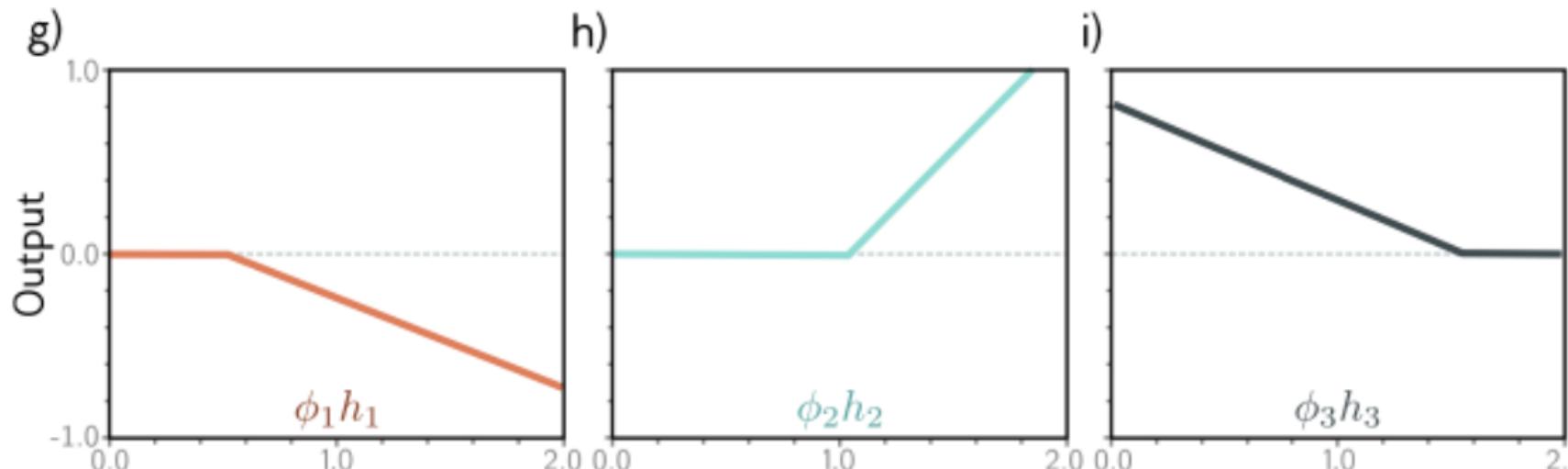
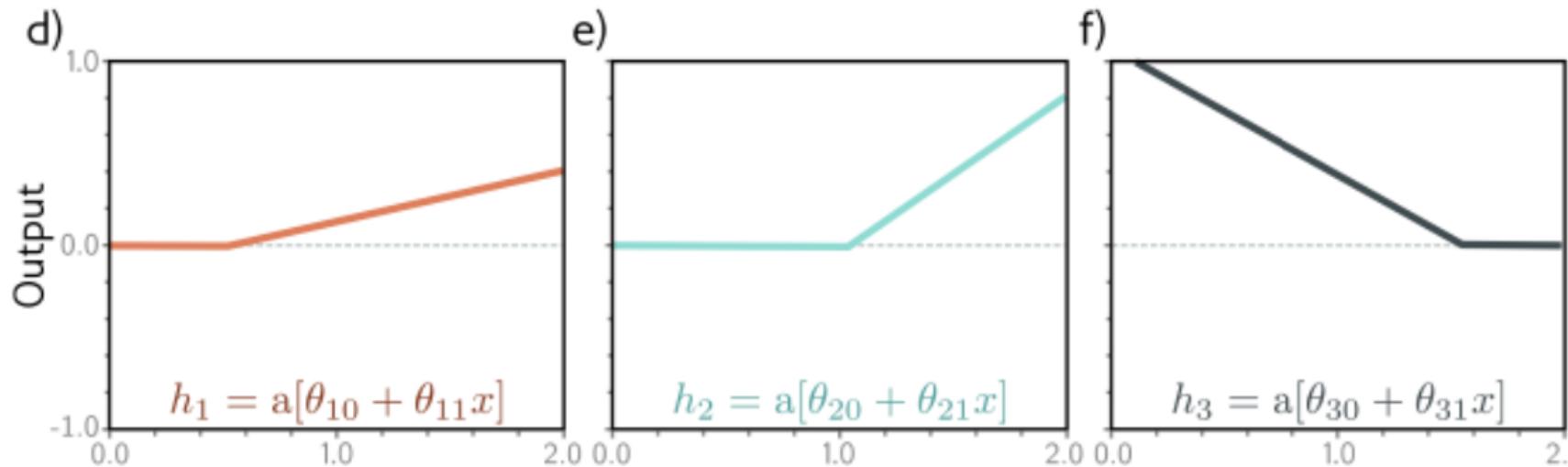
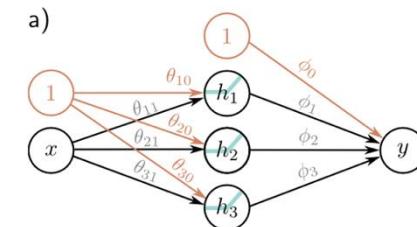
Нейросеть с ReLU

Линейное преобразование + ReLU



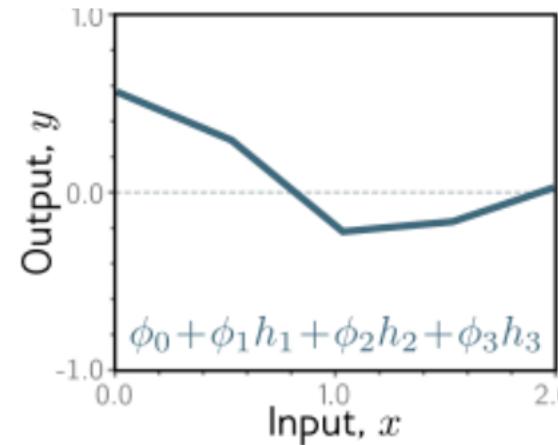
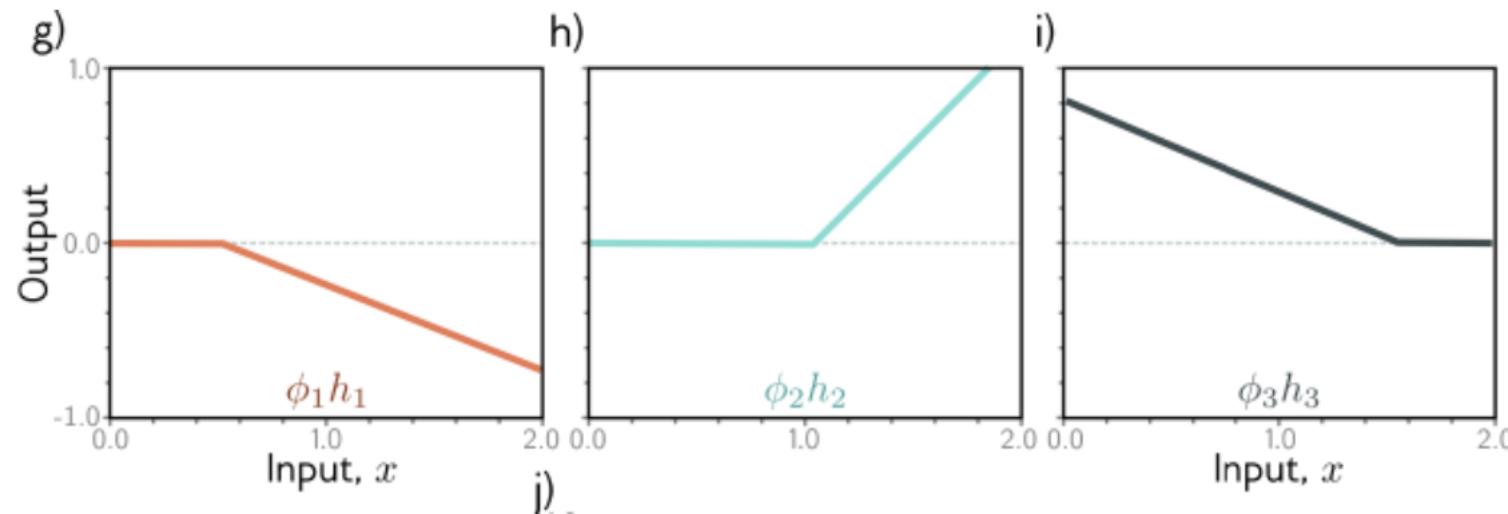
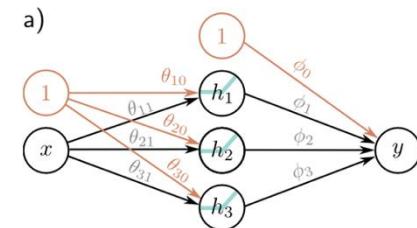
Нейросеть с ReLu

Умножение выходов скрытого слоя на веса на выходном слое

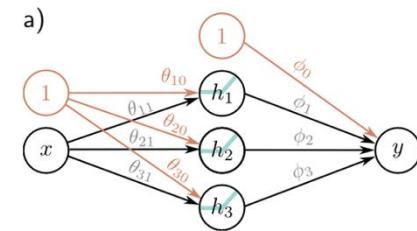


Выходной слой

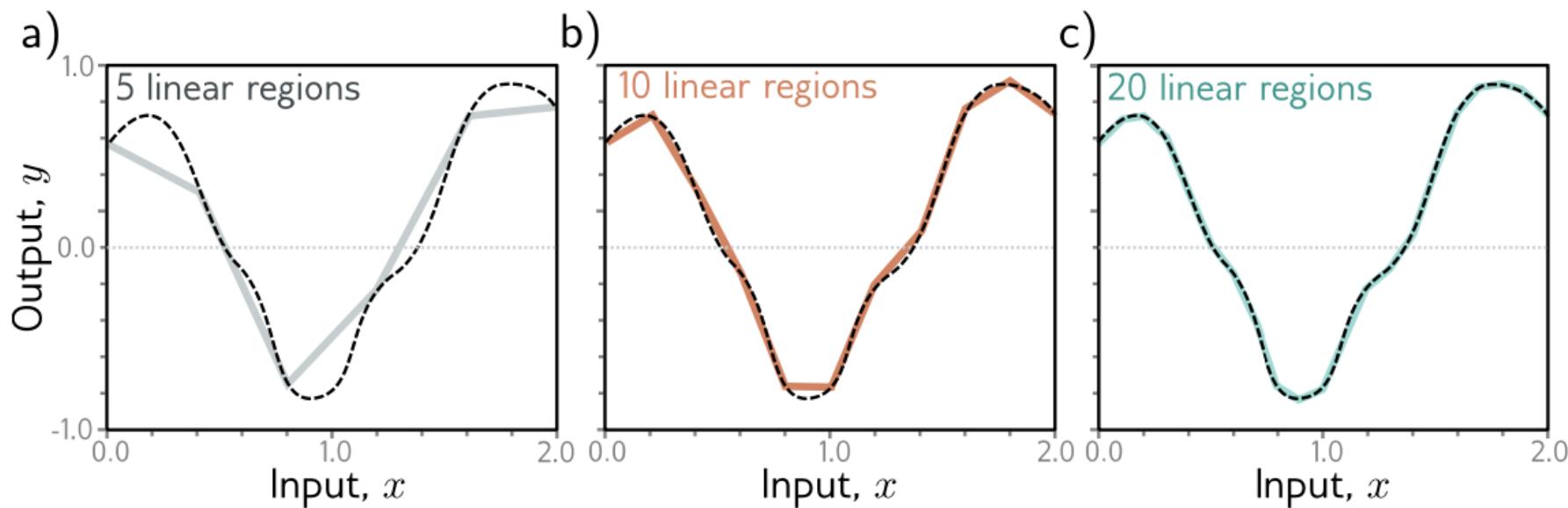
Суммирование на выходном слое



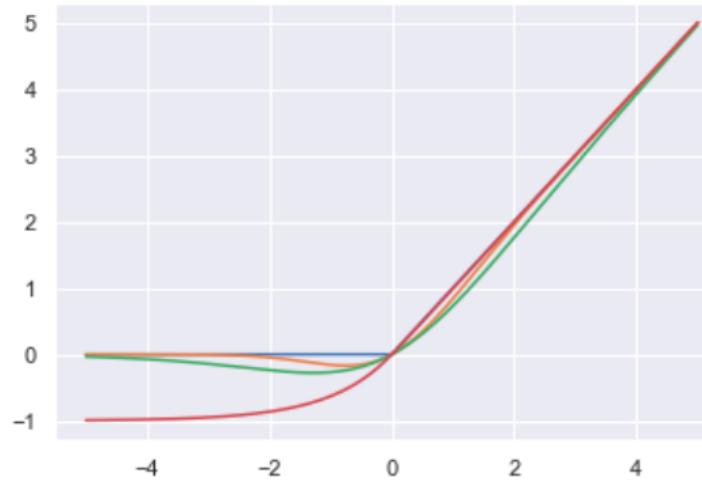
Универсальный аппроксиматор



Universal approximation theorem



Тейк Хоум

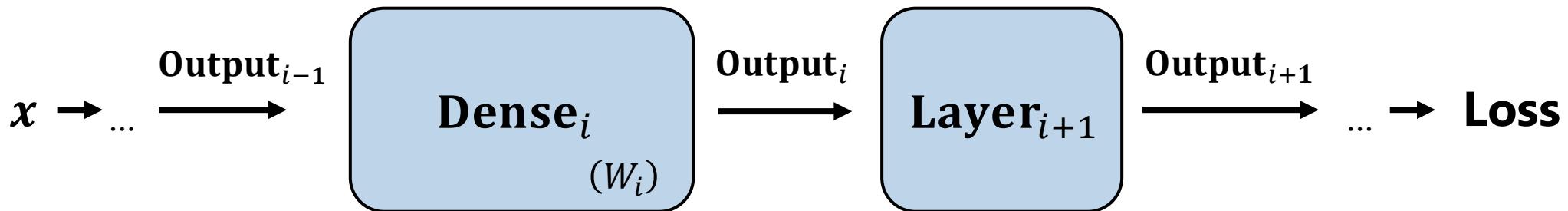


- Избегать сигмоиды
- ReLU - хороший выбор в большинстве случаев
- Модификации ReLU могут чуть-чуть улучшить качество
- Подбор функции активации - не первый приоритет

Инициализация весов



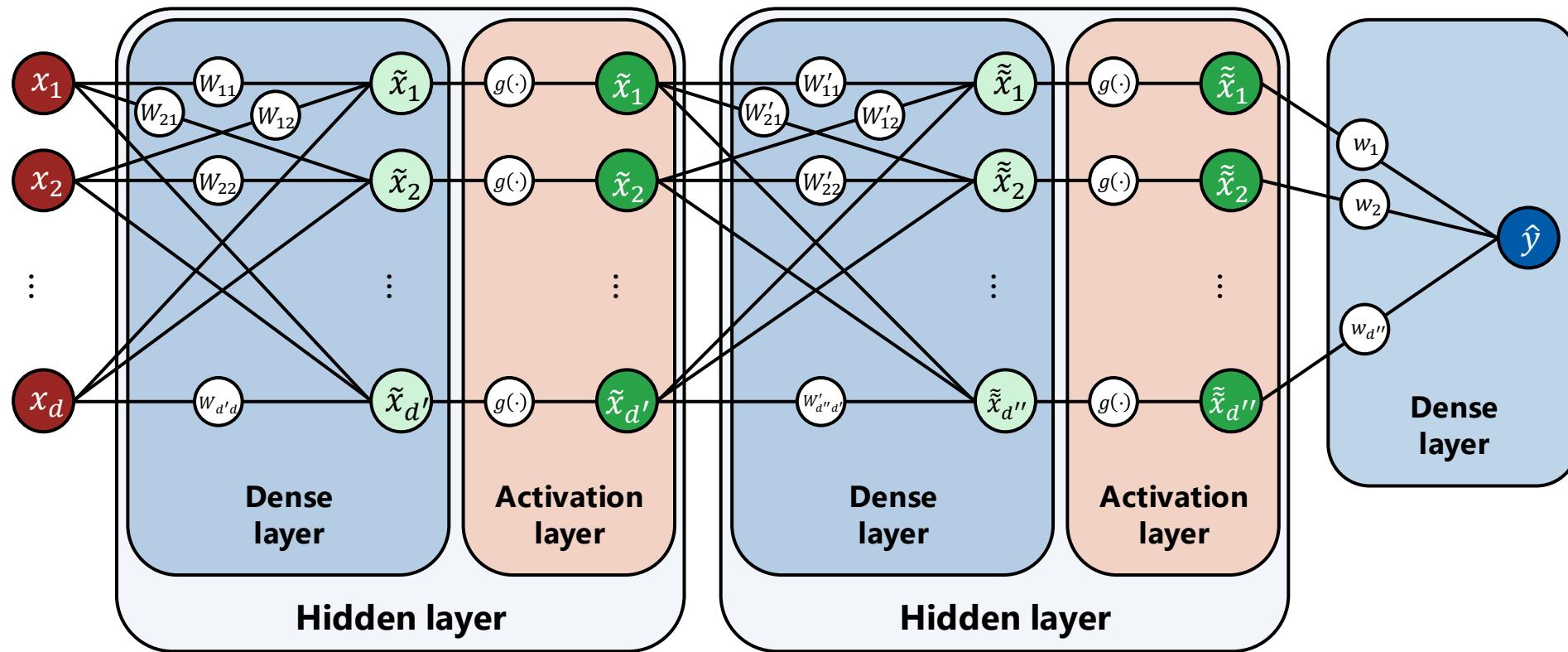
Reminder Ещё совсем много интуиции



- ▶ Аналогично, мы также хотели бы не масштабировать выходные данные на каждом этапе прямого прохода:

$$\text{Var}(\text{Layer}_{i+1}(\text{Layer}_i(\text{Output}_{i-1}))) \approx \text{Var}(\text{Layer}_i(\text{Output}_{i-1}))$$

Initialization with a constant (?)



- ▶ What happens if we initialize all weights with the same value?
- ▶ Within each layer, the gradients for each of the weights will be the same as well ⇒ **updates will be the same** ⇒ network degrades!

Случайная инициализация

$$\text{Var}\left(\frac{\partial \text{Layer}_{i+1}}{\partial \text{Output}_i} \cdot \frac{\partial \text{Layer}_i}{\partial \text{Output}_{i-1}}\right) \approx \text{Var}\left(\frac{\partial \text{Layer}_{i+1}}{\partial \text{Output}_i}\right)$$

$$\text{Var}\left(\text{Layer}_{i+1}(\text{Layer}_i(\text{Output}_{i-1}))\right) \approx \text{Var}(\text{Layer}_i(\text{Output}_{i-1}))$$

- ▶ Вообще, требования могут стать противоречивыми
- ▶ Например, для ReLU:

$$\text{Var}(W_{ij}) = \frac{2}{(\# \text{ outgoing connections})}$$

$$\text{Var}(W_{ij}) = \frac{2}{(\# \text{ incoming connections})}$$

- ▶ Часто можно использовать одно из них, или комбинацию:

$$\text{Var}(W_{ij}) = \frac{4}{(\# \text{ outgoing connections}) + (\# \text{ incoming connections})}$$

Xavier (Glorot) initialization

Understanding the difficulty of training deep feedforward neural networks -
Glorot, X. & Bengio, Y. (2010)

- Для симметричных функций активации
- Идея - дисперсии выходов и градиентов на всех слоях должны быть одинаковыми

Рассмотрим один нейрон $y = w^T x = \sum_{i=1}^{n_{in}} w_i x_i$

В силу независимости и одинаковой распределенности w_i, x_i

$$Var[y] = Var \left[\sum_{i=1}^{n_{in}} w_i x_i \right] = \sum_{i=1}^{n_{in}} Var[w_i x_i] = n_{in} Var[w_i x_i]$$

$$\begin{aligned} Var[w_i x_i] &= E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] = \\ &= Var[w_i] Var[x_i] \end{aligned}$$

Xavier (Glorot) initialization

$$Var[y] = n_{in}Var[w_i x_i] = n_{in}Var[w_i]Var[x_i]$$

Получили условие для forward pass

$$n_{in}Var[w_i] = 1$$

Можно получить такое же условие для backward pass

$$n_{out}Var[w_i] = 1$$

Как объединить?

$$Var[w_i] = \frac{2}{n_{in} + n_{out}}$$

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}\right]$$

(Kaiming) He Intialization

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification - He, K. et al. (2015)

- Для несимметричных функций активации (ReLU)
- Идея - дисперсии выходов или градиентов на всех слоях должны быть одинаковыми

$$\begin{aligned}Var[w_i x_i] &= E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] = \\&= E[x_i]^2 Var[w_i] + Var[w_i] Var[x_i] \\&= Var[w_i] (E[x_i]^2 + Var[x_i])) = Var[w_i] E[x_i^2]\end{aligned}$$

Для ReLU

$$E[x_i^2] = \frac{1}{2} Var[y_{prev}]$$

$$Var[y] = n_{in} Var[w_i x_i] = \frac{1}{2} n_{in} Var[w_i] Var[y_{prev}]$$

(Kaiming) He Intialization

Либо используем условие для forward pass

$$Var[w_i] = \frac{2}{n_{in}}$$

$$w_i \sim N(0, \sqrt{2/n_{in}})$$

или

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in}}}, \frac{\sqrt{6}}{\sqrt{n_{in}}}\right]$$

Либо условие для backward pass

$$Var[w_i] = \frac{2}{n_{out}}$$

Ортогональная инициализация

Ортогональные матрицы - повороты и отражения

$$A^T A = I$$

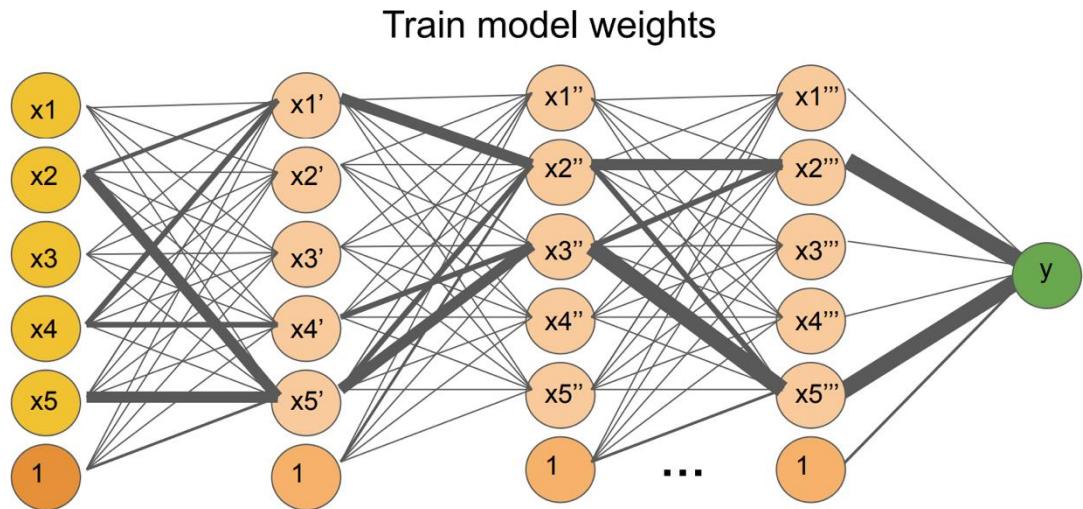
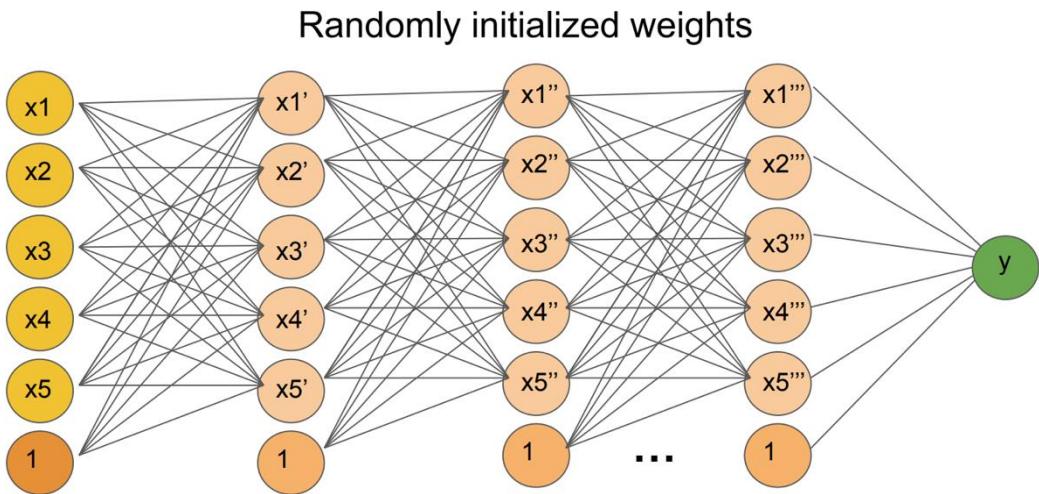
Столбцы (и строки) ортнормированы: $\sum_i A_{ij} A_{ik} = \delta_{ik}$

Не изменяют длины векторов:

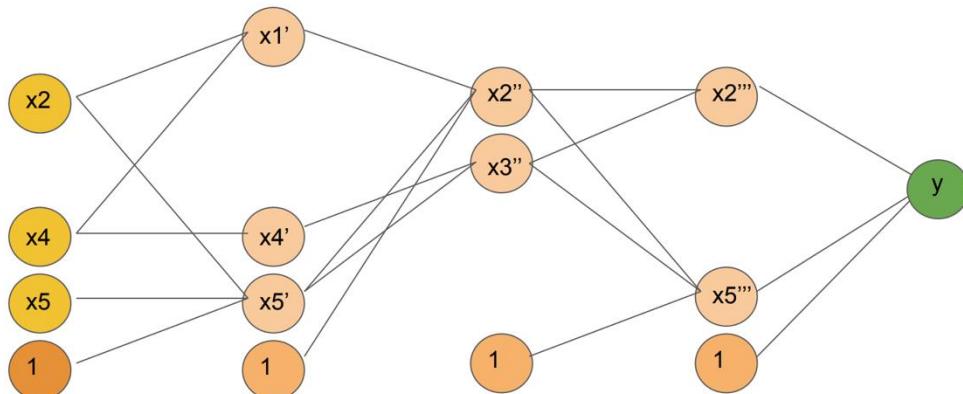
$$\|Ax\| = \|x\|$$

Поэтому градиенты не будут взрываться и затухать

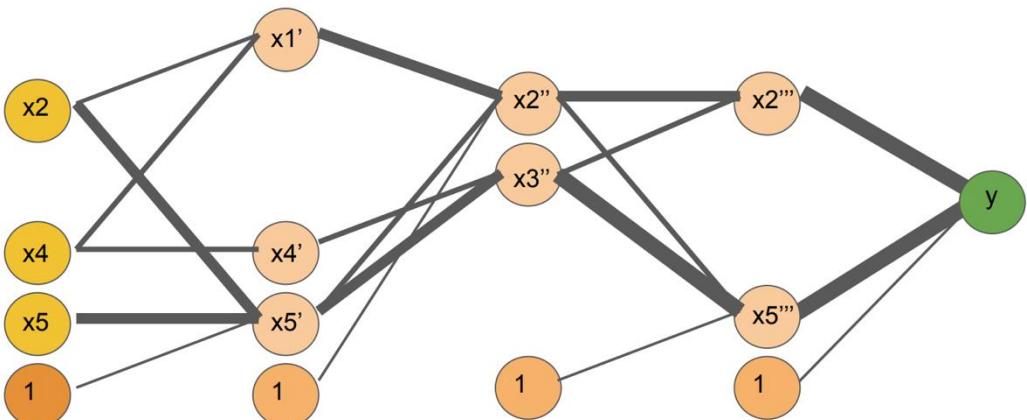
Лотерейные билеты



Restore to the same random weight initialization as before

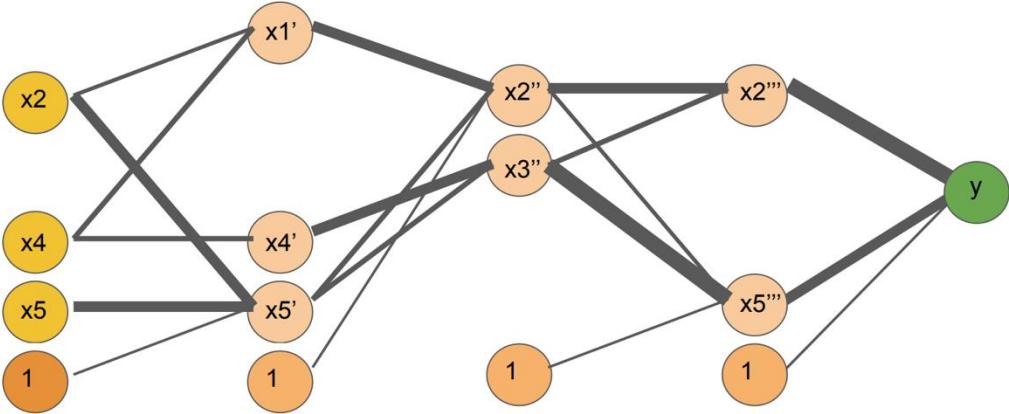


Prune trained weights



Лотерейные билеты

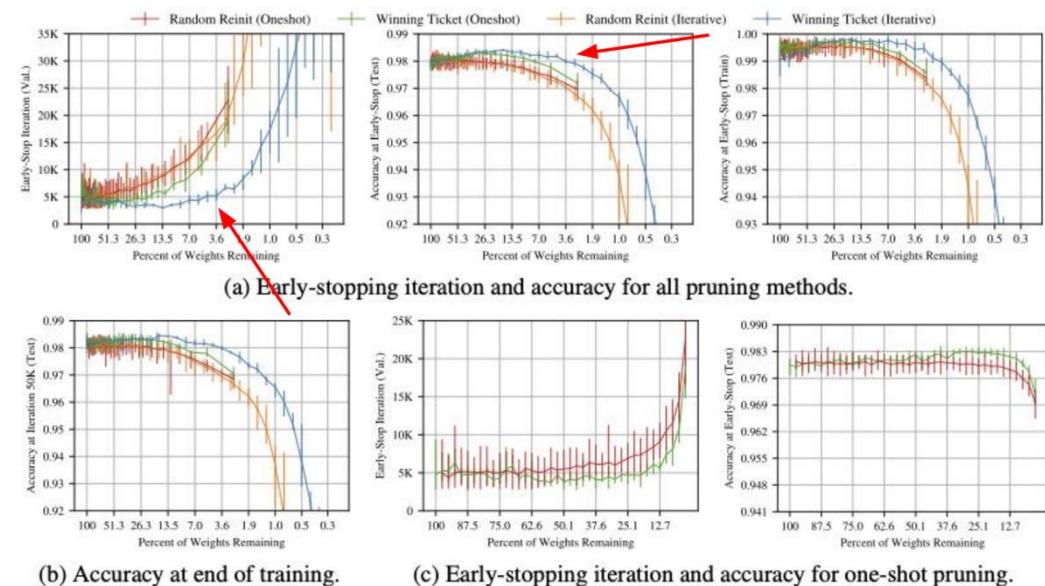
Retrain for same or even higher model performance!!!



- Глубокие нейронные сети очень перепараметризованы
- Эта перепараметризация покупает много билетов
- С таким количеством билетов один часто будет победителем!

Случайно инициализированная, плотная нейронная сеть содержит подсеть, которая инициализируется таким образом, что при обучении в изоляции она может соответствовать тестовой точности исходной сети после обучения максимум на том же количестве итераций.

<https://arxiv.org/abs/1803.03635>



Финальные мысли

- ▶ Правильная инициализация имеет гораздо большее значение, чем функция активации.
- ▶ Полностью случайная инициализация может также нарушить сходимость, потому надо использовать одну из предложенных.

Функции потерь



Оценка максимального правдоподобия

Модель $f(x, \theta)$ определяет параметры распределения вероятности $P(y|x)$:

$$P(y|x) = P(y|f(x, \theta))$$

Функция правдоподобия:

$$L(\theta) = P(D|\theta) = \prod_{i=1}^N P(y_i|f(x_i, \theta))$$

Метод максимального правдоподобия (maximum likelihood estimation):

$$\theta = \operatorname{argmax}_{\theta} L(\theta) = \operatorname{argmax}_{\theta} \left[\prod_{i=1}^N P(y_i|f(x_i, \theta)) \right]$$

Оценка максимального правдоподобия

$$\begin{aligned}\theta &= \operatorname{argmax}_{\theta} \left[\log \left(\prod_{i=1}^N P(y_i | f(x_i, \theta)) \right) \right] \\ &= \operatorname{argmax}_{\theta} \left[\sum_{i=1}^N \log (P(y_i | f(x_i, \theta))) \right] \\ &= \operatorname{argmin}_{\theta} \left[- \sum_{i=1}^N \log (P(y_i | f(x_i, \theta))) \right]\end{aligned}$$

Negative log-likelihood loss:

$$\mathcal{L} = - \sum_{i=1}^N \log (P(y_i | f(x_i, \theta)))$$

Ординарная регрессия

Предположим, что шум распределен нормально с центром в нуле:

$$y = f(x, \theta) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

то есть

$$\begin{aligned} P(y|x, \theta, \sigma) &= \mathcal{N}(y|f(x, \theta), \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y - f(x, \theta))^2}{2\sigma^2} \right] \end{aligned}$$

$$\begin{aligned} \mathcal{L} &= - \sum_{i=1}^N \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y - f(x, \theta))^2}{2\sigma^2} \right] \right) \\ &= \sum_{i=1}^N \left[-\log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \frac{(y - f(x, \theta))^2}{2\sigma^2} \right] \rightarrow (y - f(x, \theta))^2 \end{aligned}$$

Получили обычный метод наименьших квадратов

Обобщённые линейные модели и другие

- Можем предсказывать и среднее, и дисперсию:

$$\theta = \operatorname{argmin}_{\theta} \sum_{i=1}^N \left[-\log \left(\frac{1}{\sqrt{2\pi f_2(x, \theta)^2}} \right) + \frac{(y - f_1(x, \theta))^2}{2f_2(x, \theta)^2} \right]$$

- Если использовать распределение Лапласа, то получим МАЕ
- Если использовать распределение Пуассона, то можем предсказывать счетчики событий
- Если использовать бета-распределение, то можем предсказывать пропорции

Бинарная классификация

Распределение Бернулли

$$P(y|\lambda) = \lambda^y(1-\lambda)^{1-y} = \begin{cases} \lambda, & y = 1 \\ 1 - \lambda, & y = 0 \end{cases}$$

Чтобы оценивать λ - вероятность от 0 до 1, используем сигмоиду:

$$f(x, \theta) = \sigma(x, \theta)$$

$$P(y|f(x, \theta)) = f(x, \theta)^y(1 - f(x, \theta))^{1-y}$$

Функция потерь - binary cross-entropy

$$\mathcal{L}(\theta) = \sum_{i=1}^N [-y_i \log f(x_i, \theta) - (1 - y_i) \log(1 - f(x_i, \theta))]$$

Многоклассовая классификация

Softmax:

$$\text{softmax}_k(z) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

$$P(y = k | f(x, \theta)) = \text{softmax}_k [f(x, \theta)]$$

Функция потерь - cross-entropy (aka softmax loss)

$$\mathcal{L}(\theta) = - \sum_{i=1}^N \log (\text{softmax}_{y_i} [f(x, \theta)])$$

Softmax поведение на границах

$$\text{softmax}_k(z) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

Из-за численного переполнения

- При больших отрицательных значениях z можем получить 0 в знаменателе
- При больших положительных значениях z можем получить бесконечность

Трюк:

$$\frac{e^{z_k+c}}{\sum_{j=1}^K e^{z_j+c}} = \frac{e^{z_k} e^c}{\sum_{j=1}^K e^{z_j} e^c} = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

$$z_k \rightarrow z_k - \max_j z_j$$

Другие функции потерь

- Hinge loss
- Focal loss
- Dice loss
- Triplet loss
- Contrastive loss

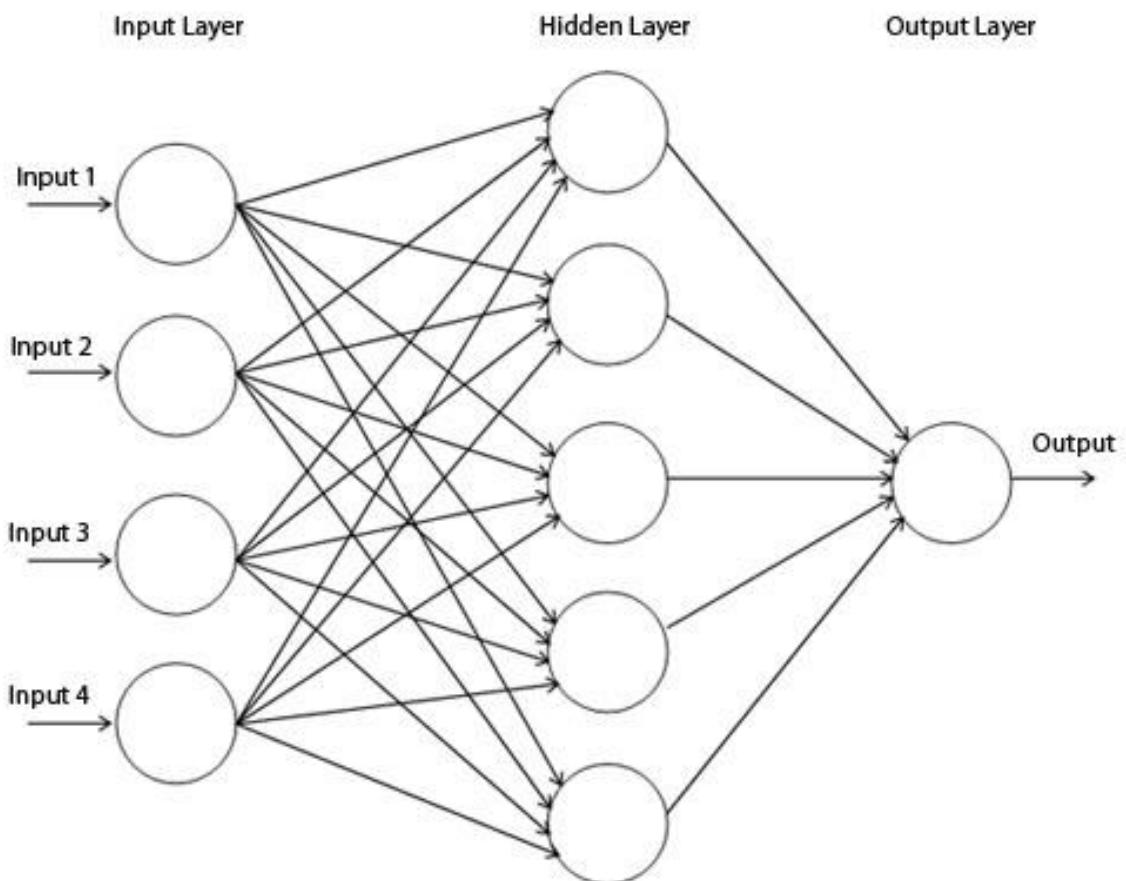
И многие другие..

Архитектуры нейронных сетей



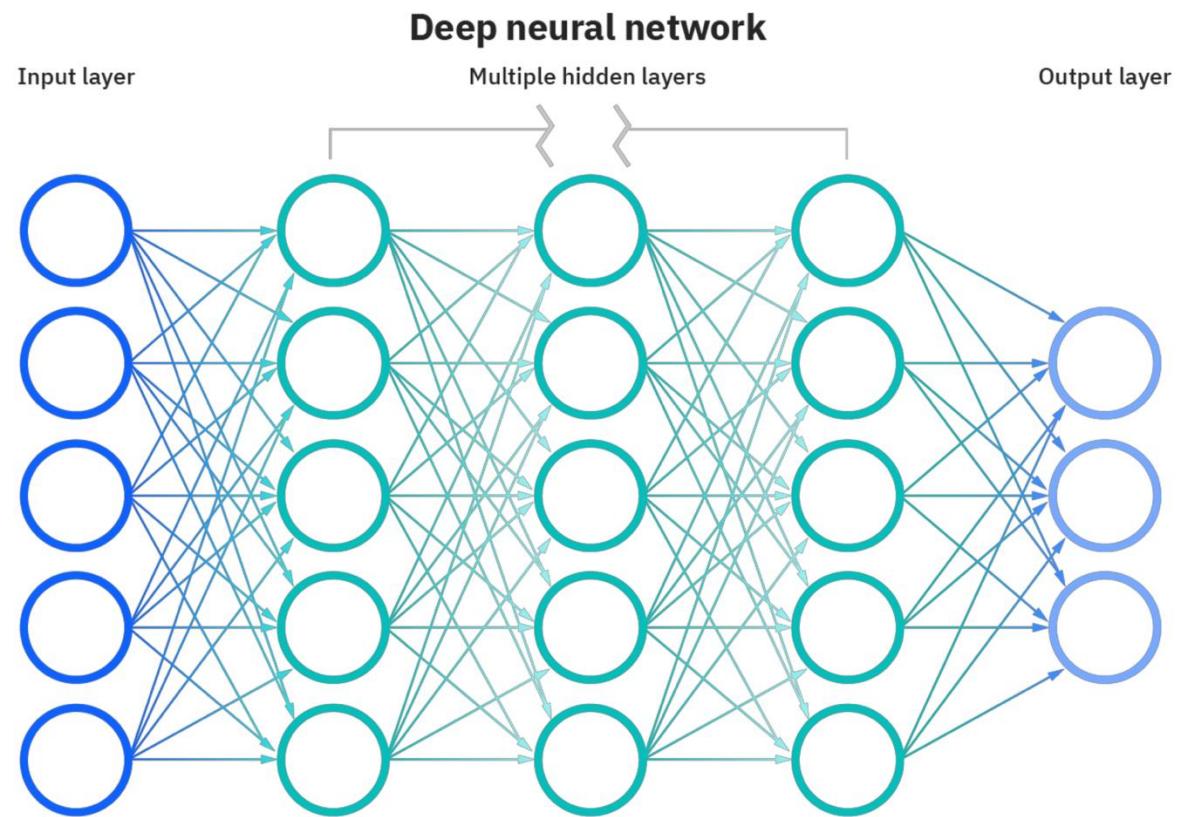
Один скрытый слой

- легко оптимизировать;
- универсальный аппроксиматор;
- часто требуется большое количество скрытых нейронов.



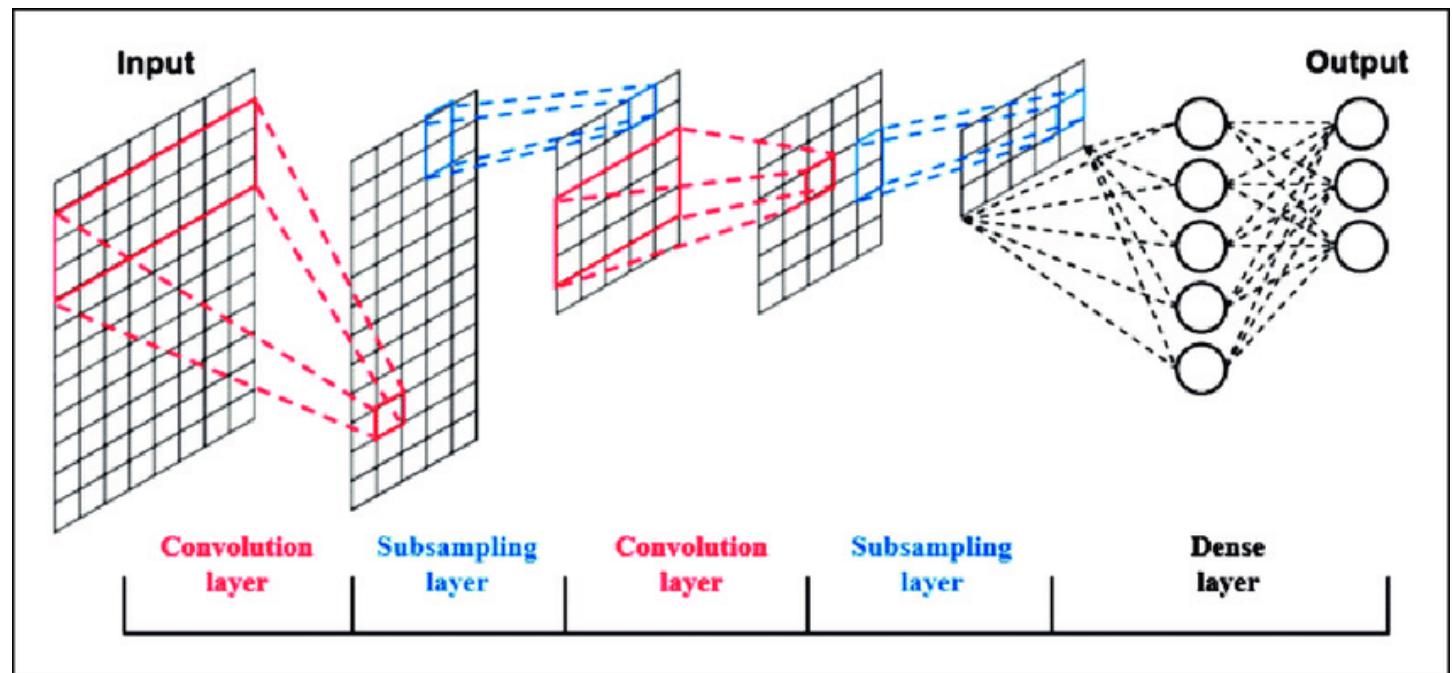
Полносвязная сеть

- труднее оптимизировать;
- лучше подходит для решения реальных задач;
- типичная сеть имеет 4-7 слоев;
- 10+ слоев - сложнее оптимизировать;
- подходит для широкого класса задач;
- начальная точка для других архитектур.



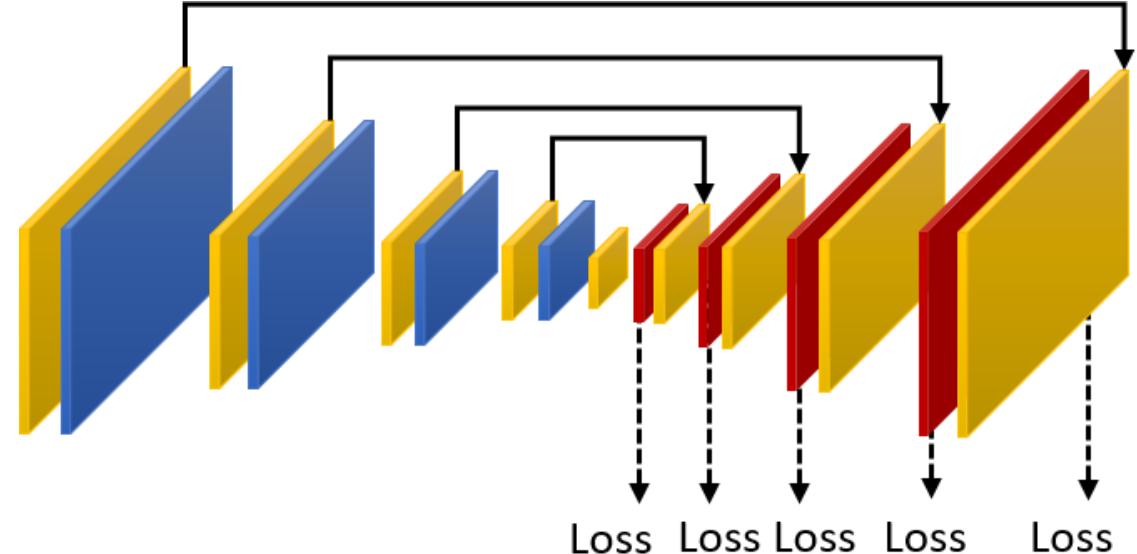
Свёрточные сети

- пространственная/временная структура;
- коммутативный с переносом;
- локальный.



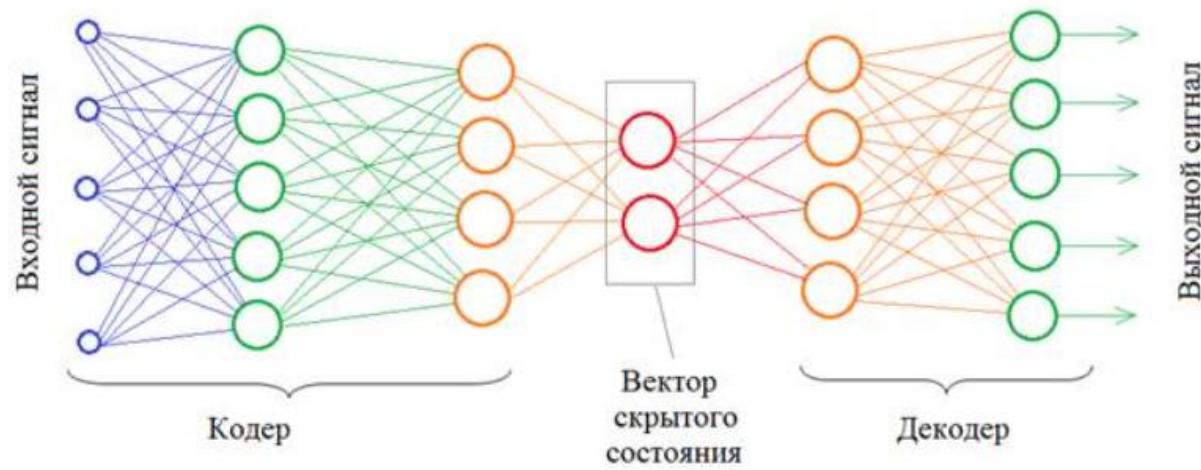
Deeply supervised

- заставляет скрытые слои создавать прогностические характеристики:
 - не всегда хорошая идея;
- сильный регуляризатор;
- прост в реализации;
- отсутствие исчезающих градиентов.



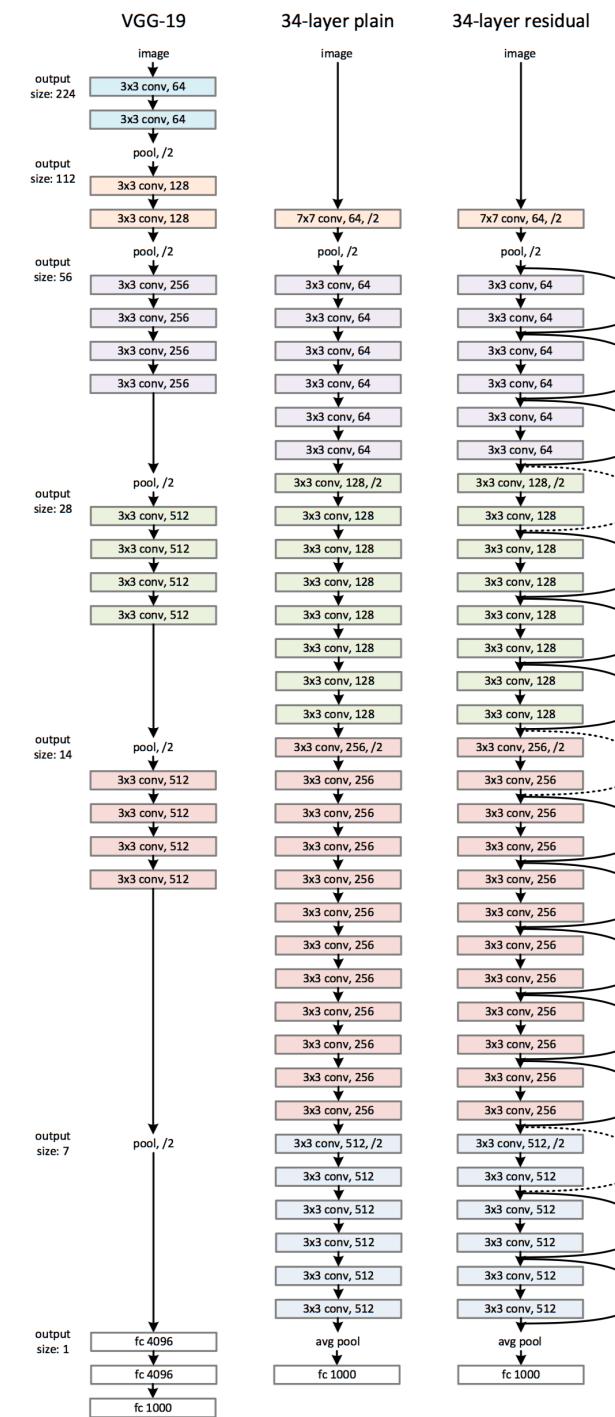
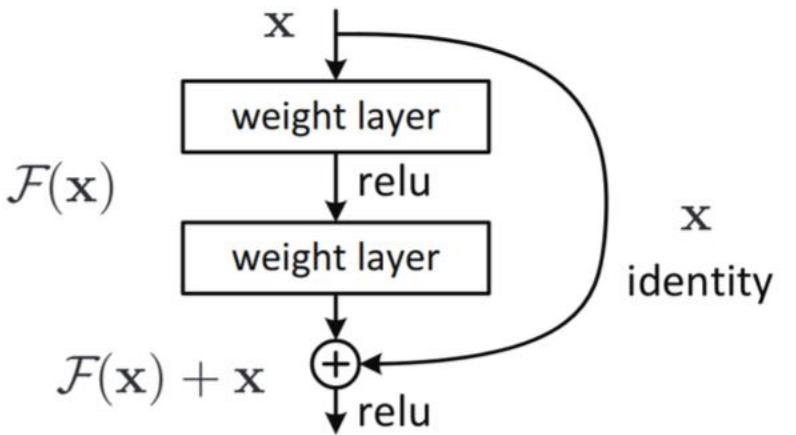
АВТОКОДИРОВЩИКИ

- ▶ dimensionality reduction:
 - $\mathcal{X} \mapsto \mathcal{Z}$;
- ▶ image to image:
 - $\mathcal{X} \mapsto \mathcal{X}$;
- ▶ anomaly detection:
 - $\text{score}(x) = \|\text{decoder}(\text{encoder}(x)) - x\|^2$;
- ▶ auxiliary:
 - denoising;
 - pretraining;
 - auxiliary loss, regularization.



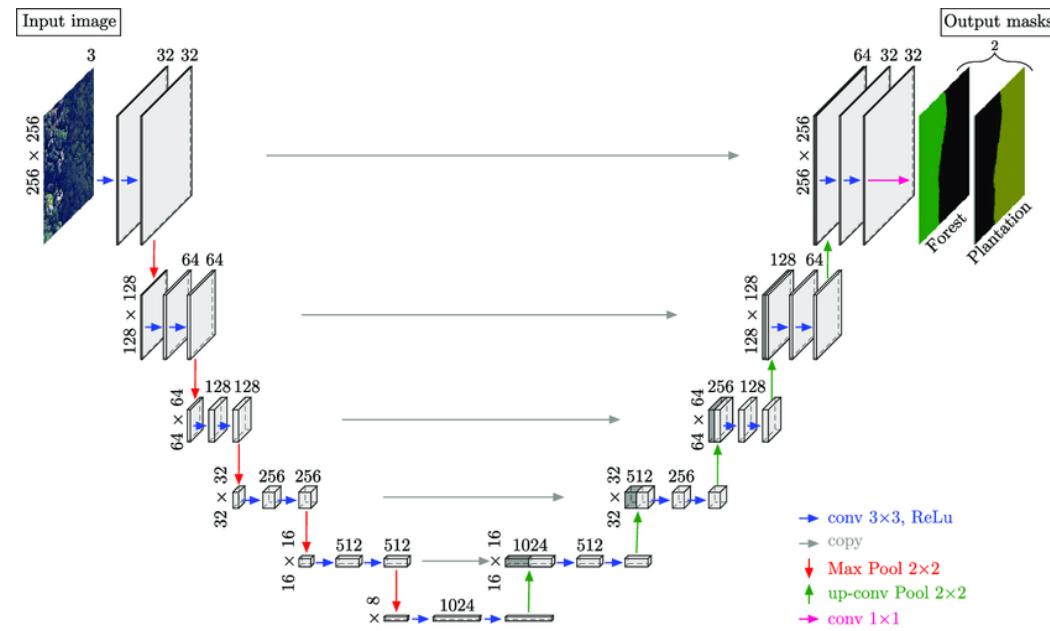
ResNet

- ▶ boosting, neural edition;
 - ▶ adaptive depth:
 - shallow initially;
 - ▶ 1000+ layers!



U-NET

- изображение к изображению:
 - увеличение/трансформация; - сегментация;
- легче обучается;
- объединяет глобальную и локальную информацию.

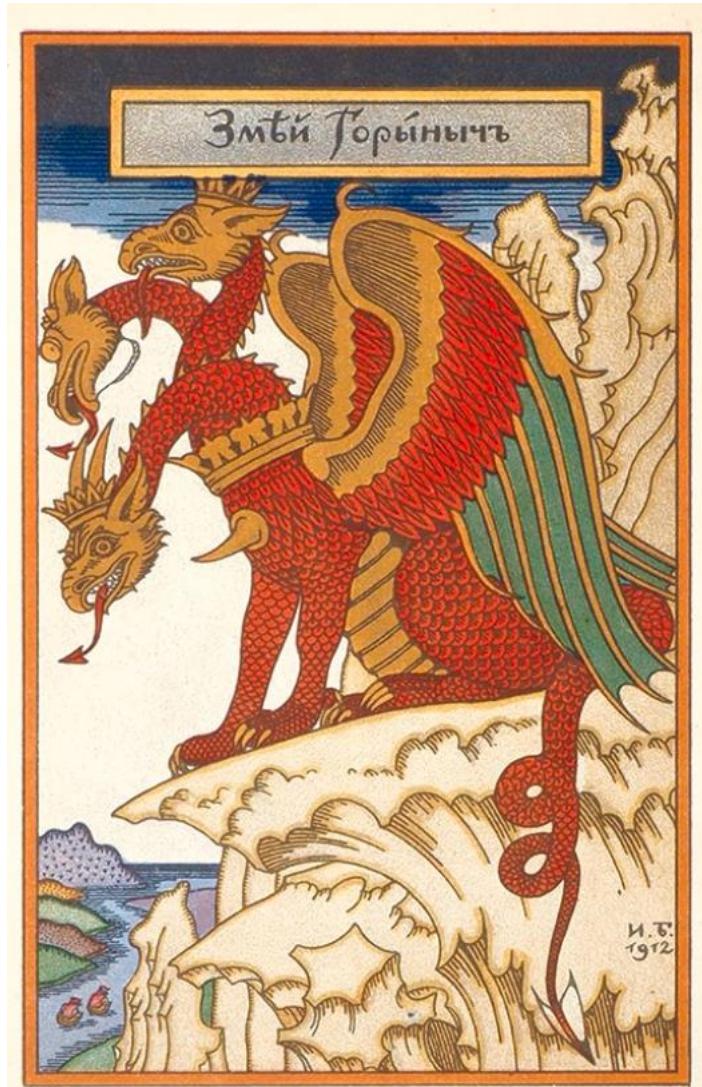


Дополнительные задачи

- ▶ similar task tends to share similar features;

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \sum_i \alpha_i \cdot \mathcal{L}_{\text{aux}}^i$$

- ▶ α_i can be decreased during training.

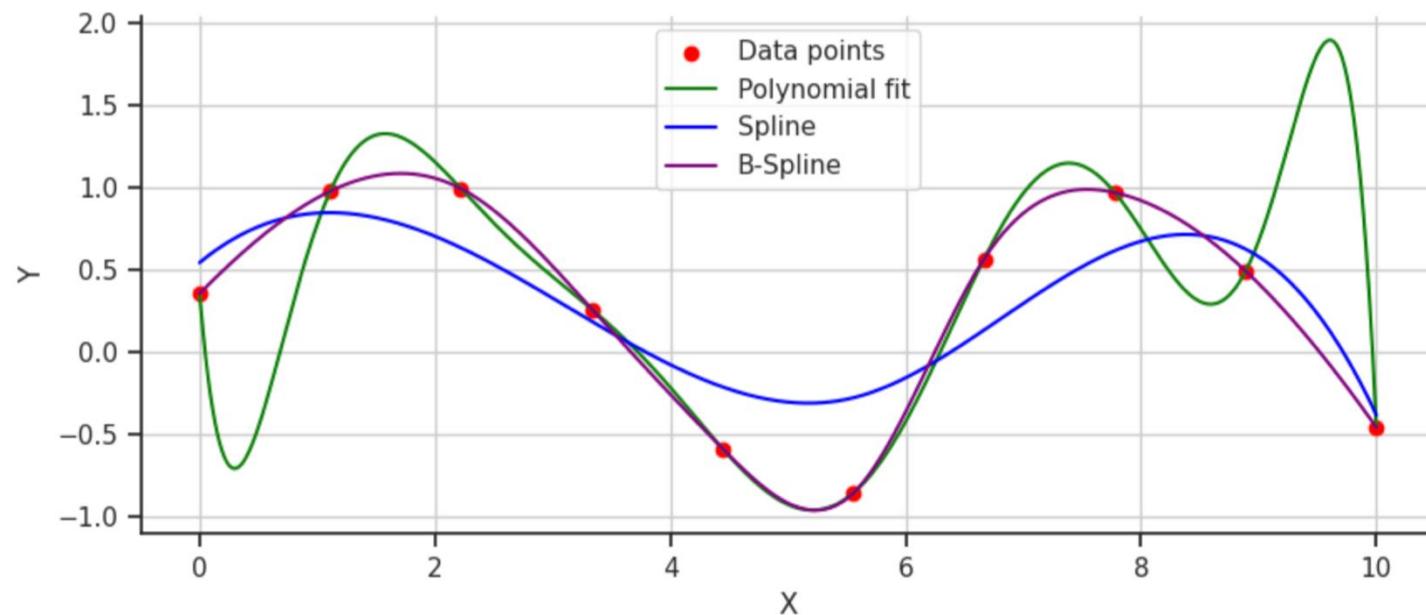


В-сплайны

- Полиномы могут иметь слишком серьёзные «флуктуации»
 - Феномен Рунге из-за положения узловых точек
- Сплайны могут не достаточно хорошо проходить через точки.
- Выход – В сплайн

$$C(t) = \sum_{i=0}^n P_i N_{i,k}(t)$$

- Дополнительные контрольные точки и базисные функции



https://ru.wikipedia.org/wiki/Феномен_Рунге

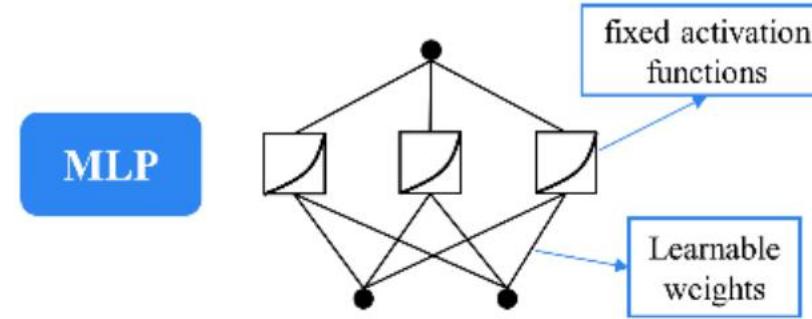
<https://daniel-bethell.co.uk/posts/kan/>

Сети Колмогорова-Смирнова

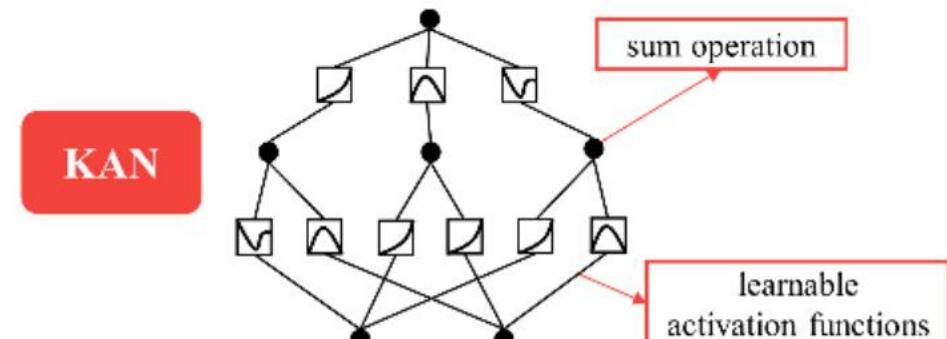
$$\text{MLP: } f(x) = \sigma(Wx + b) \rightarrow \text{KAN: } f(x) = \Phi \left(\sum_i \phi_{ij}(x_i) \right)$$

Каждое соединение становится обучаемой одномерной функцией (В-сплайном) вместо скалярного веса.

Таким образом получается обратная архитектура к обычной нейросети.



$$\text{MLP}(\mathbf{x}) = (\mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$$



$$\text{KAN}(\mathbf{x}) = (\Phi_2 \circ \Phi_1)(\mathbf{x})$$

<https://arxiv.org/html/2404.19756v1>

<https://arxiv.org/pdf/2409.03430.pdf>

MLP VS KAN

► Architecture:

- Weights: scalars $w \in \mathbb{R}$
- Activations: fixed functions (ReLU, sigmoid, tanh)
- Nonlinearity in nodes

► Advantages:

- Fast training (matrix multiplications)
- Excellent GPU scalability
- Mature ecosystem and libraries
- Effective for large-scale problems

► Limitations:

- Black box, low interpretability
- Requires many parameters for complex functions
- Fixed nonlinearities may be suboptimal
- Prone to overfitting on small datasets

► Architecture:

- Weights: functions $w : \mathbb{R} \rightarrow \mathbb{R}$
- Activations: learnable splines (B-splines)
- Nonlinearity on edges

► Advantages:

- High interpretability (visualizable functions)
- **10-100× fewer parameters** for comparable accuracy
- Superior function approximation capability
- Automatic discovery of optimal nonlinearities

► Limitations:

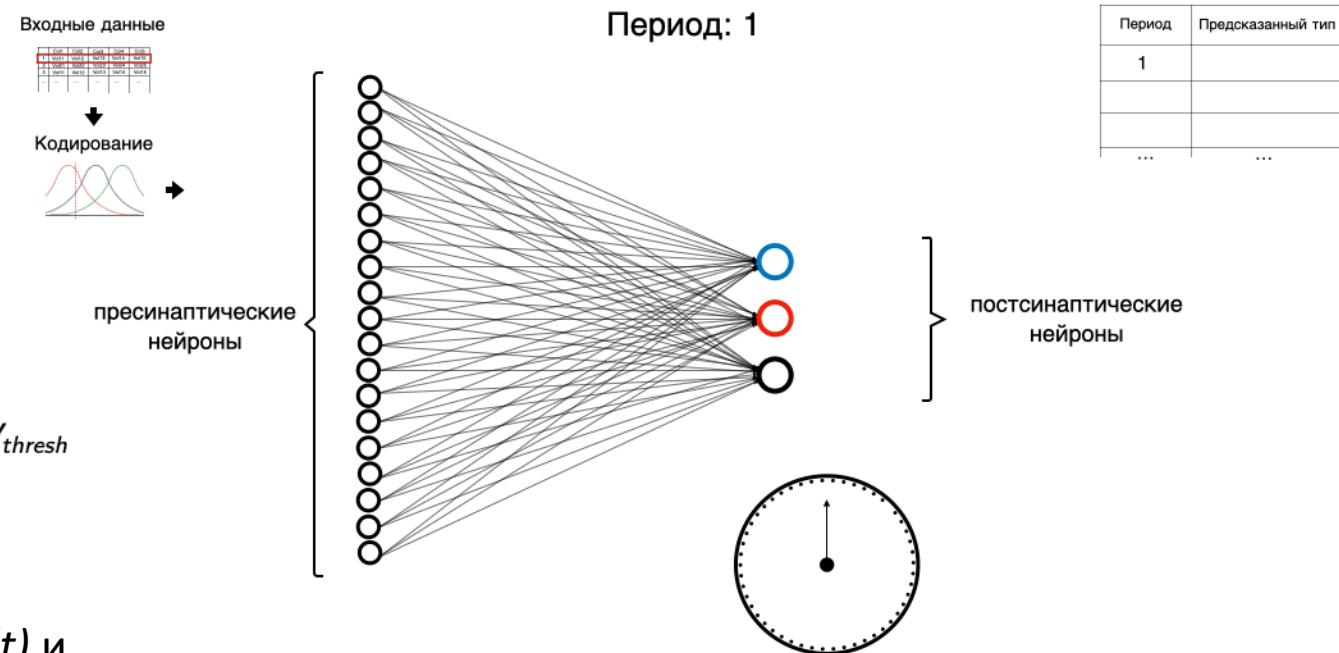
- Slower training (function optimization)
- Experimental technology (2024)
- Scaling challenges for very large models

Импульсные нейронные сети

Время как вычислительная составляющая:

SNN используют точное время возникновения импульсов для вычислений, что позволяет осуществлять энергоэффективное обучение, вдохновленное работой мозга.

$$\text{MLP: } y = f(Wx + b) \rightarrow \text{SNN: } \text{Spike}(t) = \begin{cases} 1, & \text{if } V(t) > V_{\text{thresh}} \\ 0, & \text{otherwise} \end{cases}$$



Нейроны накапливают мембранный потенциал $V(t)$ и генерируют импульсы при достижении порогового значения.

<https://habr.com/ru/articles/746762/>

MLP VS SNN

► Computation:

- Continuous-valued activations
- Synchronous, layer-by-layer
- Static input representation

► Advantages:

- Mature training algorithms (backprop)
- Excellent accuracy on standard benchmarks
- Extensive software frameworks (PyTorch, TensorFlow)
- Well-understood optimization techniques

► Limitations:

- High power consumption
- Computations always active
- Not biologically plausible
- Inefficient for temporal data processing

► Computation:

- Binary spikes (0/1) over time
- Asynchronous, event-driven
- Dynamic temporal representation

► Advantages:

- **Extremely energy efficient** (sparse computation)
- Natural for temporal/sequential data
- Compatible with neuromorphic hardware
- Closer to biological neural processing

► Limitations:

- Challenging training (non-differentiable spikes)
- Lower accuracy on some benchmarks
- Less mature software ecosystem

Зоопарк моделей

Architecture	Key Principle	Primary Applications	Era
Classical Architectures (Established)			
MLP/FCN	Fully connected layers, universal approximation	Tabular data, regression, classification	1980s
CNN	Convolution + pooling, spatial hierarchies	Computer vision, image processing	1990s
RNN/LSTM	Hidden state, sequential processing	Time series, NLP, speech recognition	1990s
Autoencoder	Bottleneck representation, reconstruction	Dimensionality reduction, anomaly detection	2000s
GAN	Adversarial training, minimax game	Image generation, data augmentation	2014
ResNet	Skip connections, residual learning	Very deep networks, image classification	2015
U-Net	Encoder-decoder with skip connections	Medical imaging, segmentation	2015
Modern Architectures (Active Research)			
Transformer	Self-attention mechanism, parallel processing	NLP (BERT, GPT), multimodal AI	2017
Graph Neural Networks (GNN)	Message passing between graph nodes	Social networks, chemistry, recommender systems	2017
Spiking Neural Networks (SNN)	Discrete spikes, event-based computation	Neuromorphic hardware, low-power AI	2000s
KAN (Kolmogorov-Arnold Networks)	Learnable splines as activation functions	Scientific ML, interpretable models	2024
Neural ODEs	Continuous-time dynamics, numerical integration	Time series, generative models, physics	2018
Diffusion Models	Reverse denoising process	High-quality image generation	2020
Mixture of Experts (MoE)	Sparse activation, adaptive routing	Large language models (e.g., Switch Transformer)	2021
Physics-Informed Neural Networks (PINN)	Physical constraints in loss function	Scientific computing, engineering simulation	2019

