

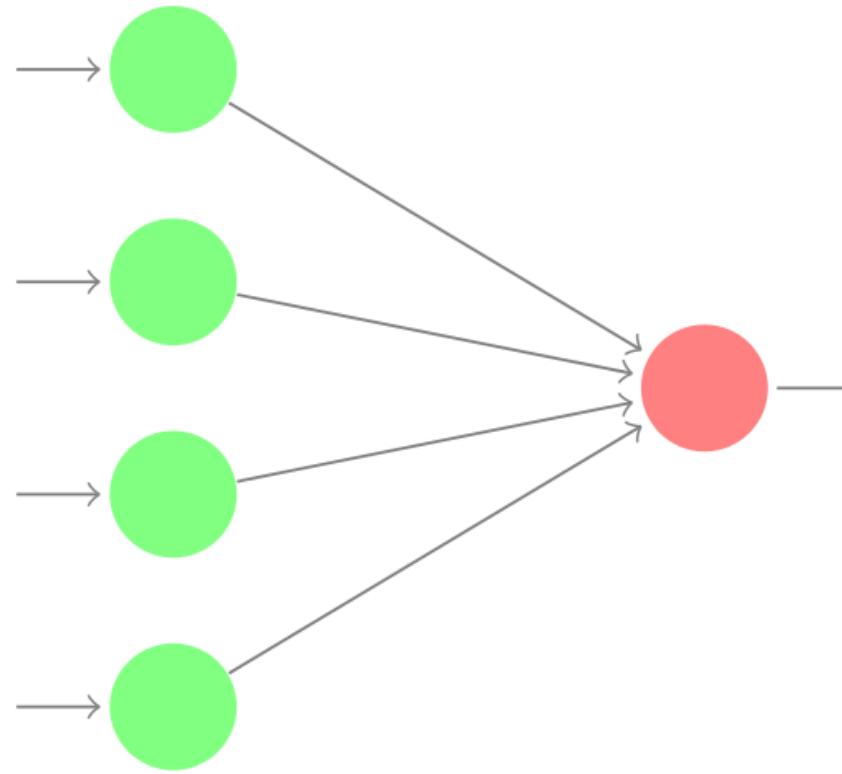
Машинное обучение

Лекция 15. Нейронные сети



От линейных моделей к нейросетям

Линейная модель



Линейная модель

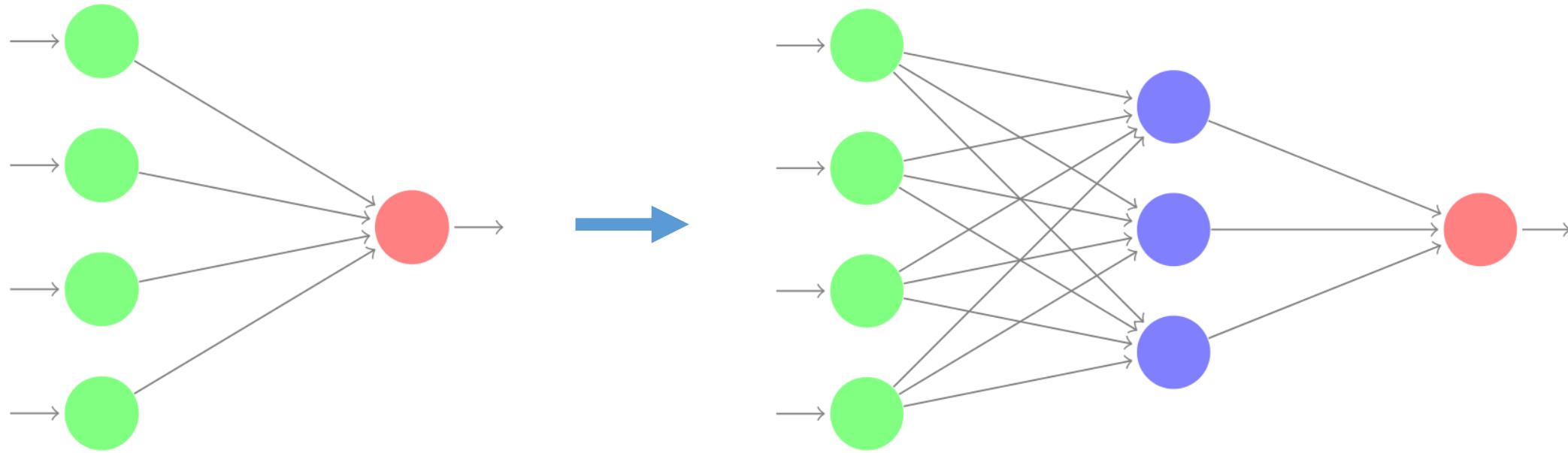
1. Быстро обучается
2. Работает с большим числом признаков
3. Выпуклый функционал
4. Интерпретируемые коэффициенты
5. Хорошо работает с линейными зависимостями

Нелинейный зависимости

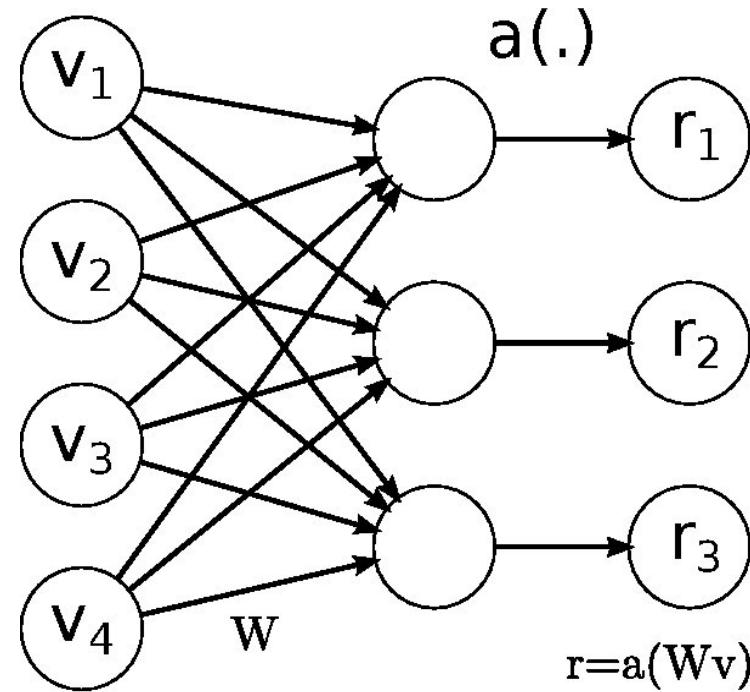
Нужно придумать признаки, для которых зависимость линейная

1. Полиномиальные
2. Любые нелинейные функции
3. Переход в спрямляющее пространство

Автоматическая генерация признаков



Полносвязная нейросеть

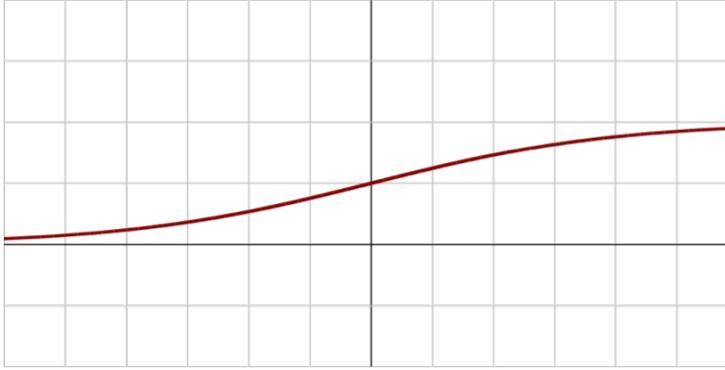


Полносвязная нейросеть

1. К входным признакам применяется линейное преобразование (преактивации)
2. Новые признаки – результат применения нелинейного преобразования к преактивациям
3. Новые признаки подаются в линейную модель
4. Параметры признаков и линейной модели учатся вместе

Функции активации

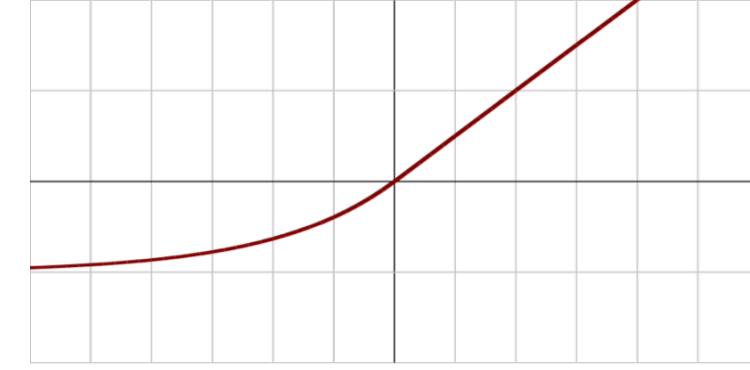
Sigmoid



ReLU



ELU



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

$$f(x) = \max(x, 0)$$

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

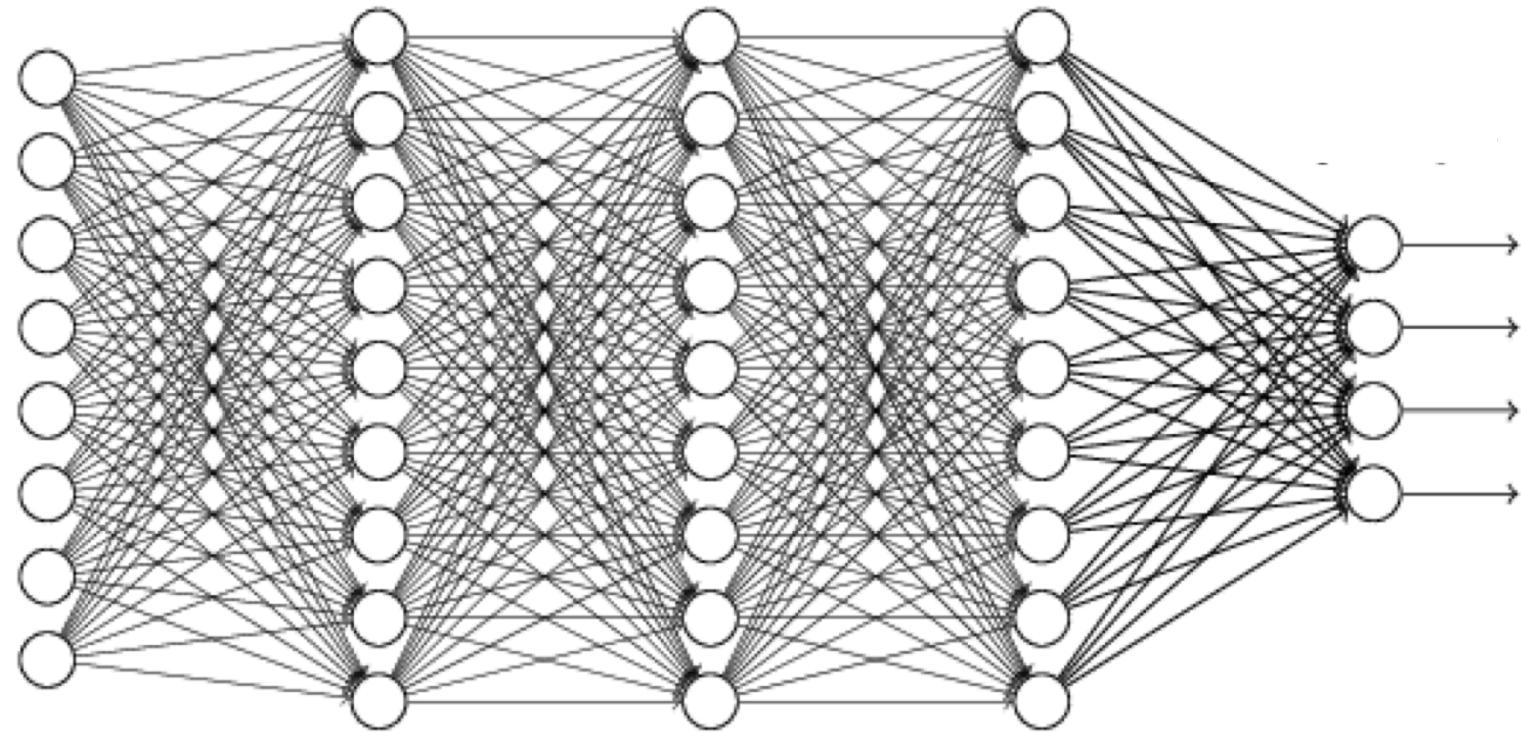
Полносвязная нейросеть

Универсальная теорема аппроксимации

Нейронная сеть с одним скрытым слоем может аппроксимировать любую непрерывную функцию многих переменных с любой точностью*

*При достаточном числе нейронов на скрытом слое и с правильными параметрами.

Полносвязная нейросеть на практике



Обучение нейросетей

Обратное распространение ошибки

$$nn_{w_1, w_2}(x) = nn(x) = f_{w_2}(f_{w_1}(x))$$

$$L(X, y, nn) = \frac{1}{N} \sum_i l(nn(x_i), y_i)$$

$$\frac{\partial}{\partial w_1} l, \frac{\partial}{\partial w_2} l - ?$$

Обратное распространение ошибки

$$nn_{w_1, w_2}(x) = nn(x) = f_{w_2}(f_{w_1}(x))$$

$$L(X, y, nn) = \frac{1}{N} \sum_i l(nn(x_i), y_i)$$

$$\frac{\partial}{\partial w_2} l(nn(x), y) = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial w_2} \Big|_x = \frac{\partial l}{\partial y'} \cdot \frac{\partial f_{w_2}}{\partial w_2} \Big|_{f_{w_1}(x)}$$

Обратное распространение ошибки

$$nn_{w_1, w_2}(x) = nn(x) = f_{w_2}(f_{w_1}(x))$$

$$L(X, y, nn) = \frac{1}{N} \sum_i l(nn(x_i), y_i)$$

$$\frac{\partial}{\partial w_2} l(nn(x), y) = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial w_2} \Big|_x = \frac{\partial l}{\partial y'} \cdot \frac{\partial f_{w_2}}{\partial w_2} \Big|_{f_{w_1}(x)}$$

$$\frac{\partial}{\partial w_1} l(nn(x), y) = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial w_1} = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial f_{w_1}} \cdot \frac{\partial f_{w_1}}{\partial w_1}$$

Обратное распространение ошибки

Прямой проход

1. Подаем на вход объект
2. Проходим по сети с начала до конца, запоминаем выходы слоев
3. Считаем функцию потерь

Обратный проход

1. Считаем градиент функции потерь
2. Идем с конца, поддерживаем градиент функции потерь по выходам текущего слоя
3. Пересчитываем градиент функции потерь по параметрам текущего слоя

Обратное распространение ошибки

Обратный проход

1. Считаем градиент функции потерь
2. Идем с конца, поддерживаем градиент функции потерь по выходам текущего слоя
3. Пересчитываем градиент функции потерь по параметрам текущего слоя

$$\frac{\partial}{\partial w_2} l(nn(x), y) = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial w_2} \Big|_x = \frac{\partial l}{\partial y'} \cdot \frac{\partial f_{w_2}}{\partial w_2} \Big|_{f_{w_1}(x)}$$

$$\frac{\partial}{\partial w_1} l(nn(x), y) = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial w_1} = \frac{\partial l}{\partial y'} \cdot \frac{\partial nn}{\partial f_{w_1}} \cdot \frac{\partial f_{w_1}}{\partial w_1}$$

Градиентный спуск

$$L(X, y, nn) = \frac{1}{N} \sum_i l(nn(x_i), y_i)$$

- Градиентный спуск

$$W^t = W^{(t-1)} - \eta \nabla L|_{W=W^{(t-1)}}$$

- Стохастический градиентный спуск

$$W^t = W^{(t-1)} - \eta \nabla \frac{1}{B} \sum_{i \in \mathcal{B}} l(nn(x_i), y_i) |_{W=W^{(t-1)}}$$

Tradeoff: скорость одной итерации vs шумность градиентов

Momentum vs NAG

$$L(X, y, nn) = \frac{1}{B} \sum_{i \in \mathcal{B}} l(nn(x_i), y_i)$$

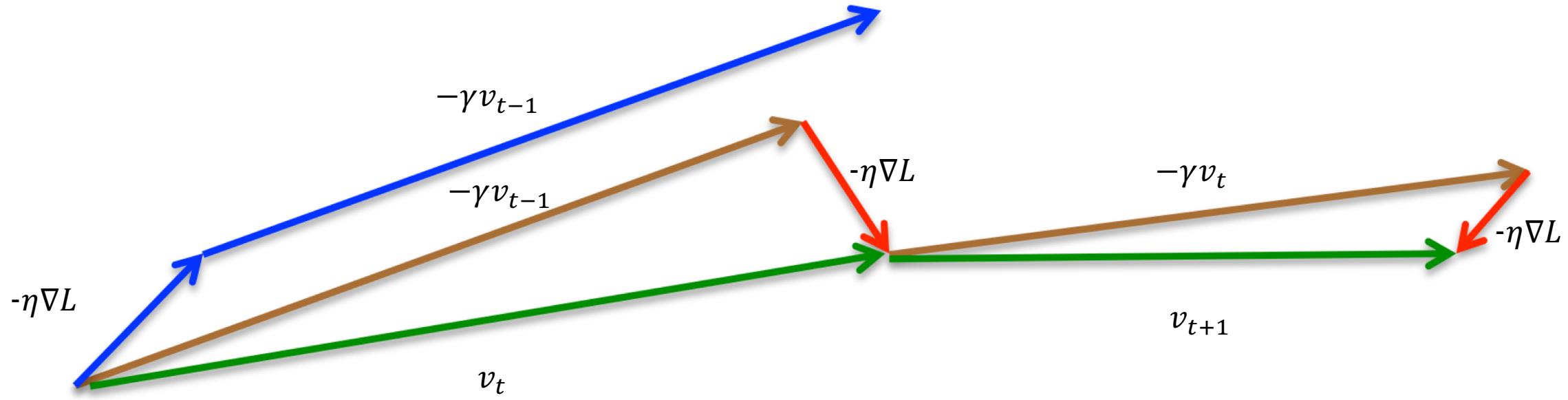
- Momentum

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla L \Big|_{W=W^{(t-1)}} \\ W^t &= W^{t-1} - v_t \end{aligned}$$

- Nesterov accelerated gradient

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla L \Big|_{W=W^{(t-1)} - \gamma v_{t-1}} \\ W^t &= W^{t-1} - v_t \end{aligned}$$

Momentum vs NAG



Adaptive learning rate

- AdaGrad

$$\begin{aligned} g_t &= \nabla L \Big|_{W=W^{(t-1)}} \\ G_t &= G_{t-1} + g_t^2 \\ W^t &= W^{(t-1)} - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t \end{aligned}$$

- RMSprop

$$G_t = \gamma G_t + (1 - \gamma) g_t^2$$

Adaptive learning rate

- AdaDelta

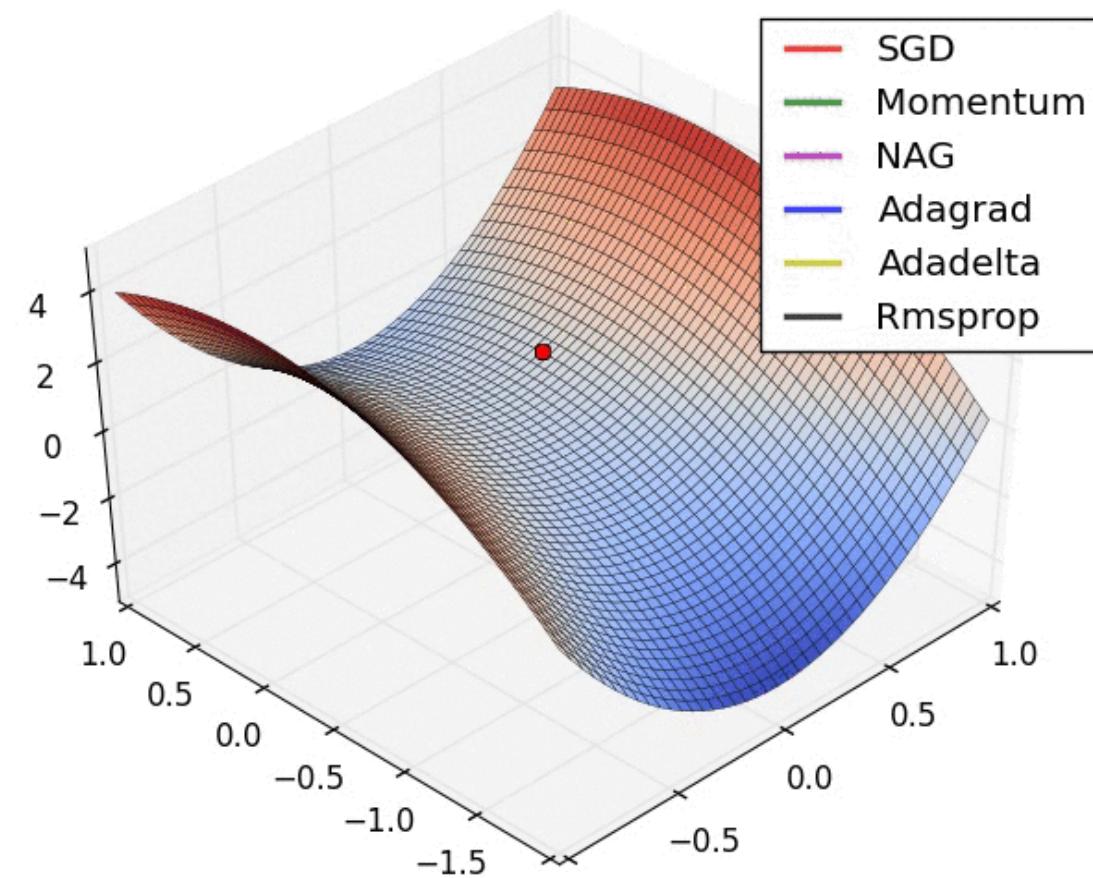
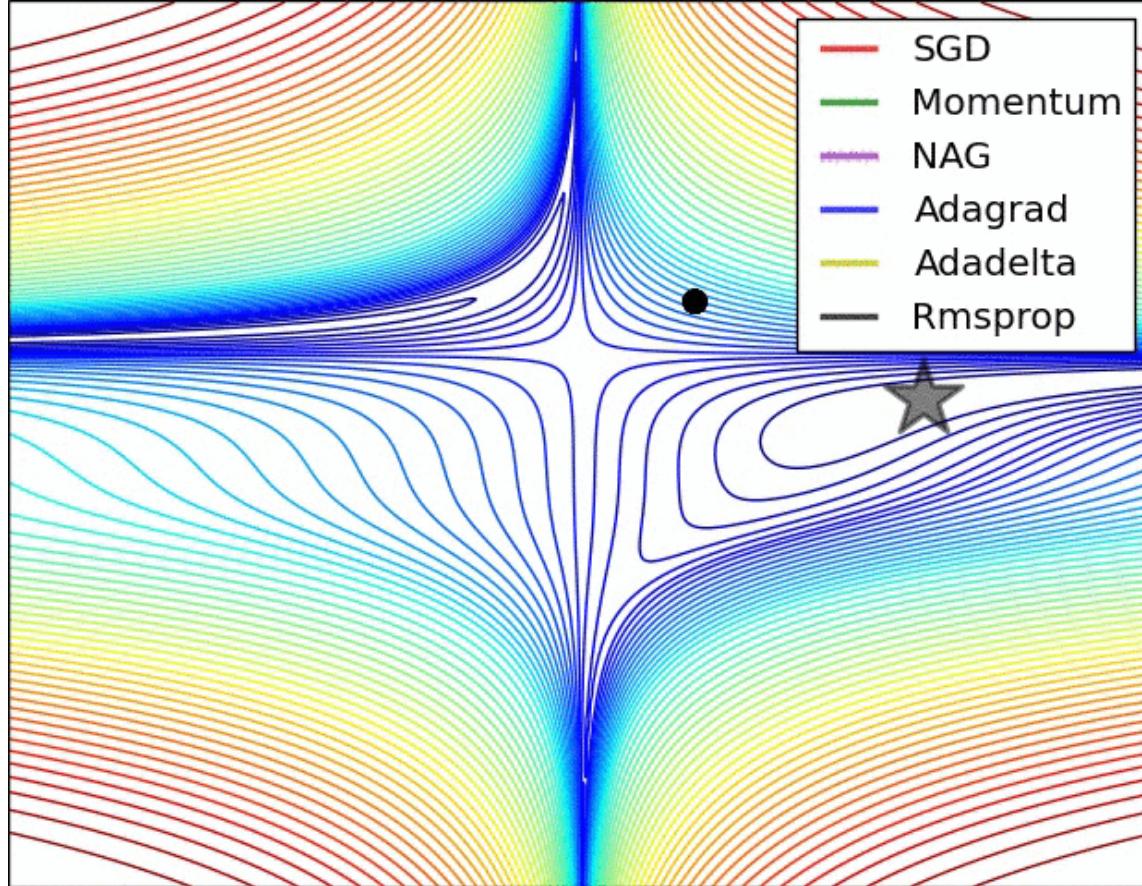
$$\begin{aligned}\Delta W_t &= -\frac{\sqrt{F_{t-1} + \epsilon}}{\sqrt{G_t + \epsilon}} g_t \\ W^t &= W^{(t-1)} + \Delta W_t \\ F_t &= \gamma F_{t-1} + (1 - \gamma) \Delta W_t^2\end{aligned}$$

Adaptive learning rate

- Adam

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\\hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \\W^{(t)} &= W^{(t-1)} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} m_t\end{aligned}$$

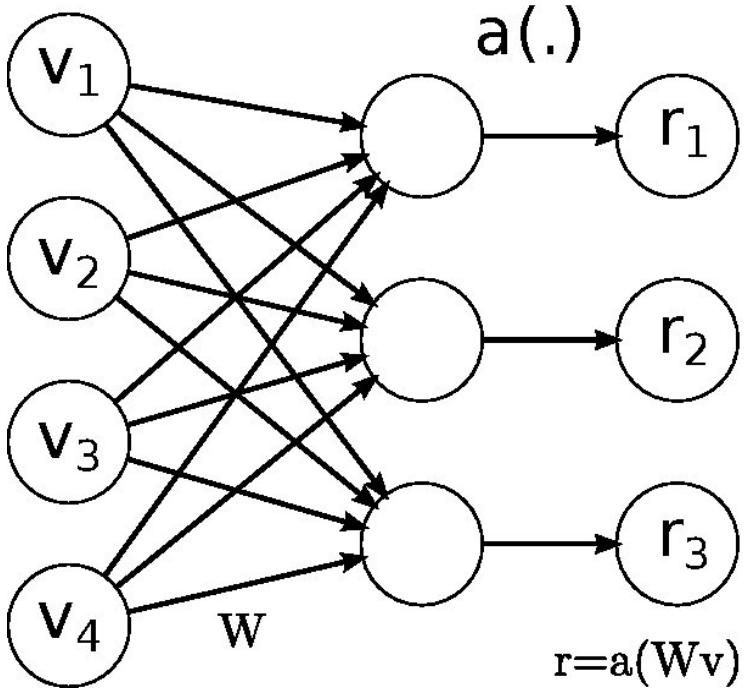
Сравнение методов



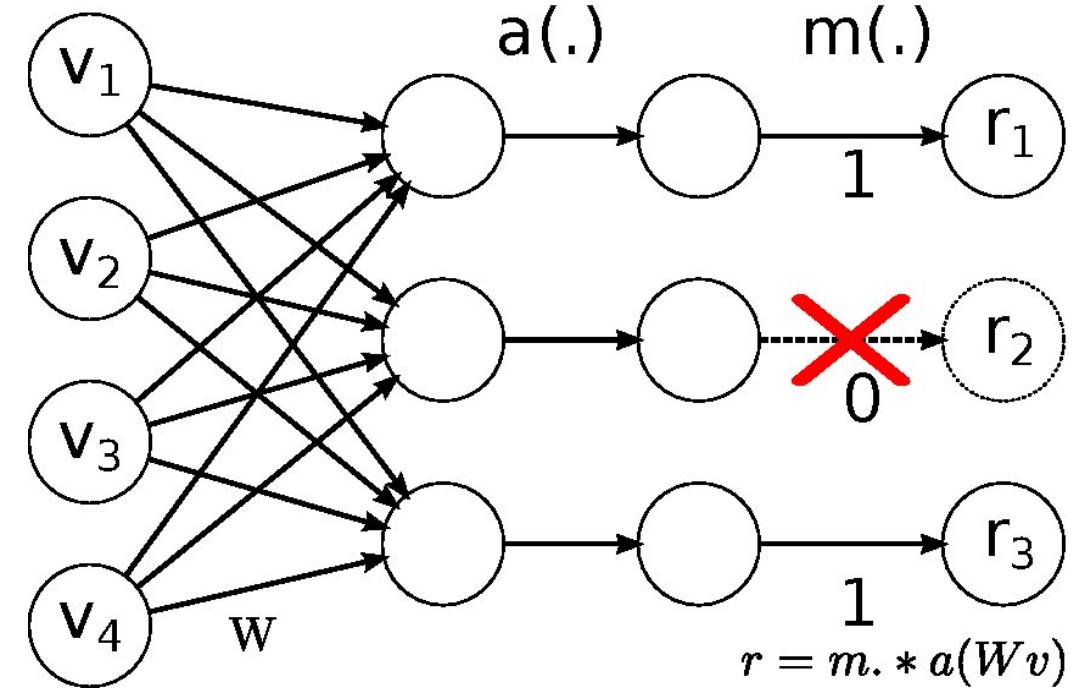
Регуляризация

Dropout

Обычная сеть



Dropout



Dropout

1. Во время обучения независимо зануляем активации с вероятностью p
2. Во время тестирования домножаем активации на $1 - p$

$$\mathbb{E}r_1 = (1 - p)a_1 + p \cdot 0$$

Dropout

1. Во время обучения независимо зануляем активации с вероятностью p
2. Во время тестирования домножаем активации на $1 - p$

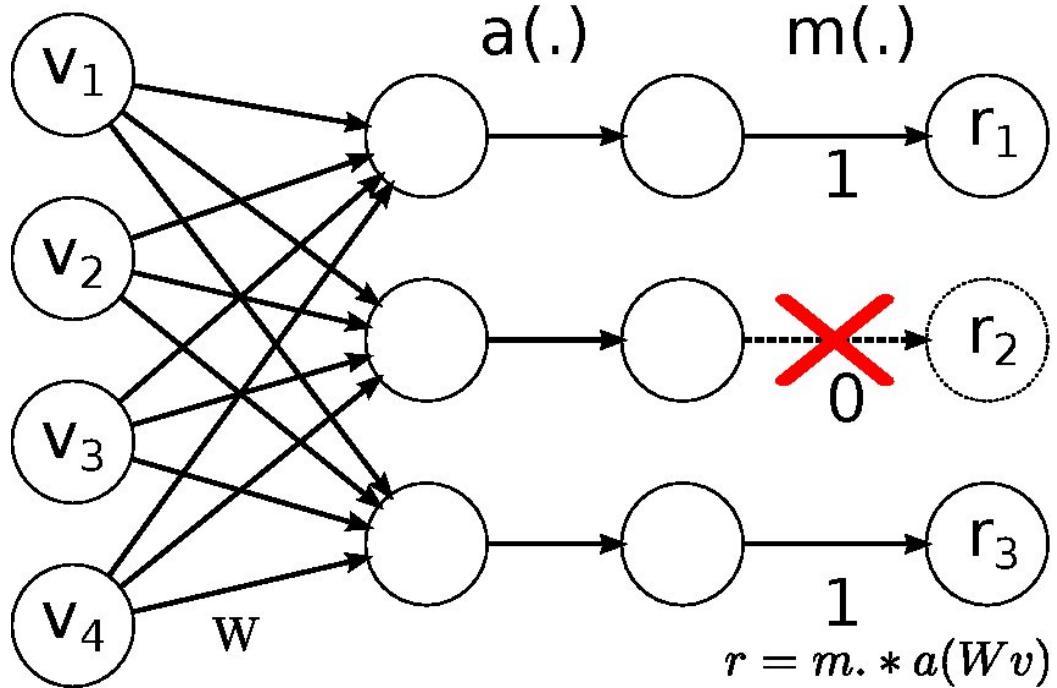
$$\mathbb{E}r_1 = (1 - p)a_1 + p \cdot 0$$

Inverted Dropout

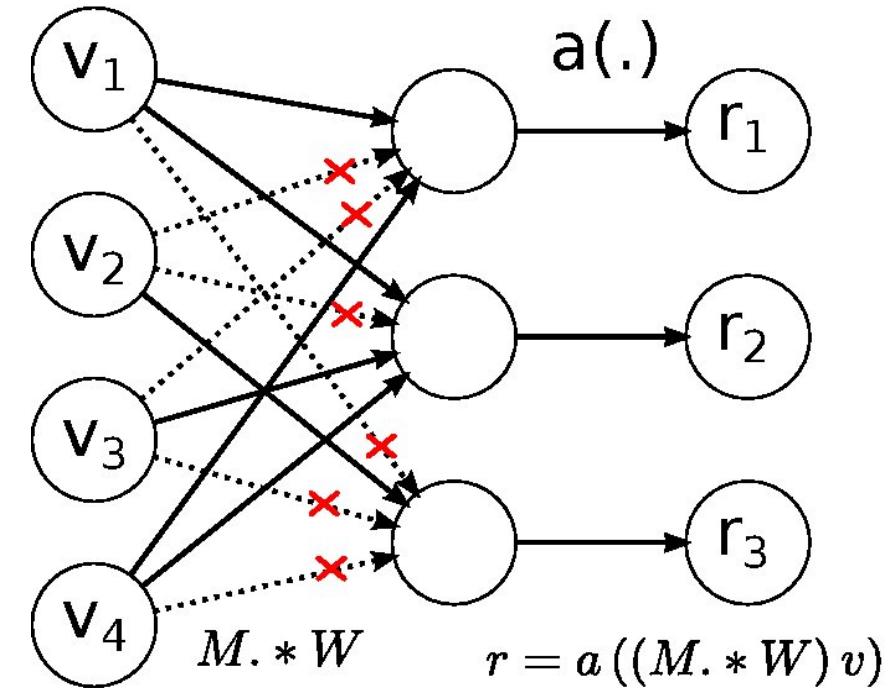
1. Во время обучения независимо зануляем активации с вероятностью p и делим их на $1 - p$

DropConnect

Dropout

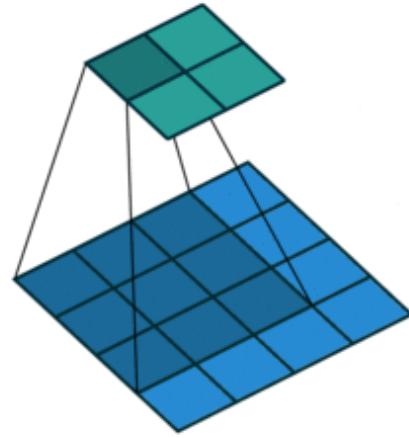


DropConnect

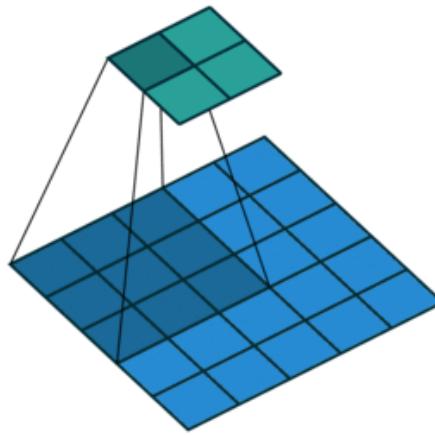


Работа с изображениями

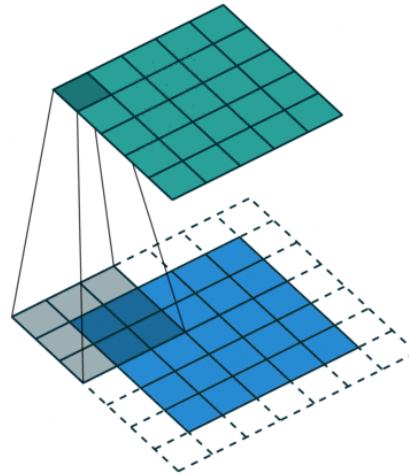
Свертки



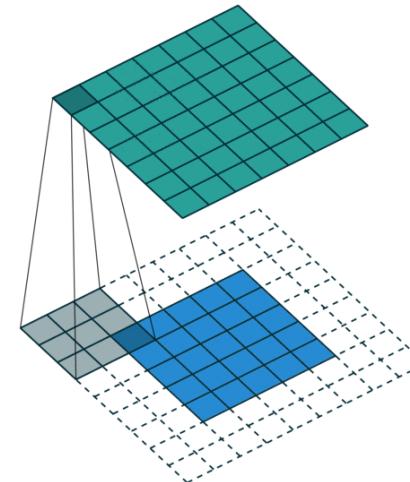
No padding, no strides



No padding, strides



Half padding, no strides



Full padding, no strides

Пулинг

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4



6	8
3	4

Пулинг

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

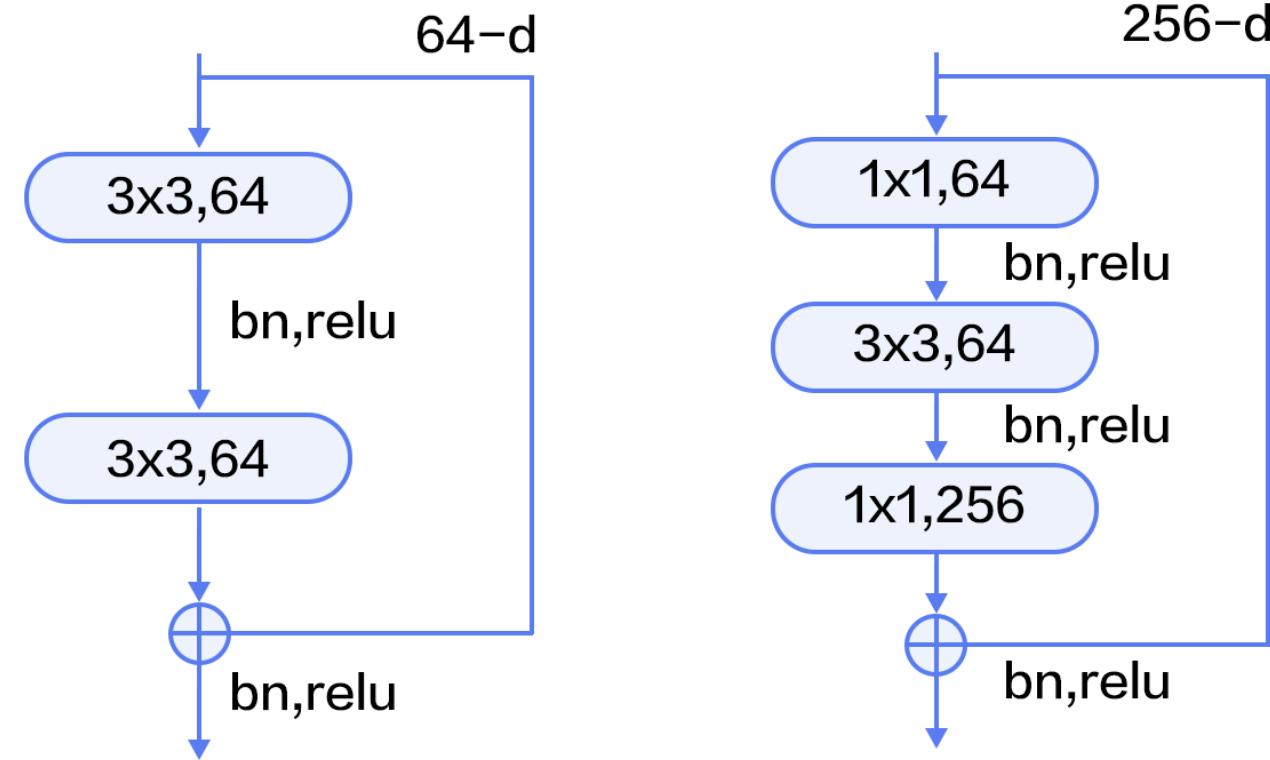


6	8
3	4

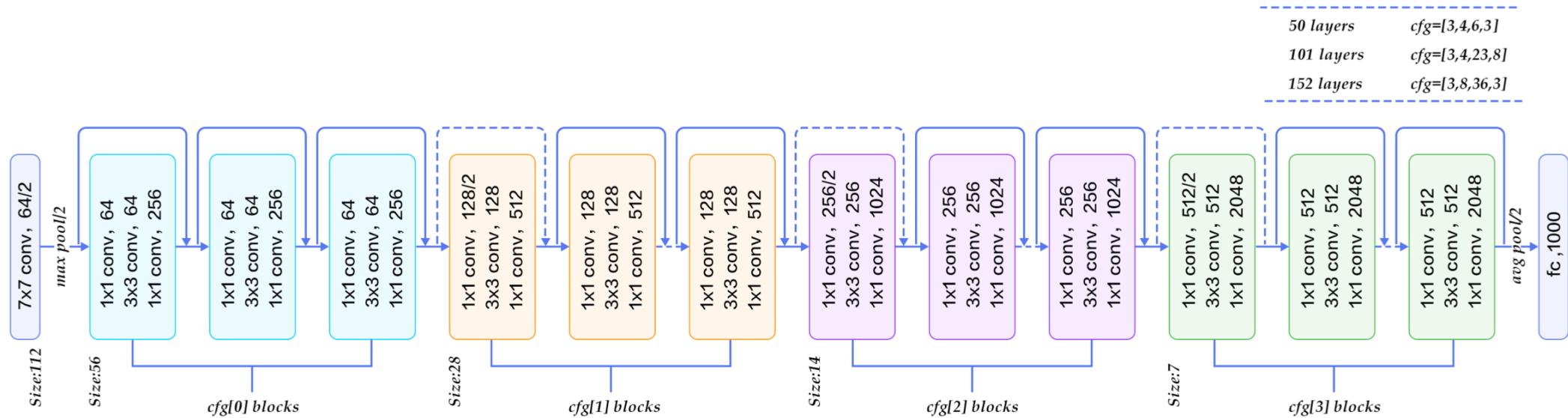
Резюме



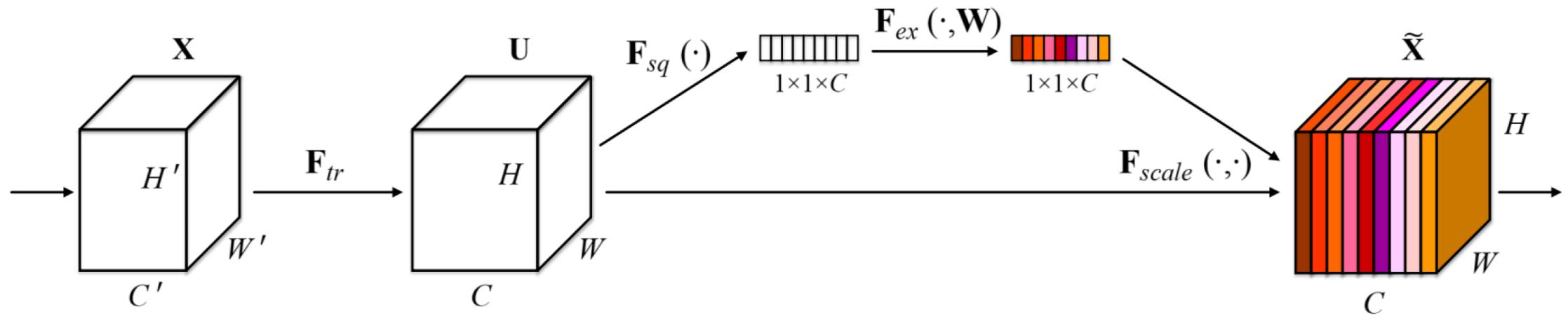
ResNet



ResNet



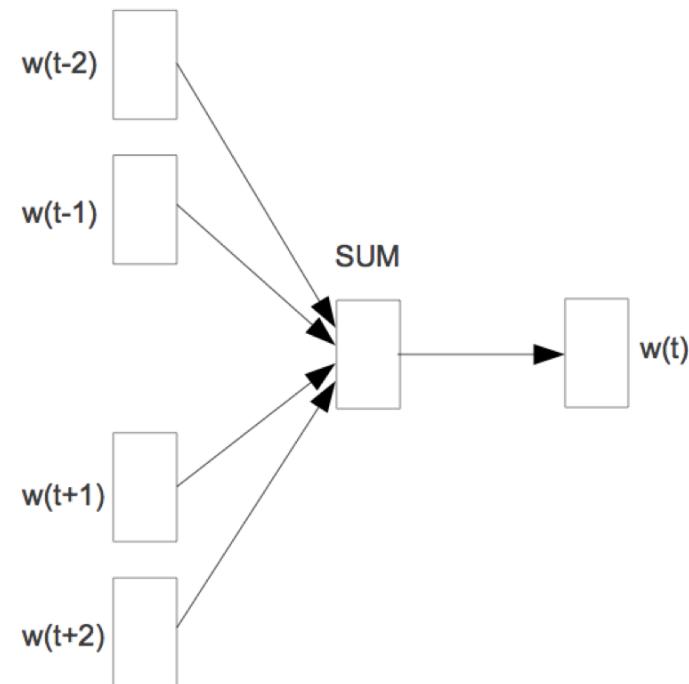
SeNet



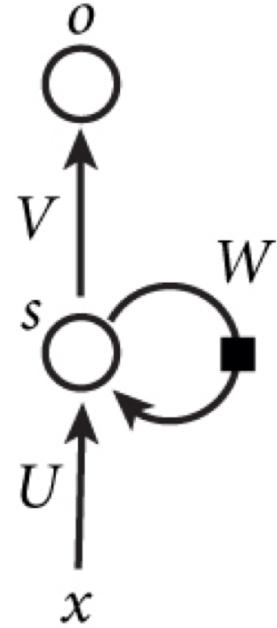
Работа с текстами

Признаки из текстов

1. One-hot / Tf-Idf – фиксированный размер
2. Эмбеддинги – каждому слову вектор фиксированного размера

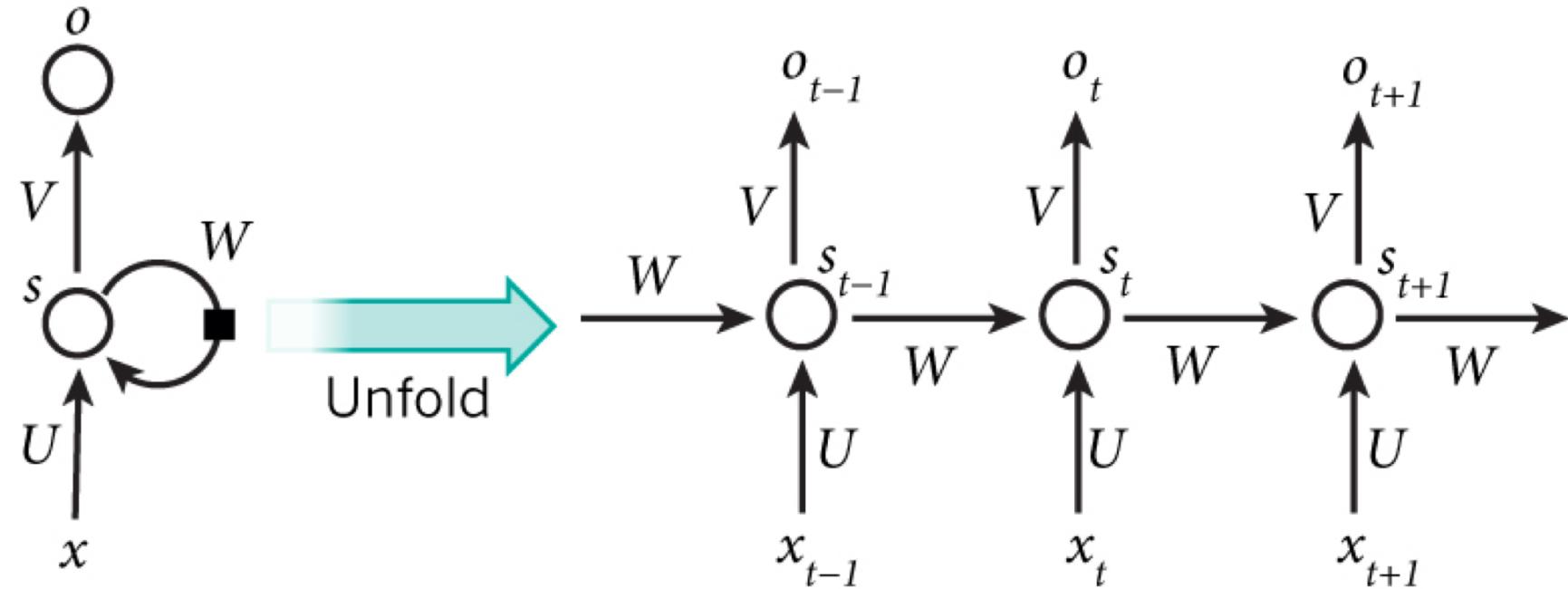


Рекуррентные сети

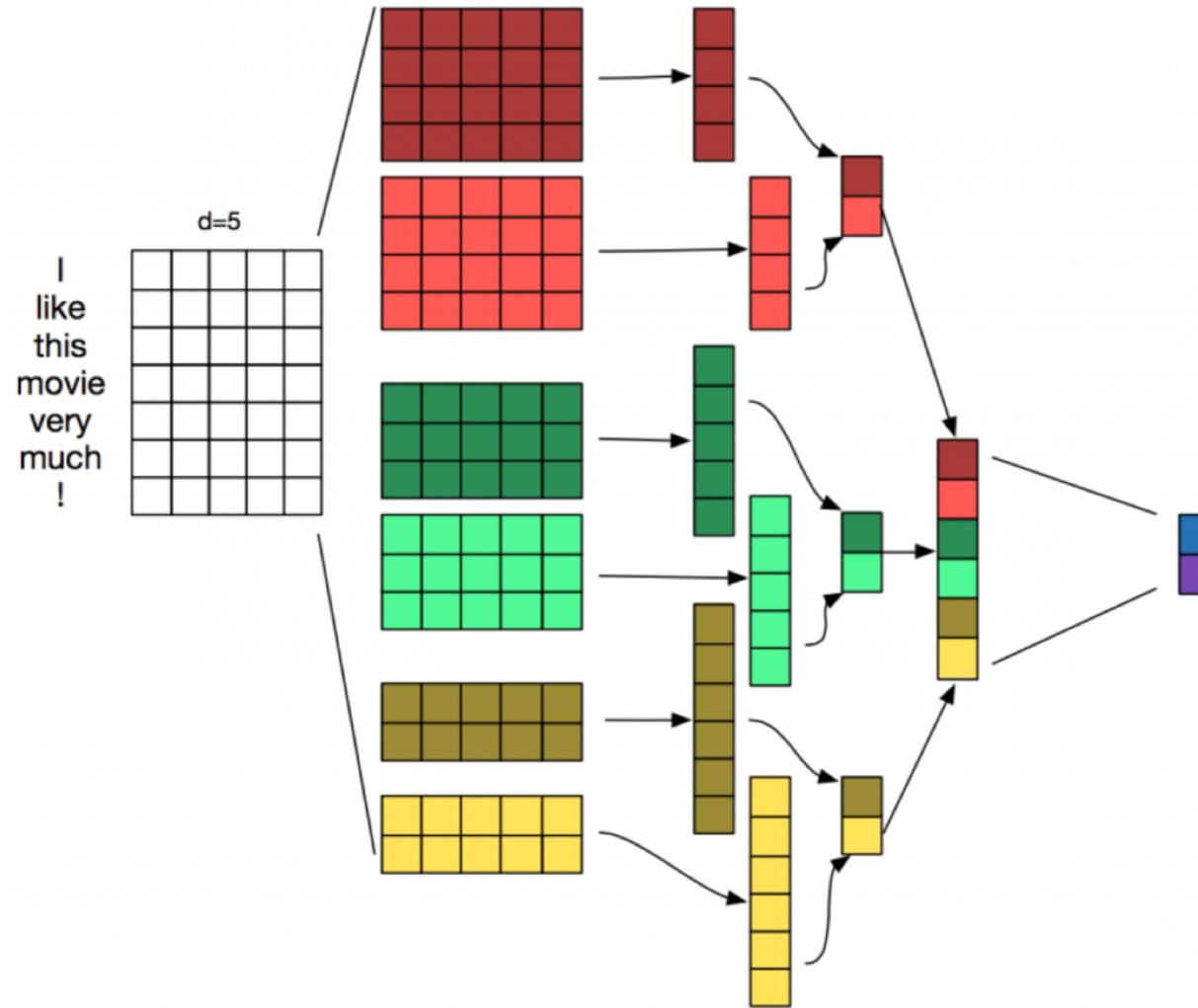


$$s_t = f(Ux_t + Ws_{t-1})$$
$$o_t = g(Vs_t)$$

Рекуррентные сети



Сверточные сети



Обзор DL фреймворков

Основные задачи

1. Построение и работа с вычислительными графами
2. Автоматическое дифференцирование
3. Оптимизация работы на CPU / GPU

Фреймворки

Статический граф

- Theano (deprecated) & Lasagne
- Tensorflow
- Caffe -> Caffe2
- CNTK

Динамический граф

- Torch -> PyTorch
- Tensorflow (eager, $\geq 1.5.0$)

Keras – высокоуровневая обертка для Theano, Tensorflow, CNTK