Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени первого Президента России Б. Н. Ельцина»

УТВЕРЖДАЮ

Директор по образовательной деятельности

_____ С.Т. Князев

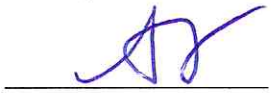
«_7_» _сентября_ 2023 г.

Программирование глубоких нейронных сетей

Учебно-методические материалы по направлению подготовки
09.03.03 Прикладная информатика
Образовательная программа «Прикладной искусственный интеллект»

Екатеринбург

РАЗРАБОТЧИКИ УЧЕБНО-МЕТОДИЧЕСКИХ МАТЕРИАЛОВ

Доцент кафедры информационных технологий и систем управления, канд.техн.наук		А.В. Созыкин
Доцент кафедры информационных технологий и систем управления, канд.техн.наук		М.В. Ронкин

СОДЕРЖАНИЕ

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.01
В которой мы начнем обсуждать что делать с картинками

Докладчик
Долганов Антон

В предыдущей лекции

- **Поиск Аномалий**
- Выбросы и Новинки
- Кластеризация?
- Метод главных компонент / кодировщики?
- **Статистический подход**
- Рисуем Эллипс
- **Метрический подход**
- Считаем плотность у соседей
- **Одноклассовый метод опорных векторов**
- SVM+Kernel Trick, но для отделения 0 от выборки
- **Случайные Проекции**
- Садим случайный лес, но не для классификации

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Про изображения

Про нейронки

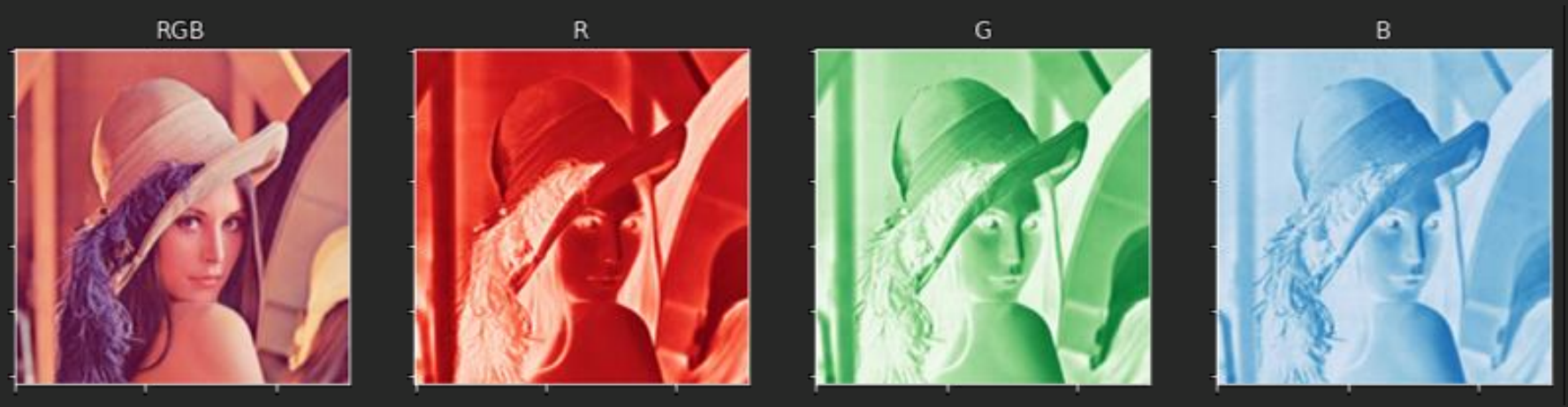
Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Изображение

$$\boldsymbol{X} \in \mathbb{R}^{m \times n \times c}$$



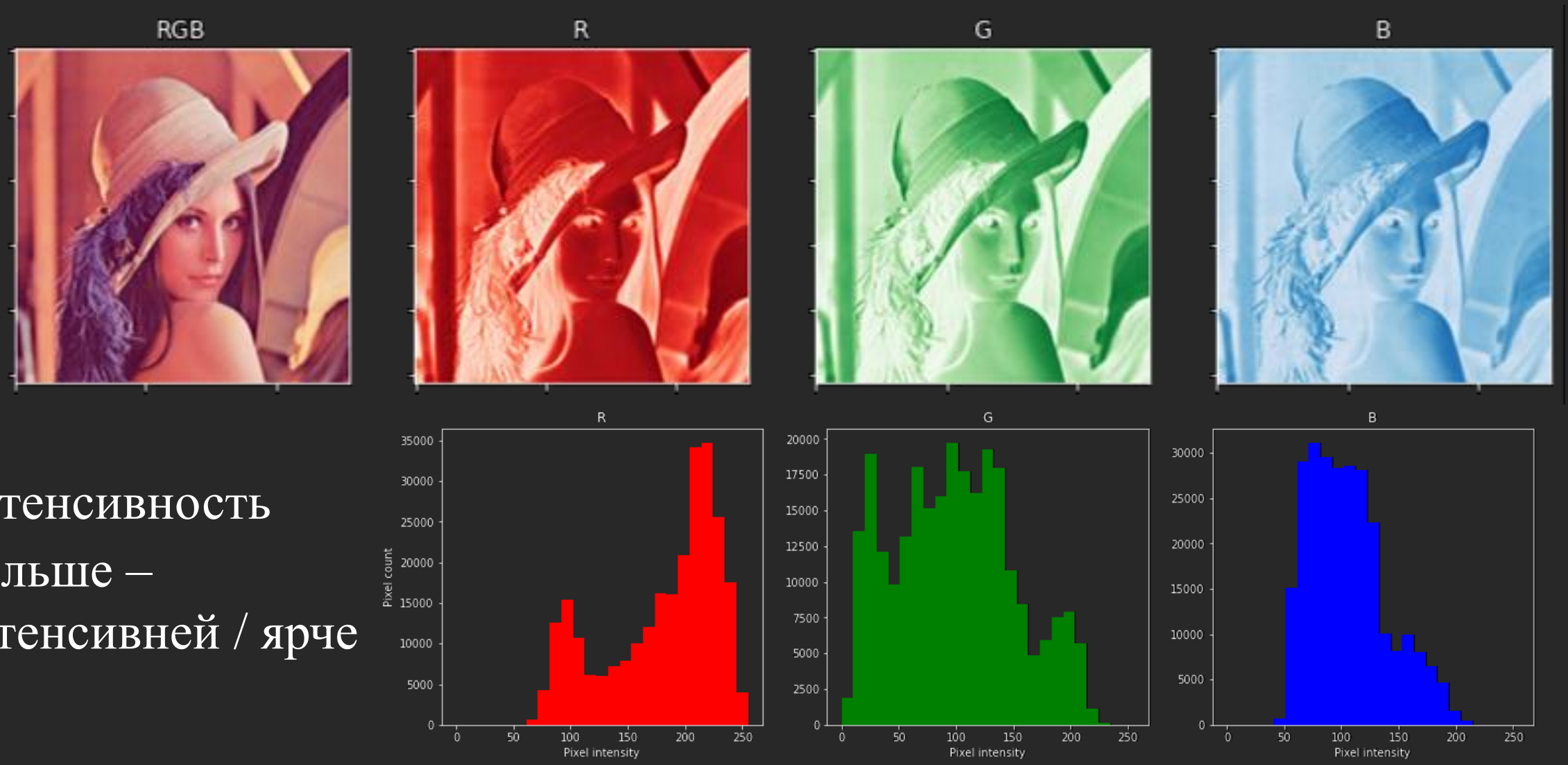
m – высота изображения,
n – ширина изображения
c – количество каналов (как правило 3)

$x_{ijk} \in \mathbb{N}$: **[0**, 1, … $2^n - 1$**]** n – битность/разрядность

n =8 **[0**, 1, … , 255**]** n =16 **[0**, 1, … ,65535**]**

$x_{ijk} \in \mathbb{R}$: **[0**, … , 1**]**

Яркость



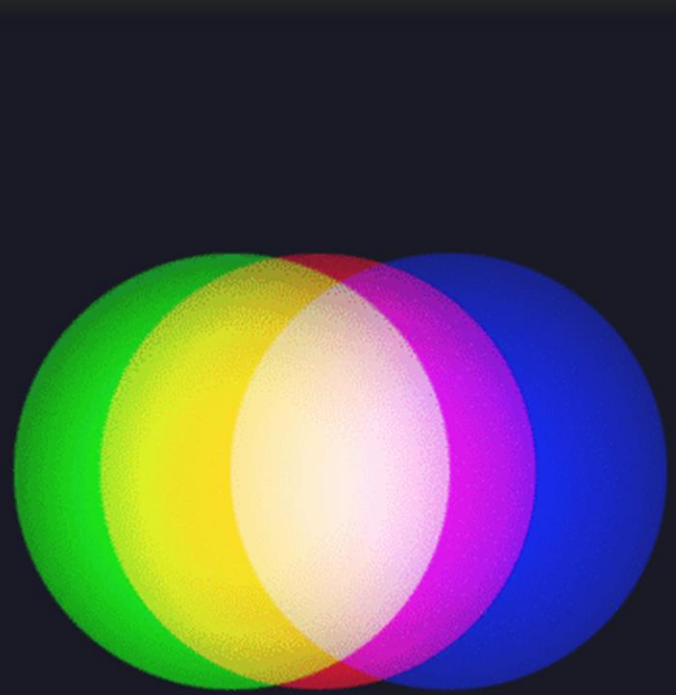
x_{ijk} интенсивность

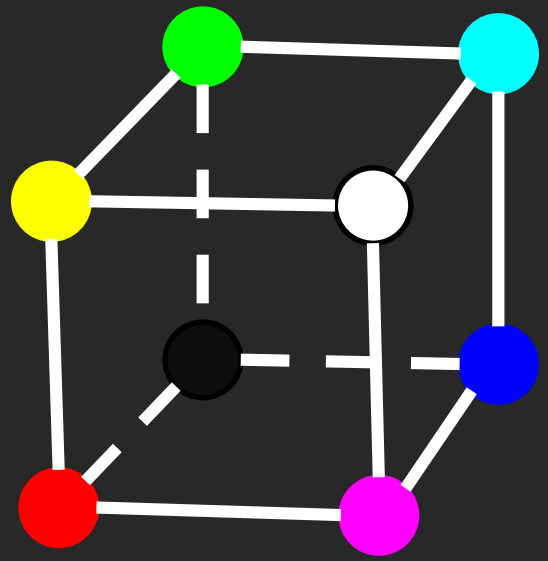
Чем больше –

тем интенсивней / ярче

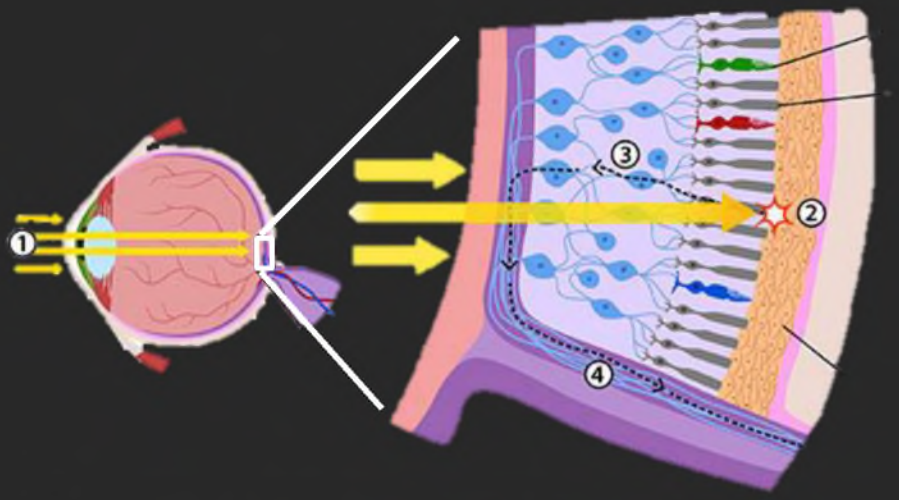
Цветовые модели RGB модель

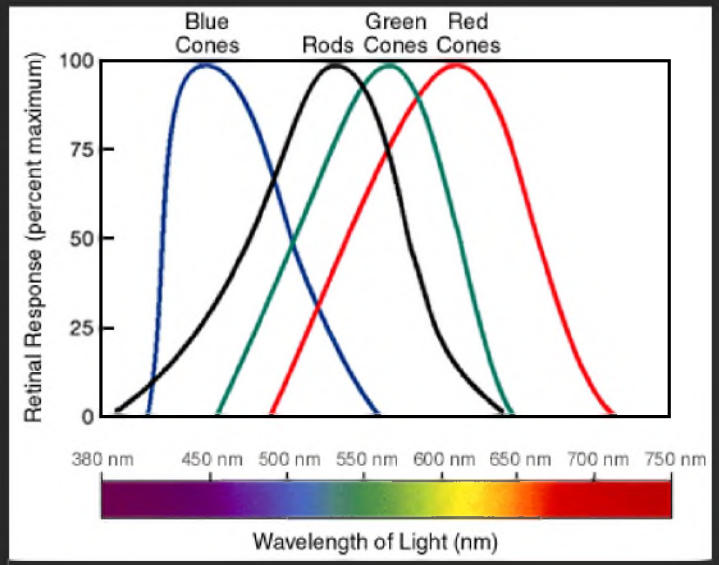
Аддитивная

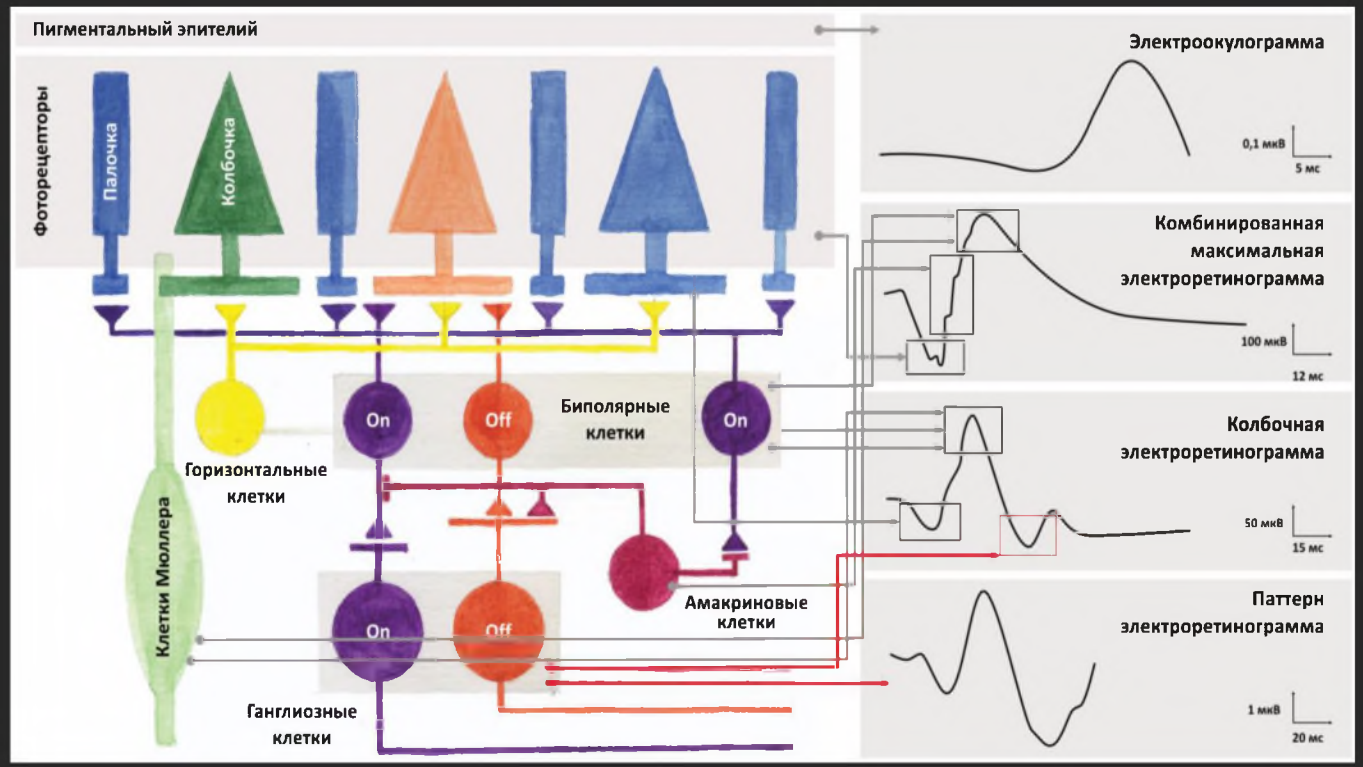




Зрение у людей: Колбочки и Палочки

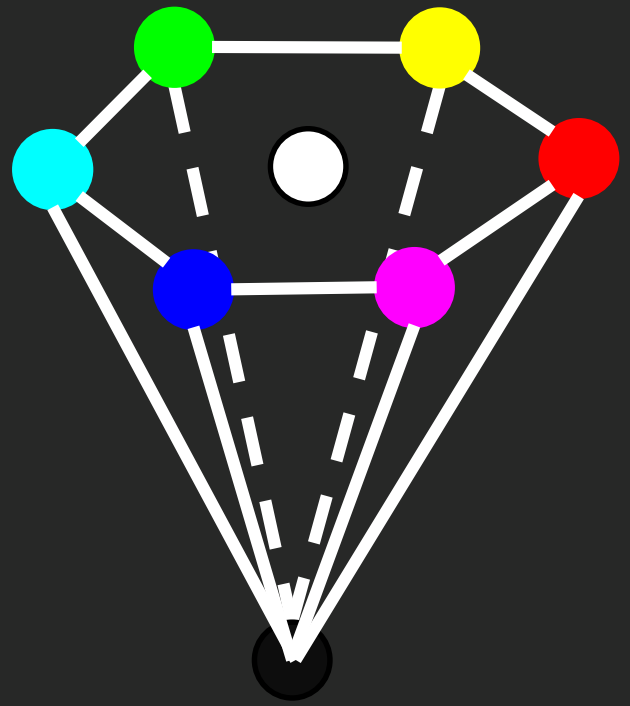


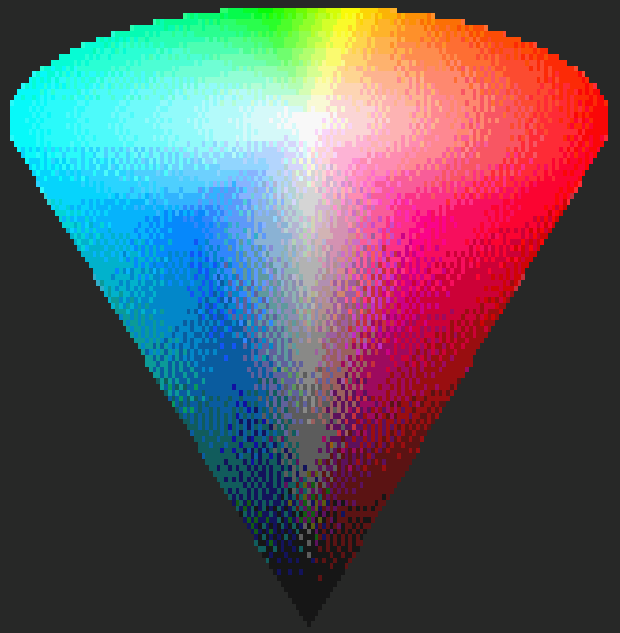




Цветовые модели HSL и HSV модели

Вспомогательная

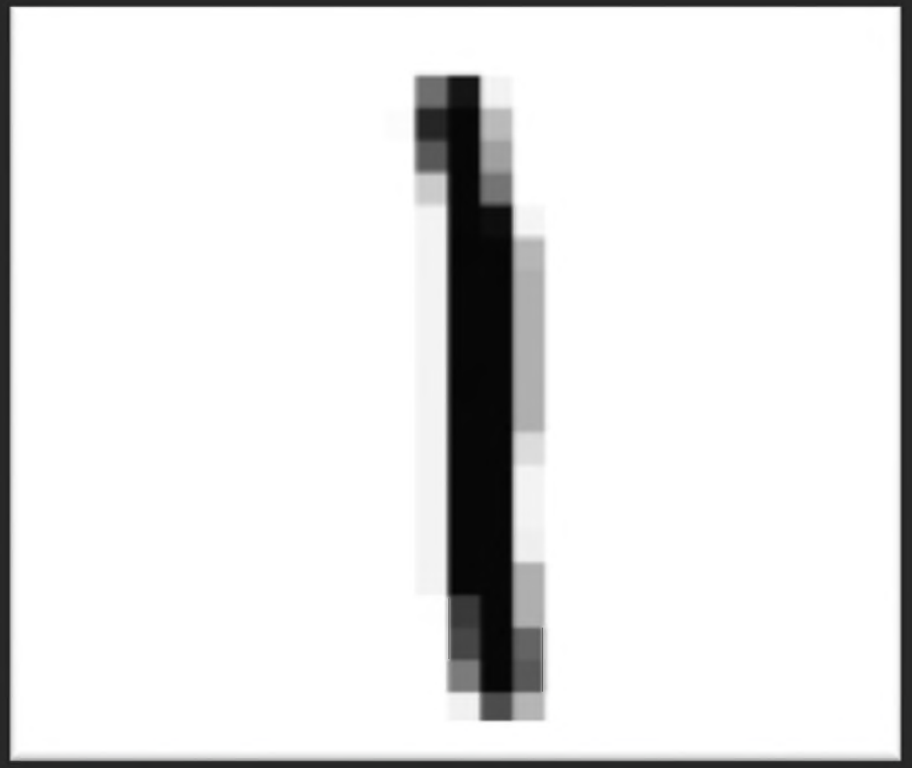




- **Hue** - Тон
- **Saturation-** Насыщенность
- **Intensity** (lightness) - Светлота / **Value** (Brightness) - Яркость

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Изображение -> Таблички?



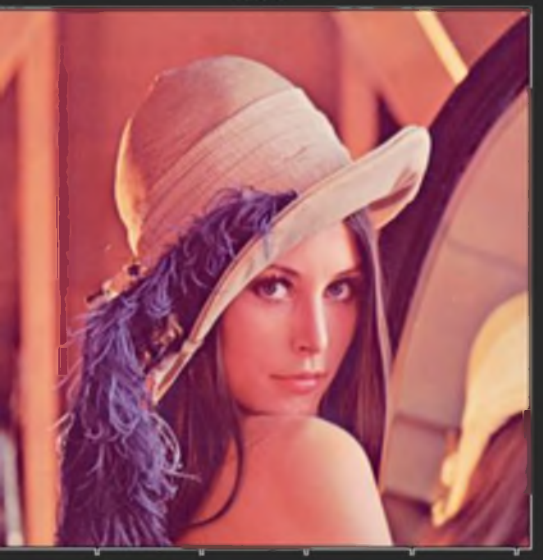
241	251	248	244	252	247	254	254	241	240
244	249	252	246	9	2	240	249	251	252
241	244	254	251	250	8	244	250	254	241
246	251	250	243	249	14	245	246	254	252
244	251	246	244	240	1	244	242	241	249
247	241	244	253	244	15	250	248	249	245
245	252	250	245	248	8	249	250	243	247
254	241	240	249	254	13	247	240	248	242
245	247	241	248	248	1	247	249	252	245
243	244	251	242	248	247	248	251	249	246

241	251	248	244	252	247	254	254	241	240	244	249	252	246	9	2	240	249	251	252

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Изображение -> Таблички?

RGB



R

195	157	250	229	56	205	110	177	165	122
226	193	97	67	179	192	245	56	114	242
131	193	128	168	214	235	130	254	109	252
93	71	122	109	135	199	93	217	60	94
192	147	147	159	180	164	71	97	80	171
220	246	129	196	96	66	58	60	66	139
68	252	103	83	223	77	151	243	96	240
108	190	71	113	104	85	200	111	126	223
171	57	179	106	191	244	142	238	202	179
173	117	235	133	197	86	218	136	207	82

G

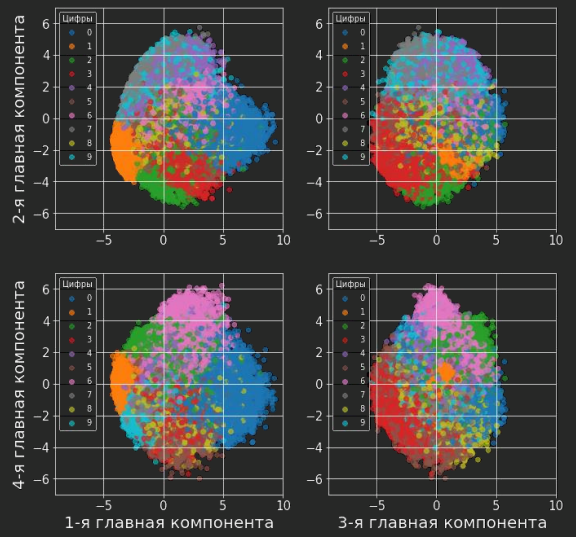
226	226	244	253	191	100	120	218	207	160
187	136	238	248	199	232	78	194	199	64
118	127	175	160	55	186	142	205	115	99
109	166	55	94	216	153	134	111	106	200
61	200	120	70	94	107	127	104	79	139
194	211	164	71	78	222	165	201	137	88
164	207	212	61	188	95	227	62	98	206
232	61	234	80	104	188	190	175	121	86
82	184	85	196	198	185	151	231	156	166
164	126	120	220	121	210	61	226	93	113

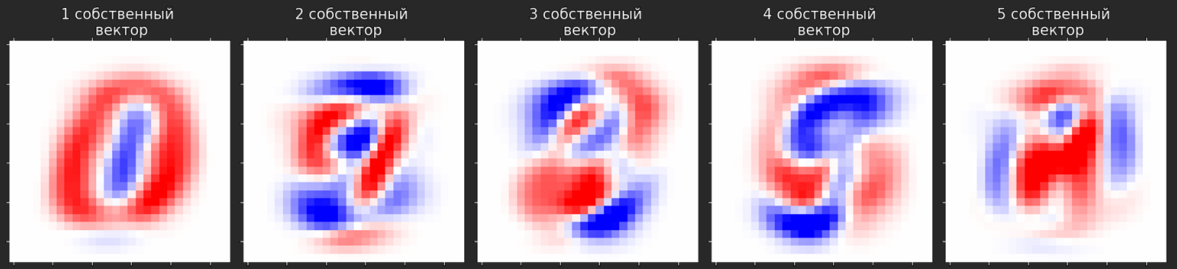
B

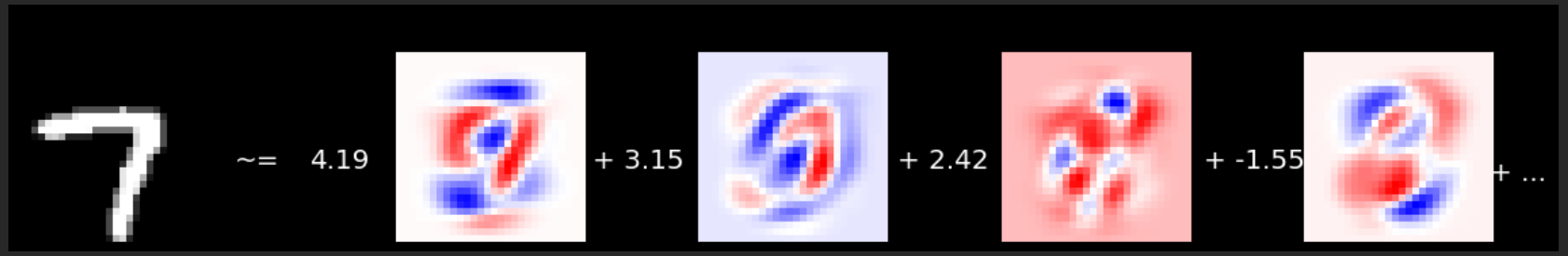
214	131	251	194	84	98	241	248	80	169
167	78	83	97	218	153	122	57	67	163
161	193	82	108	219	98	156	75	166	61
225	83	199	149	117	170	108	60	182	194
130	199	196	118	234	175	211	97	164	125
130	238	214	137	128	154	177	59	62	237
90	55	150	245	98	78	249	177	206	172
251	99	242	236	58	76	107	169	171	104
170	160	134	132	117	91	146	252	163	155
232	71	191	145	116	165	107	206	183	221

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Набор MNIST и Метод Главных Компонент

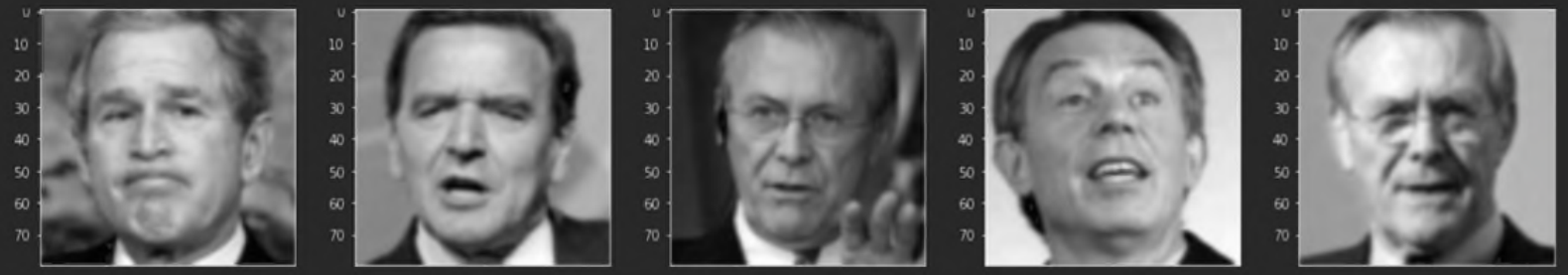


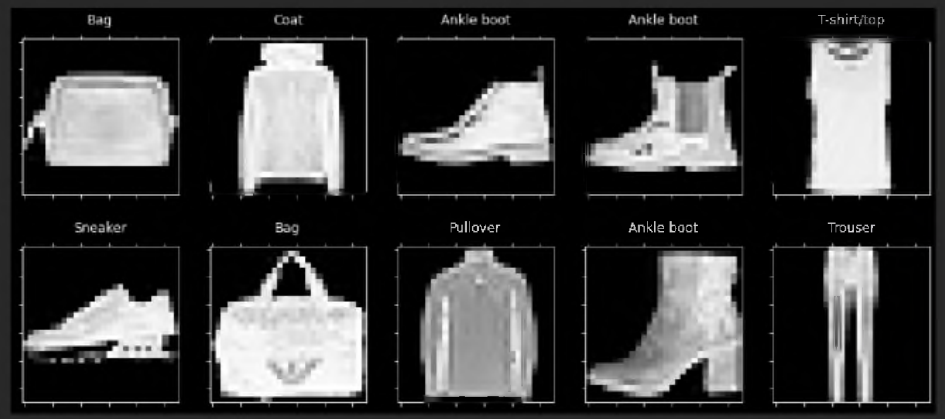


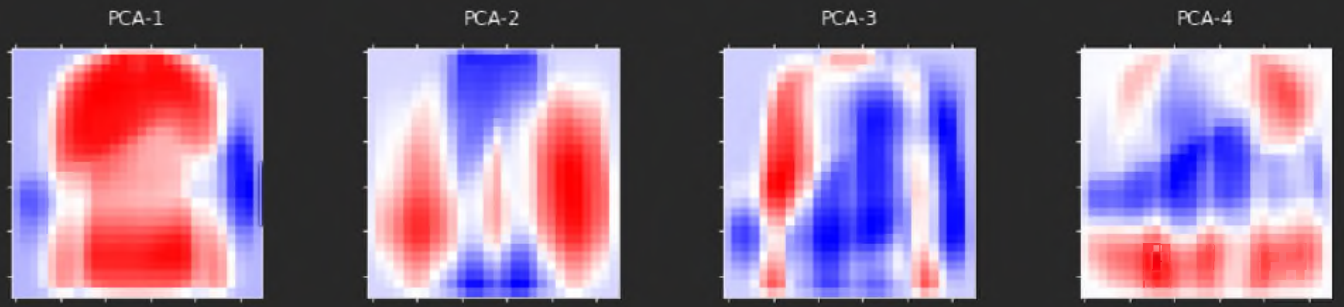


Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Что с лицом)







Фильтры

Исходное изображение

a	b	c
d	e	f
g	h	i

$\otimes$

3×3

размер фильтра

Матрица Фильтра

$$\frac{1}{\sum_{а}^{з}}$$

а	б	в
г	д	е
ё	ж	з

Результат фильтрации

$$\frac{1}{\sum_{а}^{з}} \begin{matrix} a*а+ & b*б+ & c*в+ \\ d*г+ & e*д+ & f*е+ \\ g*ё+ & h*ж+ & i*з \end{matrix}$$

a	b	c
d	ξ	f
g	h	i

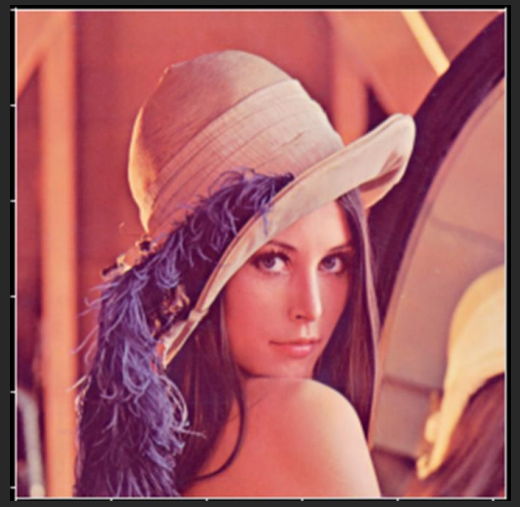
Фильтры

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

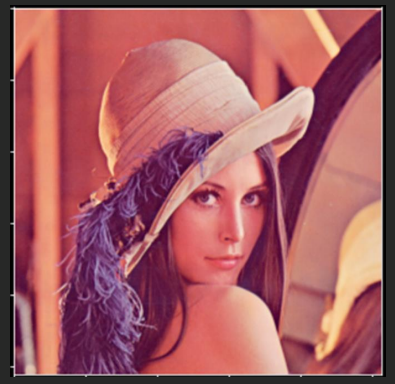
4		

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Разные Фильтры – Разный Результат

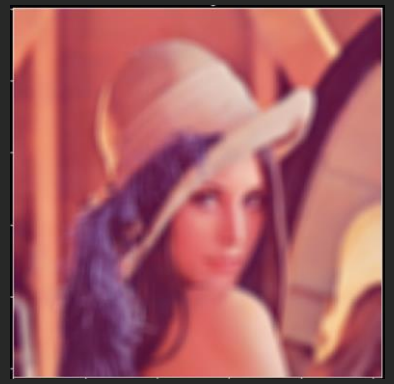


0	0	0
0	1	0
0	0	0



$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



$\frac{1}{10}$

-1	-2	-1
-2	22	-2
-1	-2	-1

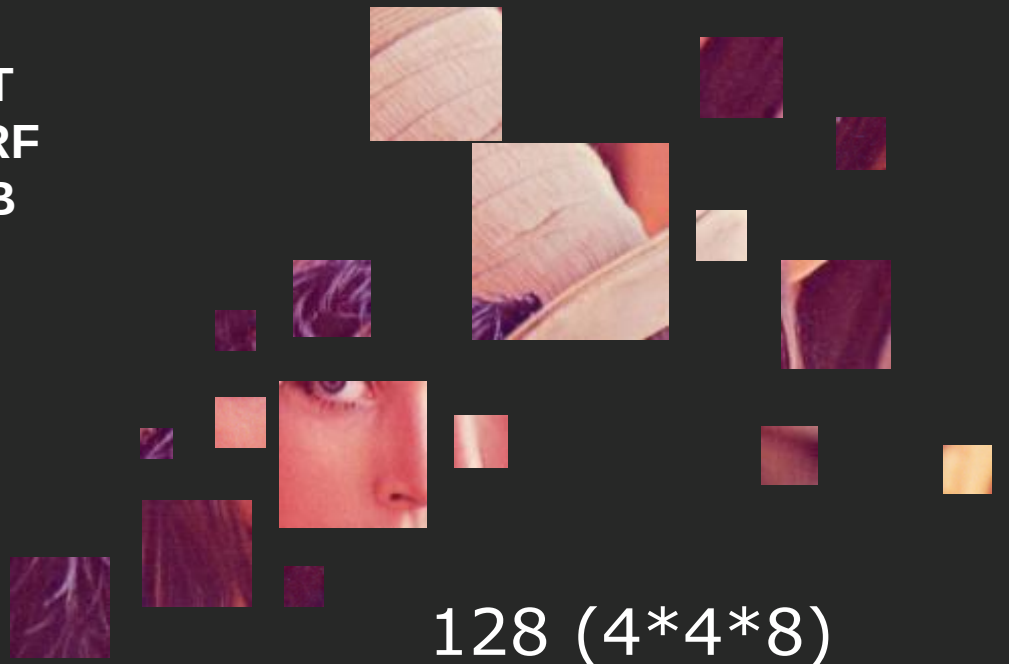
+1	0	-1
+2	0	-2
+1	0	-1

+1	+2	+1
0	0	0
-1	-2	-1

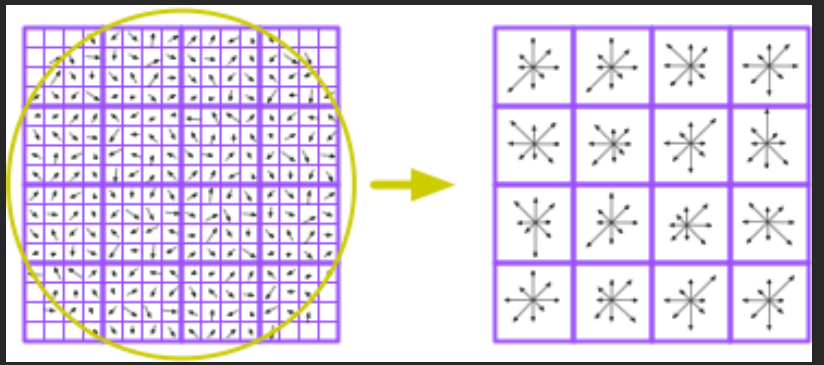
https://setosa.io/ev/image-kernels/

Инженерия признаков для Изображений

SIFT
SURF
ORB

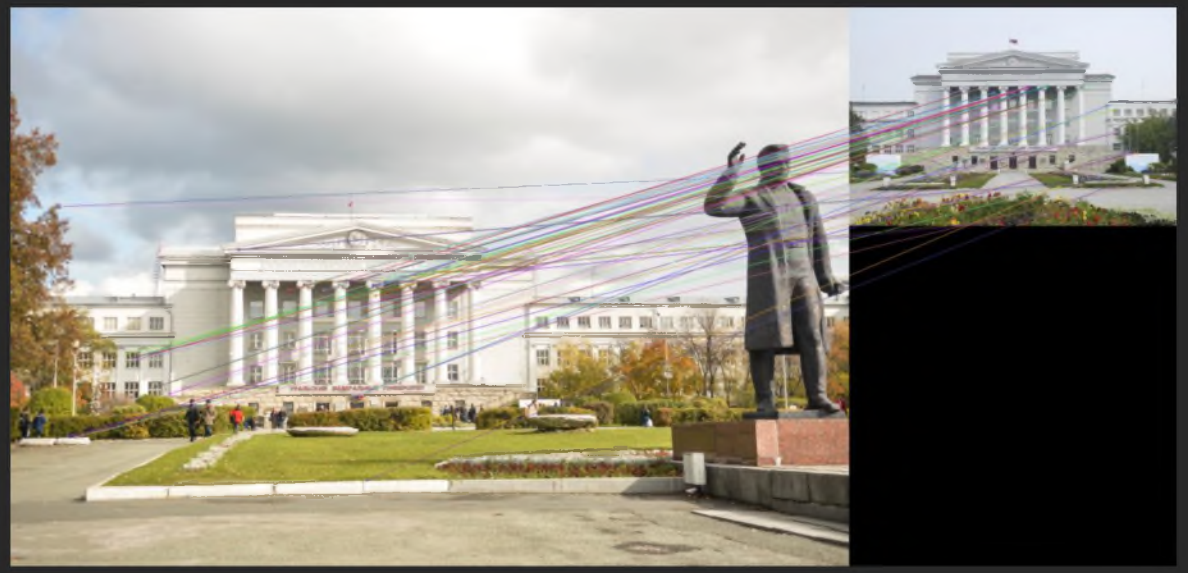


128 (4*4*8)

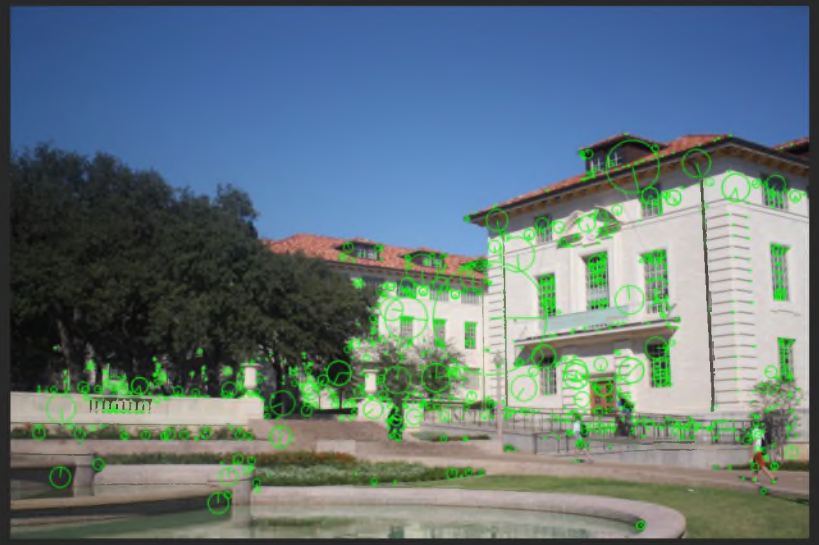


Уральский федеральный университет имени первого Президента России Б.Н. Ельцина

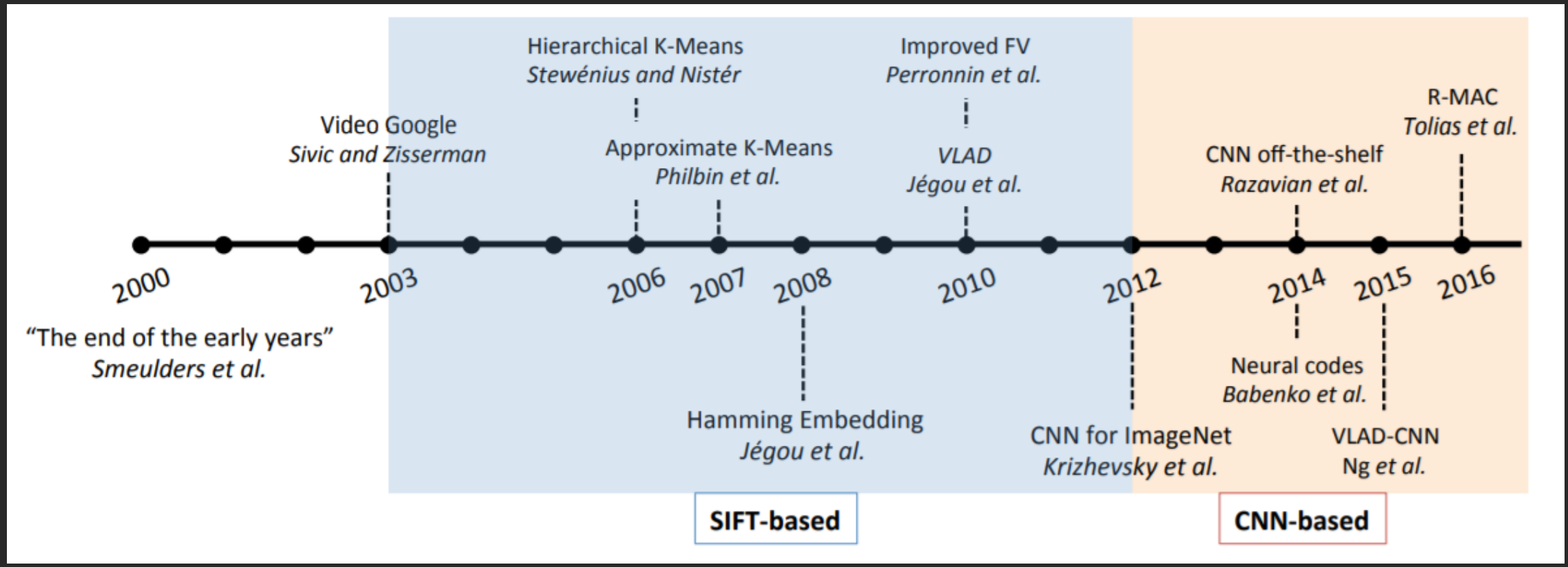
Инженерия признаков для Изображений



Инженерия признаков для Изображений



Развитие идей «простого» Машинного Обучения



Zheng, Liang, Yi Yang, and Qi Tian. "SIFT meets CNN: A decade survey of instance retrieval." *IEEE transactions on pattern analysis and machine intelligence* 40, no. 5 (2017): 1224-1244.

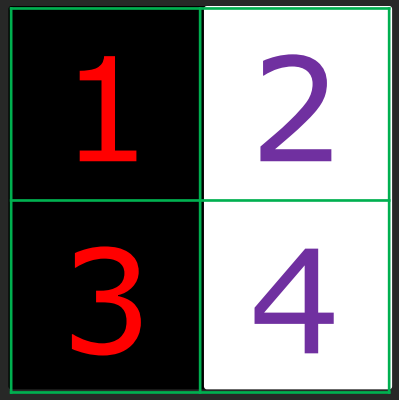
Viola Jones в OpenCV

Признаки у Виола-Джонс

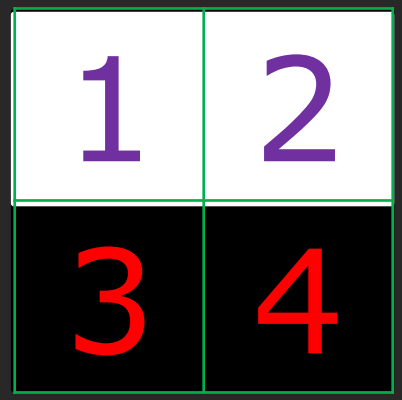
"Прямоугольные фильтры" (сложная вариация признаков Хаара)

Численно:

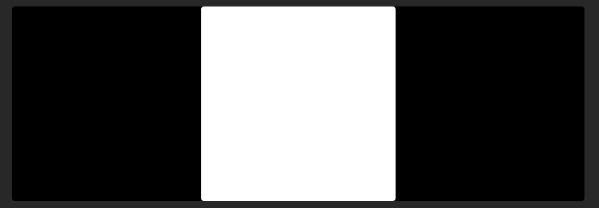
Σ(Что попало в Белую зону) –
Σ (Что попало в Черную зону)

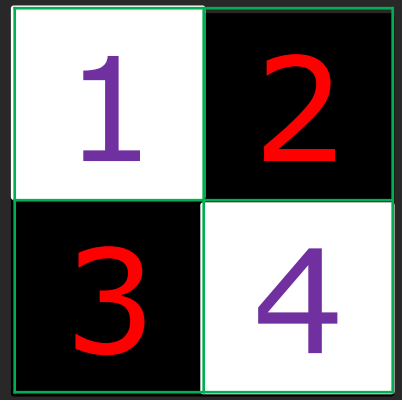


6 – 4 = 2



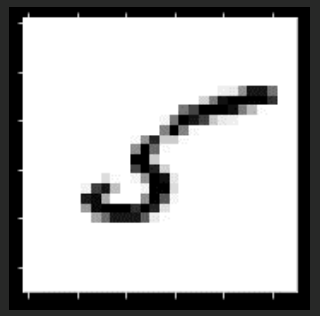
3 – 7 = –4



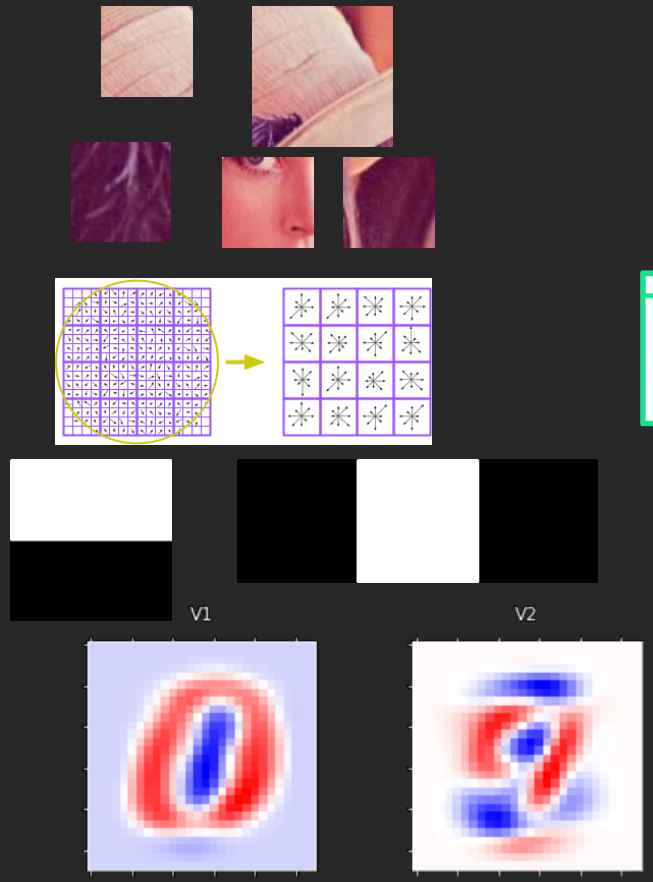


5 – 5 = 0

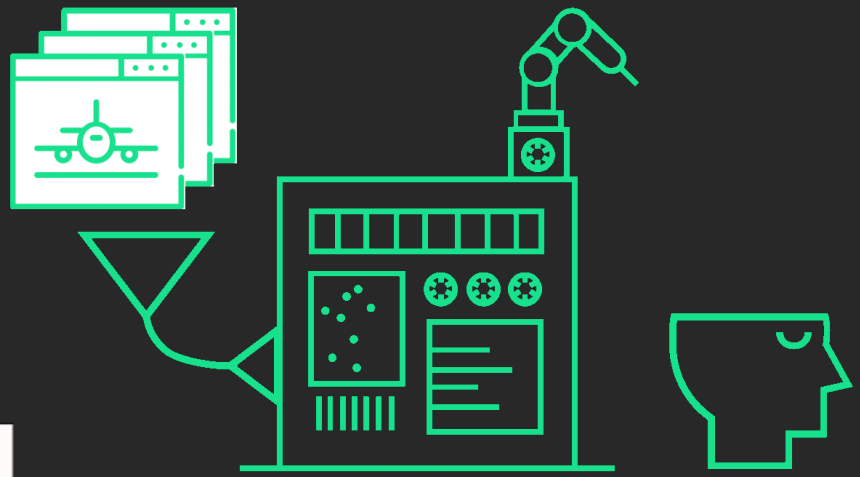
Машинное Обучение



Входные данные



Признаки



Машинное обучение

Девушка
Лена
В шляпе

Это пять!

Ожидаемый ответ

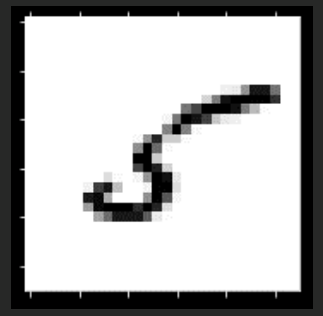
Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

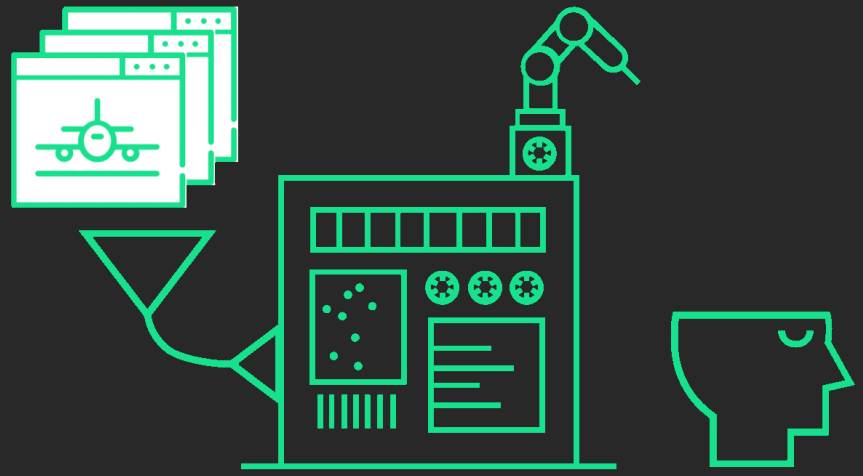
Про изображения

Про нейронки

Нейронные Сети



Входные данные



Нейронные Сети

Девушка
Лена
В шляпе

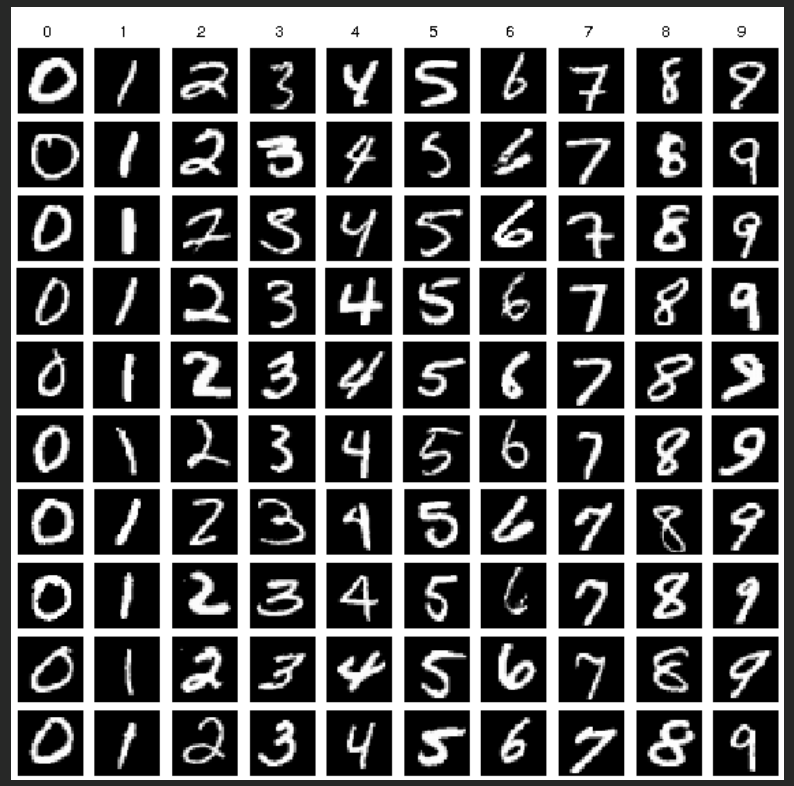
Это пять!

Ожидаемый ответ

Фреймфорки


Два лидера – Tensorflow и PyTorch
SBER AI
TensorFlow
PyTorch
Разработка
Google, 2015
Facebook , 2017
Простота разработки
Скорость обучения
Скорость инференса
Качество документации
Промышл. применение
Исследования и прототипы
Создан TensorFlow Lite – специально для мобильных устройств и Federated Learning

Пример про числа



28

0.01	0.00	0.05	0.05	0.04	0.00	0.04	0.00	0.01	0.04
0.05	0.01	0.81	0.72	0.64	0.81	0.64	0.54	0.93	0.03
0.01	0.05	0.02	0.03	0.01	0.00	0.04	0.05	0.92	0.01
0.05	0.04	0.05	0.05	0.04	0.05	0.03	0.04	0.84	0.03
0.04	0.04	0.03	0.01	0.02	0.01	0.05	0.52	0.03	0.05
0.03	0.00	0.05	0.02	0.04	0.04	0.73	0.03	0.00	0.01
0.05	0.03	0.02	0.00	0.05	0.85	0.03	0.01	0.00	0.03
0.03	0.05	0.05	0.03	0.90	0.02	0.00	0.01	0.04	0.01
0.02	0.04	0.05	0.82	0.05	0.02	0.03	0.01	0.00	0.02
0.05	0.04	0.05	0.92	0.02	0.03	0.02	0.05	0.03	0.05

28

```
from tensorflow.keras import datasets

(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

train_images, test_images = train_images / 255.0, test_images / 255.0
```

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Преобразуем входные данные

0.01	0.00	0.05	0.05	0.04	0.00	0.04	0.00	0.01	0.04
0.05	0.01	0.81	0.72	0.64	0.81	0.64	0.54	0.93	0.03
0.01	0.05	0.02	0.03	0.01	0.00	0.04	0.05	0.92	0.01
0.05	0.04	0.05	0.05	0.04	0.05	0.03	0.04	0.84	0.03
0.04	0.04	0.03	0.01	0.02	0.01	0.05	0.52	0.03	0.05
0.03	0.00	0.05	0.02	0.04	0.04	0.73	0.03	0.00	0.01
0.05	0.03	0.02	0.00	0.05	0.85	0.03	0.01	0.00	0.03
0.03	0.05	0.05	0.03	0.90	0.02	0.00	0.01	0.04	0.01
0.02	0.04	0.05	0.82	0.05	0.02	0.03	0.01	0.00	0.02
0.05	0.04	0.05	0.92	0.02	0.03	0.02	0.05	0.03	0.05

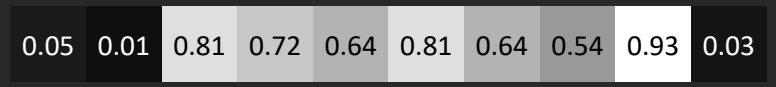
0.01	0.00	0.05	0.05	0.04	0.00	0.04	0.00	0.01	0.04
0.05	0.01	0.81	0.72	0.64	0.81	0.64	0.54	0.93	0.03
0.01	0.05	0.02	0.03	0.01	0.00	0.04	0.05	0.92	0.01
0.05	0.04	0.05	0.05	0.04	0.05	0.03	0.04	0.84	0.03
0.04	0.04	0.03	0.01	0.02	0.01	0.05	0.52	0.03	0.05
0.03	0.00	0.05	0.02	0.04	0.04	0.73	0.03	0.00	0.01
0.05	0.03	0.02	0.00	0.05	0.85	0.03	0.01	0.00	0.03
0.03	0.05	0.05	0.03	0.90	0.02	0.00	0.01	0.04	0.01
0.02	0.04	0.05	0.82	0.05	0.02	0.03	0.01	0.00	0.02
0.05	0.04	0.05	0.92	0.02	0.03	0.02	0.05	0.03	0.05

0.00 0.05 0.05 0.04 0.00 0.04 0.00 0.01 0.04 0.00 0.01 0.04 0.05 0.01 0.81 0.72 0.64 0.81 0.64 0.54 0.93 0.03 0.01 0.05 0.02 0.03 0.01 0.00 0.04 0.05 0.92 0.01 0.05 0.04 0.05 0.05 0.04 0.05 0.03 0.04 0.84 0.03 0.04 0.04 0.03 0.01 0.02 0.01 0.05 0.52 0.03 0.05 0.03 0.00 0.05 0.02 0.04 0.04 0.73 0.03 0.00 0.01 0.05 0.03 0.02 0.00 0.05 0.85 0.03 0.01 0.00 0.03 0.03 0.05 0.05 0.03 0.90 0.02 0.00 0.01 0.04 0.01 0.02 0.04 0.05 0.82 0.05 0.02 0.03 0.01 0.00 0.02 0.05 0.04 0.05 0.92 0.02 0.03 0.02 0.05 0.03 0.05

784

```
from tensorflow.keras import layers
layers.Flatten(input_shape=(28, 28))
```

Немного Терминологии

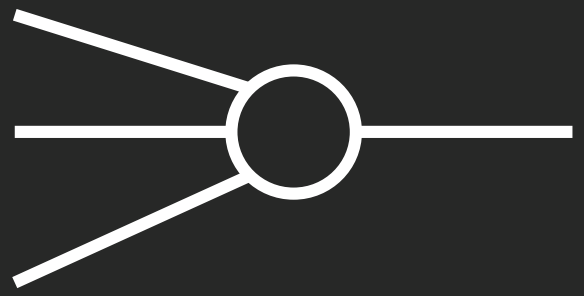


0.81 уровень активации

Нейрон
функция активации $f()$
И сумма входов

Вход

Разные X

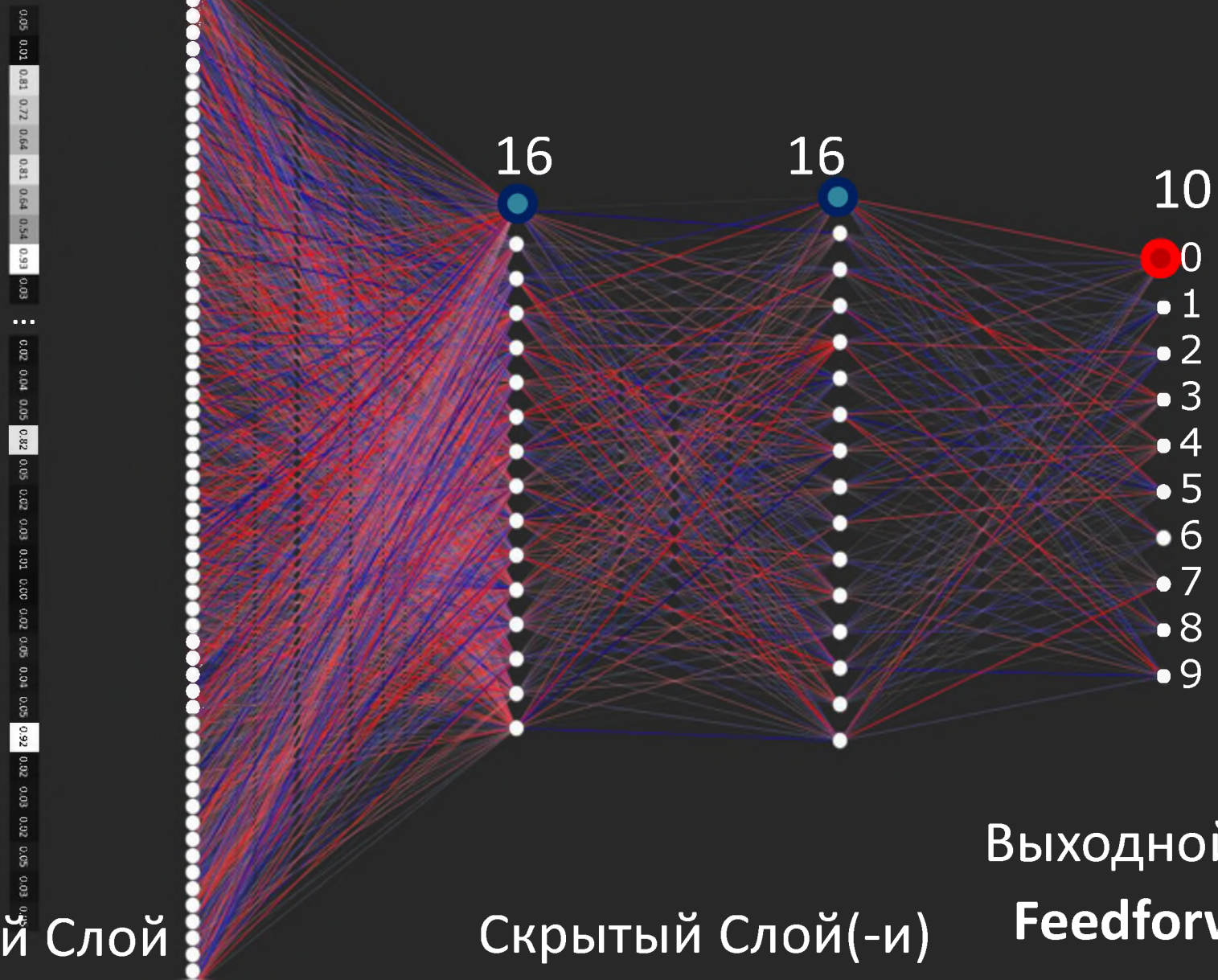


Выход

Результат Применения
$f(x_1 + x_2 + x_3)$

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Структура Нейронной Сети



Почему два скрытых слоя?

Почему в скрытых слоях 16 нейронов?

Входной Слой

Скрытый Слой(-и)

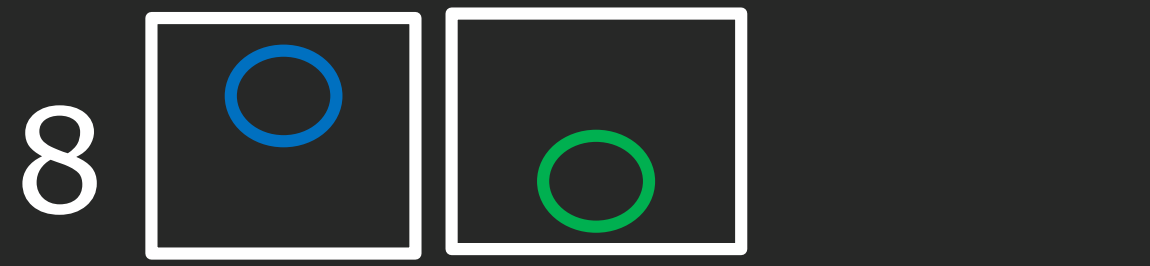
Выходной Слой

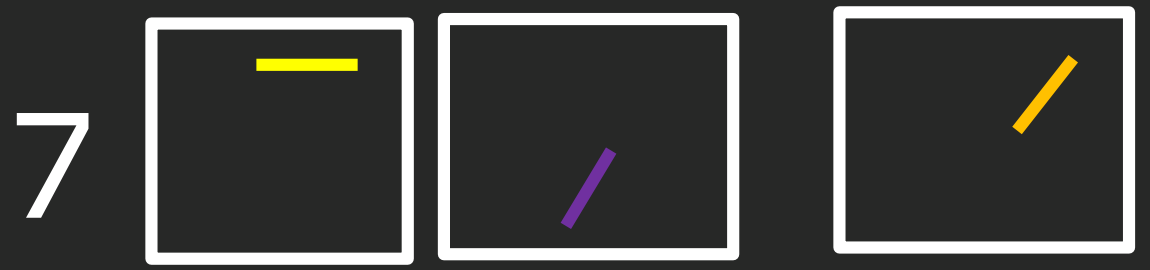
Feedforward Прямая связь

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

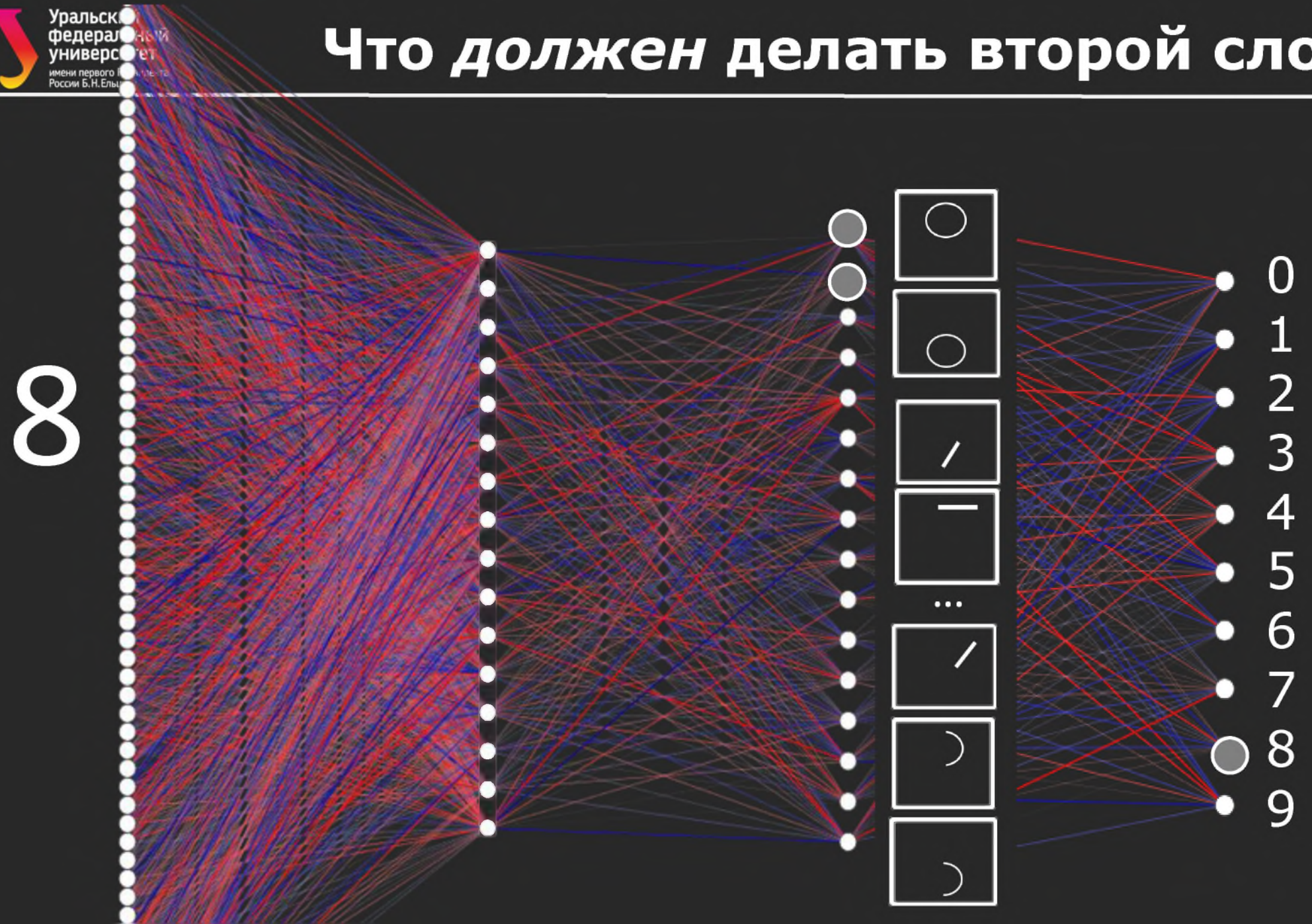
Почему это может сработать?

0 1 2 3 4 5 6 7 8 9





Что *должен* делать второй слой?

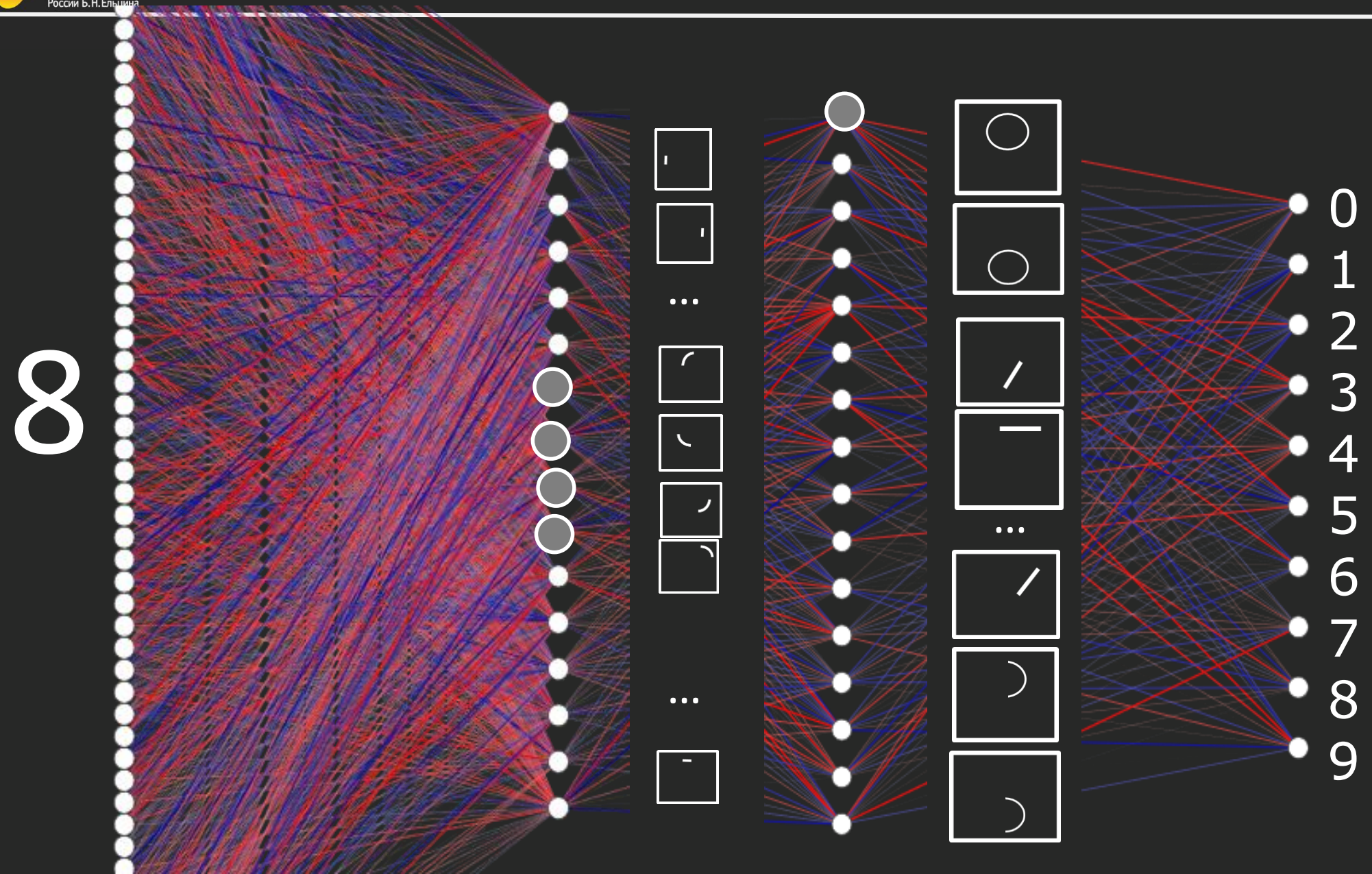


Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

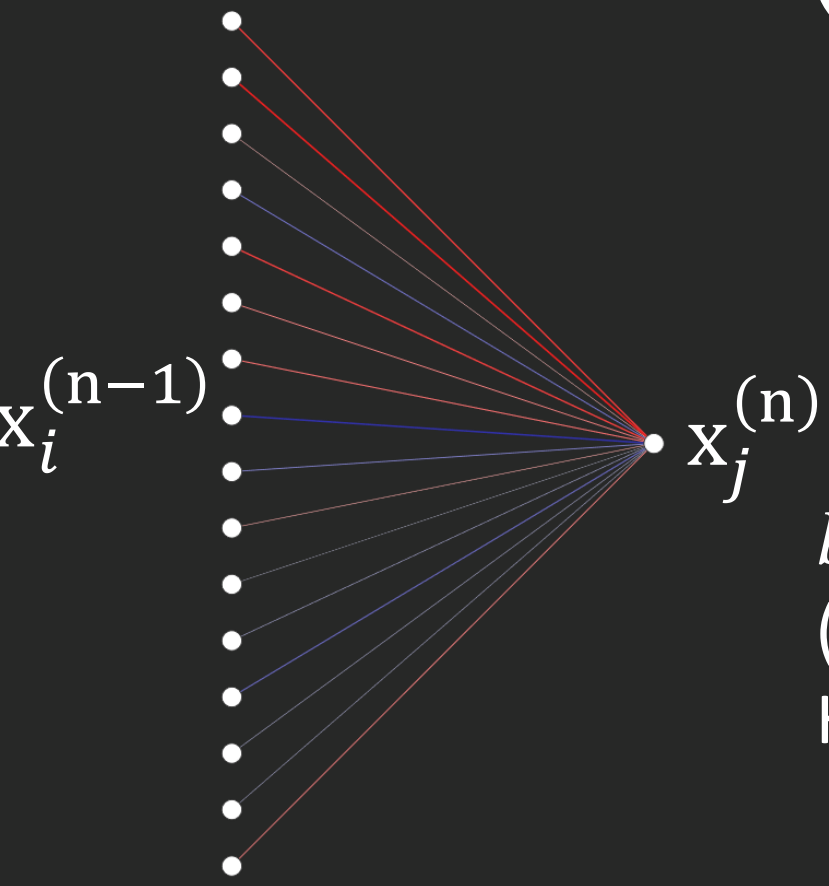
Почему это может сработать? (часть 2)

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Что *должен* делать первый слой?



Отдельные Нейроны



○ $x_j^{(n)}$ сумма (по i) всех входных нейронов $x_i^{(n-1)}$

— не только связь, но и веса $w_{n,i}$

$$w_{n,1}\, x_1^{(n)} + w_{n,2}\, x_2^{(n)} + \ldots + w_{n,N-1}\, x_{N-1}^{(n)} + w_{n,N}\, x_N^{(n)} + b_n$$

b_n значит “bias”
(предвзятость, смещение)
Независимый элемент

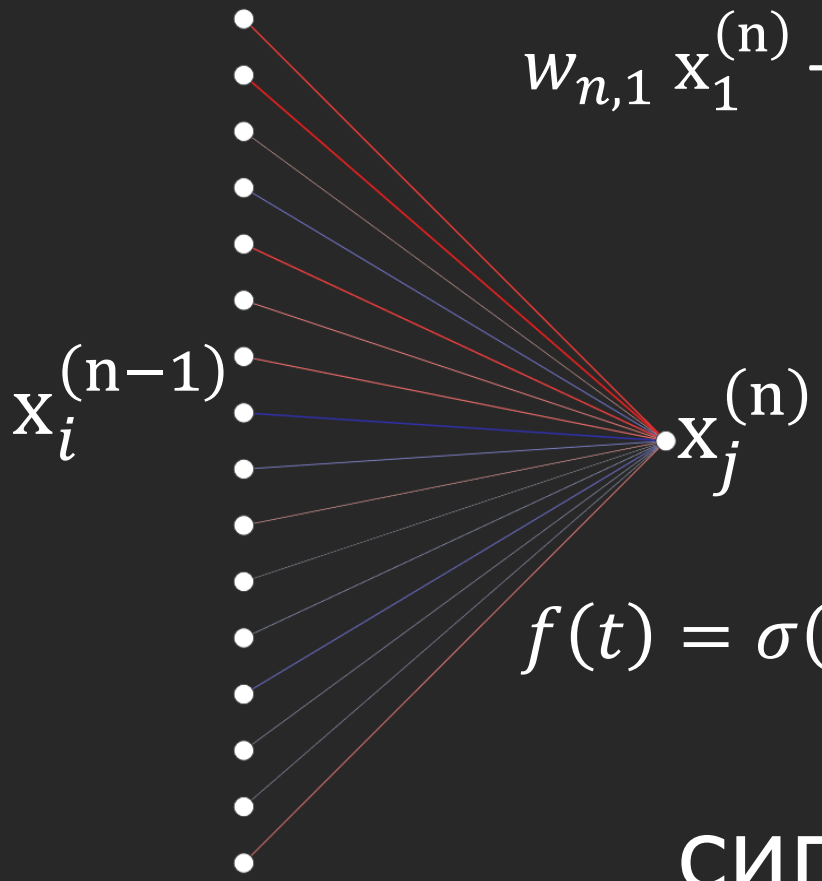
0.55	0.95	-0.89	0.80	0.49	-0.36	-0.65	-0.83	0.10	0.35
-0.75	0.58	-0.45	0.04	-0.61	0.93	0.29	-0.62	0.42	-0.47
0.95	0.49	-0.55	0.38	0.23	0.59	0.13	-0.18	-0.25	-0.27
0.20	-0.13	0.92	-0.48	-0.27	1.00	0.55	-0.47	0.72	-0.02
-0.78	-0.35	0.34	-0.35	-0.52	0.04	0.30	-0.40	0.59	-0.90
-0.51	0.14	0.66	-0.15	0.51	-0.25	-0.46	0.92	-0.17	-0.84
-0.99	-0.70	-0.37	0.37	-0.50	-0.67	0.45	-0.31	-0.42	-0.04
-0.28	0.12	-0.65	-0.10	-0.16	0.70	-0.33	0.04	-0.48	0.30
-0.57	0.06	-0.03	-0.56	-0.50	0.38	0.71	-0.37	-0.39	-0.42
0.12	0.36	0.78	-0.14	0.54	-0.92	-0.06	-0.27	0.27	0.80

Что делают Функции активации

$x_j^{(n)}$ сумма (по i) всех входных нейронов $x_i^{(n-1)}$ с весами и константой

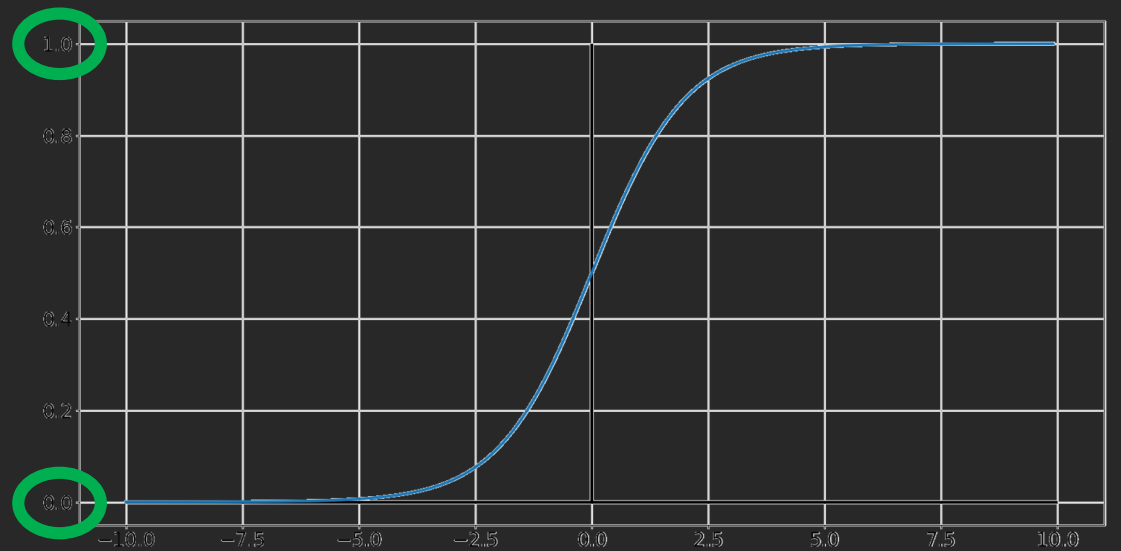
$$w_{n,1}\, x_1^{(n)} + w_{n,2}\, x_2^{(n)} + \ldots + w_{n,N-1}\, x_{N-1}^{(n)} + w_{n,N}\, x_N^{(n)} + b_n \in \{-\infty;+\infty\}$$

Хотелось бы, чтобы сумма є {0;1}

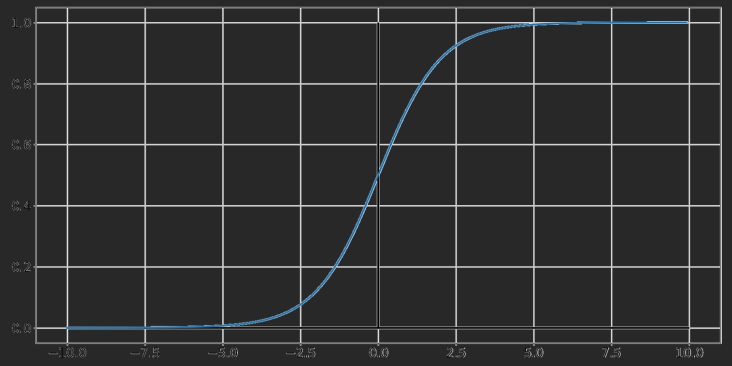


$$f(t) = \sigma(t) = \frac{1}{1 + e^{-t}}$$

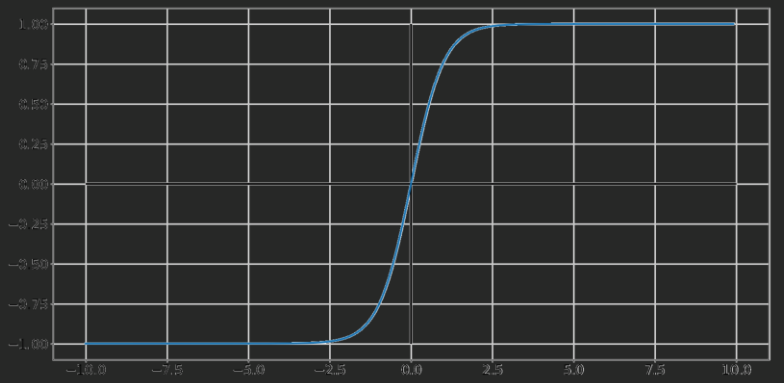
сигмоид



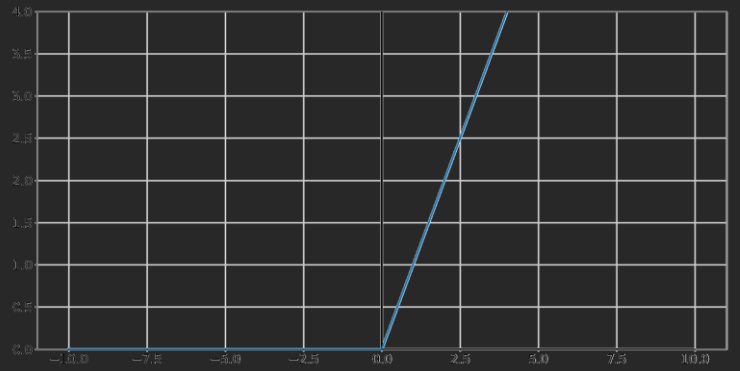
Функции Активации



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

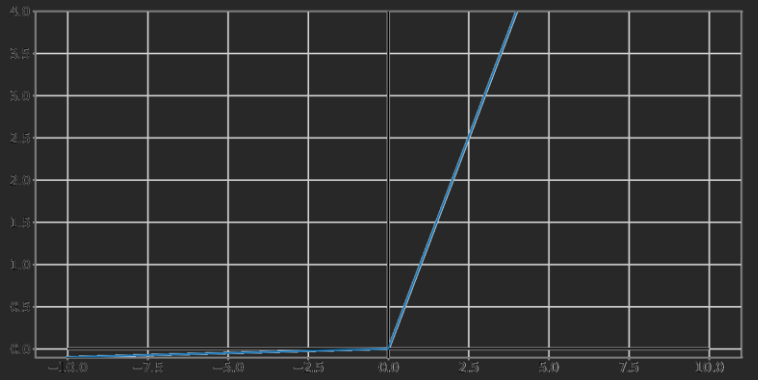


$$f(t) = \tanh(t) = \frac{e^{t}-e^{-t}}{e^{t}+e^{-t}}$$



Rectified linear unit (ReLU)

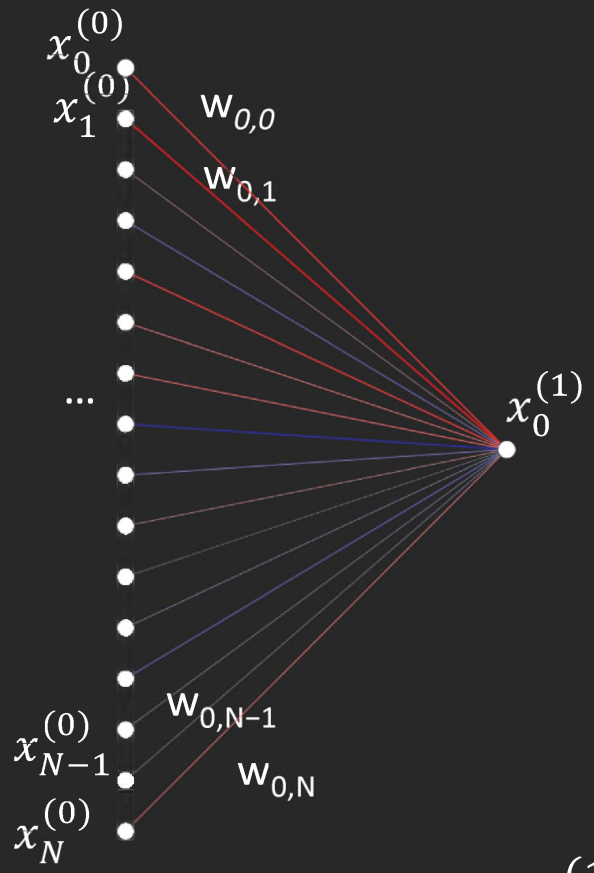
$$f(t) = ReLU(t) = \max(0, t)$$

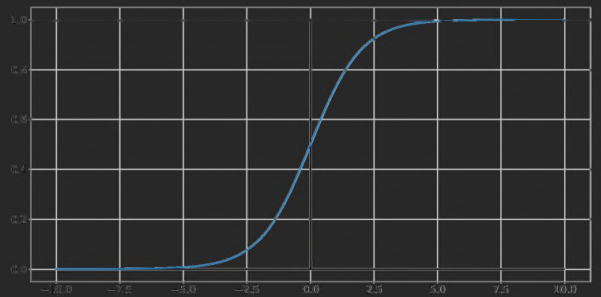


Leaky Rectified linear unit (ReLU)

$$f(t) = LReLU(t) = \max(a * t, t),$$

Что значит найти элемент?





Подобрать веса

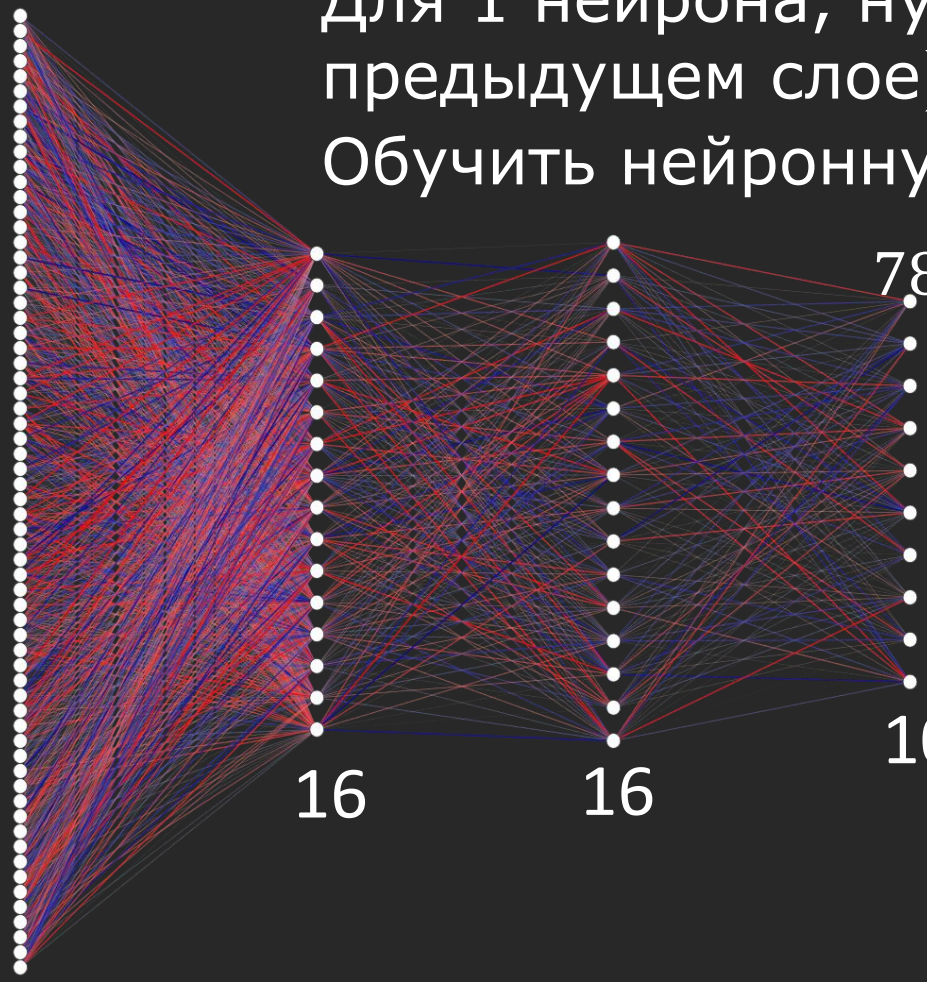
0.03	0.00	0.05	0.00	0.04	0.01	0.05	0.03	0.02	0.03
0.04	0.04	0.03	-0.49	-0.49	-0.50	-0.45	0.02	0.05	0.05
0.00	0.00	-0.47	0.23	0.42	0.34	0.44	-0.51	0.02	0.00
0.05	0.00	0.04	-0.50	-0.48	-0.47	-0.49	0.05	0.02	0.00
0.02	0.05	0.04	0.01	0.02	0.02	0.03	0.02	0.02	0.04
0.05	0.05	0.00	0.02	0.05	0.00	0.01	0.00	0.00	0.04
0.04	0.03	0.01	0.00	0.03	0.05	0.00	0.02	0.05	0.04
0.00	0.04	0.01	0.00	0.05	0.04	0.00	0.02	0.05	0.05
0.00	0.04	0.00	0.03	0.05	0.05	0.00	0.05	0.02	0.00
0.03	0.03	0.03	0.04	0.02	0.05	0.01	0.01	0.03	0.02

$$x_0^{(1)} = \sigma\left(w_{0,1}x_0^{(0)} + w_{0,2}x_1^{(0)} + \ldots + w_{0,N-1}x_{N-1}^{(0)} + w_{0,N}x_N^{(0)} - b_0\right)$$

Что значит обучить Нейронную Сеть

Для 1 нейрона, нужно найти N весов (число нейрнонов на предыдущем слое) и 1 константу bias

Обучить нейронную сеть – Найти все веса и bias



$$784 * 16 + 16 * 16 + 16 * 10 = 12960$$ Веса

$$16 + 16 + 10 = 42$$ bias

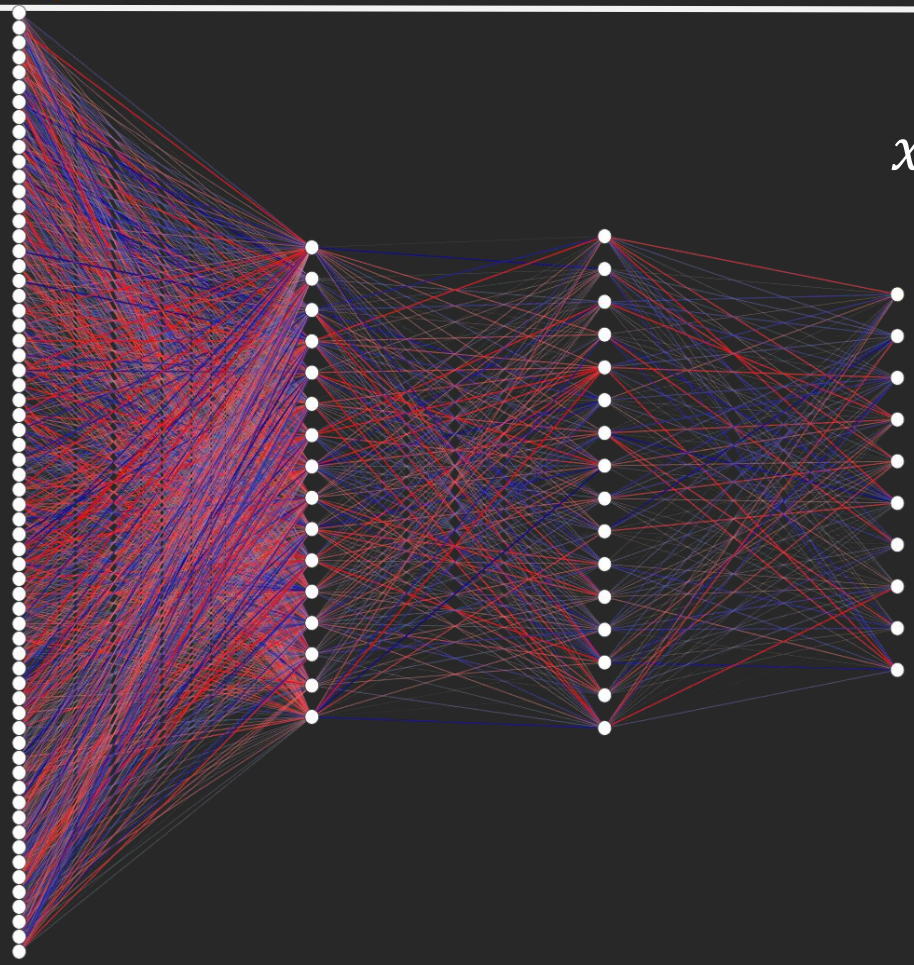
$$W_1 = \begin{bmatrix} w_{0,1} & \cdots & w_{0,N} \\ \vdots & \ddots & \vdots \\ w_{K,1} & \cdots & w_{K,N} \end{bmatrix} \quad x^{(0)} = \begin{bmatrix} x_0^{(0)} \\ \dots \\ x_K^{(0)} \end{bmatrix} \quad b = \begin{bmatrix} b_0 \\ \dots \\ b_N \end{bmatrix}$$

$$x^{(1)} = \sigma(W_1 * x^{(0)} + b)$$

y=f(X)

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Нейронка это «всего лишь» функция



$$x_0^{(1)} = f\left(w_{0,1}x_0^{(0)} + w_{0,2}x_1^{(0)} + \ldots + w_{0,N-1}x_{N-1}^{(0)} + w_{0,N}x_N^{(0)} - b_0\right)$$

$$W^{(n)} = \begin{bmatrix} w_{0,1} & \cdots & w_{0,N} \\ \vdots & \ddots & \vdots \\ w_{K,1} & \cdots & w_{K,N} \end{bmatrix} x^{(n)} = \begin{bmatrix} x_0^{(n)} \\ \ldots \\ x_K^{(n)} \end{bmatrix} b^{(n)} = \begin{bmatrix} b_0 \\ \ldots \\ b_N \end{bmatrix}$$

$$x^{(1)} = f(W^{(1)} * x^{(0)} + b^{(1)})$$

$$x^{(2)} = f(W^{(2)} * x^{(1)} + b^{(2)})$$

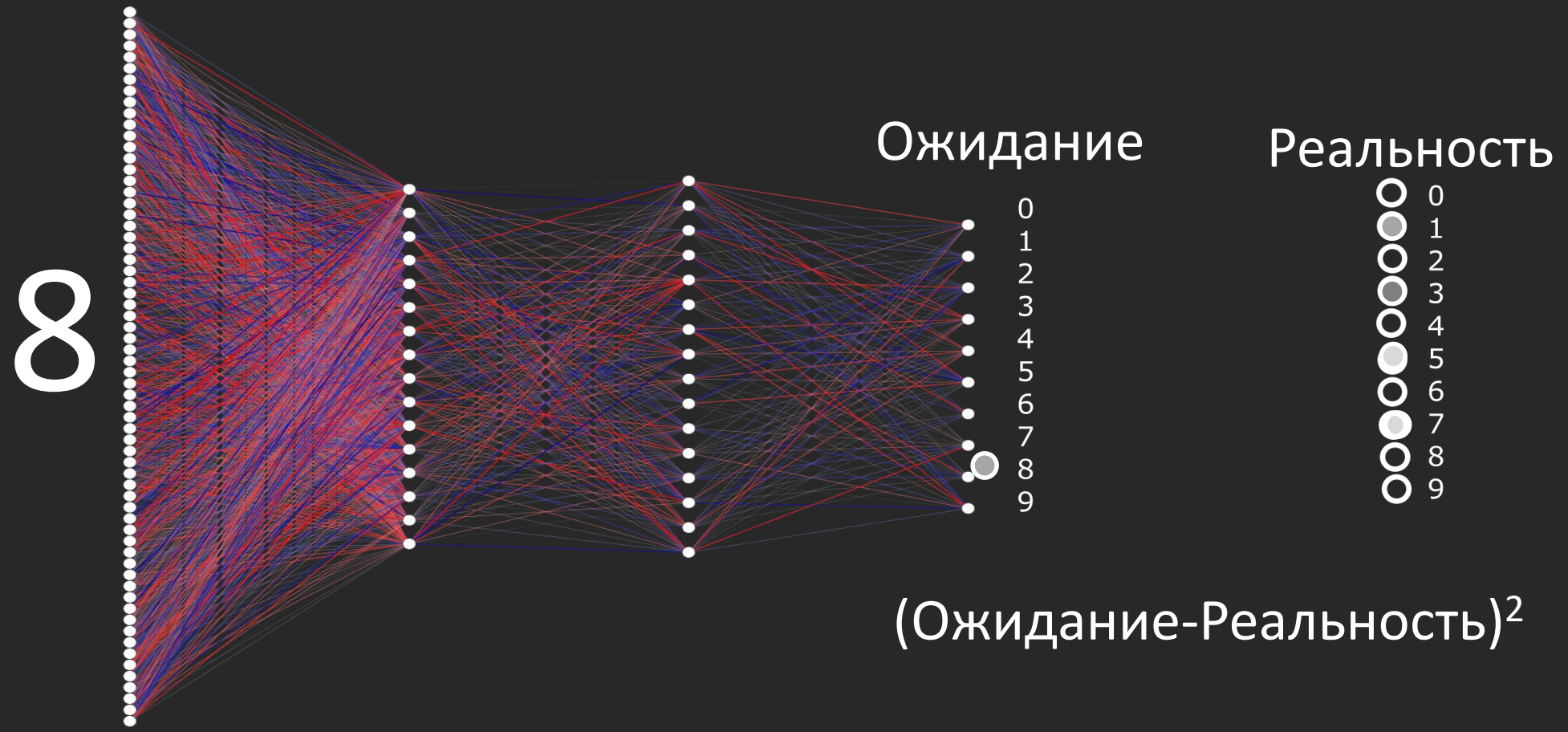
$$x^{(3)} = f(W^{(3)} * x^{(2)} + b^{(3)})$$

$$x^{(3)} = f(W^{(3)} * f(W^{(2)} * f(W^{(1)} * x^{(0)} + b^{(1)}) + b^{(2)}) + b^{(3)})$$

f НЕ линейная функция (иначе в чем смысл)

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

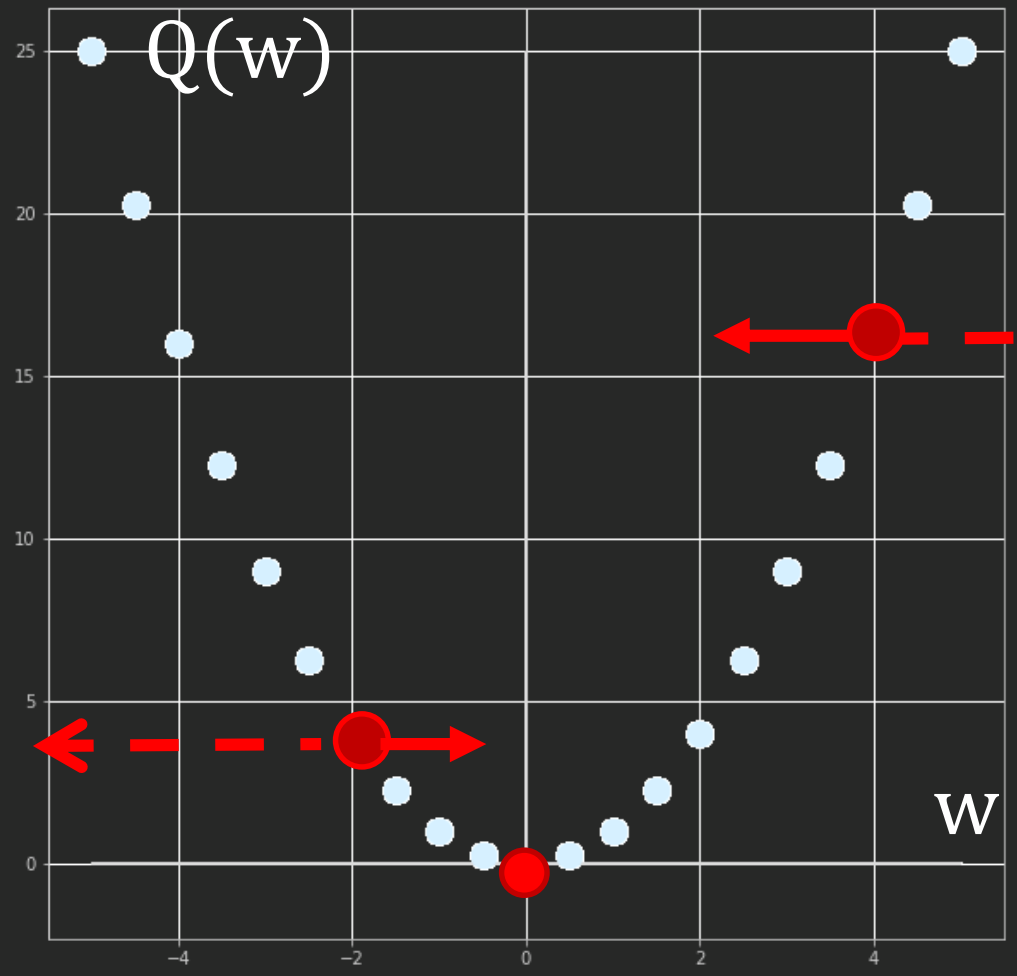
Пытаемся обучать Нейронную сеть

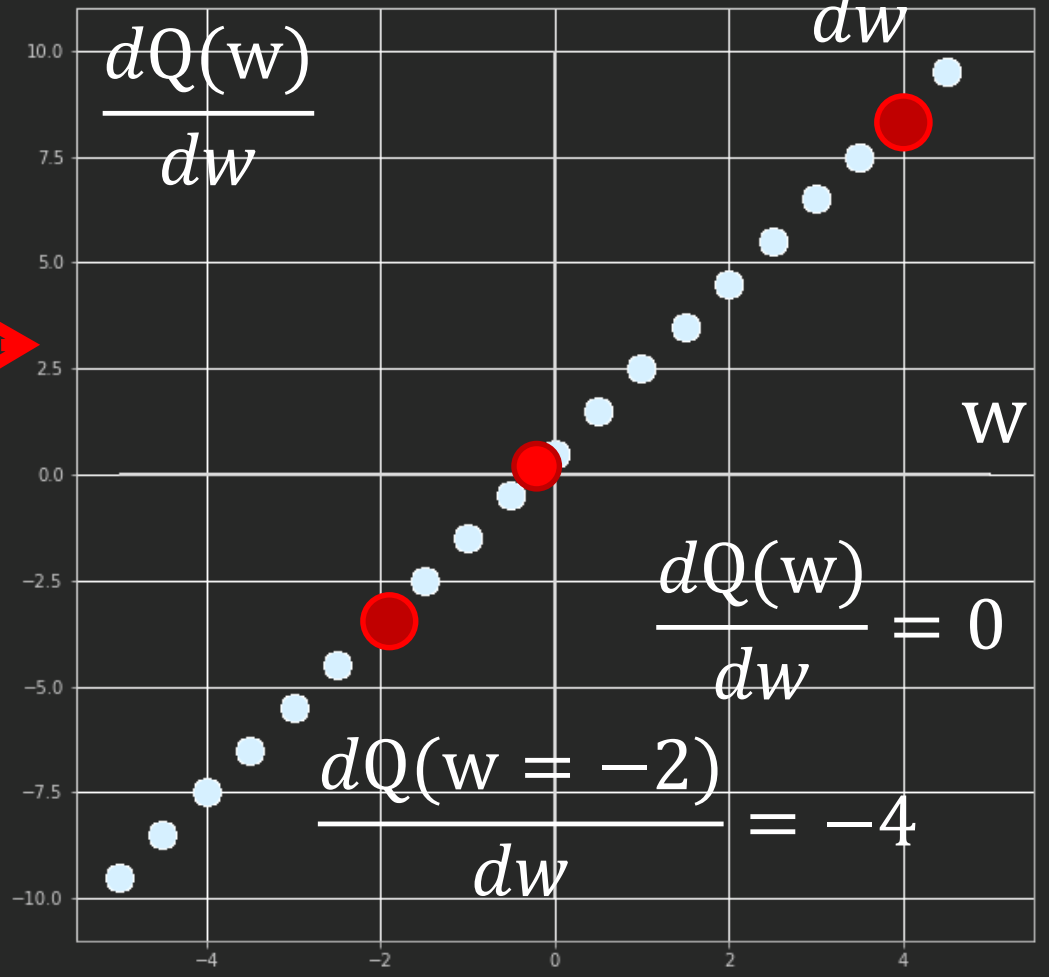


(Ожидание-Реальность)^2

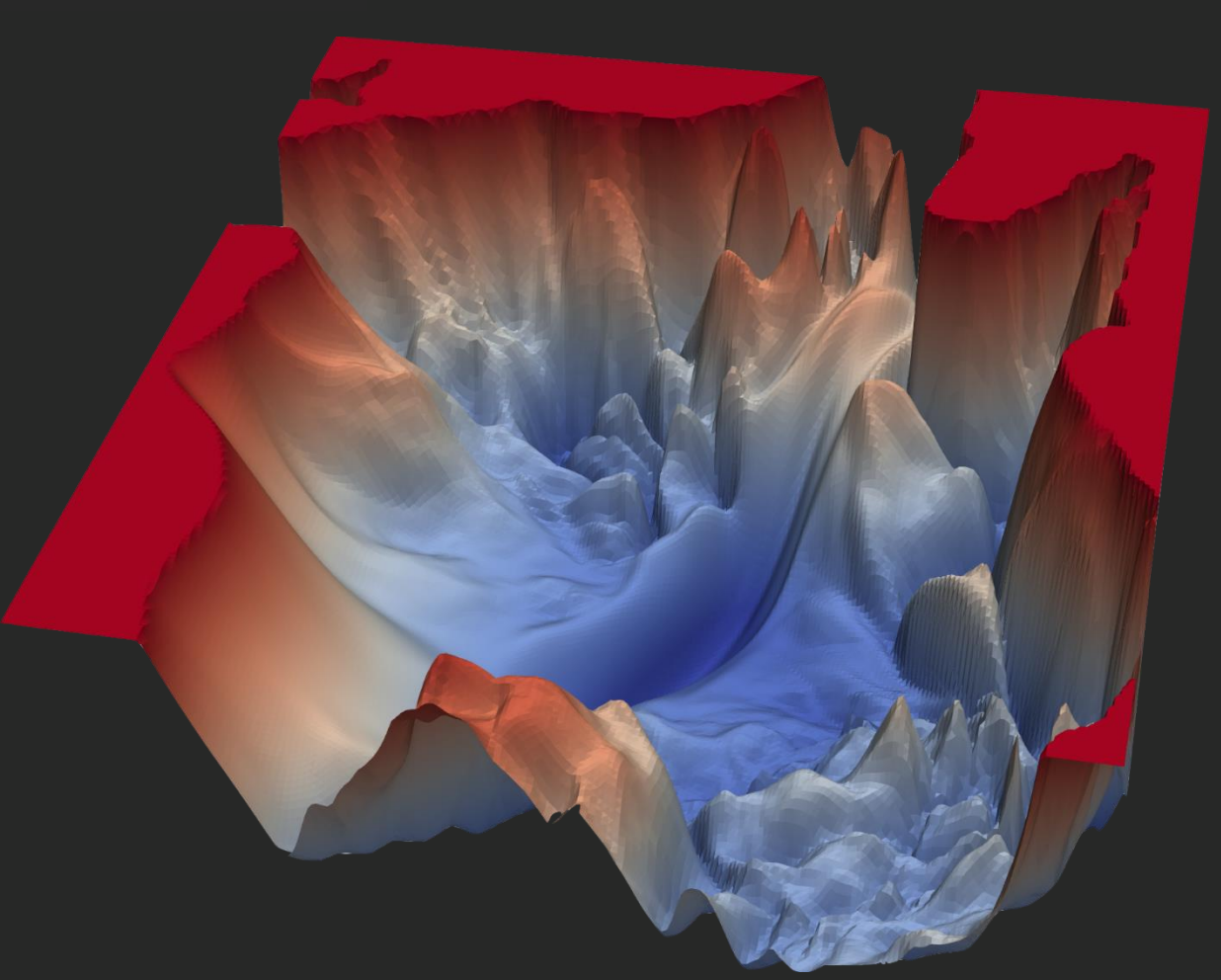
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \rightarrow \min$$

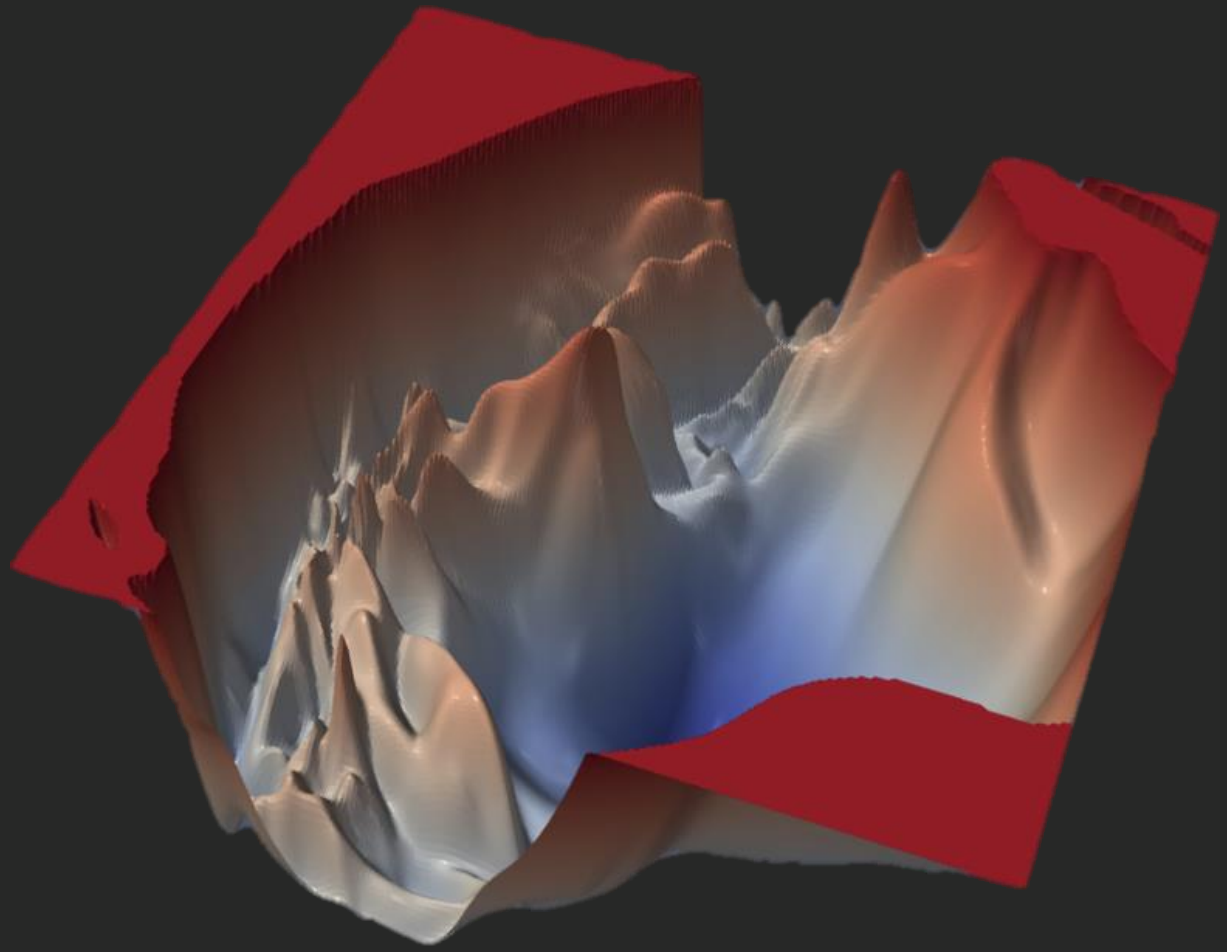
Про Оптимизацию Функции Потерь





Насколько все сложно





https://www.cs.umd.edu/~tomg/projects/landscapes/

Оптимизация на Простом Примере

$\nabla L(w_0, b_0, w_1, b_1, w_2, b_2)$

Ошибка $\rightarrow L_0=(x^{(N)}-y)^2$

$$x^{(N)} = \sigma(w_N\, x^{(N-1)} + b_N) = \sigma(z^{(N)})$$

$$z^{(N)} = w_N\, x^{(N-1)} + b_N$$

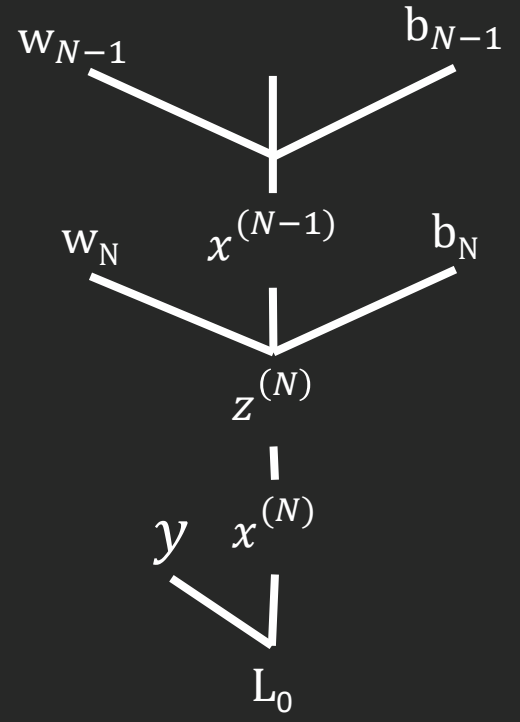
$$\frac{\partial L_0}{\partial w_N} = \frac{\partial z^{(N)}}{\partial w_N}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial L_0}{\partial x^{(N)}} = x^{(N-1)}\sigma'(z^{(N)})\, 2(x^{(N)}-y)$$

$$\frac{\partial L_0}{\partial x^{(N)}} = 2(x^{(N)}-y) \qquad \frac{\partial x^{(N)}}{\partial z^{(N)}} = \sigma'(z^{(N)}) \qquad \frac{\partial z^{(N)}}{\partial w_N} = x^{(N-1)}$$

$$\frac{\partial L_0}{\partial b_N} = \frac{\partial z^{(N)}}{\partial b_N}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial L_0}{\partial x^{(N)}} = \sigma'(z^{(N)})2(x^{(N)}-y)$$

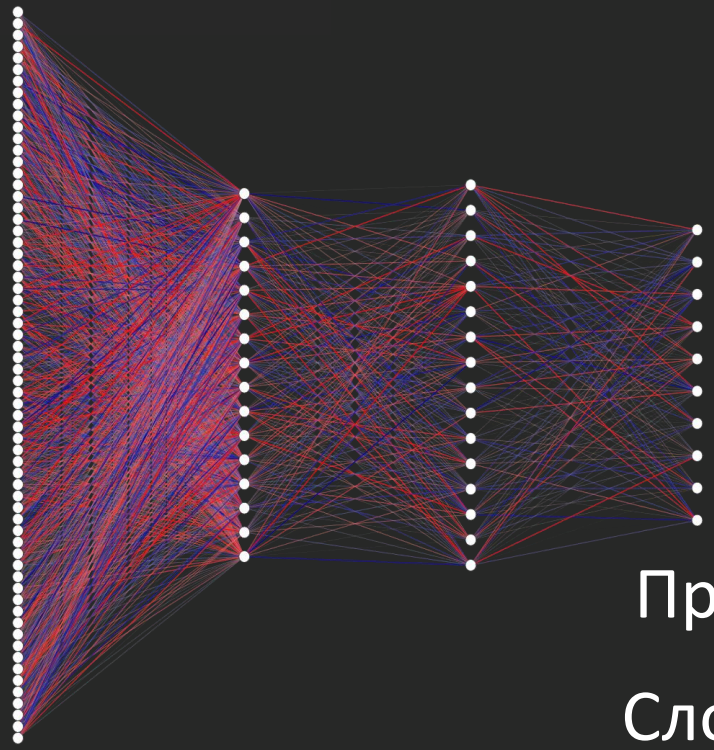
$$\frac{\partial L_0}{\partial x^{(N-1)}} = \frac{\partial z^{(N)}}{\partial x^{(N-1)}}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial L_0}{\partial x^{(N)}} = w_N\sigma'(z^{(N)})2(x^{(N)}-y)$$

Backpropagation
Обратное распространение



Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Возвращаемся к сложному примеру



Просто: Ошибка → $\mathrm{L}_0=(x^{(N)}-y)^2$

Сложно: Ошибка → $\mathrm{L}_0=\sum_{j=0}^{n_N-1}(x_j^{(N)}-y)^2$

Просто: $x^{(N)}=\sigma(\mathrm{w_N}\,x^{(N-1)}+\mathrm{b_N})=\sigma(z^{(N)})$

Сложно: $x_j^{(N)}=\sigma\{(\sum \mathrm{w}_{jk}\,x_k^{(N-1)})+\mathrm{b_N}\}=\sigma(z_j^{(N)})$

Просто: $\frac{\partial \mathrm{L}_0}{\partial x^{(N-1)}}=\frac{\partial z^{(N)}}{\partial x^{(N-1)}}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial \mathrm{L}_0}{\partial x^{(N)}}$

$\nabla \mathrm{O}(\dots,\mathrm{w}_{JK},\dots,\mathrm{b_N},\dots)$

Сложно: $\frac{\partial \mathrm{L}_0}{\partial x_K^{(N-1)}}=\sum_{J=0}^{n_N-1}\frac{\partial z_J^{(N)}}{\partial x_K^{(N-1)}}\frac{\partial x_J^{(N)}}{\partial z_J^{(N)}}\frac{\partial \mathrm{L}_0}{\partial x_J^{(N)}}$

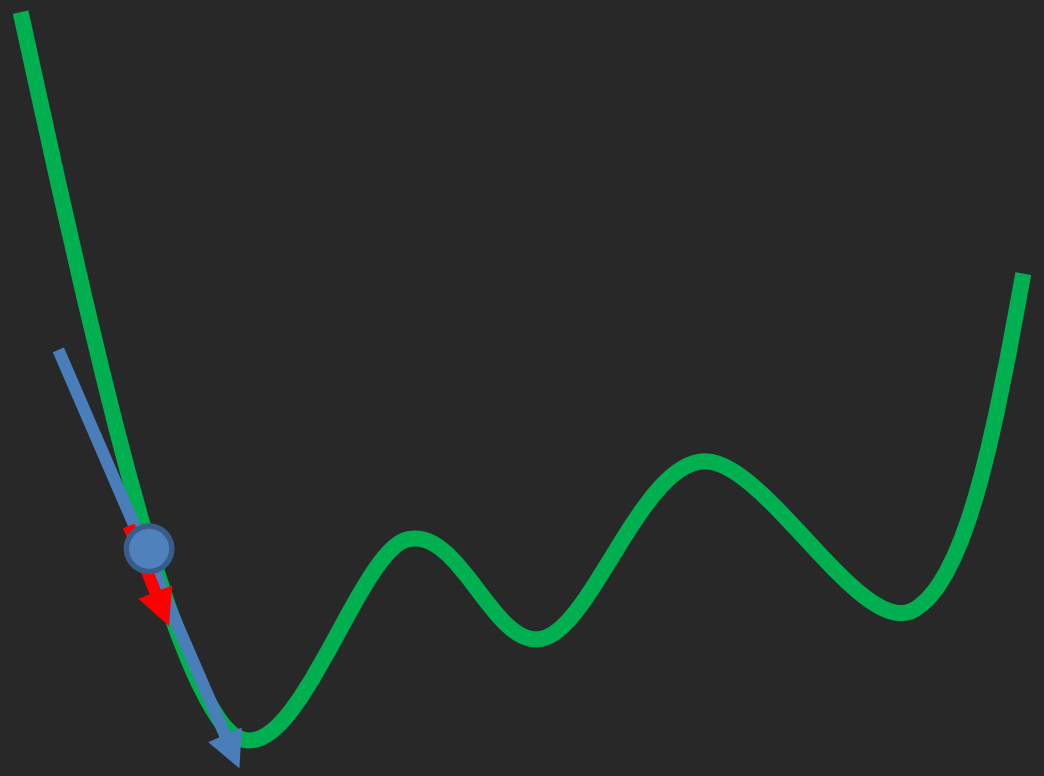
Оптимизация

Градиентный Спуск

$\nabla L(\vec{W})$

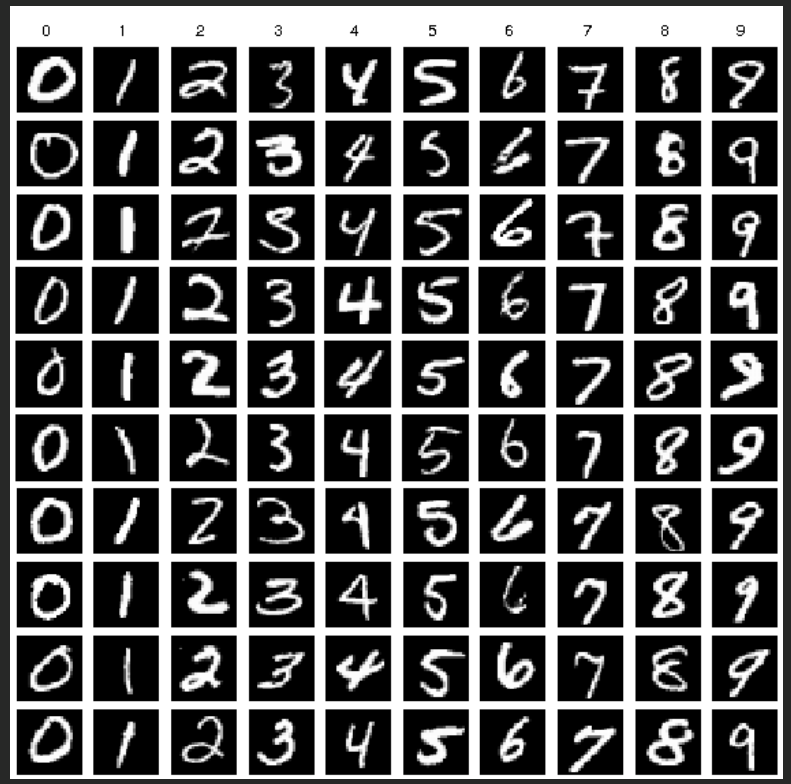
$$\vec{W}_{new} = \vec{W}_{old} - \eta \nabla L(\vec{W}_{old})$$

η – learning rate
Скорость обучения

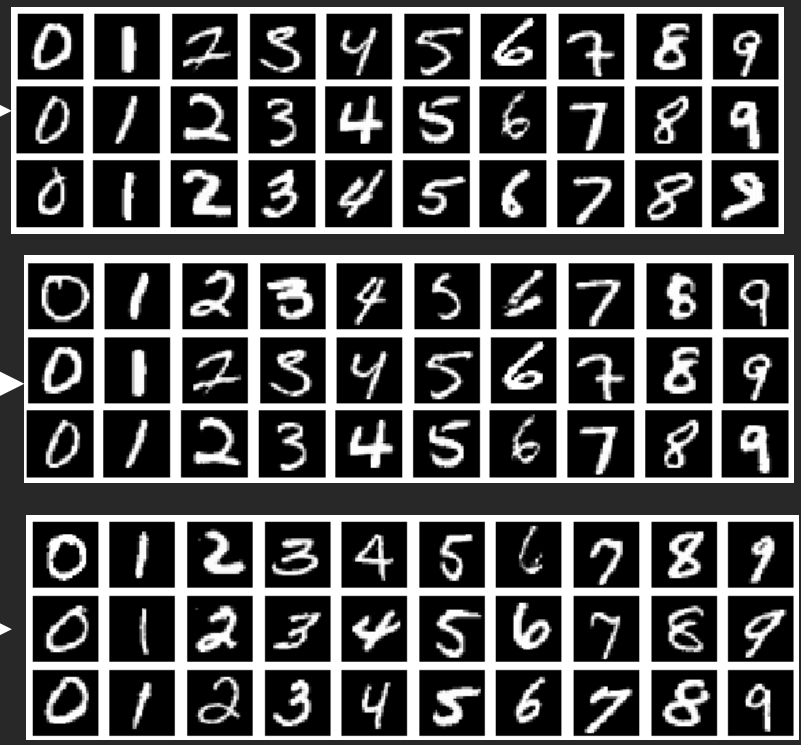


Оптимизация

(Стохастический) Градиентный Спуск



Исходная выборка



mini batch подвыборка

$$\overrightarrow{W}_{new} = \overrightarrow{W}_{mini-batch1} - \eta \nabla L\left(\overrightarrow{W}_{mini-batch2}\right)$$

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Два API в для создания сетей TensorFlow

```
import tensorflow as tf
from tensorflow.keras import layers, models
```

Sequential API

```
model = tf.keras.models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(16, activation='sigmoid'))
model.add(layers.Dense(16, activation='sigmoid'))
model.add(layers.Dense(10))
```

Functional API

```
Input = tf.keras.Input(shape = (28, 28))
Flattened = layers.Flatten()(Input)
Dense1 = layers.Dense(16, activation='sigmoid')(Flattened)
Dense2 = layers.Dense(16, activation='sigmoid')(Dense1)
Output = layers.Dense(10)(Dense2)
Model = tf.keras.Model(inputs = Input, outputs = Output )
```

Обучение Нейронной Сети

```
model.compile(optimizer='SGD',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])
```

- *Loss function* —измеряет, насколько точна модель во время обучения. Необходимо минимизировать эту функцию, чтобы «направить» модель в правильном направлении.

- *Optimizer* —Таким образом модель обновляется на основе данных, которые она видит, и функции потерь.

- *Metrics* —Используется для контроля этапов обучения и тестирования. В примере используется точность - доля правильно классифицированных изображений.

Loss function

$$MSE = \frac{1}{Y}\sum_{i=1}^{Y}(y_i - \hat{y}_i)^2$$ Mean Square Error

$$BCE = -\frac{1}{Y}\sum_{i=1}^{Y}[y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)]$$ Binary Cross-Entropy

$$CCE = -\sum_{c=1}^{C}\frac{1}{Y_c}\sum_{i:y_i=c} y_i * \log(\hat{y}_i)$$ Categorical Cross-Entropy

One-hot encoding

{1, 2, 3, 4} →

1	2	3	4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

?
0.2
0.5
0.1
0.9

https://www.tensorflow.org/api_docs/python/tf/keras/losses

Обучение модели: TensorFlow Pipeline

```
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()

train_images, test_images = train_images / 255.0, test_images / 255.0

model = tf.keras.models.Sequential()

model.add(layers.Flatten(input_shape=(28, 28)))

model.add(layers.Dense(16, activation='sigmoid'))
model.add(layers.Dense(16, activation='sigmoid'))
model.add(layers.Dense(10))

model.compile(optimizer='SDG',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                  metrics=['accuracy'])
history = model.fit(train_images, train_labels,
                    validation_data=(test_images, test_labels), epochs=10)
```

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Применение Обученной модели

```
prediction = model.predict(test_images)
```

SoftMax $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

4.80|1.15|-0.35|4.95|2.45|2.85

```
probability_model = tf.keras.Sequential([model,
                                         tf.keras.layers.Softmax()])

prediction = probability_model.predict(test_images)
```

0.411|0.011|0.002|0.478|0.039|0.059

```
label_predict=np.argmax(prediction,axis=1)
```

Что можно поменять

Количество слоев

Больше слоев – более сложные предсказания

Больше слоев – дольше считать производные

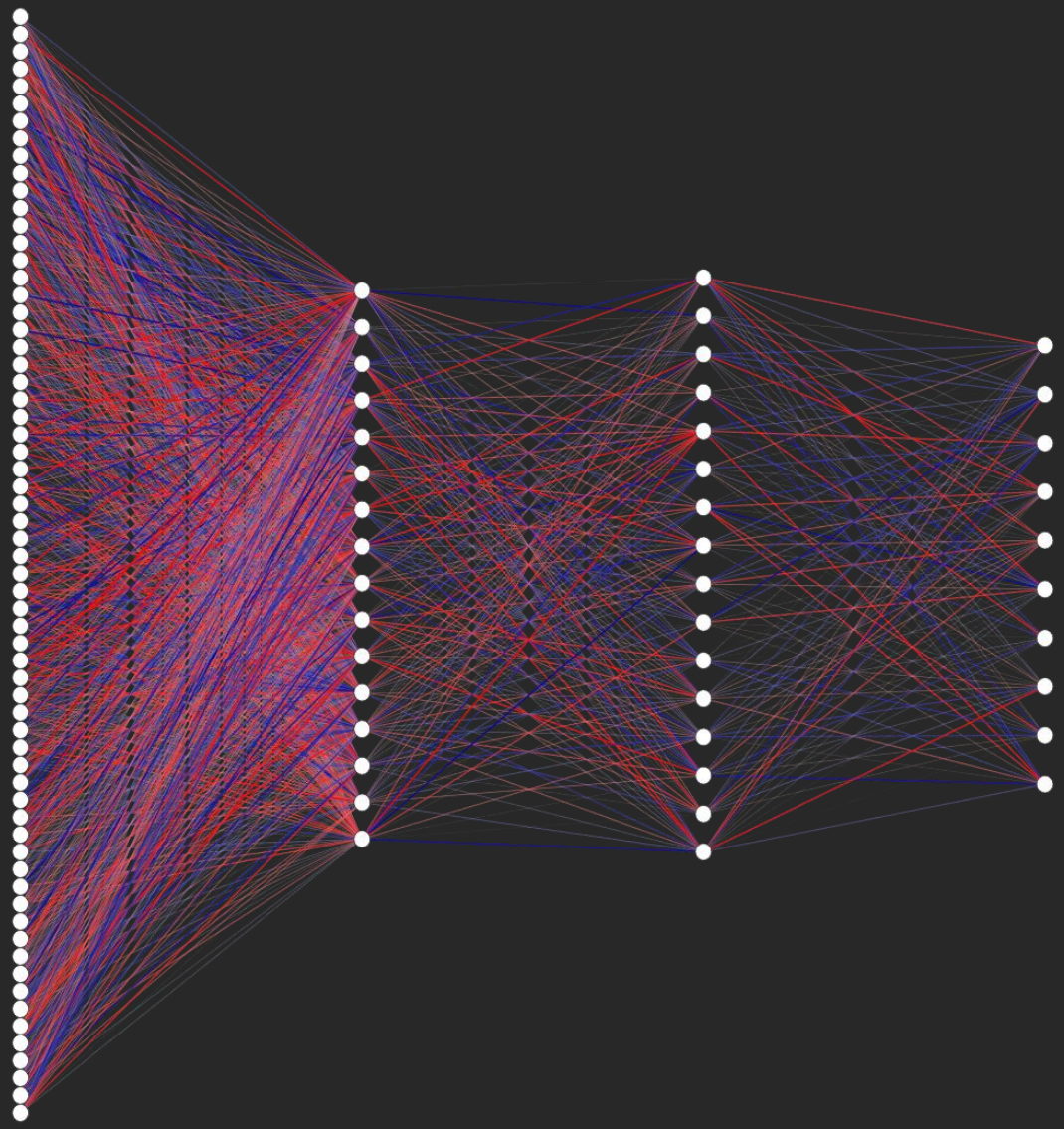
Количество нейронов в слое

----и-----

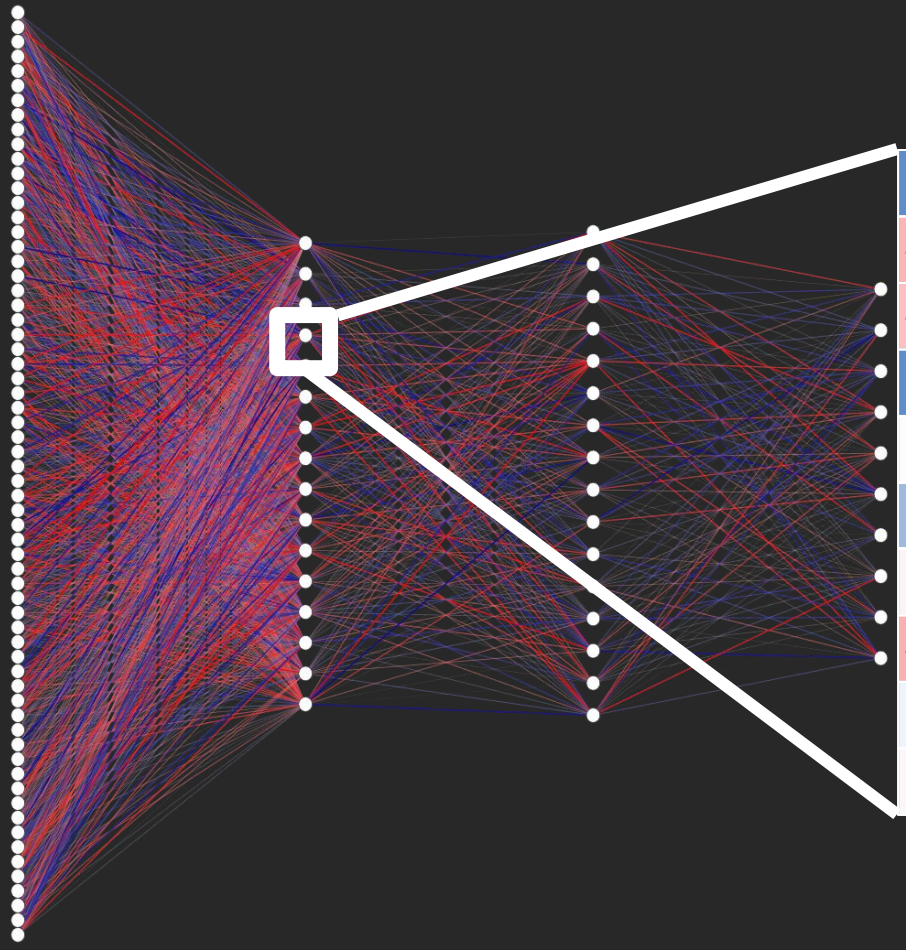
Тип функции активации

ReLU говорят неплох

Больше параметров требуется больше данных



Проблемы с Полносвязными Нейронными Сетями



0.98	0.61	-0.62	-0.81	0.38	0.76	-0.25	0.09	0.51	0.05
-0.42	-0.55	-0.78	-0.94	-0.63	0.28	-0.69	0.32	0.60	-1.00
-0.34	-0.35	0.96	0.96	0.05	0.10	-0.32	-0.17	0.07	0.31
0.97	0.49	-0.55	0.30	0.48	0.52	0.09	-0.68	-0.47	0.12
0.12	-0.15	-0.44	1.00	0.85	-0.29	0.50	-0.61	0.71	0.43
0.63	0.09	0.96	-0.96	0.73	0.48	0.93	0.04	-0.96	0.64
0.05	-0.98	0.99	-1.00	0.56	0.83	-0.15	0.57	0.64	0.87
-0.46	-0.44	0.83	-0.48	-0.42	0.49	0.53	0.11	0.56	-0.76
0.19	0.53	0.97	0.19	0.13	-0.87	-0.30	0.30	-0.87	0.38
0.07	0.84	-0.56	0.09	0.64	-0.86	0.25	0.51	-0.27	-0.55

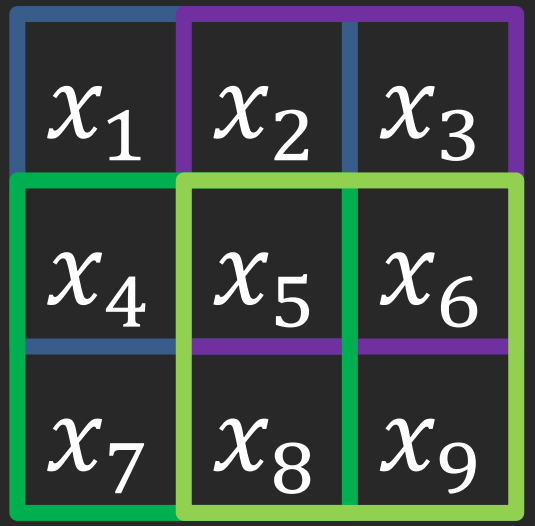
Веса

0.66	0.80	0.85	0.51	0.96	0.57	0.54	0.53	0.31	0.04
0.67	0.47	0.12	0.30	0.51	0.25	0.03	0.61	0.32	0.43
0.04	0.25	0.35	0.61	0.32	0.38	0.33	0.09	0.76	0.89
0.10	0.09	0.07	0.80	0.23	0.83	0.99	0.48	0.14	0.29
0.71	0.78	0.09	0.11	0.01	0.35	0.10	0.63	0.75	0.99
0.83	0.26	0.54	0.75	0.44	0.03	0.25	0.72	0.70	0.54
0.48	0.19	0.21	0.25	0.54	0.83	0.63	0.43	0.95	0.34
0.09	0.14	0.06	0.43	0.80	0.84	0.85	0.45	0.35	0.10
0.97	0.74	0.96	0.90	0.43	0.36	0.99	1.00	0.18	0.00
0.41	0.53	0.50	0.74	0.17	0.84	0.89	0.43	0.00	0.44

эта Пять!

Сверточные Нейронные Сети

Convolution Neural Networks



$\otimes$

$$y_1 = x_1 * w_{1,1} + x_2 * w_{1,2} + x_4 * w_{2,1} + x_5 * w_{2,2}$$
$$y_2 = x_2 * w_{1,1} + x_3 * w_{1,2} + x_5 * w_{2,1} + x_6 * w_{2,2}$$
$$y_3 = x_4 * w_{1,1} + x_5 * w_{1,2} + x_7 * w_{2,1} + x_8 * w_{2,2}$$
$$y_4 = x_5 * w_{1,1} + x_6 * w_{1,2} + x_8 * w_{2,1} + x_9 * w_{2,2}$$

$$ReLU\left(\begin{matrix} y_1 & y_2 \\ y_3 & y_4 \end{matrix} + b\right)$$

```
layers.Conv2D(10, (3, 3), activation='relu')
```

Backpropagation Для Сверток

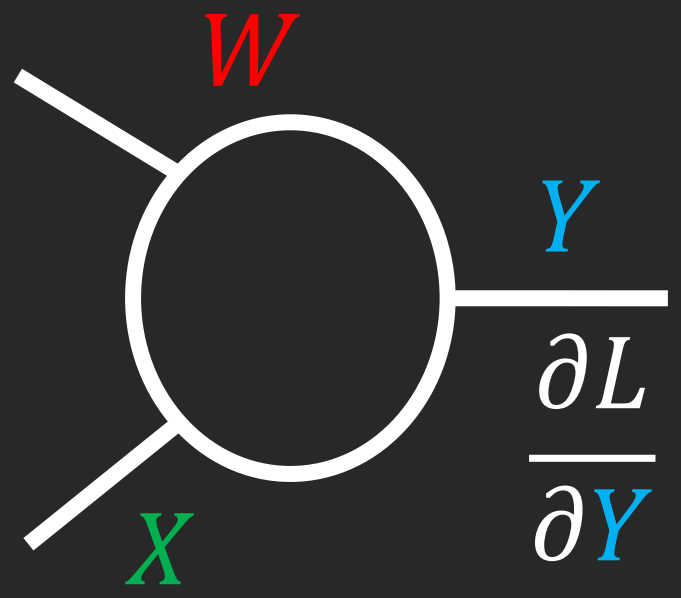
x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

$\otimes$

$\frac{\partial L}{\partial y_1}$	$\frac{\partial L}{\partial y_2}$
$\frac{\partial L}{\partial y_3}$	$\frac{\partial L}{\partial y_4}$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial W}$$

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial X}$$



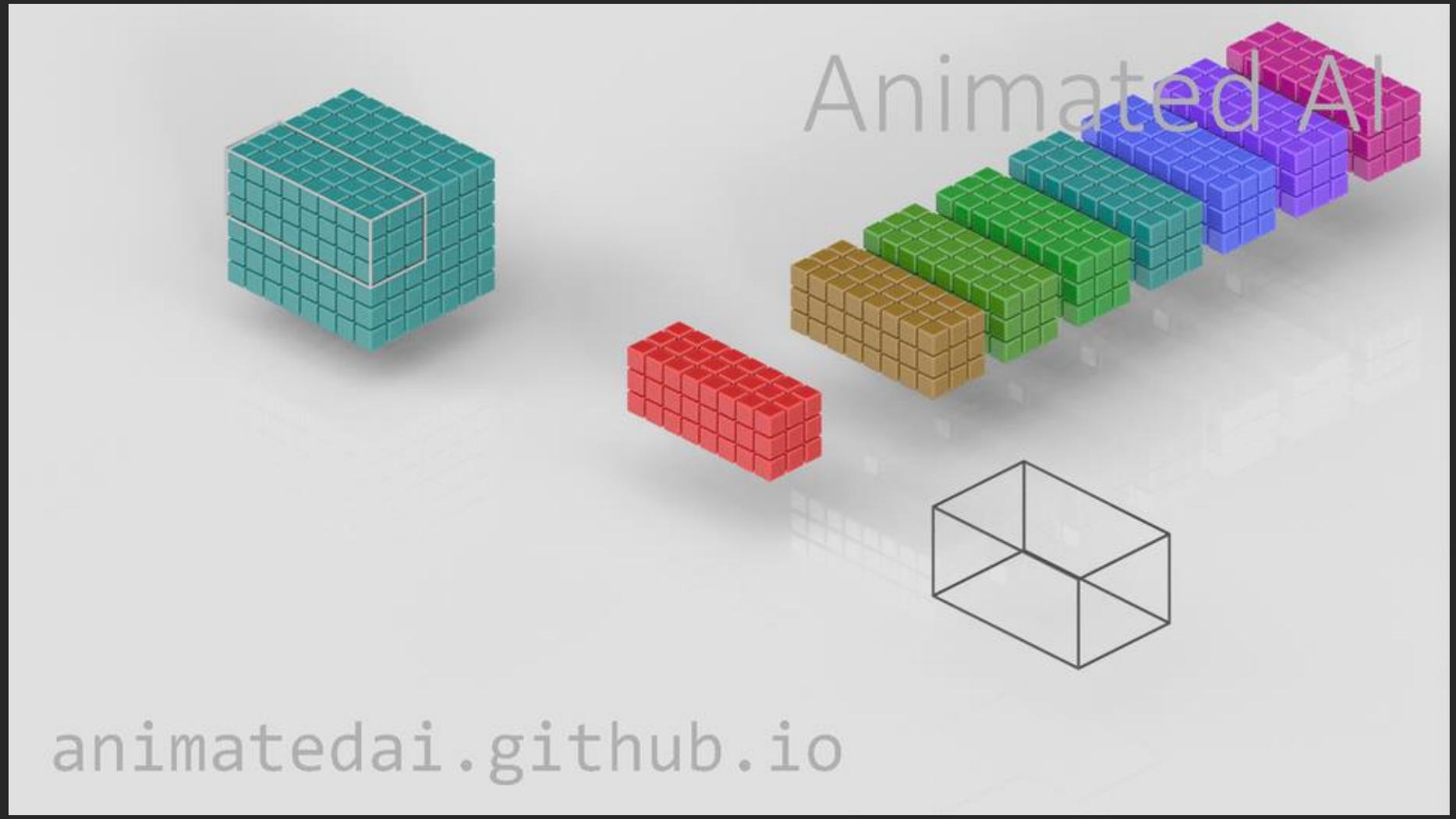
$w_{2,2}$	$w_{2,1}$
$w_{1,2}$	$w_{1,1}$

$\otimes$

$\frac{\partial L}{\partial y_1}$	$\frac{\partial L}{\partial y_2}$
$\frac{\partial L}{\partial y_3}$	$\frac{\partial L}{\partial y_4}$

Полная Свертка

Свертки в GIF



https://animatedai.github.io/

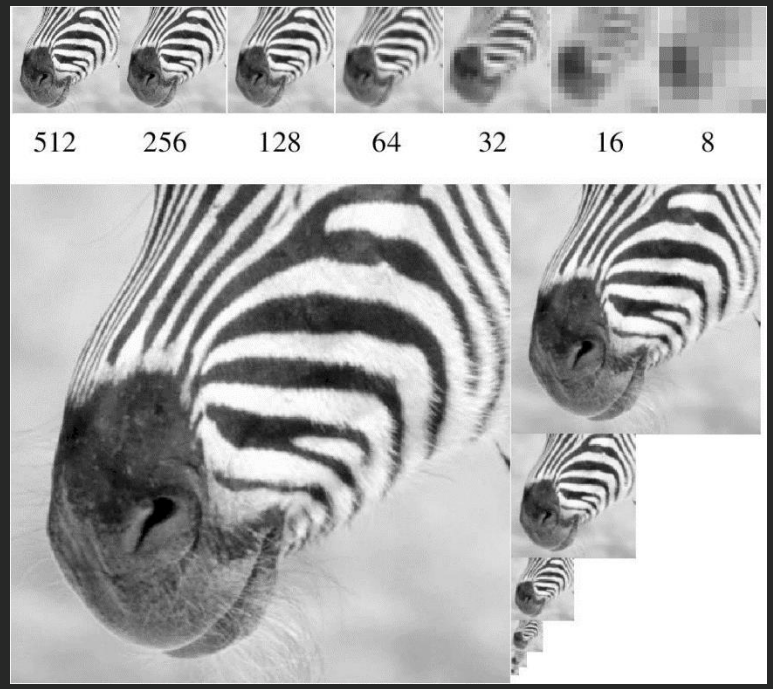
Pooling

MaxPooling

4	42	220	43
16	176	120	104
156	253	120	177
190	103	99	41

176	220
253	177

176	220
253	177



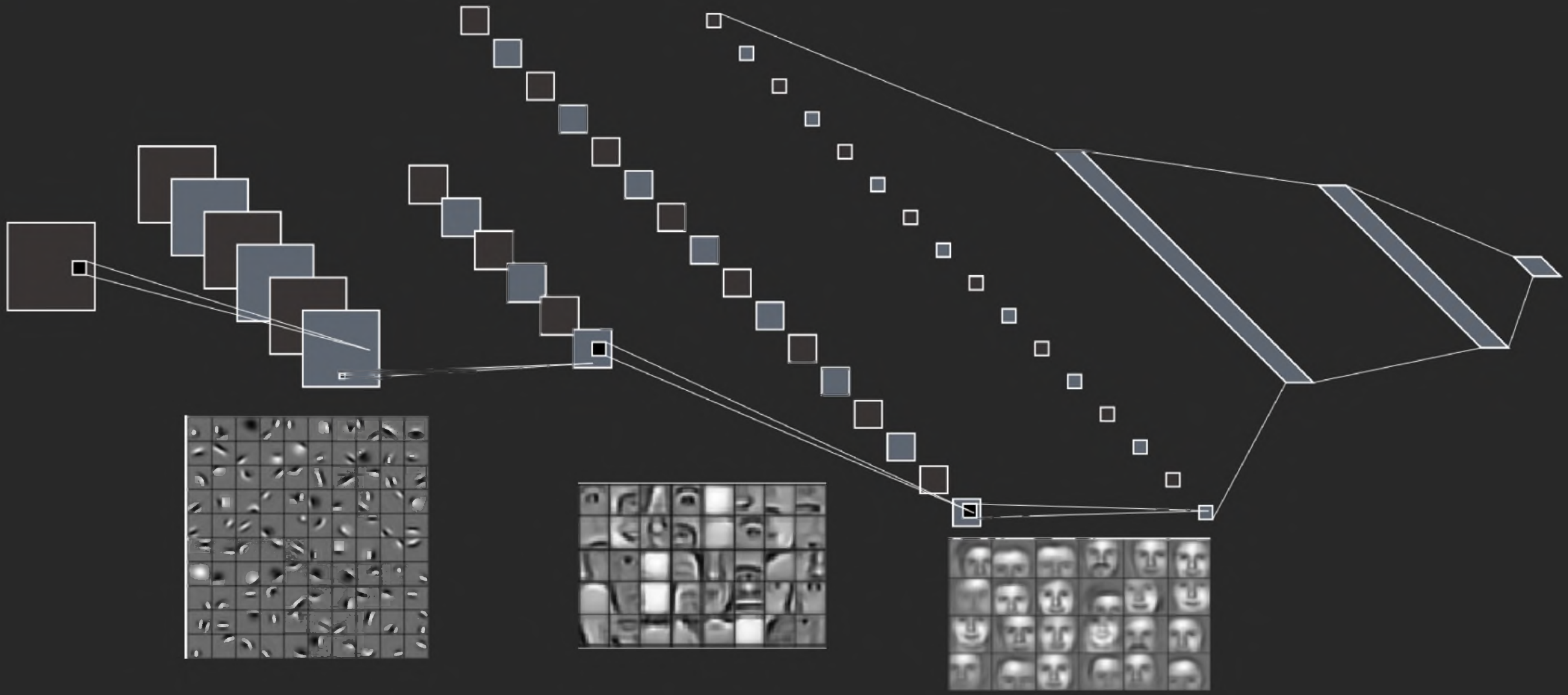
AveragePooling

59	121
175	109

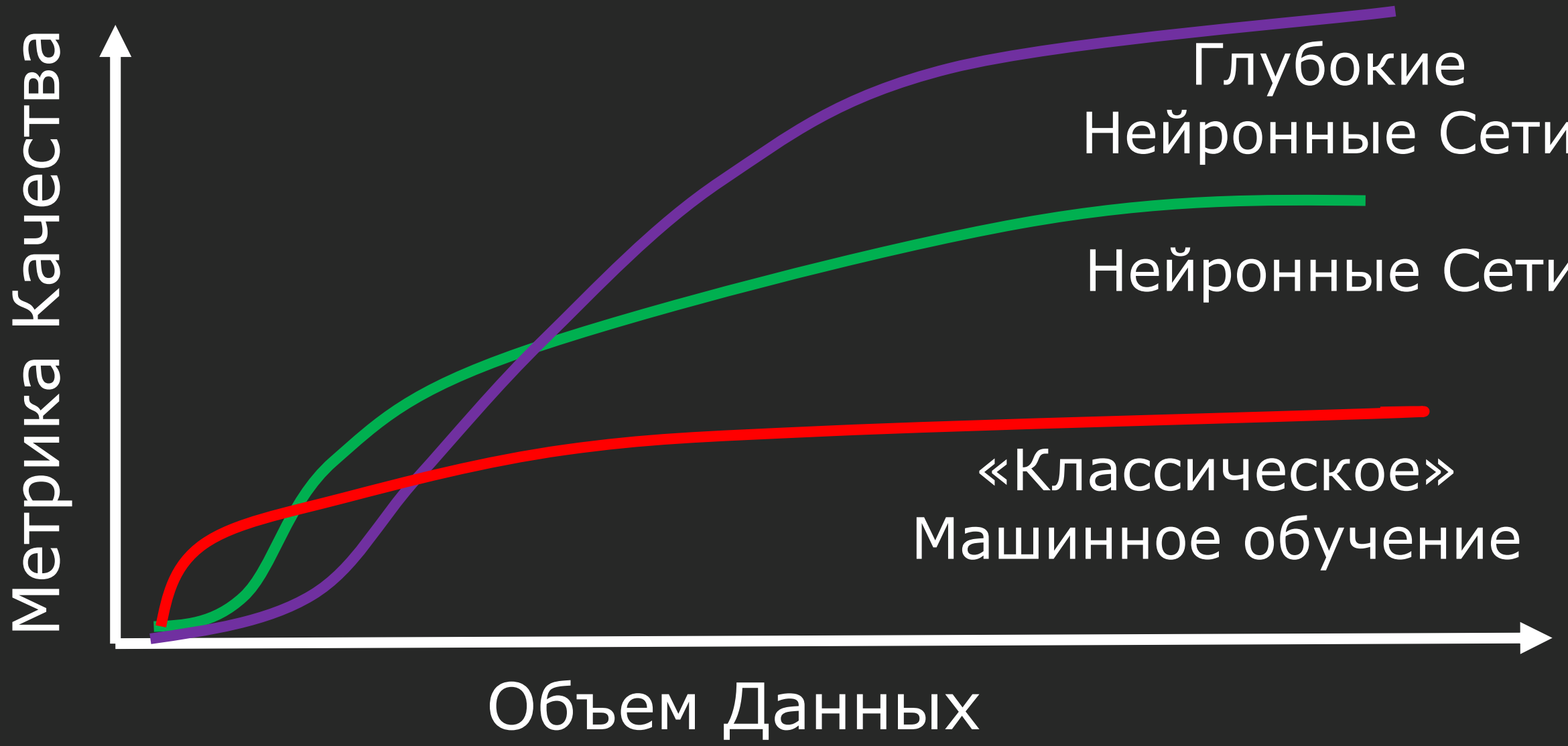
59	121
175	109

```
kernel_size = (2,2)

stride = (2,2)
```

Сверточные Нейронные Сети



Data Volume vs Methods



Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Резюме

Про изображения

- Картинки – не таблички
- Простые картинки сводятся к табличкам
- Свертки дают разную информацию в зависимости от весов
- Old School - ищем углы на изображениях
- Old School – признаки на фильтрах Хаара

Про нейронки

- Отдаем инженерию признаков на откуп модели
- Нейроны, Weights и Bias, Функции Активации
- Градиентный спуск (стохастический – скармливаем данные по batch)
- Полносвязные нейронки «Хорошо запоминают»
- Сверточные нейронки «Обучаемые» Фильтры «Хорошо извлекают Признаки»
- Pooling – для «уменьшение» размера
- ВСЕГО ЛИШЬ ОЧЕНЬ СЛОЖНАЯ ФУНКЦИЯ (Матричное умножение+нелинейные функции)

Вопросы, пожелания, предложения

?????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.08
Где мы изучаем историю свёрточных нейронных сетей для классификации изображений

Докладчик
Долганов Антон

В предыдущей лекции

Оптимизаторы

- Экспоненциально Взвешенные Средние
- Градиентный Спуск с Моментом, RMSprop, Adam

Регуляризация

- L1, L2
- Dropout
- Data Augmentation

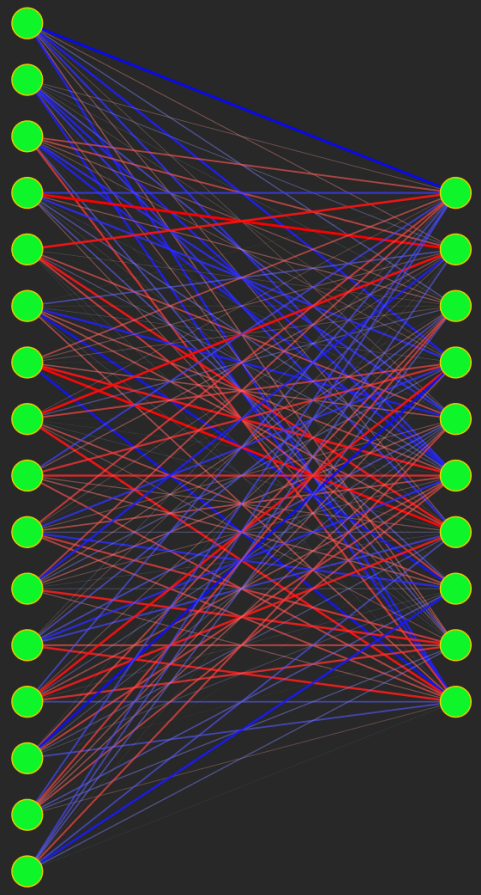
Convolutional Neural Networks/Сверточные нейронные сети

- Фильтры
- Padding и Stride / Заполнение и Шаг
- Pooling
- «Хорошо извлекают Признаки»

Нейронные Сети: Слои

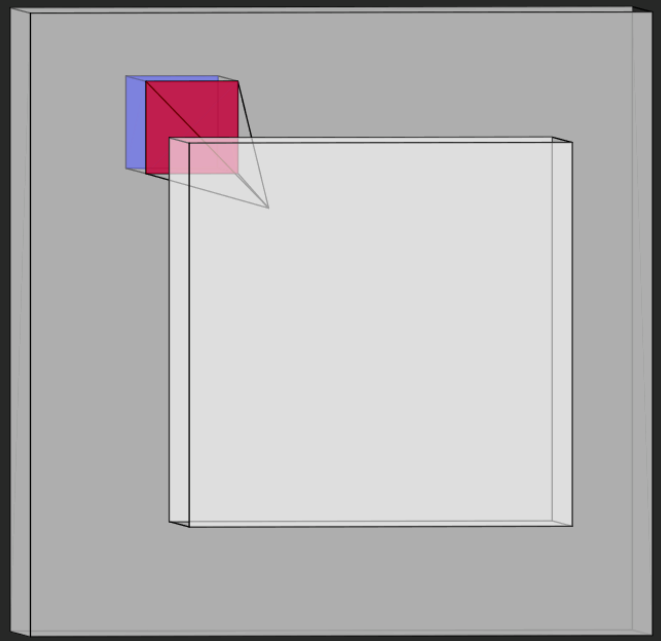
```
from tensorflow.keras import layers
```

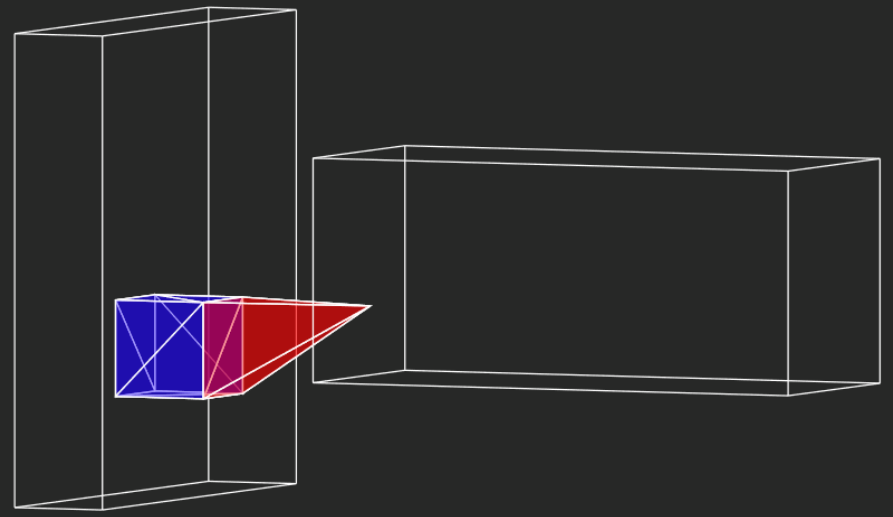
```
layers.Dense(32, activation='relu')
```



```
layers.Conv2D(64, (3, 3), activation='relu')
```

```
layers.MaxPooling2D((2, 2))
```





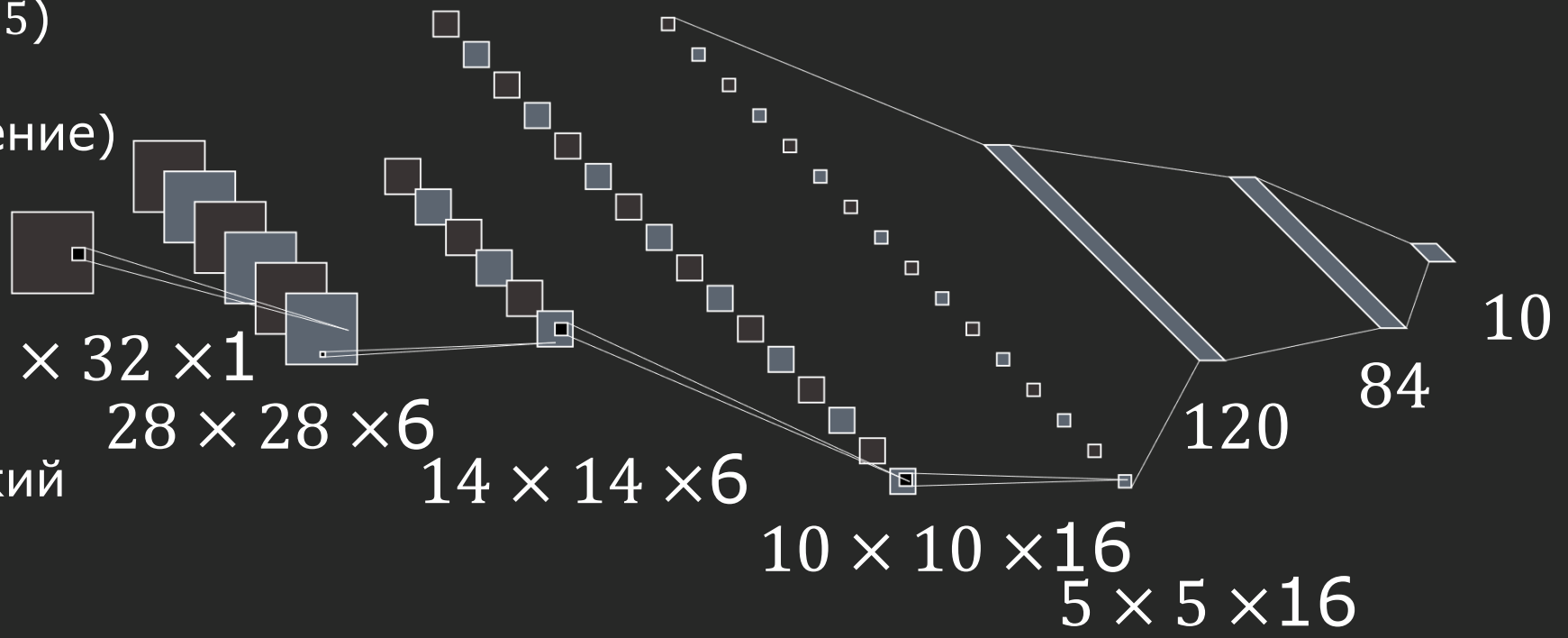
Содержание

Содержание

LeNet-5

- 2 сверточных слоя (5×5)
- 2 слоя pooling (усреднение)
- 3 полносвязных слоя

➢ Функции активации
- Сигмоид
- Тангенс Гиперболический



~60 тыс. параметров

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. (December 1989). "Backpropagation Applied to Handwritten Zip Code Recognition". *Neural Computation*. **1** (4): 541–551.

ImageNet

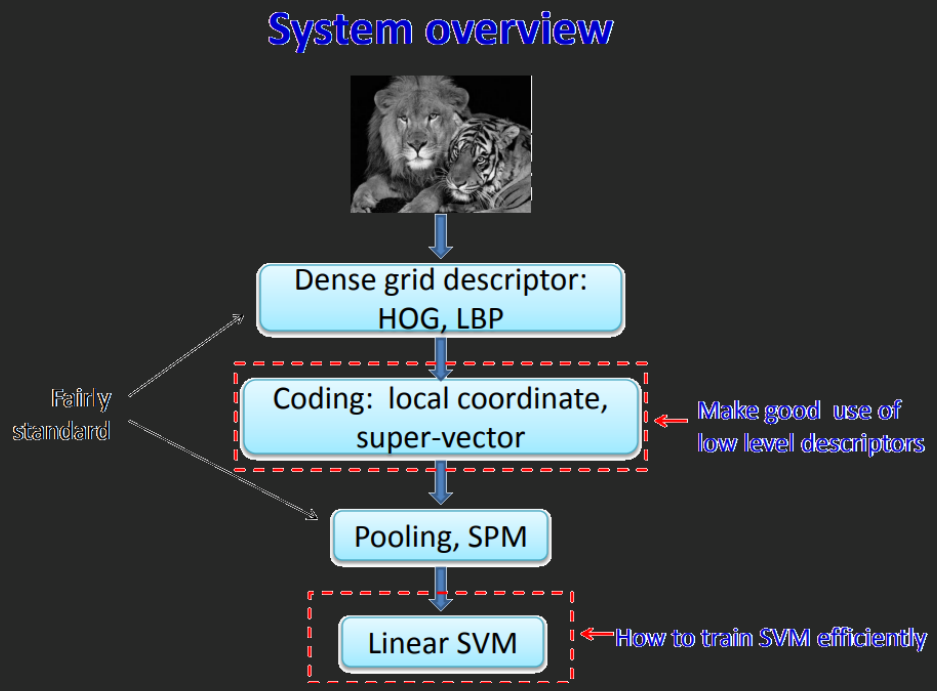
- Всего изображений : >14 000 000
- Изображений с тренировочными рамками: > 1 000 000
- Всего подклассов : >21000

Высокое Разрешение (а не эти ваши 28*28 и 32*32)




http://www.image-net.org/

ImageNet Large Scale Visual Recognition Challenge



Ranking by learning methods

- Descriptor Coding + SVM, 0.28 --- NEC-UIUC
- Fisher kernel + SVM, 0.34 --- XRCE
- LI2C, 0.58 --- NTU_WZX
- KNN, 0.61 --- LIG
- Canonical Correlation Analysis, 0.79 -- NII

https://image-net.org/challenges/LSVRC/2011/

Содержание

LeNet-5

AlexNet

VGG

GoogLeNet

ResNet

MobileNet

AlexNet

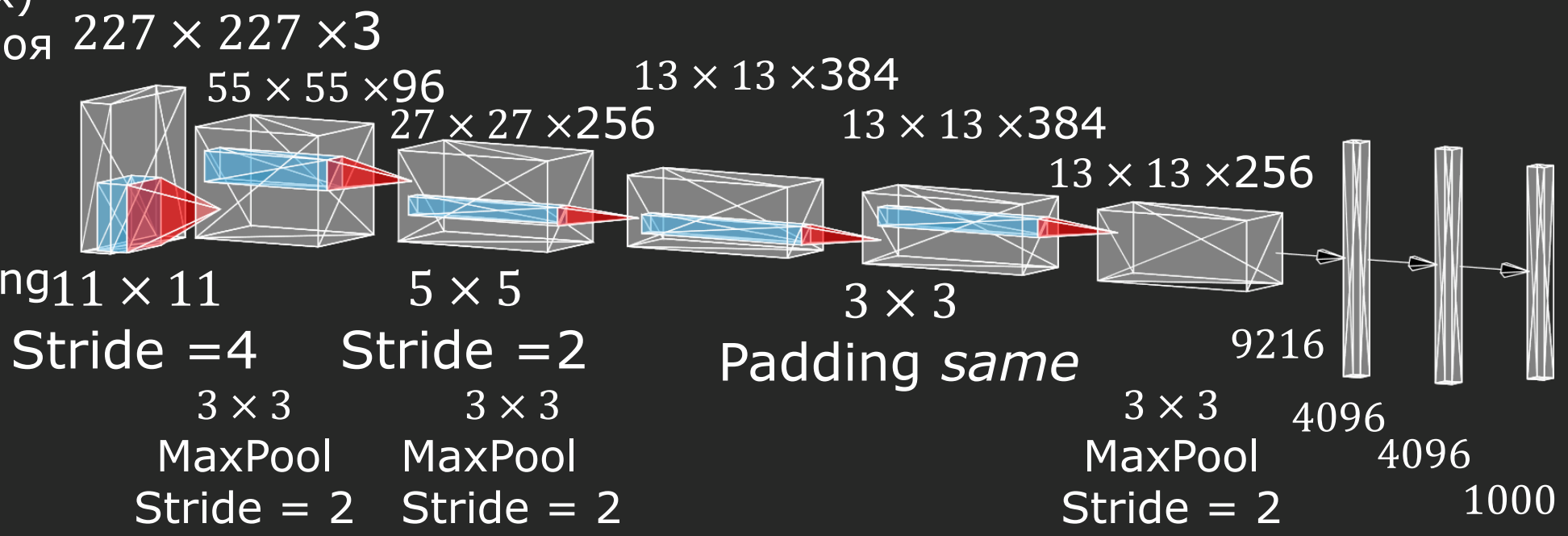
- 5 сверточных слоев
- 3 pooling слоя (Max)
- 3 полносвязных слоя

- Padding
- Stride

➢ Overlapping pooling

✓ ReLU
✓ Dropout

❖ GPU



60 миллионов параметров

AlexNet | ILSVRC Competition – 2012 (*Winner*) | Top-5 Error Rate – 16.42%

Содержание

LeNet-5

AlexNet

VGG

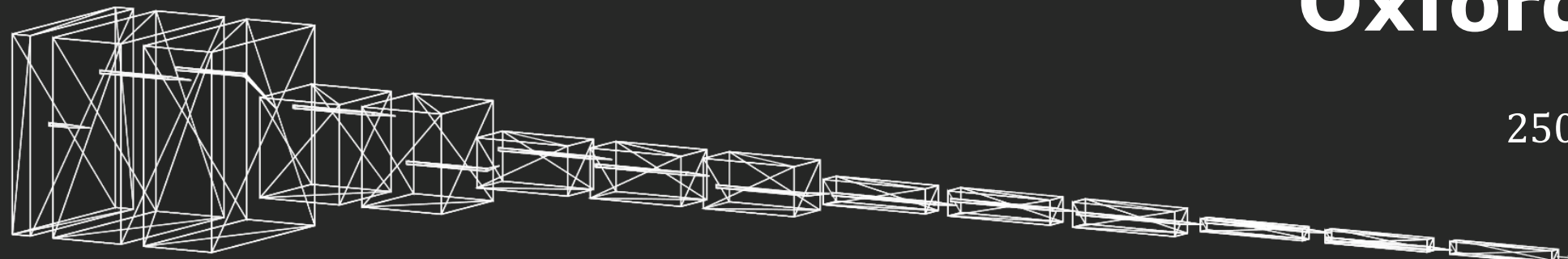
GoogLeNet

ResNet

MobileNet

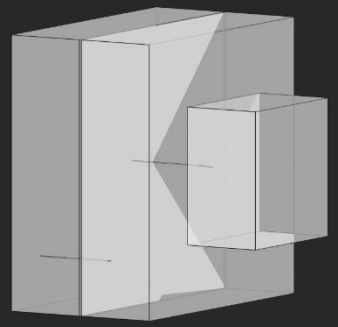
VGG Visual Geometry Group

Oxford



- Только свертки 3 x 3
  Padding same
- Max Pooling 2 x 2
- Dropout

138+ миллионов параметров



VGG-16 | ILSVRC Competition – 2014 (*Runners-Up*) | Top-5 Error Rate – 7.3%

Содержание

GoogLeNet или Inception v1

In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed Inception, which derives its name from the Network in network paper by Lin et al [12] in conjunction with the famous "we need to go deeper" internet meme [1]. In our case, the word "deep" is used in two different meanings: first of all, in the sense that we introduce a new level of organization in the form of the "Inception module" and also in the more direct sense of increased network depth. In general, one can view the Inception model as a logical culmination of [12] while taking inspiration and guidance from the theoretical work by Arora et al [2]. The benefits of the architecture are experimentally verified on the ILSVRC 2014 classification and detection challenges, on which it significantly outperforms the current state of the art.

Duerig and Ning Ye for their help on photometric distortions. Also our work would not have been possible without the support of Chuck Rosenberg and Hartwig Adam.

References

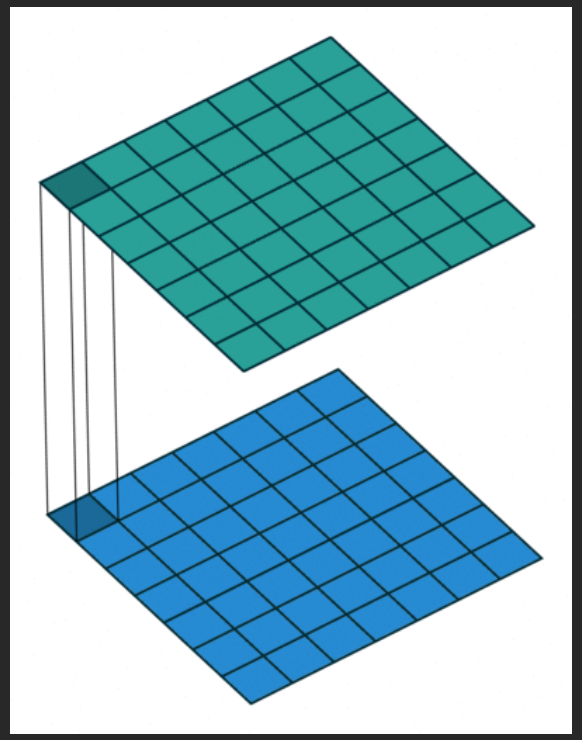
[1] Know your meme: We need to go deeper. http://knowyourmeme.com/memes/we-need-to-go-deeper. Accessed: 2014-09-15.

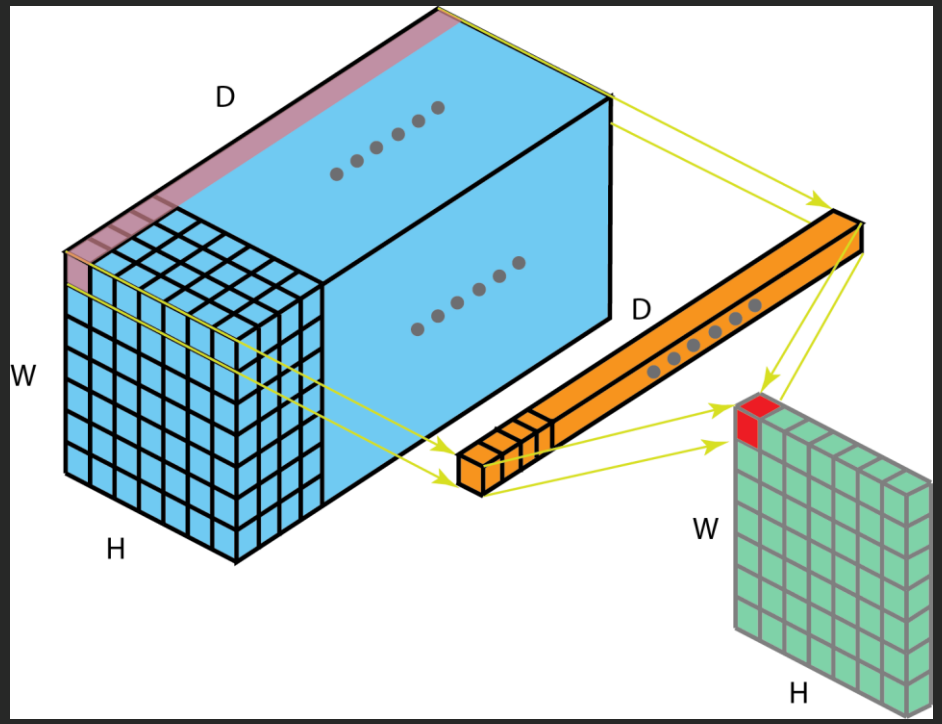
[2] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. *CoRR*, abs/1310.6343, 2013.

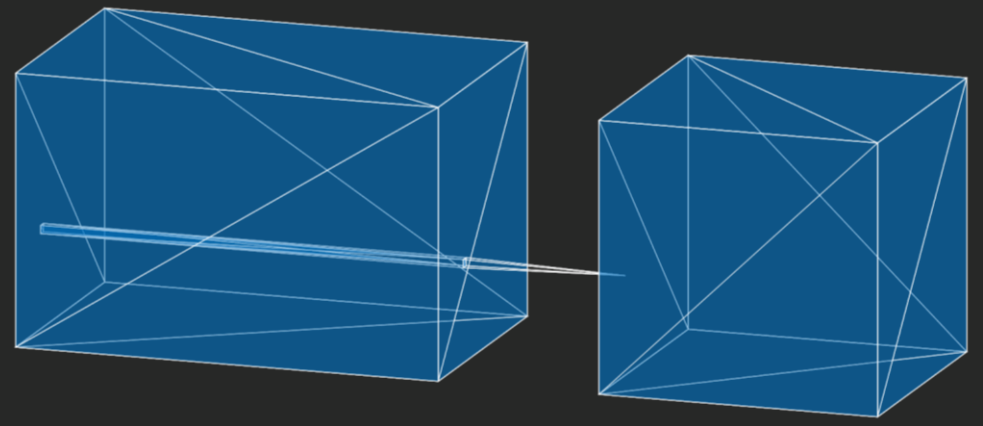
https://arxiv.org/pdf/1409.4842.pdf

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.

Свертка 1 x 1







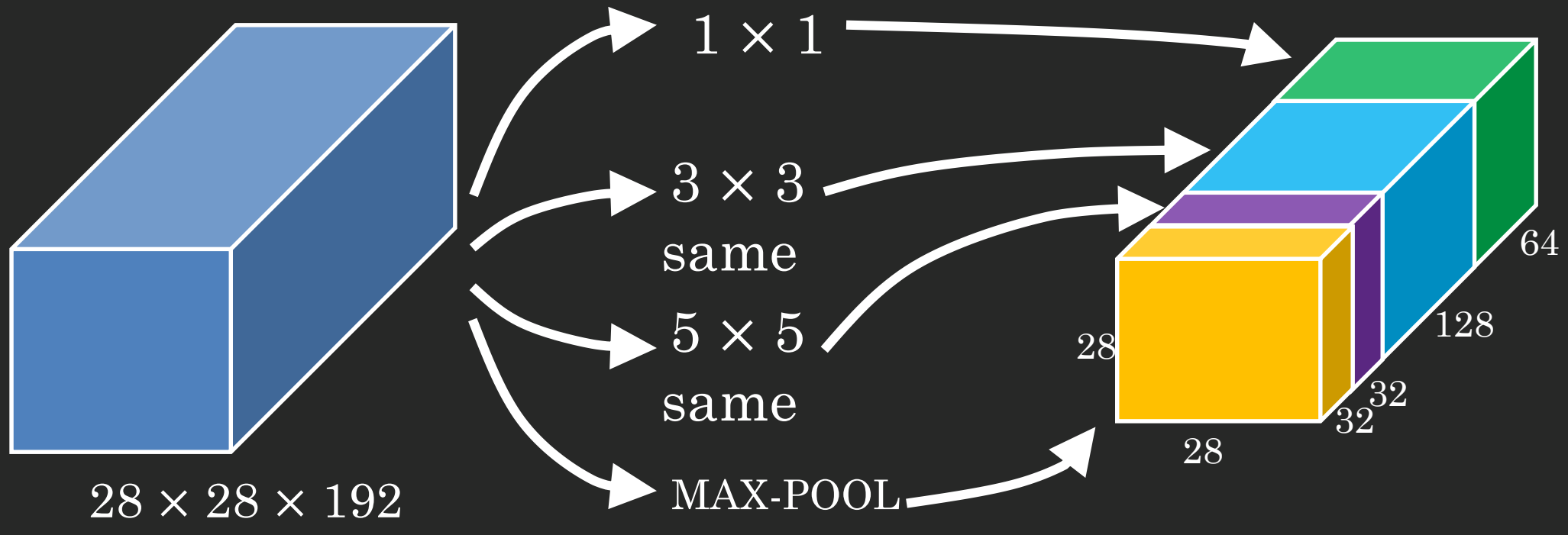
$28 \times 28 \times 256$ $28 \times 28 \times 32$

$32 : 1 \times 1 \times 256$

Свертка не по пространству, а по «каналам»

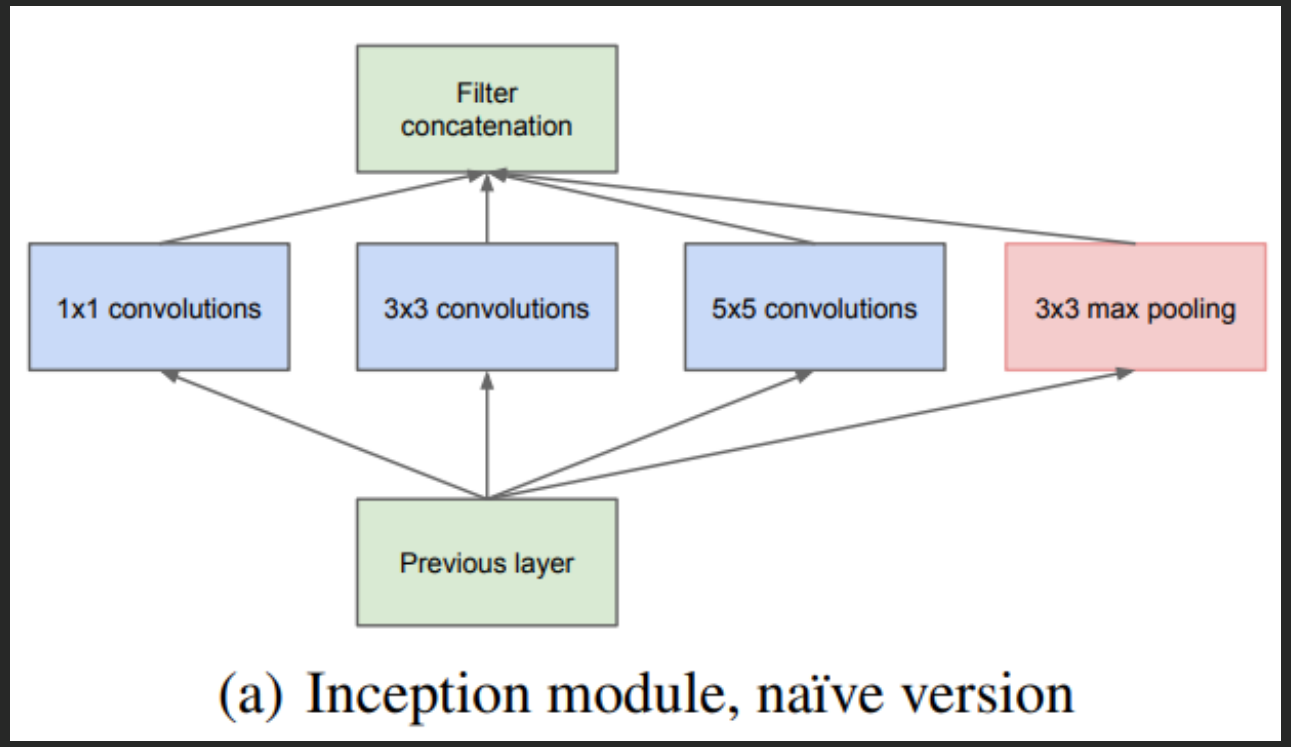
Networks in Networks / Сети внутри Сетей

“Зачем выбирать размер фильтра?”

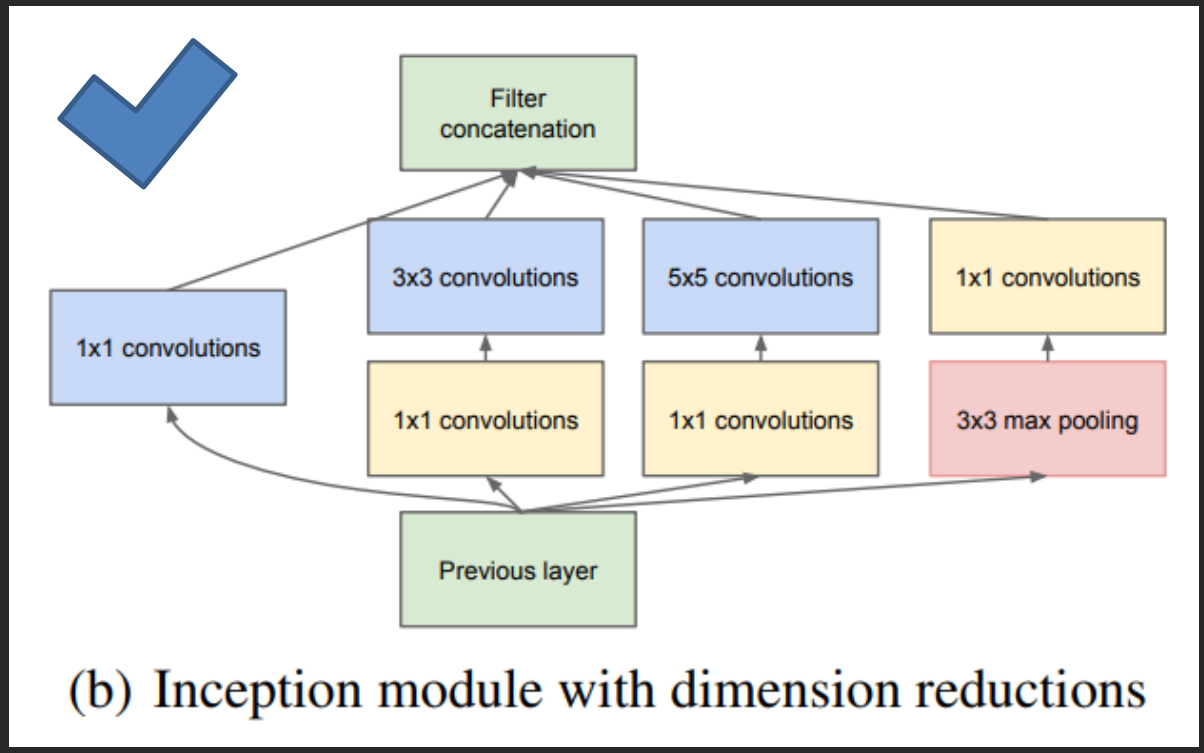


https://www.coursera.org/learn/convolutional-neural-networks

Inception Модули



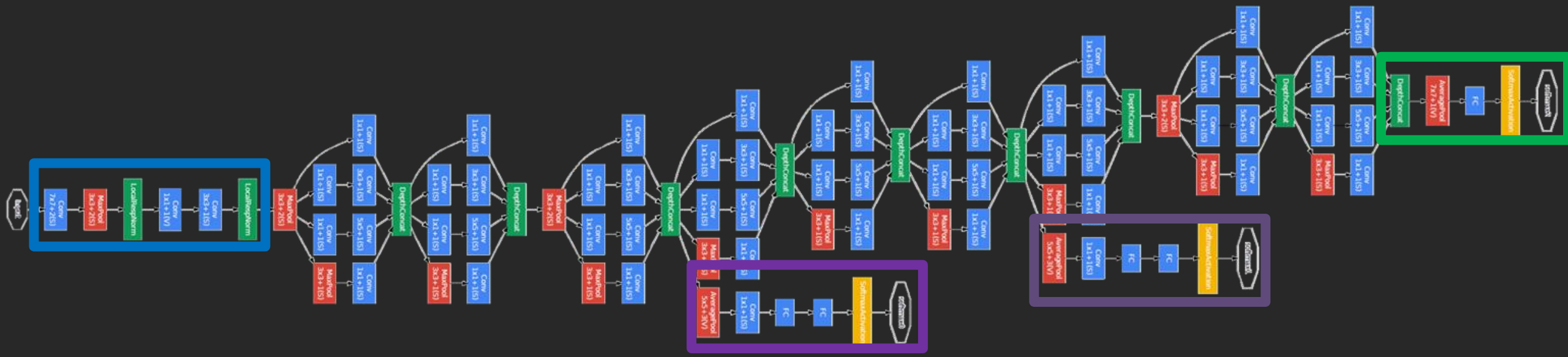
(a) Inception module, naïve version



(b) Inception module with dimension reductions

https://arxiv.org/pdf/1409.4842.pdf

GoogLeNet или Inception v1



22 слоя

- Стандартные слои Свертки + pooling
- Inception Слои
- Вспомогательные классификаторы (auxiliary)

$$total\_loss = real\_loss + 0.3 * aux\_loss\_1 + 0.3 * aux\_loss\_2$$

6.8 миллионов параметров

Inception V1 (GoogLeNet) | ILSVRC Competition – 2014 (Winner) | Top-5 Error Rate – 6.67%

Идеи для улучшения

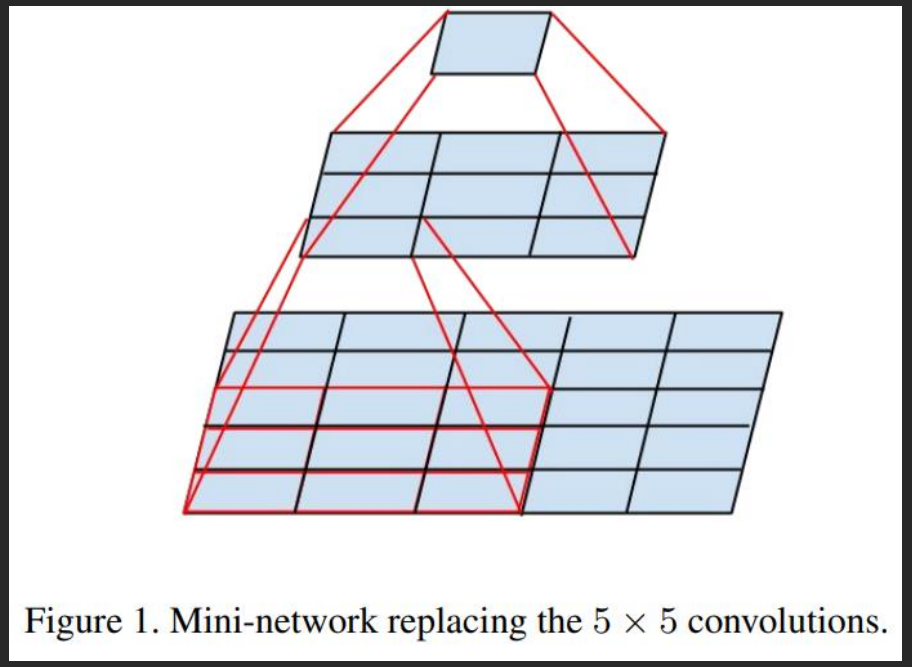


Figure 1. Mini-network replacing the 5×5 convolutions.

Свертки 5x5 в **2.78 раз более «дорогие»** чем свертки 3x3

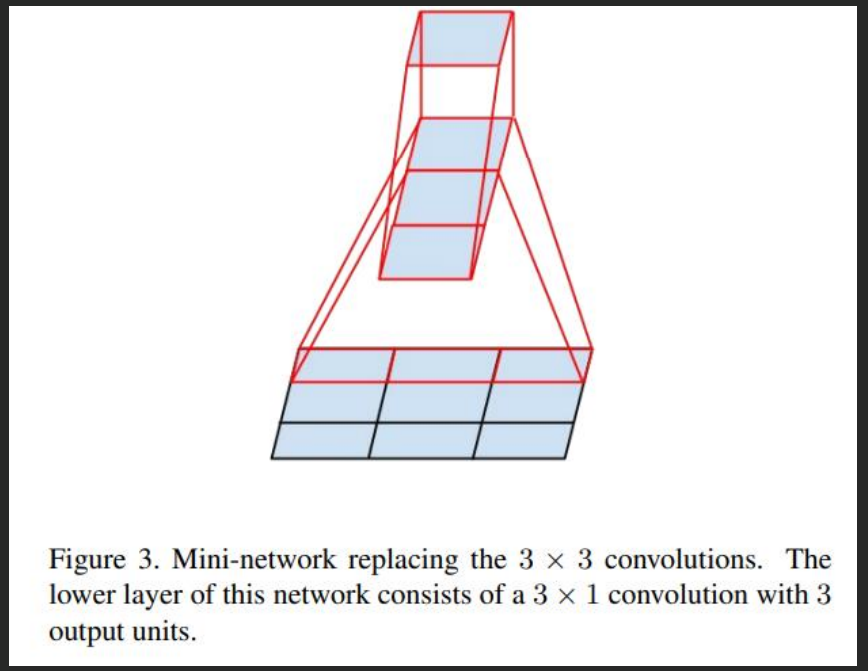


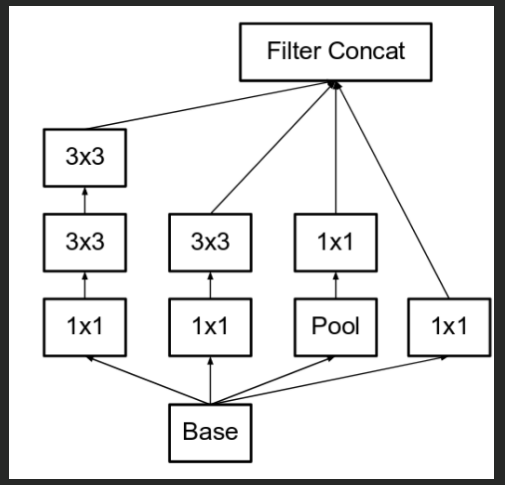
Figure 3. Mini-network replacing the 3×3 convolutions. The lower layer of this network consists of a 3×1 convolution with 3 output units.

На 33% дешевле чем просто свертка 3x3

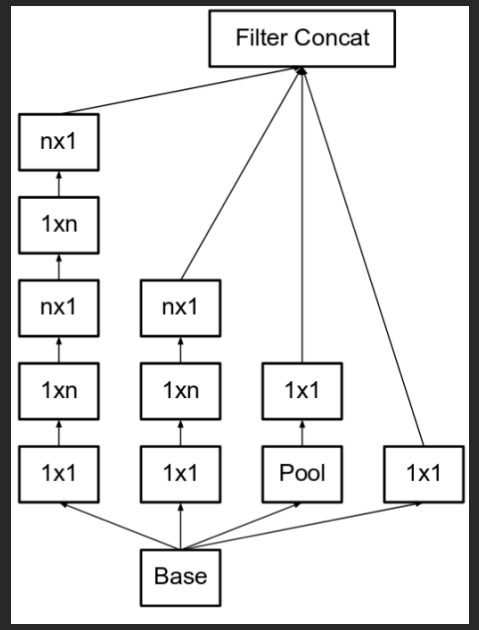
https://arxiv.org/pdf/1512.00567v3.pdf

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

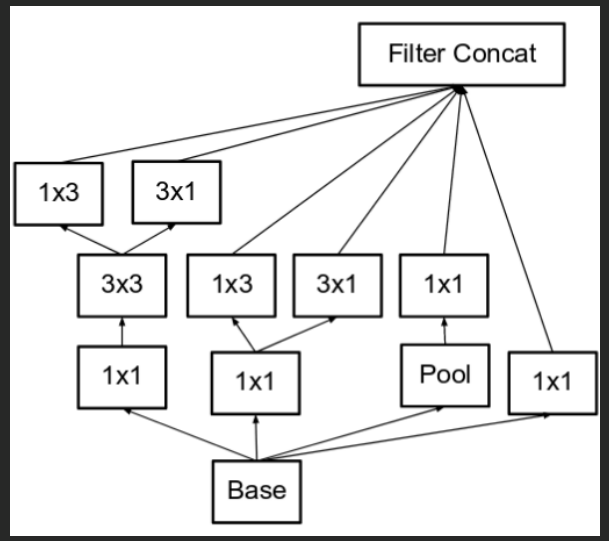
Inception v2



A: свертка 5x5 как две свертки 3x3.



B: свертка 5x5 как две свертки 3x3, которые как 1x3 и 3x1 последовательно.



C: Делаем модуль Inception шире.

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	A As in figure 5	35×35×288
5×Inception	As in B gure 6	17×17×768
2×Inception	As in figure C 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

42 слоя, но всего лишь в 2,5 раза более затратно по вычислительным ресурсам

https://arxiv.org/pdf/1512.00567v3.pdf

Inception v3

Inception v2 +

- RMSProp
- По «умному» считаем свертки 7x7
- BatchNorm во вспомогательных классификаторах
- Label Smoothing

BatchNorm

Стандартизируем выходы скрытых слоев на мини-выборке

Label Smoothing

$$q'(k|x) = (1-\varepsilon)\delta_{k,y} + \frac{\varepsilon}{K}$$

23.8+ миллиона параметров

Inception v3 | ILSVRC Competition – 2015 (Runners-Up) | Top-5 Error Rate > 3.57% (<4%)

https://arxiv.org/pdf/1512.00567v3.pdf

BatchNorm

```
tf.keras.layers.BatchNormalization(
    axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True,
    beta_initializer='zeros', gamma_initializer='ones',
    moving_mean_initializer='zeros',
    moving_variance_initializer='ones', beta_regularizer=None,
    gamma_regularizer=None, beta_constraint=None,
gamma_constraint=None, **kwargs
)
```

При обучении $$batch_{norm} = gamma * \frac{(batch - mean(batch))}{\sqrt{var(batch) + epsilon}} + beta$$

При использовании

$$moving\_mean = moving\_mean * momentum + mean(batch) * (1 - momentum)$$

$$moving\_var = moving\_var * momentum + var(batch) * (1 - momentum)$$

https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

Содержание

Вспомним про Градиенты

$\nabla L(w_0, b_0, w_1, b_1, w_2, b_2)$

Ошибка $\rightarrow L_0=(x^{(N)}-y)^2$

$$x^{(N)} = \sigma(w_N\, x^{(N-1)} + b_N) = \sigma(z^{(N)})$$

$$z^{(N)} = w_N\, x^{(N-1)} + b_N$$

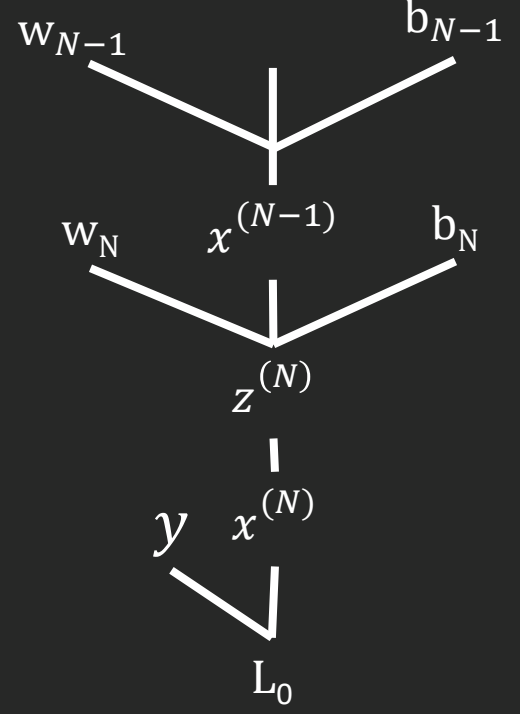
$$\frac{\partial L_0}{\partial w_N} = \frac{\partial z^{(N)}}{\partial w_N}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial L_0}{\partial x^{(N)}} = x^{(N-1)}\sigma'(z^{(N)})\, 2(x^{(N)}-y)$$

$$\frac{\partial L_0}{\partial x^{(N)}} = 2(x^{(N)}-y) \qquad \frac{\partial x^{(N)}}{\partial z^{(N)}} = \sigma'(z^{(N)}) \qquad \frac{\partial z^{(N)}}{\partial w_N} = x^{(N-1)}$$

$$\frac{\partial L_0}{\partial b_N} = \frac{\partial z^{(N)}}{\partial b_N}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial L_0}{\partial x^{(N)}} = \sigma'(z^{(N)})2(x^{(N)}-y)$$

$$\frac{\partial L_0}{\partial x^{(N-1)}} = \frac{\partial z^{(N)}}{\partial x^{(N-1)}}\frac{\partial x^{(N)}}{\partial z^{(N)}}\frac{\partial L_0}{\partial x^{(N)}} = w_N\sigma'(z^{(N)})2(x^{(N)}-y)$$

backpropagation



Затухающие Градиенты Vanishing Gradient Problem

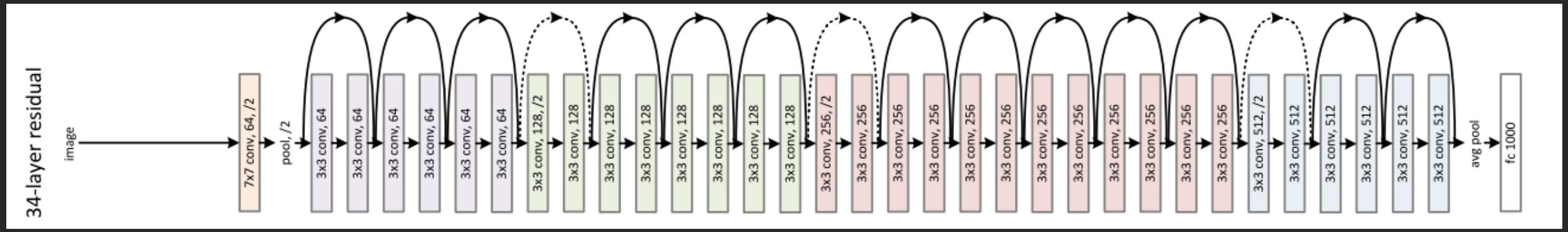
$$0.9^{100} \sim 0.00002656139$$

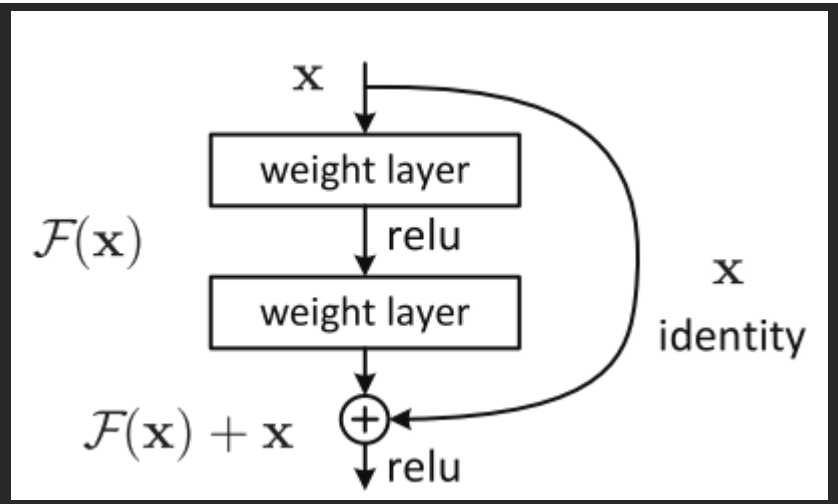
Взрывающиеся Градиенты Exploding Gradient Problem

$$1.1^{100} \sim 13780.6123398$$

ResNet Residual Networks

Microsoft Research





остаточные блоки и пропуски слоев

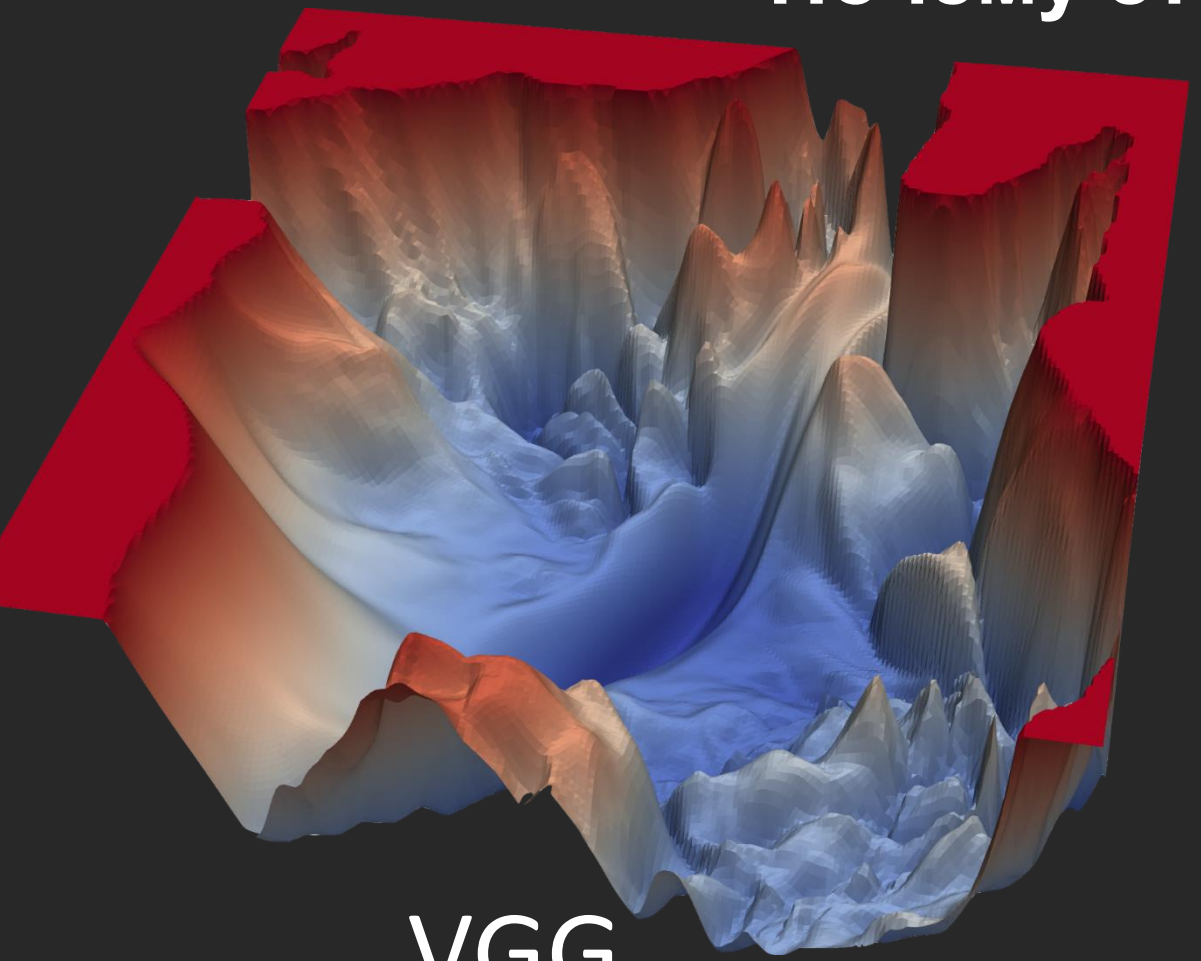
(34/50/101/152) слоя

23/25 миллиона параметров

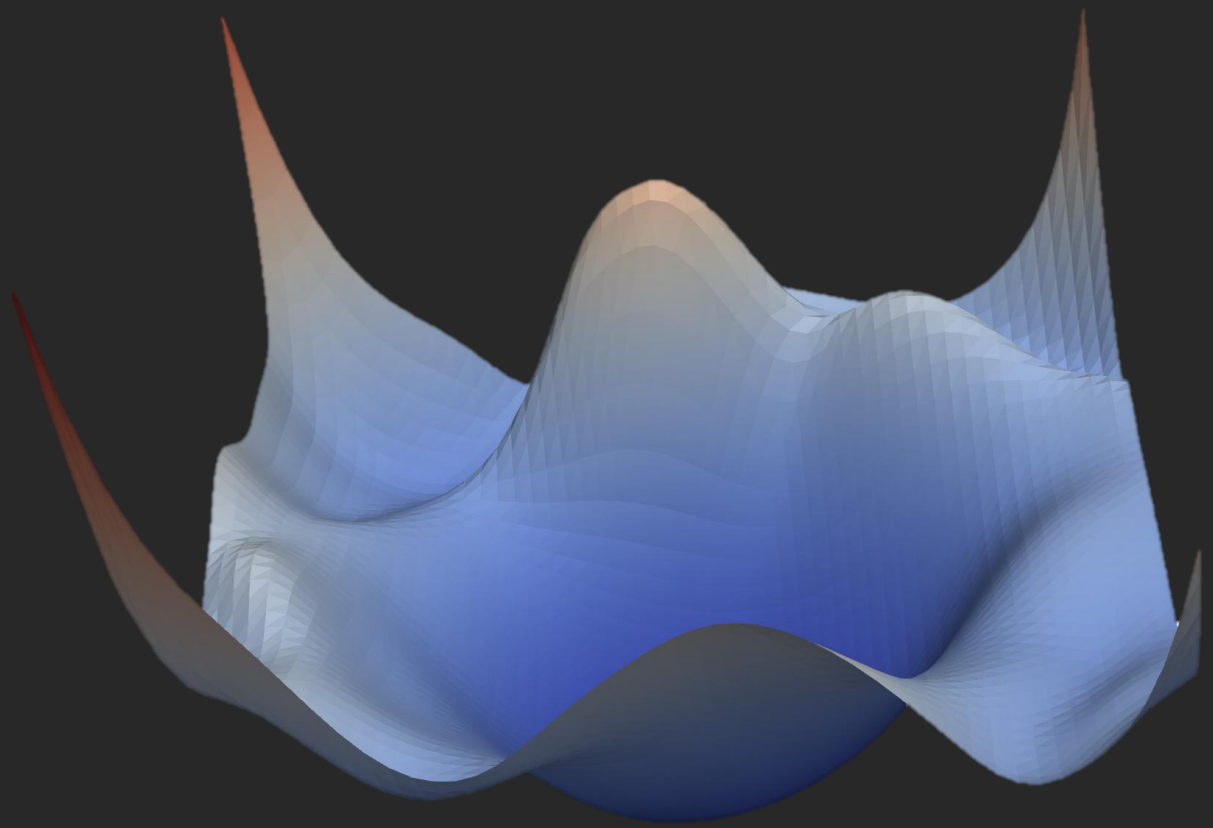
He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

ResNet | ILSVRC Competition – 2015 (Winner) | Top-5 Error Rate – 3.56%

Почему это работает?



VGG



ResNet

https://www.cs.umd.edu/~tomg/projects/landscapes/

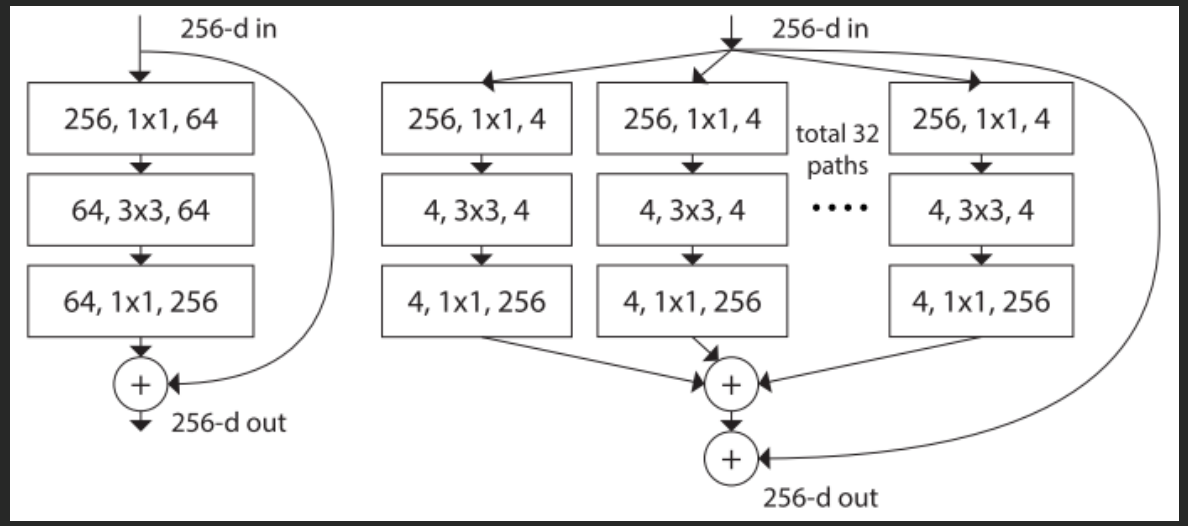
Пример Реализации в TensorFlow

```
tf.keras.layers.Add()

input1 = tf.keras.layers.Input(shape=(16,))
x1 = tf.keras.layers.Dense(8, activation='relu')(input1)

input2 = tf.keras.layers.Input(shape=(32,))
x2 = tf.keras.layers.Dense(8, activation='relu')(input2)

added = tf.keras.layers.Add()([x1, x2])

out = tf.keras.layers.Dense(4)(added)

model = tf.keras.models.Model(inputs=[input1, input2], outputs=out)
```

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Add

Развитие ResNet

ResNeXt

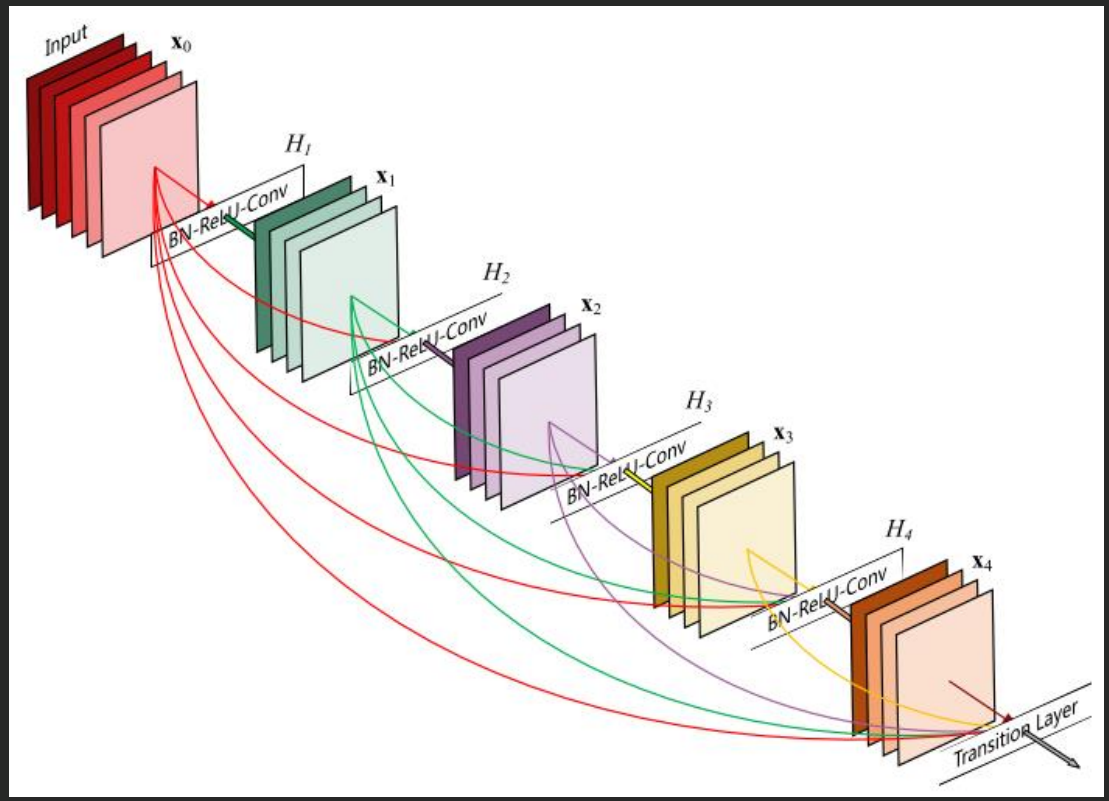


ResNeXt-10 | ILSVRC Competition – 2016 (Runners Up) | Top-5 Error Rate – 3.03%

Проиграли ансамблю других моделей командой Trimps-Soushen (2.99%)

- Inception v3
- Inception v4
- Inception-Resnet-v2
- Wide ResNet

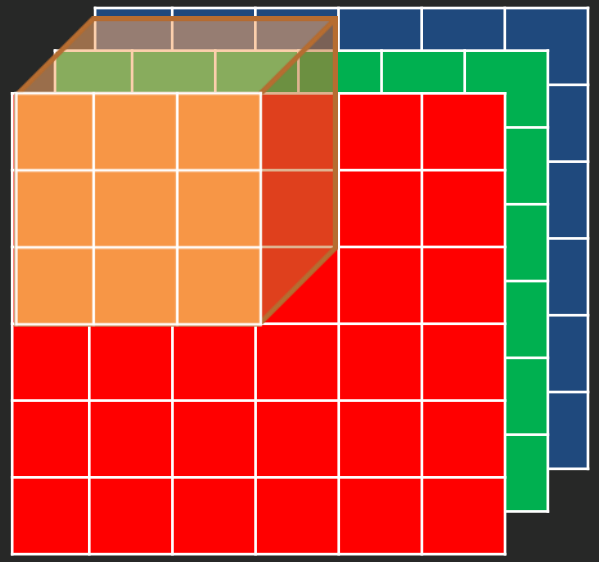
DenseNet



Содержание

Что не может пойти не так? Итог

Изображение

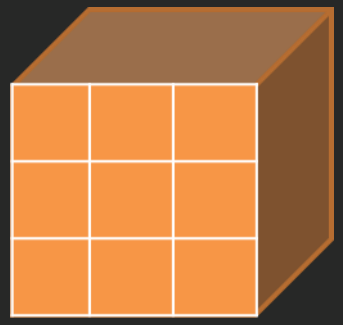


$6 \times 6 \times 3$

$n_{in} \times m_{in} \times \mathrm{n_c}$

$\otimes$

Фильтр

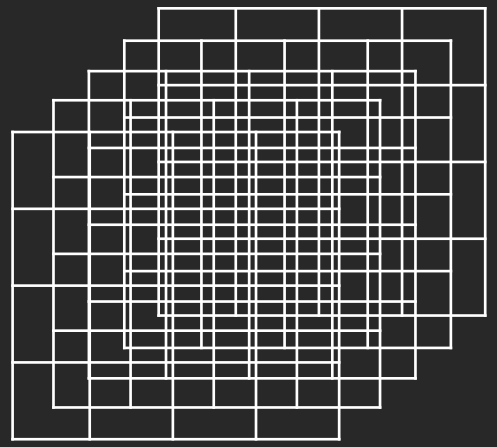


$\times\, \mathrm{n_f}$

$\times\, 5$

$3 \times 3 \times 3$

$f \times f \times \mathrm{n_c}$



$4 \times 4 \times 5$

$n_{out} \times m_{out} \times \mathrm{n_f}$

Количество вычислений

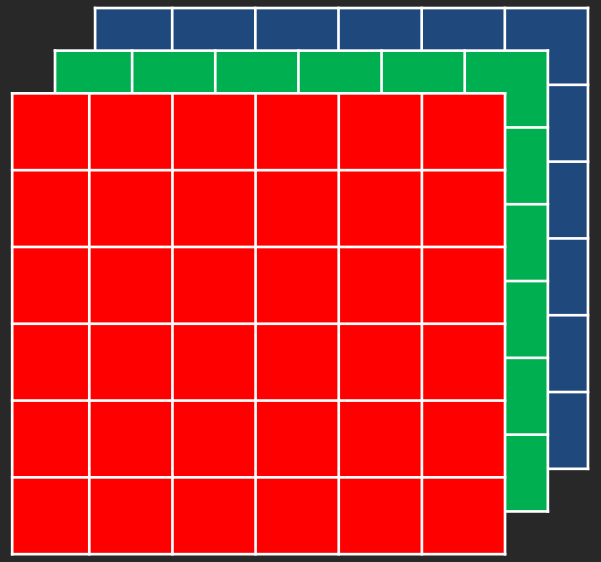
#параметров фильтра × #позиций фильтра × #фильтров

$f \times f \times \mathrm{n_c} \times n_{out} \times m_{out} \times \mathrm{n_f}$

$3 \times 3 \times 3 \times 4 \times 4 \times 5 = 2160$

Depthwise Свертка

Изображение

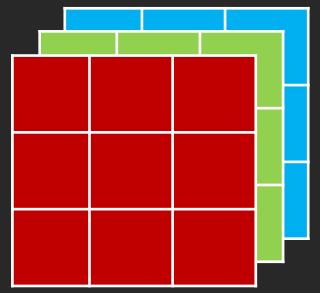


$6 \times 6 \times 3$

$n_{in} \times m_{in} \times \mathrm{n_c}$

$\otimes$

Фильтры



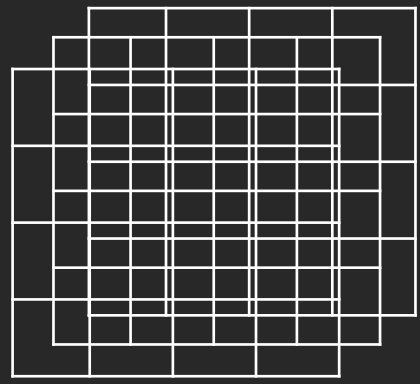
$\times \mathrm{n_c}$

$\times 3$

3×3

$f \times f$

Результат



$4 \times 4 \times 3$

$n_{out} \times m_{out} \times \mathrm{n_c}$

Количество вычислений

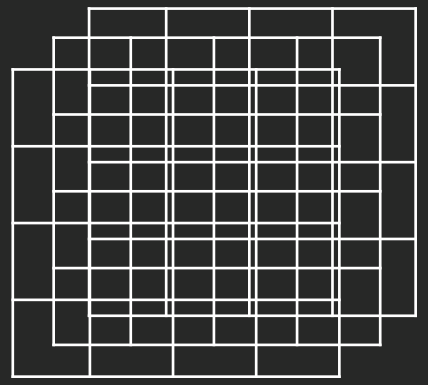
#параметров фильтра × #позиций фильтра × #фильтров

$f \times f \times n_{out} \times m_{out} \times \mathrm{n_c}$

$3 \times 3 \times 4 \times 4 \times 3 = 432$

Pointwise Свертка

Результат



$4 \times 4 \times 3$

$n_{out} \times m_{out} \times \mathrm{n_c}$

Фильтры

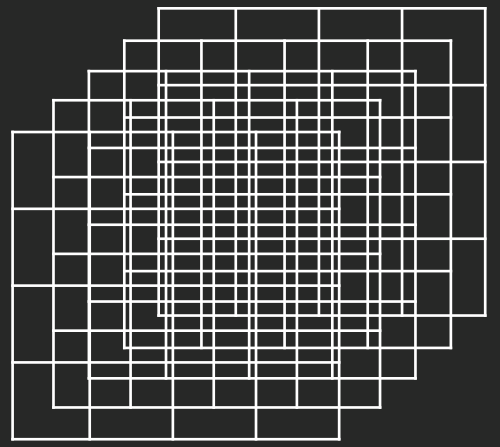
$\times \mathrm{n}_f$

$\times 5$

$1 \times 1 \times 3$

$f' \times f' \times \mathrm{n_c}$

Итог



$4 \times 4 \times 5$

$n_{out} \times m_{out} \times \mathrm{n_f}$

Количество вычислений

#параметров фильтра × #позиций фильтра × #фильтров

$$f' \times f' \times \mathrm{n_c} \times n_{out} \times m_{out} \times \mathrm{n_f}$$

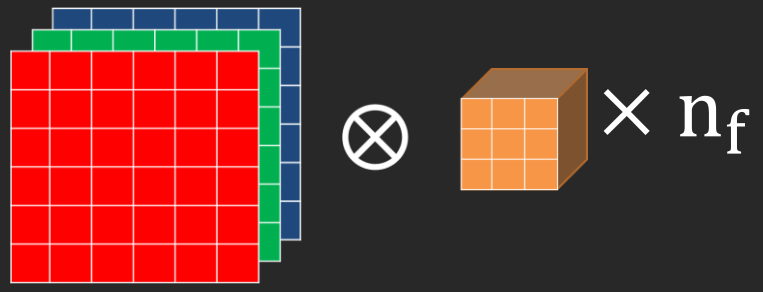
$$1 \times 1 \times 3 \times 4 \times 4 \times 5 = 240$$

Сравнение вычислений

Простая Свертка

Количество вычислений

$$f \times f \times \mathrm{n_c} \times n_{out} \times m_{out} \times \mathrm{n_f} \qquad 2160$$



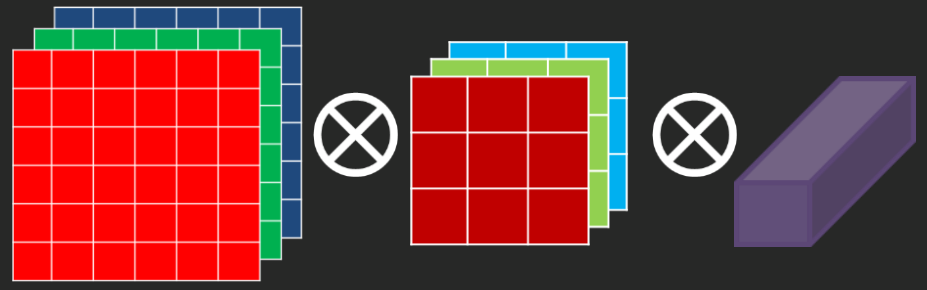
Depthwise seperable Свертка (Depthwise+Pointwise)

Количество вычислений

$$f \times f \times n_{out} \times m_{out} \times \mathrm{n_c} \qquad 432$$

$$f' \times f' \times \mathrm{n_c} \times n_{out} \times m_{out} \times \mathrm{n_f} \qquad 240$$

$$672$$



В общем случае “Выигрыш”

$$\frac{1}{n_f} + \frac{1}{f^2}$$

$$\frac{672}{2160} \sim 0.31$$

Пример Реализации в TensorFlow

```
tf.keras.layers.DepthwiseConv2D(
    kernel_size, strides=(1, 1), padding='valid', depth_multiplier=1,
    data_format=None, dilation_rate=(1, 1), activation=None, use_bias=True,
    depthwise_initializer='glorot_uniform',
    bias_initializer='zeros', depthwise_regularizer=None,
    bias_regularizer=None, activity_regularizer=None, depthwise_constraint=None,
    bias_constraint=None, **kwargs
)
```

```
tf.keras.layers.SeparableConv2D(
    filters, kernel_size, strides=(1, 1), padding='valid',
    data_format=None, dilation_rate=(1, 1), depth_multiplier=1, activation=None,
    use_bias=True, depthwise_initializer='glorot_uniform',
    pointwise_initializer='glorot_uniform',
    bias_initializer='zeros', depthwise_regularizer=None,
    pointwise_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    depthwise_constraint=None, pointwise_constraint=None, bias_constraint=None,
    **kwargs
)
```

https://www.tensorflow.org/api_docs/python/tf/keras/layers/DepthwiseConv2D
https://www.tensorflow.org/api_docs/python/tf/keras/layers/SeparableConv2D

MobileNet v1

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

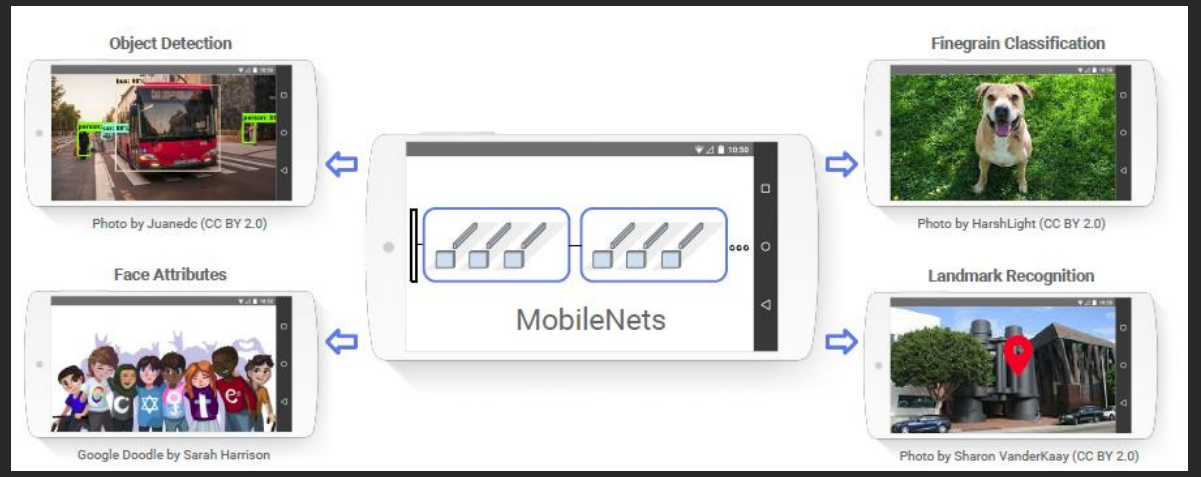
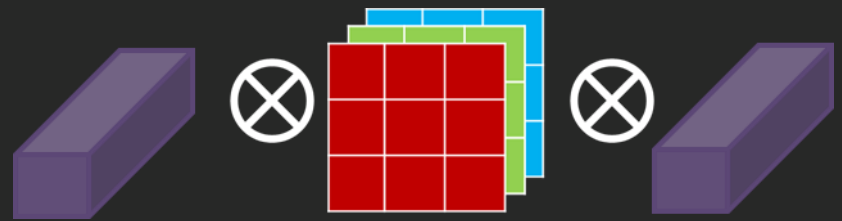


Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

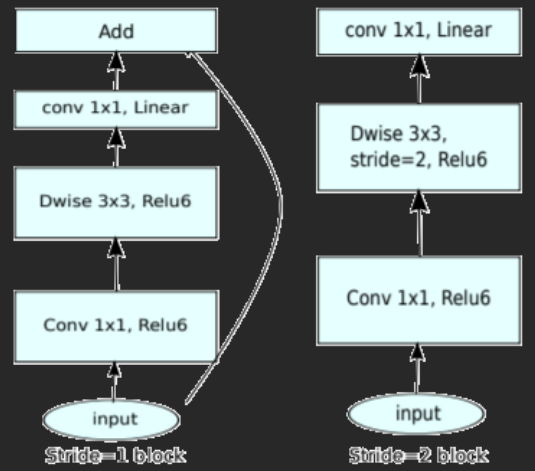
MobileNet v2



Expansion Pointwise

Depthwise

Bottleneck Слой



(d) Mobilenet V2

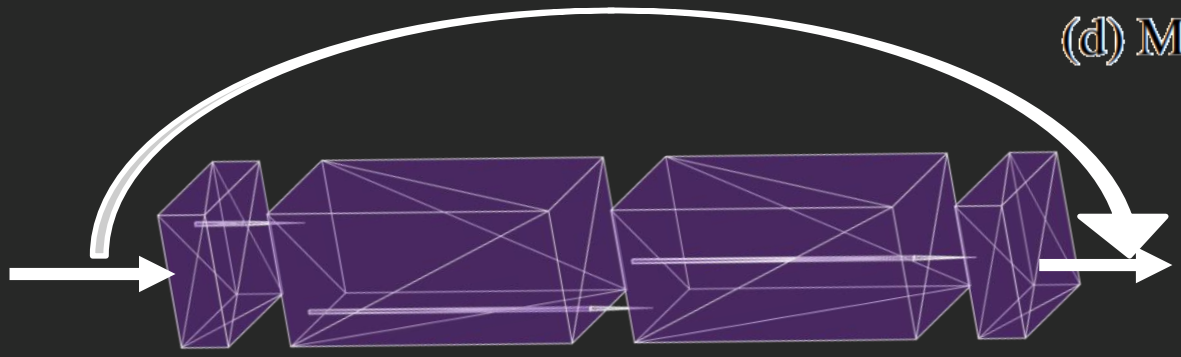


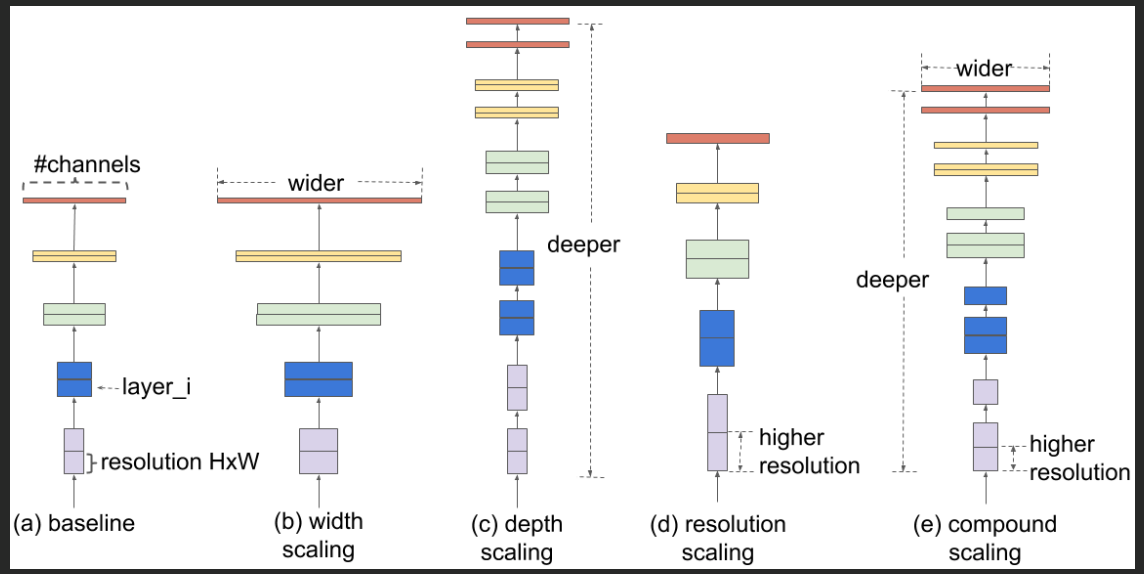
Table 2: MobileNetV2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510-4520. 2018.

Google EfficientNet



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

$$\max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big)$$

$$\text{Memory}(\mathcal{N}) \le \text{target\_memory}$$

$$\text{FLOPS}(\mathcal{N}) \le \text{target\_flops}$$

$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \ge 1, \beta \ge 1, \gamma \ge 1$$

Tan, M. and Le, Q., 2019, May. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (pp. 6105-6114). PMLR.

И зачем нам нужна вся эта история? Это все «там» А мы «здесь»

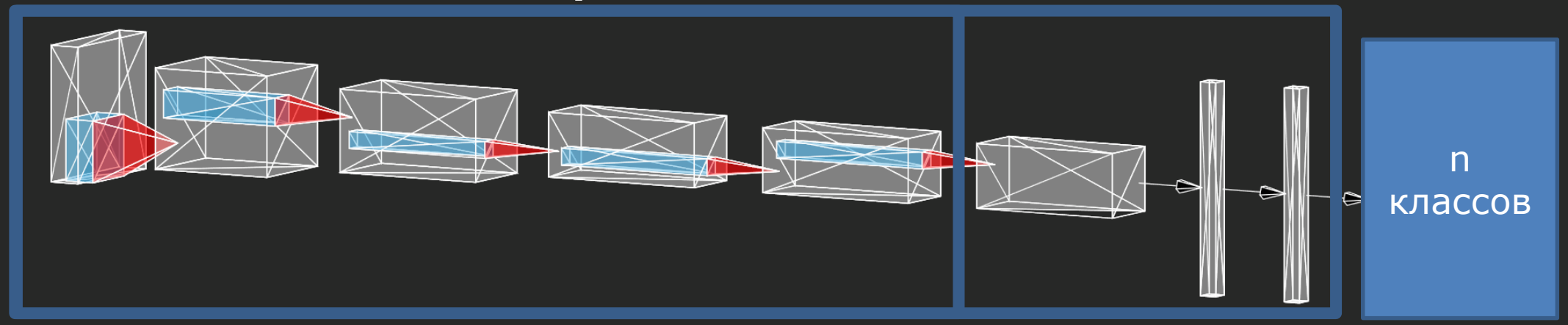
Загрузка Готовой Модели

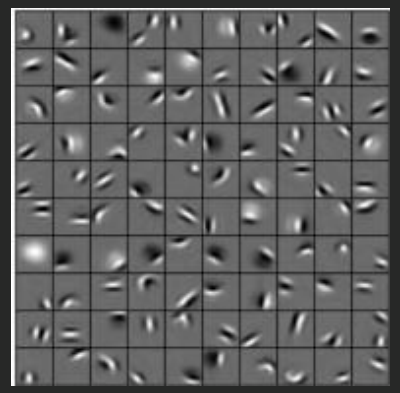
```
tf.keras.applications.DenseNet121(
    include_top=True, weights='imagenet', input_tensor=None,
    input_shape=None, pooling=None, classes= 1000
)
```

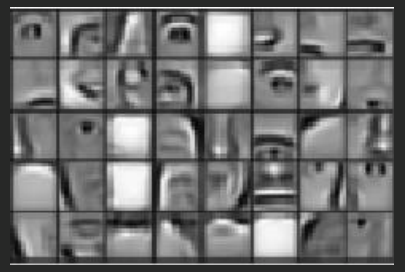
```
tf.keras.applications.MobileNet(
    input_shape=None, alpha= 1.0, depth_multiplier= 1, dropout=0.001,
    include_top=True, weights='imagenet', input_tensor=None, pooling=None,
    classes= 1000, classifier_activation='softmax', **kwargs
)
```

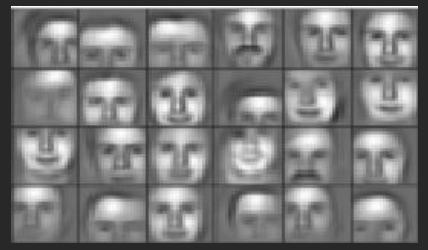
https://www.tensorflow.org/api_docs/python/tf/keras/

Продолжение Обучения
Transfer Learning
Замораживаем веса





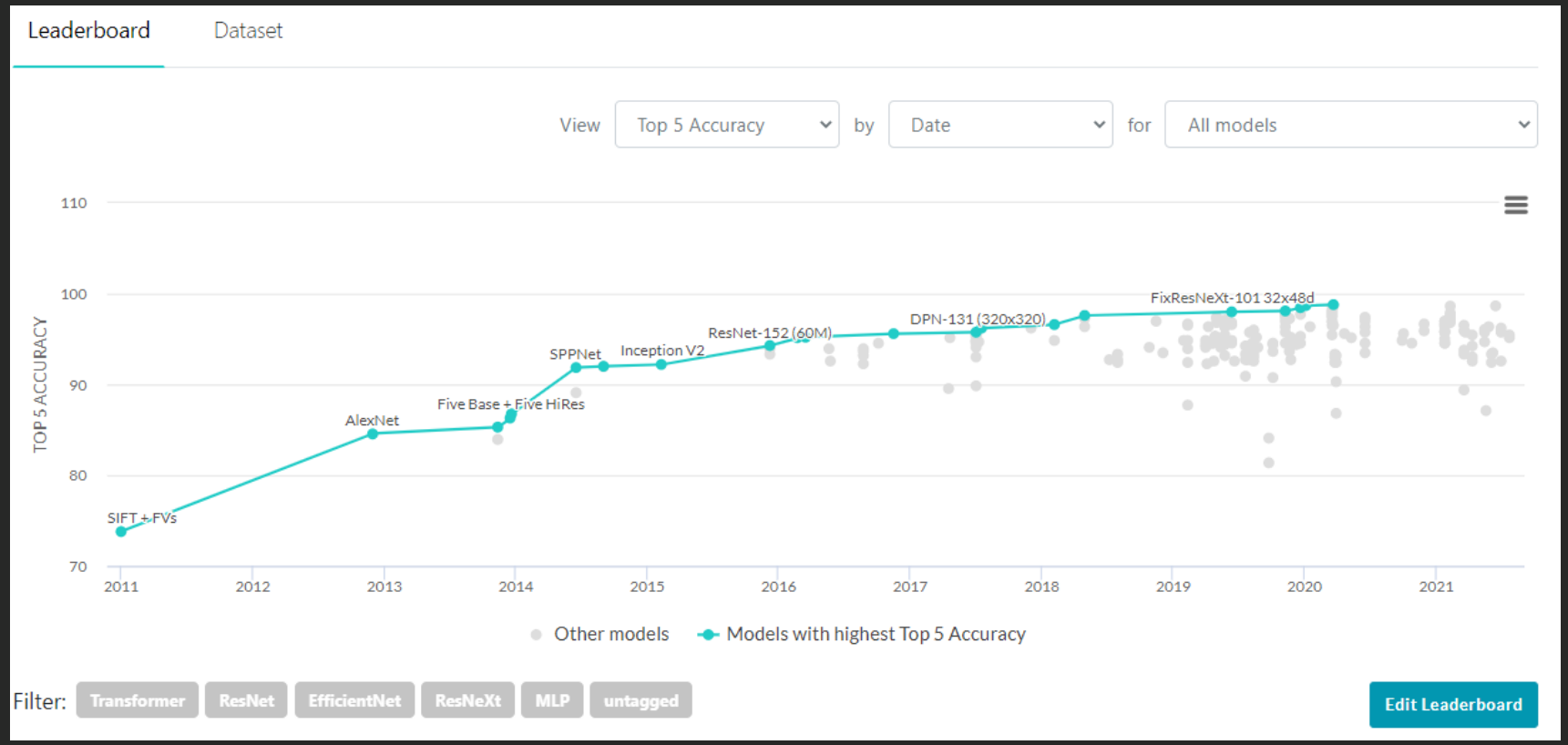




Продолжение Обучения
Transfer Learning

```
base_model = keras.applications.MobileNet(weights='imagenet',
input_shape=(150, 150, 3),include_top=False)

base_model.trainable = False

inputs = keras.Input(shape=(150, 150, 3))
x = base_model(inputs, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
outputs = keras.layers.Dense(1)(x)
model = keras.Model(inputs, outputs)

model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=[keras.metrics.BinaryAccuracy()])


model.fit(new_dataset, epochs=10, callbacks=..., validation_data=...)
```

Ссылки на Архитектуры и Статьи



https://paperswithcode.com/sota/image-classification-on-imagenet?metric=Top%205%20Accuracy

Резюме

LeNet-5

Классика, Свертки + Pooling + Полносвязные

AlexNet

Глубокое обучение не так уж и плохо, GPU

VGG

Много (очень) сверток 3x3

GoogLeNet

Свертки 1x1, Inception слои, упрощение сверток

ResNet

Пропуски (Residuals), можно еще глубже сети

MobileNet

меньше мат. операций «за ту же точность»

Вопросы, пожелания, предложения

????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.09
Где мы изучаем историю свёрточных нейронных сетей для поиска объектов

Докладчик
Долганов Антон

В предыдущем выпуске

LeNet-5

Классика, Свертки + Pooling + Полносвязные

AlexNet

Глубокое обучение не так уж и плохо, GPU

VGG

Много (очень) сверток 3x3

GoogLeNet

Свертки 1x1, Inception слои, упрощение сверток

ResNet

Пропуски (Residuals), можно еще глубже сети

MobileNet

меньше мат. операций «за ту же точность»

Содержание

Классификация с локализацией

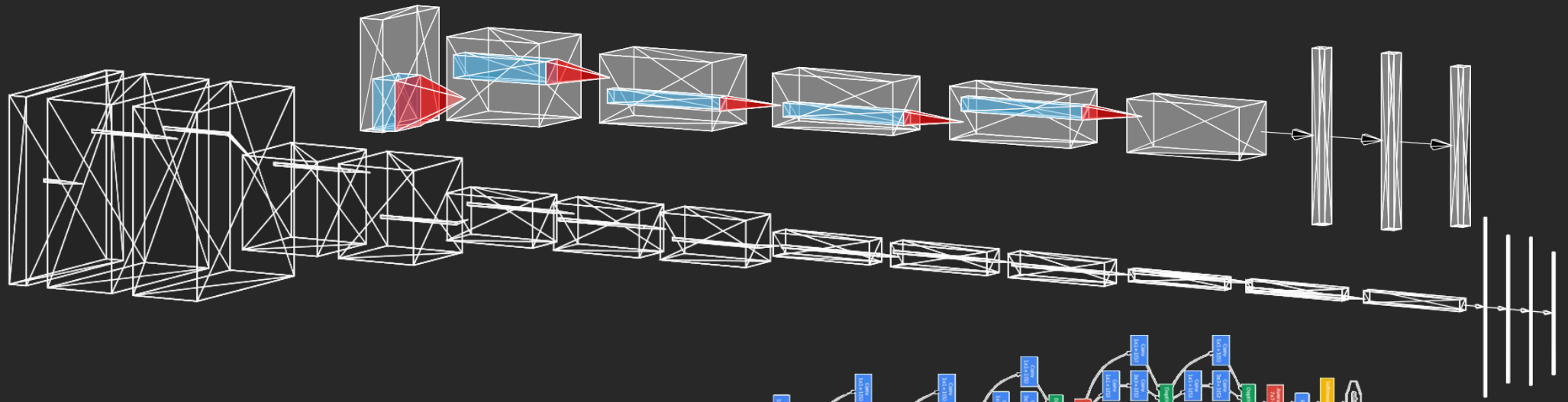
R-CNN

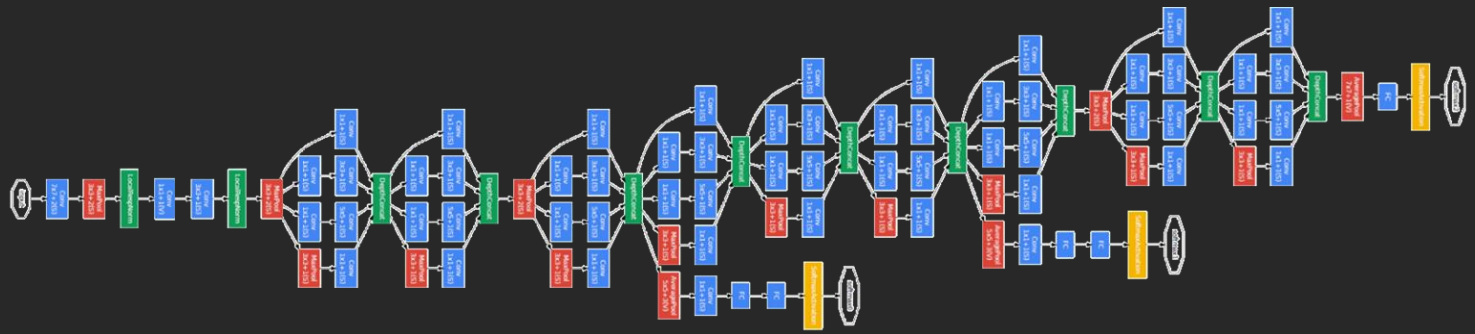
YOLO

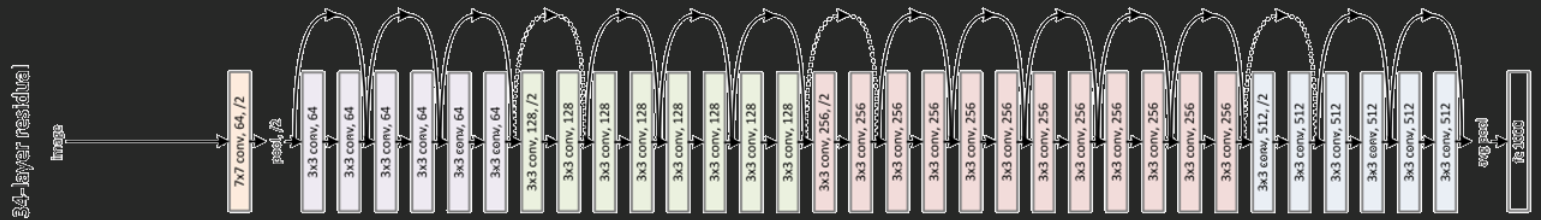
SSD

CenterNet

Архитектуры для Классификации Изображений

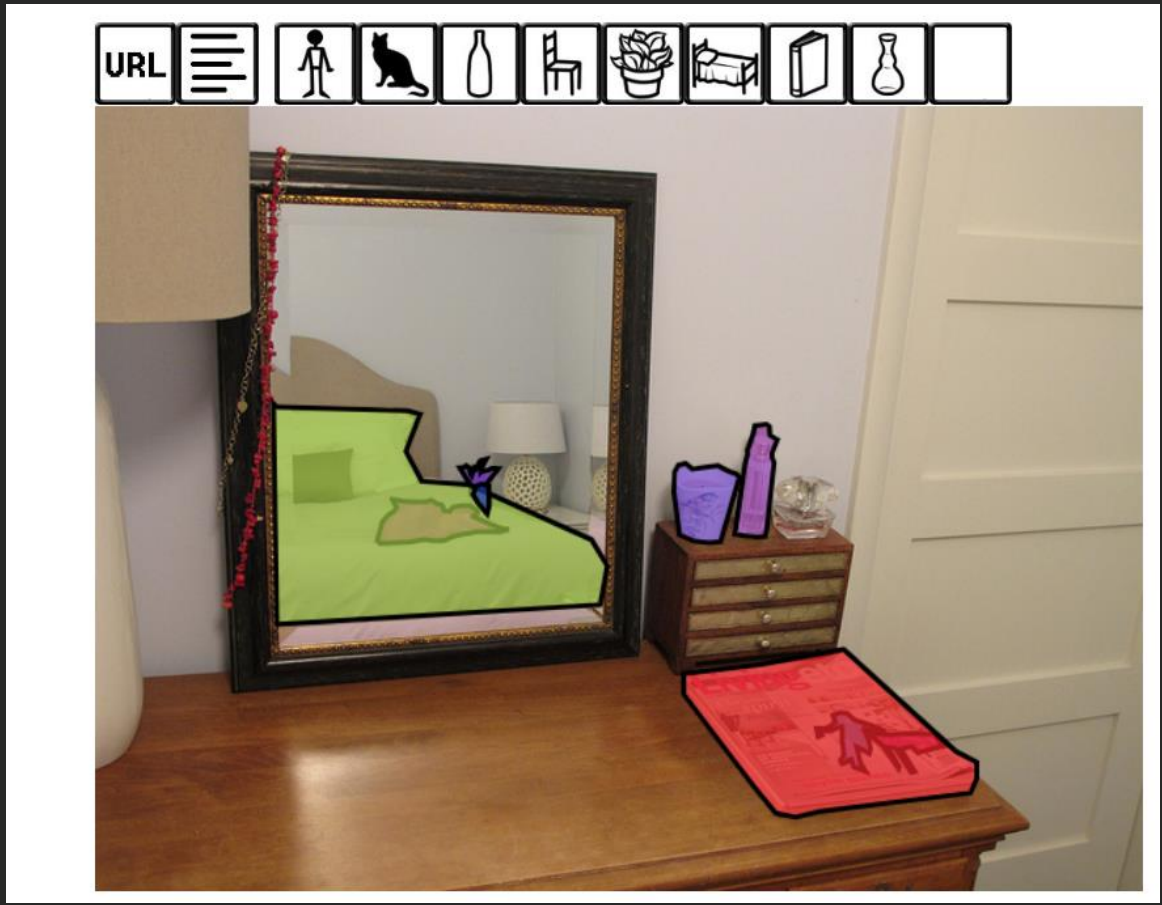






Компьютерное Зрение для Машин

COCO - ImageNet для Детекции



Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

https://cocodataset.org/

Содержание

Классификация с локализацией

R-CNN

YOLO

SSD

CenterNet

Классификация с Локализацией

(0,0)

(1,1)

Классы Интереса

1. Пешеход
2. Машина
3. Велосипед
4. Фон

$$y = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Функция Потерь

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

BCE Binary Cross-Entropy

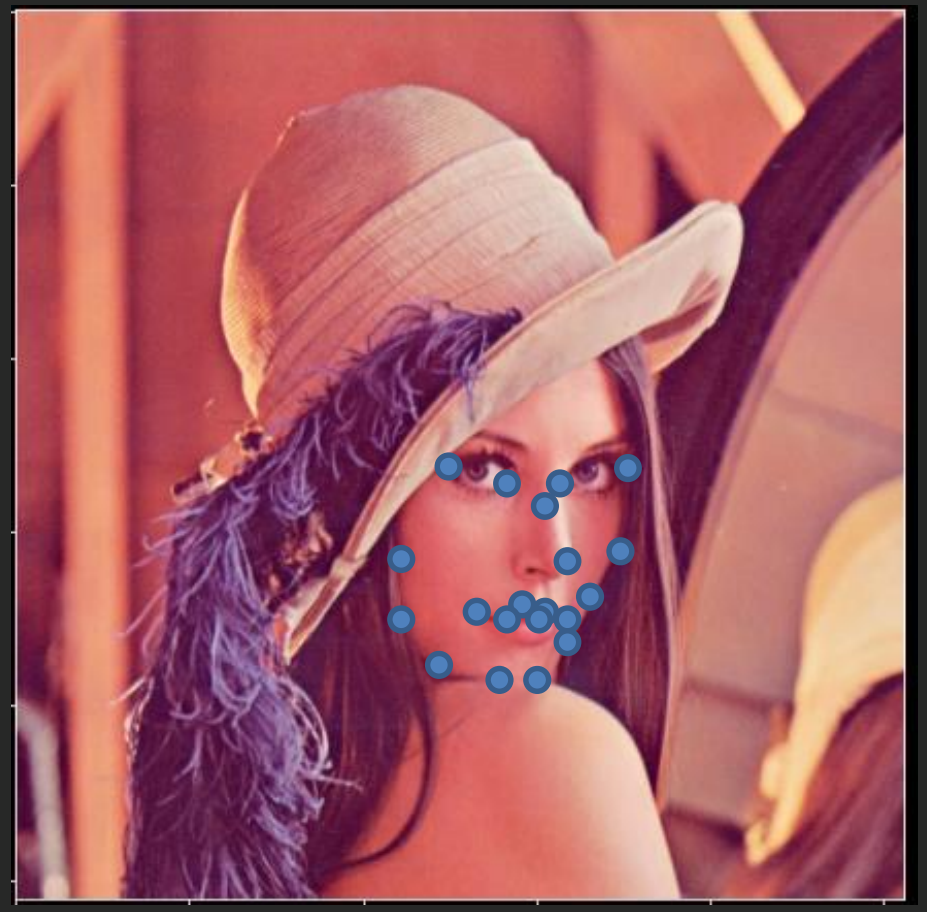
MSE Mean Square Error

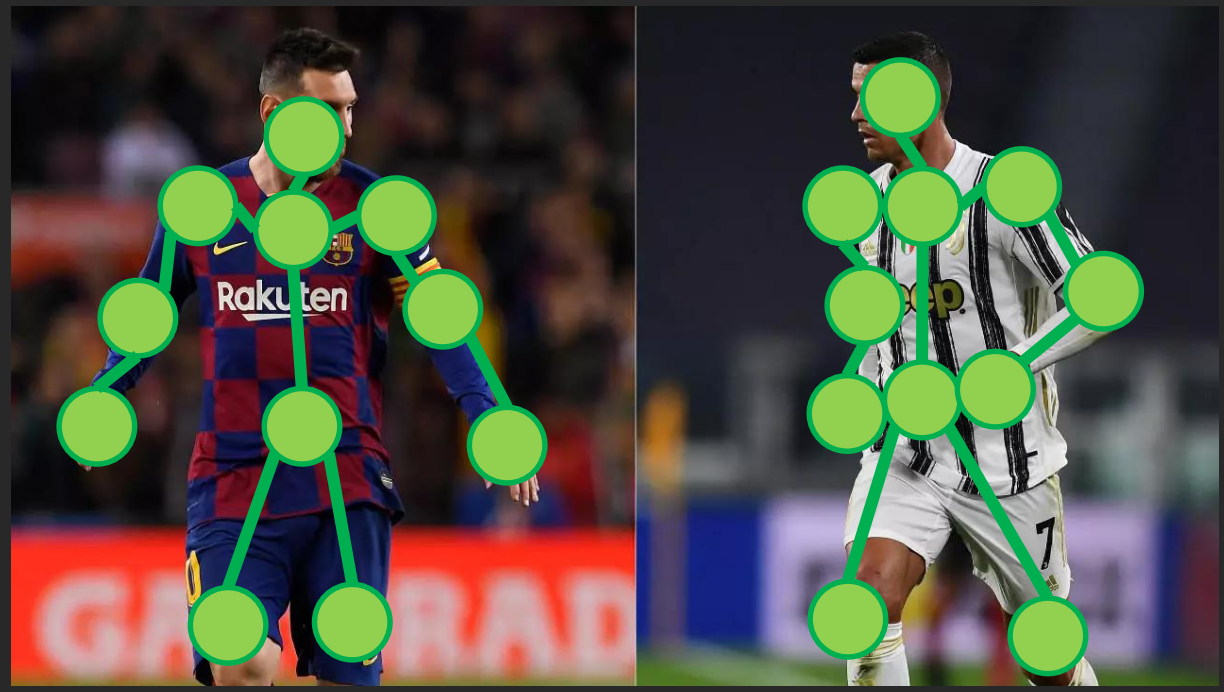
CCE Categorical Cross-Entropy

$$Loss_c = BCE(p_c) + MSE(b_x) + MSE(b_y) + MSE(b_h) + MSE(b_w) + CCE(c_1, c_2, c_3)$$

$$Loss_{\text{фон}} = BCE(p_c)$$

Поиск Особых Точек

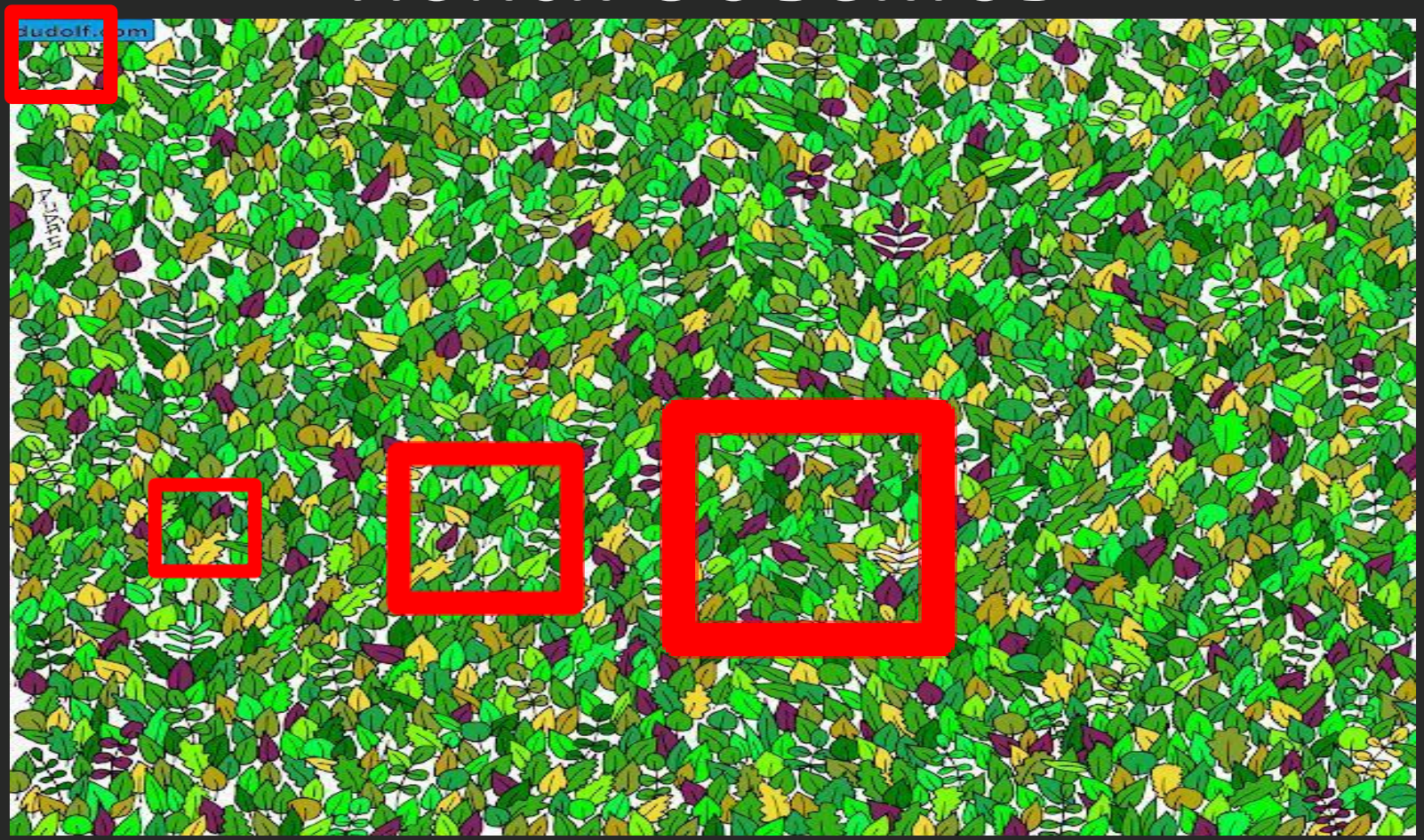




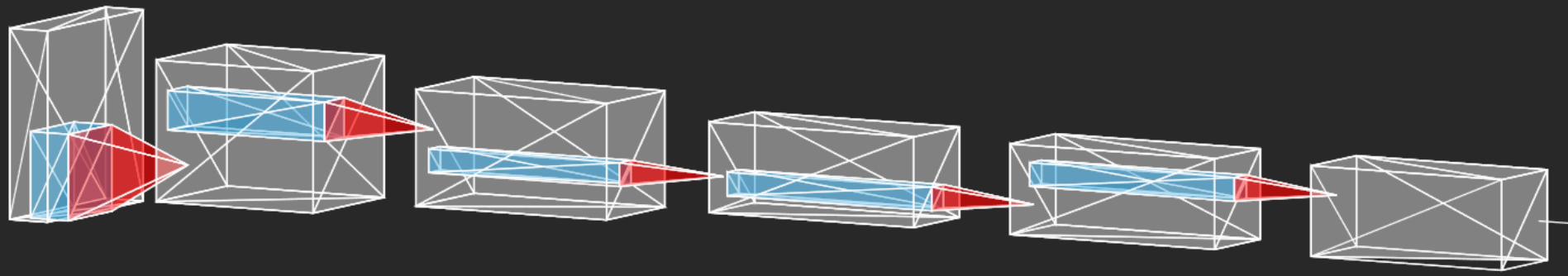
$$y = \begin{bmatrix} \dots \\ l_1 x \\ l_1 y \\ l_2 x \\ l_2 y \\ \dots \\ l_N x \\ l_N y \end{bmatrix}$$

Поиск Объектов

Найти лягушку

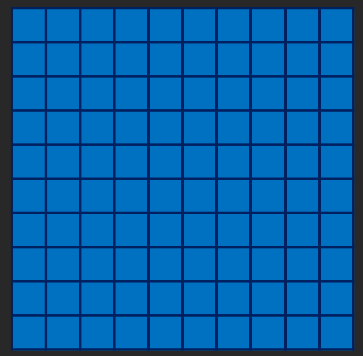


Вспомним AlexNet



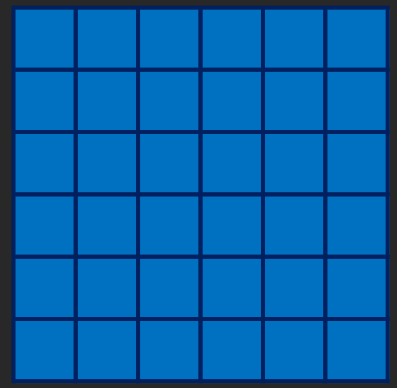
Смотрим как на Свертки 1x1, а не полносвязные Слои

Сверточный Подход



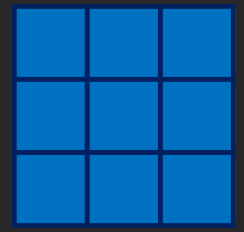
10 x 10 x 3

5 x 5



6 x 6 x 16

MaxPool
2 x 2



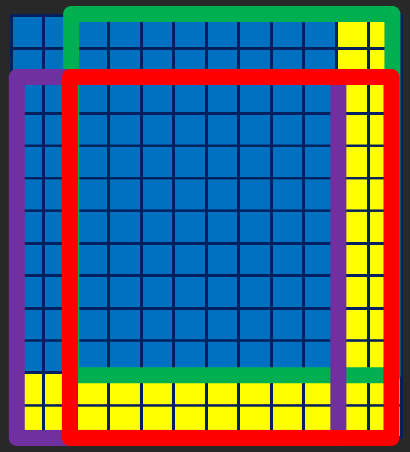
3 x 3 x 16

1 x 1 x 144

3 x 3

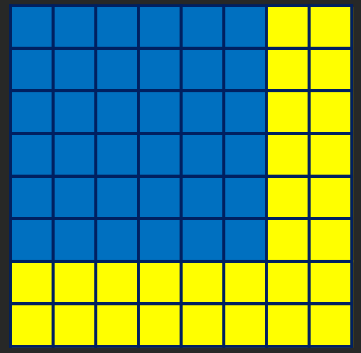
1 x 1

1 x 1 x 144



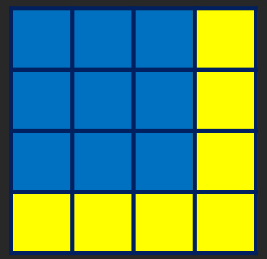
12 x 12 x 3

5 x 5



8 x 8 x 16

MaxPool
2 x 2



4 x 4 x 16

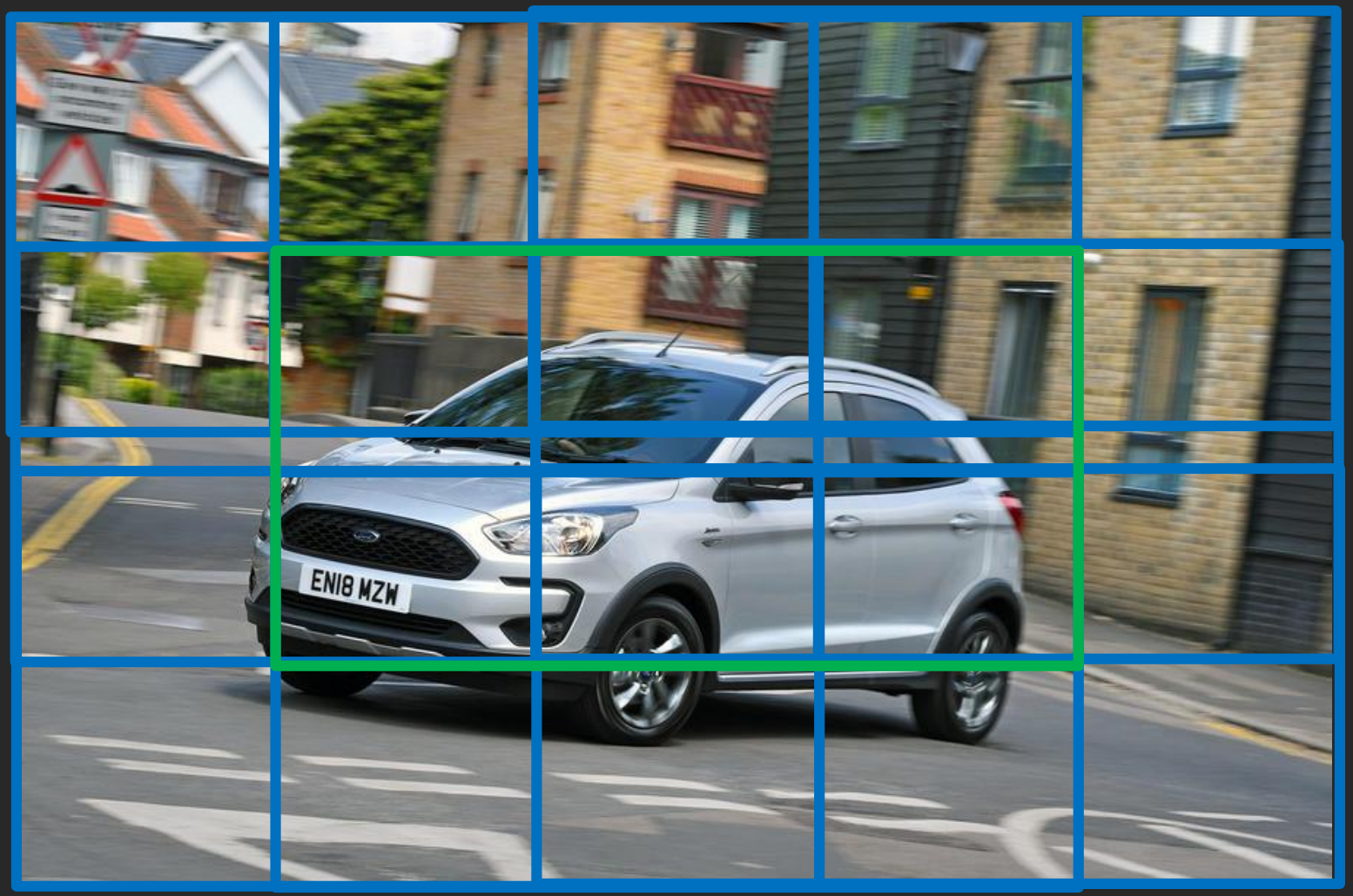
2 x 2 x 144

3 x 3

1 x 1

2 x 2 x 144

Классификация с Локализацией



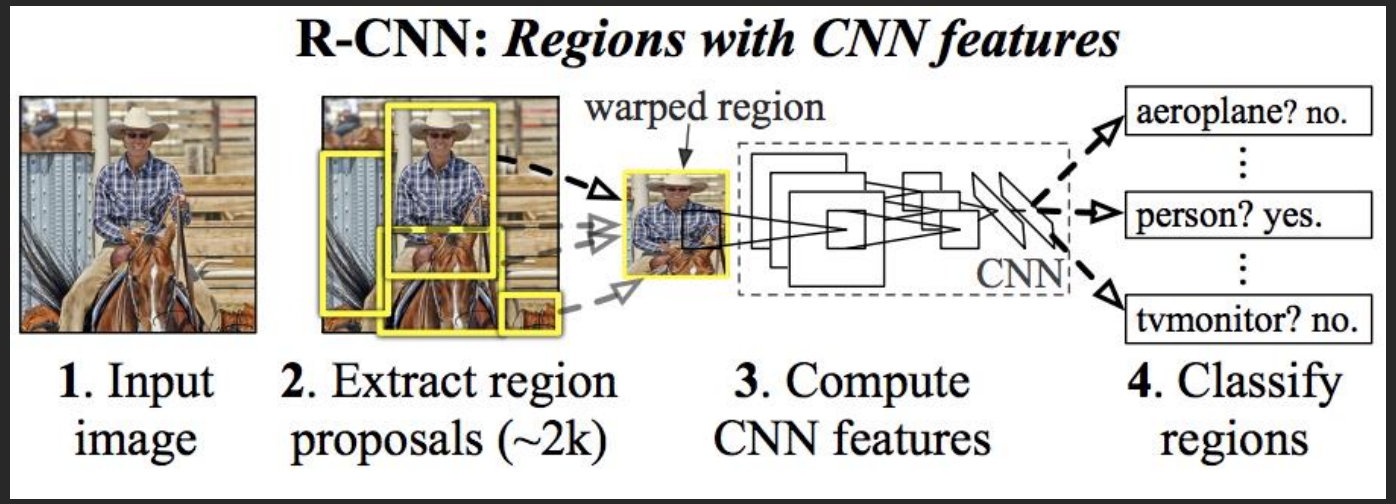
Содержание

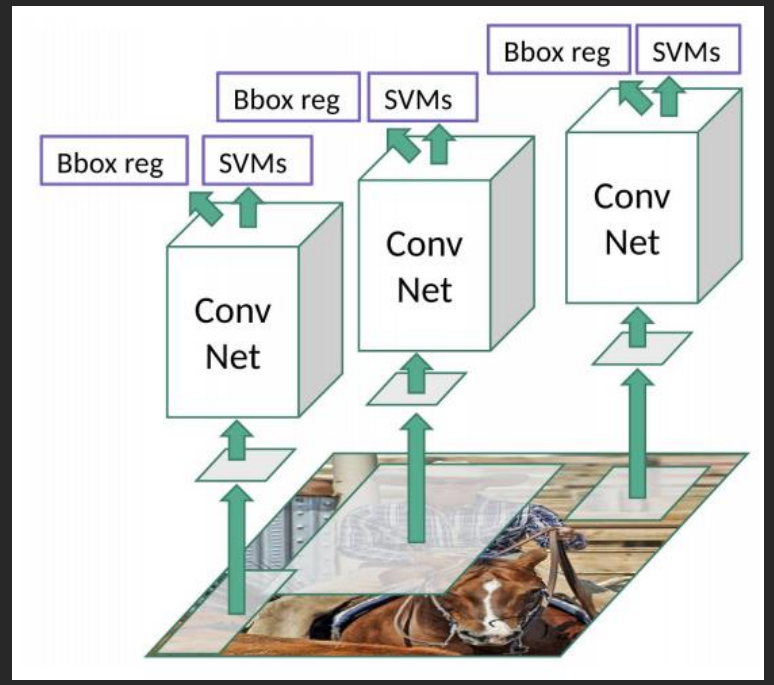
R-CNN - Region Based Convolutional Neural Networks

Выборочный поиск:

1. Создаем начальную подсегментацию, много регионов-кандидатов
2. Жадный алгоритм для рекурсивного объединения похожих регионов в более крупные.
3. Подготовка финальных предложений для регионов-кандидатов

2000 регионов



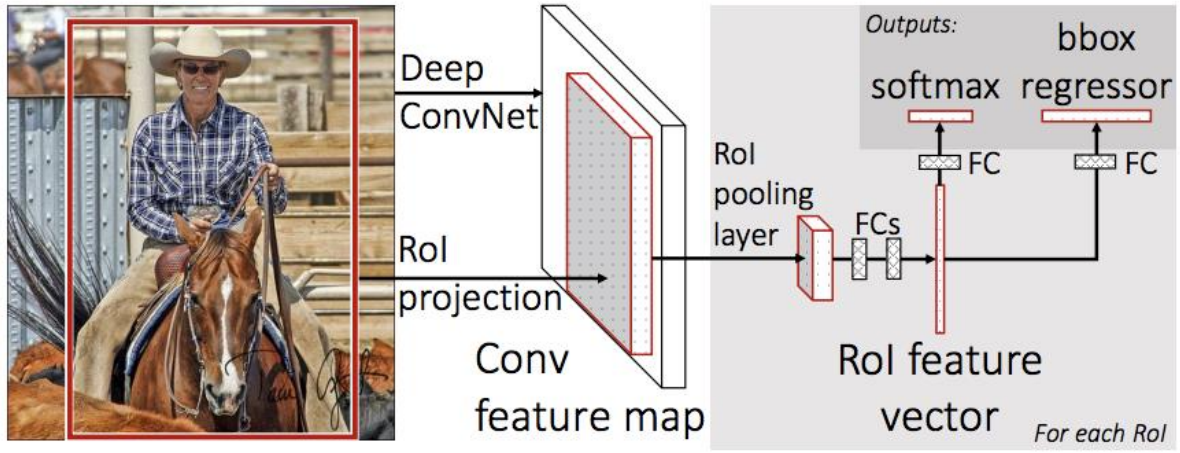


Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587. 2014.

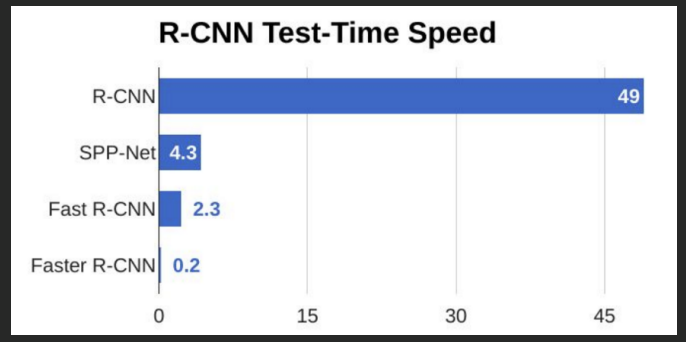
R-CNN – Это плохо?

- На обучение сети по-прежнему уходит огромное количество времени, так как вам нужно классифицировать 2000 предложений регионов для каждого изображения;
- Это невозможно сделать в реальном времени, так как для каждого тестового изображения требуется около 49 секунд;
- Алгоритм выборочного поиска - это фиксированный алгоритм. Следовательно, на данном этапе обучение не происходит. Это может привести к плохим предложениям для регионов-кандидатов.

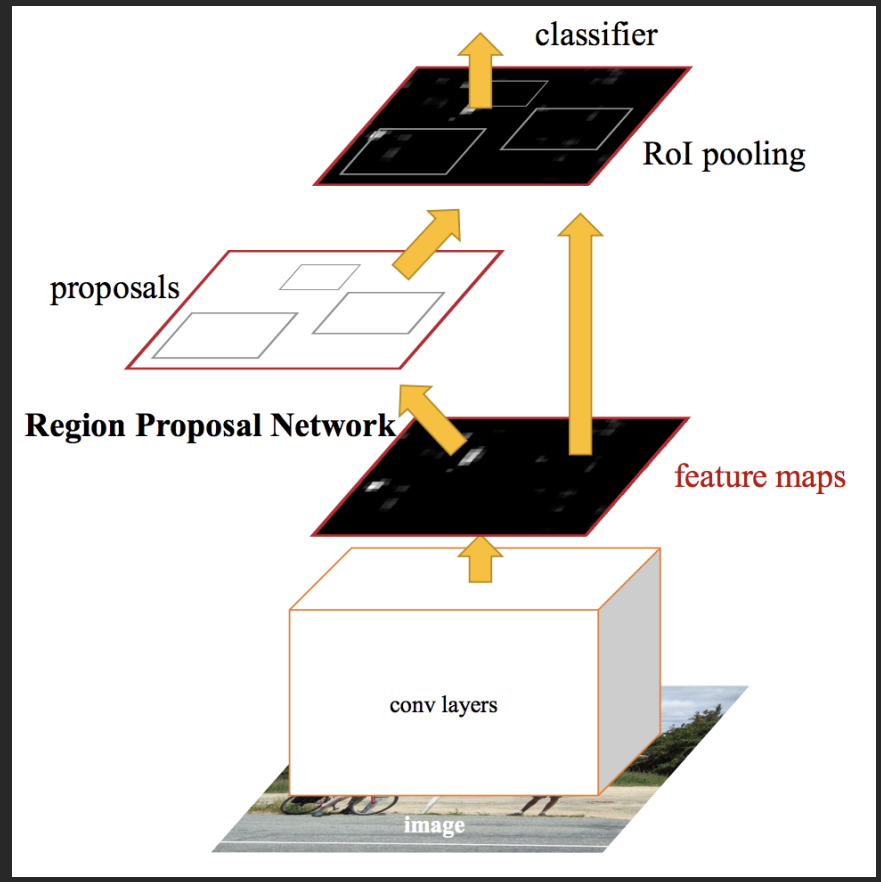
Fast(-er) R-CNN



Fast R-CNN



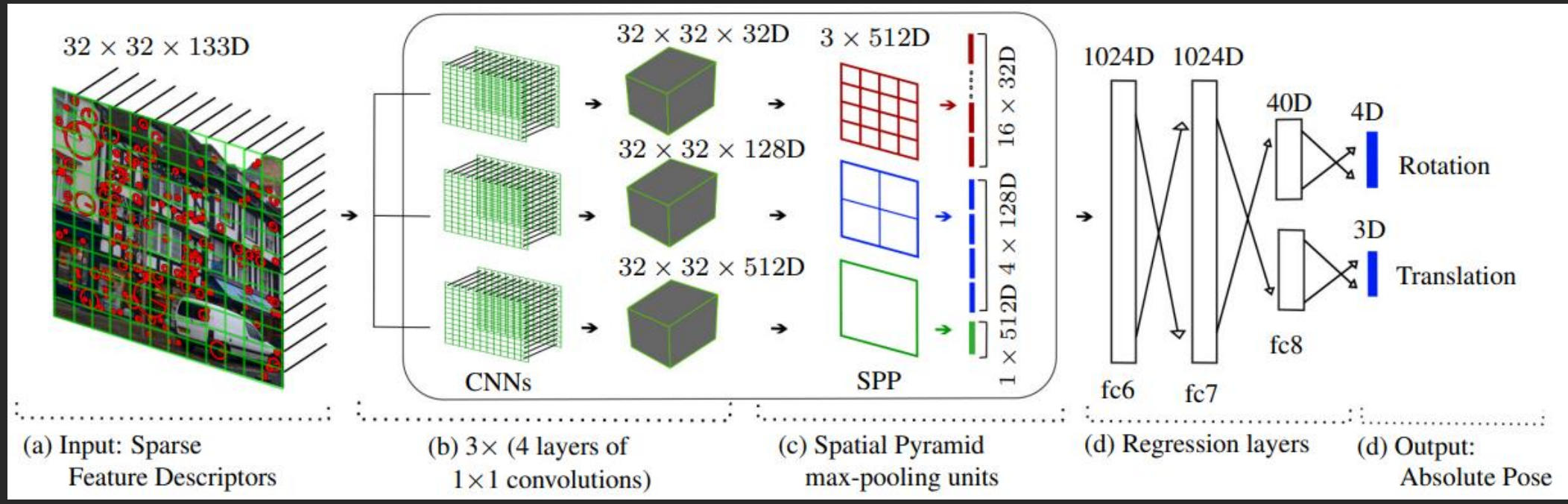
Girshick, Ross. "Fast r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.



Faster R-CNN

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28, pp.91-99.

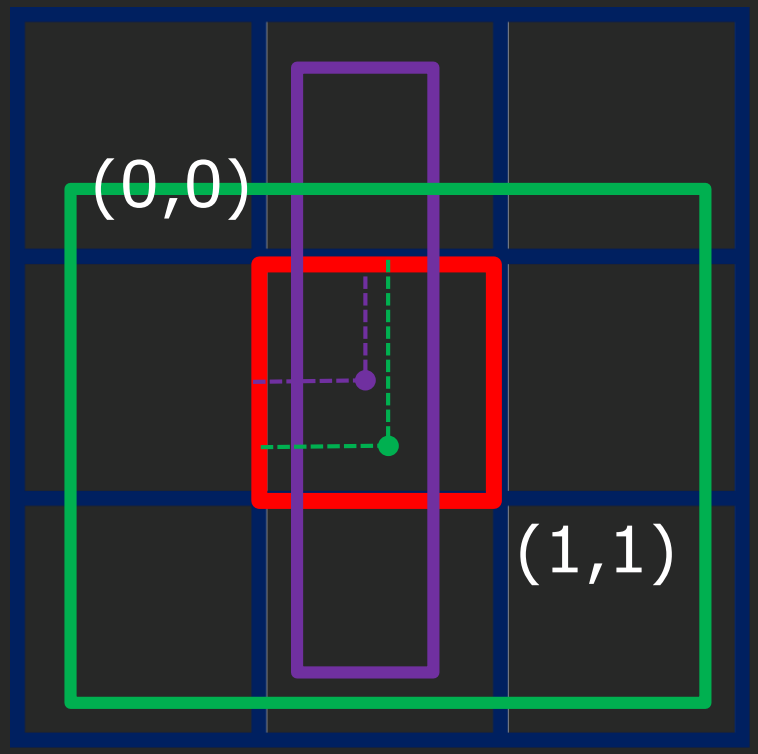
SPP-NET
Spatial Pyramid Pooling Layer



Purkait, Pulak, Cheng Zhao, and Christopher Zach. "SPP-Net: Deep absolute pose regression with synthetic views." arXiv preprint arXiv:1712.03452 (2017).

Содержание

YOLO — You Only Look Once



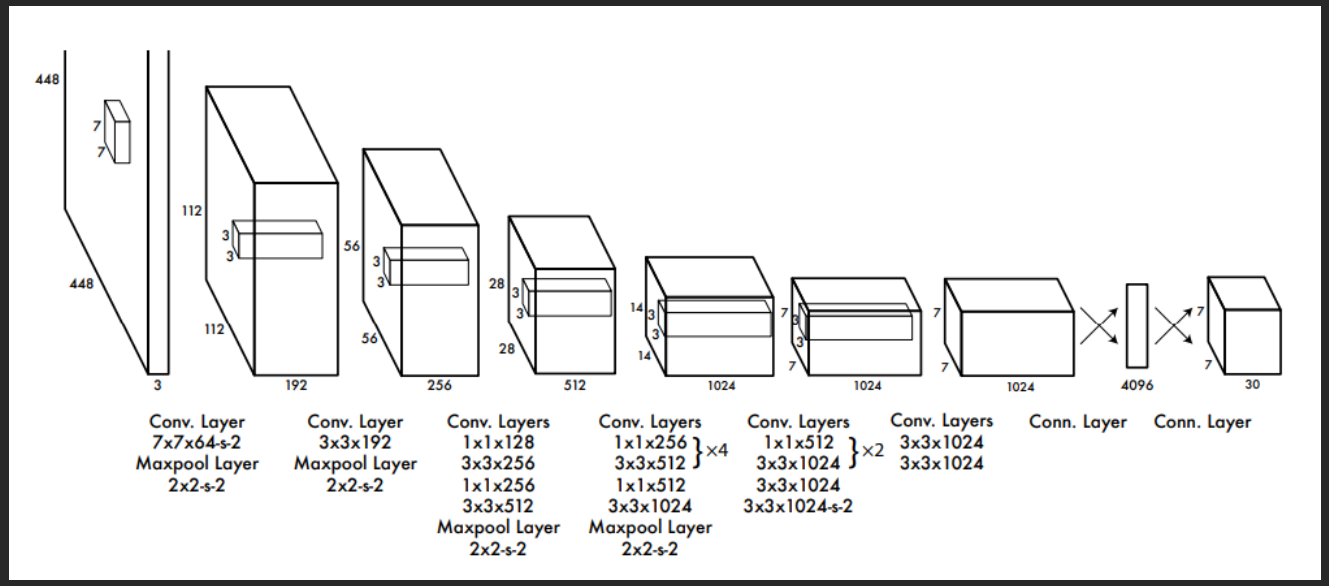
$$\begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w \end{bmatrix} \quad \begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

$SoftMax$

20 Классов $\begin{pmatrix} 0.8 \\ 0.5 \\ 0.1 \\ \vdots \\ 0.2 \\ 0.3 \end{pmatrix}$ 0.9

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016.

YOLO: Архитектура Модели



24 Сверточных Слоя
2 Полносвязных Слоя

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

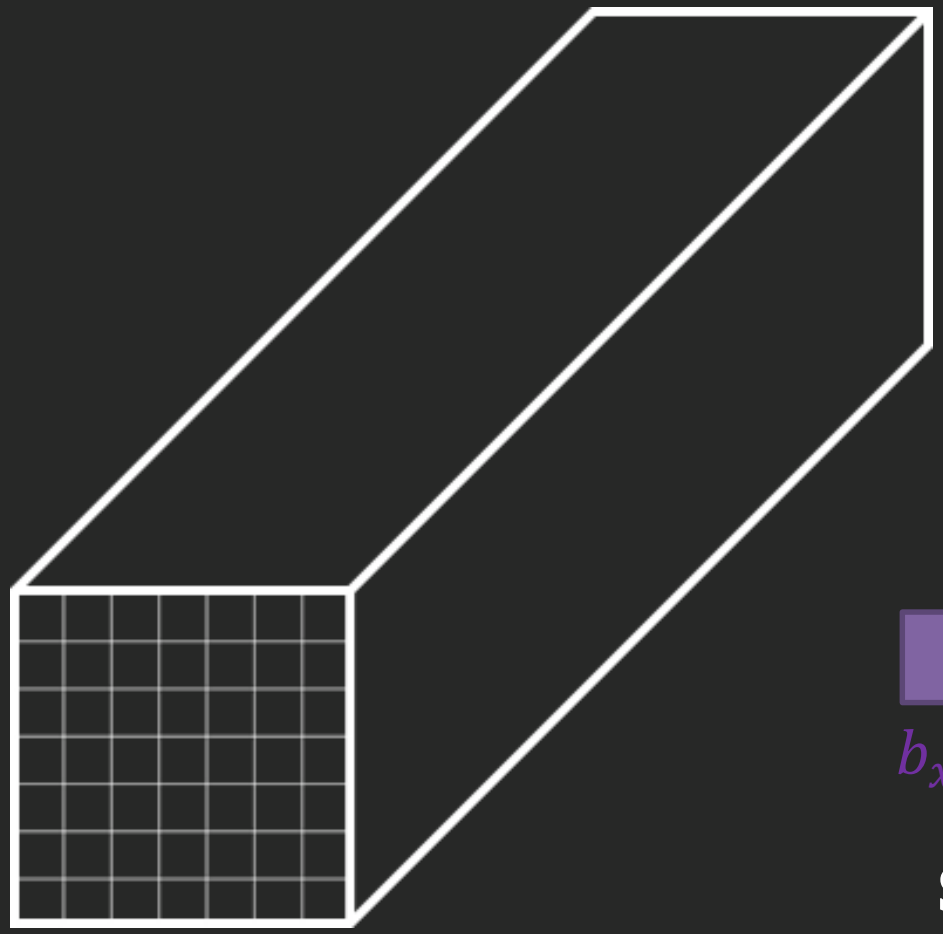
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Функционал Потерь

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016.

YOLO: А что на выходе модели?

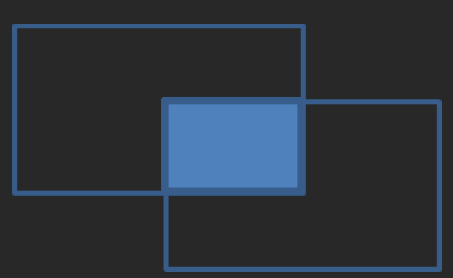


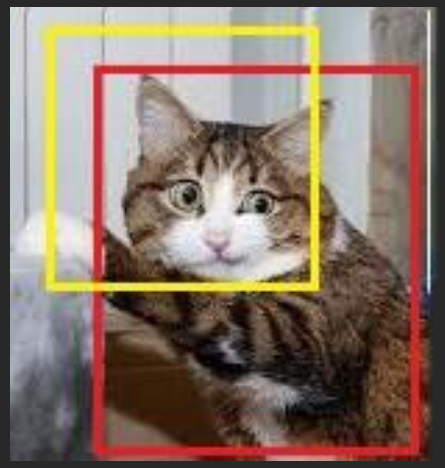
S × S × (B × 5 + Классы)

7 × 7 × 30

YOLO

$$IOU = \frac{Intersection}{\text{Union}}$$





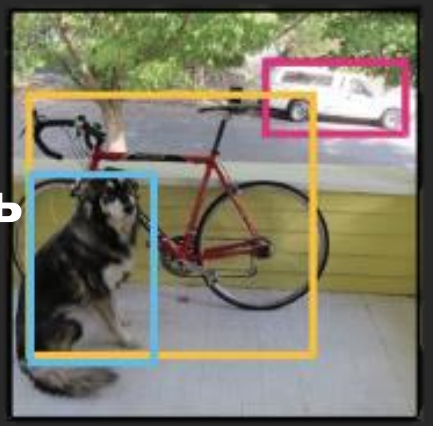
YOLO: Как выбрать нужную рамку

Non-maximal suppression

Для каждого класса (автомобили, пешеходы и т.д.)
Выполните:

1. Отказаться от всех рамок с уверенностью $C < C_{threshold}$ (например, $C < 0.5$)
2. Отсортируйте прогнозы по уверенности C.
3. Выберите поле с наибольшим C и выведите его как активный прогноз.
4. Отбросьте любую рамку у которой IOU с рамку на предыдущем шаге $> IOU_{threshold}$.
5. Начните снова с шага (3), пока не будут проверены все оставшиеся прогнозы.

Рамки и Уверенность



Классы

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016.

YOLO: Плюсы и Минусы

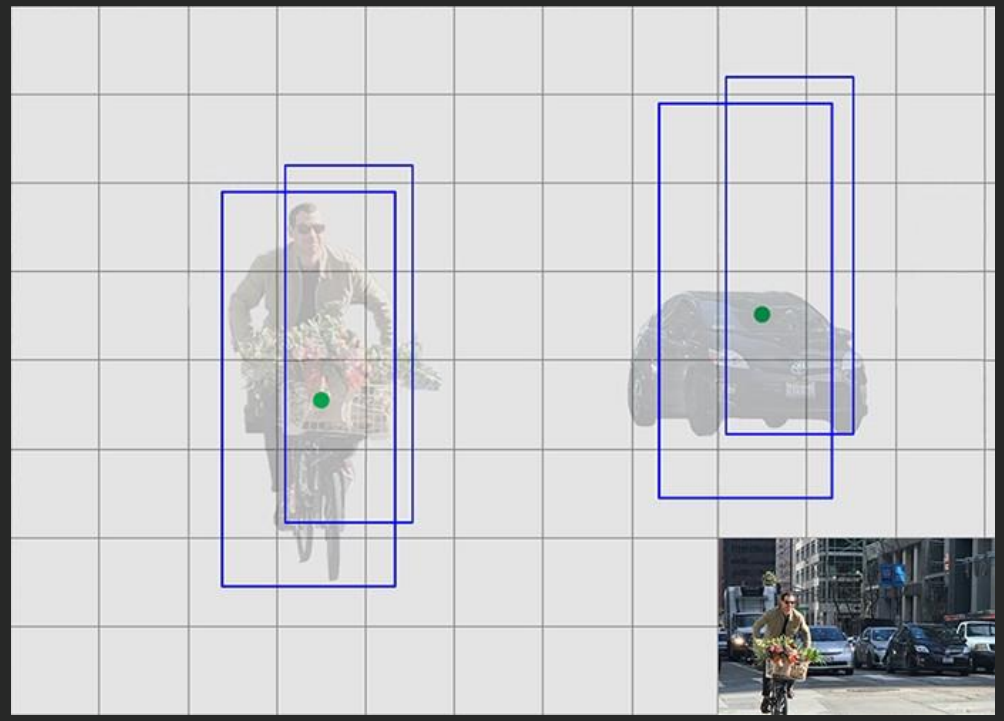
1) Поскольку каждая ячейка сетки предсказывает только две рамки и может иметь только один класс, это ограничивает количество близлежащих объектов, которые может предсказать YOLO, особенно для небольших объектов, которые появляются в группах, таких как стаи птиц.

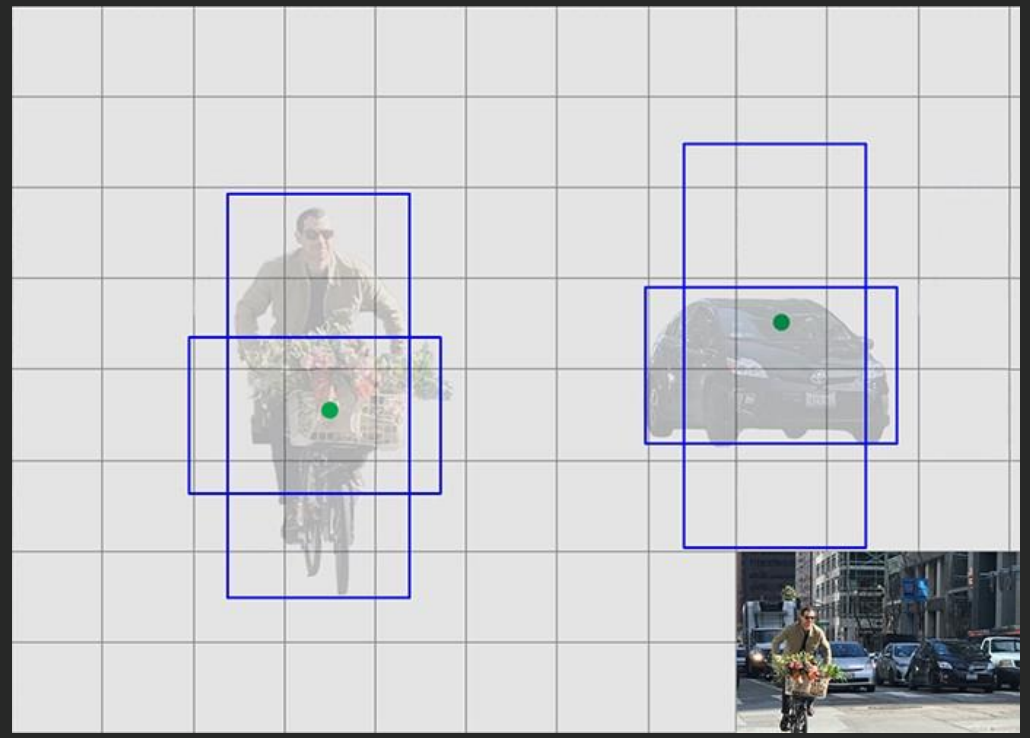
2) YOLO может обнаружить только 49 объектов.

3) Относительно высокая ошибка локализации.

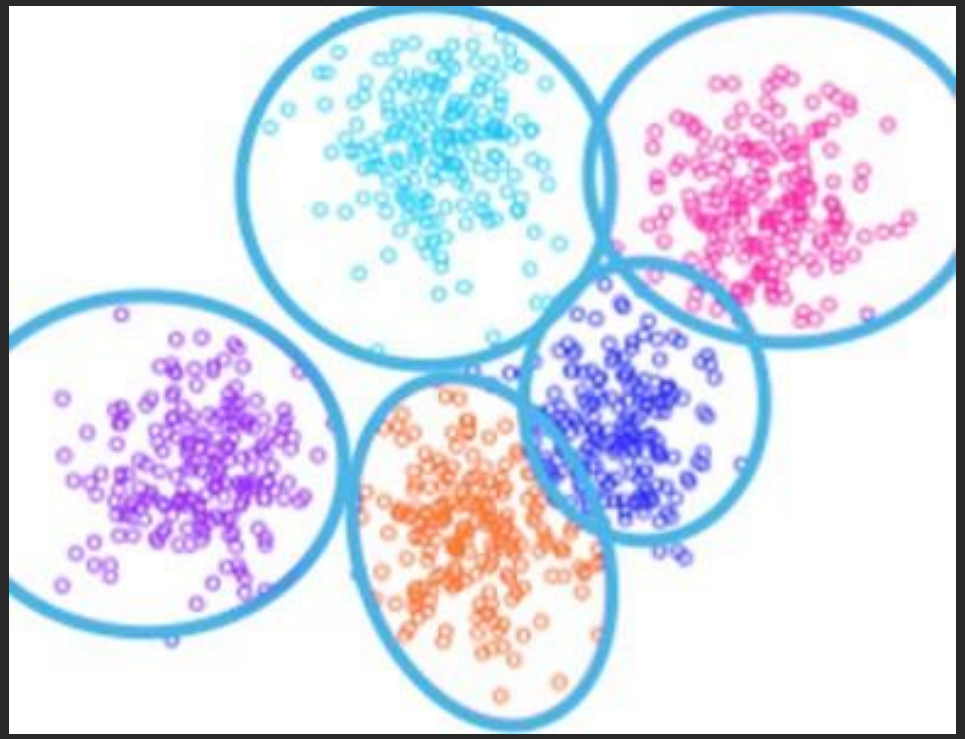
4) 45 кадров в секунду — лучше чем real-time

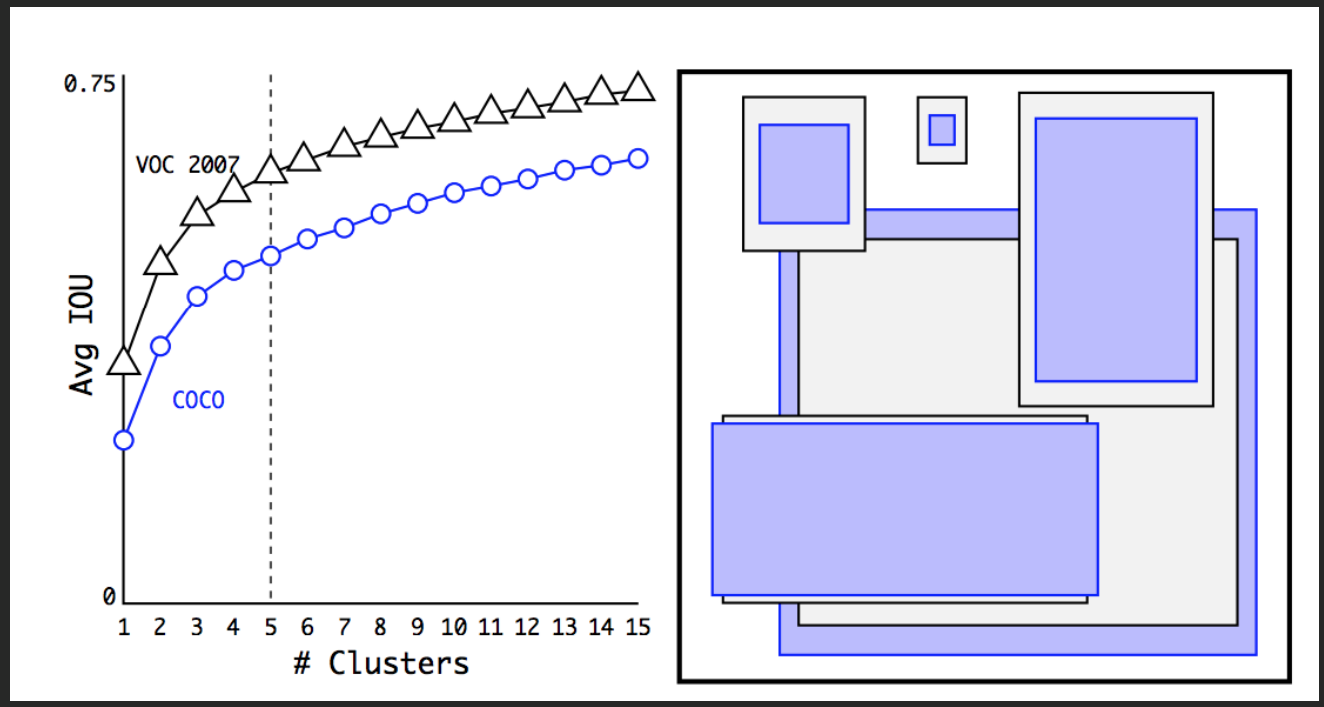
YOLO v2: что можно улучшить





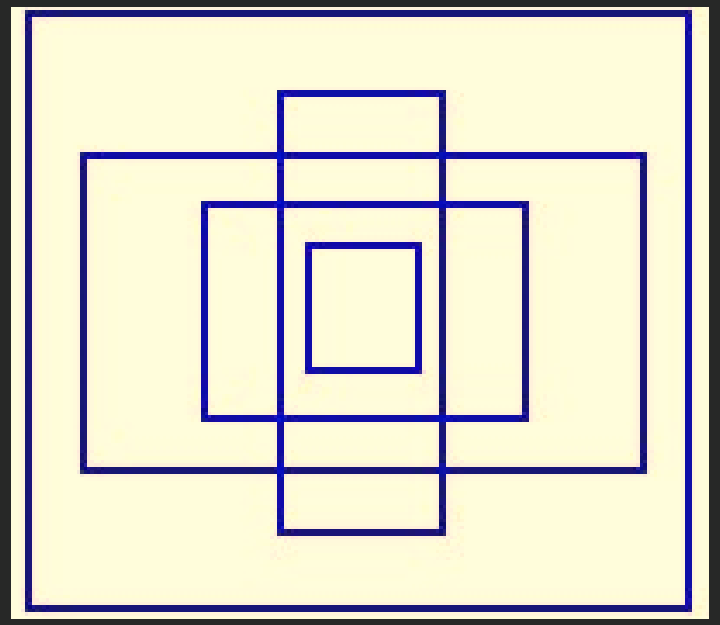
YOLO v2: Dimension Clusters

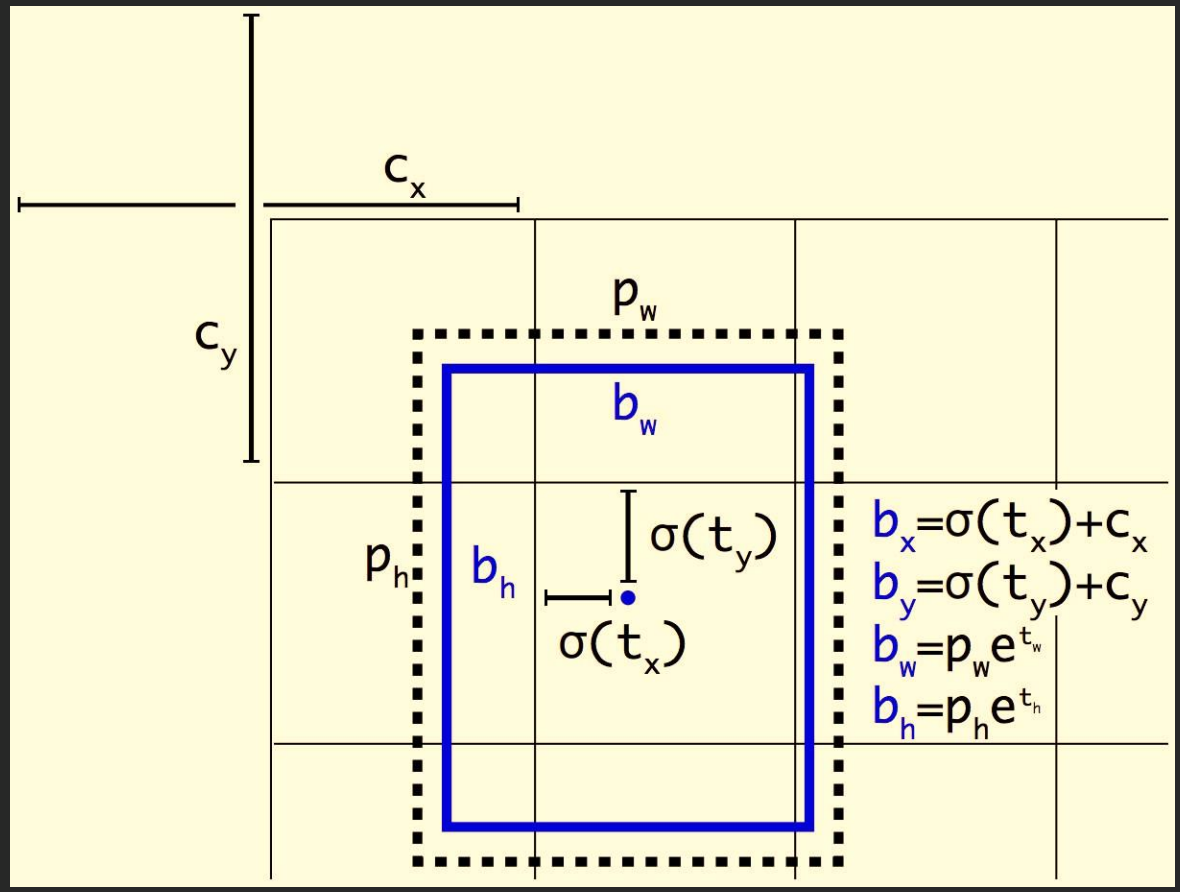




Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

YOLO v2: Direct location prediction





Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

YOLO v2: DarkNet

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

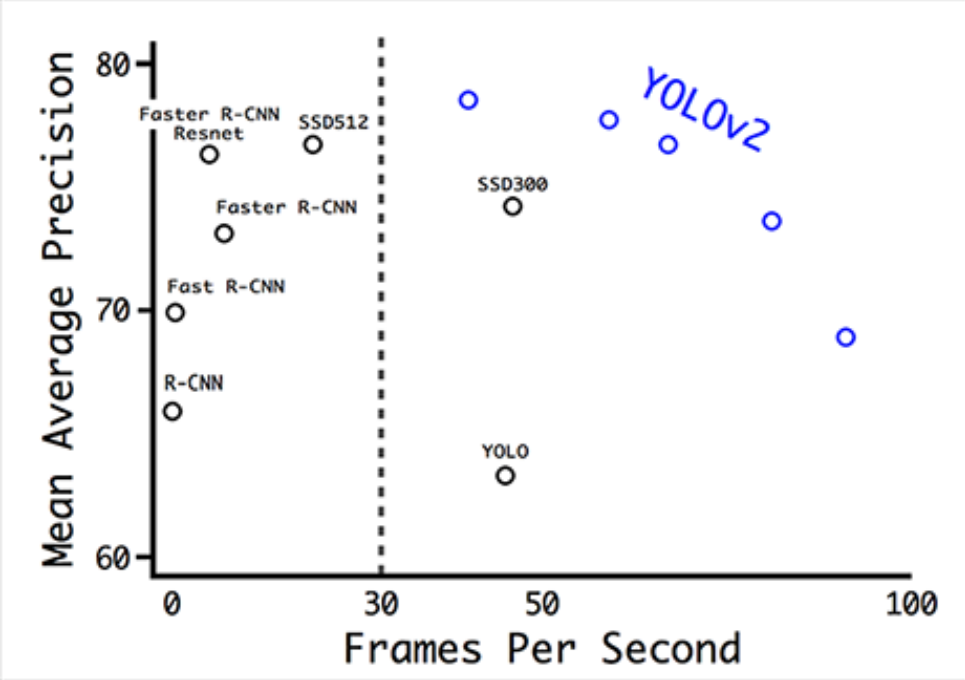
S × S × (AB*{1 + 4 + Классы})

13 x 13 x 125

Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

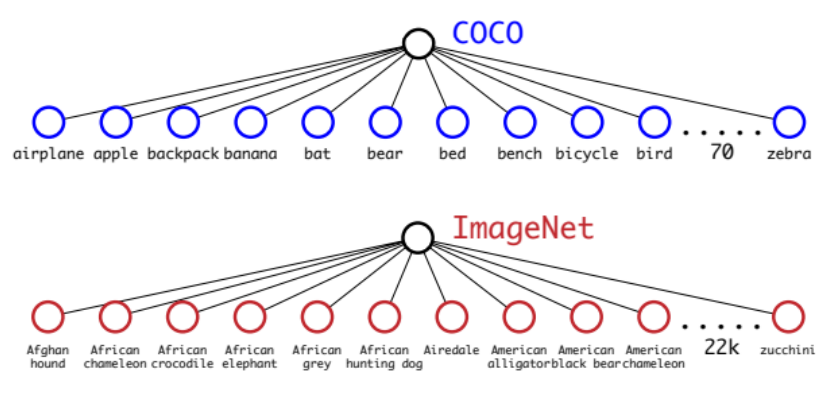
YOLO v2: много «небольших» изменений

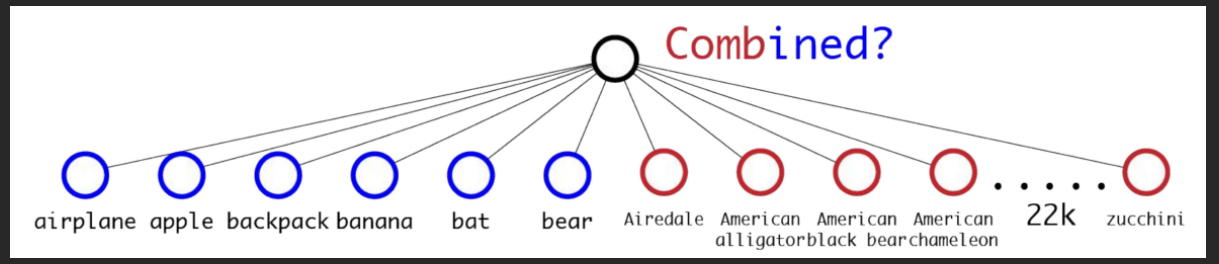
	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?								✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

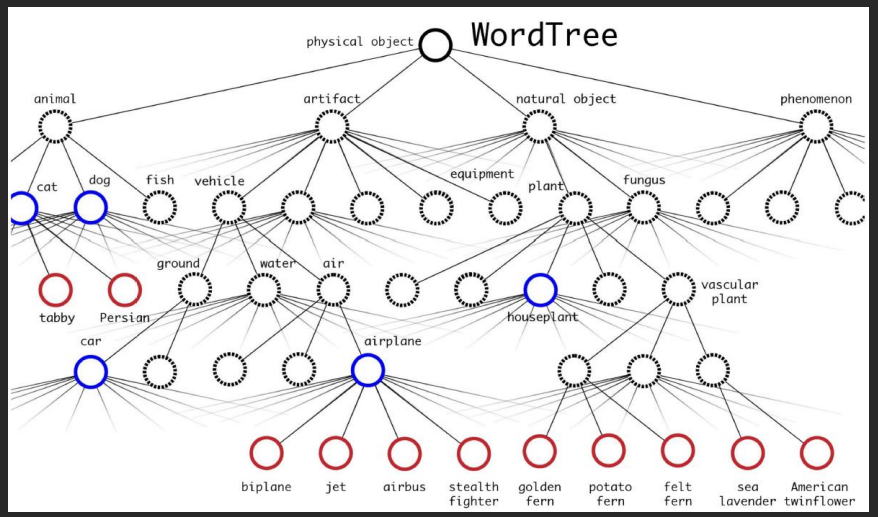


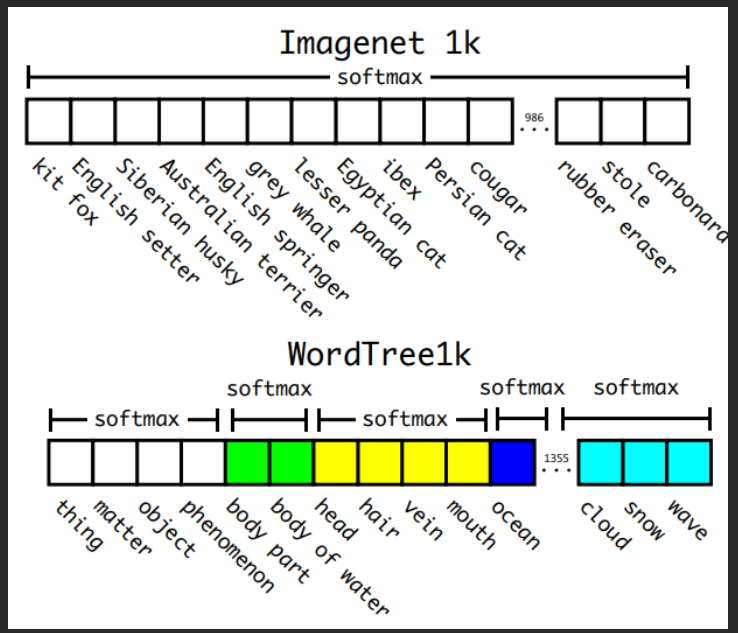
Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

YOLO9000: Better, Faster, Stronger



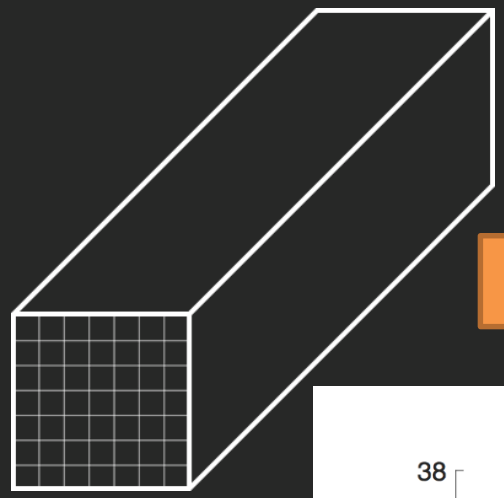


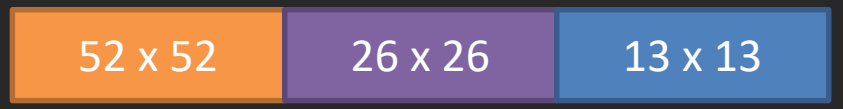


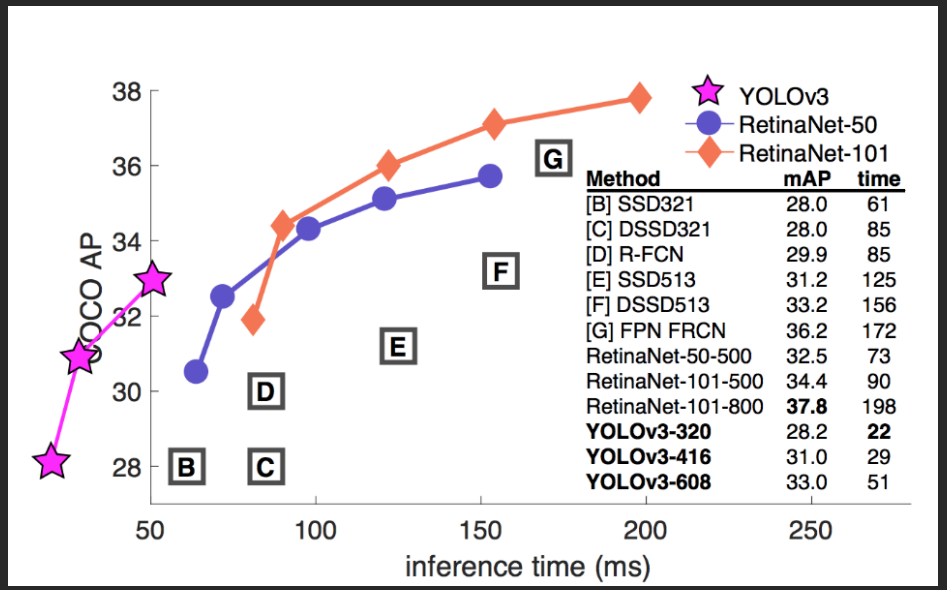


Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

YOLO v3







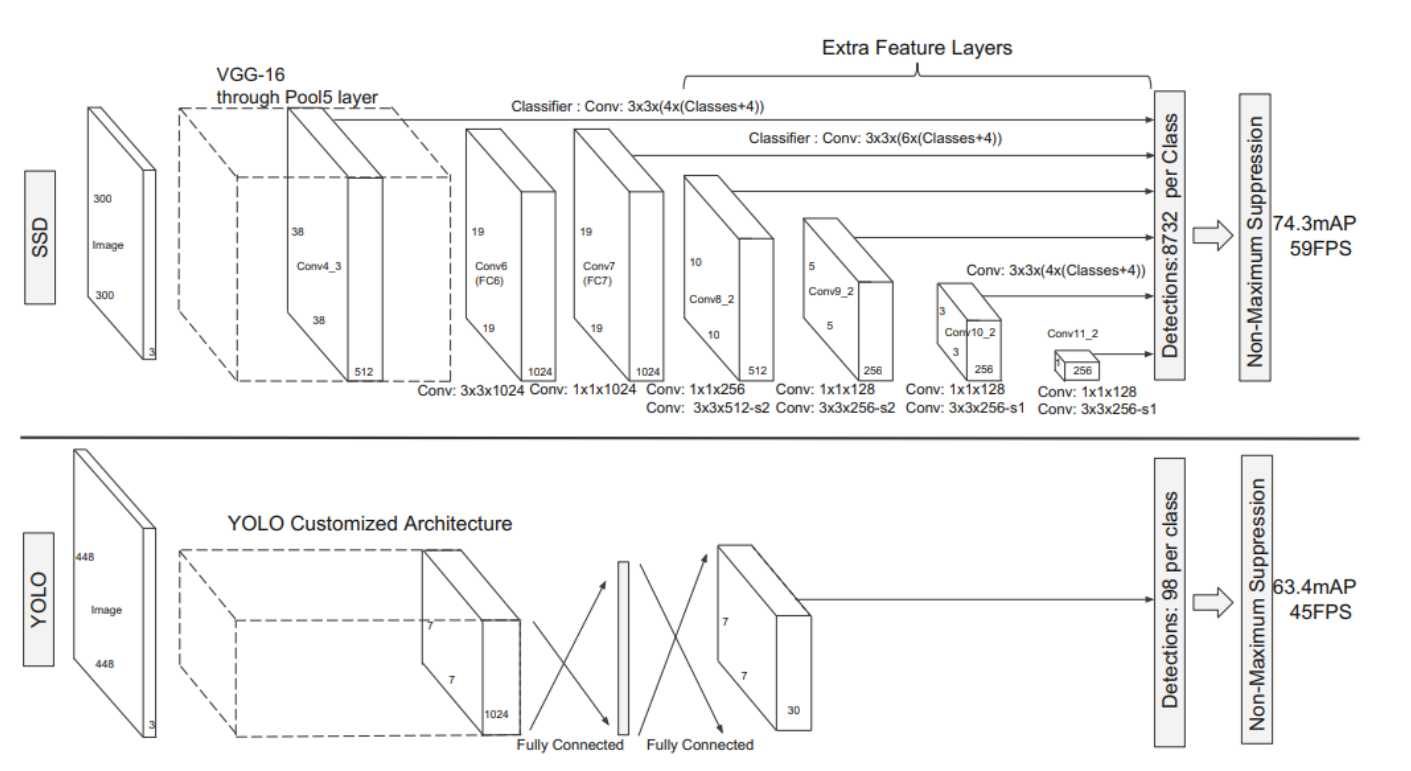
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

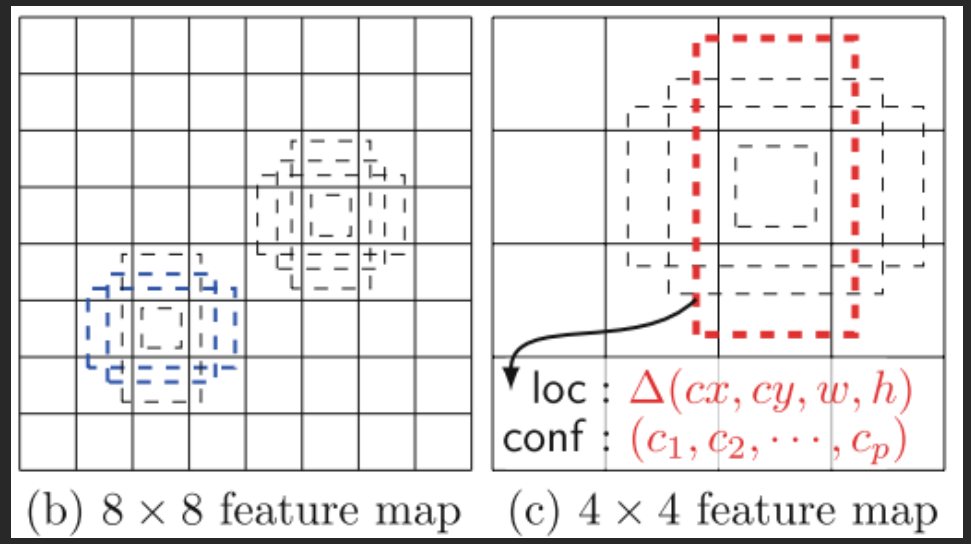
Table 1. **Darknet-53.**

Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

Содержание

SSD
Single Shot MultiBox Detector





Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. “SSD: Single shot multibox detector." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.

Содержание

Классификация с локализацией

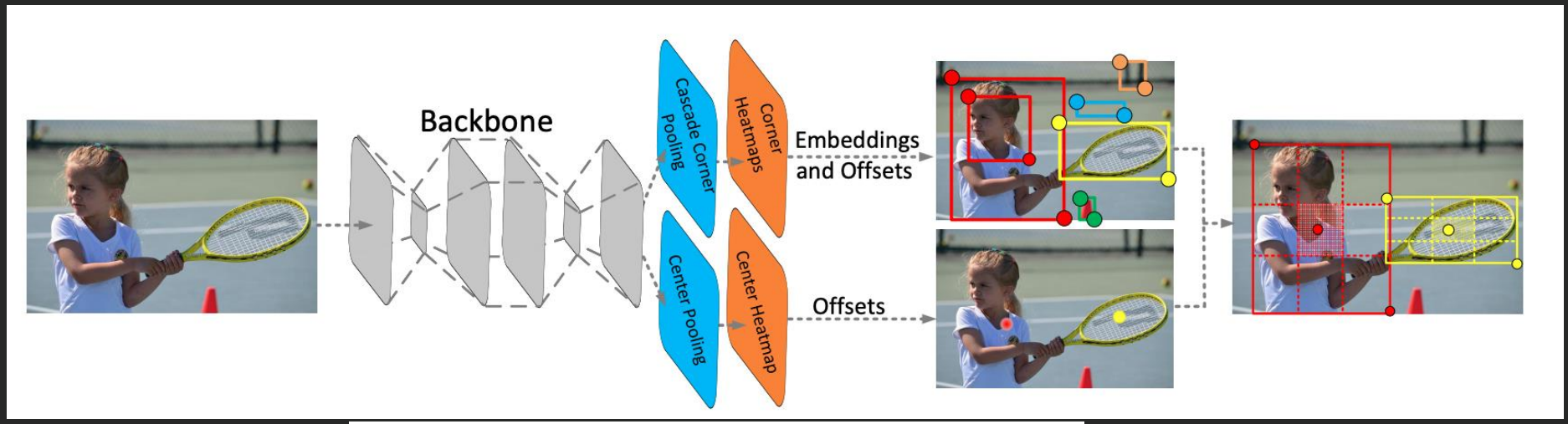
R-CNN

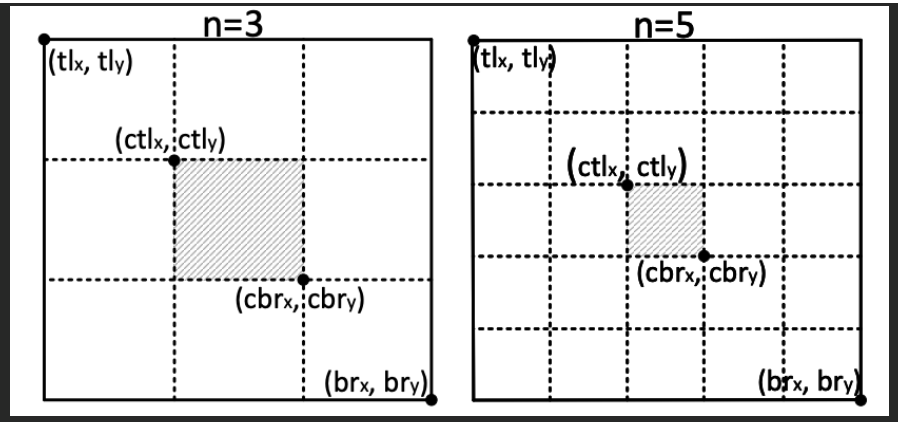
YOLO

SSD

CenterNet

CenterNet





Duan, Kaiwen, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. "Centernet: Keypoint triplets for object detection." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6569-6578. 2019.

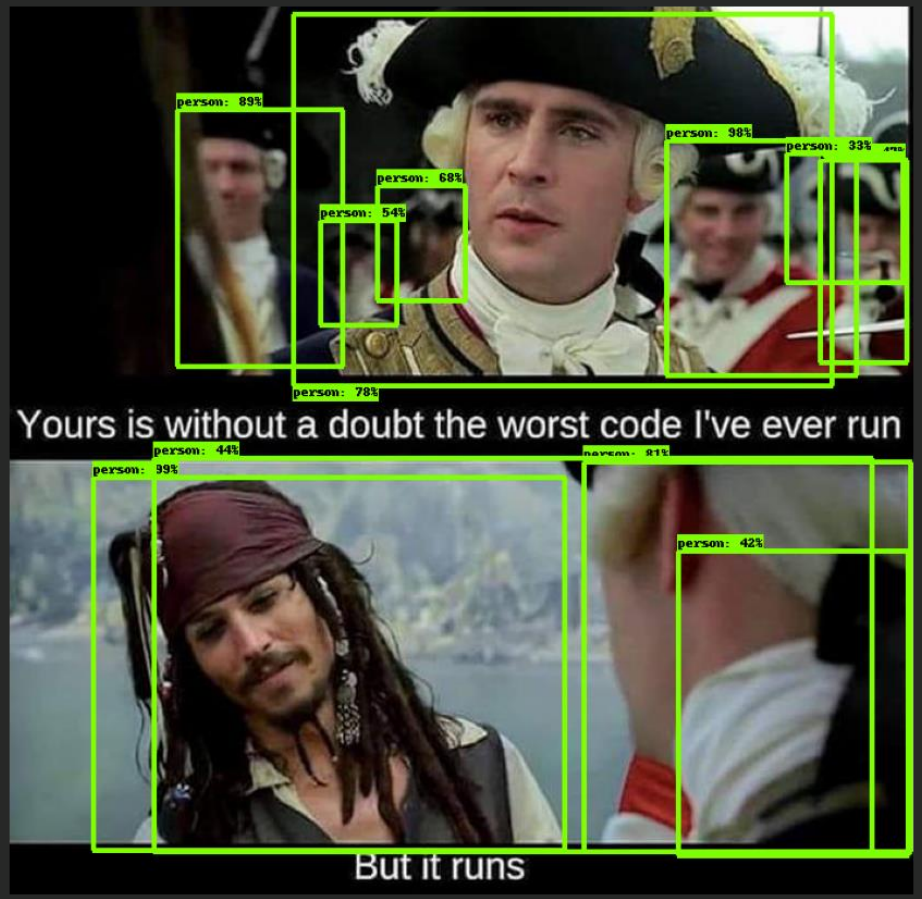
Tensorflow Hub

репозиторий обученных моделей нейронок

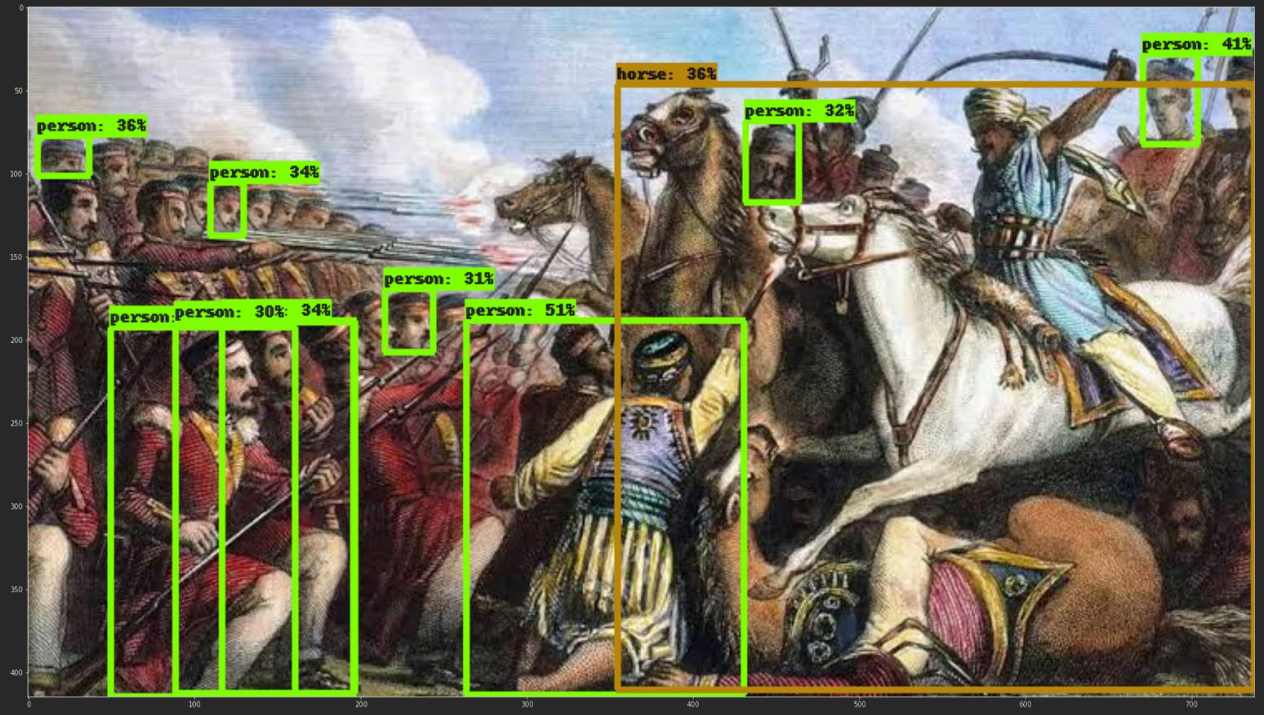
```
import tensorflow_hub as hub

hub_model = hub.load(model_handle)

results = hub_model(image)

result = {key:value.numpy() for key,value in results.items()}
```

https://www.tensorflow.org/hub

Tensorflow Hub

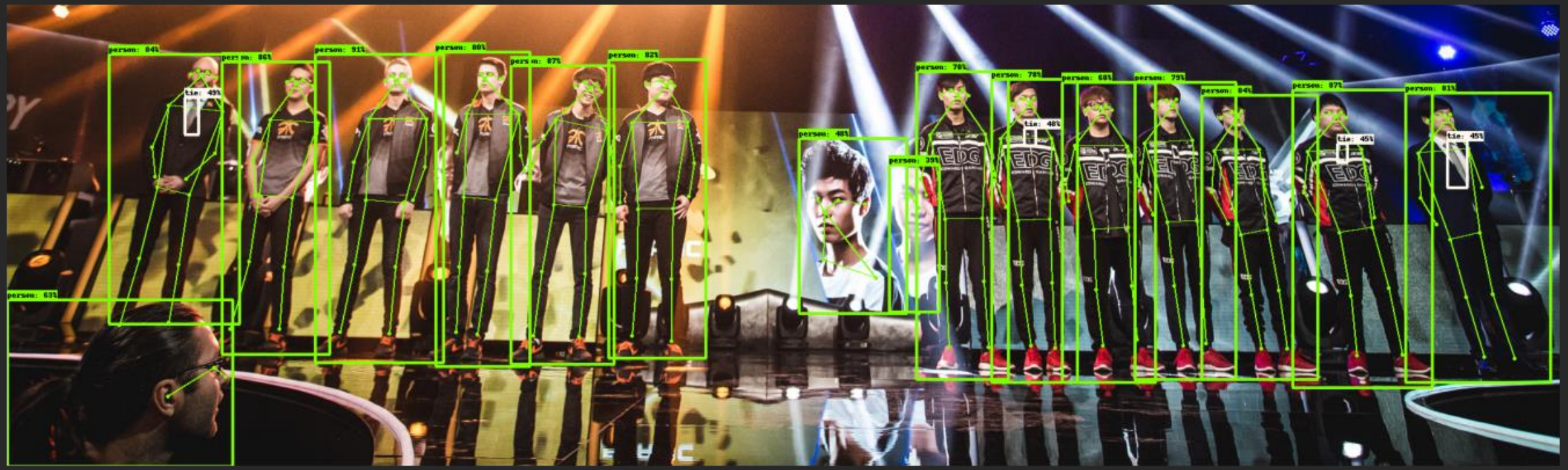


Faster R-CNN ResNet50 V1 1024x1024



CenterNet HourGlass104 Keypoints 1024x1024

Tensorflow Hub



CenterNet HourGlass104 Keypoints 1024x1024

Резюме

Классификация с локализацией

R-CNN: Регионы Кандидаты и Признаки от Сверток;

YOLO: Сетка, Рамки, Non-maximal suppression;
YOLOv2: Много «небольших» изменений которые приводят к успеху;
YOLO9000: WordTree;
YOLOv3: -и- добавляем разные масштабы;

SSD: разные масштабы (только больше), фиксированные рамки

CenterNet: ищем по триплетам (центр и два угла);

Вопросы, пожелания, предложения

?????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.03
Где мы изучаем как нейронные сети рисуют

Докладчик
Долганов Антон

В предыдущей лекции

LeNet-5

Классика, Свертки + Pooling + Полносвязные

AlexNet

Глубокое обучение не так уж и плохо, GPU

VGG

Много (очень) сверток 3x3

GoogLeNet

Свертки 1x1, Inception слои, упрощение сверток

ResNet

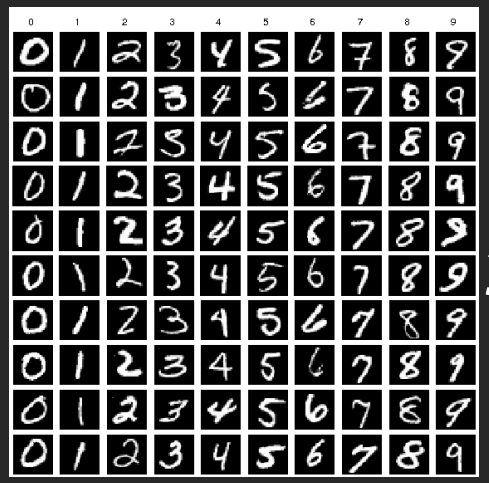
Пропуски (Residuals), можно еще глубже сети

MobileNet

меньше мат. операций «за ту же точность»

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

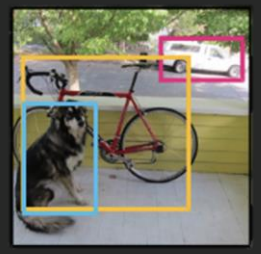
Выходные Слои Нейронных Сетей



$$y = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \end{bmatrix}$$

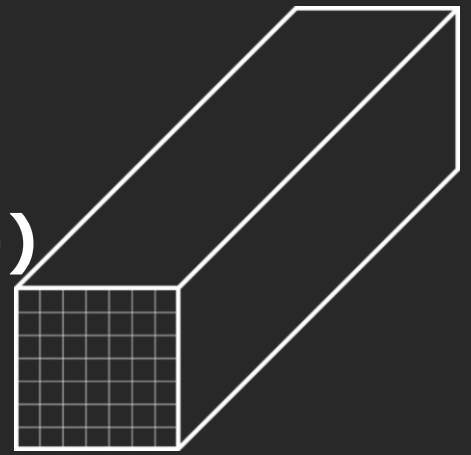
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$y = \begin{bmatrix} \dots \\ l_1 x \\ l_1 y \\ l_2 x \\ l_2 y \\ \dots \\ l_N x \\ l_N y \end{bmatrix}$$

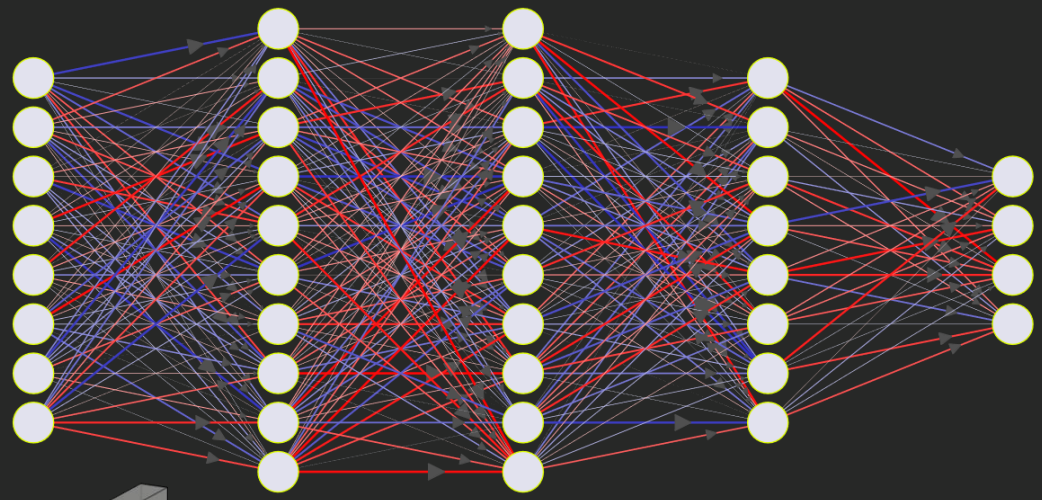


S × S × (B ∗5 + Классы)

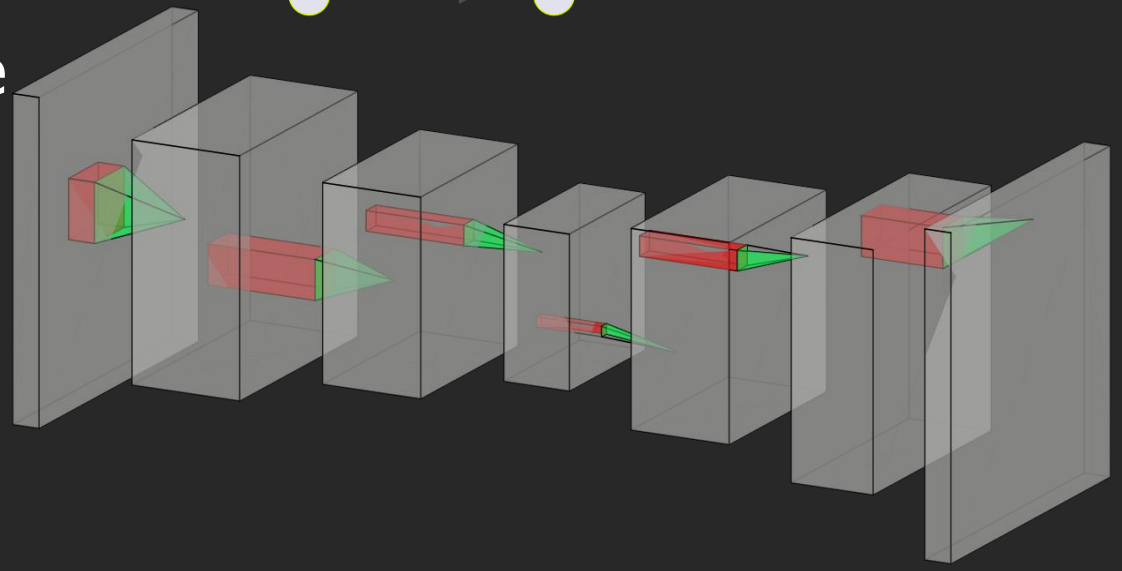
S × S × (AB*{1 + 4 + Классы})



Может ли полноценное изображение быть выходом



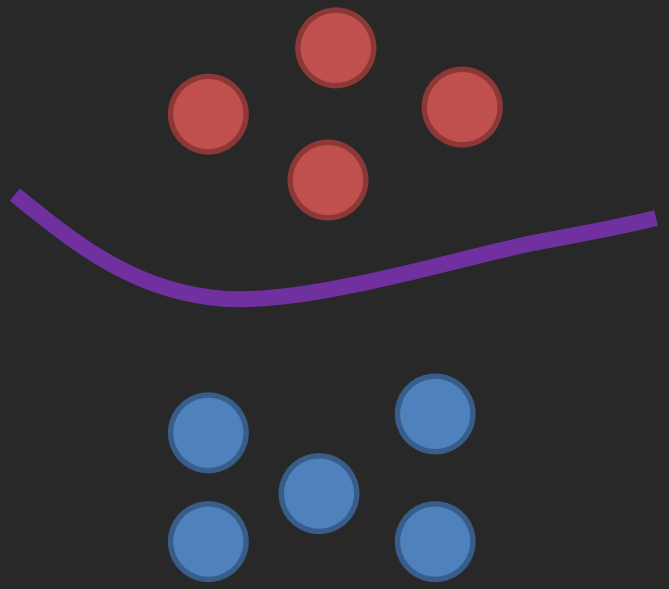
Что-то на входе



Новое Изображение

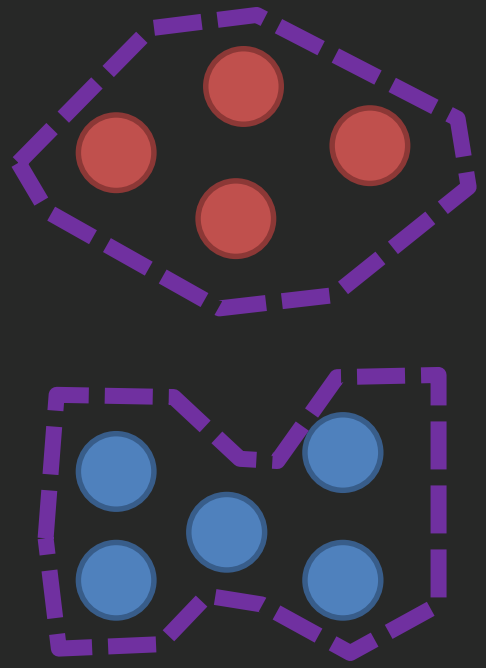
Генеративные и Дискриминационные модели

Дискриминационные модели



- Деревья Решений
- Логистическая Регрессия
- Ближайшие Соседи

Генеративные модели



- Наивный Байес
- Дискриминантный Анализ

Какой кот Настоящий?

https://thiscatdoesnotexist.com/

https://thisxdoesnotexist.com/

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Генеративные Модели

Перенос Стиля

Содержание

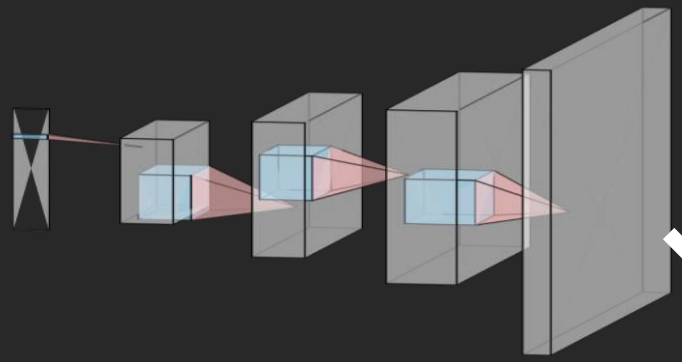
Генеративные Модели

GANы

Перенос Стиля

GAN Generative Adversarial Network

Генеративно-Состязательные сети

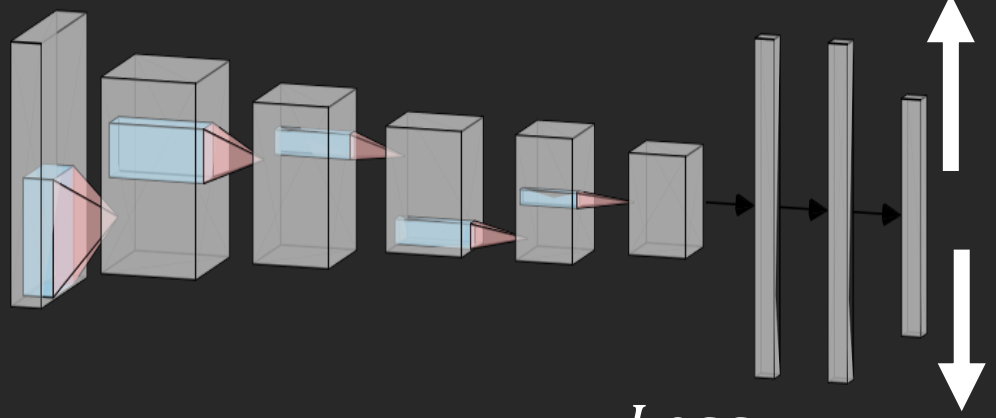


Генератор

- Пытается подделать настоящее изображение
- То что получается хорошо используется как Отрицательный класс для дискриминатора

Дискриминатор

- Учится различать Подделки от Реальных изображений
- Обратная связь генератору «делай лучше»



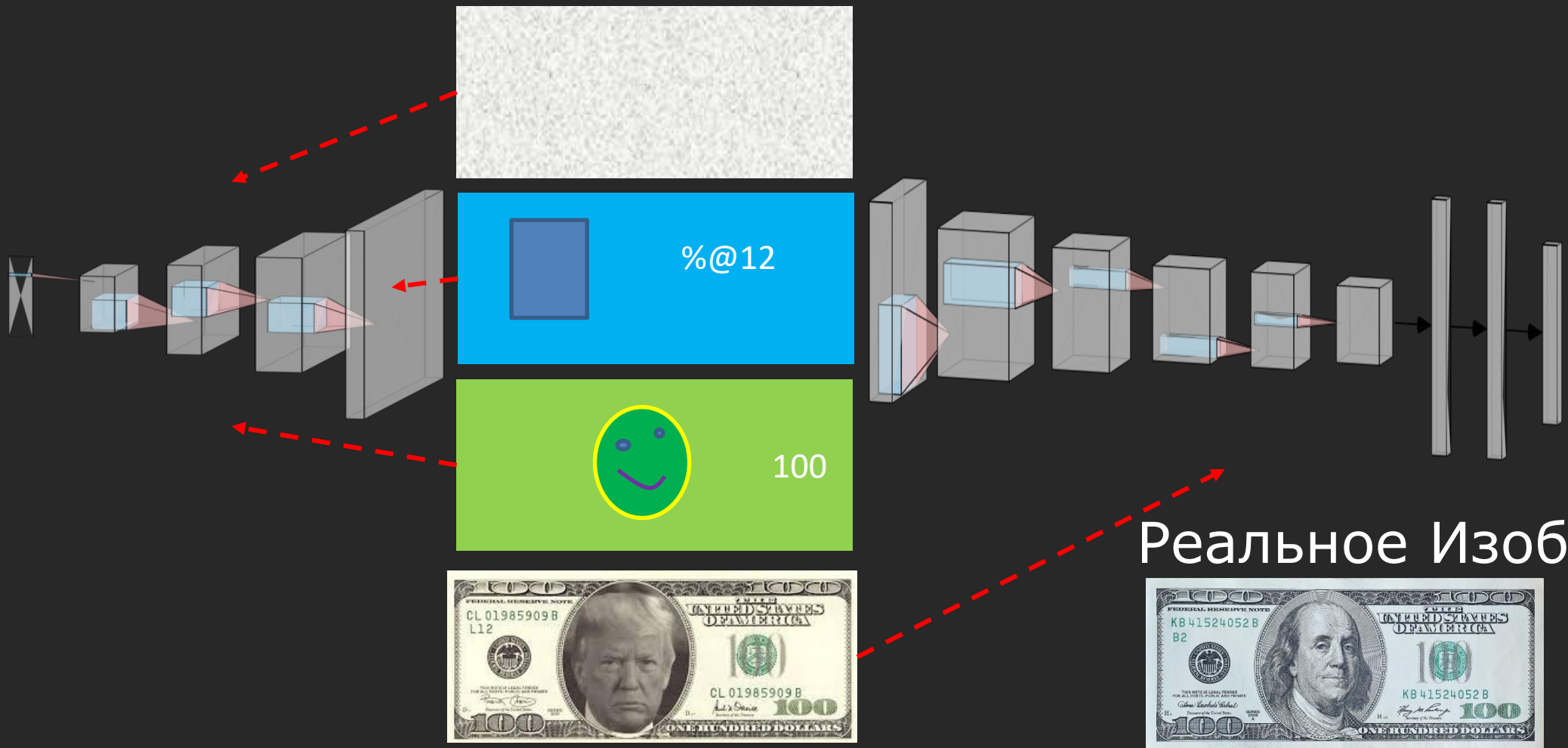
$Loss_{Generator} \sim BCE_{max}$

$Loss_{Discriminator} \sim BCE_{min}$

Реальные Изображения

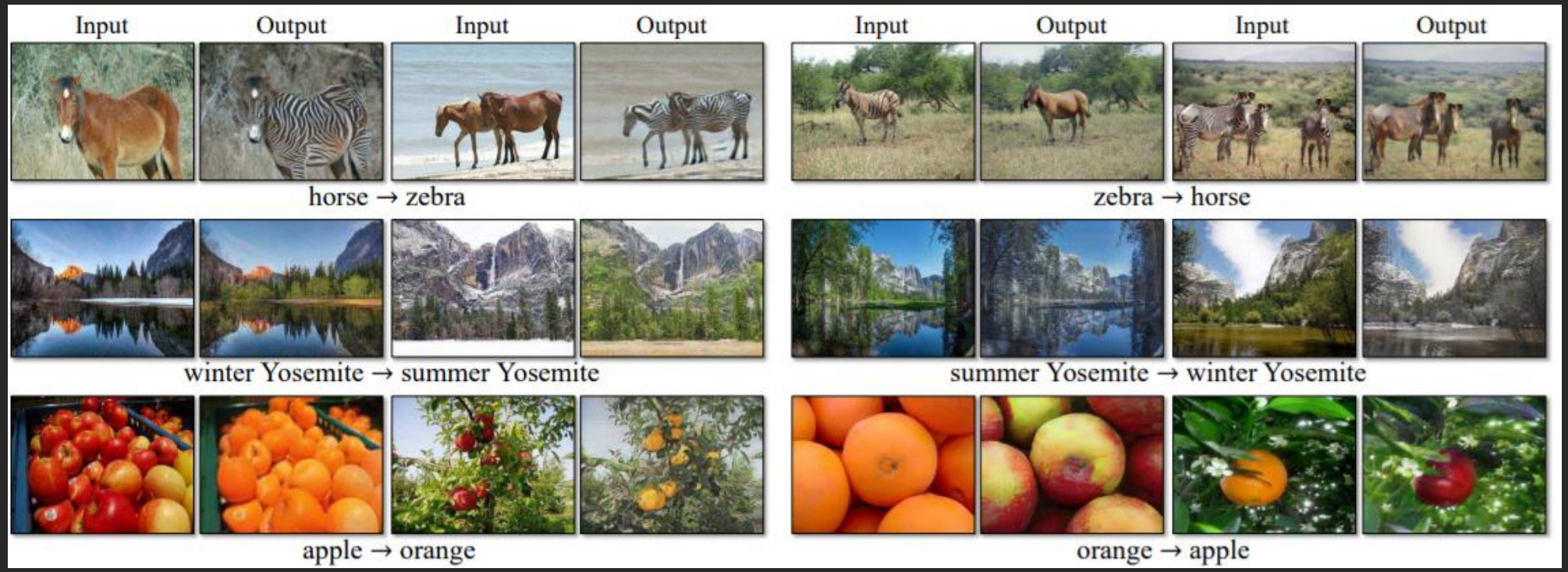
Генеративно-Состязательные сети

.



Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

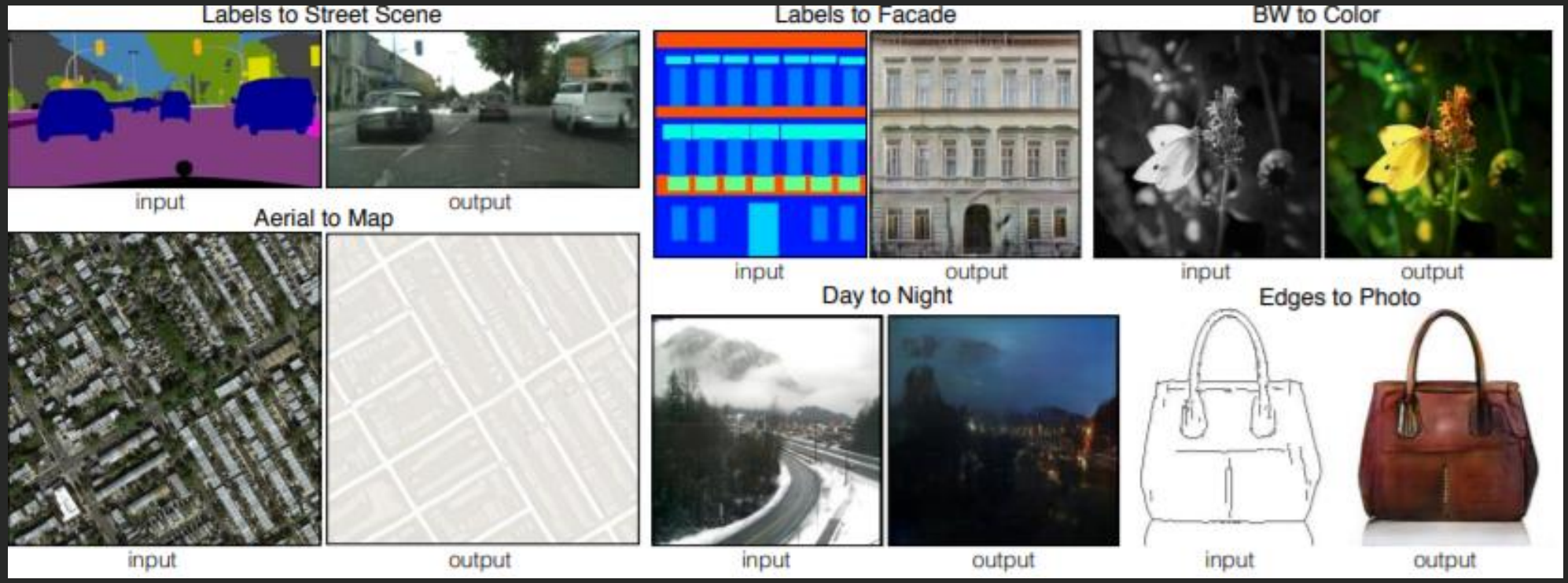
Цикличные GAN
Cycle-Consistent Adversarial Networks



https://openaccess.thecvf.com/content_ICCV_2017/papers/Zhu_Unpaired_Image-To-Image_Translation_ICCV_2017_paper.pdf

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." In *Proceedings of the IEEE international conference on computer vision*, pp. 2223-2232. 2017.

«Перевод Изображений»

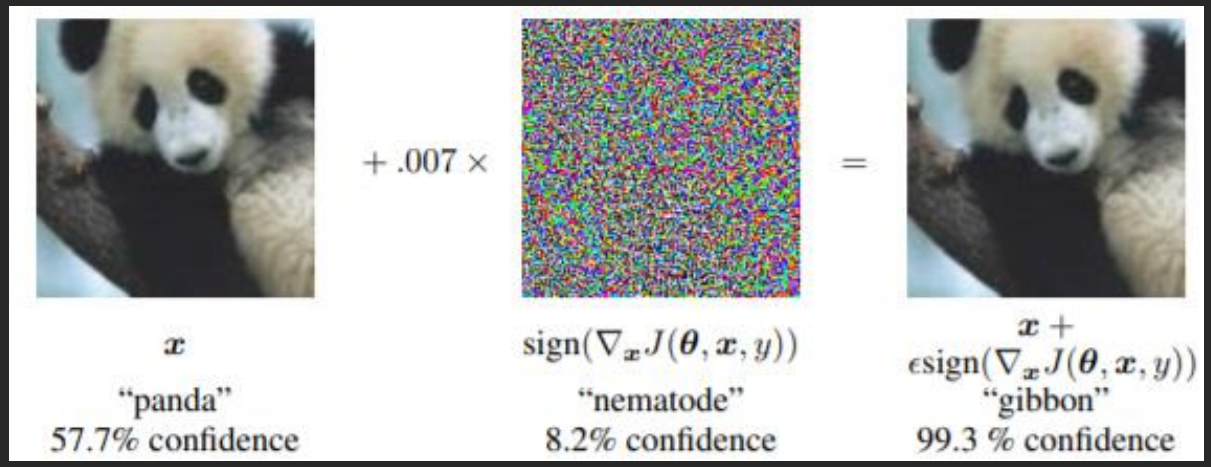


https://arxiv.org/pdf/1611.07004.pdf

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-image translation with conditional adversarial networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125-1134. 2017.

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

«Атаки» на Нейронные Сети



https://arxiv.org/pdf/1412.6572.pdf

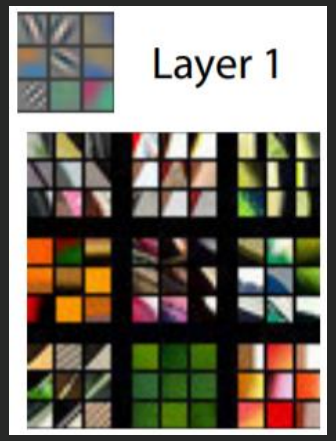
Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014).

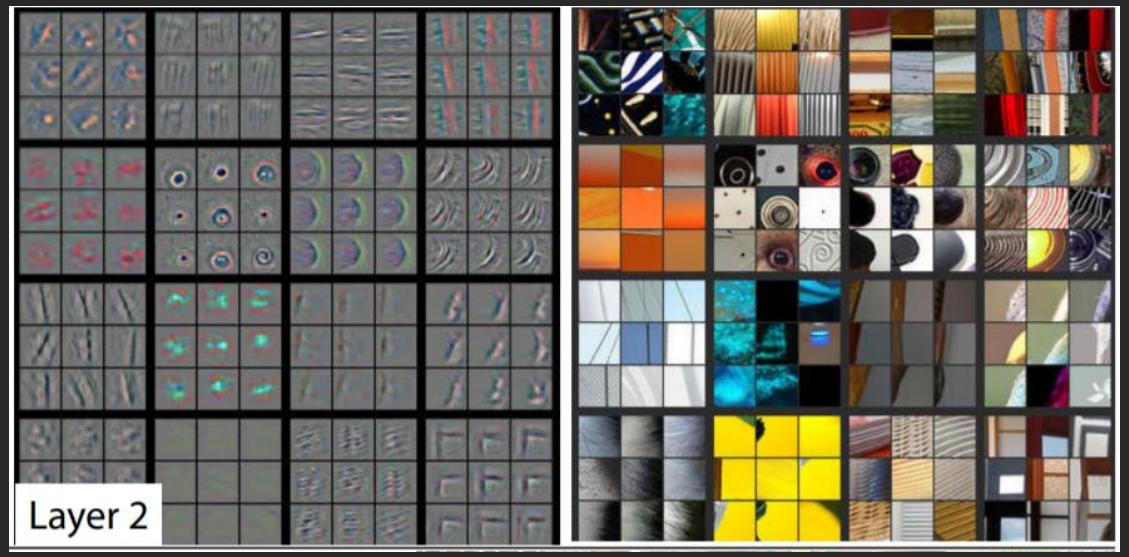
Su, Jiawei, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks." *IEEE Transactions on Evolutionary Computation* 23, no. 5 (2019): 828-841.

https://arxiv.org/pdf/1710.08864.pdf

Содержание

Нейронные Сети
«Визуализация Слоев»

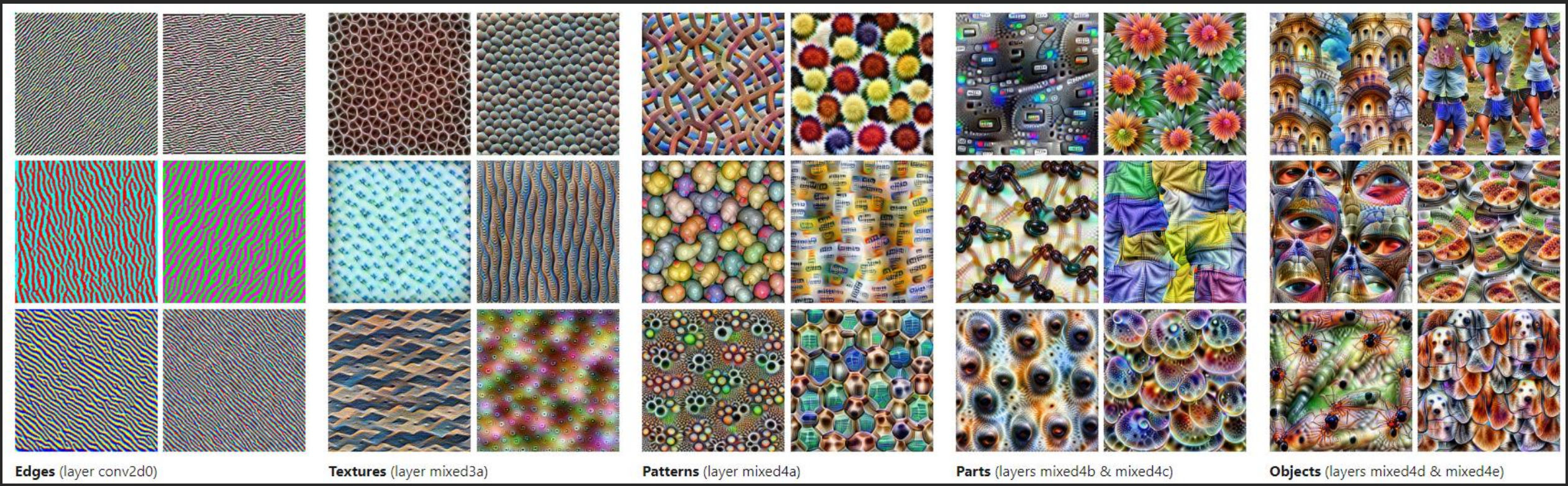




Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." In *European conference on computer vision*, pp. 818-833. Springer, Cham, 2014.

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

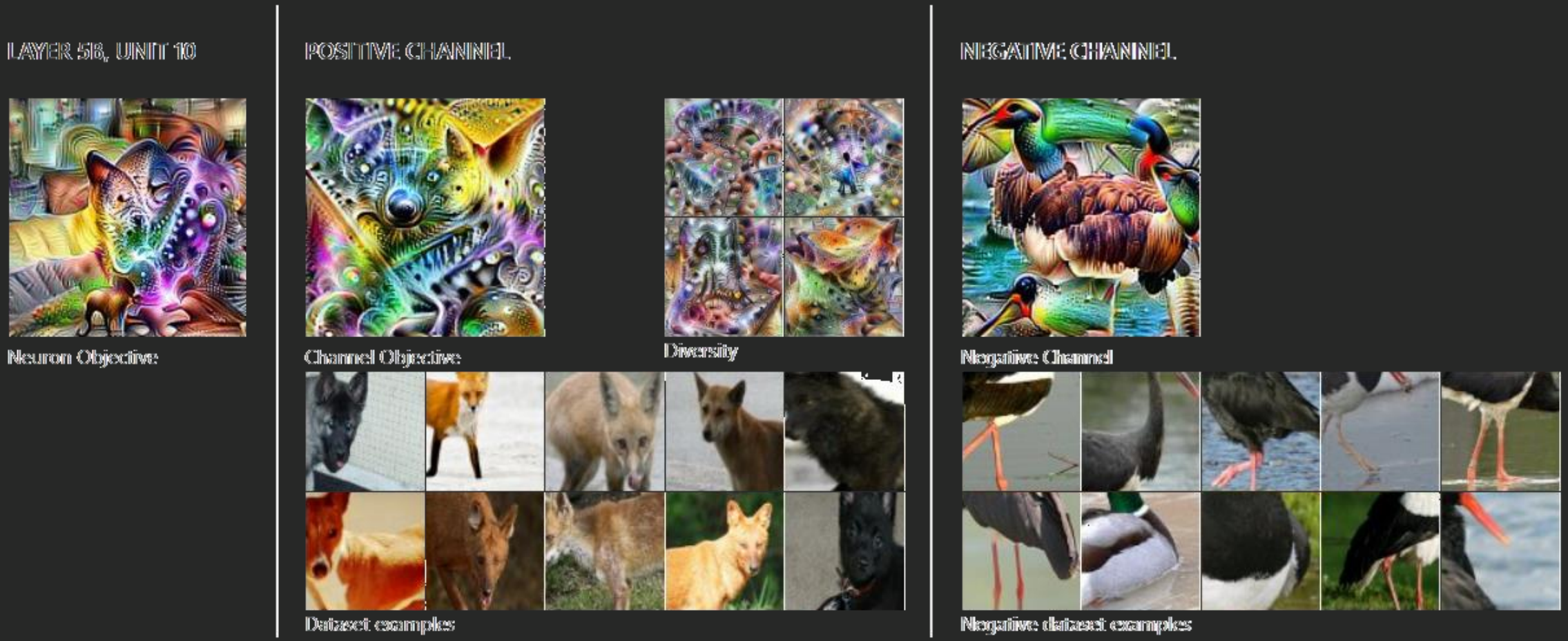
Нейронные Сети «Визуализация Слоев»



https://distill.pub/2017/feature-visualization/

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Нейронные Сети «Визуализация Слоев»



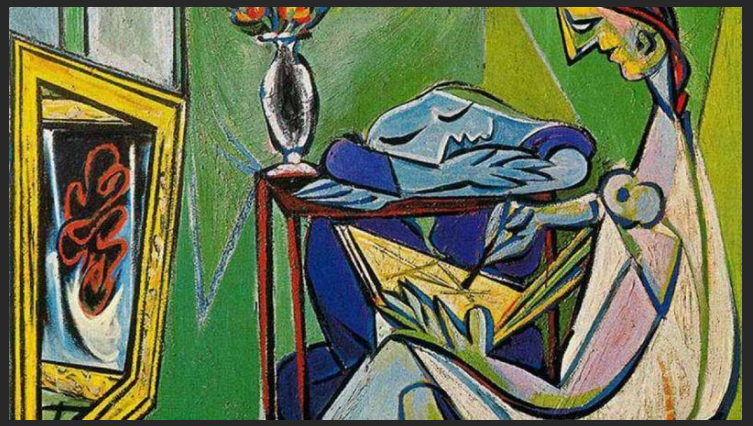
https://distill.pub/2017/feature-visualization/

Вспоминаем МХК

Vincent van Gogh

Andy Warhol

Pablo Picasso

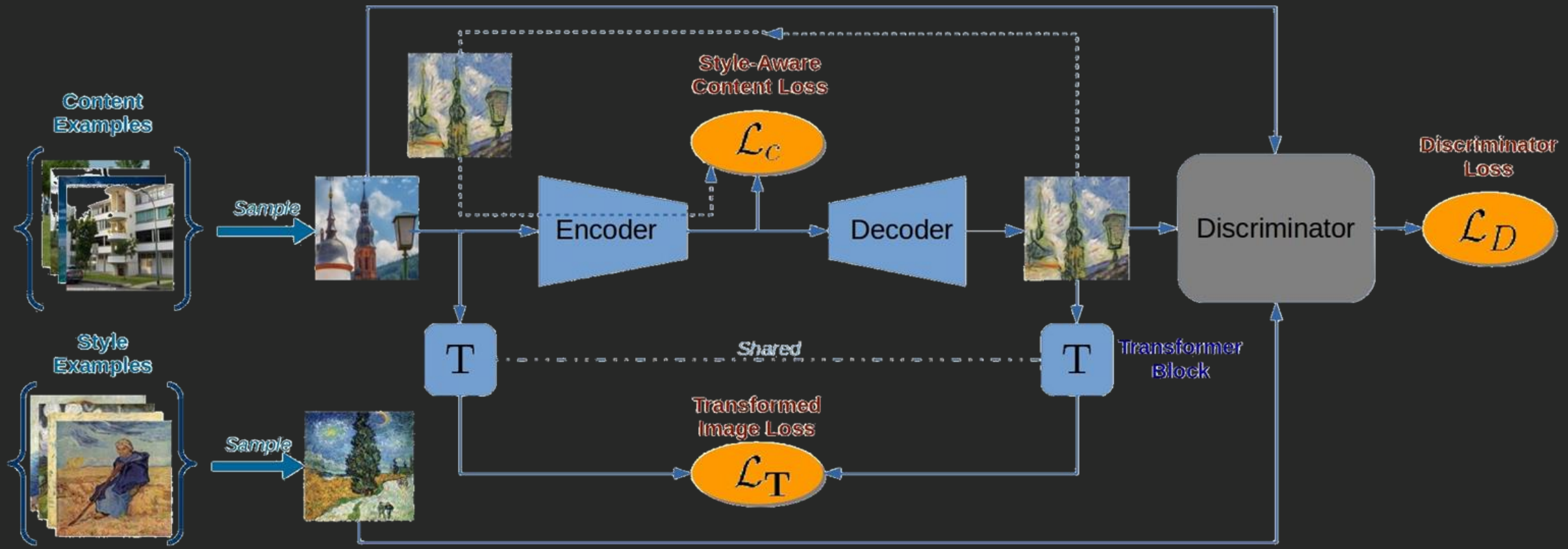


Leonardo da Vinci

Claude Monet

Виктор Васнецов

Перенос Стиля



https://neurohive.io/en/state-of-the-art/a-style-aware-content-loss-for-real-time-hd-style-transfer/

Перенос Стиля

https:// Нашел когда-то в интернетах

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Перенос Стиля

Original

Style

Styled Image

Original content image

Style image

Stylized image

Original content image

Style image

Stylized image

Original content image

Style image

Stylized image

Tensorflow Hub

репозиторий обученных моделей нейронок

```
import tensorflow_hub as hub

hub_model = hub.load(model_handle)

results = hub_model(image)

result = {key:value.numpy() for key,value in results.items()}
```

https://www.tensorflow.org/hub

Содержание

Генеративные Модели

Автокодировщики

Перенос Стиля

Вложения и Представления

.

Метод Главных Компонент

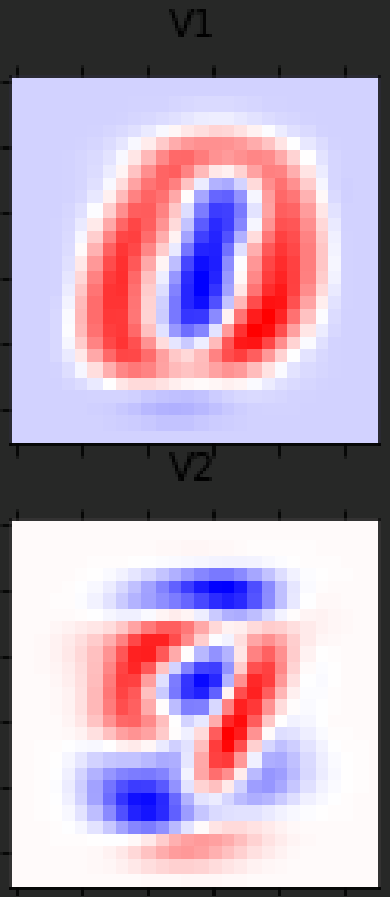
$$\Sigma = \frac{1}{n} X' X'^{T}$$

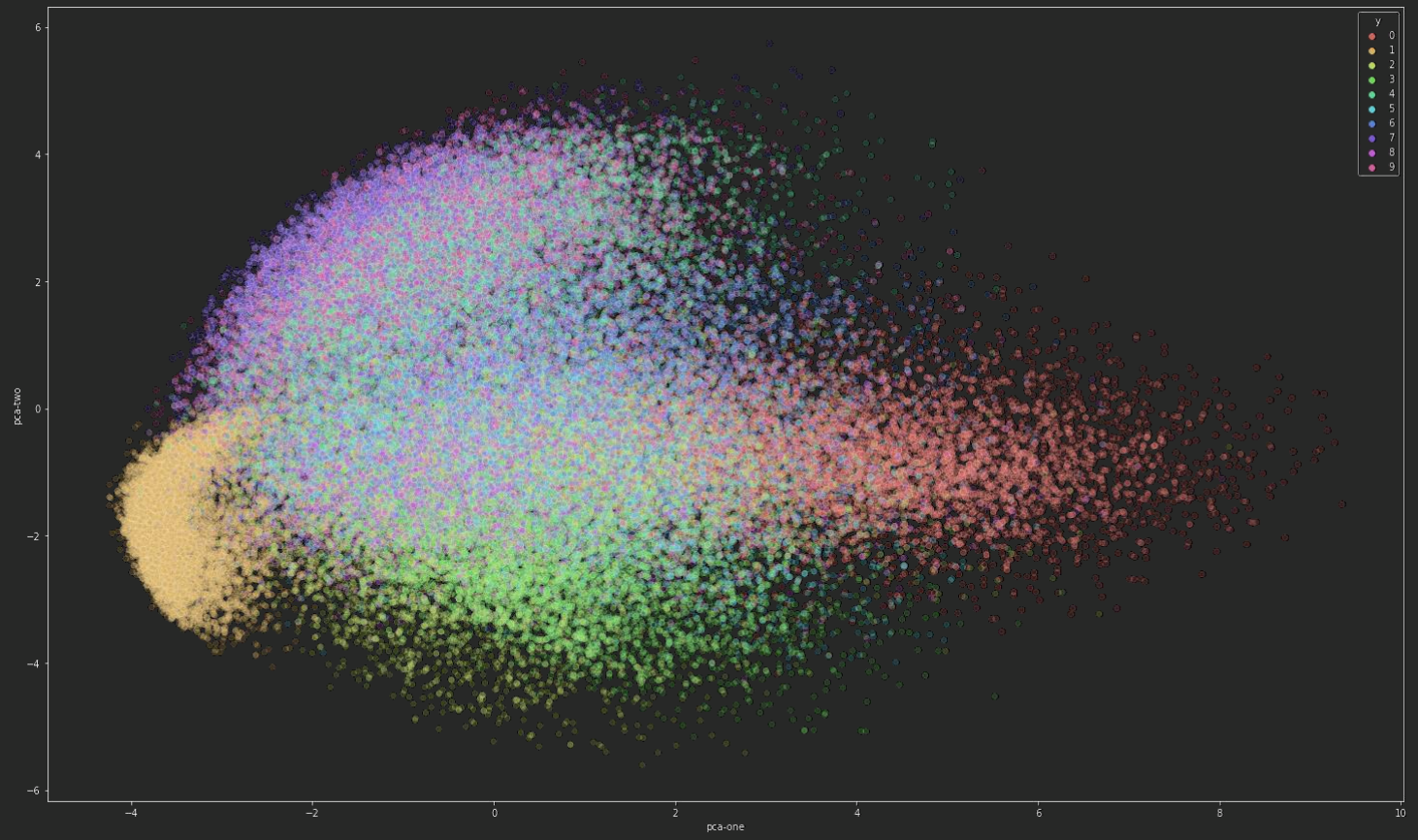
$$\Sigma \vec{v} = \lambda \vec{v}$$

$$\Lambda = \lambda I \in \mathrm{R}^{m \times m}$$

$$\Sigma = \mathrm{W} \Lambda W^{-1}$$

$$\mathrm{Y} = X' * \mathrm{W}^{T}$$





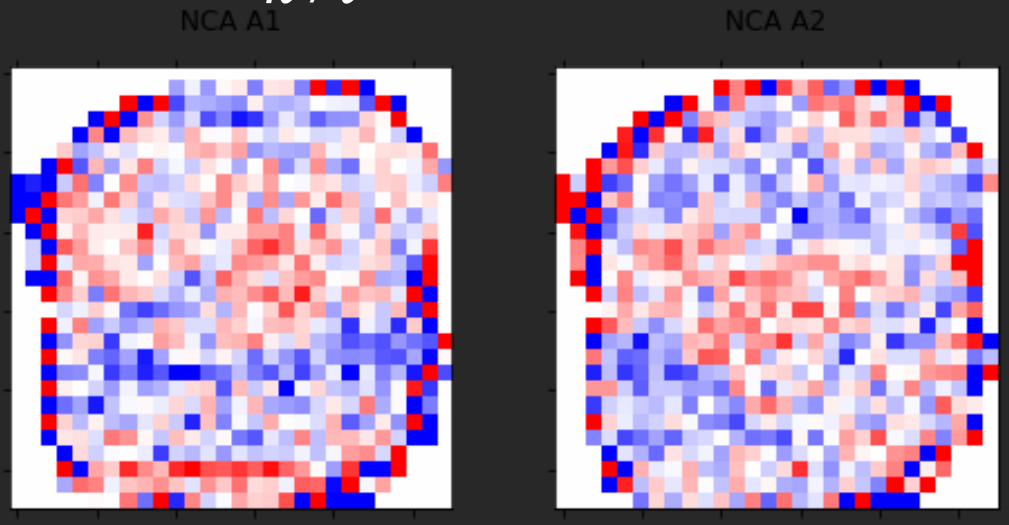
Вложения и Представления

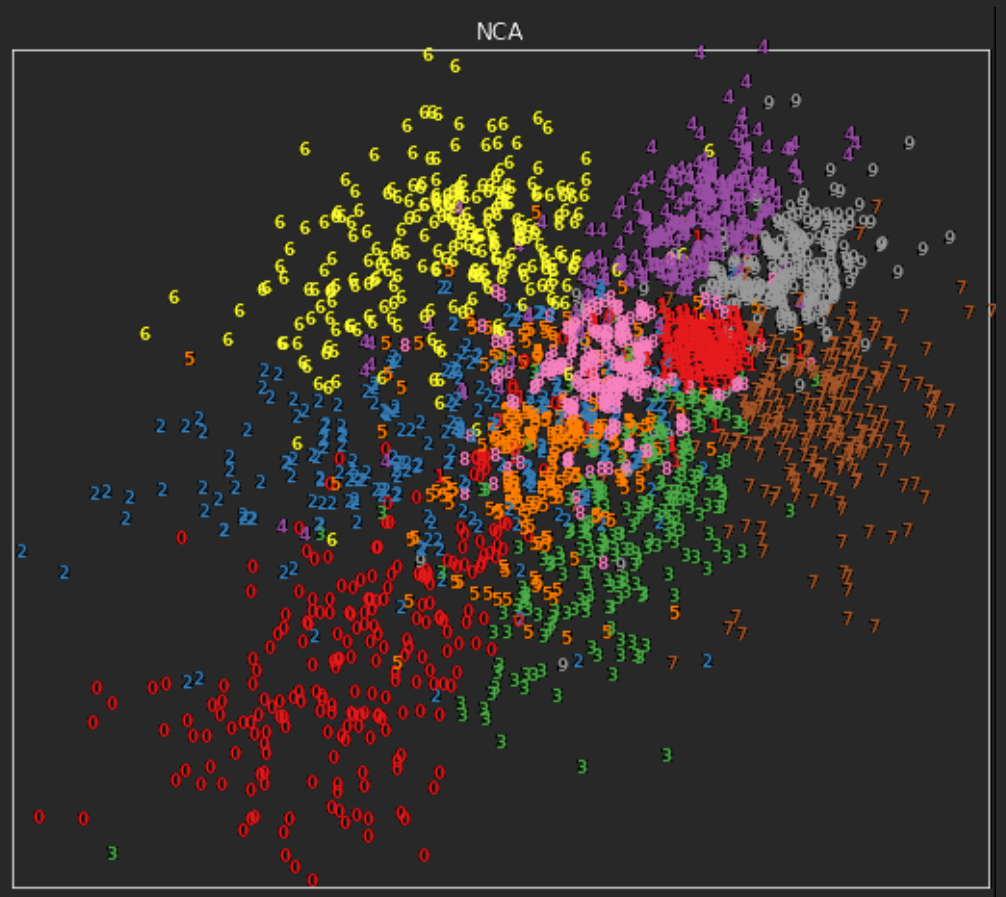
.

Neighborhood Components Analysis

$$Y = AX$$

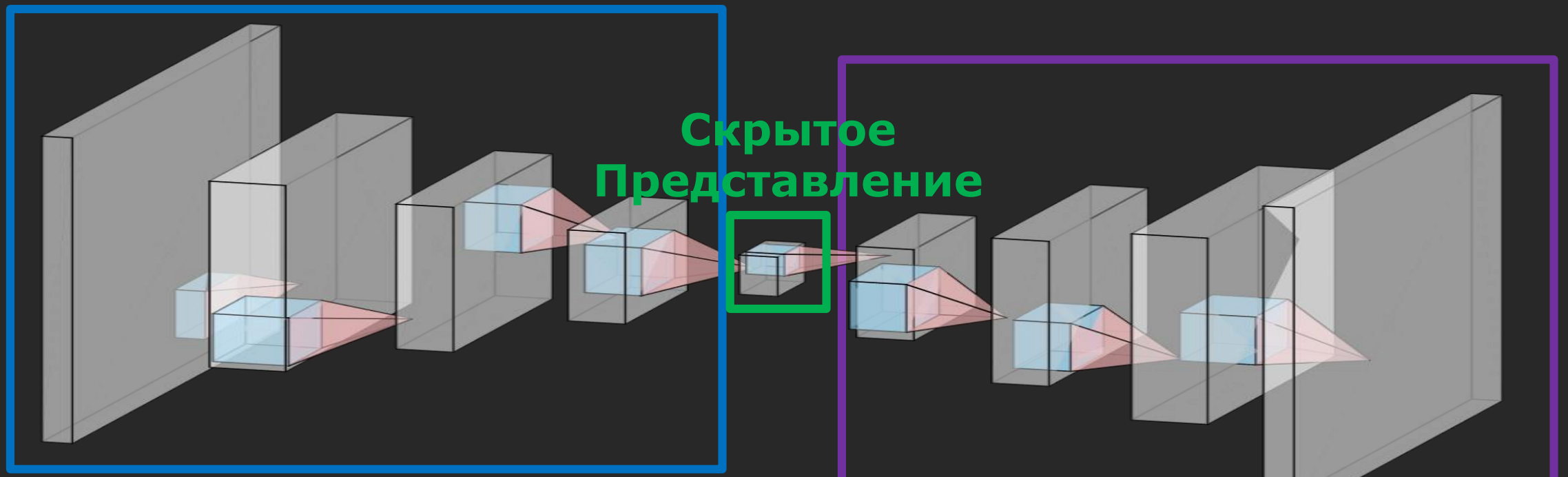
$$p_{ij} = \frac{e^{(-||Ax_i - Ax_j||^2)}}{\sum_{k \neq i} e^{(-||Ax_i - Ax_k||^2)}}$$





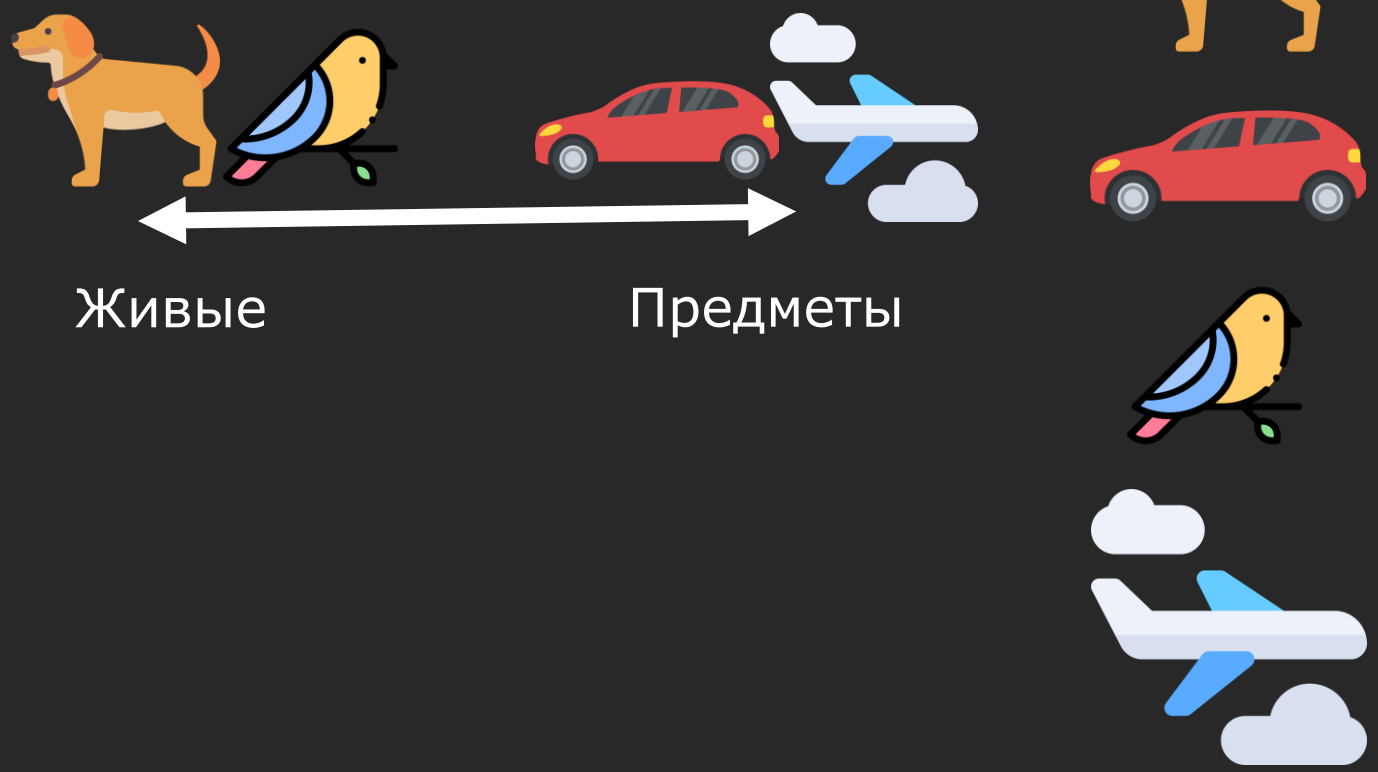
Auto-Encoders

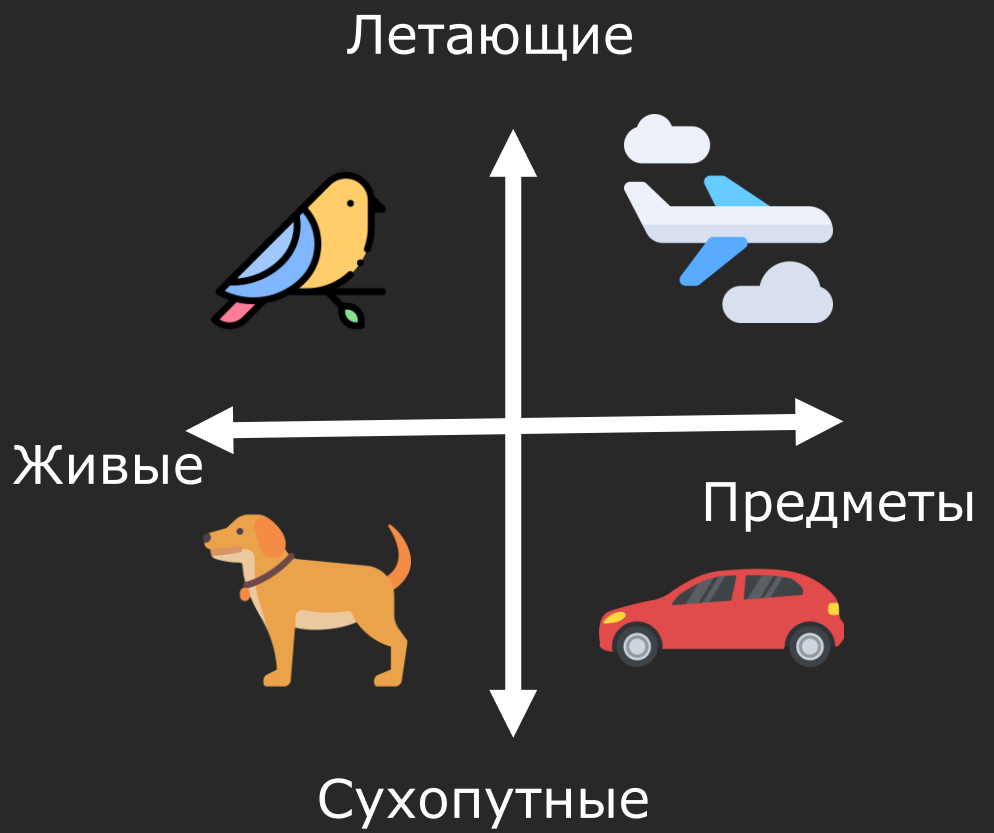
Авто-Энкодеры



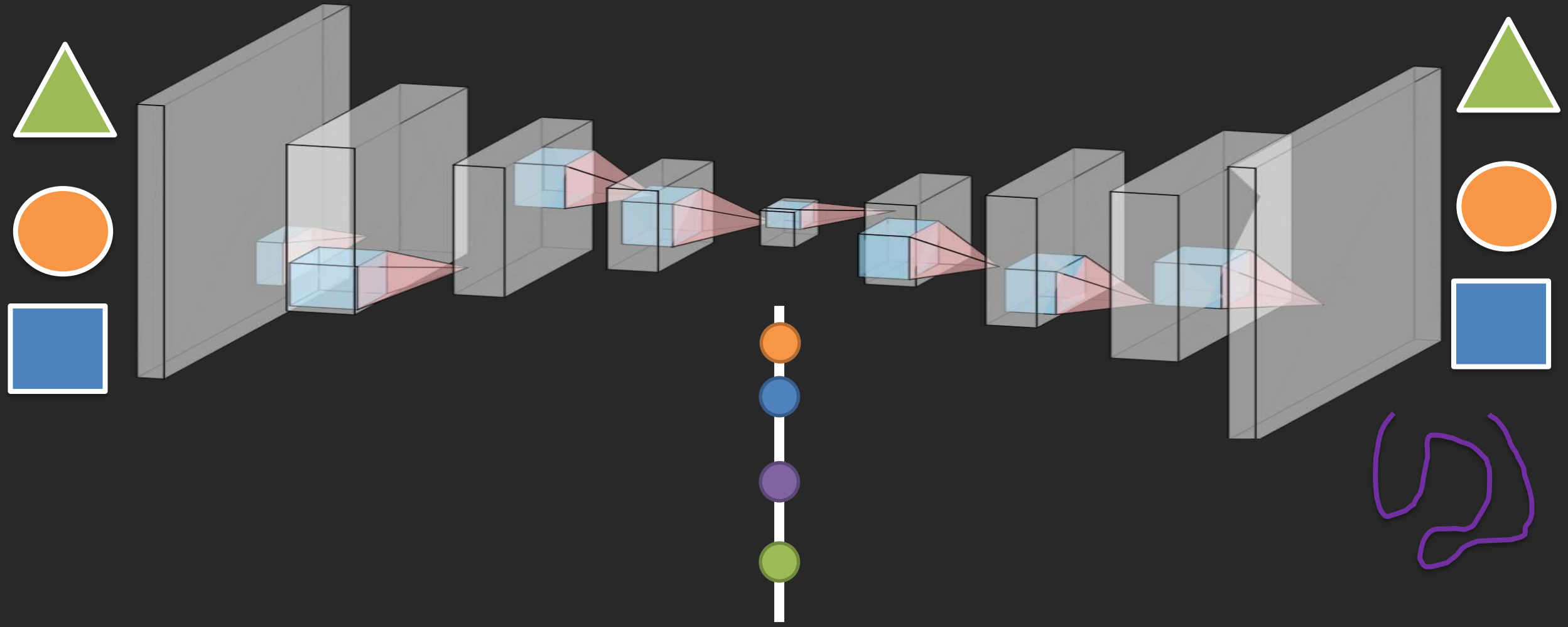
$$Loss \sim MSE(\text{Вход} - \text{Выход})$$

Скрытое Представление



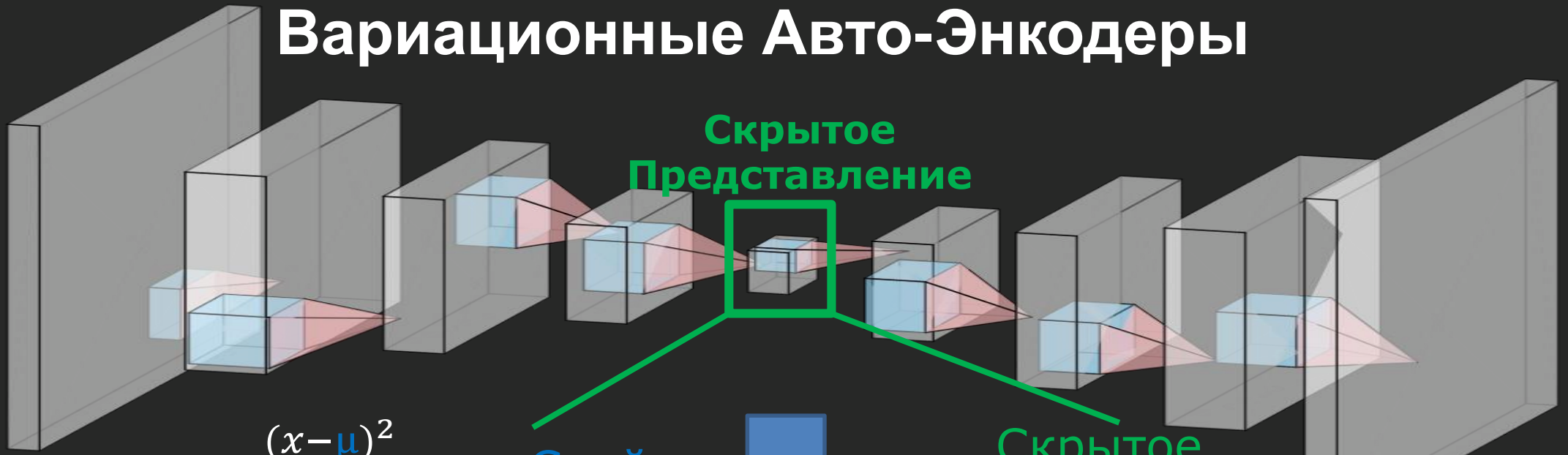


Авто-Энкодеры



Variational Auto-Encoders

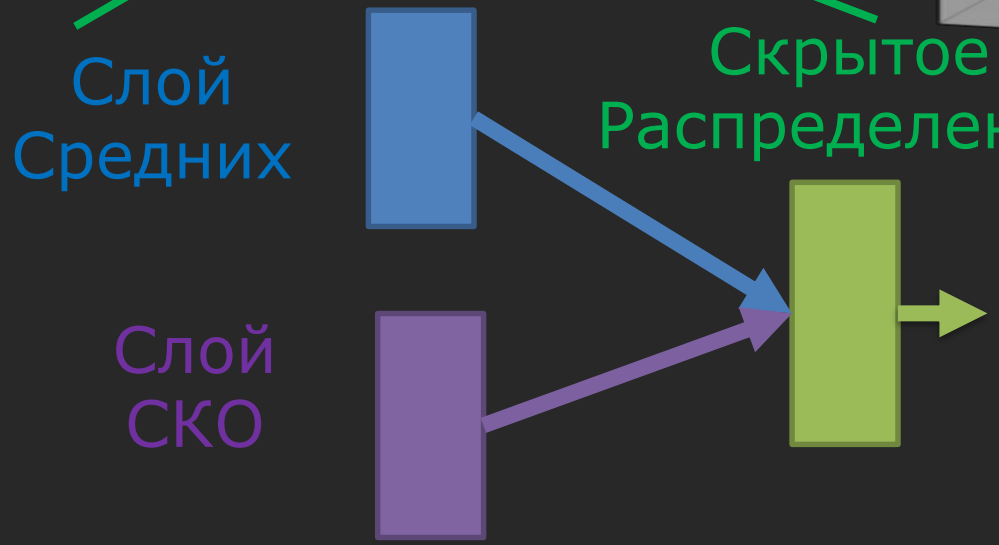
Вариационные Авто-Энкодеры



$$G(\mu,\sigma) = C * e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

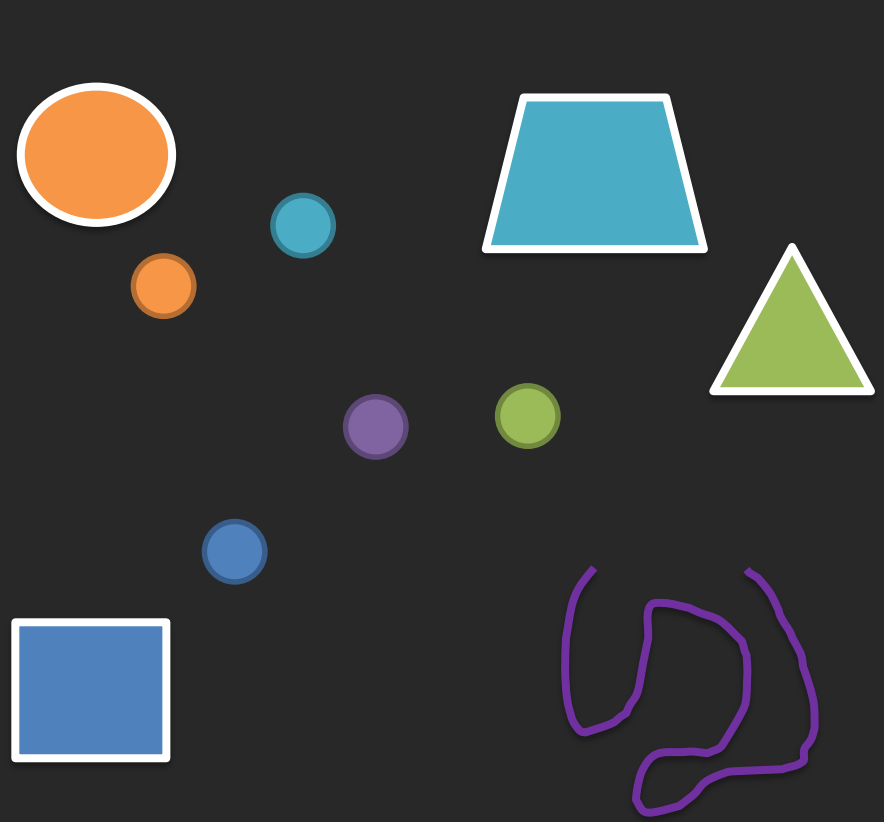
$$Loss \sim MSE(\text{Вход} - \text{Выход}) + D_{K|L}\{G(\mu,\sigma), G(0,1)\}$$

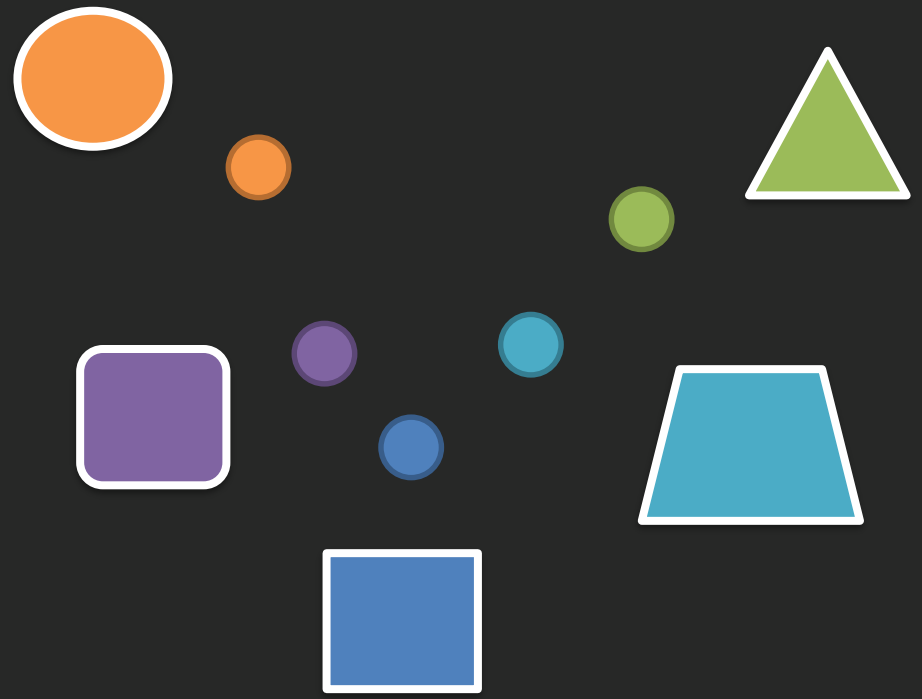
Дивергенция Кульбака – Лейблера



Вариационные Авто-Энкодеры

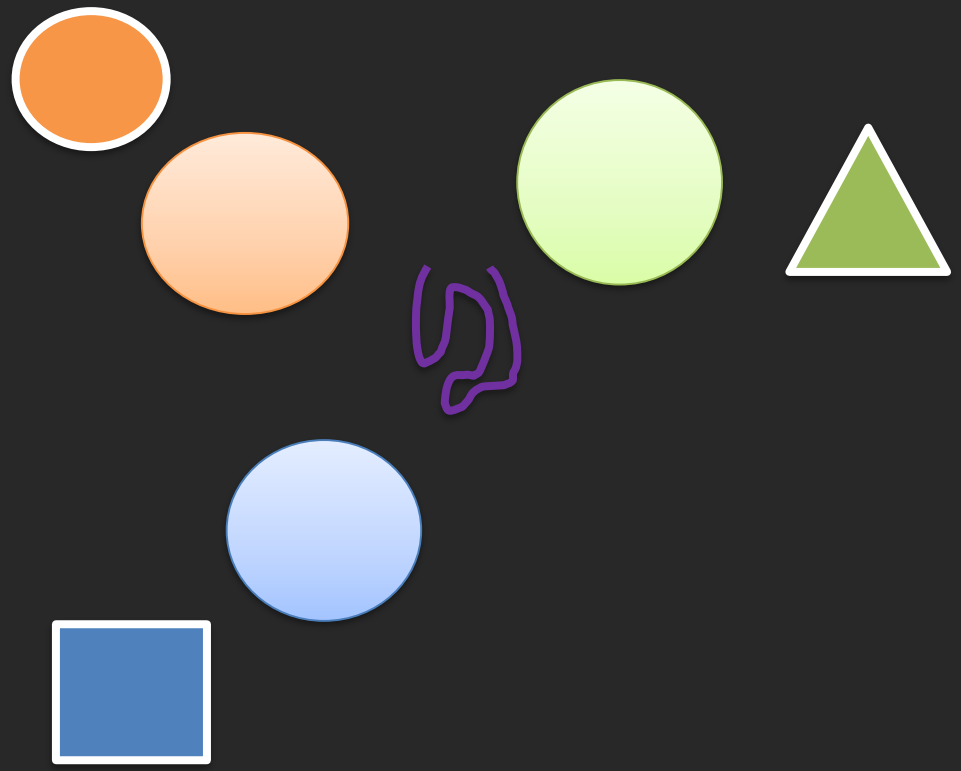
Что дает «регуляризация» Скрытого Распределения

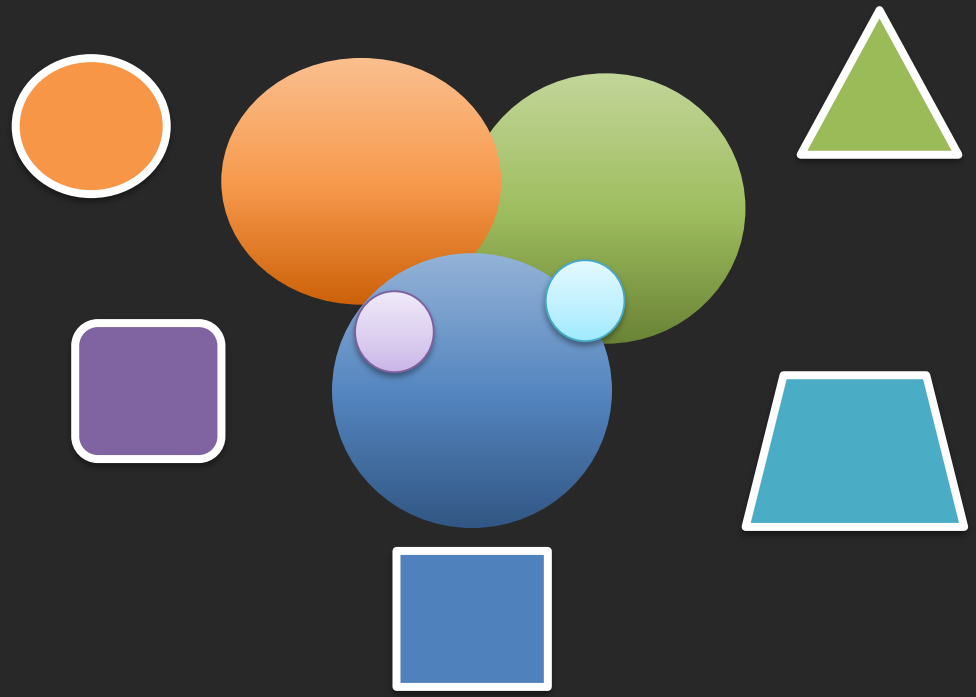




Вариационные Авто-Энкодеры

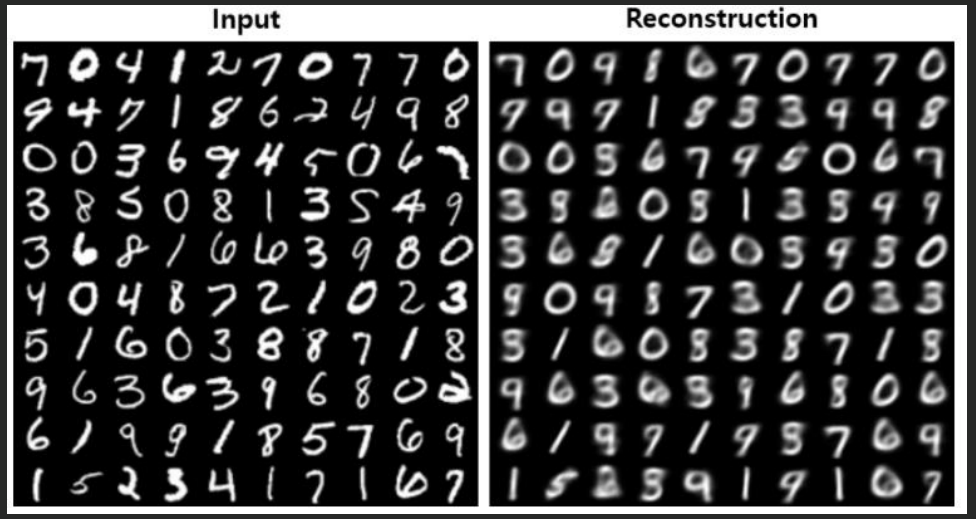
Что дает «регуляризация» Скрытого Распределения



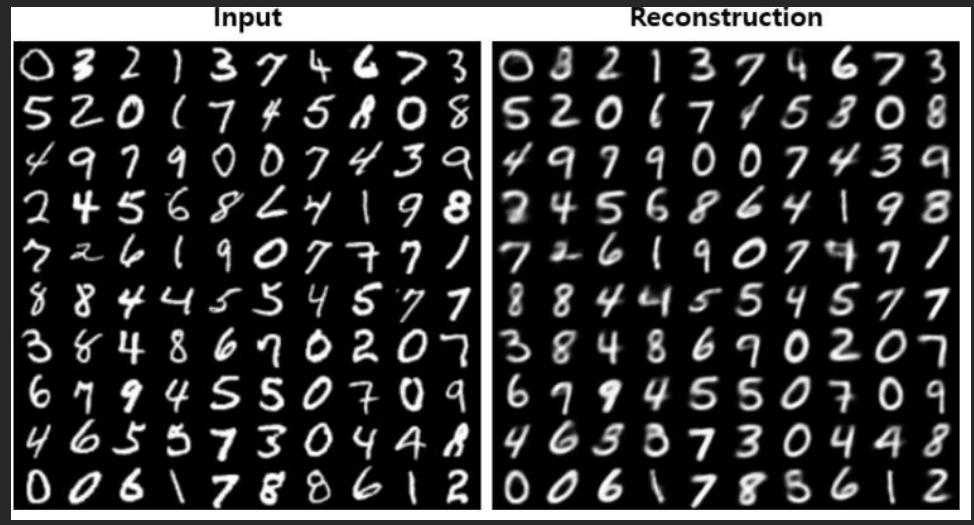


Вариационные Авто-Энкодеры

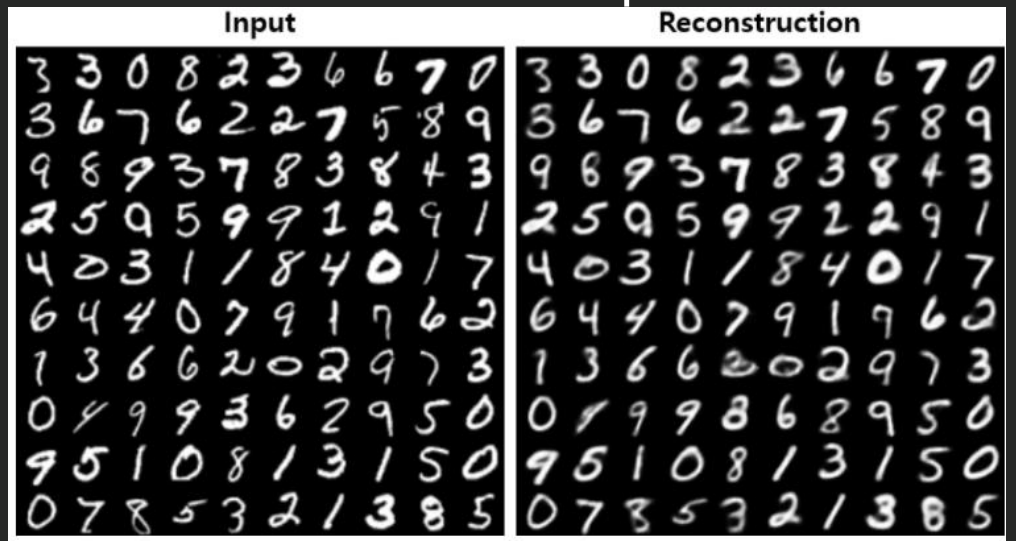
•2-D latent space



•5-D latent space



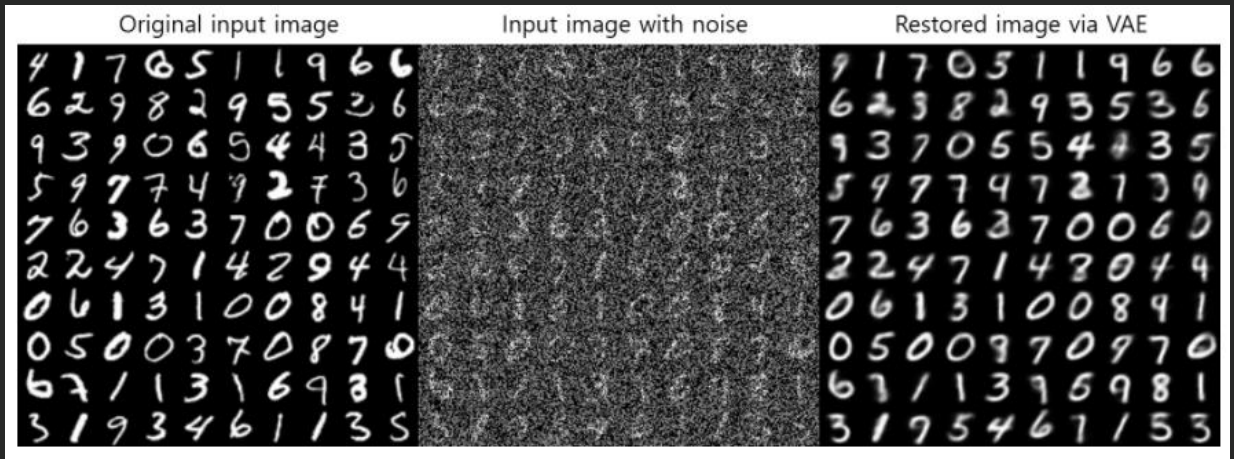
•10-D latent space

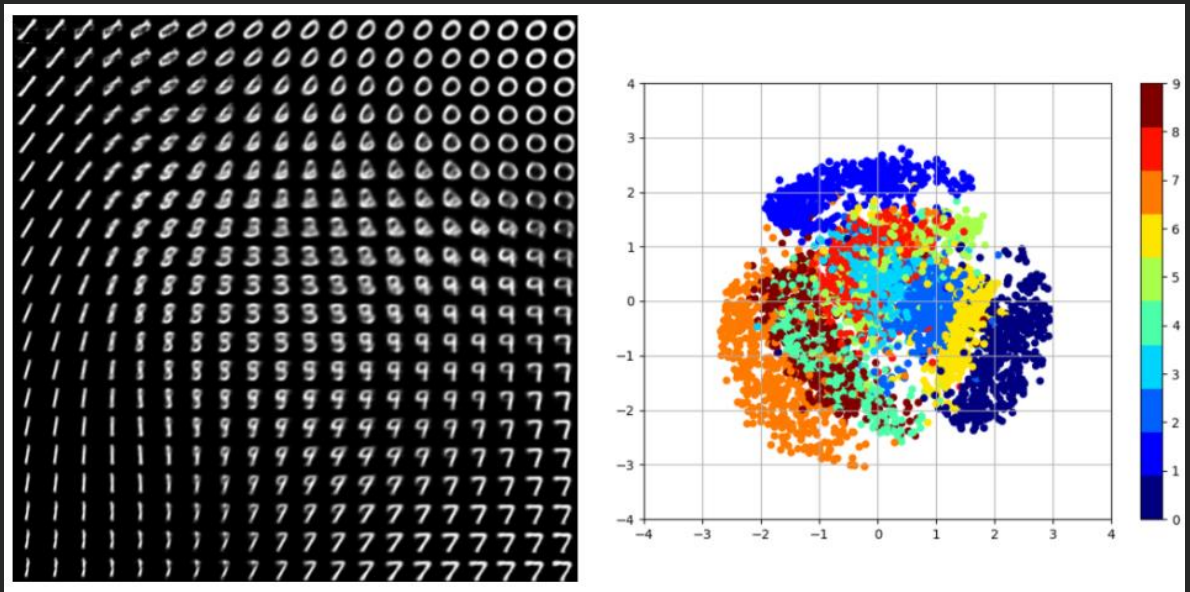


https://github.com/ChengBinJin/VAE-Tensorflow

Вариационные Авто-Энкодеры

.

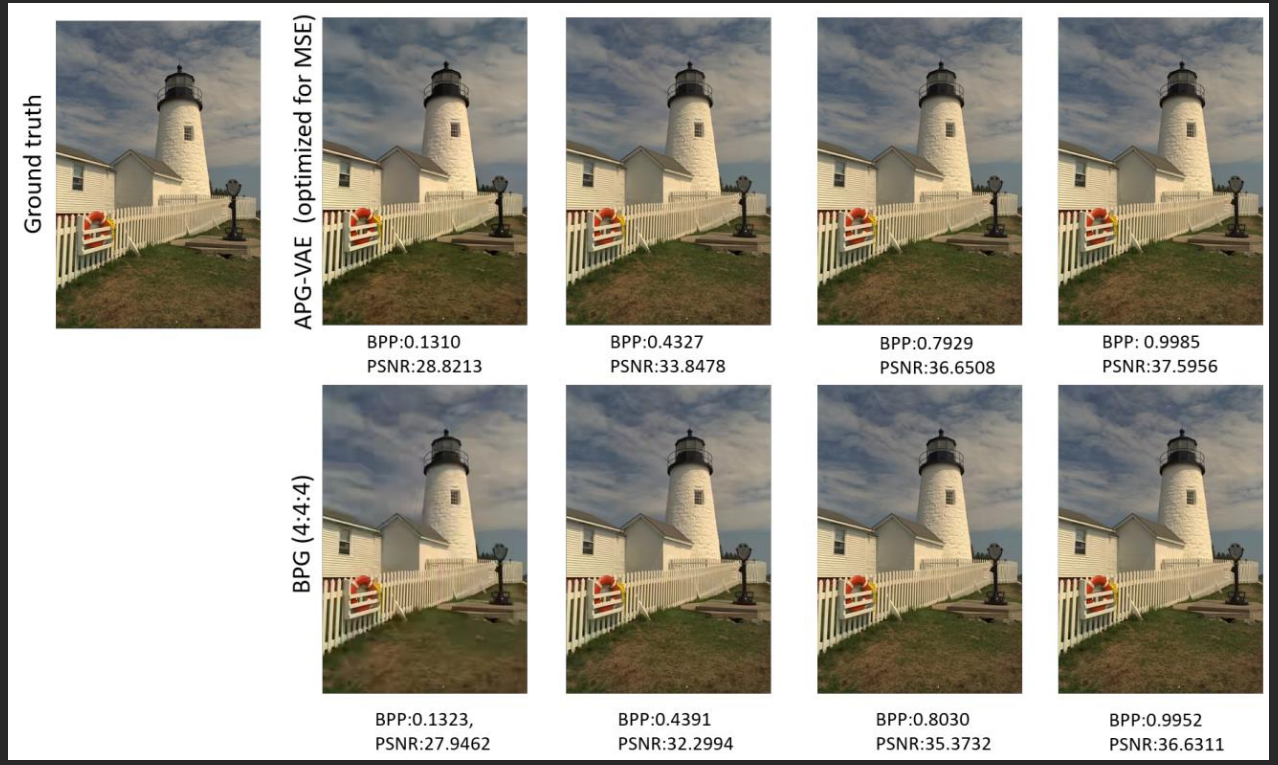




https://github.com/ChengBinJin/VAE-Tensorflow

Сжатие Изображений

.



Cui, Ze, Jing Wang, Bo Bai, Tiansheng Guo, and Yihui Feng. "G-VAE: A continuously variable rate deep image compression framework." *arXiv preprint arXiv:2003.02012* (2020).

https://arxiv.org/pdf/2003.02012.pdf

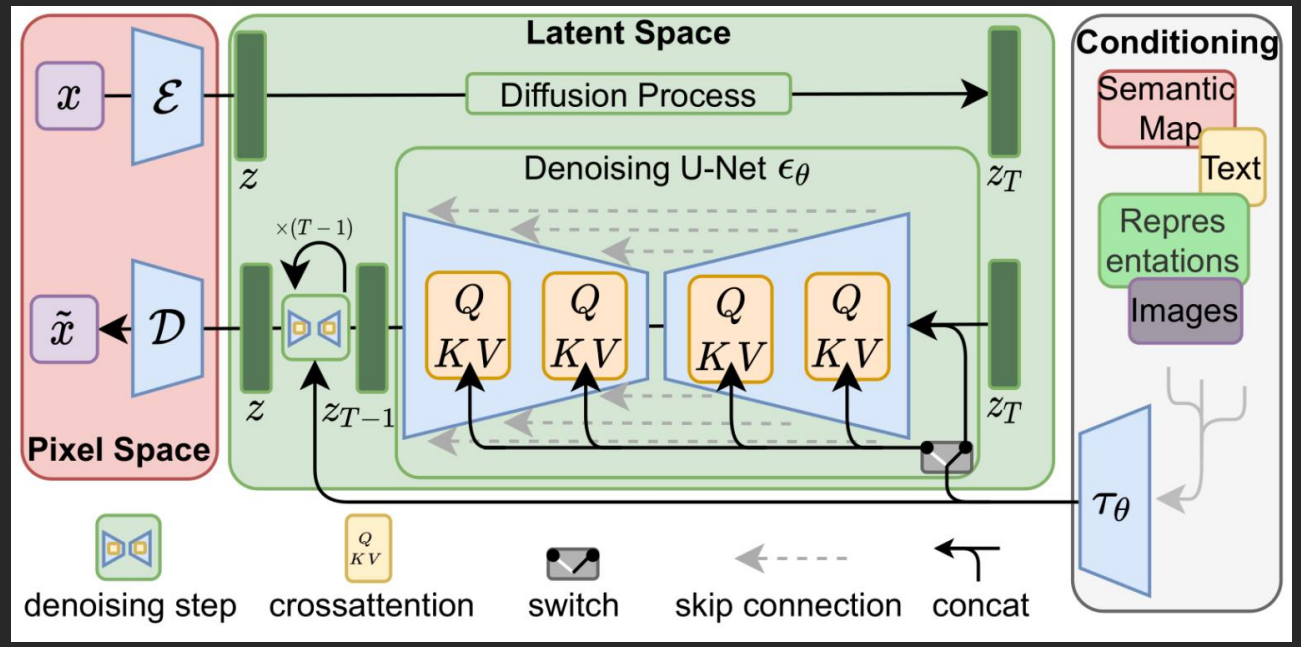
Содержание

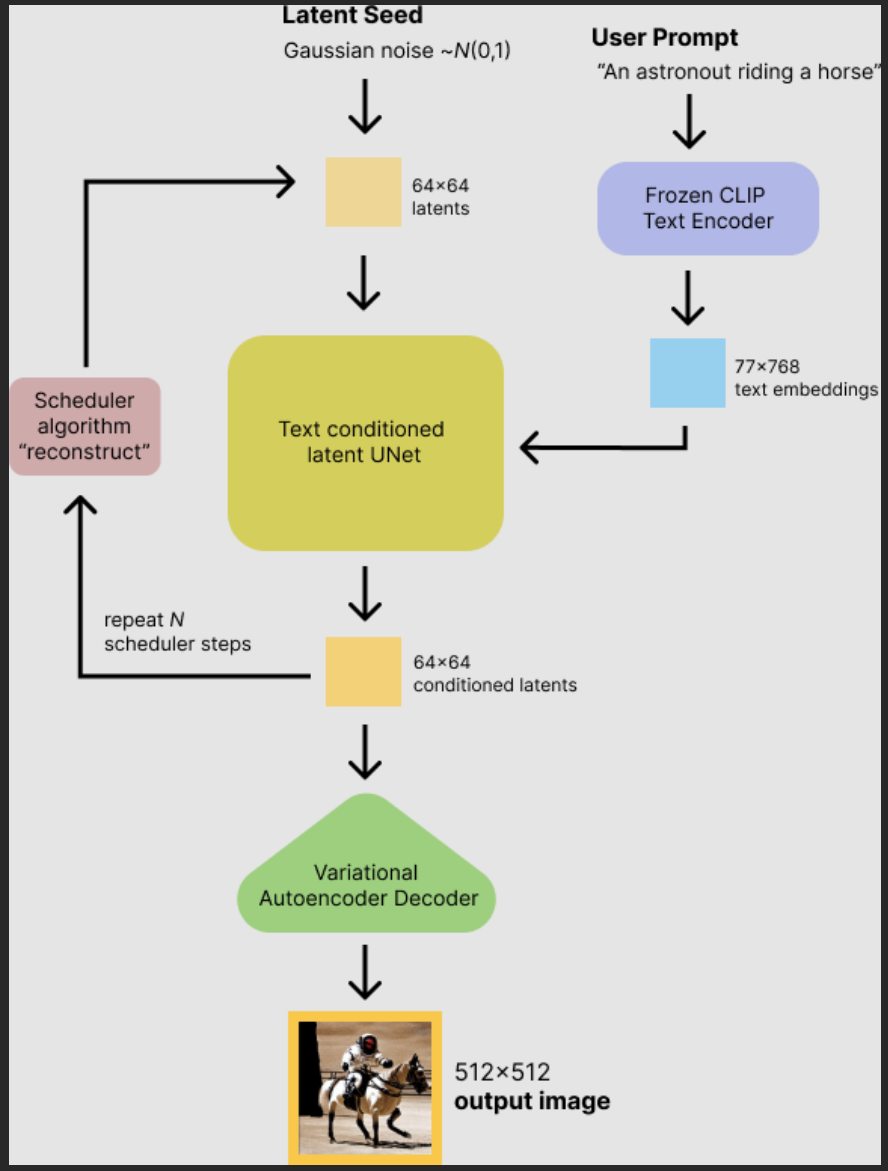
Генеративные Модели
Диффузия

Перенос Стиля

Про диффузии

1. An autoencoder (VAE)
2. A U-Net
3. A text-encoder, e.g. CLIP's Text Encoder



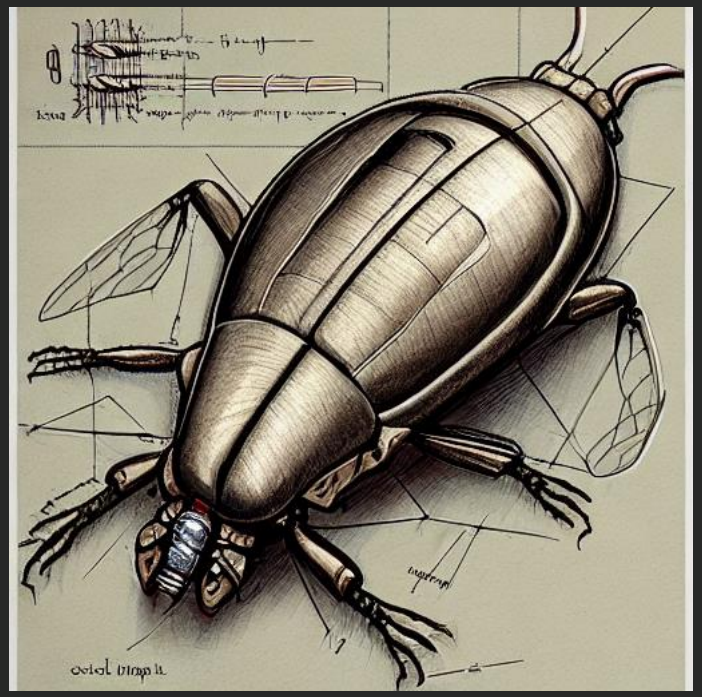


https://huggingface.co/blog/stable_diffusion

On the Diffusion Networks (GIF)

Some examples

"technical drawing of a bug, anatomy, mechanical features, steam punk, cyberpunk, 80s, computer chip, words, notes, text, roadmap, incredible detail, full page, washed out colors, davinci,"



'car on a curvey moutain road at night while raining, highly detailed, vray, unreal engine, 8k, intricate, bloom, wall paper'

Some examples

'A berth on the new year and a moored ship and in the distance the ship goes into the sunset, oil painting style'

'Many students staring at their smartphones and laughing, aquarelle'

Произведение искусства

бЕздУшнОЕ твОрЕнИЕ кОмпУктЕрА

Резюме

Генеративные Модели

Генеративно-Состезательные Сети

Генератор Дискриминатор

(*Вариационные*) *Авто-Энкодеры*

Энкодеры, Декодеры, Скрытое Пространство

Еще есть Диффузии

Перенос Стиля

Содержание, Стиль

Вопросы, пожелания, предложения

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.11
С текстами начинаем работу

Докладчик
Долганов Антон

Ранее

Генеративные Модели

Генеративно-Состезательные Сети

Генератор Дискриминатор

(*Вариационные*) *Авто-Энкодеры*

Энкодеры, Декодеры, Скрытое Пространство

Еще есть Диффузии

Перенос Стиля

Содержание, Стиль

DeepFakes

Содержание

Цели и задачи обработки естественного языка

Regular Expressions

Содержание

Цели и задачи обработки естественного языка

Regular Expressions

Обработка естественного языка
Natural Language Processing

- Символический анализ (1950-1990-ые)
- Статистический анализ (1990-2010-ые)
- Машинное обучение (2000-2010-ые)
- Анализ нейронных сетей (сейчас)

Обработка естественного языка: задачи

- Токенизация
- Определение границ предложения
- Мелкий синтаксический анализ
  - Тегирование части речи
  - Разбиение фраз на части речи
- Синтаксический анализ
  - Семантика
  - Семантическая маркировка ролей
  - Анализ семантических зависимостей
- Анализ настроений / анализ мнений
- Устранение неоднозначности / индукция смысла слова
- Определение имен собственных / классификация
- Временное выражение
- Распознавание/нормализация
- Поиск связей в предложении
- Извлечение информации
- Извлечение терминологии
- Обобщение
- Сходство
  - Атрибуциальное сходство (сходство слов)
  - Относительное сходство
  - Сходство фраз
  - Сходство предложений
  - Определение перефразирования
- Генерация естественного языка
- Распознавание речи
- Синтез речи
- Наполнение онтологий
- Ответы на вопросы
- Машинный перевод
- Поисковая системы
- Согласованность текста
- Обнаружение фейковых новостей

Обработка естественного языка: задачи

- **Классификация текстов**
  - Спам/Не Спам; + / - рецензия, и т.д.
- **Машинный перевод**
  - Google translate xD
- **Правка текстов**
  - Helo mu friend → hello my friend
- **Диалоговые системы**
  - Alexa, play Despacito
- **Поисковики**
  - Ok, Google…
- **Резюмирование**
  - TL;DR

Содержание

Цели и задачи обработки естественного языка

Regular Expressions

Regular Expressions

Регулярное выражение (Regular Expressions, RegEx) — это способ определить шаблон для поиска или манипулирования строками.

Мы можем использовать регулярное выражение для сопоставления, поиска, замены и манипулирования внутри текстовых данных.

- Модуль Python RE
- Регулярные выражения и их синтаксис
- Метасимволы регулярных выражений, специальные последовательности и классы символов
- Методы и объекты регулярных выражений

Модуль Python RE

```
import re
```

Регулярные выражения и их синтаксис

. (DOT) любой символ, кроме новой строки

^ (Caret) шаблон только в начале строки

$ (Dollar)шаблон на конце строки

* (asterisk) 0 или более повторений регулярного выражения

? (Question mark)0 или 1 повторение регулярного выражения

+ (Plus) 1 или более повторений регулярного выражения

\ (backslash) экранировать специальные символы

Регулярные выражения и их синтаксис

[ ] (Square brackets)набор символов.
Соответствует любому одиночному символу в скобках

[^] любой одиночный символ не в скобках

( ) все внутри это шаблон в целом

{} - узорные скобки указать конкретное количество совпадений

Некоторые специальные классы

\d эквивалентно [0-9] любой цифра

\D эквивалентно [^0-9] любой не цифре

\s эквивалентно [\t\n\x0b\r\f] любому пробельному символу

\S эквивалентно [^ \t\n\x0b\r\f] любому непробельному символу

\w эквивалентно [a-zA-Z_0-9] любому буквенно-цифровому символу

\W эквивалентно [^a-zA-Z_0-9] любому небуквенно-цифровому символу

\b - найти пустую строку в начале или в конце слова

\B как-то наоборот

Методы и объекты регулярных выражений

`re.compile('pattern')`
сохранить шаблон как переменную для использования в будущем

`re.match(pattern, str)`
вернуть совпадение в начале строки

`re.findall(pattern, str)`
Сканирует шаблон регулярного выражения по всей строке и возвращает все совпадения

`re.split(pattern, str)`
разбивает строку на список совпадений в соответствии с заданным шаблоном рег
улярного выражения

`re.sub(pattern, replacement, str)`
Замените одно или несколько вхождений шаблона в строке на `replacement`

Предварительная обработка

Крутяк, я изучаю машинное обучение в #УрФУ **#**Наука **! https://lo.st/12**:)

- **Удаление хэштегов, гиперссылок и т.д.**
- Токенизация строки
- Нижний регистр
- Удаление стоп-слов и знаков препинания
- Стемминг

Предварительная обработка

Удаление хэштегов, гиперссылок и т.д.

Крутяк, я изучаю машинное обучение в #УрФУ **#**Наука **! https://lo.st/12**:)

```
import re

tweet = re.sub(r'https?:\/\/.*[\r\n]*', '', tweet)
tweet = re.sub(r'#', '', tweet)
```

Крутяк, я изучаю машинное обучение в УрФУ Наука **!** :)

Резюме

Цели и задачи обработки естественного языка

Regular Expressions

- Модуль Python RE
- Регулярные выражения и их синтаксис
- Метасимволы регулярных выражений, специальные последовательности и классы символов
- Методы и объекты регулярных выражений

Вопросы, пожелания, предложения

????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.04
Истории про то, как сделать из слов Векторы

Докладчик
Долганов Антон

Ранее

Генеративные Модели

Генеративно-Состезательные Сети

Генератор Дискриминатор

(*Вариационные*) *Авто-Энкодеры*

Энкодеры, Декодеры, Скрытое Пространство

Еще есть Диффузии

Перенос Стиля

Содержание, Стиль

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

.

Обработка естественного языка

Natural Language Processing

- Символический анализ (1950-1990-ые)
- Статистический анализ (1990-2010-ые)
- Машинное обучение (2000-2010-ые)
- Анализ нейронных сетей (сейчас)

Обработка естественного языка: задачи

- Токенизация
- Определение границ предложения
- Мелкий синтаксический анализ
  - Тегирование части речи
  - Разбиение фраз на части речи
- Синтаксический анализ
  - Семантика
  - Семантическая маркировка ролей
  - Анализ семантических зависимостей
- Анализ настроений / анализ мнений
- Устранение неоднозначности / индукция смысла слова
- Определение имен собственных / классификация
- Временное выражение
- Распознавание/нормализация
- Поиск связей в предложении
- Извлечение информации
- Извлечение терминологии
- Обобщение
- Сходство
  - Атрибуциальное сходство (сходство слов)
  - Относительное сходство
  - Сходство фраз
  - Сходство предложений
  - Определение перефразирования
- Генерация естественного языка
- Распознавание речи
- Синтез речи
- Наполнение онтологий
- Ответы на вопросы
- Машинный перевод
- Поисковая системы
- Согласованность текста
- Обнаружение фейковых новостей

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Обработка естественного языка: задачи

- **Классификация текстов**
  - Спам/Не Спам; + / - рецензия, и т.д.
- **Машинный перевод**
  - Google translate xD
- **Правка текстов**
  - Helo mu friend → hello my friend
- **Диалоговые системы**
  - Alexa, play Despacito
- **Поисковики**
  - Ok, Google…
- **Резюмирование**
  - TL;DR

Естественный Язык .

Проблемы «цифровизации» слов

1. Семантика

Прилагательное:
имеющий цвет растущей листвы;
заросший зеленью, растительностью;
перен. техн. жарг. комп. жарг. Экологичный;
перен. незрелый, неспелый
перен. молодой, неопытный, плохо знающий жизнь

Глагол:
увеличиваться в размерах
перен. воспитываться
перен. совершенствоваться

Зеленая трава растет быстро

Существительное:
растение, не образующее одревесневающих стеблей;
только ед. ч.: покров земли из таких растений;
жарг. любое наркотическое вещество, предназначенное для курения

Наречие:
нареч. к быстрый;
через небольшой промежуток времени

Естественный Язык .

Проблемы «цифровизации» слов

1. Семантика
2. Позиция слова (относительная и абсолютная)

- You have completed the task **well**. [adverb]
- You are doing **well** in machine learning. [adjective]
- **Well**, not everyone are participating equally. [interjection]
- The **well** is empty. [noun]
- Tears are starting to **well** in his eyes. [verb]

Естественный Язык

Проблемы «цифровизации» слов

1. Семантика
2. Позиция слова (относительная и абсолютная)
3. Грамматические связи (“внимание”)

- Собака не пошла по дороге потому что она мокрая

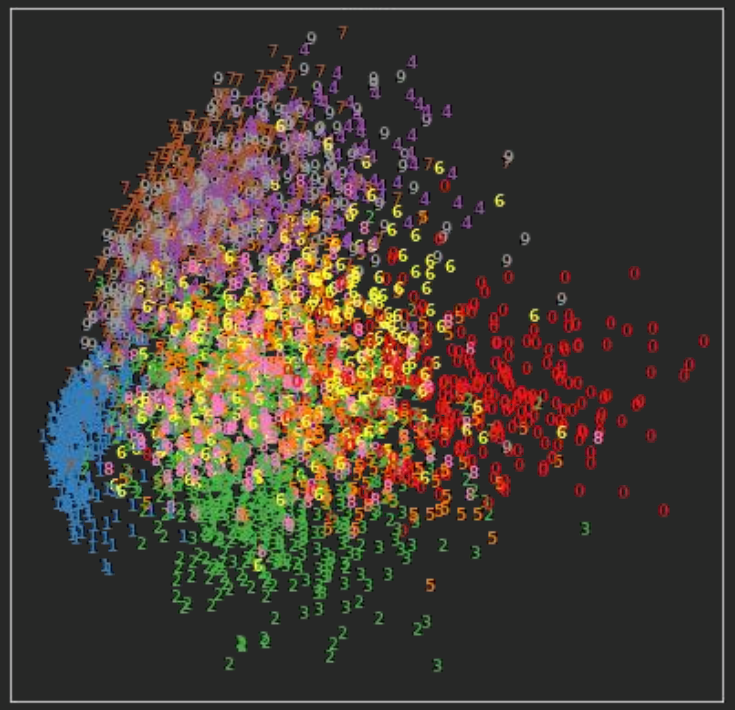
- Собака не пошла по дороге потому что она боится

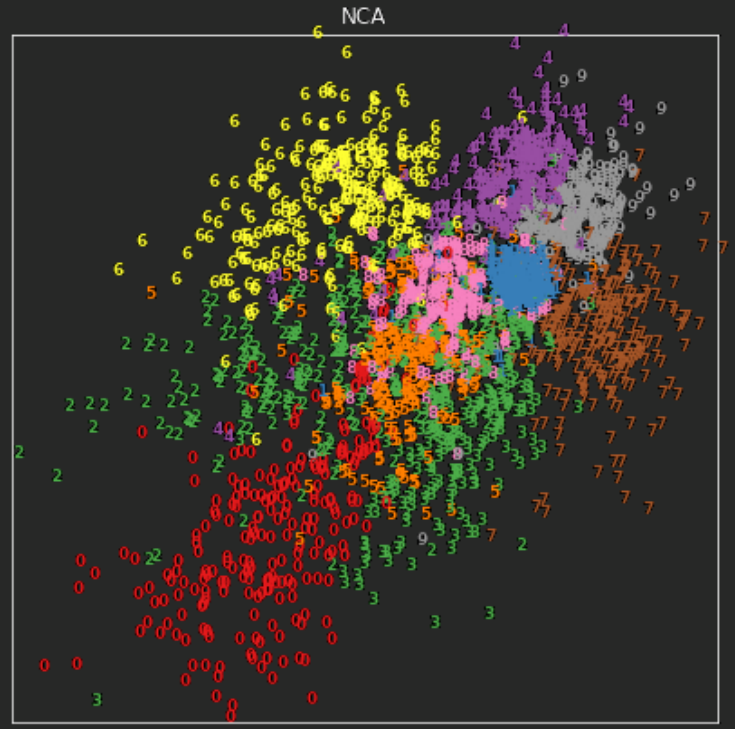
Содержание

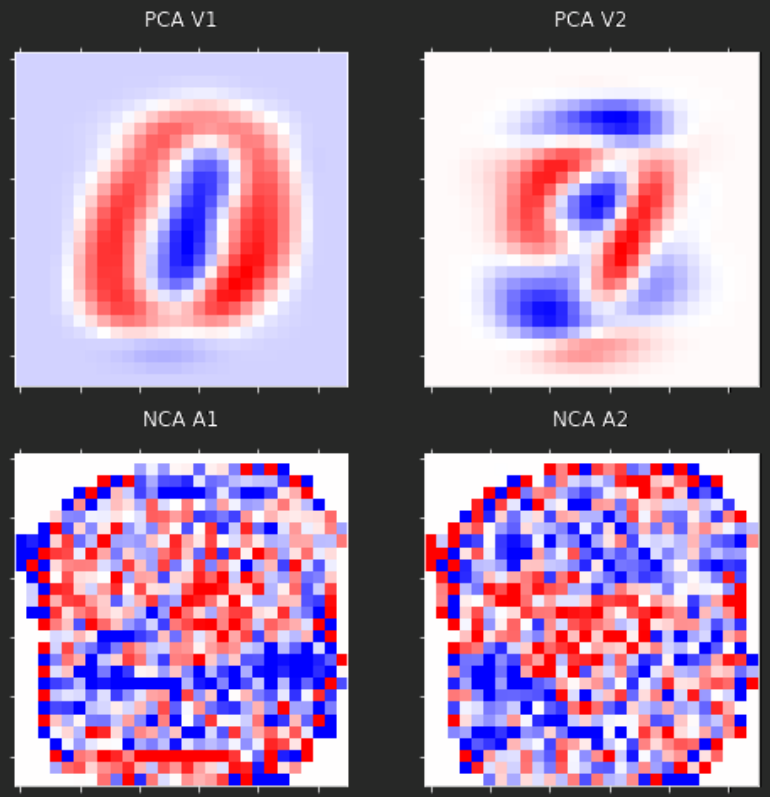
.

Про хитрые представления

Neighbourhood Components Analysis и Метод Главных Компонент PCA

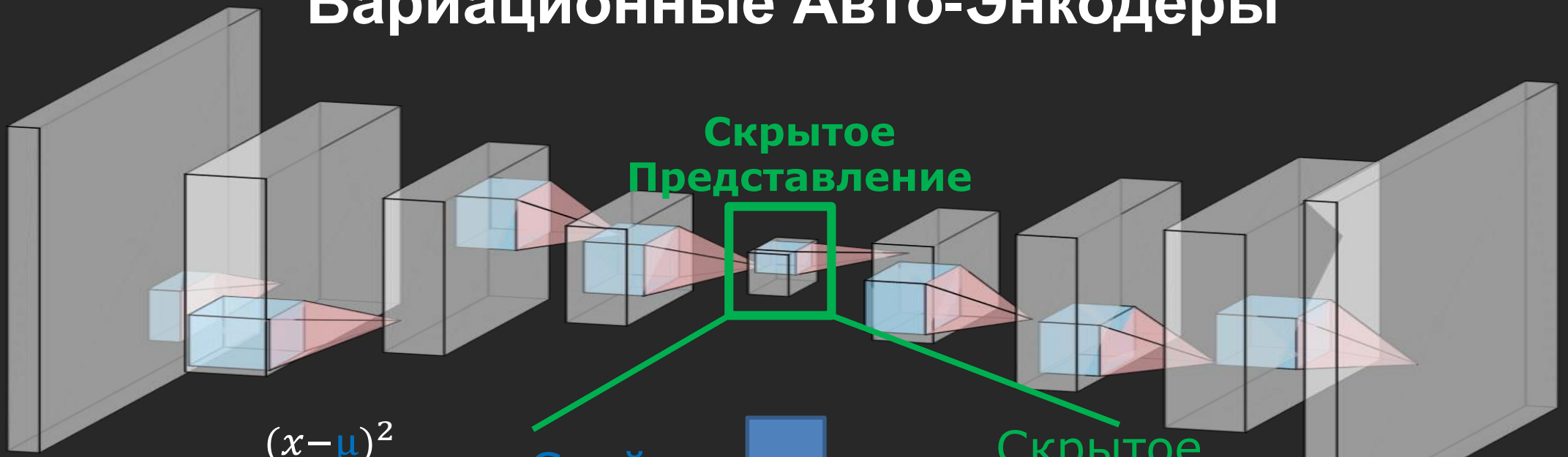






Variational Auto-Encoders

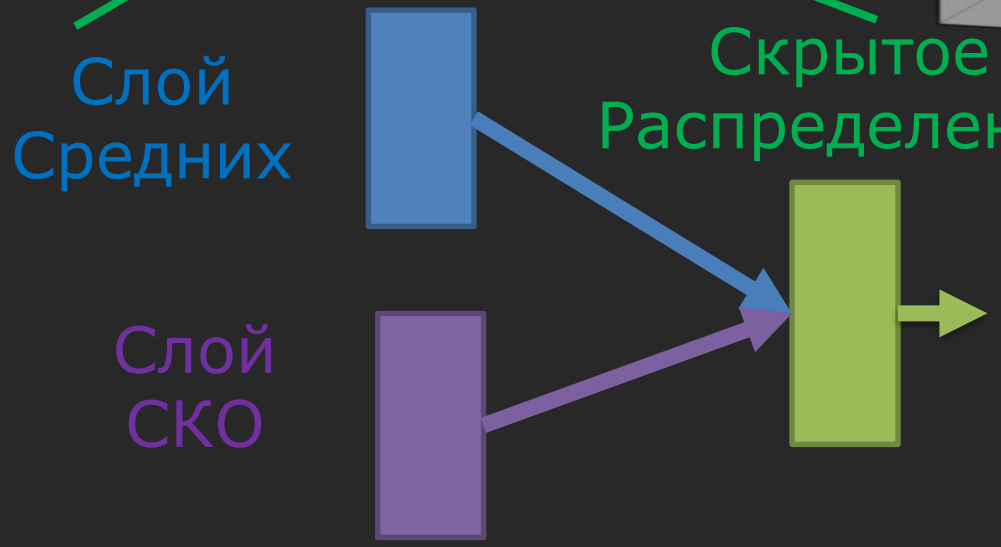
Вариационные Авто-Энкодеры



$$G(\mu, \sigma) = C * e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$Loss \sim MSE(\text{Вход} - \text{Выход}) + D_{K|L}\{G(\mu, \sigma), G(0,1)\}$$

Дивергенция Кульбака – Лейблера



Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Word Embeddings (Вложения?)

Текст

Happy, I study machine learning at UrFU Study ! :)

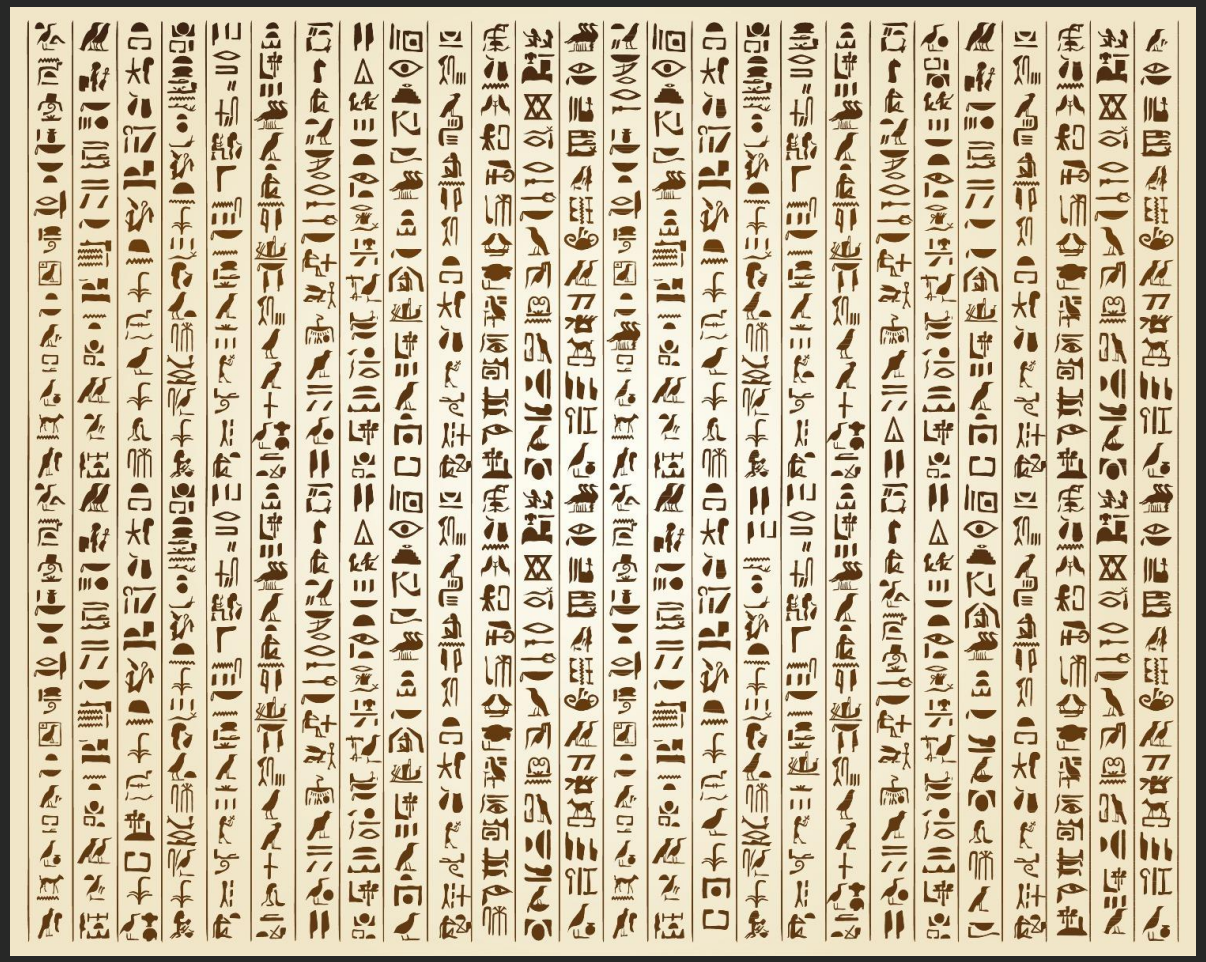
Токены

```
['happy', ',', 'i', 'study', 'machine',
'learning', 'at', 'urfu', 'study', '!', ':)']
```

Векторы?

$$\begin{bmatrix} 4 \\ 8 \\ 15 \\ \dots \end{bmatrix} \quad \begin{bmatrix} 16 \\ 23 \\ 42 \\ \dots \end{bmatrix} \quad \begin{bmatrix} \dots \\ \dots \\ \dots \\ \dots \end{bmatrix} \quad \begin{bmatrix} 69 \\ 420 \\ 1337 \\ \dots \end{bmatrix}$$

Про токенизацию



Египетские символы

https://usefulcharts.com/blogs/charts/evolution-of-the-english-alphabet

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Про токенизацию

по словам

```
['happy',
',', 'i',
'study',
'machine',
'learning',
'at',
'urfu',
'study',
'!', ':)']
```

по частям слова

```
['happ',’#y’,
',', 'i',
'stud',’#y’,
'machine',
'learn', ’#ing’,
'at', 'urfu',
'stud',’#y’,
'!', ':)']
```

по буквам

```
[‘h’,’a’,’p’,’p’,’y’,’
‘,’,
‘i’,
‘s’,’t’,’u’,’d’,’y’,
‘m’,’a’,’c’,’h’,’i’,’n’,’e’,
‘l’,’e’,’a’,’r’,’n’,’i’,’n’,’g’,
‘a’,’t’,
‘u’,’r’,’f’,’u’,’
‘s’,’t’,’u’,’d’,’y’,
 ‘!’,’ ‘:)’]
```

Стемминг / Лемматизация

Стемминг (выделение корней) подразумевает просто удаление несколько последних символов слова, что часто приводит к неправильному значению и правописанию.

Лемматизация учитывает контекст и преобразует слово в его осмысленную базовую форму, которая называется **леммой**. Сначала нужно определить часть речи слова.

Иногда одно и то же слово может иметь несколько разных лемм.

Целью как стемминга, так и лемматизации является уменьшение морфологической изменчивости.

Стемминг / Лемматизация

Word	Lemma	Stem
caring	(to) care	car
(someone) stripes	(to) strip	strip
(multiple) stripes	stripe	strip
walking	(to) walk	walk

Стемминг

- **уч**ить
- об**уч**ение
- **уч**итель
- из**уч**ать

⇩

уч

- **крут**як
- **крут**ой
- **крут**о

⇩

крут

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
tweets_stem = []

for word in tweets_clean:
    stem_word = stemmer.stem(word)
    tweets_stem.append(stem_word)
```

```
['крутяк', 'изучаю', 'машинное', 'обучение', 'урфу', 'наука', ':)']

 ['крут', 'уч', 'машин', 'уч', 'урфу', 'наук', ':)']
```

Лемматизация

```
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.tag.pos_tag(word)
```

```
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()
wnl.lemmatize(word, tag)
```

tokenized	pos_tags	wordnet_pos	lemmatized
[would, responded, going]	[(would, MD), (responded, VBD), (going, VBG)]	[(would, n), (responded, v), (going, v)]	[would, respond, go]
[sooo, sad, miss, san, diego]	[(sooo, JJ), (sad, JJ), (miss, NN), (san, NN),...	[(sooo, a), (sad, a), (miss, n), (san, n), (di...	[sooo, sad, miss, san, diego]
[boss, bullying]	[(boss, IN), (bullying, VBG)]	[(boss, n), (bullying, v)]	[bos, bully]
[interview, leave, alone]	[(interview, NN), (leave, VBP), (alone, RB)]	[(interview, n), (leave, v), (alone, r)]	[interview, leave, alone]
[sons, could, put, releases, already, bought]	[(sons, NNS), (could, MD), (put, VB), (release...	[(sons, n), (could, n), (put, v), (releases, n...	[son, could, put, release, already, buy]

Содержание

.

Вложения (Embedding)

Статистический подход

Контекстный подход

Векторные модели

Словарь

Структурированная коллекция всех известных слов

0	I
1	am
2	happy
3	study
4	machine
5	learning
6	sad
7	hate
8	deep
9	…

I am happy because I study machine learning

[1, 1, 1, 1, 1, 1, 0, 0, 0, …]

I am sad because I hate deep learning

[1, 1, 0, 1, 0, 1, 1, 1, 1, …]

…

Анализ тональности
Простой вектор признаков

I am happy I study machine learning
[1, 13, 9]

I am happy
[1, 7, 4]

I am sad I hate deep learning
[1, 9, 13]

I hate learning
[1, 4, 7]

Word	Pos Count	Neg Count
I	3	3
am	2	1
happy	2	0
study	1	0
machine	1	0
learning	1	2
sad	0	1
hate	0	2
deep	0	1
…	…	…

Частоты слов

Python

```
result = {}
for y, tweet in zip(ys, tweets):
    for word in pre_process(tweet):
        pair = (word,y)

        if pair in result:
            result[pair] += 1
        else:
            result[pair] = 1
```

Ну или по серьезному Python

```
from sklearn.feature_extraction.text import CountVectorizer

count_vectorizer = CountVectorizer(max_features=N, ngram_range=(1,n))


  count_vectorizer.fit(corpus)

  X_count = count_vectorizer.transform(corpus)

  dic_vocabulary_count = count_vectorizer.vocabulary_
```

Логистическая регрессия

.

$$\boldsymbol{X} \in \mathbb{R}^{n \times 3} = [1, PosCount, NegCount]$$

$$\boldsymbol{y} \in \mathbb{R}^{n \times 1}$$

$$\hat{y} = \sigma(X * B^T) = \frac{1}{1 + e^{-(X * B^T)}}$$

$$B \in \mathbb{R}^{1 \times 3}$$

$$\hat{y}\,(X) \geq 0.5 \rightarrow \text{y}=1$$

$$\hat{y}\,(X) < 0.5 \rightarrow \text{y}=0$$

Частоты слов Term Frequency TF

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

Word	Pos Count	Neg Count	PosFreq	NegFreq
I	3	3	0.3	0.3
am	2	1	0.2	0.1
happy	2	0	0.2	0
study	1	0	0.1	0
machine	1	0	0.1	0
learning	1	2	0.1	0.2
sad	0	1	0	0.1
hate	0	2	0	0.2
deep	0	1	0	0.1
Total	**10**	**10**		

Word	PosFreq	NegFreq
I	0.21053	0.21053
am	0.15789	0.10526
happy	0.15789	0.05263
study	0.10526	0.05263
machine	0.10526	0.05263
learning	0.10526	0.15789
sad	0.05263	0.10526
hate	0.05263	0.15789
deep	0.05263	0.10526

$$P(\text{word}\,|PosFreq) = \frac{word + 1}{N_{PosFreq} + V}$$

$$P(\text{word}\,|\text{NegFreq}) = \frac{word + 1}{N_{\text{NegFreq}} + V}$$

Лапласовское сглаживание

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Обратная частота документа Inverse Document Frequency IDF

$$idf(t, D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|}$$

$|D|$ число документов в коллекции

I am happy I study machine learning

I am happy

I am sad I hate deep learning

I hate learning

Word	|D|	|{d_i D| t d_i }|	IDF
I	4	4	0.00
am	4	3	0.12
happy	4	2	0.30
study	4	2	0.30
machine	4	1	0.60
learning	4	3	0.12
sad	4	1	0.60
hate	4	2	0.30
deep	4	1	0.60

Term Frequency TF
+ (*)
Inverse Document Frequency IDF
= win

$$tf \cdot idf = tf(t, d) * idf(t, D)$$

Word
I
am
happy
study
machine
learning
sad
hate
deep

IDF	Pos TF	Neg TF	Pos TF IDF	Neg TF IDF
0.00	0.21	0.21	0.00	0.00
0.12	0.16	0.11	0.02	0.01
0.30	0.16	0.05	0.05	0.02
0.30	0.11	0.05	0.03	0.02
0.60	0.11	0.05	0.06	0.03
0.12	0.11	0.16	0.01	0.02
0.60	0.05	0.11	0.03	0.06
0.30	0.05	0.16	0.02	0.05
0.60	0.05	0.11	0.03	0.06

Ну или по серьезному Python

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer =  TfidfVectorizer(max_features=N, ngram_range=(1,n))


  tfidf_vectorizer.fit(corpus)

  X_tfidf = tfidf_vectorizer.transform(corpus)

  dic_vocabulary_tfidf = tfidf_vectorizer.vocabulary_
```

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Теорема Байеса

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} = \frac{P(E|H)P(H)}{P(E|H)P(H) + P(E|\neg H)P(\neg H)}$$

E – Evidence / Событие

H – Hypothesis / Гипотеза

$P(H|E)$ - апостериорная вероятность **Гипотезы** при наличии **События**

$P(E|H)$ - правдоподобие, вероятность наблюдения **События** данной **гипотезы**

$P(H)$ - априорная вероятность **гипотезы** до того, как наблюдаются **События**

$P(E)$ - сумма произведений всех априорных вероятностей взаимоисключающих **Гипотез** и соответствующих правдоподобий

Байесовский подход

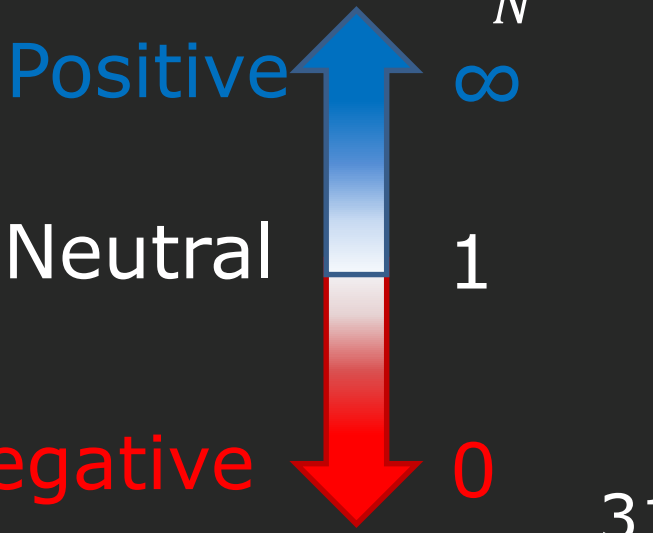
Word	PosFreq	NegFreq
I	0.21053	0.21053
am	0.15789	0.10526
happy	0.15789	0.05263
study	0.10526	0.05263
machine	0.10526	0.05263
learning	0.10526	0.15789
sad	0.05263	0.10526
hate	0.05263	0.15789
deep	0.05263	0.10526

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

$$P(\text{word}\,|PosFreq) = \frac{word + 1}{N_{PosFreq} + V} \qquad P(PosFreq) = \frac{N_{PosFreq}}{N}$$

$$P(\text{word}\,|\text{NegFreq}) = \frac{word + 1}{N_{\text{NegFreq}} + V} \qquad P(\text{NegFreq}) = \frac{N_{\text{NegFreq}}}{N}$$

$$\frac{P(PosFreq)}{P(\text{NegFreq})} \prod_{i=1}^{m} \frac{P(\text{word}_i|PosFreq)}{P(\text{word}_i|\text{NegFreq})}$$



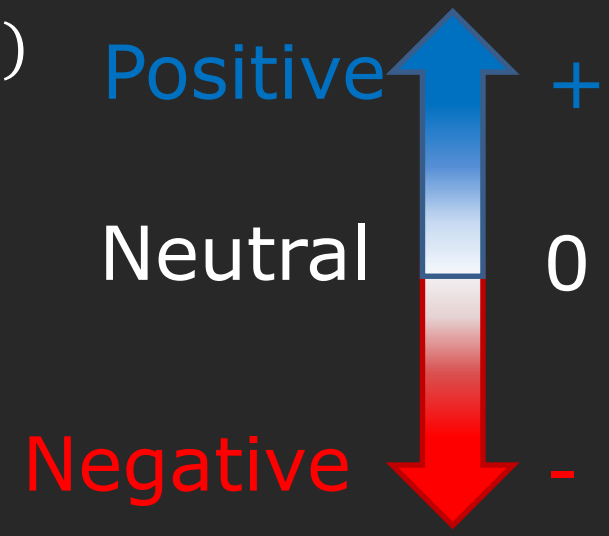
Байесовский подход

Логарифмируем

$$\frac{P(PosFreq)}{P(NegFreq)}\prod_{i=1}^{m}\frac{P(\text{word}_i|PosFreq)}{P(\text{word}_i|NegFreq)} \qquad \log\frac{P(PosFreq)}{P(NegFreq)}+\sum_{i=1}^{m}\log\frac{P(\text{word}_i|PosFreq)}{P(\text{word}_i|NegFreq)}$$

Log Likelihood $$\sum_{i=1}^{m}\log\frac{P(\text{word}_i|PosFreq)}{P(\text{word}_i|NegFreq)}=\sum_{i=1}^{m}\lambda(\text{word}_i)$$

Log Prior $$\log\frac{P(PosFreq)}{P(NegFreq)}$$



Байесовский подход

Применение

- Идентификация автора $\frac{P(Shakespere\ |\text{Книга})}{P(Rowling|\text{Книга})}$
- Фильтрация спама $\frac{P(Mail|Not\ Spam)}{P(Mail|\text{Spam})}$
- Поиск информации $P(\text{документ}|\text{запрос}) \propto \prod_{i=1}^{m} P(\text{запрос}_i|\text{документ})$
- Неоднозначность слов $\frac{P(\text{ЗначениеСлова1}|\text{Контекст})}{P(\text{ЗначениеСлова2}|\text{Контекст})}$

Замок

Байесовский подход

Ограничения

- Порядок слов

I am **not** happy because I do study Machine Learning

I am happy because I do **not** study Machine Learning

- Состязательные (Adversarial) атаки
  - сарказм
  - ирония
  - эвфемизмы

Содержание

.

Вложения (Embedding)

Статистический подход

Контекстный подход

Векторные модели

Контекст

You shall know a word by the company it keeps
(Firth, J. R. 1957:11)

Word by Word

Word by Document

Word2Vec

Word by Word

I like to study machine learning

Deep learning is group of machine learning methods

Корпус

k=2 соседи

learning

study	deep	machine	methods	like	I
1	1	2	1	0	0

learning ~ [1,1,2,1,0,0]

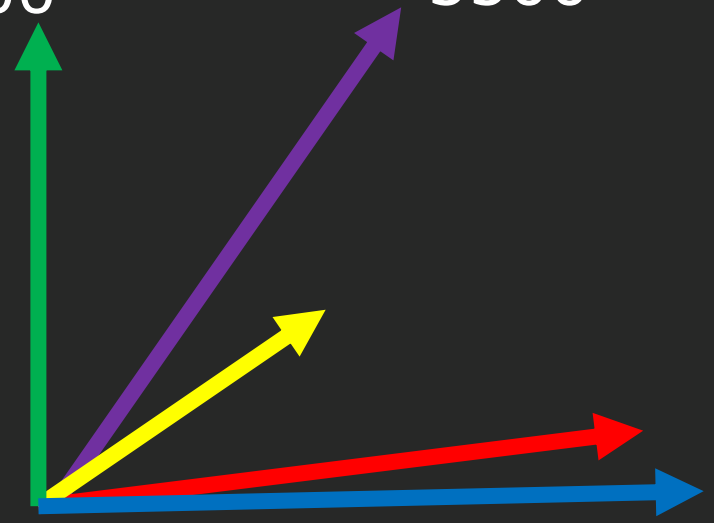
Word by Document

.

Корпус

	Economy	Sport	Machine Learning
data	5000	500	10000
goal	2000	6000	3500

Machine Learning ~ [10000,3500]



Cosine Distance

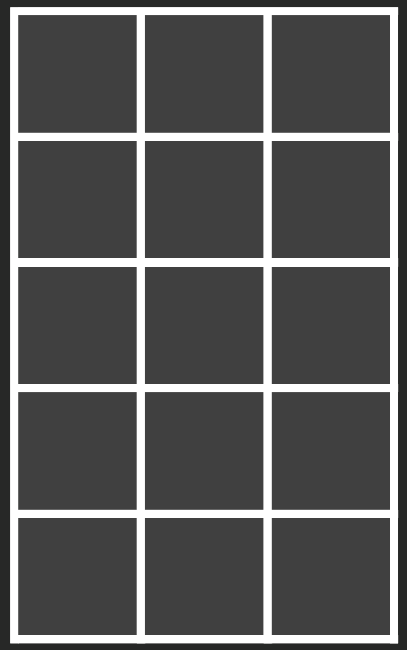
$$\cos\theta = \frac{A \cdot B}{|A||B|}$$

Word Embeddings

Singular Value Decomposition $A = USV^T$

документы

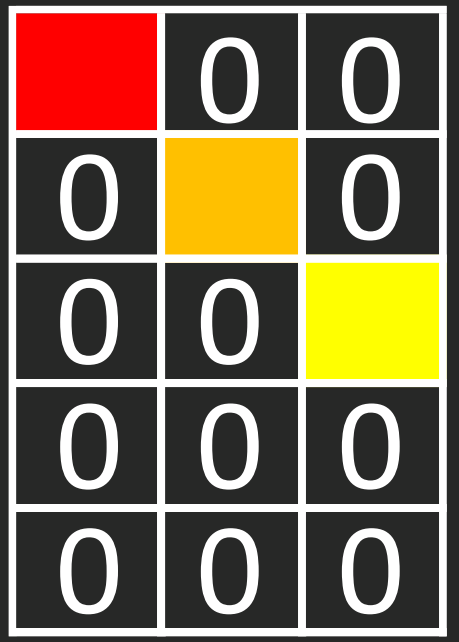
слова



$A \in \mathbb{R}^{n \times m}$

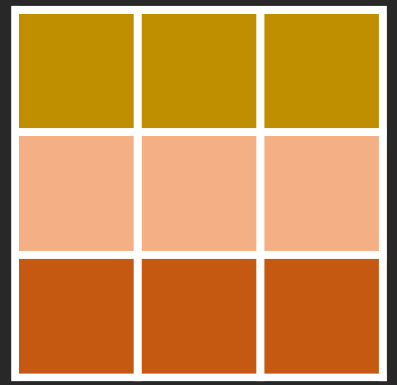
Скрытое пространство слов

$U \in \mathbb{R}^{n \times n}$



$S \in \mathbb{R}^{n \times m}$

Скрытое пространство документов



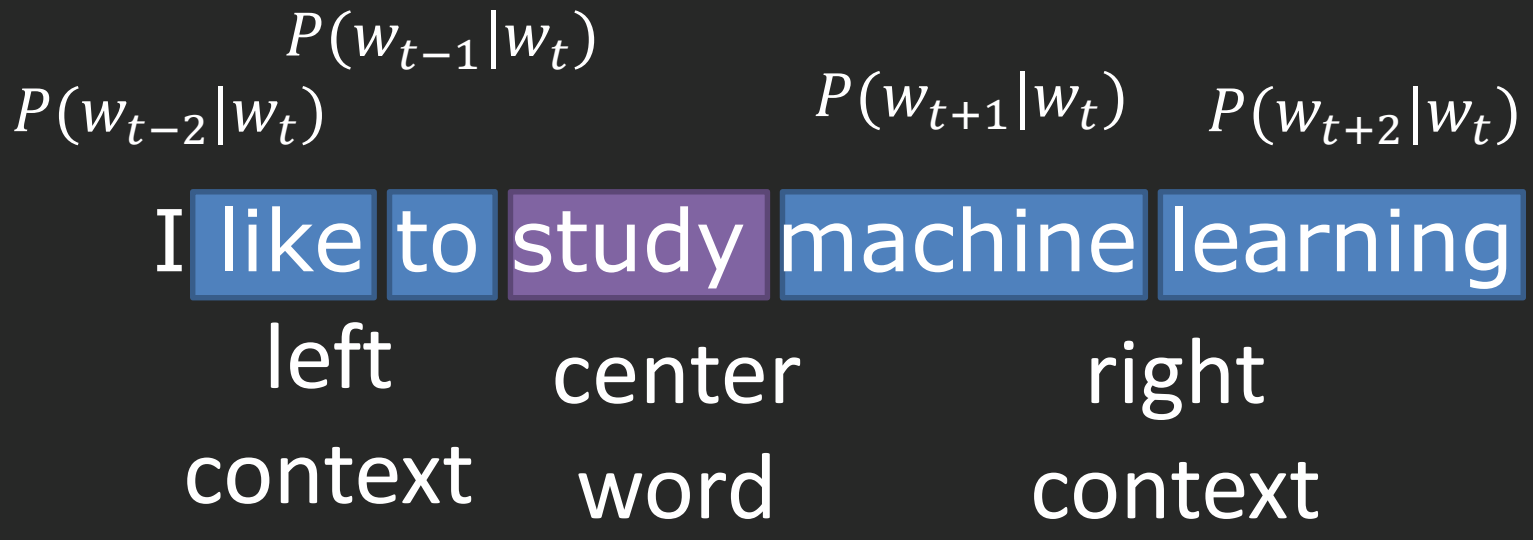
$V^T \in \mathbb{R}^{m \times m}$

Содержание

.

Word2Vec

$$P(w_{t+j}|w_t)$$



Word2Vec

$$Likelihood = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j}|w_t;\theta)$$

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j}|w_t;\theta)$$

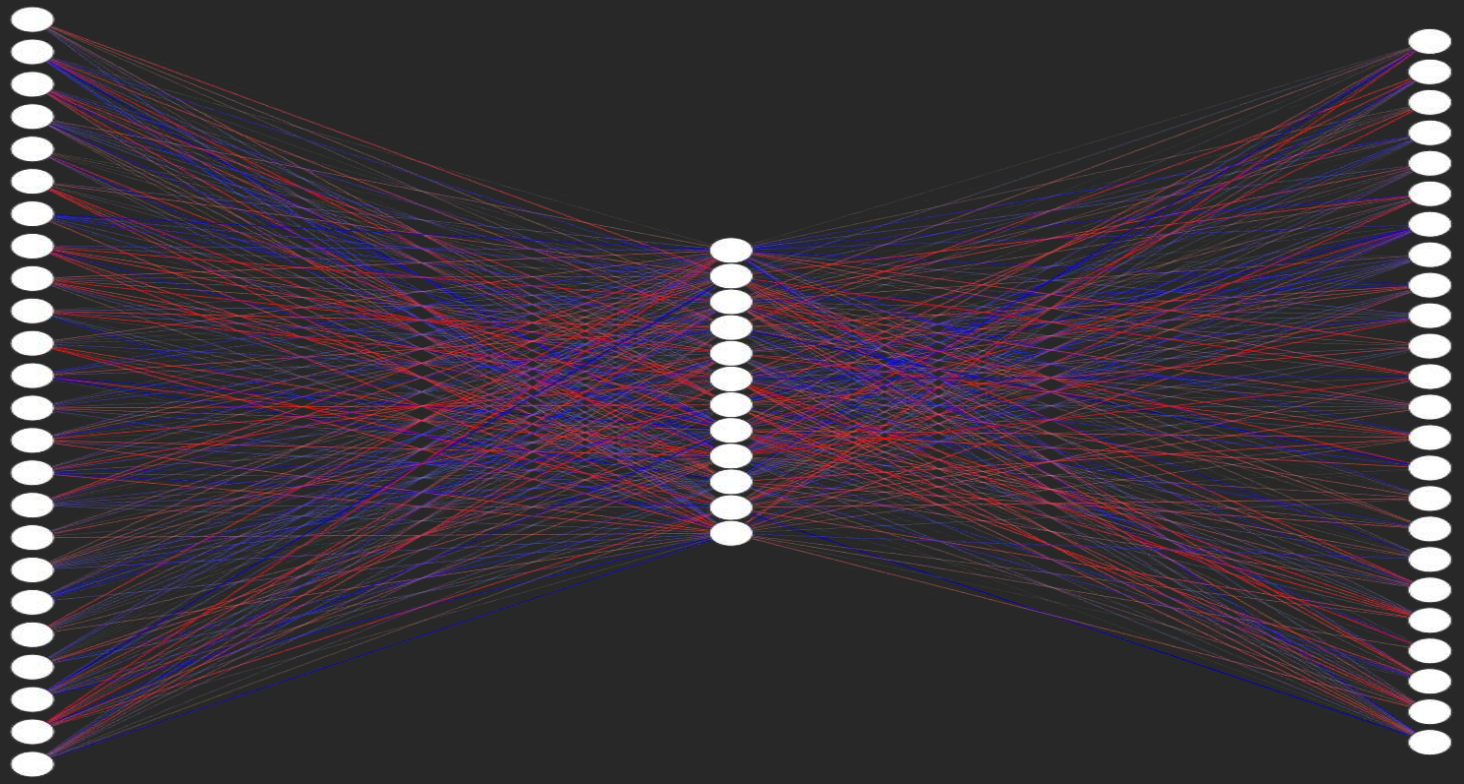
Пусть для каждого слова w есть два вектора

- v_w когда w центральное слово
- u_w когда w слово контекст

Тогда можно записать для центрального слова c и контекста o

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Word2Vec



Скрытый Слой

Вход (Размер ~ Словарь)

Выход (Размер ~ Словарь)

https://patents.google.com/patent/US9037464B1/en

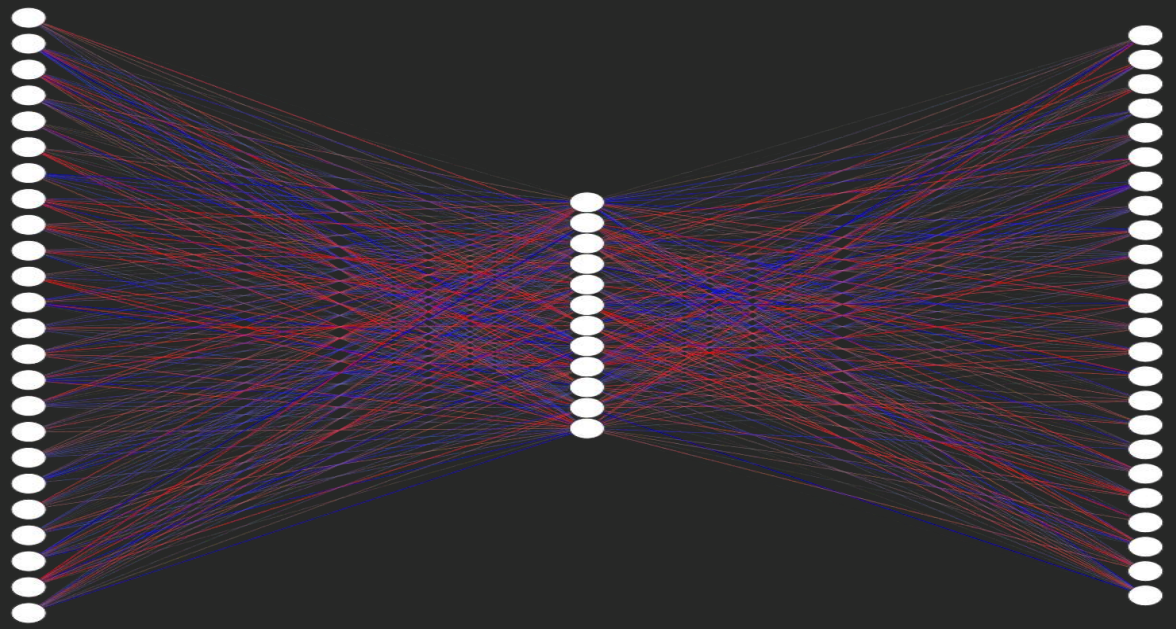
Word2Vec

I like to study machine learning

like to ____ machine learning

study

Контекст



Целевое Слово

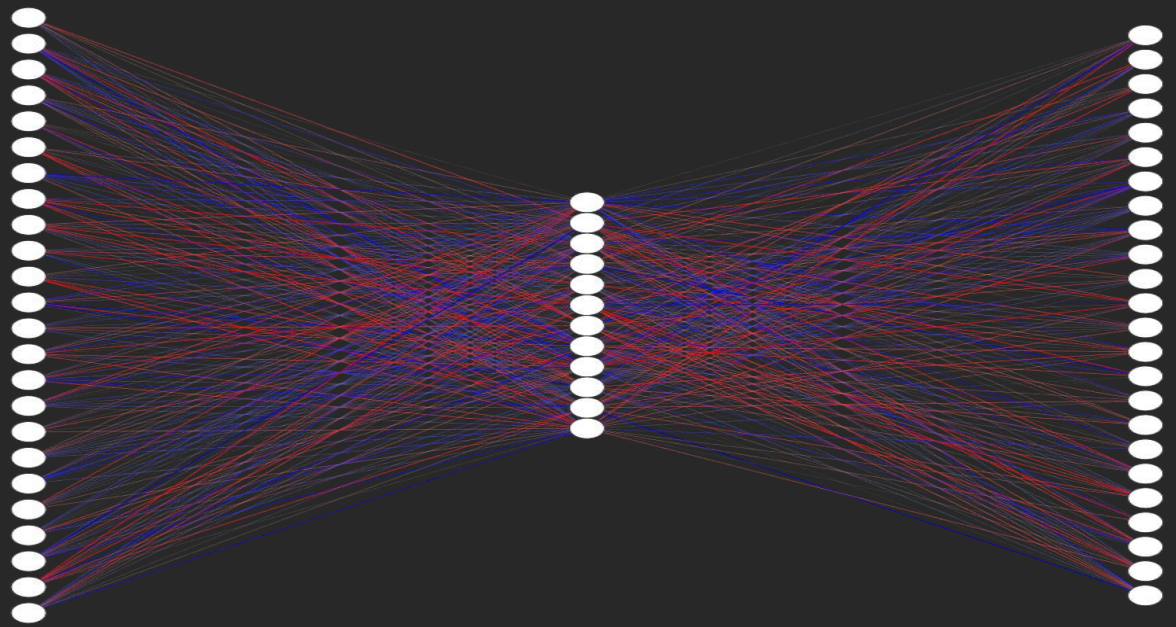
Continuous Bag of Words

Word2Vec

I like to study machine learning

study

like to ____ machine learning



Целевое Слово

Контекст

Continuous Skip Grams

Представление слов и one-hot encoding

0	I
1	am
2	happy
3	study
4	machine
5	learning
6	sad
7	hate
8	deep
9	…

I

[1, 0, 0, 0, 0, 0, 0, 0, 0, …]

am

[0, 1, 0, 0, 0, 0, 0, 0, 0, …]

Размер ~ Словарь

Word2Vec

Continuous Bag of Words

[0, 0, 0, 1, 0, 0, 0, 0, 0, …]

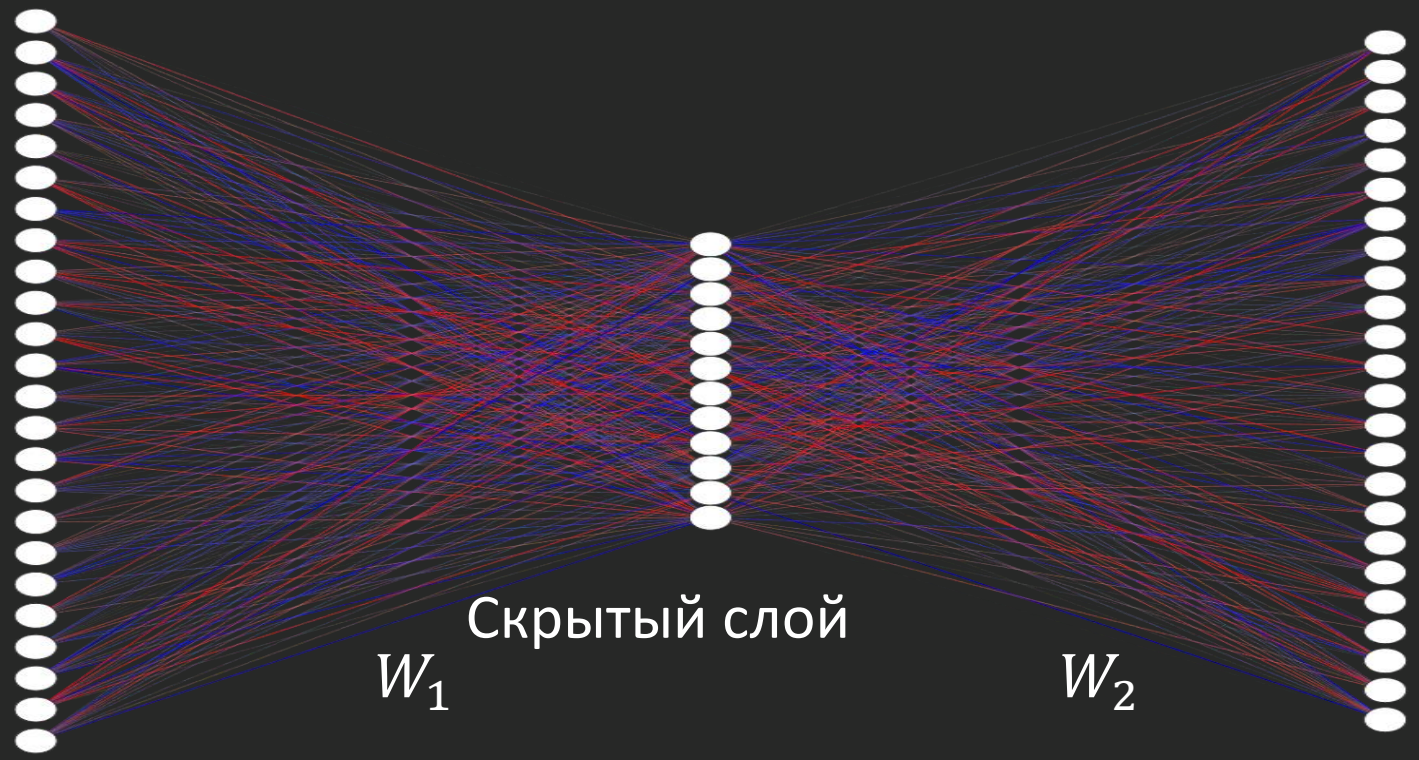
study

$$Target\ Word_t = Word_t$$

like to _____ machine learning

$$Context_t \sim \frac{1}{N}(Word_{t-\frac{N}{2}} + Word_{t-\frac{N}{2}+1} + \cdots + Word_{t+\frac{N}{2}-1} + Word_{t+\frac{N}{2}})$$

Word2Vec

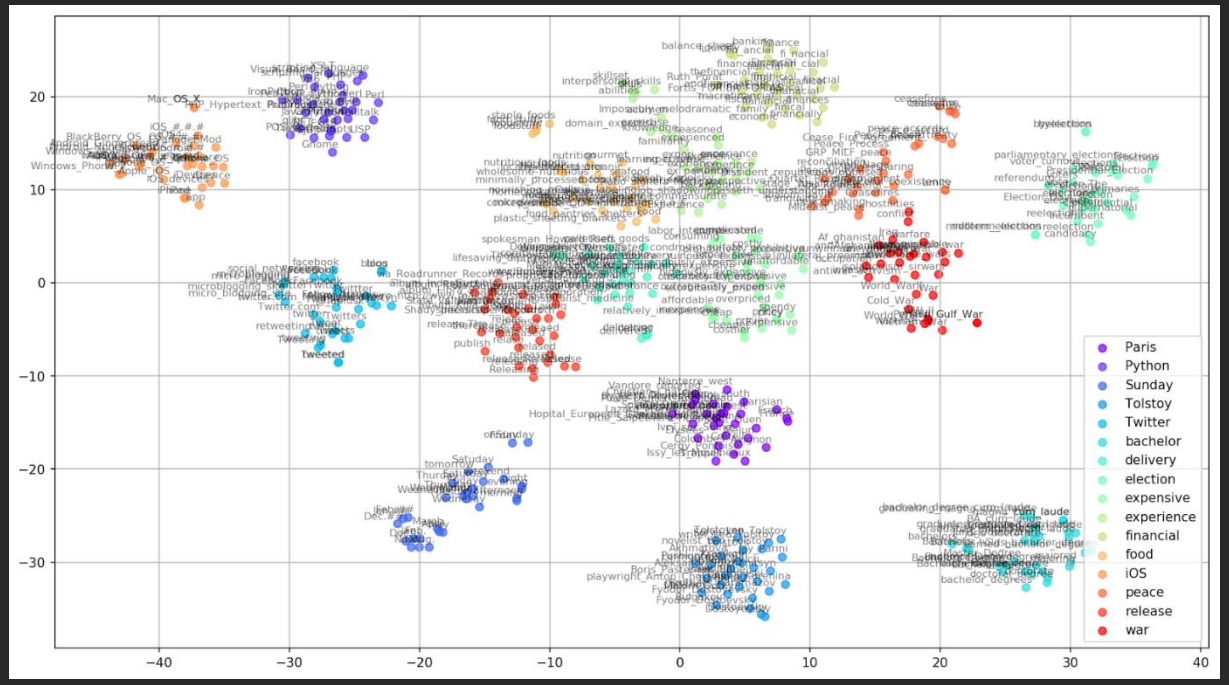


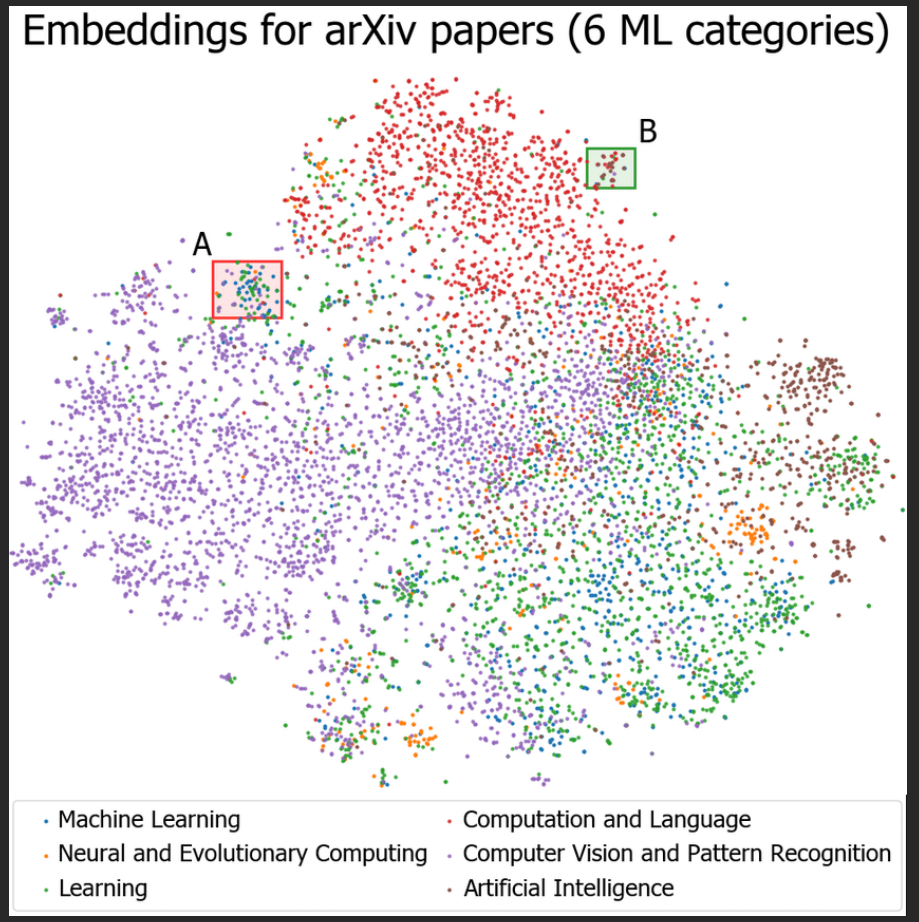
$$CCE = -\sum_{c=1}^{C} \frac{1}{Y_c} \sum_{i:y_i=c} y_i * \log(\hat{y}_i)$$

Скрытое вложение embedding (Матрица Весов)

W_1 W_2^T $W_1 + W_2^T$

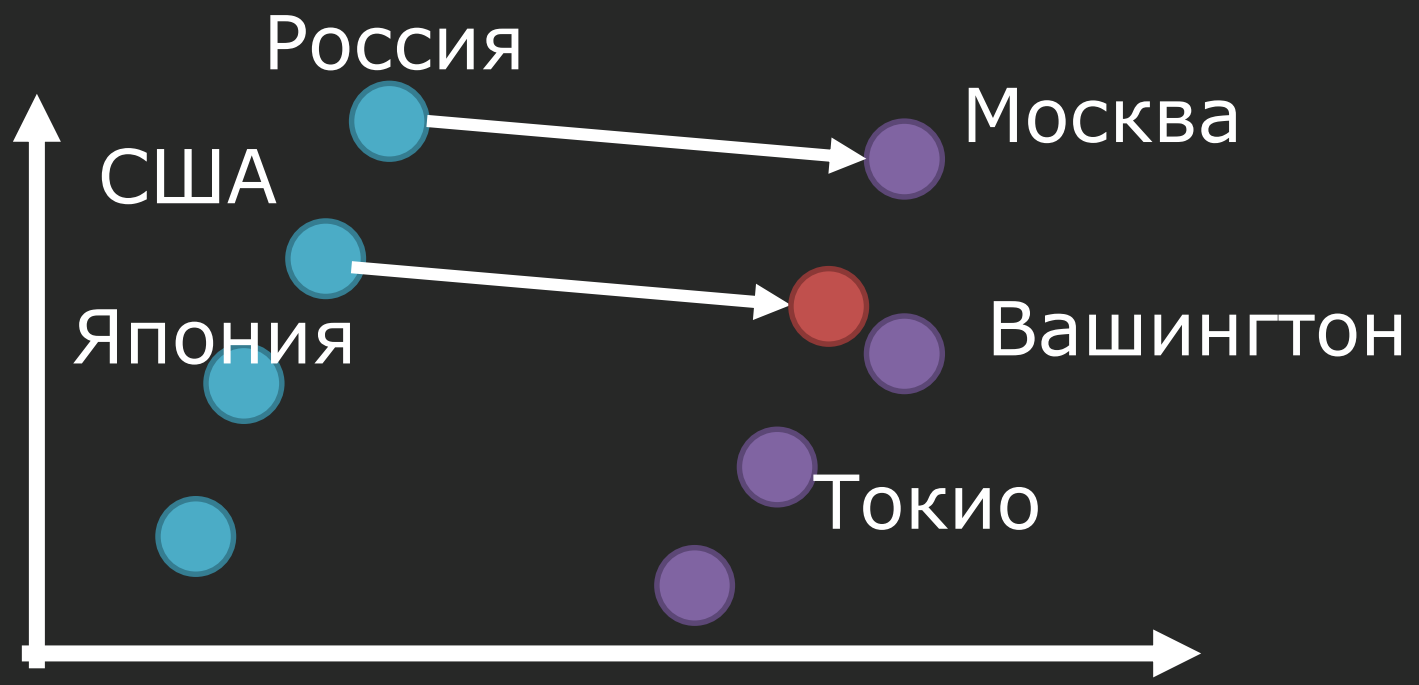
Word2Vec


Paris
Python
Sunday
Tolstoy
Twitter
bachelor
delivery
election
expensive
experience
financial
food
iOS
peace
release
war


Embeddings for arXiv papers (6 ML categories)
A
B
Machine Learning
Neural and Evolutionary Computing
Learning
Computation and Language
Computer Vision and Pattern Recognition
Artificial Intelligence

Содержание

Векторные модели



США – Вашингтон = Россия – Москва

США – Россия + Москва = Вашингтон

Королева – Женщина = Король – Мужчина

Королева = Король – Мужчина + Женщина

Векторные модели

```
import gensim
from gensim.models import word2vec
from gensim.models import KeyedVectors

!wget -P /root/input/ -c "https://s3.amazonaws.com/dl4j-
distribution/GoogleNews-vectors-negative300.bin.gz"

EMBEDDING_FILE = '/root/input/GoogleNews-vectors-negative300.bin.gz'

word_vectors = KeyedVectors.load_word2vec_format(EMBEDDING_FILE, binary=True)

word_vectors.most_similar(positive=['woman', 'king'], negative=['man'], topn=5)
```

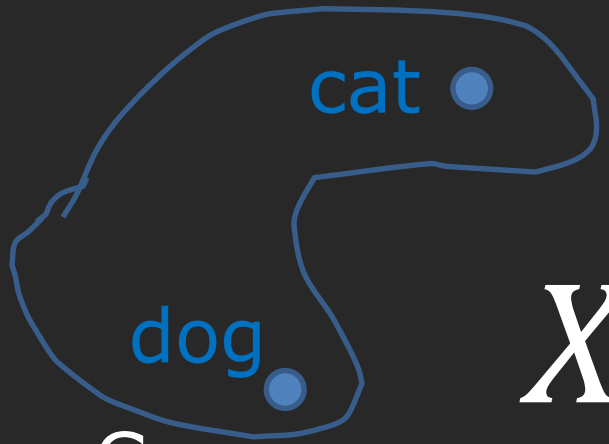
```
[('queen', 0.7118192911148071),
('monarch', 0.6189674139022827),
('princess', 0.5902431011199951),
('crown_prince', 0.5499460697174072),
('prince', 0.5377321243286133)]
```

Queen = King – Male + Female

Векторные модели

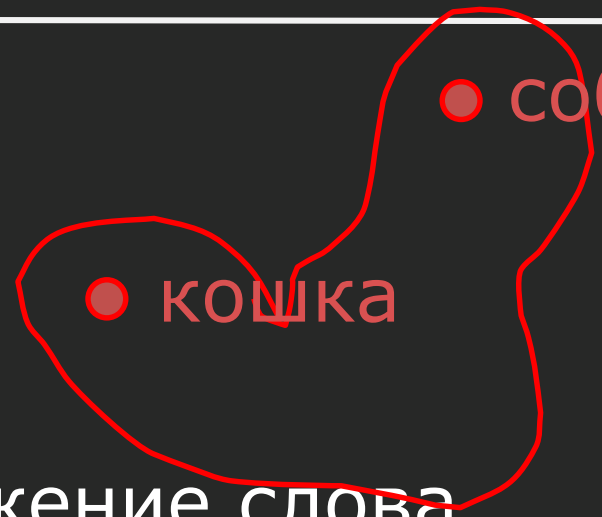
```
word_vectors.most_similar(positive=['Russia', 'White_House'], negative=['Kremlin'], topn=5)
    ('Oval_Office', 0.553366065025329 6),
    ('United_States', 0.5473676518997192),
    ('Obama', 0.5402511954307556),
    ('President_Barack_Obama', 0.5351332426071167),
    ('Bush', 0.534263730049133 3)

word_vectors.most_similar(positive=['Vodka', 'Japan'], negative=['Russia'], topn=5)
    ('Shochu', 0.5533198118209839),
    ('shochu', 0.5222346782684326),
    ('Momokawa', 0.5214118957519531),
    ('saké', 0.519989252090454 1),
    ('Sadaharu', 0.5143933892250061)

word_vectors.most_similar(positive=['balalaika', 'Hawaii'], negative=['Russia'], topn=5)

    ('ukulele', 0.572131872177124),
    ('hula_halau', 0.5431629419326782),
    ('Makaha_Sons', 0.5396252870559692),
    ('ukulele_virtuoso_Jake_Shimabukuro', 0.5336641073226929),
    ('kumu', 0.525770783424377 4)
```

Наивный машинный перевод



X

Скрытое вложение слова в английском языке



Y

Скрытое вложение слова в русском языке

R Матрица Перевода

$$R: \arg\min_R ||XR - Y||$$

Train Data → (Loss, Gradient Descent) →??? → Profit

Поиск документов

Модель «мешок слов»

Каждое предложение представляет собой сумму векторных вложений слов в нем

Поиск ближайшего к вашему предложение из Корпуса

Резюме

Вложения (Embedding)

- Токены – цифры
- Перед началом - Лемматизация, стемминг

Статистический подход

- Счетчики / Частоты слов
- TF-IDF
- Классические алгоритмы МО

Контекстный подход

- Word by Word, Word by Doc
- Word2Vec

Векторные модели

- Алгебра над векторными представлениями
- Наивный машинный перевод

Вопросы, пожелания, предложения

?????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.12
Теория вероятности на страже Обработки текстов

Докладчик
Долганов Антон

Ранее в Машинном Обучении

Цели и задачи обработки естественного языка

Regular Expressions

- Модуль Python RE
- Регулярные выражения и их синтаксис
- Метасимволы регулярных выражений, специальные последовательности и классы символов
- Методы и объекты регулярных выражений

Содержание

Предварительная обработка

Autocorrect

Autocomplete

Содержание

Предварительная обработка

Autocorrect

Autocomplete

Предварительная обработка

Крутяк, я изучаю машинное обучение в #УрФУ **#**Наука **! https://lo.st/12**:)

- Удаление хэштегов, гиперссылок и т.д.
- Токенизация строки
- Нижний регистр
- Удаление стоп-слов и знаков препинания
- Стемминг

Предварительная обработка

Удаление хэштегов, гиперссылок и т.д.

Крутяк, я изучаю машинное обучение в #УрФУ **#**Наука **! https://lo.st/12**:)

```
import re

tweet = re.sub(r'https?:\/\/.*[\r\n]*', '', tweet)
tweet = re.sub(r'#', '', tweet)
```

Крутяк, я изучаю машинное обучение в УрФУ Наука **!** :)

Предварительная обработка

Токенизация строки и нижний регистр

Крутяк, я изучаю машинное обучение в УрФУ Наука **!** :)

```
from nltk.tokenize import TweetTokenizer
tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, reduce_len=True)
tweet_tokens = tokenizer.tokenize(tweet)
```

```
['крутяк', ',', 'я', 'изучаю', 'машинное',
'обучение', 'в', 'урфу', 'наука', '!', ':)']
```

Предварительная обработка

Удаление стоп-слов и знаков препинания

```
['крутяк', ',', 'я', 'изучаю', 'машинное', 'обучение',
'в', 'урфу', 'наука', '!', ':)']
```

```
from nltk.corpus import stopwords
import string

stopwords_russian = stopwords.words('russian')
tweets_clean = []

for word in tweet_tokens:
    if (word not in stopwords_russian and word not in string.punctuation):
        tweets_clean.append(word)
```

```
['крутяк', 'изучаю', 'машинное', 'обучение', 'урфу', 'наука', ':)']
```

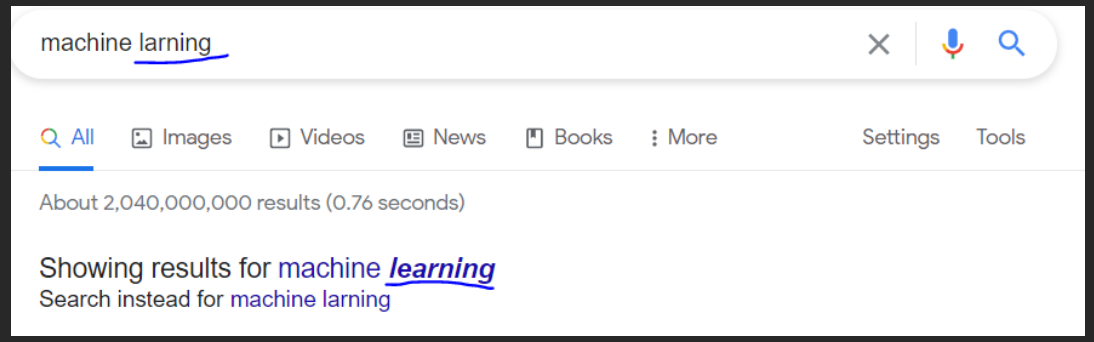
Содержание

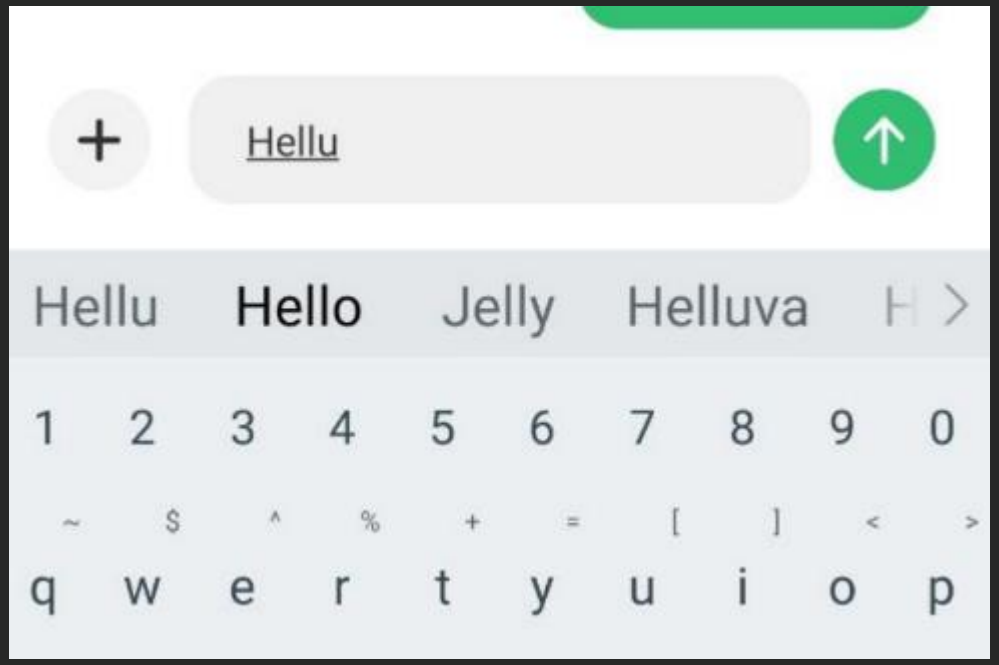
Предварительная обработка

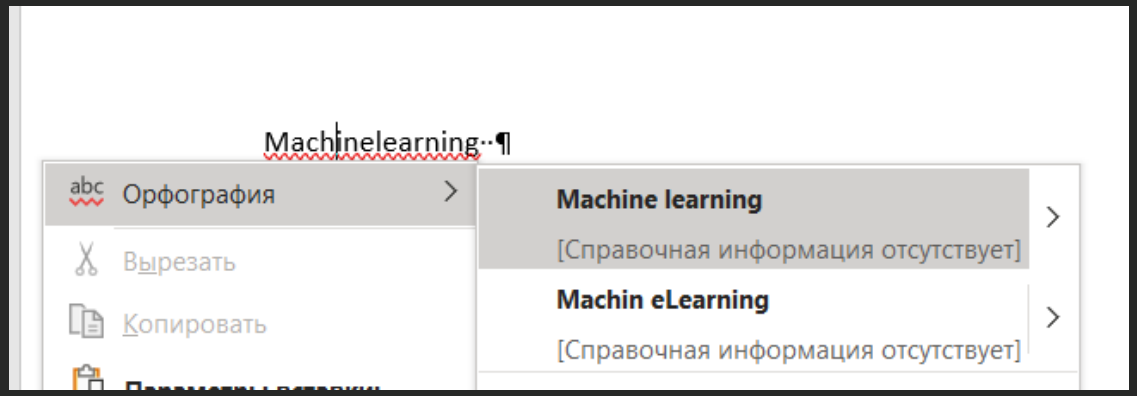
Autocorrect

Autocomplete

AutoCorrect







AutoCorrect

Алгоритм

1. Найдите слово с ошибкой
2. Создайте возможных кандидатов за n правок (n-edits)
3. Отфильтруйте кандидатов
4. Оцените вероятность слов

https://norvig.com/spell-correct.html

AutoCorrect

Найдите слово с ошибкой

leern

```
if word not in Vocab:
    misspelled = True
```

https://norvig.com/spell-correct.html

AutoCorrect

Создайте возможных кандидатов за n правок (n-edits)

Edit – некоторая операция, выполняемая над строкой, чтобы изменить ее

Insert (добавить букву)

```
insert_letter('re')  {'are', 'ere', 'ore', 'red'}
```

Delete (убрать букву)

```
delete_letter('wiall') {'wall', 'will'}
```

Switch (поменять местами 2 соседние буквы)

```
switch_letter('taem') {'tame', 'team'}
```

Replace (заменить 1 букву на другую)

```
replace_letter('red')
{'bed', 'fed', 'led', 'med', 'ned', 'rid', 'rud', 'wed'}
```

n – количество операций редактирования

https://norvig.com/spell-correct.html

AutoCorrect

Создайте возможных кандидатов за n правок (n-edits)

```
def edits1(word):
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]
    deletes    = [L + R[1:]               for L, R in splits if R]
    switches = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in letters]
    inserts    = [L + c + R               for L, R in splits for c in letters]
    return set(deletes +  switches  + replaces + inserts)

def edits2(word):
    return (e2 for e1 in edits1(word) for e2 in edits1(e1))
```

https://norvig.com/spell-correct.html

AutoCorrect

Отфильтруйте кандидатов

```
def candidates(word):
    return (known([word]) or known(edits1(word)) or known(edits2(word)) or [word])

def known(words):
    return set(w for w in words if w in Vocab)
```

https://norvig.com/spell-correct.html

AutoCorrect

Оцените вероятность слов

```
def P(word,Vocab):
    N=sum(Vocab.values())
    return Vocab[word] / N

def correction(word):
    return max(candidates(word), key=P)
```

https://norvig.com/spell-correct.html

AutoCorrect

Minimum Edit Distance

Динамическое программирование

Инициализация

$$D[0,0] = 0$$
$$D[i,0] = D[i-1,0] + del_{cost}$$
$$D[0,j] = D[0,j-1] + ins_{cost}$$

$$del_{cost} = 1$$
$$ins_{cost} = 1$$
$$rep_{cost} = 2$$

	#	h	a	t	e
#	0	1	2	3	4
I	1	2	3	4	5
o	2	3	4	5	6
v	3	4	5	6	7
e	4	5	6	7	6

Операции

$$D[i,j] = \min \begin{cases} D[i-1,j] + del_{cost} \\ D[i,j-1] + ins_{cost} \\ D[i-1,j-1] + \begin{cases} rep_{cost};\ if\ source[i] \neq target[j] \\ 0, if\ source[i] = target[j] \end{cases} \end{cases}$$

	#	f	e	a	r
#	0	1	2	3	4
f	1	0	1	2	3
u	2	1	2	3	4
r	3	2	3	4	3

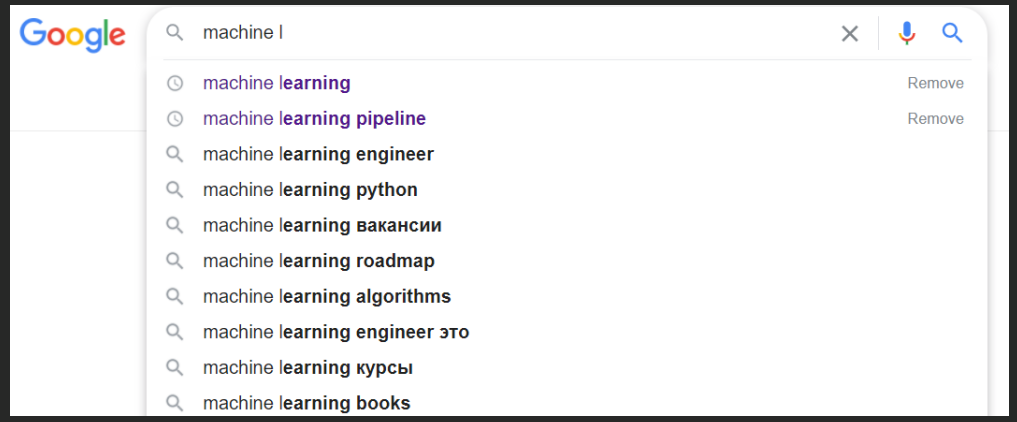
Содержание

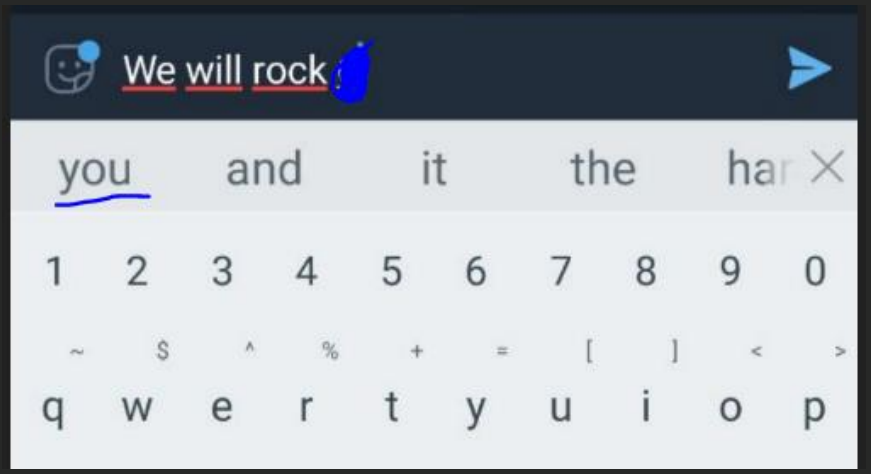
Предварительная обработка

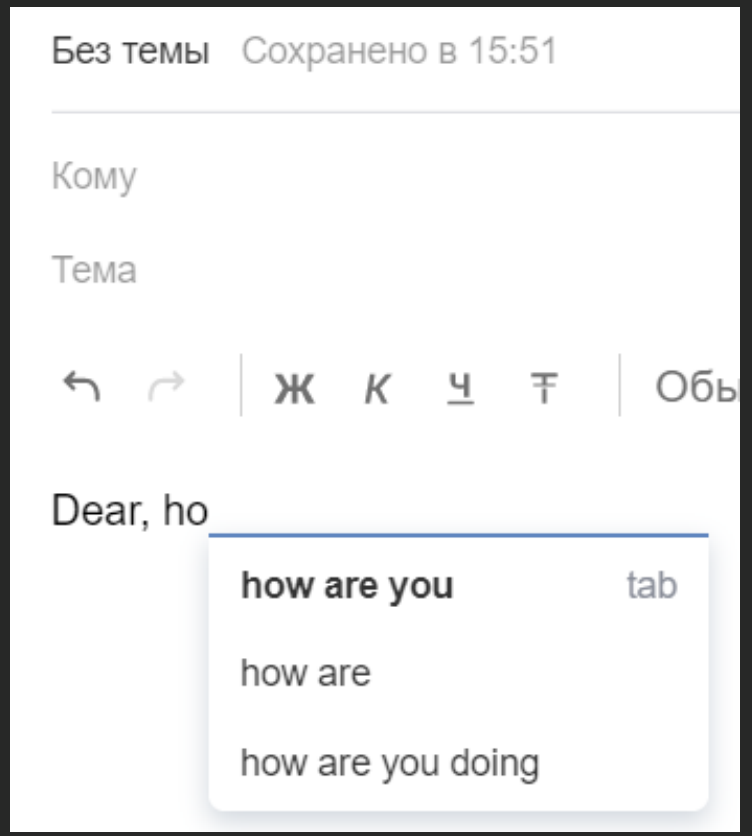
Autocorrect

Autocomplete

Autocomplete







Autocomplete

Языковая модель

Генерируем следующее слово с учетом предыдущих слов

Корпус → Языковая модель → Предсказание

Распознавание речи

Орфографическая коррекция

Дополняющая коммуникация

N-gram (N-Граммы)

Последовательности N-слов

I like machine learning

Deep learning is like machine learning

Uni-gram:

```
{('<s>',): 2, ('i',): 1, ('like',): 2, ('machine',): 2,
('learning',): 3, ('</s>',): 2, ('deep',): 1, ('is',): 1}
```

Bi-gram:

```
{('<s>', 'i'): 1, ('i', 'like'): 1, ('like', 'machine'): 2,
('machine', 'learning'): 2, ('learning', '</s>'): 2, ('<s>', 'deep'): 1,
('deep', 'learning'): 1, ('learning', 'is'): 1, ('is', 'like'): 1}
```

Tri-gram:

```
{('<s>', 'i', 'like'): 1, ('i', 'like', 'machine'): 1,
('like', 'machine', 'learning'): 2, ('machine', 'learning', '</s>'): 2,
('<s>', 'deep', 'learning'): 1, ('deep', 'learning', 'is'): 1,
('learning', 'is', 'like'): 1, ('is', 'like', 'machine'): 1}
```

Вероятности

Unigrams (1 слово) $P(w) = \frac{C(w)}{m} \quad m = 10$

I like machine learning Deep learning is like machine learning

Uni-gram:

```
{('<s>',): 2, ('i',): 1, ('like',): 2, ('machine',): 2, ('learning',): 3,
('</s>',): 2, ('deep',): 1, ('is',): 1}
```

Bigram (2 слова) $P(y|x) = \frac{C(x\ y)}{\sum_w C(x\ w)} = \frac{C(x\ y)}{C(x)}$

Bi-gram:

```
{('<s>', 'i'): 1, ('i', 'like'): 1, ('like', 'machine'): 2,
('machine', 'learning'): 2, ('learning', '</s>'): 2, ('<s>', 'deep'): 1,
('deep', 'learning'): 1, ('learning', 'is'): 1, ('is', 'like'): 1}
```

Trigram $P(w_3|w_1^2) = \frac{C(w_1^2\ w_3)}{C(w_1^2)} \quad C(w_1^2\ w_3) = C(w_1\ w_2\ w_3) = C(w_1^3)$

N-gram $P(w_N|w_1^{N-1}) = \frac{C(w_1^{N-1}\ w_N)}{C(w_1^{N-1})} \quad C(w_1^{N-1}\ w_N) = C(w_1^N)$

Вероятность последовательности

I like machine learning

$$P(A,B,C,D) = P(A) * P(B|A) * P(C|A,B) * P(D|A,B,C)$$

$P(\text{I like machine learning }) = P(I) * P(like|I) * P(machine|I\ like) * P(learning|I\ like\ machine)$

$\cong P(I) * P(like|I) * P(machine|like) * P(learning|machine)$

Марковское предположение: имеют значение только последние N слов

Bigram $P(w_n|w_1^{n-1}) \cong P(w_n|w_{n-1})$

N-gram $P(w_n|w_1^{n-1}) \cong P(w_n|w_{n-N+1}^{n-1})$

Языковая модель на N-Граммах

I like machine learning

Deep learning is like machine learning

bigram probabilities

	like	deep	is	learning	machine	i	</s>	<unk>
(<s>,)	0.100	0.200	0.100	0.100	0.100	0.200	0.100	0.100
(is,)	0.222	0.111	0.111	0.111	0.111	0.111	0.111	0.111
(like,)	0.100	0.100	0.100	0.100	0.300	0.100	0.100	0.100
(i,)	0.222	0.111	0.111	0.111	0.111	0.111	0.111	0.111
(deep,)	0.111	0.111	0.111	0.222	0.111	0.111	0.111	0.111
(learning,)	0.091	0.091	0.182	0.091	0.091	0.091	0.273	0.091
(machine,)	0.100	0.100	0.100	0.300	0.100	0.100	0.100	0.100

	like	deep	is	learning	machine	i	</s>	<unk>
(<s>, i)	0.222	0.111	0.111	0.111	0.111	0.111	0.111	0.111
(machine, learning)	0.100	0.100	0.100	0.100	0.100	0.100	0.300	0.100
(<s>, <s>)	0.100	0.200	0.100	0.100	0.100	0.200	0.100	0.100
(is, like)	0.111	0.111	0.111	0.111	0.222	0.111	0.111	0.111
(i, like)	0.111	0.111	0.111	0.111	0.222	0.111	0.111	0.111
(like, machine)	0.100	0.100	0.100	0.300	0.100	0.100	0.100	0.100
(learning, is)	0.222	0.111	0.111	0.111	0.111	0.111	0.111	0.111
(<s>, deep)	0.111	0.111	0.111	0.222	0.111	0.111	0.111	0.111
(deep, learning)	0.111	0.111	0.222	0.111	0.111	0.111	0.111	0.111

<unk> - незнакомые слова или не часто встречающиеся слова

$$P(w_N|w_1^{N-1}) = \frac{C(w_1^{N-1}\, w_N)}{C(w_1^{N-1})} \qquad P(w_N|w_1^{N-1}) = \frac{C(w_1^{N-1}\, w_N) + k}{C(w_1^{N-1}) + k * V} \qquad \log P(w_N|w_1^{N-1})$$

Языковая модель на N-Граммах

Разделить корпус на подмножества Train-Validation-Test

Если корпус небольшой: 80% - 10% - 10%

Если корпус большой : 98% - 1% -1% (типа и так много)

Непрерывный текст

Случайные короткие последовательности

Метрики для оценки
Perplexity

$$PP(W) = P(s_1, s_2, \dots, s_N)^{-\frac{1}{m}}$$

s_i - i-е предложение в выборке W,

m – количество уникальных слов

Чем меньше perplexity – тем вроде лучше

Пусть $m = 100$

$P(W) = 0.95$ $PP(W) \sim 1.000513$

$P(W) = 10^{-100}$ $PP(W) \sim 10$

Bigram

$$PP(W) = \sqrt[m]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

$$\log PP(W) = -\frac{1}{m} \sum_{i=1}^{N} \log_2 (P(w_i|w_{i-1}))$$

Метрики для оценки
Perplexity

I like machine learning

Deep learning is like machine learning

```
Perplexity for first train sample: 3.1049

Perplexity for second train sample: 3.3014
```

I like learning

learning like life

```
Perplexity for test sample: 3.9874    Perplexity for test sample: 5.1017
```

```
The previous words are 'machine learning’,
and the suggested word is `is` with a probability of 0.1818

The previous words are 'machine learning', the suggestion must start with `d`
and the suggested word is `deep` with a probability of 0.0909

The previous words are 'machine learning', the suggestion must start with `c`
and the suggested word is `None` with a probability of 0.0000
```

Итого

Предварительная обработка

Тут нам помогает nltk

Разбиваем на слова, удаляем ненужное и т.д.

Autocorrect

Редактирование строки (n-edit)

Edit Distance и Динамическое программирование

Autocomplete

Языковые модели

N-Граммы

Perplexity

Вопросы, пожелания, предложения

?????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.05
Нейронки для NLP, let’s go

Докладчик
Долганов Антон

Ранее в Машинном Обучении

Вложения (Embedding)

- Токены – цифры
- Перед началом - Лемматизация, стемминг

Статистический подход

- Счетчики / Частоты слов
- TF-IDF
- Классические алгоритмы МО

Контекстный подход

- Word by Word, Word by Doc
- Word2Vec

Векторные модели

- Алгебра над векторными представлениями
- Наивный машинный перевод

Содержание

Embeddings (опять)

1D-Светрки

Рекуррентные Нейронные сети
Recurrent Neural Networks

Содержание

Embeddings (опять)

1D-Свётрки

Рекуррентные Нейронные сети
Recurrent Neural Networks

Предварительная обработка в TensorFlow

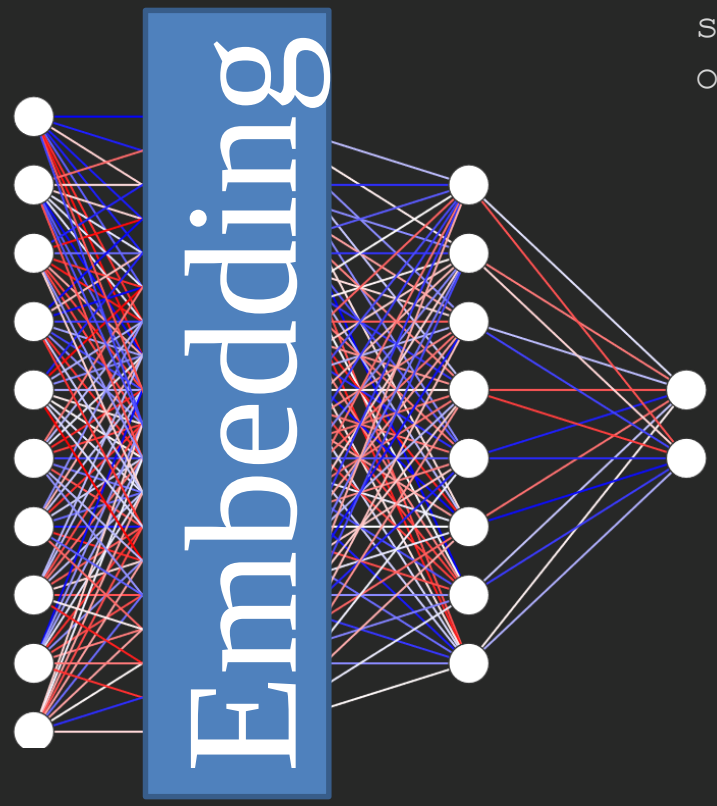
```
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization


import re ; import string; import tensorflow as tf


def custom_standardization(input_data):
  lowercase = tf.strings.lower(input_data)
  stripped_html = tf.strings.regex_replace(lowercase, '<br />', ' ')
  return tf.strings.regex_replace(stripped_html,
                                  '[%s]' % re.escape(string.punctuation), '')


vectorize_layer = TextVectorization(
standardize=custom_standardization,max_tokens=vocab_size,
output_mode='int', output_sequence_length=sequence_length)
```

0	I
1	am
2	happy
3	study
4	machine
5	learning
6	sad
7	hate
8	deep
9	…

https://www.tensorflow.org/tutorials/text/word_embeddings

Слой Embedding

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Activation, Dense, Embedding, GlobalAveragePooling1D
```




```
vectorize_layer = TextVectorization(
standardize=custom_standardization,max_tokens=vocab_size,
output_mode='int', output_sequence_length=sequence_length)
```

$$\begin{bmatrix} 4 \\ 8 \\ 15 \\ \dots \end{bmatrix} \quad \begin{bmatrix} 16 \\ 23 \\ 42 \\ \dots \end{bmatrix} \quad \begin{bmatrix} \dots \\ \dots \\ \dots \end{bmatrix} \quad \begin{bmatrix} 69 \\ 420 \\ 1337 \\ \dots \end{bmatrix}$$

0	I
1	am
2	happy
3	study
4	machine
5	learning
6	sad
7	hate
8	deep
9	…

Turns positive integers (indexes) into dense vectors of fixed size.

```
model = Sequential([
  vectorize_layer,
  Embedding(vocab_size, embedding_dim, name="embedding"),
  GlobalAveragePooling1D(),
  Dense(16, activation='relu'),
  Dense(2)])
```

https://www.tensorflow.org/tutorials/text/word_embeddings

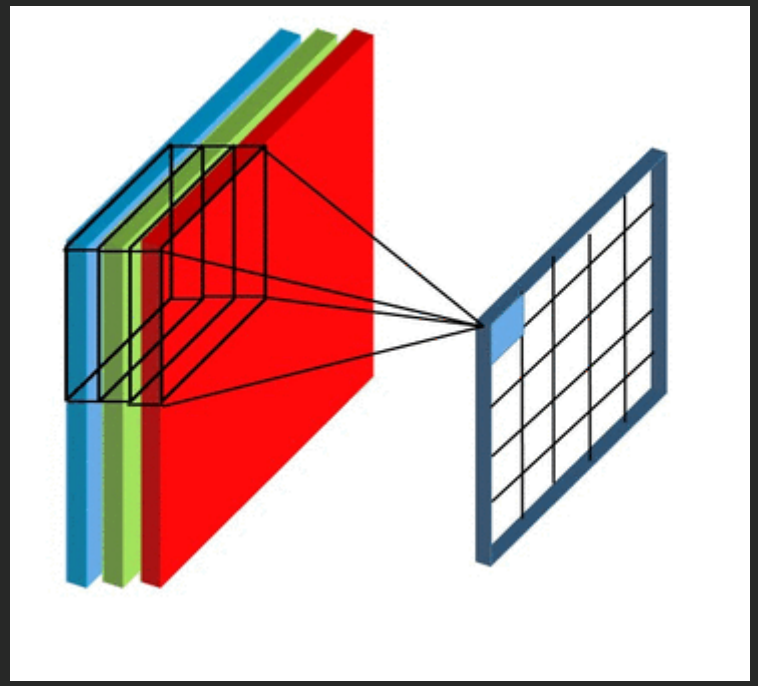
Содержание

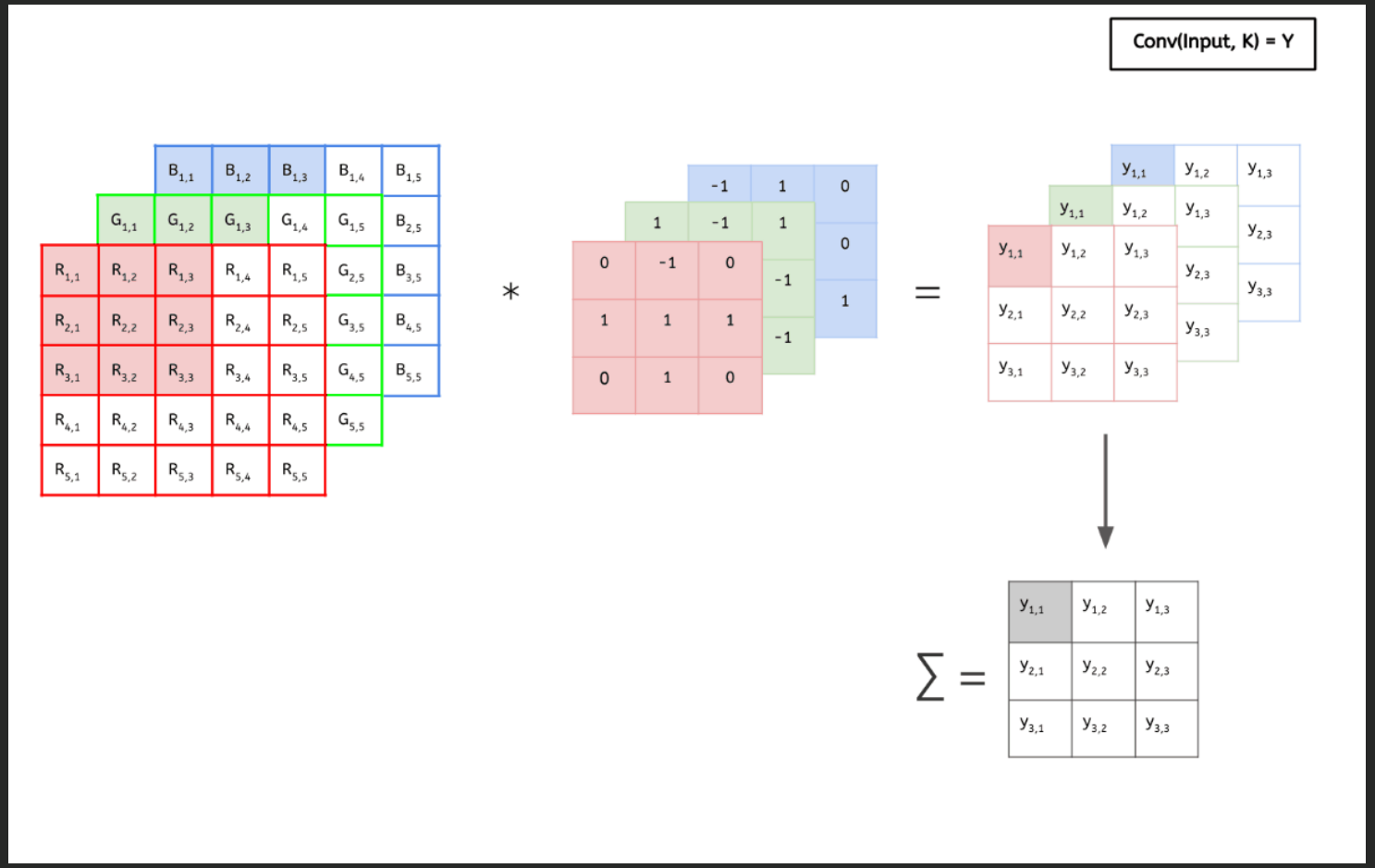
Embeddings (опять)

1D-Свёртки

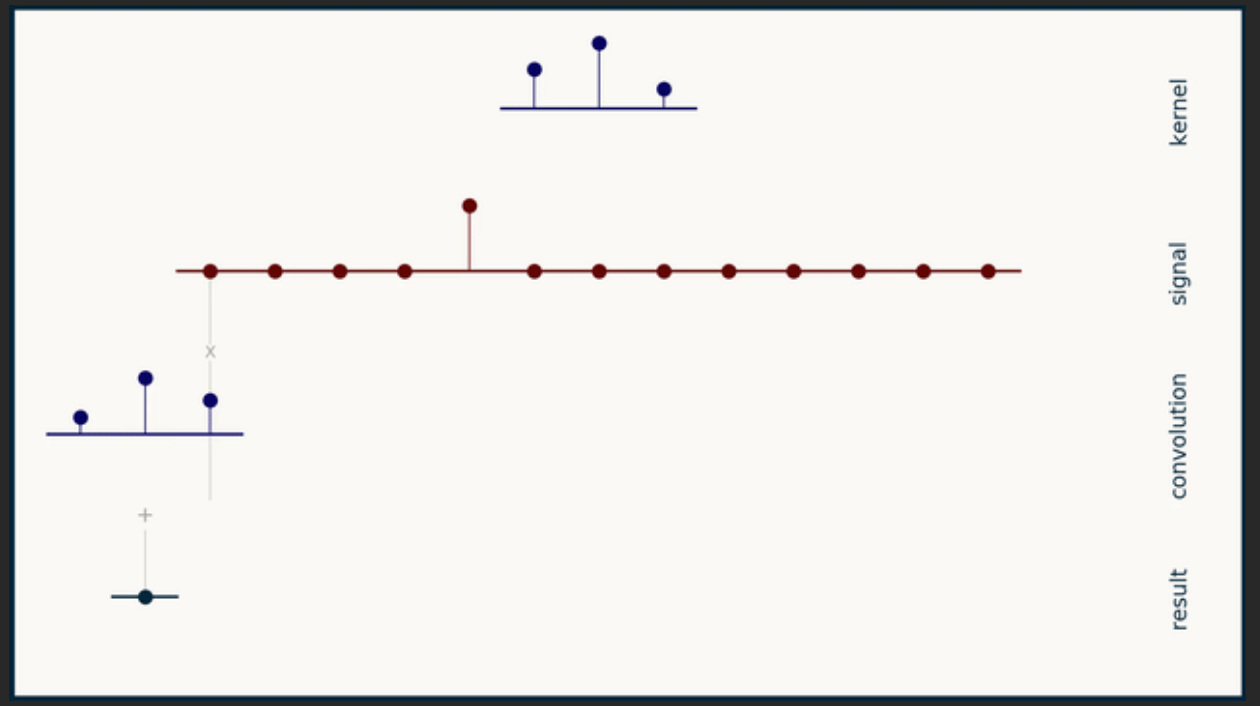
Рекуррентные Нейронные сети
Recurrent Neural Networks

Convolution Neural Networks



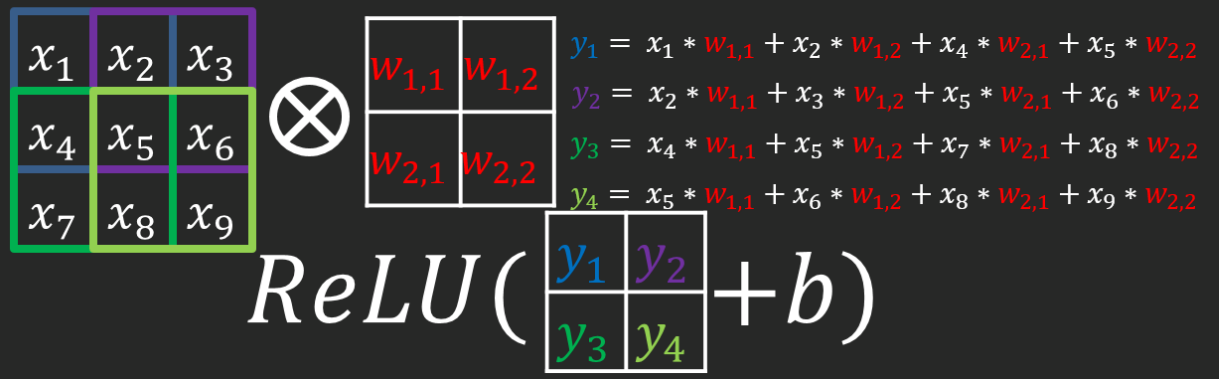


1-D Convolutions



1-D Свертки

Сверточные Нейронные Сети
Convolution Neural Networks



ЭТО НЕ Свёртки 1x1

$y_1 = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4$

```
layers.Conv1D(10, 4, activation='relu')
```

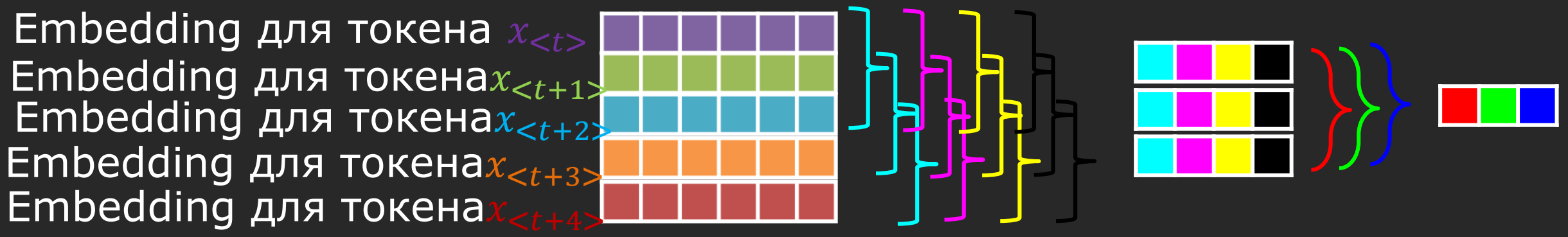
$y_2 = x_2 * w_1 + x_3 * w_2 + x_4 * w_3 + x_5 * w_4$

$y_3 = x_3 * w_1 + x_4 * w_2 + x_5 * w_3 + x_6 * w_4$

$y_4 = x_4 * w_1 + x_5 * w_2 + x_6 * w_3 + x_7 * w_4$

Примеры из TensorFlow
1-D Сверток

```
model = tf.keras.Sequential([
      layers.Embedding(vocab_size, 64, mask_zero=True),
      layers.Conv1D(64, 5, padding="valid", activation="relu", strides=2),
      layers.GlobalMaxPooling1D(),
      layers.Dense(num_labels)
  ])
```



https://www.tensorflow.org/tutorials/load_data/text

Содержание

Embeddings (опять)

1D-Свётрки

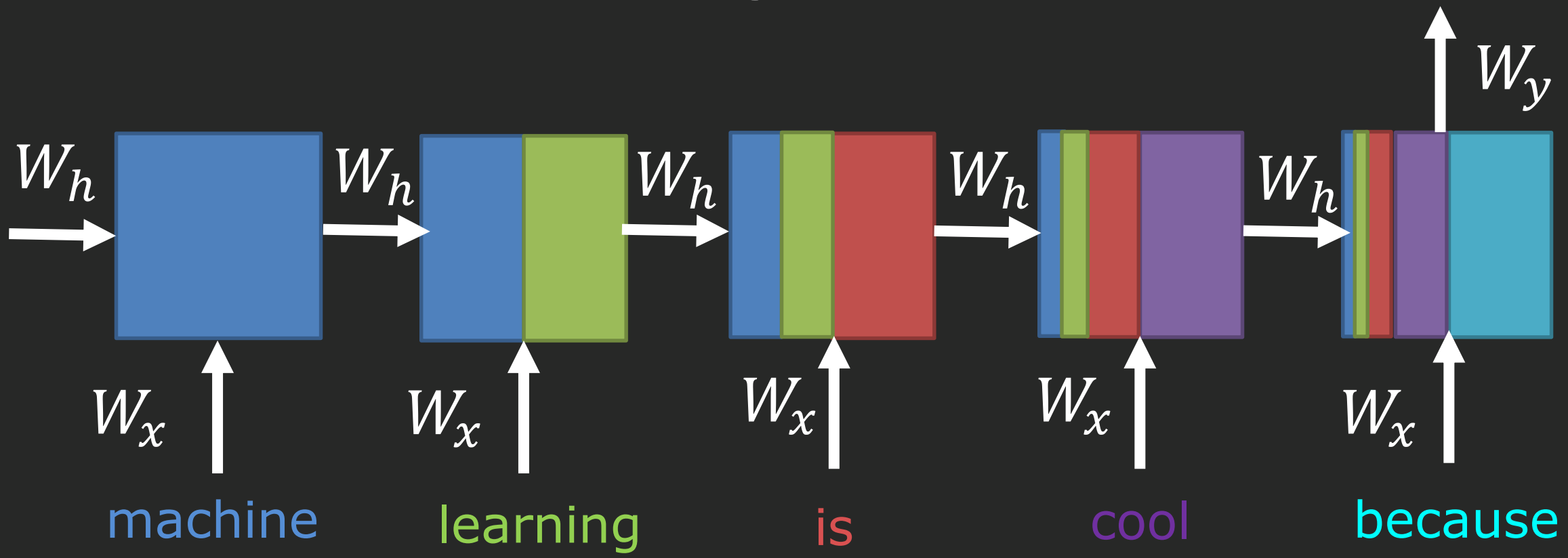
Рекуррентные Нейронные сети
Recurrent Neural Networks

Когда N-Gram могут подводить

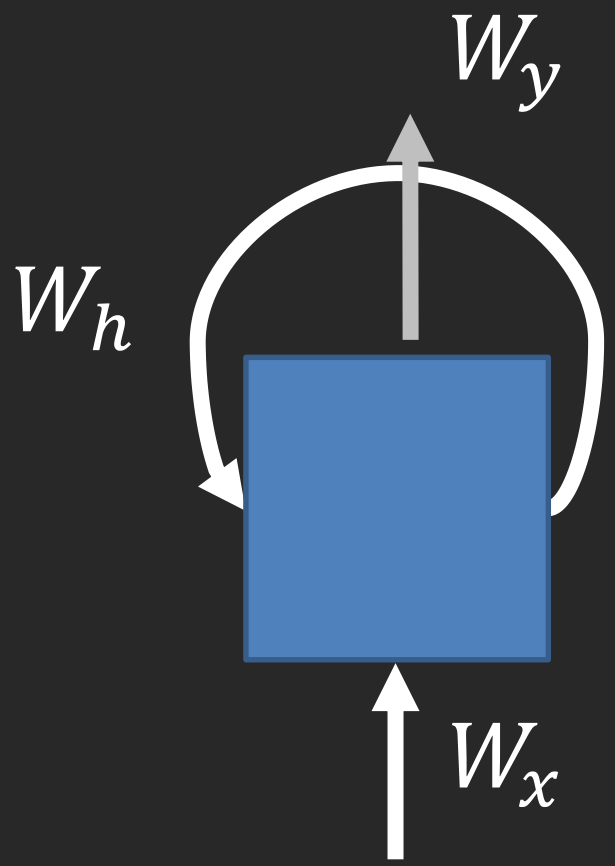
Tom, a boy who lived in the next house, run to meet his father.

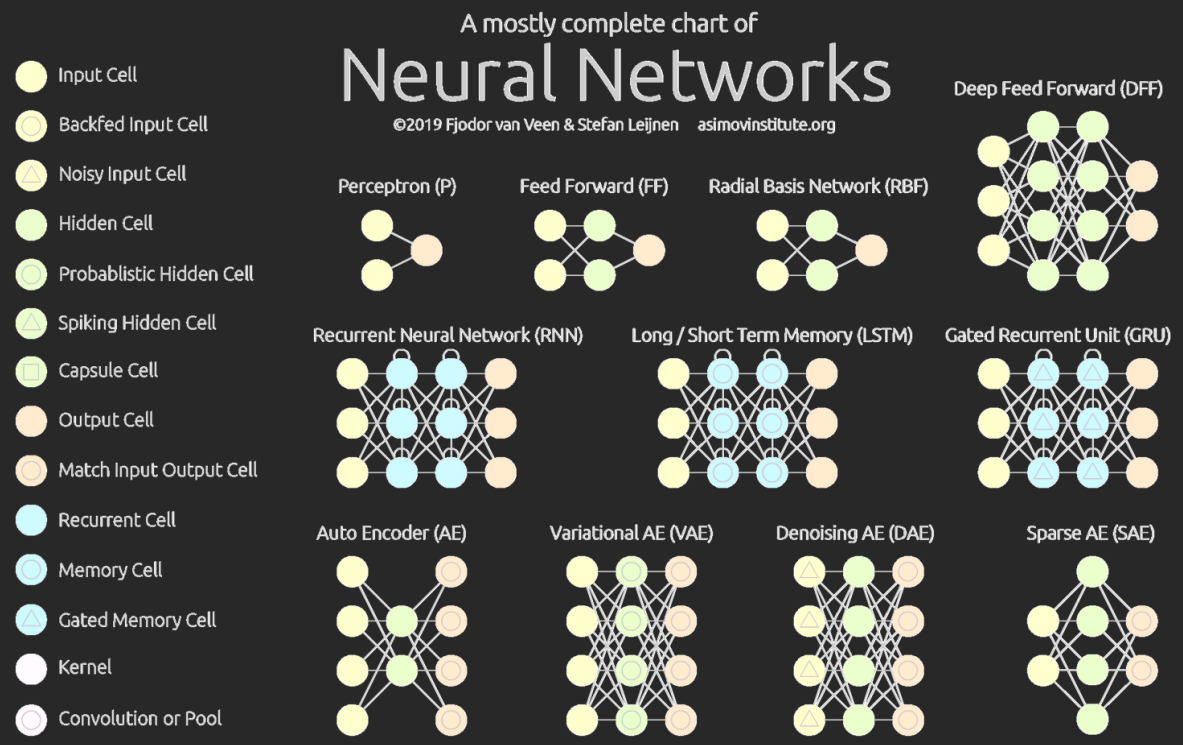
Recurrent Neural Networks
Рекуррентные Нейронные Сети

machine learning is cool because _____



Recurrent Neural Networks





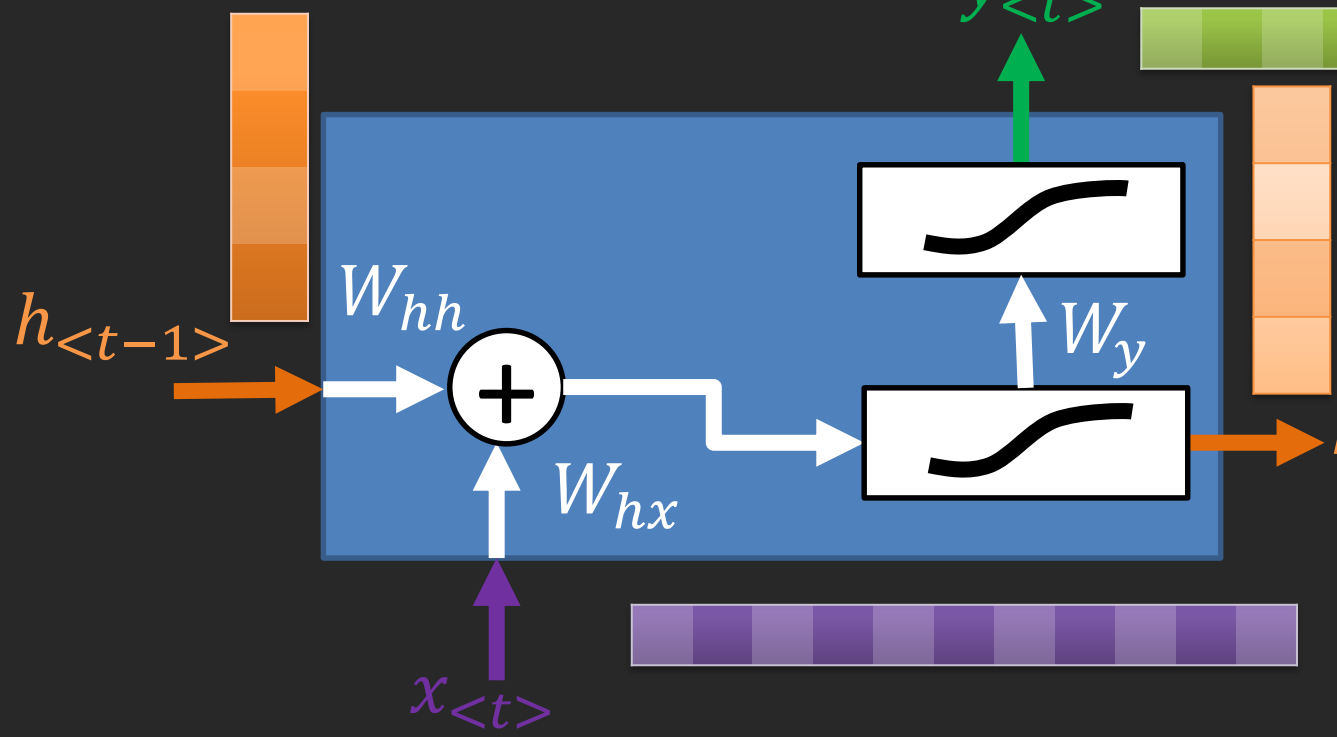
https://www.asimovinstitute.org/neural-network-zoo/

Recurrent Neural Networks
Математика Внутри

$$h_{<t>} = g(W_h[h_{<t-1>}, x_{<t>}] + b_h)$$

$$\hat{y}_{<t>} = g(W_y h_{<t>} + b_y)$$

```
import tensorflow as tf
tf.keras.layers.SimpleRNN(units,
                          activation='tanh')
```

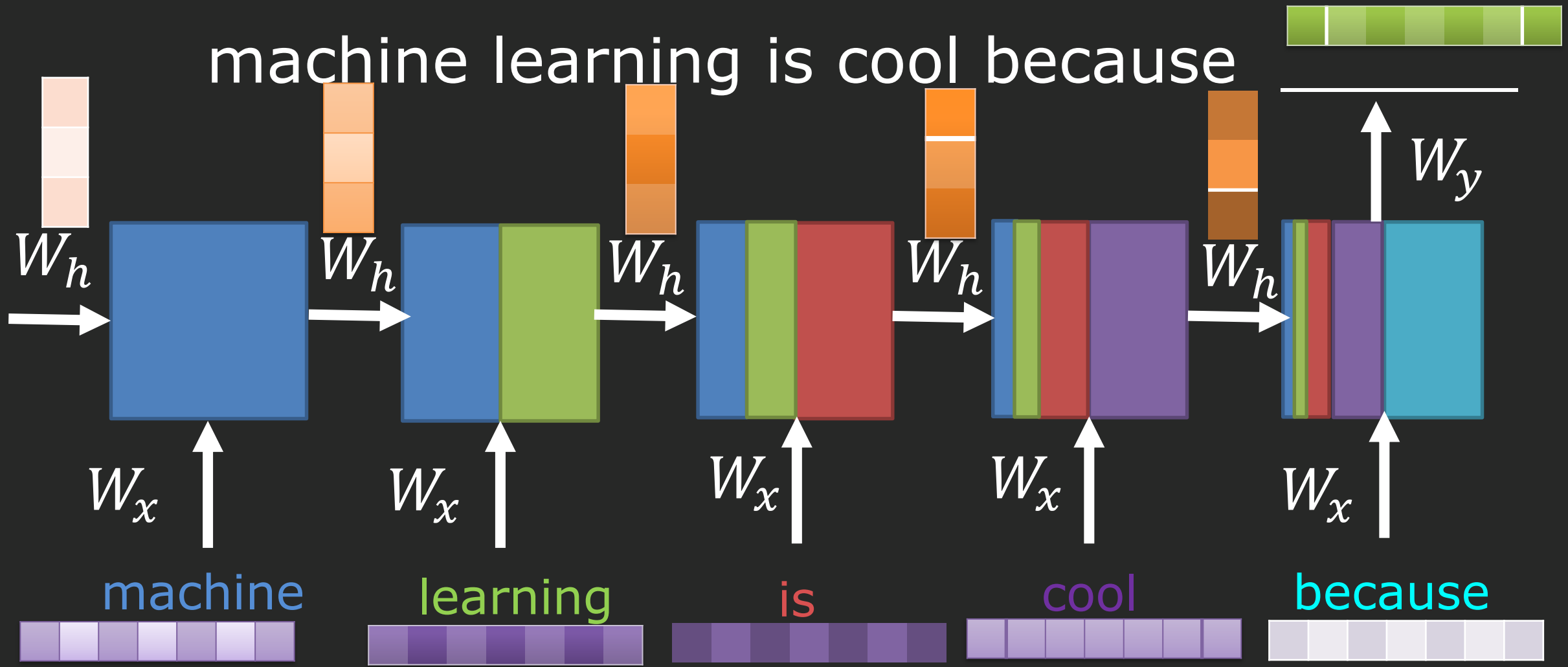


Embedding для токена $\hat{y}_{<t>}$

$$Cost = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{K} y_j^{<t>} \log \hat{y}_j^{<t>}$$

Embedding для токена $x_{<t>}$

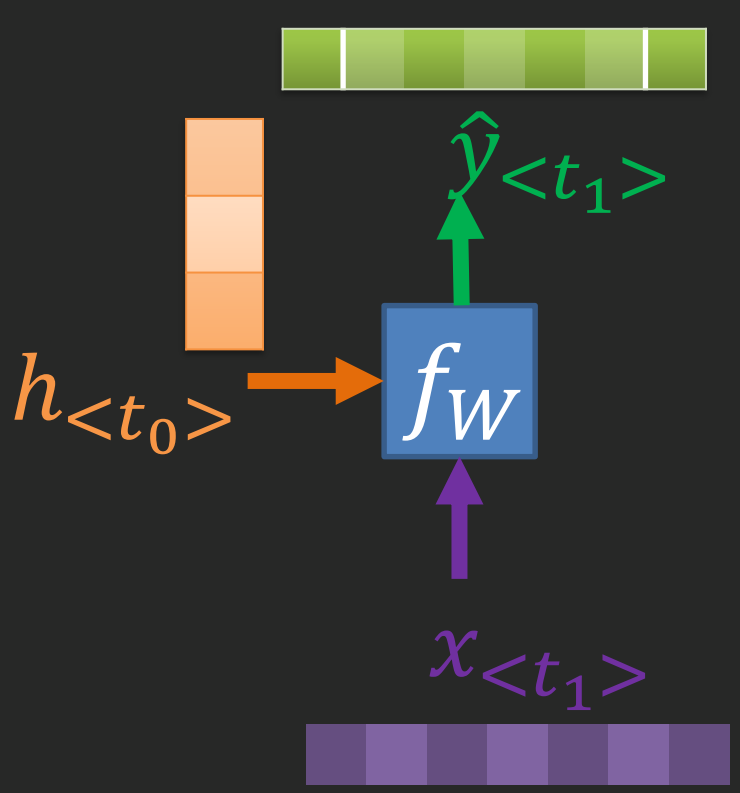
Recurrent Neural Networks



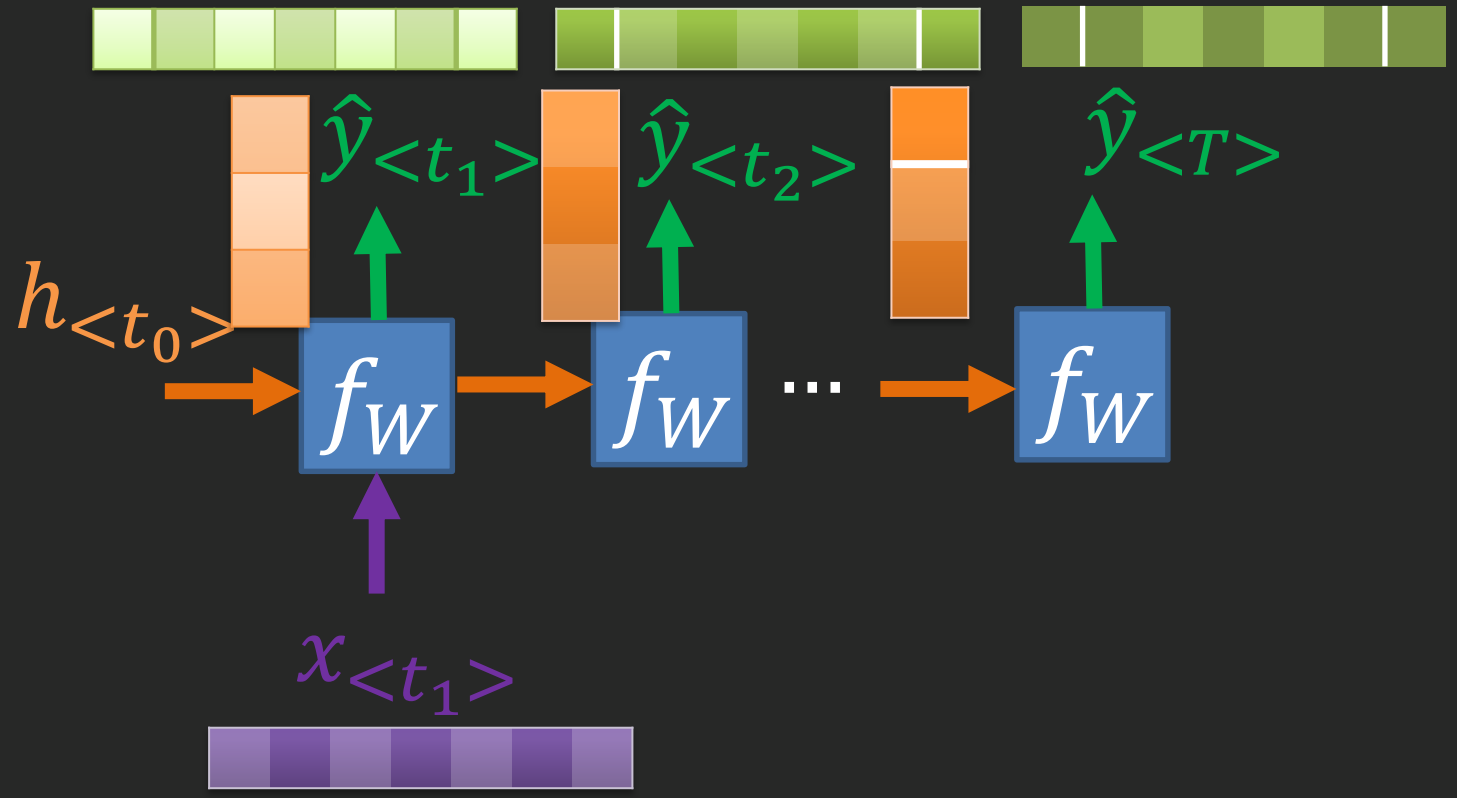
Recurrent Neural Networks

Задачи

One to One

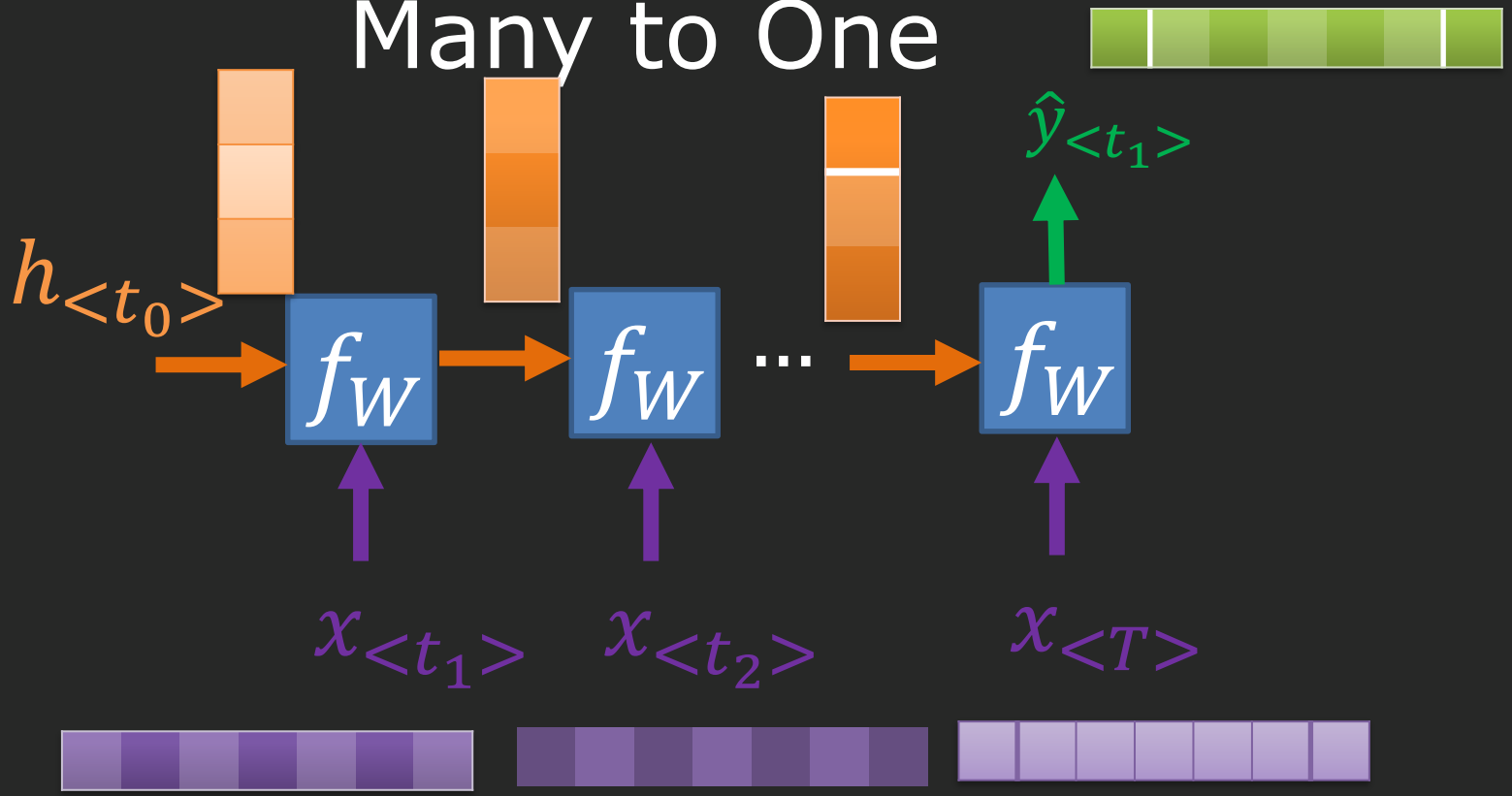


One to Many

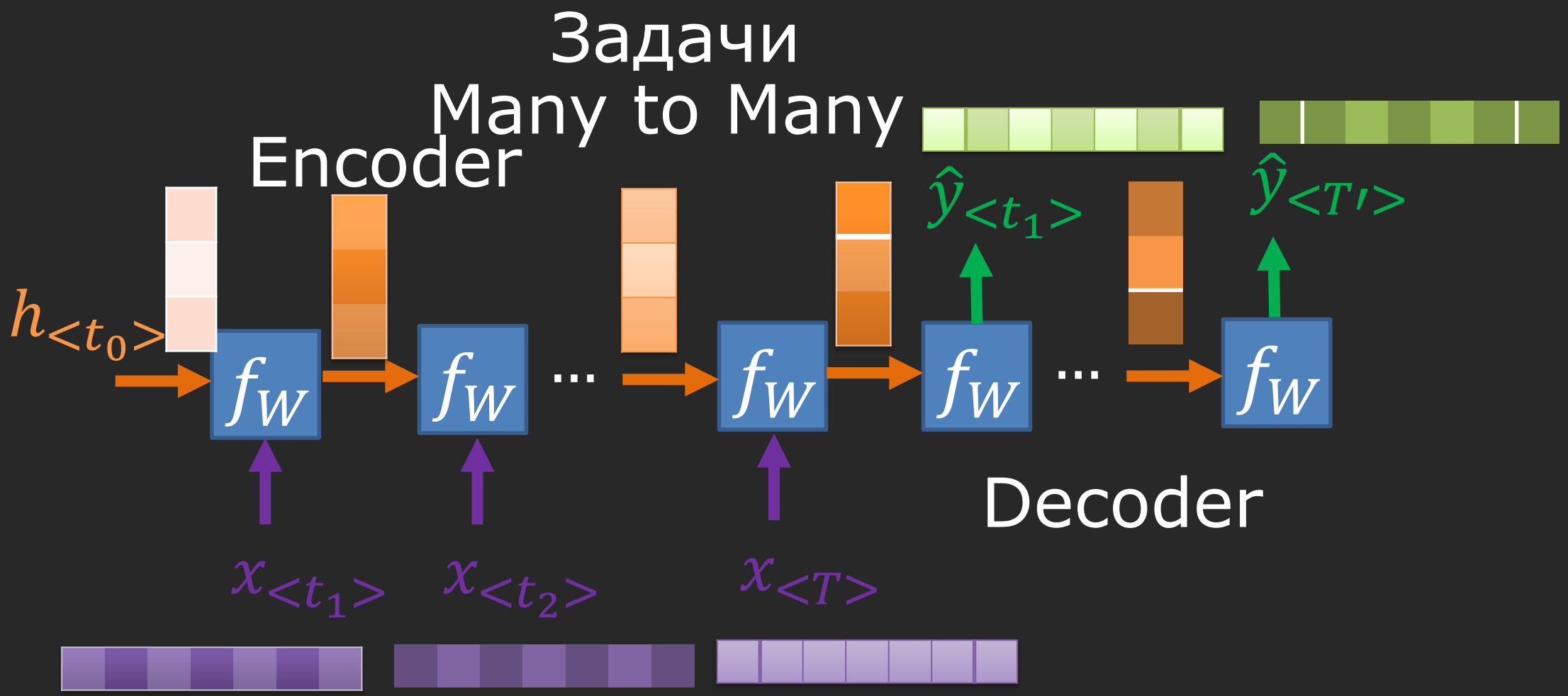


Recurrent Neural Networks

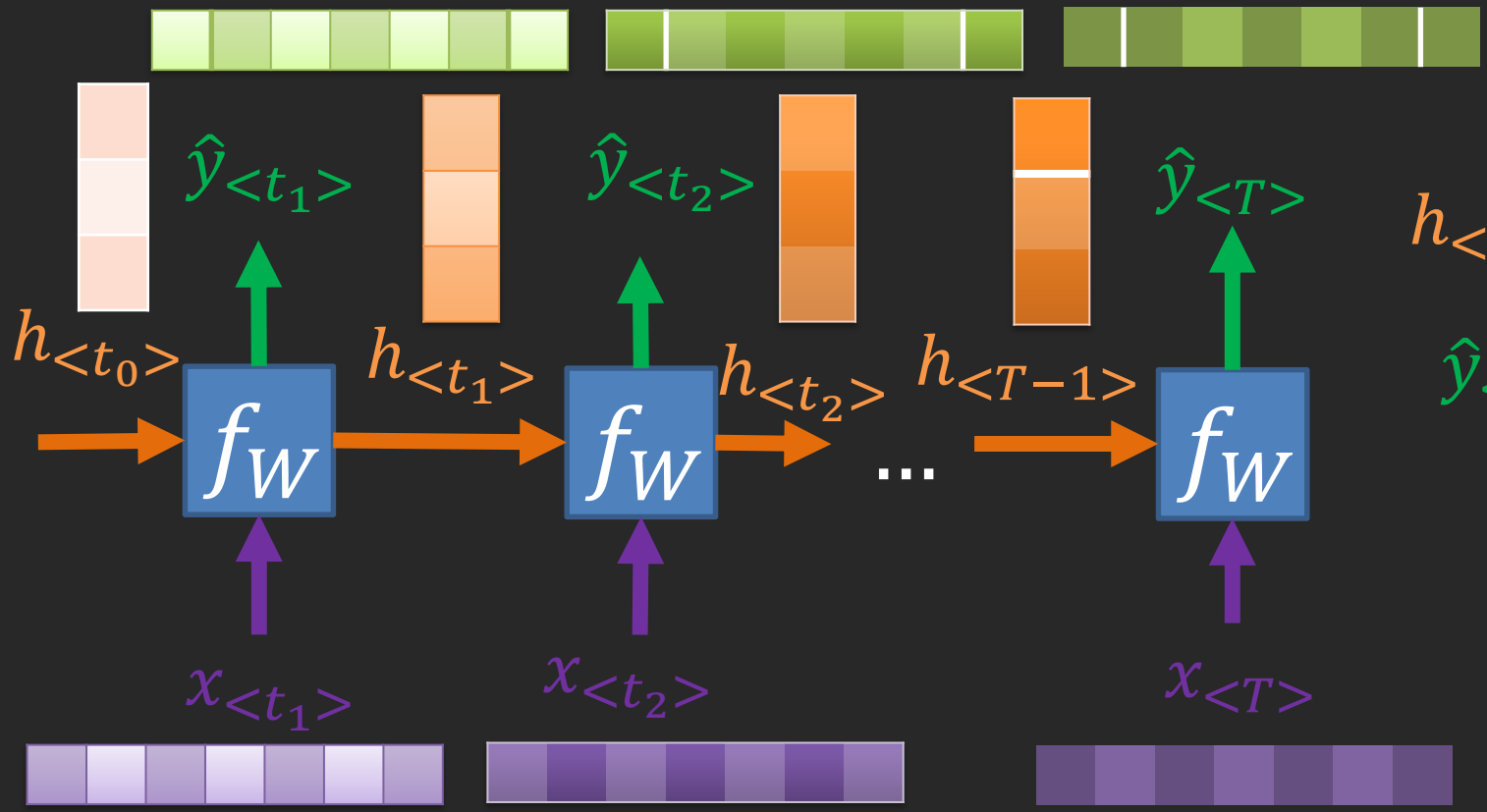
Задачи
Many to One



Recurrent Neural Networks



Recurrent Neural Networks
Математика Внутри



$$h_{<t>} = g(W_h[h_{<t-1>}, x_{<t>}] + b_h)$$

$$\hat{y}_{<t>} = g(W_y h_{<t>} + b_y)$$

Содержание

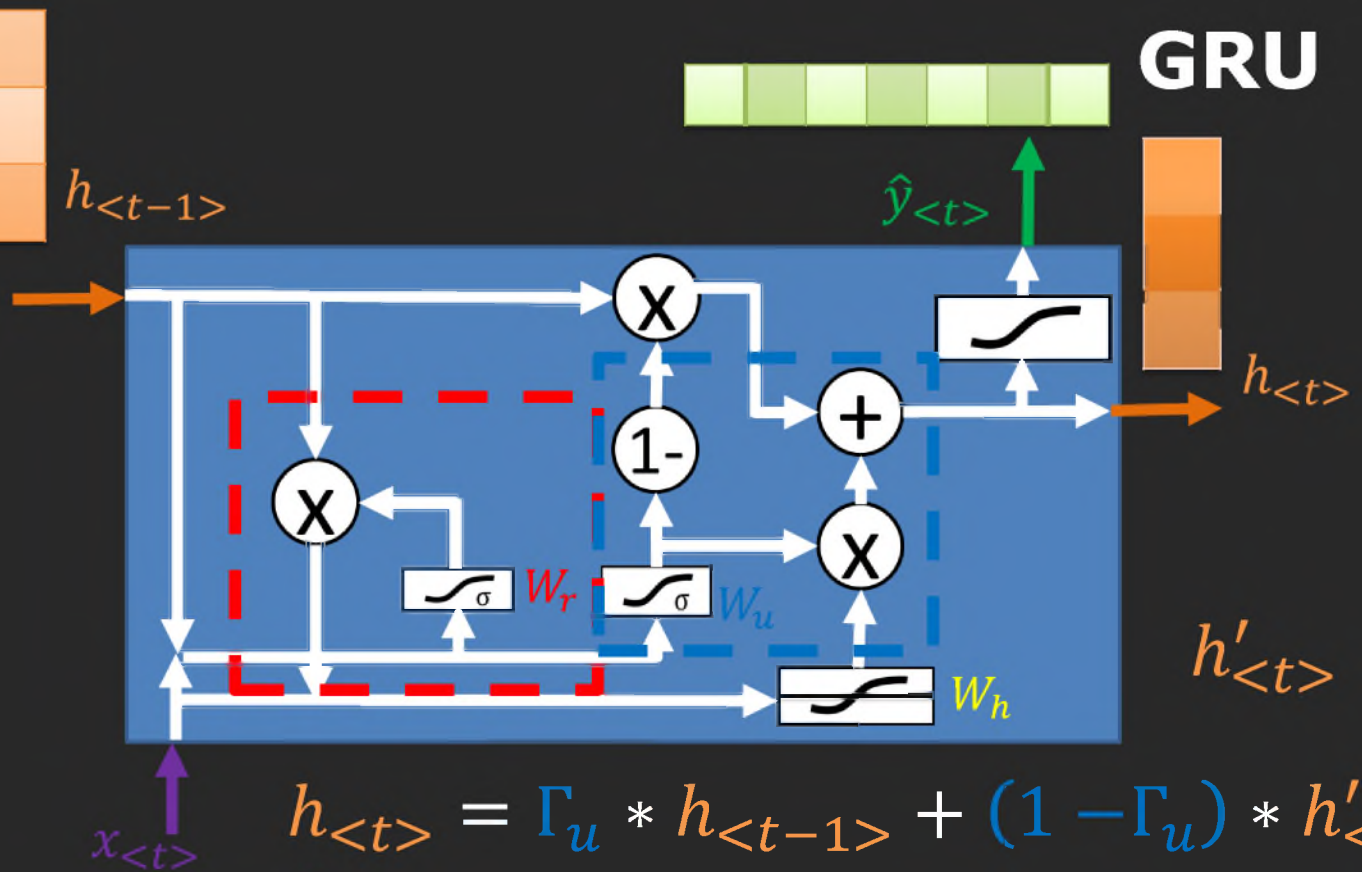
Embeddings (опять)

1D-Свётрки

Рекуррентные Нейронные сети
Recurrent Neural Networks
GRU

Gated Recurrent Units

GRU



Gate to remove information

$$\Gamma_r = \sigma(W_r[h_{<t-1>}, x_{<t>}] + b_r)$$

Gate to update information

$$\Gamma_u = \sigma(W_u[h_{<t-1>}, x_{<t>}] + b_u)$$

Hidden state candidate

$$h'_{<t>} = \tanh(W_h[\Gamma_r * h_{<t-1>}, x_{<t>}] + b_h)$$

$$h_{<t>} = \Gamma_u * h_{<t-1>} + (1 - \Gamma_u) * h'_{<t>}$$

$$\hat{y}_{<t>} = g(W_y h_{<t>} + b_y)$$

```
import tensorflow as tf
tf.keras.layers.GRU(units)
```

Содержание

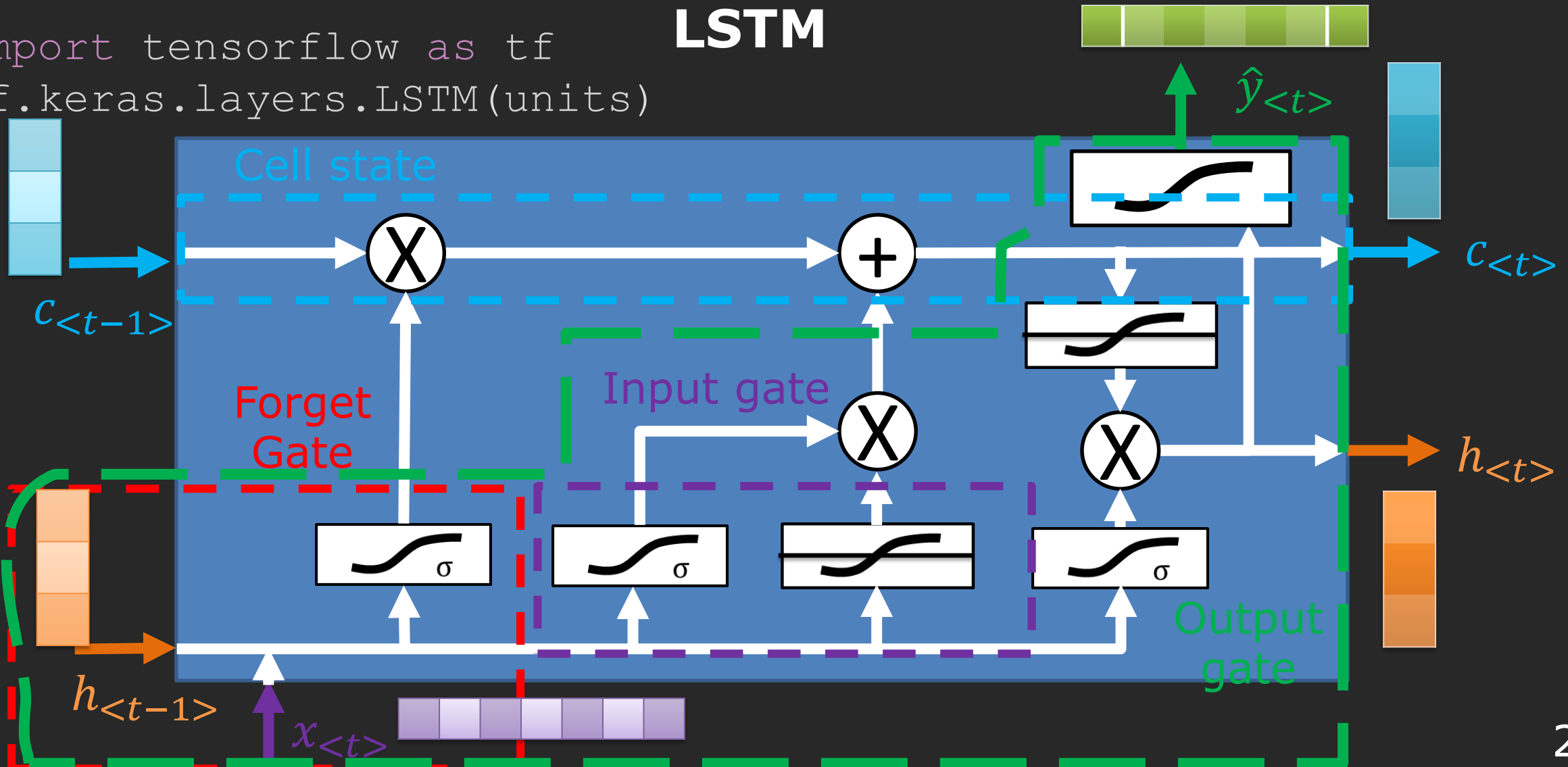
Embeddings (опять)

1D-Свётрки

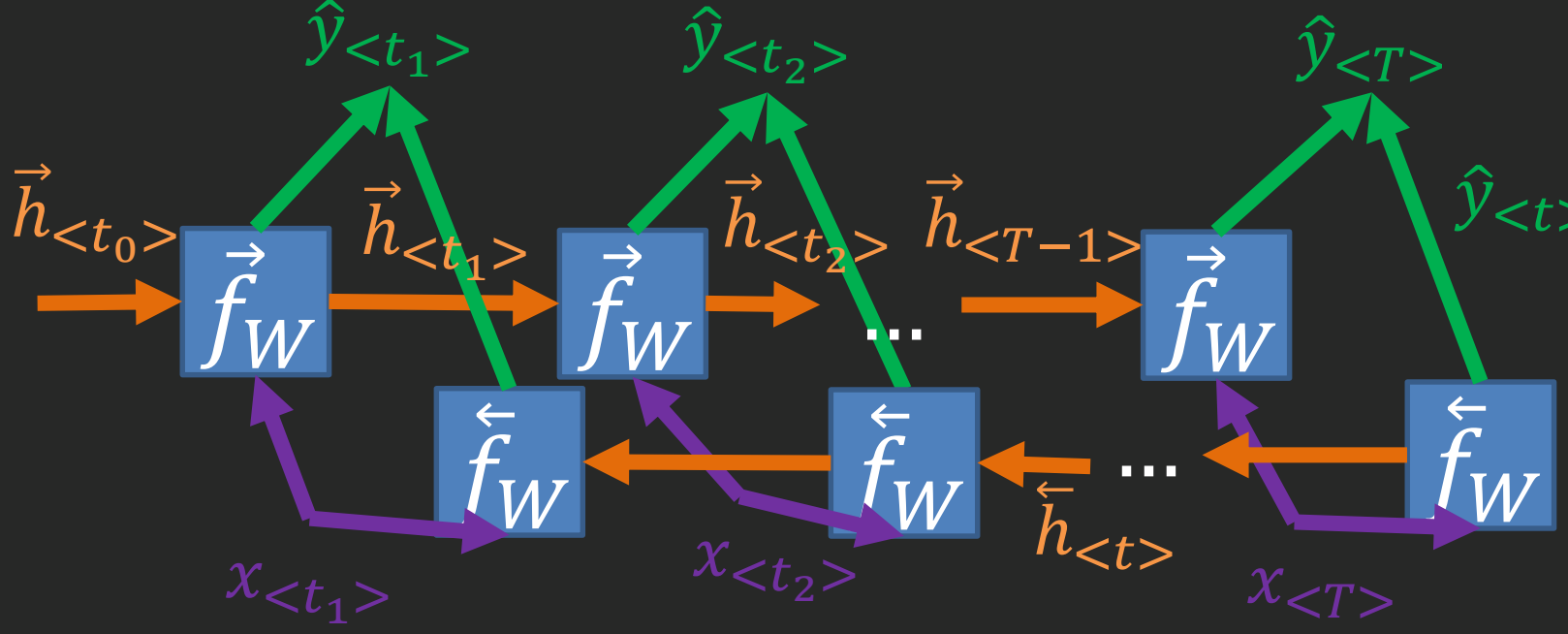
Рекуррентные Нейронные сети
Recurrent Neural Networks
LSTM

Long Short-Term Memory LSTM

```
import tensorflow as tf
tf.keras.layers.LSTM(units)
```



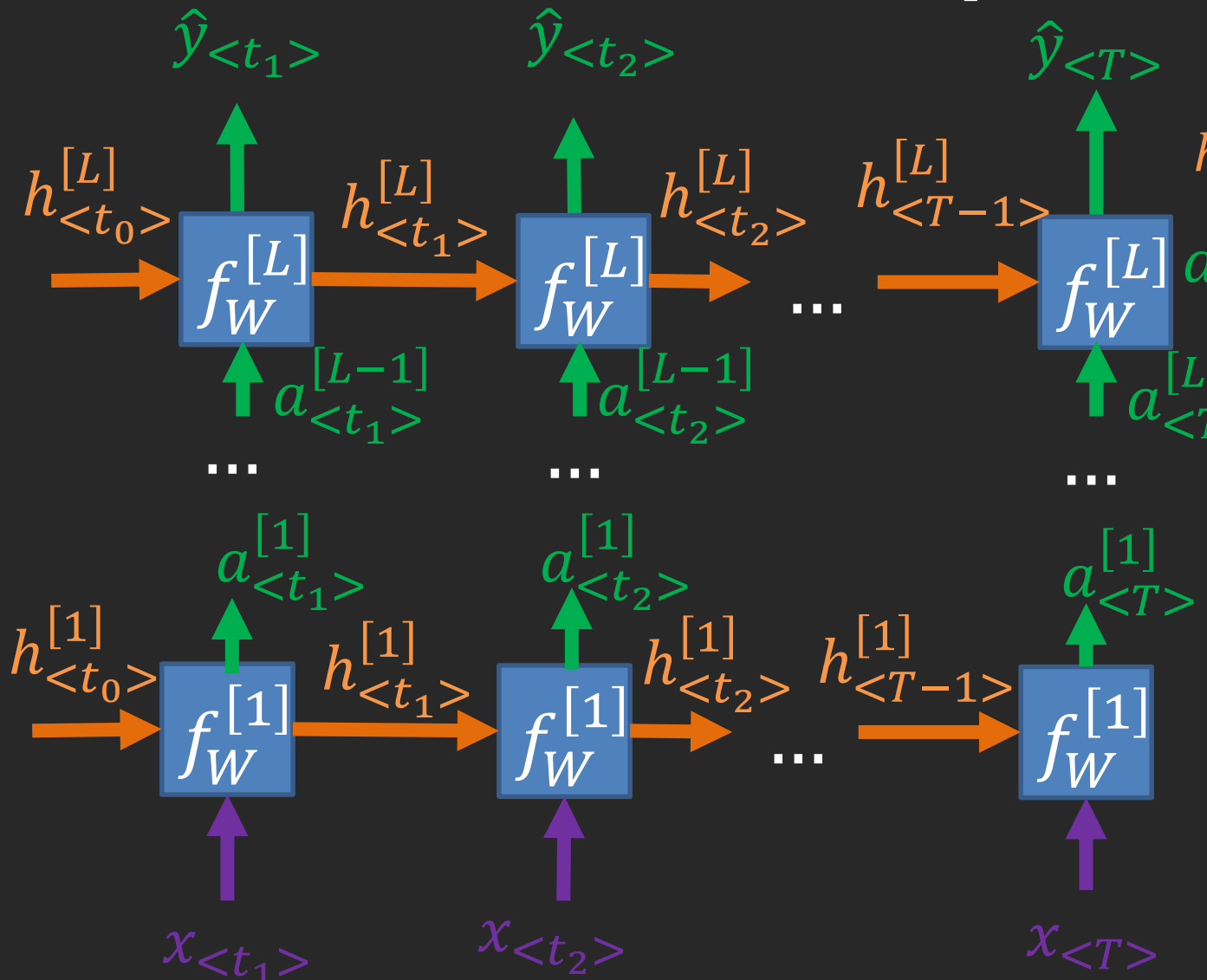
Bi-directional RNN



$$\hat{y}_{<t>} = g(W_y[\vec{h}_{<t_0>}, \overleftarrow{h}_{<t>}] + b_y)$$

```
import tensorflow as tf
tf.keras.layers.Bidirectional(layer)
```

Deep RNN



$$h^{[l]}_{<t>} = g^{[l]}(W^{[l]}_h \left[h^{[l]}_{<t-1>}, a^{[l-1]}_{<t>}\right] + b^{[l]}_h)$$

$$a^{[l]}_{<t>} = g^{[l]}(W^{[l]}_a h^{[l]}_{<t>} + b^{[l]}_a)$$

ELMo

“Эволюция” Continuous Bag of Words

I like to study machine learning because it is fun

like to ____ machine learning study

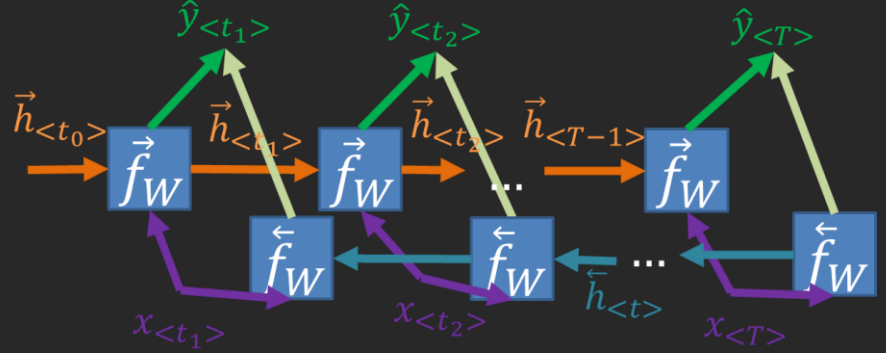
Word2Vec / FastText фиксированное окно

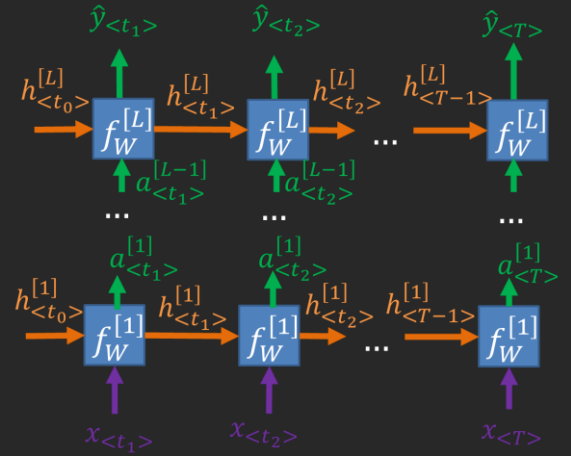
“Transfer Learning”

ELMo Embeddings from Language Model

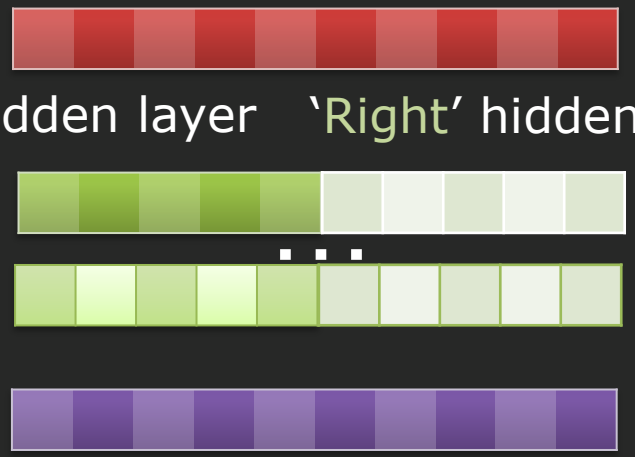
full context

Deep Bi-directional LSTM





ElMO Embedding для токена $x_{<t>}$



https://jalammar.github.io/illustrated-bert/

Embedding для токена $x_{<t>}$

ELMo

```
import tensorflow_hub as hub
import tensorflow as tf

elmo = hub.load("https://tfhub.dev/google/elmo/3")

# Некое предложение
x = tf.constant(["Roasted ants are a popular snack in Columbia"])

# ELMo Embedding
embeddings = elmo.signatures["default"](x)["elmo"]
embeddings
```

```
<tf.Tensor: shape=(1, 8, 1024), dtype=float32)>
```

TensorFlow Примеры

```
model = keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.LSTM(128))
model.add(layers.Dense(10))
```

```
model = keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))
model.add(layers.GRU(256, return_sequences=True))
model.add(layers.SimpleRNN(128))
model.add(layers.Dense(10))
```

```
model = keras.Sequential()
model.add(layers.Bidirectional(layers.LSTM(64, return_sequences=True),
                               input_shape=(5, 10)))
model.add(layers.Bidirectional(layers.LSTM(32)))
model.add(layers.Dense(10))
```

https://www.tensorflow.org/guide/keras/rnn

TensorFlow Примеры
Seq2Seq

```
encoder_input = layers.Input(shape=(None,))
encoder_embedded = layers.Embedding(input_dim=encoder_vocab,
                                    output_dim=64)(encoder_input)
output, state_h, state_c = layers.LSTM(64, return_state=True,
                                       name="encoder")(encoder_embedded)
encoder_state = [state_h, state_c]
decoder_input = layers.Input(shape=(None,))
decoder_embedded = layers.Embedding(input_dim=decoder_vocab,
                                    output_dim=64)(decoder_input)
decoder_output = layers.LSTM(64,
                     name="decoder")(decoder_embedded, initial_state=encoder_state)
output = layers.Dense(10)(decoder_output)
model = keras.Model([encoder_input, decoder_input], output)
```

https://www.tensorflow.org/guide/keras/rnn

Content

Embeddings

- Обучаемое Представление словаря

1D-Convolutions

- Почему бы и нет? (ЭТО НЕ 1x1 СВЁРТКИ!!!)

Recurrent Neural Networks

Рекуррентные Нейронные Сети

- По-по-по-повторяем полносвязный слой
- Скрытое состояние (hidden state)
- Deep RNN, Bi-Directional

➢ **GRU**

- hidden state, remove gate, update gate

➢ **LSTM**

- cell state, forget gate, input gate, output gate
- ELMo (более модные вектора для слов)

Вопросы, пожелания, предложения

?????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.06
Внимание все что вам нужно и вот это всё

Докладчик
Долганов Антон

Ранее в Машинном Обучении .

Embeddings
- Обучаемое Представление словаря

1D-Convolutions
- Почему бы и нет? (ЭТО НЕ 1x1 СВЁРТКИ!!!)

Recurrent Neural Networks

Рекуррентные Нейронные Сети
- По-по-по-повторяем полносвязный слой
- Скрытое состояние (hidden state)
- Deep RNN, Bi-Directional

➢ **GRU**
- hidden state, remove gate, update gate

➢ **LSTM**
- cell state, forget gate, input gate, output gate
- ELMo (более модные вектора для слов)

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Про Токенизацию (опять)

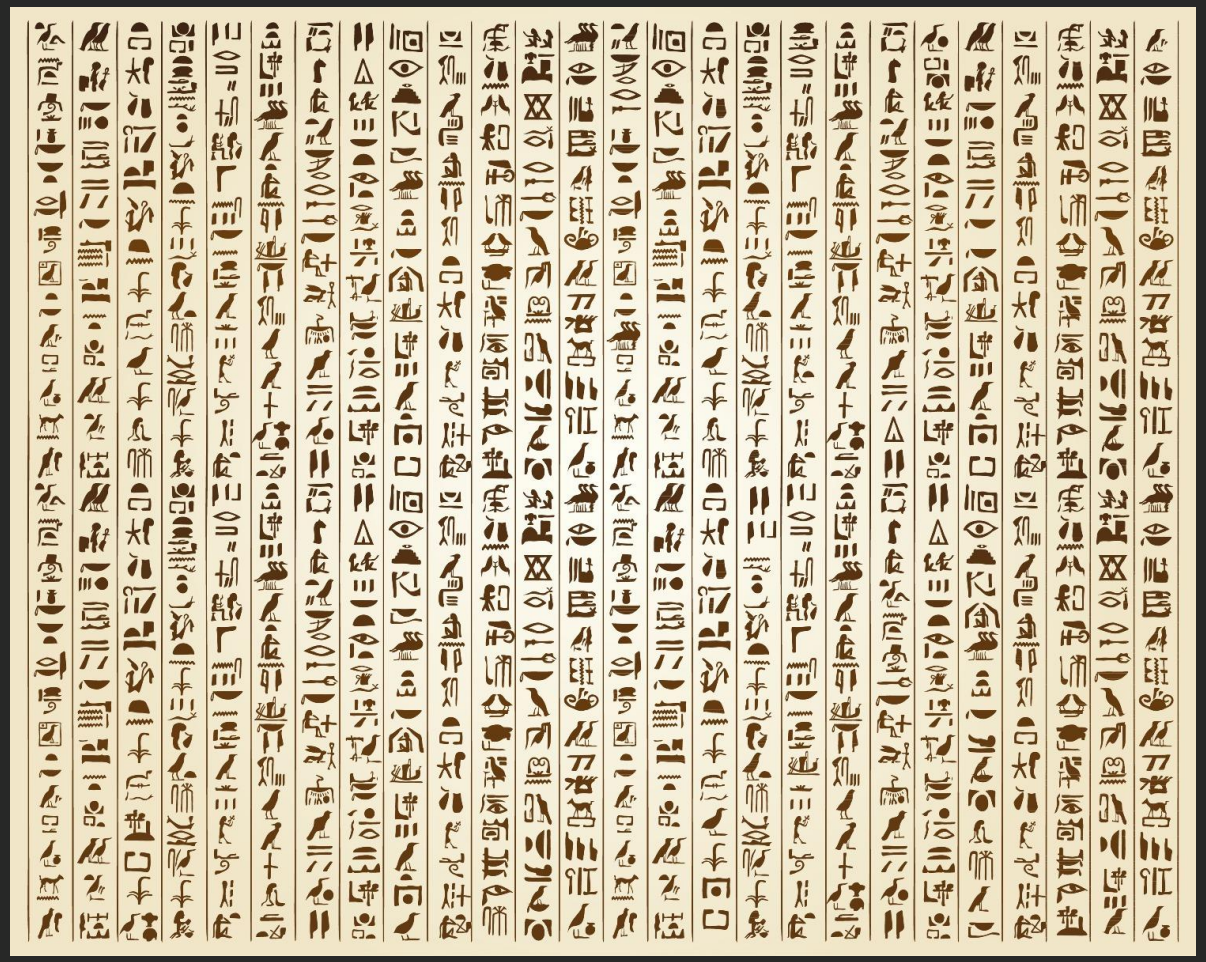
Внимание

Трансформеры

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Про Токенизацию



Египетские символы

https://usefulcharts.com/blogs/charts/evolution-of-the-english-alphabet

Byte-Pair Encoding (BPE)

Словарь (Слов)	("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
Алфавит (Букв)	("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)
Самая частая пара	("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)
	. . .
После нескольких итераций	("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

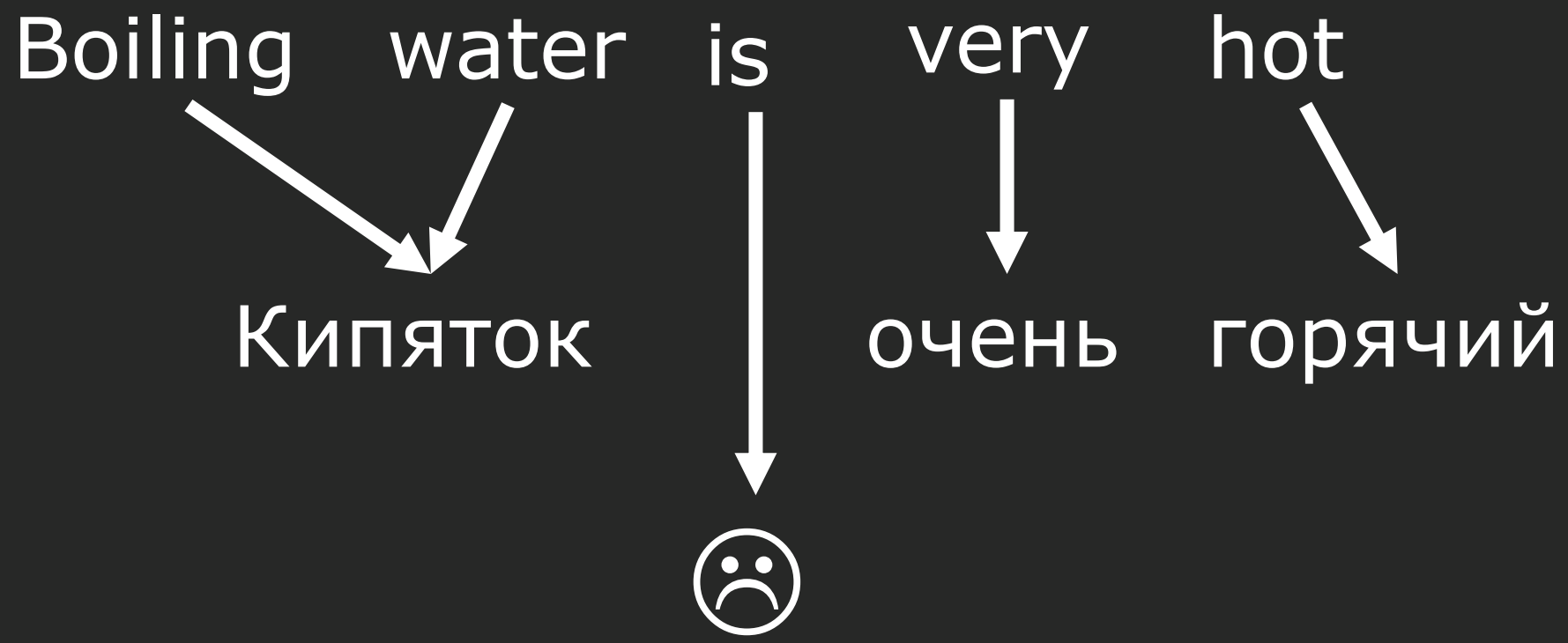
https://huggingface.co/docs/transformers/tokenizer_summary

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

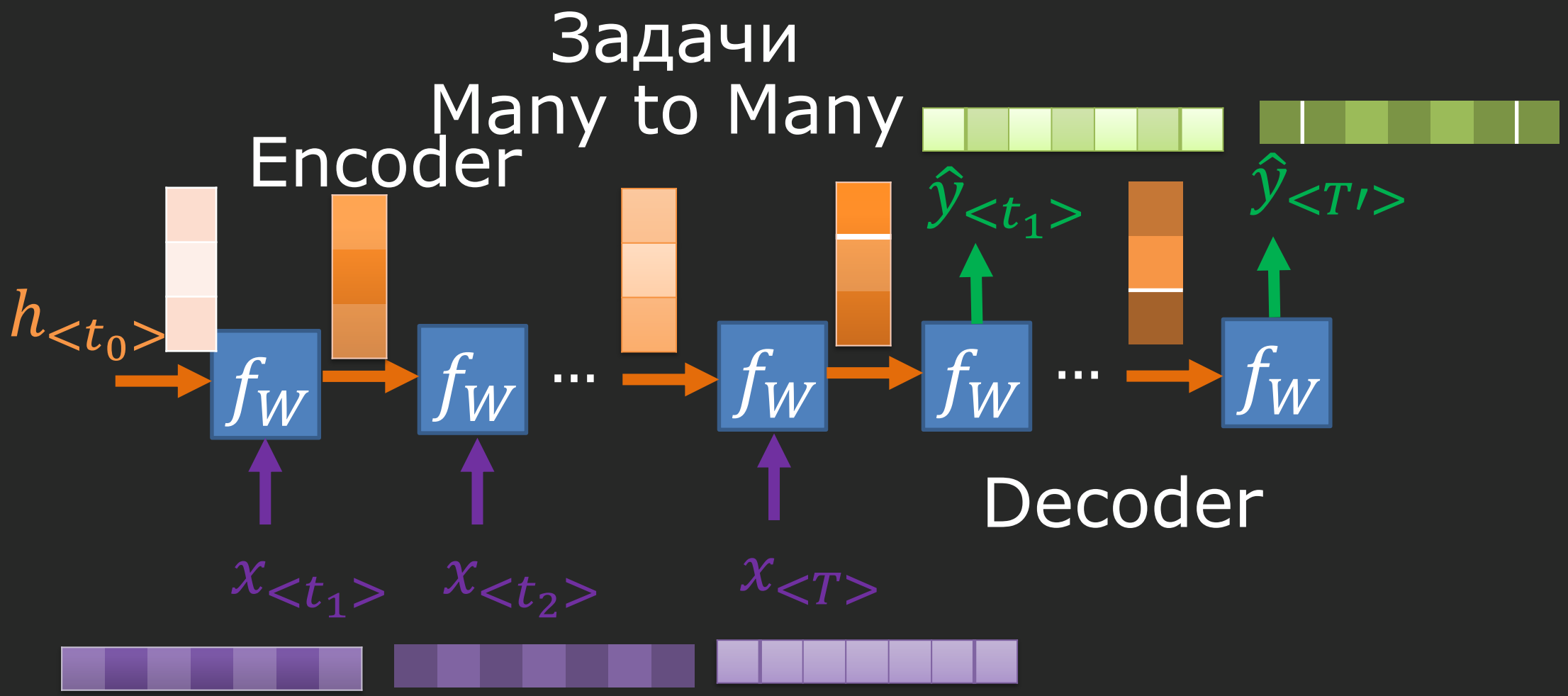
Attention is All You Need

Alignment (Выравнивание)

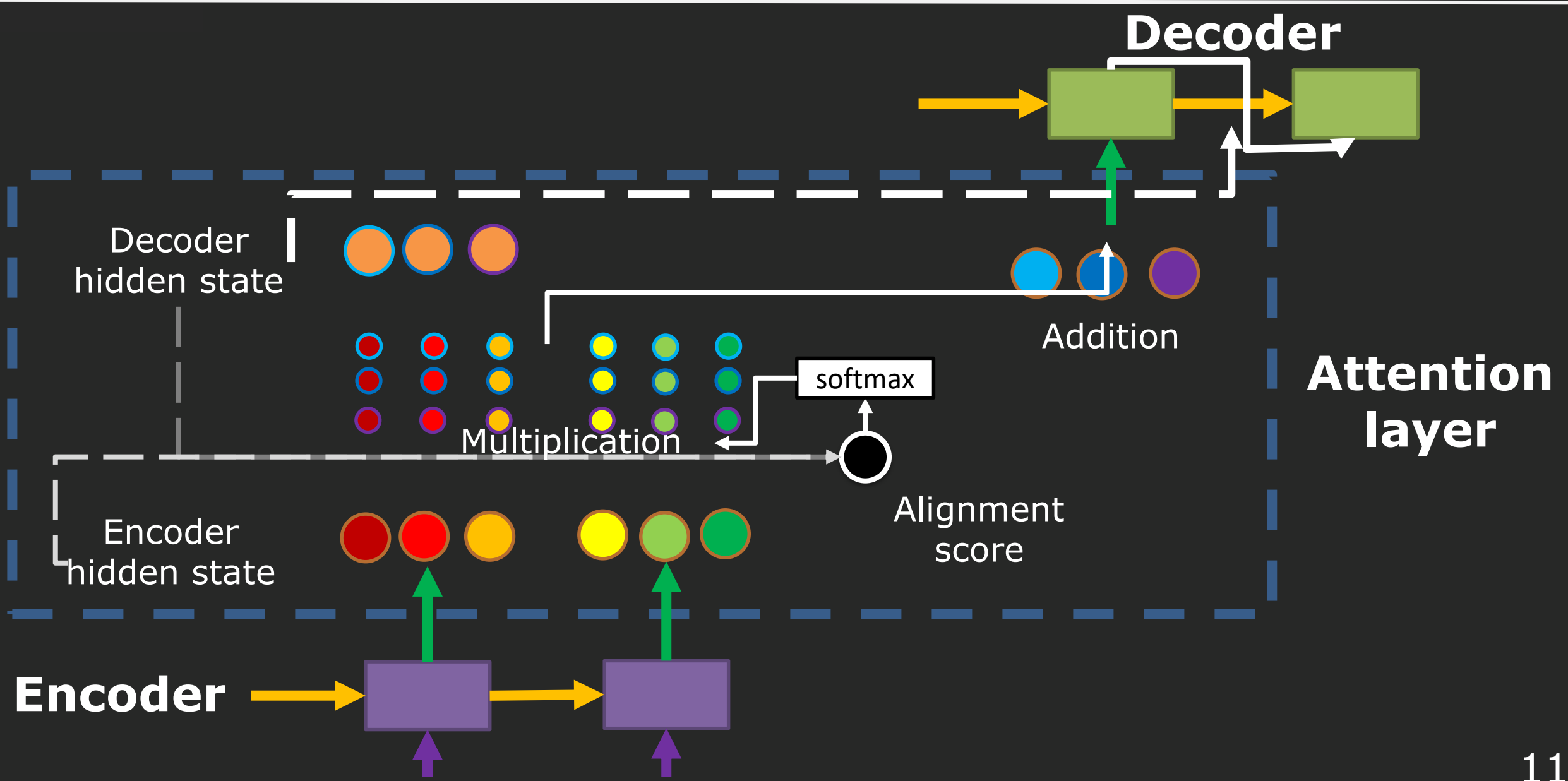


Recurrent Neural Networks

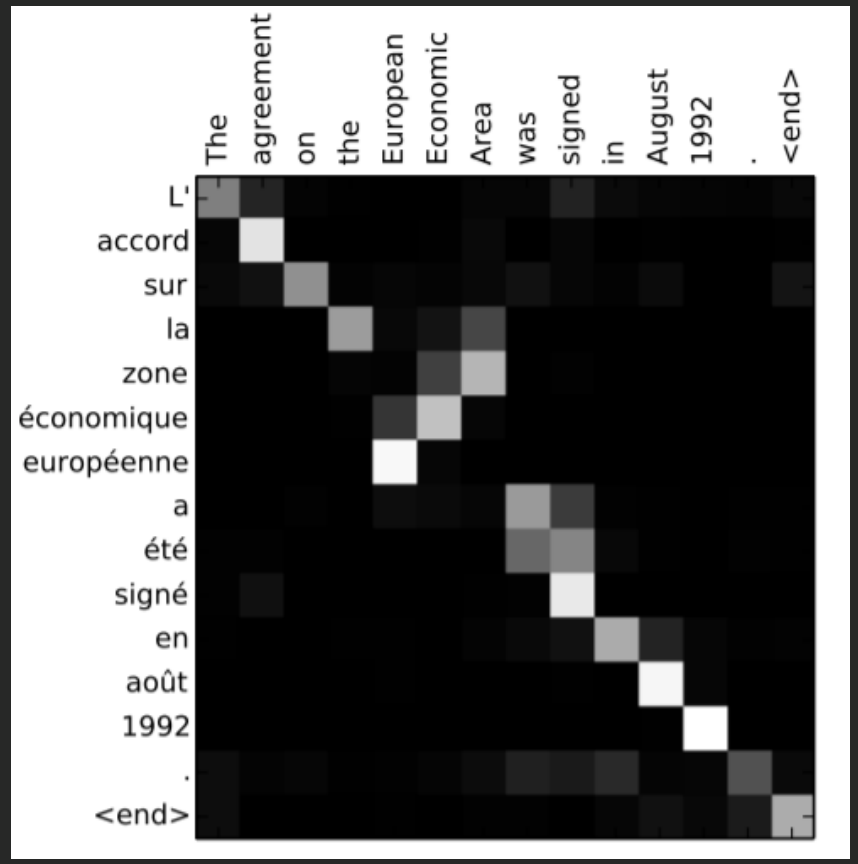
.

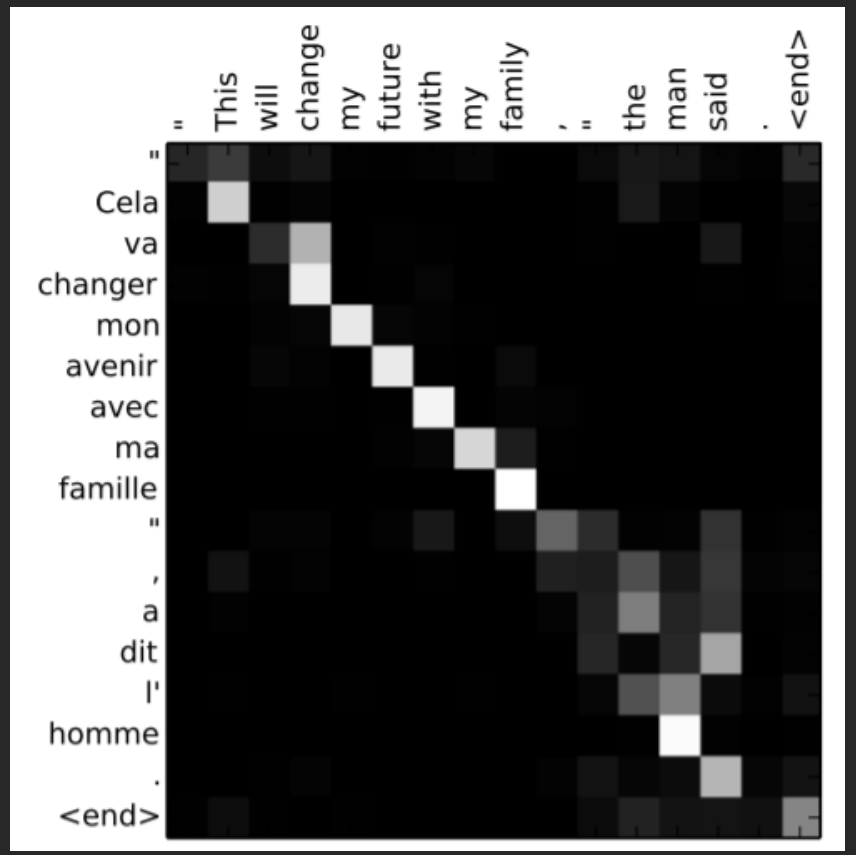


Alignment


Decoder
Decoder hidden state
Addition
Attention layer
softmax
Multiplication
Alignment score
Encoder hidden state
Encoder

Alignment

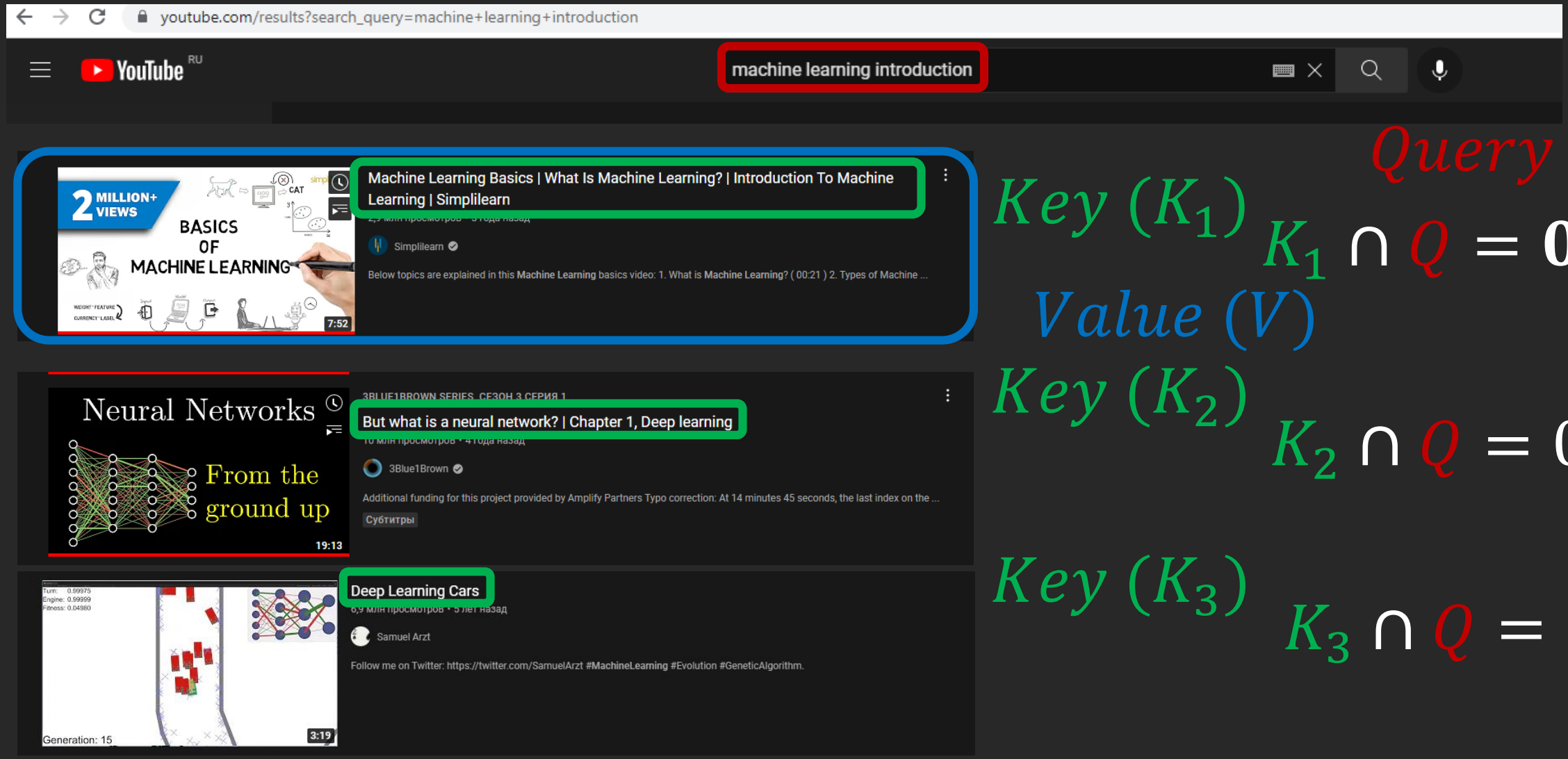




Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).

https://arxiv.org/pdf/1409.0473.pdf

Поиск в Базе Данных



$Query\ (Q)$

$Key\ (K_1)$ $K_1 \cap Q = 0.99$

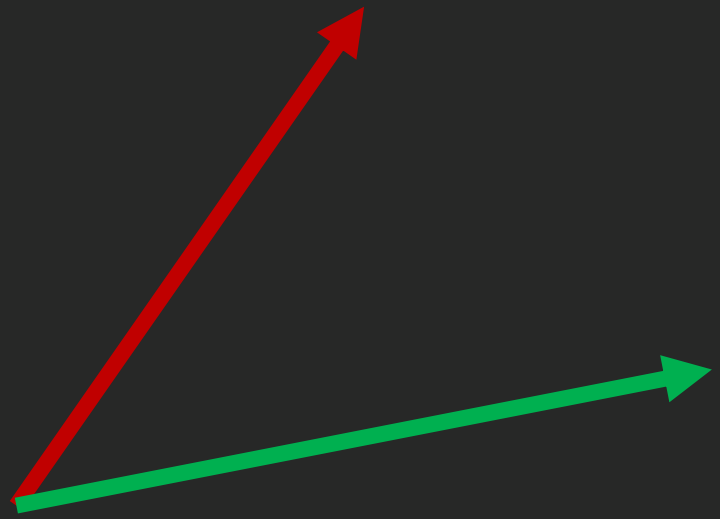
$Value\ (V)$

$Key\ (K_2)$ $K_2 \cap Q = 0.55$

$Key\ (K_3)$ $K_3 \cap Q = 0.25$

Похожесть между векторами

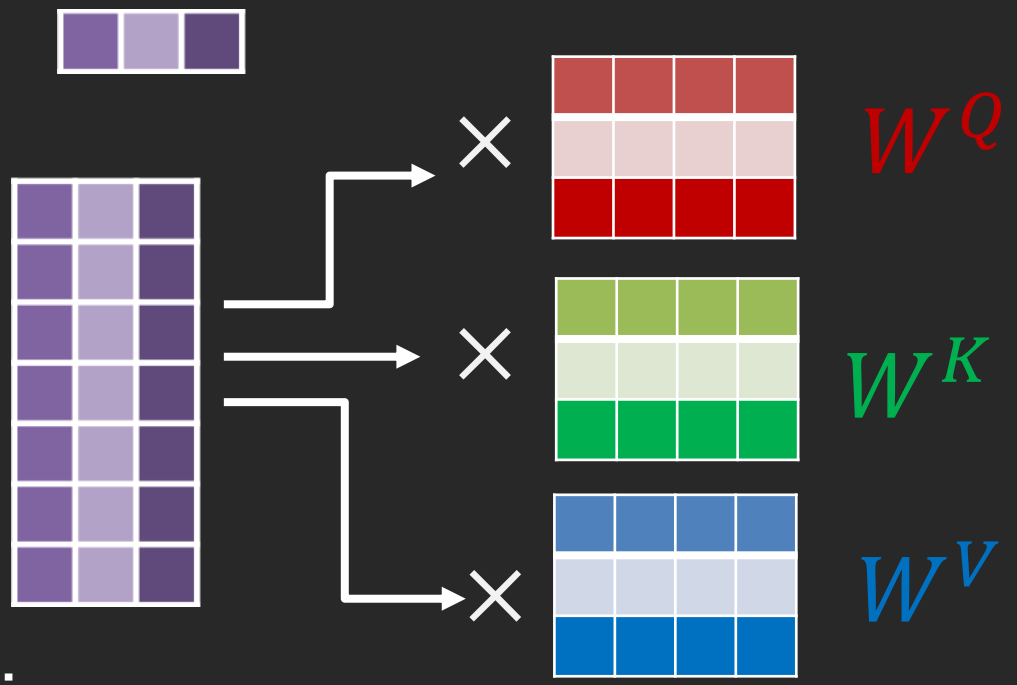
$$\text{Cosine Distance} = \frac{A \cdot B}{|A||B|}$$



$$\text{Similarity}(\text{Query},\text{Key}) = \frac{\text{Query} \cdot \text{Key}}{Scaling}$$

А между чем мы ищем подобие?

Embedding для токена $x_{<t>}$



Embedding для токенов

W^Q

W^K

W^V

Query

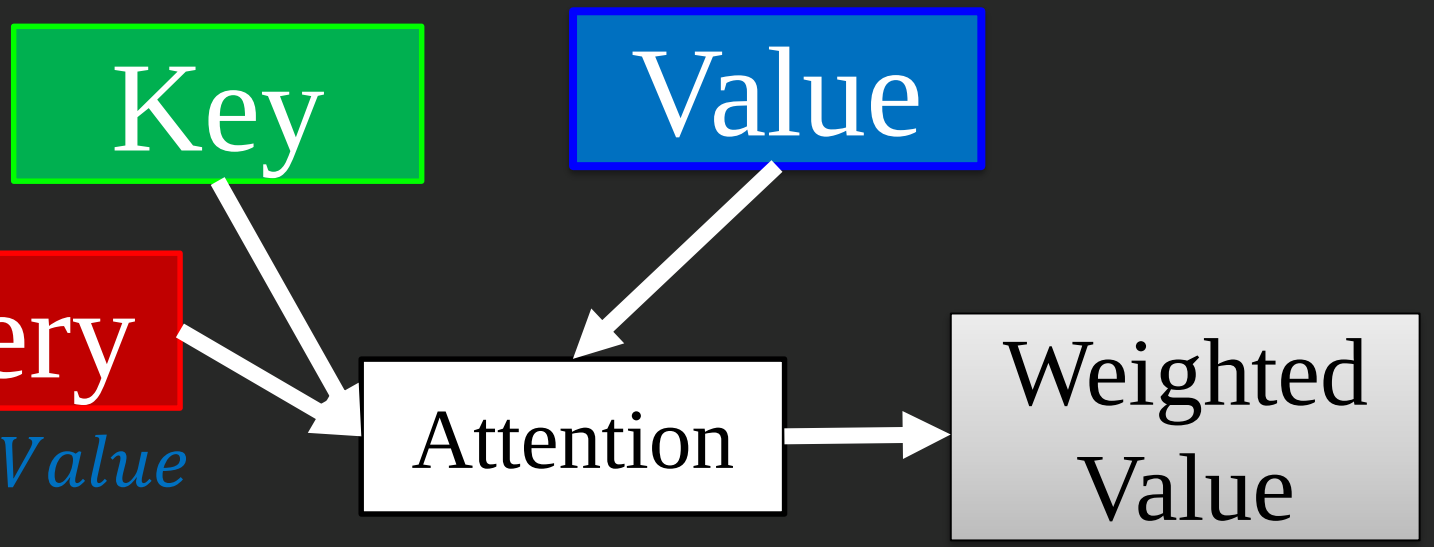
Key

Value

Различные веса

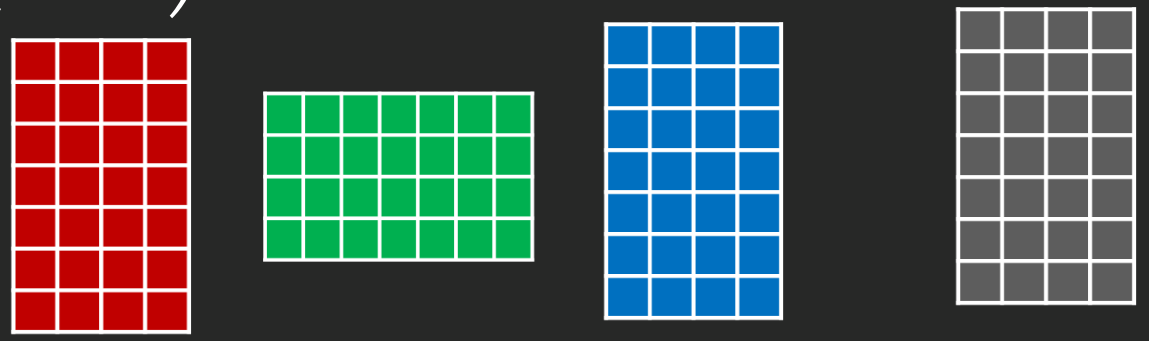
Attention (Внимание)

Key $Value$ $Query$

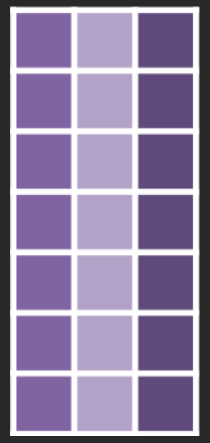


$$softmax(Query * Key^T) * Value$$

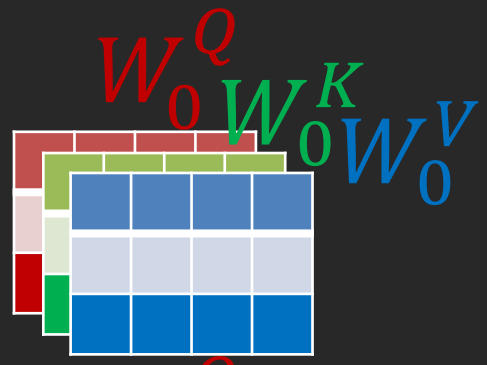
$$softmax\left(\frac{Query * Key^T}{\sqrt{d_K}}\right) * Value$$

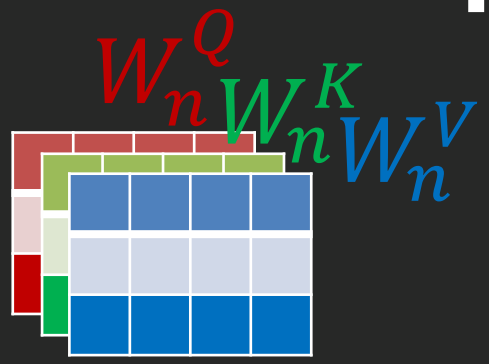


Multi-Head Attention

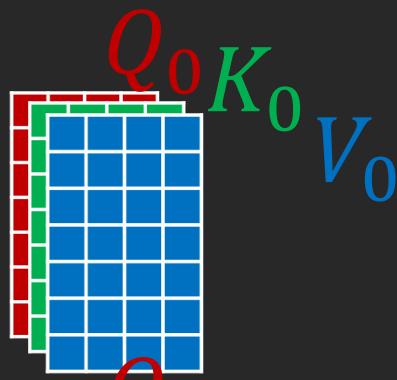


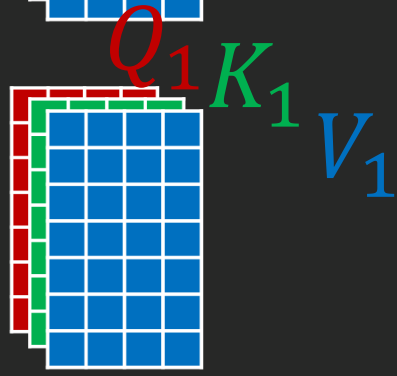
Embedding для токенов

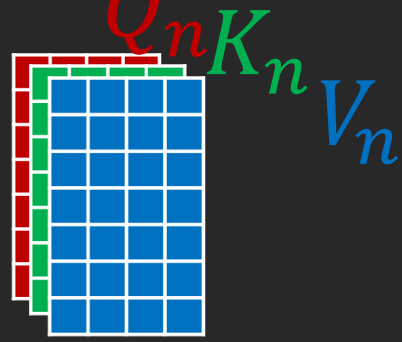


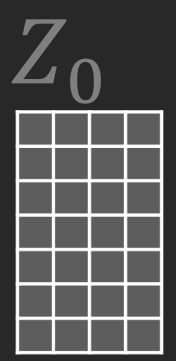


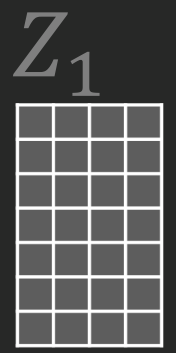
...

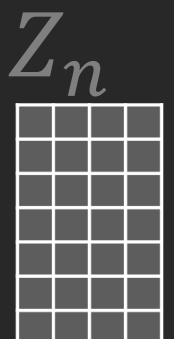


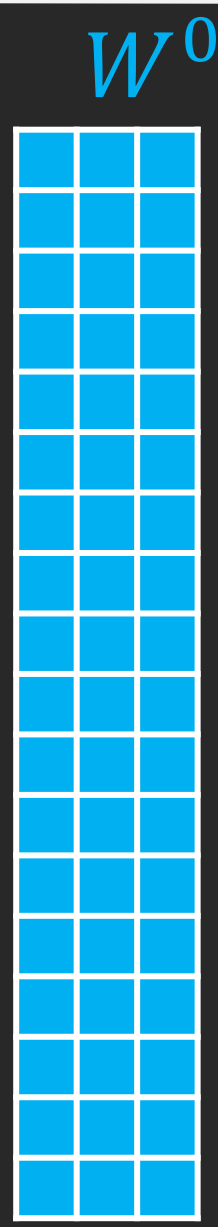


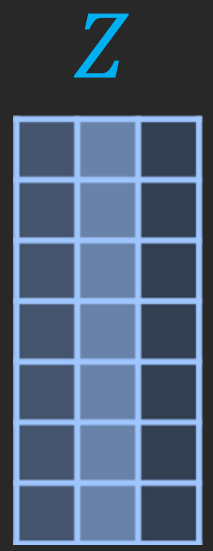






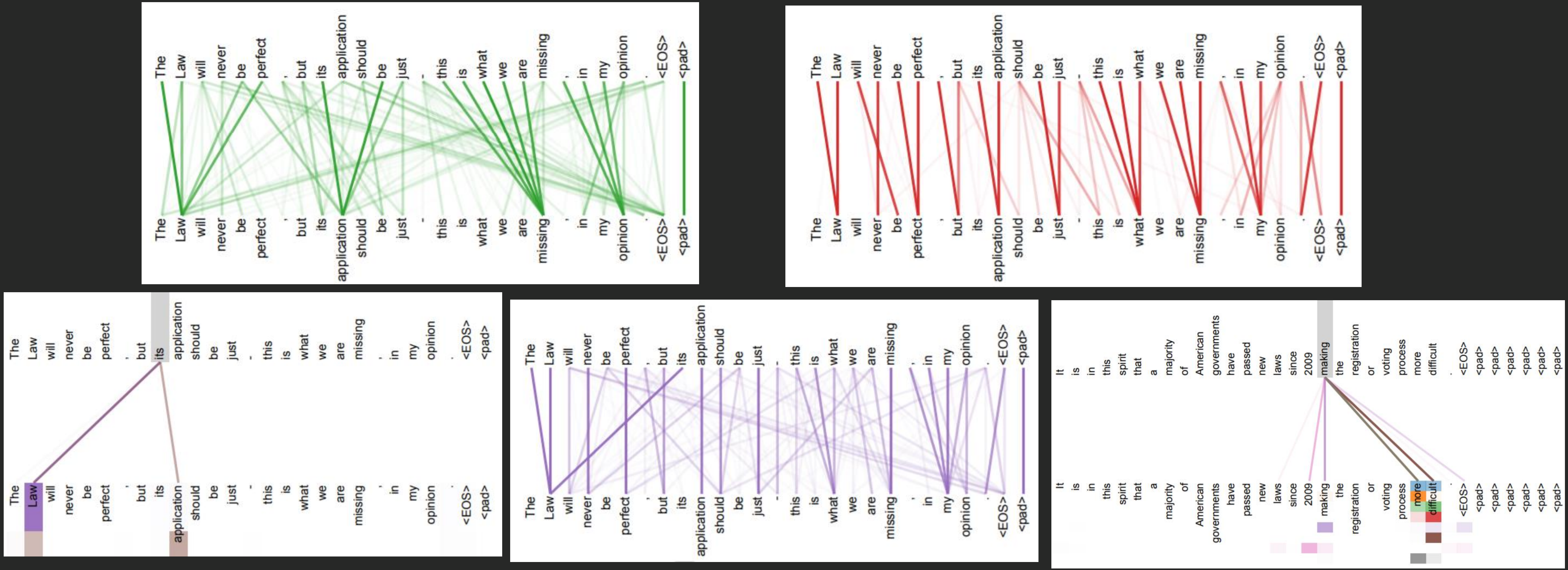






Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Multi-Head Attention

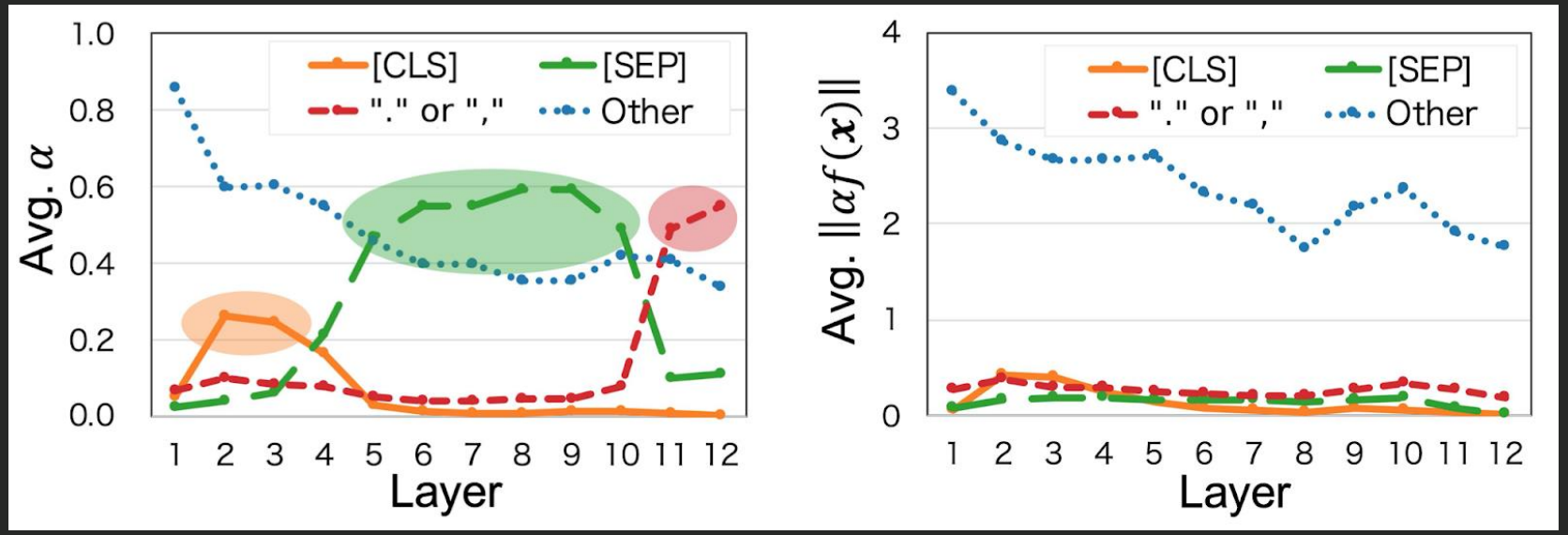


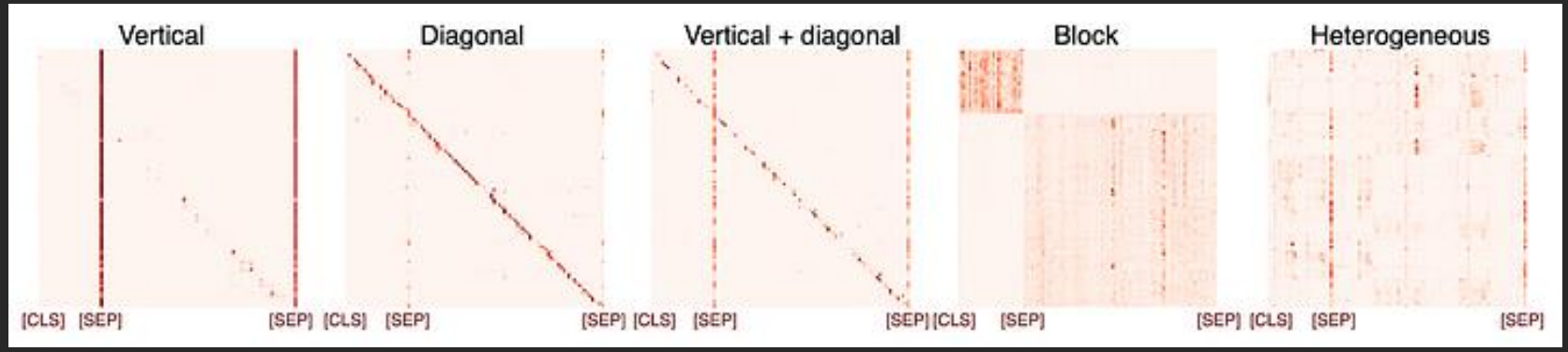
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

https://arxiv.org/pdf/1706.03762.pdf

Multi-Head Attention

.





https://arxiv.org/pdf/2004.10102.pdf

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Про Токенизацию (опять)

Внимание

Трансформеры

Transformer Encoder Decoder

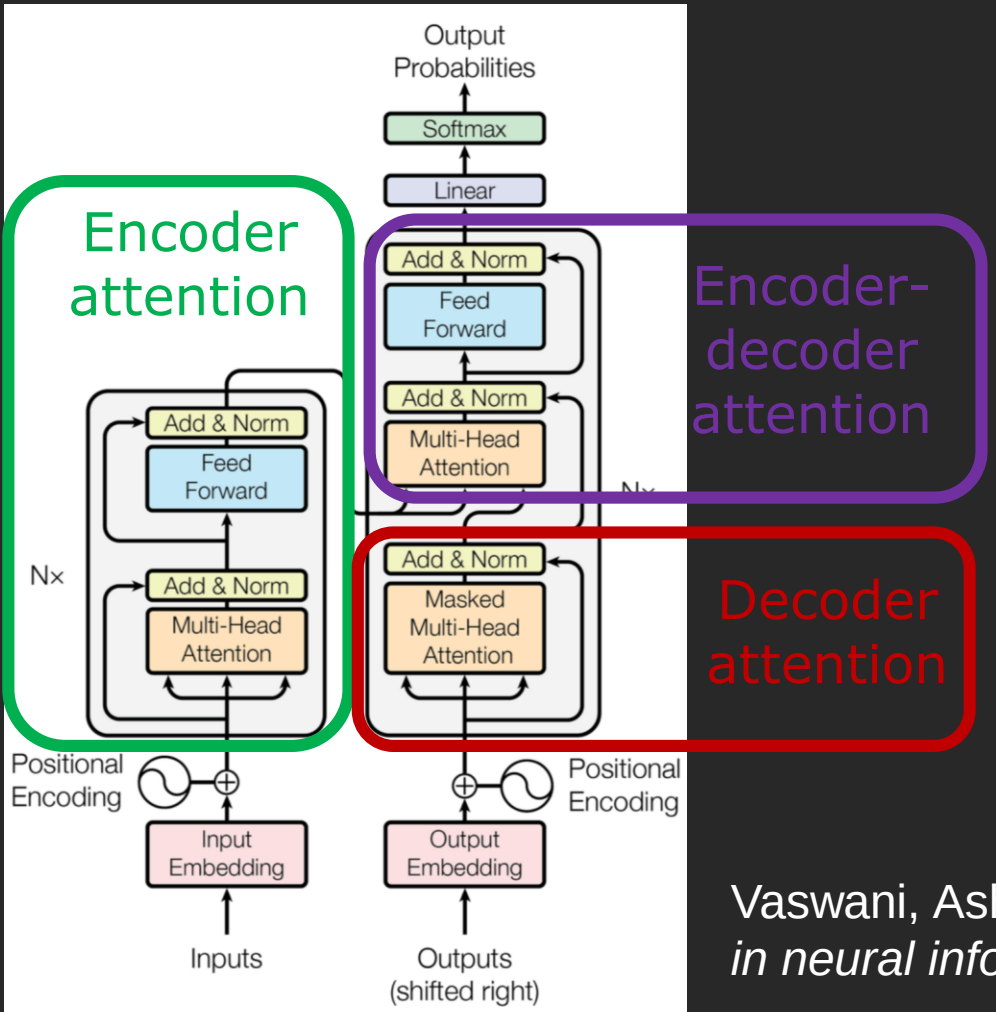


Figure 1: The Transformer - model architecture.

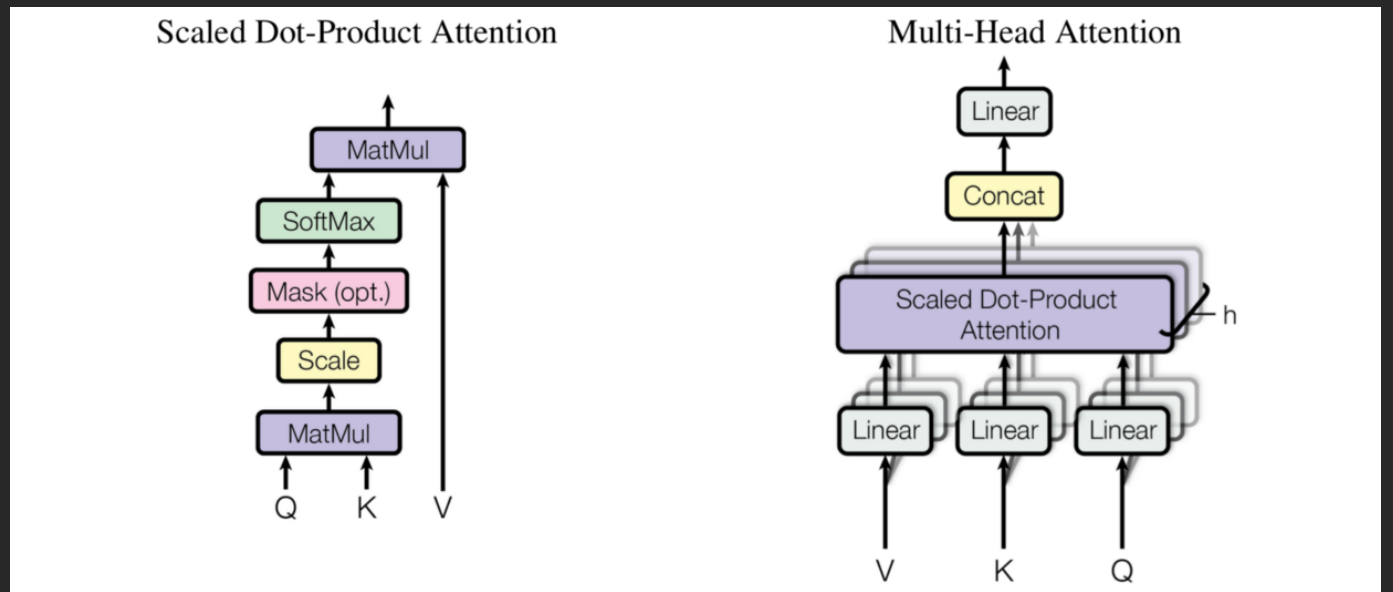


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

https://arxiv.org/pdf/1706.03762.pdf

Transformer Encoder Decoder

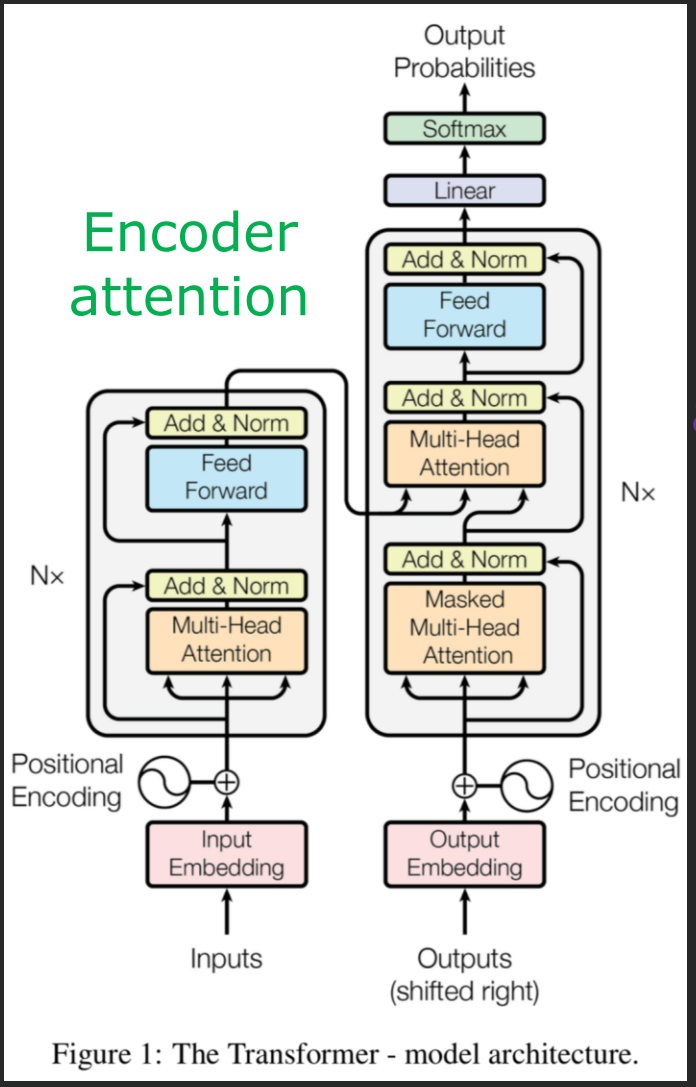


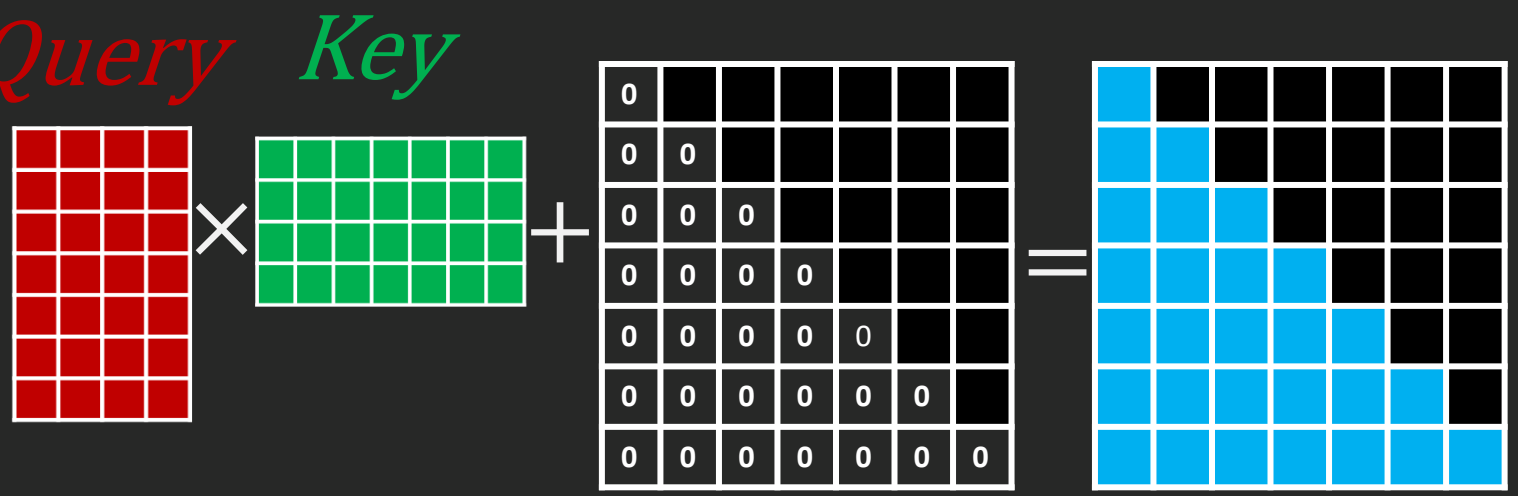
Figure 1: The Transformer - model architecture.

Encoder attention

Encoder-decoder attention

Decoder attention

Masked Attention



https://arxiv.org/pdf/1706.03762.pdf

Transformer Encoder Decoder

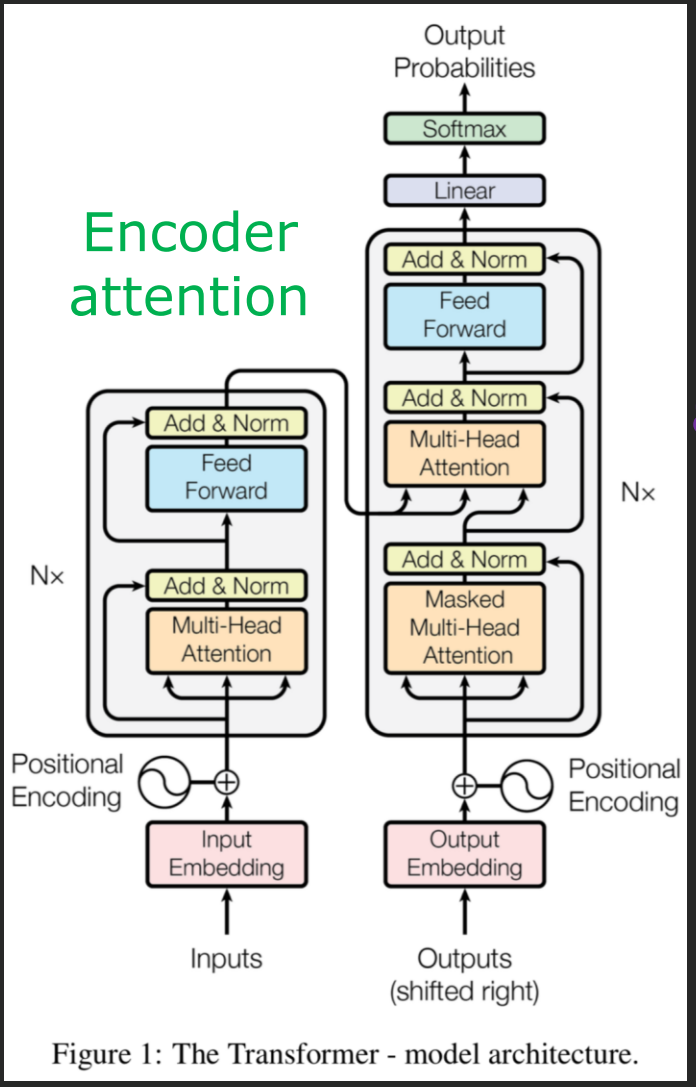


Figure 1: The Transformer - model architecture.

Encoder attention

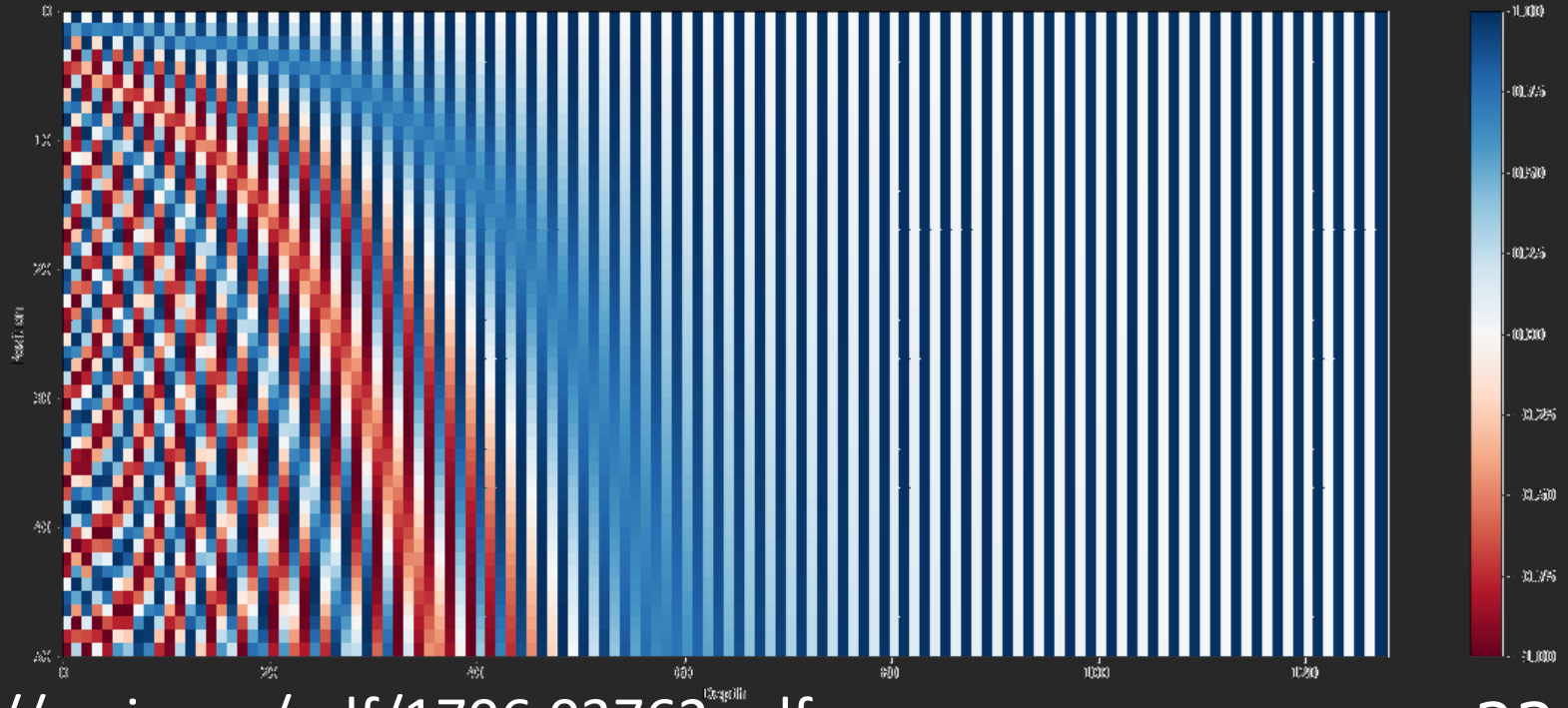
Encoder-decoder attention

Decoder attention

Positional Encoding

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



https://arxiv.org/pdf/1706.03762.pdf

Transformer Encoder Decoder

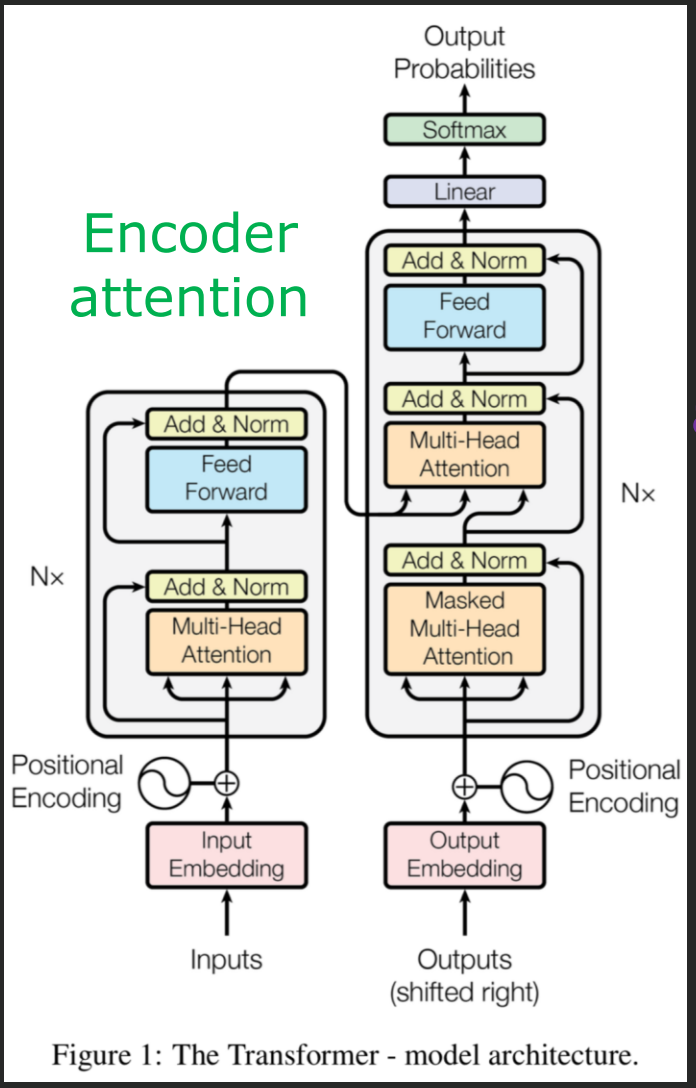


Figure 1: The Transformer - model architecture.

Encoder attention

Encoder-decoder attention

Decoder attention

‘Классическая’ архитектура

$d_{model} = 512$

Encoder $N = 6$ Слоев

Decoder $N = 6$ Слоев

Attention heads $h = 8$

$d_{key} = d_{value} = d_{query} = 64$

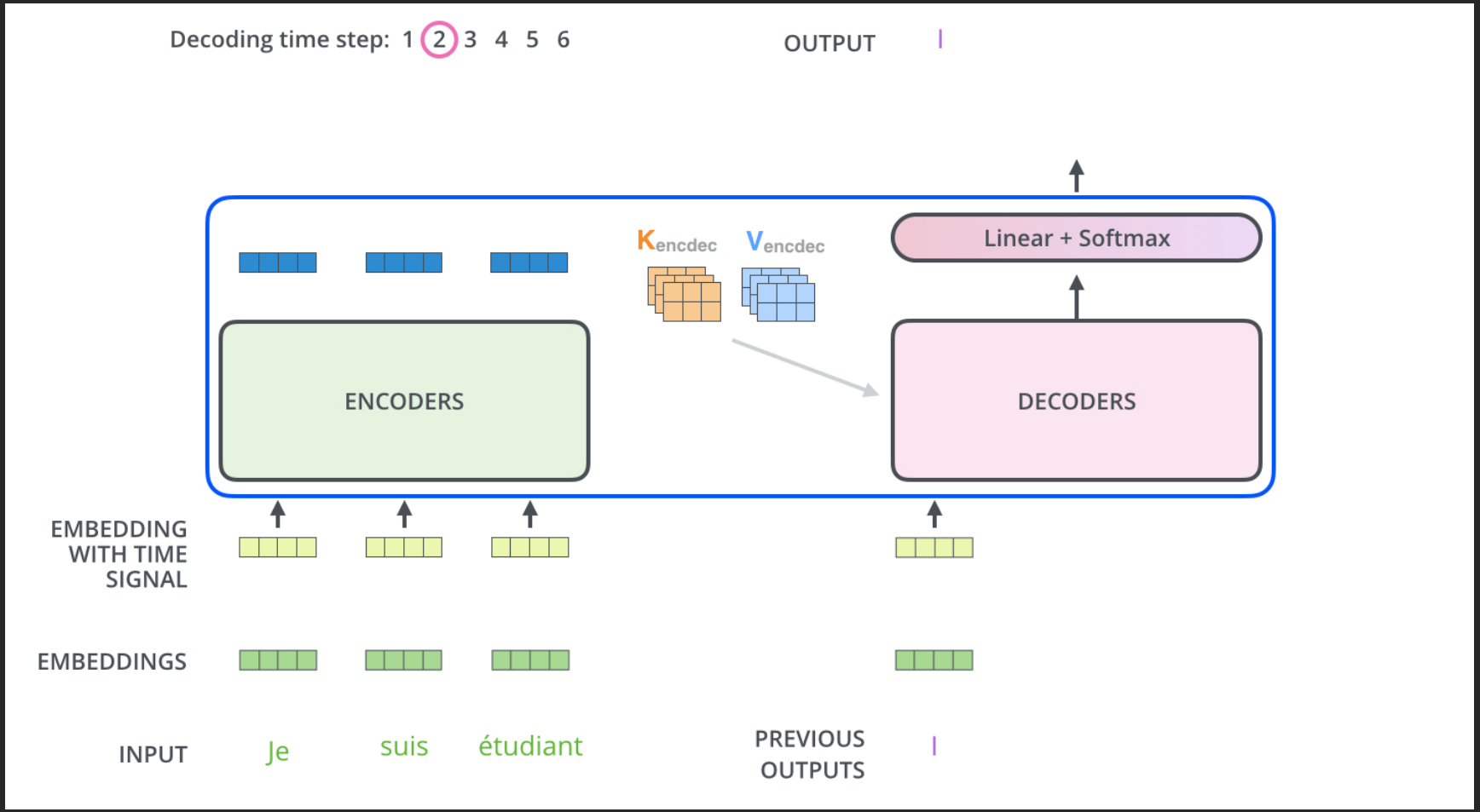
65 млн. параметров

Применение

Google Translate

https://arxiv.org/pdf/1706.03762.pdf

Transformer Encoder Decoder .



https://jalammar.github.io/illustrated-transformer/

Содержание

Transformer Encoder

BERT
Bidirectional Encoder Representations from Transformers

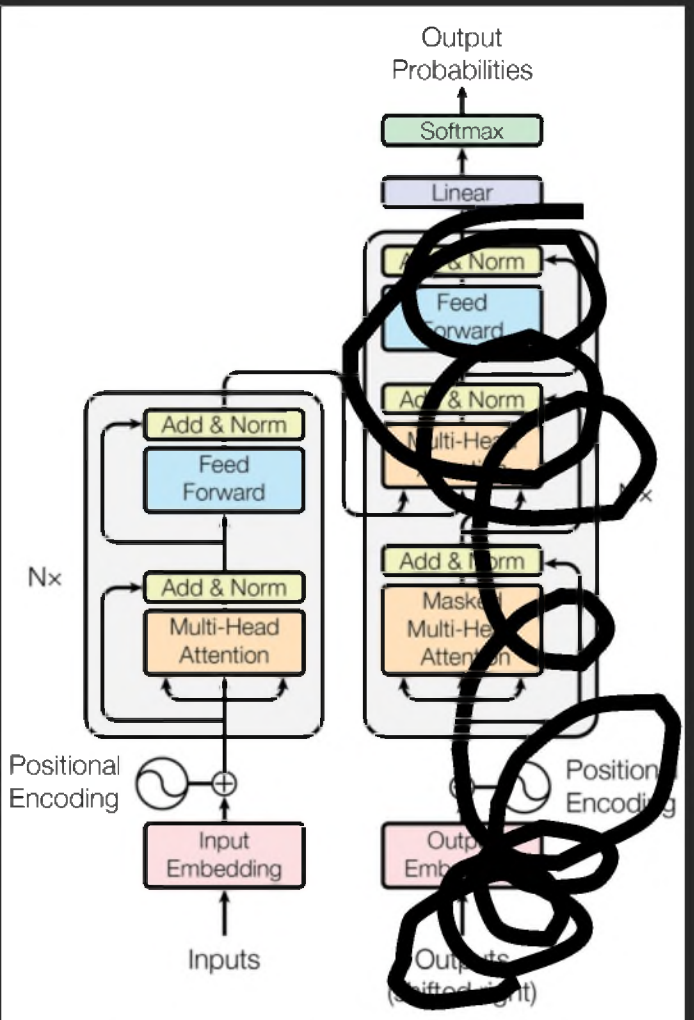
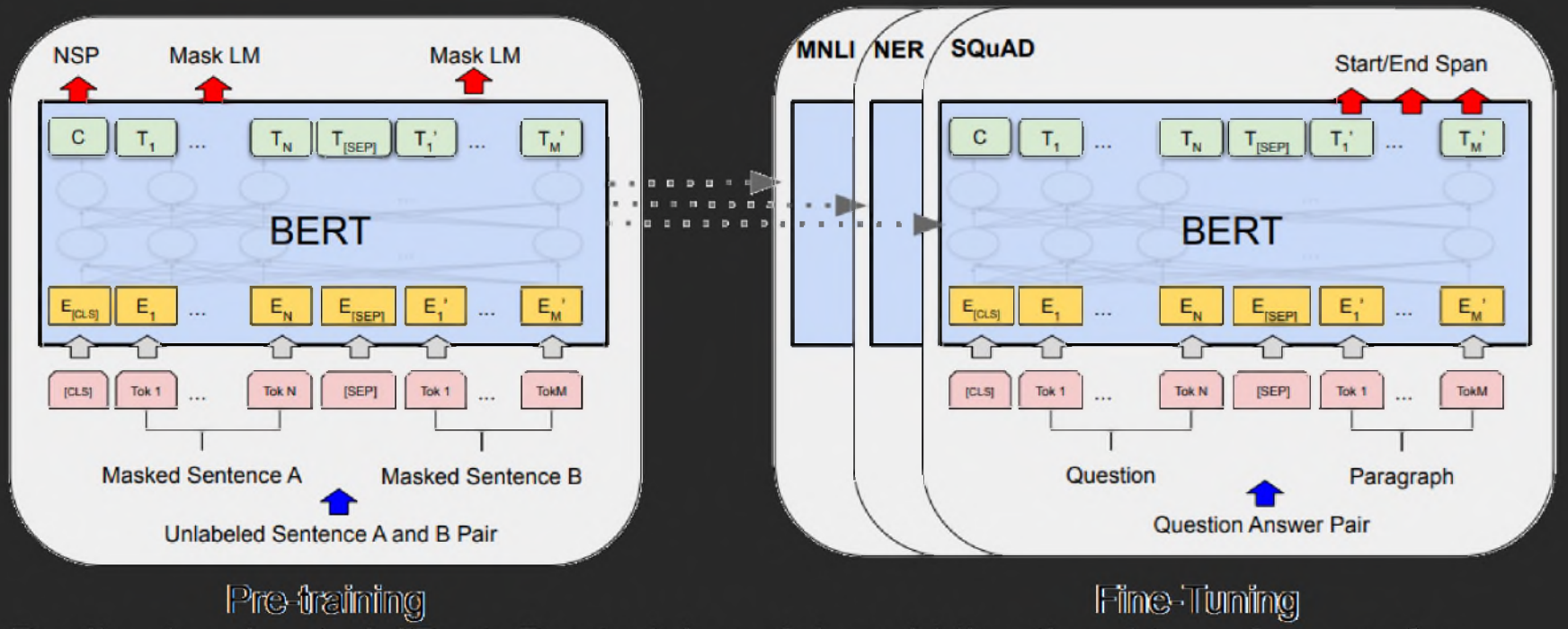


Figure 1: The Transformer - model architecture.



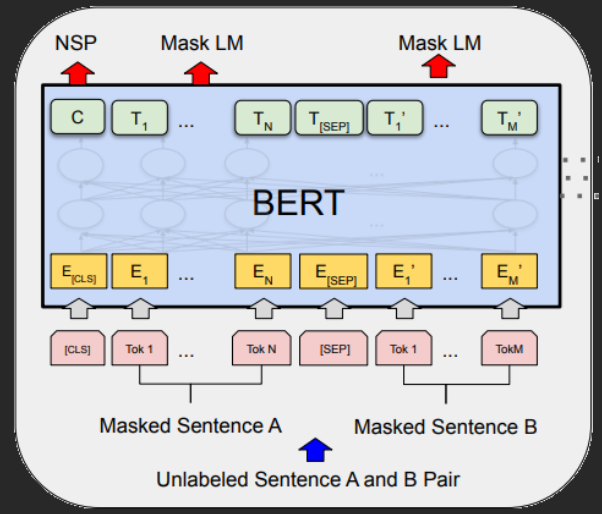
Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Transformer Encoder

BERT

Задачи для Пред-обучения

- **Маскированная языковая модель (masked LM)**
  Процедура маскирования
  - для 15% токенов
    - 80%: менялись на токен [MASK]
    - 10%: менялись на случайный токен
    - 10%: ничего не делали
- **Предсказание следующего текста**



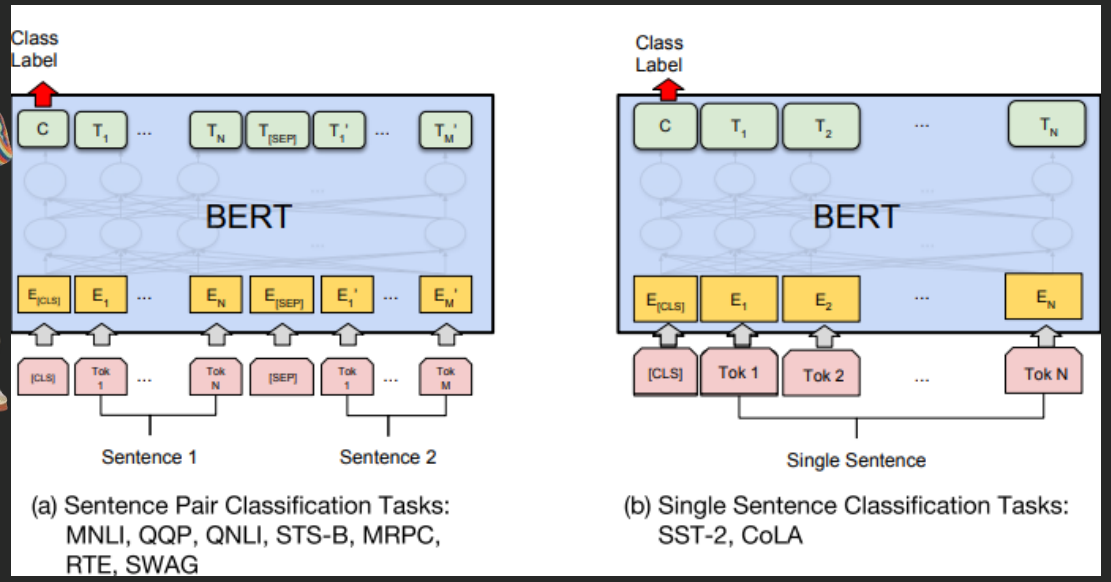
Pre-training

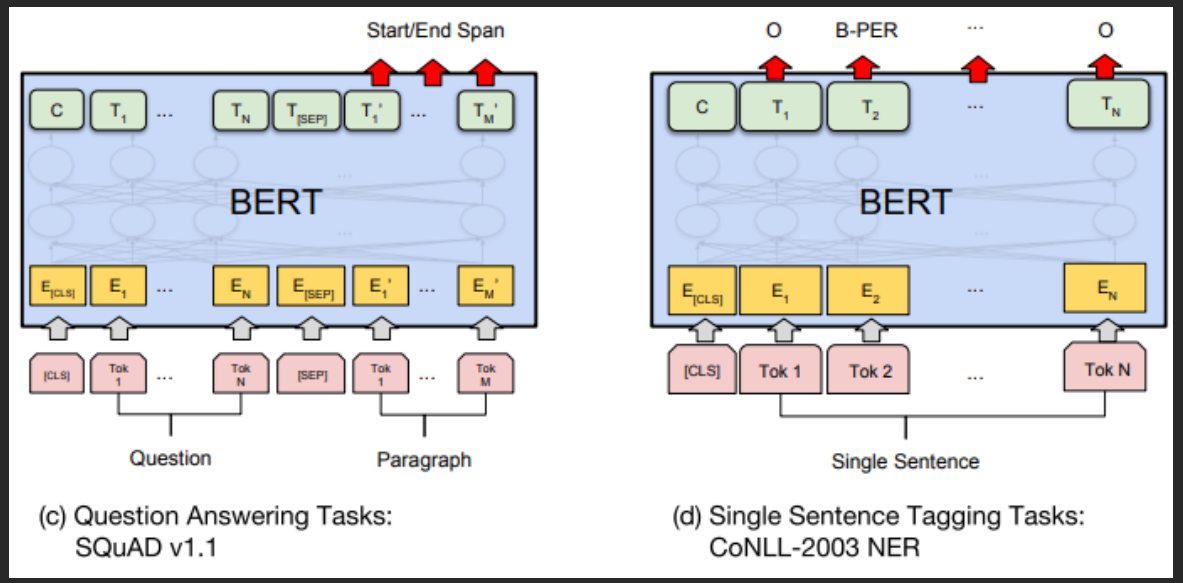
Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Transformer Encoder

BERT

До-обучение (Fine-Tuning)





Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Transformer Encoder

BERT

$BERT_{base}$: $L=12, H=768, A=12$

Всего параметров: 110 млн.

$BERT_{large}$: $L=24, H=1024, A=16$

Всего параметров: 340 млн.

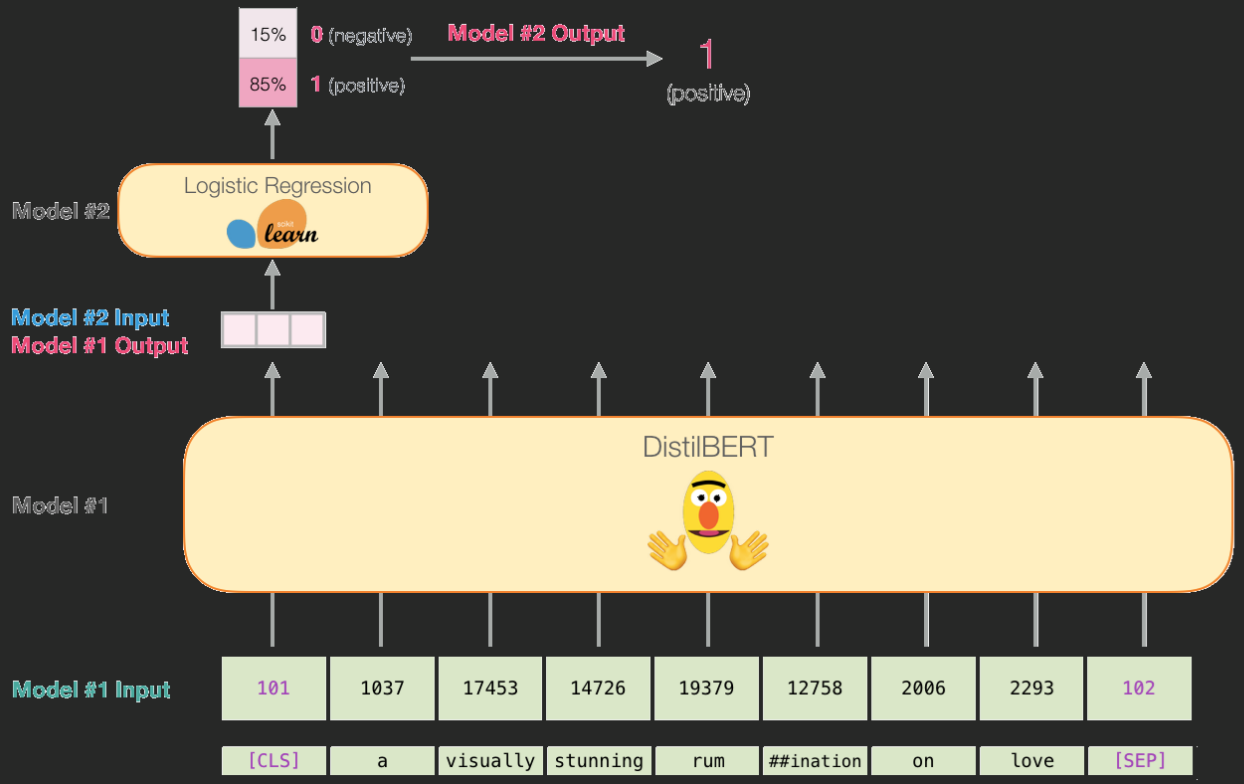
L – число Attention Layers
H – размерность (attention weights)
A – количество Attention heads

https://github.com/google-research/bert

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Transformer Encoder

BERT



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Содержание

Transformer Decoder

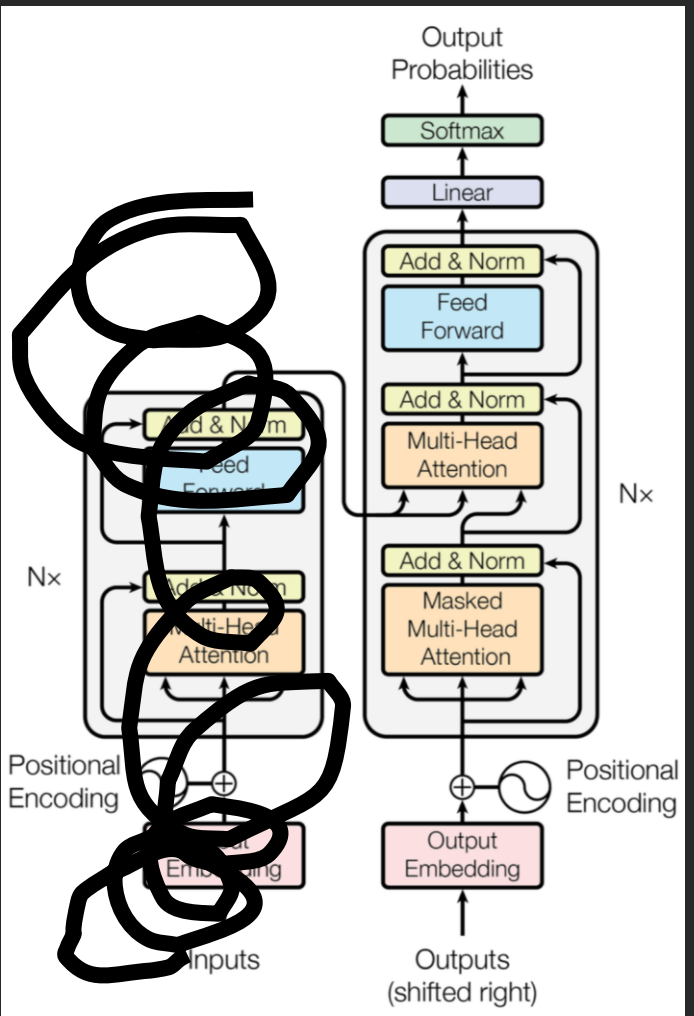


Figure 1: The Transformer - model architecture.

GPT
Generative Pre-Training

‘…просто предсказывает следующее слово…’

Год	Модель	Млрд Параметров	Контекст, Токены	Размер данных, Гб
2018	GPT	0.117	512	4
2019	GPT-2	0.345/0.762/1.542	1024	40
2020	GPT-3	175	2048	570

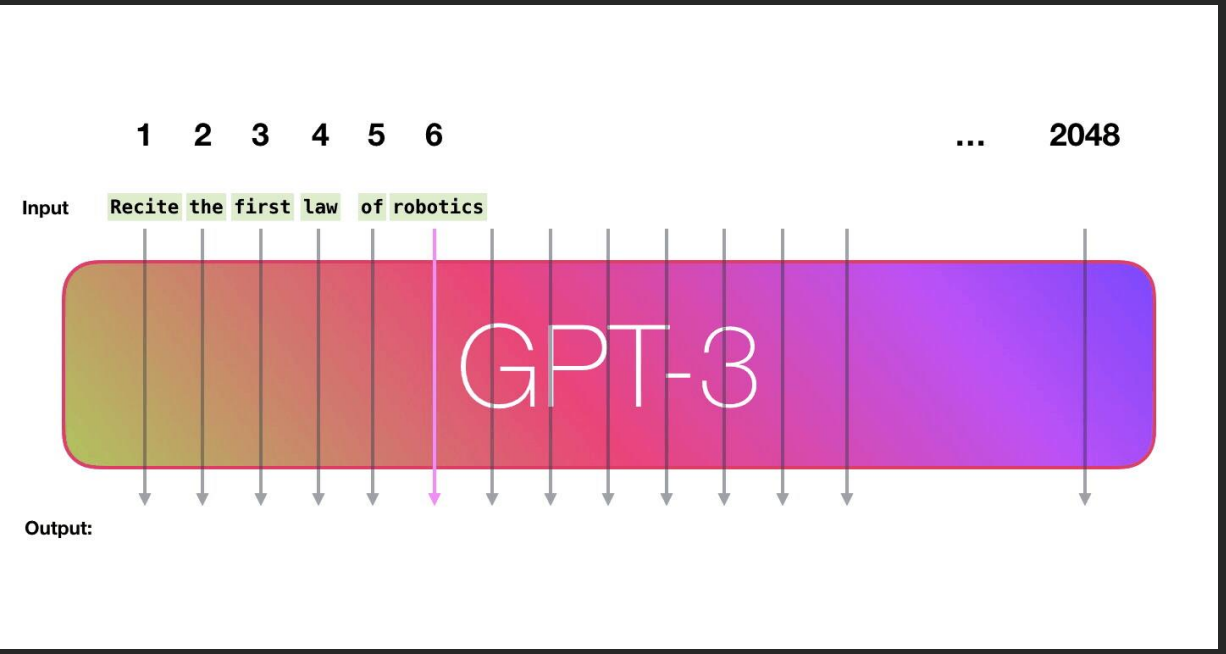
https://openai.com/blog/language-unsupervised/

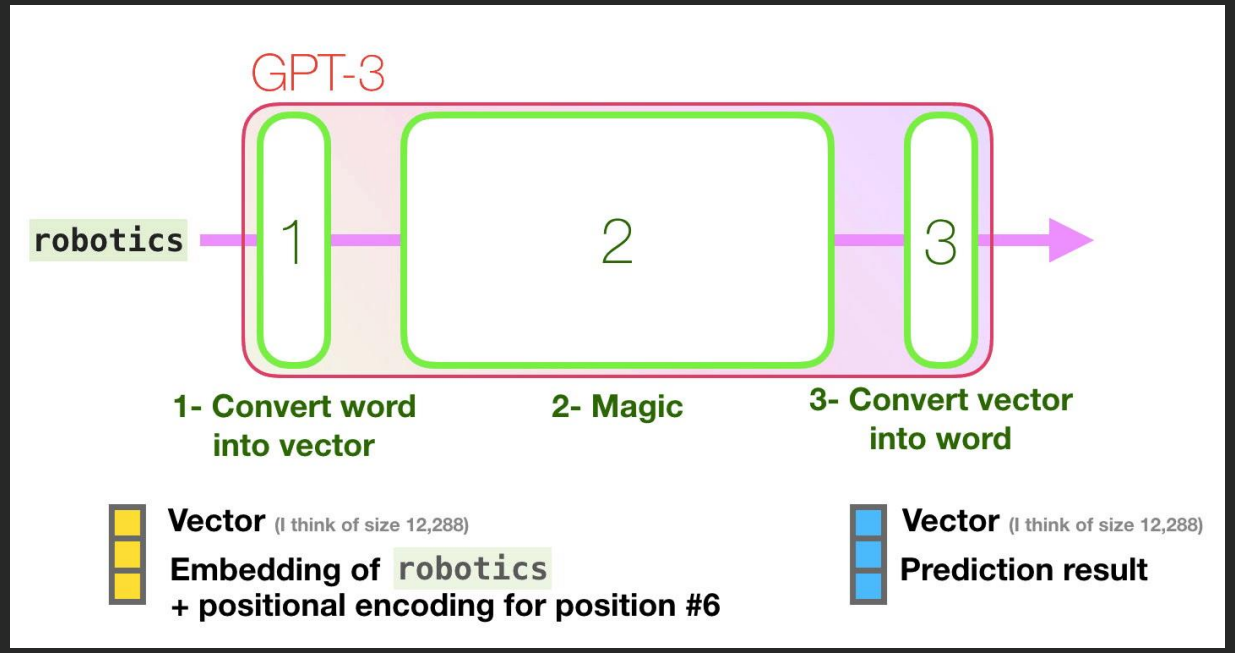
https://openai.com/blog/better-language-models/

Transformer Decoder

.

GPT





https://jalammar.github.io/how-gpt3-works-visualizations-animations/

Transformer Decoder

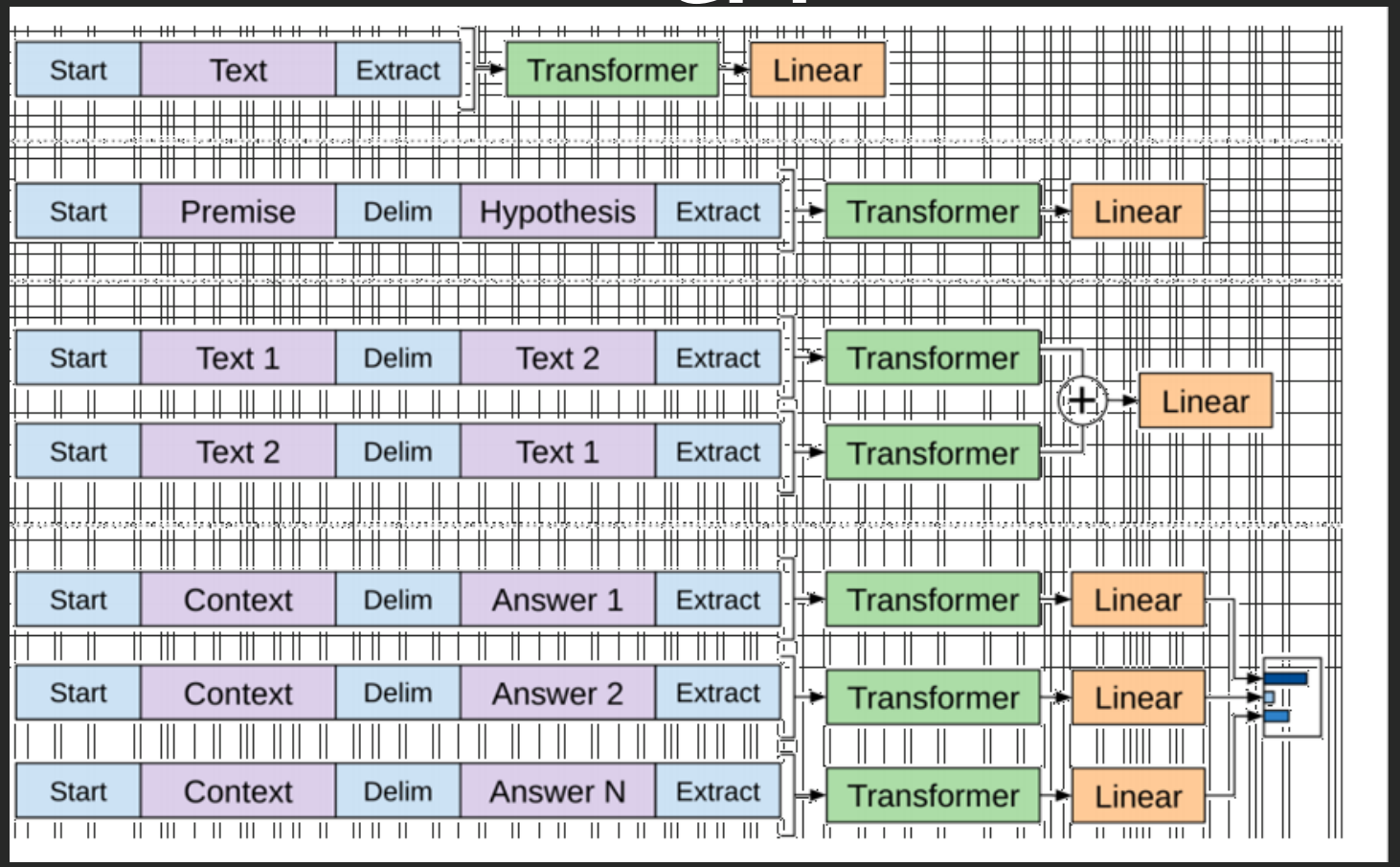
GPT

Классификация

Логическое следствие

Сходство

Множественный выбор



https://jalammar.github.io/illustrated-bert/

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

.

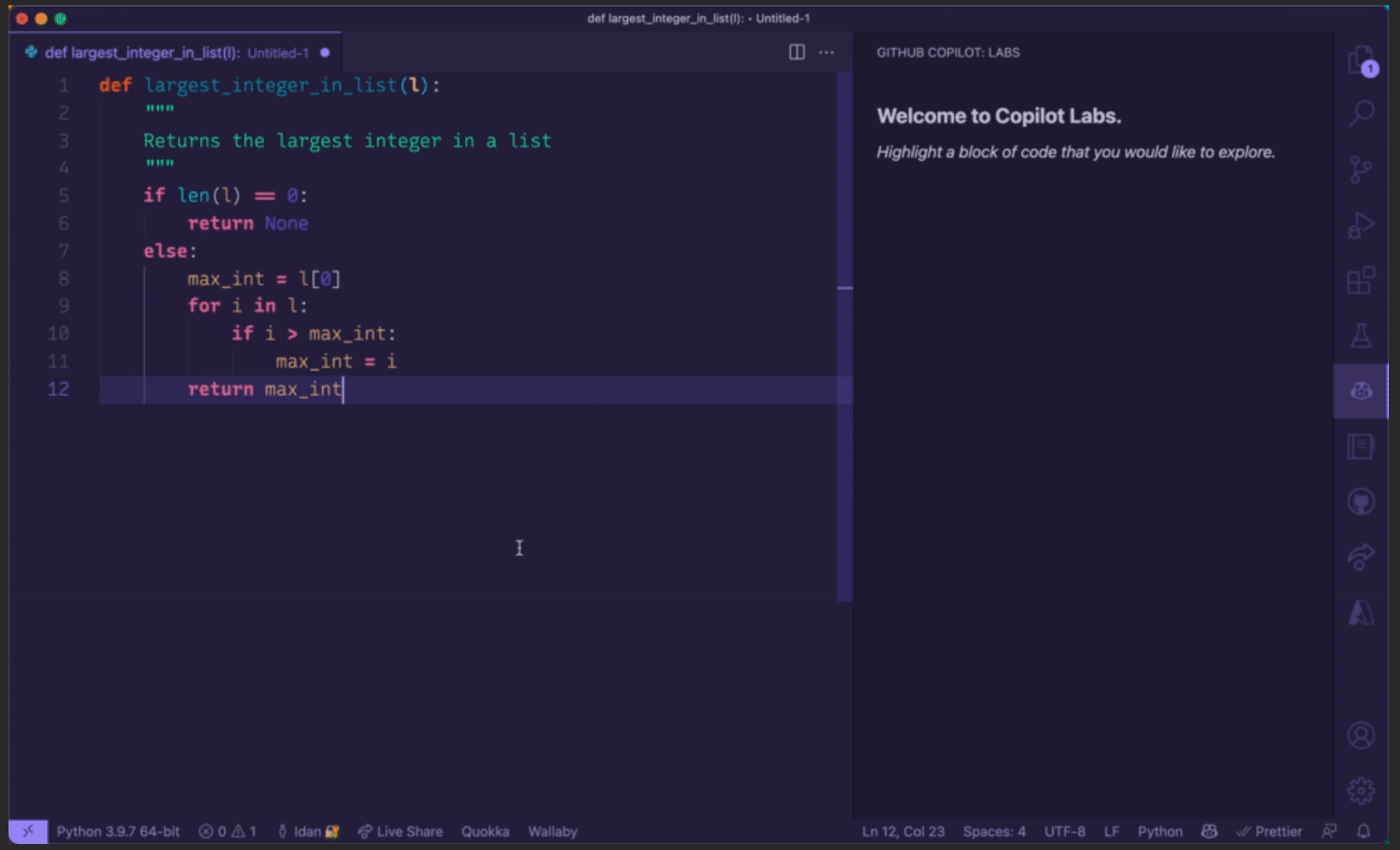
Про Токенизацию (опять)

Внимание

Трансформеры

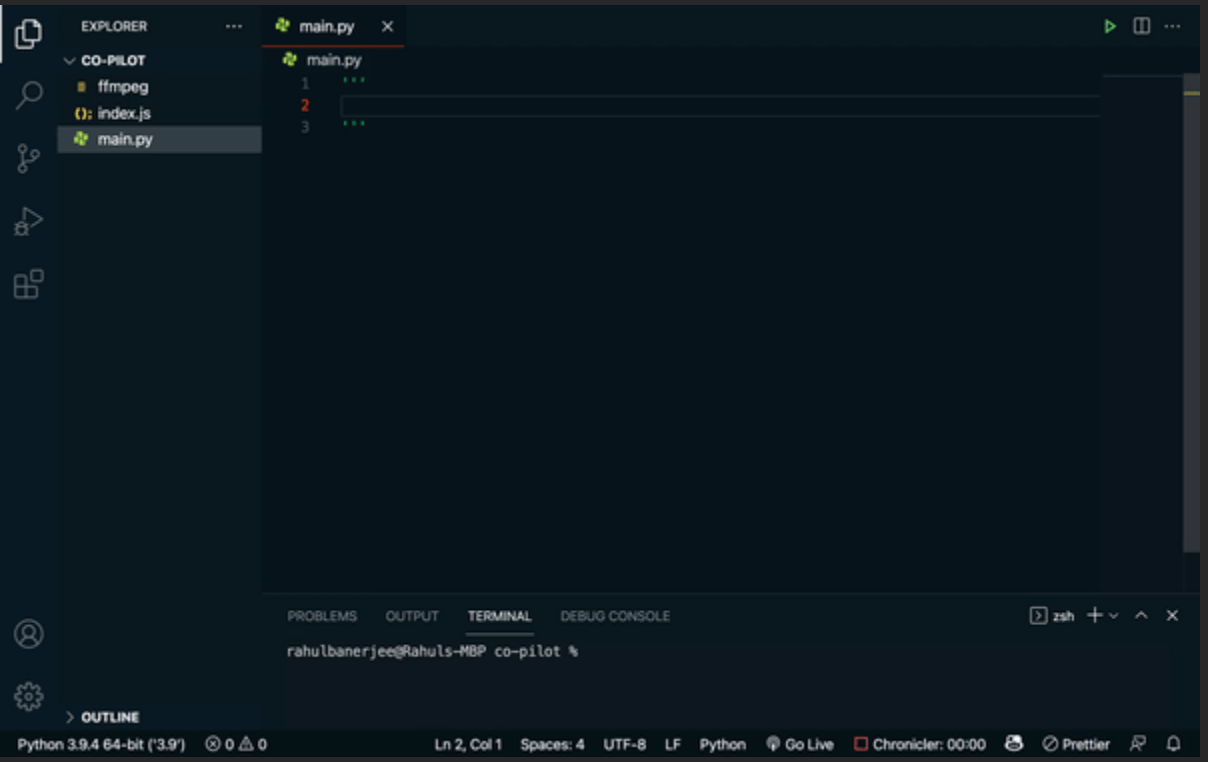
В продакшн

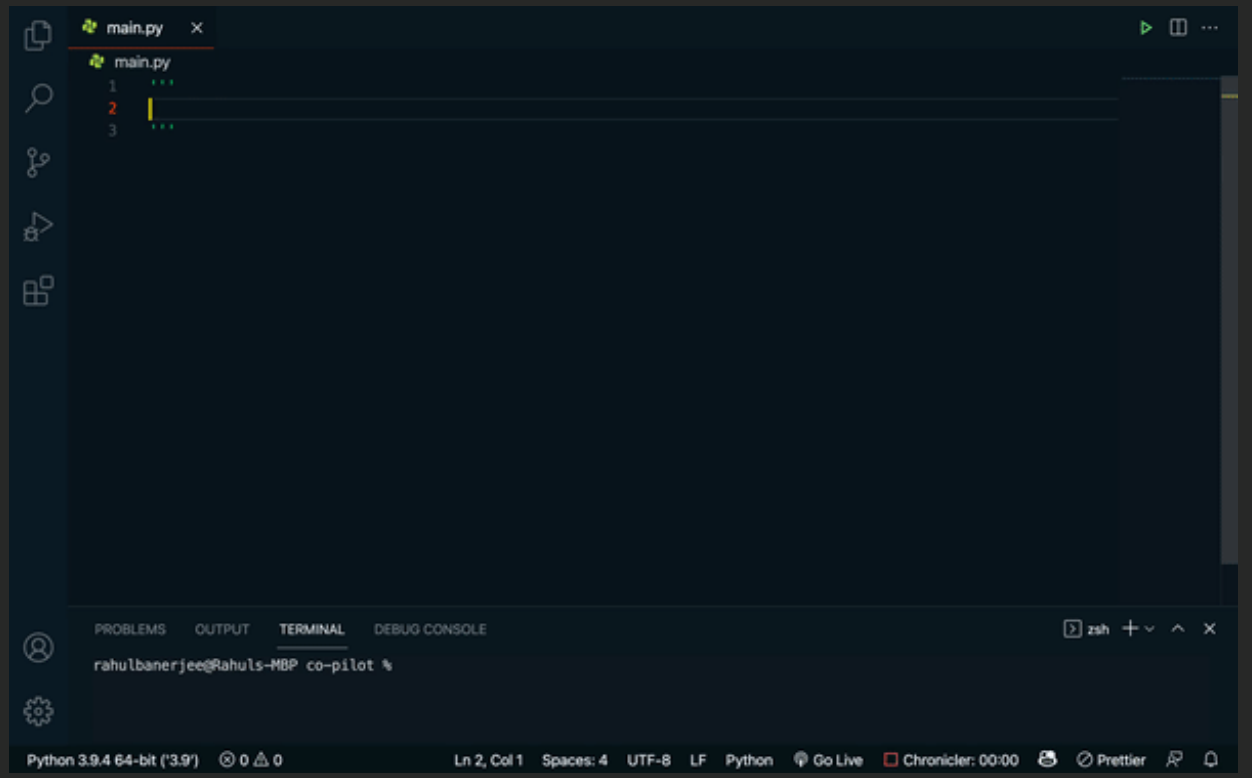
Copilot



https://github.com/github/feedback/discussions/8308

Copilot





https://devsday.ru/blog/details/49410

Содержание

.

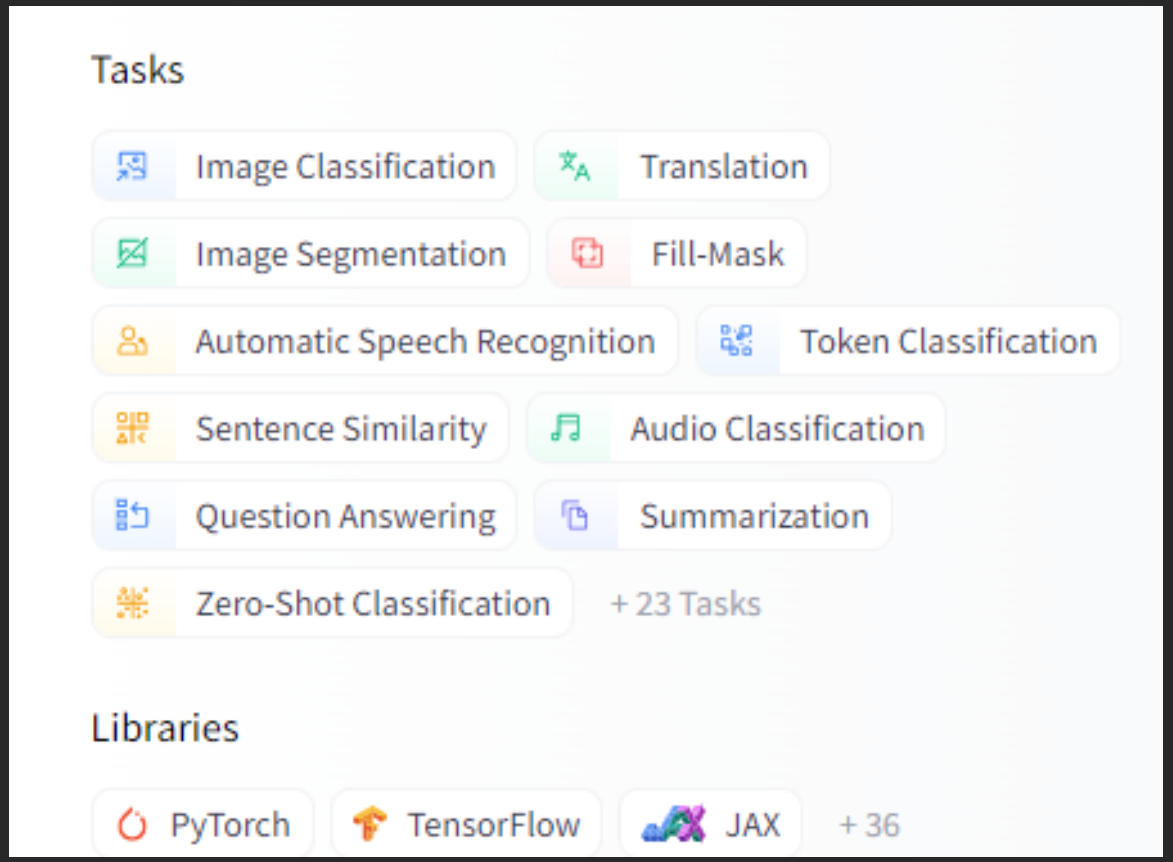
Про Токенизацию (опять)

Внимание

Трансформеры

А можно нам с этим поиграть?

Hugging Face

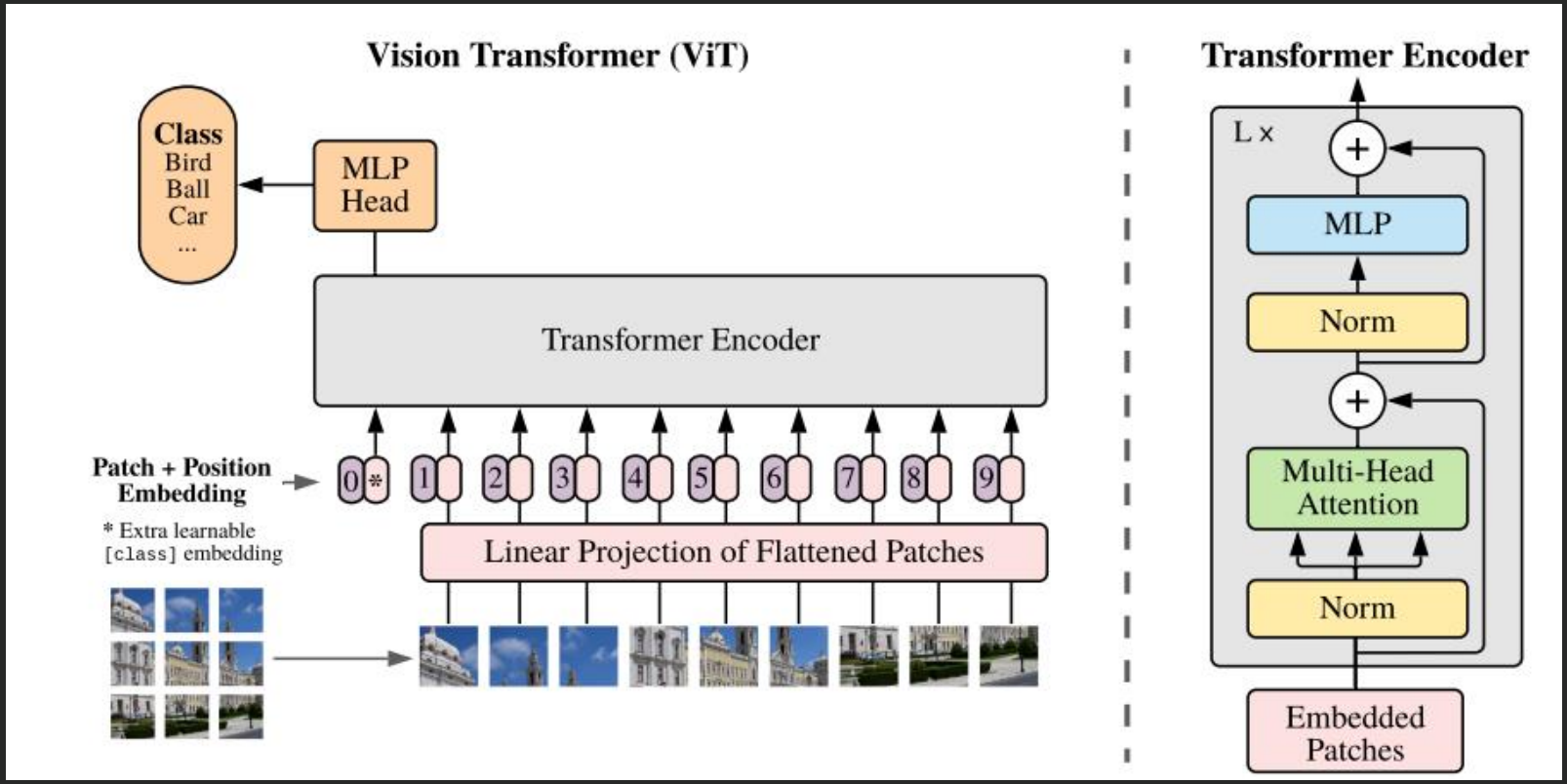


https://huggingface.co/models?sort=downloads

Содержание

.

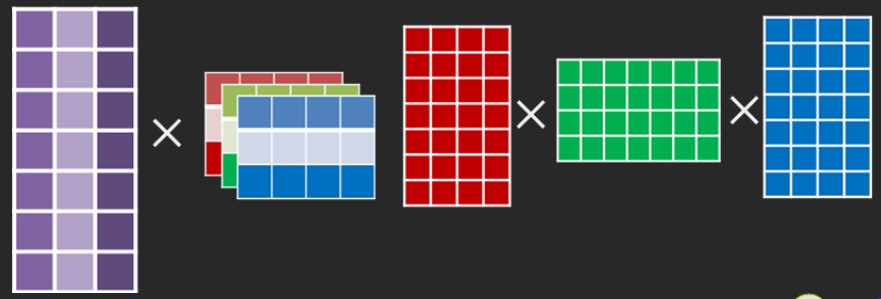
Не только NLP

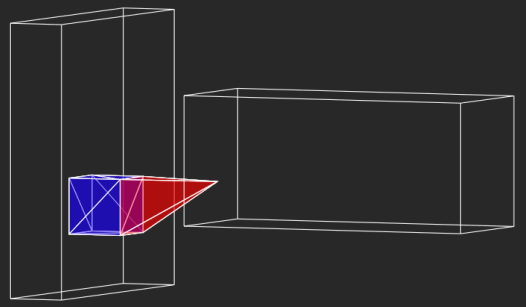


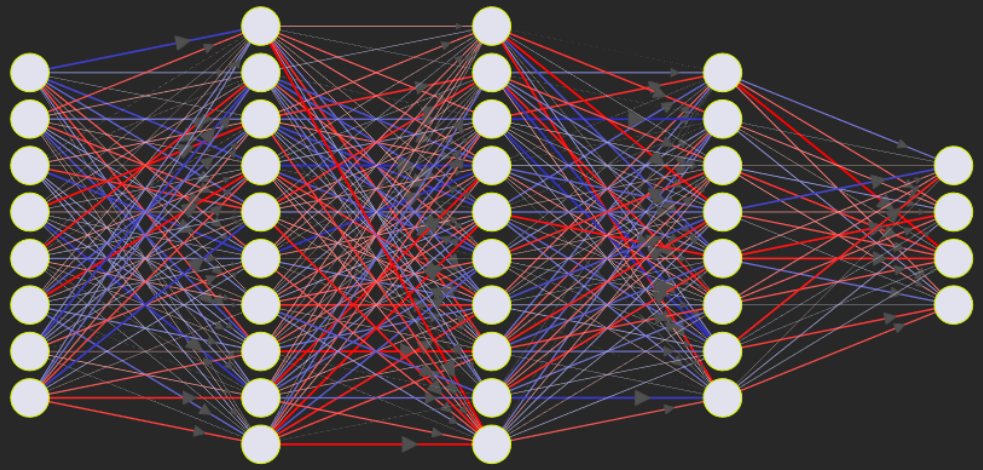
Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

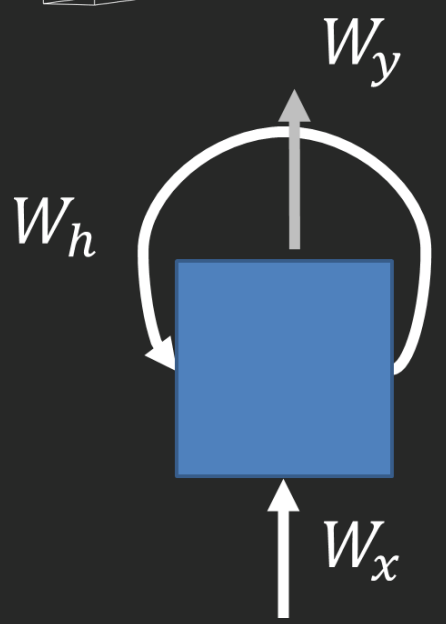
https://arxiv.org/abs/2010.11929

Трансформеры vs Полносвязные слои vs Свертки /Рекуррентные слои









Резюме

Про токенизацию

- Byte-Pair Encoding

Внимание Attention

- Key, Query, Value
- Multi-Head attention

Transformers

- Encoder-Decoder (оригинал для перевода)
  - Masked Attention, Positional Encoding
- Encoder (пропускаем данные через Внимание)
  - Семейство BERT
- Decoder («просто» генерируем текст)
  - Семейство GPT

Вопросы, пожелания, предложения

?????????

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Машинное Обучение

Лекция 2.07
Введение в базовые основы Промпт Инженеринга

Докладчик
Долганов Антон

Ранее в Машинном Обучении .

Про токенизацию

- Byte-Pair Encoding

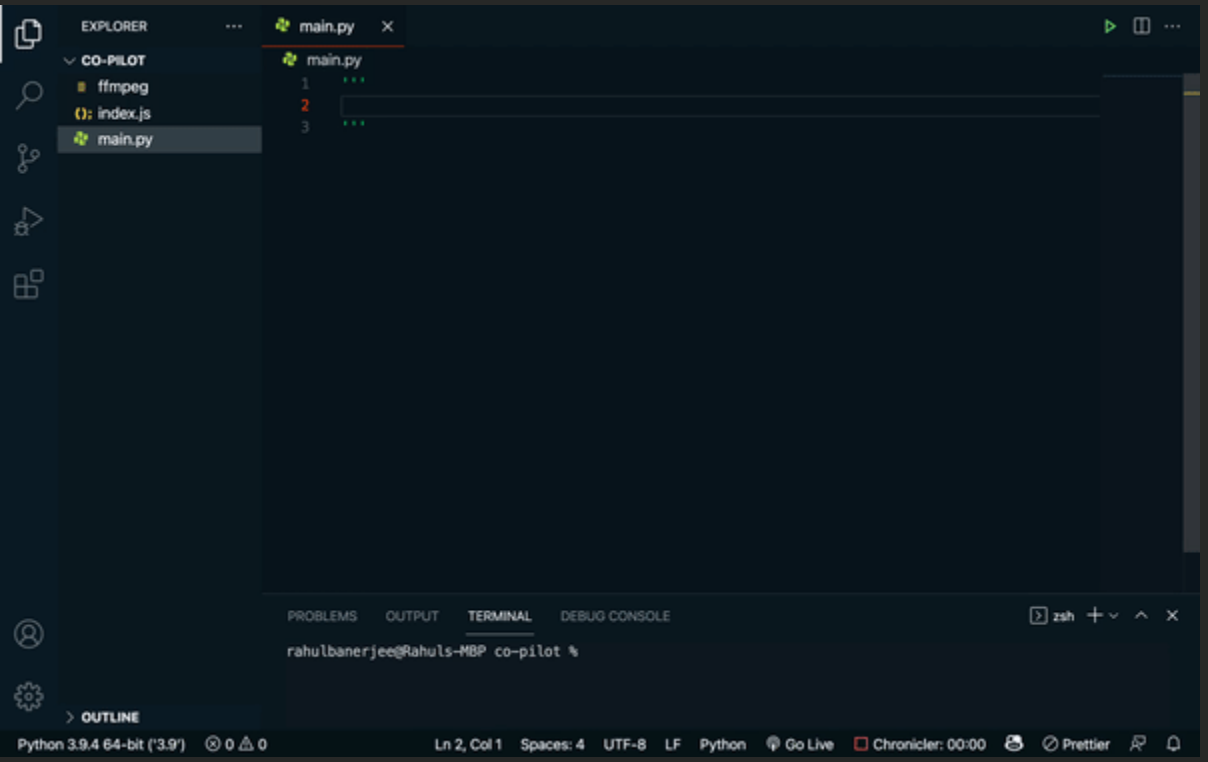
Внимание Attention

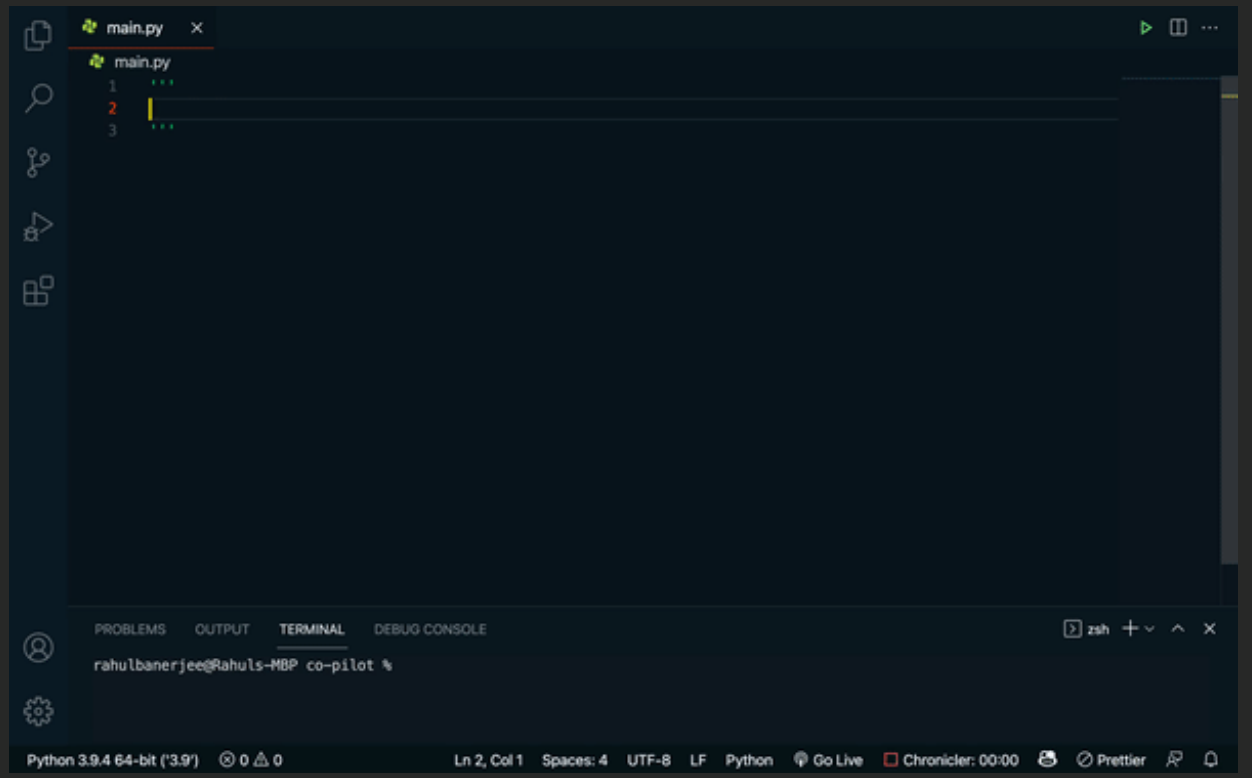
- Key, Query, Value
- Multi-Head attention

Transformers

- Encoder-Decoder (оригинал для перевода)
  - Masked Attention, Positional Encoding
- Encoder (пропускаем данные через Внимание)
  - Семейство BERT
- Decoder («просто» генерируем текст)
  - Семейство GPT

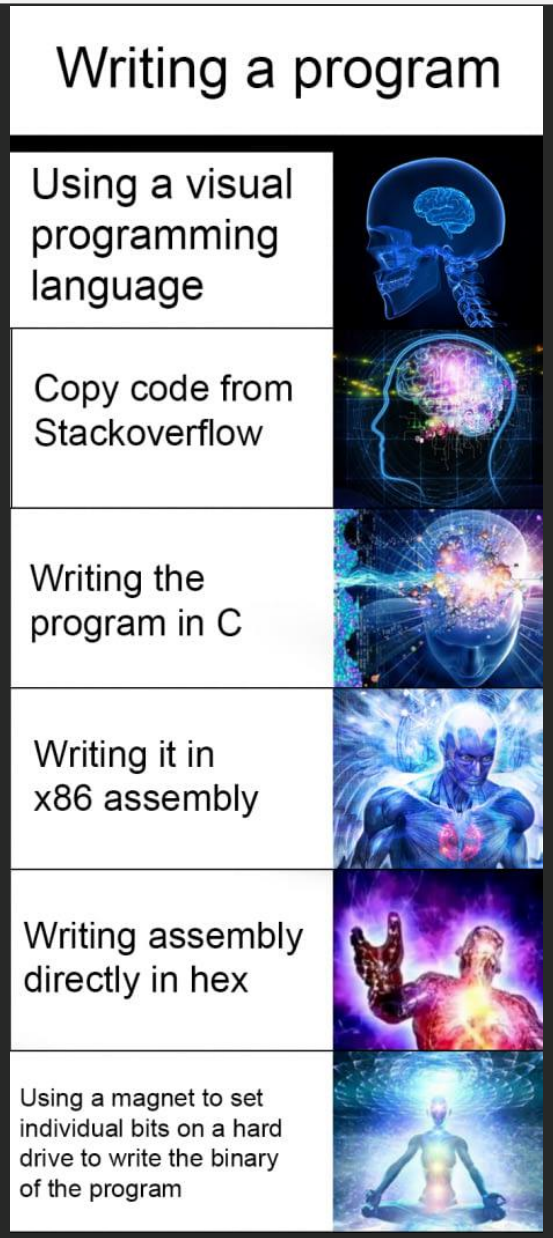
Copilot





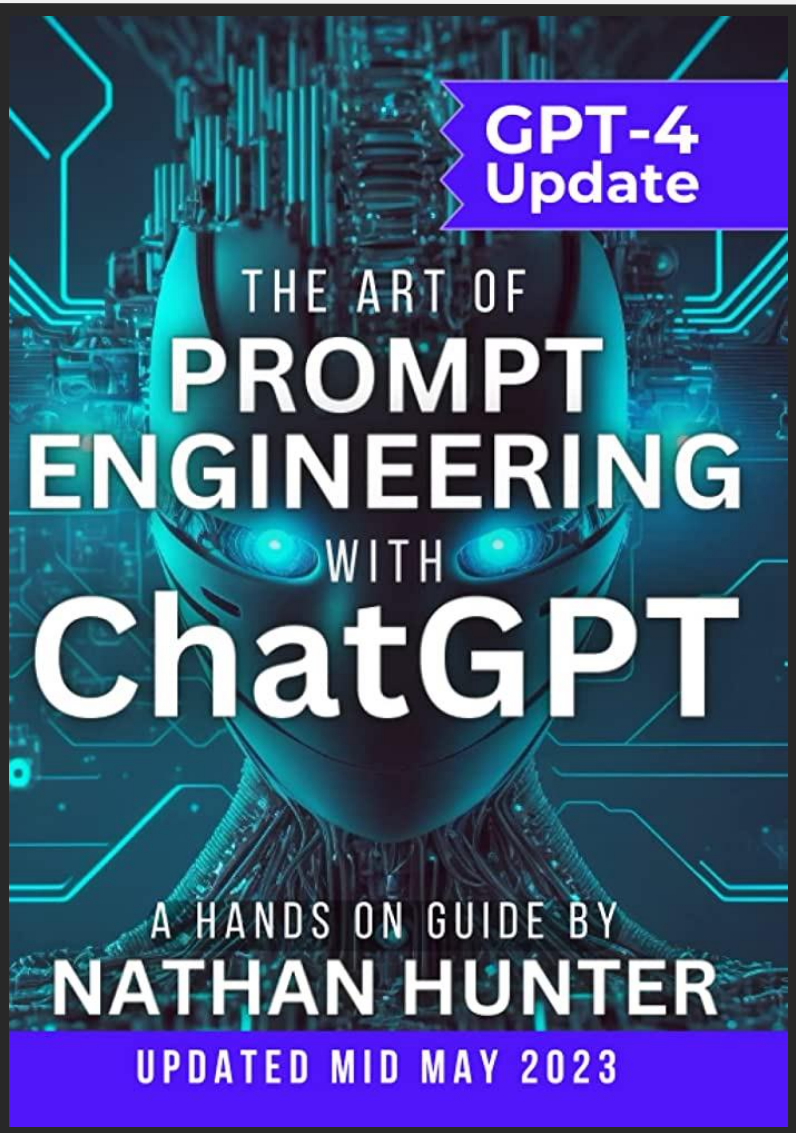
https://devsday.ru/blog/details/49410

Yet Another Уровень Абстракции?



Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Как обычно нужно было учиться еще вчера .



Prompt Engineer

з/п не указана

Требуемый опыт работы: 3–6 лет

Полная занятость, полный день

Условно полезное .

ChatGPT Prompt Engineering for Developers

https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/

Prompt Engineering Guide

https://github.com/dair-ai/Prompt-Engineering-Guide

Некая База

.

Принцип 1: «Нормально проси – Нормально будет»

- Используйте разделители в тексте чтобы структурировать информацию в промпте ({}, <>, “””””””)
- Конкретизируйте как должен выглядеть выход (json, html, просто ответ «да»/«нет»
- Просите не только делать ответ, а сначала просите проверить некоторые условия
- Дайте модели примеры как надо выполнить задачу (few-shot learning)

Некая База

Принцип 2: «Это всего лишь модель которая предсказывает следующий токен, а не ИИ уровня СкайНет»

- Указывайте конкретные шаги, которые вы ожидаете что она сделает
- Просите модель сначала самой решить задачу, а потом давать оценку правильности / неправильности
- Модель может уверенно обманывать «галлюцинировать»

Во избежание можете сначала попросить «давай только корректную информацию»

- Готовьтесь к Итеративному процессу работы

Немного примеров

.

Write a small intro to workshop aimed for a beginner students. The topic of workshop is a swift introduction to Python, in particular Pandas and scikitlearn libriaries . Last part of the workshop will be devoted to image generation using Stable Diffusion

Welcome to our beginner's workshop on Python! In this workshop, we will provide you with a swift introduction to Python, one of the most popular programming languages today. We will focus on two essential libraries, Pandas and scikit-learn, which are widely used for data analysis and machine learning.

By the end of this workshop, you will have a good understanding of how to use these libraries to perform data analysis and build machine learning models. Additionally, we will also explore the Stable Diffusion library and learn how to generate images using it.

Whether you are a complete beginner or someone who has some basic knowledge of Python, this workshop is designed to help you get started with these essential libraries and take your skills to the next level. So, let's dive in and explore the exciting world of Python together!

Rewrite part about stable diffusion with more catchy style

But wait, there's more! We'll also be exploring the Stable Diffusion library and learning how to generate stunning images with it. Get ready to unleash your creativity and produce mesmerizing visuals that will leave your audience in awe. With Python and Stable Diffusion, the possibilities are endless. Join us on this journey and let's bring your artistic vision to life!

Немного примеров

Добро пожаловать на наш мастер-класс для начинающих по Python! На этом семинаре мы познакомим вас с Python, одним из самых популярных сегодня языков программирования. Мы сосредоточимся на двух важных библиотеках, Pandas и scikit-learn. К концу этого семинара вы будете хорошо понимать, как использовать эти библиотеки для анализа данных и построения моделей машинного обучения.

Независимо от того, являетесь ли вы полным новичком или имеете некоторые базовые знания Python, этот семинар предназначен для того, чтобы помочь вам начать работу с этими важными библиотеками и вывести свои навыки на новый уровень. Итак, давайте погрузимся и исследуем захватывающий мир Python вместе!

Но подождите, это еще не все! Мы также изучим библиотеку Stable Diffusion и научимся создавать с ее помощью потрясающие изображения. Приготовьтесь раскрыть свой творческий потенциал и создать завораживающие визуальные эффекты, которые вызовут трепет у вашей аудитории. С Python и Stable Diffusion возможности безграничны. Присоединяйтесь к нам в этом путешествии, и давайте воплотим ваше художественное видение в жизнь!

Вопросы, пожелания, предложения

????????