Лекция 2. CNN в техническом зрении, архитектуры CNN, устройство CNN. Основные задачи, решаемые CNN. Слои и параметры. Типовые архитектуры CNN: AlexNet, VGG, GoogleNet, ResNet, ResNEXT, DenseNet, MobileNet

В.С. Горбацевич

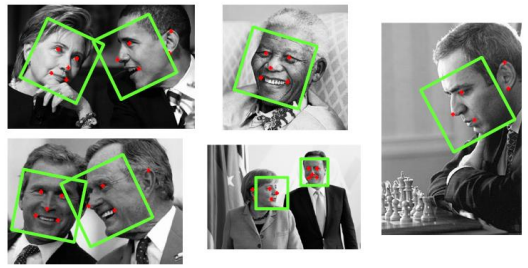
начальник лаборатории 3050

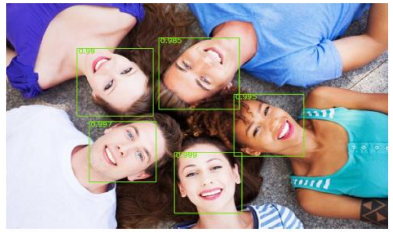
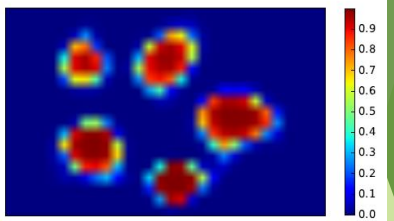
ФГУП «Государственный научно-исследовательский институт авиационных систем»

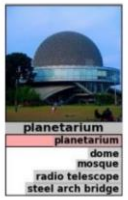
Москва, ГосНИИАС, 31.10.2018

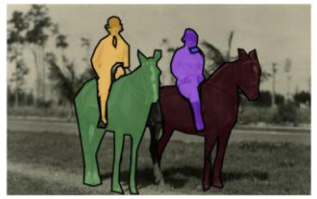
Задачи, решаемые с помощью глубоких конволюционных сетей

+Классификация
+Семантическая сегментация
+Поиск объектов
+Распознавание лиц
+Распознавание объектов
+Улучшение изображений
+Денойсинг
+Стерео отождествление
+Деблур
+Оценка ядер смаза
+Оценка характеристик объекта
+Построение признаковых описаний
+Оценка активности(динамических характеристик)
+Аннотирование
+Пстроение 3D моделей
+Суперразрешение
+Косплексирование
+Распознавание жестов/действий
+Антиспуффинг
+Оценка положения/калибровка
+Выделение краев
+Распознавание текста/номеров/надписей
+Нормализация
+Треккинг



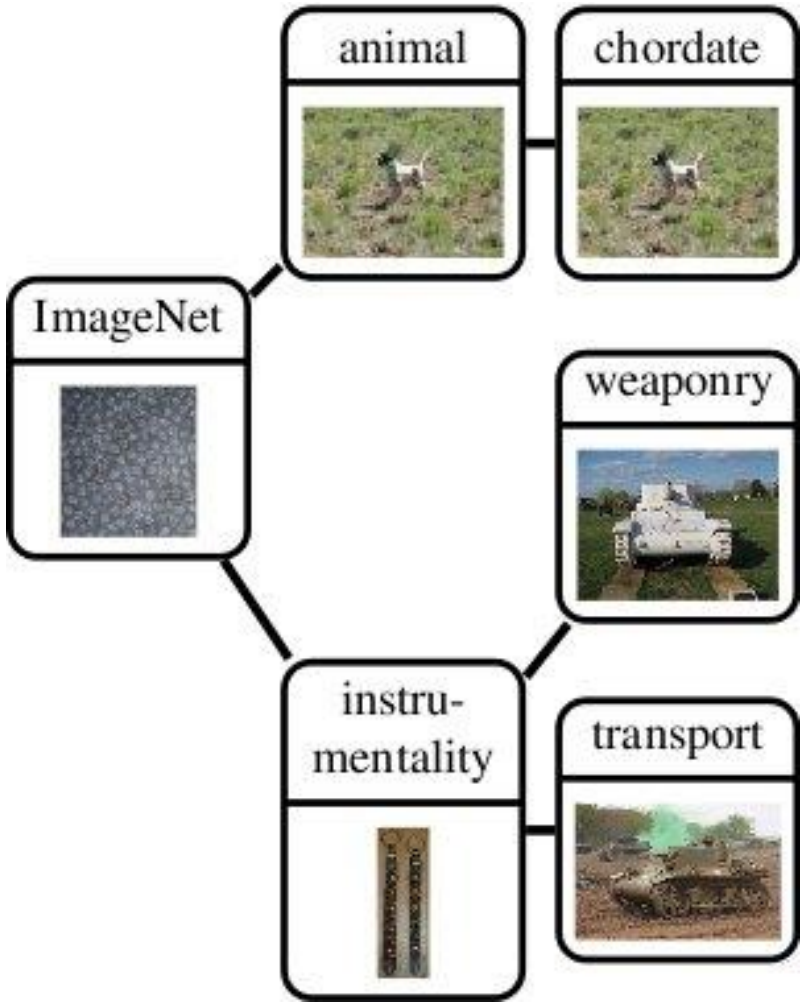








Практически все задачи технического зрения !!!

Классификация



Вход:

Выход:

Что это?

Классификация

Постановка: Есть n-классов изображений необходимо каждому изображению поставить в соответствие метку класса.

Примеры:

Определение марки машины
Определение пола по фото
Определение цифры/буквы

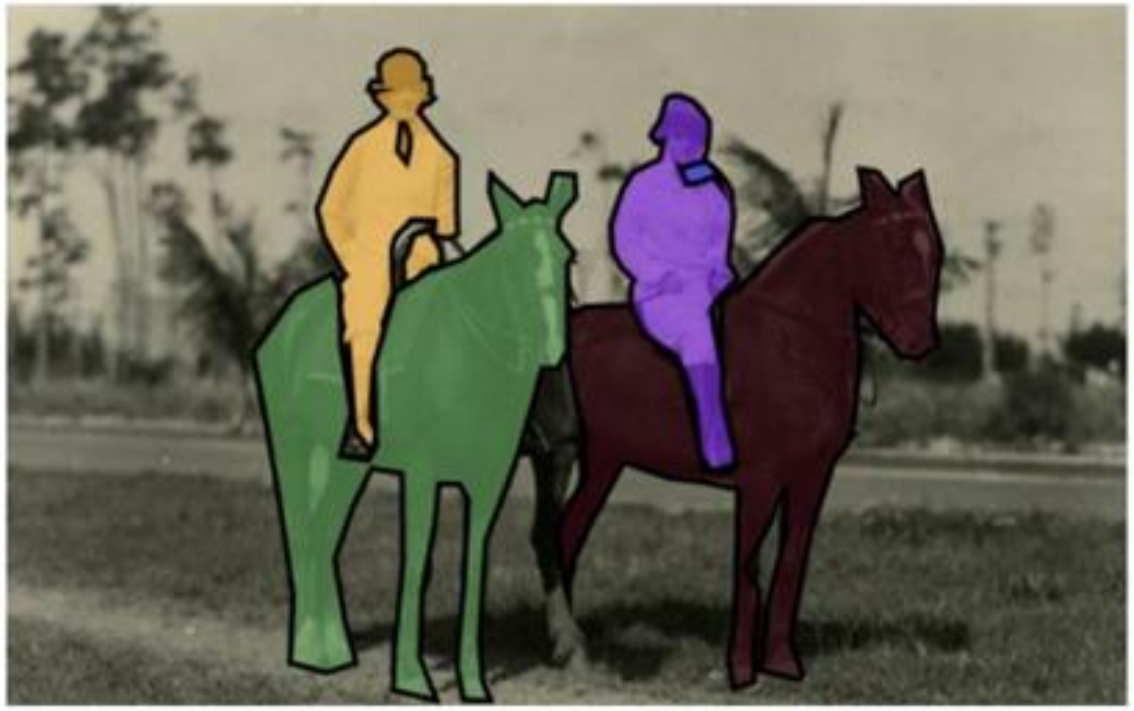
?

Семантическая сегментация

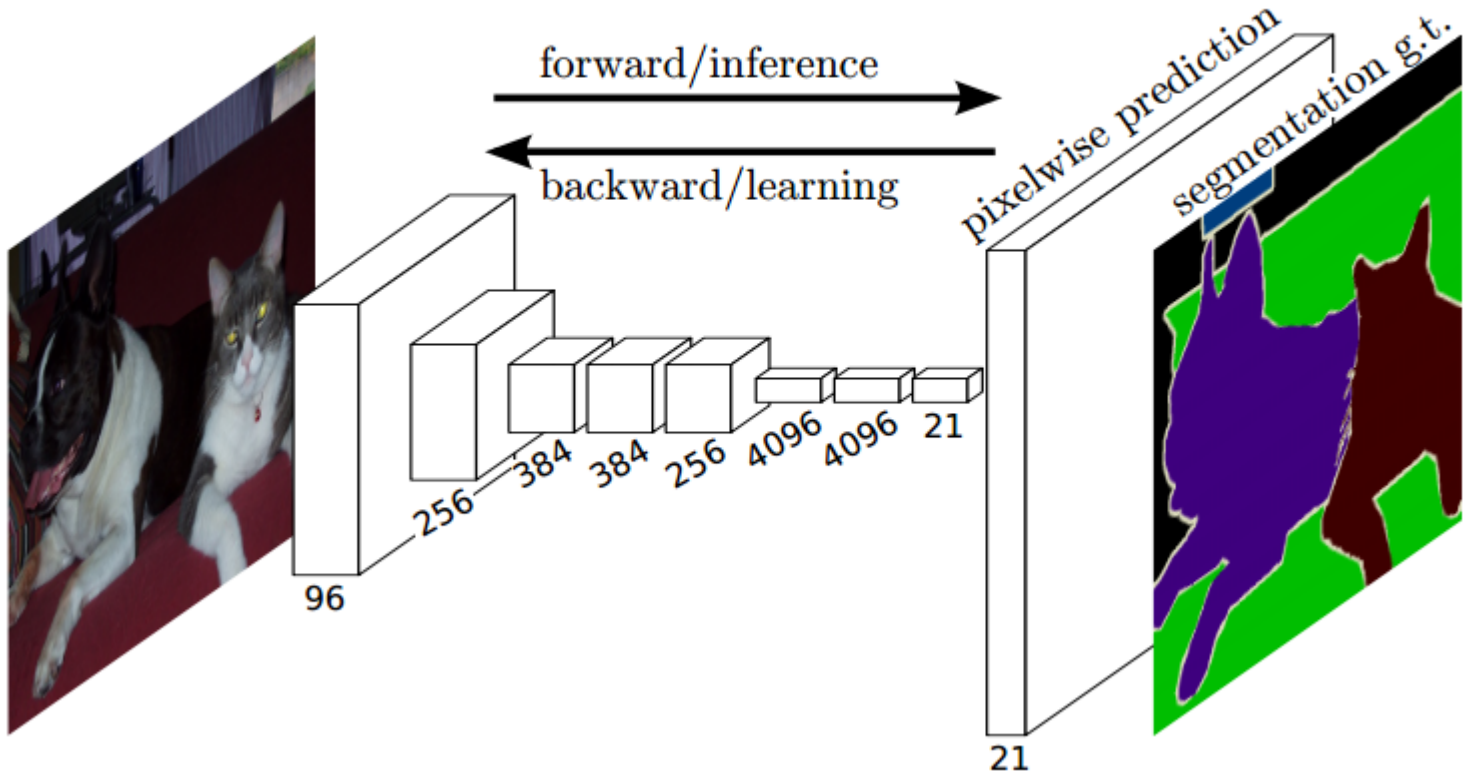
Постановка: Есть n-классов необходимо для каждого изображения создать сегментированное изображение причём сегменты должны соответствовать объектам этих классов.

Примеры:

автоматическое составление карт



Семантическая сегментация



Fully Convolutional Networks for Semantic Segmentation
Jonathan Long* Evan Shelhamer* Trevor Darrell
https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
Code: https://github.com/shelhamer/fcn.berkeleyvision.org

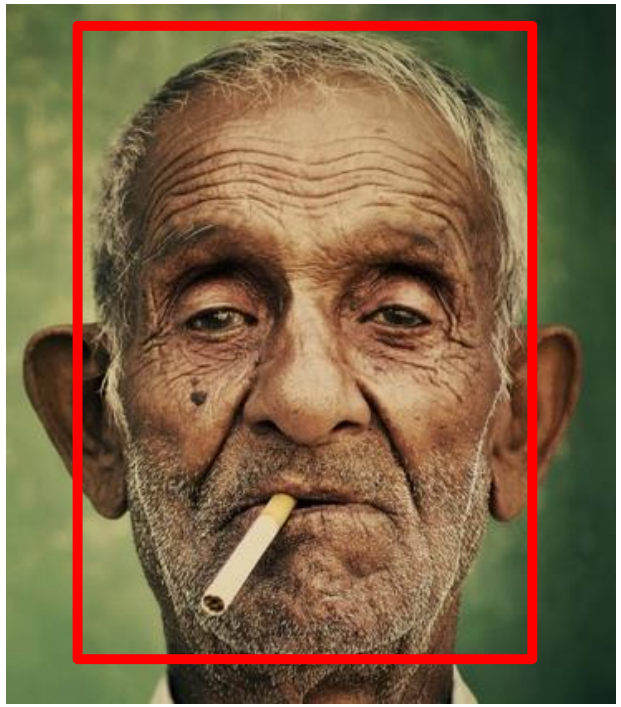
Обнаружение объектов

Постановка: Есть изображение на нем необходимо выделить описывающие прямоугольники определённого класса объектов.

Примеры:

Поиск машин/самолетов и т.д

Поиск глаз/рта/ушей



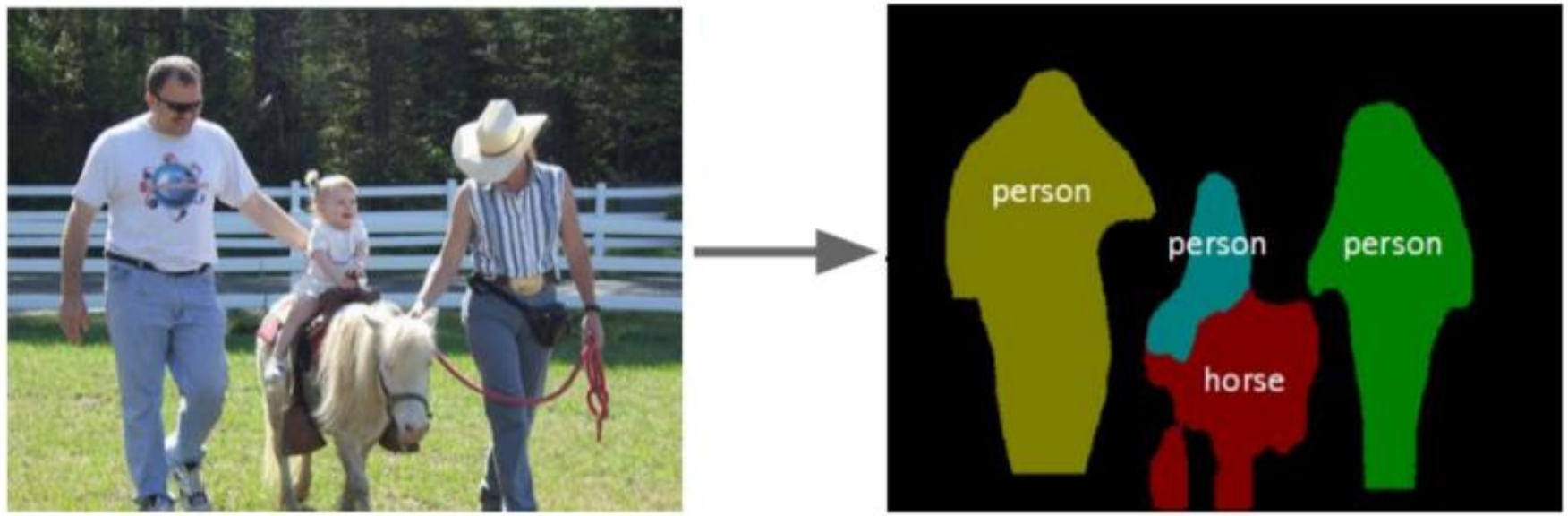
Обнаружение объектов

Описывающий прямоугольник:

Алгоритмы:
YOLO,DSOD,R-CNN,R-FPN, Light-head R-CNN

Обнаружение объектов

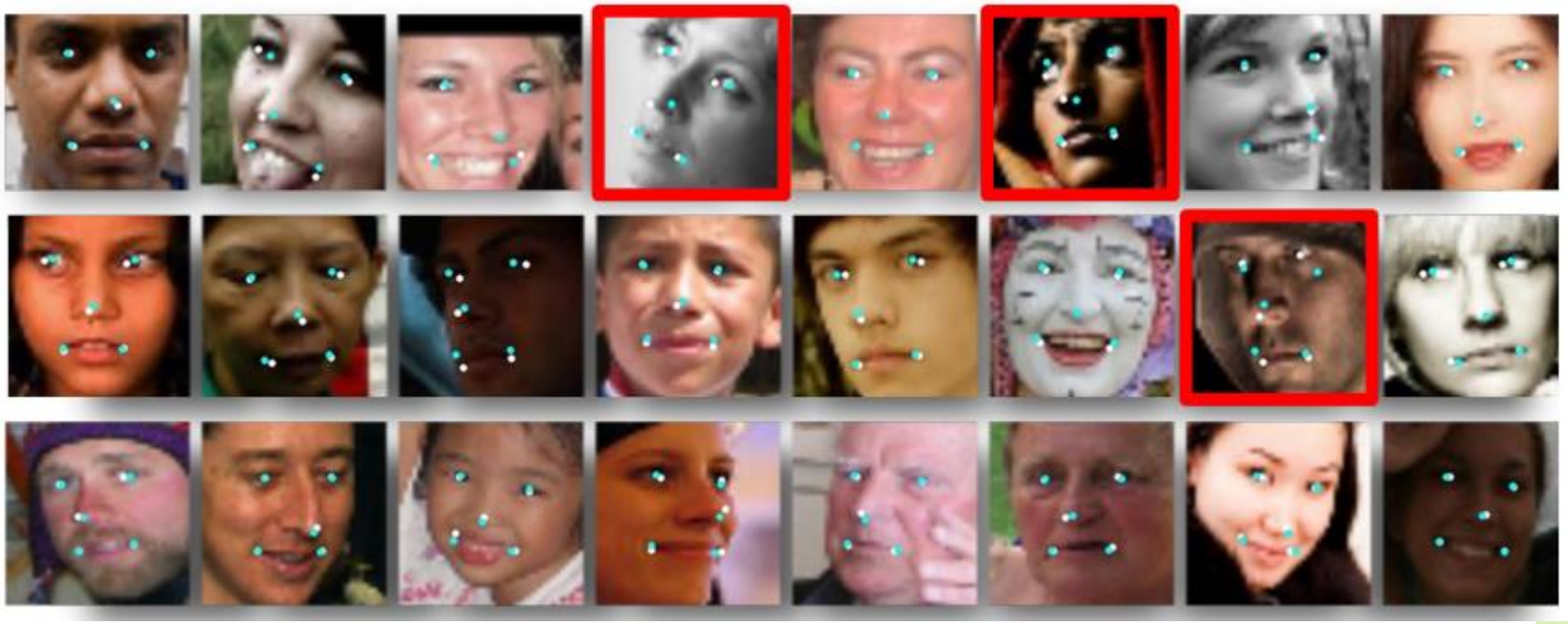
Точные границы объекта:



Алгоритмы:
Mask R-CNN, Box2Pix

Обнаружение объектов

Координаты точек:

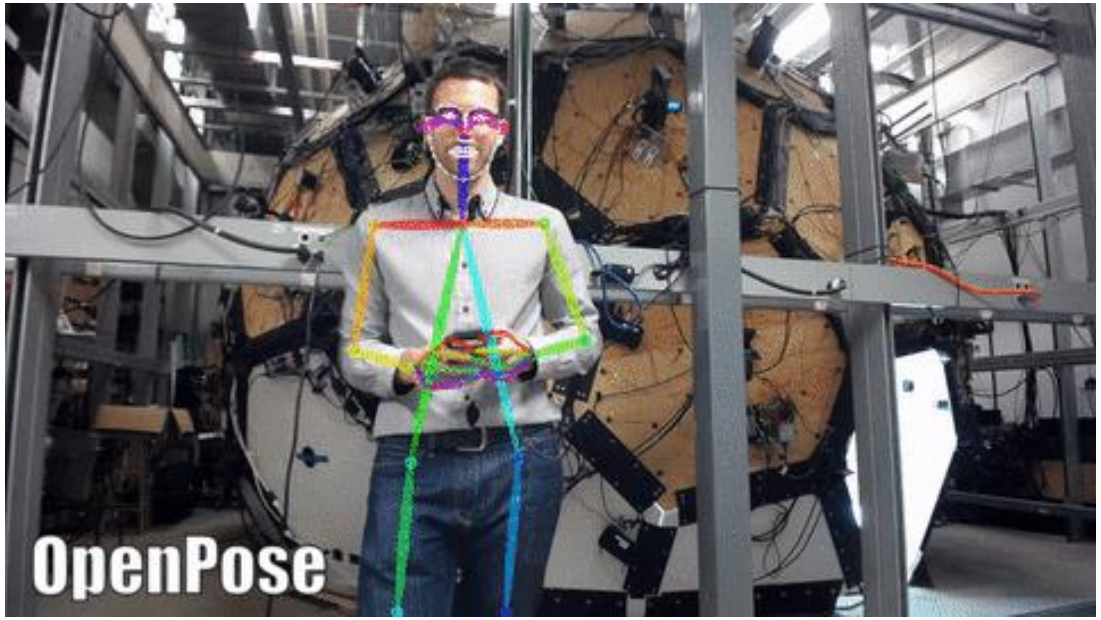


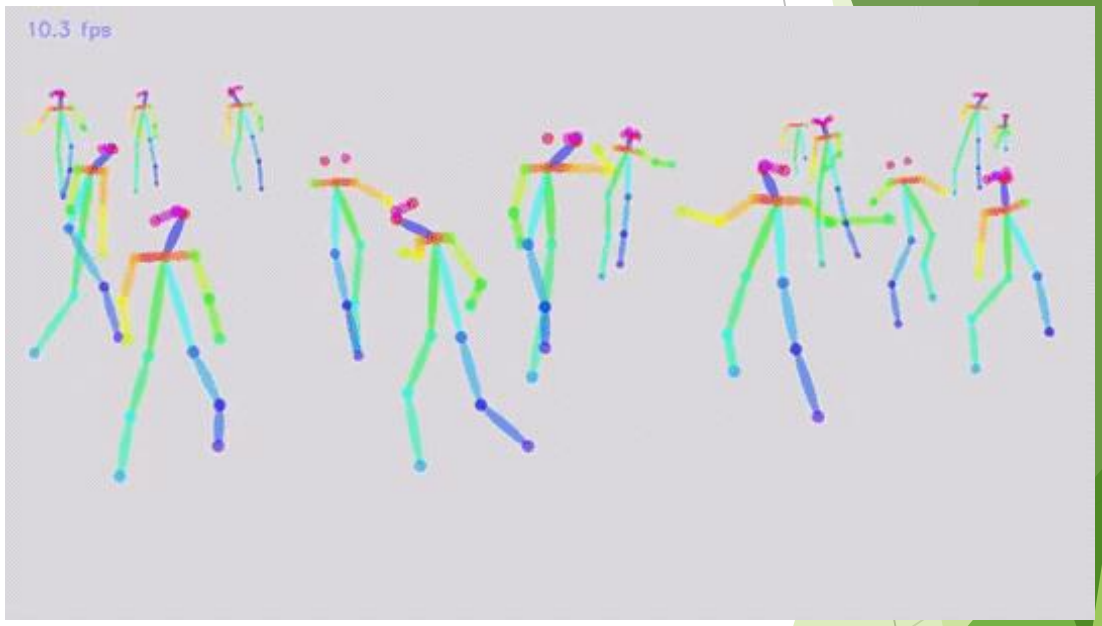
Vanilla net:

https://gist.github.com/ishay2b/58248e5f3c3bf575ac40

Обнаружение объектов

Координаты точек:





Open Pose:
https://github.com/CMU-Perceptual-Computing-Lab/openpose

Регрессия/предсказание

Постановка: Есть изображение необходимо построить по нему некий сигнал/характеристики.

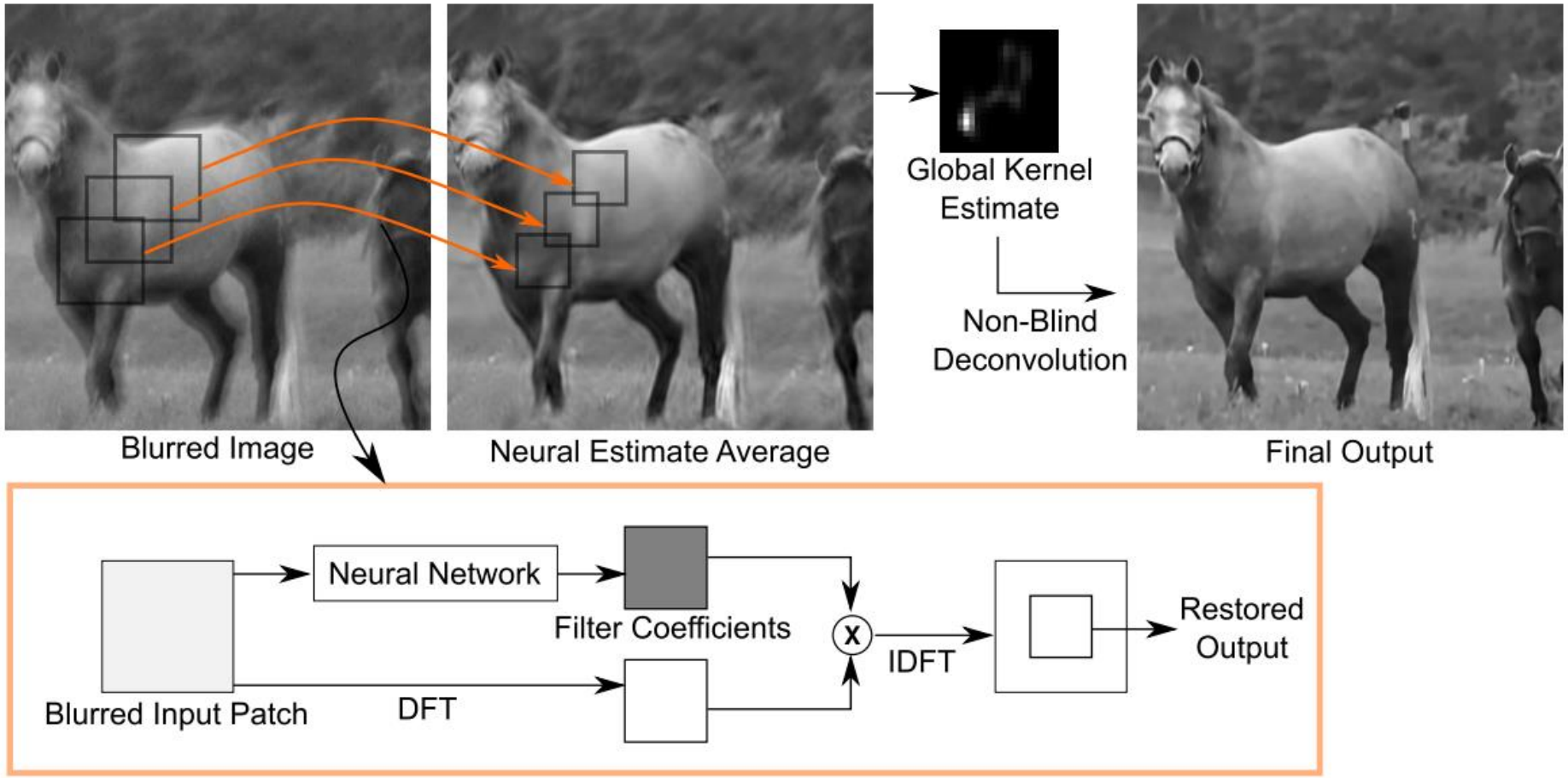
Примеры:

Определение ядра смаза

Определение вектора скорости

Определение вектора g

Регрессия/предсказание



A Neural Approach to Blind Motion Deblurring
Ayan Chakrabarti ECCV 2016

Code: https://github.com/ayanc/ndeblur

Обучение дескрипторов/сравнение

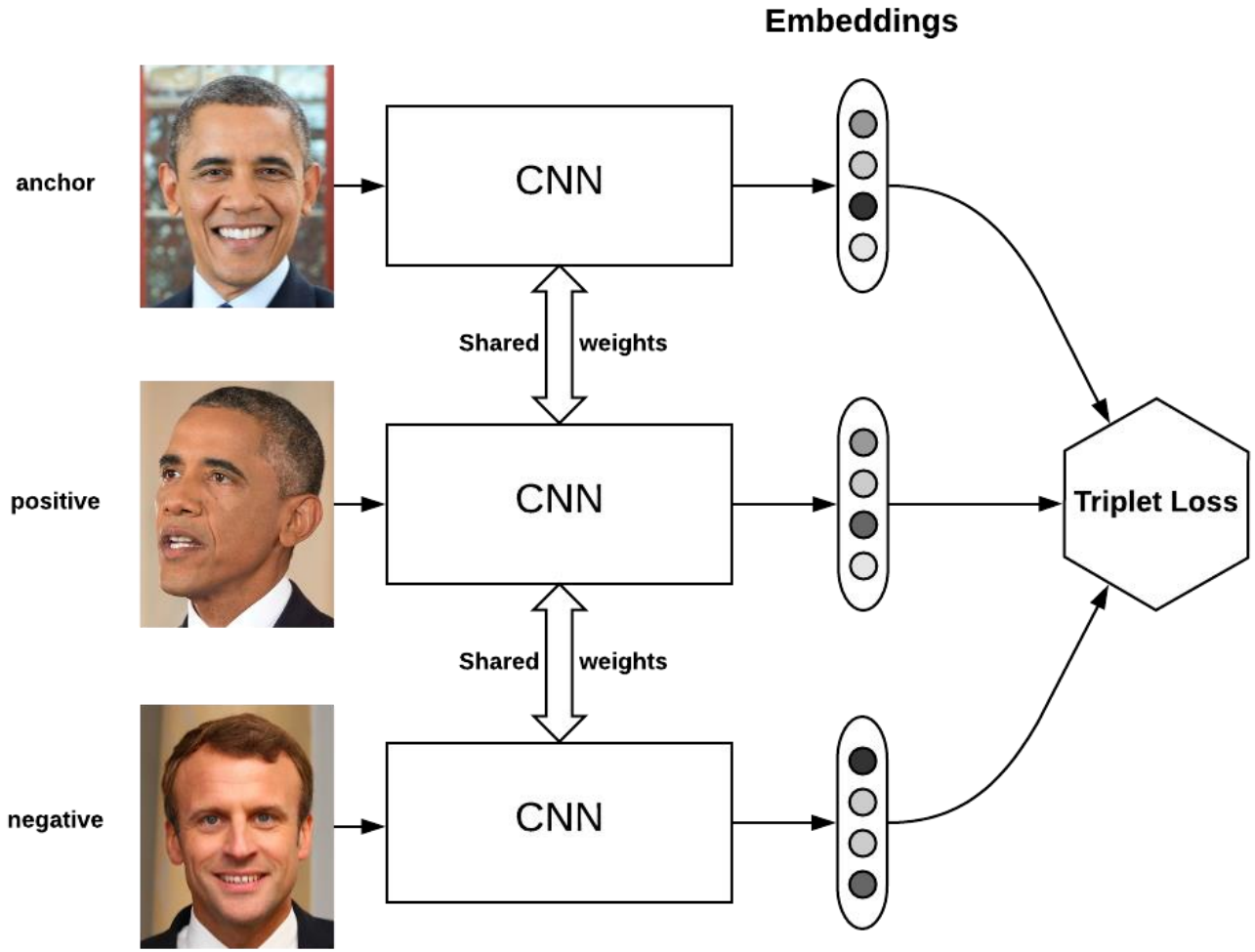
Постановка: Есть некоторый набор объектов задача построить информативные с точки зрения сравнения объектов дескрипторы либо напрямую сравнение изображений.

Примеры:

Поиск человека по фото

Поиск в базе

Обучение дескрипторов/сравнение

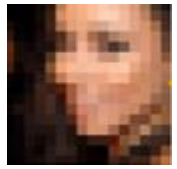


Суперразрешение

Постановка: Есть изображение его необходимо увеличить до некоторого размера с максимальным качеством.

.

оригинал

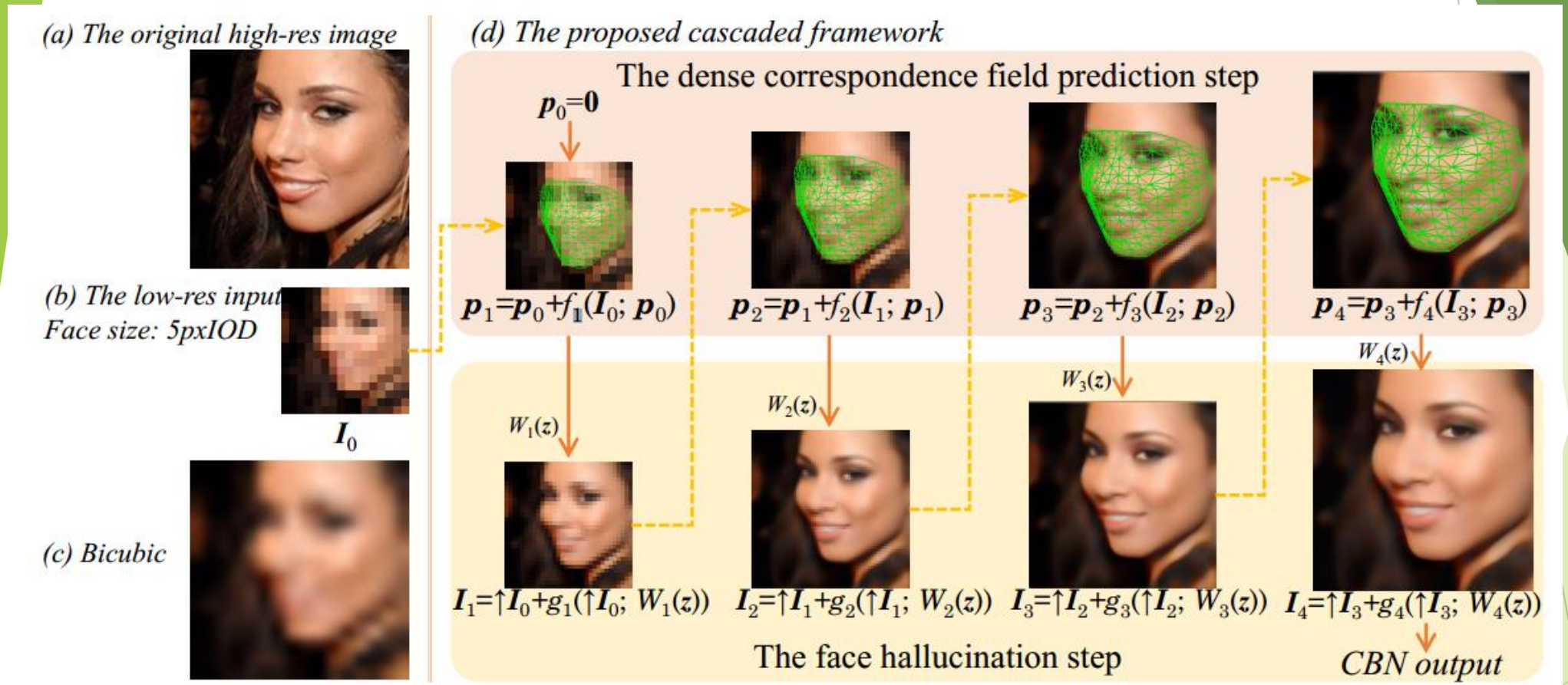


изображение 16x16

Бикубическая интерполяция

Увеличение ГКНС

Суперразрешение



Deep Cascaded Bi-Network for Face Hallucination

Zhu, Shizhan and Liu, Sifei and Loy, Chen Change and Tang, Xiaoou ECCV 2016

Code https://github.com/zhusz/ECCV16-CBN

Деконволюция

Постановка: необходимо устранить смаз/размытие на изображении

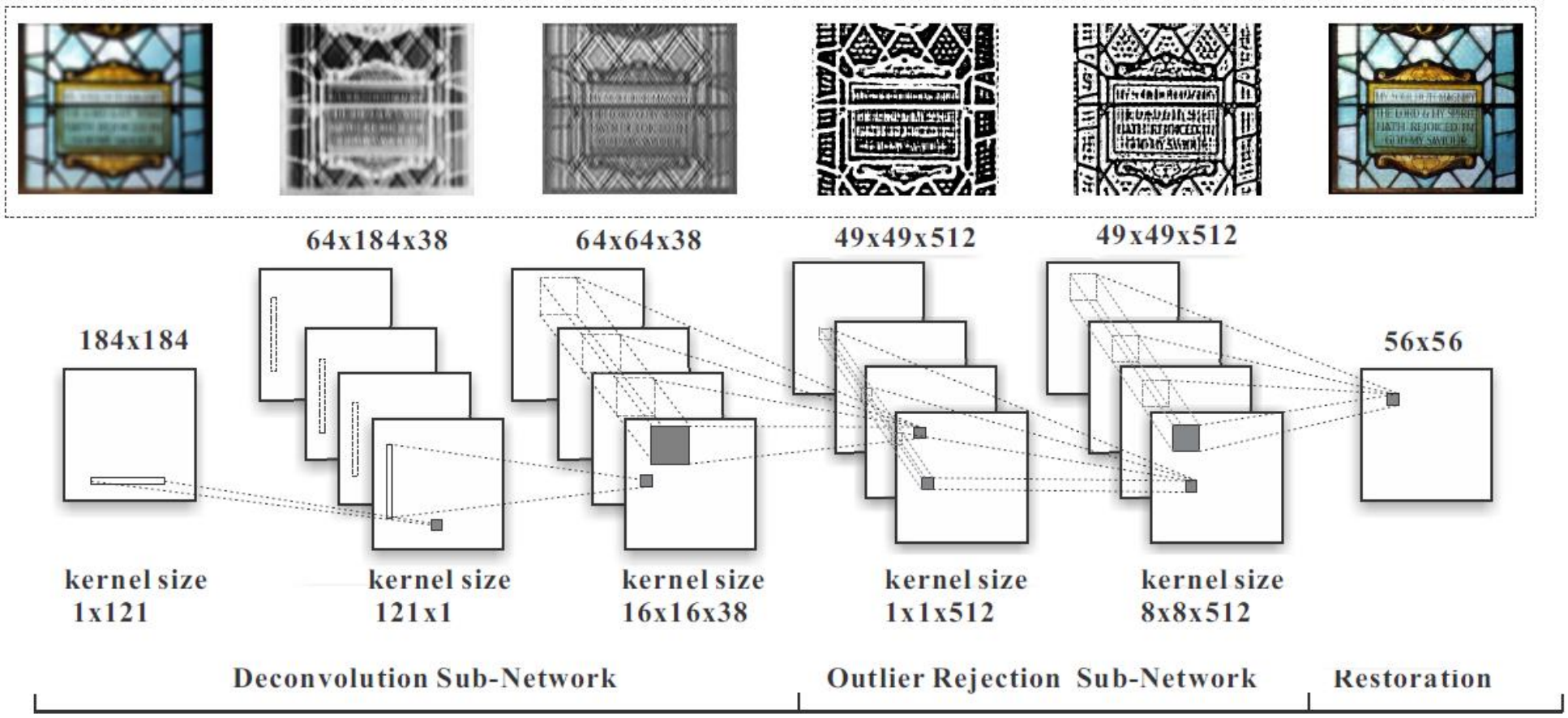


Figure 6: Our complete network architecture for deep deconvolution.

Деконволюция

(a) input (b) SSDAE [15] (d) Ours

Денойсинг

Постановка: необходимо устранить шумы с изображения

noisy ($\sigma = 25$)PSNR:20.16dB

ours: PSNR:30.03dB

noisy ($\sigma = 25$)PSNR:20.19dB

ours: PSNR:29.21dB

Image denoising: Can plain Neural Networks compete with BM3D? Harold C. Burger, Christian J. Schuler, and Stefan Harmeling CVPR 2012

Денойсинг

"stripe" noise: 20.23 dB

salt and pepper noise: 12.39 dB

JPEG quantization: 27.33 dB

our result: **30.09** dB

our result: **34.50** dB

our result: **28.97** dB

3D from image

Постановка: По одному входному изображению требуется построить 3-мерную вексельную модель сцены.

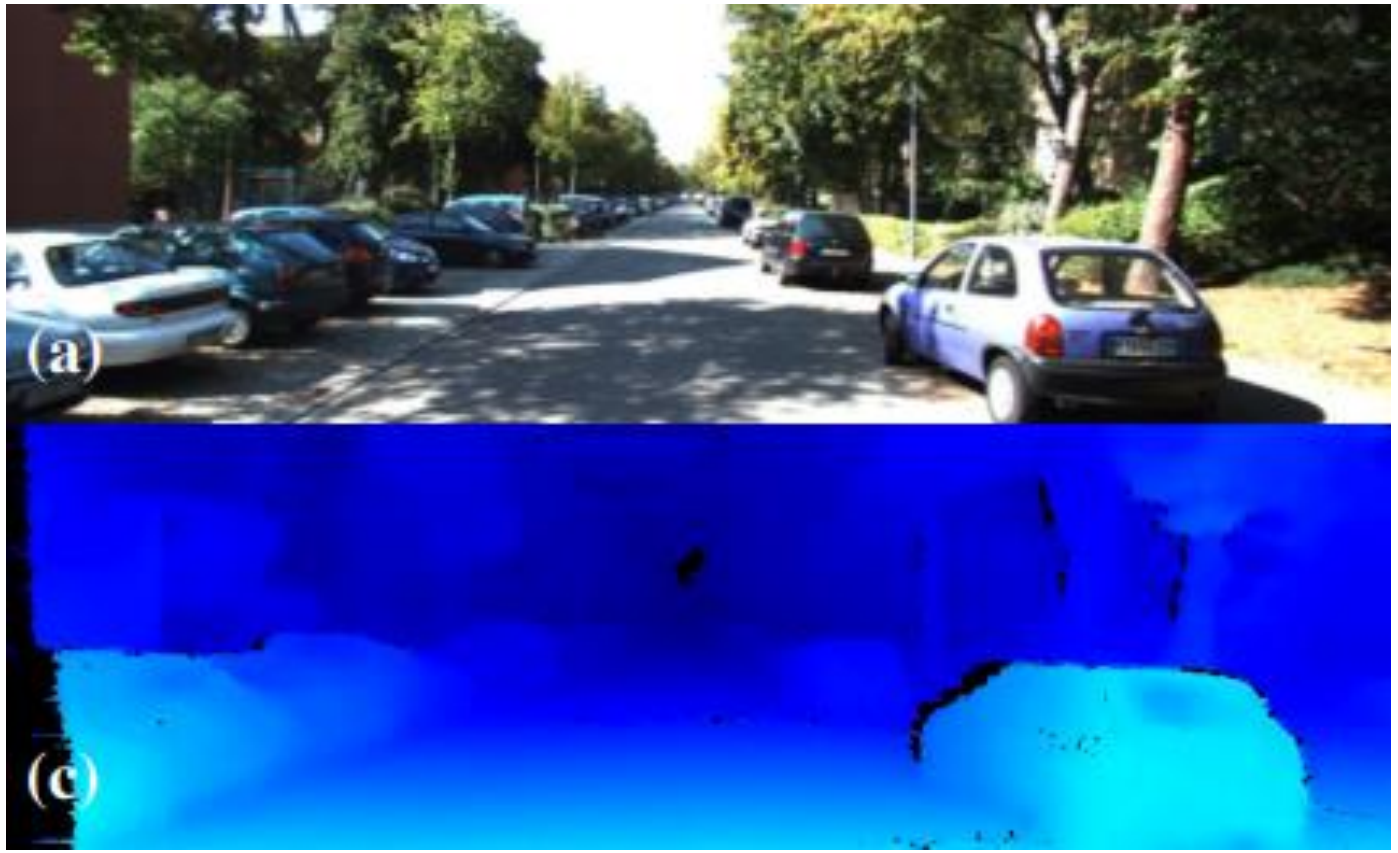


Image inpainting

Постановка: восстановление неизвестной области изображения с максимальным качеством.

Преобразование изображение в изображение

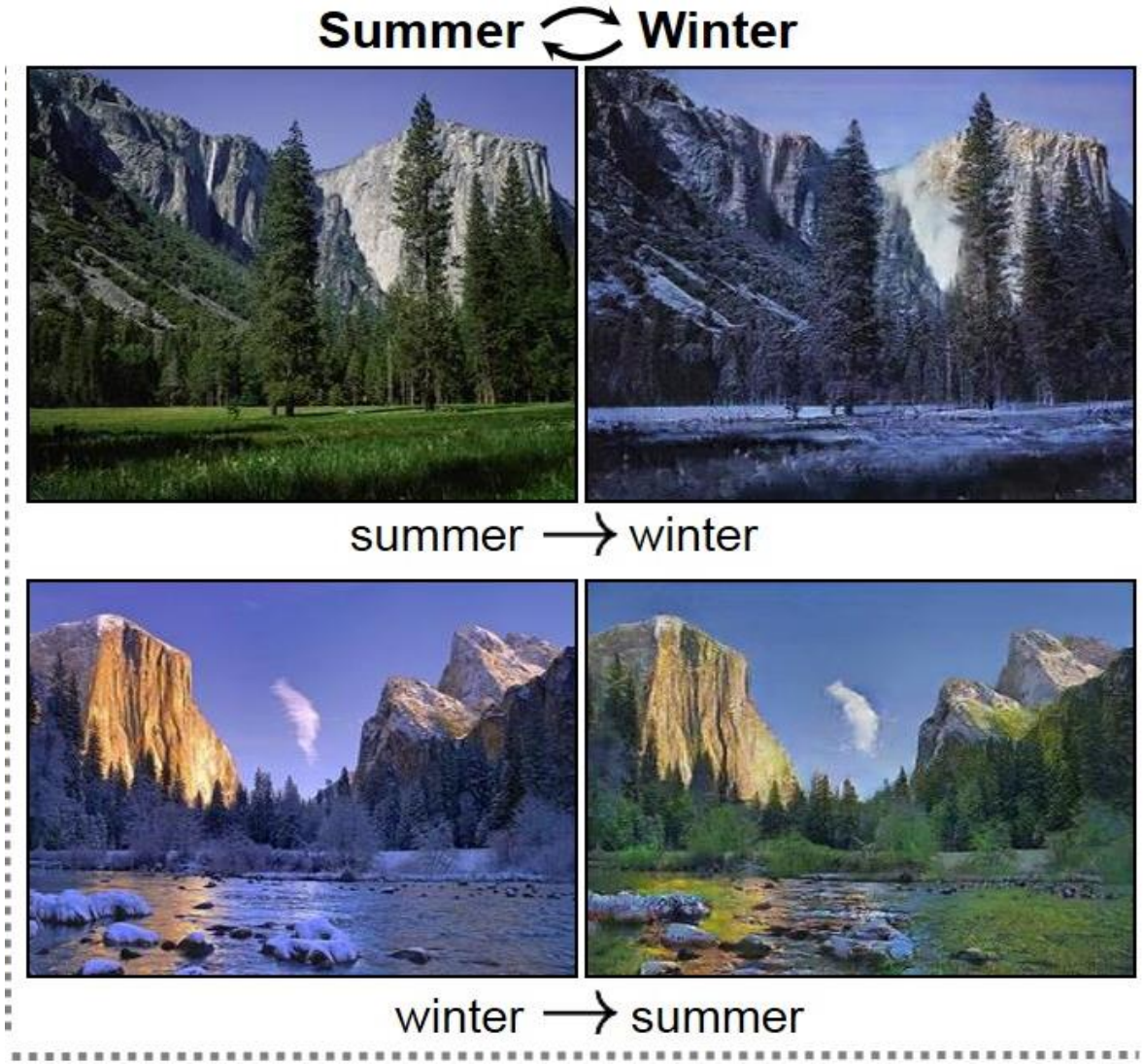
Постановка: Преобразование изображение в изображение по произвольному правилу(например лошади в зебры)

Cycle GAN:

https://junyanz.github.io/CycleGAN/

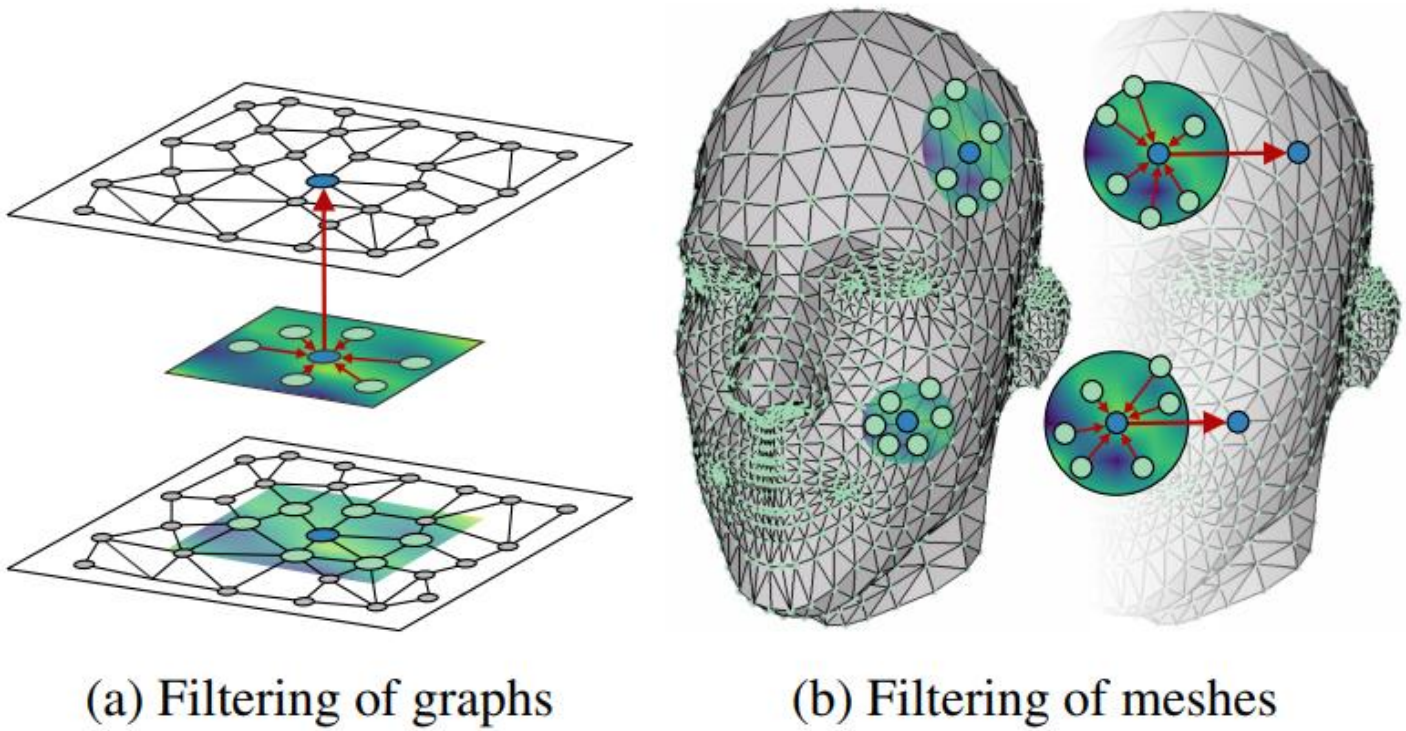
Генеративные соревнующиеся сети

Генеративные соревнующиеся сети



Генеративные соревнующиеся сети

Нерегулярные данные

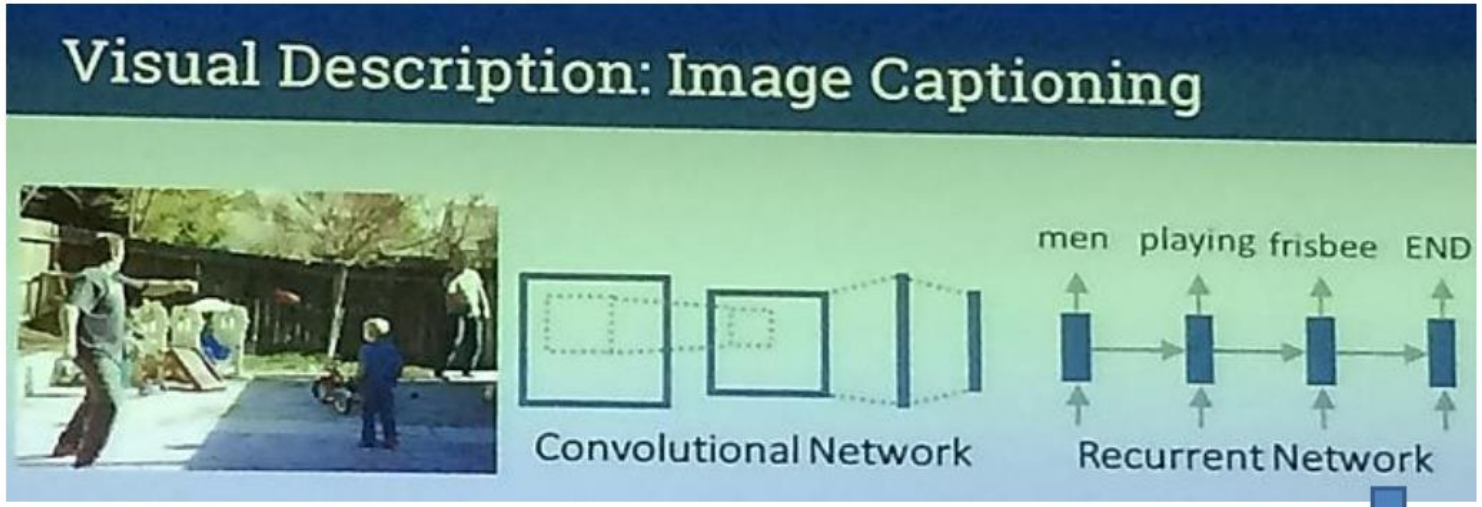


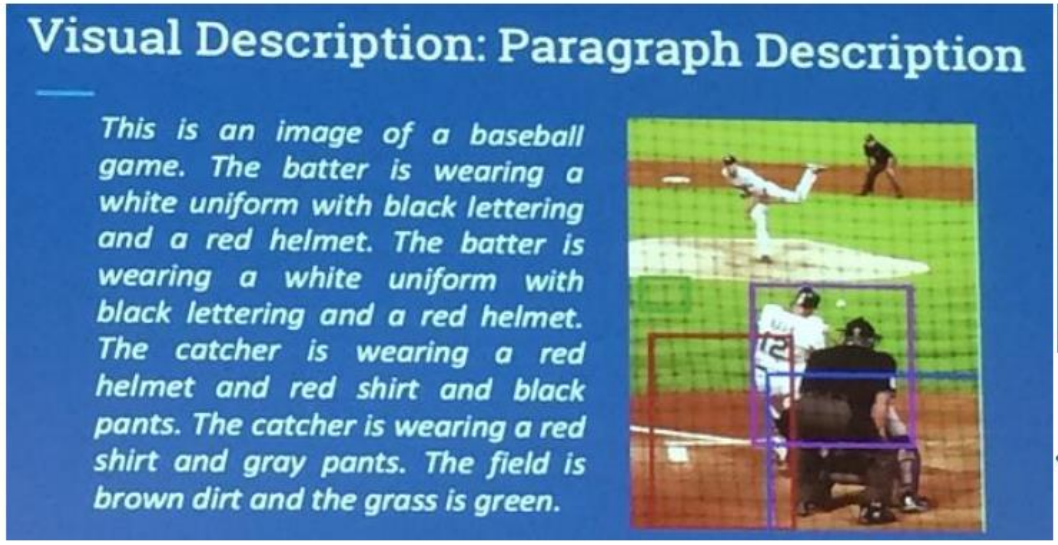
(a) Filtering of graphs (b) Filtering of meshes

Работа с 3-мерными данными
Задачи химии

https://github.com/rusty1s/pytorch_geometric

Аннотирование и рассказ историй



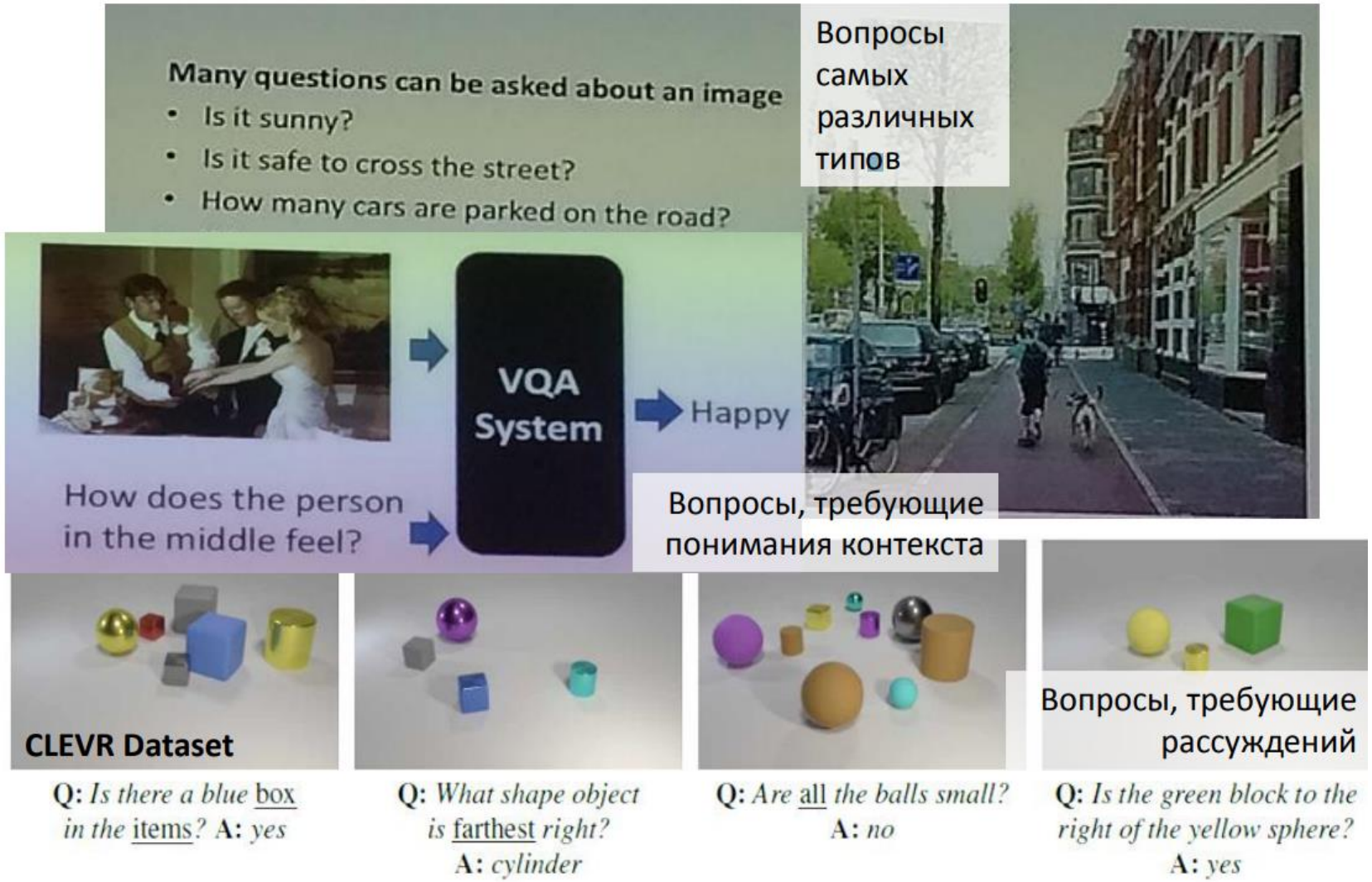


Работа с текстом:
Text2vec
LSTM

Составление связного описания содержания изображения и видеопоследовательности

Визильтер Ю.В. ИОИ - 2018

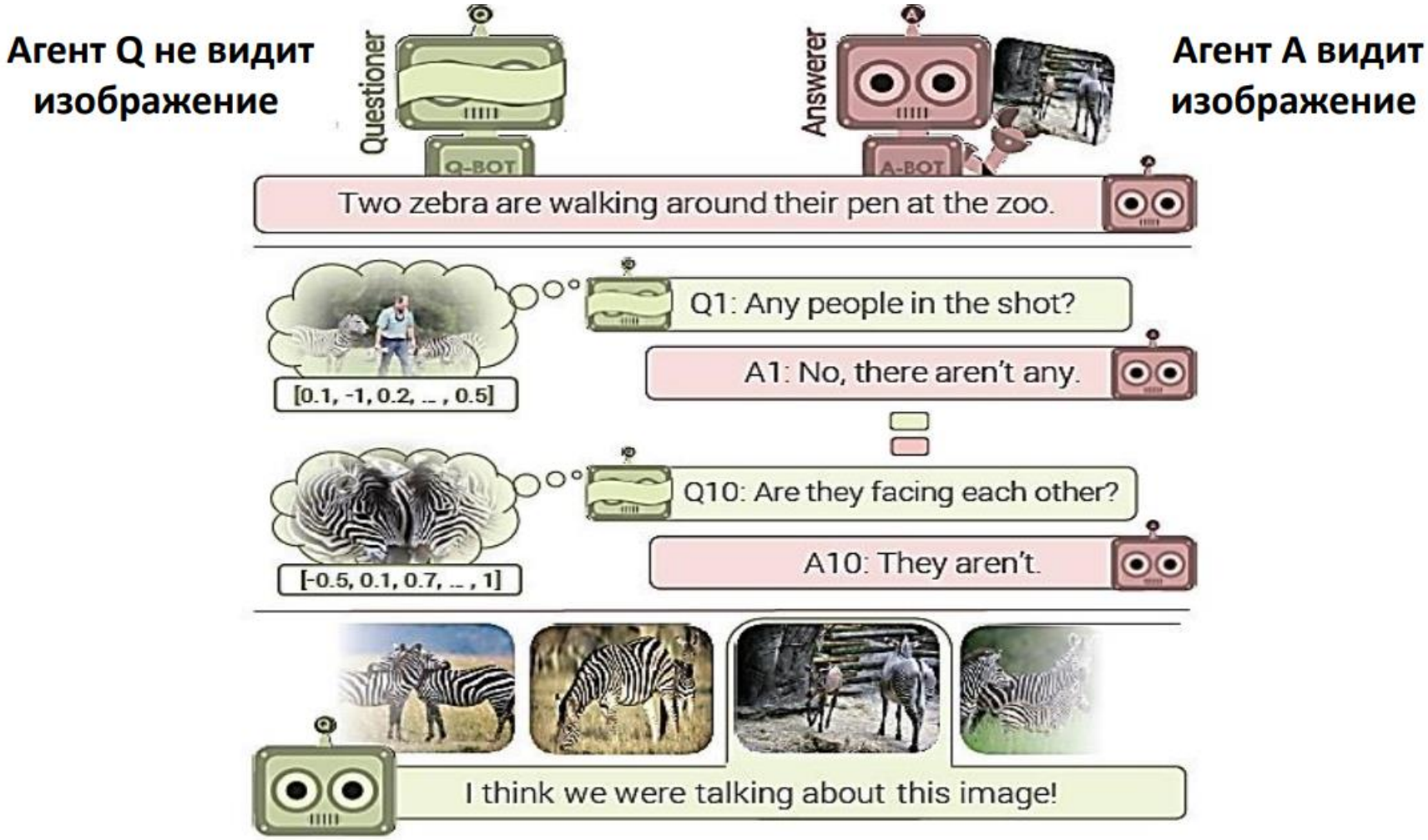
VQA:Ответы на вопросы по изображению



Работа с текстом:
Text2vec
LSTM

Closing the Loop Between Video and Language, ICCV Workshop, 2017

Обучение с подкреплением +ГКНС



Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning, Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, Dhruv Batra, ICCV, 2017

ГКНС.Базовые понятия

Архитектура ГКНС

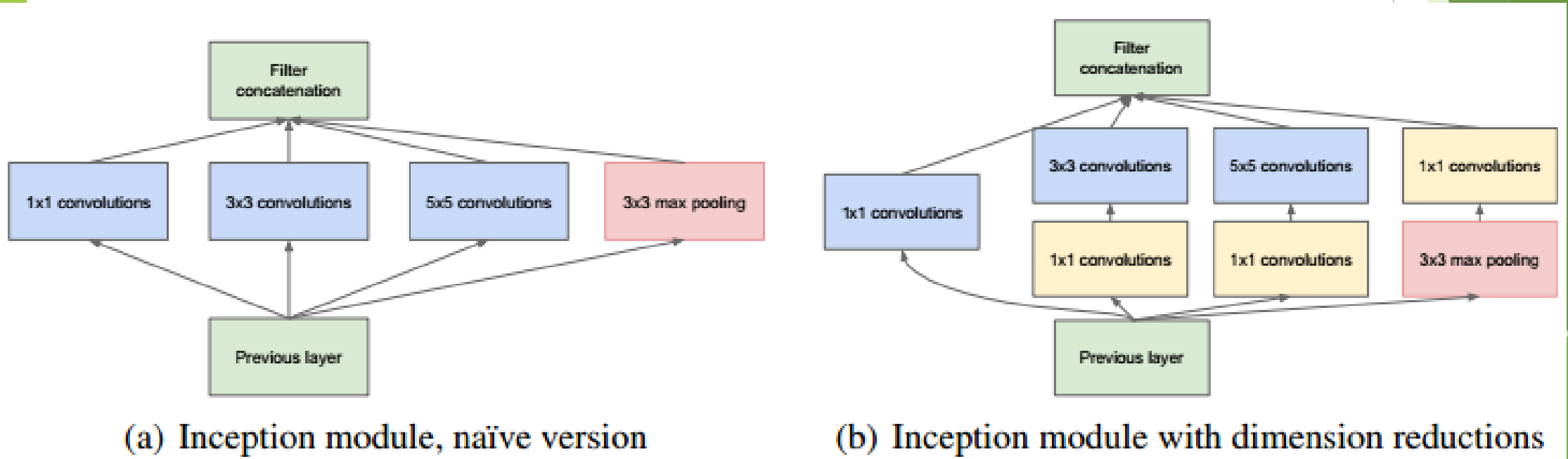
ГКНС состоит из слоев(операций):

1. Конволюционный
2. Слой нелинейности
3. Пулинга
4. Батч-нормализации
5. Вспомогательных слоев(загрузки данных и т.д.)
6. Конкатенации
8. Разделения
9. Слой поэлементной арифметики
10. Полносвязный
11. Арифметические операции с константой
12. Функция потерь

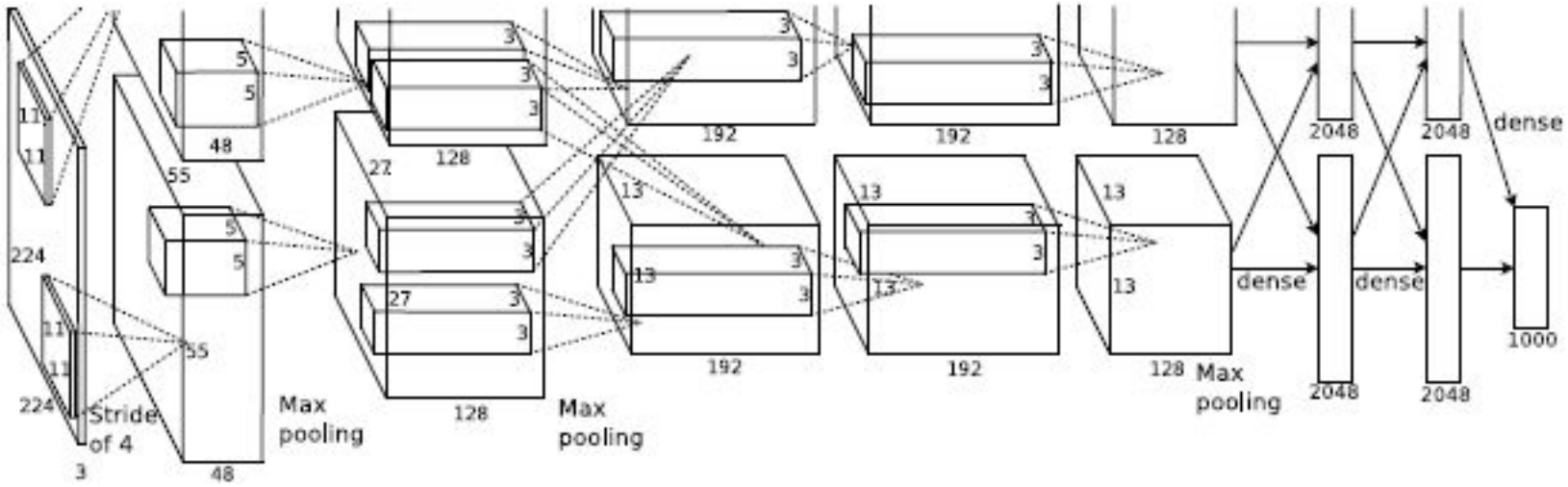
Архитектура ГКНС представляет собой набор слоёв ГКНС, а также последовательность передачи данных от слоя к слою в процессе работы/обучения ГКНС.
Архитектура ГКНС не содержит в себе коэффициентов фильтров и других параметров, подбираемых при обучении.
Архитектура ГКНС обычно не привязана к конкретной задаче, а привязана скорее к типу задач например Классификация, обнаружение объектов, семантическая сегментация

Архитектура ГКНС

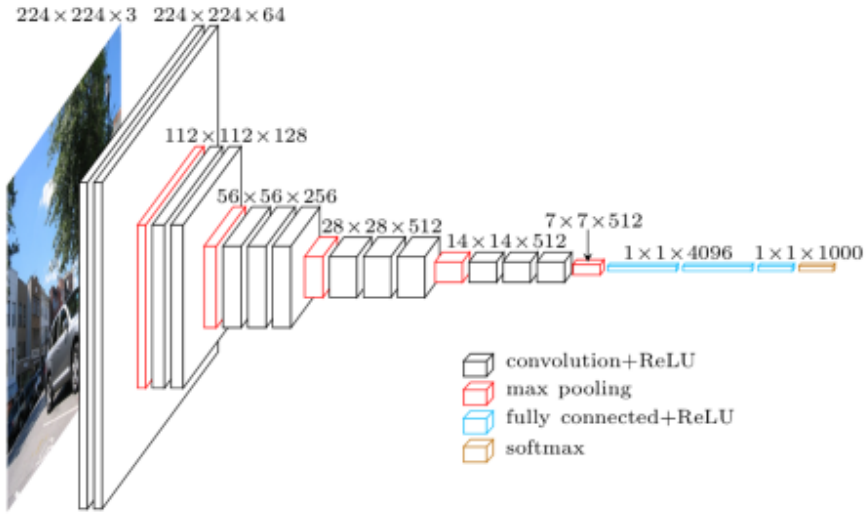
Модуль – набор слоёв и связей между ними, служащий для упрощения процесса задания архитектуры ГКНС.



Архитектура ГКНС



Схематическое представление архитектуры



Архитектура ГКНС

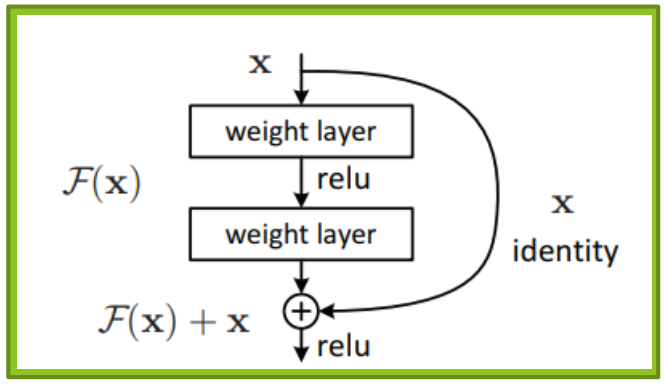
Архитектура как таблица:

Input	64x64
Convolution 1	3x3@16
Convolution 2	3x3@32
Pooling	2x2
Convolution 3	3x3@64
Pooling	2x2
Convolution 4	3x3@64
Pooling	2x2
Fully-connected	500
Fully-connected	2

Layer	Filter size, stride	Output W×H×N
Input	-	$400 \times 400 \times 3$
Conv	10×10, 2	$196 \times 196 \times 80$
ReLU+LRN	-	$196 \times 196 \times 80$
Max-pool	6×6, 4	$49 \times 49 \times 80$
Conv	5×5, 1	$45 \times 45 \times 120$
ReLU+LRN	-	$45 \times 45 \times 120$
Max-pool	3×3, 2	$22 \times 22 \times 120$
Conv	3×3, 1	$20 \times 20 \times 160$
ReLU	-	$20 \times 20 \times 160$
Conv	3×3, 1	$18 \times 18 \times 200$
ReLU	-	$18 \times 18 \times 200$
Max-pool	3×3, 2	$9 \times 9 \times 200$
FC	-	320
ReLu+Drop	-	320
FC	-	320
ReLu+Drop	-	320
FC	-	Dataset dependent
Softmax	-	Dataset dependent

Архитектура ГКНС

Архитектура как схема + таблица(прежде всего для модульных сетей):

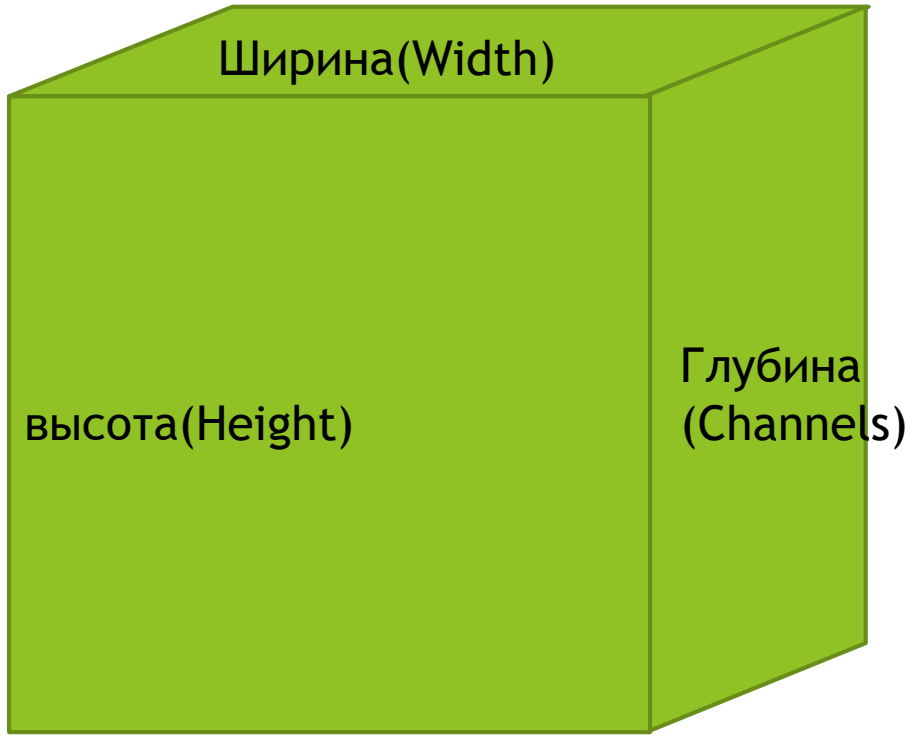


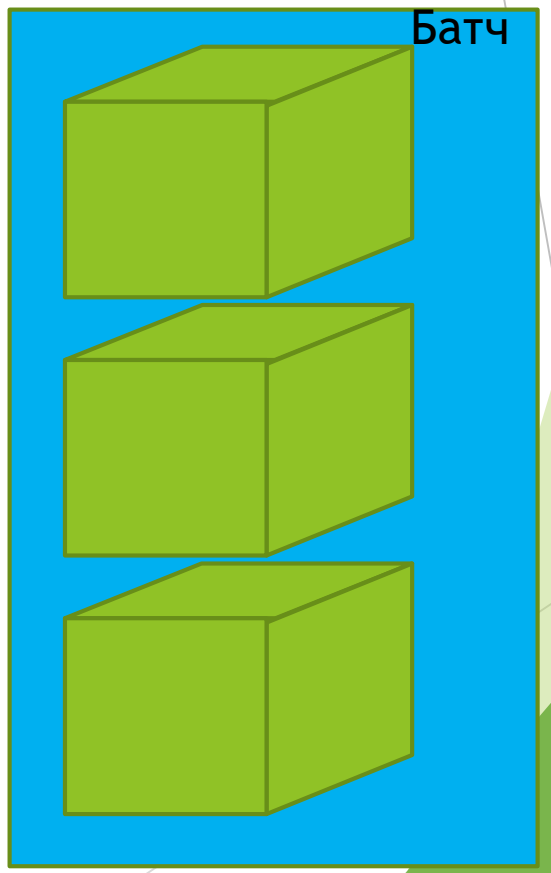
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Термины и определения

Батч - набор изображений по которым проводится обработка

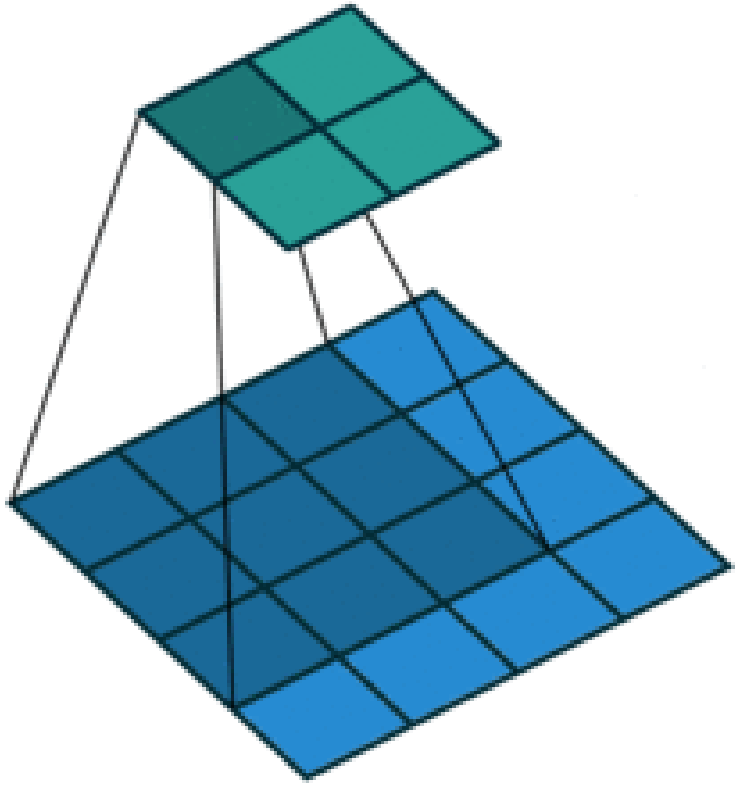
Блоб – данные являющиеся либо входными либо выходными для слоев





Блоб – это тензор ранга 4. Т.е. четырёхмерный массив данных

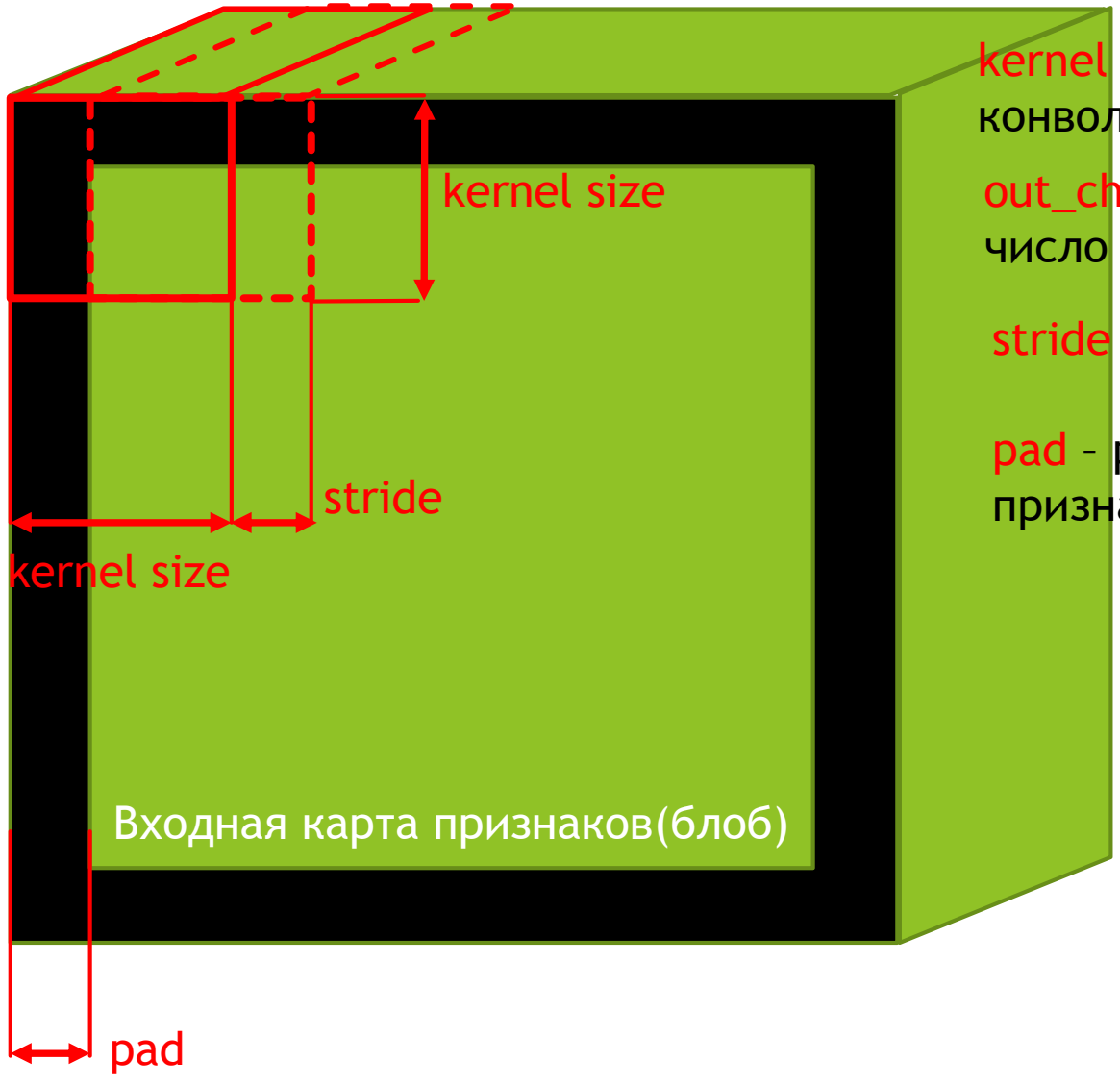
Архитектура ГКНС



Конволюционный(сверточный) слой

Архитектура ГКНС

Конволюционный(сверточный) слой:



kernel size(filter size) – размер ядра фильтра конволюционного слоя(обычно фильтры квадратные)

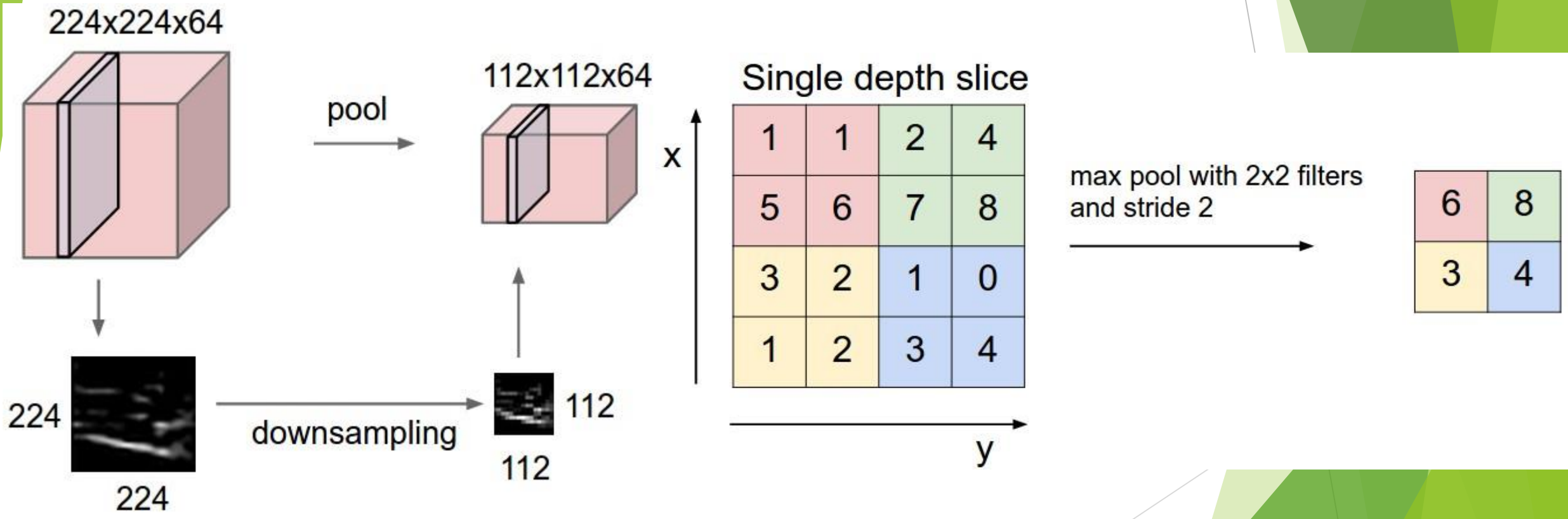
out_channels, num_output, filter_cnt – число фильтров в слое

stride – шаг фильтра

pad – размер полей добавляемых к входной карте признаков перед обработкой

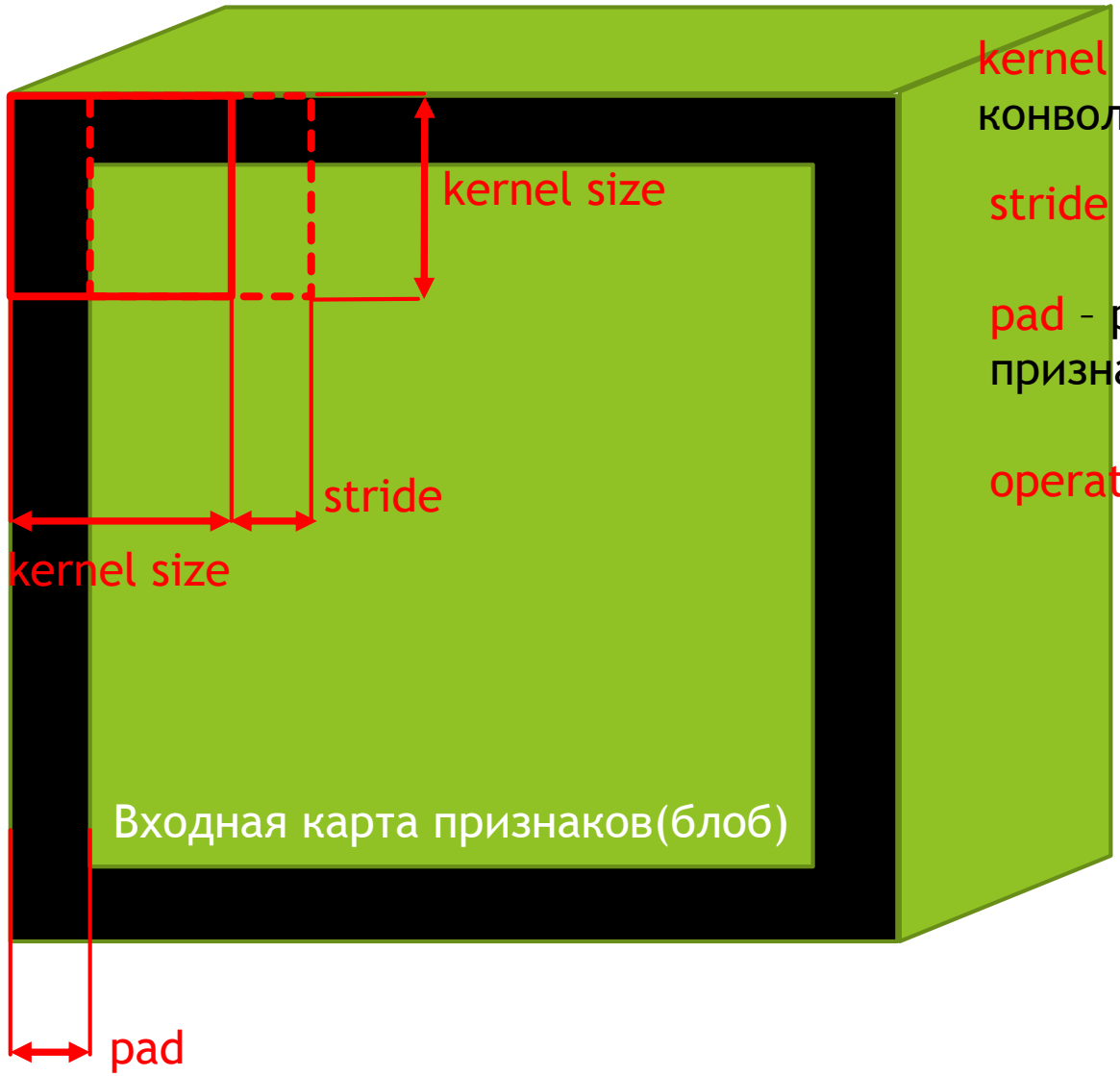
Архитектура ГКНС

Пулинг



Архитектура ГКНС

Слой пулинга:



kernel size(filter size) – размер ядра фильтра конволюционного слоя(обычно фильтры квадратные)

stride – шаг фильтра

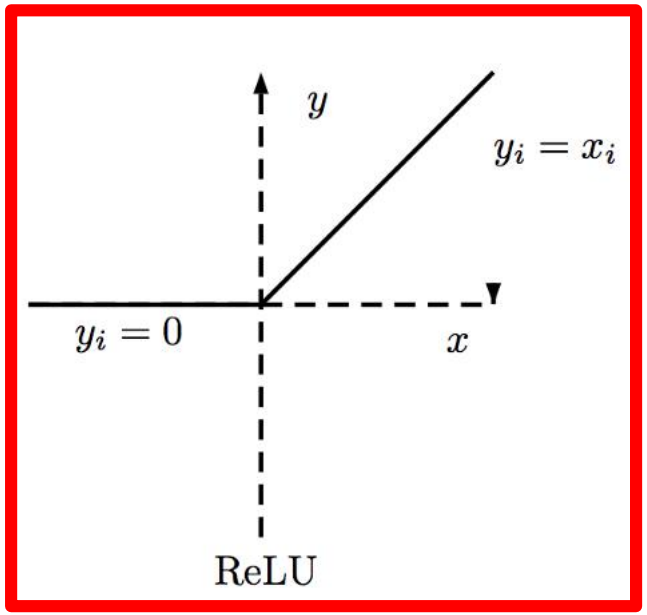
pad – размер полей добавляемых к входной карте признаков перед обработкой

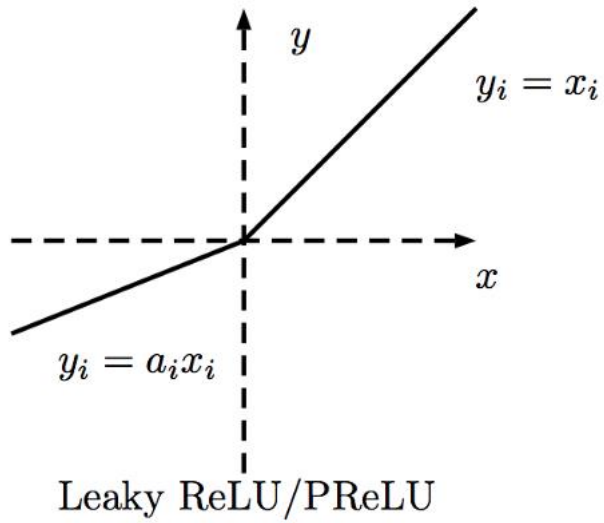
operator – тип пулинга:

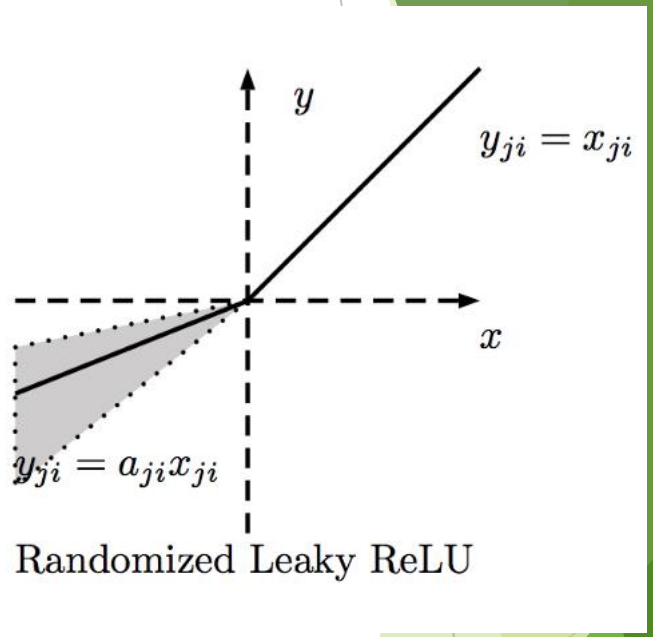
1. по максимуму
2. по среднему

Архитектура ГКНС

Слой нелинейности:





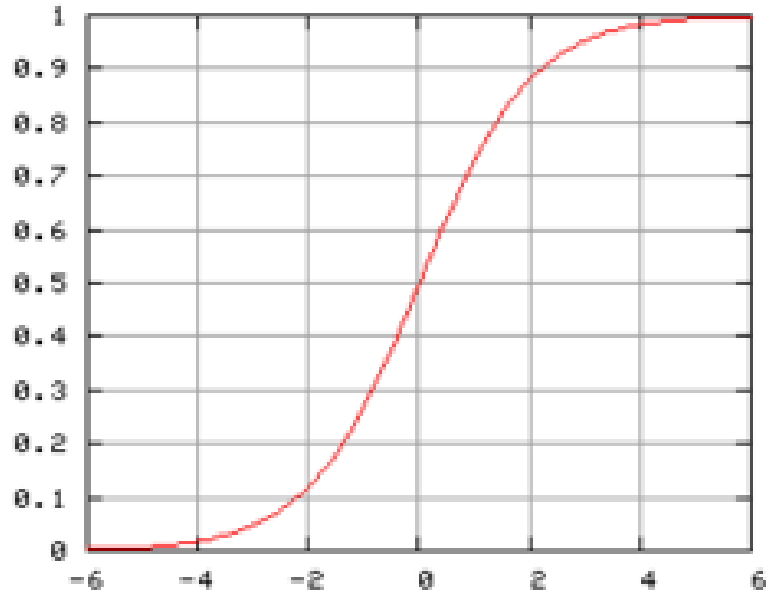


Применяет к каждому элементу входной карты признаков нелинейное преобразование

Параметры – параметры нелинейного преобразования. Для ReLU параметров нет.

Архитектура ГКНС

Слой нелинейности:

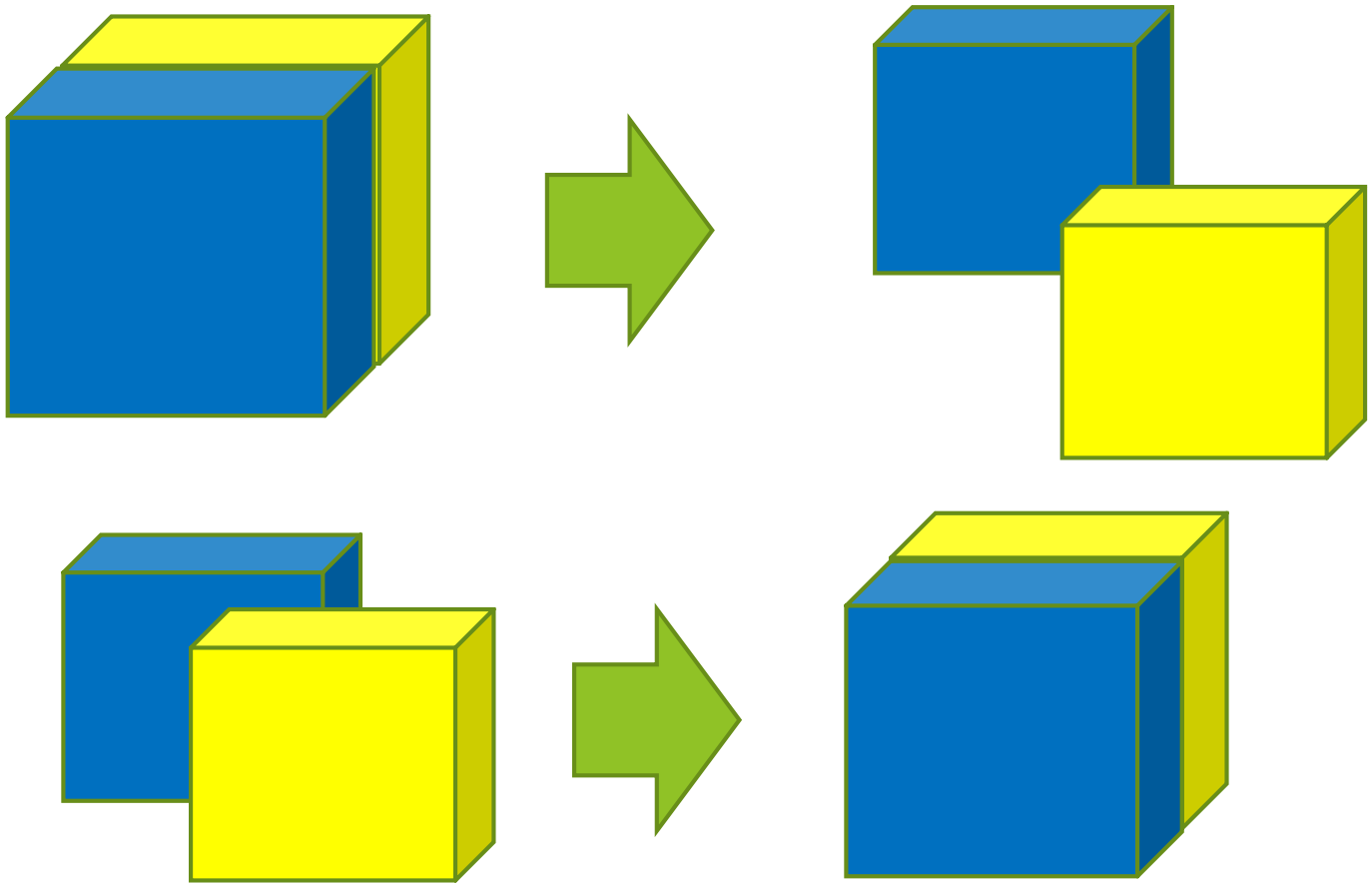


$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Сигмойда вход любой – выход [0..1]
Параметров нет

Архитектура ГКНС

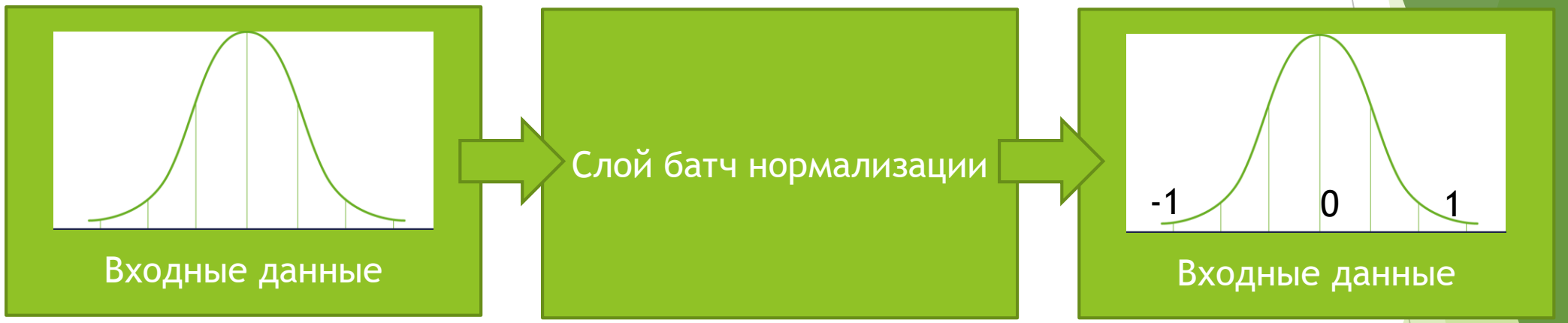
Слой конкатенации/разделения:



Разделяет входную карту признаков на два и более частей.
Объединяет несколько карт признаков в одну

Архитектура ГКНС

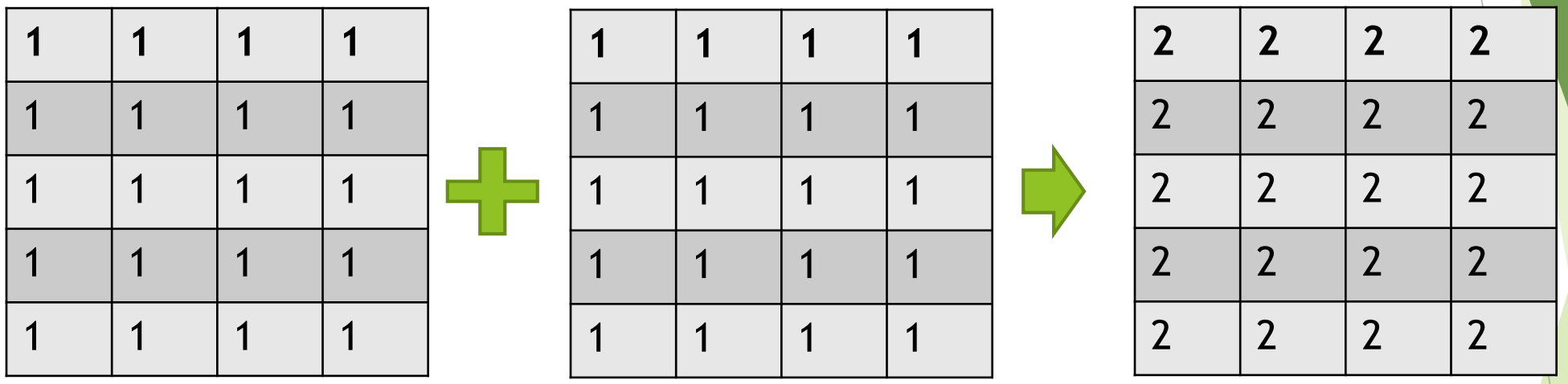
Слой батч нормализации:



Слой батч нормализации нормирует входные данные – в простейшем случае это вычитание МО и деление на СКО. Слой повышает скорость обучения сети и делает сеть менее чувствительной к начальной инициализации. В процессе обучения нормировочные коэффициенты рассчитываются внутри батча. В процессе работы сети после обучения нормировочные коэффициенты заменяются средними по обучающей выборке.

Архитектура ГКНС

Слой поэлементной арифметики:



Слой реализует поэлементное умножение/сложение двух входных карт признаков(блобов). Выходом слоя является результат операции. Входные карты признаков должны быть одного размера.

Архитектура ГКНС

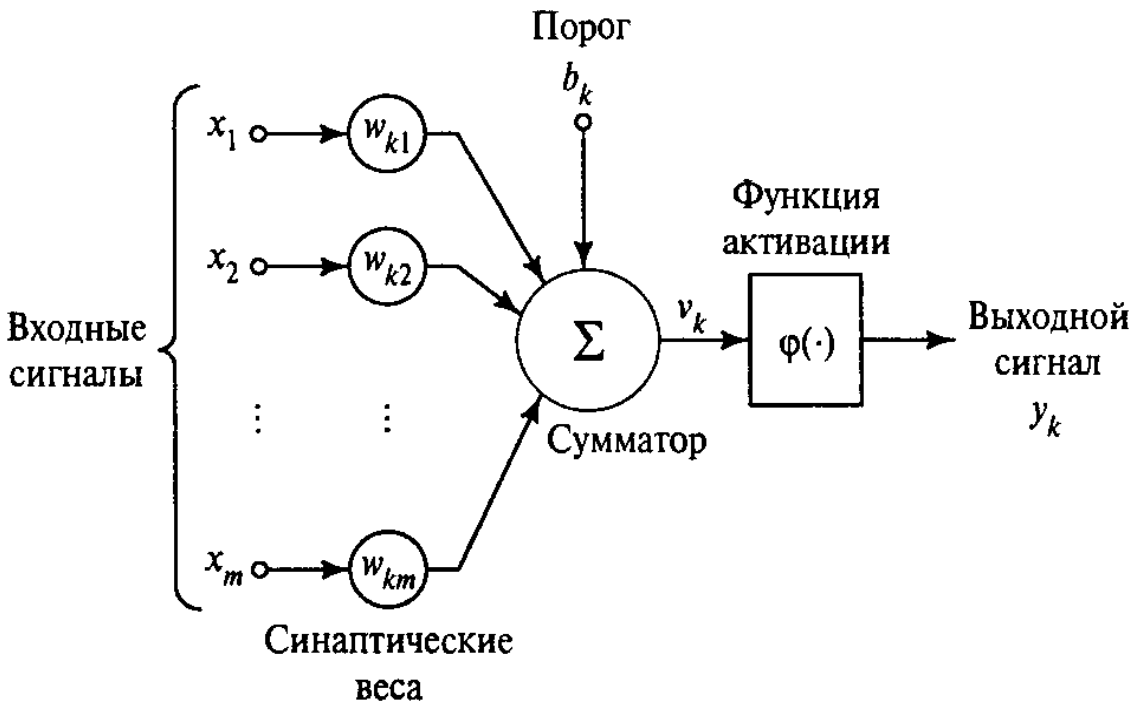
Арифметические операции с константой :

$$A * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{B} + C$$

Слой реализует умножение каждого значения входной карты признаков на число, возведение в степень, сложение с константой.

Архитектура ГКНС

Полносвязный слой: Linear, Inner Product(IP), Fully Connected(FC)

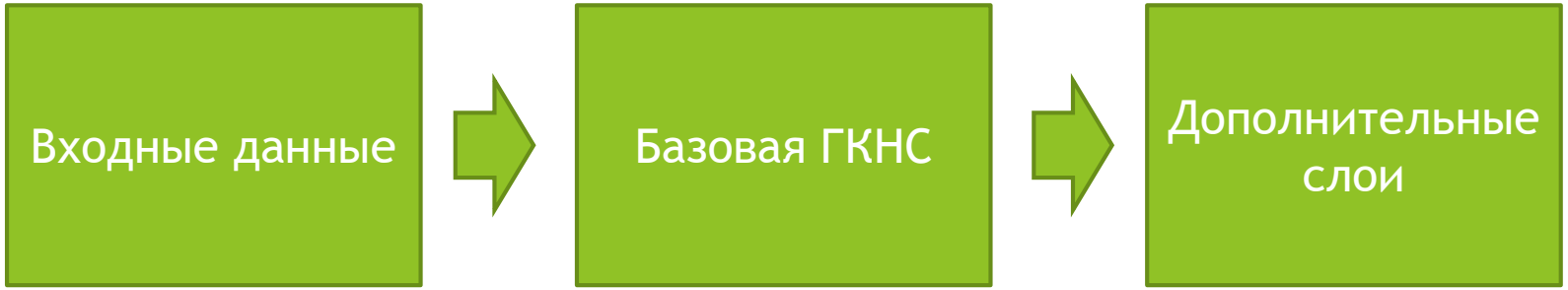


num_output, out_channels – число нейронов

Bias – выбор типа нейрона с порогом/без порога

Набор классических нейронов. Выходом слоя является вектор. Входным вектором является входная карта признаков в не зависимости от ее формы.

Базовые ГКНС



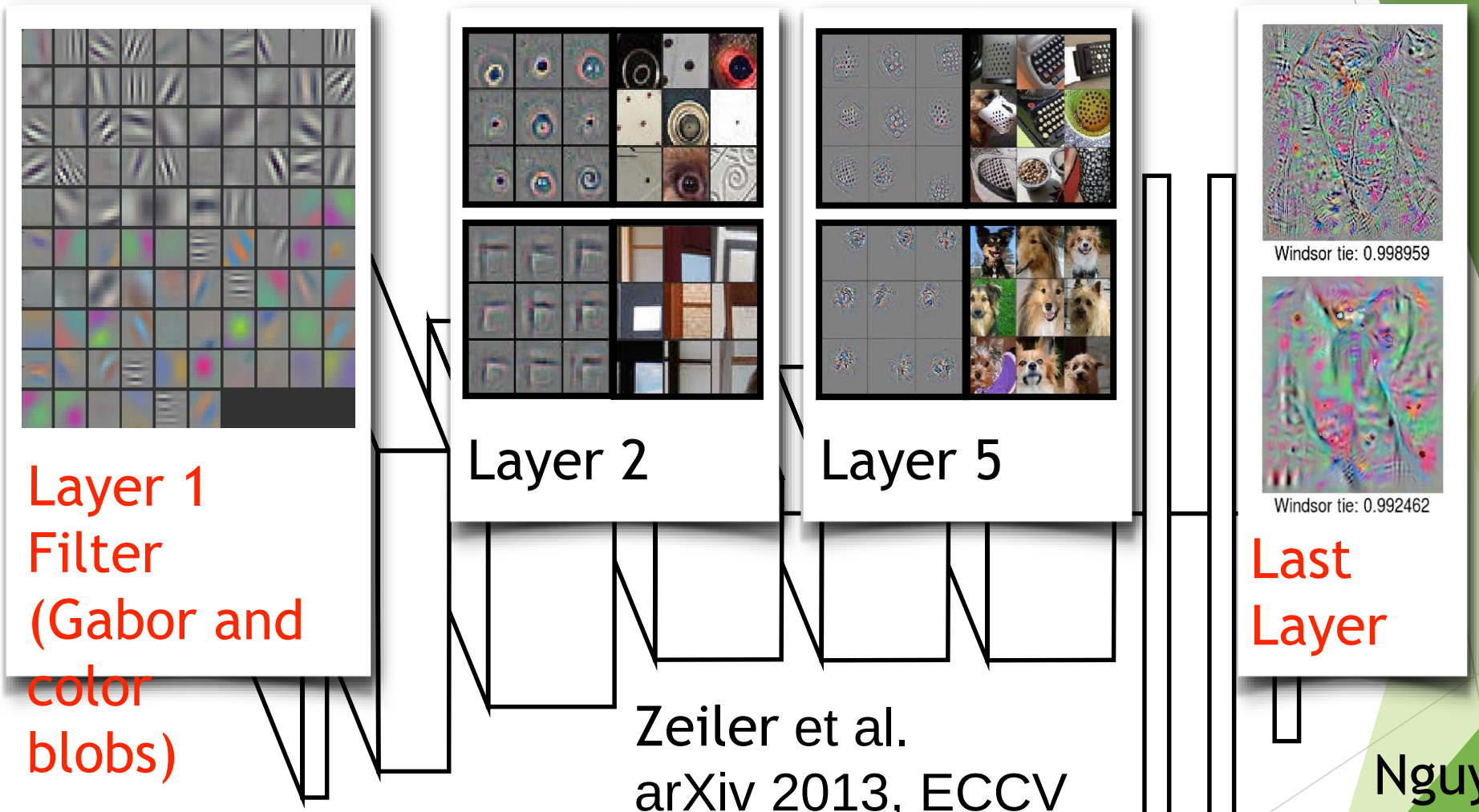
Обычно при решении практических задач используются т.н. базовые ГКНС.

1. Базовые ГКНС обычно разработаны для решения задачи классификации по ImageNet

2. Для решения финальной задачи к базовой ГКНС добавляются дополнительные слои, однако при этом сама базовая ГКНС обычно практически не меняется

3. Для базовых ГКНС доступны готовые наборы весов с которых удобно проводить дообучение

2. Fine-tuning

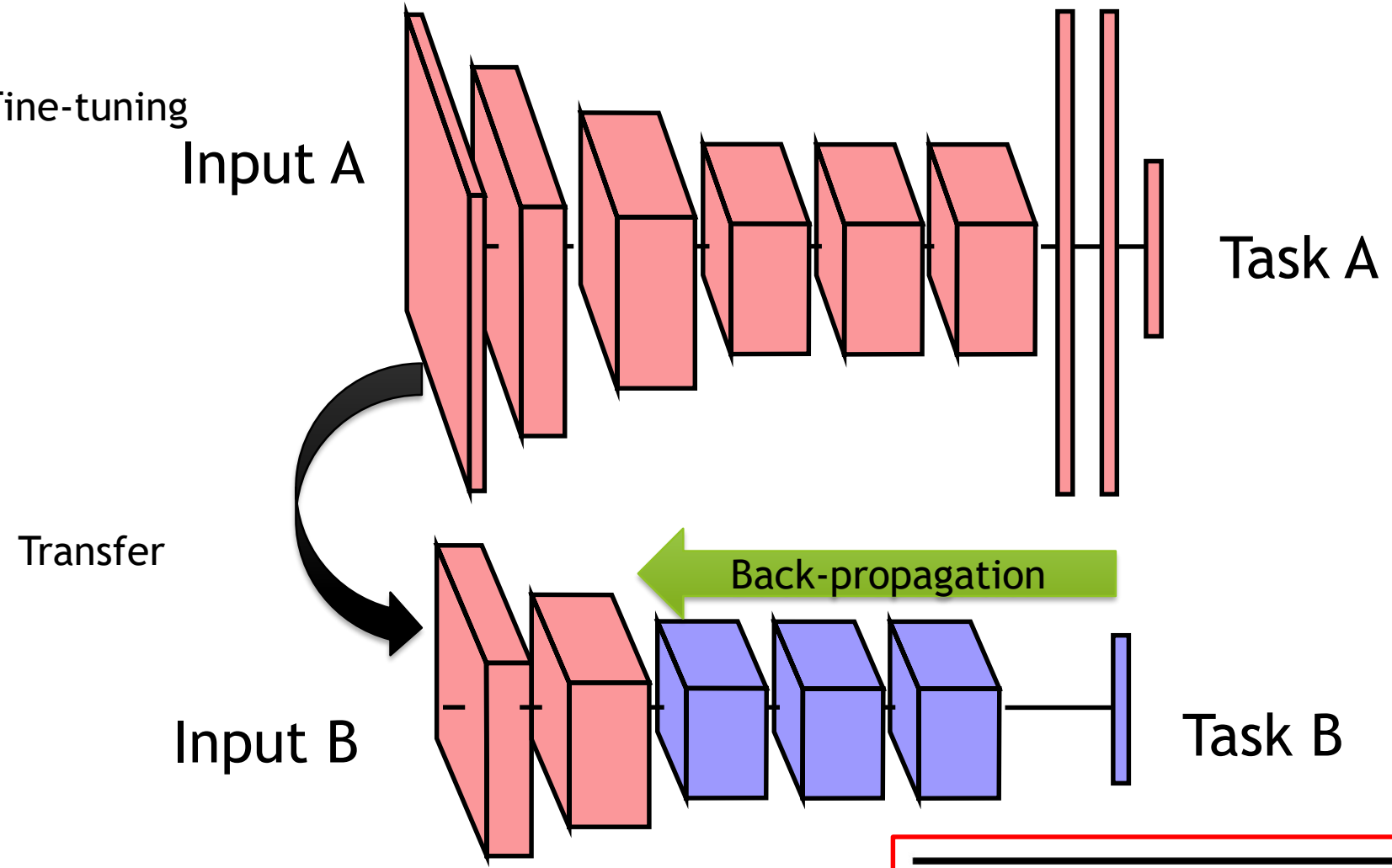


Zeiler et al.
arXiv 2013, ECCV 2014

Nguyen et al.
arXiv 2014

Gabor filter: linear filters used for edge detection with similar orientation representations to the human visual system

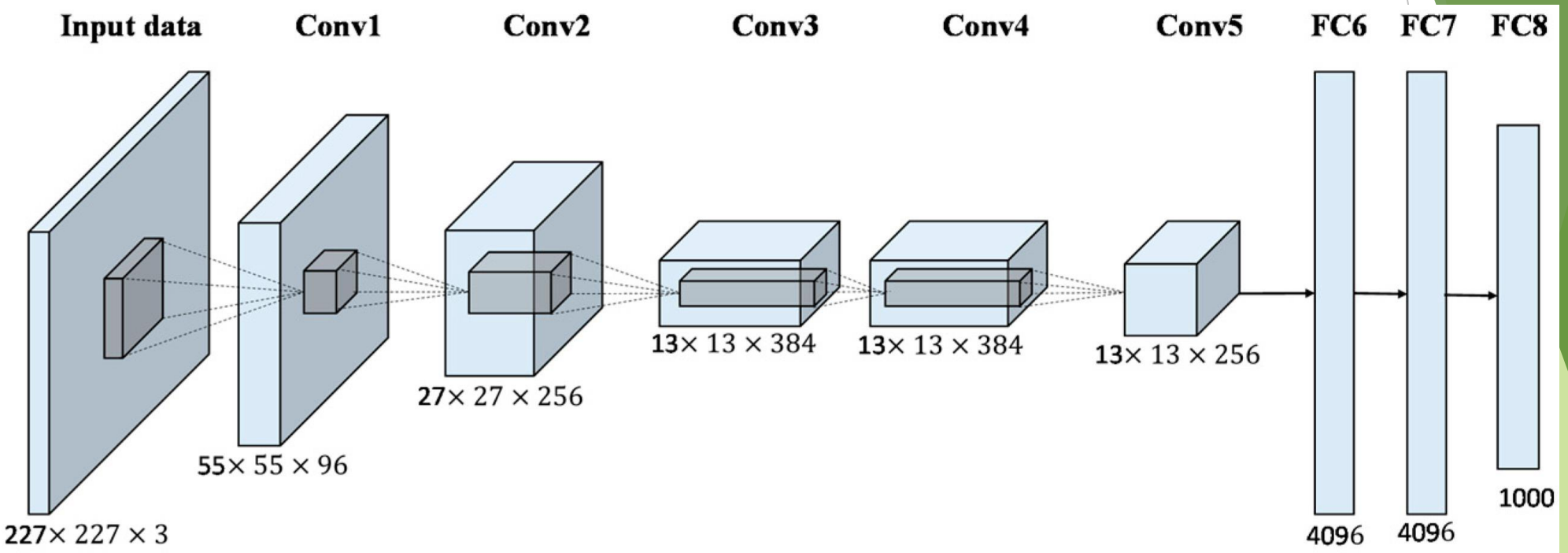
2. Fine-tuning



How transferable are features in deep neural networks?

Jason Yosinski,[1] **Jeff Clune,**[2] **Yoshua Bengio,**[3] **and Hod Lipson**[4]
[1] Dept. Computer Science, Cornell University
[2] Dept. Computer Science, University of Wyoming
[3] Dept. Computer Science & Operations Research, University of Montreal
[4] Dept. Mechanical & Aerospace Engineering, Cornell University

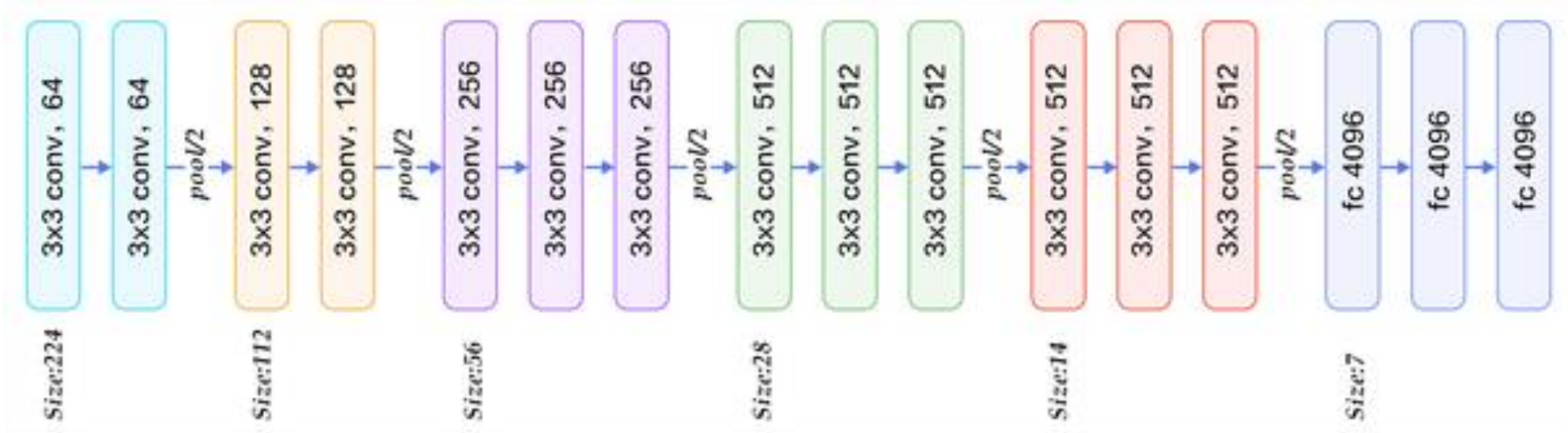
AlexNet



Предложена в 2012 году Алексом Крживецким, по сути с этой ГКНС началась революция в машинном обучении. Часто используется в статьях по аппаратной реализации.

https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf

VGG



Предложена в 2014 году в рамках исследования по максимальной глубине и архитектуре ГКНС.
Особенности:

- Конволюции только 3x3
- До 19 слоев

Karen Simonyan, Andrew Zisserman VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNIT
https://arxiv.org/pdf/1409.1556.pdf

VGG

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

До сих пор активно используется в практических задачах(напр. SSD,GAN и т.д).

Используются только VGG-16 и VGG-19

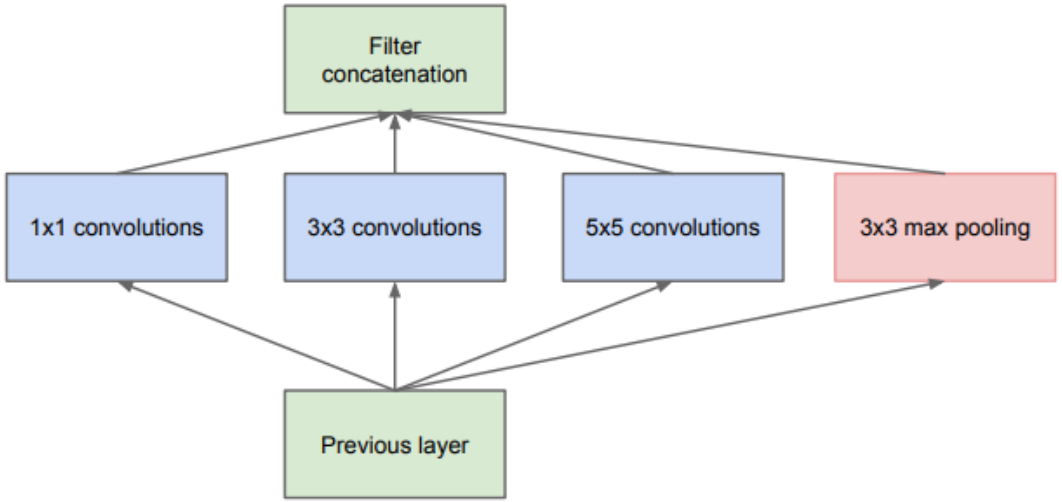
Дальше 19 слоев не учится!

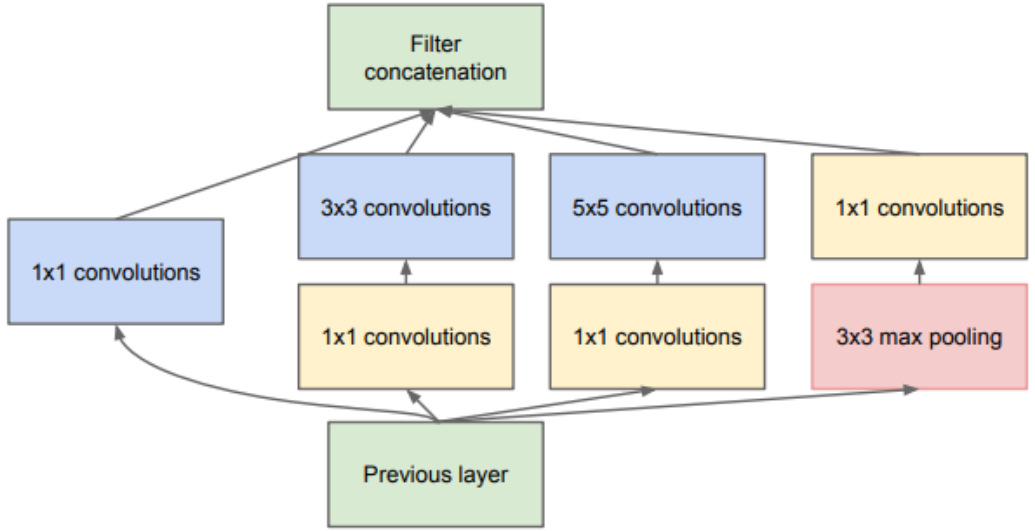
Медленная

Хорошо сжимается и упрощается

Karen Simonyan, Andrew Zisserman VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNIT
https://arxiv.org/pdf/1409.1556.pdf

Google Net





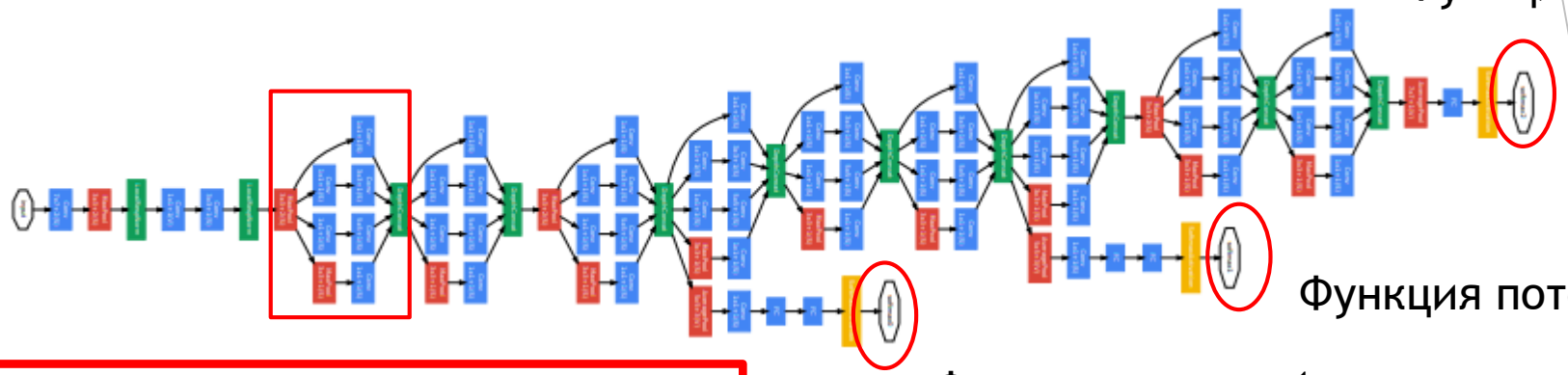
Предложена в 2014 году, основная идея использовании широких модулей и особой методике обучения.

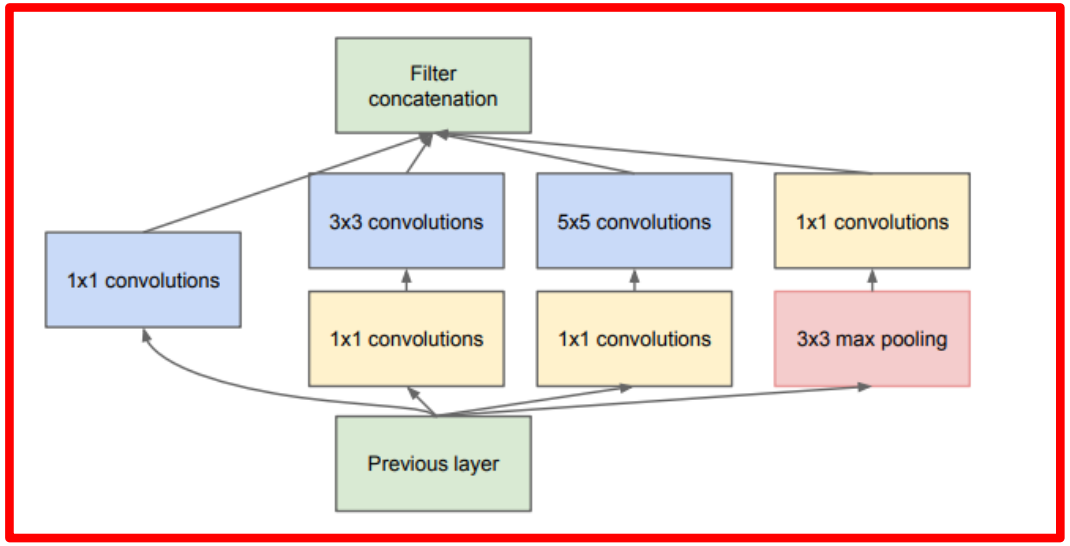
Особенности:

1. Одна из первых модульных сетей
2. Использование многих функций потерь

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going deeper with convolutions
https://arxiv.org/pdf/1409.1556.pdf

Google Net





Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going deeper with convolutions
https://arxiv.org/pdf/1409.1556.pdf

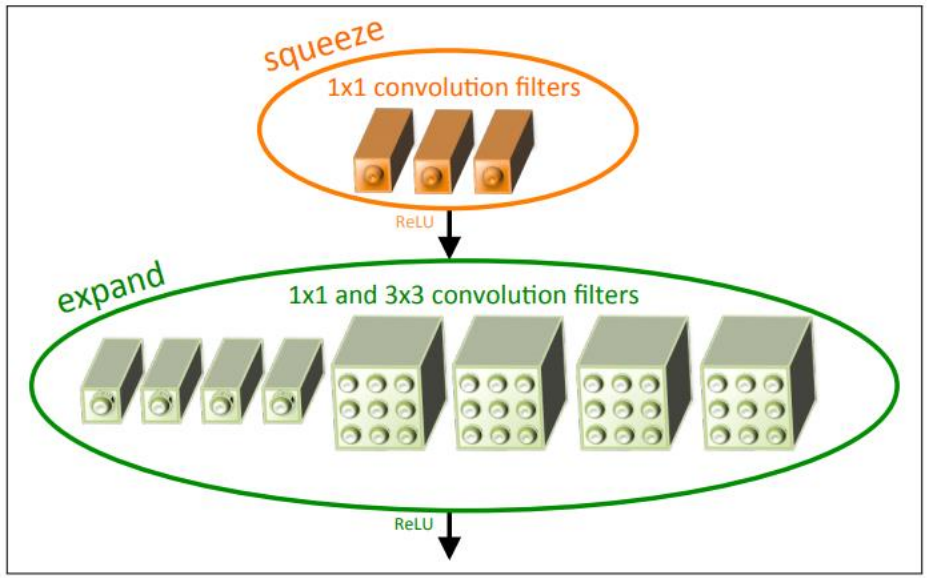
Google Net

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

В настоящий момент не используется

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going deeper with convolutions
https://arxiv.org/pdf/1409.1556.pdf

Squeeze Net

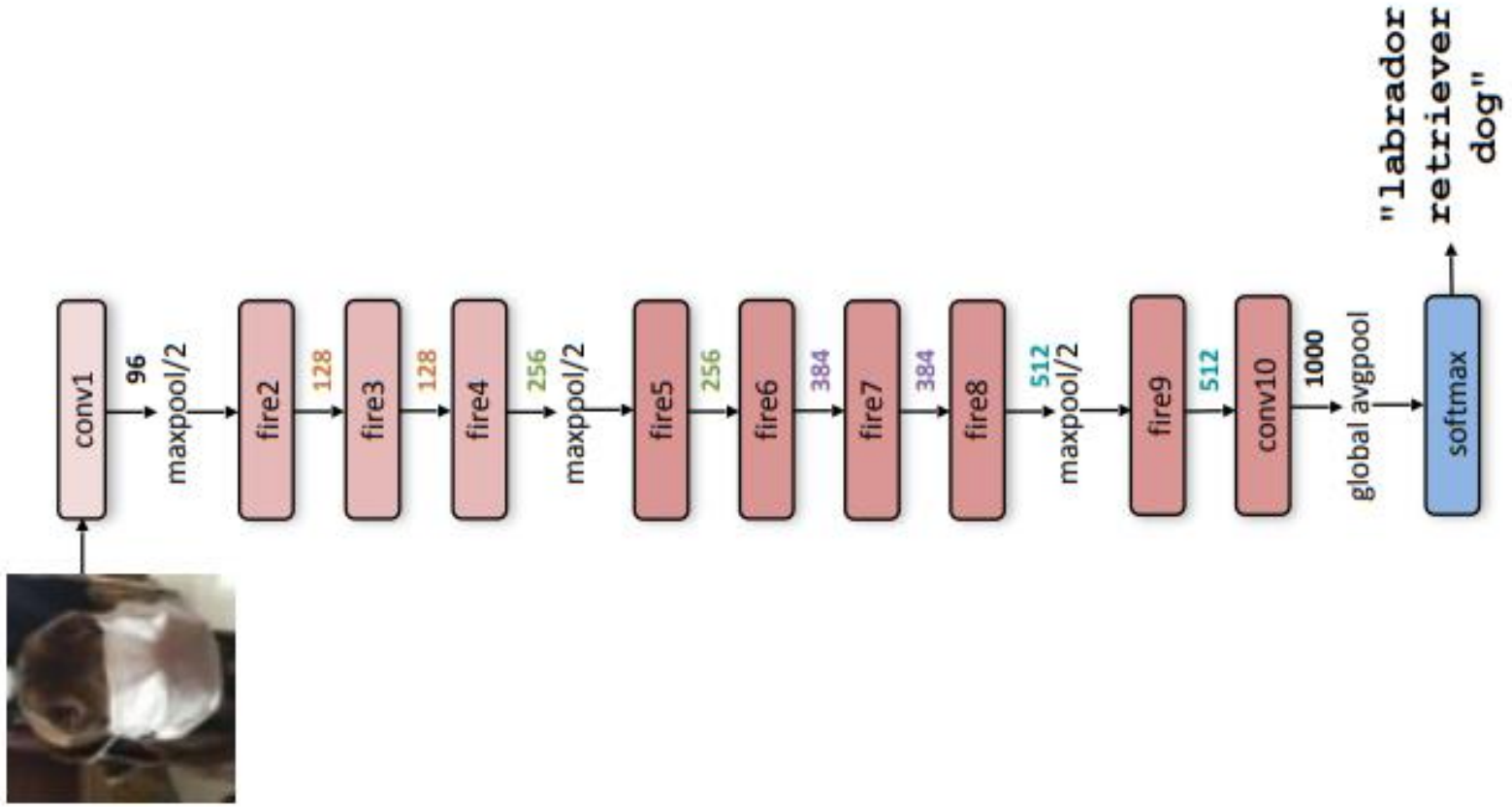


Предложена в 2016 году, задача максимального ускорения и упрощения ГКНС для портирования.
Особенности:

Использование модулей Fire module.

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer
SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size
https://arxiv.org/pdf/1602.07360.pdf

Squeeze Net



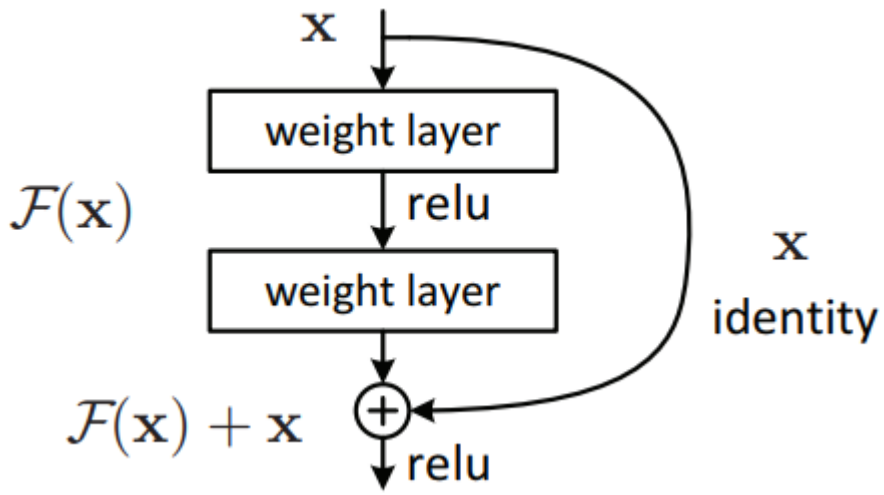
Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer
SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size
https://arxiv.org/pdf/1602.07360.pdf

Squeeze Net

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations			parameters			compression info				1,248,424 (total)	**421,098** (total)

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer
SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size
https://arxiv.org/pdf/1602.07360.pdf

ResNet



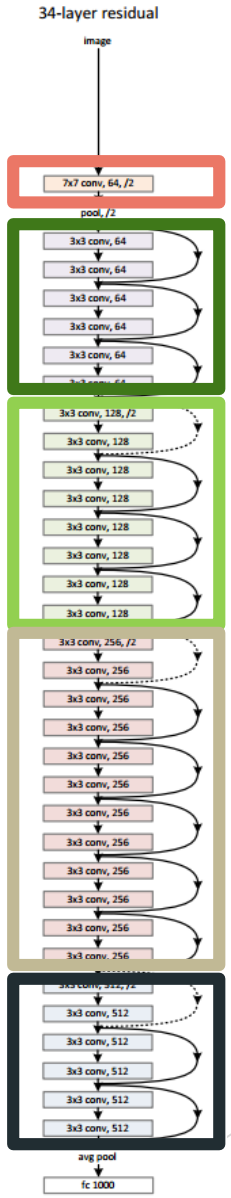
Предложена в 2015 году, основная идея использовании резидуальных блоков.
Особенности:

- Свёртки только 3x3 за исключением первого слоя
- Резидуальные блоки
- Неограниченное число слоев (ResNet 1001)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition
https://arxiv.org/pdf/1512.03385.pdf

ResNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



блок 1

снижение размерности

блок 2

снижение размерности

блок 3

снижение размерности

блок 4

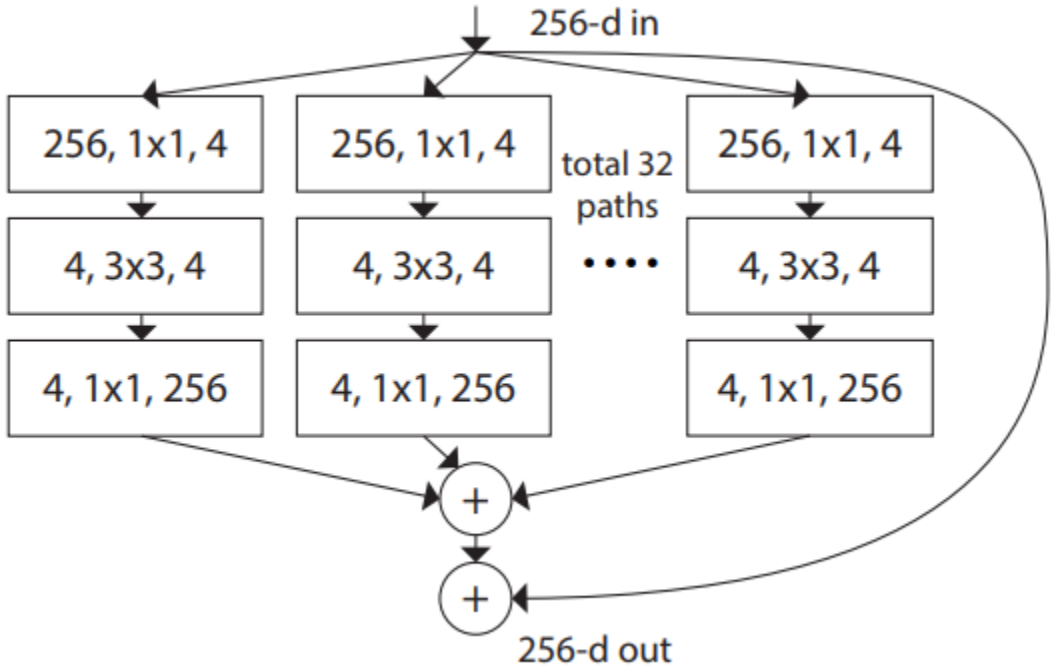
снижение размерности

блок 5

Одна из наиболее популярных сетей.
Актуальна до сих пор.
Хорошо изучена и очень стабильна.
Не сильно уступает state of the art.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition
https://arxiv.org/pdf/1512.03385.pdf

ResNeXT

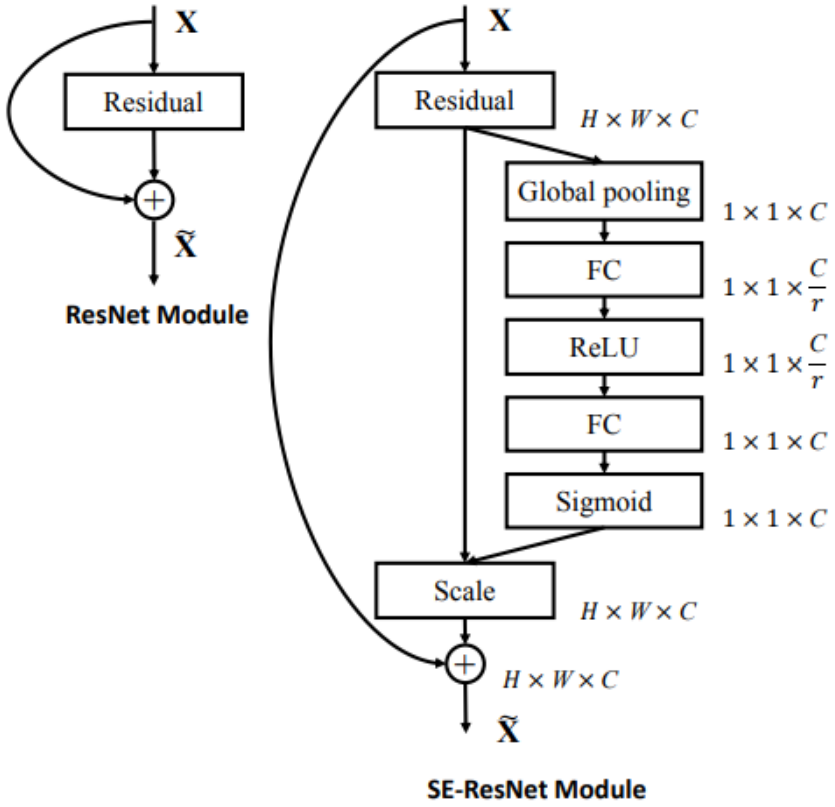


Предложена в 2017-ом году.
Одна из state of the art.
Широко распростанена
Если хочется улучшить resnet -> resnext

Синтез идей GoogleNet и ResNet

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He Aggregated Residual Transformations for Deep Neural Networks https://arxiv.org/pdf/1611.05431.pdf

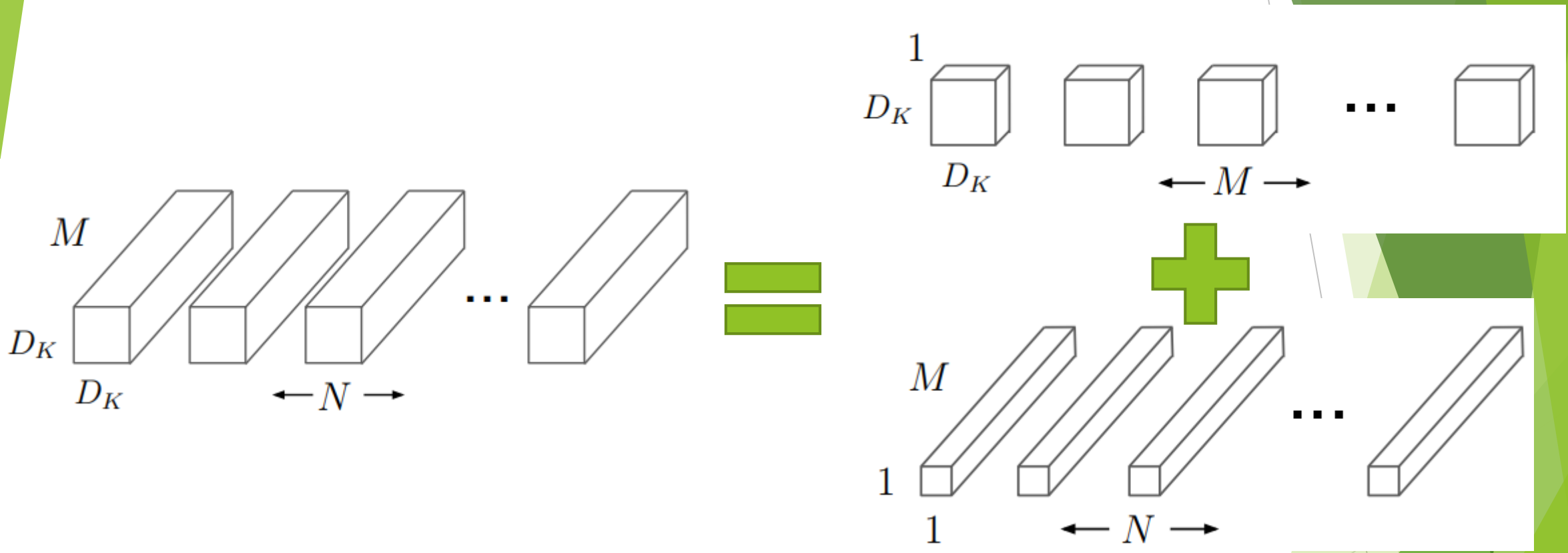
SE ResNet



Предложена в 2018-ом году.
Одна из state of the art.
Участвует в конкурсах
Идея использование слоя поканального “забывания”
Существует вариант Inception

By Jie Hu, Li Shen, Gang Sun Squeeze-and-Excitation Networks
https://arxiv.org/pdf/1611.05431.pdf

MobileNet



Предложена в 2017-ом году.
Основная идея замена N конволюционных слоев 3x3xM на M поканальных сверток 3x3 и N сверток 1x1xM.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications https://arxiv.org/pdf/1704.04861.pdf

MobileNet

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

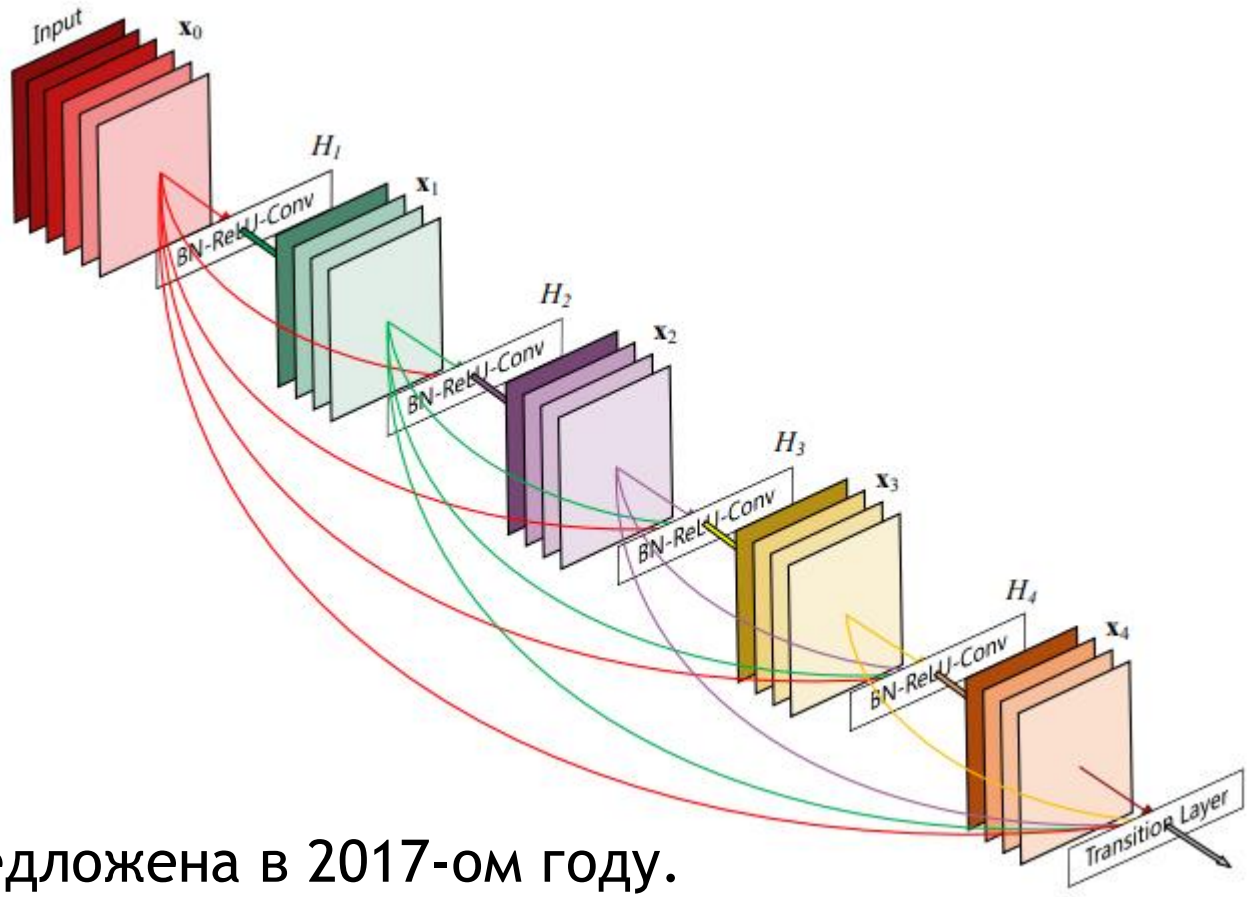
Table 8. MobileNet Comparison to Popular Models

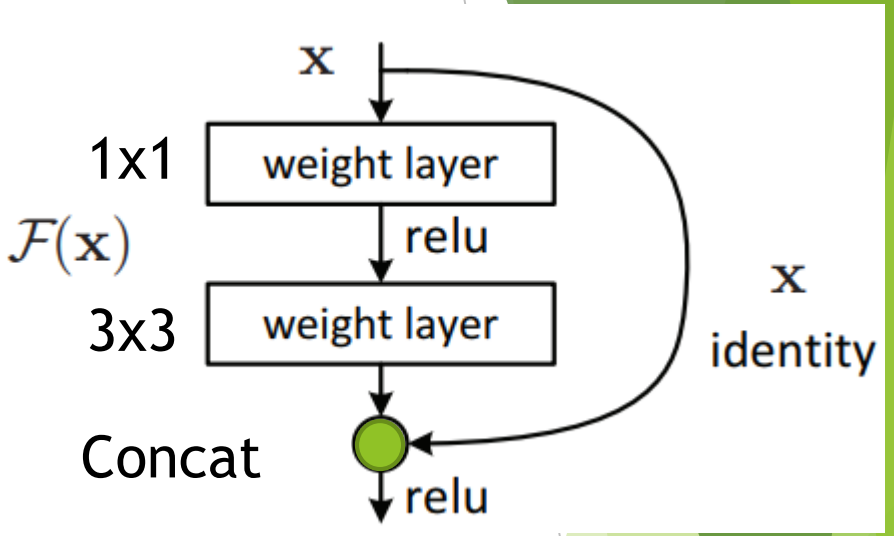
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Часто используется в практических работах
Необходима низкоуровневая поддержка архитектуры.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications https://arxiv.org/pdf/1704.04861.pdf

Dense Net





Предложена в 2017-ом году.
Одна из state of the art по маленьким изображениям
Использование конкатенации вместо сложения

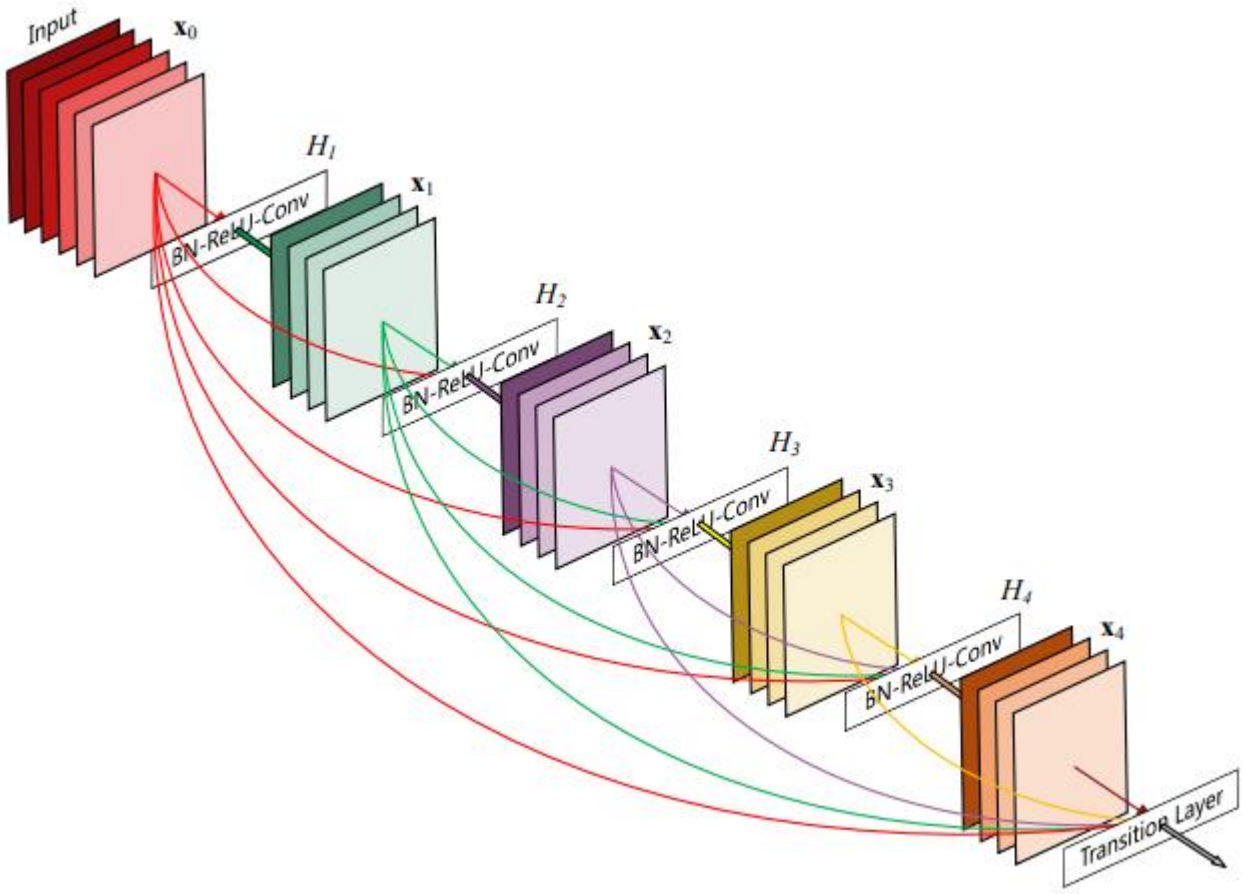
Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger **Densely Connected Convolutional Networks** https://arxiv.org/pdf/1608.06993.pdf

Dense Net

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger **Densely Connected Convolutional Networks** https://arxiv.org/pdf/1608.06993.pdf

Dense Net



Мало сверток.
Не теряются данные
Быстро растет глубина

Не особо популярные

Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger **Densely Connected Convolutional Networks** https://arxiv.org/pdf/1608.06993.pdf