

SOTA in Computer Vision

Наталья Ханжина
ML Lab
CV DL Senior Researcher
PhD student



Роман Лебедев
JetBrains Research
CV DL Intern Researcher
PhD student

План лекции

- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA model zoo
 - Регуляризация
 - Наборы данных
- Детекция изображений
 - Модули
 - Виды детекции и SOTA model zoo
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA model zoo
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

План лекции

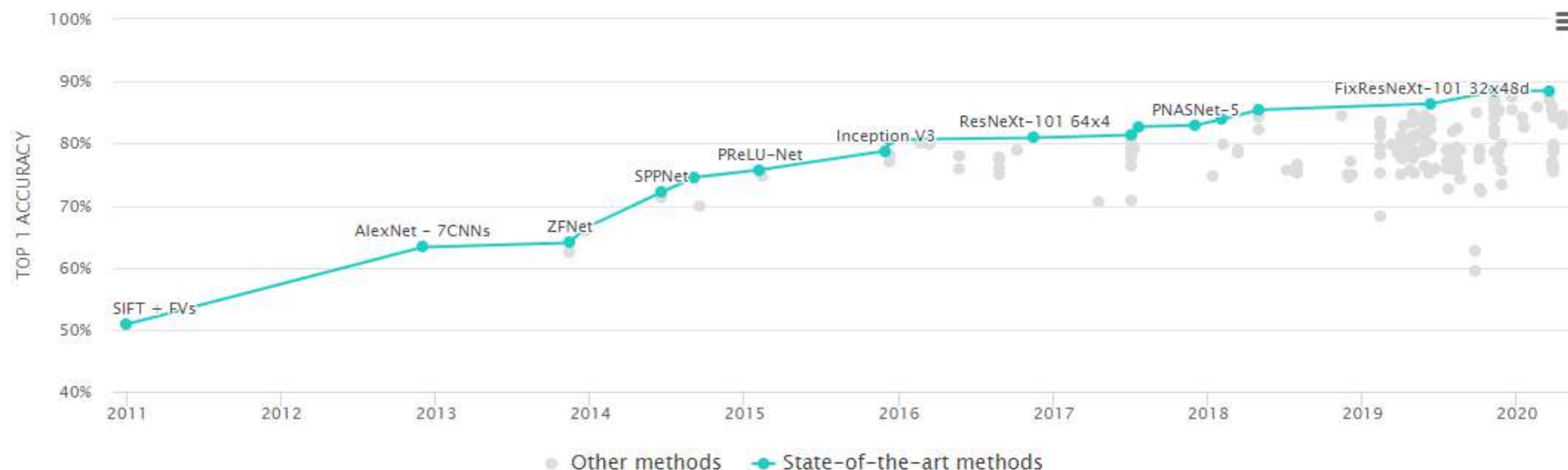
- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA model zoo
 - Регуляризация
 - Наборы данных
- Детекция изображений
 - Модули
 - Виды детекции и SOTA model zoo
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA model zoo
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

Short history of computer vision

1957	Персептрон Розенблата
1974	Многослойный персептрон и Back propagation algorithm (Rumelhart, Werbos, Galushkin)
1978	Распознавание образов (Gonzalez)
1980	Convolutional NN (Fukushima)
1998	LeNet : Gradient descent for convolutional NN (LeCun et al.)
2006	Deep belief nets (Hinton, Osindero and Teh)
2007	Глубокое обучение (Hinton)

Modern history of computer vision

- 2012 Архитектура **AlexNet** (Hinton, Krizhevsky, and Sutskever) выиграла ImageNet. С этого началась современная история компьютерного зрения
- 2015 Архитектура **ResNet** превзошла человеческий уровень ошибки на ImageNet

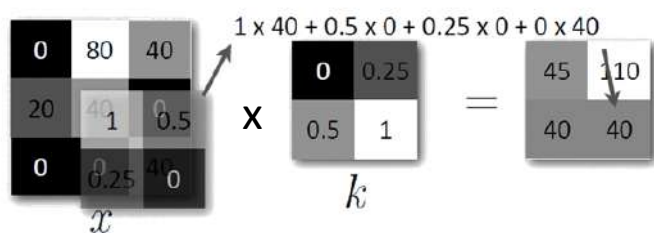


План лекции

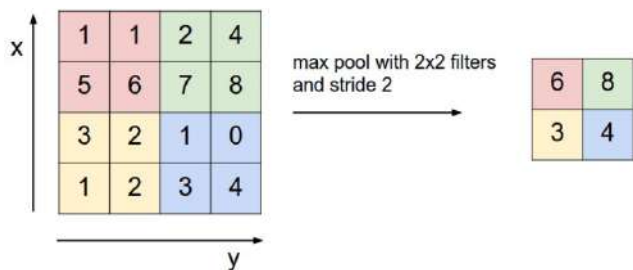
- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA модели
 - Регуляризация
 - Наборы данных
- Детекция изображений
 - Модули
 - Виды детекции и SOTA модели
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA модели
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

Базовые слои

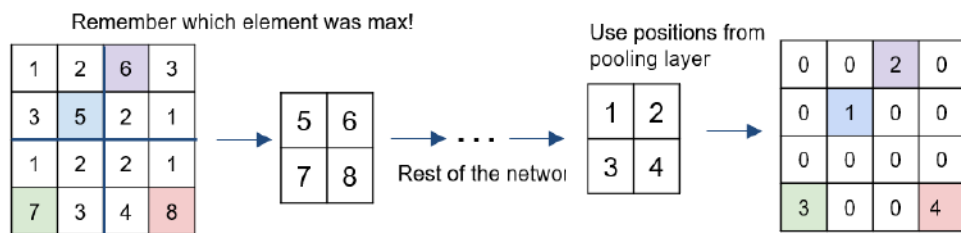
- Convolution



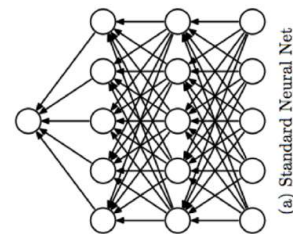
- Pooling



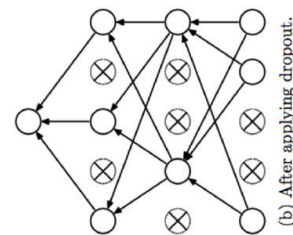
- UpSampling



- Dense (fully-connected)



- Dropout



- Batch normalization

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

// normalize

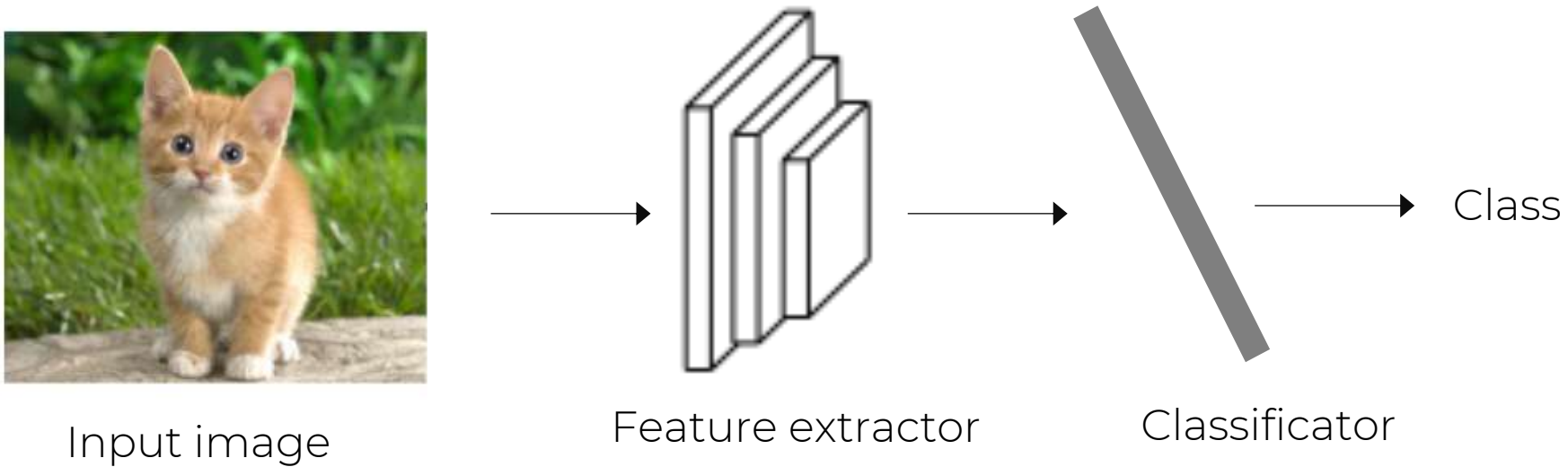
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$$

// scale and shift

Module

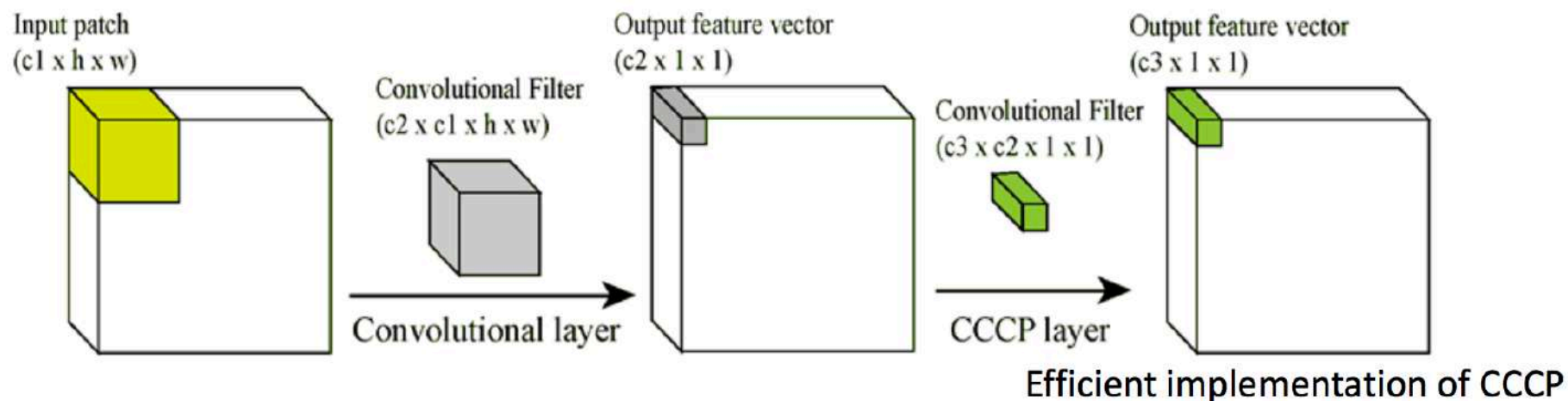
- Модуль, или блок – структурная единица современных моделей нейросетей
- Каждый модуль – объект или функция $a = h(x; \theta)$, которая
 - Имеет настраиваемые параметры(θ)
 - В качестве аргумента принимает входной сигнал x
 - Возвращает выходное значение a , пропущенное через функцию активации h
 - Функция активации должна иметь (по крайней мере) производные первого порядка (почти) везде
 - Обучается с помощью **chain rule**
 - Может состоять из базовых слоев (операций)

Basic classification model



1x1 convolution

- Свертка 1x1 часто используется в сверточных модулях, она нужна для контроля размерности промежуточных тензоров



- Также известна как Cascaded Cross-Channel pooling (CCCP)

Residual block (2015)

- Стакать слои не всегда помогает
- А если добавить не слой, а остаточный сигнал?

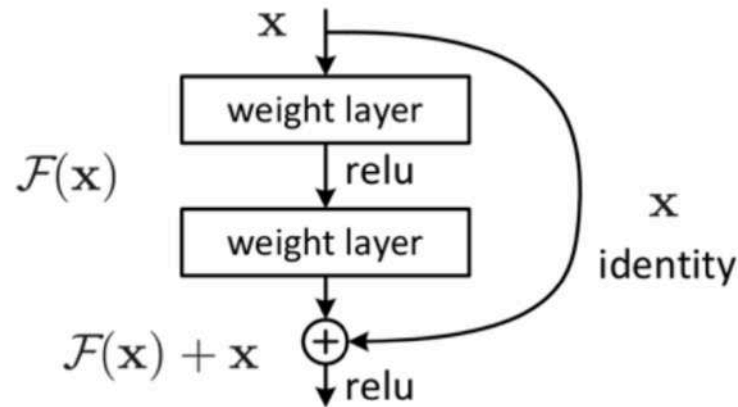


Figure 2. Residual learning: a building block.

$\mathcal{H}(x)$ is the true function we want to learn

Let's pretend we want to learn

$$\mathcal{F}(x) := \mathcal{H}(x) - x$$

instead.

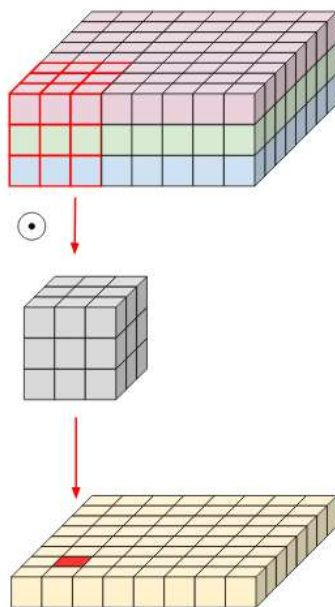
The original function is then

$$\mathcal{F}(x) + x$$

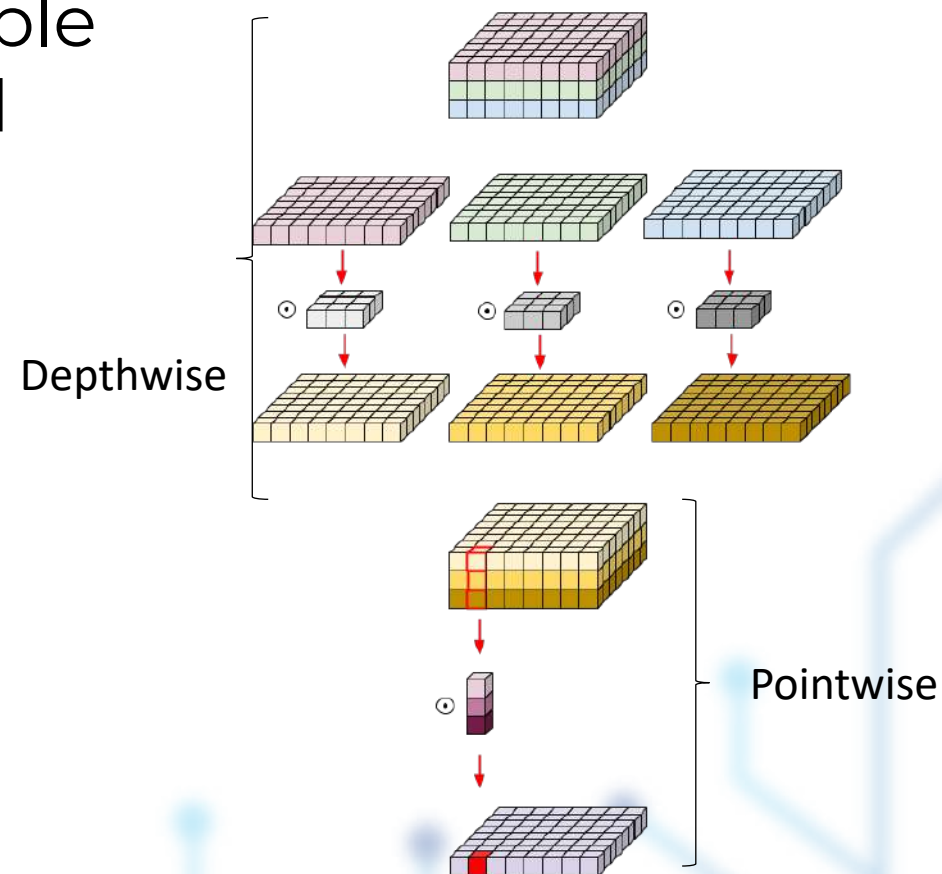
- ResNets

Depth-wise Separable Convolution (2017)

Conv2d



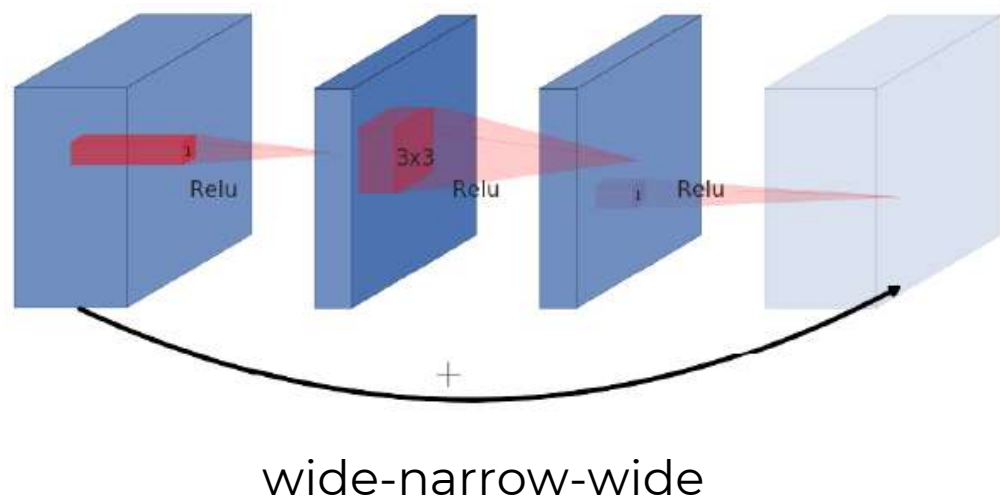
Separable
Conv2d



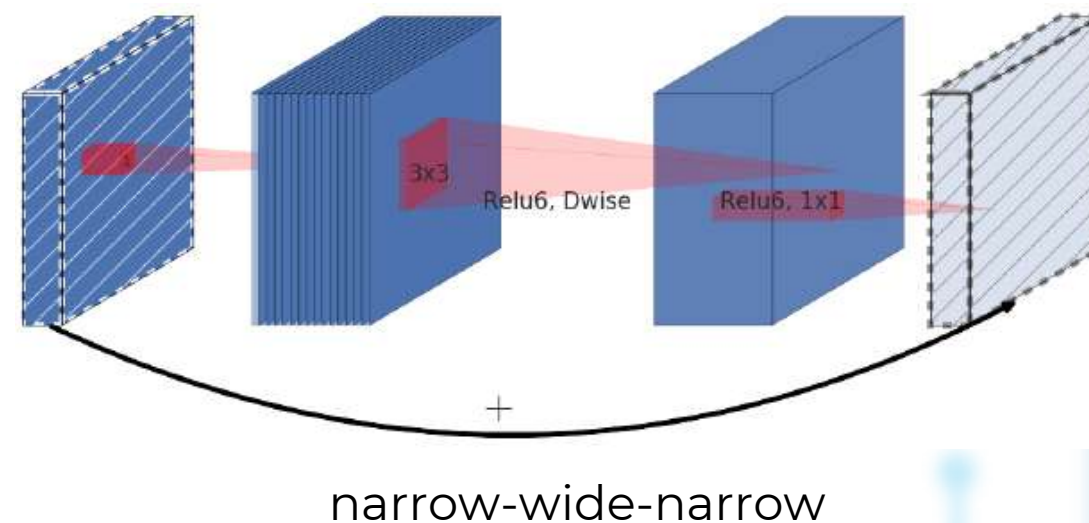
- В три раза меньше обучаемых параметров \min
- MobileNet, Xception

Residual Inverted block (2018)

Residual block



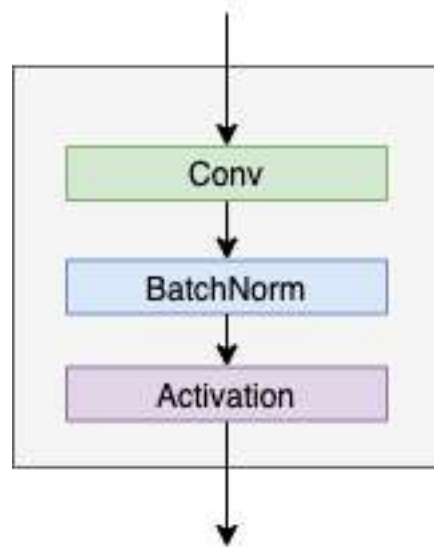
Inverted Residual block



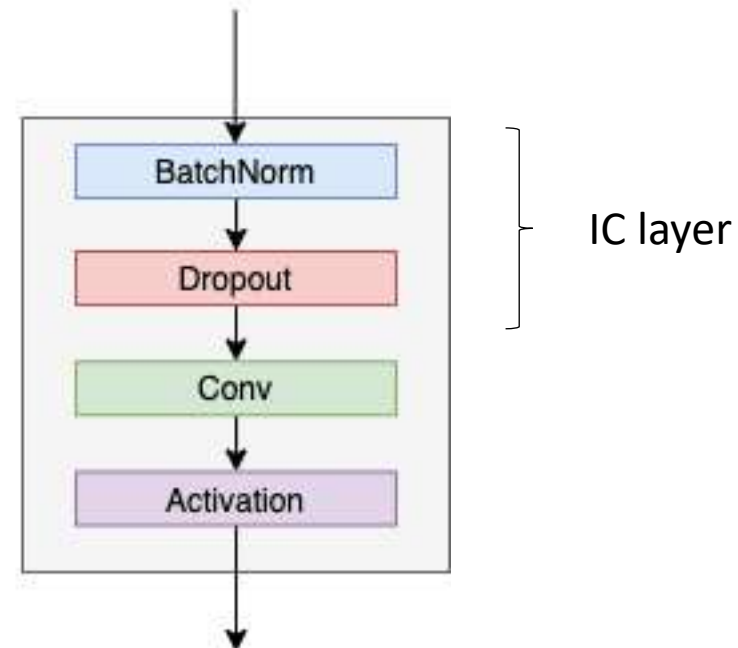
- Меньше параметров
- MobileNetV2, EfficientNet

Independent-Component (IC) layer (2019)

BatchNorm conv
block



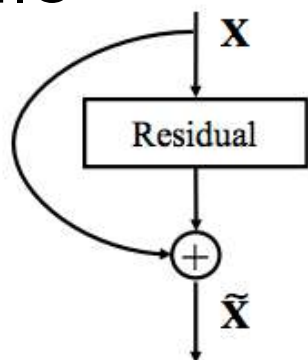
IC block



- +3% accuracy на CIFAR100
- Tested on ResNet

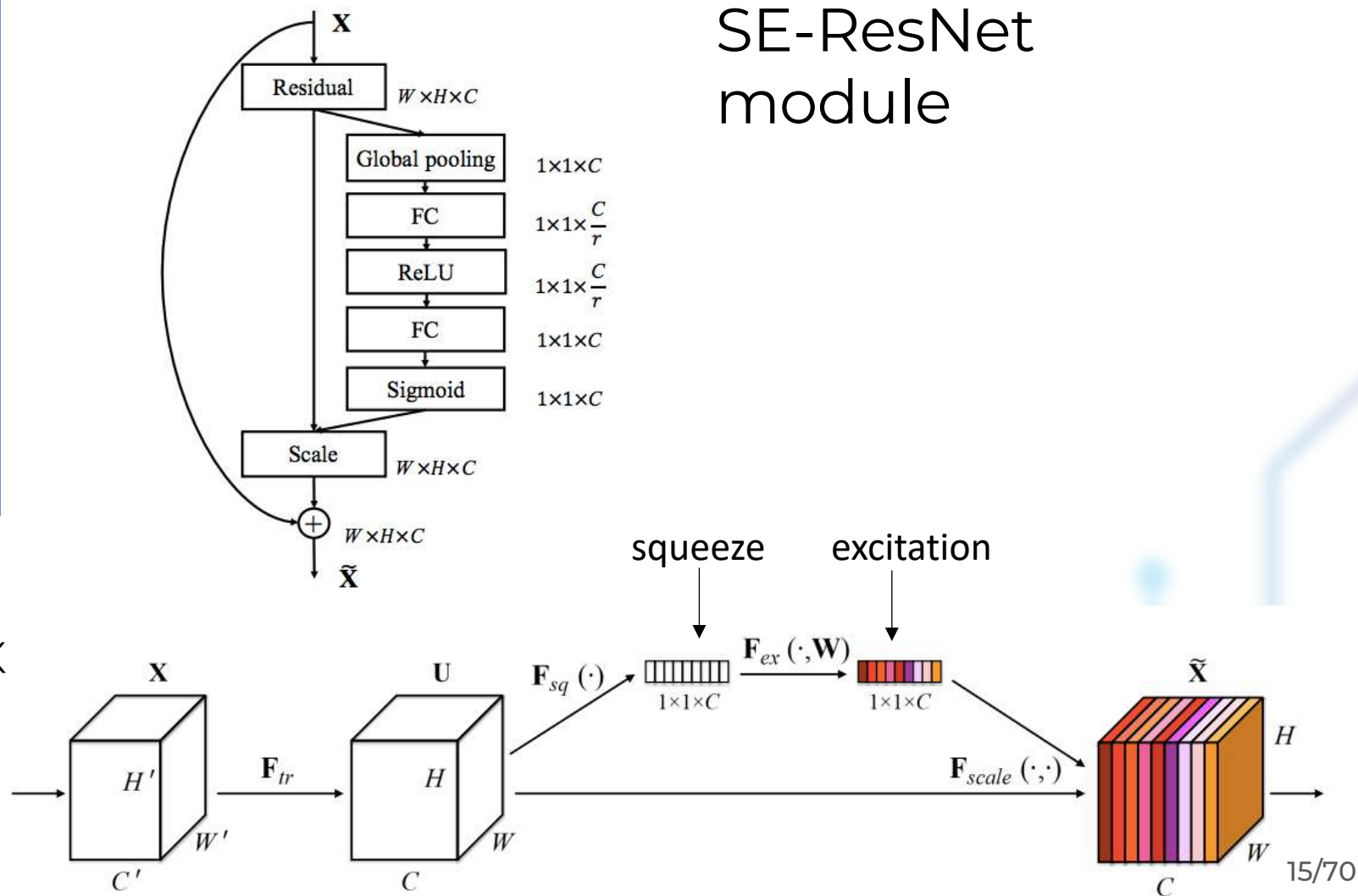
Squeeze-and-Excitation block (2017)

ResNet module



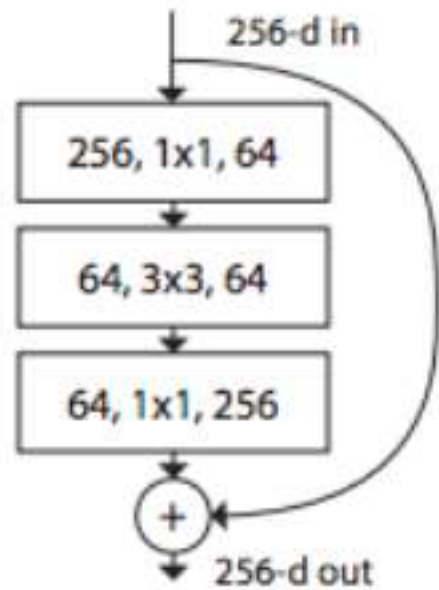
- Attention для картинок
- 25% boost on ImageNet
- SENets, EfficientNet DeepSEED...

SE-ResNet module

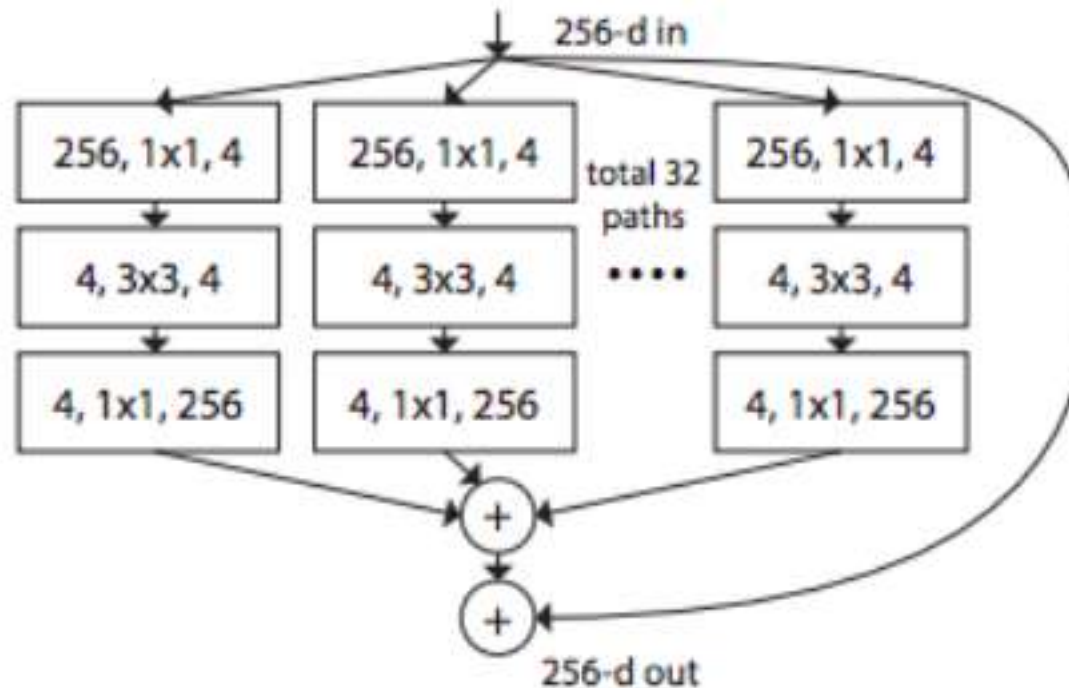


ResNeXt (2016)

ResNet block

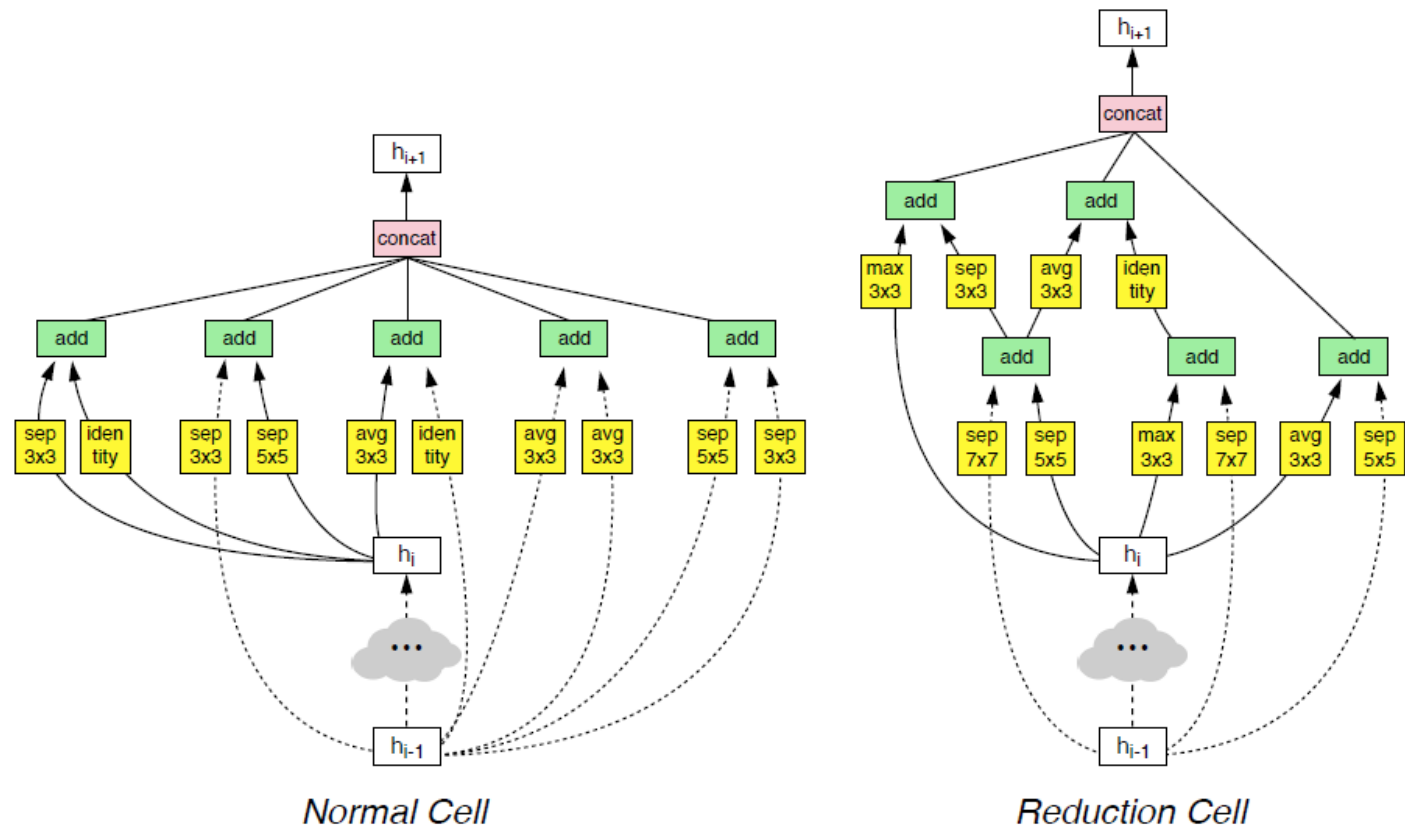


ResNeXt shortcut

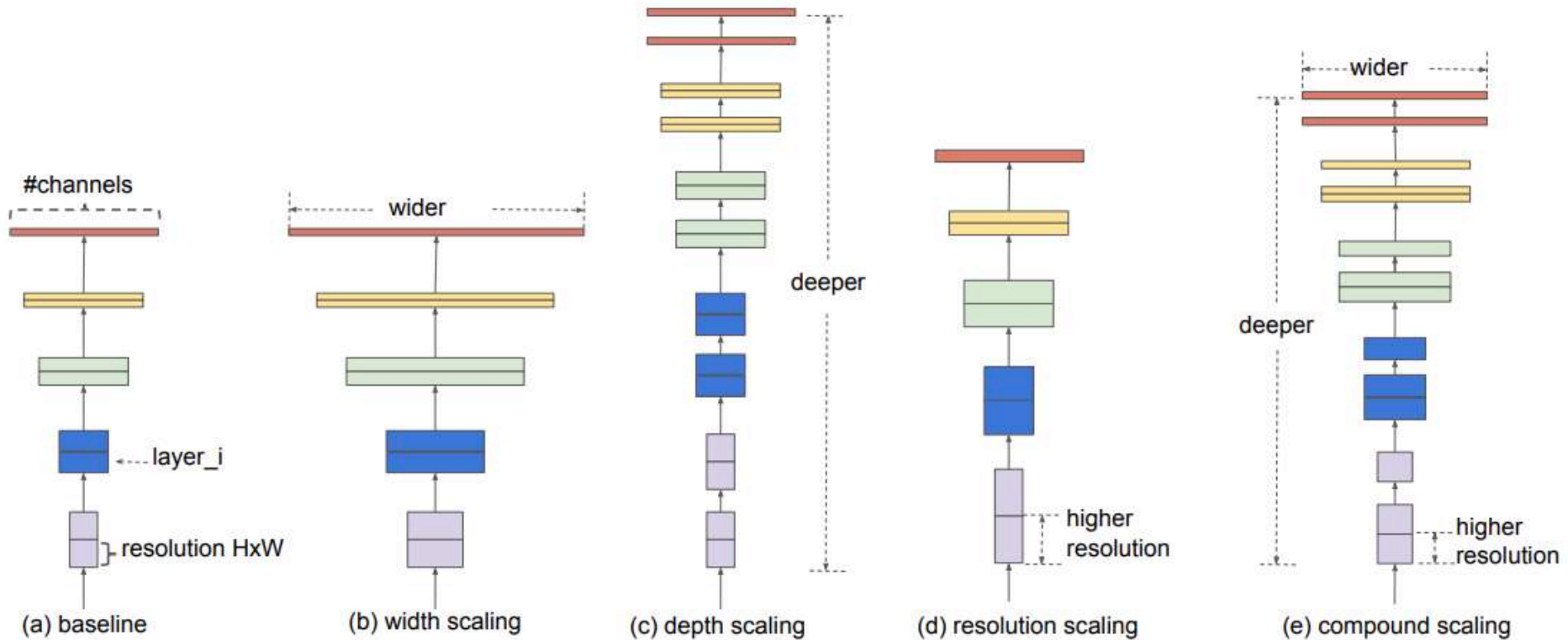


NASNet (2018)

- Найдена с помощью AutoML
- Использует ScheduledDropPath
- 82,7% accuracy on ImageNet



EfficientNet (2019)

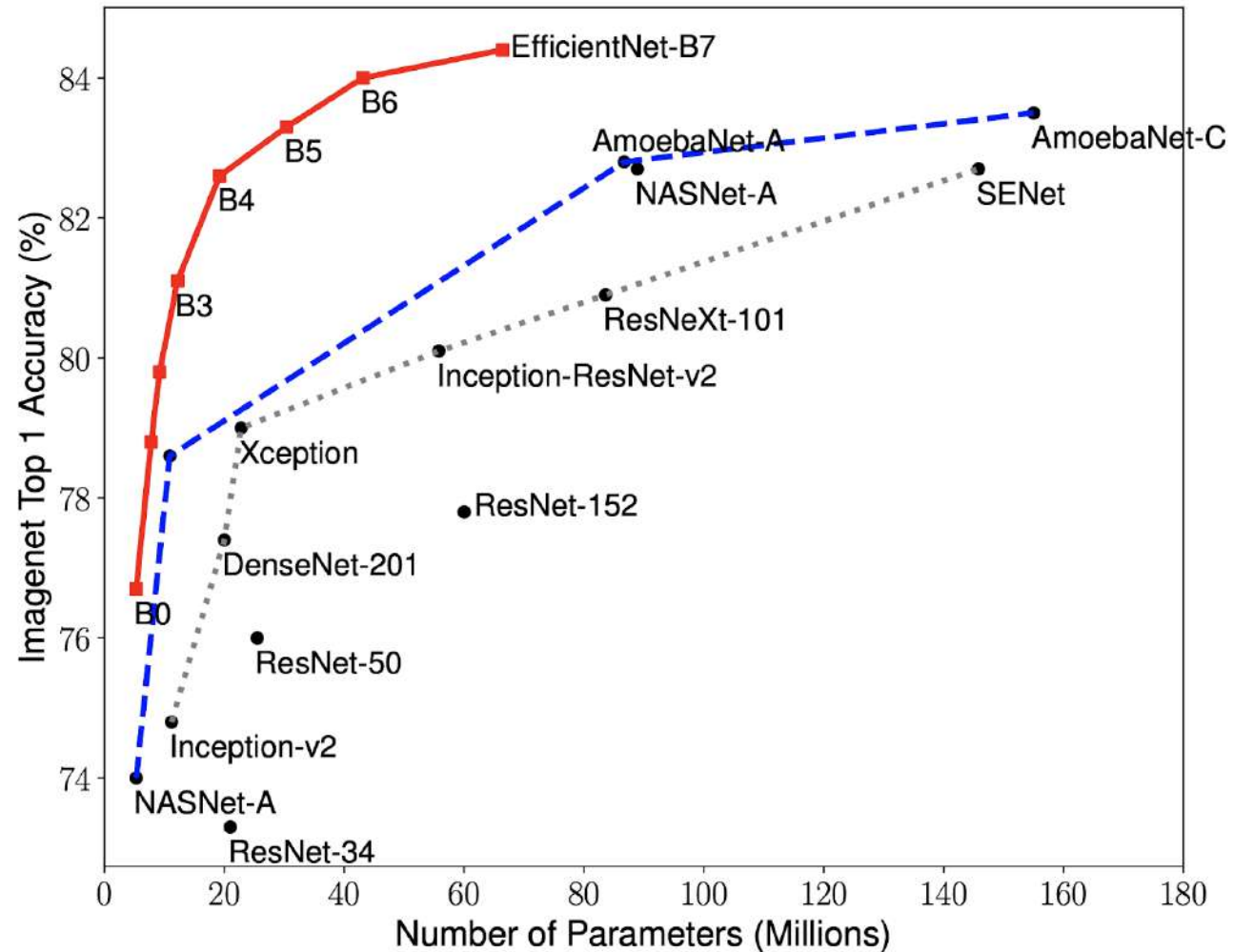


- Top accuracy on ImageNet (84,4%)
- Использует Compound model scaling coefficient ϕ

Классификация: SOTA model zoo

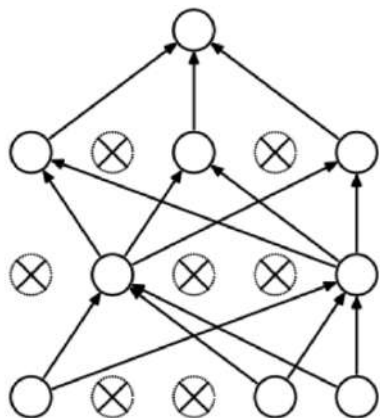
- Классификация:
 - [EfficientNet](#)
 - [AmoebaNet](#)
 - [ResNeXt](#)
 - [NASNet](#)
 - [SENet](#)
 - [Inception-ResNet](#)
 - [Xception](#)
 - [MobileNetV2](#)

Network comparison

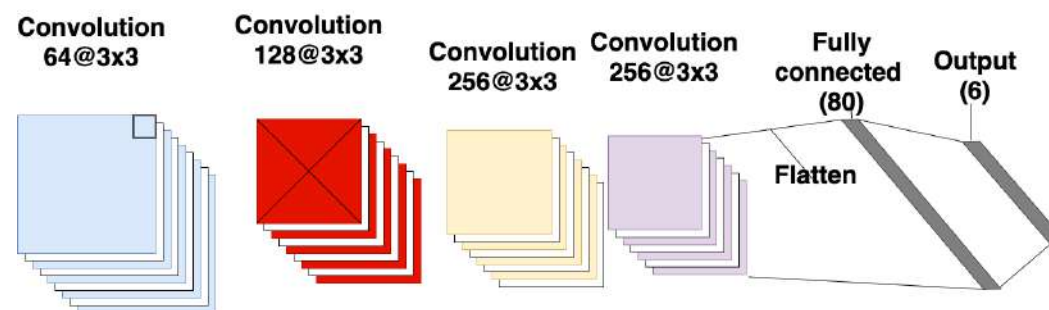


Регуляризация

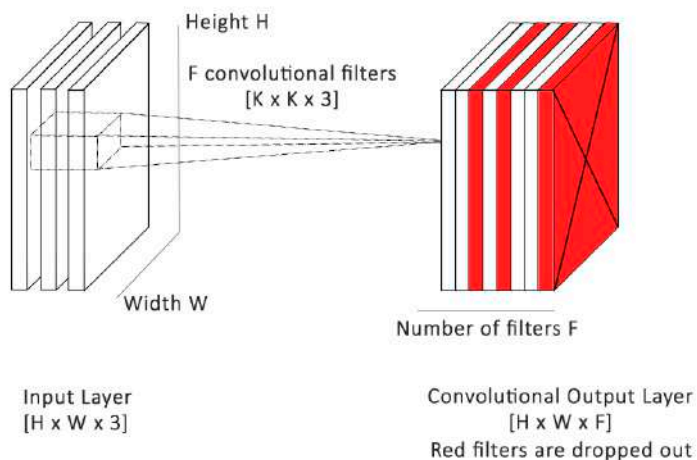
Dropout



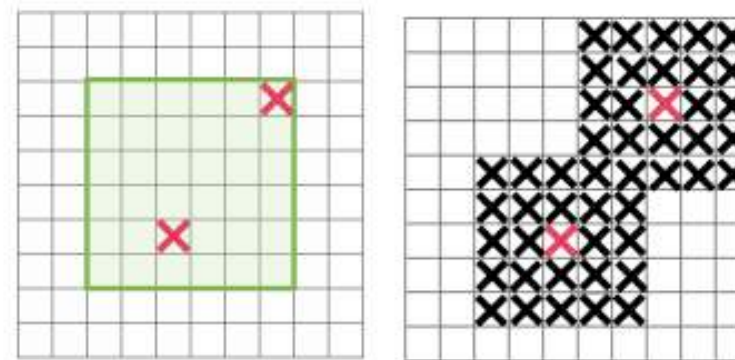
DropLayer



DropChannel



DropBlock



Benchmarks for classification

ImageNet(ILSVRC)	1000 classes	data
CIFAR.....	100/10 classes	data
MNIST	10 classes	data
FashionMNIST.....	10 classes	data

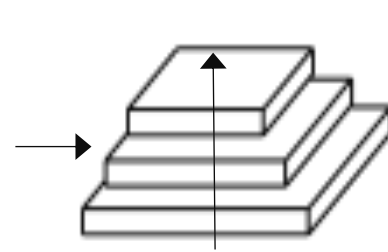
План лекции

- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA model zoo
 - Регуляризация
 - Наборы данных
- **Детекция изображений**
 - Модули
 - Виды детекции и SOTA model zoo
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA model zoo
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

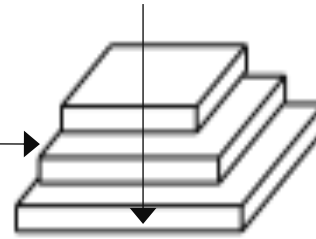
Basic detection model



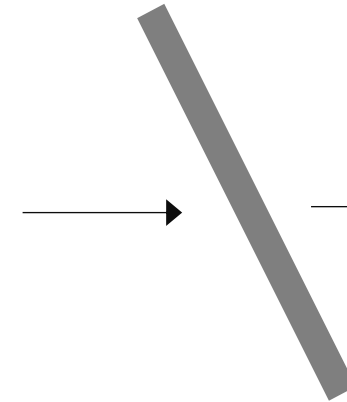
Input image



Feature
extractor



FPN



FC

Bbox

Class

Quality Assessment and Metrics

- mean average precision (mAP)

- Для каждого класса c_i вычислить average precision $ap_i = AP(c_i)$
- Посчитать среднее по всем ap_i значениям для каждого класса

- intersection over union (IoU)

- Каждый bounding box (результат детекции) обладает определенной уверенностью предсказания (confidence)
- Оценивается близость *predicted bboxes* с *ground truth bboxes* как истинные/ложные срабатывания по уровню их пересечения
- Чтобы считаться корректным обнаружением (т. е. истинно положительным), площадь пересечения aoi между *predicted bounding box* BB_p и *ground truth bounding box* BB_{gt} должна быть больше 0,5 согласно формуле

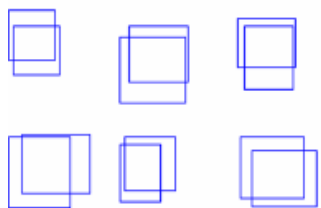
$$area_{oi} = \frac{area(BB_p \cap BB_{gt})}{area(BB_p \cup BB_{gt})} \quad (1)$$

$area_{oi}$ часто называют *intersection over union* (IoU)

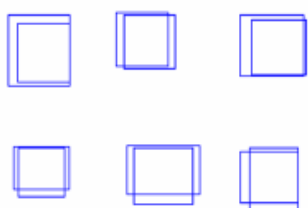
Quality Assessment and Metrics

IoU

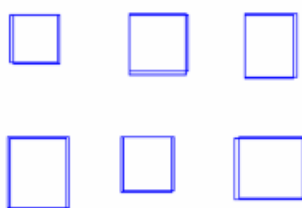
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



IoU = 0.5



IoU = 0.7



IoU = 0.9

Generalized IoU (2019)

Algorithm 1: Generalized Intersection over Union

input : Two arbitrary convex shapes: $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$

output: $GIoU$

1 For A and B , find the smallest enclosing convex object C ,
where $C \subseteq \mathbb{S} \in \mathbb{R}^n$

2 $IoU = \frac{|A \cap B|}{|A \cup B|}$

3 $GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$

Quality Assessment

- Non-Maximum Suppression (NMS):

```
# code is taken from https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/
```

```
# loop over all indexes in the indexes list
```

```
for pos in xrange(0, last):
```

```
# grab the current index
```

```
j = idxs[pos]
```

```
# find the largest (x, y) coordinates for the start of
```

```
# the bounding box and the smallest (x, y) coordinates
```

```
# for the end of the bounding box
```

```
xx1 = max(x1[i], x1[j])
```

```
yy1 = max(y1[i], y1[j])
```

```
xx2 = min(x2[i], x2[j])
```

```
yy2 = min(y2[i], y2[j])
```

```
# compute the width and height of the bounding box
```

```
w = max(0, xx2 - xx1 + 1)
```

```
h = max(0, yy2 - yy1 + 1)
```

```
# compute the ratio of overlap between the computed
```

```
# bounding box and the bounding box in the area list
```

```
overlap = float(w * h) / area[j]
```

```
# if there is sufficient overlap, suppress the
```

```
# current bounding box
```

```
if overlap > overlapThresh:
```

```
suppress.append(pos)
```

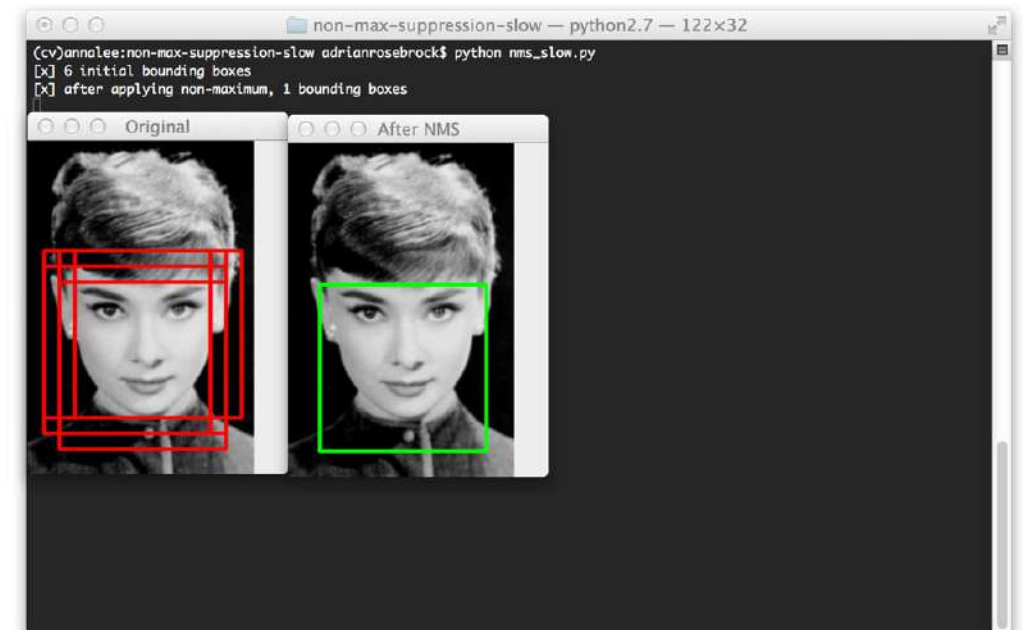
```
# delete all indexes from the index list that are in the
```

```
# suppression list
```

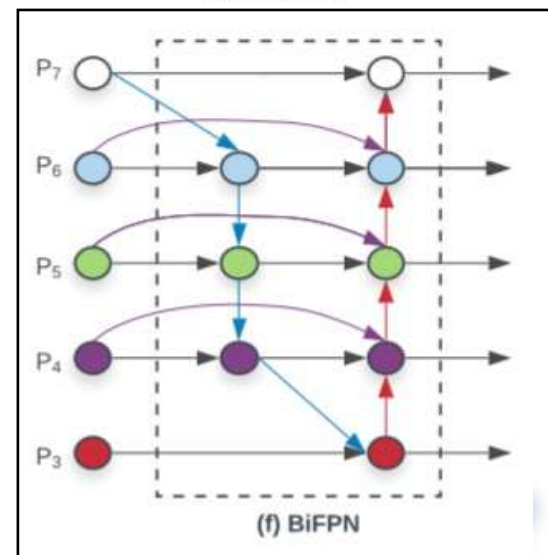
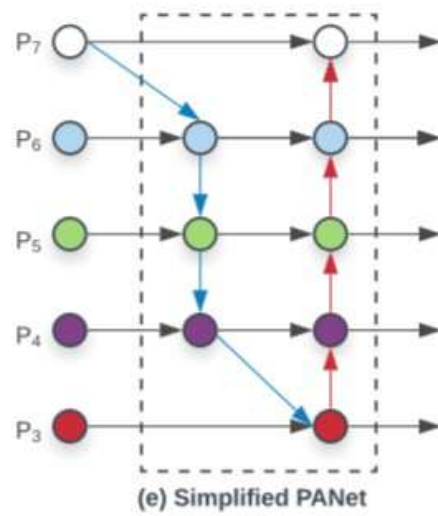
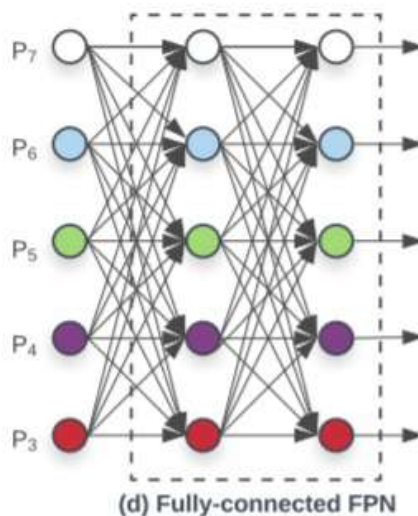
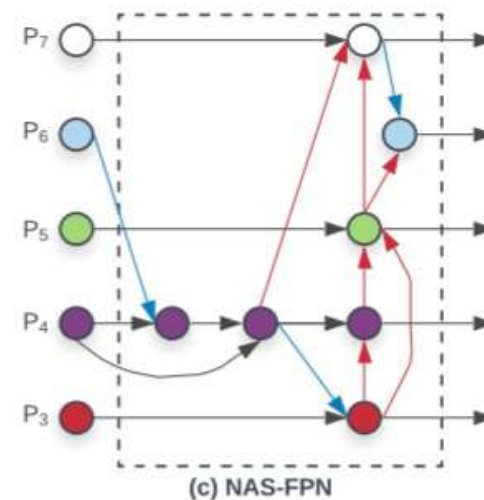
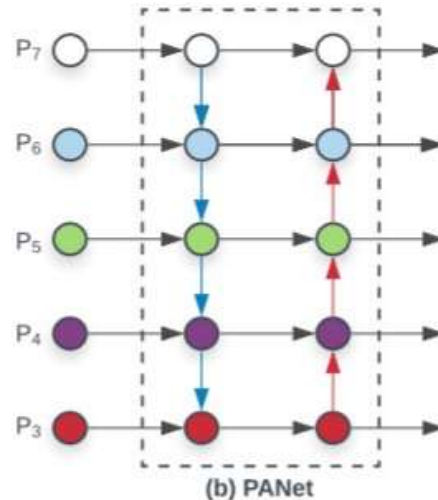
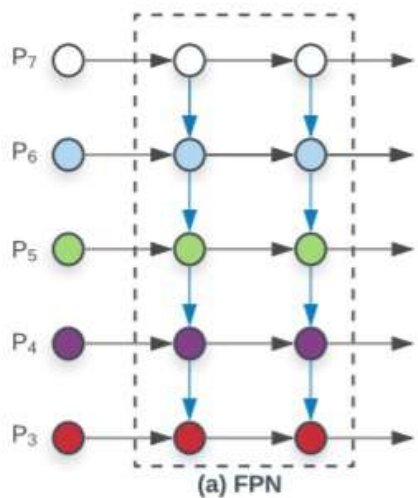
```
idxs = np.delete(idxs, suppress)
```

```
# return only the bounding boxes that were picked
```

```
return boxes[pick]
```

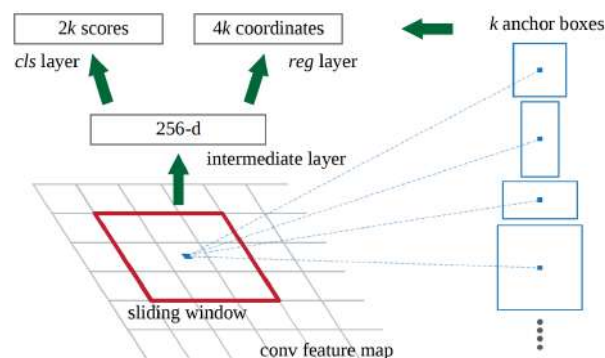


Feature pyramid networks

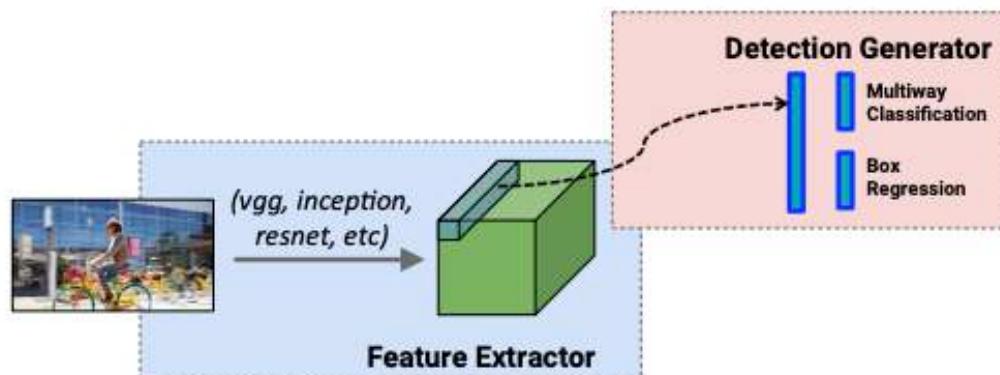


Model zoo

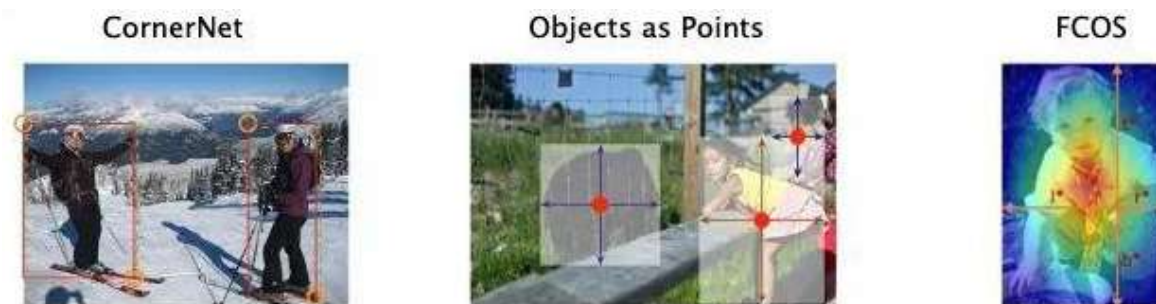
- Anchor-based:



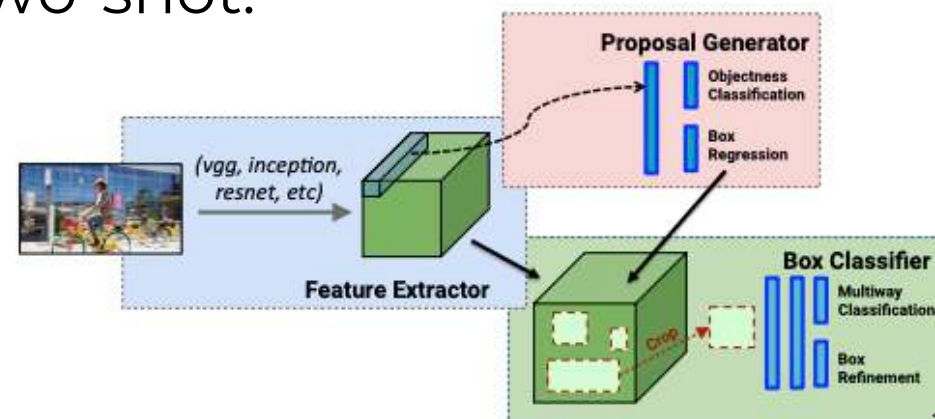
- One-shot:



- Key-point-based:



- Two-shot:



Model zoo

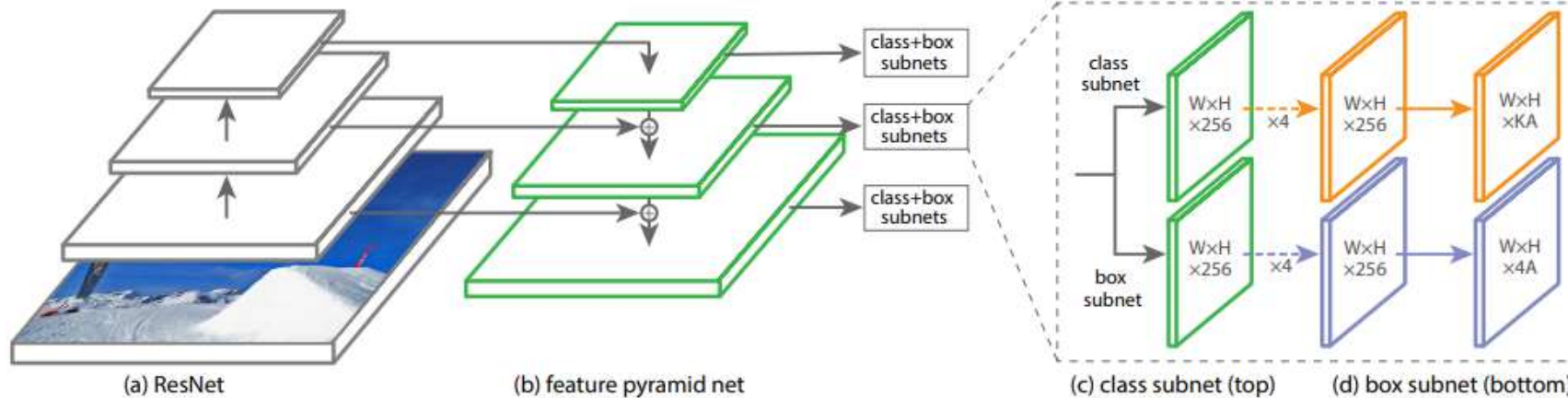
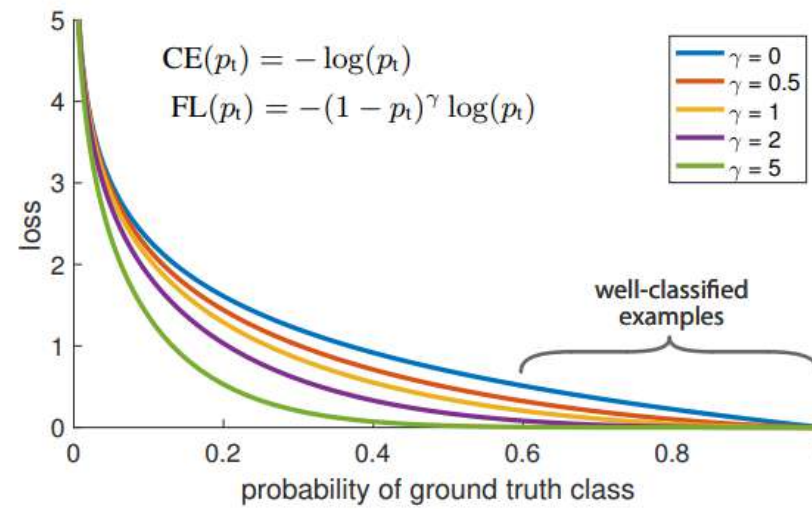
- Anchor-based:
 - [EfficientDet](#)
 - YOLO v4
 - RetinaNet
 - SSD (MobileNet v2)
- One-shot:
 - YOLO v4
 - RetinaNet
 - RPN
 - [FCOS](#)
 - SSD
- Key-point-based:
 - [CornerNet](#)
 - CenterNet
 - [FCOS](#)
- Two-shot:
 - Faster R-CNN
 - R-FCN

RetinaNet (2017)

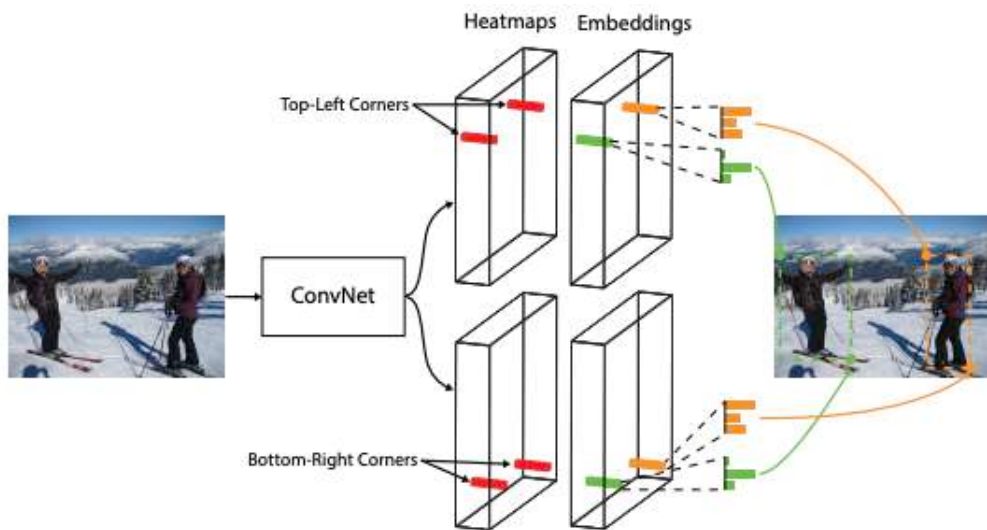
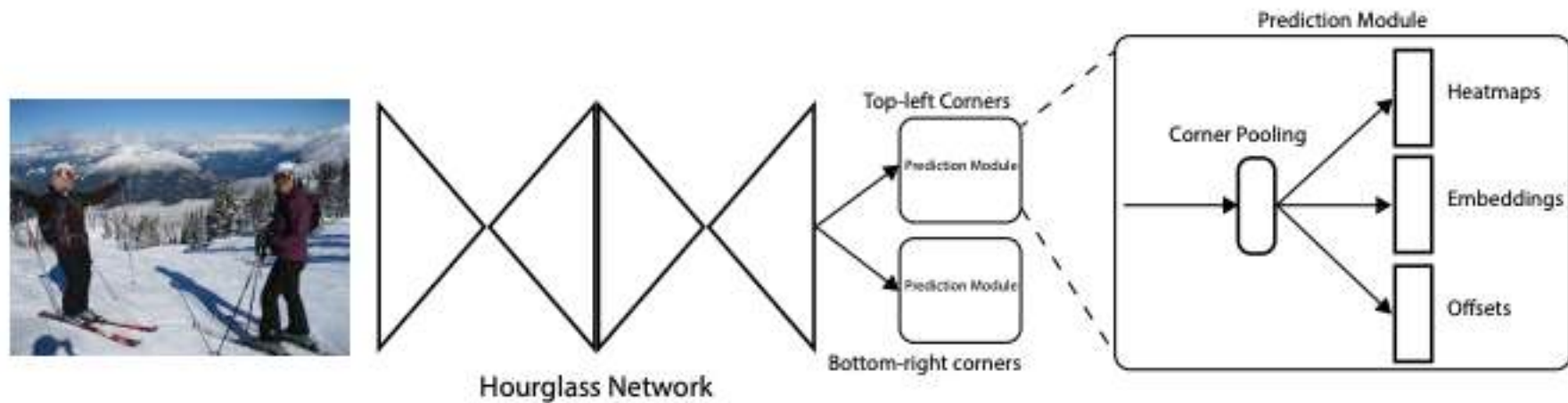
- Использует Focal loss
- Использует FPN

$$\text{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1 - p) & \text{otherwise.} \end{cases}$$

$$\text{FL}(p, y) = \begin{cases} -\alpha(1 - p)^\gamma \log p, & \text{if } y = 1 \\ -p^\gamma \log(1 - p), & \text{otherwise.} \end{cases}$$



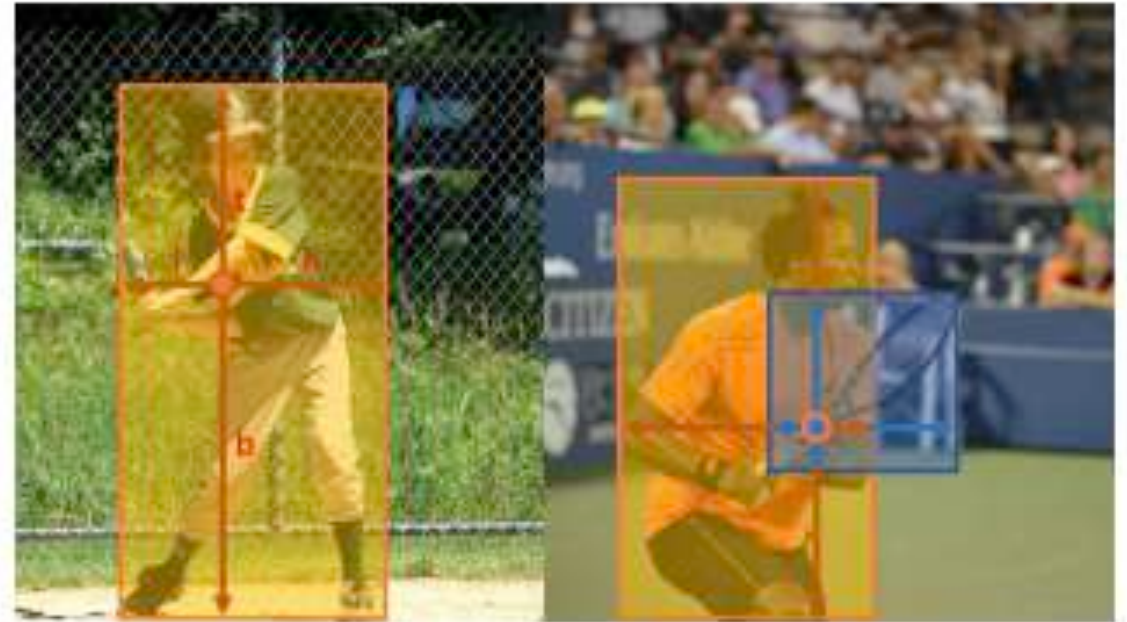
CornerNet (2019)



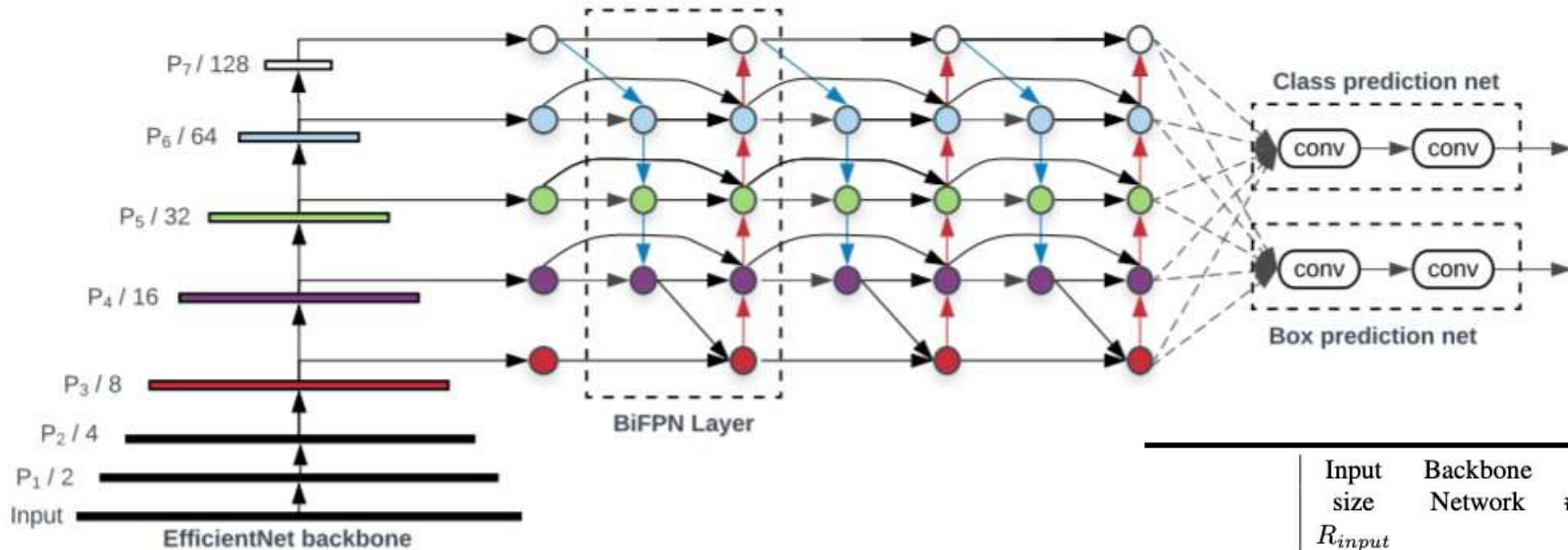
	AP	AP ⁵⁰	AP ⁶⁰	AP ⁷⁰	AP ⁸⁰	AP ⁹⁰
RetinaNet (Lin et al., 2017)	39.8	59.5	55.6	48.2	36.4	15.1
Cascade R-CNN (Cai and Vasconcelos, 2017)	38.9	57.8	53.4	46.9	35.8	15.8
Cascade R-CNN + IoU Net (Jiang et al., 2018)	41.4	59.3	55.3	49.6	39.4	19.5
CornerNet	40.6	56.1	52.0	46.8	38.8	23.4

FCOS (2019)

- Предсказывает вектор (l, t, r, b) удаленности точки от для каждого пикселя
- Отказ от кандидатов
- Проблема пересечения классов
- Для решения предсказываем класс наименьшего размера



EfficientDet (2020)



- Единый масштабирующий коэффициент ϕ
- Сгенерированные архитектуры EfficientDet превосходят по скорости сравнимые по точности архитектуры

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1280	B6	384	8	5
D7 ($\phi = 7$)	1536	B6	384	8	5
D7x	1536	B7	384	8	5

Anchor vs anchor-free detectors

- [Авторы](#) добавили главные фишки FCOS в RetinaNet

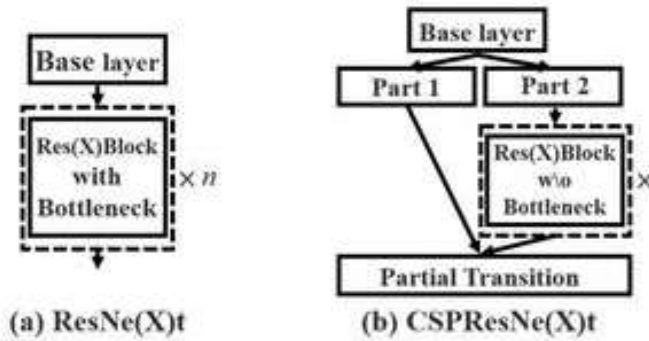
Inconsistency	FCOS	RetinaNet (#A=1)					
GroupNorm	✓	✓	✓	✓	✓	✓	✓
GIoU Loss	✓		✓	✓	✓	✓	✓
In GT Box	✓			✓	✓	✓	✓
Centerness	✓				✓	✓	✓
Scalar	✓						✓
AP (%)	37.8	32.5	33.4	34.9	35.3	36.8	37.0

- Оказалось, что их точность стала сравнима (разница 0,8%)
- Таким образом, оба подхода к детекции объектов достигают одинаковой точности при равных условиях

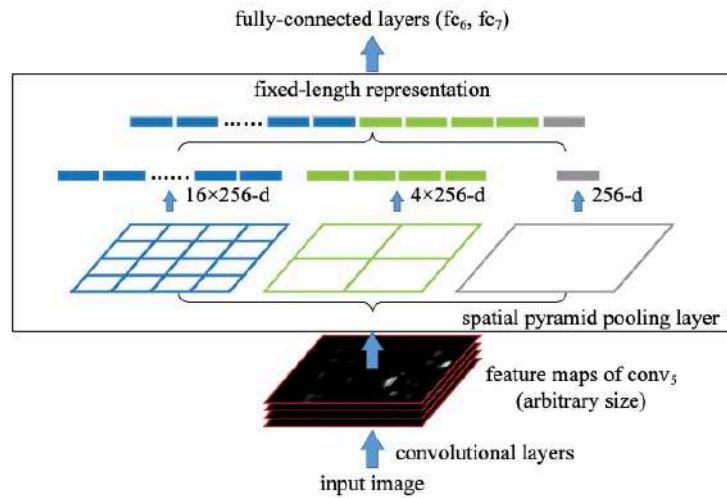
Classification	Regression	Box	Point
	Intersection over Union	37.0	36.9
	Spatial and Scale Constraint	37.8	37.8

YOLOv4 (2020)

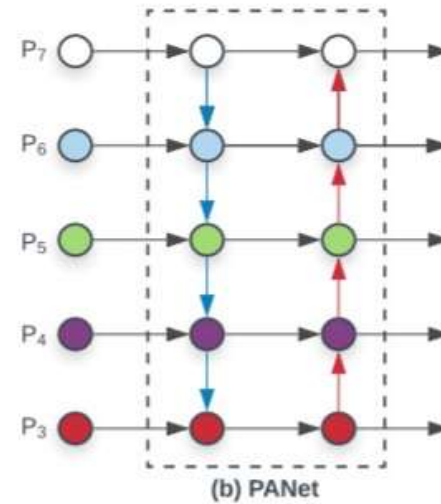
- CSPDarknet backbone



- SPP (Spatial Pyramid Pooling)



- Modified PANet as FPN



- YOLOv3 head
- Modified Spatial attention module
- Mosaic Augmentation
- Self-adversarial training

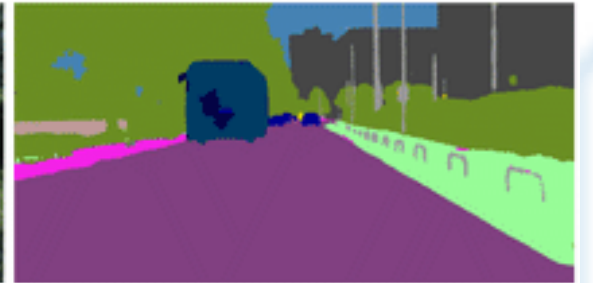
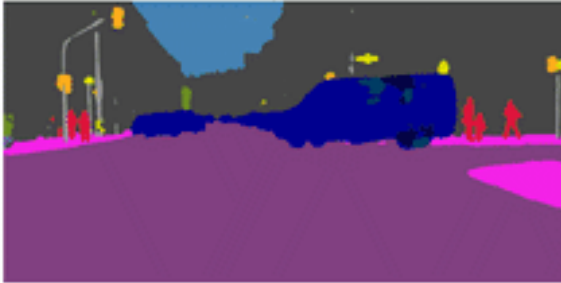
Public Datasets for detection

Inria Aerial Image Labeling	2017	paper	data
DAVIS Challenge	2017	paper	data
Mapillary Vistas	2017	paper	data
ADE20K	2016	paper	data
SYNTHIA	2016	paper	data
SpaceNet	2016	N/A	data
Playing for Data	2016	paper	data
SUN RGB-D	2015	paper	data
Common Objects in Context (COCO)	2014	paper	data
Oxford RoboCar Dataset	2014	paper	data
KITTI Vision Benchmark Suite	2012	paper	data
Visual Object Classes 2012 (VOC12)....	2012	[1] [2]	data
NYU Depth Dataset V2	2012	paper	data
Caltech Pedestrian Detection Benchmark	2009	[1] [2]	data

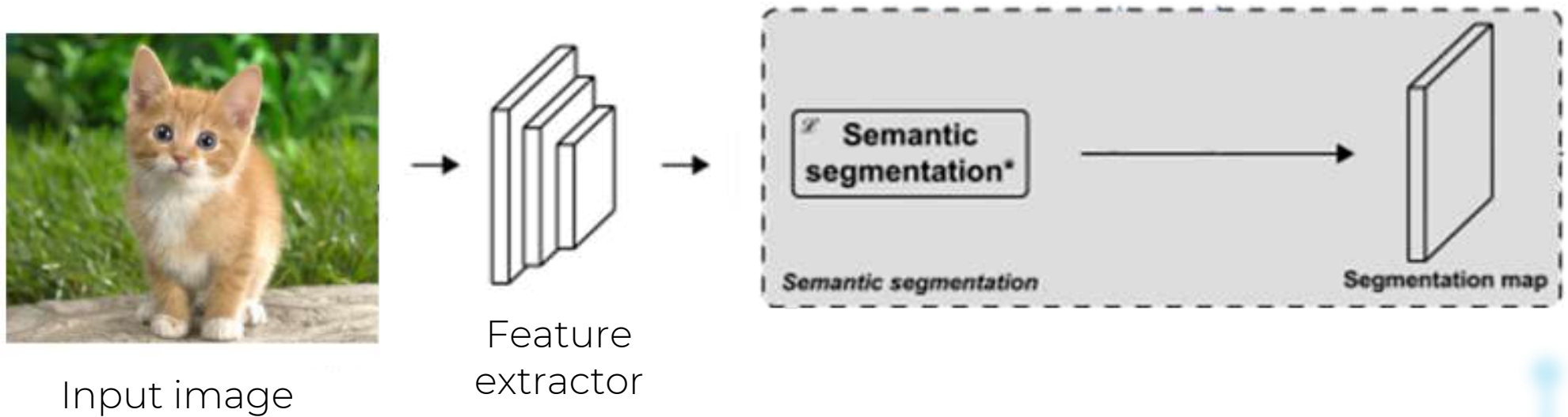
План лекции

- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA model zoo
 - Регуляризация
 - Наборы данных
- Детекция изображений
 - Модули
 - Виды детекции и SOTA model zoo
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA model zoo
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

Semantic segmentation



Basic semantic segmentation model

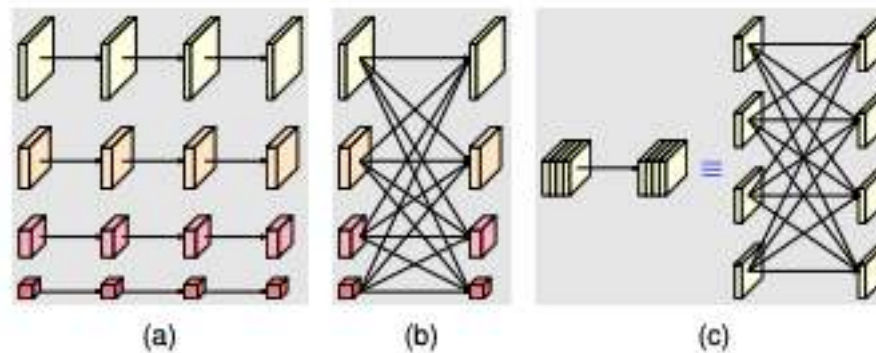
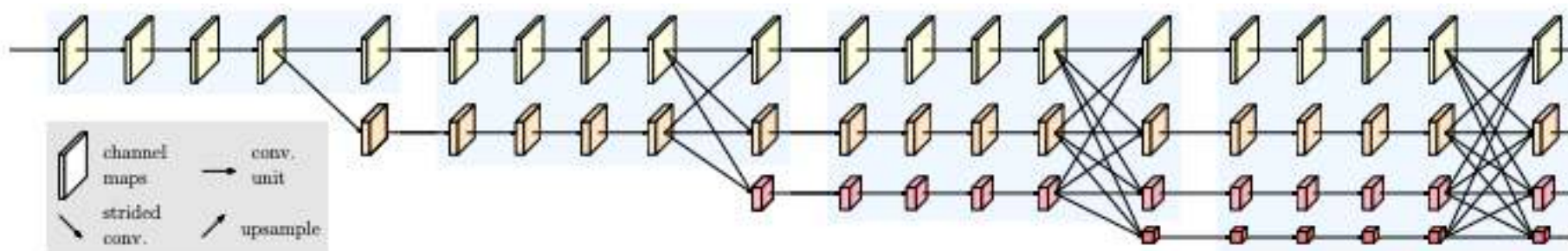


Model zoo

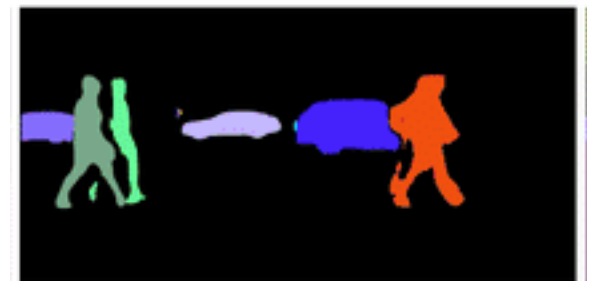
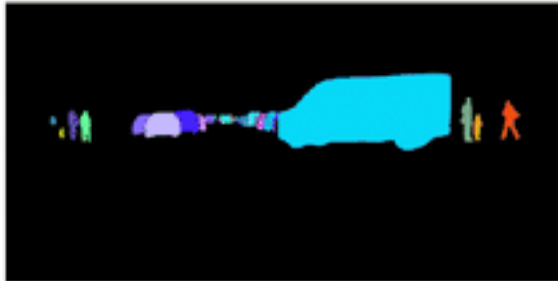
- Семантическая сегментация:
 - [HRNet](#)
 - PSPNet
 - UNet++
 - DeepLabV3
 - Mask R-CNN

HRNet

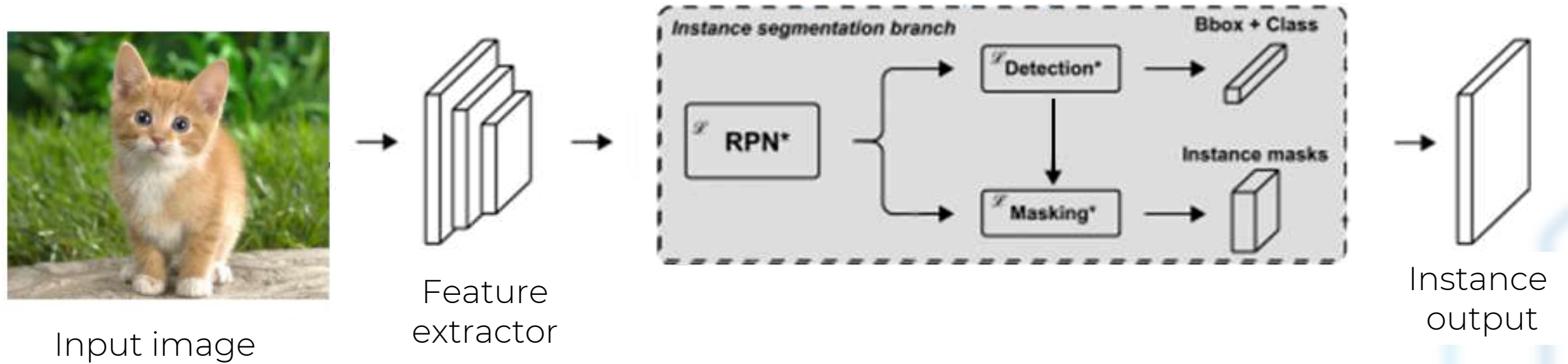
- Построение признаков высокого разрешения



Instance segmentation



Basic instance segmentation model



Mask R-CNN (2017)

Results on COCO test images using ResNet-101-FPN at 5 fps.



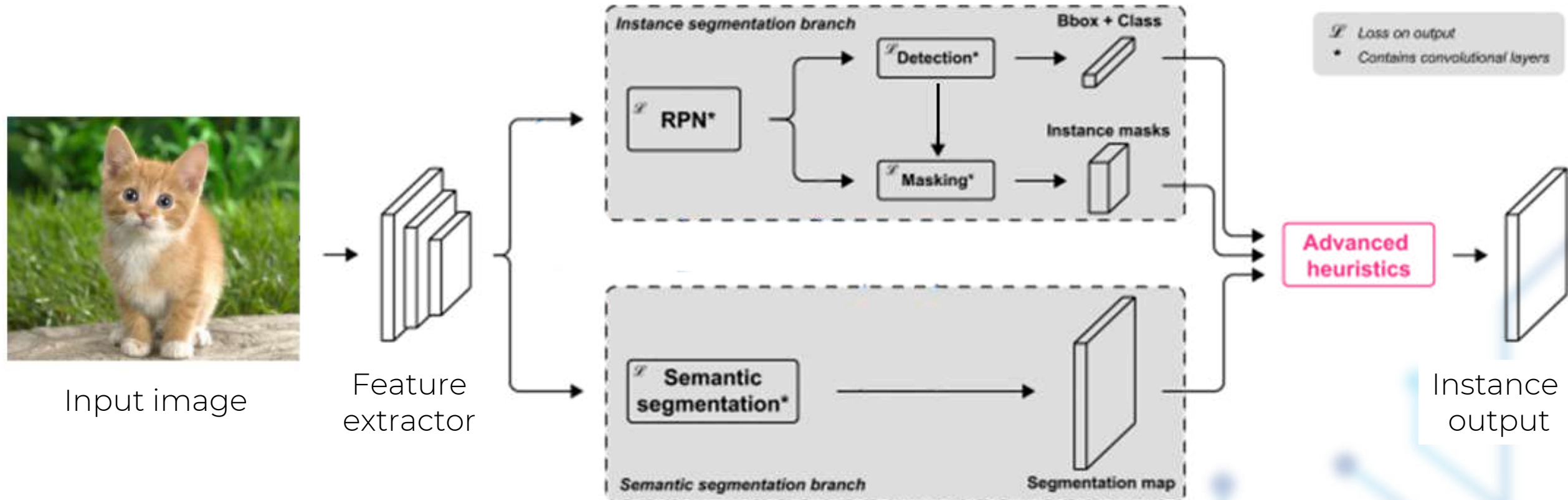
SOTA model zoo

- Инстанс сегментация:
 - Mask R-CNN
 - Detectron
 - Detectron2

Panoptic segmentation



Basic panoptic segmentation model

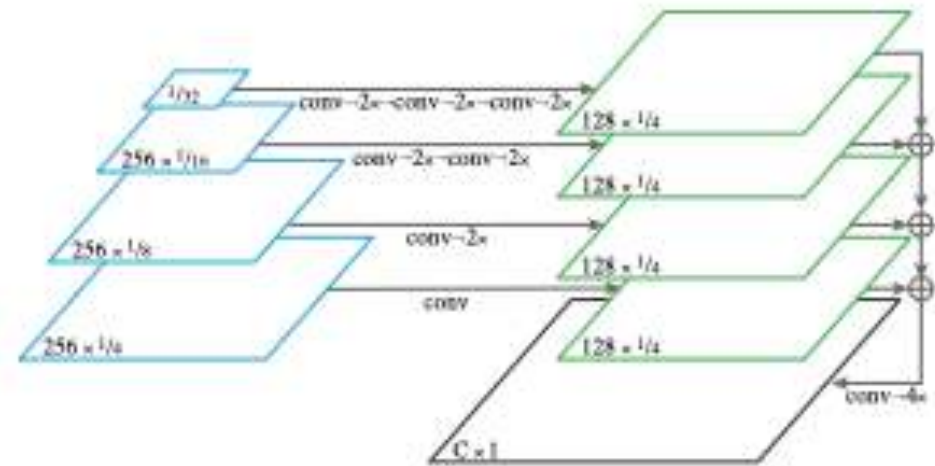
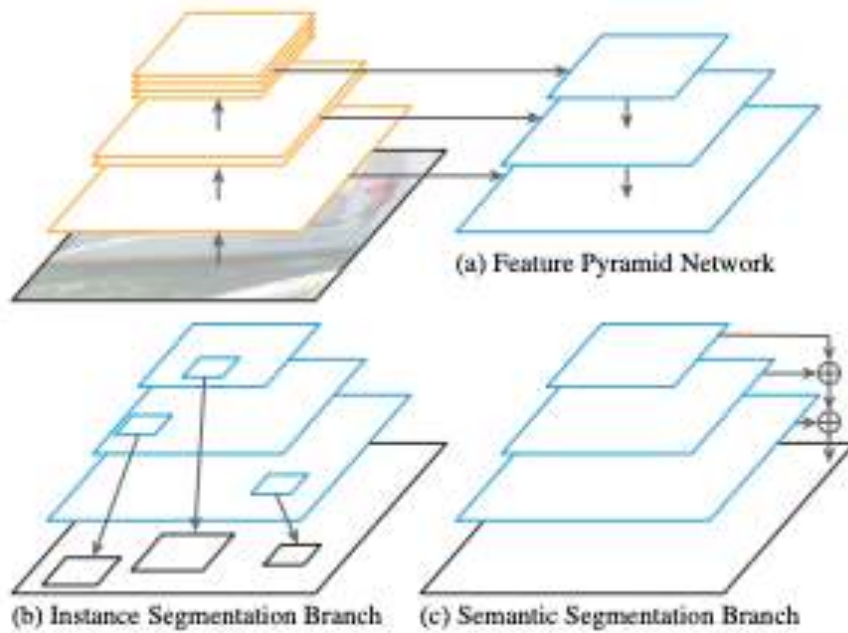


Model zoo

- Паноптическая сегментация:
 - [Panoptic FPN](#)
 - [CenterMask](#)
 - AdaptIS (Adaptive Instance Segmentation)

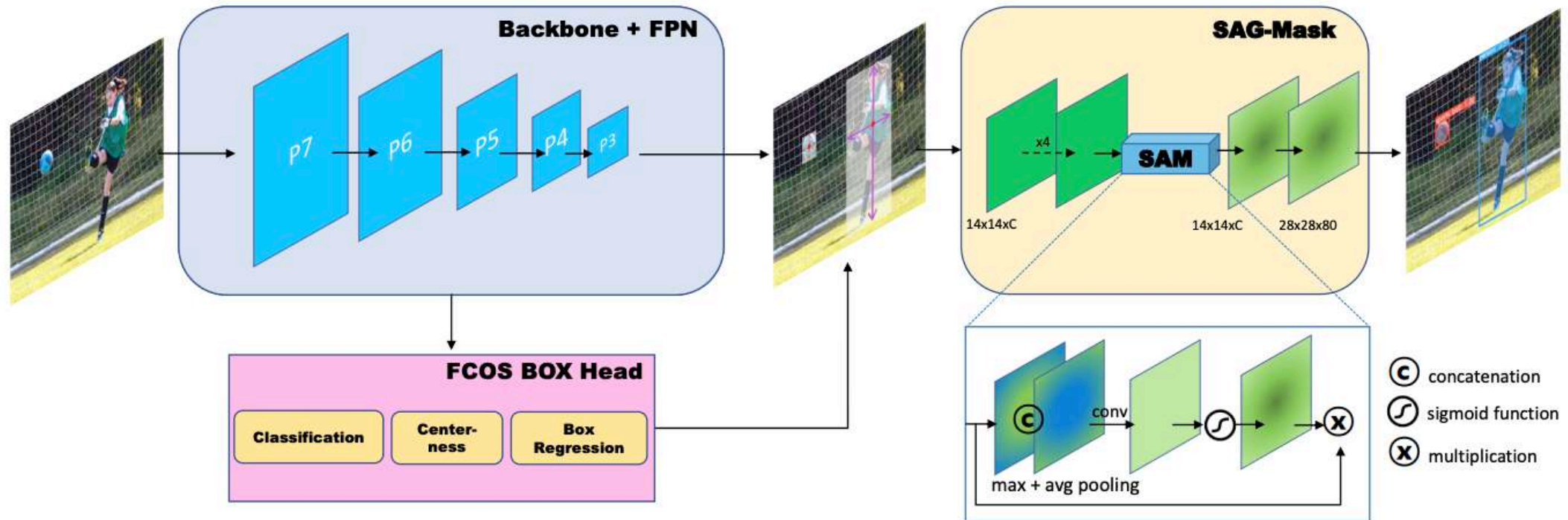
Panoptic Feature Pyramid Networks (2019)

Mask R-CNN, дополненная веткой семантической сегментации



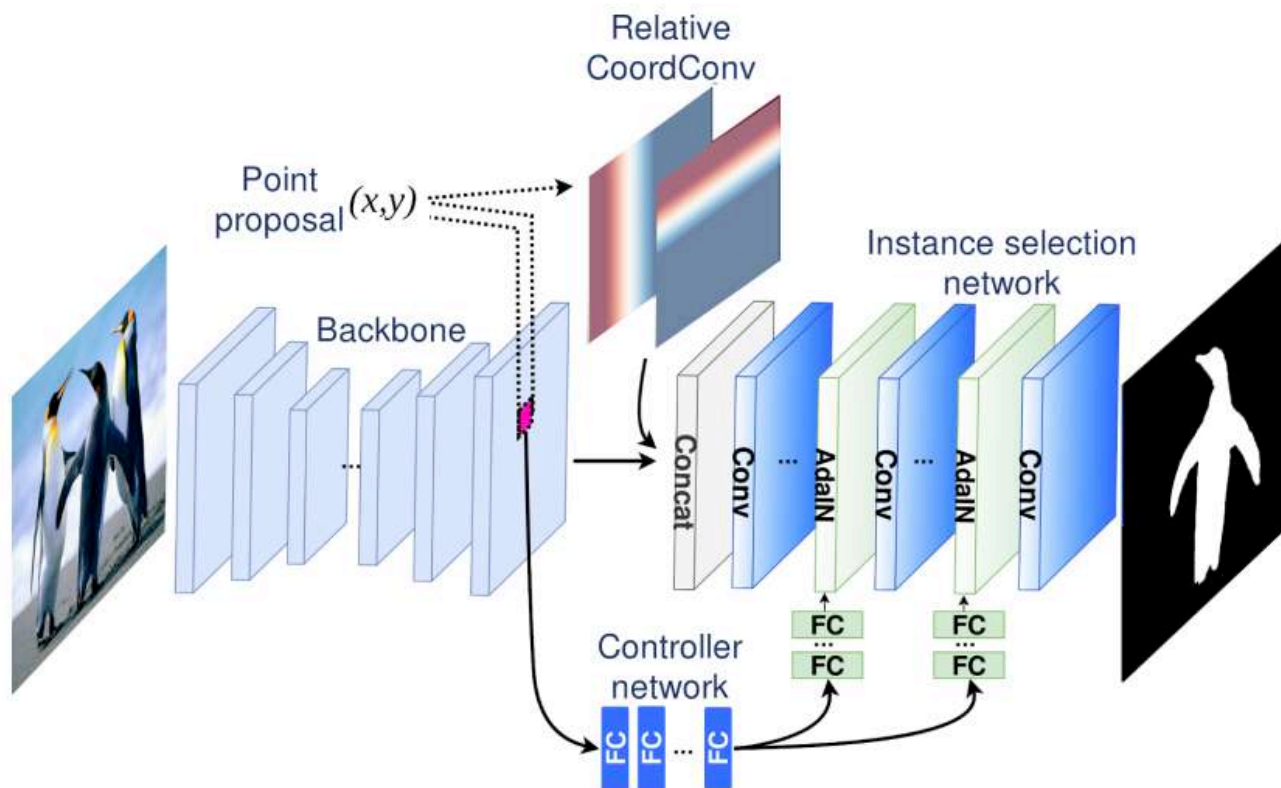
CenterMask (2020)

- FCOS + Mask R-CNN
- Для каждого объекта генерируется его маска



AdaptIS (2019)

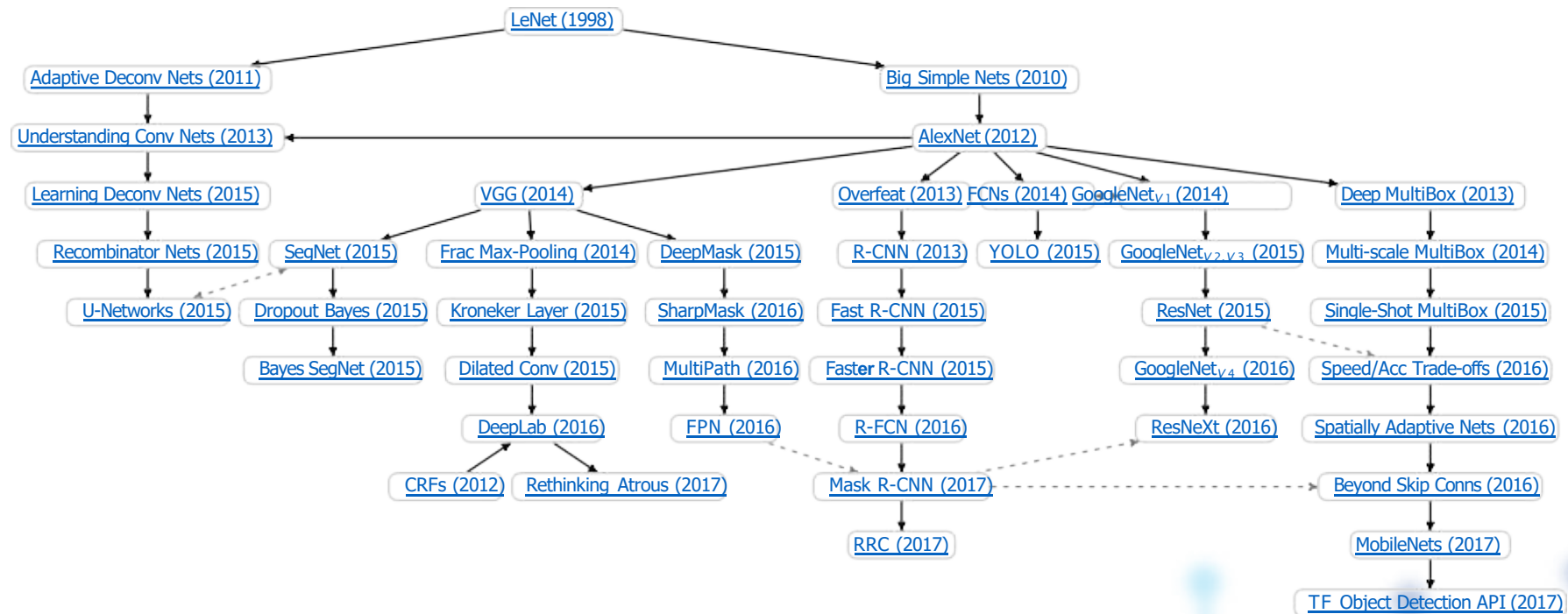
- Использует Adaptive Instance Normalization (AdaIN)
- AdaIN – управляющая сеть



Public Datasets for segmentation

Inria Aerial Image Labeling	2017	paper	data
DAVIS Challenge	2017	paper	data
Mapillary Vistas	2017	paper	data
ADE20K	2016	paper	data
SYNTHIA	2016	paper	data
SpaceNet	2016	N/A	data
Playing for Data	2016	paper	data
SUN RGB-D	2015	paper	data
Cityscapes.....	2015	paper	data
Common Objects in Context (COCO)	2014	paper	data
Oxford RoboCar Dataset	2014	paper	data
KITTI Vision Benchmark Suite	2012	paper	data
Visual Object Classes 2012 (VOC12)....	2012	[1][2]	data
NYU Depth Dataset V2	2012	paper	data
Caltech Pedestrian Detection Benchmark	2009	[1][2]	data
CamVid: Motion-based Segmentation...	2008	paper	data

Detection and Segmentation Atlas



Nodes are hyperlinks to the associated [arXiv.org](https://arxiv.org) papers.

План лекции

- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA model zoo
 - Регуляризация
 - Наборы данных
- Детекция изображений
 - Модули
 - Виды детекции и SOTA model zoo
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA model zoo
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

Practical tips: Bag of freebies

- SOTA Augmentations:

- Mixup



- Cutout



- CutMix (2 images)

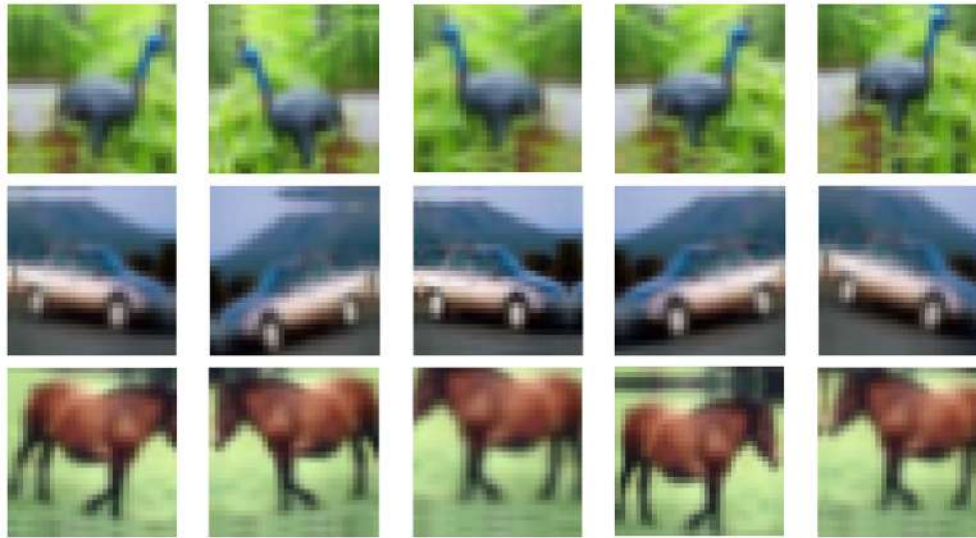


- **Mosaic (4 images)**



Practical tips: Bag of freebies

- Test-time Augmentation (TTA)
 - Позволяет повысить точность на 3%
 - Увеличивает время inference



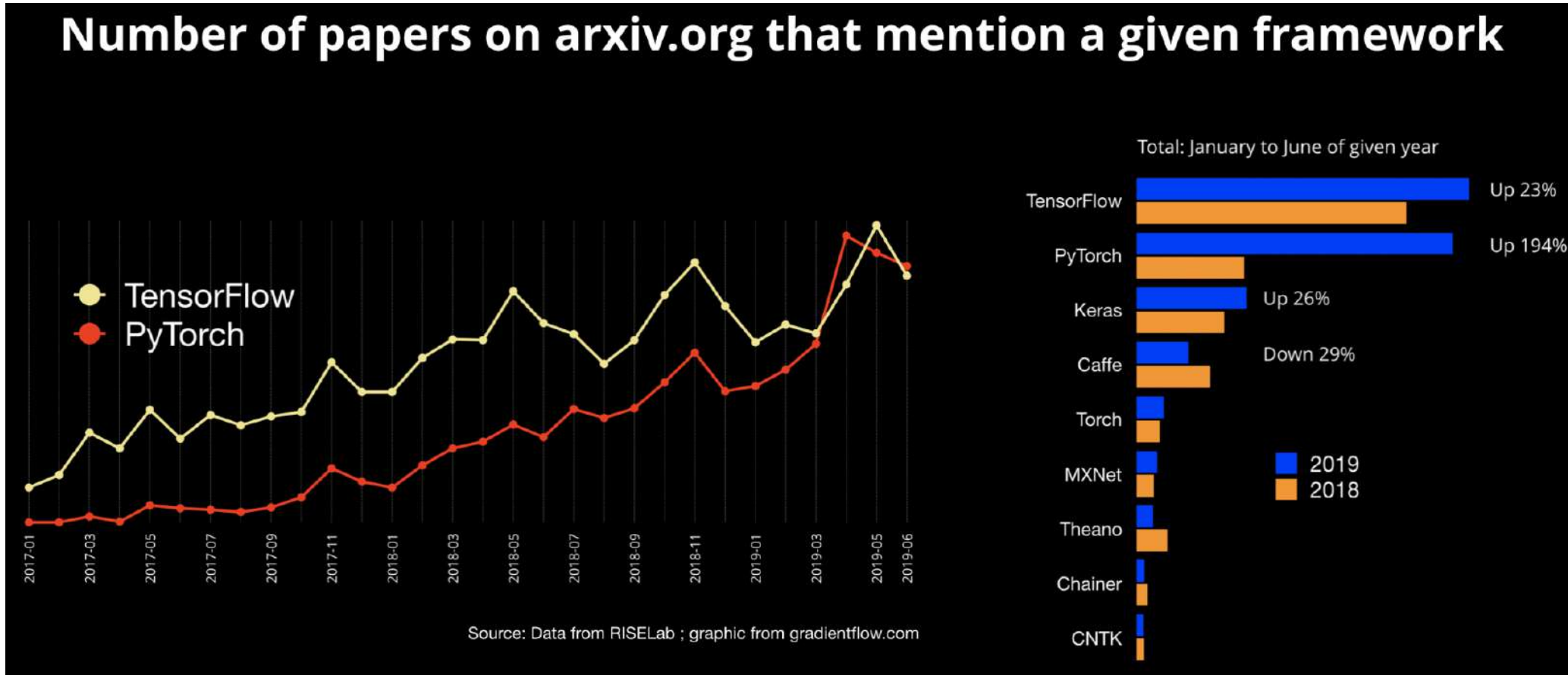
Practical tips: Bag of specials

- ATSS (Adaptive training sample selection)
- Self-adversarial training
- Uncertainty estimation with MC Dropout

План лекции

- Краткая история компьютерного зрения
- Классификация изображений
 - Модули
 - SOTA model zoo
 - Регуляризация
 - Наборы данных
- Детекция изображений
 - Модули
 - Виды детекции и SOTA model zoo
 - Losses
 - Метрики
 - Наборы данных
- Сегментация
 - Виды сегментации
 - Модули
 - SOTA model zoo
 - Метрики
 - Наборы данных
- Practical tips
 - Bag of freebies
 - Bag of specials
- SOTA frameworks

Common trends



Low-level frameworks



- › tf.errors
- › tf.estimator
- › tf.experimental 🧪
- › tf.feature_column
- › tf.graph_util
- › tf.image
- › tf.io
- › **tf.keras**
- › tf.linalg
- › tf.lite
- › tf.lookup
- › tf.math
- › tf.mixed_precision
- › tf.mlir
- › tf.nest
- › tf.nn
- › tf.quantization



- Низкоуровневый, но
- TFLearn, TFLite etc
- Много шаблонного кода
- Подходит для продакшн
- Визуализация Tensorboard
- Tensorflow Serving
- Quick Tutorial



- Проще и
- FastAI, Lightning advanced stuff
- Много готовых модулей
- Подходит для прототипирования
- Подходит для ризерча
- Визуализация visdom
- Flask
- Quick Tutorial

Catalyst & Pytorch Lightning

Catalyst

VS

```
21 # model runner
22 runner = SupervisedRunner()
23
24 # model training
25 runner.train(
26     model=model,
27     criterion=criterion,
28     optimizer=optimizer,
29     scheduler=scheduler,
30     loaders=loaders,
31     callbacks=[
32         PrecisionCallback(),
33         EarlyStoppingCallback()
34     ],
35     logdir=logdir,
36     n_epochs=n_epochs,
37     verbose=True
38 )
39
```

```
1 import argparse
2
3 import torch
4 import torch.nn as nn
5 import torch.nn.parallel
6 import torch.optim
7 import torch.utils.data
8 import torchvision.transforms as transforms
9 import torchvision.datasets as datasets
10 import torchvision.models as models
11
12 # experiment setup
13 logdir = "./logdir"
14 n_epochs = 42
15 args = ...
16
17 # data
18 loaders = {'train': train_loader, 'valid': test_loader}
19
20 use_cuda = not args.no_cuda and torch.cuda.is_available()
21 device = torch.device("cuda" if use_cuda else "cpu")
22
23 # model, criterion, optimizer
24 model = Net()
25
26 if args.gpu is not None:
27     torch.cuda.set_device(args.gpu)
28     model = model.cuda(args.gpu)
29 else:
30     # DataParallel will divide and allocate batch_size to all available GPUs
31     model = torch.nn.DataParallel(model).cuda()
32
33 # define loss function (criterion) and optimizer
34 criterion = nn.CrossEntropyLoss().cuda(args.gpu)
35
36 optimizer = torch.optim.SGD(
37     model.parameters(), args.lr,
38     momentum=args.momentum,
39     weight_decay=args.weight_decay)
40
41 for epoch in range(args.start_epoch, args.epochs):
42     adjust_learning_rate(optimizer, epoch, args)
43
44     # train for one epoch
45     train(train_loader, model, criterion, optimizer, epoch, args)
46
47     # evaluate on validation set
48     acc1 = validate(val_loader, model, criterion, args)
49
50     # remember best acc1 and save checkpoint
51     is_best = acc1 > best_acc1
52     best_acc1 = max(acc1, best_acc1)
53
54     save_checkpoint({
55         'epoch': epoch + 1,
56         'arch': args.arch,
57         'state_dict': model.state_dict(),
58         'best_acc1': best_acc1,
59         'optimizer': optimizer.state_dict(),
60     }, is_best)
61
62
63
64
```

```
65 def train(train_loader, model, criterion, optimizer, epoch, args):
66     batch_time = AverageMeter()
67     data_time = AverageMeter()
68     losses = AverageMeter()
69     top1 = AverageMeter()
70     top5 = AverageMeter()
71
72     # switch to train mode
73     model.train()
74
75     end = time.time()
76     for i, (input, target) in enumerate(train_loader):
77         # measure data loading time
78         data_time.update(time.time() - end)
79
80         if args.gpu is not None:
81             input = input.cuda(args.gpu, non_blocking=True)
82             target = target.cuda(args.gpu, non_blocking=True)
83
84         # compute output
85         output = model(input)
86         loss = criterion(output, target)
87
88         # measure accuracy and record loss
89         acc1, acc5 = accuracy(output, target, topk=(1, 5))
90         losses.update(loss.item(), input.size())
91         top1.update(acc1[0], input.size())
92         top5.update(acc5[0], input.size())
93
94         # compute gradient and do SGD step
95         optimizer.zero_grad()
96         loss.backward()
97         optimizer.step()
98
99         # measure elapsed time
100         batch_time.update(time.time() - end)
101         end = time.time()
102
103         if i % args.print_freq == 0:
104             print('Epoch: [{0}][{1}/{2}] \t'
105                   '\t time (batch_time.val, {3}) \t (batch_time.avg, {4}) \t'
106                   '\t data (data_time.val, {5}) \t (data_time.avg, {6}) \t'
107                   '\t loss (loss.val, {7}) \t (loss.avg, {8}) \t'
108                   '\t Acc@1 (top1.val, {9}) \t (top1.avg, {10}) \t'
109                   '\t Acc@5 (top5.val, {11}) \t (top5.avg, {12})'.format(
110                     epoch, i, len(train_loader), batch_time=batch_time,
111                     data_time=data_time, loss=losses, top1=top1, top5=top5))
112
113 def validate(val_loader, model, criterion, args):
114     batch_time = AverageMeter()
115     losses = AverageMeter()
116     top1 = AverageMeter()
117     top5 = AverageMeter()
118
119     # switch to evaluate mode
120     model.eval()
121
122     with torch.no_grad():
123         end = time.time()
124         for i, (input, target) in enumerate(val_loader):
125             if args.gpu is not None:
126                 input = input.cuda(args.gpu, non_blocking=True)
127                 target = target.cuda(args.gpu, non_blocking=True)
128
129             # compute output
130             output = model(input)
131             loss = criterion(output, target)
132
133             # measure accuracy and record loss
134             acc1, acc5 = accuracy(output, target, topk=(1, 5))
135             losses.update(loss.item(), input.size())
136             top1.update(acc1[0], input.size())
137             top5.update(acc5[0], input.size())
138
139             # measure elapsed time
140             batch_time.update(time.time() - end)
141             end = time.time()
142
143             if i % args.print_freq == 0:
144                 print('Test: [{0}/{1}] \t'
145                       '\t time (batch_time.val, {3}) \t (batch_time.avg, {4}) \t'
146                       '\t loss (loss.val, {5}) \t (loss.avg, {6}) \t'
147                       '\t Acc@1 (top1.val, {7}) \t (top1.avg, {8}) \t'
148                       '\t Acc@5 (top5.val, {9}) \t (top5.avg, {10})'.format(
149                         i, len(val_loader), batch_time=batch_time, loss=losses,
150                         top1=top1, top5=top5))
151
152         print(' * Acc@1 (top1.avg, {3}) Acc@5 (top5.avg, {4})'
153               '.format(top1=top1, top5=top5))
154
155         return top1.avg
156
157 def save_checkpoint(state, is_best, filename='checkpoint.pth.tar'):
158     torch.save(state, filename)
159     if is_best:
160         shutil.copyfile(filename, 'model_best.pth.tar')
161
162 class AverageMeter(object):
163     """Computes and stores the average and current value"""
164     def __init__(self):
165         self.reset()
166
167     def reset(self):
168         self.val = 0
169         self.avg = 0
170         self.sum = 0
171         self.count = 0
172
173     def update(self, val, n=1):
174         self.val = val
175         self.sum += val * n
176         self.count += n
177         self.avg = self.sum / self.count
178
179 def adjust_learning_rate(optimizer, epoch, args):
180     """Sets the learning rate to the initial LR decayed by 10 every 30 epochs"""
181     lr = args.lr * (0.1 ** (epoch // 30))
182     for param_group in optimizer.param_groups:
183         param_group['lr'] = lr
184
185 def accuracy(output, target, topk=(1,)):
186     """Computes the accuracy over the k top predictions for the specified values of k"""
187     with torch.no_grad():
188         batch_size = target.size(0)
189         _, grad = output.topk(maxk, 1, True, True)
190         pred = grad.t()
191         correct = pred.eq(target.view(1, -1).expand_as(pred))
192
193     res = []
194     for k in topk:
195         correct_k = correct[:, k].sum(0, keepdim=True)
196         res.append(correct_k.mul_(100.0 / batch_size))
197     return res

```

200+ lines!

Detection frameworks

MMDetection

MMDetection

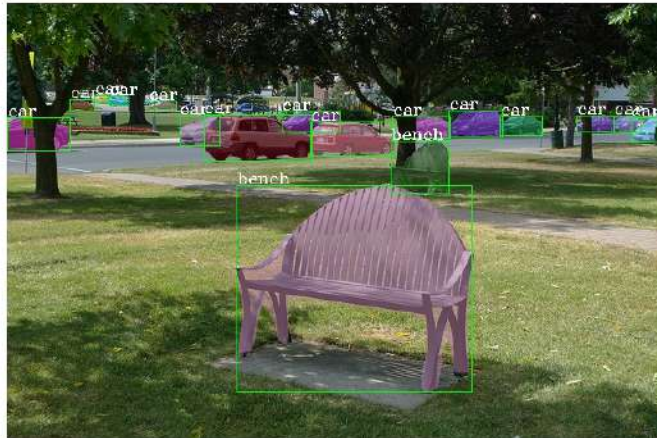
News: We released the technical report on [ArXiv](#).

Documentation: <https://mmdetection.readthedocs.io/>

Introduction

The master branch works with **PyTorch 1.1 to 1.4**.

mmdetection is an open source object detection toolbox based on PyTorch. It is a part of the open-r by [Multimedia Laboratory, CUHK](#).



Detectron



Segmentation frameworks

MMSegmentation



Documentation: <https://mmssegmentation.readthedocs.io/>

Introduction

MMSegmentation is an open source semantic segmentation toolbox based on PyTorch. It is a part of the OpenMMLab project.

The master branch works with **PyTorch 1.3 to 1.6**.

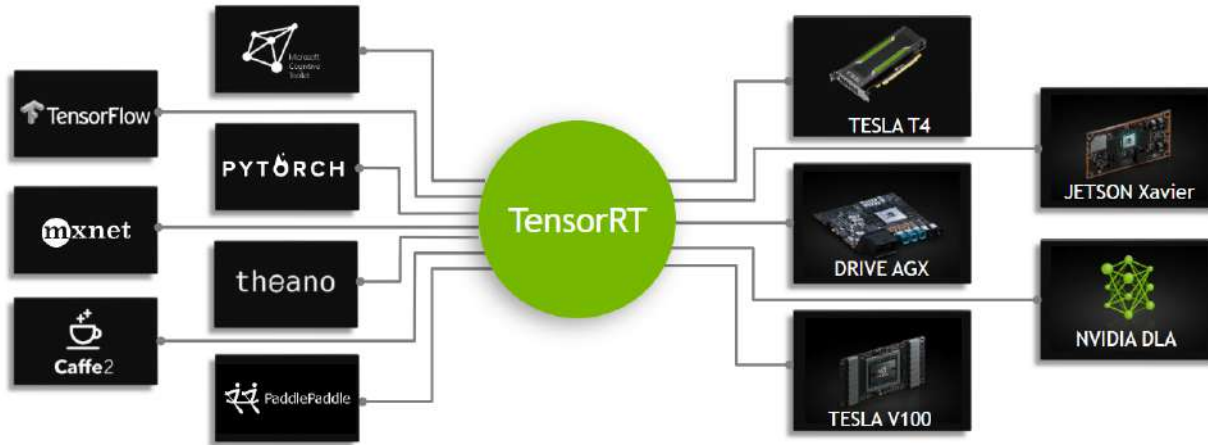


Detectron



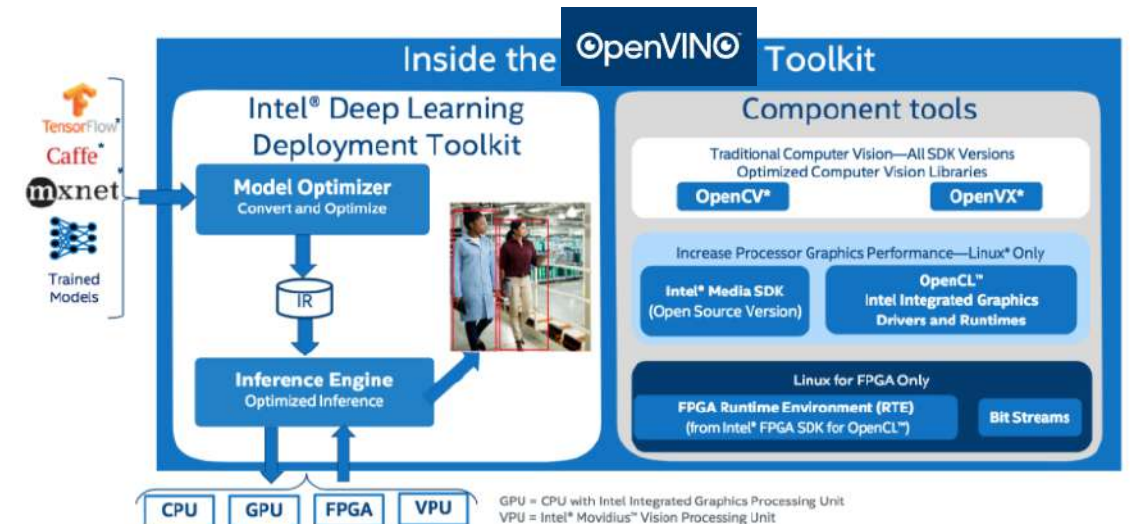
Inference acceleration

GPU



- 40x faster than CPU
- Model Zoo

CPU



- 20x faster than CPU
- Model optimization
- Model Zoo

Clouds for training

Colab.google



- <https://colab.research.google.com>
- 12h session
- Notebooks saved to google drive
- Not full jupyter keyboard shortcuts

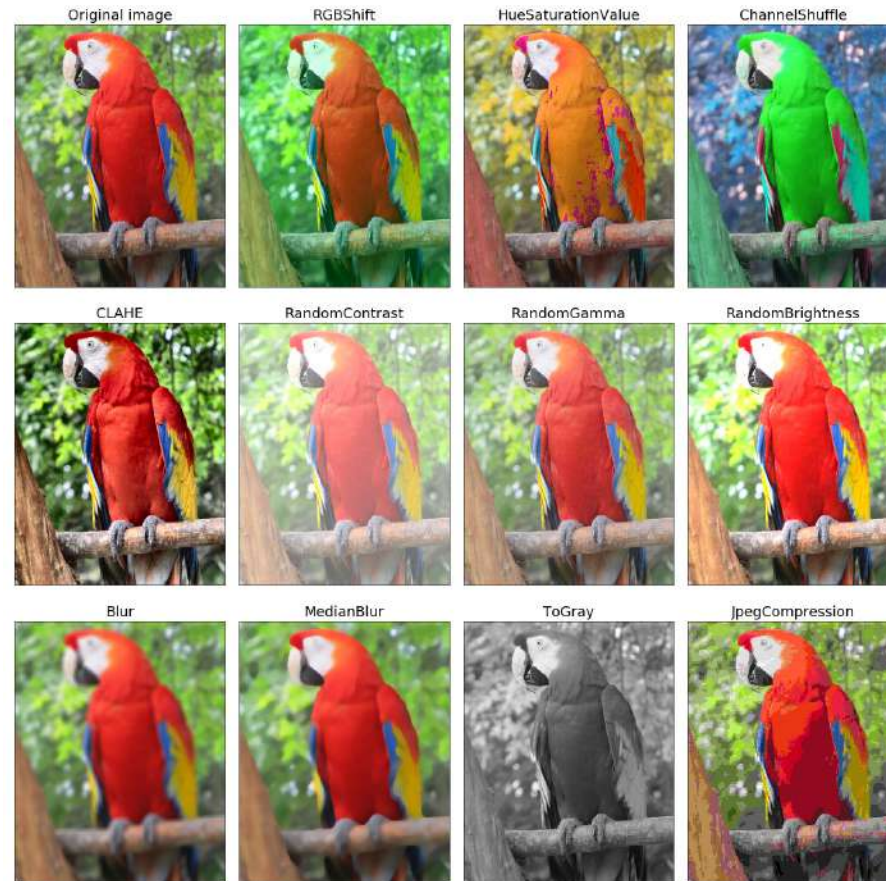
Kaggle kernels

kaggle

- <https://www.kaggle.com/notebooks>
- 9h session
- Notebooks easy savings
- Full jupyter keyboard shortcuts

Preprocessing and augmentations

- [Albumentations](#)
- FastAI preprocessing
- Kornia [augmentations](#)
- [NVIDIA DALI](#) (on GPU)



Acknowledgments

- Презентация подготовлена на основе слайдов Andrej Karpathy, Артура Кузина, Антона Конушина, Андрея Фильченкова, Всеволода Коняхина

ВОПРОСЫ? ПИШИТЕ

Наталья Ханжина
@natkaha



Роман Лебедев
@lebedev_rv

