ITMO.Hack

ITMO ML LAB

BOTAN INVESTMENTS

SOTA ML School

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

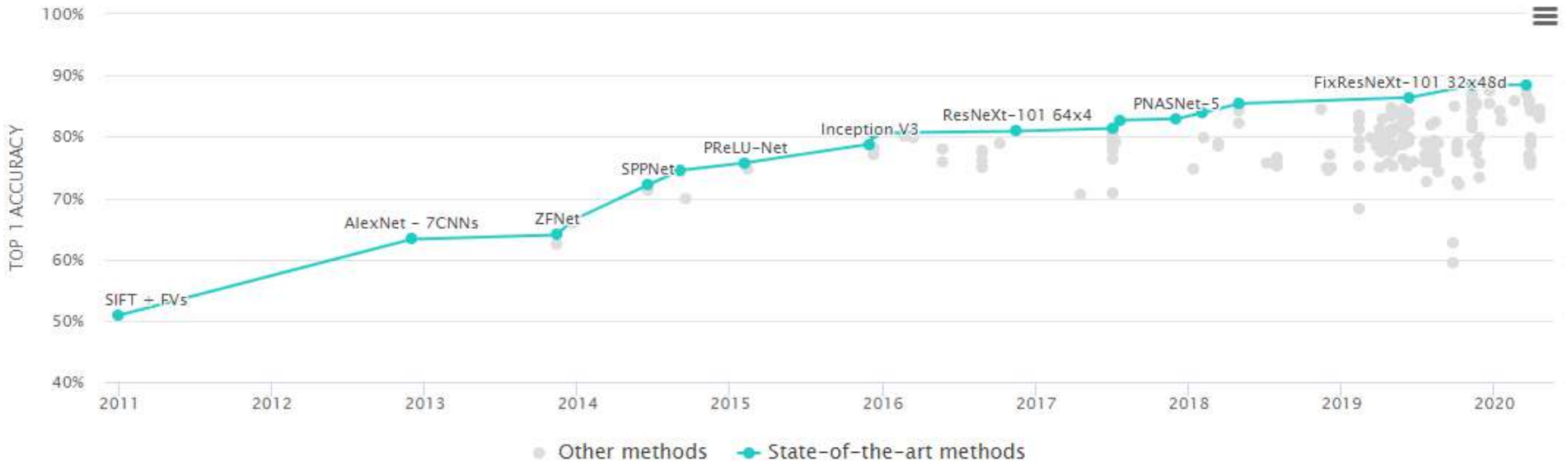
Short history of computer vision

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

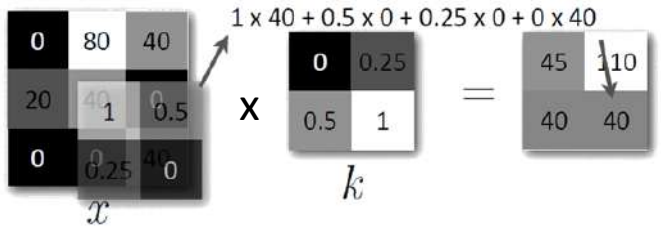


План лекции

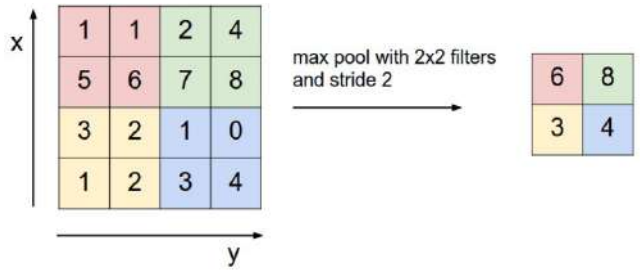
- Краткая история компьютерного зрения
- Классификация изображений
  - Модули
  - SOTA модели
  - Регуляризация
  - Наборы данных
- Детекция изображений
  - Модули
  - Виды детекции и SOTA модели
  - Losses
  - Метрики
  - Наборы данных
- Сегментация
  - Виды сегментации
  - Модули
  - SOTA модели
 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

Базовые слои

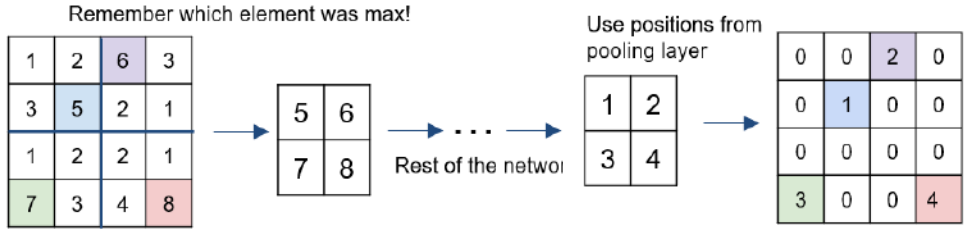
- Convolution



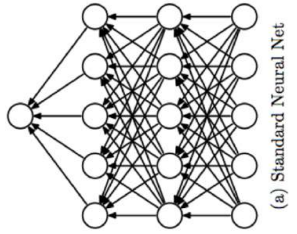
- Pooling



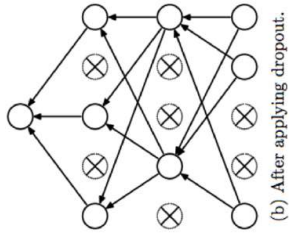
- UpSampling



- Dense (fully-connected)



- Dropout



- Batch normalization

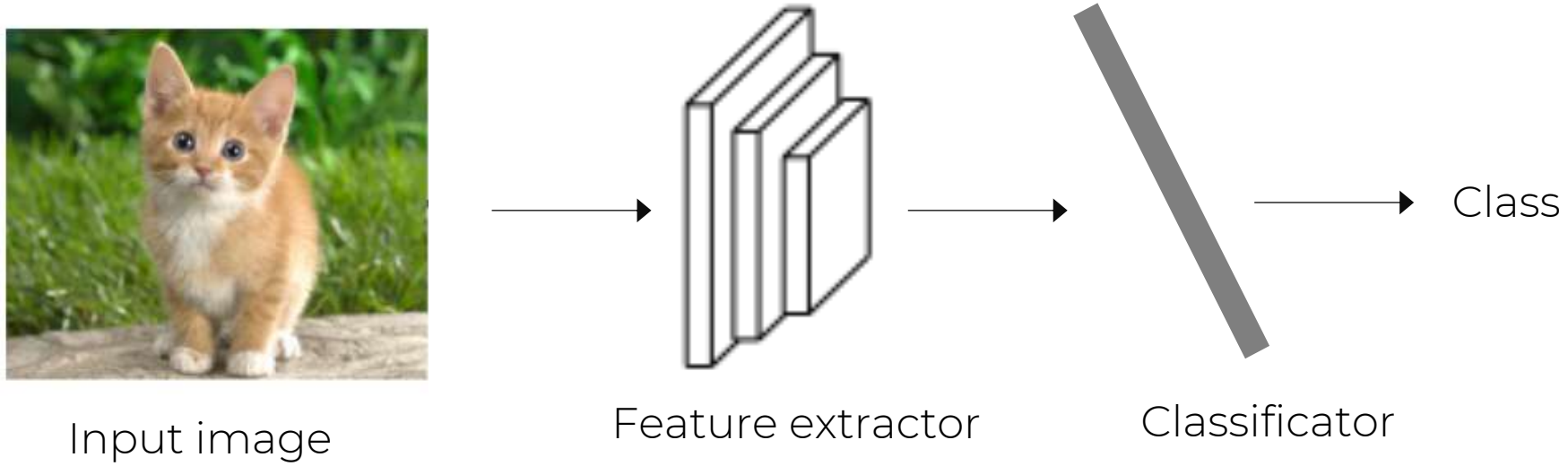
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Module

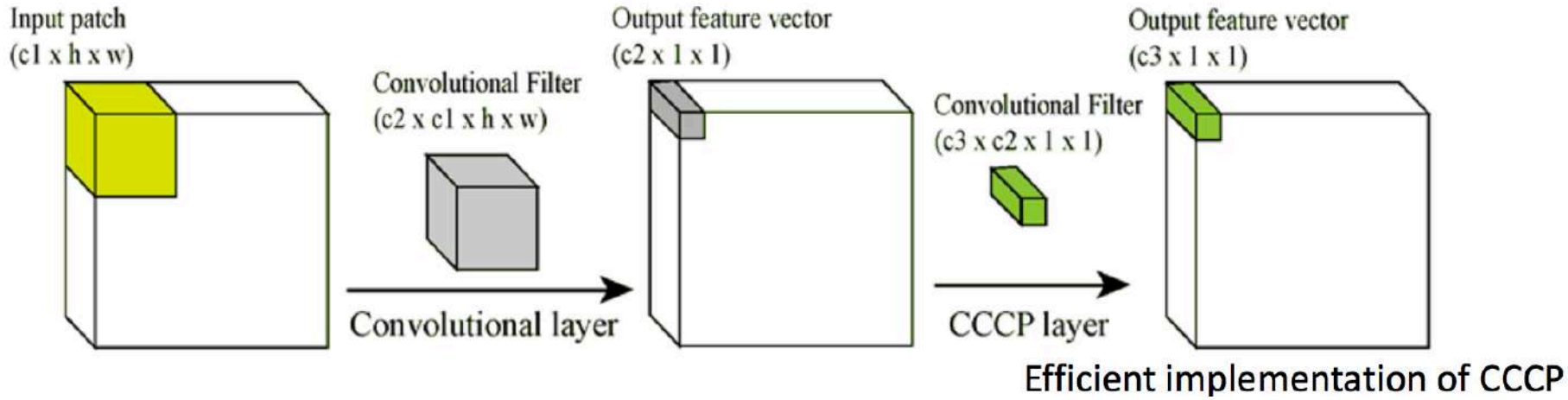
- **Модуль**, или блок – структурная единица современных моделей нейросетей
- Каждый модуль – объект или функция $a = h(x; \theta)$, которая
  - Имеет настраиваемые параметры(θ)
  - В качестве аргумента принимает входной сигнал x
  - Возвращает выходное значение a, пропущенное через функцию активации 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?

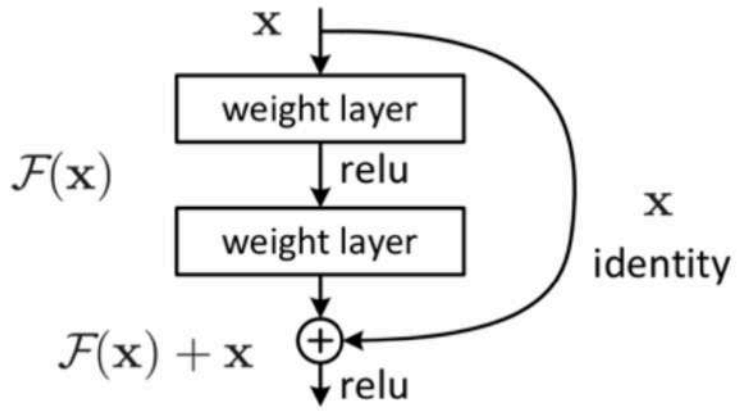


Figure 2. Residual learning: a building block.

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

Let's pretend we want to learn

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

instead.

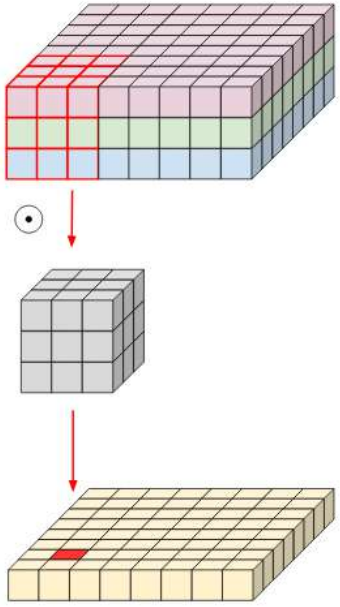
The original function is then

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

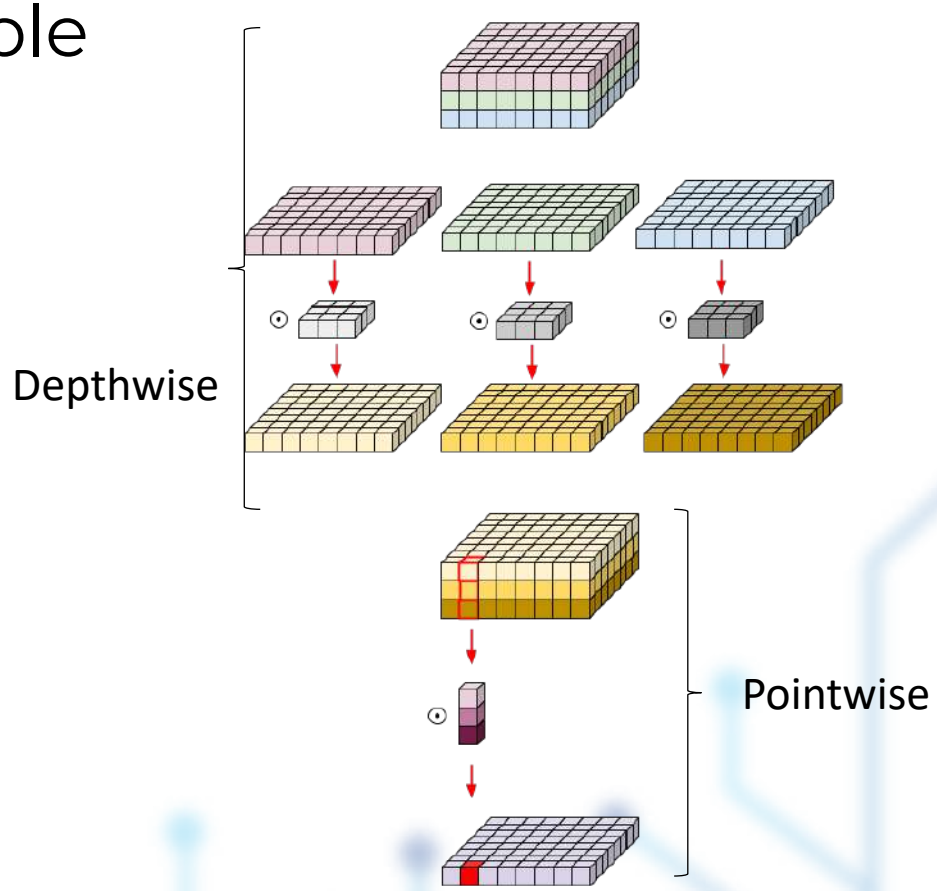
- ResNets

Depth-wise Separable Convolution (2017)

Conv2d



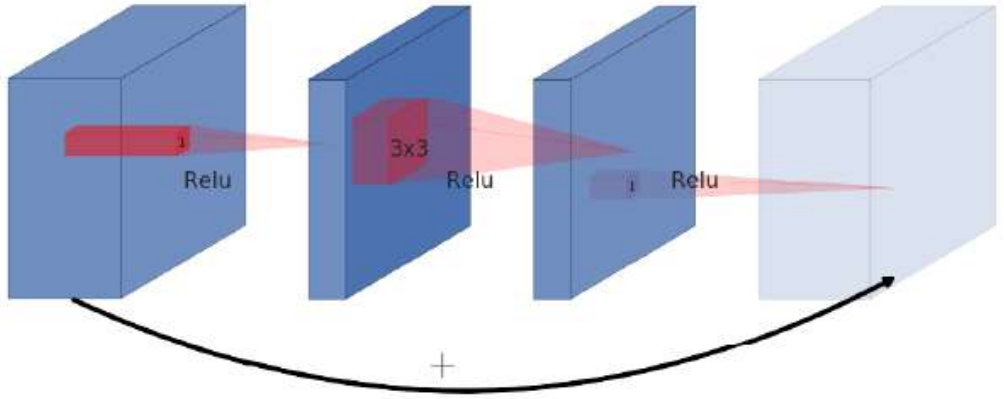
Separable Conv2d



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

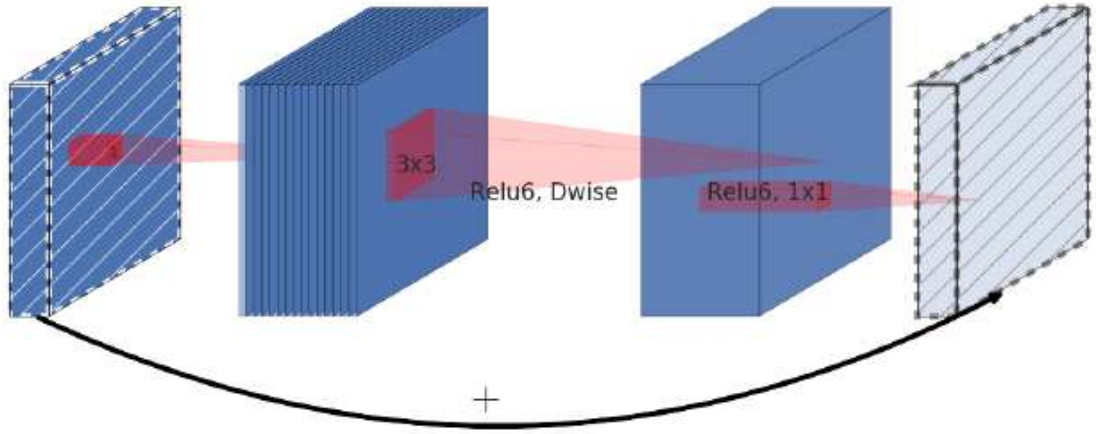
Residual Inverted block (2018)

Residual block



wide-narrow-wide

Inverted Residual block

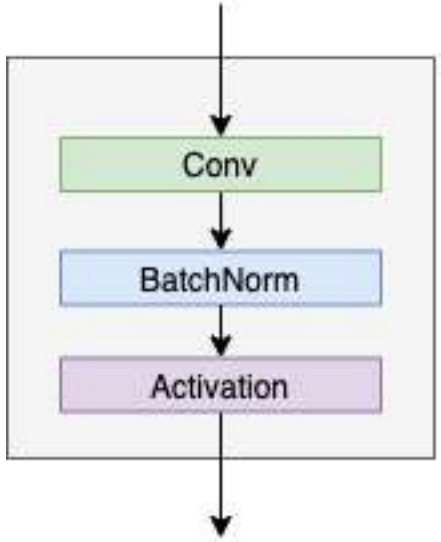


narrow-wide-narrow

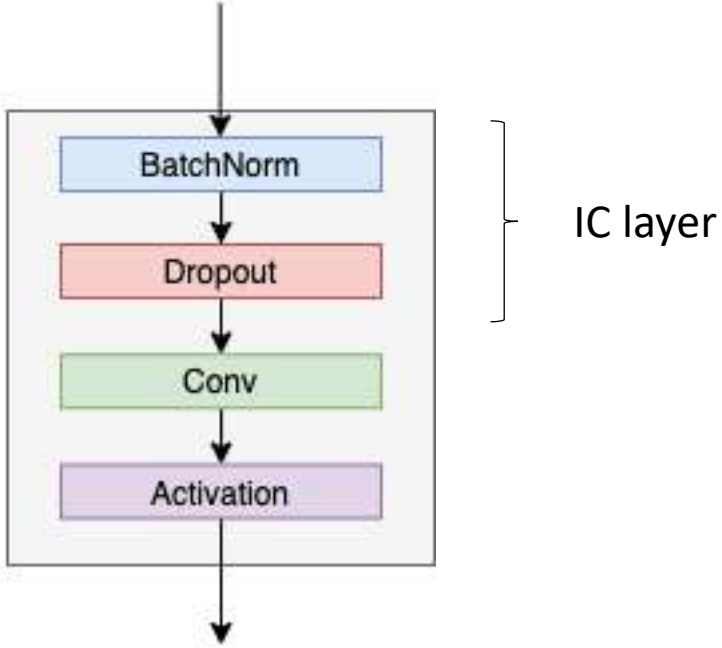
- Меньше параметров
- MobileNetV2, EfficientNet

Independent-Component (IC) layer (2019)

BatchNorm conv block



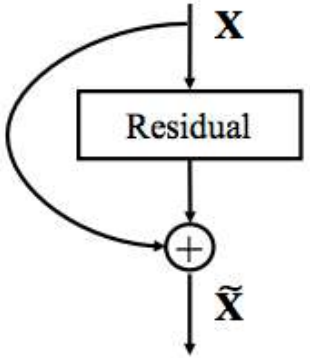
IC block



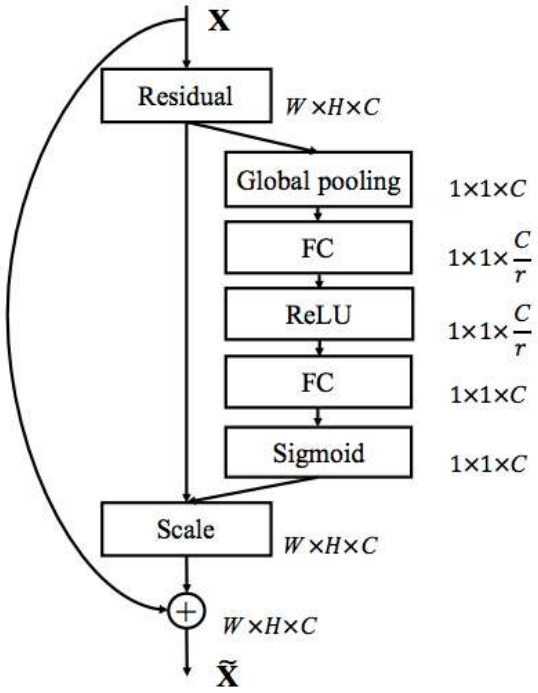
- +3% accuracy на CIFAR100
- Tested on ResNet

Squeeze-and-Excitation block (2017)

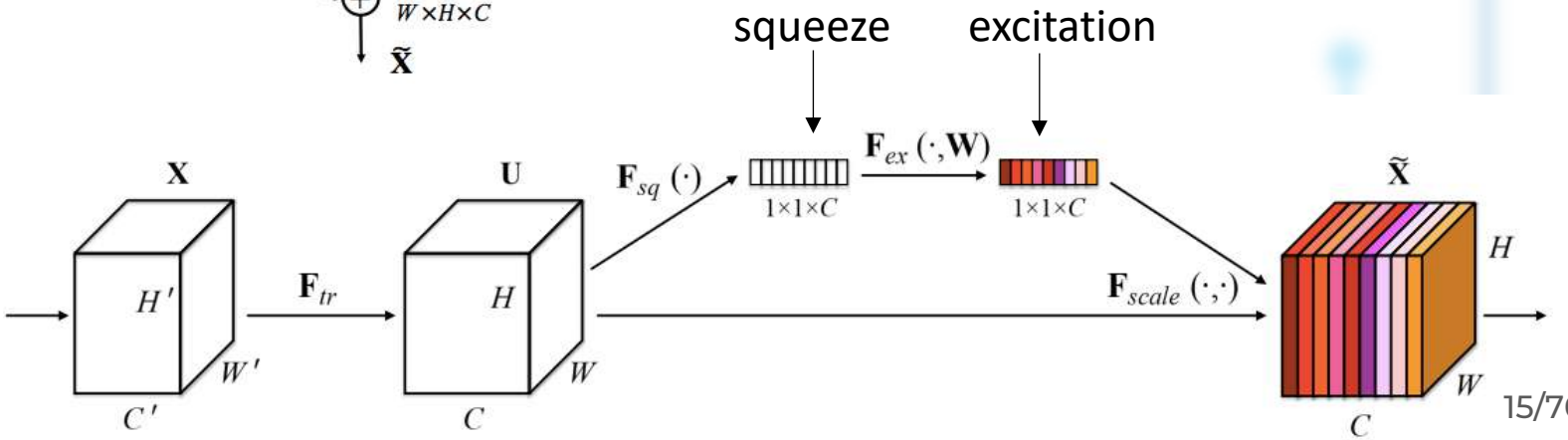
ResNet module



SE-ResNet module

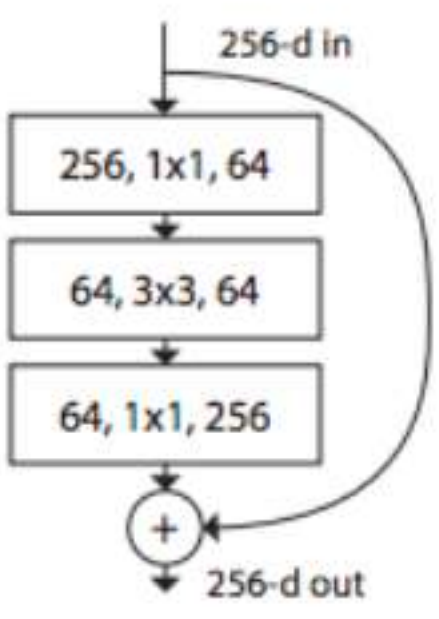


- Attention для картинок
- 25% boost on ImageNet
- SENets, EfficientNet DeepSEED...

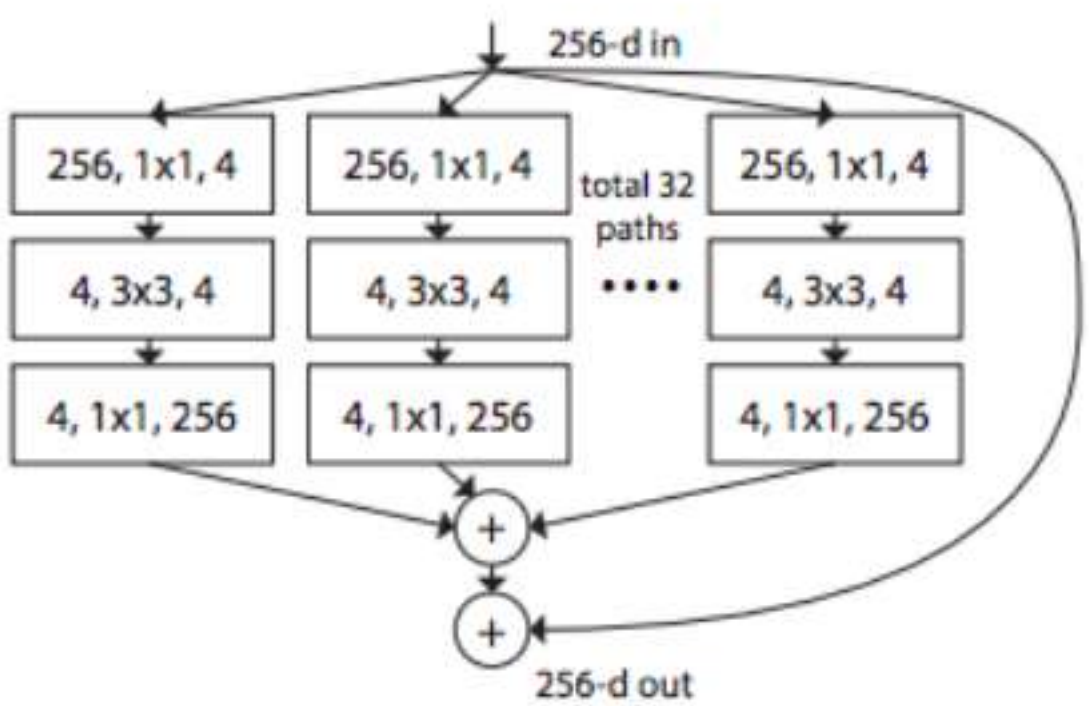


ResNeXt (2016)

ResNet block

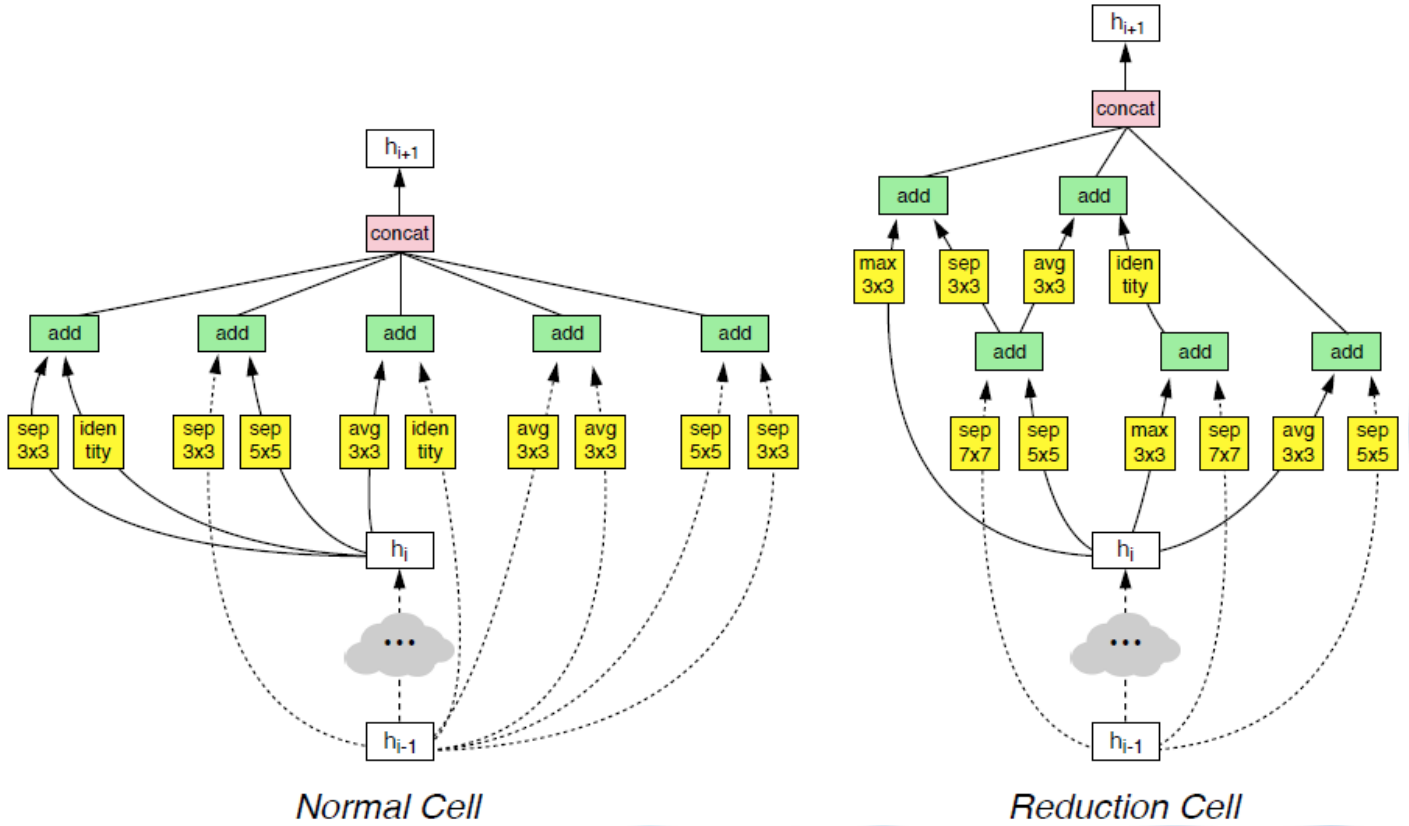


ResNeXt shortcut

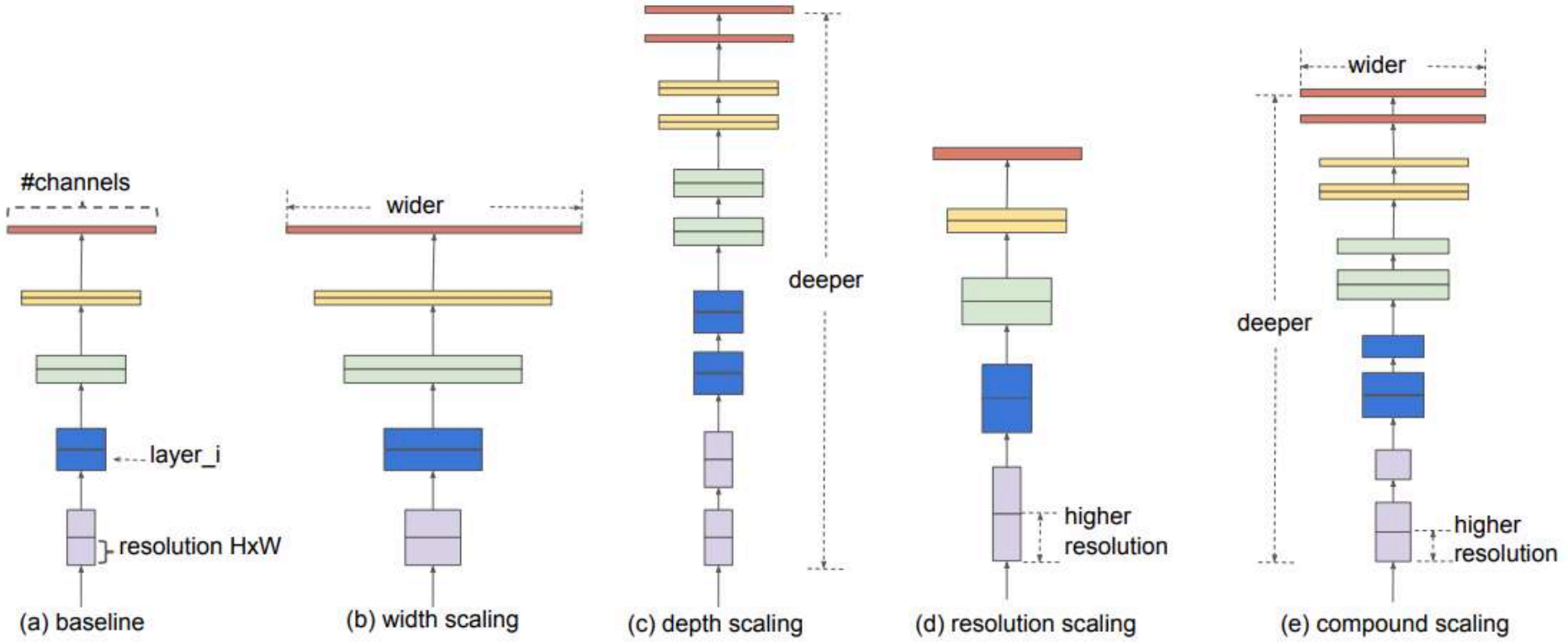


NASNet (2018)

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



EfficientNet (2019)

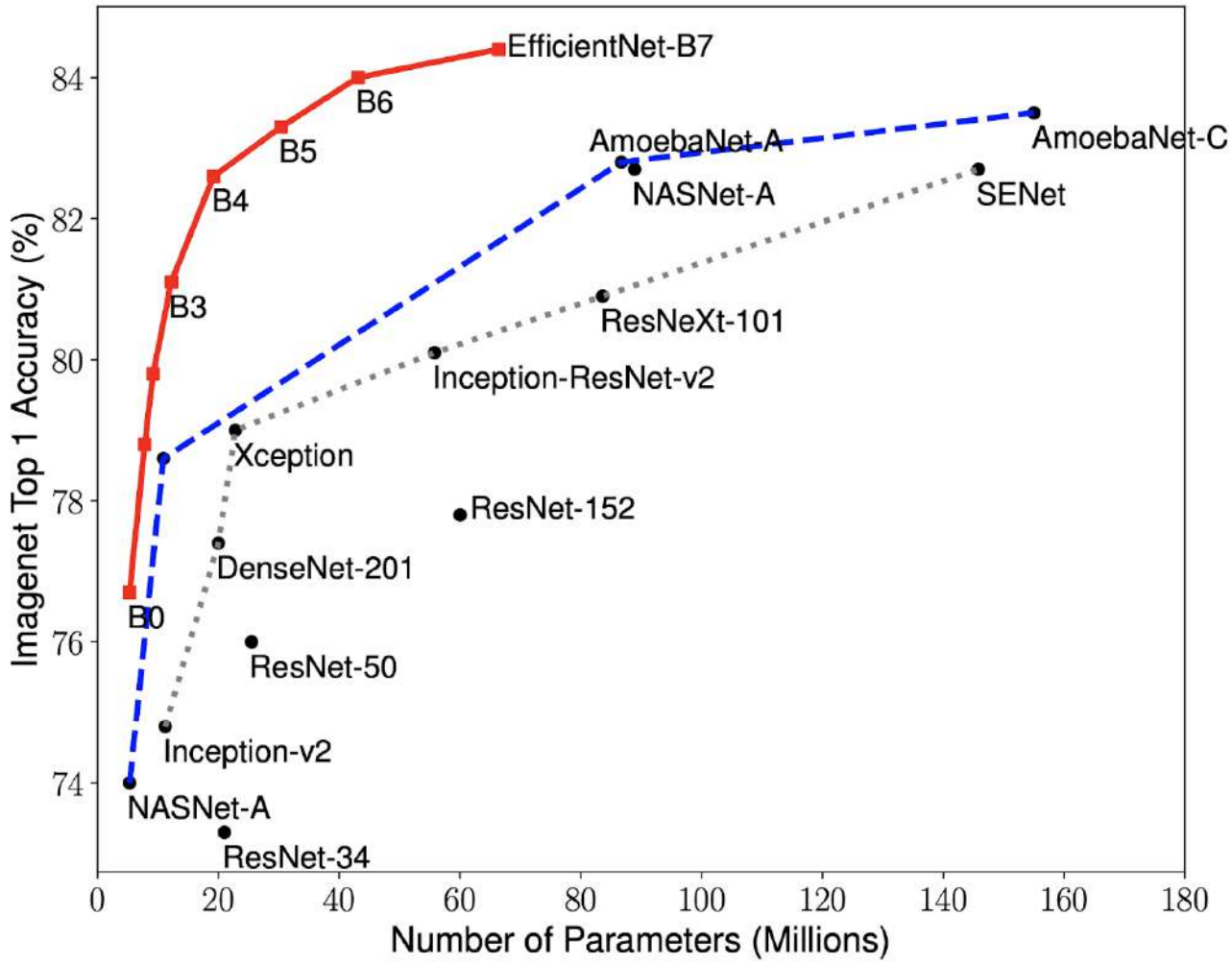


- Top accuracy on ImageNet (84,4%)
- Использует Compound model scaling coefficient ϕ

Классификация: SOTA model zoo

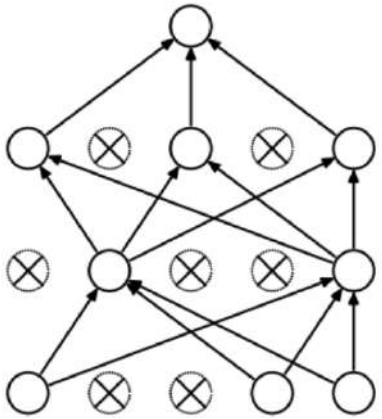
- Классификация:
  - EfficientNet
  - AmoebaNet
  - ResNeXt
  - NASNet
  - SENet
  - Inception-ResNet
  - Xception
  - MobileNetV2

Network comparison

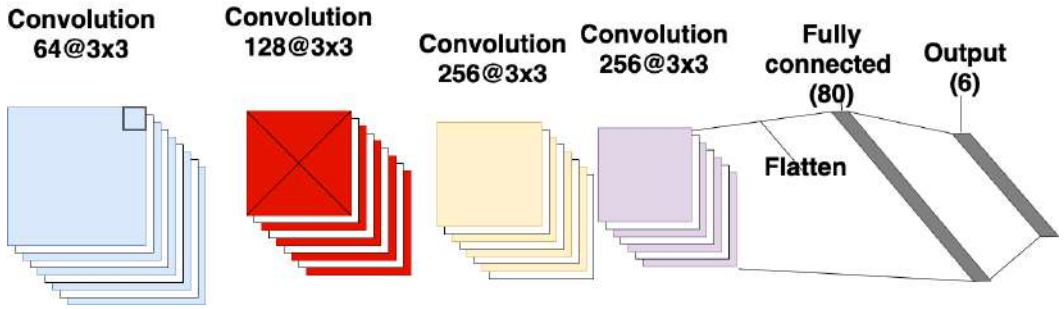


Регуляризация

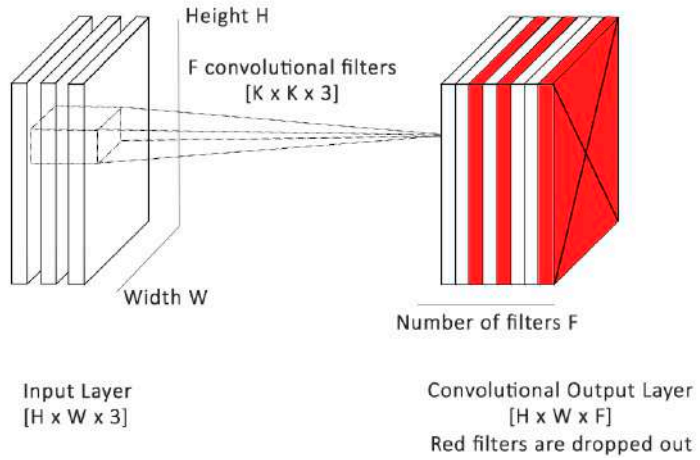
Dropout (2014)



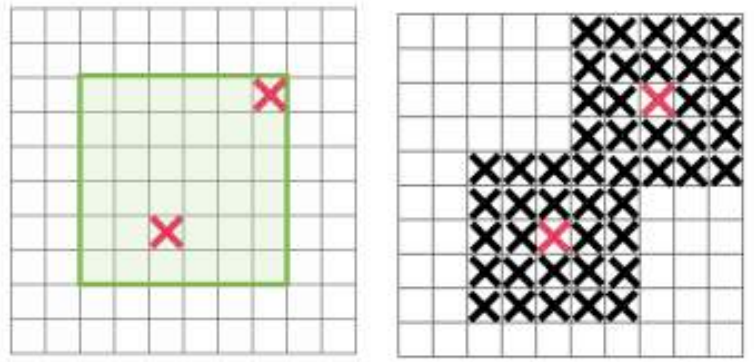
DropLayer (2016)



DropChannel (2018)



DropBlock (2015)



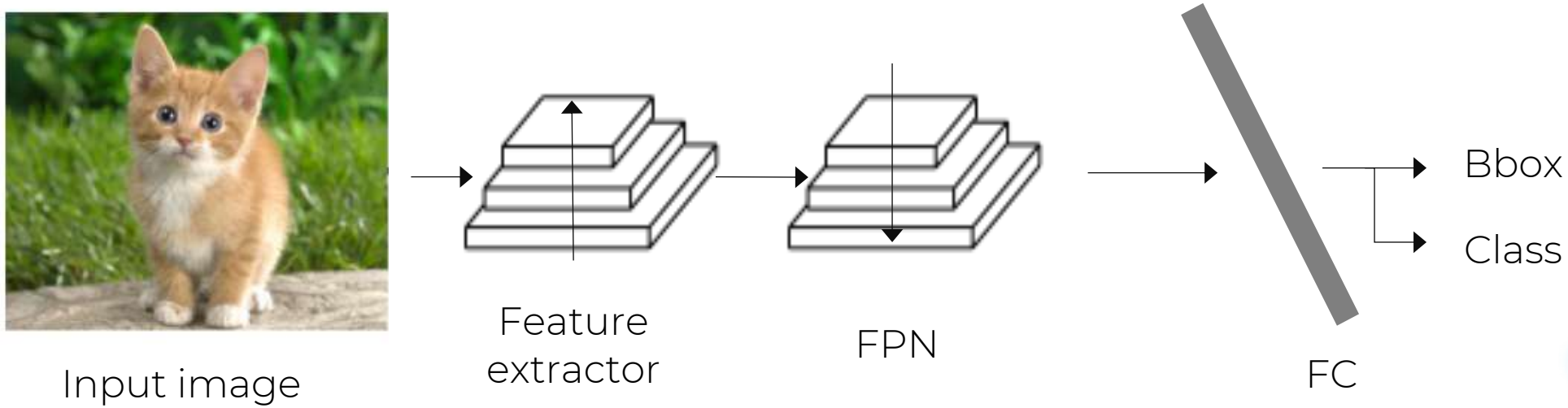
Benchmarks for classification

ImageNet(ILSVRC)	1000 classes	data
CIFAR	100/10 classes	data
MNIST	10 classes	data
FashionMNIST	10 classes	data
Food-11	11 classes	data
KITTI Vision Benchmark Suite	8 classes	data
Visual Object Classes 2012 (VOC12)	20 classes	data
Common Objects in Context (COCO)	80 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



Quality Assessment and Metrics

- mean average precision (mAP)
  - Для каждого класса ci 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 a_{ovl} между *predicted bounding box* BB_p и *ground truth bounding box* BB_{gt} должна быть больше 0,5 согласно формуле

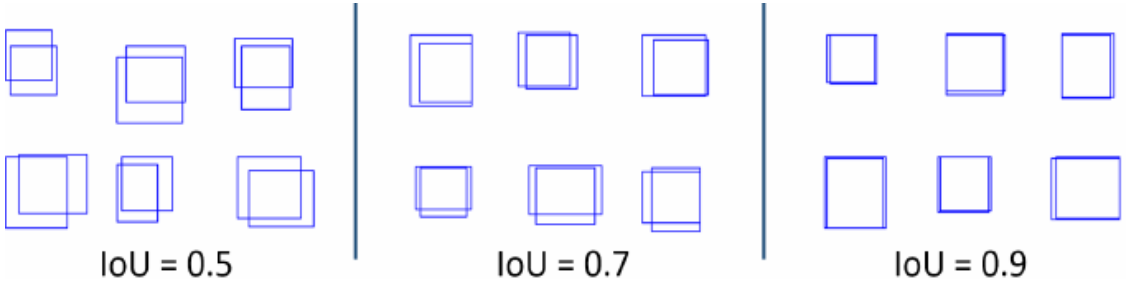
$$area_{ovl} = \frac{area(BB_p \cap BB_{gt})}{area(BB_p \cup BB_{gt})} \quad (1)$$

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

Quality Assessment and Metrics

IoU

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



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 $C \subseteq \mathbb{S} \in \mathbb{R}^n$

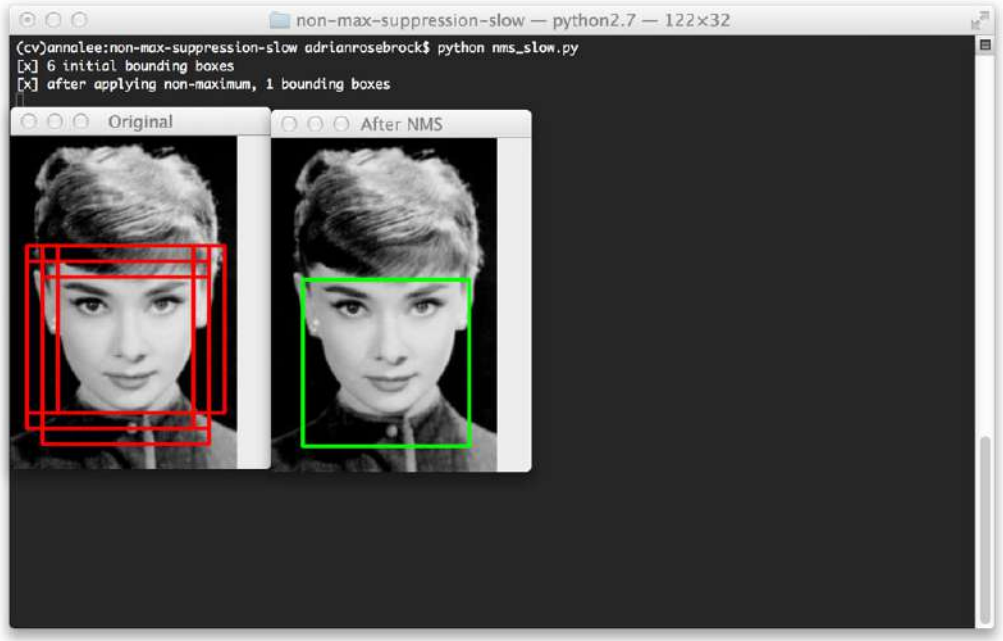
2 $IoU = \frac{|A \cap B|}{|A \cup B|}$

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

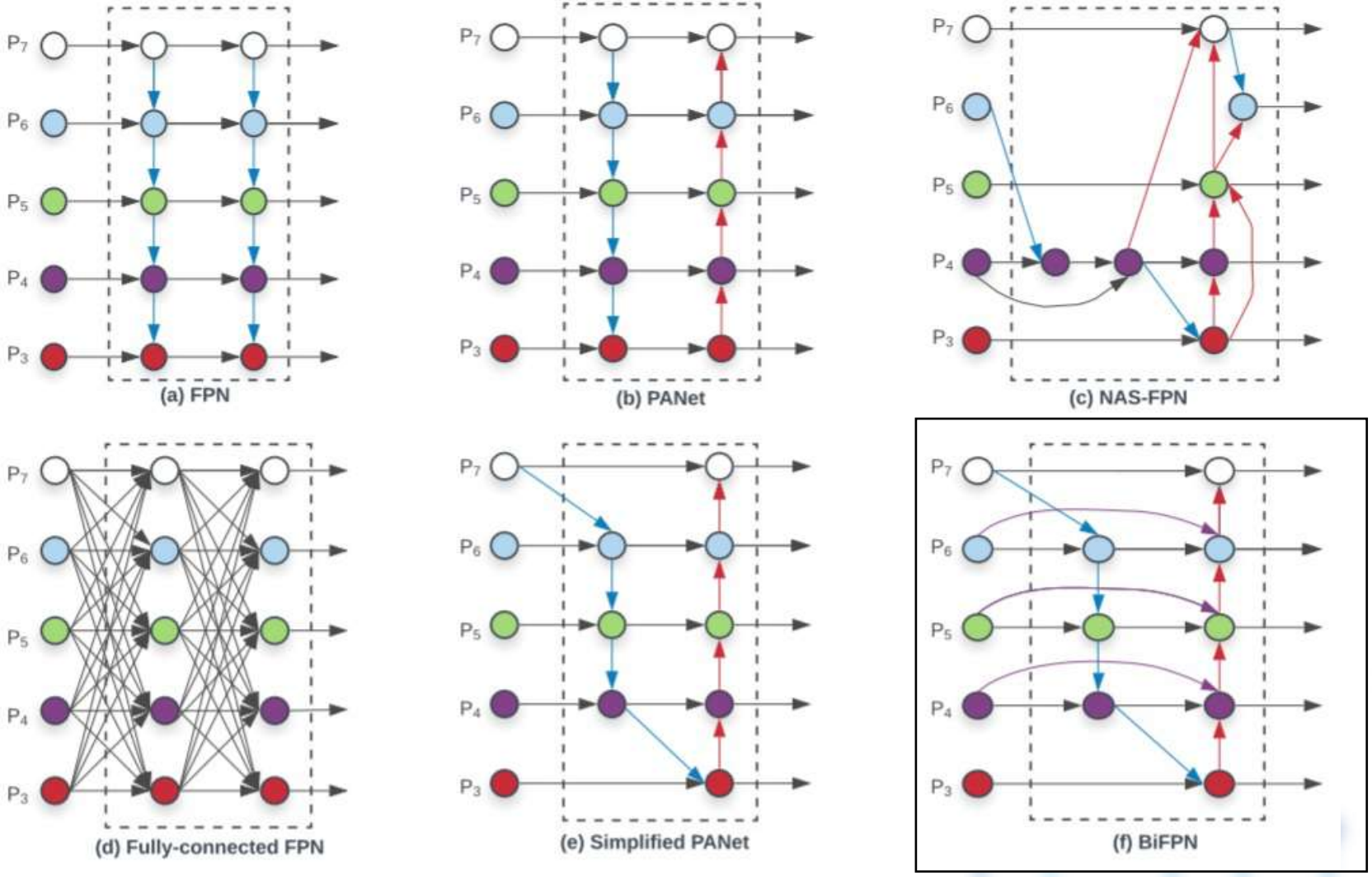
Quality Assessment

- 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]
```

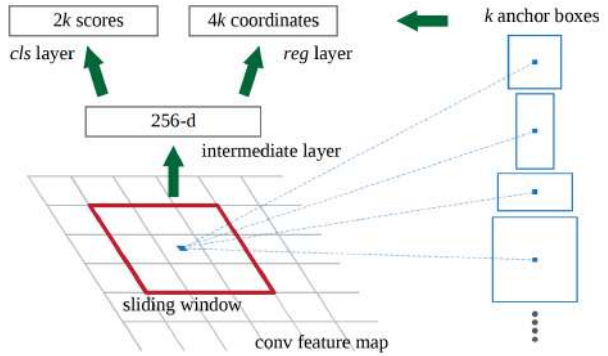


Feature pyramid networks

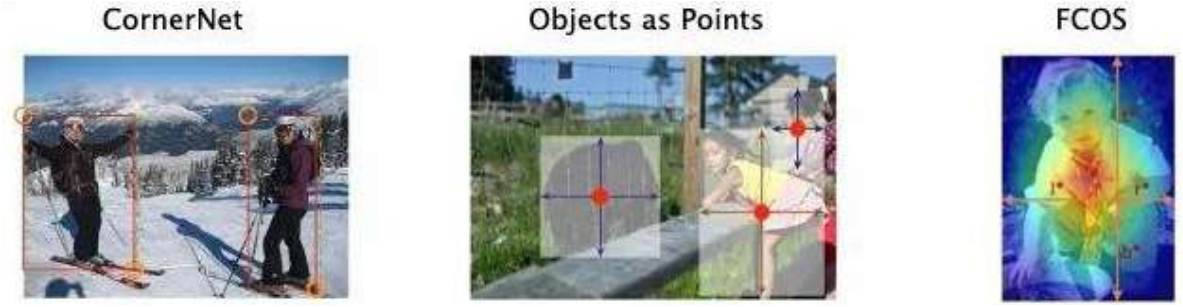


Model zoo

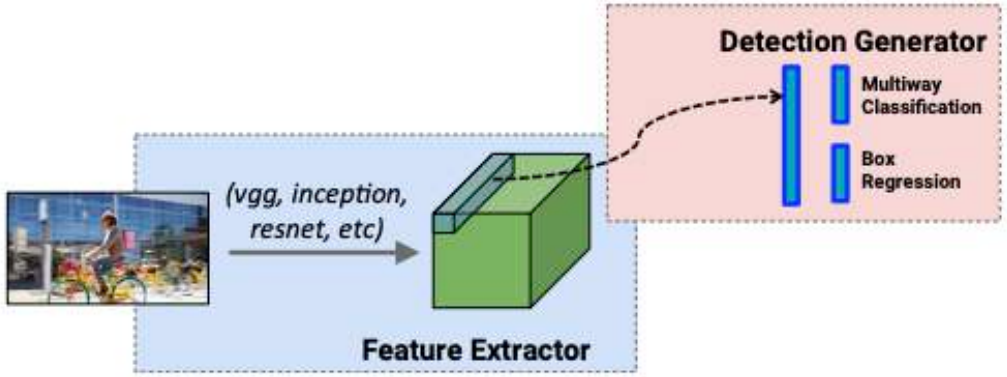
- Anchor-based:



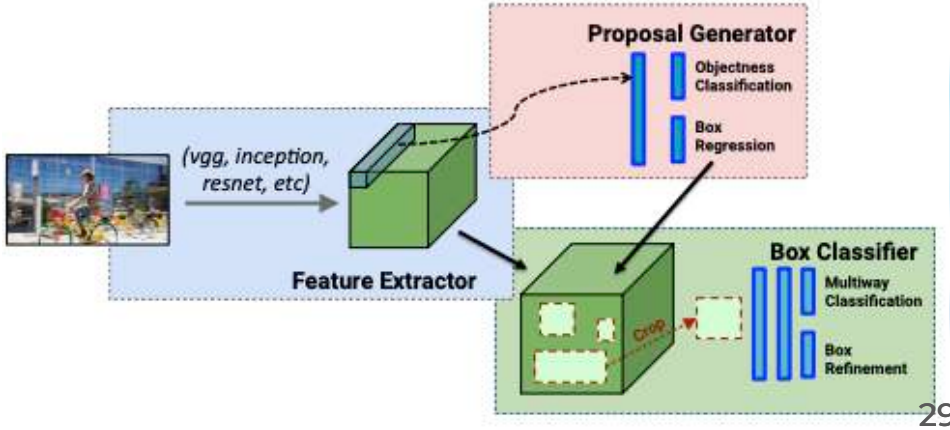
- Key-point-based:



- One-shot:



- Two-shot:



Model zoo

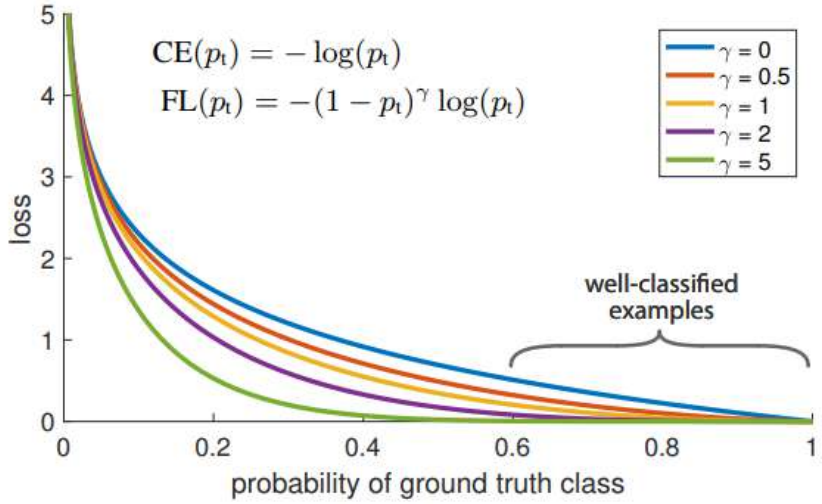
- Anchor-based:
  - EfficientDet
  - YOLO v4
  - RetinaNet
  - SSD (on MobileNet v2)
- Key-point-based:
  - CornerNet
  - CenterNet
  - FCOS
- One-shot:
  - YOLO v4
  - RetinaNet
  - RPN
  - FCOS
  - SSD
- Two-shot:
  - Faster R-CNN
  - R-FCN

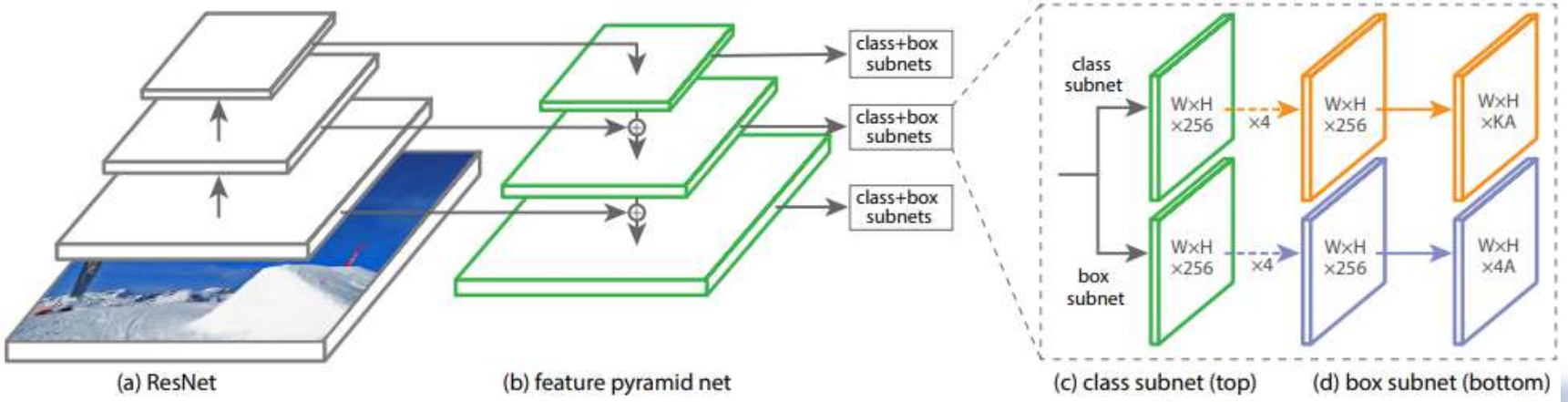
RetinaNet (2017)

- Использует Focal loss
- Использует FPN

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

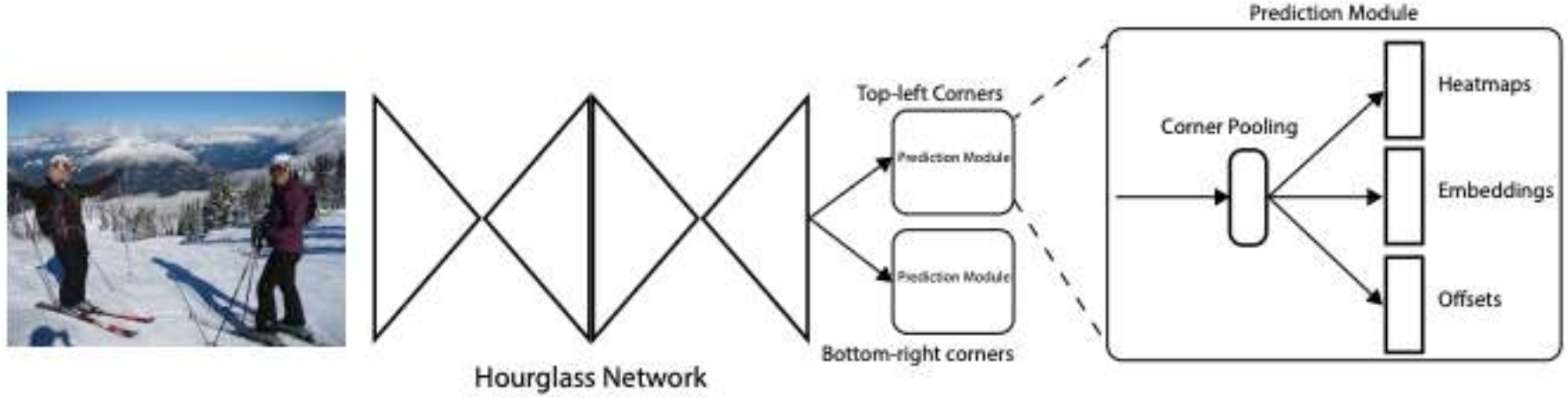
$$\mathrm{FL}(p, y) = \begin{cases} -\alpha(1-p)^{\gamma}\log p, & \text{if } y = 1 \\ -p^{\gamma}\log(1-p), & \text{otherwise.} \end{cases}$$

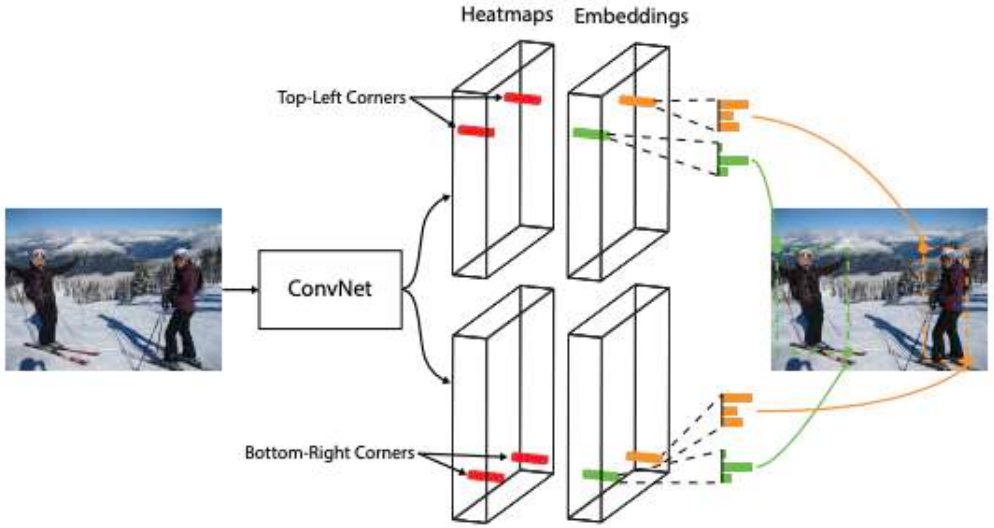




Lin, Tsung-Yi, et al. "Focal loss for dense object detection." IEEE transactions on pattern analysis and machine intelligence (2018).

CornerNet (2019)

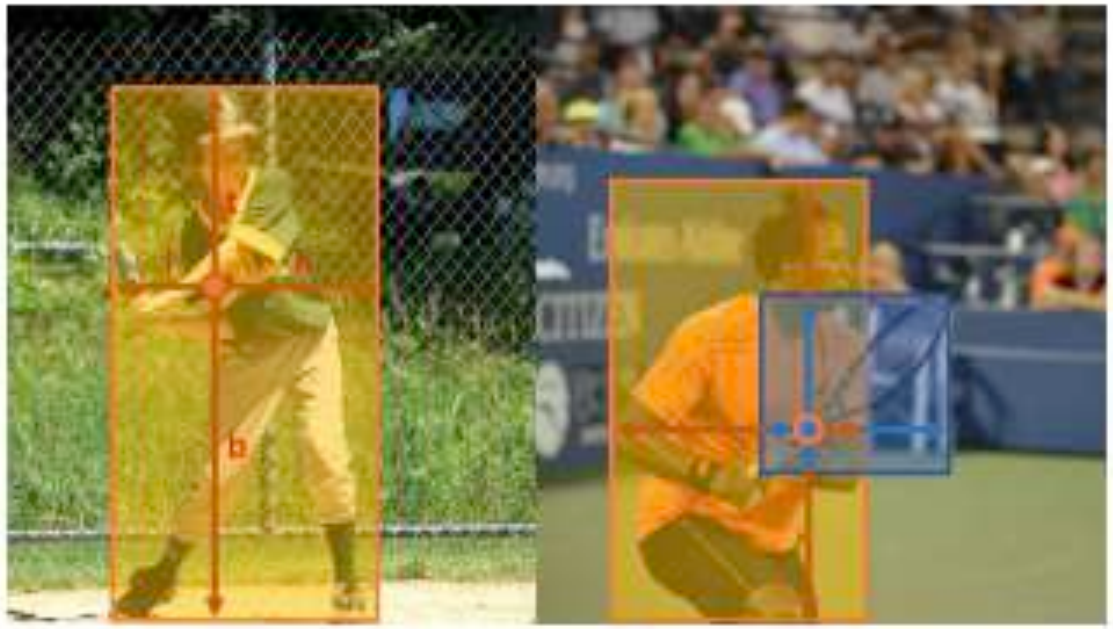




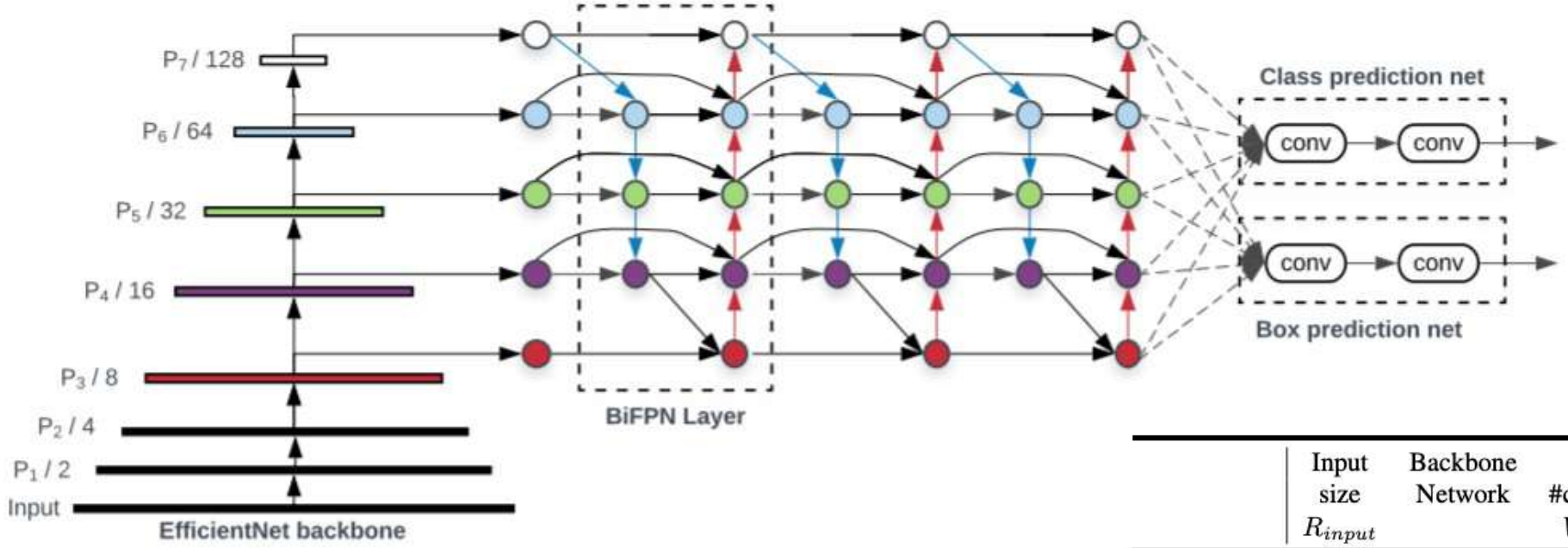
	AP	AP^{50}	AP^{60}	AP^{70}	AP^{80}	AP^{90}
RetinaNet (Lin et al., 2017)	39.8	59.5	55.6	48.2	36.4	15.1
Cascade R-CNN (Cai and Vasconcelos, 2017)	38.9	57.8	53.4	46.9	35.8	15.8
Cascade R-CNN + IoU Net (Jiang et al., 2018)	41.4	59.3	55.3	49.6	39.4	19.5
CornerNet	40.6	56.1	52.0	46.8	38.8	23.4

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 фичи FCOS в RetinaNet

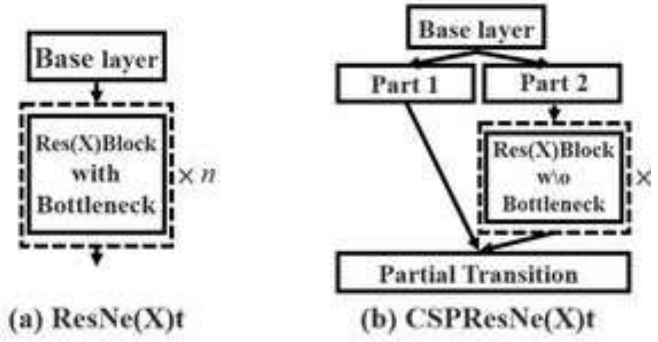
Inconsistency	FCOS	RetinaNet (#A=1)					
GroupNorm	✓		✓	✓	✓	✓	✓
GIoU Loss	✓			✓	✓	✓	✓
In GT Box	✓				✓	✓	✓
Centerness	✓					✓	✓
Scalar	✓						✓
AP (%)	37.8	32.5	33.4	34.9	35.3	36.8	37.0

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

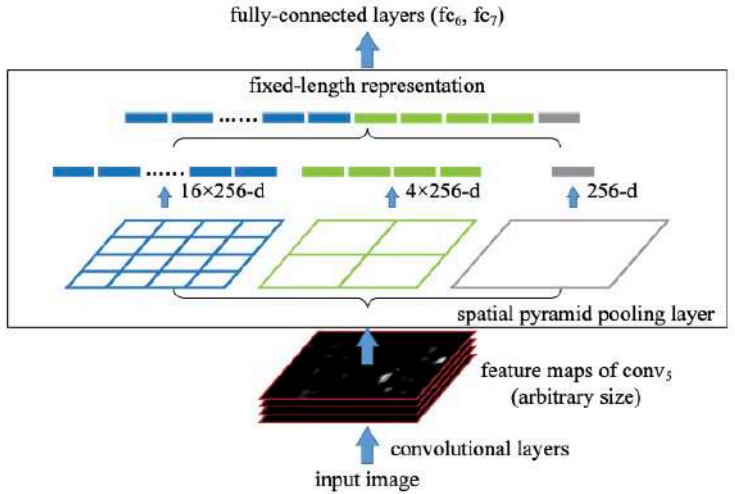
Classification \ Regression	Box	Point
Intersection over Union	37.0	36.9
Spatial and Scale Constraint	37.8	37.8

YOLOv4 (2020)

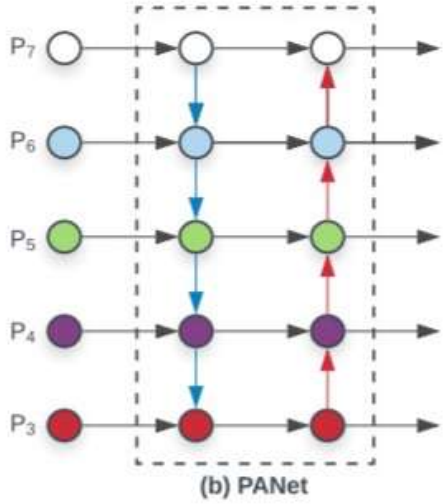
- CSPDarknet backbone



- SPP (Spatial Pyramid Pooling)



- Modified PANet as FPN



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

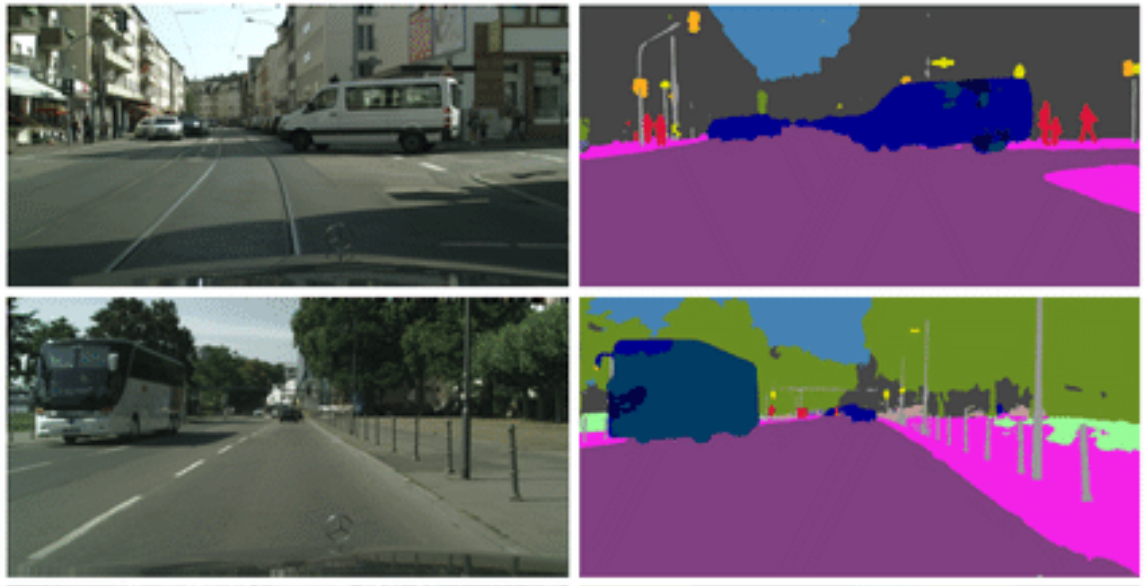
Public Datasets for detection

ICDAR	2013-2019	OCR	paper	data
COCO-Text	2016	OCR	paper	data
SynthText	2016	OCR	paper	data
LUNA16	2016	CT/lung	paper	data
SUN RGB-D	2015	indoor	paper	data
Common Objects in Context (COCO)	2014	general	paper	data
Oxford RoboCar Dataset	2014	street	paper	data
KITTI Vision Benchmark Suite	2012	street	paper	data
Visual Object Classes 2012 (VOC12)	2012	street	[1][2]	data
LIDC-IDRI	2011	CT/lung	paper	data
Caltech Pedestrian Detection Benchmark	2009	street	[1][2]	data

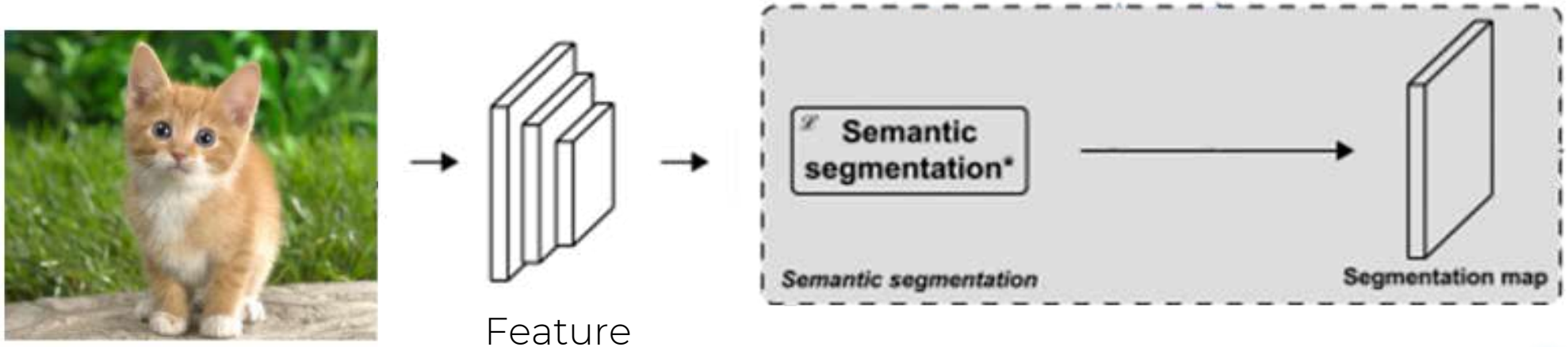
План лекции

- 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



Input image

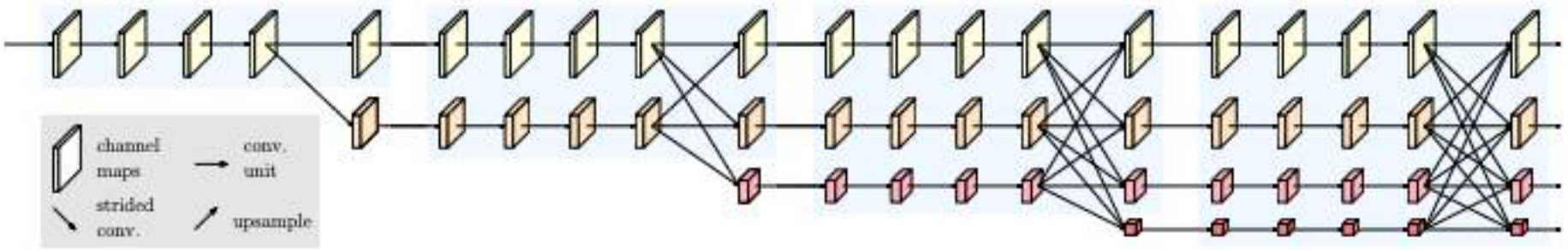
Feature extractor

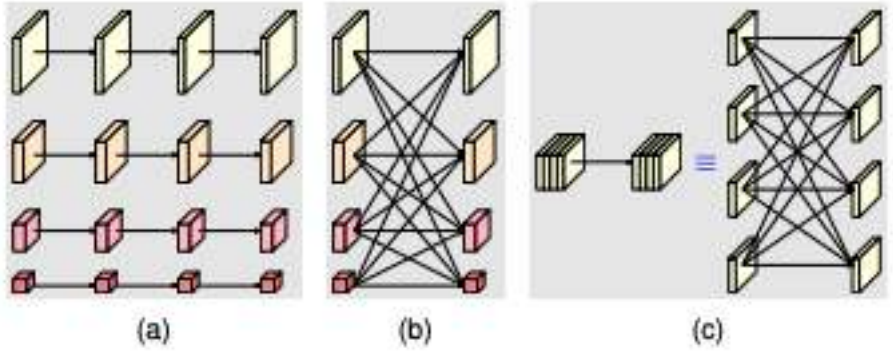
Model zoo

- Семантическая сегментация:
  - HRNet
  - PSPNet
  - UNet++
  - DeepLabV3
  - Mask R-CNN

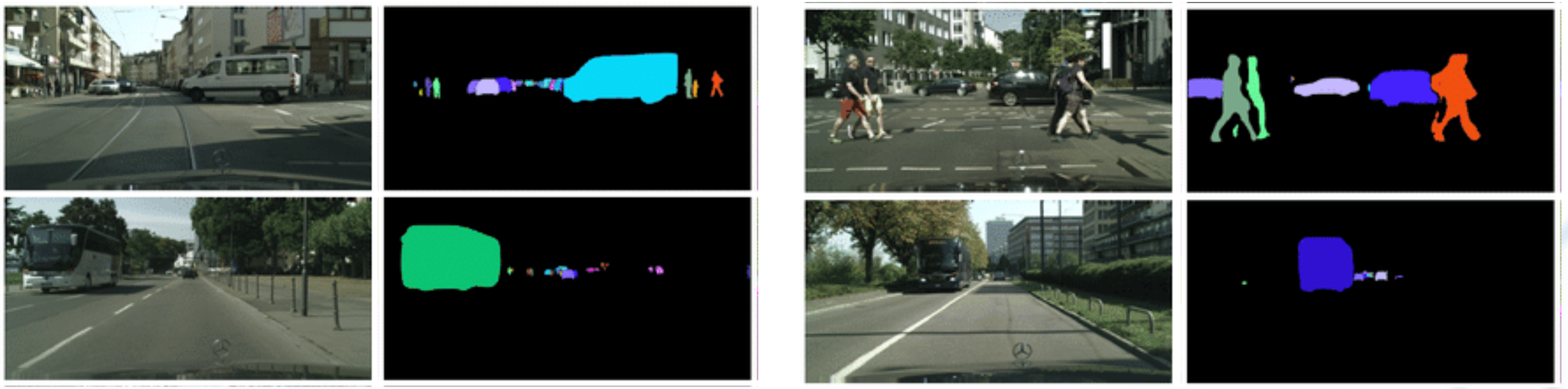
HRNet (2020)

- Построение признаков высокого разрешения





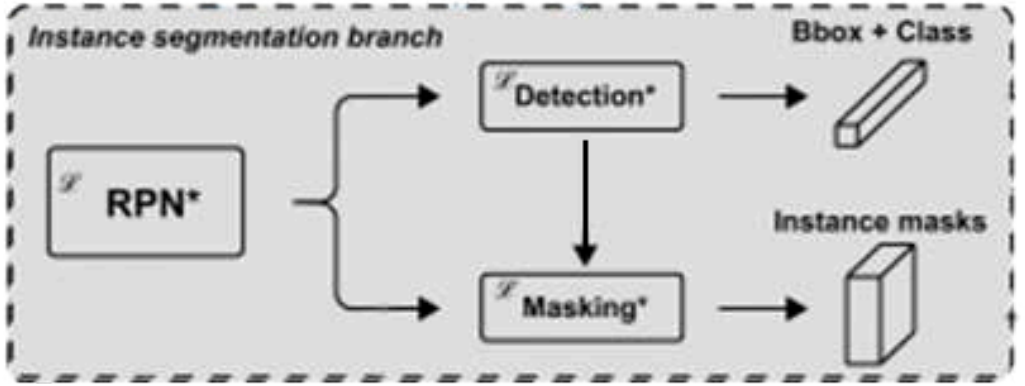
Instance segmentation



Basic instance segmentation model

Input image

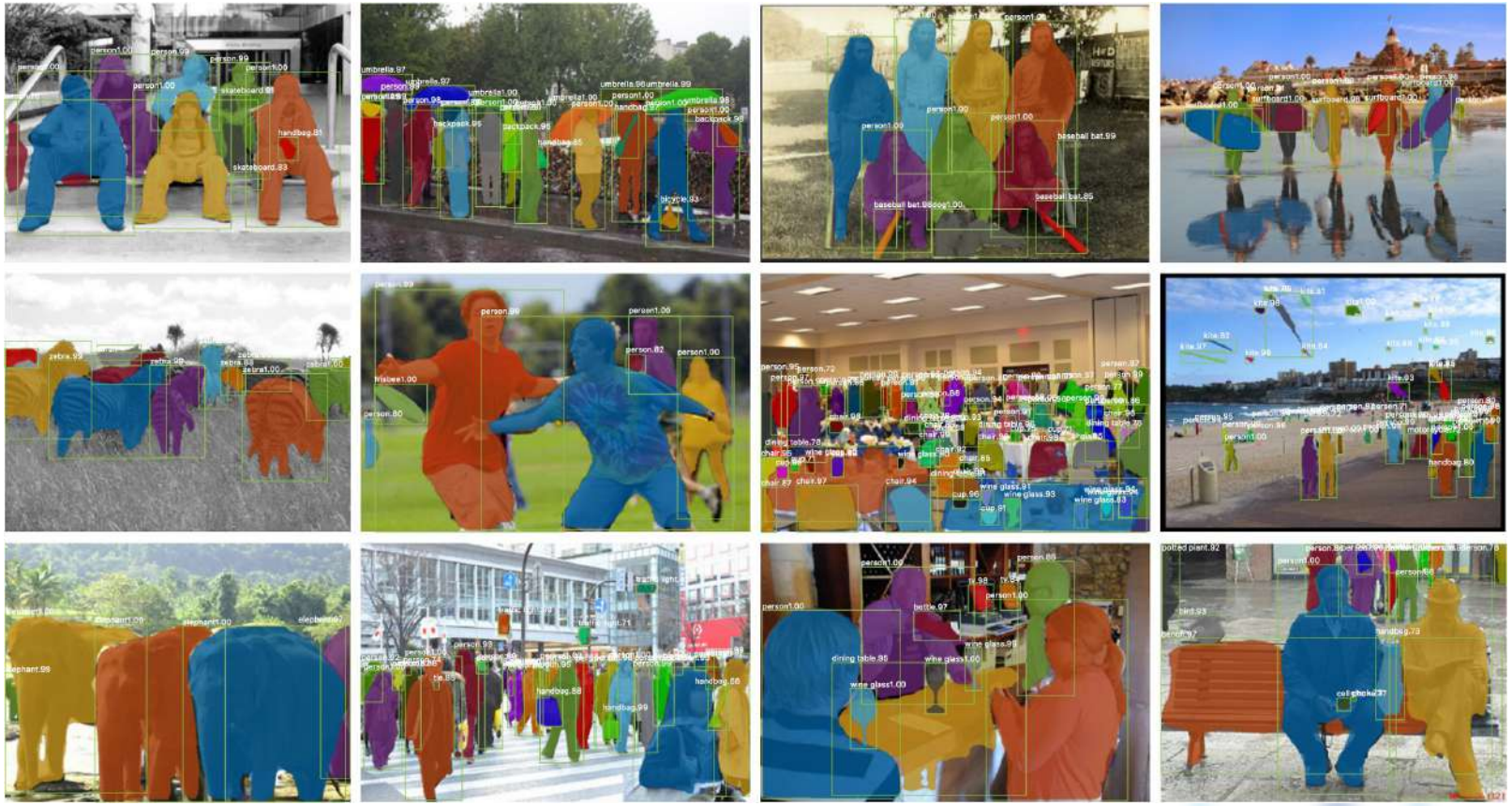
Feature extractor



Instance output

Mask R-CNN (2017)

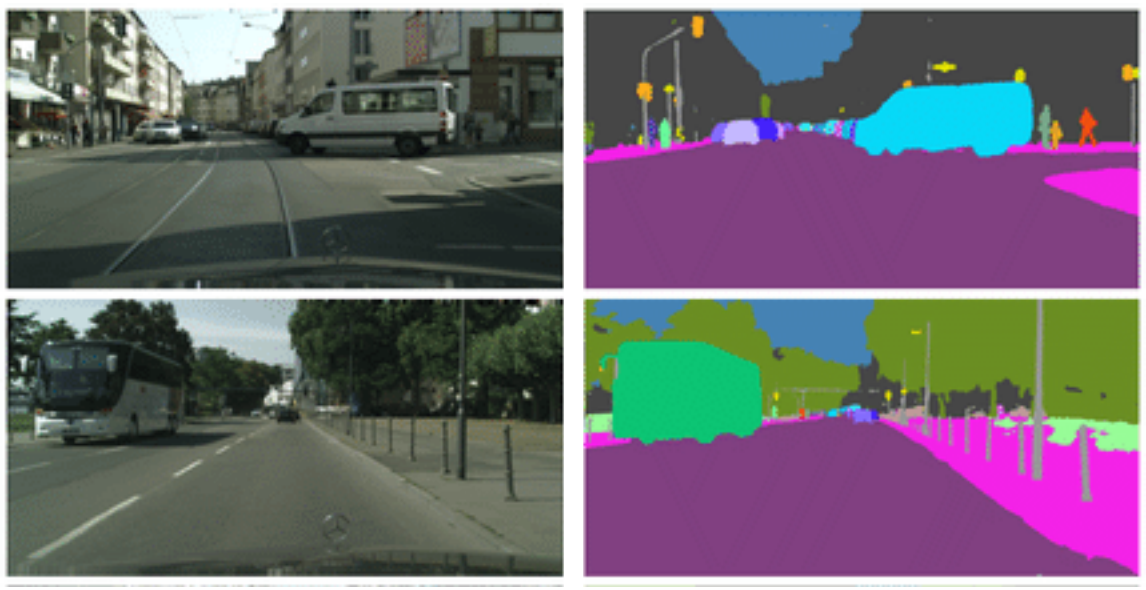
Results on COCO test images using ResNet-101-FPN at 5 fps.

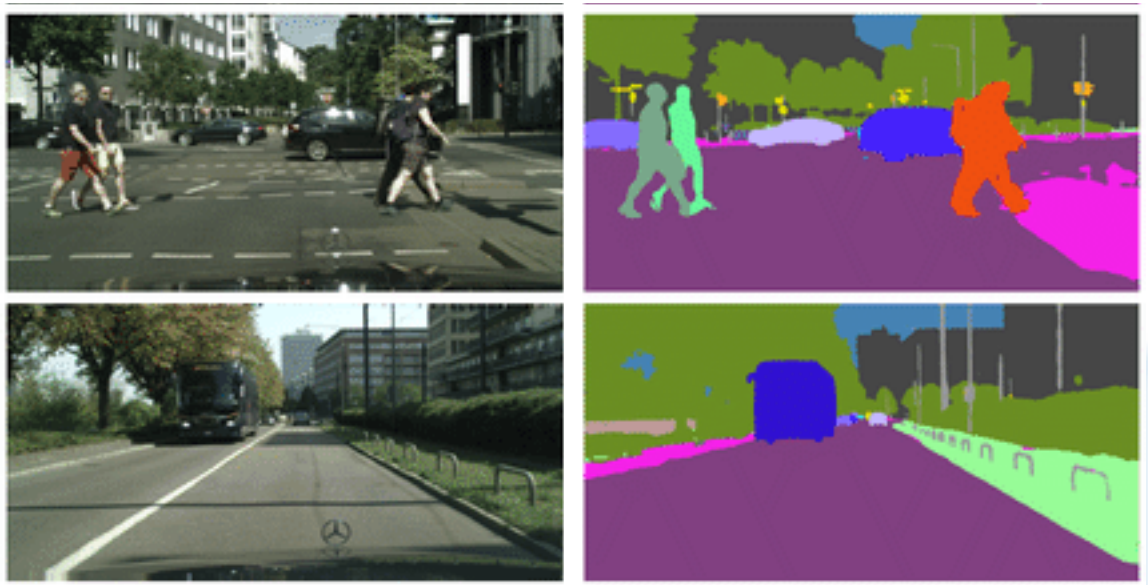


SOTA model zoo

- Инстанс сегментация:
  - Mask R-CNN
  - Detectron (see model zoo)
  - Detectron2 (see model zoo)

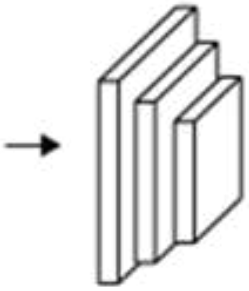
Panoptic segmentation



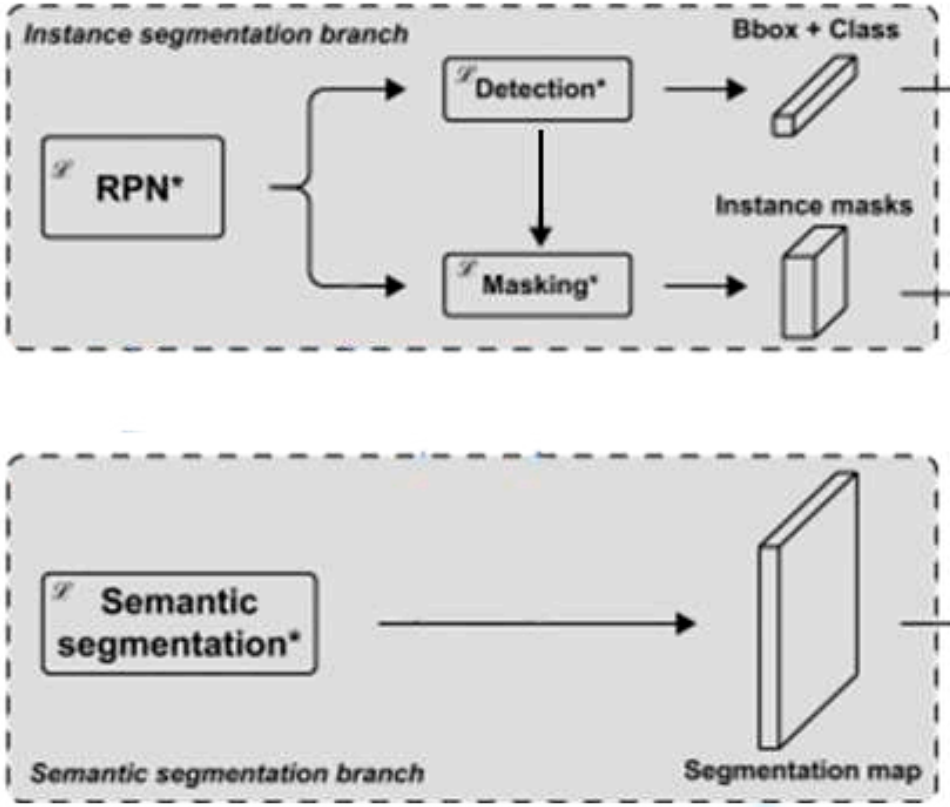


Basic panoptic segmentation model

Input image



Feature extractor



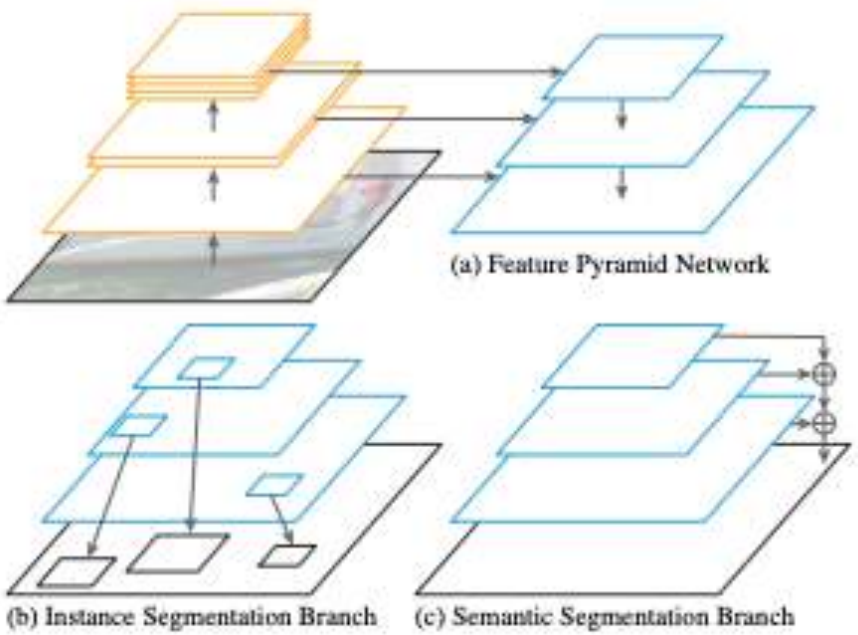
Instance output

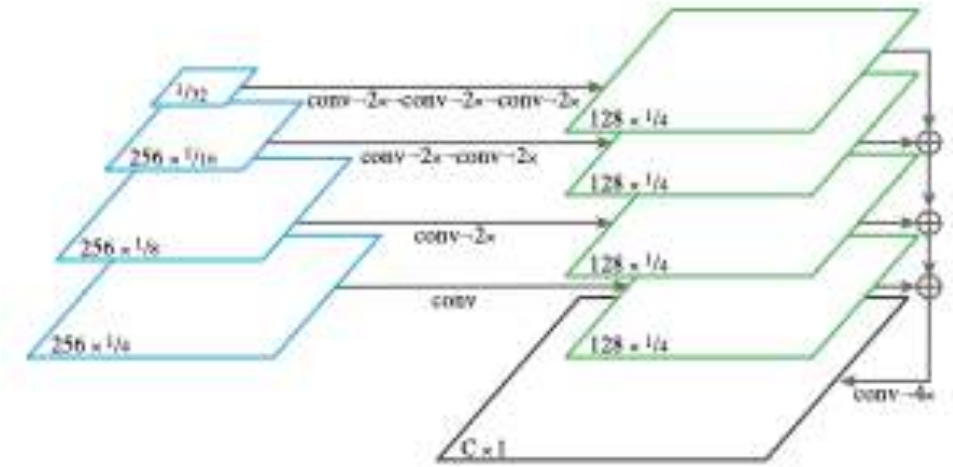
Model zoo

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

Panoptic Feature Pyramid Networks (2019)

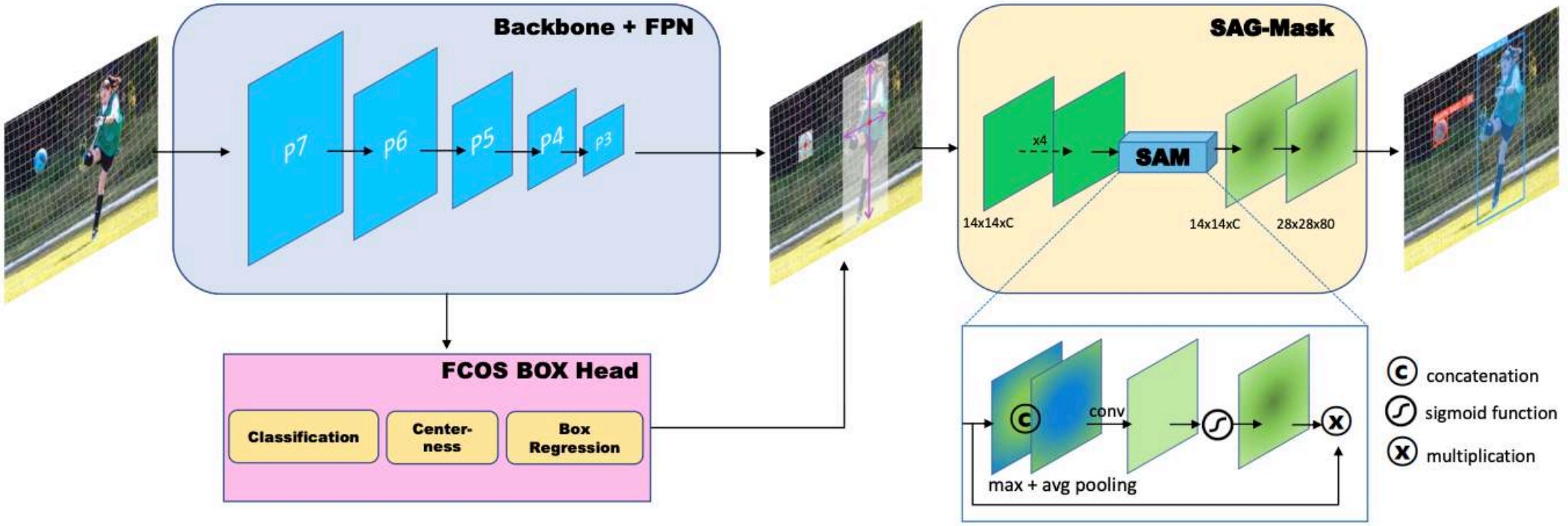
Mask R-CNN, дополненная веткой семантической сегментации





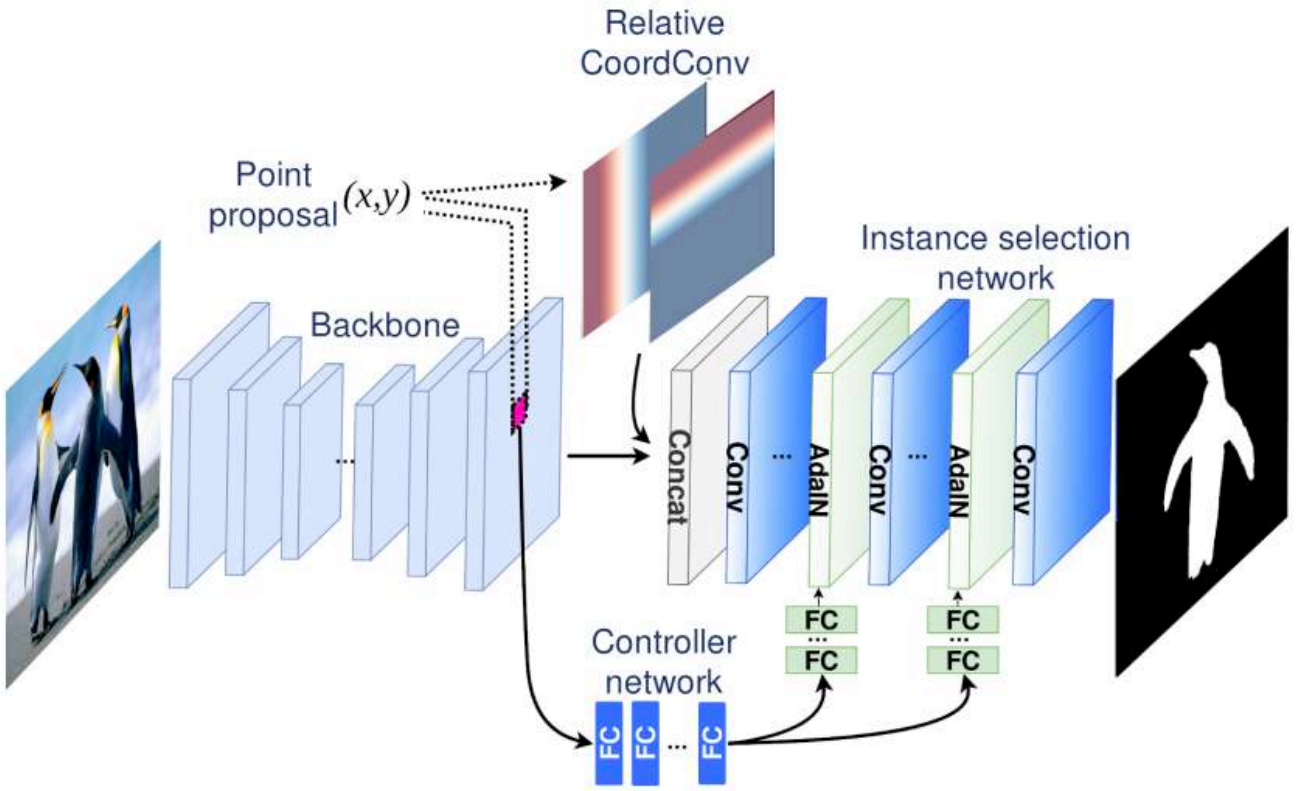
CenterMask (2020)

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



AdaptIS (2019)

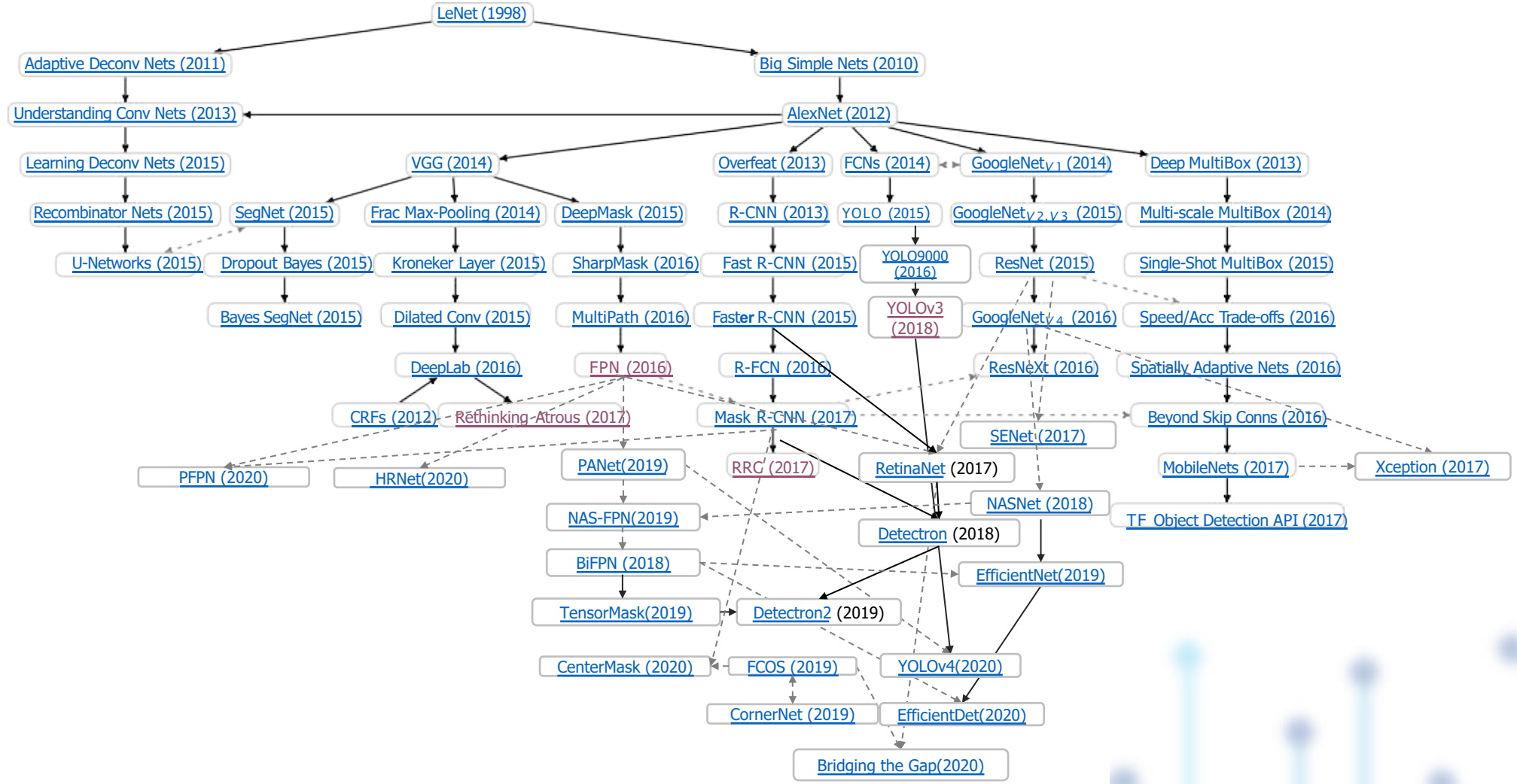
- Использует Adaptive Instance Normalization (AdaIN)
- AdaIN – управляющая сеть



Public Datasets for segmentation

BRaTS	2012-2020	MRI/brain	paper	data
Inria Aerial Image Labeling	2017	UAV	paper	data
DAVIS Challenge	2017	general	paper	data
Mapillary Vistas	2017	street	paper	data
ADE20K	2016	indoor	paper	data
SYNTHIA	2016	street	paper	data
SpaceNet	2016	UAV	N/A	data
Playing for Data	2016	street	paper	data
SUN RGB-D	2015	indoor	paper	data
Cityscapes	2015	street	paper	data
Common Objects in Context (COCO)	2014	general	paper	data
Oxford RoboCar Dataset	2014	street	paper	data
KITTI Vision Benchmark Suite	2012	street	paper	data
Visual Object Classes 2012 (VOC12)	2012	street	[1][2]	data
NYU Depth Dataset V2	2012	indoor	paper	data
CamVid: Motion-based Segmentation	2008	street	paper	data

Detection and Segmentation Atlas



Nodes are hyperlinks to the associated scholar 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

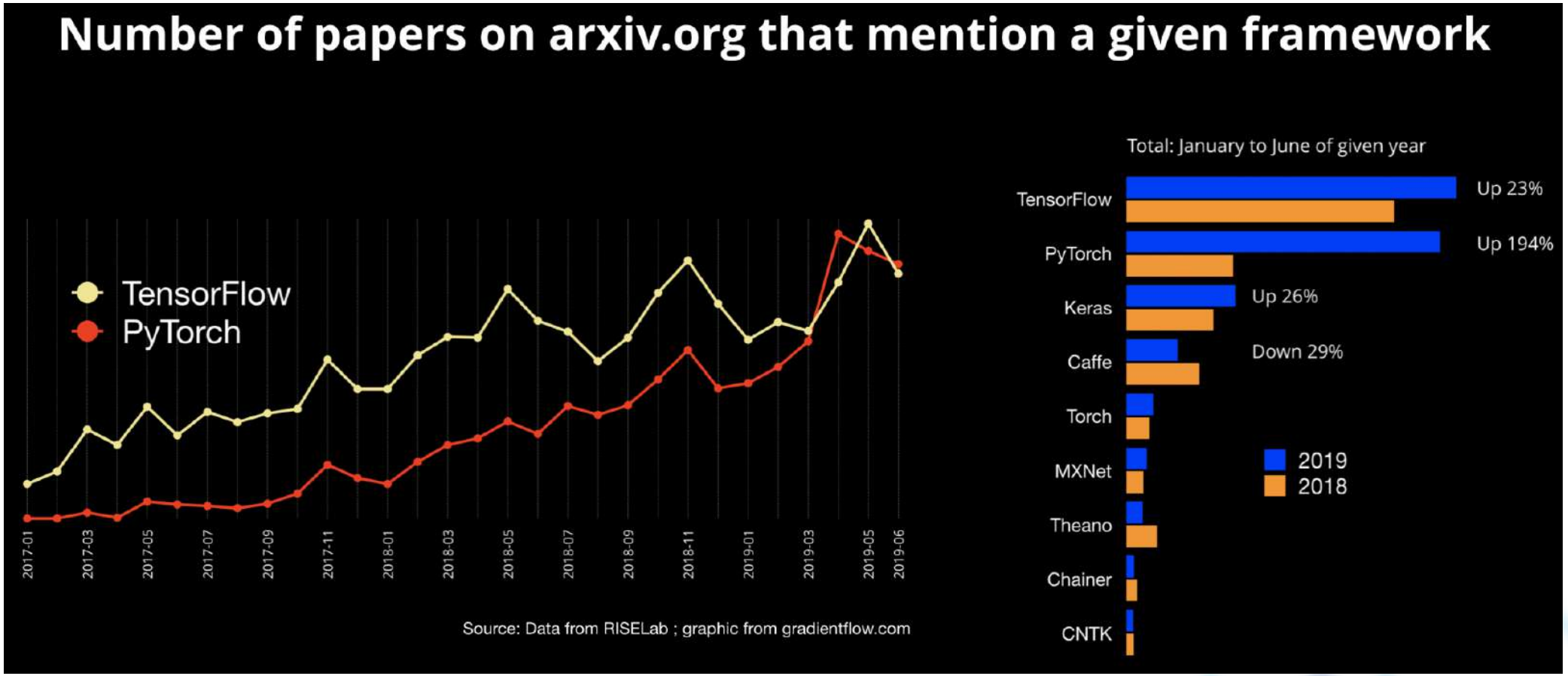
Practical tips: Bag of specials

- 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

TensorFlow

- 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

Keras inside

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

PyTorch

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

Catalyst & Pytorch Lightning

```python
# model runner
runner = SupervisedRunner()

# model training
runner.train(
    model=model,
    criterion=criterion,
    optimizer=optimizer,
    scheduler=scheduler,
    loaders=loaders,
    callbacks=[
        PrecisionCallback(),
        EarlyStoppingCallback()
    ],
    logdir=logdir,
    n_epochs=n_epochs,
    verbose=True
)
```

VS

Catalyst

```python
import argparse

import torch
import torch.nn as nn
import torch.nn.parallel
import torch.optim
import torch.utils.data
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models


# experiment setup
logdir = "./logdir"
n_epochs = 42
args = ...

# data
loaders = {"train": train_loader, "valid": test_loader}

use_cuda = not args.no_cuda and torch.cuda.is_available()
device = torch.device("cuda" if use_cuda else "cpu")


# model, criterion, optimizer
model = Net()

if args.gpu is not None:
    torch.cuda.set_device(args.gpu)
    model = model.cuda(args.gpu)
else:
    # DataParallel will divide and allocate batch_size to all available GPUs
    model = torch.nn.DataParallel(model).cuda()

# define loss function (criterion) and optimizer
criterion = nn.CrossEntropyLoss().cuda(args.gpu)

optimizer = torch.optim.SGD(
    model.parameters(), args.lr,
    momentum=args.momentum,
    weight_decay=args.weight_decay)

for epoch in range(args.start_epoch, args.epochs):
    adjust_learning_rate(optimizer, epoch, args)

    # train for one epoch
    train(train_loader, model, criterion, optimizer, epoch, args)

    # evaluate on validation set
    acc1 = validate(val_loader, model, criterion, args)

    # remember best acc@1 and save checkpoint
    is_best = acc1 > best_acc1
    best_acc1 = max(acc1, best_acc1)

    save_checkpoint({
        'epoch': epoch + 1,
        'arch': args.arch,
        'state_dict': model.state_dict(),
        'best_acc1': best_acc1,
        'optimizer' : optimizer.state_dict(),
    }, is_best)
```

```python
def train(train_loader, model, criterion, optimizer, epoch, args):
    batch_time = AverageMeter()
    data_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()
    top5 = AverageMeter()

    # switch to train mode
    model.train()

    end = time.time()
    for i, (input, target) in enumerate(train_loader):
        # measure data loading time
        data_time.update(time.time() - end)

        if args.gpu is not None:
            input = input.cuda(args.gpu, non_blocking=True)
        target = target.cuda(args.gpu, non_blocking=True)

        # compute output
        output = model(input)
        loss = criterion(output, target)

        # measure accuracy and record loss
        acc1, acc5 = accuracy(output, target, topk=(1, 5))
        losses.update(loss.item(), input.size(0))
        top1.update(acc1[0], input.size(0))
        top5.update(acc5[0], input.size(0))

        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        if i % args.print_freq == 0:
            print('Epoch: [{0}][{1}/{2}]\t'
                  'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
                  'Data {data_time.val:.3f} ({data_time.avg:.3f})\t'
                  'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
                  'Acc@1 {top1.val:.3f} ({top1.avg:.3f})\t'
                  'Acc@5 {top5.val:.3f} ({top5.avg:.3f})'.format(
                   epoch, i, len(train_loader), batch_time=batch_time,
                   data_time=data_time, loss=losses, top1=top1, top5=top5))
```

```python
def validate(val_loader, model, criterion, args):
    batch_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()
    top5 = AverageMeter()

    # switch to evaluate mode
    model.eval()

    with torch.no_grad():
        end = time.time()
        for i, (input, target) in enumerate(val_loader):
            if args.gpu is not None:
                input = input.cuda(args.gpu, non_blocking=True)
            target = target.cuda(args.gpu, non_blocking=True)

            # compute output
            output = model(input)
            loss = criterion(output, target)

            # measure accuracy and record loss
            acc1, acc5 = accuracy(output, target, topk=(1, 5))
            losses.update(loss.item(), input.size(0))
            top1.update(acc1[0], input.size(0))
            top5.update(acc5[0], input.size(0))

            # measure elapsed time
            batch_time.update(time.time() - end)
            end = time.time()

            if i % args.print_freq == 0:
                print('Test: [{0}/{1}]\t'
                      'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
                      'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
                      'Acc@1 {top1.val:.3f} ({top1.avg:.3f})\t'
                      'Acc@5 {top5.val:.3f} ({top5.avg:.3f})'.format(
                       i, len(val_loader), batch_time=batch_time, loss=losses,
                       top1=top1, top5=top5))

        print(' * Acc@1 {top1.avg:.3f} Acc@5 {top5.avg:.3f}'
              .format(top1=top1, top5=top5))

    return top1.avg
```

```python
def save_checkpoint(state, is_best, filename='checkpoint.pth.tar'):
    torch.save(state, filename)
    if is_best:
        shutil.copyfile(filename, 'model_best.pth.tar')


class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()

    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count


def adjust_learning_rate(optimizer, epoch, args):
    """Sets the learning rate to the initial LR decayed by 10 every 30 epochs"""
    lr = args.lr * (0.1 ** (epoch // 30))
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr


def accuracy(output, target, topk=(1,)):
    """Computes the accuracy over the k top predictions for the specified values of k"""
    with torch.no_grad():
        maxk = max(topk)
        batch_size = target.size(0)

        _, pred = output.topk(maxk, 1, True, True)
        pred = pred.t()
        correct = pred.eq(target.view(1, -1).expand_as(pred))

        res = []
        for k in topk:
            correct_k = correct[:k].view(-1).float().sum(0, keepdim=True)
            res.append(correct_k.mul_(100.0 / batch_size))
        return res
```

200+ lines!

Detection frameworks

MMDetection

Detectron

Segmentation frameworks

MMSegmentation

MMSegmentation

pypi v0.7.0 | docs latest | build passing | codecov 84% | license Apache-2.0 | issue resolution 2 d | open issues 13%

Documentation: https://mmsegmentation.readthedocs.io/

Introduction

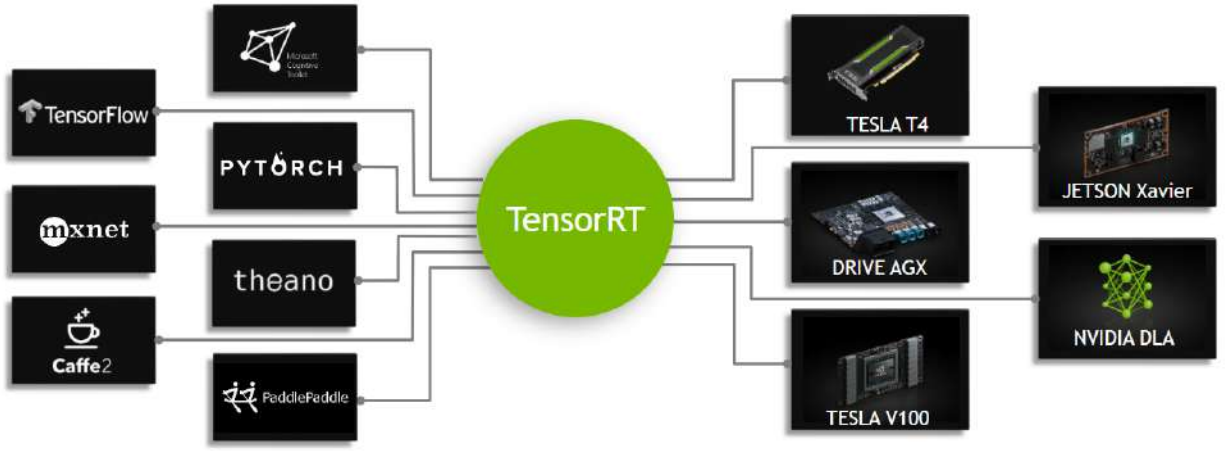
MMSegmentation is an open source semantic segmentation toolbox based on PyTorch. It is a part of the OpenMMLab project.

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

Detectron

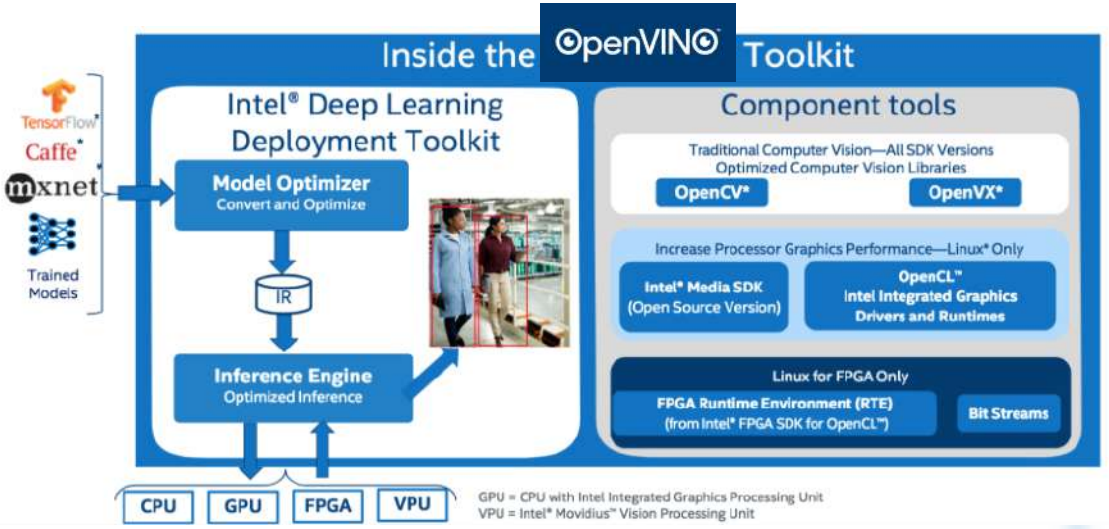
Inference acceleration

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

SOTA ML School

State-Of-The-Art Machine Learning School

ВОПРОСЫ?

ПИШИТЕ

Наталья Ханжина
@natkaha

Роман Лебедев
@lebedev_rv