Сверточные нейронные сети для задачи сегментации

Кафедра технологий проектирования сложных технических систем

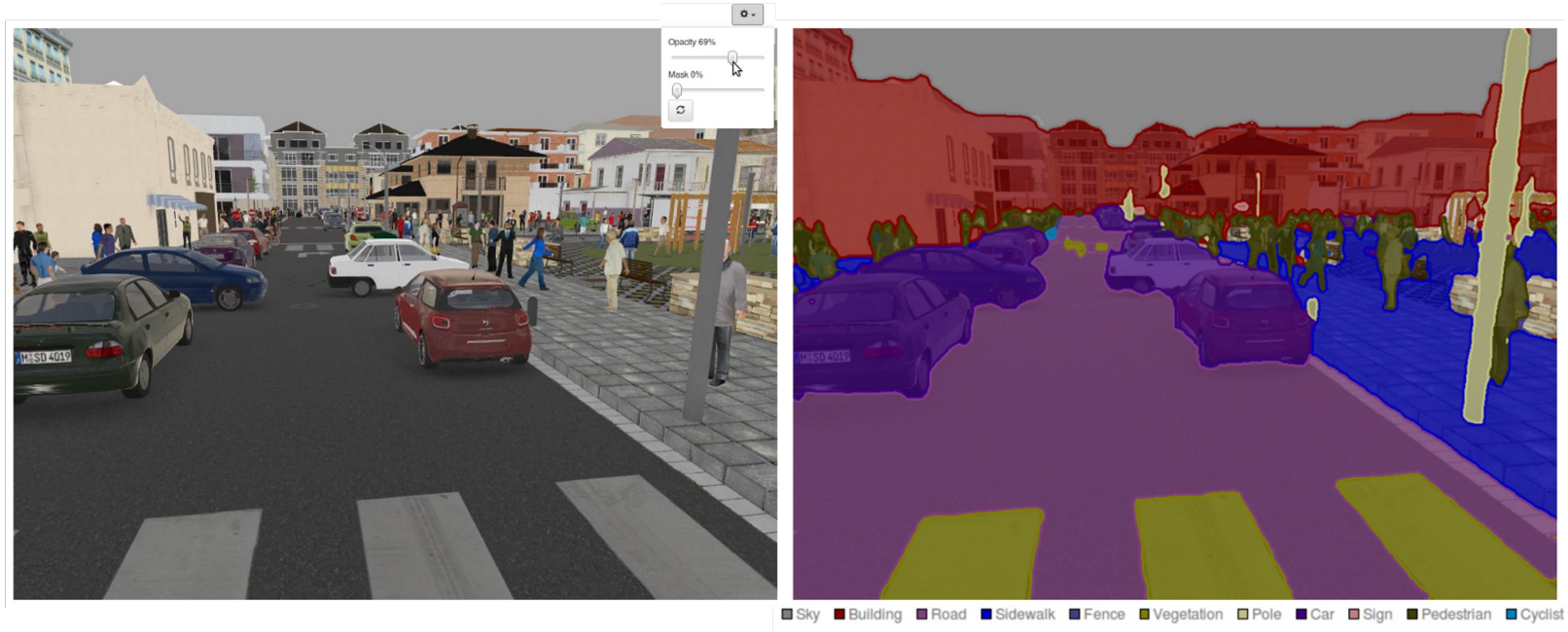
План лекции

- Постановка задачи сегментации изображений
- Базовые подходы к построению архитектур encoder-decoder
- Некоторые техники обучения моделей

Основные направления в CV

Сегментация

Решается задача классификации, но попиксельной

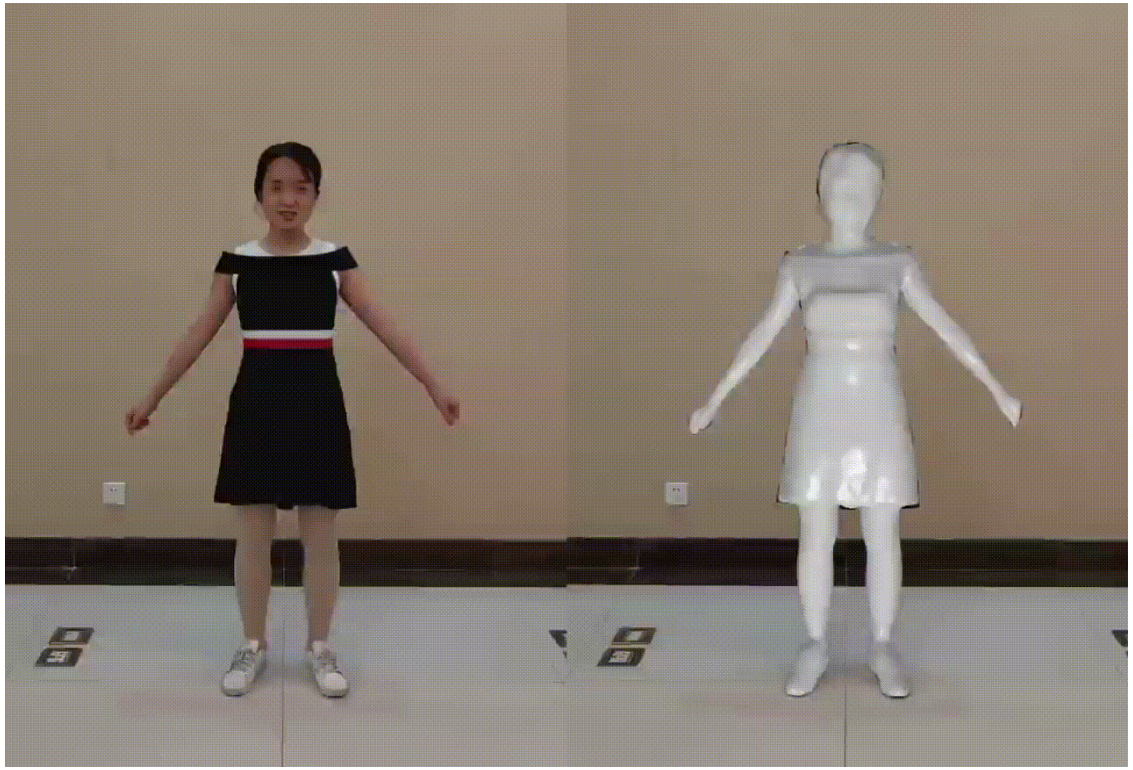


Изображение: ссылка на источник

Основные направления в CV

Построение 3D Объектов

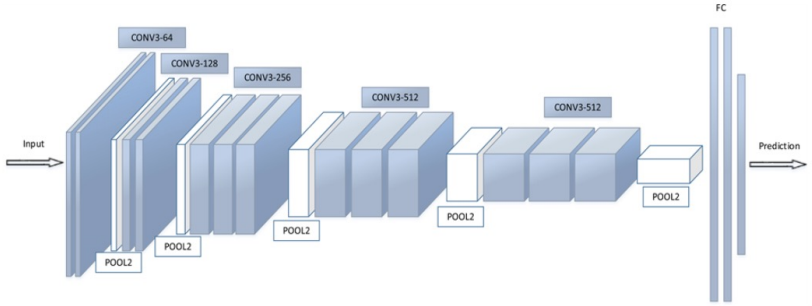
Решается задача построения объемного объекта (obj-файл) по двумерному изображению



Гифка: ссылка на источник, [Jiang et al. 2022]

Классификация

Классификация изображения



Cat (0.99)
Dog(0.01)

Классификация
пикселей изображения

- Сегментация — попиксельная классификация
- Не требует большого количества данных
- Все сегментационные модели — это архитектуры вида FCN

Типы сегментации

Semantic segmentation

segmented

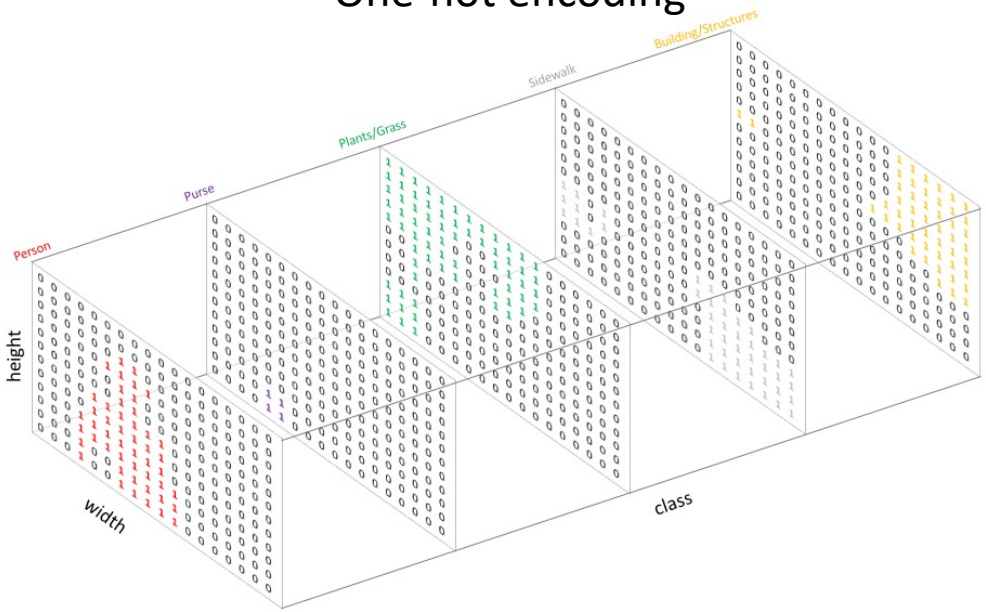
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	3	5	5	5	5	5	5
3	3	3	3	3	1	1	1	1	3	3	3	5	5	5	5	5	5
3	3	3	3	3	3	1	1	3	3	3	5	5	5	5	5	5	5
5	5	3	3	3	3	1	1	3	3	5	5	5	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	5	5	5	5	5
4	4	3	4	1	1	1	1	1	1	4	4	4	4	5	5	5	
4	4	4	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4
3	3	3	1	2	2	1	1	1	1	1	4	4	4	4	4	4	4

Input

Semantic Labels

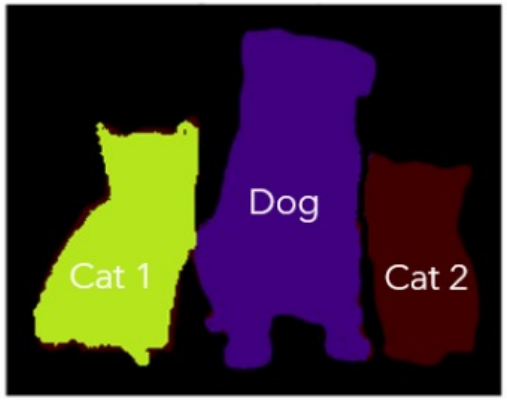
One-hot encoding



Instance segmentation

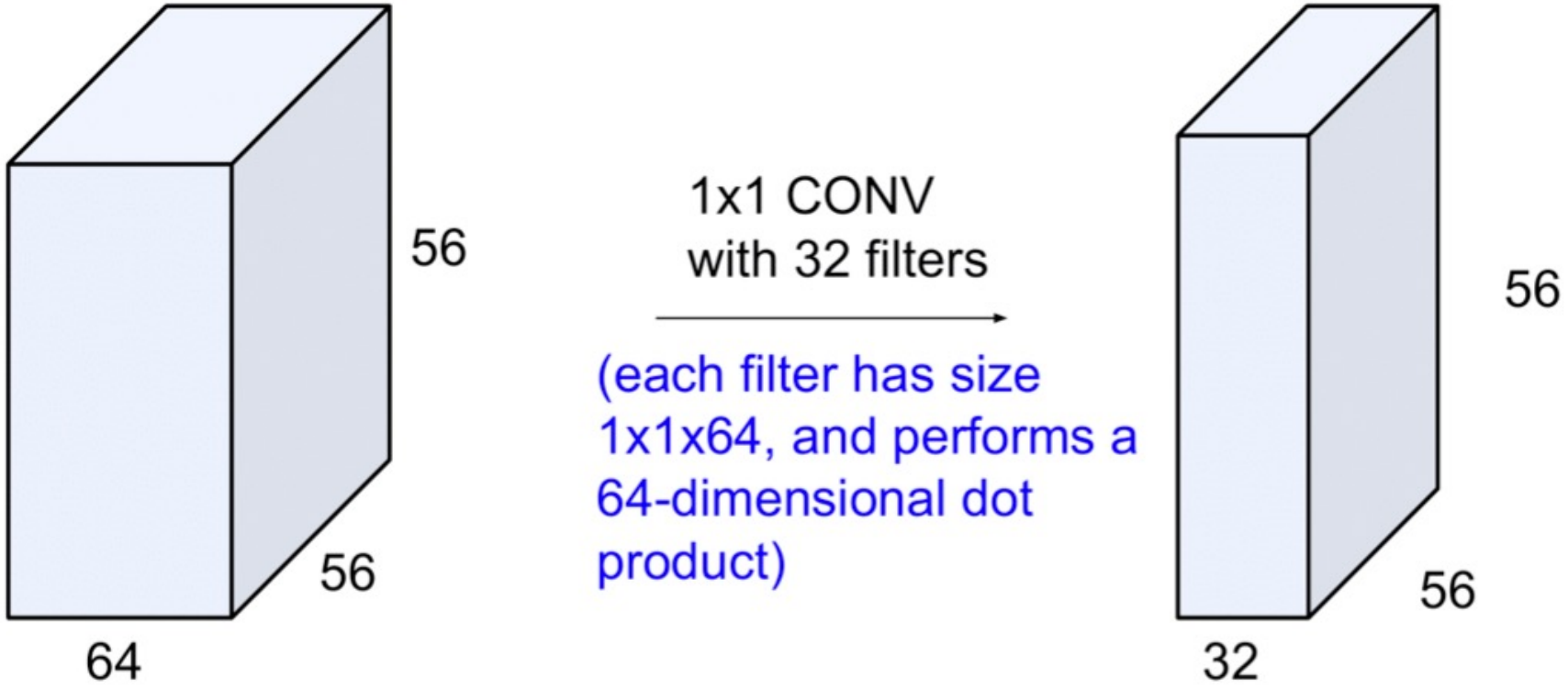
Original Image

Instance Segmentation



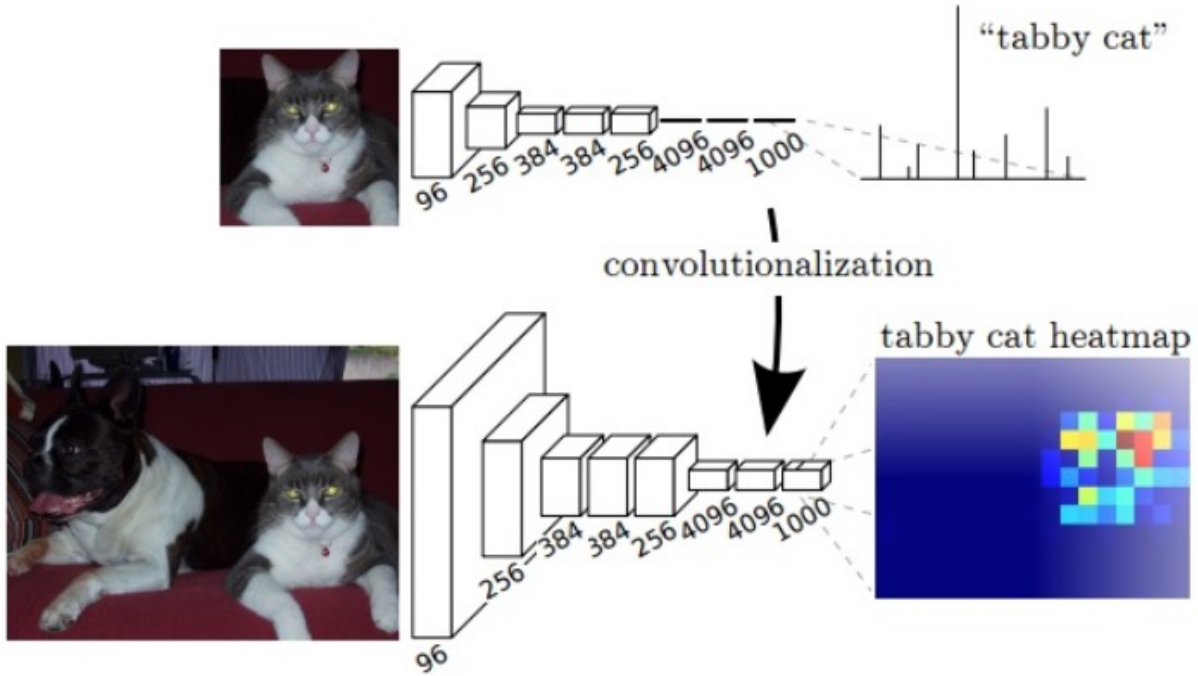
A 2021 guide to Semantic Segmentation

Conv1x1 Recap



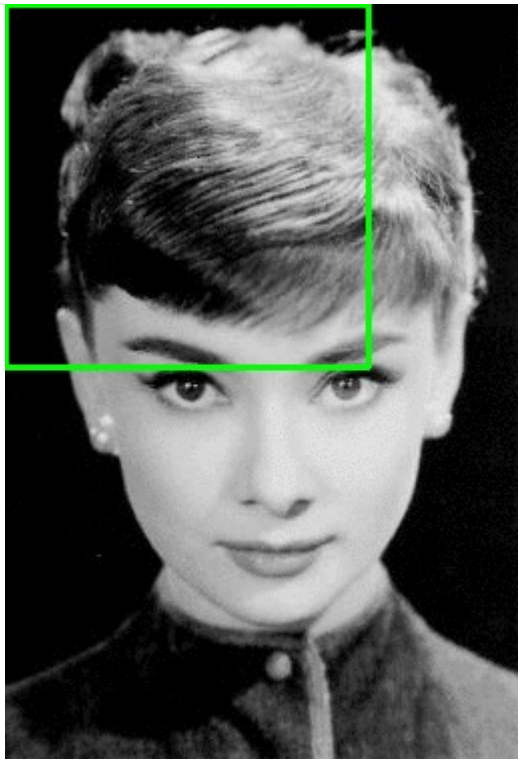
Source: Stanford CS231n Lecture 5 2017 by Fei-Fei Li & Justin Johnson & Serena Yeung

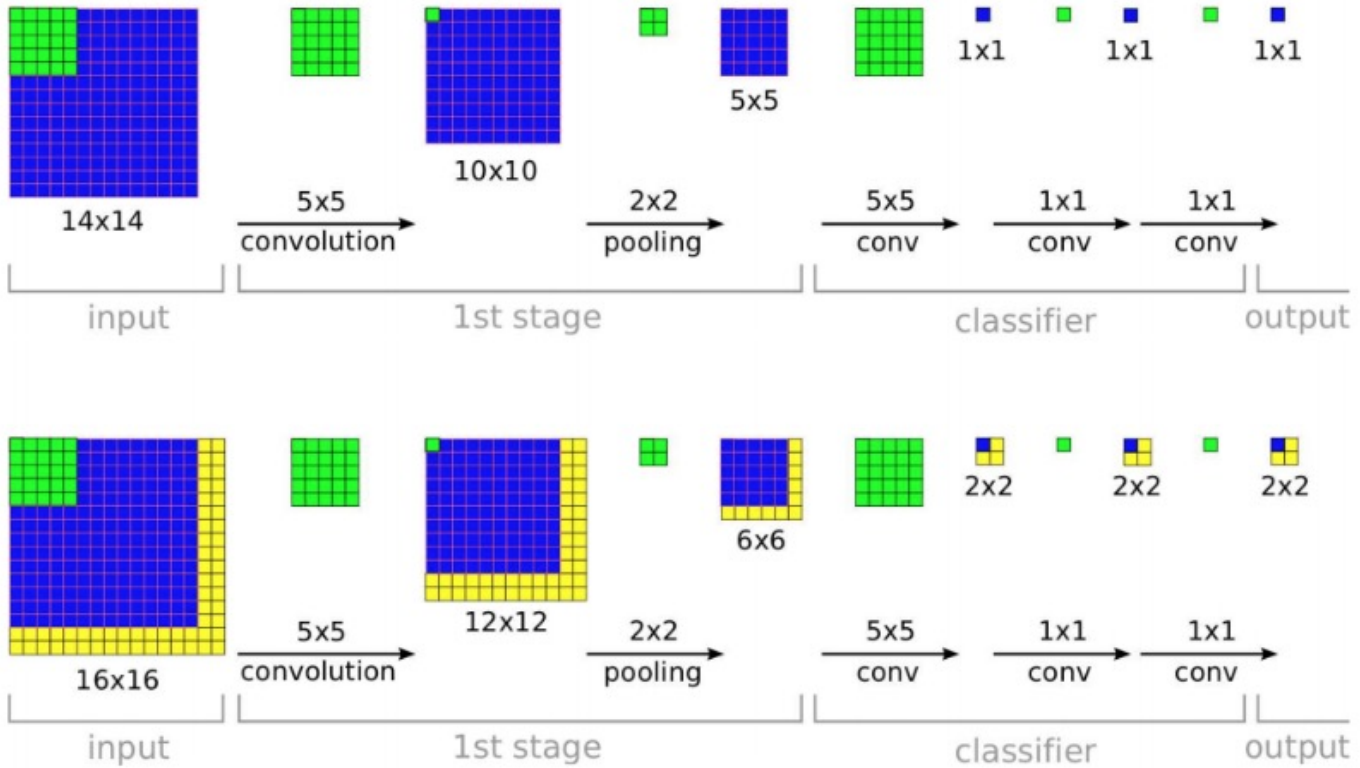
Fully Convolutional Network: FCN



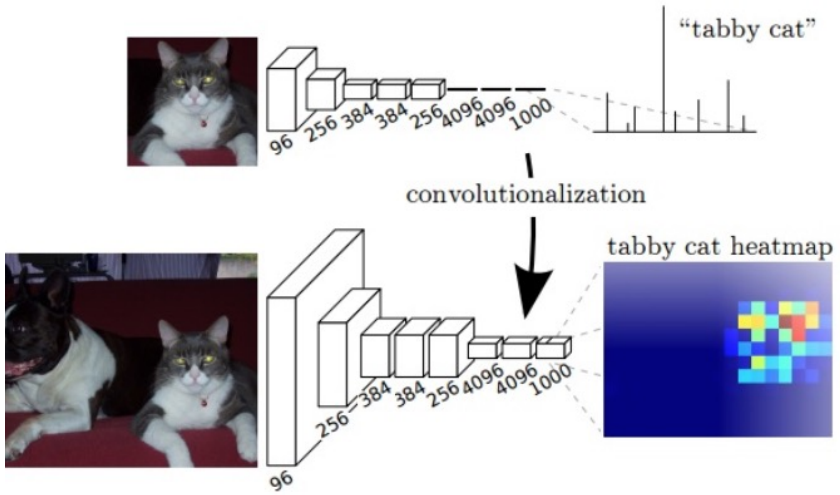
- Убираем FC и заменяем на Conv
- Получаем меньше обучаемых параметров
- На вход может получить изображение любого размера

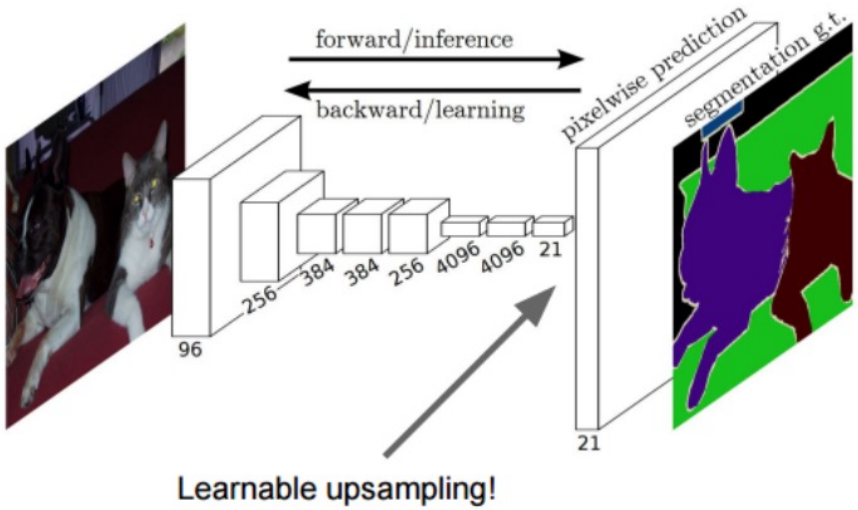
FCN = Efficient Sliding Window





Fully Convolutional Network: FCN

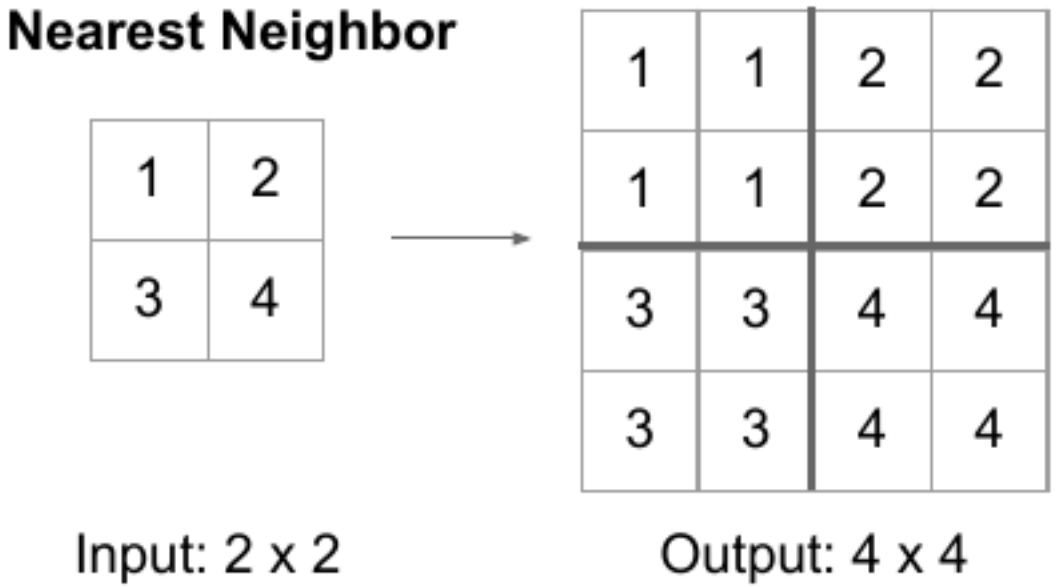




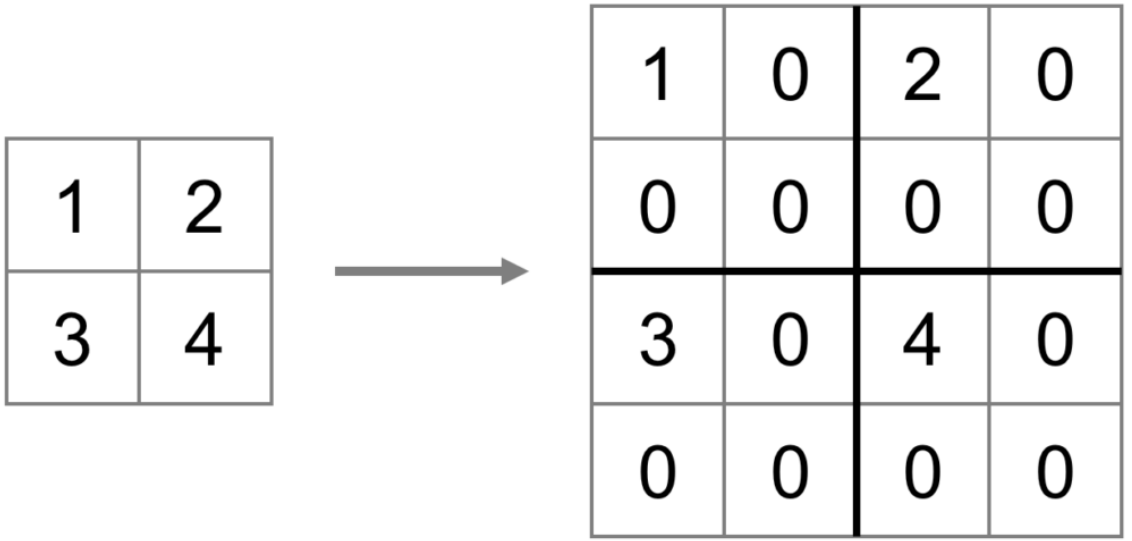
- Убираем FC и заменяем на Conv
- Добавляем декодер

Upsampling

Nearest neighbor

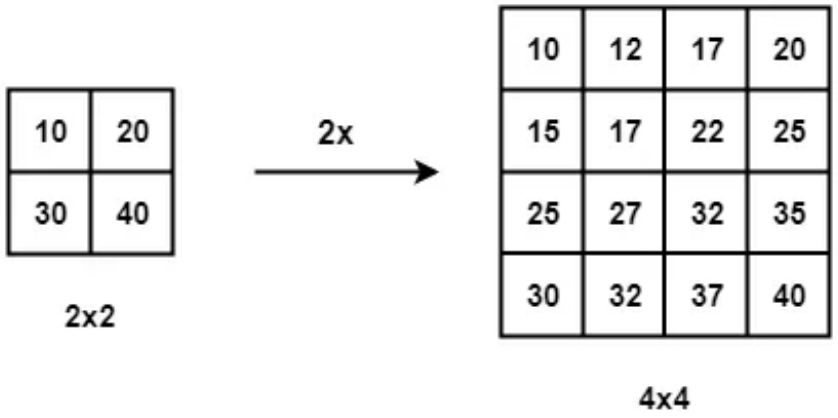


Bed of nails

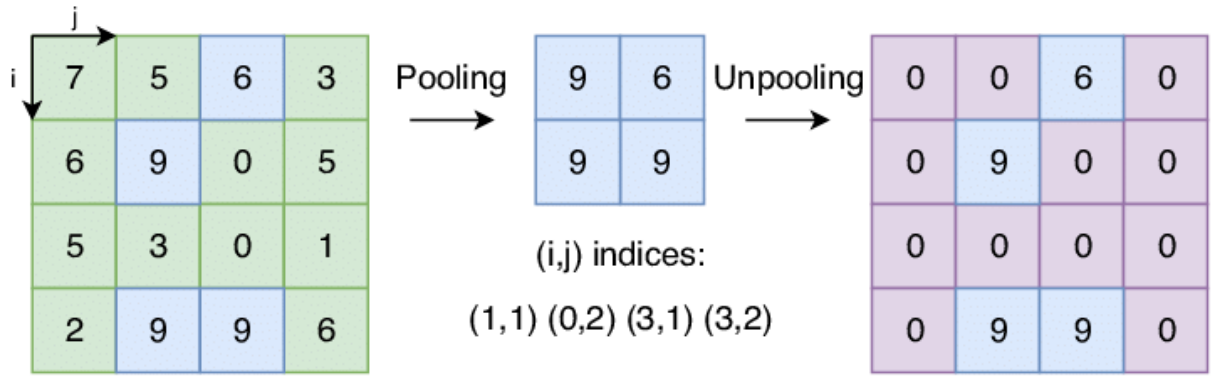


Upsampling

Bilinear interpolation

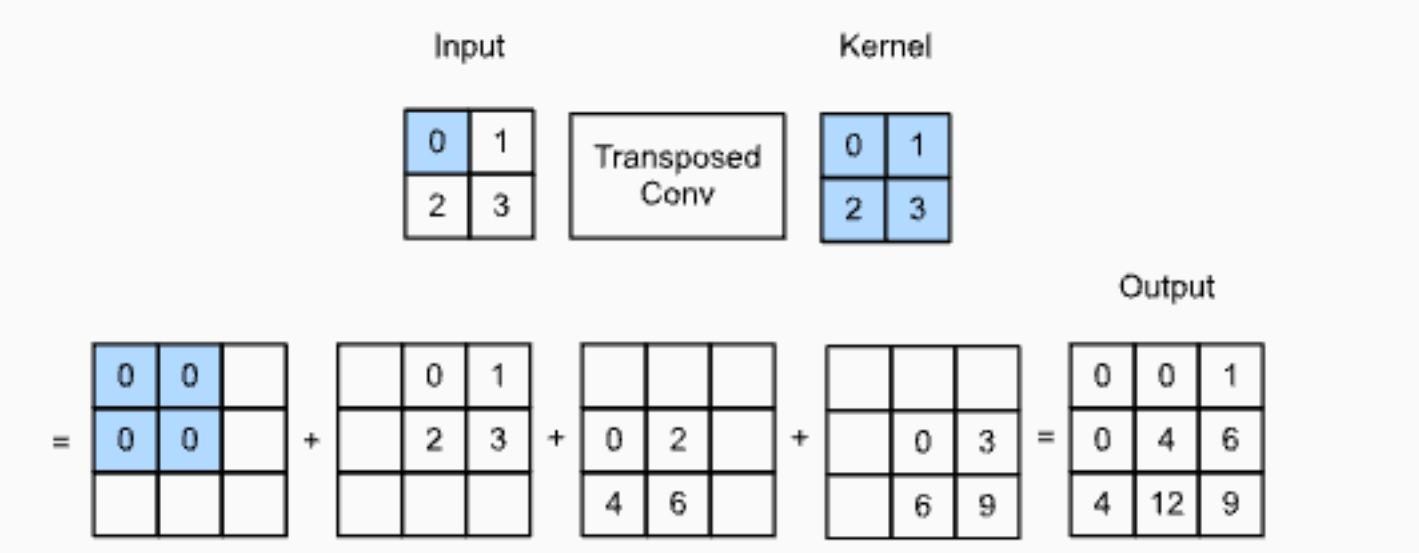


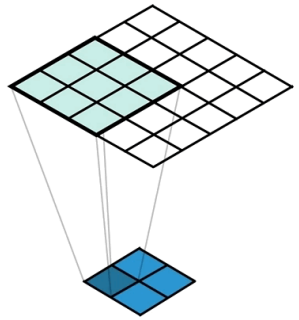
Max-Unpooling



Upsampling

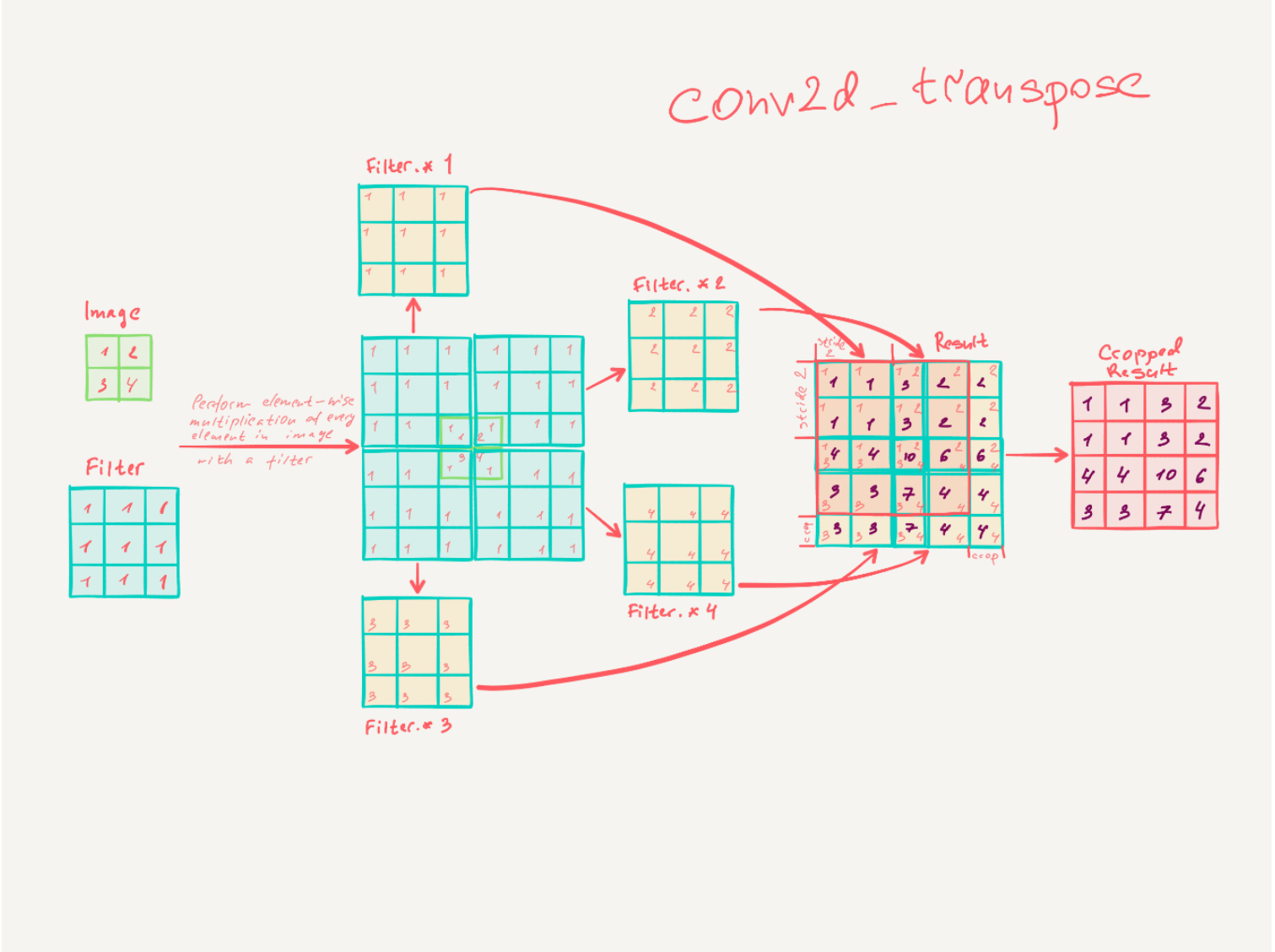
Transposed convolutions – also called fractionally strided convolutions – work by swapping the forward and backward passes of a convolution





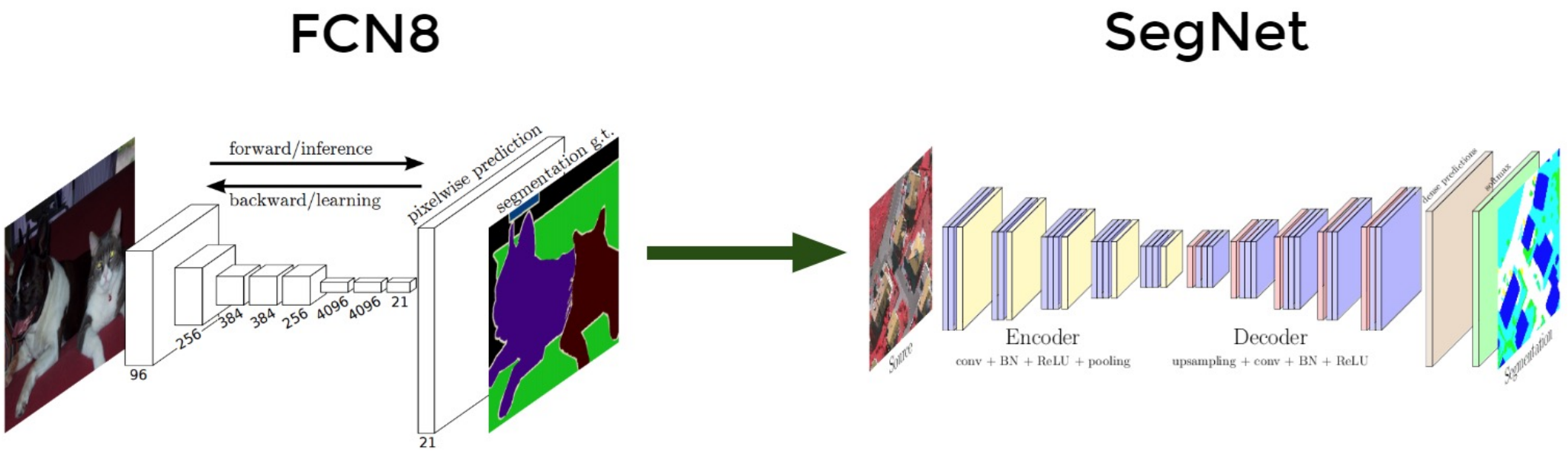
[Matthew D. Zeiler et al. CVPR 2010]

Upsampling



Source

FCN8 to SegNet

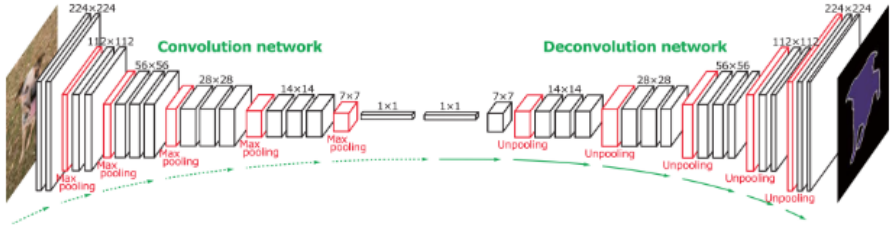


Заменить Upsampling на иерархический Upsampling

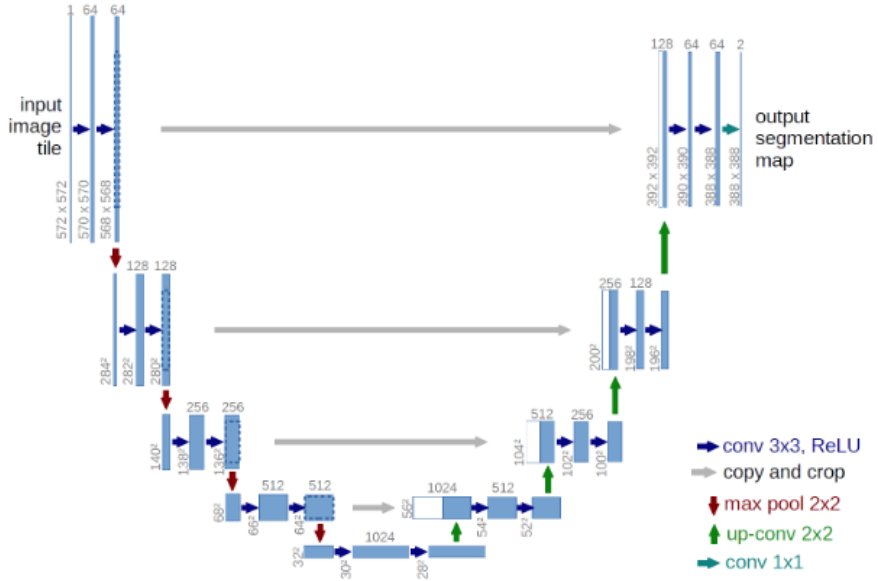
V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” arXiv:1511.00561, 2015

Upsampling

SegNet



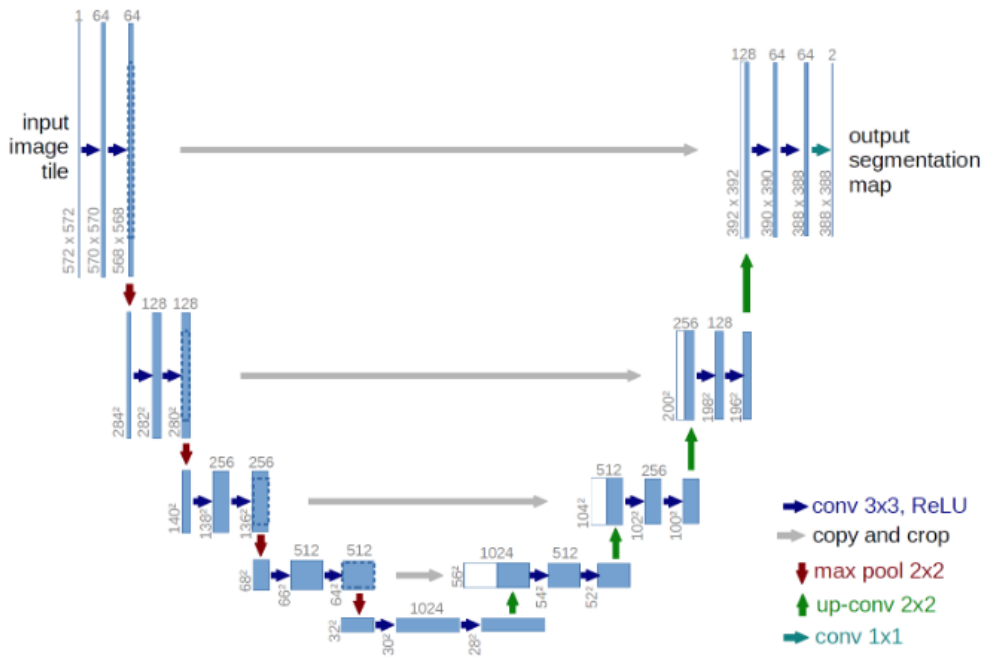
UNet



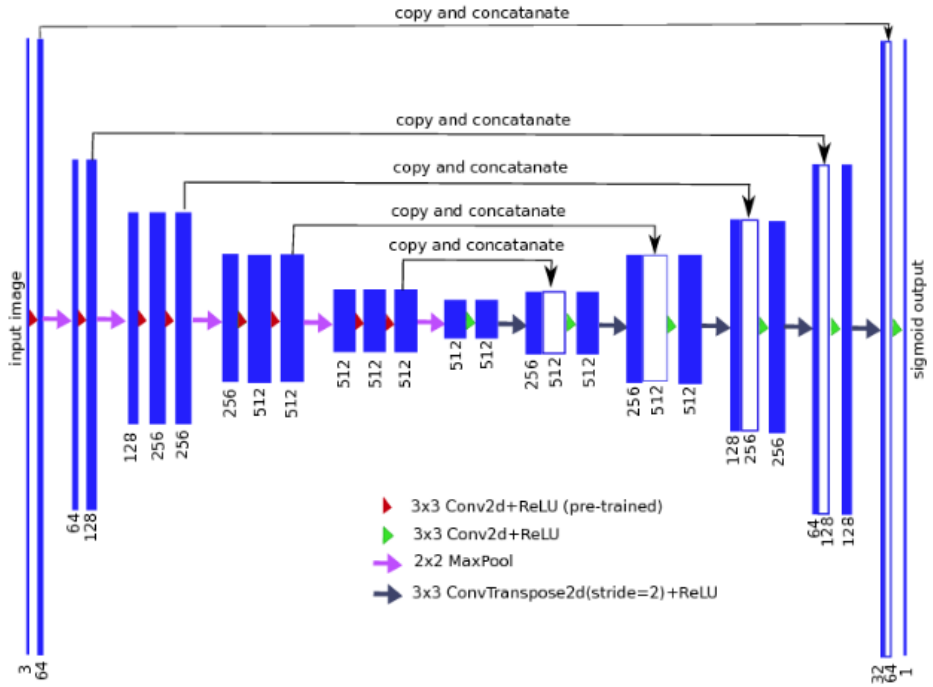
Added skip connections

O. Ronneberger P. Fischer T. Brox "U-net: Convolutional networks for biomedical image segmentation" Proc. Med. Image Comput. Comput.-Assisted Intervention pp. 234-241 2015.

Unet => TernausNet



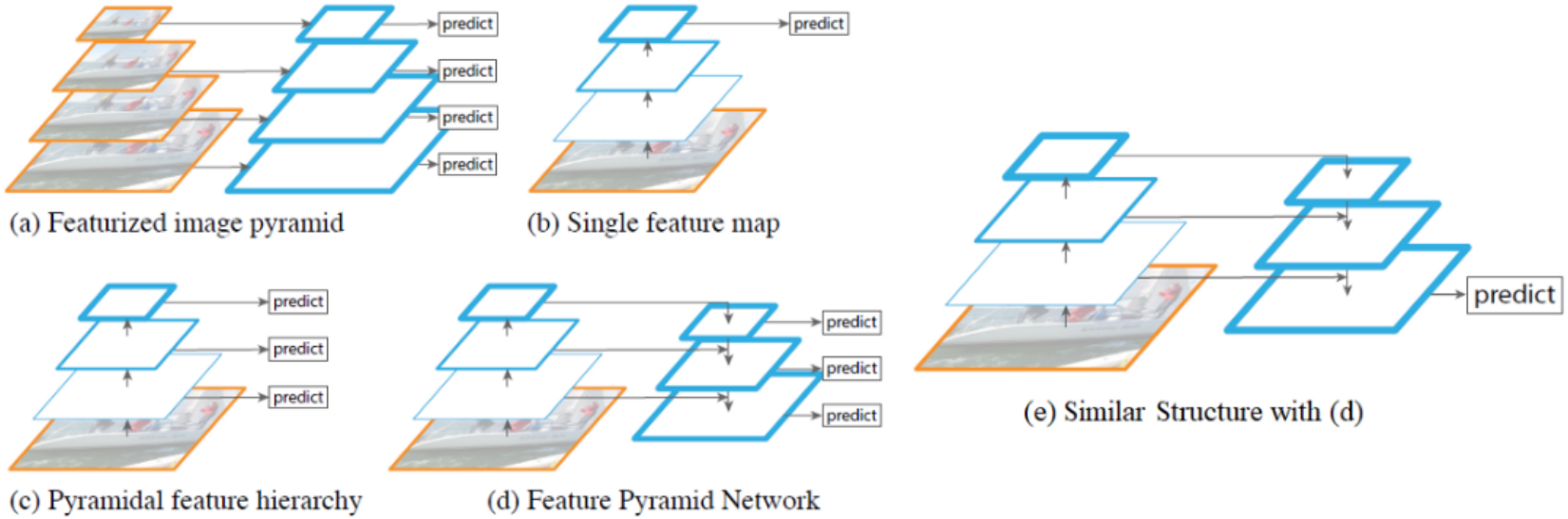
Text



Энкодер инициализируем весами с ImageNet

Iglovikov, V., Shvets, A.: Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. arXiv preprint arXiv:1801.05746 (2018)

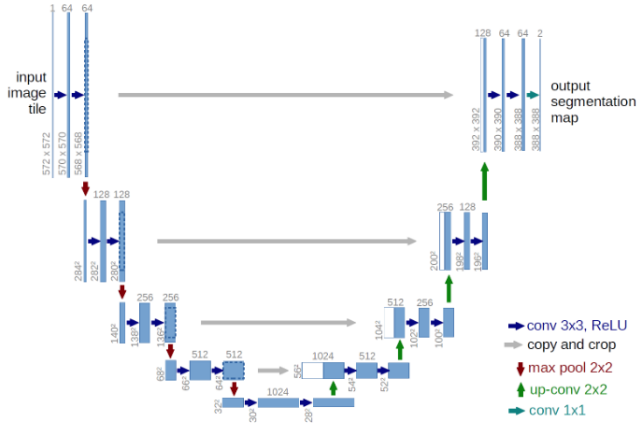
Feature Pyramid Networks (FPN)

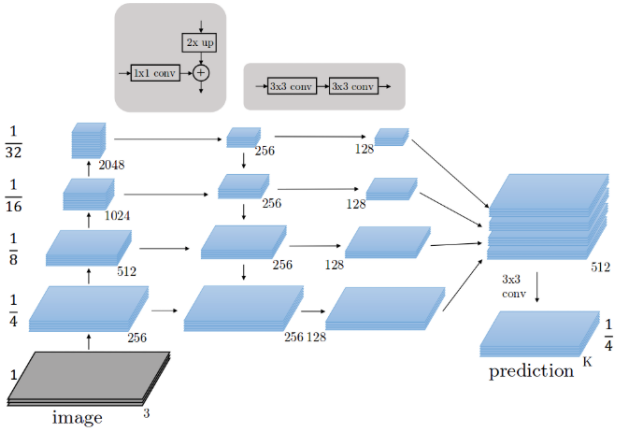


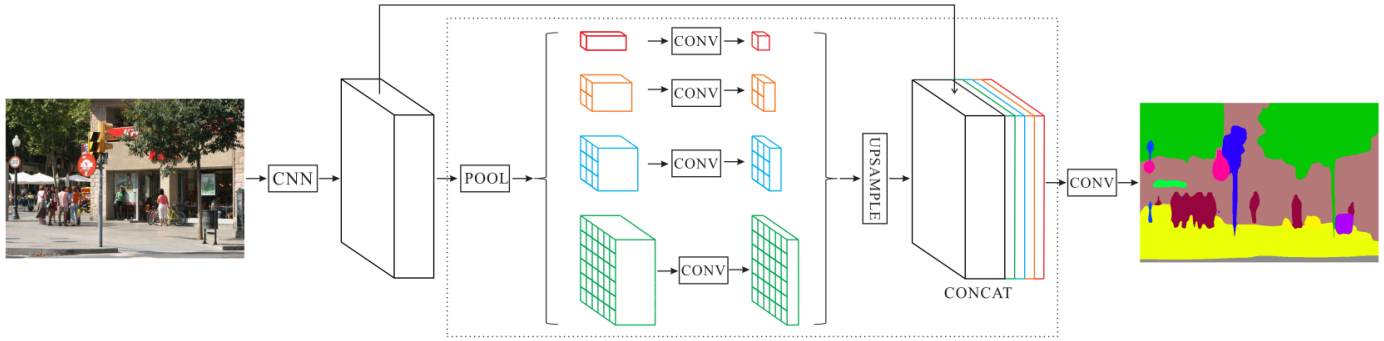
1. Легко добавить во многие архитектуры.
2. Помогает с multiscale

Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117-2125

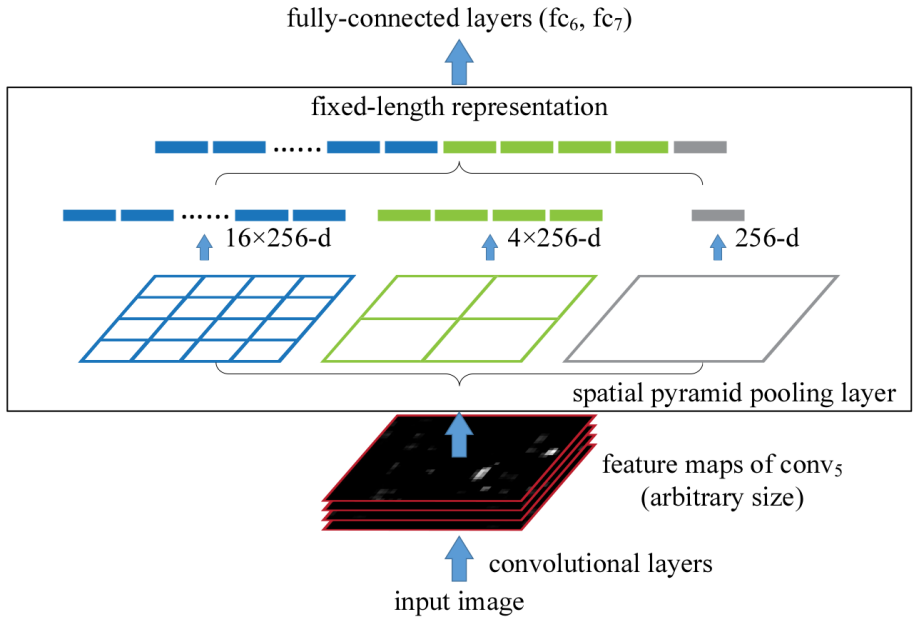
Unet + FPN



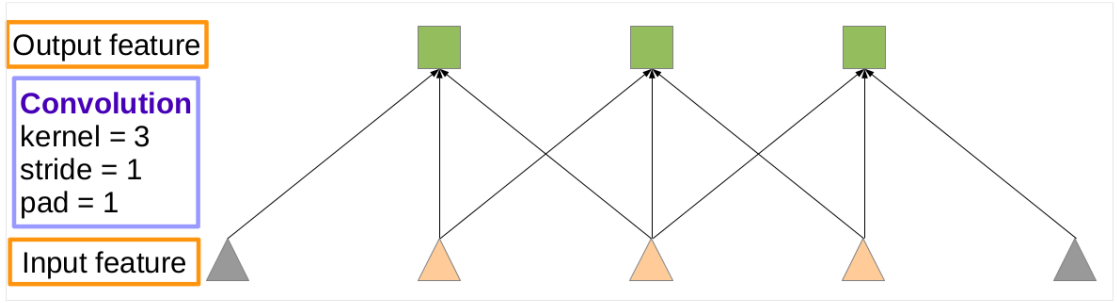




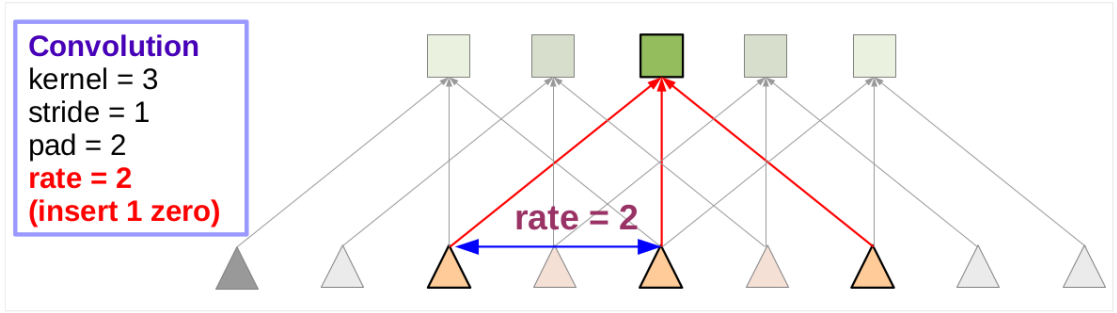
(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction



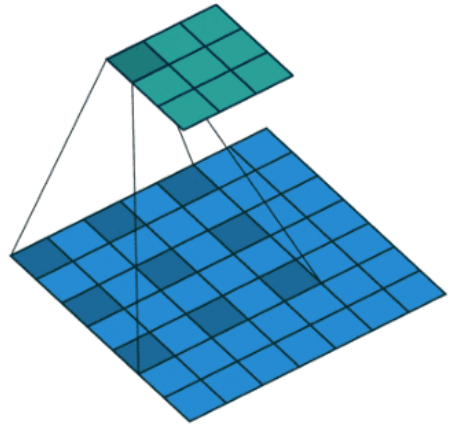
Atrous Spatial Pyramid Pooling

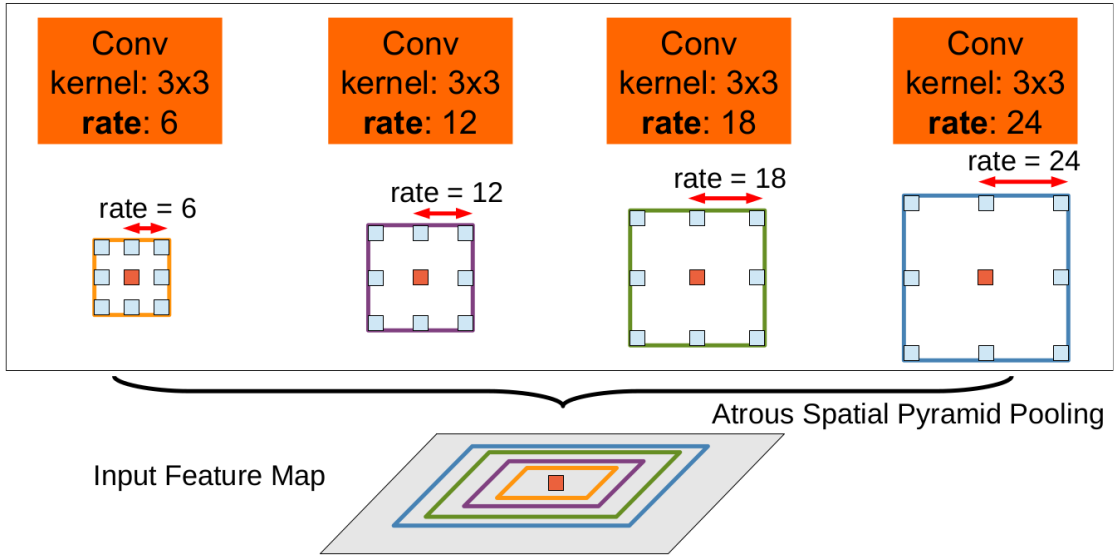


(a) Sparse feature extraction

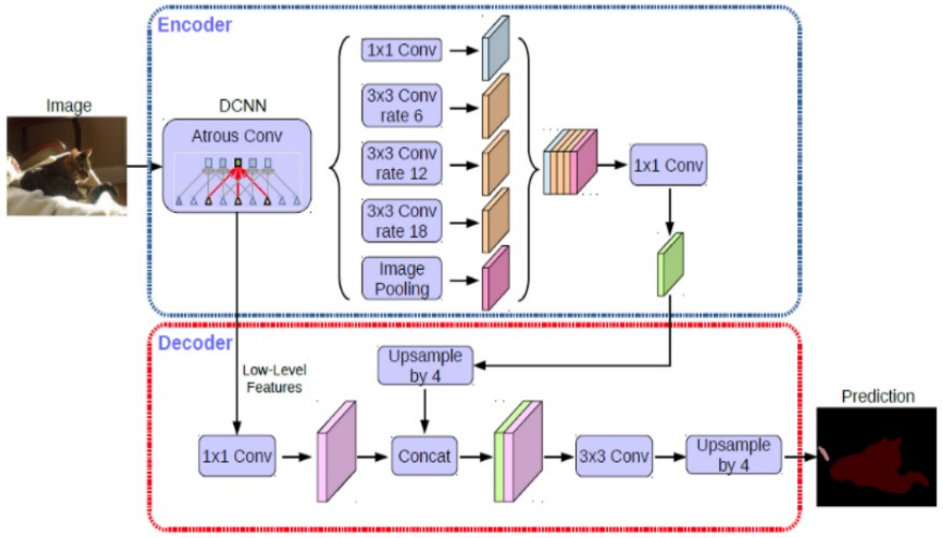


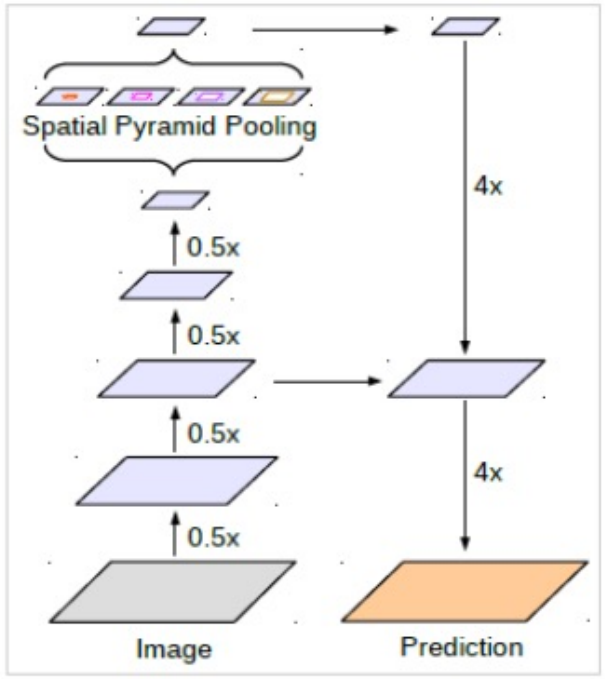
(b) Dense feature extraction

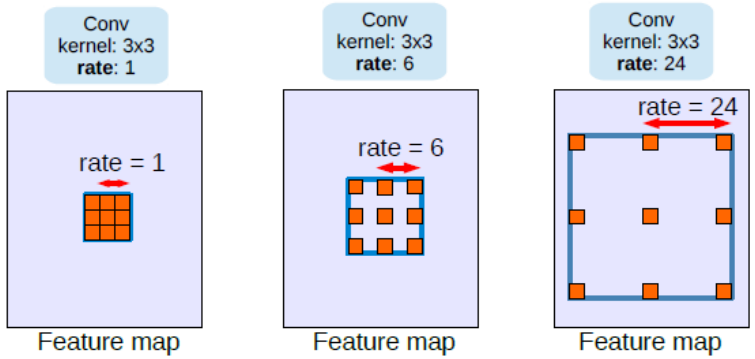




DeepLabV3







Atrous Separable Convolution

Segmentation Loss Function

Каждый пиксель классификатор => Categorical / Binary Cross Entropy(CCE, BCE)

Но! Метрика Dice / Jaccard

Dice / Jaccard недифференцируемы =>

Soft Dice / Soft Jaccard

и добавляем в loss

Lovasz-Softmax loss

Использовать для FineTune

$$CCE = \sum_c p(x) \log q(x)$$

$$LOSS = BCE - \ln(DICE)$$

$$BCE = -\sum_i (y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i))$$

$$DICE = 2\frac{\sum_i y_i p_i}{\sum_u y_i + \sum p_i}$$

Berman, M., Rannen Triki, A., Blaschko, M.B.: The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4413–4421 (2018)

Label smoothing

Было

$$L = \sum_{i=1}^{n} H_i(p, q_\theta)$$
$$= -\sum_{i=1}^{n}\sum_{y=1}^{K} p(y|x_i)\log q_\theta(y|x_i)$$

Вместо использования вектора с одним активным битом мы вводим шумовое распределение u(y|x) вида (y|θ). Наша новая истинная метка для данных (xi,yi) (i=1,...,N) будет

$$p'(y|x_i) = (1-\varepsilon)p(y|x_i) + \varepsilon u(y|x_i)$$
$$= \begin{cases} 1-\varepsilon+\varepsilon u(y|x_i) & \text{if } y = y_i \\ \varepsilon u(y|x_i) & \text{otherwise} \end{cases}$$

$$\varepsilon \in [0,1] \qquad \sum_{y=1}^{K} p'(y|x_i) = 1$$

Мы используем эту новую истинную метку вместо метки с одним активным битом в нашей функции потерь.

$$L' = -\sum_{i=1}^{n}\sum_{y=1}^{K} p'(y|x_i)\log q_\theta(y|x_i)$$
$$= -\sum_{i=1}^{n}\sum_{y=1}^{K} \left[(1-\varepsilon)p(y|x_i) + \varepsilon u(y|x_i)\right]\log q_\theta(y|x_i)$$

$$L' = \sum_{i=1}^{n}\left\{(1-\varepsilon)\Big[-\sum_{y=1}^{K} p(y|x_i)\log q_\theta(y|x_i)\Big] + \varepsilon\Big[-\sum_{y=1}^{K} u(y|x_i)\log q_\theta(y|x_i)\Big]\right\}$$
$$= \sum_{i=1}^{n}\Big[(1-\varepsilon)H_i(p, q_\theta) + \varepsilon H_i(u, q_\theta)\Big]$$

Как закодировать в обычном случае

$$y_k^{LS} = y_k(1-\alpha) + \frac{\alpha}{K}$$

```
import torch
import torch.nn as nn

class LabelSmoothingLoss(nn.Module):
    def __init__(self, smoothing=0.0):
        super(LabelSmoothingLoss, self).__init__()
        self.smoothing = smoothing

    def forward(self, x, target):
        logprobs = torch.nn.functional.log_softmax(x, dim=-1)
        nll_loss = -logprobs.gather(dim=-1, index=target.unsqueeze(1)).squeeze
(1)
        smooth_loss = -logprobs.mean(dim=-1)
        loss = (1 - self.smoothing) * nll_loss + self.smoothing * smooth_loss
        return loss.mean()
```


Rethinking the Inception Architecture for Computer Vision

Exponential Moving Average

Экспоненциальное скользящее среднее (EMA) - это техника, используемая в глубоком обучении, чтобы стабилизировать процесс обучения, сглаживая колебания в обучающих данных.

$$\mathrm{EMA}_t = \begin{cases} p_1 & \text{if } t = 1 \\ \alpha p_t + (1-\alpha)\mathrm{EMA}_{t-1} & \text{else} \end{cases}$$

Псевдокод

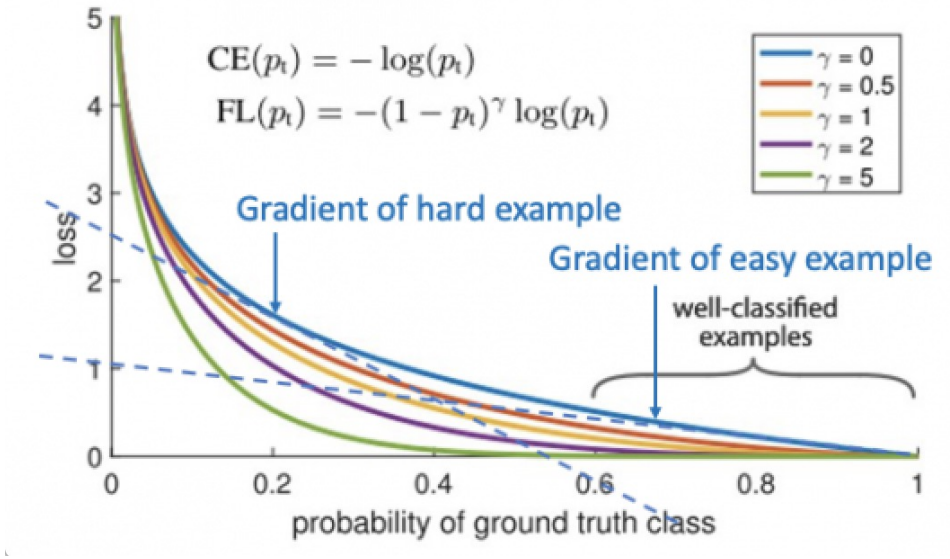
```
# Initialize the EMA parameters
ema_alpha = 0.99
ema_weights = model.get_weights()

# Iterate through the training data
for i in range(num_epochs):
    for batch in training_data:
        # Perform forward and backward pass on the batch
        loss = model.train_on_batch(batch)

        # Update the model weights with stochastic gradient descent
        weights = model.get_weights()
        new_weights = [w - learning_rate * dw for w, dw in zip(weights, gradient
s)]

        # Update the EMA weights
        ema_weights = [ema_alpha * w + (1 - ema_alpha) * new_w for w, new_w in z
ip(ema_weights, new_weights)]

        # Set the model weights to the EMA weights
        model.set_weights(ema_weights)
```

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim.swa_utils import AveragedModel, SWALR

# define your neural network
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(784, 100)
        self.fc2 = nn.Linear(100, 10)

    def forward(self, x):
        x = x.view(-1, 784)
        x = nn.functional.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# define your optimizer
optimizer = optim.SGD(net.parameters(), lr=0.01)

# define the EMA
ema_model = AveragedModel(net)

# define the SWA learning rate scheduler
swa_scheduler = SWALR(optimizer, swa_lr=0.05)

# train your model
for epoch in range(num_epochs):
    for i, (inputs, labels) in enumerate(train_loader):
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = loss_function(outputs, labels)
        loss.backward()
        optimizer.step()
        ema_model.update_parameters(net)
    swa_scheduler.step()

# evaluate your model using the EMA
ema_model.eval()
with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = ema_model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
accuracy = 100 * correct / total
```

Focal Loss

Фокальная функция потерь - это функция потерь, разработанная для задач обнаружения объектов в глубоком обучении.
Она разработана для решения проблемы несбалансированности классов, когда количество примеров в одном классе значительно превышает количество примеров в других классах.

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t)$$




```
def focal_loss(y_true, y_pred, alpha=0.25, gamma=2.0):
    # Convert predictions to probabilities
    y_pred = K.softmax(y_pred, axis=-1)
    # Calculate the cross entropy loss
    ce_loss = -y_true * K.log(y_pred)
    # Calculate the focal loss
    pt = K.sum(y_true * y_pred, axis=-1)
    fl_loss = alpha * K.pow(1 - pt, gamma) * ce_loss
    return K.mean(fl_loss)
```

Метрики

Intersection over Union (IOU) or Jaccard Index = Area of Overlap / Area of Union

Чаще всего используют Dice - особенно в медицинских снимках и Jaccard (IoU)

Table 1. The three similarity coefficients

Similarity Coefficient (X,Y)	Actual Formula
Dice Coefficient	$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$
Cosine Coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} . \lvert Y \rvert^{1/2}}$
Jaccard Coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$

Заключение

Рассмотрена постановка задачи сегментации изображений

Изучены базовые подходы архитектур encoder-decoder
- pooling
- upconv, upsampling
- unet

Рассмотрены техники обучения моделей
- Label smoothing
- EMA
- Focal loss

Ссылки

Kaggle - Airbus Ship Detection Challenge, пример на задачу iinstance segmentation