Введение и обзор задач Computer Vision

Макоян Артем
Романов Игнат
Якшимамедов Владимир

А что было до DL

Методы обработки изображений

- dense
  - обрабатывают каждый пиксель
- sparse
  - работают с особыми точками

Кто такие особые точки?

- *Отличимость (distinctness)* – особая точка должна явно выделяться на фоне и быть отличимой (уникальной) в своей окрестности.
- *Инвариантность (invariance)* – определение особой точки должно быть независимо к аффинным преобразованиям.
- *Стабильность (stability)* – определение особой точки должно быть устойчиво к шумам и ошибкам.
- *Уникальность (uniqueness)* – кроме локальной отличимости, особая точка должна обладать глобальной уникальностью для улучшения различимости повторяющихся паттернов.
- *Интерпретируемость (interpretability)* – особые точки должны определяться так, чтобы их можно было использовать для анализа соответствий и выявления интерпретируемой информации из изображения.

Как искать?

Moravec (1977)

1) Берем область 3x3 вокруг каждого пикселя и считаем сумму квадратов разностей интенсивностей с центральным пикселем
2) Отбрасываем точки с слишком маленьким значением

Chris Harris and Mike Stephens (1988)

1) Смотрим на те же самые квадраты разностей

$$f(\Delta x, \Delta y) = \sum_{(x_k, y_k) \in W} (I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y))^2$$

2) Раскладываем I в ряд тейлора

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y(x, y)\Delta y$$

3) Переписываем

$$f(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix},$$

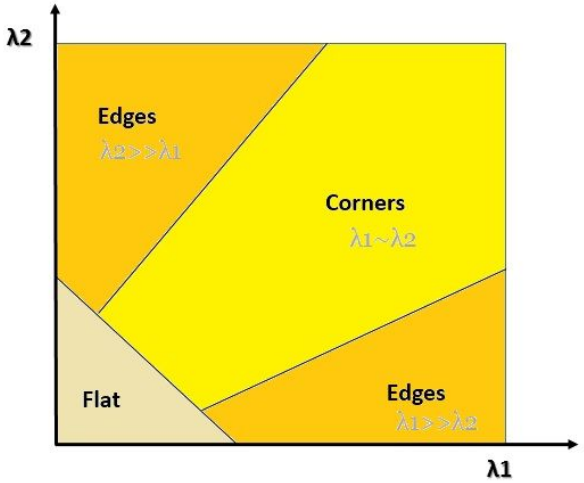
4) Где M:

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}$$

5) Нас в целом интересуют собственные значения M

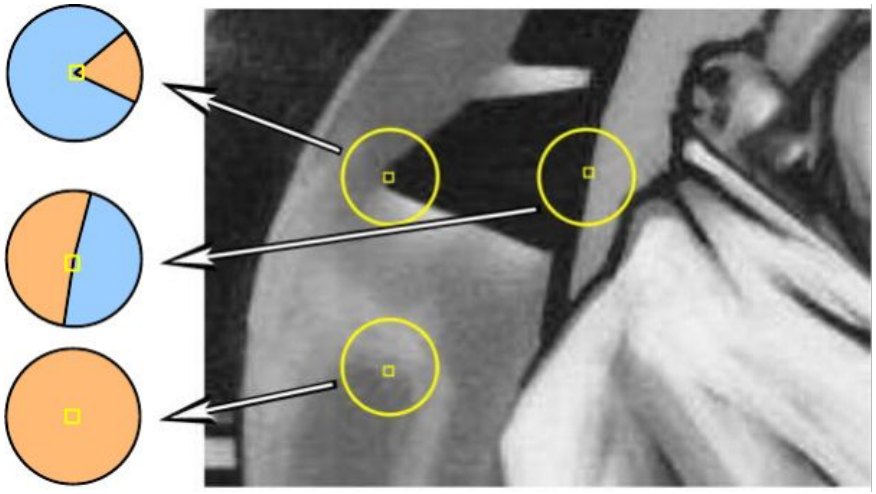
6) Предлагается использовать некоторую аппроксимацию

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(M) - k\,\mathrm{tr}(M)^2$$



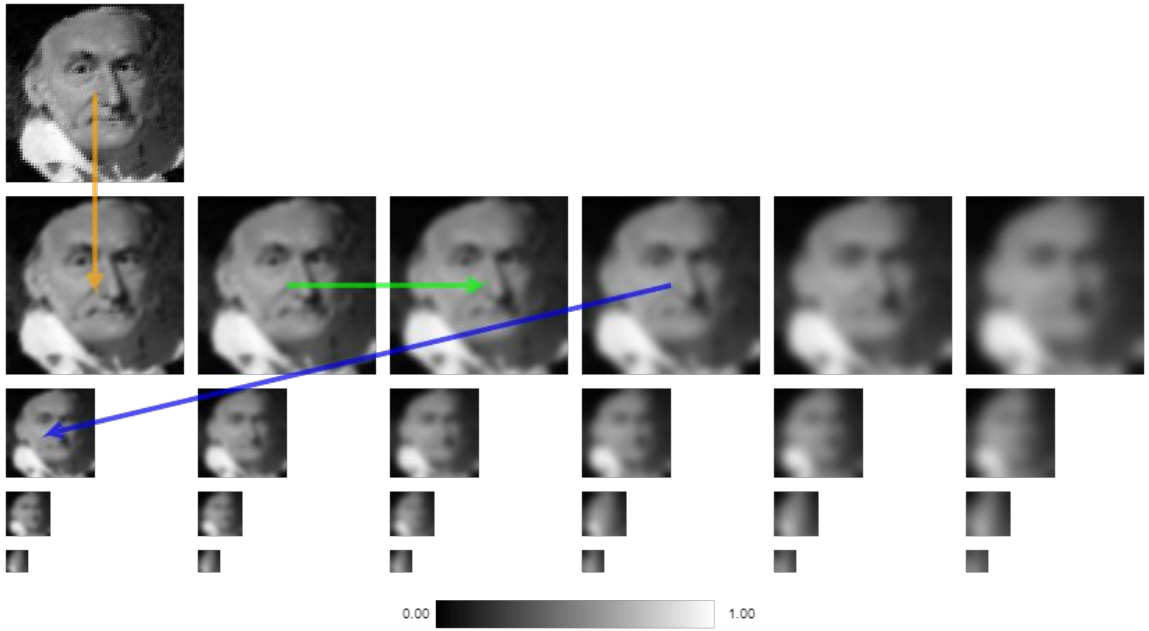
Smith и Brady (1997)

1) Строим для пикселя окрестность и разбиваем ее на похожие по интенсивности пиксели и не похожие
2) Там где площадь похожих маленькая скорее всего угол
3) Определяем центроиды и ложноположительные срабатывания

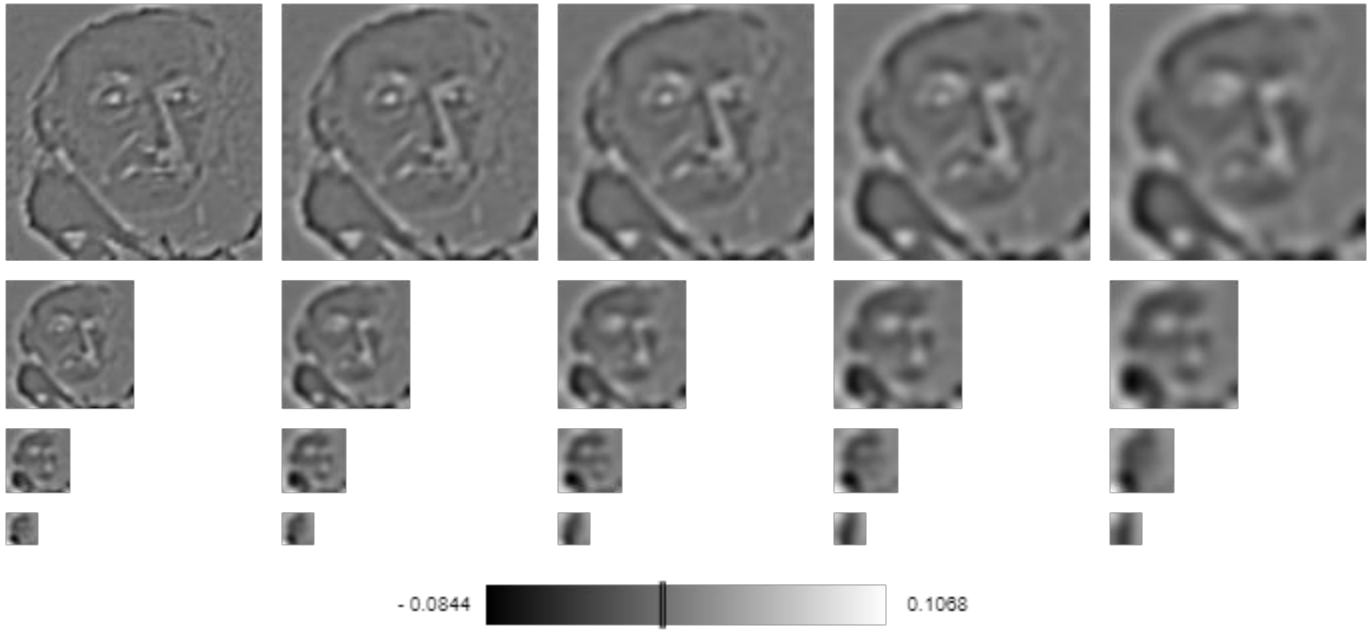


* Пикселям близким к центру можно дать больший вес

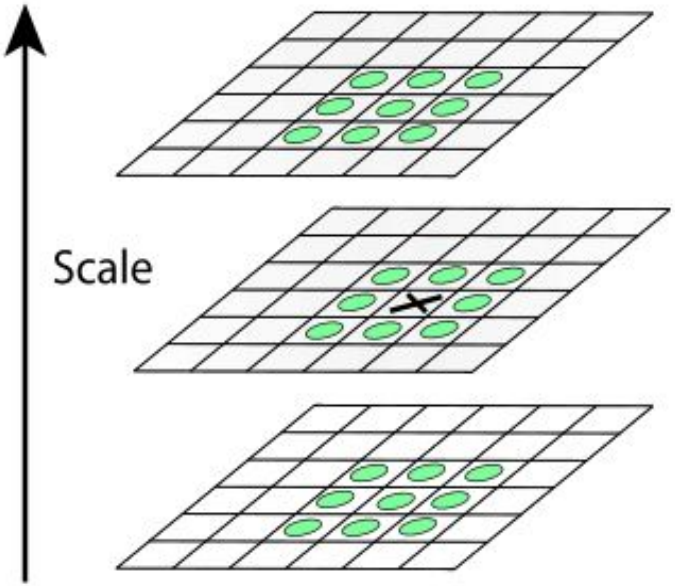
SIFT (1999)



Размываем и сжимаем



Вычитаем соседние картинки друг из друга



1) Ищем экстремумы

2) Уточняем

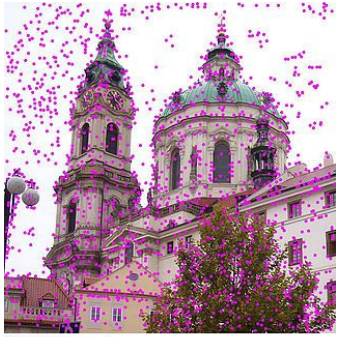
$$D(\mathbf{z}_0 + \mathbf{z}) \approx D(\mathbf{z}_0) + \left(\frac{\partial D}{\partial \mathbf{z}} \Big|_{\mathbf{z}_0} \right)^T \mathbf{z} + \frac{1}{2} \mathbf{z}^T \left(\frac{\partial^2 D}{\partial \mathbf{z}^2} \Big|_{\mathbf{z}_0} \right) \mathbf{z}$$

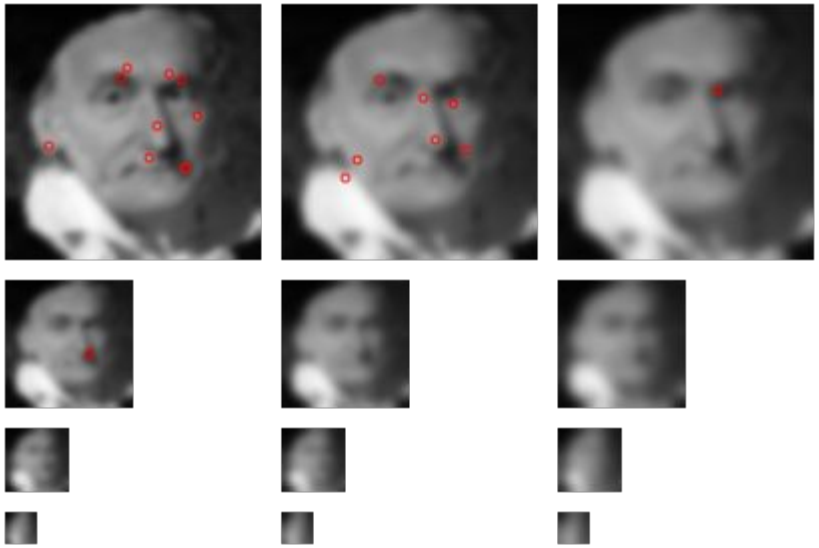
3) Отсеиваем шумы

4) Проверяем границы

$$\alpha = r\beta$$

$$\frac{\mathrm{Tr}^2(\mathbf{H})}{\mathrm{Det}(\mathbf{H})} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} < \frac{(r+1)^2}{r}$$





Находим ориентацию точки

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

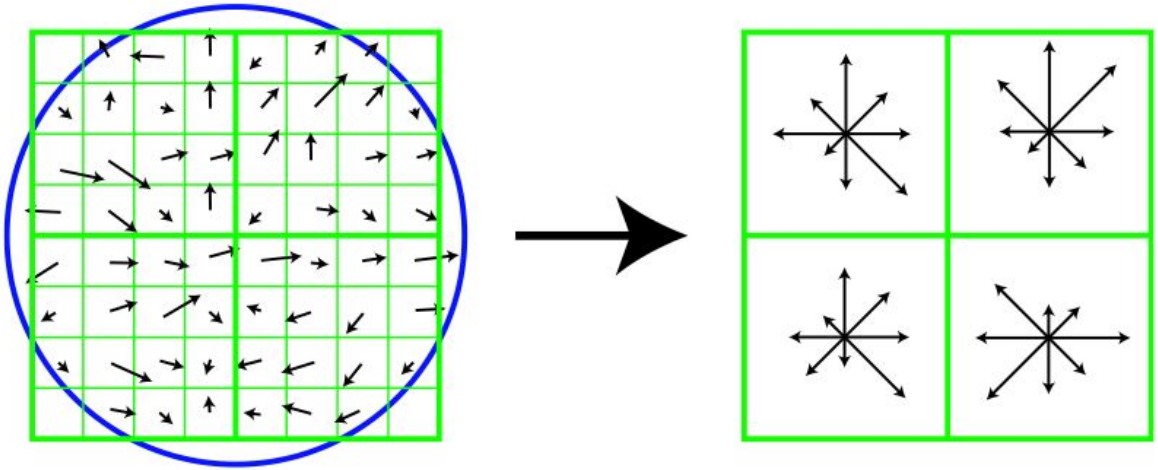
1) Берем 3 сигма окно
2) Разбиваем все направления на куски по 10 градусов, каждая точка в окне вносит вклад в направление по размеру своего градиента
3) Выбираем пик гистограммы и соседние с ним направления интерполируем направления и присваиваем результат нашей точке
4) Если встречаются еще достаточно большие направления их тоже добавляем

Мы научились строить особые точки

Интуитивные примеры

1) Панорамы
2) Стереопары
3) Реконструкция объекта из проекций
4) Распознавание объекта
5) Слежение по потоку снимков

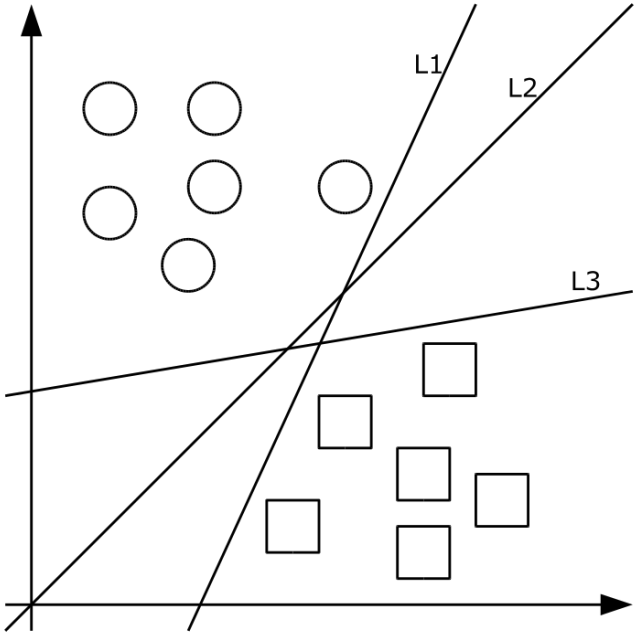
SIFT возвращение джедая



Вычисляем гистограммы для средних квадратов с учетом направления и масштаба ключевой точки и пишем в вектор

Полученные вектора дескрипторов классифицируем

1) Например SVM



Оптический поток

1) $I(x, y, t) \approx I(x + \delta x, y + \delta y, t + \delta t)$

2) $I(x + \delta x, y + \delta y, t + \delta t) \approx I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t$

3) $\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$

4) $\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0$

5) $\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$

6)
$$\begin{cases} I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ \dots \\ I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{cases}$$

7)
$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

8) $Av = b$

9) $\mathbf{v} = (A^T A)^{-1} A^T b$

10) $$\begin{bmatrix} V_x \\ \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \\ \sum_i I_x(q_i) I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

11) Строим пирамиду

Гистограмма ориентированных градиентов (2005)

1) [-1, 1], [-1, 1].T
2) $[-1, 0, 1]\ [-1, 0, 1]^T.$
3) $\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \qquad \mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$
4) $\begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix}.$

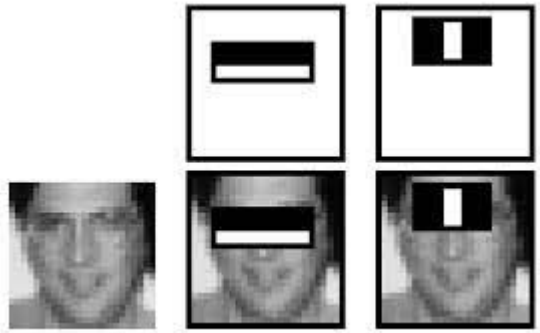
1) Разбиваем картинку на квадратные или прямоугольные ячейки
2) Для каждой ячейки строим гистограмму на 360 градусов с шагом в 10 градусов. Каждый пиксель в ячейке дает вклад в направление в соответствии с модулем своего градиента
3) Сдвигаем окно с заданным шагом (обычно с перекрытием)
4) Пишем гистограммы в вектор
5) Нормируем

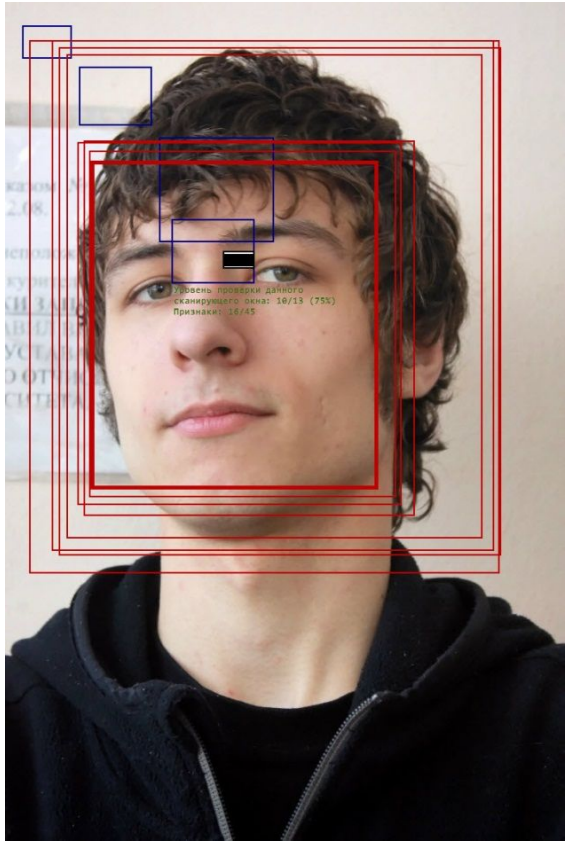
$$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

6) Классифицируем

Метод Виолы-Джонса (2001)

1) Используем сканирующее окно, ищущее признаки
2) Переводим изображение в интегральный формат (в ячейке сумма пикселей левее и выше)
3) Ищем признаки (как разность на белых пикселях и на черных)



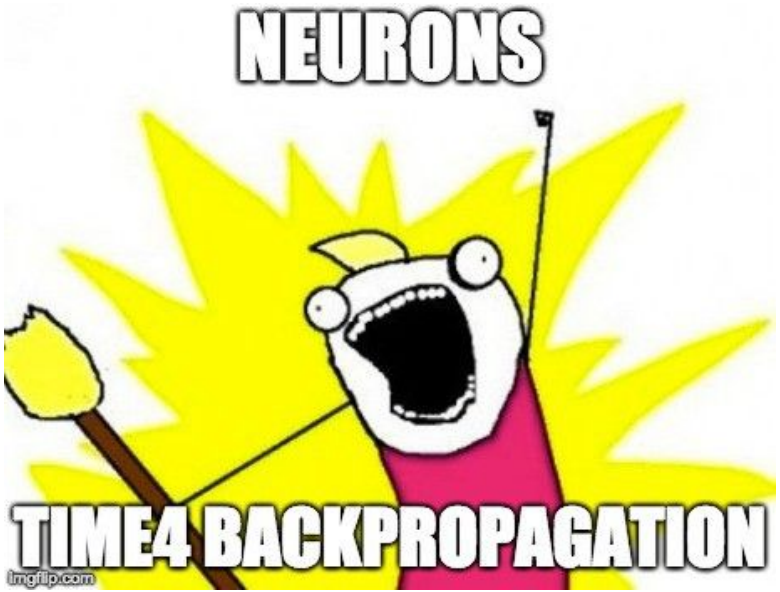


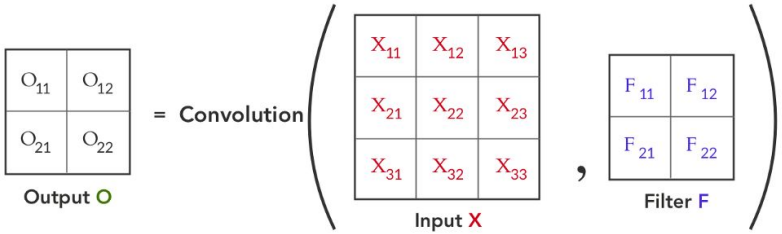
4) Строим признаковое описание $x = (f_1(x),\ldots,f_n(x))$

5) Масштабируем окно

6) Каскад классификаторов

И еще чуть-чуть о свертках





Local Gradients → (A)

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Finding derivatives with respect to F_{11} , F_{12} , F_{21} *and* F_{22}

$$\frac{\partial O_{11}}{\partial F_{11}} = X_{11} \quad \frac{\partial O_{11}}{\partial F_{12}} = X_{12} \quad \frac{\partial O_{11}}{\partial F_{21}} = X_{21} \quad \frac{\partial O_{11}}{\partial F_{22}} = X_{22}$$

Similarly, we can find the local gradients for O_{12} , O_{21} *and* O_{22}

For every element of F

$$\frac{\partial L}{\partial F_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial F_i}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{11}}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{12}}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{21}}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{22}}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

$$\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix} = \text{Convolution} \left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

where

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} = \text{Input X}$$

$$\begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} = \frac{\partial L}{\partial O}$$

Loss gradient from previous layer

Local Gradients: → (B)

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

Differentiating with respect to X_{11}, X_{12}, X_{21} *and* X_{22}

$$\frac{\partial O_{11}}{\partial X_{11}} = F_{11} \quad \frac{\partial O_{11}}{\partial X_{12}} = F_{12} \quad \frac{\partial O_{11}}{\partial X_{21}} = F_{21} \quad \frac{\partial O_{11}}{\partial X_{22}} = F_{22}$$

Similarly, we can find local gradients for O_{12}, O_{21} *and* O_{22}

For every element of X_i

$$\frac{\partial L}{\partial X_i} = \sum_{k=1}^{M} \frac{\partial L}{\partial O_k} * \frac{\partial O_k}{\partial X_i}$$

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial O_{11}} * F_{11}$$

$$\frac{\partial L}{\partial X_{12}} = \frac{\partial L}{\partial O_{11}} * F_{12} + \frac{\partial L}{\partial O_{12}} * F_{11}$$

$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial O_{12}} * F_{12}$$

$$\frac{\partial L}{\partial X_{21}} = \frac{\partial L}{\partial O_{11}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{11}$$

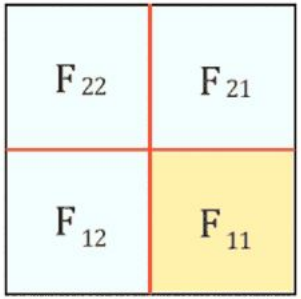
$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial O_{11}} * F_{22} + \frac{\partial L}{\partial O_{12}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{12} + \frac{\partial L}{\partial O_{22}} * F_{11}$$

$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial O_{12}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{12}$$

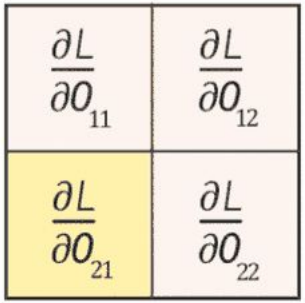
$$\frac{\partial L}{\partial X_{31}} = \frac{\partial L}{\partial O_{21}} * F_{21}$$

$$\frac{\partial L}{\partial X_{32}} = \frac{\partial L}{\partial O_{21}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{21}$$

$$\frac{\partial L}{\partial X_{33}} = \frac{\partial L}{\partial O_{22}} * F_{22}$$

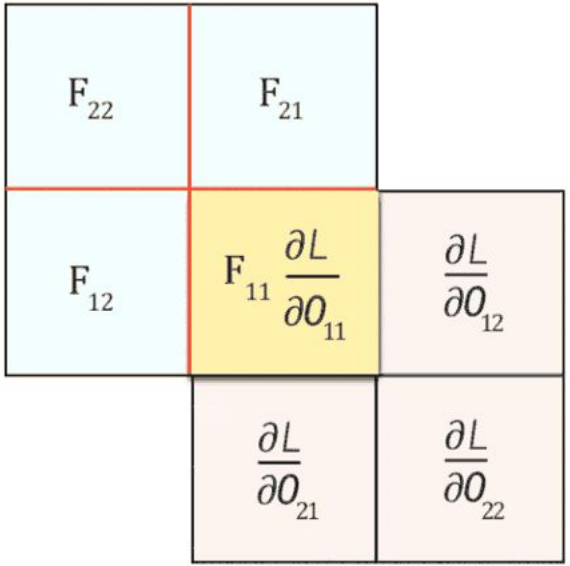


Filter F



Loss Gradient $\frac{\partial L}{\partial O}$

$$\frac{\partial L}{\partial X_{11}} = F_{11} * \frac{\partial L}{\partial O_{11}}$$




созрел вопрос

Быстрое вычисление свертки

$$\begin{pmatrix} x1 & x2 & x3 \\ x4 & x5 & x6 \\ x7 & x8 & x9 \end{pmatrix}$$

матрица

$$\begin{pmatrix} k1 & k2 \\ k3 & k4 \end{pmatrix}$$

фильтр

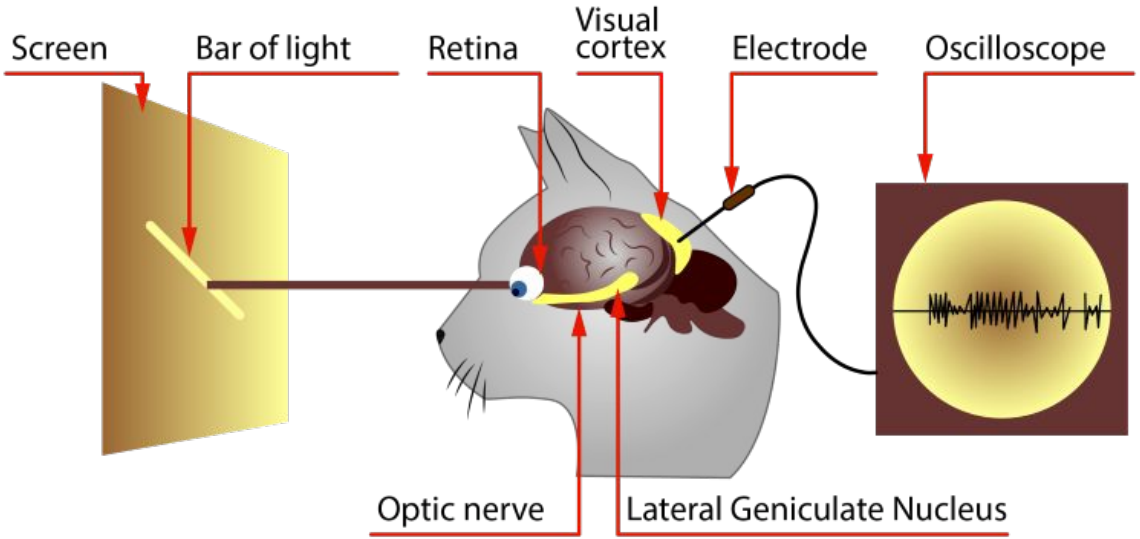
$$\begin{pmatrix} k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 & 0 \\ 0 & k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 & 0 \\ 0 & 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 \end{pmatrix} \cdot \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \end{pmatrix}$$

Теорема Пусть $C \in \mathbb{C}^{n \times n}$ – циркулянт, тогда

$$C = F_n^{-1} \operatorname{diag}(F_n c) F_n$$

c – первый столбец C

David Hubel, Torsten Wiesel (1959)

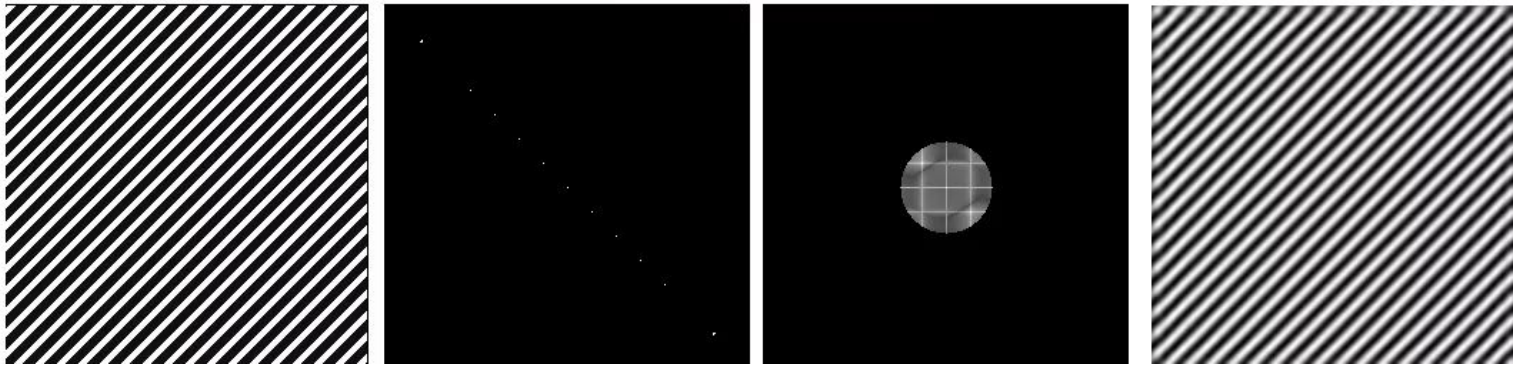


Свертка и преобразования Фурье

$$h(x) = f \otimes g = \int_{-\infty}^{\infty} f(x-u)g(u)\,du = \mathcal{F}^{-1}\left(\sqrt{2\pi}\mathcal{F}[f]\mathcal{F}[g]\right)$$

$$\text{feature map} = \text{input} \otimes \text{kernel} = \sum_{y=0}^{\text{columns}} \left(\sum_{x=0}^{\text{rows}} \text{input}(x-a, y-b)\,\text{kernel}(x,y) \right) = \mathcal{F}^{-1}\left(\sqrt{2\pi}\mathcal{F}[\text{input}]\mathcal{F}[\text{kernel}]\right)$$

Оригинал *Преобразование Фурье* *Фильтр* *Отфильтрованное изображение*



Механика жидкостей

$$Z = \mathrm{softmax}\left[\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}\right] = \begin{pmatrix} 0.0001 & 0.0001 & 0.0001 \\ 0.0001 & 0.9992 & 0.0001 \\ 0.0001 & 0.0001 & 0.0001 \end{pmatrix}$$

$$Z = \mathrm{softmax}\left[\begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}\right] = \begin{pmatrix} 0.105 & 0.1125 & 0.105 \\ 0.1125 & 0.13 & 0.1125 \\ 0.105 & 0.1125 & 0.105 \end{pmatrix}$$

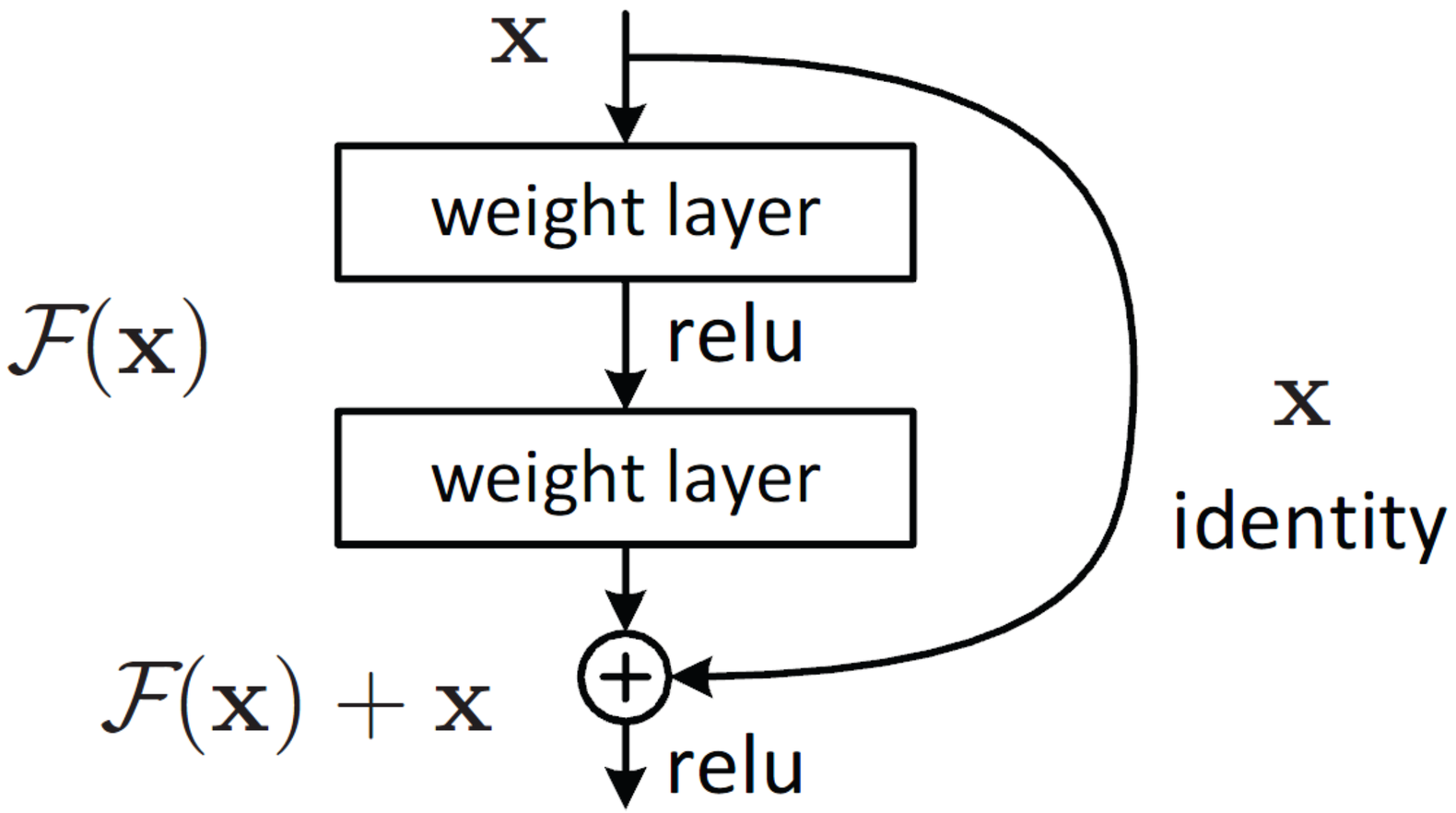
Classification

Supervised classification - training a classifier to detect certain objects

Unsupervised classification - for example, grouping objects into groups

Different fun structures (that were discussed in DL):
AlexNet - conv: 11 x 11, 5x5, 3x3; max pool, full net at the end
Visual Geometry Group - smaller conv layers, block system + 1x1 conv
ResNet: residual blocks!

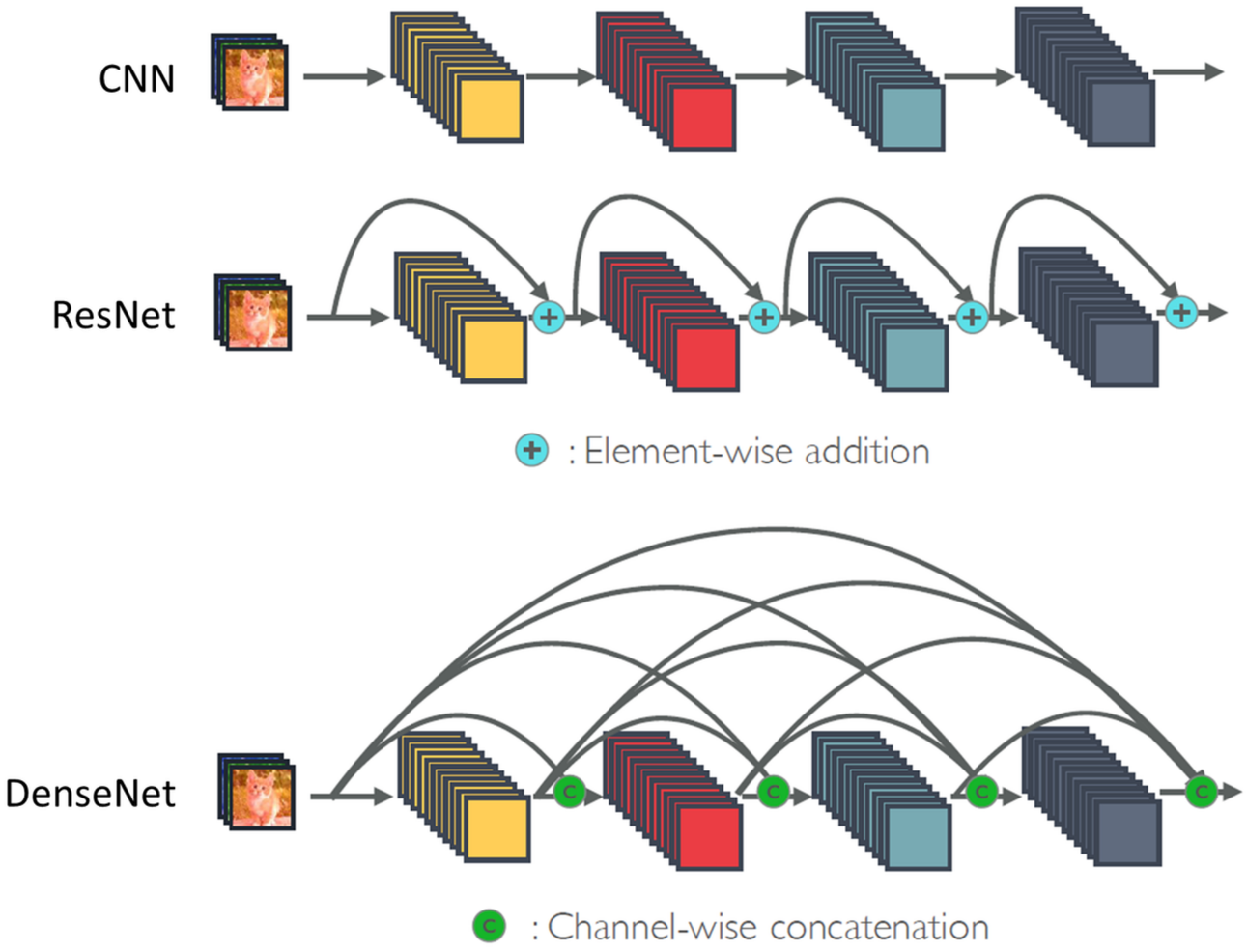
ResNet



Effective error propagation allows to train networks with greater depth while increasing accuracy

But what if we built ResNet on steroids?

DenseNet (2017)



Direct error propagation to earlier levels

Maintains low complexity features

More efficient than ResNet (computation)

DSNet

$$
\begin{aligned}
f_l &= H_l * (X_0/X_1/.../X_l) \\
&= (H_l^0/H_l^1/.../H_l^l) * (X_0/X_1/.../X_l) \\
&= H_l^0 * X_0 + H_l^1 * X_1 + \cdots + H_l^l * X_l,
\end{aligned}
$$

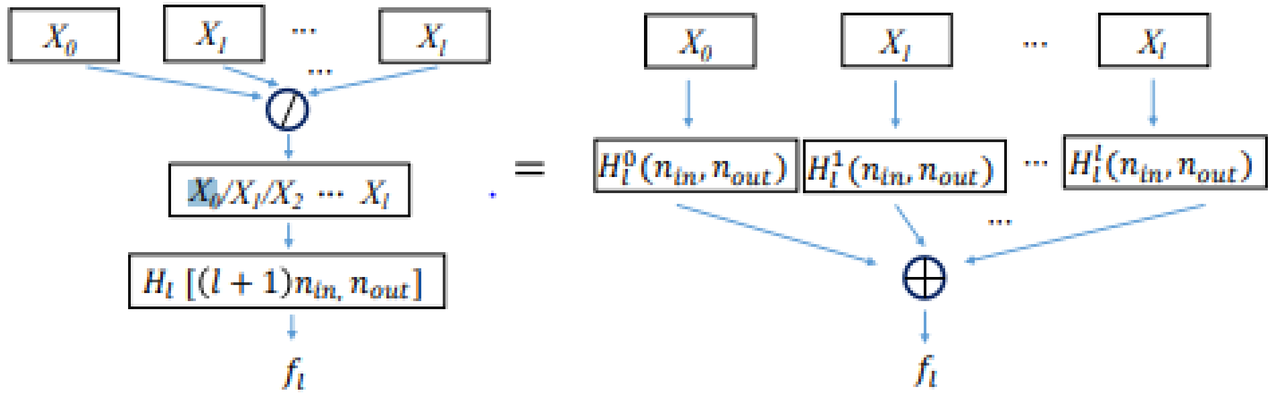


Figure 2. Equivalence of Dense concatenation before convolution and Dense summation after convolution.

$$
\begin{aligned}
f_l = H_l * DS_l^0(X_0) + H_l * DS_l^1(X_1) + \ldots \\
+ H_l * DS_l^{l-1}(X_{l-1}) + H_l * X_l.
\end{aligned} \tag{10}
$$

which is equivalent to

$$
\begin{aligned}
f_l = H_l * (DS_l^0(X_0) + DS_l^1(X_1) + \ldots \\
+ DS_l^{l-1}(X_{l-1}) + X_l),
\end{aligned} \tag{11}
$$

where "$DS()$" indicates dense shortcut. It refers to dense weighted normalized shortcut consisting of normalization and channel-wise weight. Specifically, $DS_l^i(X_i) = W_l^i \times N(X_i)$, where W_l^i represents channel-wise weight and and N indicates normalization operation. In this work, the term

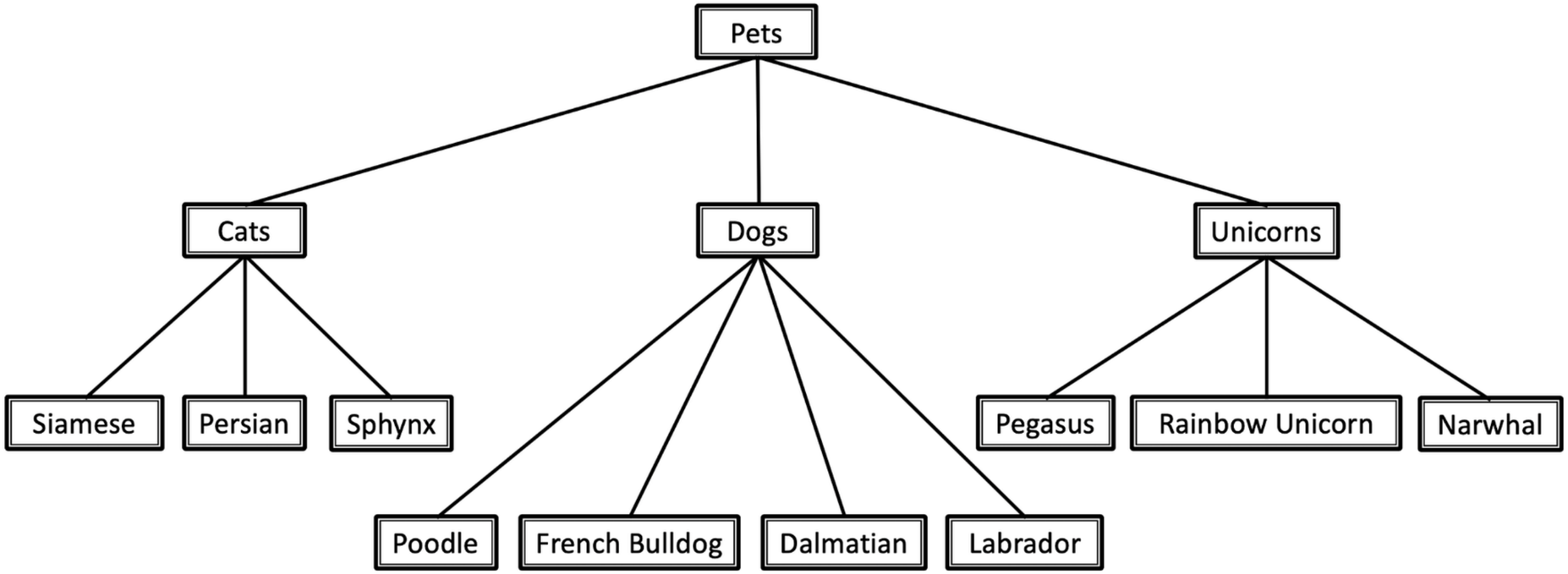
Getting rid of increasing CNN network sizes

Beats ResNet, relatively easy to train

Classification in a large number of categories

How do we turn a multi-class classification problem into a binary one?

OvO - One vs One	OvR - One vs Rest
Train $\frac{n(n-1)}{2}$ classifiers for each pair of classes	Train n classifiers for each class
Choose by softmax	Choose the most "certain" classifier

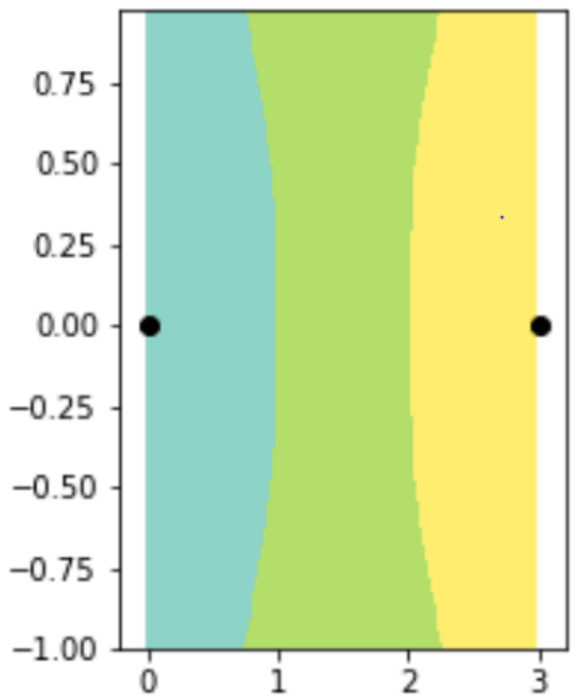

Pets
Cats
Dogs
Unicorns
Siamese
Persian
Sphynx
Poodle
French Bulldog
Dalmatian
Labrador
Pegasus
Rainbow Unicorn
Narwhal

SLaPKNN (Less than One-Shot)

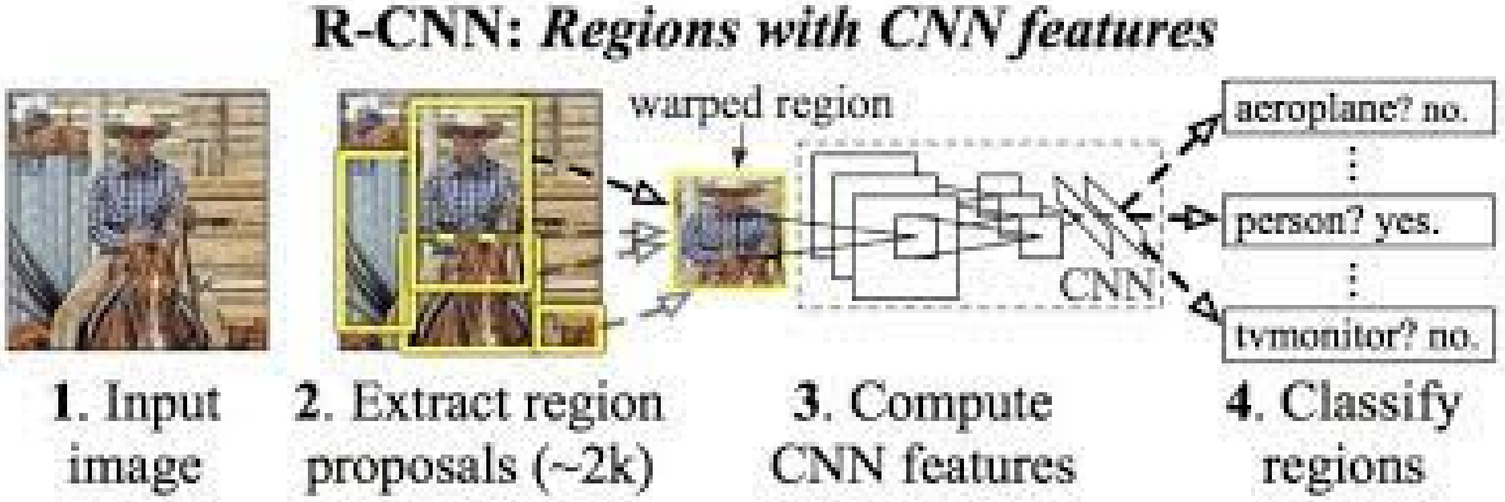
It's possible for a model to learn N classes with M < N objects

Probabiistic Soft Label = [0.6, 0.4, 0]
Soft Label = [3, 2, 0]

It is even possible for a SLaPKNN to separate n classes with only 2 soft labels.



Object detection: RCNN



Selective Search points out areas that are likely to be objects using segmentation

On average: about 2000 areas

Every area is treated as an object and gets processed by CNN.

R-CNN disadvantages

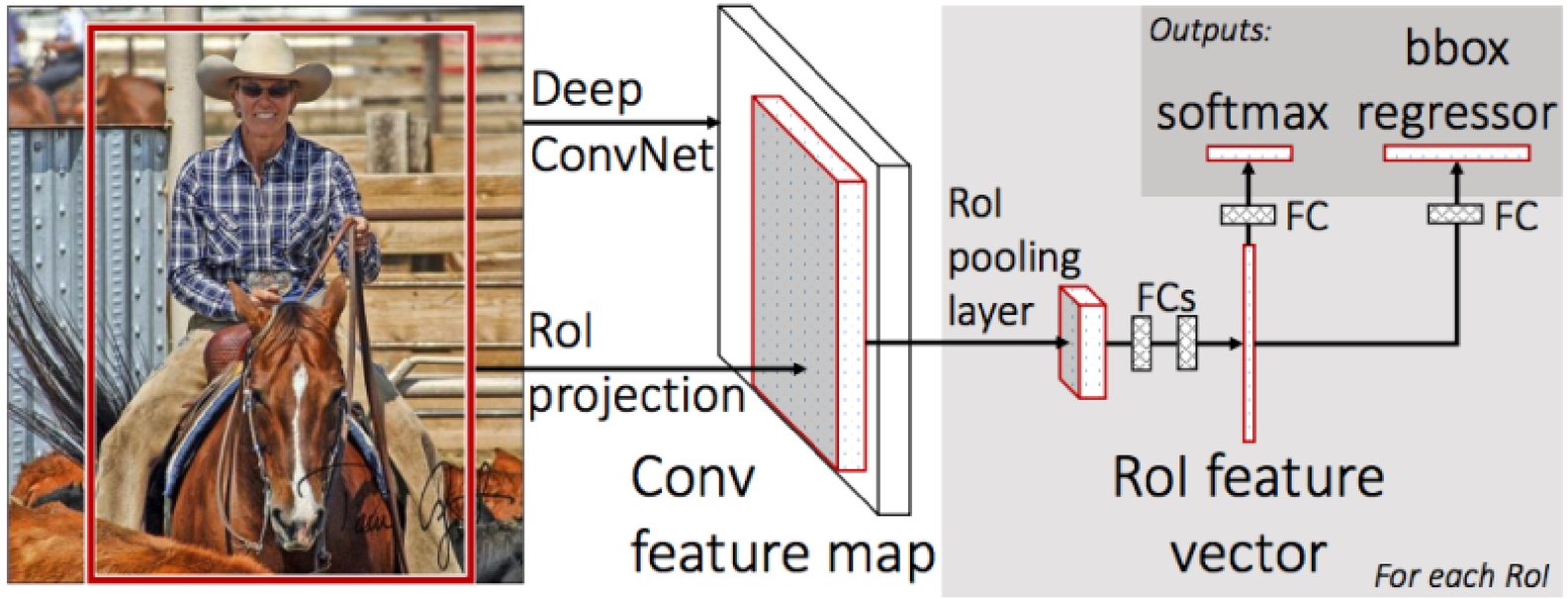
Time complexity: Impossible to use in real time or in videos

Takes too long to train (2000 regions)

Selective search is not an ML algorithm and is not trainable

Solution: Linear regression on box sizes (bbox regressor)

Fast R-CNN



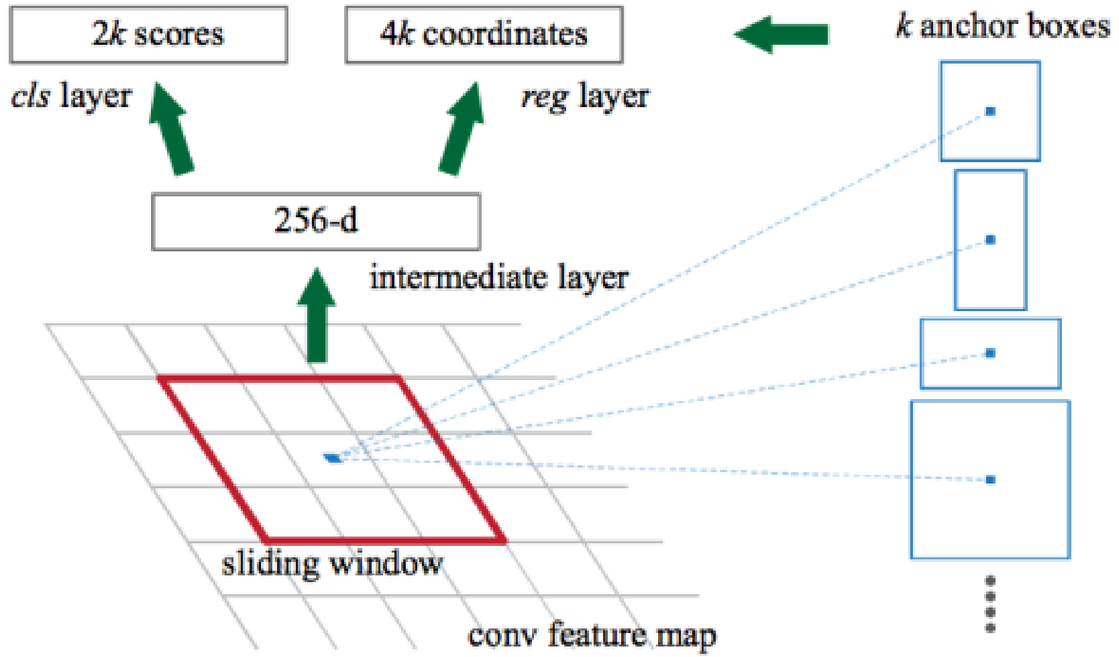
Processing the whole image at once and using regions on the feature map

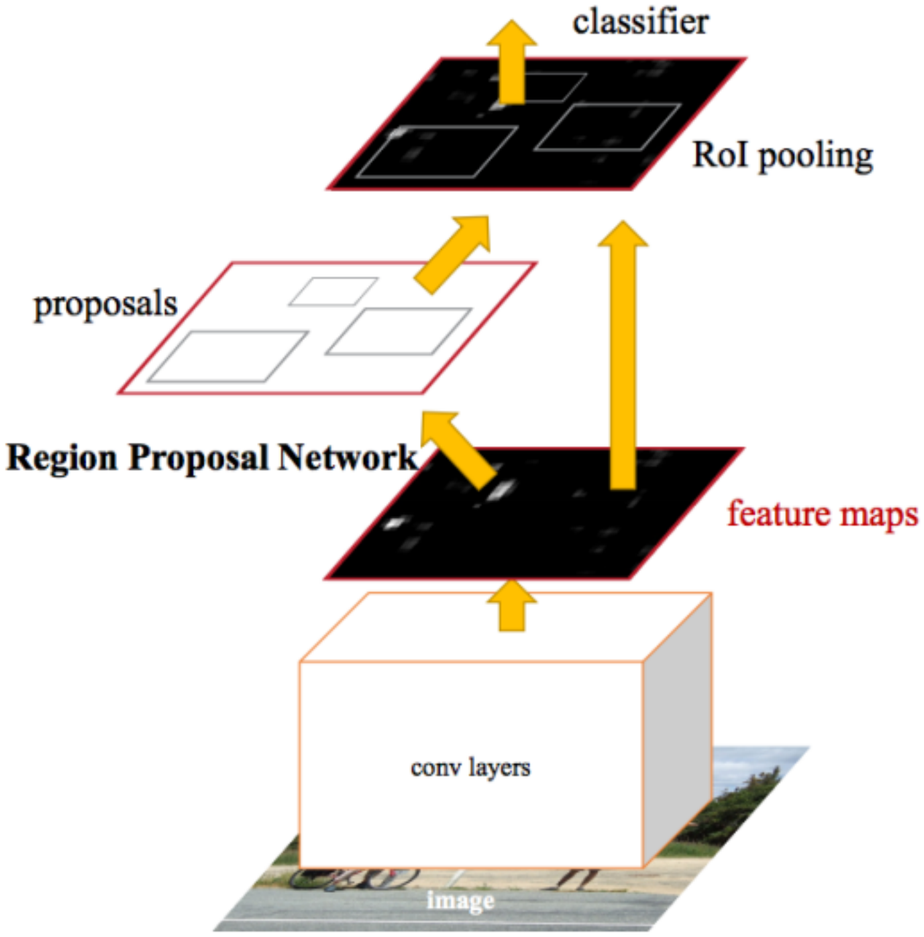
CNN, Softmax and bbox regressor are all training simultaneously

Fastererer R-CNN

Generating regions using feature maps

RPN:

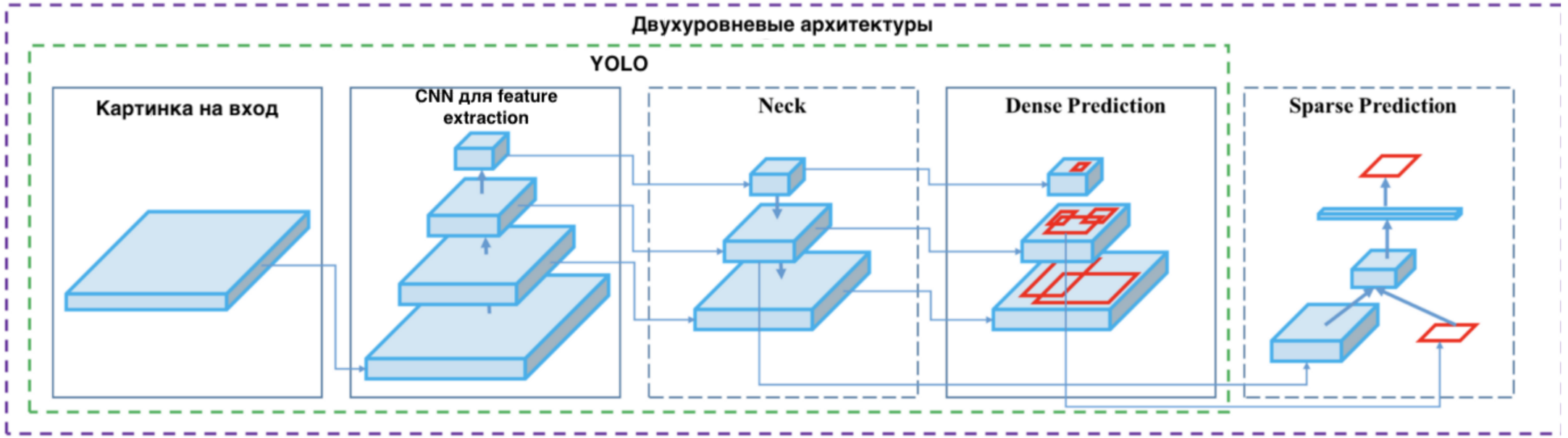


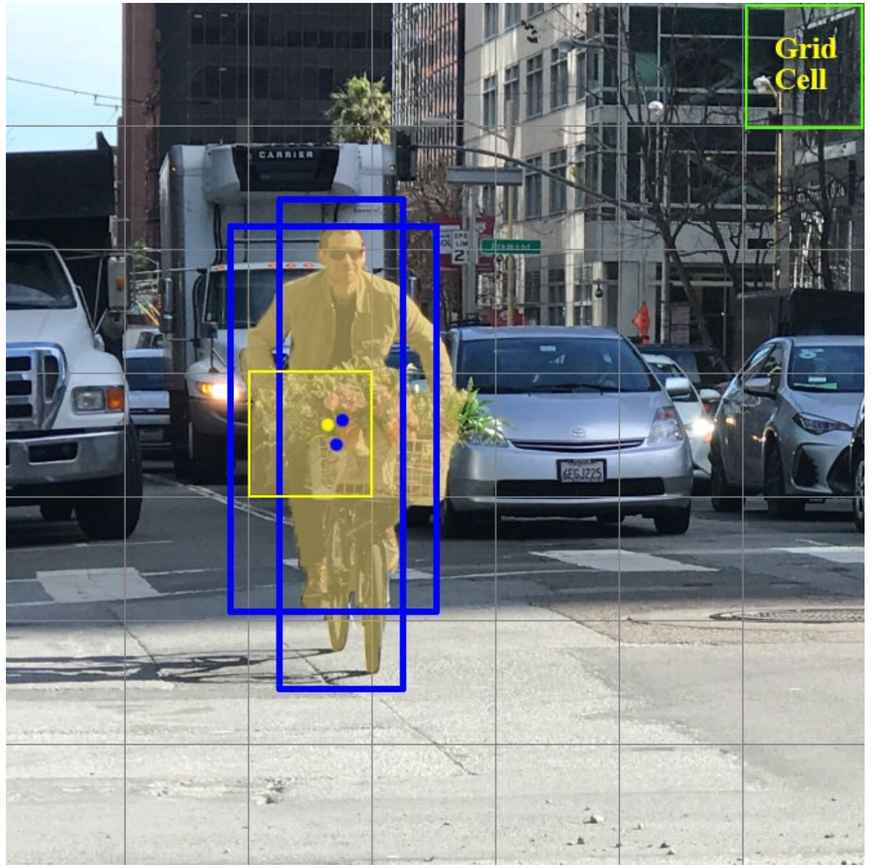


Cls = probability that this region contains an object

Object Detection

YOLOOOOO - (You Only Look Once)





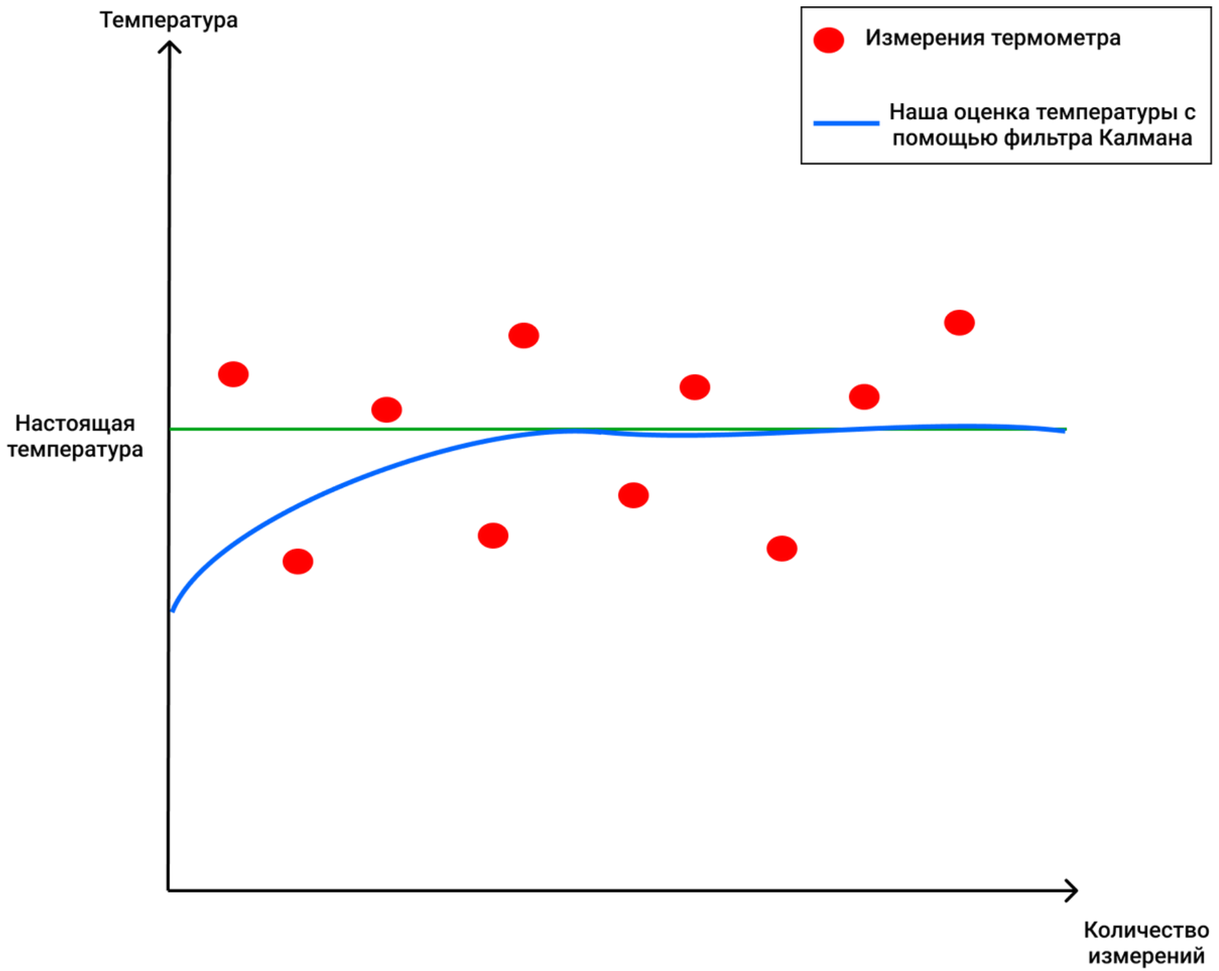
For each box we set a confidence score, that helps us combine them

Kalman Gain

$$KG = \frac{E_{EST}}{E_{EST} + E_{MEA}} \quad (1)$$

$$EST_t = EST_{t-1} + KG * (MEA - EST_{t-1})$$

$$E_{EST_t} = \frac{E_{MEA} * E_{EST_{t-1}}}{E_{MEA} + E_{EST_{t-1}}} = (1 - KG) * E_{EST_{t-1}}$$

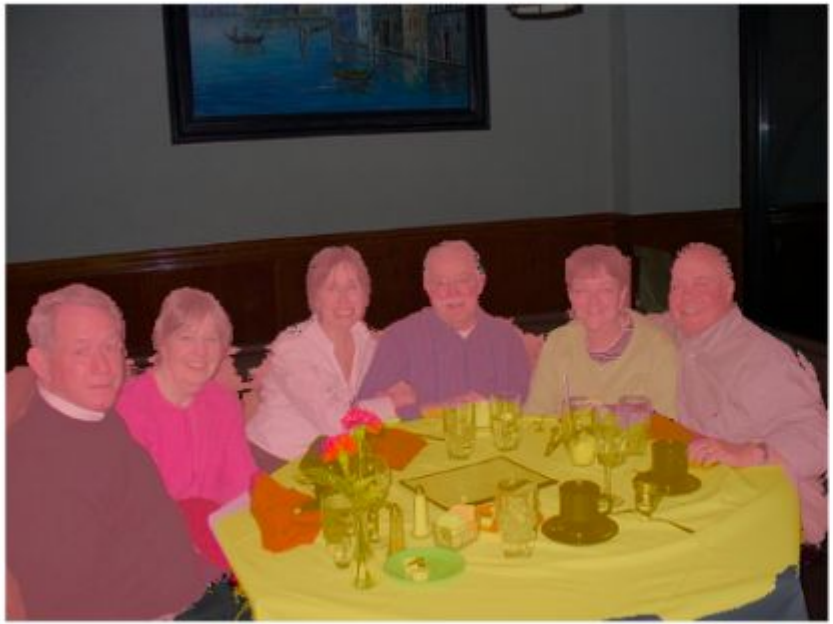


DeepSort

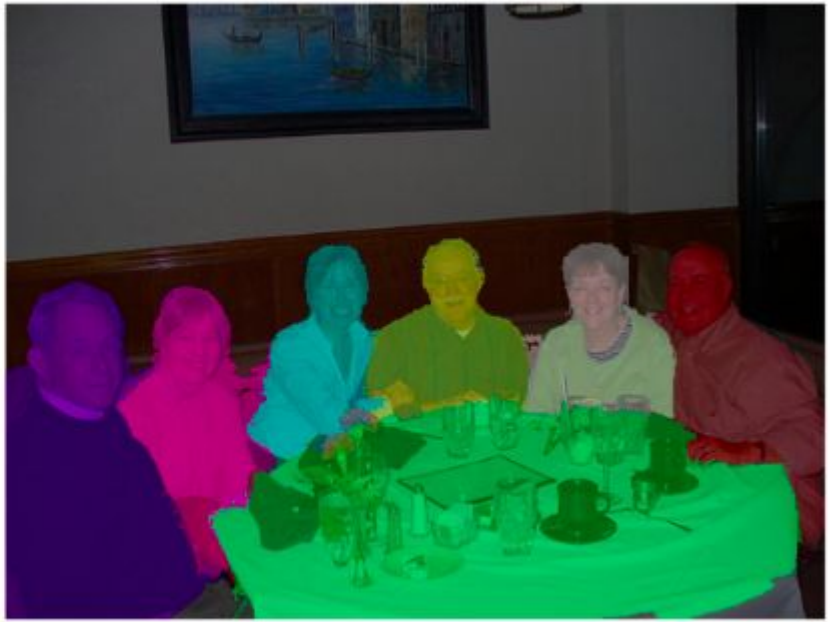
Using YOLO, detect objects and **classify their appearance.** Then, if an object is blocked by another object for a couple of frames, it is assigned the same label as before.

Using Kaplan Gain, count the distance between a new YOLO object and previous ones.

Segmentation



Semantic Segmentation



Instance Segmentation

Loss functions

Balanced Cross-Entropy

$$L_{BCE}(y, \hat{y}) = -(ylog(\hat{y}) + (1-y)log(1-\hat{y}))$$

Weighted Balanced Cross-Entropy

$$L_{W-BCE}(y, \hat{y}) = -(\beta * ylog(\hat{y}) + (1-y)log(1-\hat{y}))$$

Focal Loss

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

Dice Loss

$$DL(y, \hat{p}) = 1 - \frac{2y\hat{p} + 1}{y + \hat{p} + 1}$$

Tversky Loss

$$TL(p, \hat{p}) = 1 - \frac{1 + p\hat{p}}{1 + p\hat{p} + \beta(1-p)\hat{p} + (1-\beta)p(1-\hat{p})}$$

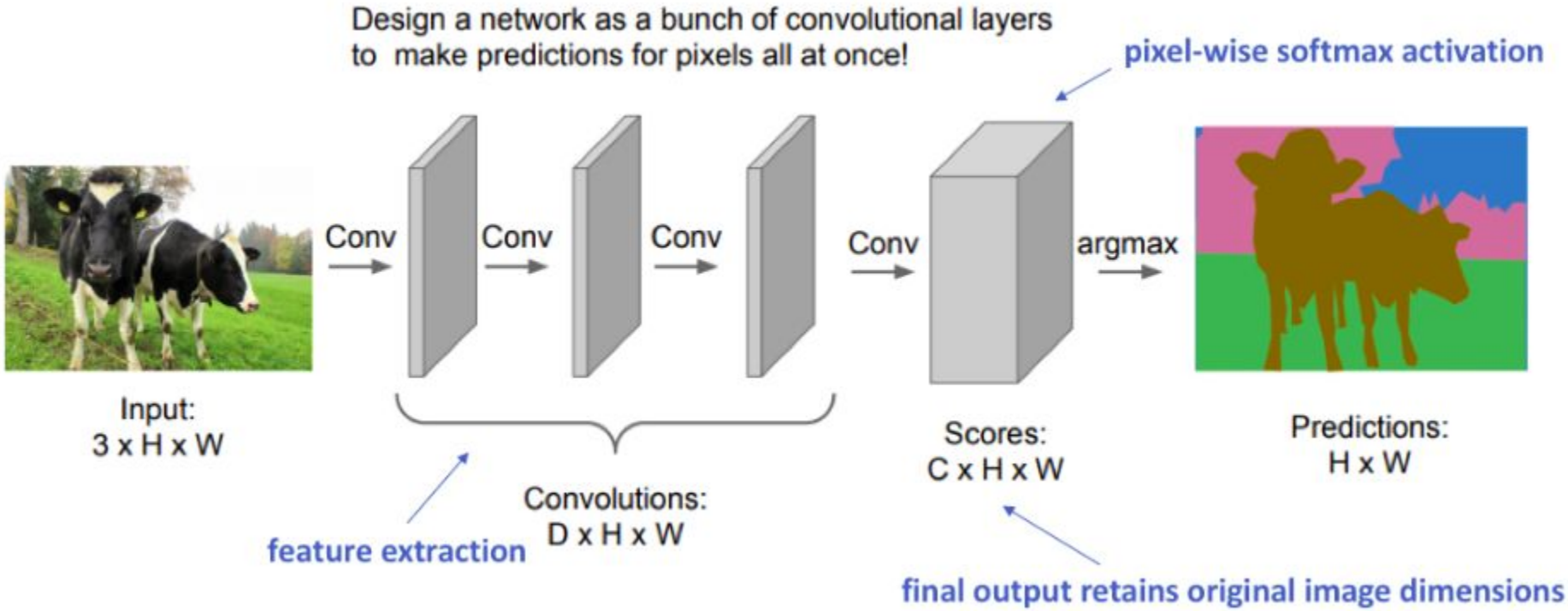
Sensitivity Specificity Loss

$$sensitivity = \frac{TP}{TP + FN} \quad specificity = \frac{TN}{TN + FP}$$

Loss Functions

Loss Function	Use cases
Binary Cross-Entropy	Works best in equal data distribution among classes scenarios Bernoulli distribution based loss function
Weighted Cross-Entropy	Widely used with skewed dataset Weighs positive examples by β coefficient
Balanced Cross-Entropy	Similar to weighted-cross entropy, used widely with skewed dataset weighs both positive as well as negative examples by β and $1 - \beta$ respectively
Focal Loss	works best with highly-imbalanced dataset down-weight the contribution of easy examples, enabling model to learn hard examples
Distance map derived loss penalty term	Variant of Cross-Entropy Used for hard-to-segment boundaries
Dice Loss	Inspired from Dice Coefficient, a metric to evaluate segmentation results. As Dice Coefficient is non-convex in nature, it has been modified to make it more tractable.
Sensitivity-Specificity Loss	Inspired from Sensitivity and Specificity metrics Used for cases where there is more focus on True Positives.
Tversky Loss	Variant of Dice Coefficient Add weight to False positives and False negatives.

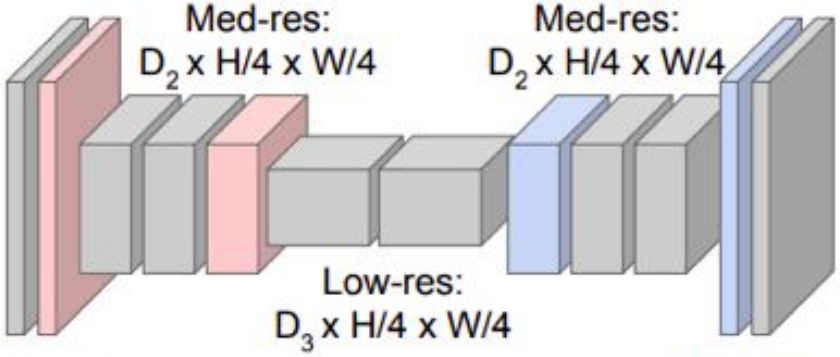
Naive architecture



Encoder-Decoder Design

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W

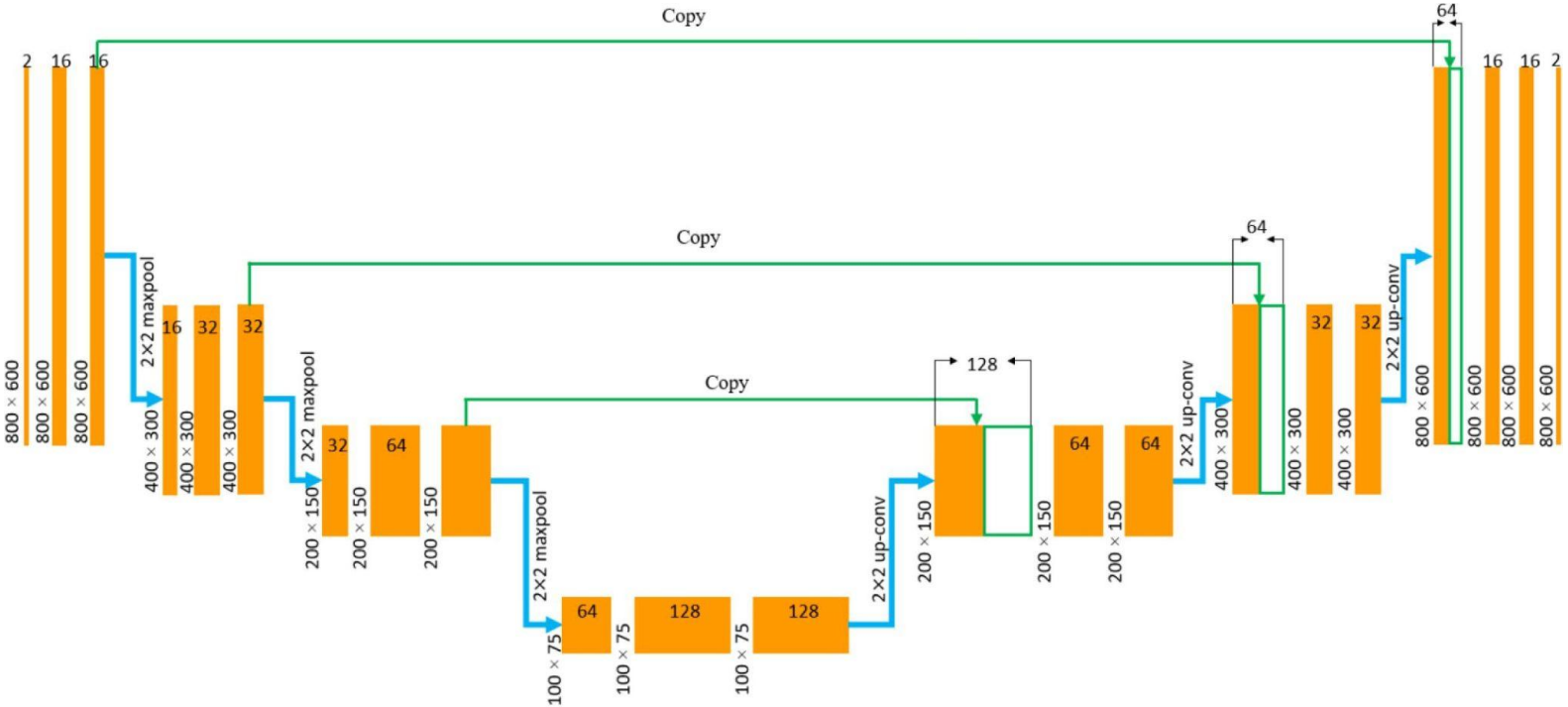


High-res:
D_1 x H/2 x W/2

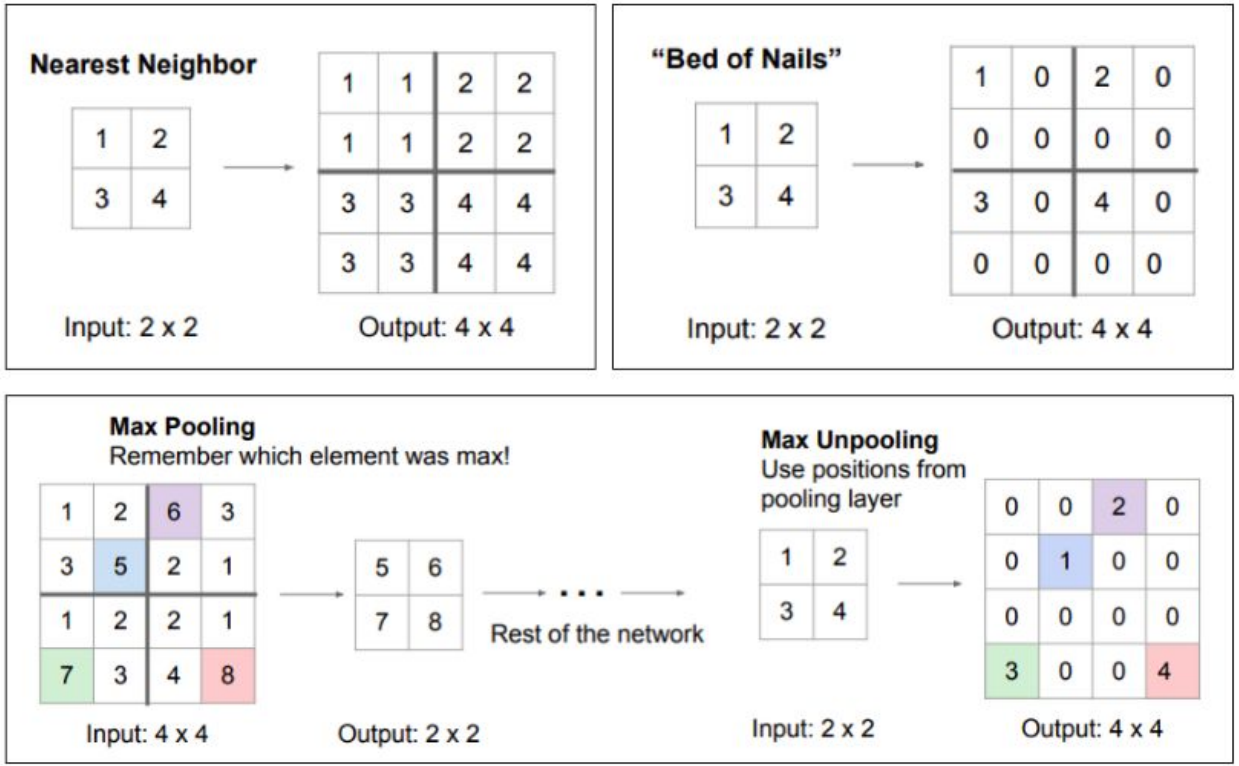
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

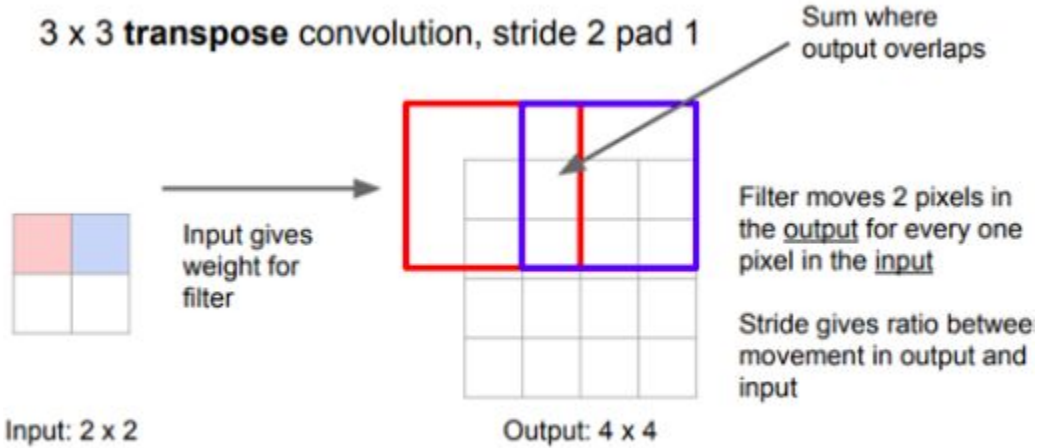
U-NET

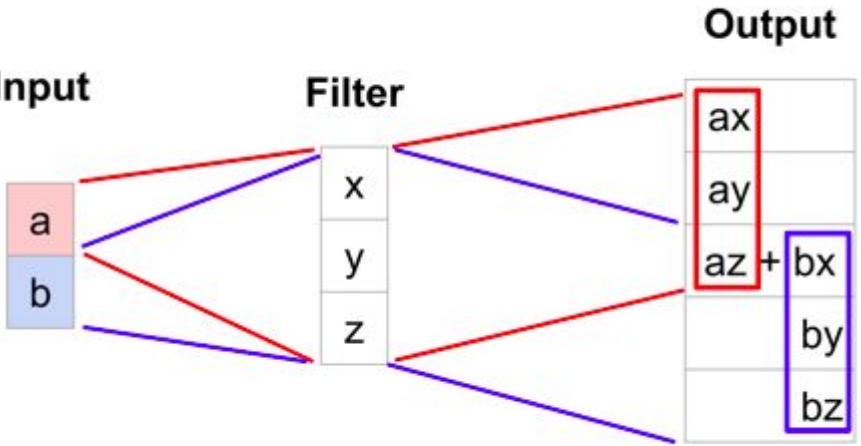

Copy
Copy
Copy
2
16
16
800 × 600
800 × 600
800 × 600
2×2 maxpool
16
32
32
400 × 300
400 × 300
400 × 300
2×2 maxpool
32
64
200 × 150
200 × 150
200 × 150
2×2 maxpool
64
128
128
100 × 75
100 × 75
100 × 75
2×2 up-conv
128
64
64
200 × 150
200 × 150
200 × 150
2×2 up-conv
64
32
32
400 × 300
400 × 300
400 × 300
2×2 up-conv
64
16
16
2
800 × 600
800 × 600
800 × 600
800 × 600
800 × 600
800 × 600
All convolutions are 3×3, Relu

Upsample

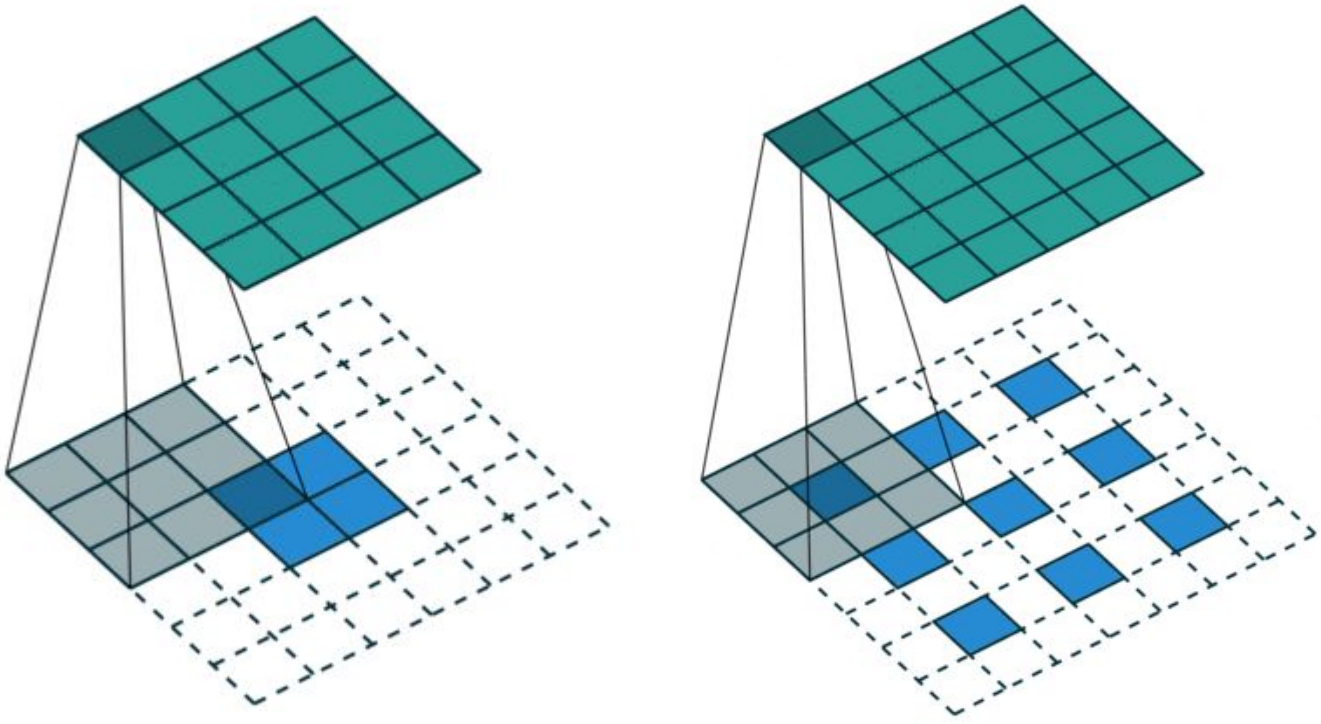

Nearest Neighbor
1 2
3 4
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
Input: 2 x 2
Output: 4 x 4
"Bed of Nails"
1 2
3 4
1 0 2 0
0 0 0 0
3 0 4 0
0 0 0 0
Input: 2 x 2
Output: 4 x 4
Max Pooling
Remember which element was max!
1 2 6 3
3 5 2 1
1 2 2 1
7 3 4 8
5 6
7 8
Rest of the network
Input: 4 x 4
Output: 2 x 2
Max Unpooling
Use positions from pooling layer
1 2
3 4
0 0 2 0
0 1 0 0
0 0 0 0
3 0 0 4
Input: 2 x 2
Output: 4 x 4

Upsample. Transpose Convolutions





Upsample. Transpose Convolutions



Upsample. Transpose Convolutions

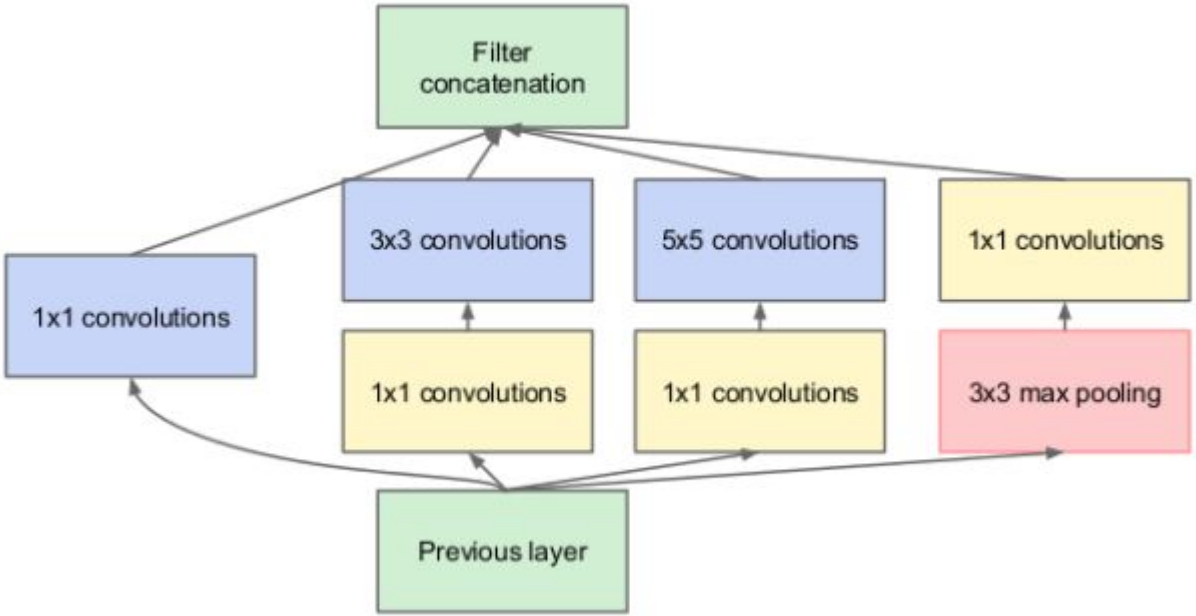
Deconv in last two layers.
Other layers use resize-convolution.
Artifacts of frequency 2 and 4.

Deconv only in last layer.
Other layers use resize-convolution.
Artifacts of frequency 2.

All layers use resize-convolution.
No artifacts.

Нужно избавиться от overlap!

Inception


Filter concatenation
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

Depth-Wise Convolution

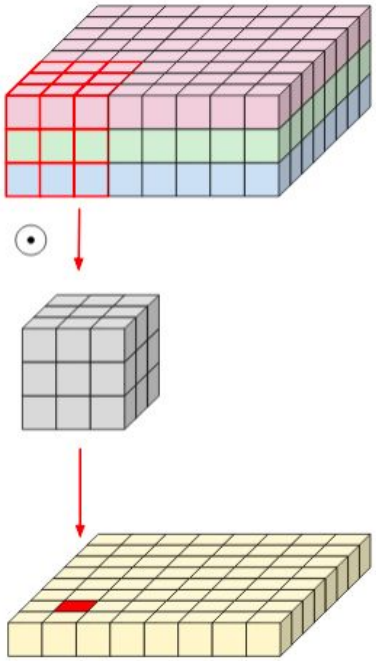


Fig 1. Normal convolution

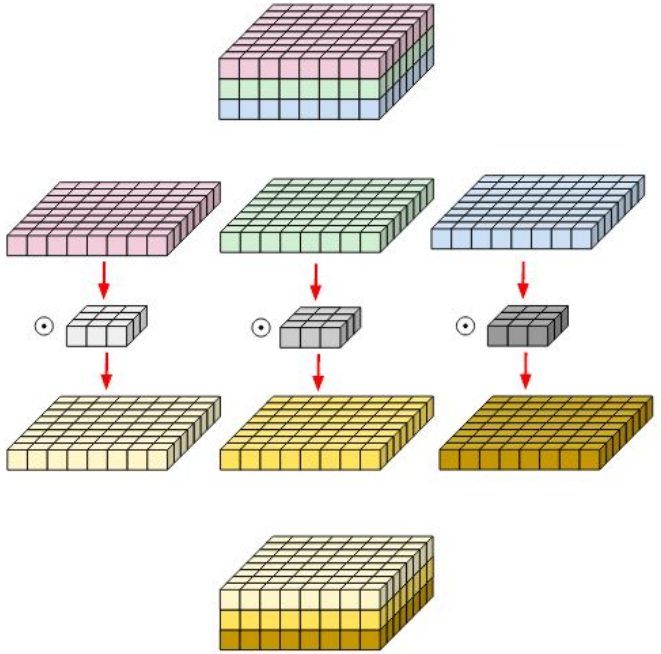
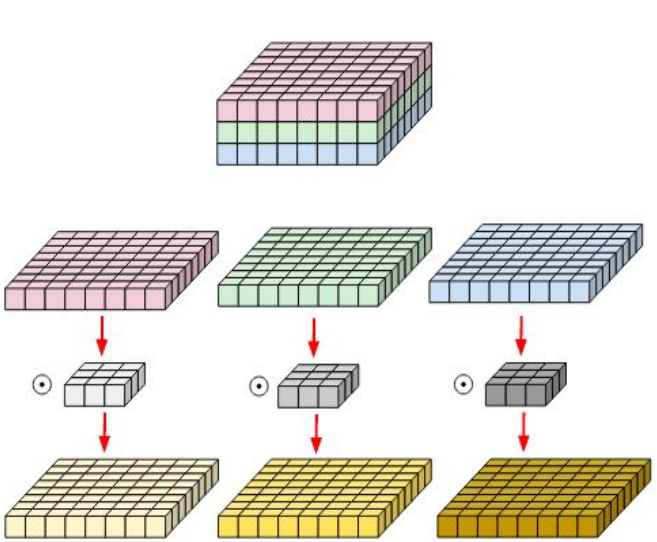


Fig 2. Depth-wise convolution. Filters and image have been broken into three different channels and then convolved separately and stacked thereafter

Depth-Wise Separable Convolution



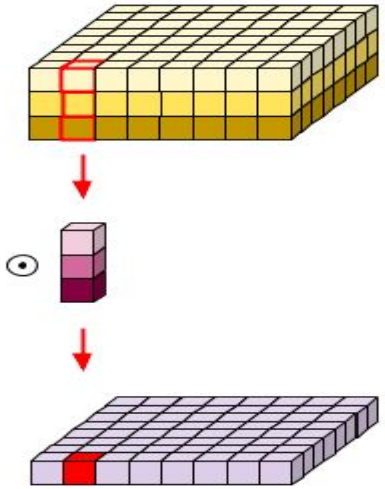
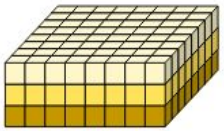


Fig 4. Depth-wise separable convolution



	Стандартная свертка	Depthwise Separation Convolutions	Выигрыш (DSC / стандартная)
Память	$K \cdot L \cdot C \cdot C'$	$K \cdot L \cdot C + 1 \cdot 1 \cdot C \cdot C'$	$1 / C' + 1 / (K \cdot L)$
Время	$K \cdot L \cdot N \cdot M \cdot C \cdot C'$	$K \cdot L \cdot N \cdot M \cdot C + 1 \cdot 1 \cdot C \cdot C' * N * M$	$1 / C' + 1 / (K \cdot L)$

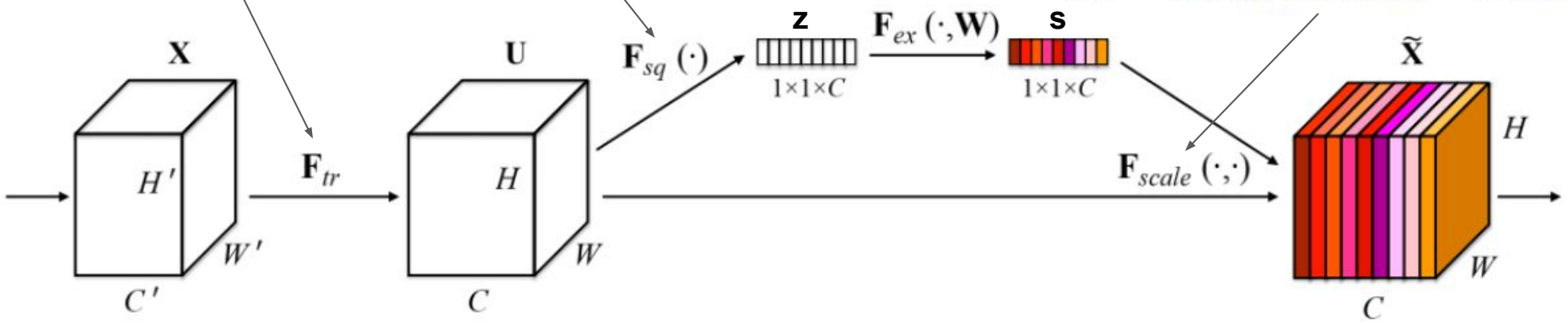
Squeeze-and-Excitation Blocks

$$\mathbf{u}_c = \mathbf{v}_c * \mathbf{X} = \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s$$

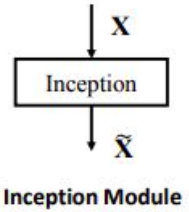
$$\mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j)$$

$$\mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W})) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z}))$$

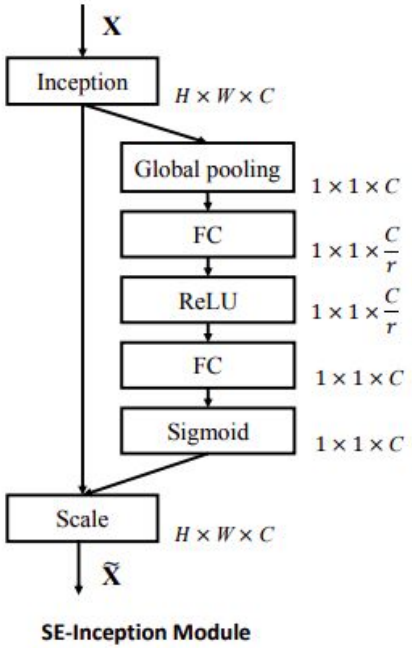
$$\tilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \mathbf{u}_c$$



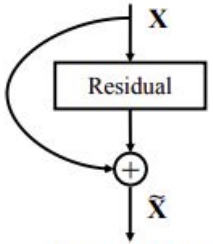
Squeeze-and-Excitation Blocks



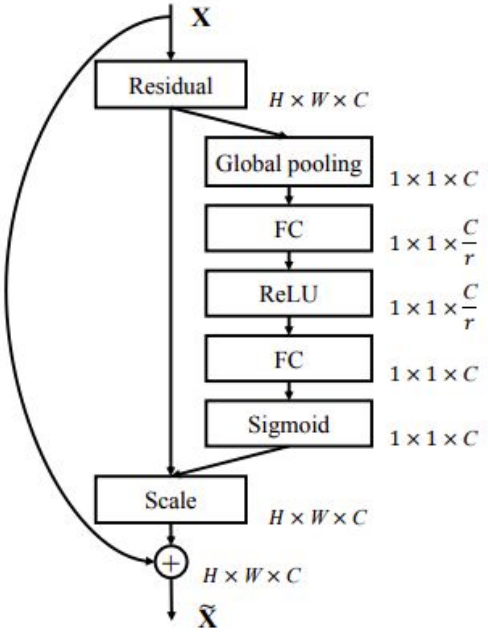
Inception Module



SE-Inception Module

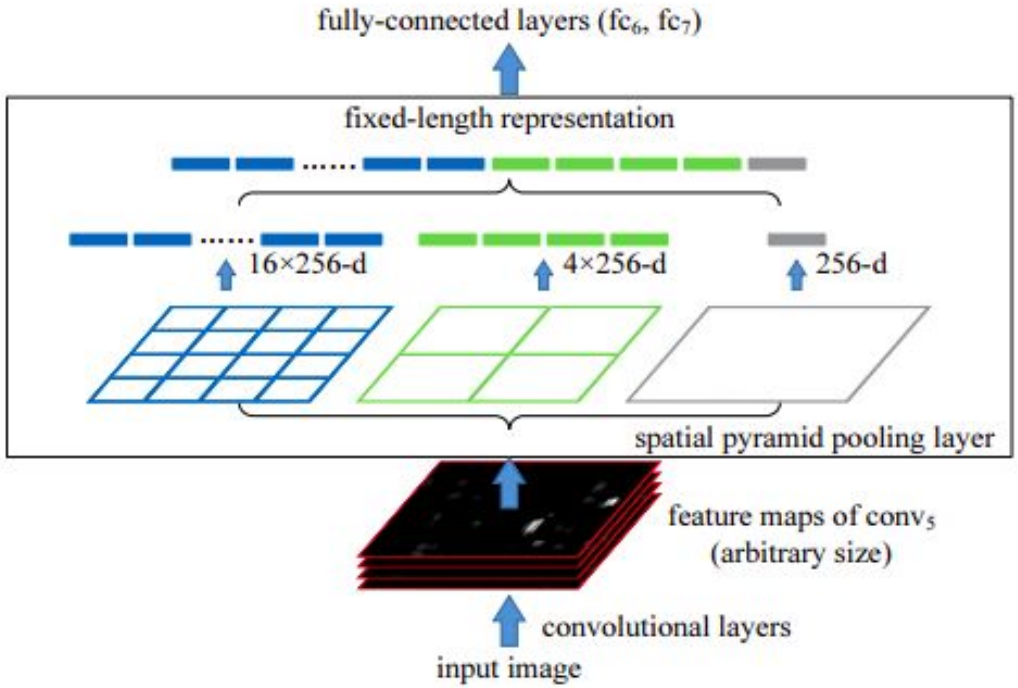


ResNet Module

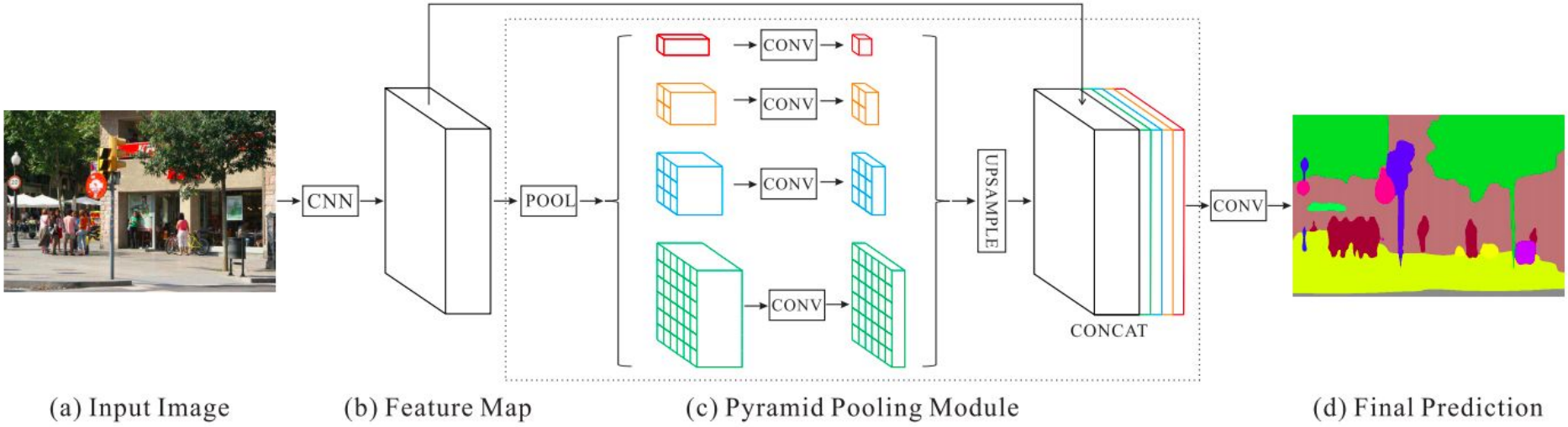


SE-ResNet Module

Spatial Pyramid Pooling



Pyramid Scene Parsing Network



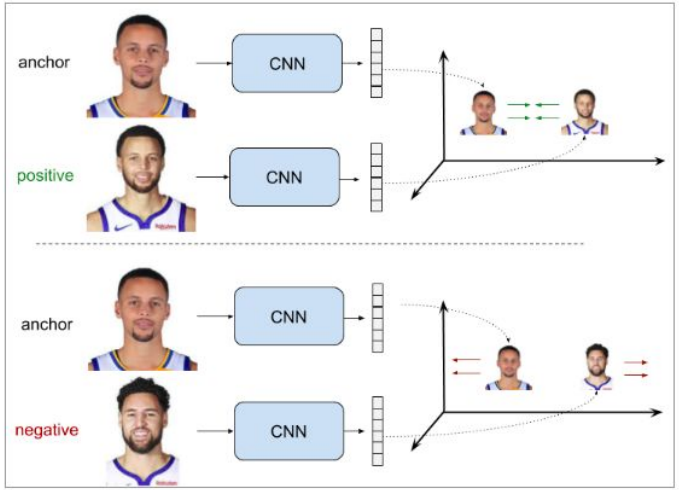
Re-identification

WHO IS

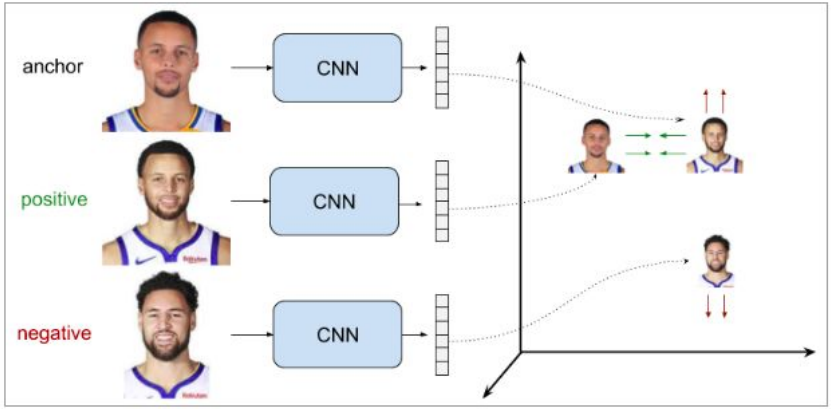
OR

Ranking Loss

Pairwise Ranking Loss



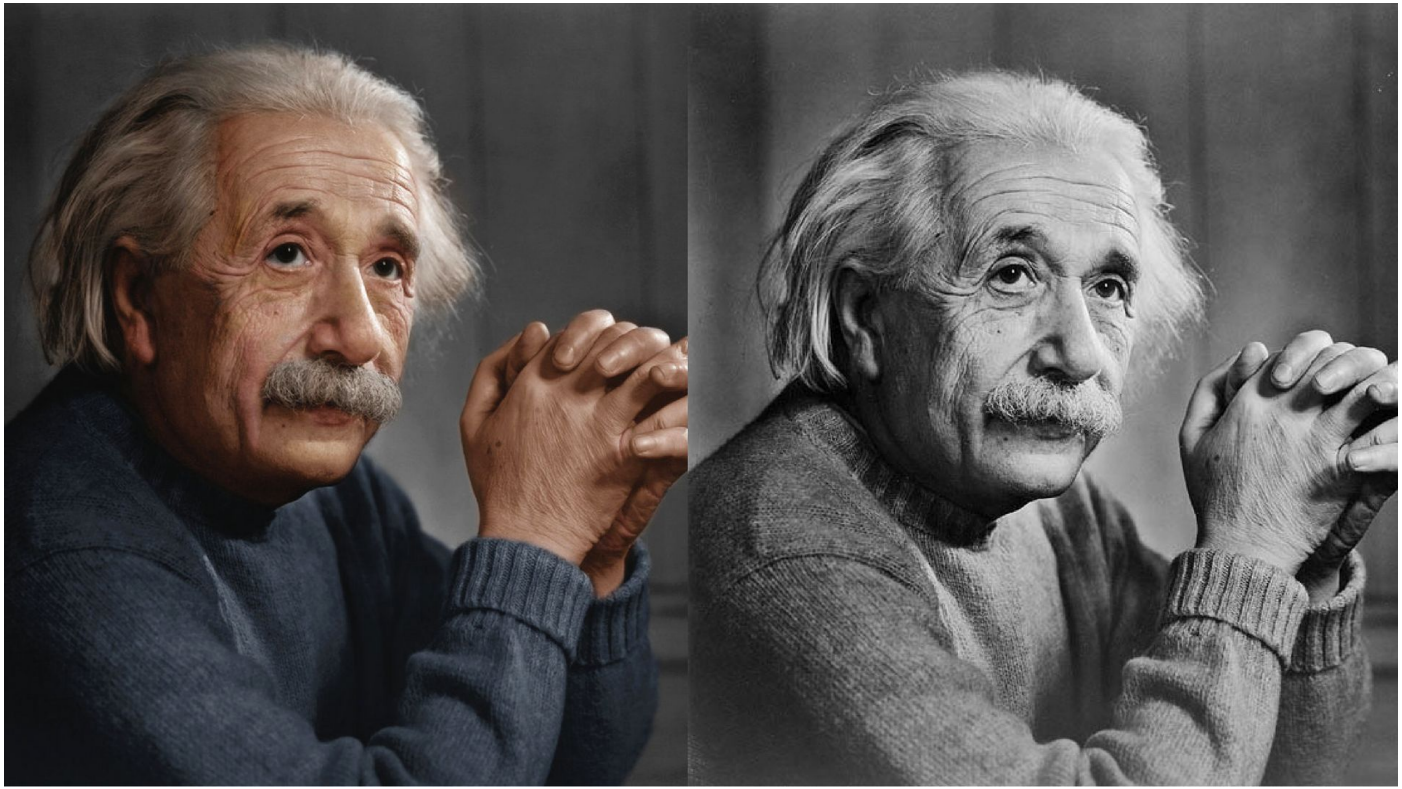
Triplet Ranking Loss



$$L = \begin{cases} d(r_a, r_p) & if \quad PositivePair \\ max(0, m - d(r_a, r_n)) & if \quad NegativePair \end{cases}$$

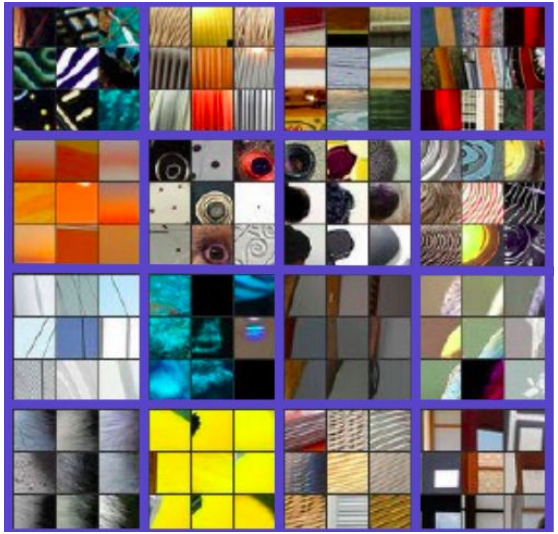
$$L(r_a, r_p, r_n) = max(0, m + d(r_a, r_p) - d(r_a, r_n))$$

Image Colorization



Style Transfer

Layer 1



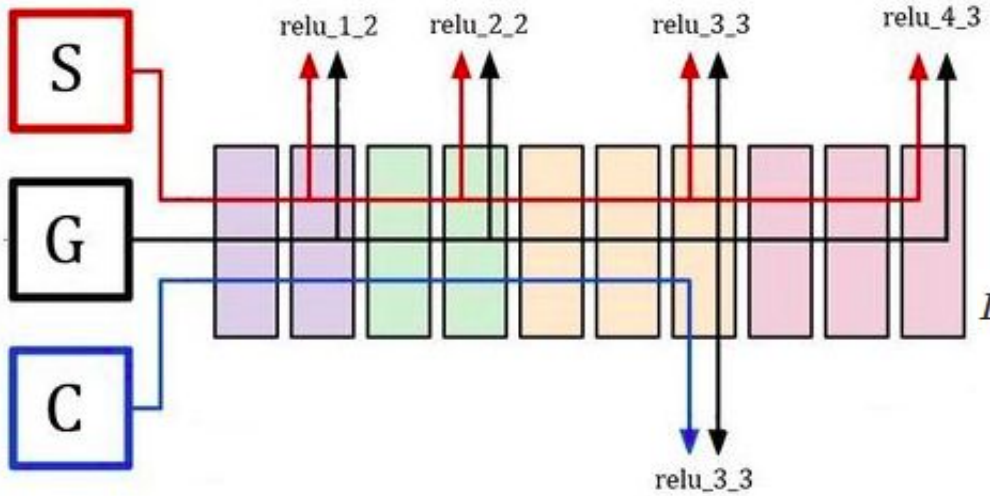
Layer 2

Layer 3

Layer 4

Layer 5

Style Transfer



$$L_{total}(S, C, G) = \alpha * L_{content}(C, G) + \beta * L_{style}(S, G)$$

$$L_{content}(C, G, L) = \frac{1}{2} \sum_{ij} (a[l](C)_{ij} - a[l](G)_{ij})^2$$

$$L_{GM}(S, G, l) = \frac{1}{4N_l^2 M_l^2} \sum_{ij} (GM[l](S)_{ij} - GM[l](G)_{ij})^2$$

$$L_{style}(S, G) = \sum_{l=0}^{L} w_l * L_{GM}(S, G, l)$$

Alpha Matting

$$\mathbf{C}_i = \boldsymbol{\alpha}_i \mathbf{F}_i + (1 - \boldsymbol{\alpha}_i)\mathbf{B}_i$$

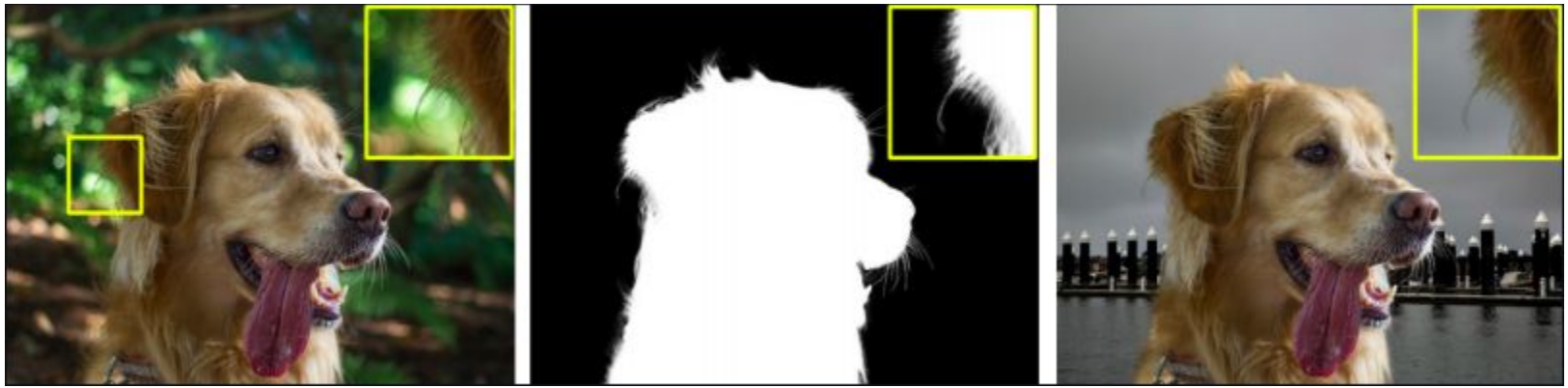


Image Matting Losses

$$\mathcal{L}_1^{\alpha} = \sum_i \|\hat{\boldsymbol{\alpha}}_i - \boldsymbol{\alpha}_i\|_1$$

$$\mathcal{L}_c^{\alpha} = \sum_i \|\mathbf{C}_i - \hat{\boldsymbol{\alpha}}_i \mathbf{F}_i - (1 - \hat{\boldsymbol{\alpha}}_i)\mathbf{B}_i\|_1$$

$$\mathcal{L}_{\text{lap}}^{\alpha} = \sum_{s=1}^{5} 2^{s-1} \left\| L_{\text{pyr}}^s(\boldsymbol{\alpha}) - L_{\text{pyr}}^s(\hat{\boldsymbol{\alpha}}) \right\|_1$$

$$\mathcal{L}_g^{\alpha} = \sum_i \|\nabla\hat{\boldsymbol{\alpha}}_i - \nabla\boldsymbol{\alpha}_i\|_1$$

$$\mathcal{L}_1^{\text{FB}} = \sum_i \left\|\hat{\mathbf{F}}_i - \mathbf{F}_i\right\|_1 + \left\|\hat{\mathbf{B}}_i - \mathbf{B}_i\right\|_1$$

$$\mathcal{L}_{\text{excl}}^{\text{FB}} = \sum_i \|\nabla\mathbf{F}_i\|_1 \|\nabla\mathbf{B}_i\|_1$$

$$\mathcal{L}_c^{\text{FB}} = \sum_i \left\|\mathbf{C}_i - \boldsymbol{\alpha}_i\hat{\mathbf{F}} - (1 - \boldsymbol{\alpha}_i)\hat{\mathbf{B}})\right\|_1$$

$$\mathcal{L}_{\text{lap}}^{\text{FB}} = \mathcal{L}_{\text{lap}}^{\mathbf{F}} + \mathcal{L}_{\text{lap}}^{\mathbf{B}}$$

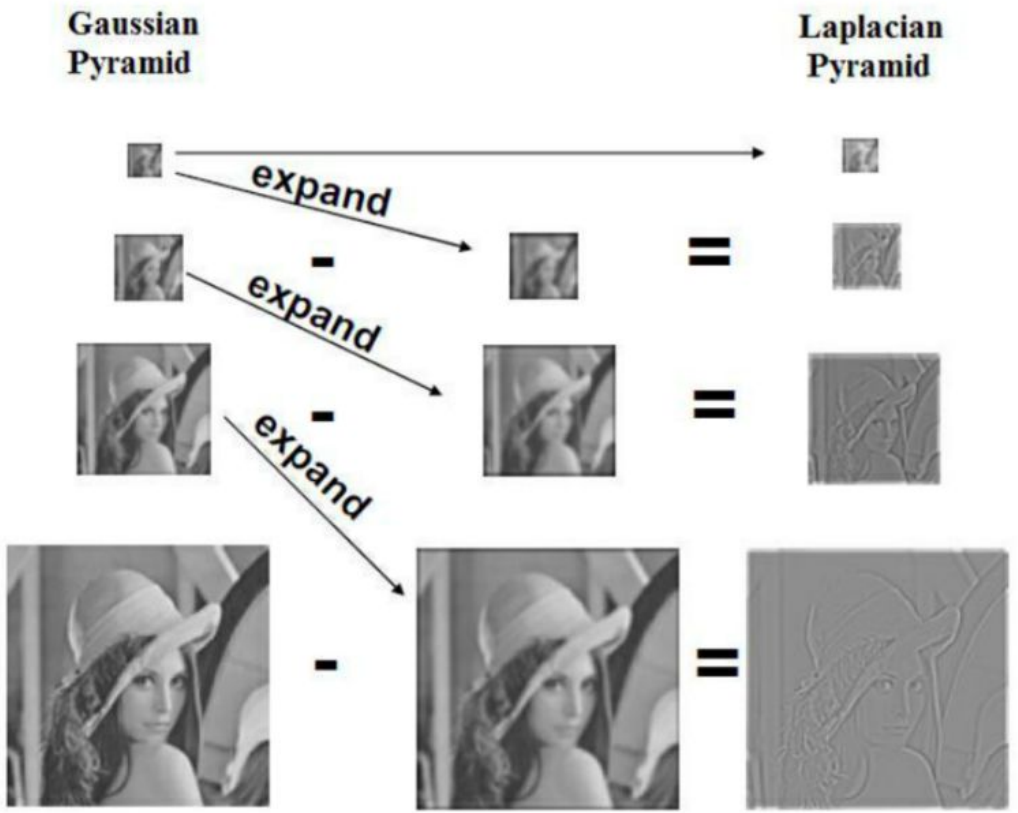
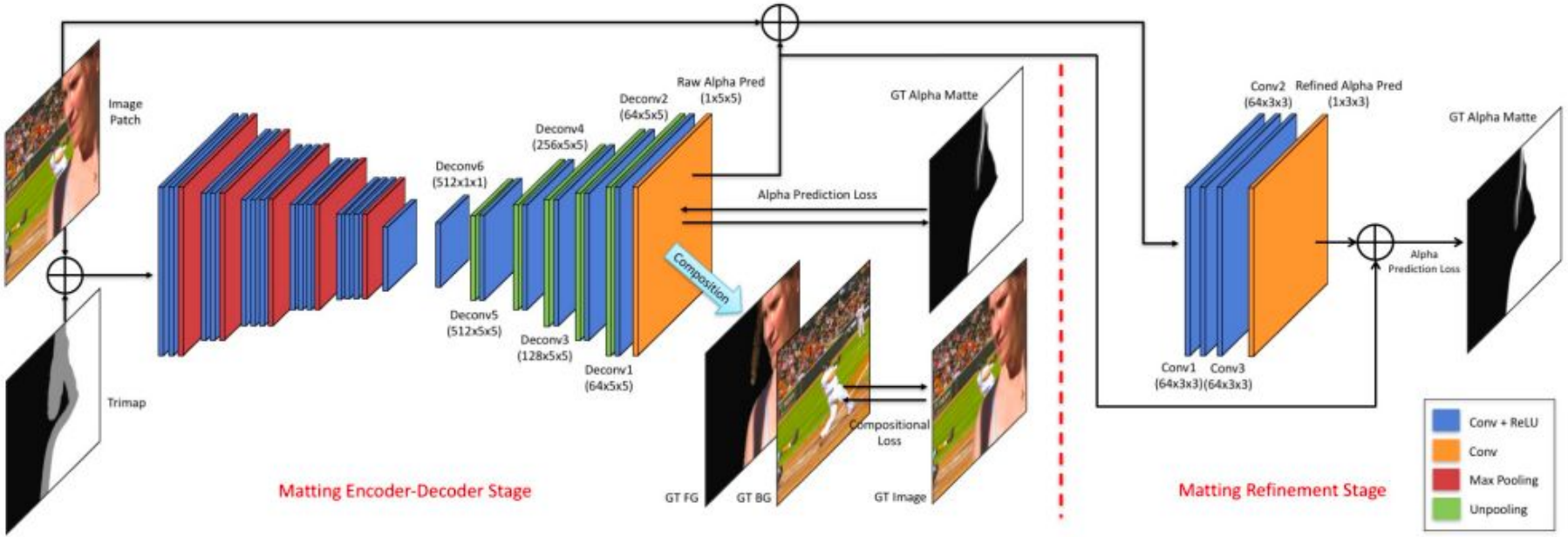


Image Matting



Automatic Image Matting

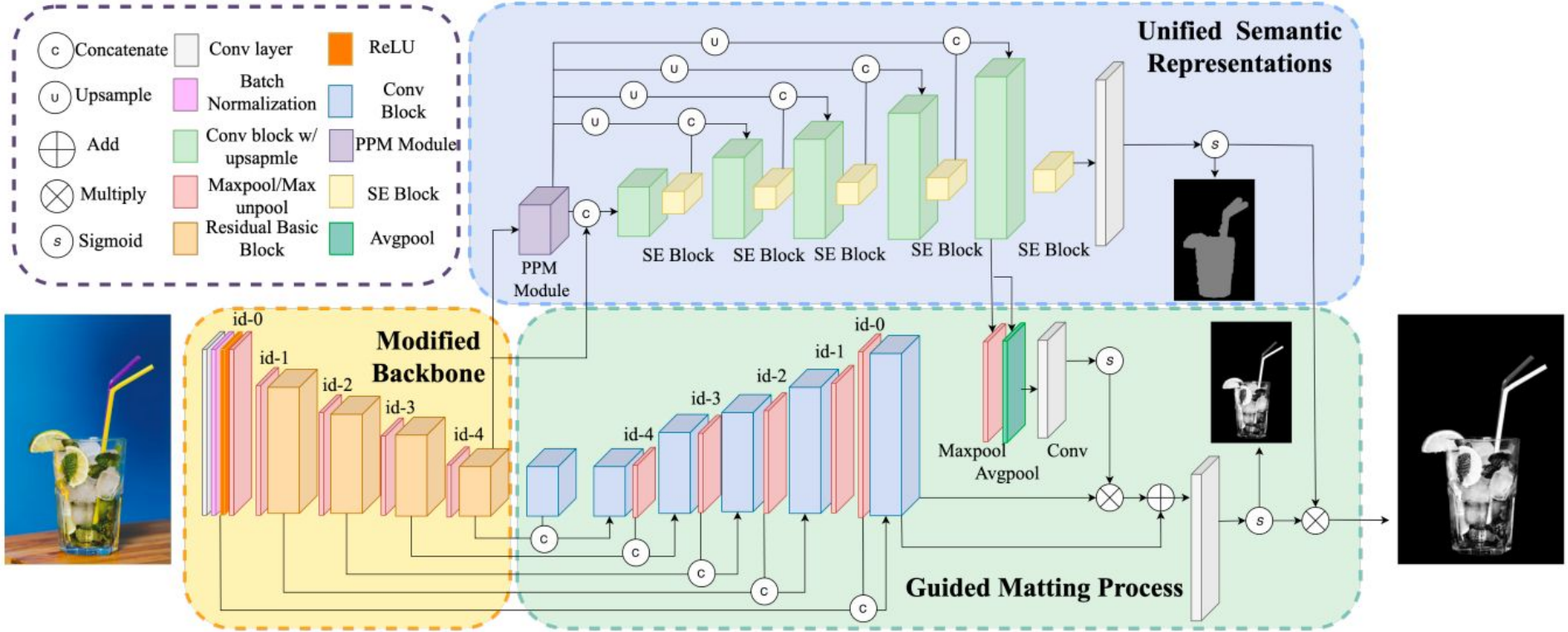


Figure 3: The structure of our matting network for AIM.

Использованная литература

- https://arxiv.org/abs/2010.12496
- https://arxiv.org/pdf/2009.08449.pdf
- https://arxiv.org/abs/1608.06993
- https://arxiv.org/abs/1506.02640
- https://www.reg.ru/blog/svyortka-v-deep-learning-prostymi-slovami/
- https://habr.com/ru/post/244541/
- http://weitz.de/sift/
- https://towardsdatascience.com/sift-scale-invariant-feature-transform-c7233dc60f37
- https://habr.com/ru/post/201406/
- https://habr.com/ru/post/133826/
- https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c
- https://arxiv.org/pdf/2006.14822.pdf
- https://arxiv.org/abs/1505.04597
- https://www.jeremyjordan.me/semantic-segmentation/#upsampling
- https://arxiv.org/pdf/1409.4842.pdf
- https://habr.com/ru/post/347564/
- https://arxiv.org/abs/1706.03059
- https://arxiv.org/abs/1709.01507
- https://distill.pub/2016/deconv-checkerboard/
- https://arxiv.org/abs/1406.4729
- https://arxiv.org/abs/1612.01105
- https://arxiv.org/abs/1703.03872v3
- https://paperswithcode.com/paper/f-b-alpha-matting
- https://paperswithcode.com/method/laplacian-pyramid
- https://www.getklap.com/blog/what-is-reidentification
- https://arxiv.org/abs/1311.2901
- https://neerc.ifmo.ru/wiki/index.php?title=Neural_Style_Transfer