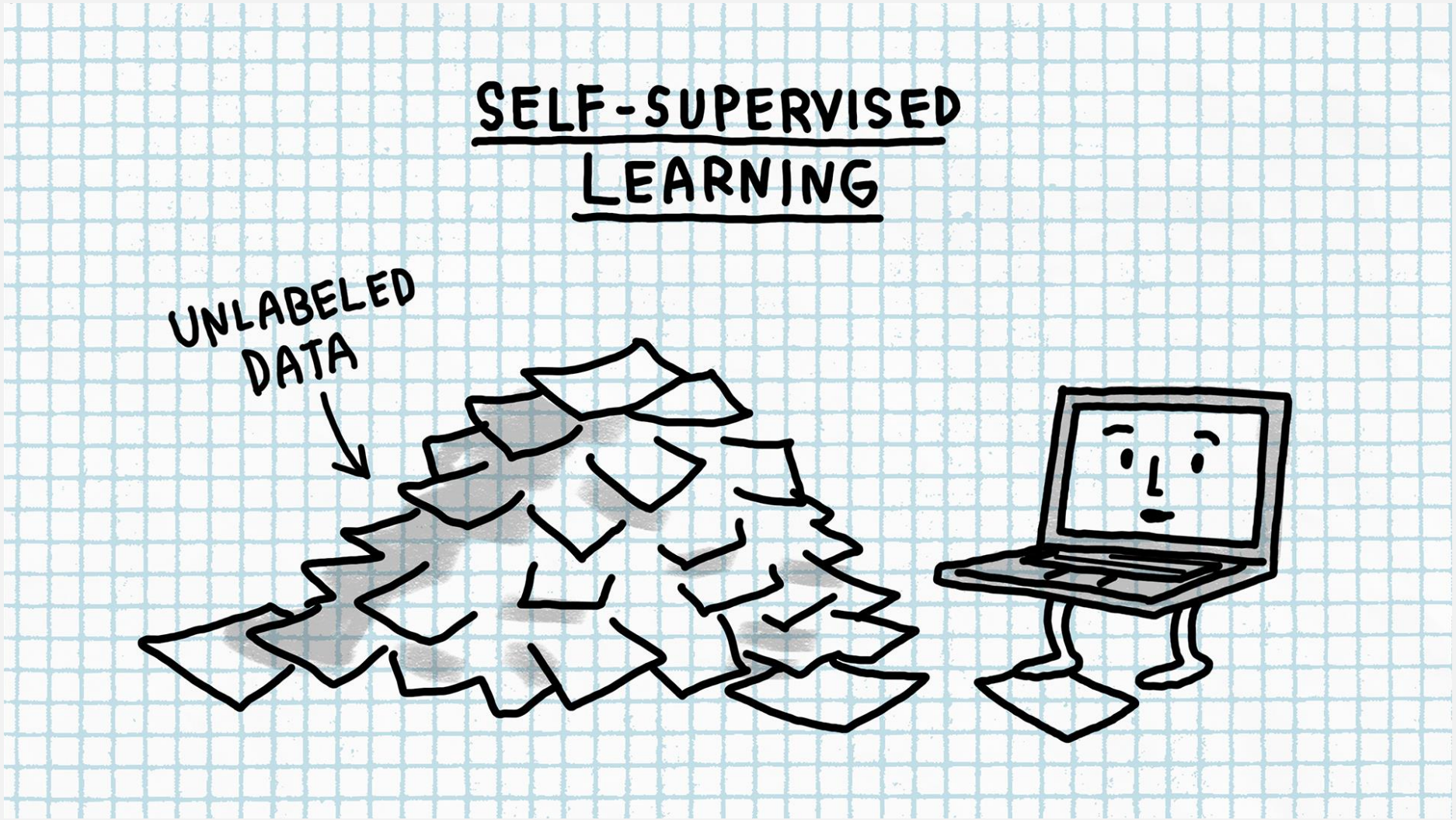
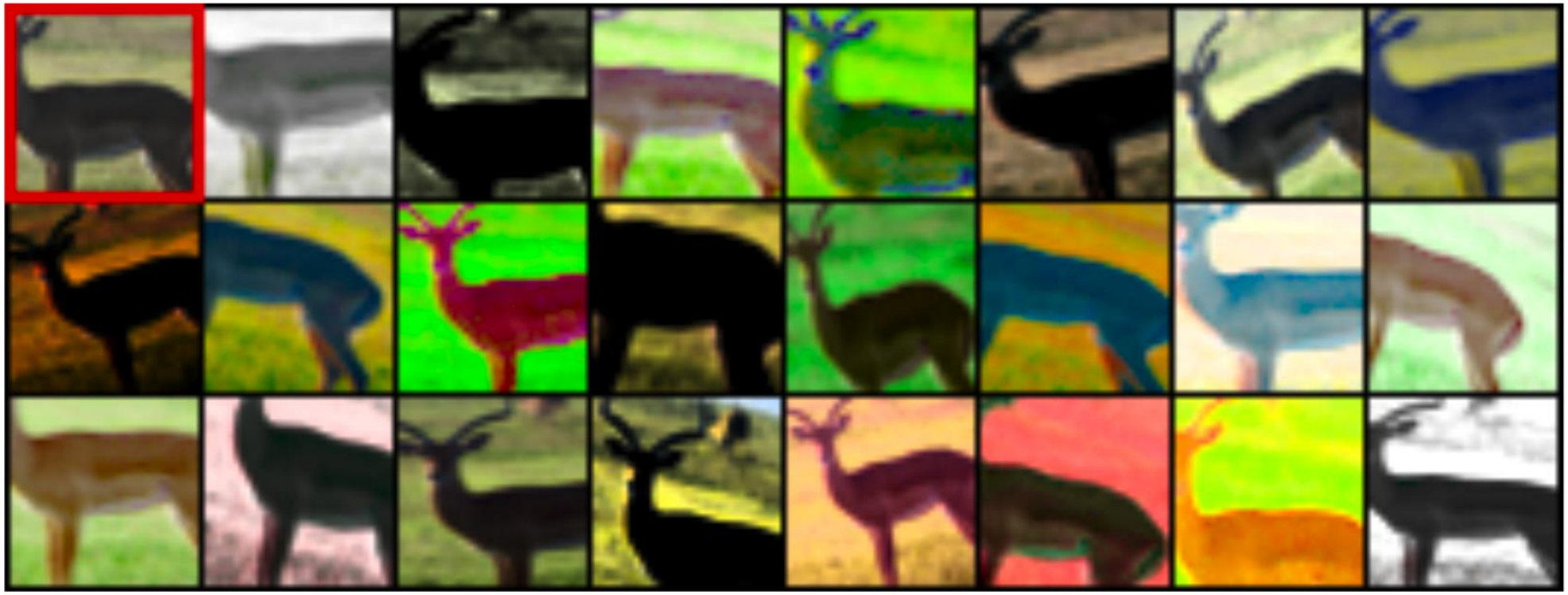
SELF-SUPERVISED LEARNING

Федорова Анна, БПМИ-191

ПРОБЛЕМА РАЗМЕТКИ ДАННЫХ



АУГМЕНТАЦИИ



ПОВОРОТЫ

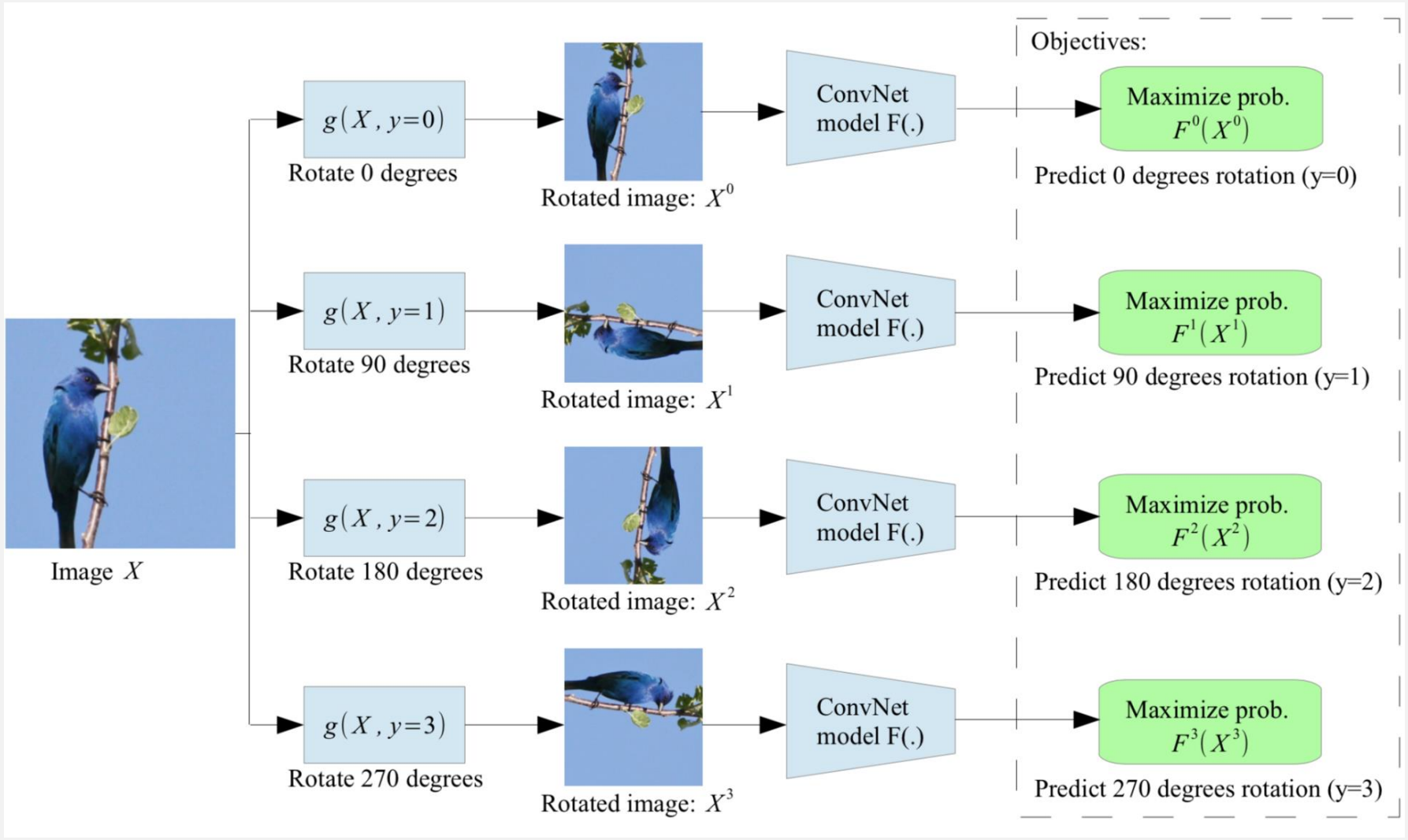
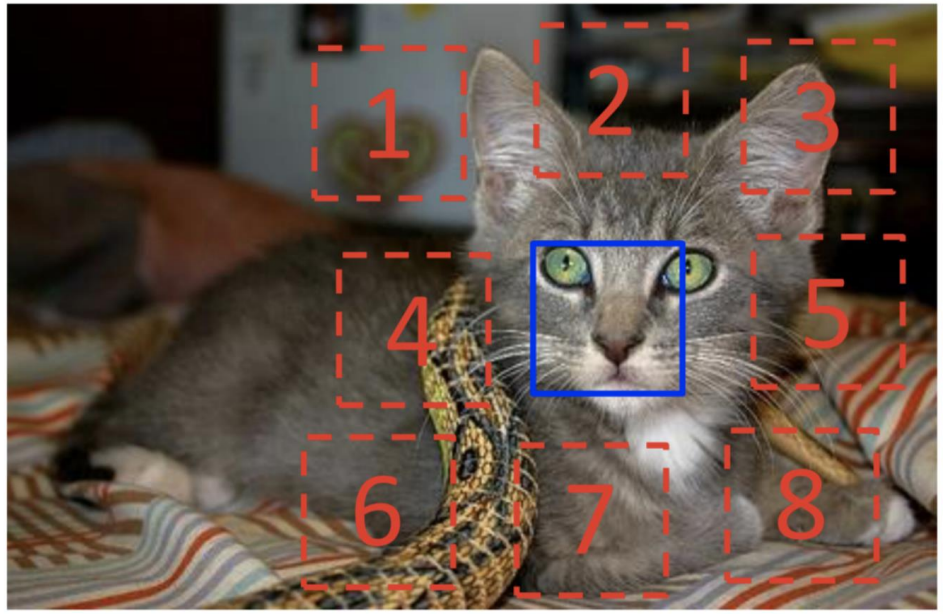


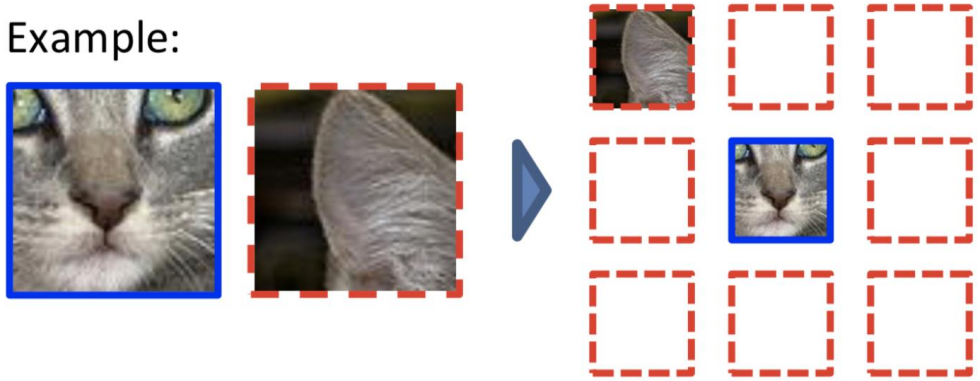
Image X
$g(X, y=0)$
Rotate 0 degrees
Rotated image: X^0
$g(X, y=1)$
Rotate 90 degrees
Rotated image: X^1
$g(X, y=2)$
Rotate 180 degrees
Rotated image: X^2
$g(X, y=3)$
Rotate 270 degrees
Rotated image: X^3
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
ConvNet model F(.)
Objectives:
Maximize prob. $F^0(X^0)$
Predict 0 degrees rotation (y=0)
Maximize prob. $F^1(X^1)$
Predict 90 degrees rotation (y=1)
Maximize prob. $F^2(X^2)$
Predict 180 degrees rotation (y=2)
Maximize prob. $F^3(X^3)$
Predict 270 degrees rotation (y=3)

ВЗАИМОРАСПОЛОЖЕНИЕ ФРАГМЕНТОВ



X = (,); Y = 3

Example:



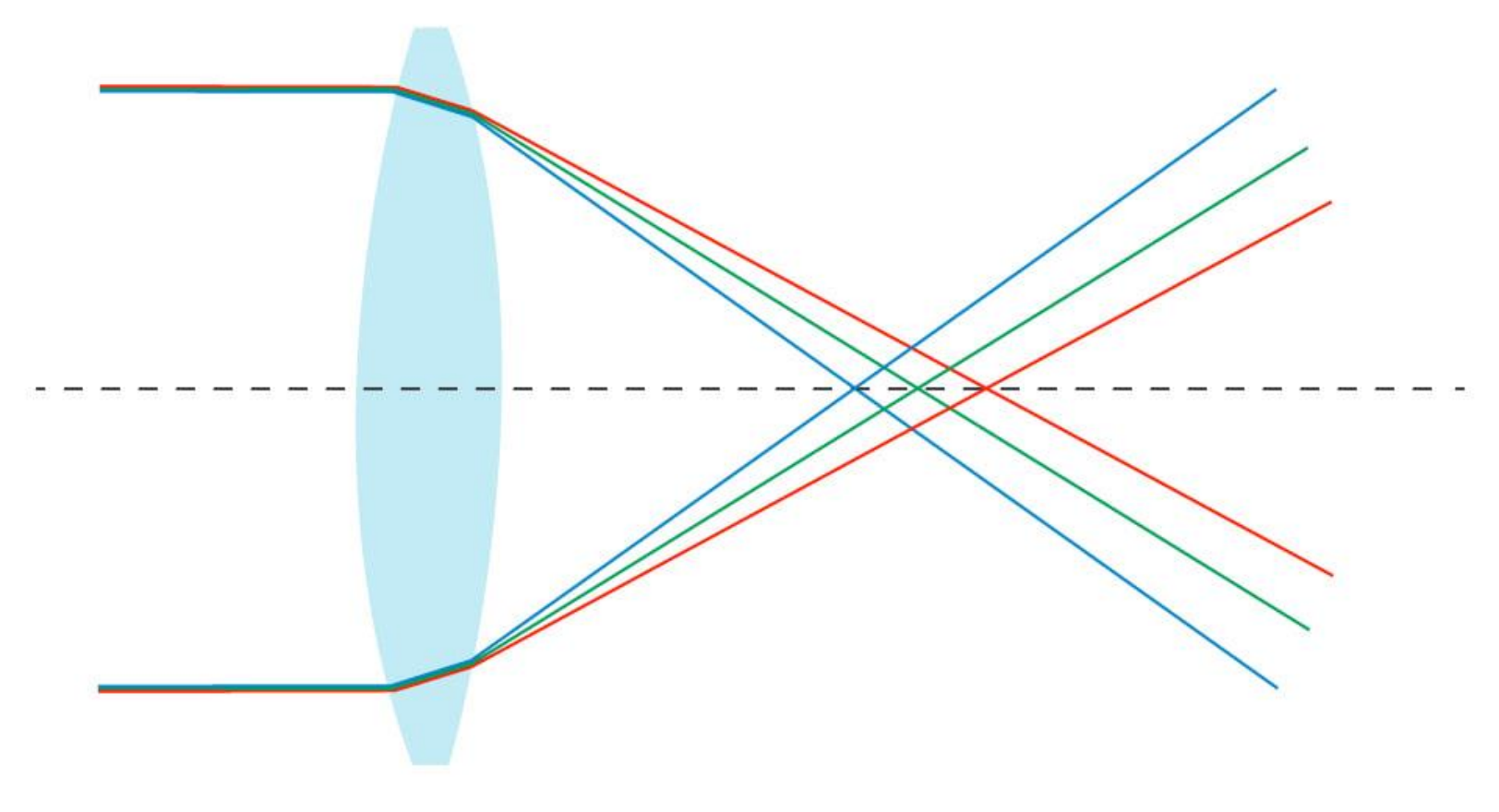
Question 1:

?

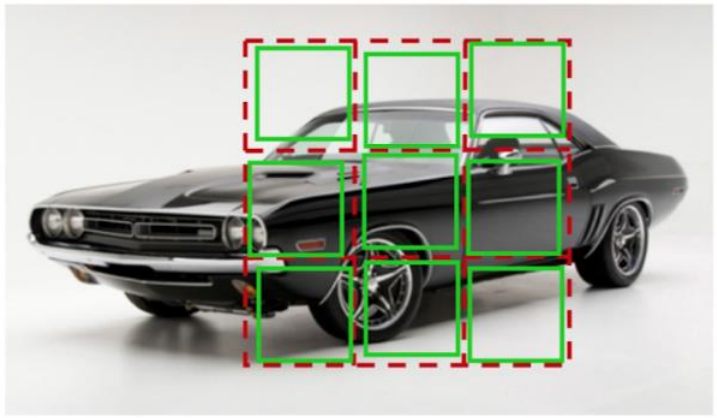
Question 2:

?

ПОДВОХ ЦВЕТОВЫХ КАНАЛОВ



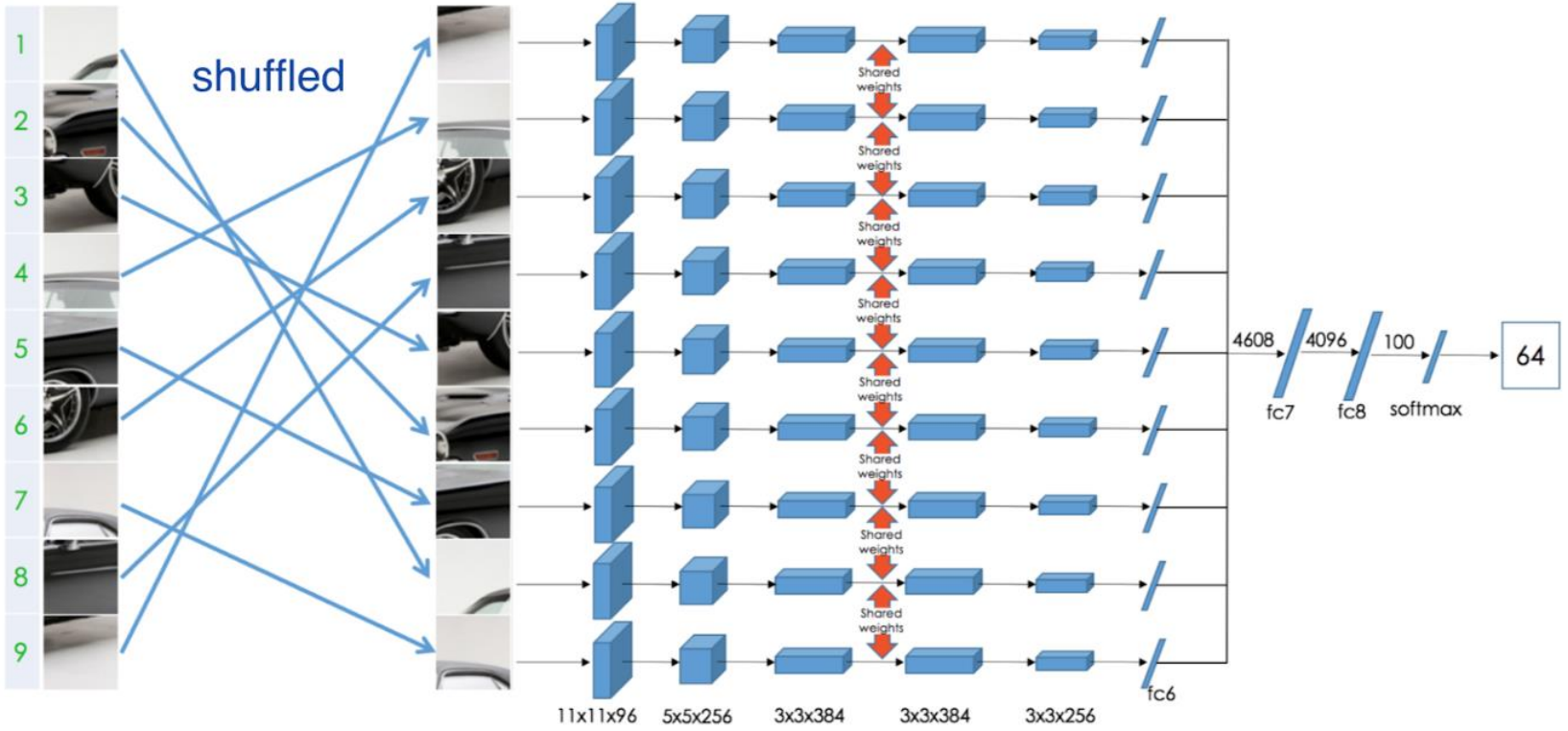
ПАЗЛЫ



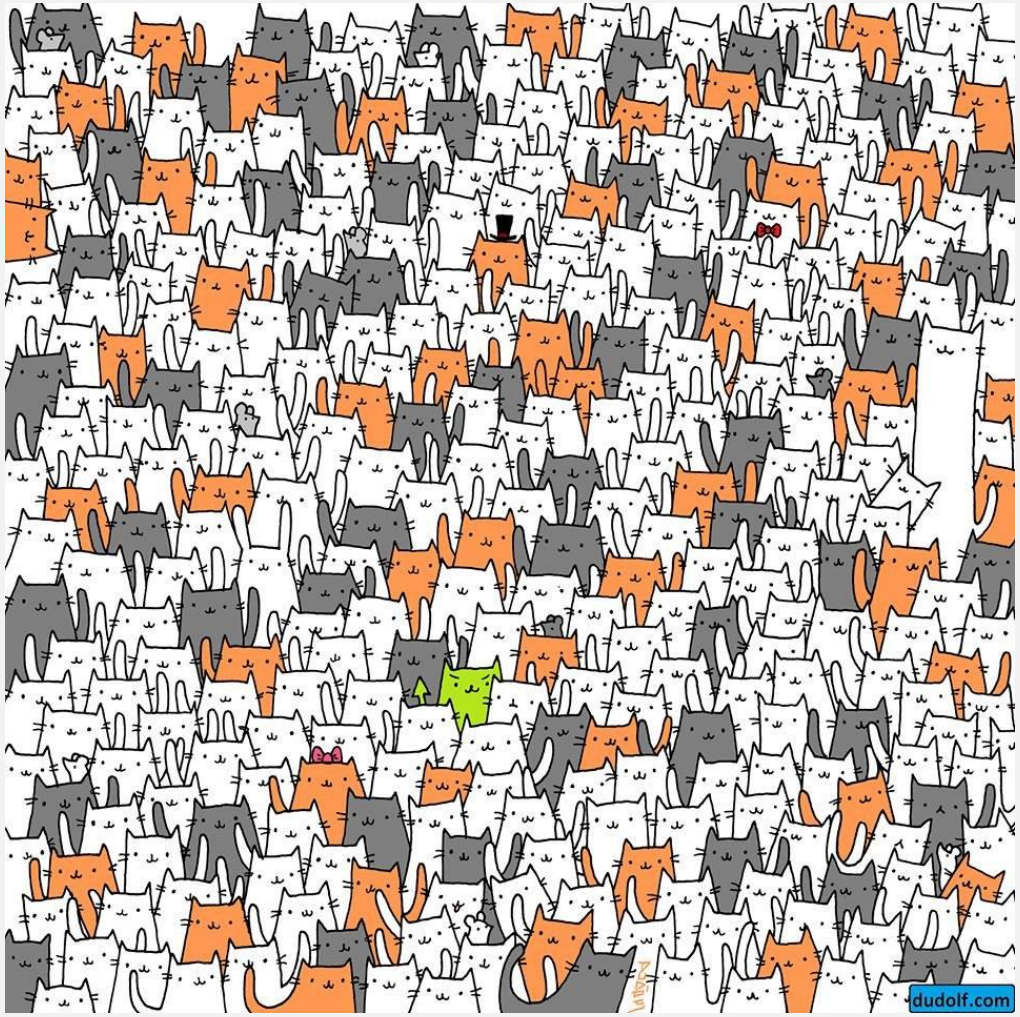
Permutation Set

index	permutation
64	9,4,6,8,3,2,5,1,7

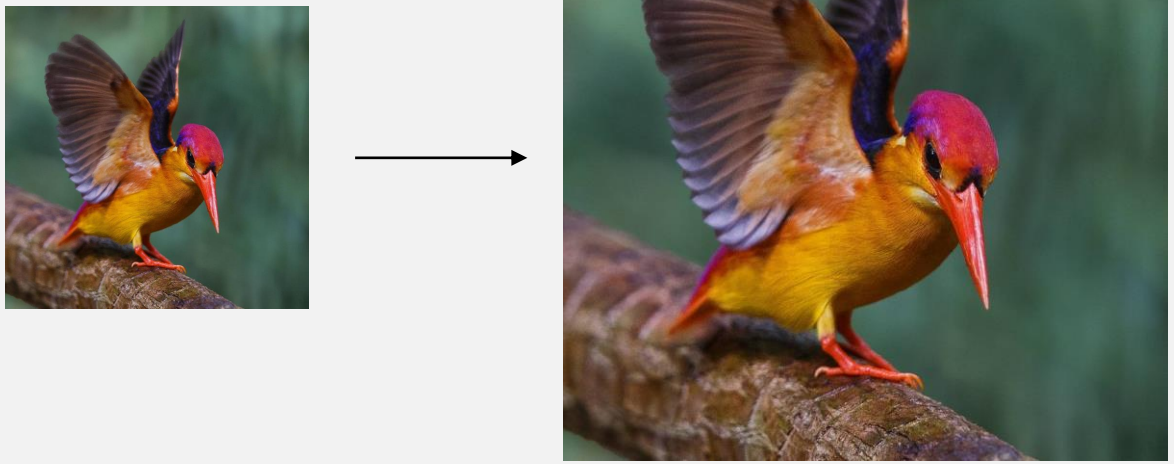
Reorder patches according to the selected permutation



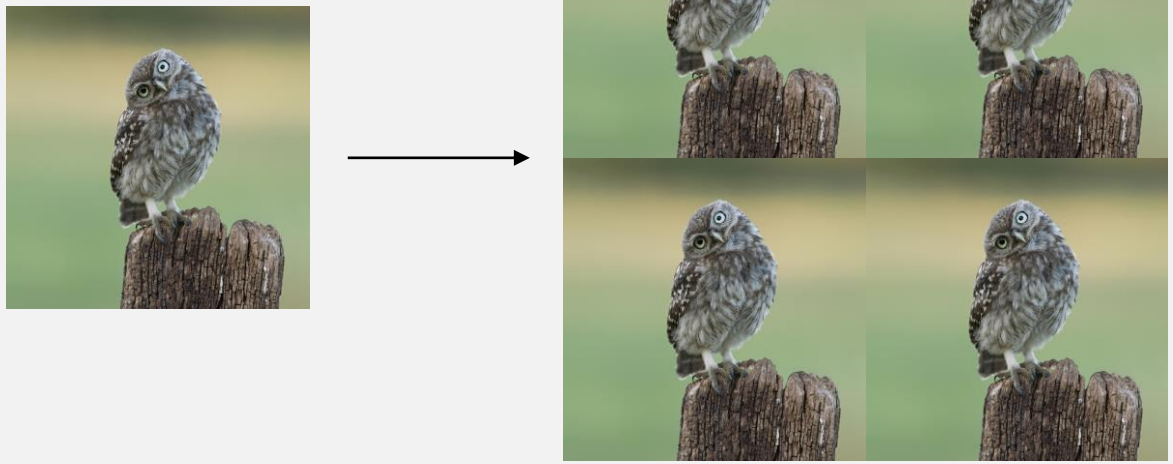
ПОДСЧЕТ ВИЗУАЛЬНЫХ ПРИМИТИВОВ



ПРЕОБРАЗОВАНИЕ ИЗОБРАЖЕНИЙ



Масштабирование



Плитка 2x2

ФОРМУЛЫ ДЛЯ ОБУЧЕНИЯ

$$\phi(x) = \phi(D \circ x) = \sum_{i=1}^{4} \phi(T_i \circ x)$$

Функция, которую ищет модель

D – обратно масштабирует изображение
T_i – возвращает i-ый элемент плитки 2×2

Функции, обращающие наши преобразования

ФУНКЦИЯ ОШИБКИ

$$L_{feat} = \|\phi(D \circ x) - \sum_{i=1}^{4} \phi(T_i \circ x)\|_2^2$$

Функция ошибки

$$L_{diff} = \max(0, c - \|\phi(D \circ y) - \sum_{i=1}^{4} \phi(T_i \circ x)\|_2^2)$$

Исключаем тождественный ноль

$$L = L_{feat} + L_{diff} = \|\phi(D \circ x) - \sum_{i=1}^{4} \phi(T_i \circ x)\|_2^2 + \max(0, c - \|\phi(D \circ y) - \sum_{i=1}^{4} \phi(T_i \circ x)\|_2^2)$$

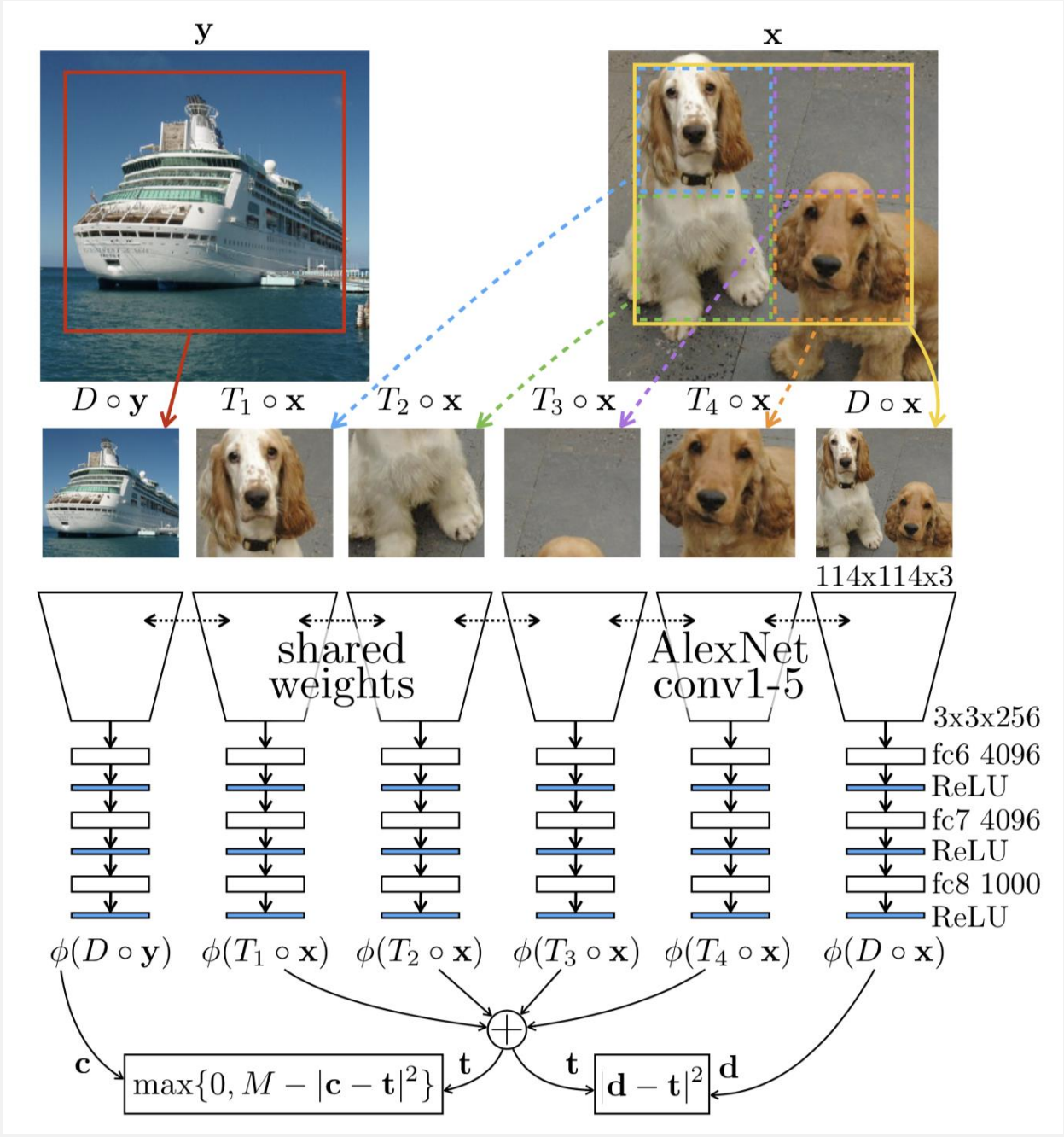
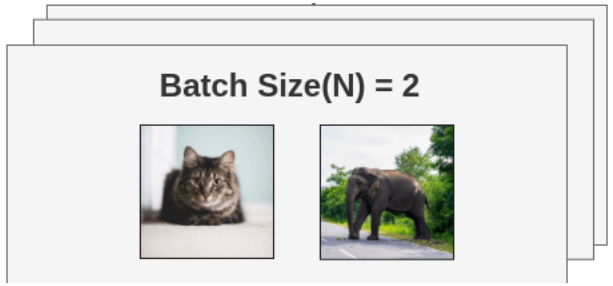


СХЕМА РАБОТЫ НЕЙРОННОЙ СЕТИ

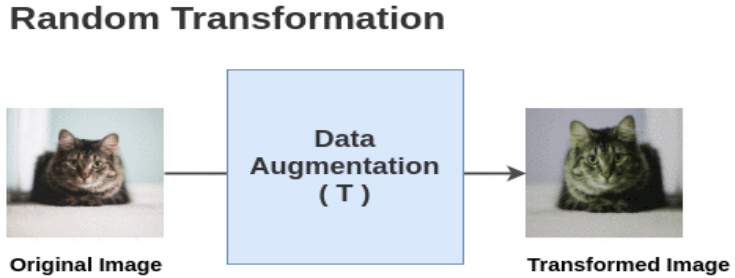
СПАСИБО ЗА ВНИМАНИЕ

Шаг 0.1) Нам нужны примеры пар изображений, которые похожи, и которые отличаются

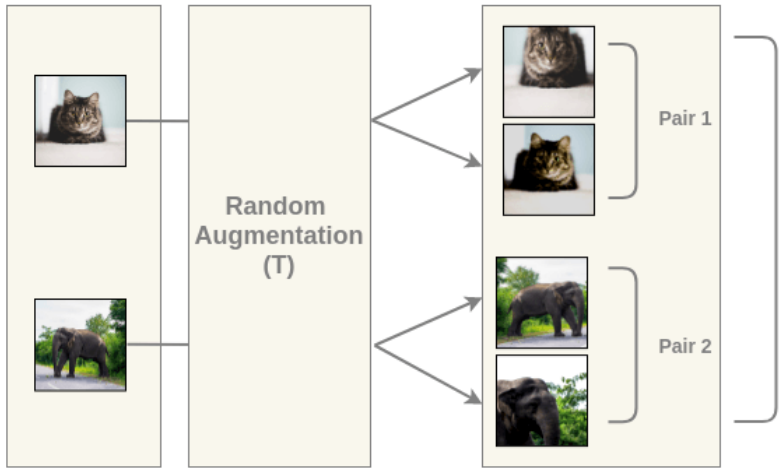
Шаг 0.2) Делим весь массив фотографий на батчи



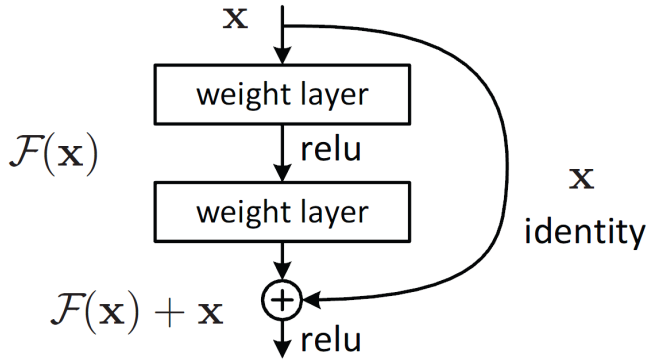
Шаг 1.1) Трансформируем изображения random (crop + flip + color jitter + grayscale) – обрезаем (берём фрагмент), поворачиваем, приводим к серому, смещения зеленого и пурпурного в сторону серого, случайное удаление 2 из 3 цветовых каналов



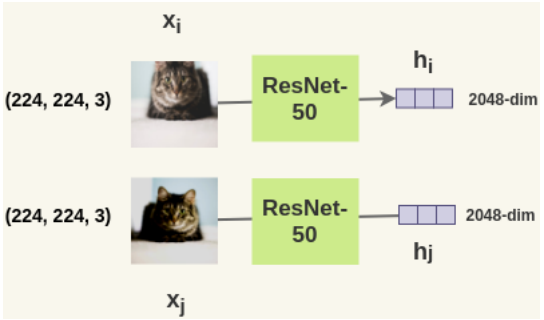
Шаг 1.2) Заменяем исходные изображения в батче их трансформированными копиями



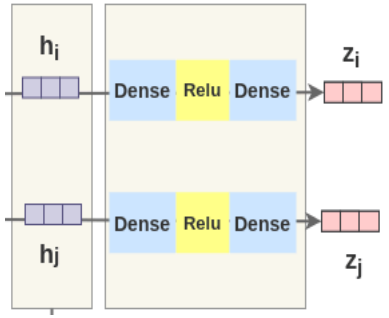
Что такое ResNet или «остаточная сеть», объяснять не буду, вот красивая картинка:



Шаг 2) «Сжимаем» изображение, например с помощью сети ResNet-50 (можно использовать и другие)

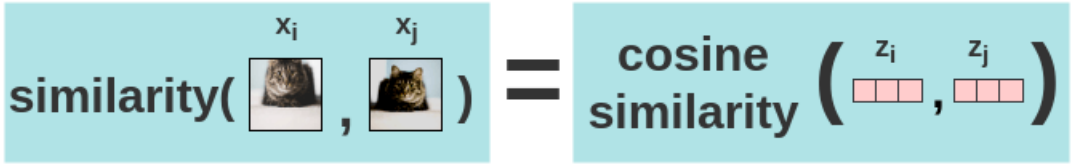


f



g

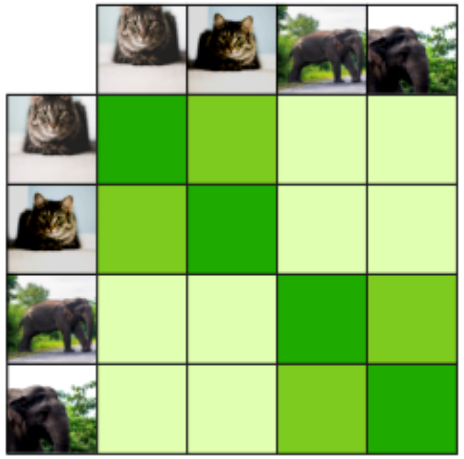
Запомните



Шаг 3.0) Считаем Cosine Similarity

$$s_{i,j} = \frac{z_i^T z_j}{(\tau ||z_i|| ||z_j||)}$$

=>



Напоминание:

similarity(x_i , x_j) = cosine similarity (z_i , z_j)

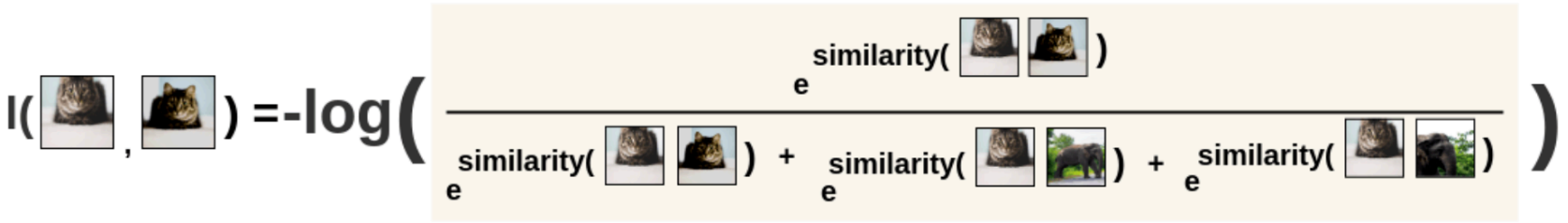
Шаг 3.1) Для каждой (!) пары изображений считаем Softmax

Pair 1

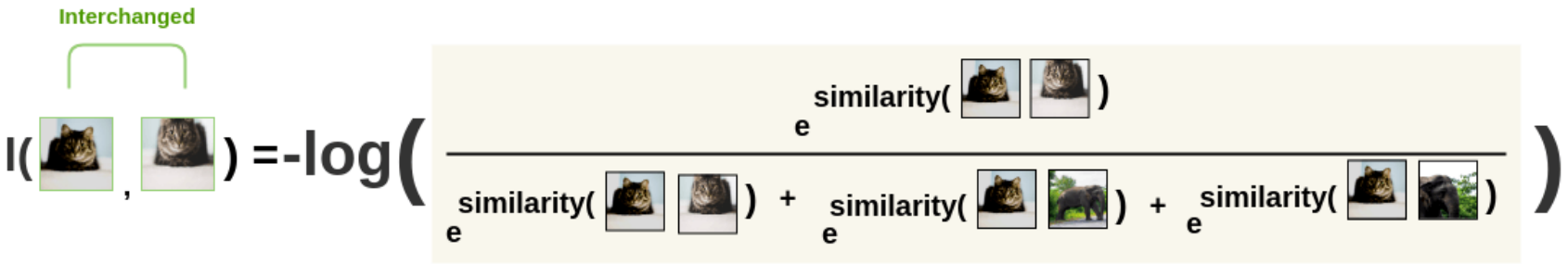
$$\text{Softmax} = \frac{e^{\text{similarity}(\,\cdot\,,\,\cdot\,)}}{e^{\text{similarity}(\,\cdot\,,\,\cdot\,)} + e^{\text{similarity}(\,\cdot\,,\,\cdot\,)} + e^{\text{similarity}(\,\cdot\,,\,\cdot\,)}}$$

Шаг 4.1) Посчитаем I (image1, image2) по формуле ниже

$$l(i,j) = -log\frac{exp(s_{i,j})}{\sum_{k=1}^{2N} l_{[k!=i]} exp(s_{i,k})}$$

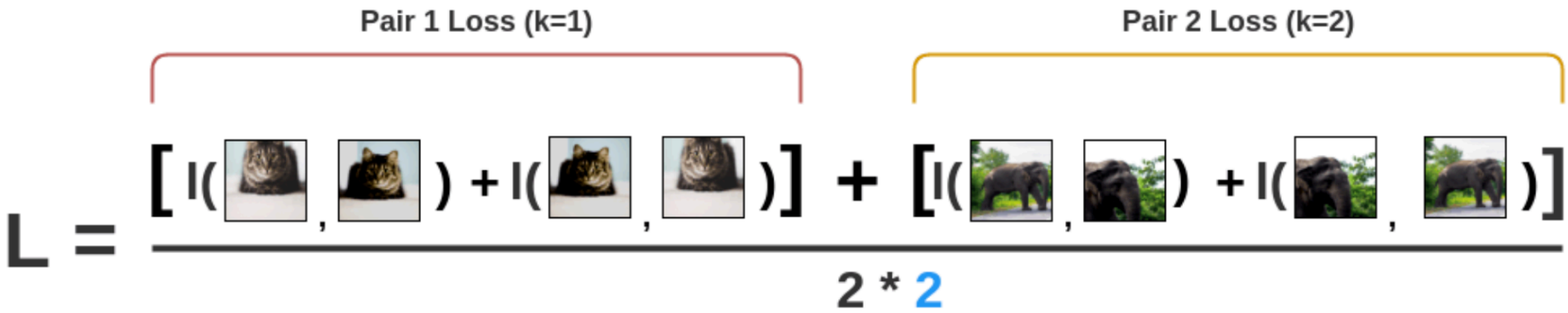


Шаг 4.2) Также посчитаем I (image2, image1)



Шаг 4.3) Теперь можем посчитать L для нашего батча

$$L = \frac{1}{2N}\sum_{k=1}^{N}[l(2k-1, 2k) + l(2k, 2k-1)]$$



Шаг 4.4) Обновляем параметры сети и возвращаемся к шагу 1.1 с новым батчем

P.S. Сам алгоритм (из статьи)

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of f, g, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

P.S.S. Сравнение с другими методами (также из статьи)