

Глубинное обучение

Лекция 8

Перенос обучения. Распознавание лиц.

Михаил Гущин

mhushchyn@hse.ru

НИУ ВШЭ, 2024



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

В предыдущей серии

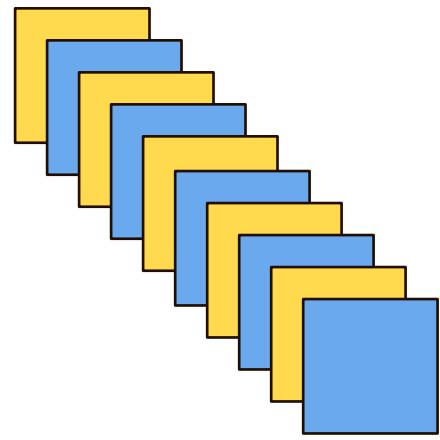
ImageNet



Подробнее: <https://image-net.org/challenges/LSVRC>

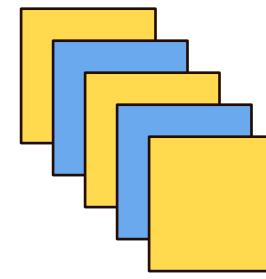
- ▶ ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- ▶ 1000 классов изображений
- ▶ Более 1 000 000 изображений

Свертка 1×1



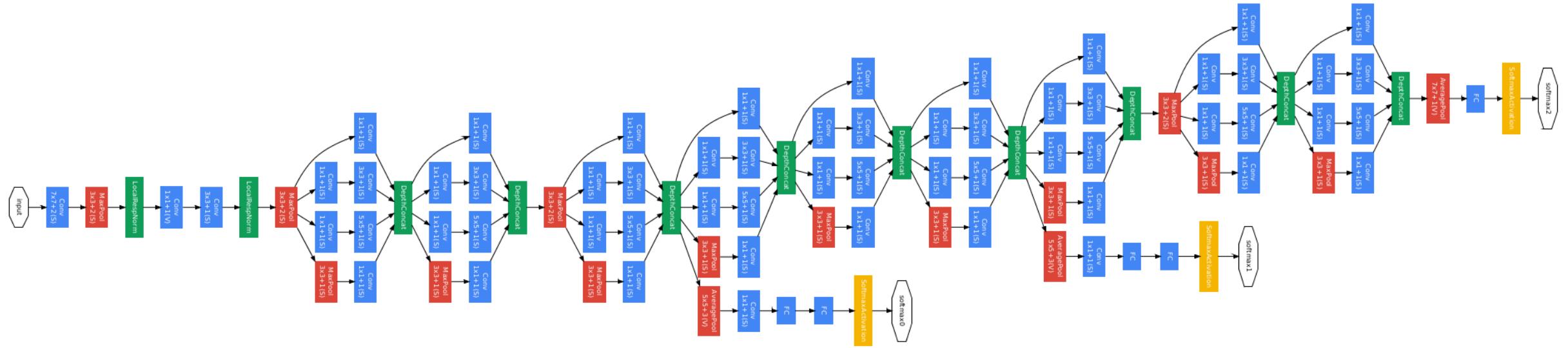
Входное
изображение
 $(28 \times 28 \times 10)$

1×1 свертка
с **5** фильтрами

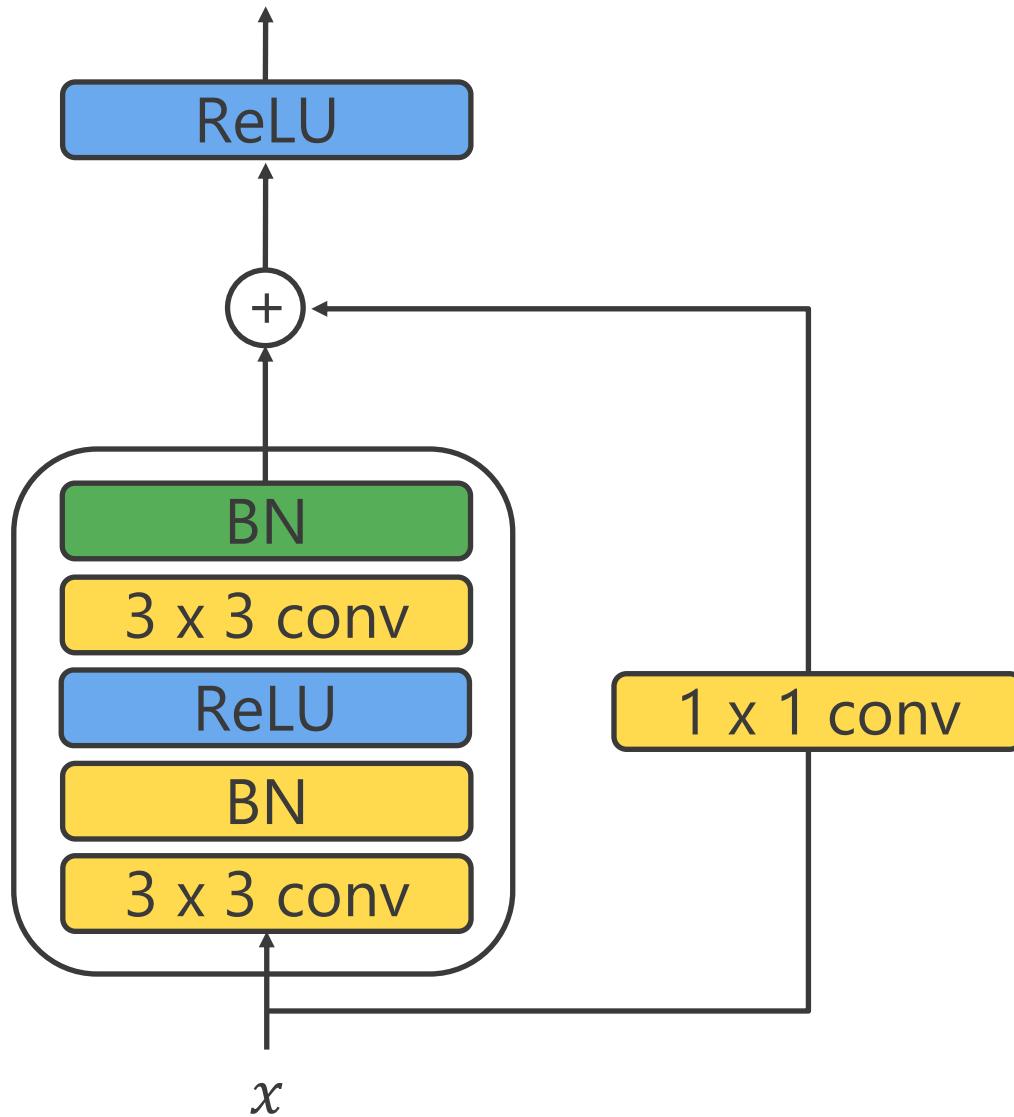


Выходное
изображение
 $(28 \times 28 \times 5)$

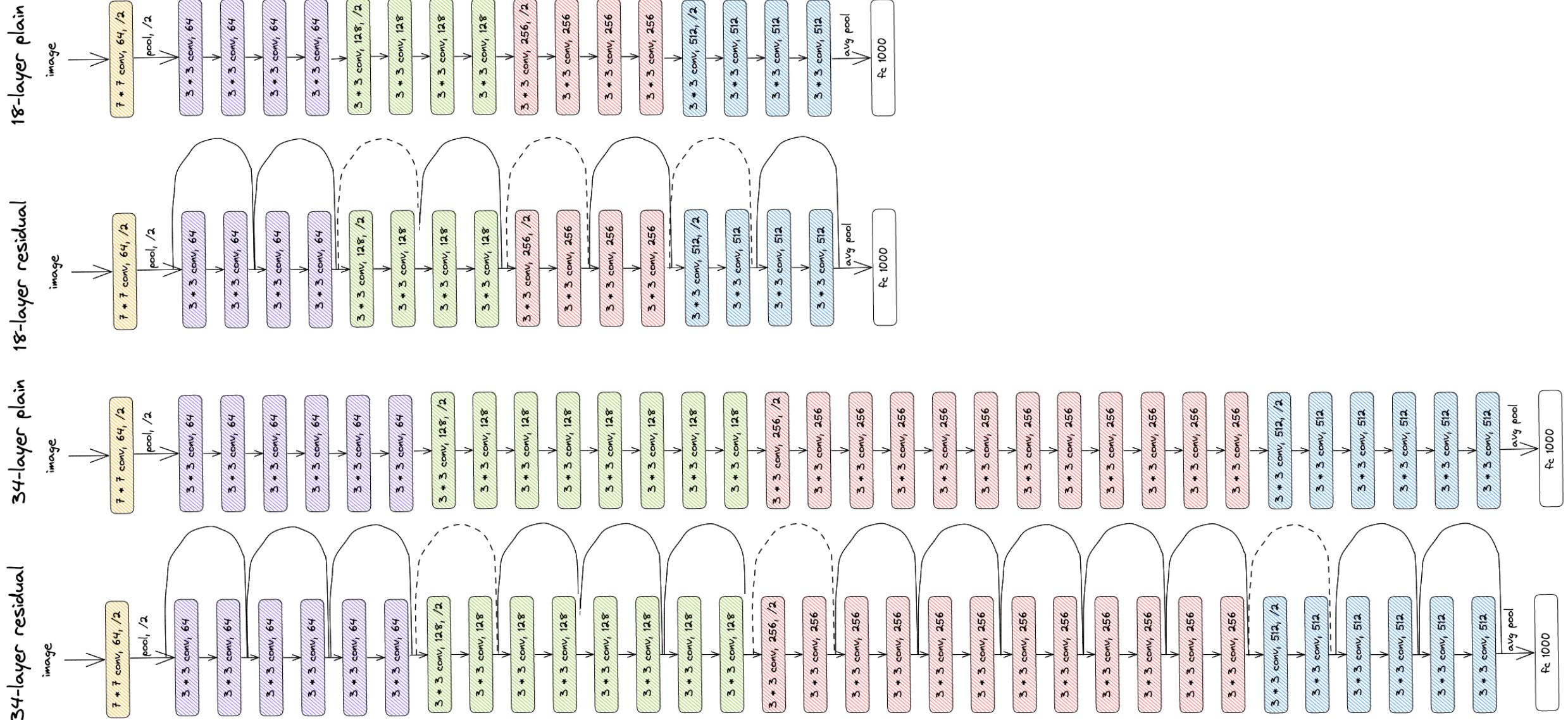
GoogLeNet



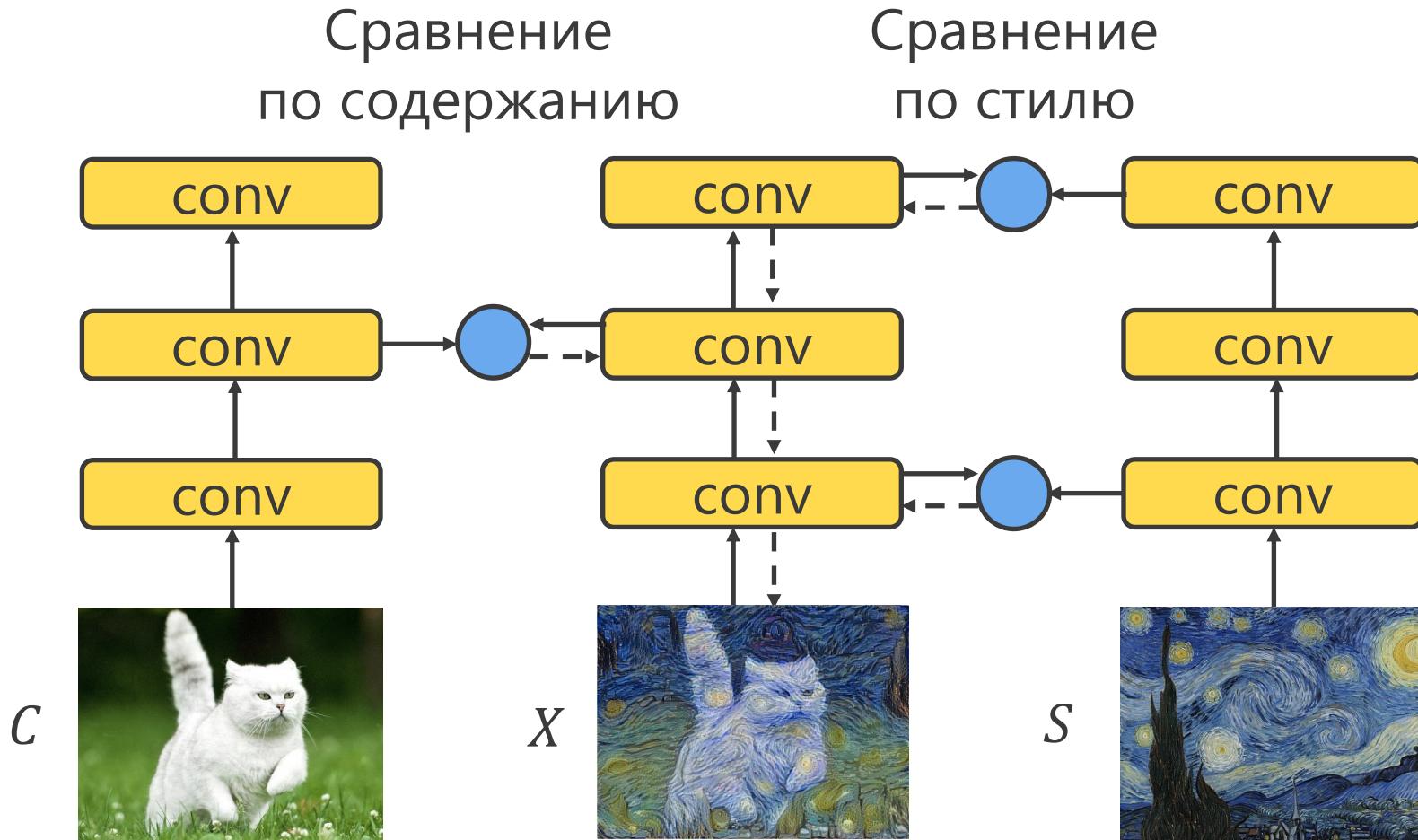
Остаточный (residual) блок



ResNet 18 и 34



Перенос стиля



EfficientNet

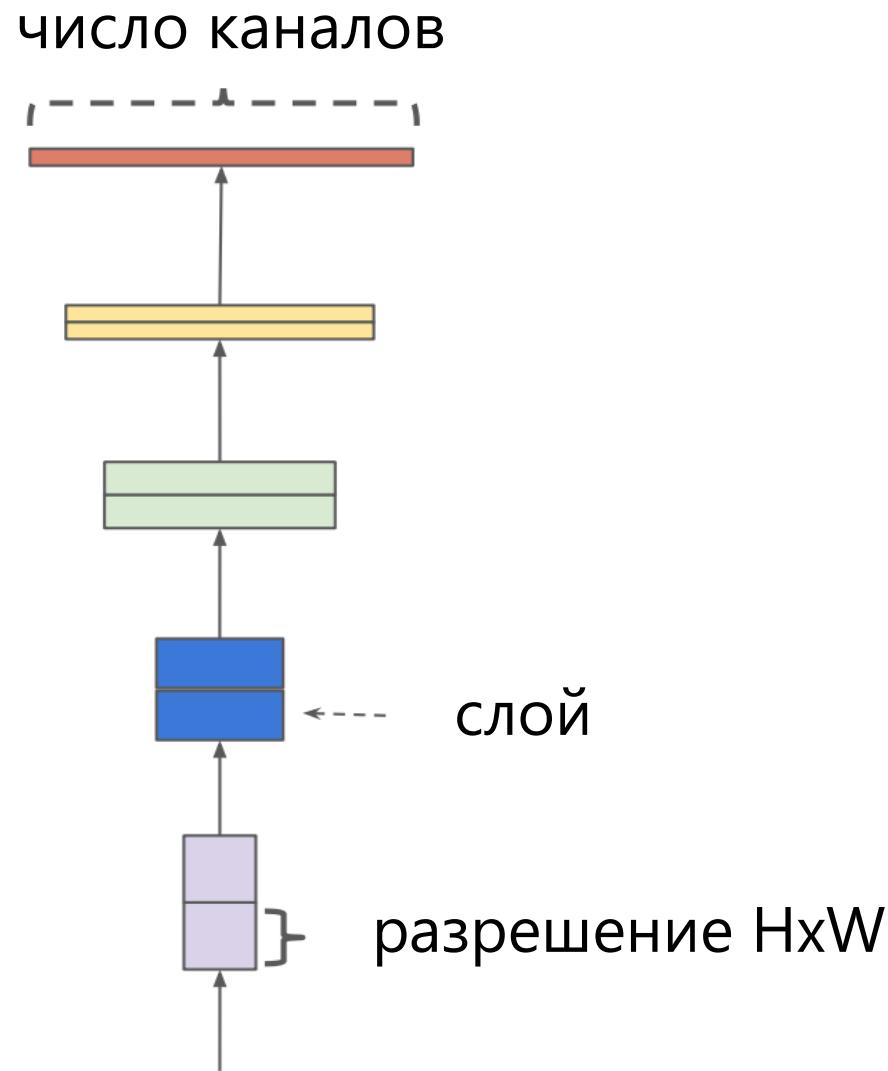
EfficientNet

- ▶ Как подобрать число слоев сети?
- ▶ Как выбрать количество каналов в каждом слое?
- ▶ Изображения с каким разрешением более оптимально?

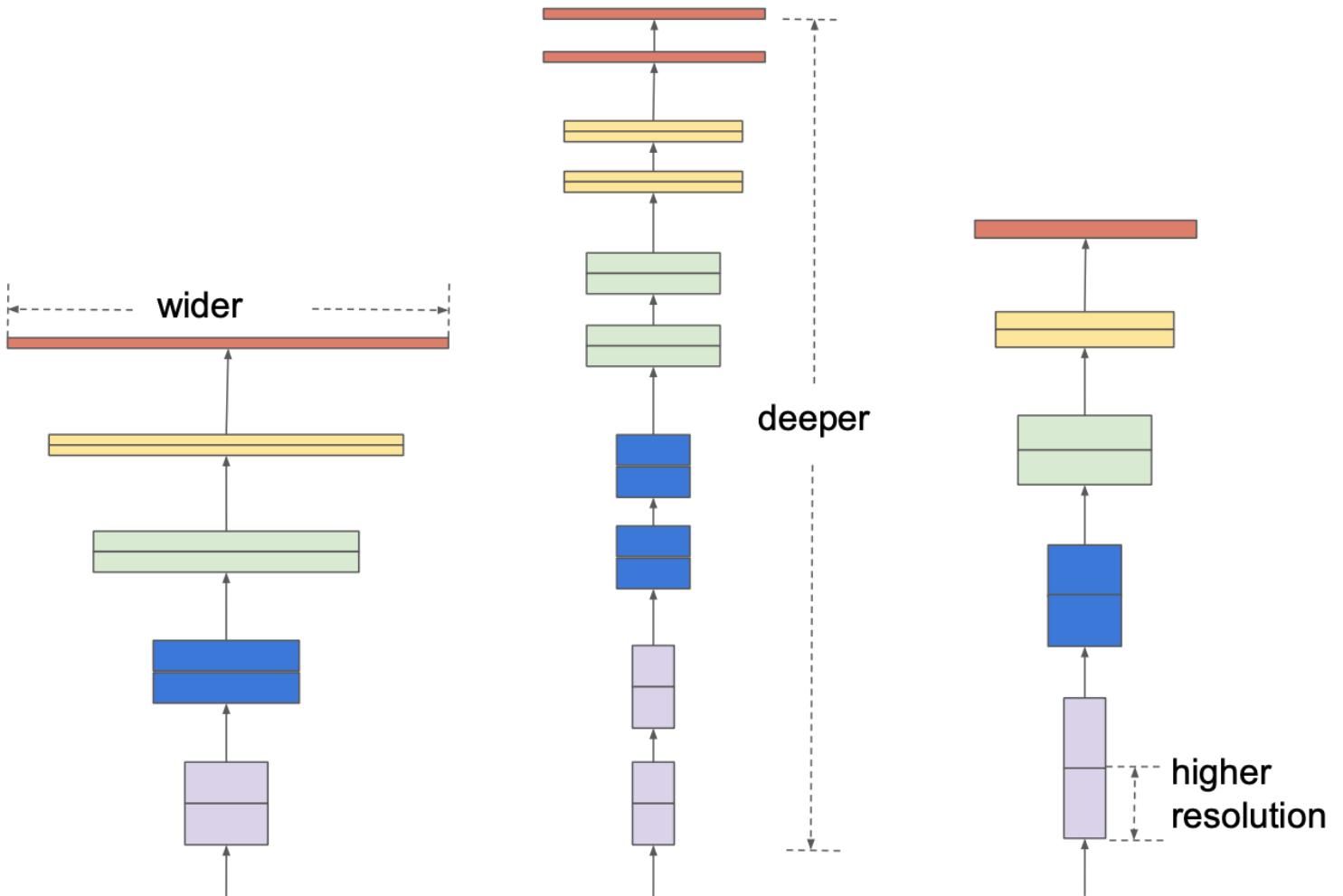
EfficientNet

- ▶ Оптимизирует архитектуру сверточной сети
- ▶ Максимизирует качество классификации
- ▶ Учитывает ограничения на объем вычислений

Базовая архитектура сети



Оптимизация архитектуры



EfficientNet

- ▶ Берем базовую сверточную сеть
- ▶ Масштабируем число каналов / глубину сети / размер изображения умножением на $\alpha^\phi, \beta^\phi, \gamma^\phi$
 - ϕ — настраиваемый параметр
 - α, β, γ — константы, которые нужно найти

EfficientNet

- ▶ α, β, γ находим поиском по сетке значений, оптимизируя целевую функцию
- ▶ Целевая функция:

$$Accuracy * \left(\frac{FLOPS}{T} \right)^{-0.07}$$

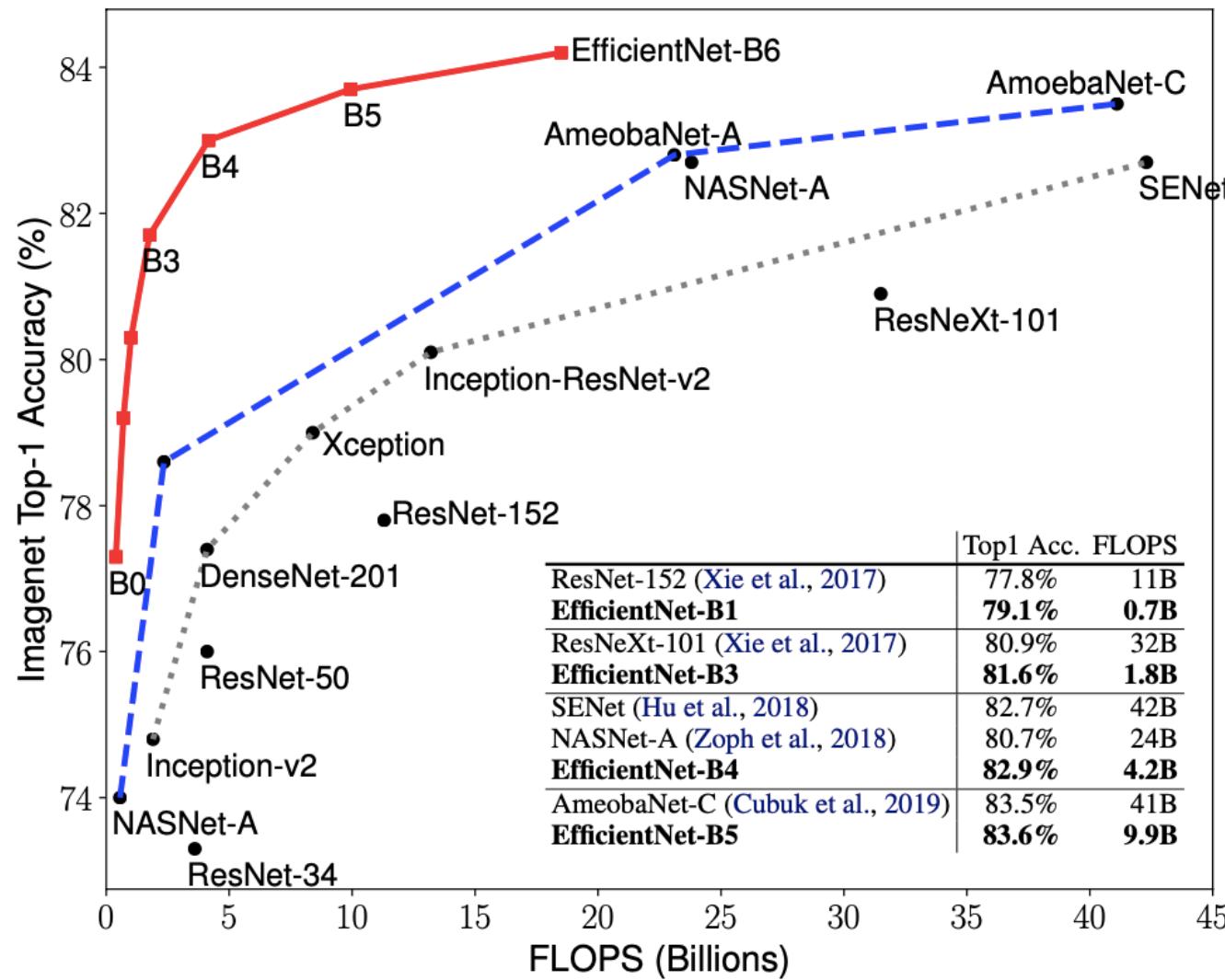
- ▶ T — наше целевое значение числа операций (FLOPS)
- ▶ Ограничение: $\alpha\beta^2\gamma^2 \approx 2$

EfficientNet

Алгоритм:

- ▶ Берем $\phi = 1$
- ▶ Находим α, β, γ поиском по сетке значений (B0)
- ▶ Для разных ϕ , масштабируем число каналов / глубину сети / размер изображения умножением на $\alpha^\phi, \beta^\phi, \gamma^\phi$ (B1-6)

EfficientNet





Перенос обучения (transfer learning)

ImageNet



Подробнее: <https://image-net.org/challenges/LSVRC>

- ▶ ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- ▶ 1000 классов изображений
- ▶ Более 1 000 000 изображений

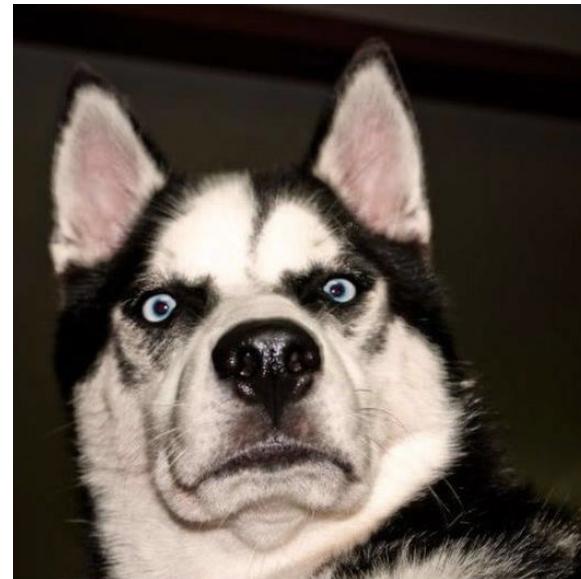
ImageNet

- ▶ Много обученных моделей
- ▶ Тысячи часов на GPU для обучения
- ▶ Можно ли переиспользовать обученные модели в других задачах?

Задача

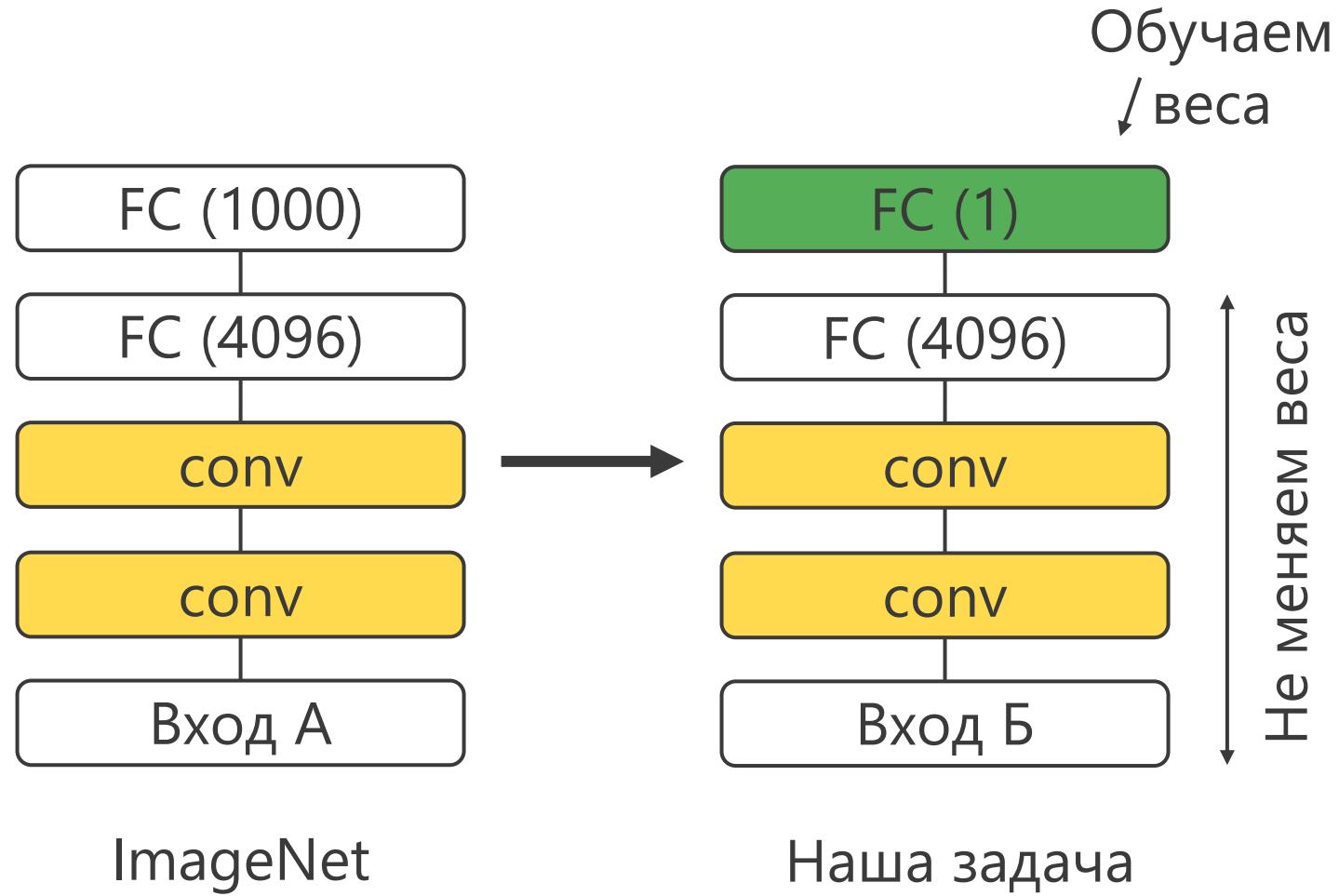


VS



- ▶ Рассмотрим произвольную задачу классификации изображений
- ▶ Как использовать модели, обученные на ImageNet?

Дообучение

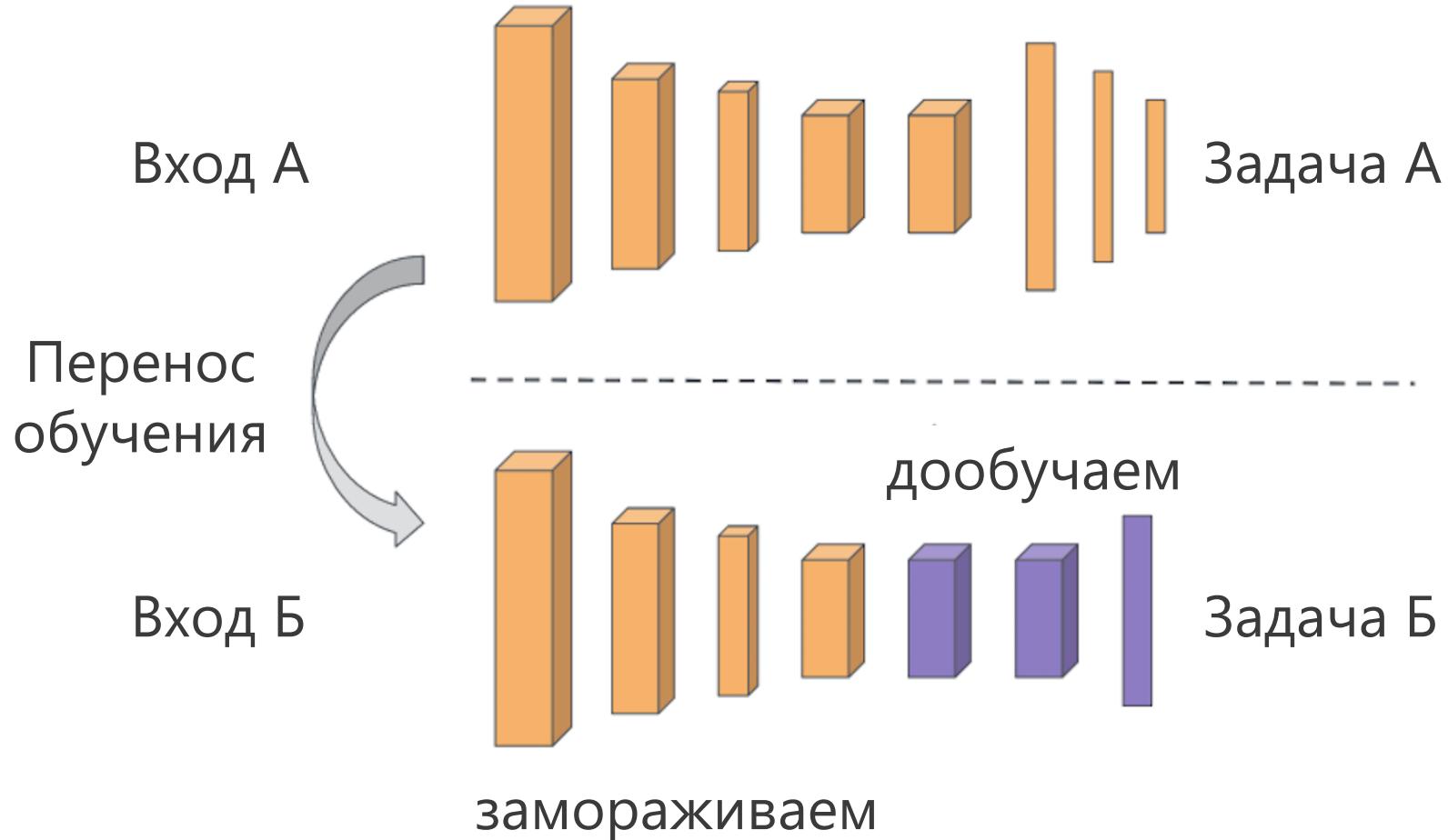


Дообучение

Если данных мало:

- ▶ Берем обученную модель из другой задачи (ImageNet)
- ▶ Заменяем выходной слой на один слой с нужным числом выходов
- ▶ Обучаем на своих данных только новый слой
- ▶ Остальные слои модели замораживаем (не обучаем)

Перенос обучения (fine-tuning)



Дообучение

Если данных достаточно много:

- ▶ Берем обученную модель из другой задачи (ImageNet)
- ▶ Заменяем несколько последних слоев на **несколько новых**
- ▶ Обучаем только новые слои
- ▶ Остальные слои модели замораживаем

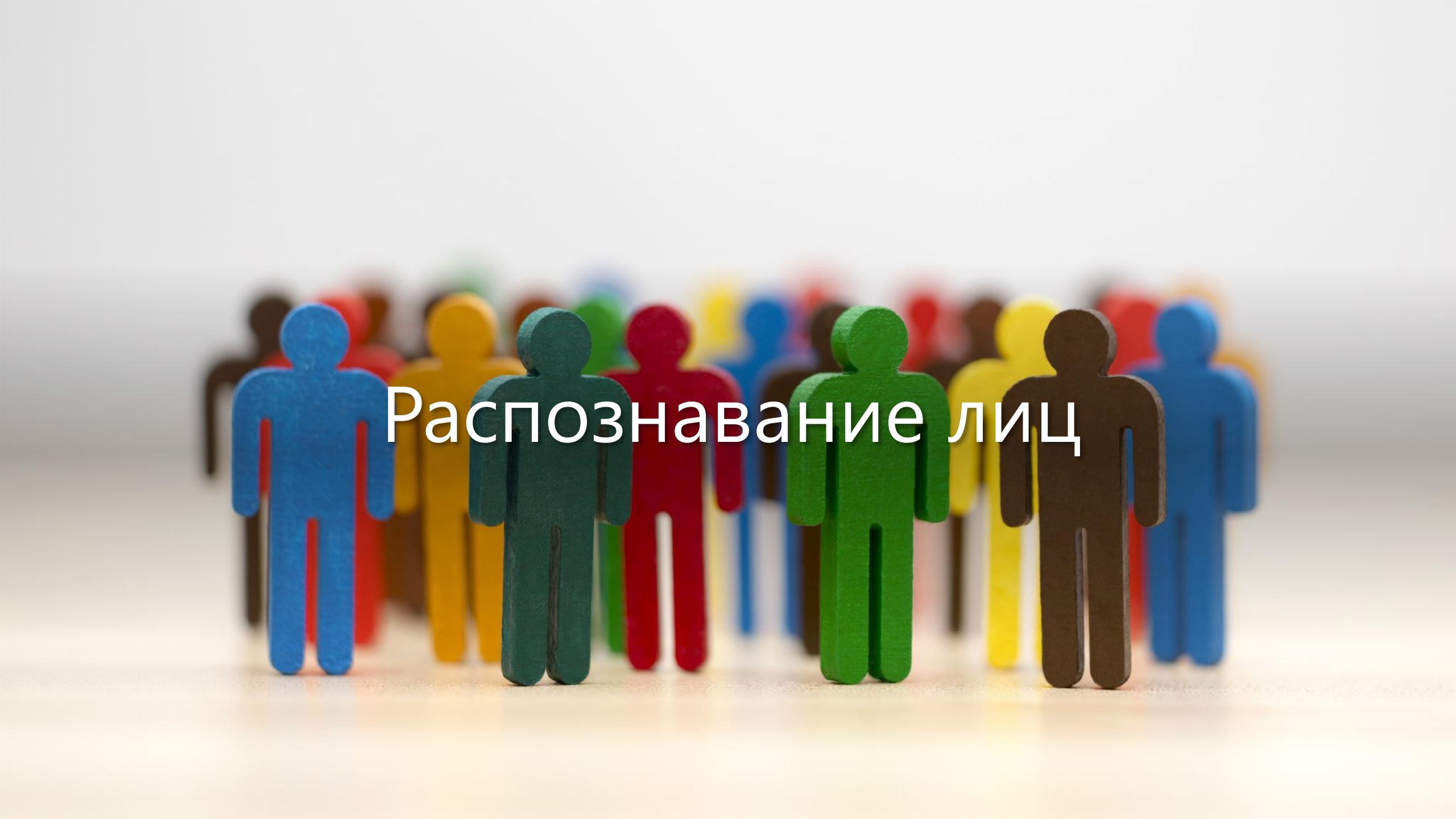
Дообучение

- ▶ Остальные слои обученной модели можно не замораживать
- ▶ Их можно дообучить на новых данных
- ▶ Но с меньшим шагом обучения (learning rate)

Перенос обучения

Когда перенос обучения работает:

- ▶ Когда задача А (ImageNet) похожа на нашу задачу В
- ▶ Одинаковые входы модели
- ▶ Для задачи А много больше данных, чем для задачи В



Распознавание лиц

Задача



Anchor



Positive



Anchor



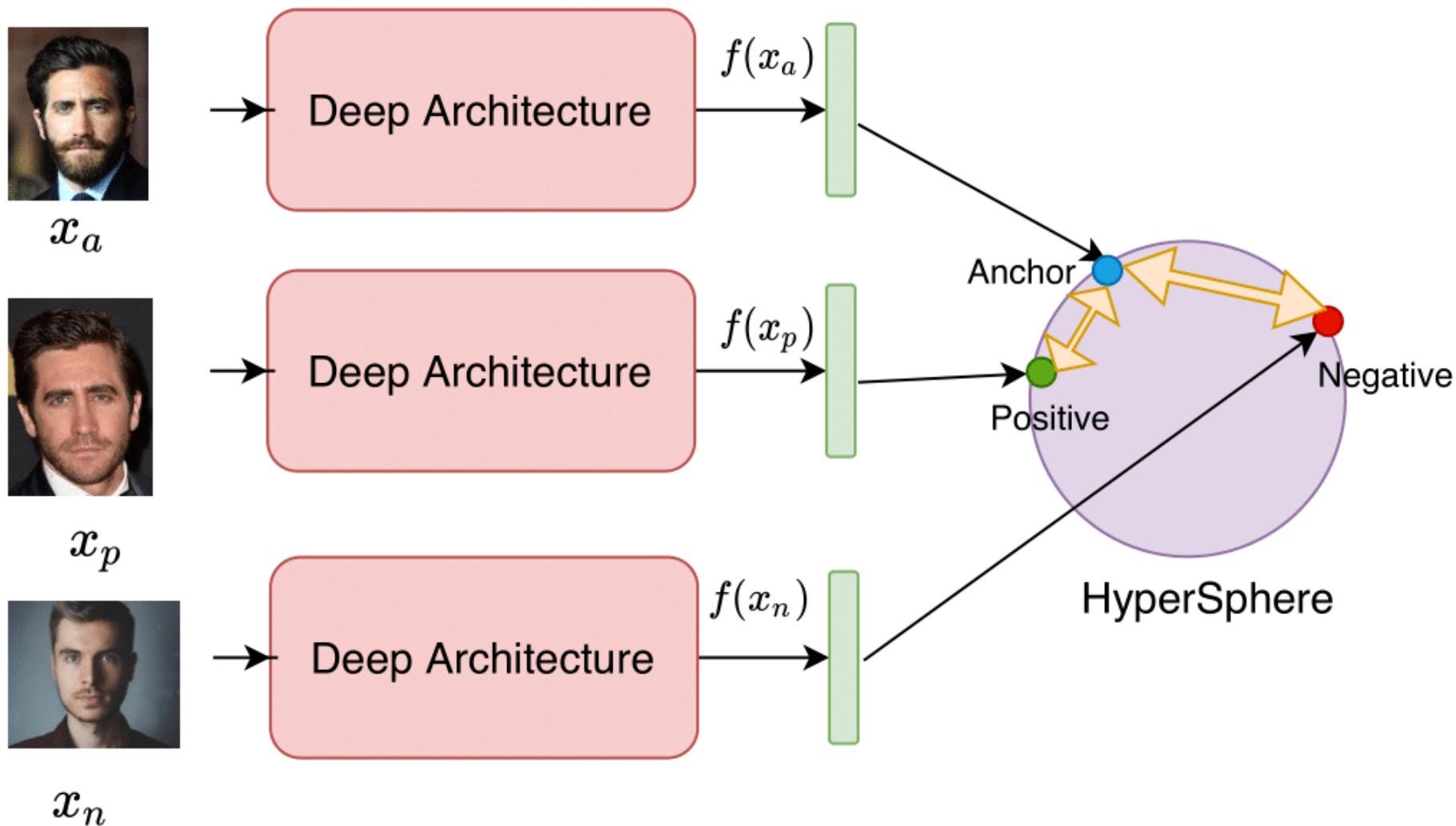
Negative

- ▶ Дан набор фотографий студентов вашей группы ☺
- ▶ Нужно обучить нейронную сеть распознавать студентов вашей группы по фото

Распознавание лиц

- ▶ Нейронная сеть должна распознавать вас на любой вашей фото
- ▶ Она также должна определить **любого** студента НЕ вашей группы
- ▶ Для обучения есть только фото студентов вашей группы
- ▶ Как будете действовать? 😊

Архитектура нейронной сети



Источник: <https://pyimagesearch.com/2023/03/06/triplet-loss-with-keras-and-tensorflow/>

Архитектура нейронной сети

- ▶ За основу берем любу (предобученную) нейронную сеть
- ▶ Создаем тройки фотографий
- ▶ Для каждой фото получаем вектор в скрытом представлении (эмбеддинг) $f(x)$
- ▶ Хотим, чтобы расстояние от якоря $f(x_a)$ до положительного примера $f(x_p)$, было меньше, чем до негативного $f(x_n)$

Triplet loss

Для обучения сети используем triplet loss:

$$L_{triplet} = \text{Max} \left(\|f(x_a) - f(x_p)\|_2^2 - \|f(x_a) - f(x_n)\|_2^2 + \alpha, 0 \right) \rightarrow \min_f$$

α – константа, гиперпараметр. Минимальное расстояние между якорем и негативным примером.

Triplet loss

Обозначим:

$$\text{apDistance} = \|f(x_a) - f(x_p)\|_2^2$$

$$\text{anDistance} = \|f(x_a) - f(x_n)\|_2^2$$

Тогда

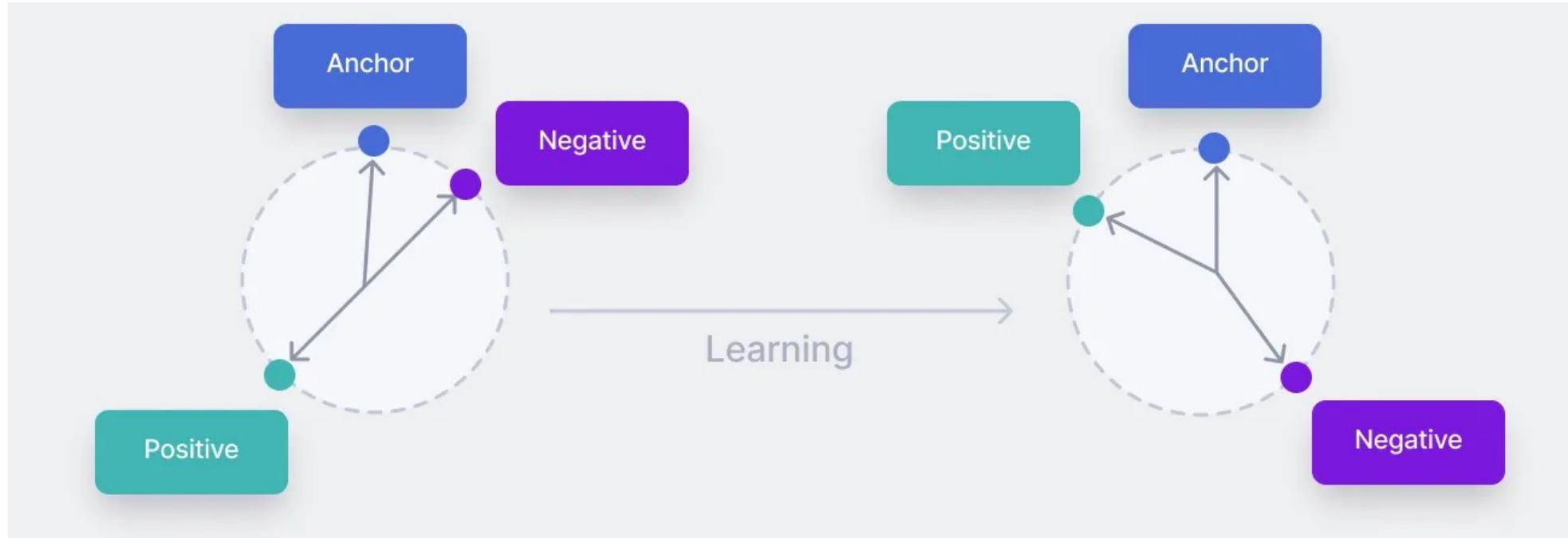
$$L_{triplet} = \text{Max}(\text{apDistance} - \text{anDistance} + \alpha, 0) \rightarrow \min_f$$

Минимум достигается когда $L_{triplet} = 0$:

$$\text{apDistance} - \text{anDistance} + \alpha \leq 0$$

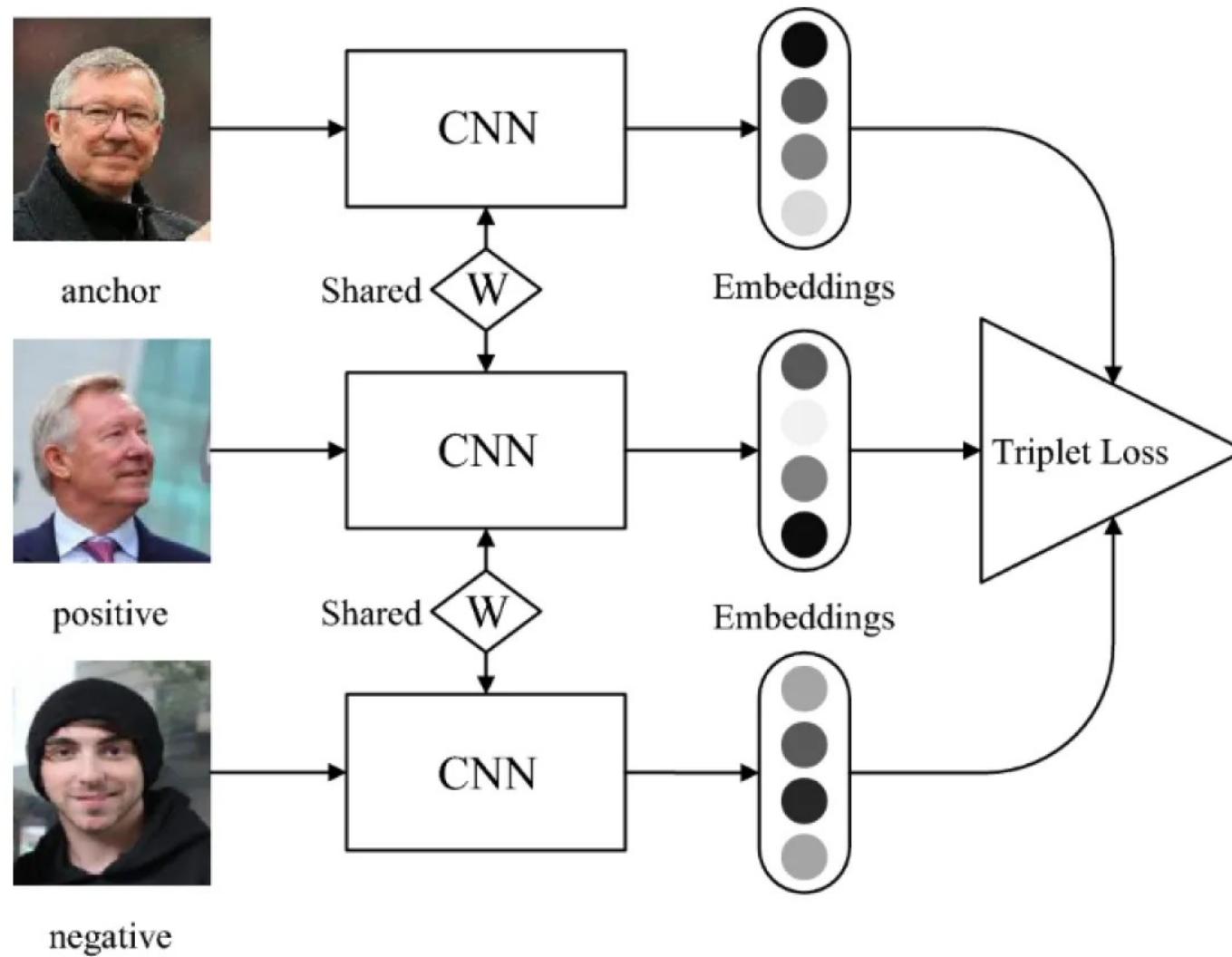
$$\text{anDistance} \geq \text{apDistance} + \alpha$$

Triplet loss



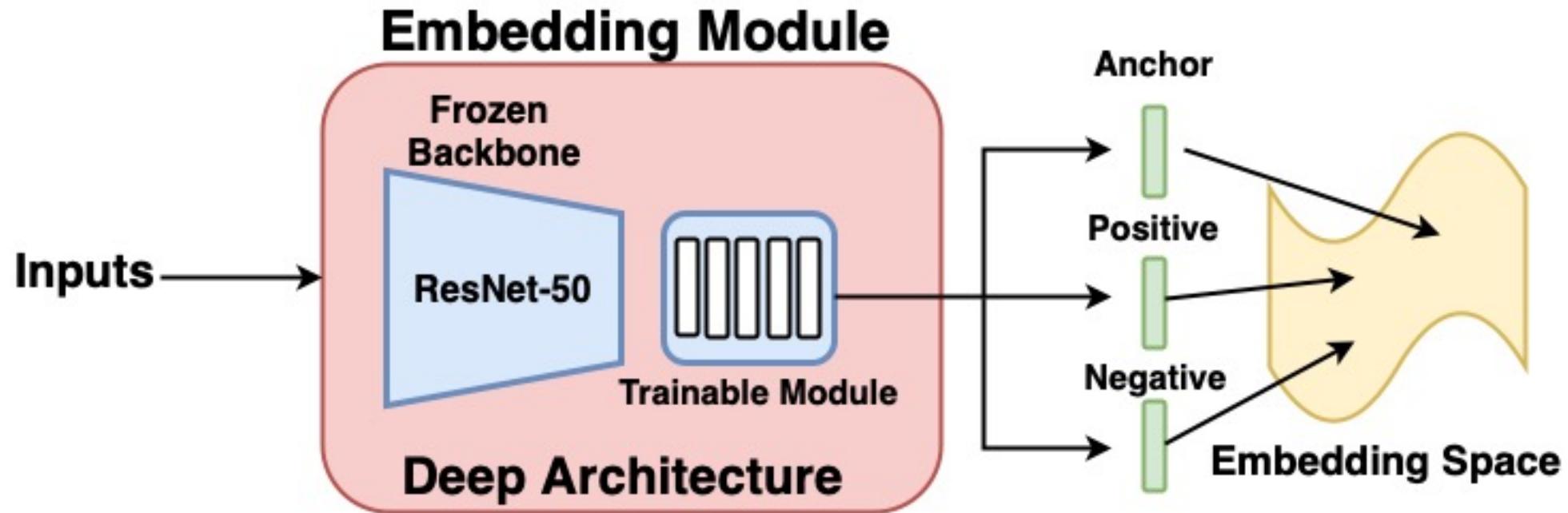
Источник: <https://www.v7labs.com/blog/triplet-loss>

Обучение сети



Обучение сети

Источник: <https://pyimagesearch.com/2023/03/06/triplet-loss-with-keras-and-tensorflow/>



- ▶ Используем перенос обучения
- ▶ За основу берем обученную сеть на ImageNet. Например, ResNet-50)