

Нейронные сети

Интерпретация моделей и
metric learning

Данил Кашин



Не забывайте
отмечаться и
оставлять
отзыв



Содержание лекции

1. Часть 1. Объяснение предсказаний NN:

- a. LIME
- b. Градиентные методы
- c. CAM
- d. Grad-CAM

2. Часть 2. Metric learning:

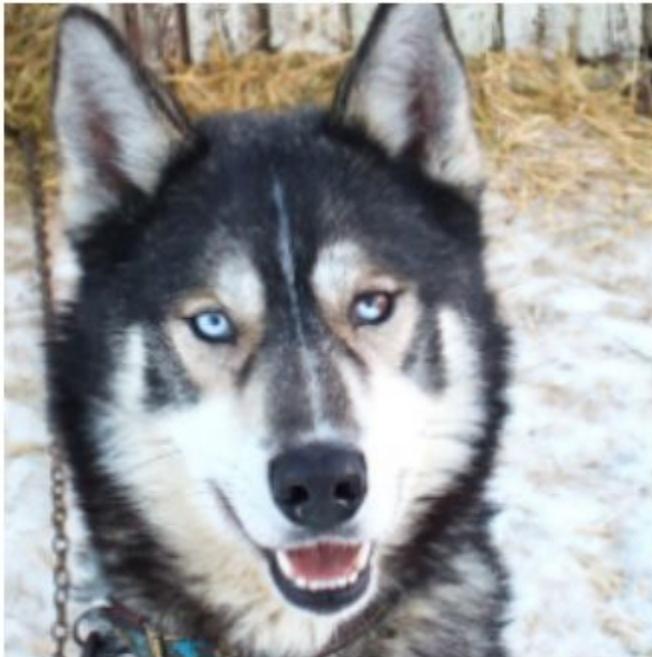
- a. Contrastive loss
- b. Triplet loss
- c. Center loss
- d. Arcface



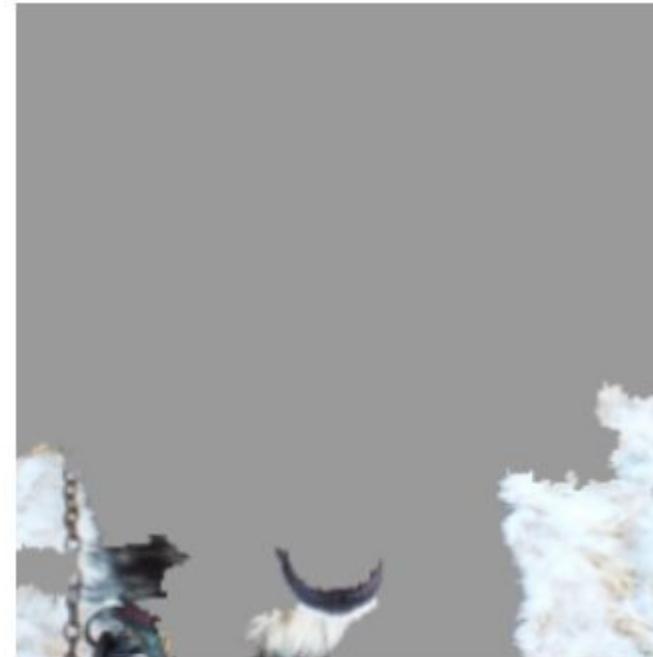
Объяснение предсказаний NN

• • •

Объяснение предсказаний NN



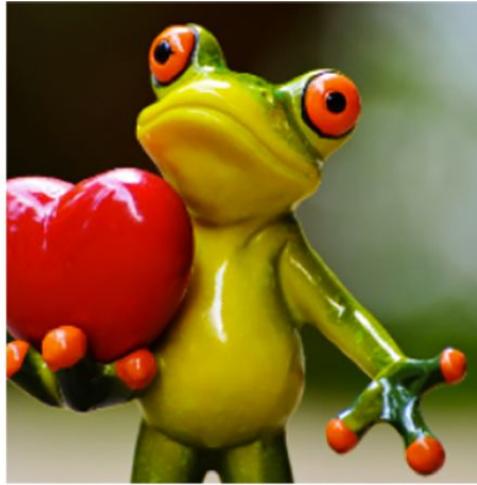
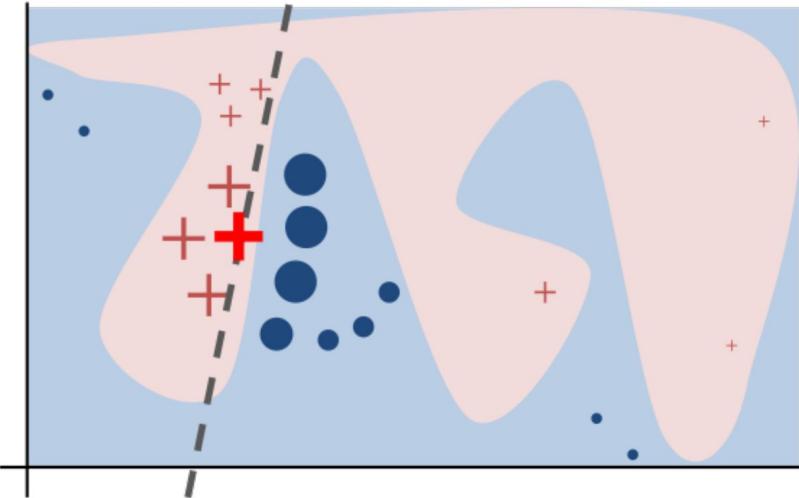
(a) Husky classified as wolf



(b) Explanation

Figure 11: Raw data and explanation of a bad model’s prediction in the “Husky vs Wolf” task.

LIME (BlackBox)



Original Image

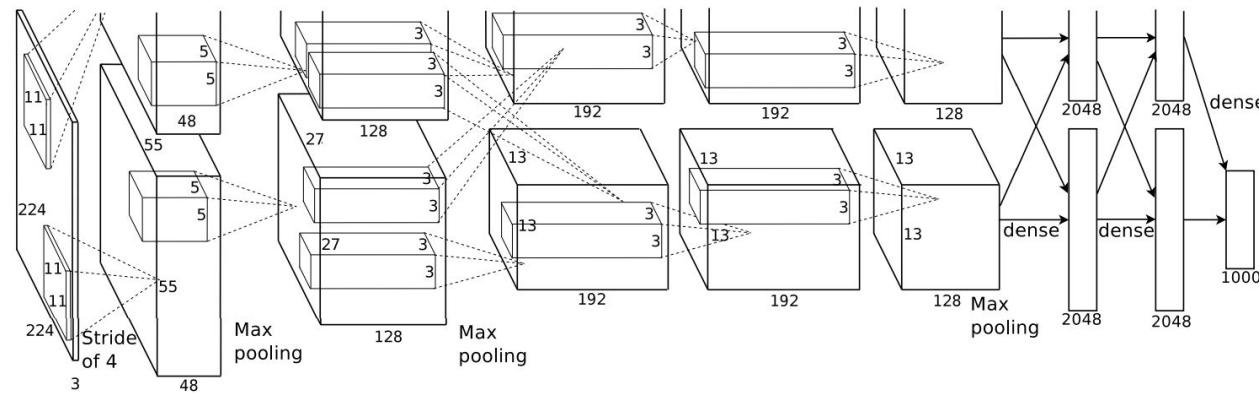


Interpretable Components

- Подходит для любой модели
- Работает медленно (около 1000 сэмплов)
- Сложно делать хороший сэмплинг

<https://arxiv.org/pdf/1602.04938.pdf>

Визуализация и анализ CNN



ImageNet Classification with Deep Convolutional Neural Networks

https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Визуализация промежуточных слоёв

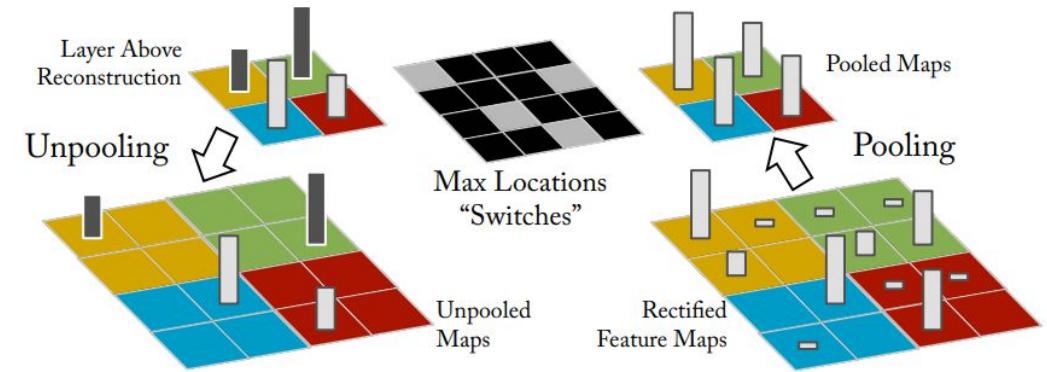
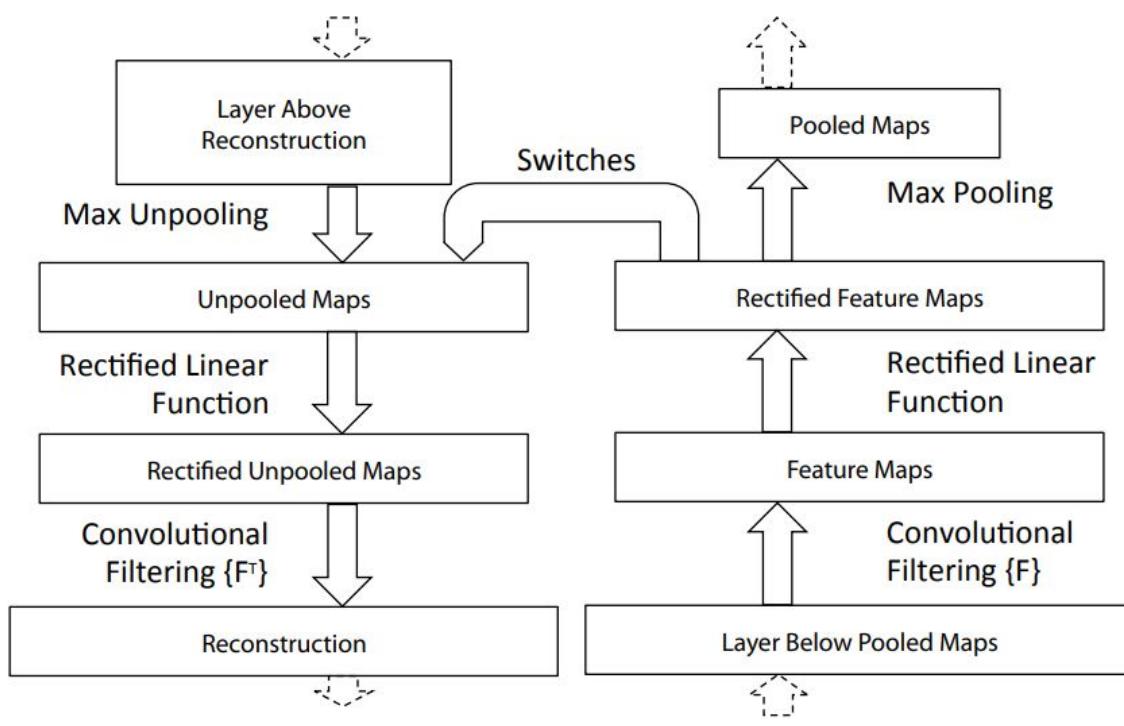
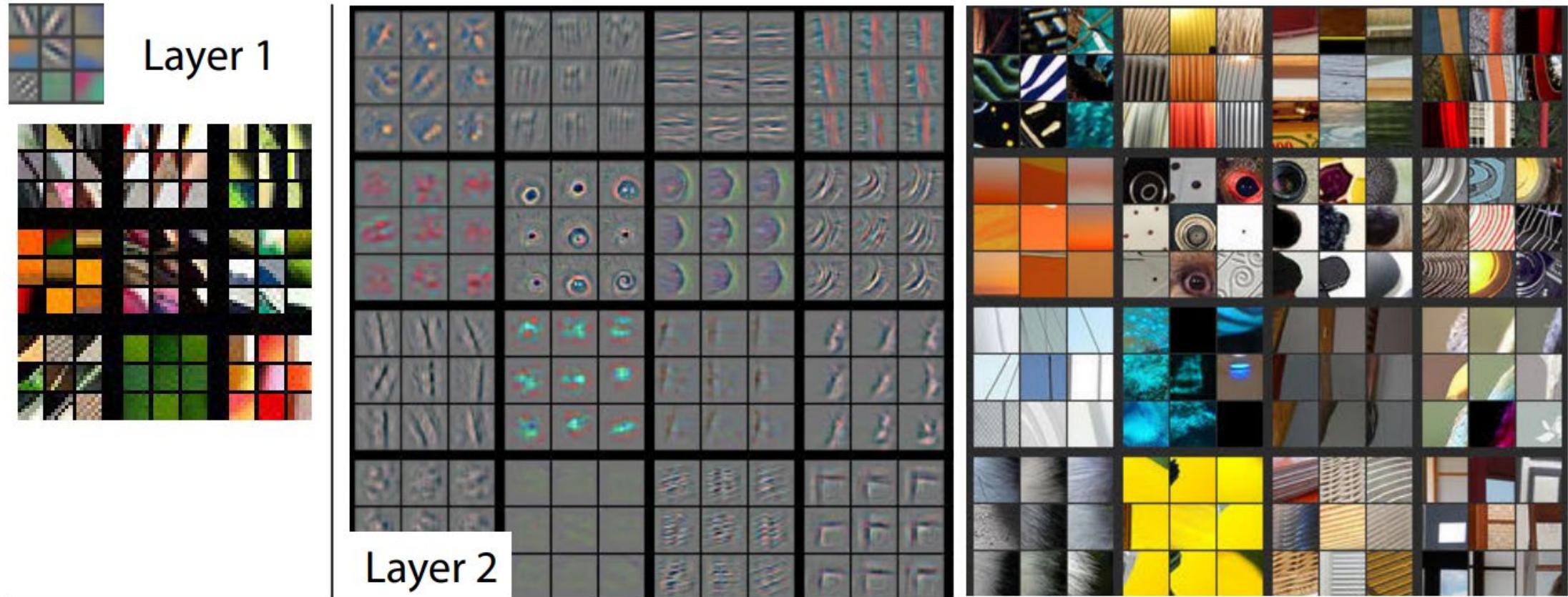
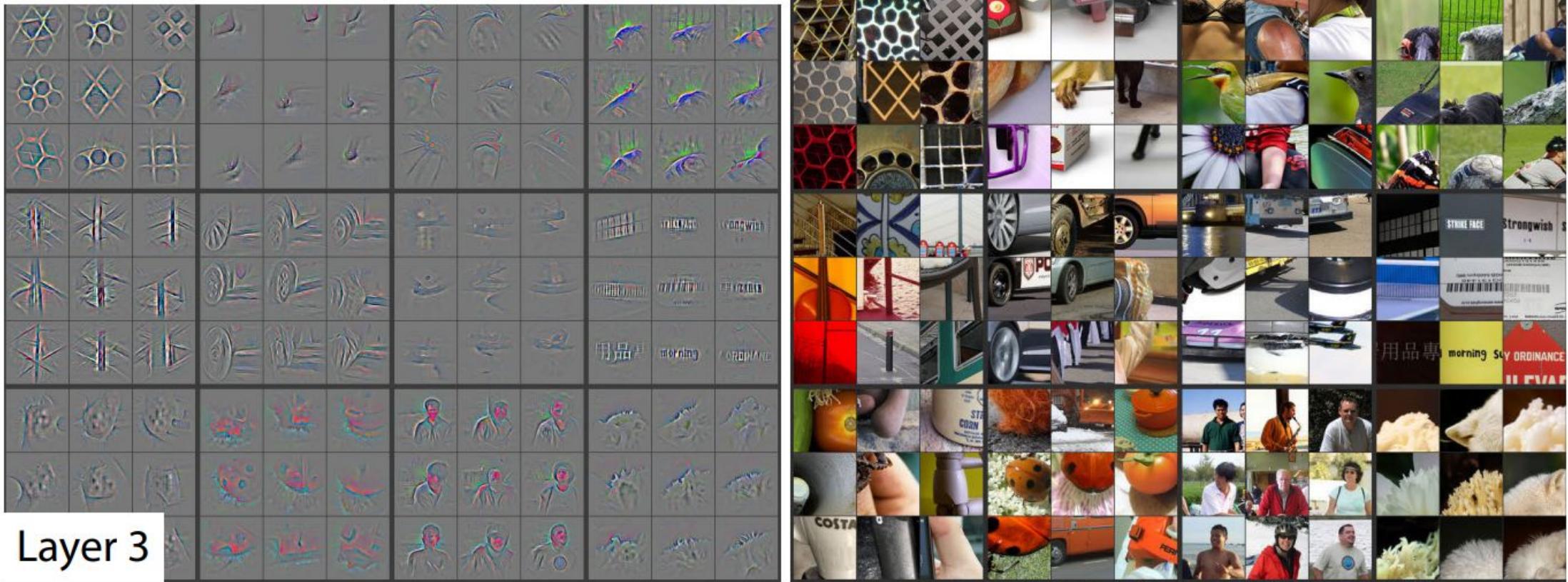


Figure 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet.

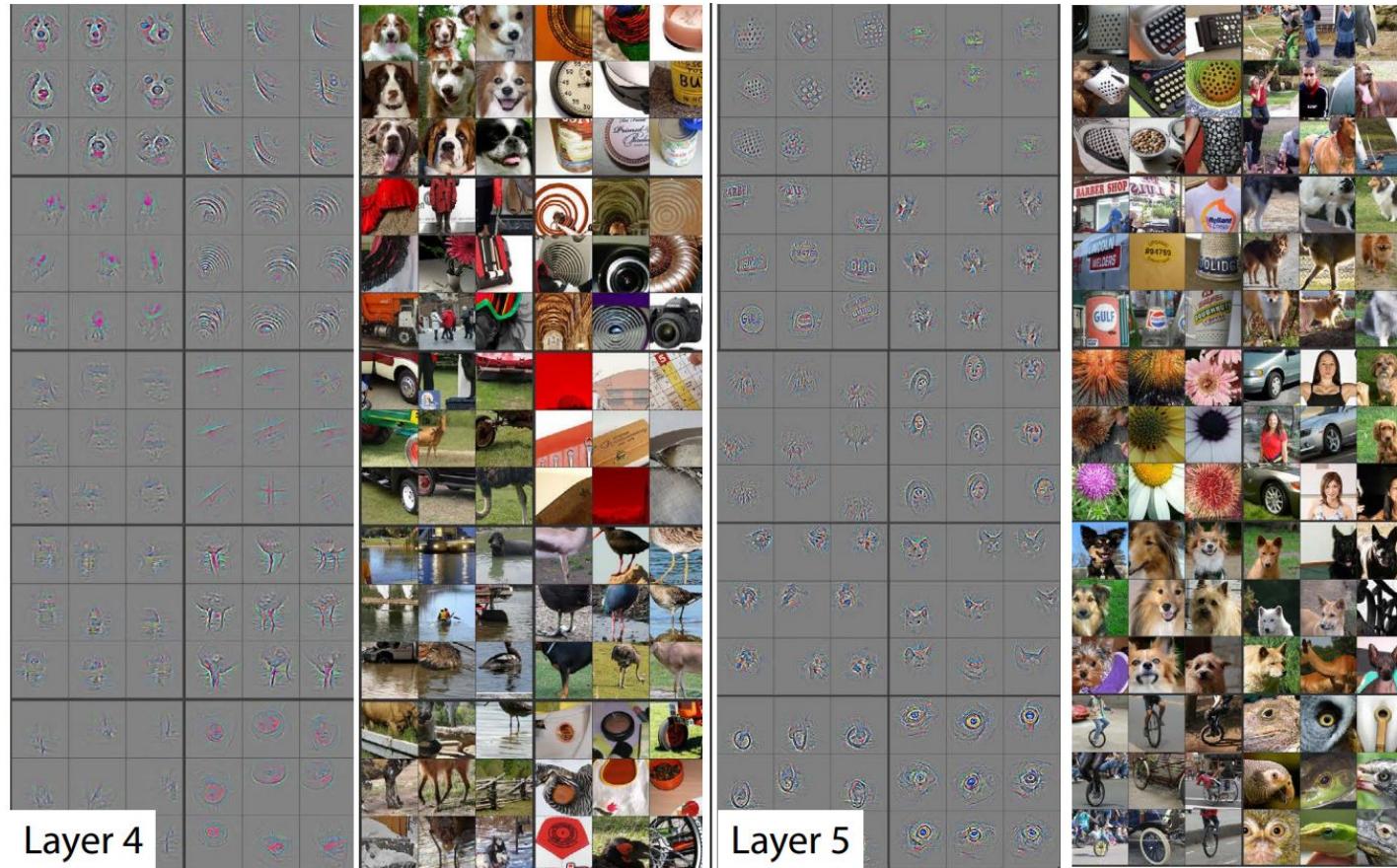
Визуализация промежуточных слоёв



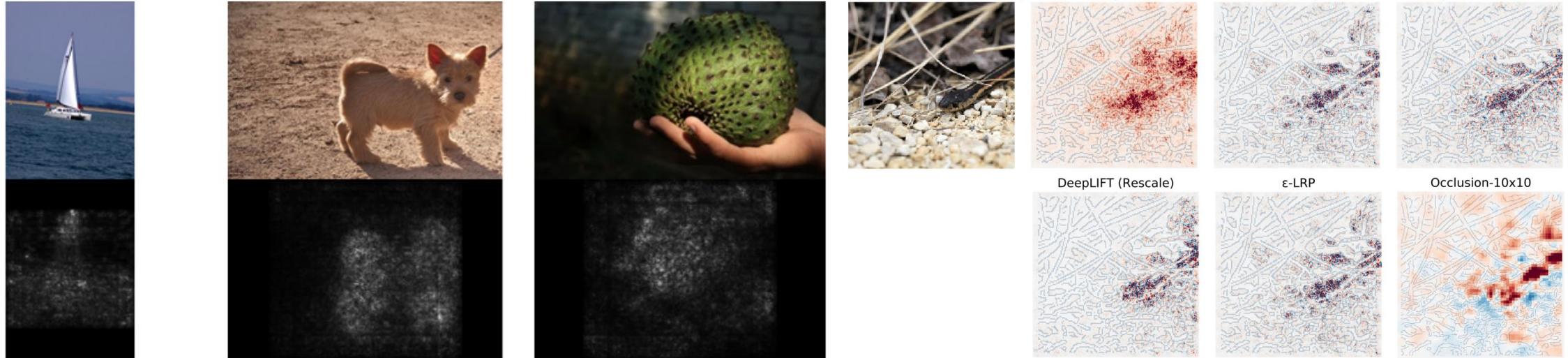
Визуализация промежуточных слоёв



Визуализация промежуточных слоёв



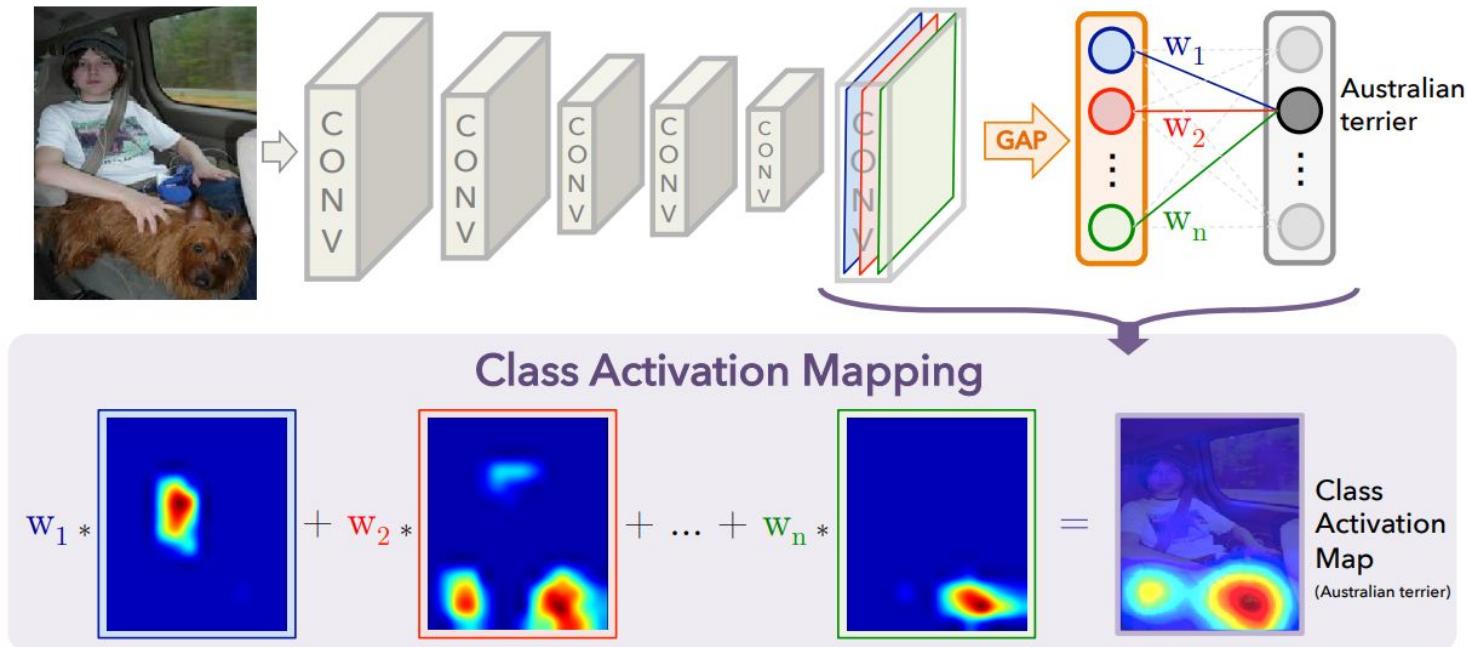
Градиентные методы. Saliency-maps



- Идея: производная выхода по отношению к входу (как изменится выход, если немного изменим вход)
- Быстрые
- Подходят для любой дифференцируемой модели

- ❖ <https://arxiv.org/pdf/1312.6034.pdf>
- ❖ <https://arxiv.org/pdf/1711.06104.pdf>

Class Activation Mapping



- Быстрые
- Подходят для любой дифференцируемой модели
- Только для GAP (Global Average Pooling) (!)

Class Activation Mapping

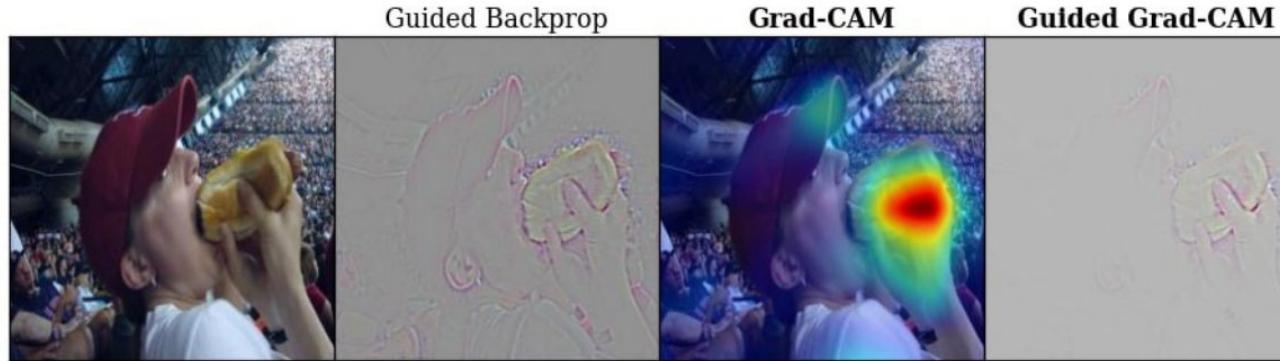
```
def returnCAM(feature_conv, weight_softmax, class_idx):
    # generate the class activation maps upsample to 256x256
    size_upsample = (256, 256)
    bz, nc, h, w = feature_conv.shape
    output_cam = []
    for idx in class_idx:
        cam = weight_softmax[idx].dot(feature_conv.reshape((nc, h*w)))
        cam = cam.reshape(h, w)
        cam = cam - np.min(cam)
        cam_img = cam / np.max(cam)
        cam_img = np.uint8(255 * cam_img)
        output_cam.append(cv2.resize(cam_img, size_upsample))
    return output_cam
```

Average pooling + softmax

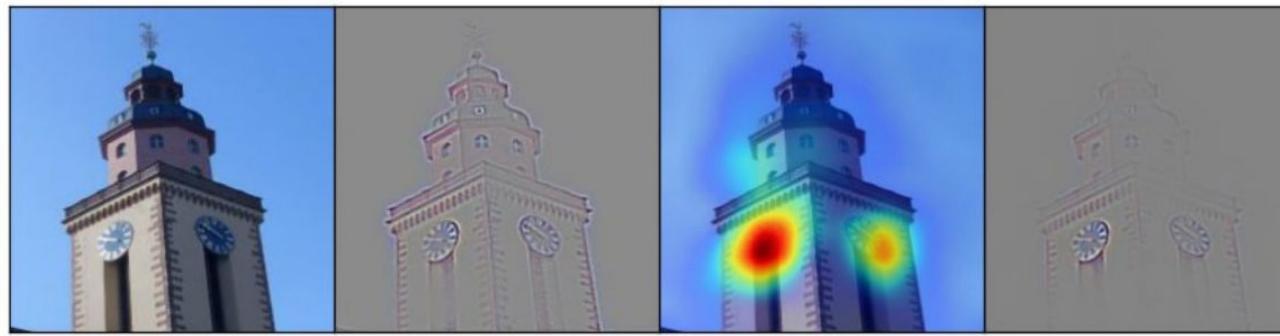
- ResNet
- DenseNet
- SqueezeNet
- Inception
- ...

https://github.com/zhoubolei/CAM/blob/master/pytorch_CAM.py

Grad-CAM



Задача Image captioning

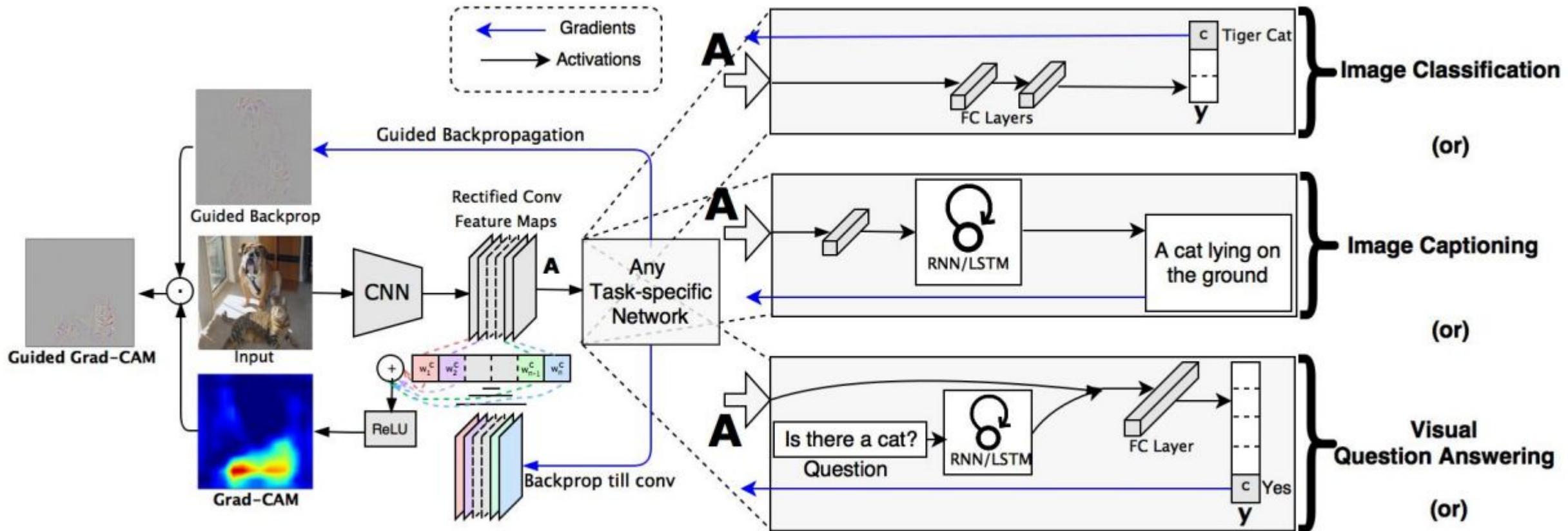


Хочется использовать САМ с более сложной головой

- два полно связанных слоя
- предсказание текста
- или с любым другим выходом

<https://arxiv.org/pdf/1610.02391.pdf>

Grad-CAM



Берём производные не к выходу, а
к активации на конкретном слое

Grad-CAM

```
def get_cam_image(self,
                  input_tensor: torch.Tensor,
                  target_layer: torch.nn.Module,
                  targets: List[torch.nn.Module],
                  activations: torch.Tensor,
                  grads: torch.Tensor,
                  eigen_smooth: bool = False) -> np.ndarray:

    weights = self.get_cam_weights(input_tensor,
                                    target_layer,
                                    targets,
                                    activations,
                                    grads)

    weighted_activations = weights[:, :, None, None] * activations

    if eigen_smooth:
        cam = get_2d_projection(weighted_activations)
    else:
        cam = weighted_activations.sum(axis=1)

    return cam
```

```
class GradCAM(BaseCAM):
    def __init__(self, model, target_layers, use_cuda=False,
                 reshape_transform=None):
        super(
            GradCAM,
            self).__init__(
                model,
                target_layers,
                use_cuda,
                reshape_transform)

    def get_cam_weights(self,
                        input_tensor,
                        target_layer,
                        target_category,
                        activations,
                        grads):
        return np.mean(grads, axis=(2, 3))
```

Вопросы?

Metric learning

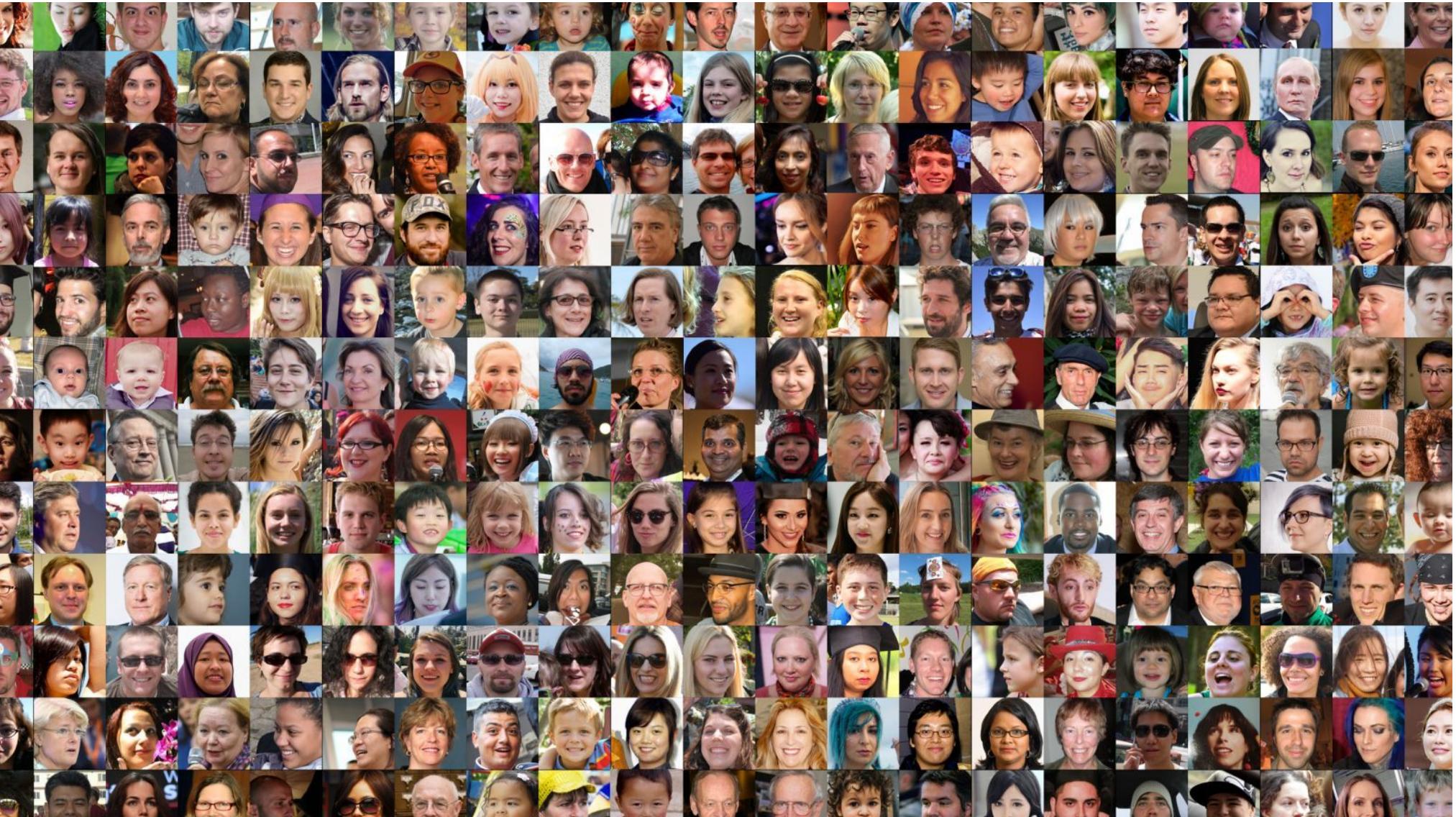
• • •

Введение

1. При решении задач компьютерного зрения достаточно редки ситуации, когда имеется много изображений, характеризующих каждый интересующий класс объектов
2. Чаще всего имеется достаточно сильная несбалансированность классов



Введение

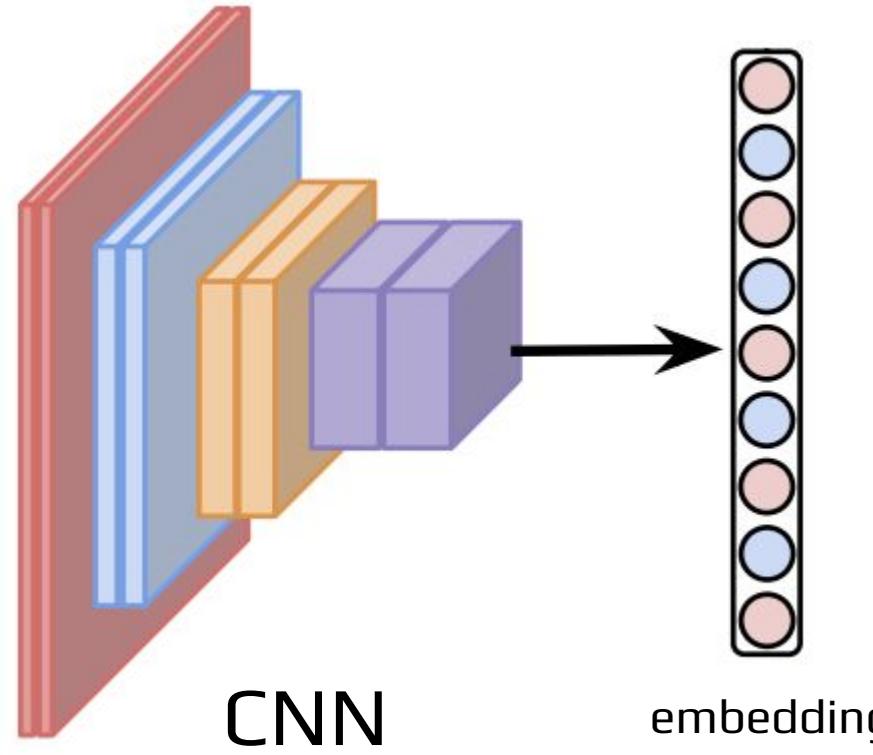


Введение

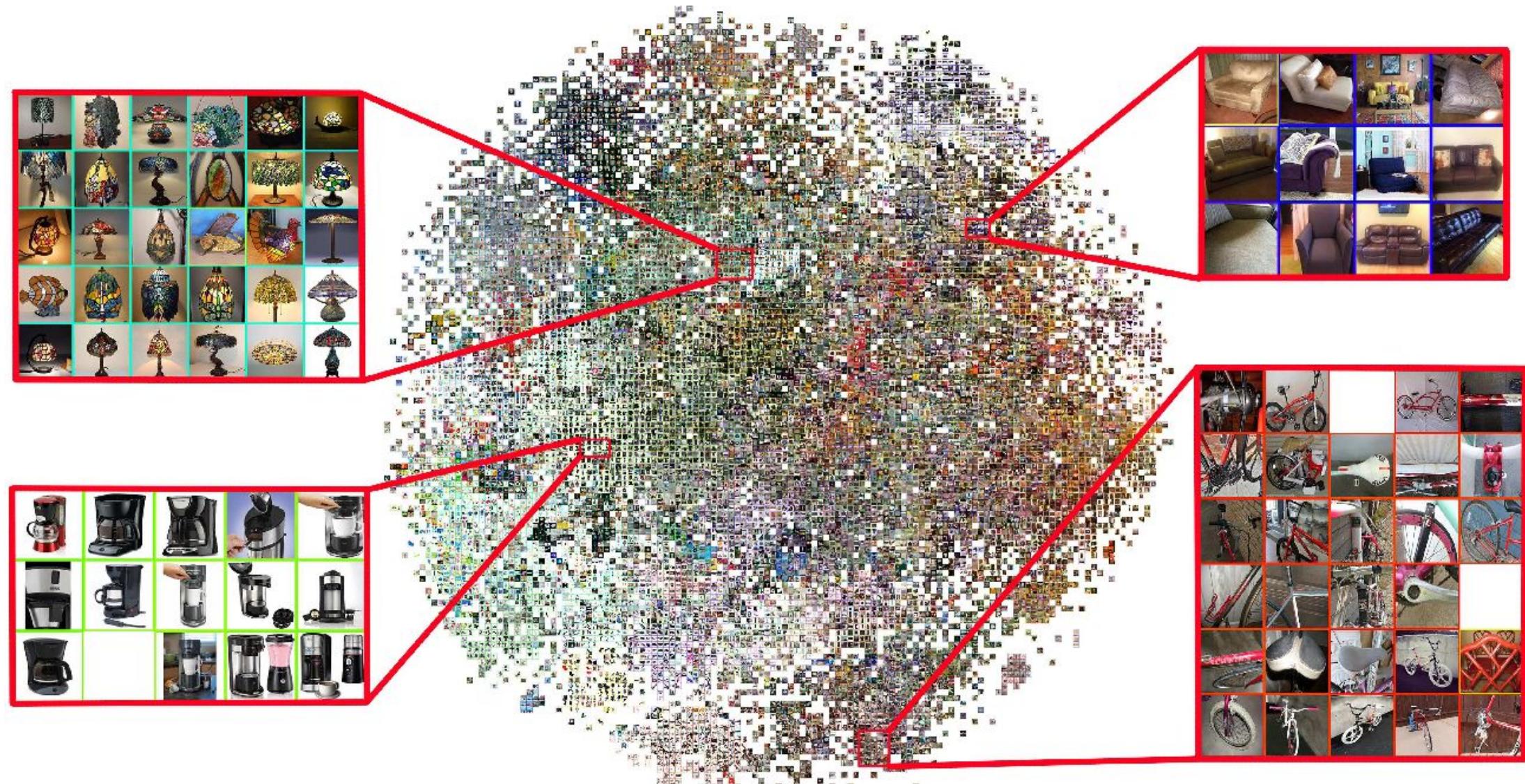
1. Создание модели, хорошо описывающей изображения
2. Обучение модели таким образом, чтобы сохранялась «близость» между embedding'ами изображений, имеющих семантическое сходство
3. Возможность использовать полученную модель для предсказания классов, которые не были представлены в обучающей выборке

$$\phi \left(\begin{array}{c} \text{[Image of a silver car]} \\ \text{[Image of a silver car]} \end{array} \right) < \phi \left(\begin{array}{c} \text{[Image of a silver car]} \\ \text{[Image of a green lawn mower]} \end{array} \right) < \phi \left(\begin{array}{c} \text{[Image of a silver car]} \\ \text{[Image of an airplane]} \end{array} \right)$$

Image embedding



«Близость» между embedding'ами изображений



Формулировка

- Имеется некоторое отображение для изображений в embedding'и

$$f(x_i): R^L \rightarrow R^D$$

- Метрика схожести объектов

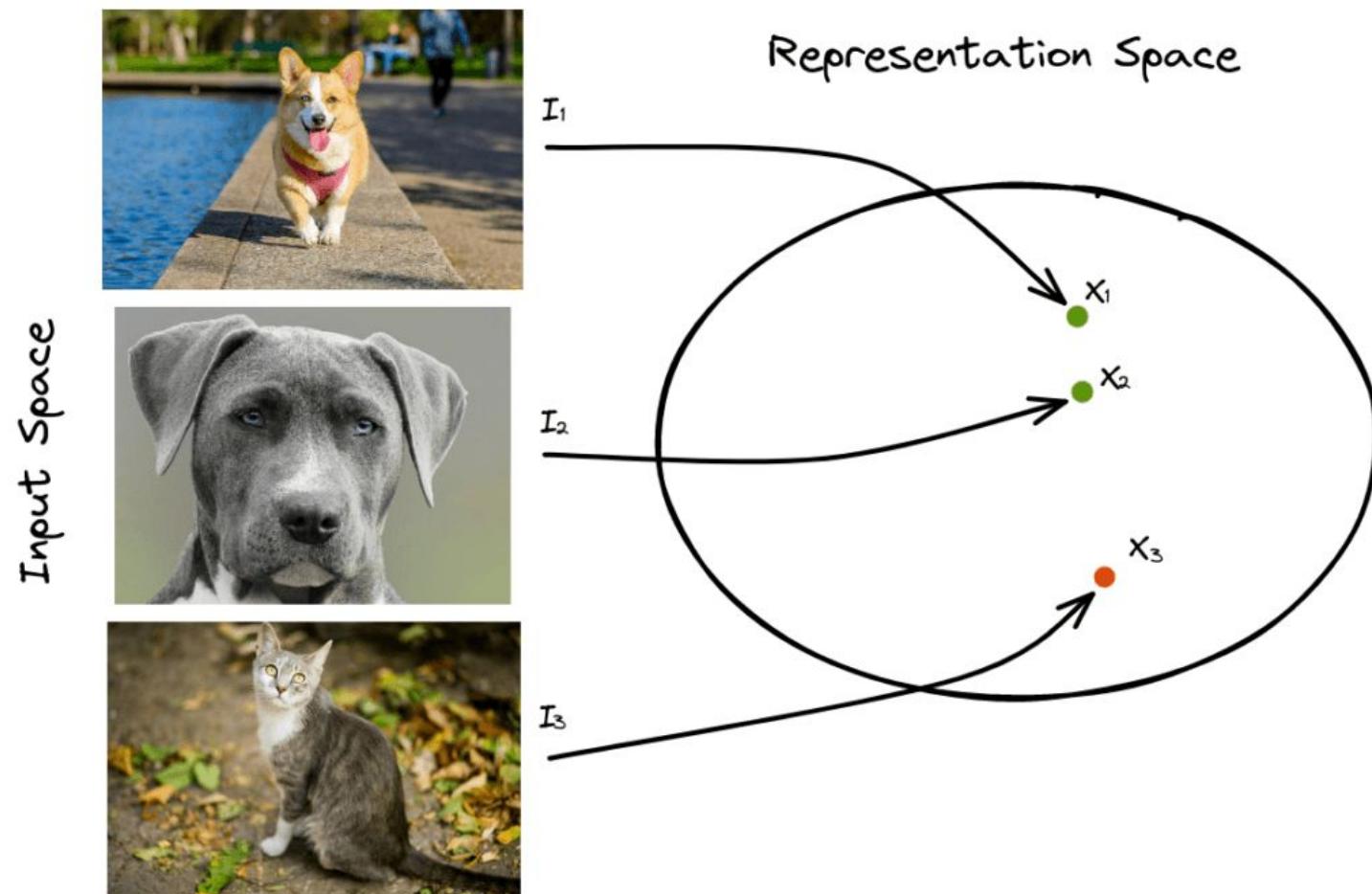
$$\phi(x_i, x_j) = \text{distance}\left(f(x_i), f(x_j)\right),$$

где *distance* может быть евклидовым, косинусным расстоянием и т.д.

$$\phi \left(\begin{array}{c} \left[\begin{array}{c} \text{Image 1} \\ \text{Image 2} \end{array} \right] \end{array} \right) < \phi \left(\begin{array}{c} \left[\begin{array}{c} \text{Image 1} \\ \text{Image 3} \end{array} \right] \end{array} \right) < \phi \left(\begin{array}{c} \left[\begin{array}{c} \text{Image 1} \\ \text{Image 4} \end{array} \right] \end{array} \right)$$

Контрастные методы

1. Contrastive Loss
2. Triplet Loss
3. Quadruplet Loss
4. N-Pair Loss
5. other

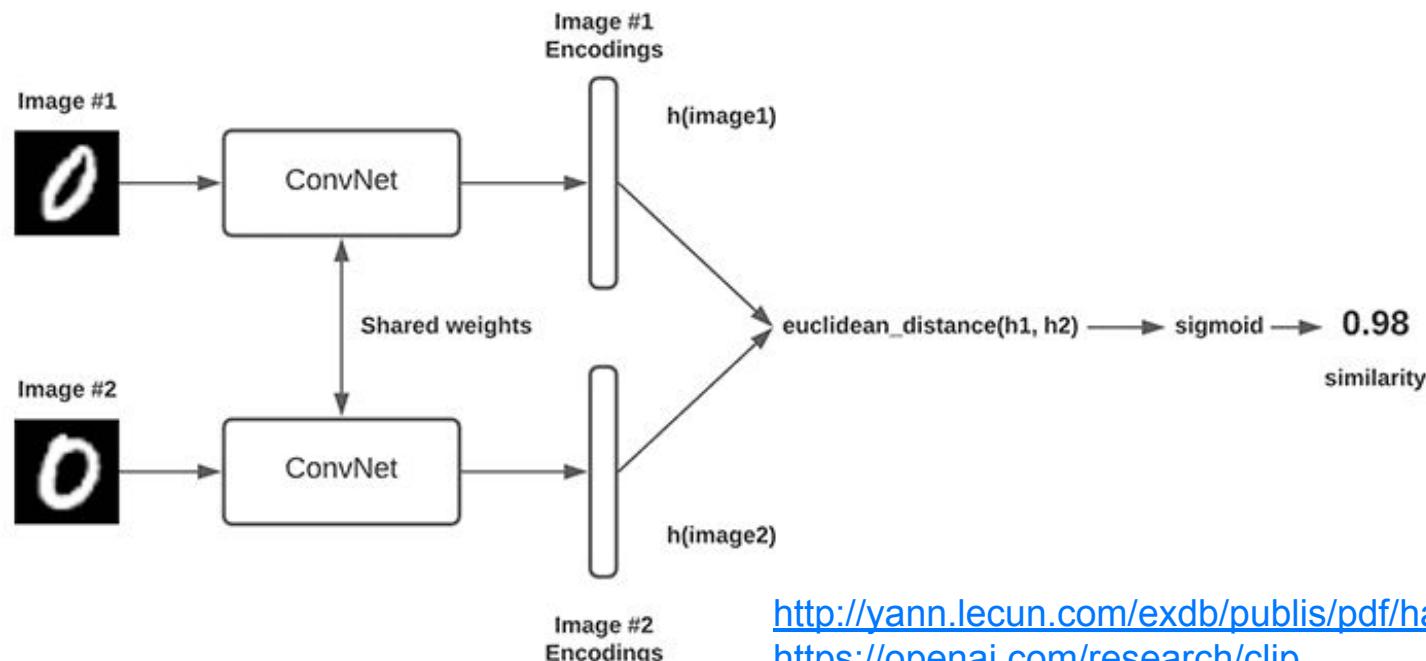


Contrastive Loss

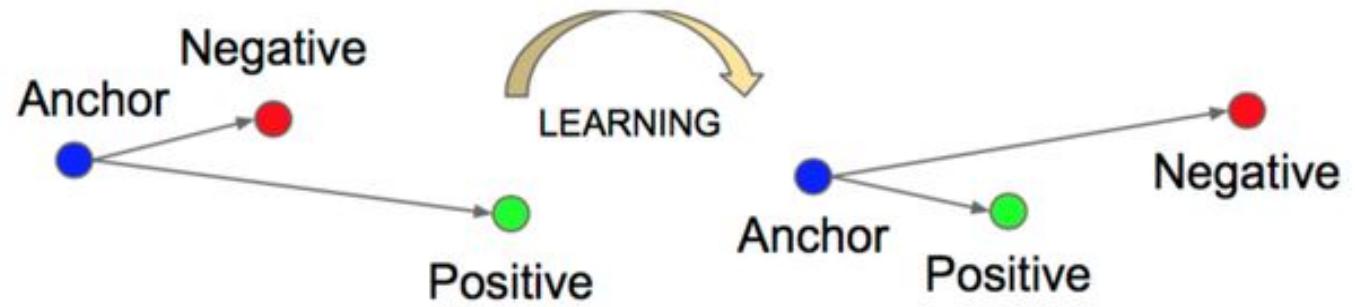
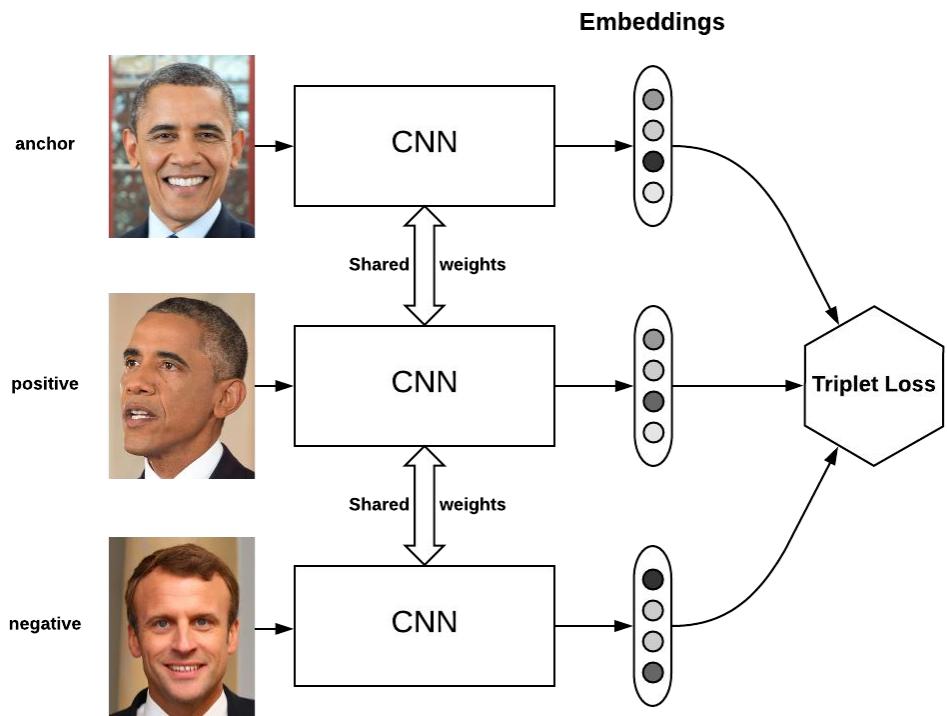
Разделяем выборку на positive и negative пары

Ставим метку $y_{ij} = 1$ для positive пар и $y_{ij} = -1$ для negative пар

$$L(X, \Theta) = \frac{1}{2} \sum_{i=1}^m y_{ij} \|f(x_i) - f(x_j)\|_2^2 + (1 - y_{ij}) \max(0, [\alpha - \|f(x_i) - f(x_j)\|_2]^2)$$



Triplet Loss



$$L(X, \Theta) = \frac{1}{2} \sum_{i=1}^m \max(0, \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha)$$

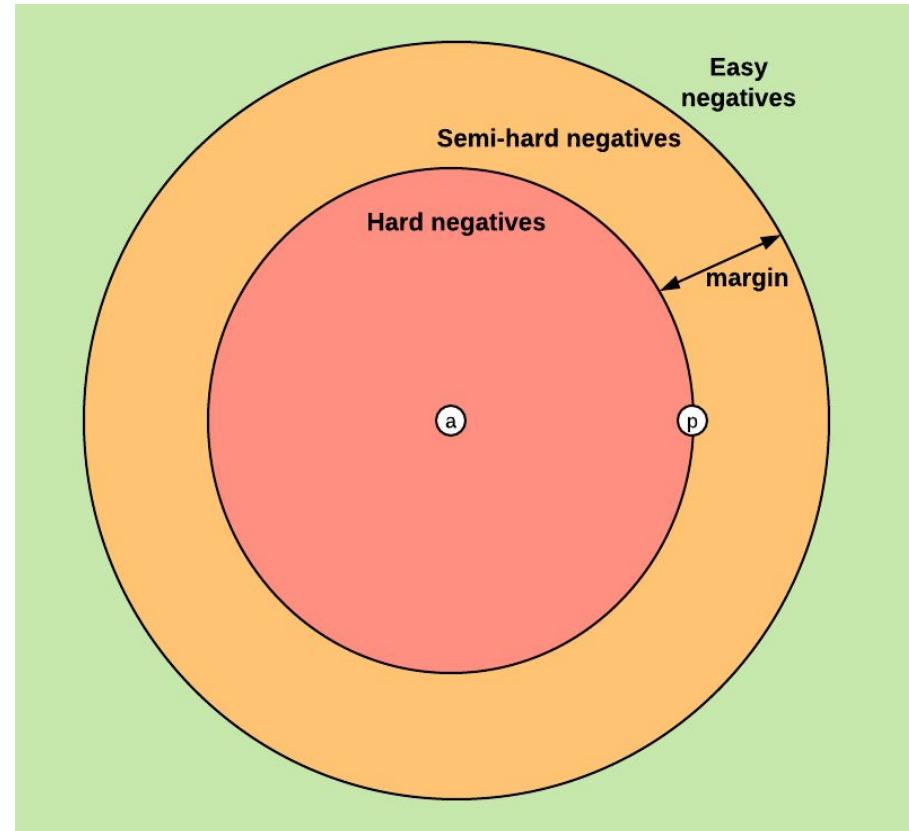
<https://arxiv.org/pdf/1412.6622.pdf>

Mining

Based on the definition of the loss, there are three categories of triplets:

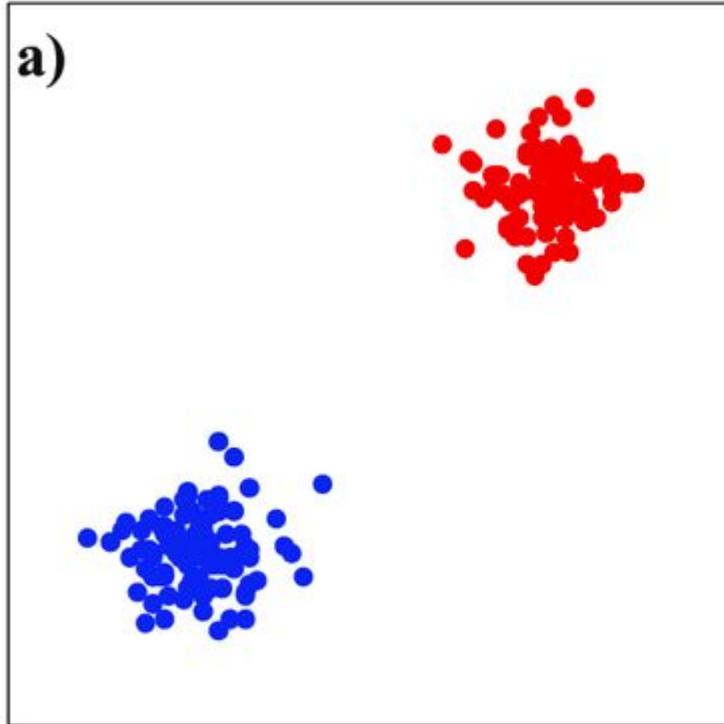
- **easy triplets**: triplets which have a loss of 0, because $d(a, p) + \text{margin} < d(a, n)$
- **hard triplets**: triplets where the negative is closer to the anchor than the positive, i.e. $d(a, n) < d(a, p)$
- **semi-hard triplets**: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss:
$$d(a, p) < d(a, n) < d(a, p) + \text{margin}$$

Each of these definitions depend on where the negative is, relatively to the anchor and positive. We can therefore extend these three categories to the negatives: **hard negatives, semi-hard negatives or easy negatives**.

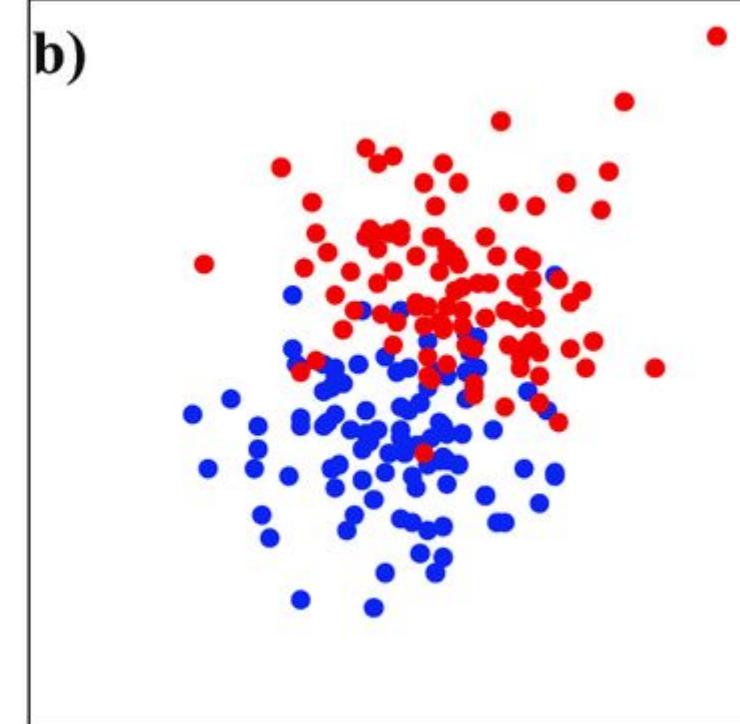


<https://omoindrot.github.io/triplet-loss#triplet-mining>

Inter-class and Intra-class variances concept



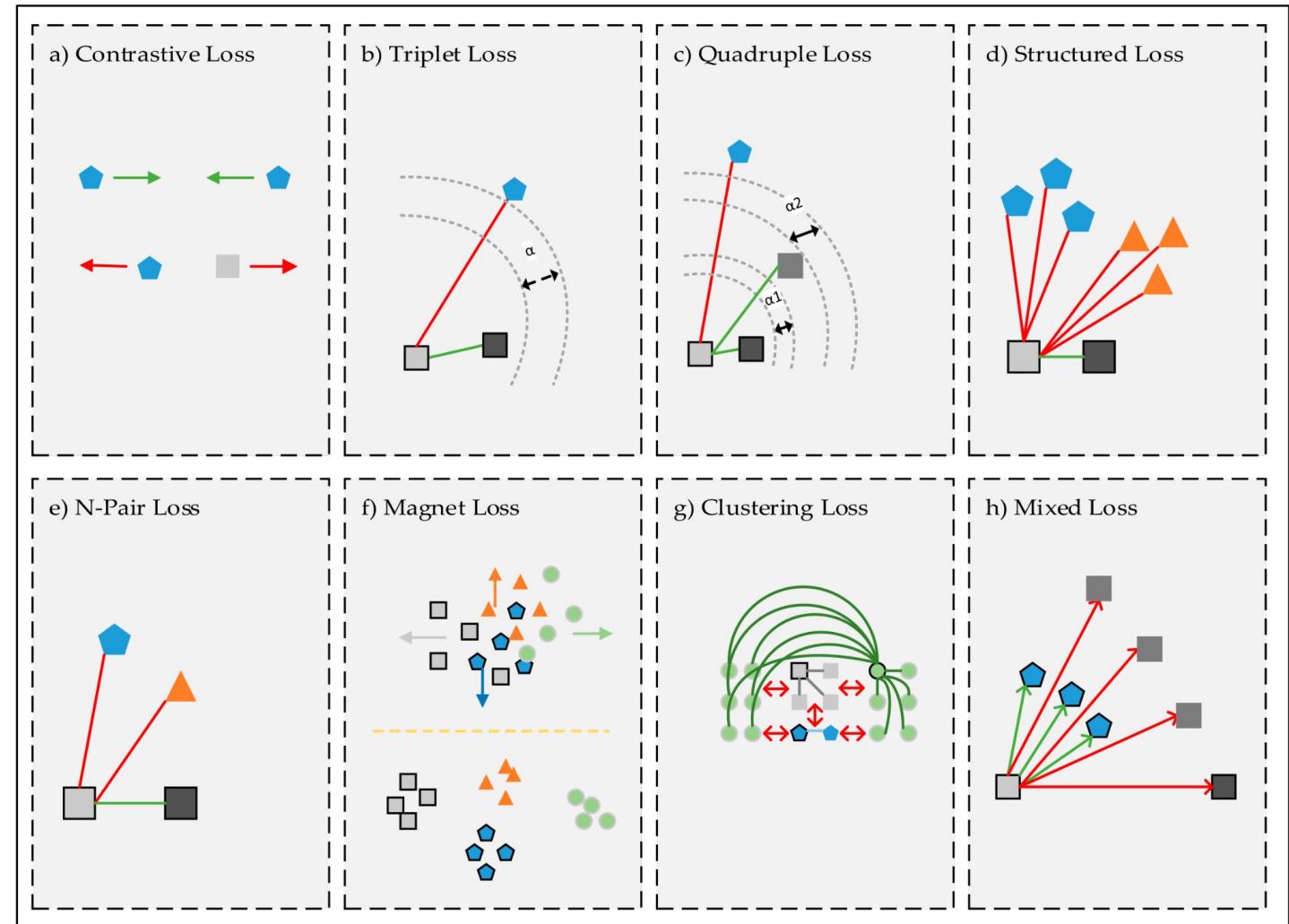
a) Low intra-class variance and high inter-class variance: compact well separated clusters.



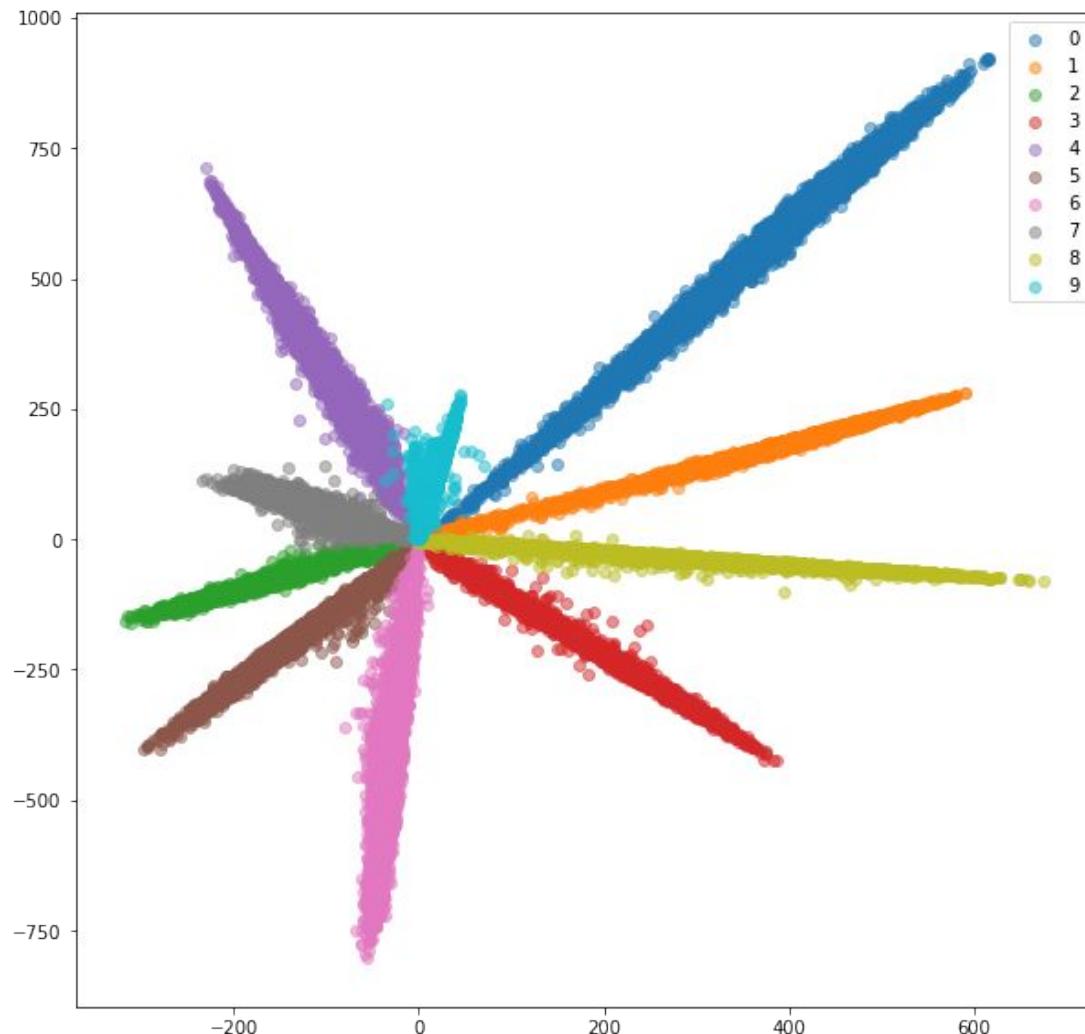
b) High intra-class variance and low inter-class variance: wide clusters without a clear frontier.

Quadruplet Loss

$$\mathcal{L}_{\text{quadruplet}} = \max \left(0, \mathcal{D}_{f_\theta}^2(x_a, x_p) - \mathcal{D}_{f_\theta}^2(x_a, x_s) + \alpha_1 \right) \\ + \max \left(0, \mathcal{D}_{f_\theta}^2(x_a, x_s) - \mathcal{D}_{f_\theta}^2(x_a, x_n) + \alpha_2 \right)$$



Softmax Loss



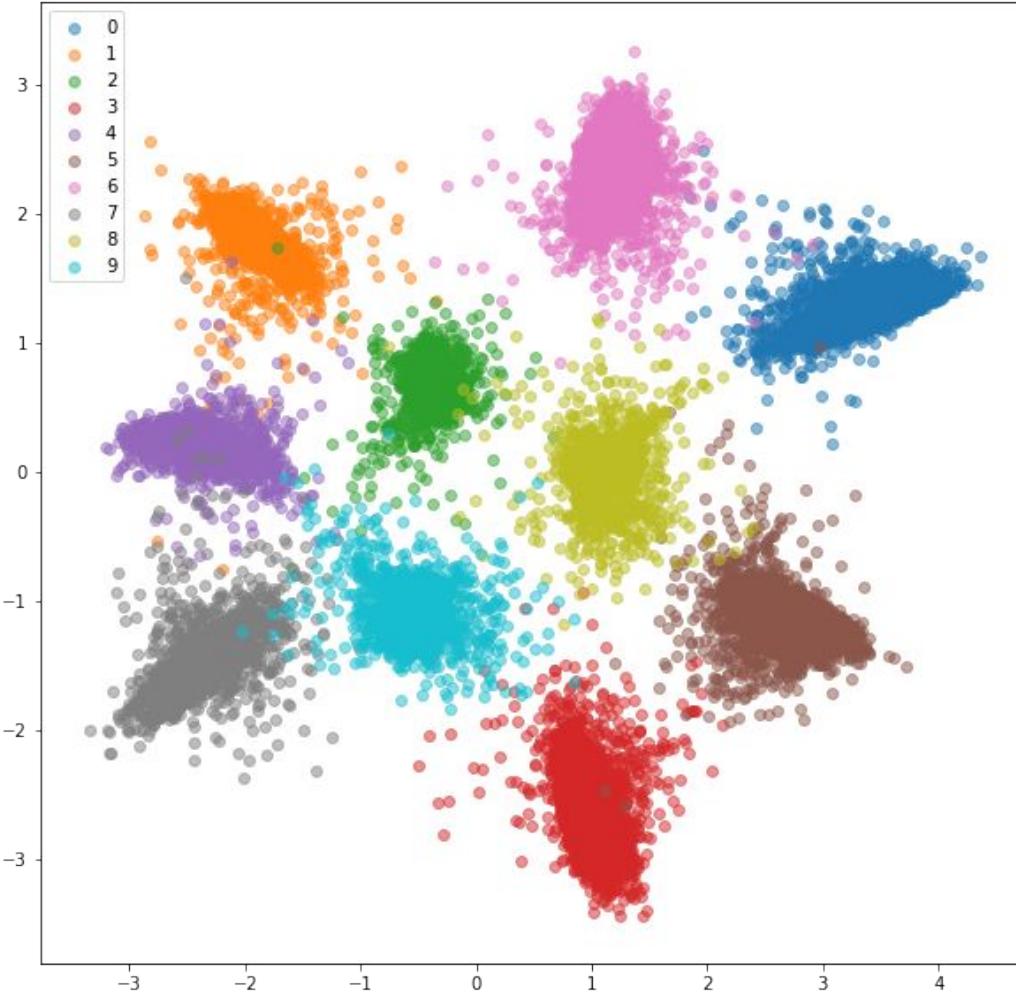
Преимущества:

1. Простая реализация
2. Можно получить компактный embedding

Недостатки:

1. Отсутствует «близость»

Triplet Loss



Преимущества:

1. + Гибкость формирования внутреннего представления

Недостатки:

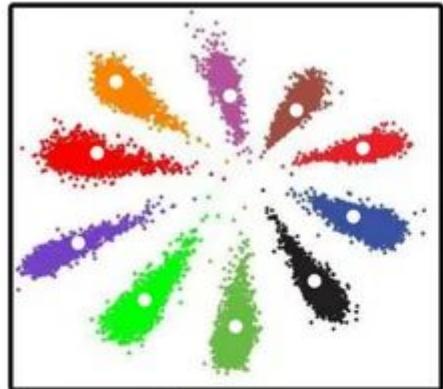
1. Частое попадание в локальные оптимумы
2. Необходимость подбора гиперпараметра
3. Формирование батчей

Методы с искусственным центром

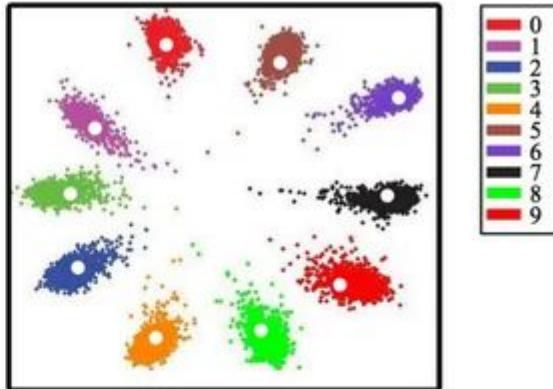
1. Center Loss
2. SphereFace
3. ArcFace
4. CosFace
5. Sub-center ArcFace



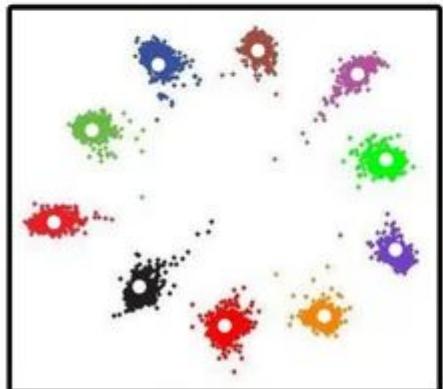
Center Loss



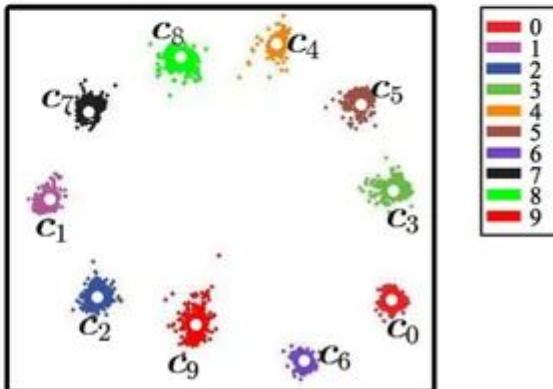
(a) $\lambda = 0.001$



(b) $\lambda = 0.01$



(c) $\lambda = 0.1$



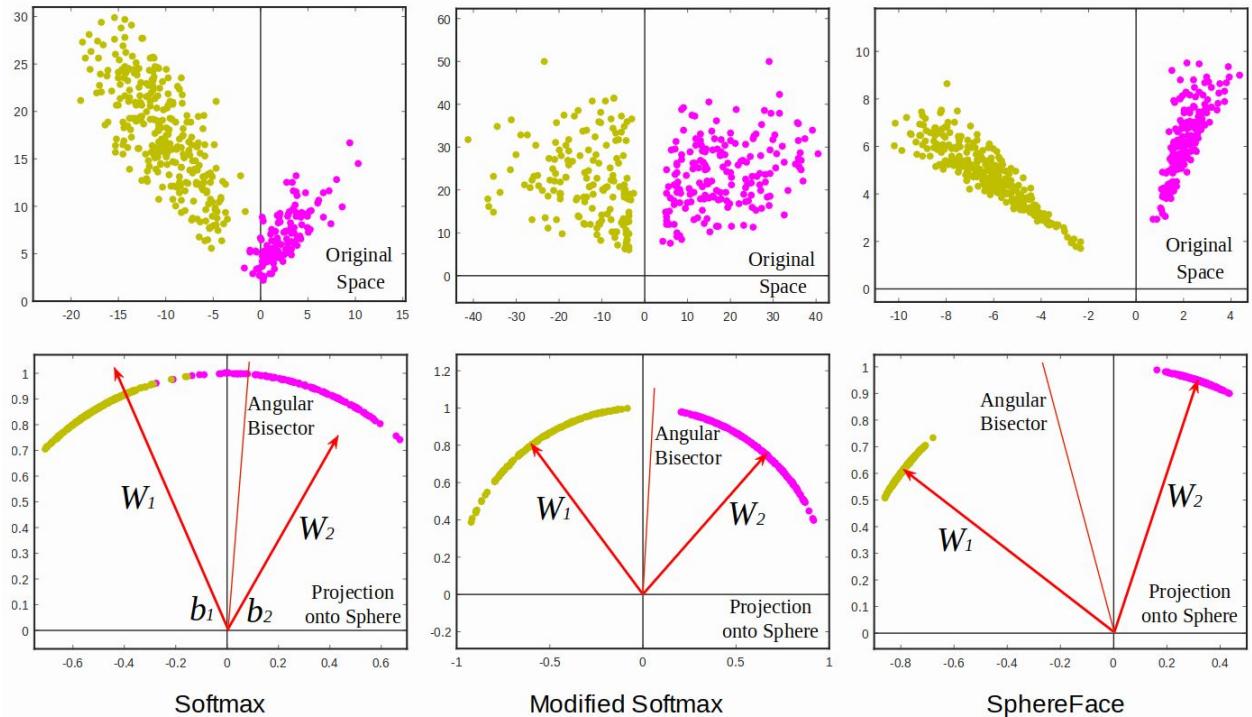
(d) $\lambda = 1$

$$\mathcal{L}_{\text{center}} = \mathcal{L}_{\text{softmax}} + \frac{\lambda}{2} \sum_{i=1}^N \|z_i - c_{y_i}\|_2^2$$

<https://arxiv.org/pdf/1707.07391.pdf>

SphereFace

$$\begin{aligned}
 \mathcal{L}_{\text{mod. softmax}} &= -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\{W_{y_i}^\top z_i + b_{y_i}\}}{\sum_{j=1}^m \exp\{W_j^\top z_i + b_j\}} \\
 &= -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\{\|W_{y_i}\| \|z_i\| \cos(\theta_{y_i,i}) + b_{y_i}\}}{\sum_{j=1}^m \exp\{\|z_i\| \cos(\theta_{j,i}) + b_j\}} \\
 &= -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\{\|z_i\| \cos(\theta_{y_i,i})\}}{\sum_{j=1}^m \exp\{\|z_i\| \cos(\theta_{j,i})\}}
 \end{aligned}$$

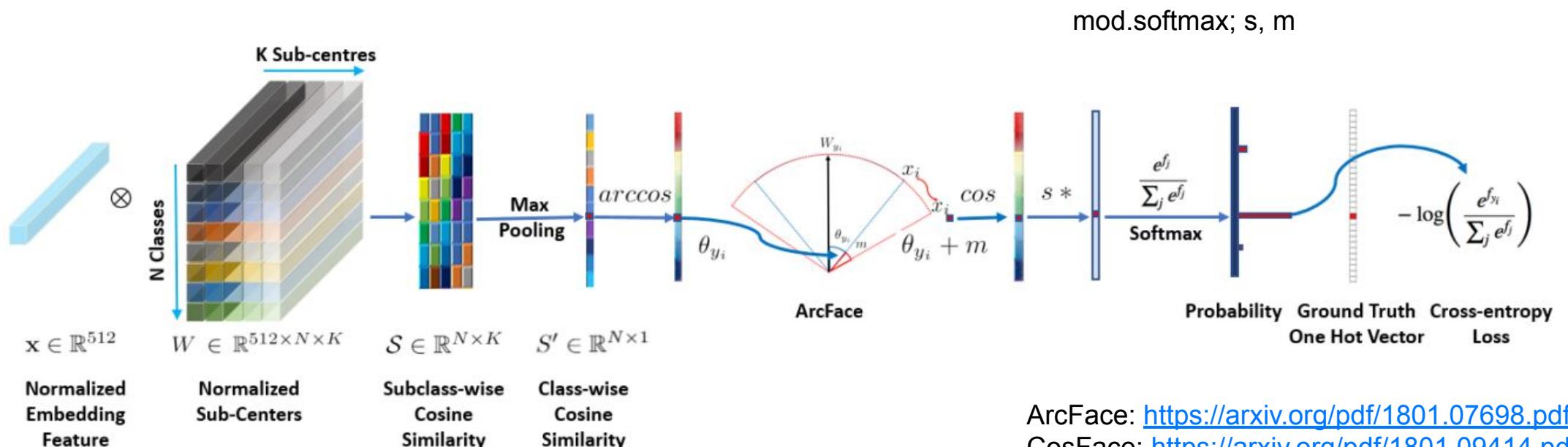


$$\mathcal{L}_{\text{SphereFace}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\{\|z_i\| \psi(\mu \theta_{y_i,i})\}}{\exp\{\|z_i\| \psi(\mu \theta_{y_i,i})\} + \sum_{j \neq y_i} \exp\{\|z_i\| \psi(\theta_{j,i})\}}$$

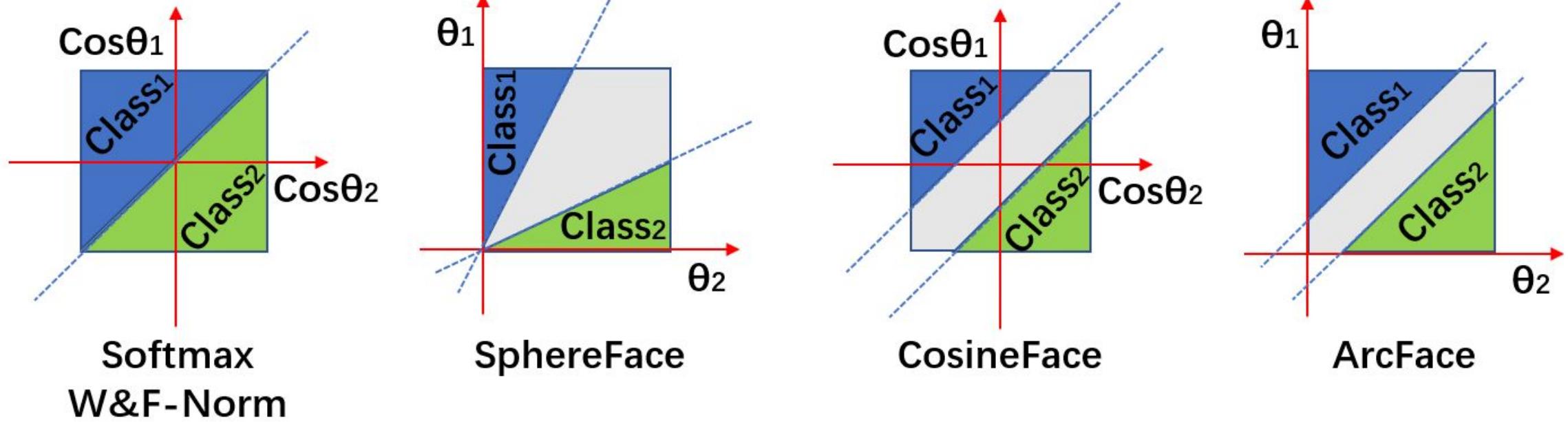
CosFace, ArcFace

$$\mathcal{L}_{\text{CosFace}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\{s(\cos(\theta_{y_i,i}) - m)\}}{\exp\{s(\cos(\theta_{y_i,i}) - m)\} + \sum_{j \neq y_i} \exp\{s \cos(\theta_{j,i})\}}$$

$$\mathcal{L}_{\text{ArcFace}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp\{s \cos(\theta_{y_i,i} + m)\}}{\exp\{s \cos(\theta_{y_i,i} + m)\} + \sum_{j \neq y_i} \exp\{s \cos(\theta_{j,i})\}}$$



Сравнение методов



Вопросы?



Спасибо
за внимание!