

Использование CNN для классификации изображений

Проблемы и их решения

Алексей Журавлев,

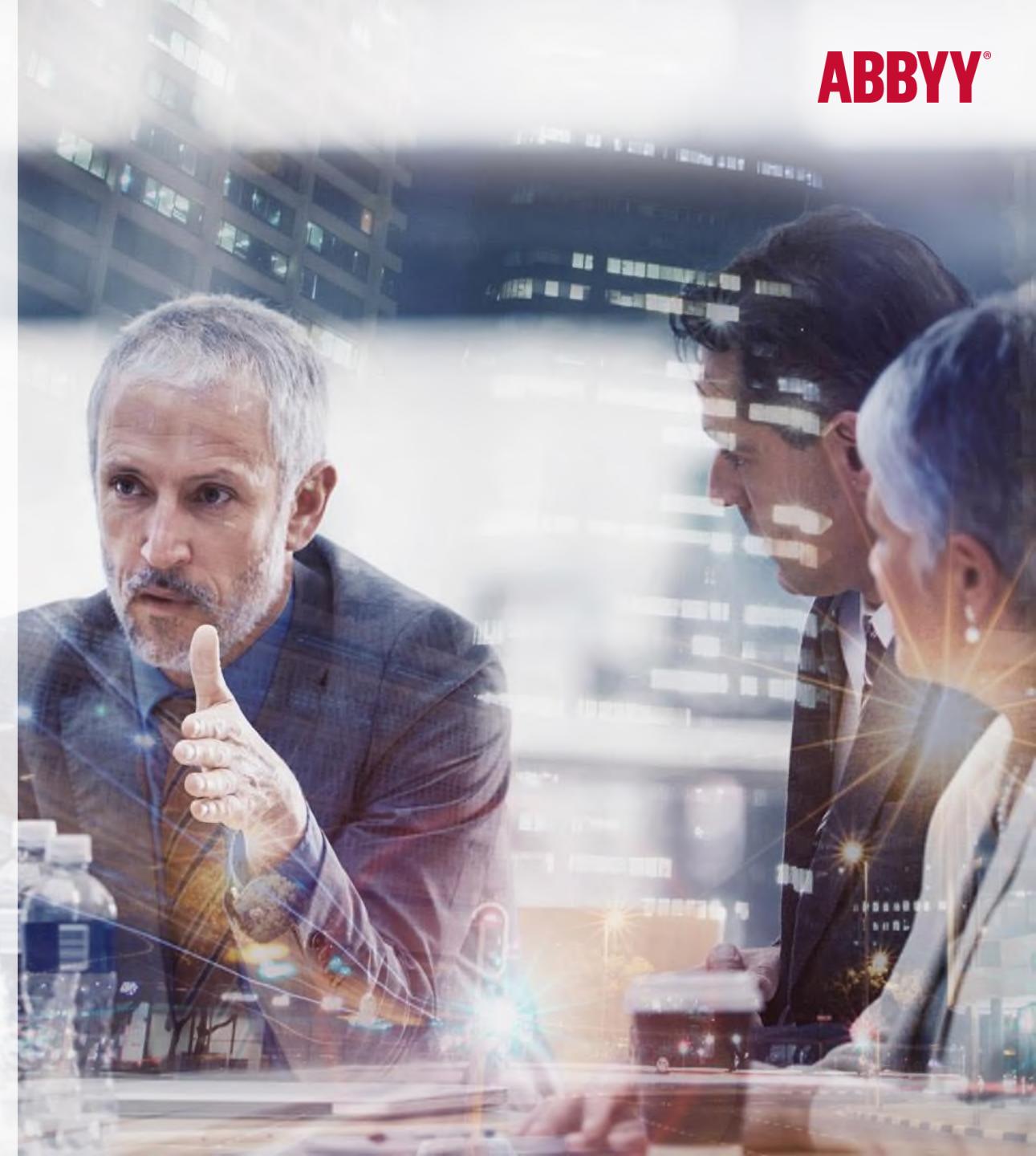
ABBYY OCR & ICR Group

1 декабря 2017



План лекции:

- Постановка задачи
- Популярные датасеты
- Решение задачи с помощью CNN
- Частный случай: распознавание текста
- Обучение «умных» представлений в постановке задачи классификации
 - Center Loss
 - Magnet Loss
- Semi-supervised Associative Learning
- Проблемы смешённой обучающей выборки, использование техник
 - Fine-Tuning
 - Domain Adaptation



Постановка задачи классификации изображений

- Есть набор изображений, и множество классов {1..K}, к которому относится (может относится) каждое из этих изображений
- На вход алгоритму: изображение (ч/б, серое или разложенное на несколько каналов)
- На выходе: класс этого изображения (или набор классов)

Постановка задачи классификации изображений



Horse?



Human?



Carriage?

Виды классификации



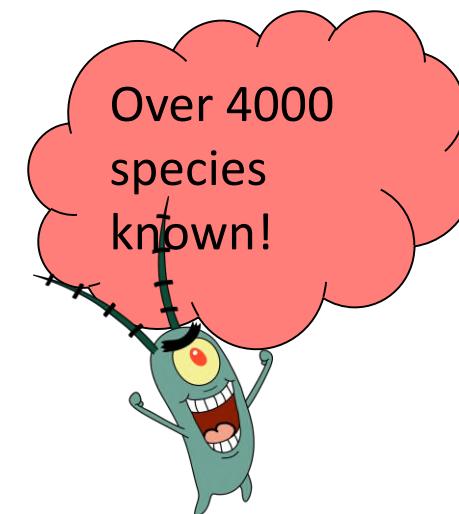
Многоклассовая:

- Какой это из S видов планктона?
- Ответ: $c \in \{1, \dots, S\}$



Бинарная:

- Это пешеход?
- $y \in \{0,1\}$, 0 – нет, 1 – да



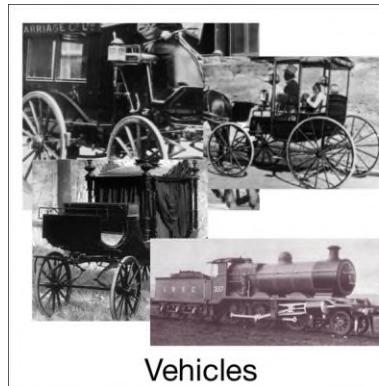
Правильный выбор целевого множества важен!



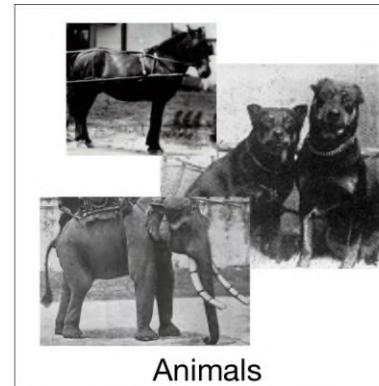
Horse?

Human?

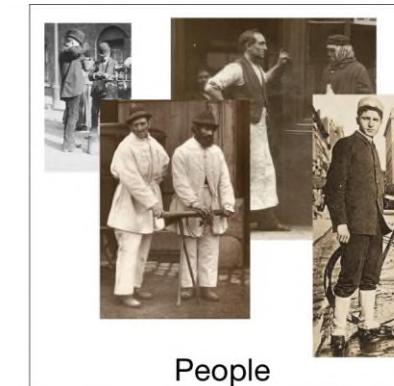
Carriage?



Vehicles

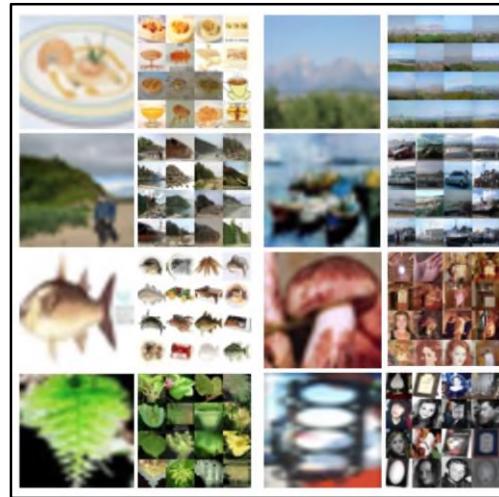


Animals



People

Крупнейшие датасеты по классификации изображений



80 million tiny images (2008)

<http://groups.csail.mit.edu/vision/TinyImages/>



CalTech 101 (2004)

[http://www.vision.caltech.edu/
Image_Datasets/Caltech101/](http://www.vision.caltech.edu/Image_Datasets/Caltech101/)



Pascal VOC (2005-2012)

<http://host.robots.ox.ac.uk/pascal/VOC/>



ImageNet (2005-2012)

<http://www.image-net.org/>

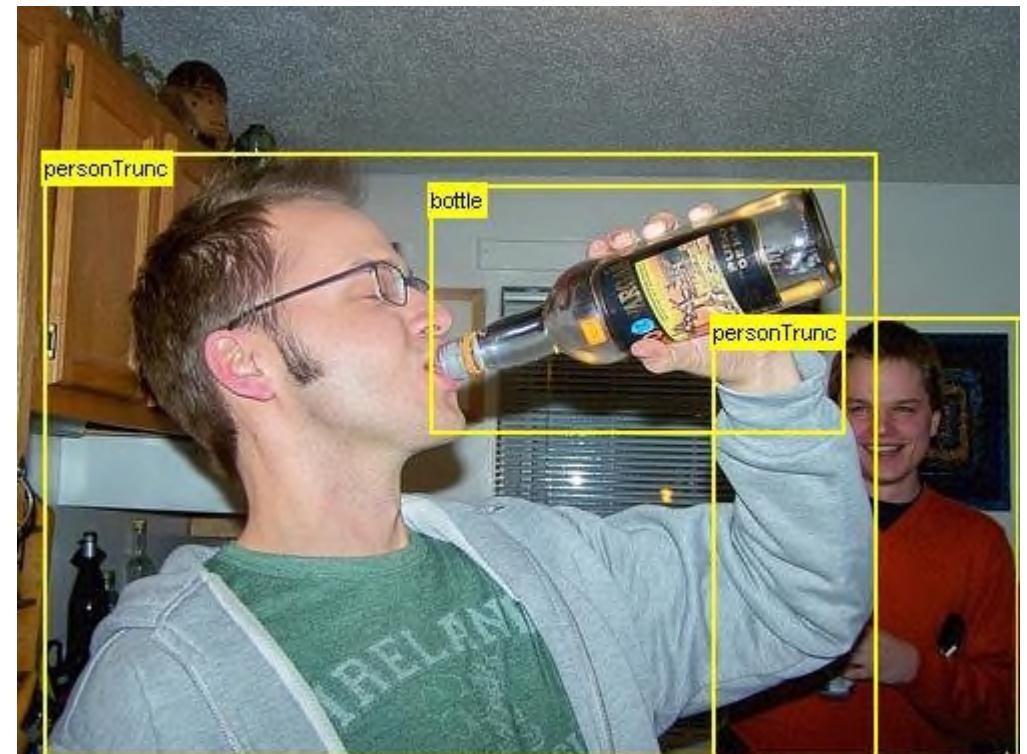
CalTech dataset

- [http://www.vision.caltech.edu/Image Datasets/Caltech101/](http://www.vision.caltech.edu/Image_Datasets/Caltech101/)
- Первая версия – 2004 год
- 101 класс объектов
- От 40 до 800 изображений на класс
- В последней версии число категорий увеличено до 256, число изображений – до 30607



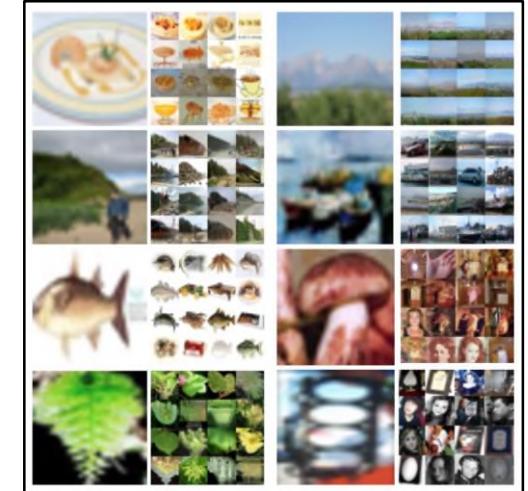
Pascal VOC dataset

- <http://host.robots.ox.ac.uk/pascal/VOC>
- Чуть более крупный датасет
- 2005: 4 класса, 2000 объектов
- 2012: 20 классов, 30000 объектов

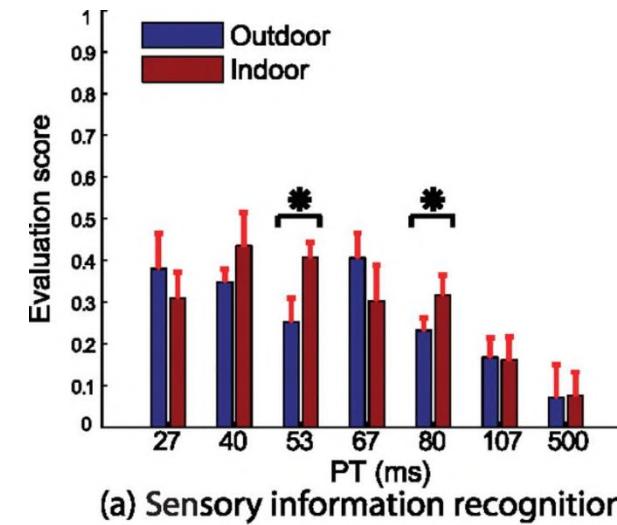
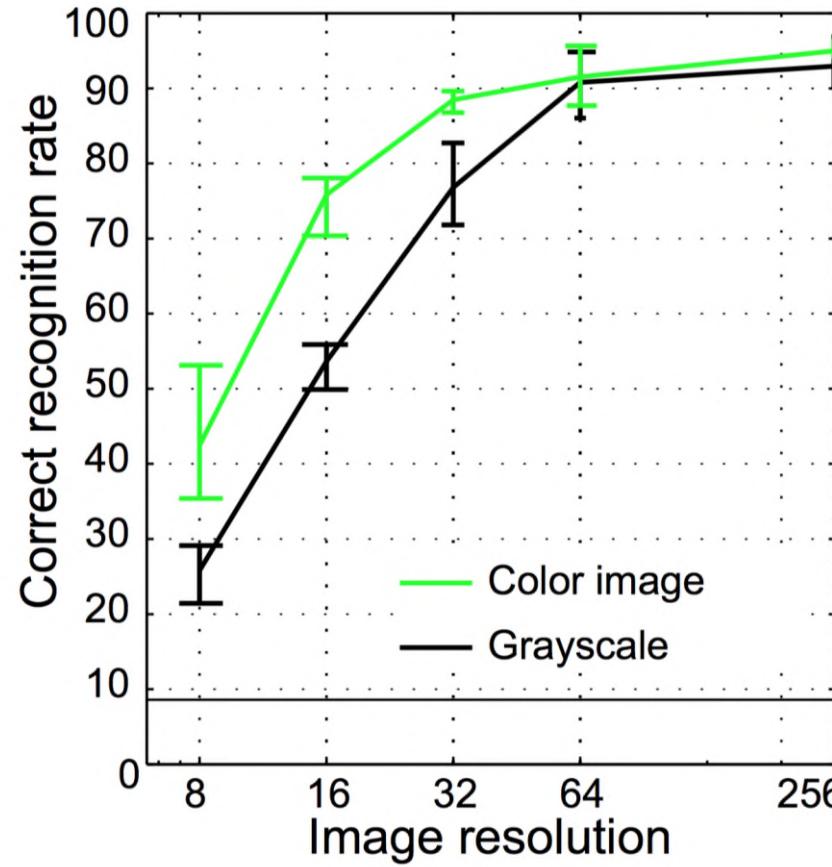


Tiny Images dataset

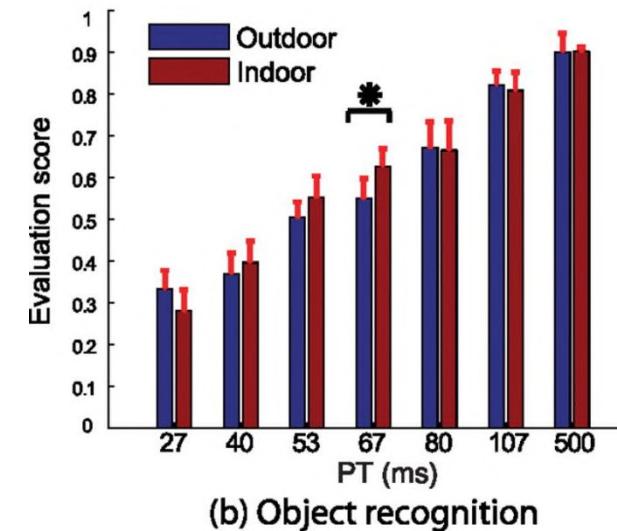
- Первая попытка создать датасет ~10 млн изображений
- ~50000 классов
- Исследовали способность человека к распознаванию изображений



Человеческие способности к распознаванию изображений



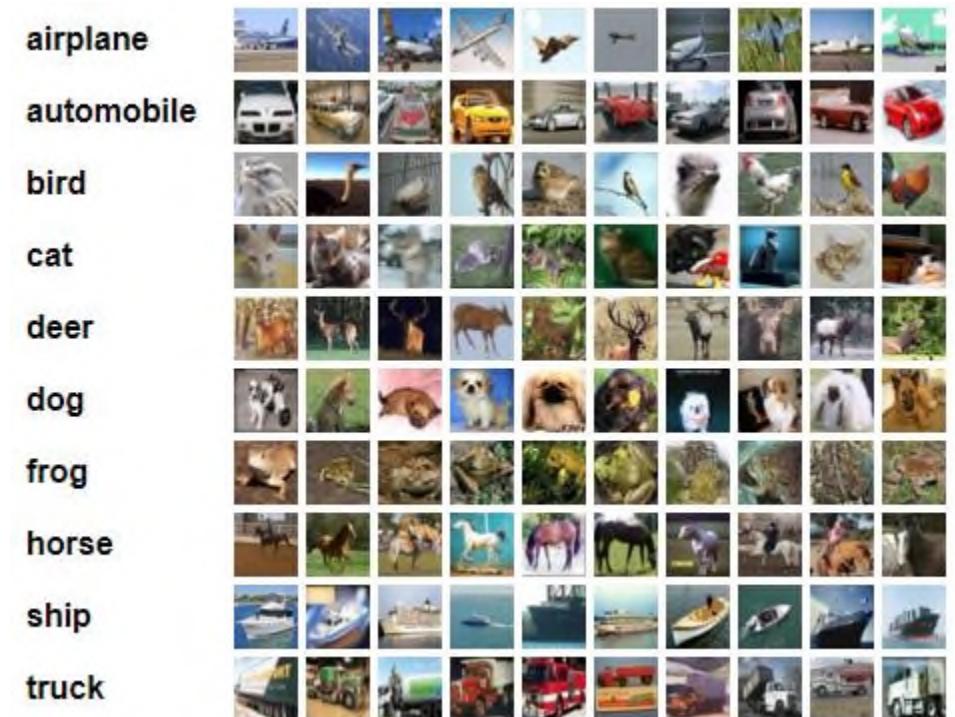
(a) Sensory information recognition



(b) Object recognition

CIFAR-10 и CIFAR-100

- Подмножества Tiny Images
- 10 классов по 6000 изображений
- 100 классов по 600 изображений
- CIFAR-100 разбит на 20 суперклассов



ImageNet

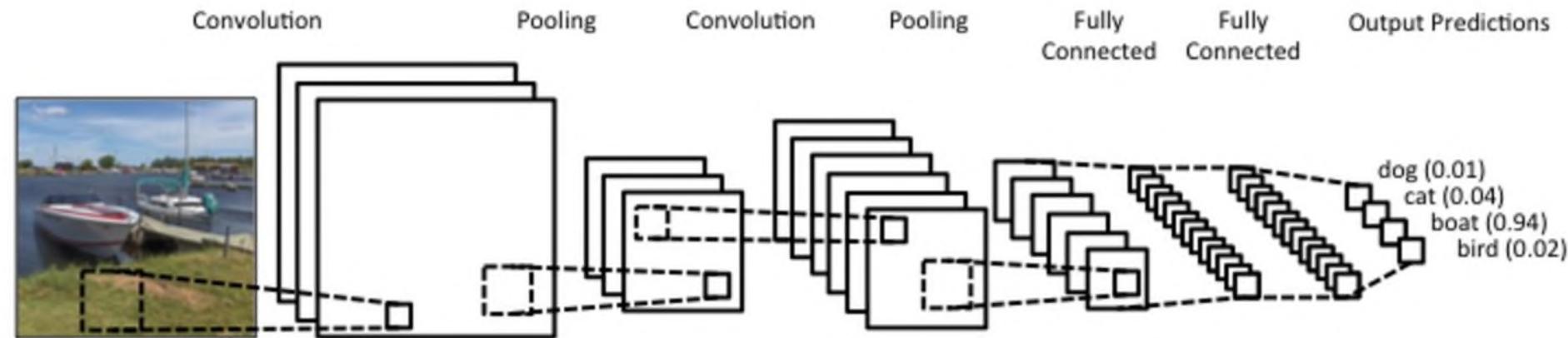
- ImageNet database: <http://image-net.org>
- Цель: собрать хотя бы по 1000 изображений на каждую из известных категорий
- Сейчас в датасете 14,197,122 изображения из 21,841 классов
- Также размечено ~1M bounding box-ов
- Собран web-краудсорсингом

Most vanilla: MNIST dataset



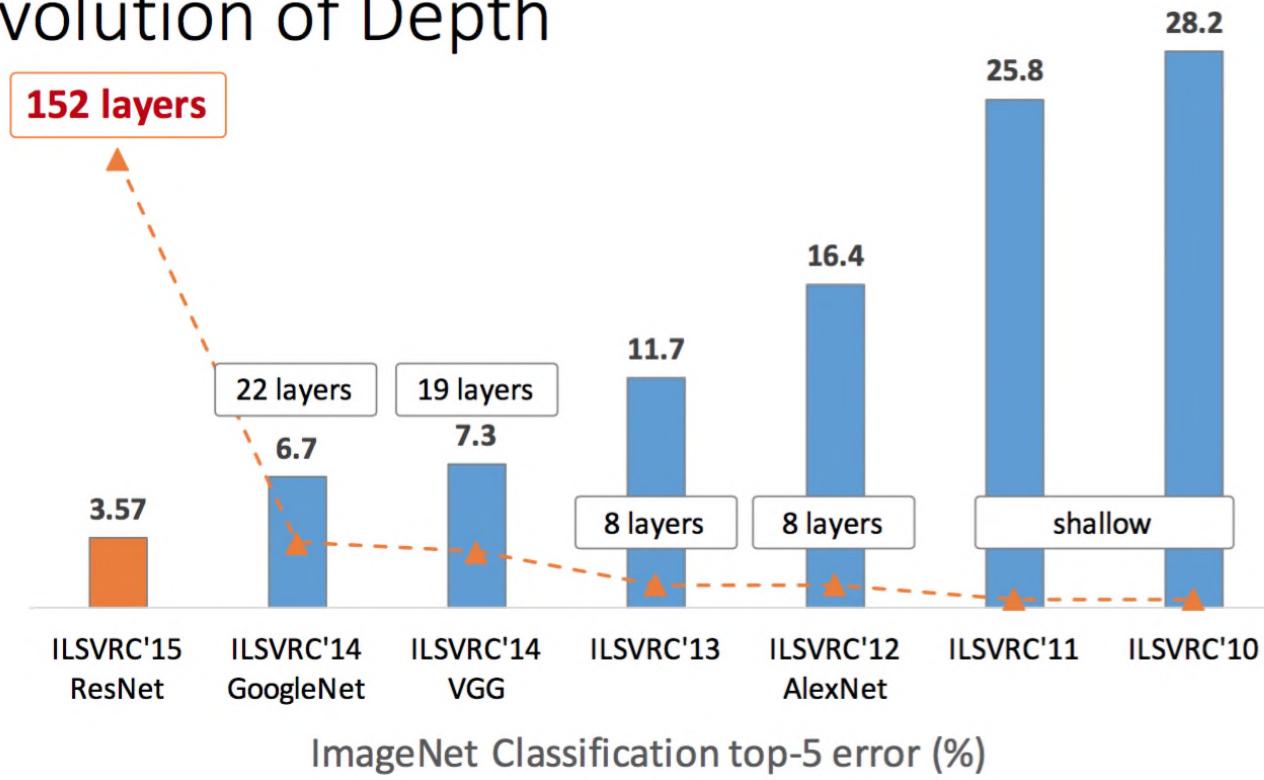
Решение задачи с помощью CNN

- Строим глубокую свёрточную сеть, на выходе которой – распределение вероятностей
- Выбор одного класса: softmax + crossentropy
- Выбор нескольких классов: sigmoid + multiclass crossentropy



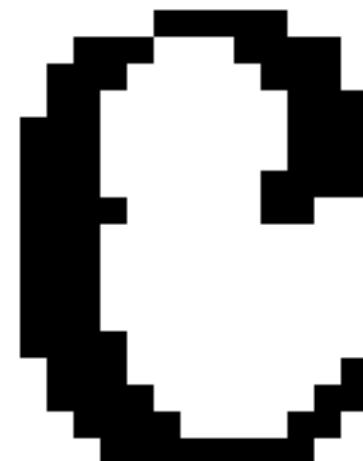
Эволюция нейросетей: ImageNet классификация

Revolution of Depth



Распознавание отдельных символов как пример задачи классификации

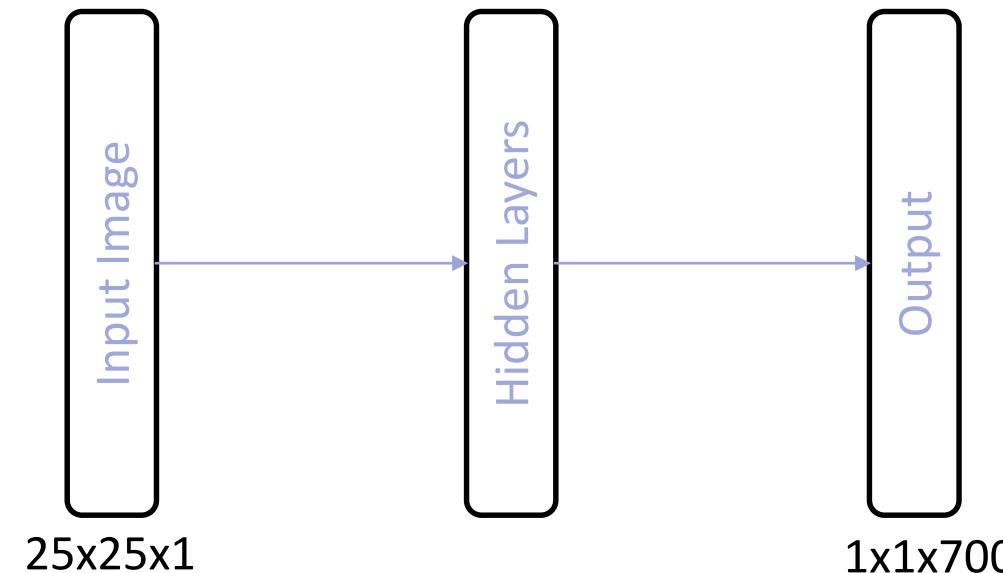
- Вход: изображение символа
- Выход: список вариантов распознавания с уверенностью каждого варианта



C	55
e	53
G	40
Q	30
O	28

Одна свёрточная нейронная сеть для распознавания всех символов

- Вход – изображение символа
- Выход – распределение вероятностей для всех символов алфавита
- Простым отсечением получим из него список вариантов распознавания
- Архитектурно мы строим обычный многоклассовый классификатор, получающий на вход сырое изображение
- Число выходов равно числу графем, которое нужно поддерживать (для английского – 130, для всех европейских символов – 700)



Одна свёрточная нейронная сеть для распознавания всех символов – плюсы и минусы

- Основной плюс – высокая точность классификатора, когда на вход приходят изображения символов
- Минусы:
- Что будет возвращать сеть, если на вход приходит не изображение символа, а мусор?
- Число выходов фиксировано, его нельзя изменить, даже если используется лишь некоторое подмножество европейских символов (английский)
- При высоком качестве скорость достаточно низкая (ниже, чем у обычного алгоритма распознавания, в европейском), на счету микросекунды
- Как выставлять уверенность варианта?
- Нужный независимый классификатор мусора (его правда тоже можно делать с помощью сети)



Дифференциальный классификатор из свёрточной нейронной сети

- Одну сеть можно использовать для дифференциальной сортировки вариантов
- Классификация мусора не имеет отношения к этой задаче
- Сортировка вариантов производится по отклику сети
- Сети известно про все варианты распознавания, поэтому она всегда может произвести полную классификацию
- Можно добиться крайне высокого качества, иногда даже выгодно разменяв его на скорость
- Большая часть старых минусов остаётся в силе
- Сеть работает сильно «вхолостую» из-за отсутствия маргинализации
- При этом цель дифференциального уровня – как можно сильнее улучшить качество метода распознавания, как можно слабее замедлив его

Дифференциальный классификатор из нескольких простых сетей

- Мотивация: чем меньше символов сети нужно уметь сравнивать, тем проще можно сделать её архитектуру без потери общего качества
- Идея: обычно с приходом на диф. уровень путаются определённые группы символов, а не каждый символ с каждым
- Почему бы не выделить набор особых «дифференциальных множеств», в рамках которых символы сильно путаются
- На каждое из множеств можно обучить отдельную нейронную сеть
- При классификации выбирать одну наиболее подходящую сеть и производить её запуск – ту, которая способна сравнить как можно больше верхних вариантов распознавания

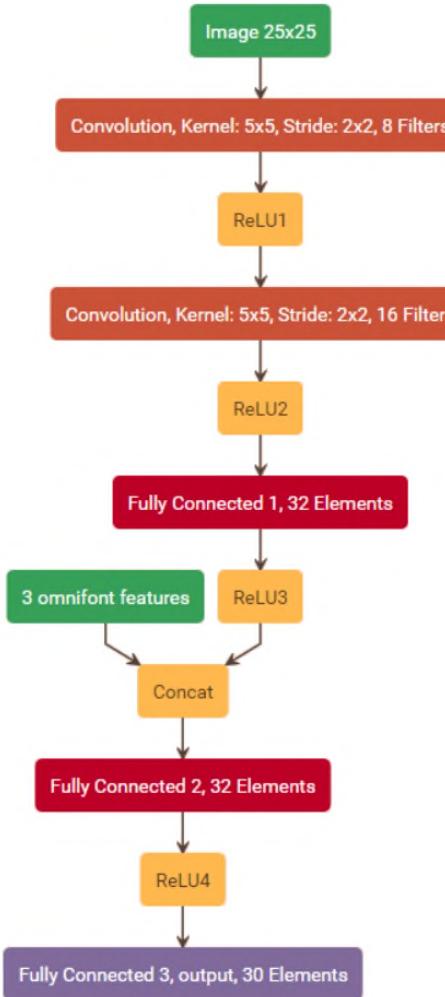
Примеры дифференциальных множеств

ばばぼぼほけ社はげ世s任せ
お旧aた女々な七力上だt也
あむ色ち去きまをも右屯电
か丸九止がガ#キひびん*ホ
る巧是わねれ打枚札礼吾ぎ
さ倉含寺友安宏芝查支左仑
者李妾吉音

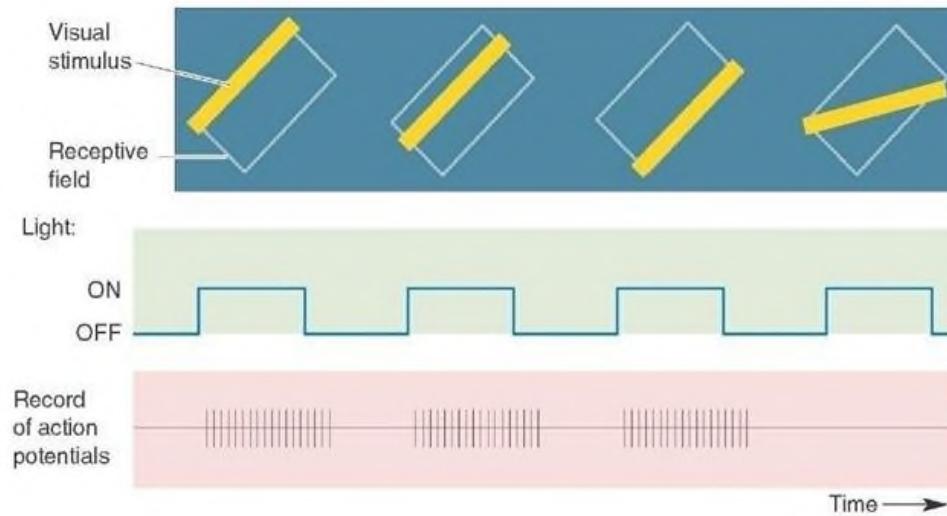
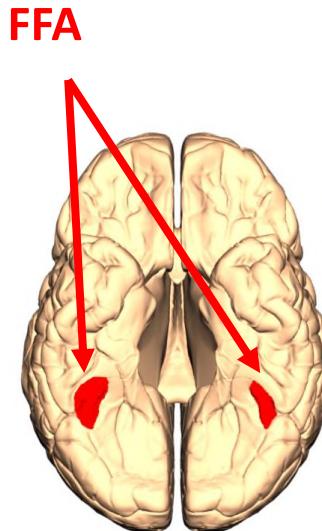
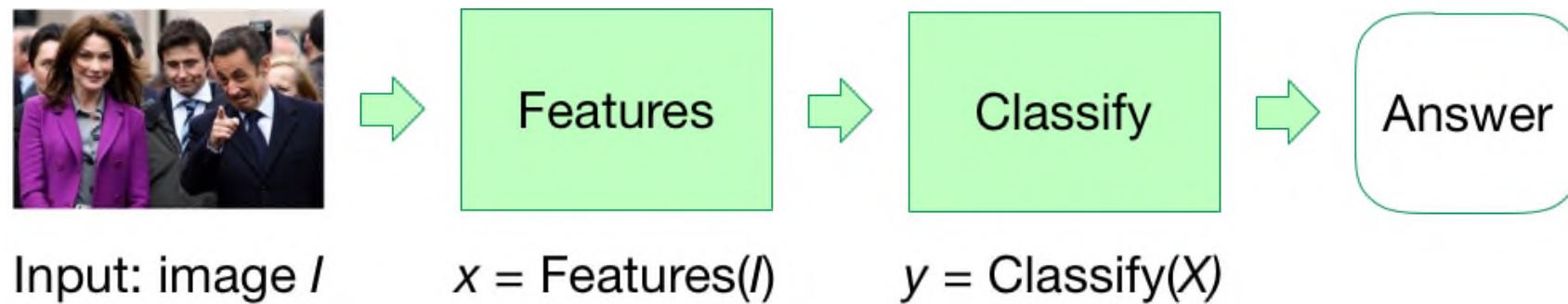
間問聞閒閻開閑閤閻閻
閨閻閭閨閏閨閨閨閨
國園圓圓團圓函面圓圓
困圈回因而圓園圓團圓
商西周·川苗思兩月闲再角
甬向圖間圓巨自圓用片貝戶
甩

Пример архитектуры, применяемой для распознавания символов, дифференциальный уровень

- Схема второго уровня с дифференциальными множествами применялась для европейских языков
- Обучено 30 сетей, каждая из которых различает множество из 30 графем
- Время запуска одной сети ~25мкс
- Среднее качество всех сетей на тестовой базе символов: 99,3%
- Количество ошибок алгоритма распознавания в среднем уменьшено на 10,5%
- Схема стала полной заменой структурному дифференциальному уровню, обогнав его по скорости и по качеству



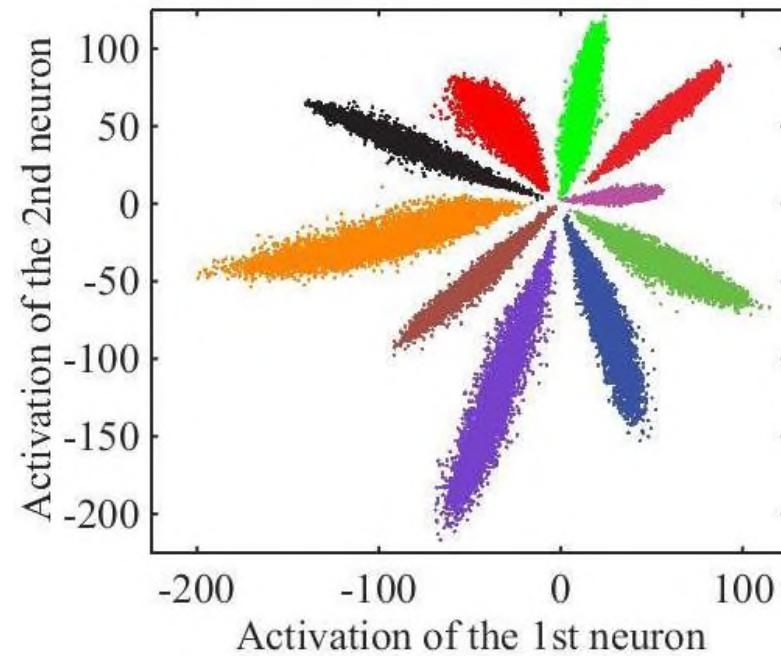
Стандартный поток классификации



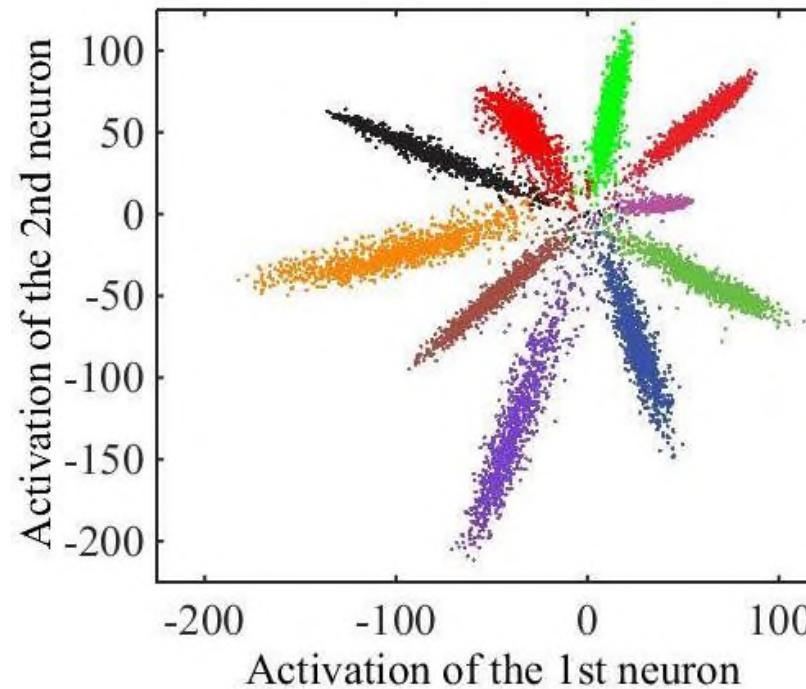
Задача классификации как обучение представления методами Deep Learning

- Плюсы:
 - Не приходится выбирать признаки вручную
 - В рамках поставленной задачи и выбранного множества классов полученные признаки будут точно лучшими (при большом размере выборки!)
- Минусы
 - Переобучение
 - Отсутствие обобщающей способности на классы, не встречающиеся в целевом множестве меток
 - Слабая метрическая интерпретируемость

Проблема softmax: обучение радиальных признаков



MNIST Train



MNIST Test

Одно из возможных решений: Center Loss (2015)

- Частично удалось решить проблему с помощью использования Center Loss

$$\mathcal{L}_C = \frac{1}{2} \sum_{i=1}^m \| \mathbf{x}_i - \mathbf{c}_{y_i} \|_2^2$$

- Физический смысл: ищем центр масс элементов каждого класса в пространстве признаков, штрафуем элементы за удалённость от своего центра
- Минусы:
- Для полного подсчёта и прорасывания градиентов нужно запустить сеть на всей обучающей выборке
- Неустойчивость к шуму: далёкие элементы могут сильно смешать центр кластера

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= - \sum_{i=1}^m \log \frac{e^{W_{y_i}^T \mathbf{x}_i + b_{y_i}}}{\sum_{j=1}^n e^{W_j^T \mathbf{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^m \| \mathbf{x}_i - \mathbf{c}_{y_i} \|_2^2\end{aligned}$$

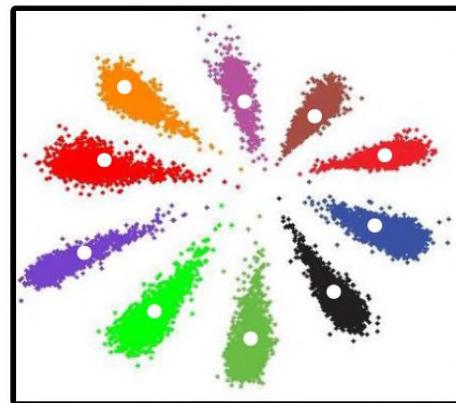
Выделение признаков: Center Loss

- Для прорасывания градиента будем использовать minibatch
- Обновление центра будем производить также итеративно по минибатчу
- Используем дополнительный коэффициент α – скорость сходимости центра на основе минибатча, контролируем влияние выбросов

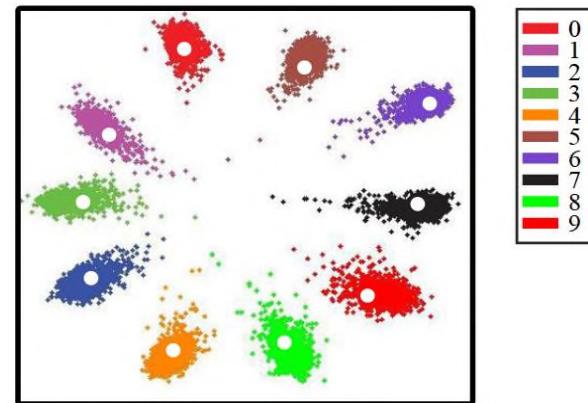
$$\frac{\partial \mathcal{L}_C}{\partial \mathbf{x}_i} = \mathbf{x}_i - \mathbf{c}_{y_i}$$
$$\Delta \mathbf{c}_j = \frac{\sum_{i=1}^m \delta(y_i = j) \cdot (\mathbf{c}_j - \mathbf{x}_i)}{1 + \sum_{i=1}^m \delta(y_i = j)}$$

$$\mathbf{c}_j^{t+1} = \mathbf{c}_j^t - \alpha \cdot \Delta \mathbf{c}_j^t.$$

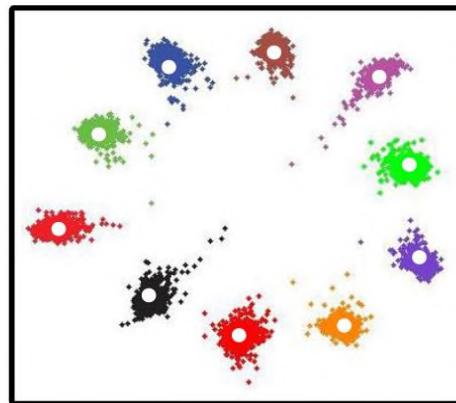
Center Loss – визуализация



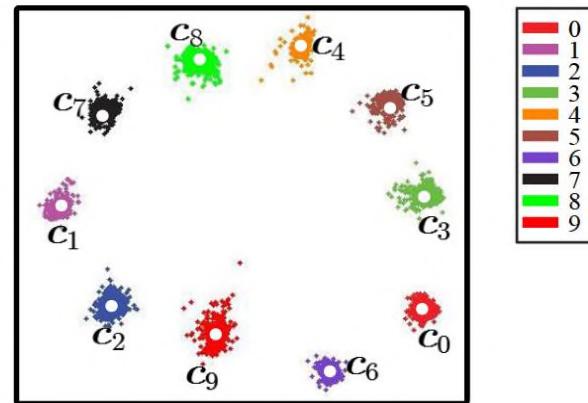
(a) $\lambda = 0.001$



(b) $\lambda = 0.01$



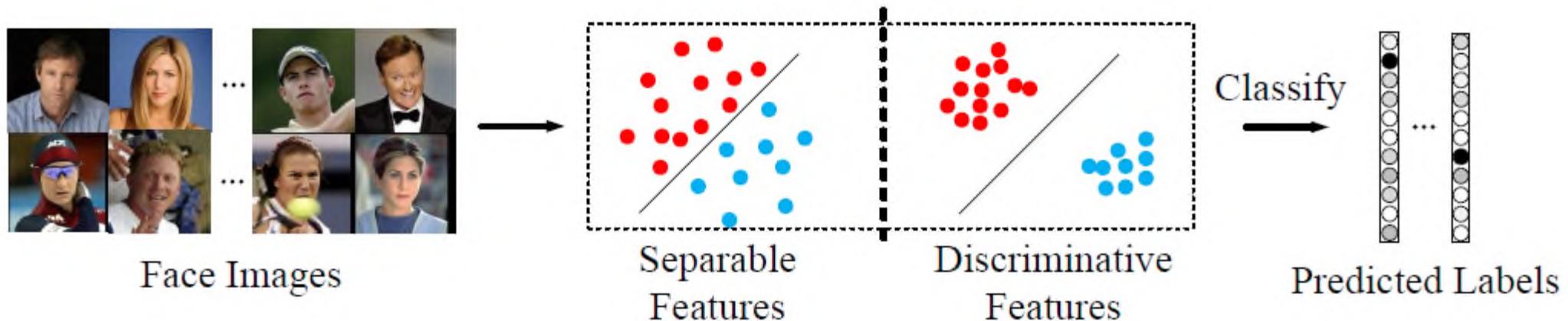
(c) $\lambda = 0.1$



(d) $\lambda = 1$

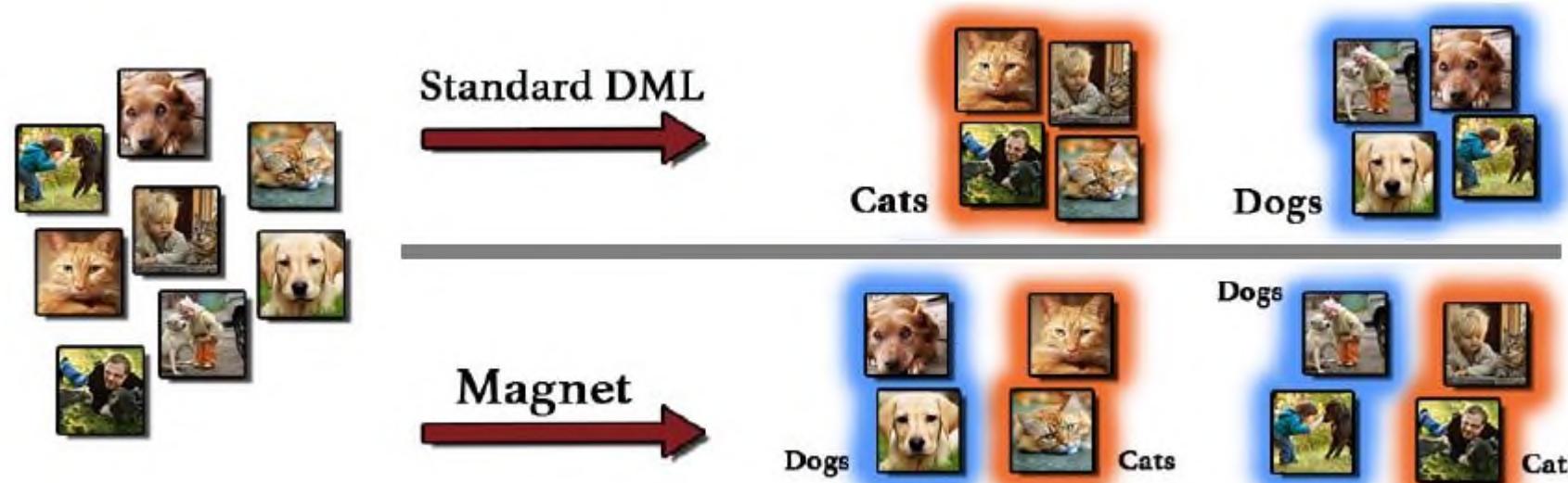
Center Loss: на практике

- Оригинальная статья: A Discriminative Feature Learning Approach for Deep Face Recognition (изначально применялся для лиц)
- Позволил увеличить обобщающую способность сетей на отсутствующие в выборке лица
- На символах показал похожий результат

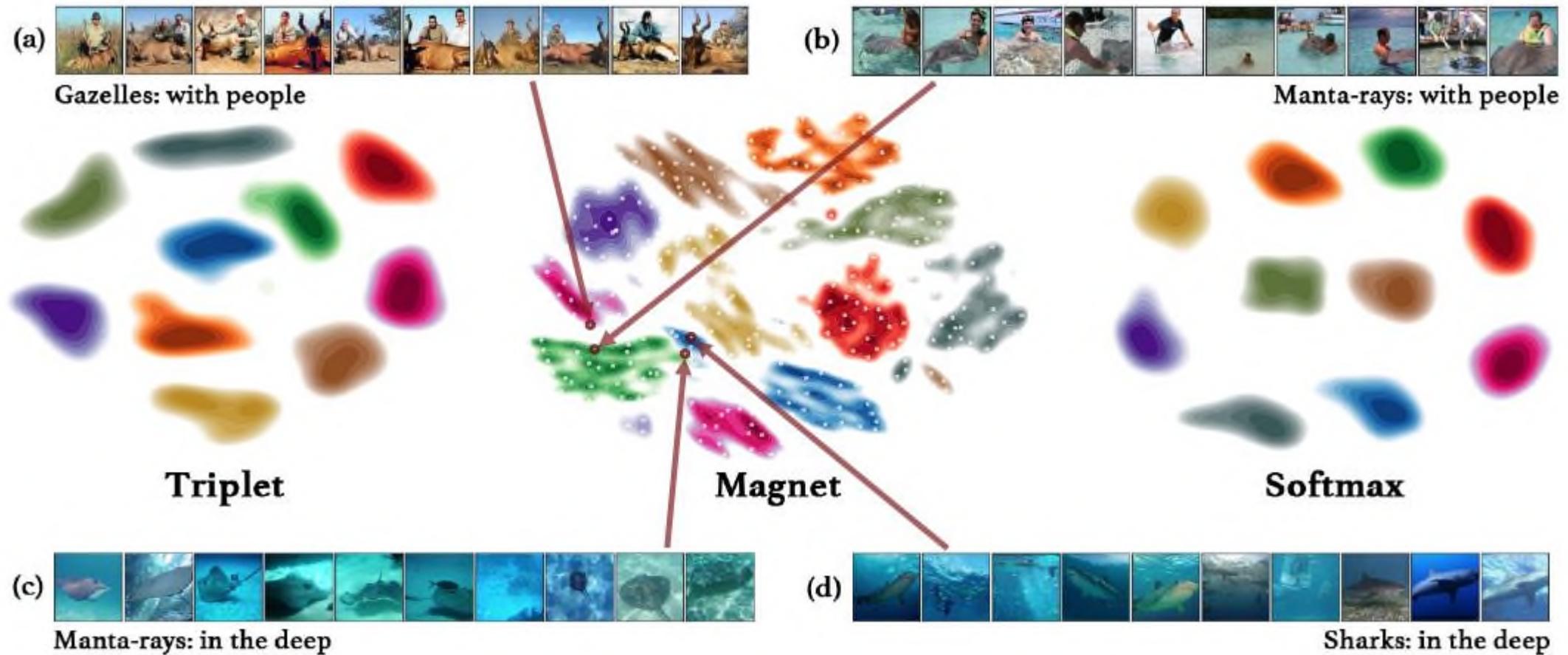


Что делать, если внутри класса возможна сильная вариативность?

- State of the art в таком случае считается Magnet Loss
- Оригинальная статья: Metric Learning With Adaptive Density Discrimination (2016)



Magnet Loss представления



Magnet Loss: математика

$$\mathcal{I}_1^c, \dots, \mathcal{I}_K^c = \arg \min_{I_1^c, \dots, I_K^c} \sum_{k=1}^K \sum_{\mathbf{r} \in I_k^c} \|\mathbf{r} - \boldsymbol{\mu}_k^c\|_2^2 ,$$

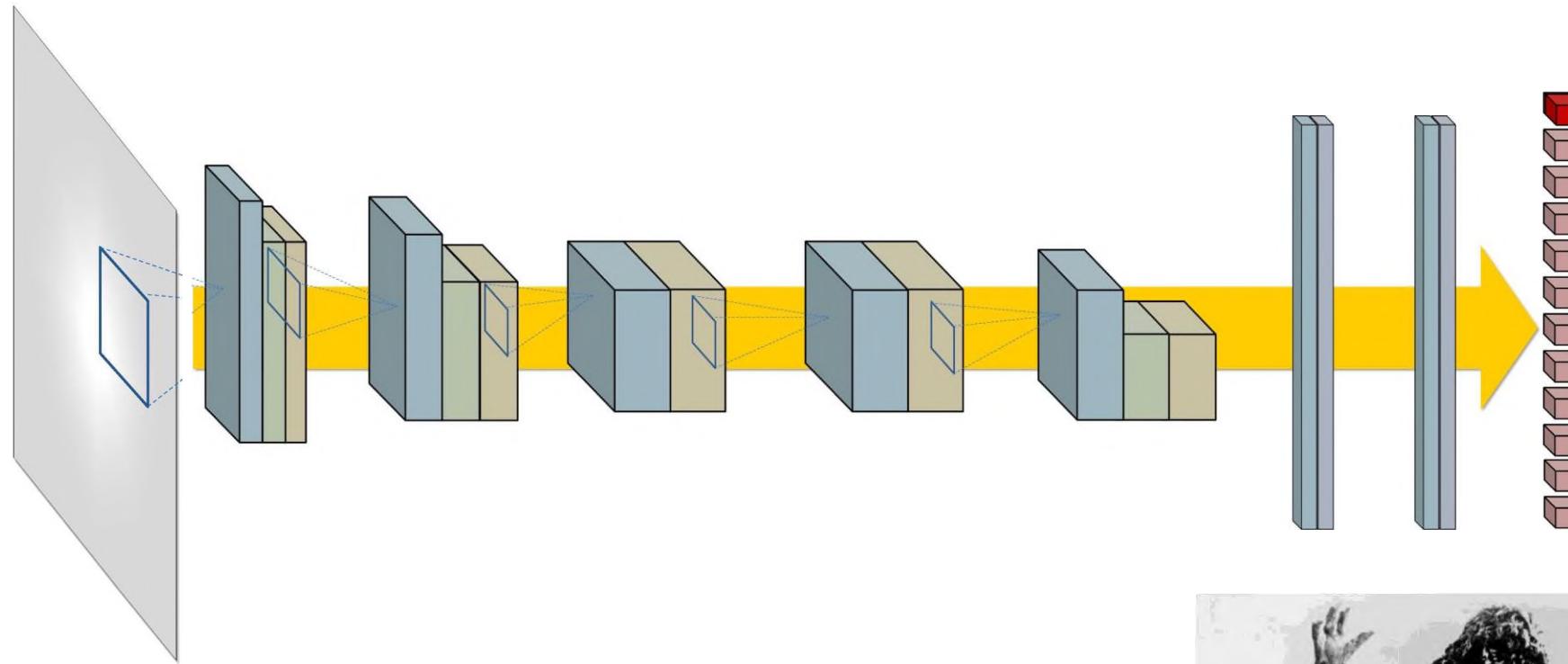
$$\boldsymbol{\mu}_k^c = \frac{1}{|I_k^c|} \sum_{\mathbf{r} \in I_k^c} \mathbf{r} .$$

$$\mathcal{L}(\boldsymbol{\Theta}) = \frac{1}{N} \sum_{n=1}^N \left\{ -\log \frac{e^{-\frac{1}{2\sigma^2} \|\mathbf{r}_n - \boldsymbol{\mu}(\mathbf{r}_n)\|_2^2 - \alpha}}{\sum_{c \neq C(\mathbf{r}_n)} \sum_{k=1}^K e^{-\frac{1}{2\sigma^2} \|\mathbf{r}_n - \boldsymbol{\mu}_k^c\|_2^2}} \right\}_+ +$$

Обучение представлений с помощью классификации: выводы

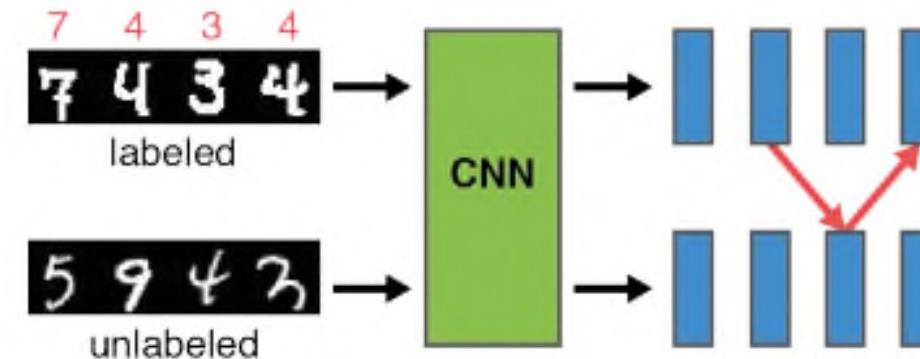
- В контексте отсутствия необходимости решать параллельно задачу классификации чаще всего используют традиционные Triplet Loss, Quadruplet Loss, Angular Loss и т.д.
- При многоклассовой классификации одно из двух:
 - Небольшой внутриклассовый разброс – Center Loss
 - Большой внутриклассовый разброс + желание его учесть – Magnet Loss
- На деле куда больше, чем двух: Range Loss и т.д.

Проблема: голодные нейросети (хотят много данных)



Semi-supervised learning – решение когда мало лишь размеченных данных

- Один из state of the art в этой сфере: Associative Learning
- Оригинальная статья: Learning by Association – A versatile semi-supervised training method for neural networks (2017)



Математика: Associative Loss

$$A_i := \phi(\mathbf{x}_i^s), B_j := \phi(\mathbf{x}_j^t)$$

$$M_{ij} = \langle A_i, B_j \rangle$$

$$P_{ij}^{ab} = \mathbb{P}(B_j|A_i) := \frac{\exp(M_{ij})}{\sum_{j'} \exp(M_{ij'})}$$

$$P_{ij}^{aba} := (P^{ab} P^{ba})_{ij}$$

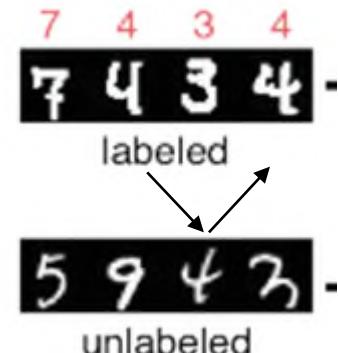
$$\mathcal{L}_{\text{assoc}} = \beta_1 \mathcal{L}_{\text{walker}} + \beta_2 \mathcal{L}_{\text{visit}}$$

$$T_{ij} := \begin{cases} 1/|A_i| & \text{class}(A_i) = \text{class}(A_j) \\ 0 & \text{else} \end{cases}$$

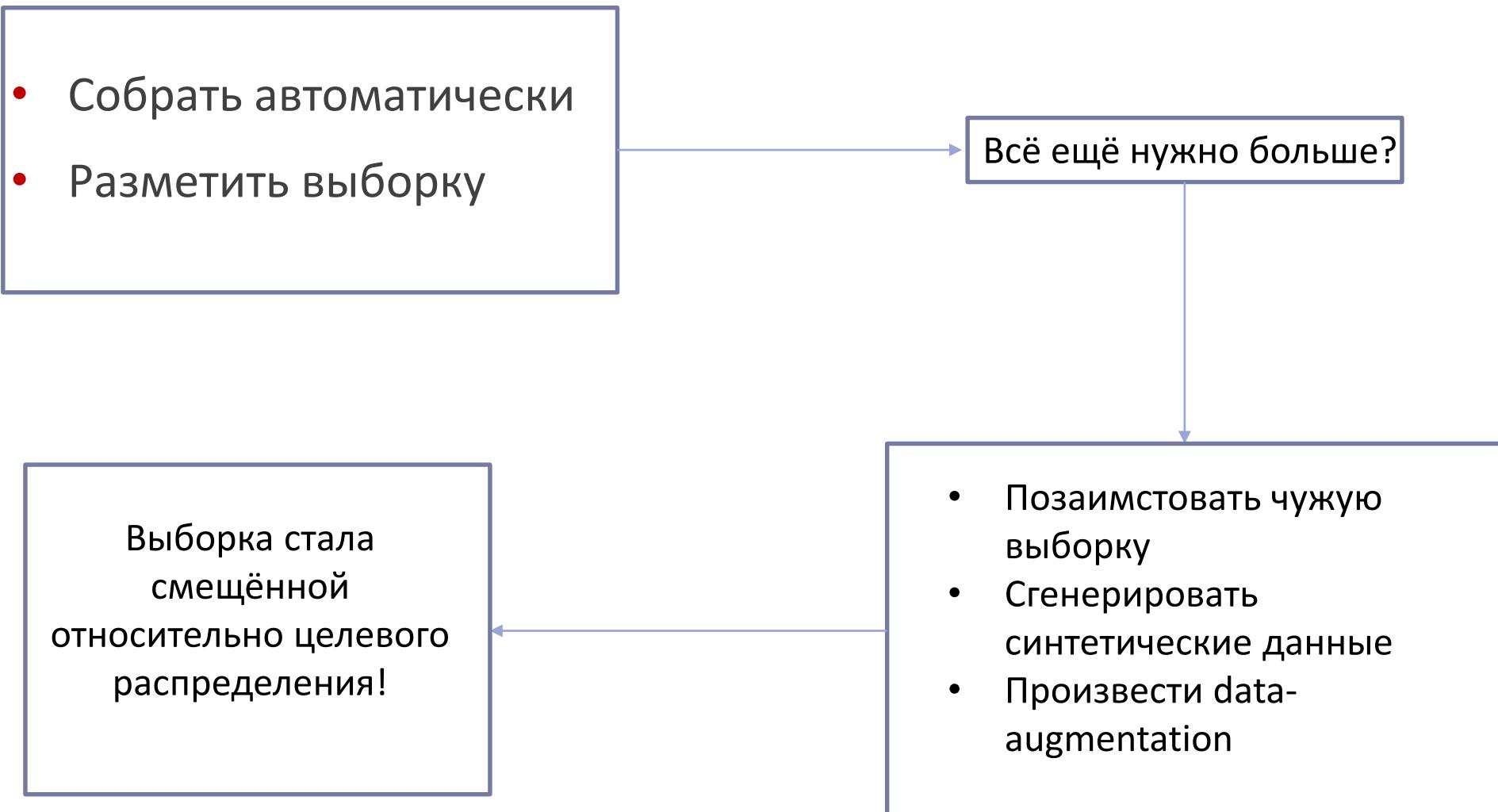
$$\mathcal{L}_{\text{walker}} := H(T, P^{aba})$$

$$P_j^{\text{visit}} := \sum_{\mathbf{x}_i \in \mathcal{D}_s} P_{ij}^{ab}, \quad V_j := \frac{1}{|B|}.$$

$$\mathcal{L}_{\text{visit}} := H(V, P^{\text{visit}})$$



Проблема: Где взять данные?

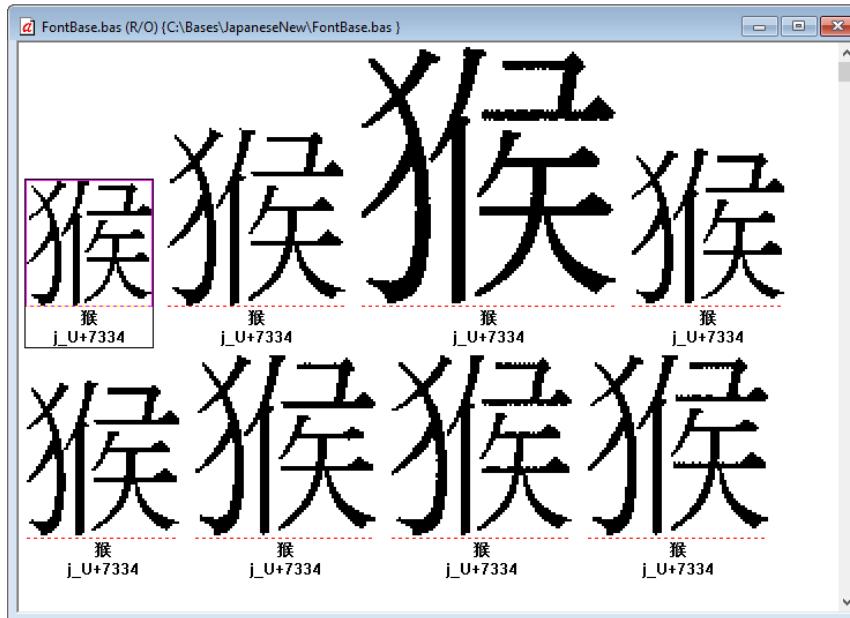


Пример: картинки из интернета -> сфотографированные объекты

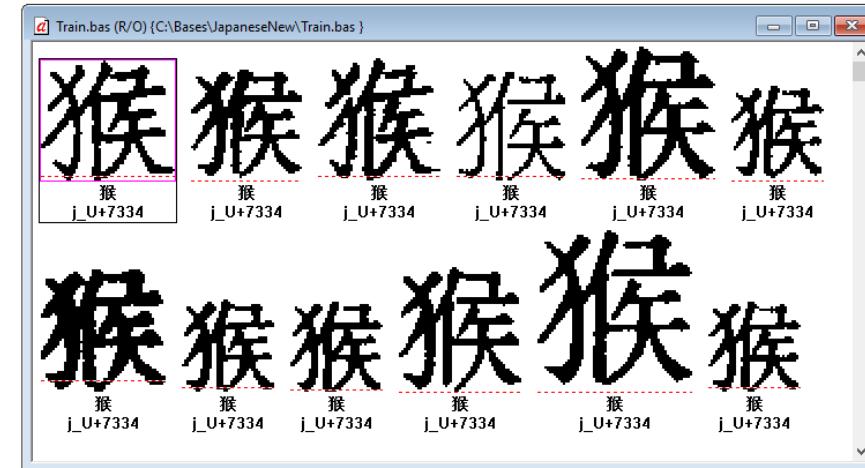


Пример: различные изображения символов

Синтетические (рендеринг шрифтов)

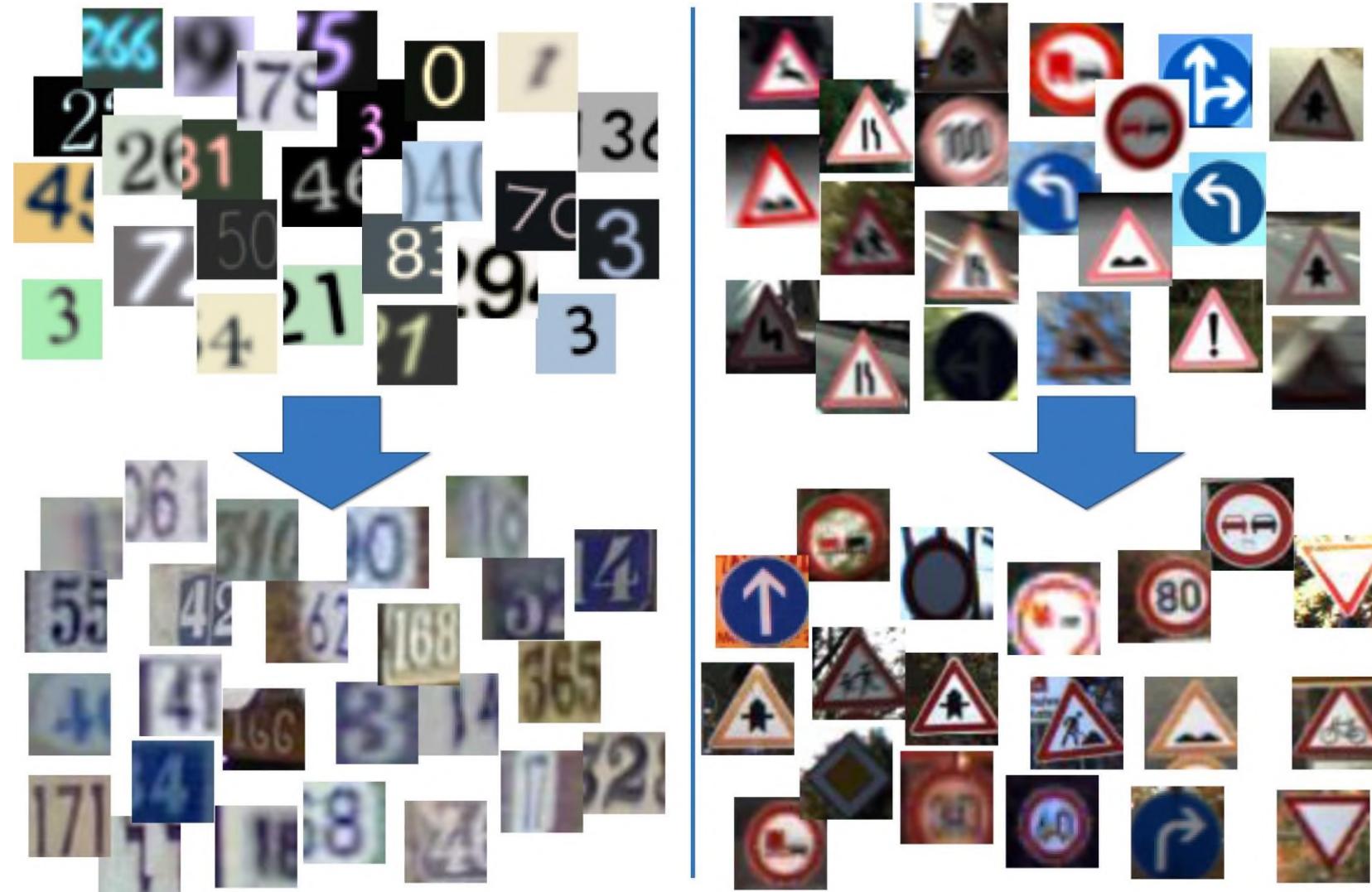


Реальные (из документов)



Полученные с помощью Data Augmentation

Пример: синтетика(возможно искажённая) -> реальные
объекты



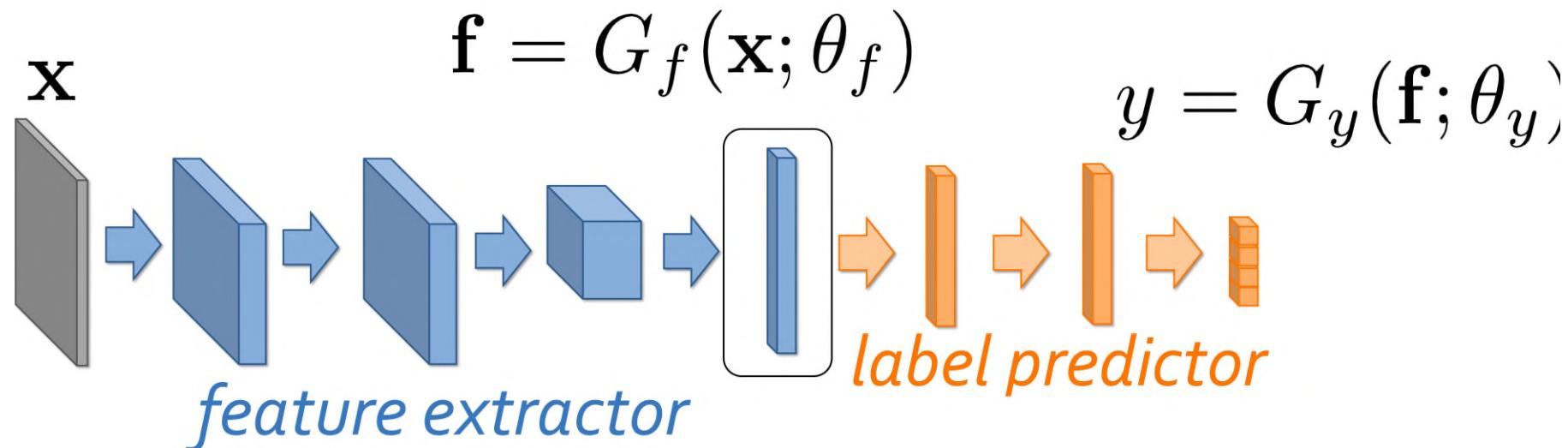
Выводы и цели

- Есть огромное количество **размеченных** данных из исходного домена (например синтетические картинки)
- Также есть множество **неразмеченных** данных из целевого домена
- **Итоговая цель:** обучить с помощью этого всего глубокую нейросеть, которая будет хорошо работать на целевом домене

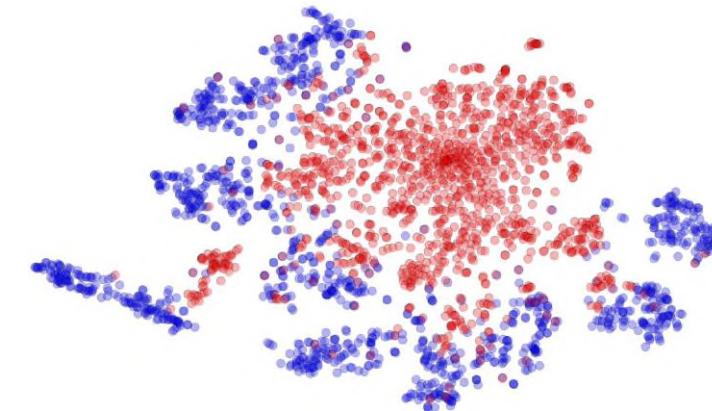
Проблема настройки обученной модели на целевые данные

- Fine-Tuning
 - Настройка модели на целевой выборке (даже небольшой) после обучения на исходной
 - Чаще всего обучаются при этом несколько последних слоёв
- Domain Adaptation
 - Обучение производится совместно на исходных и целевых данных
 - Признаки должны быть «независимы» от домена: иметь одинаковое распределение

Архитектура нейронной сети



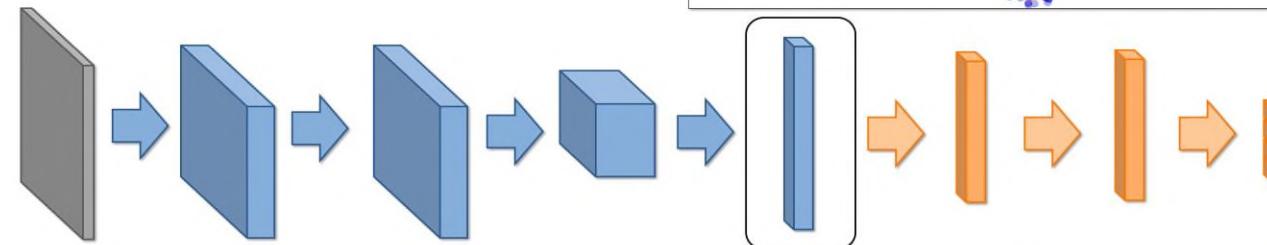
Обучение на исходном домене



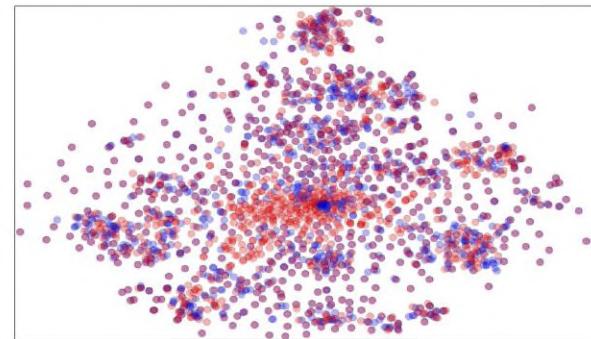
$$S(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \mid \mathbf{x} \sim S(\mathbf{x})\}$$
$$T(\mathbf{f}) = \{G_f(\mathbf{x}; \theta_f) \mid \mathbf{x} \sim T(\mathbf{x})\}$$

Основная идея: хотим научиться извлекать признаки, которые будут инвариантными относительно домена

Имеющееся распределение признаков:



Наша цель (после адаптации):

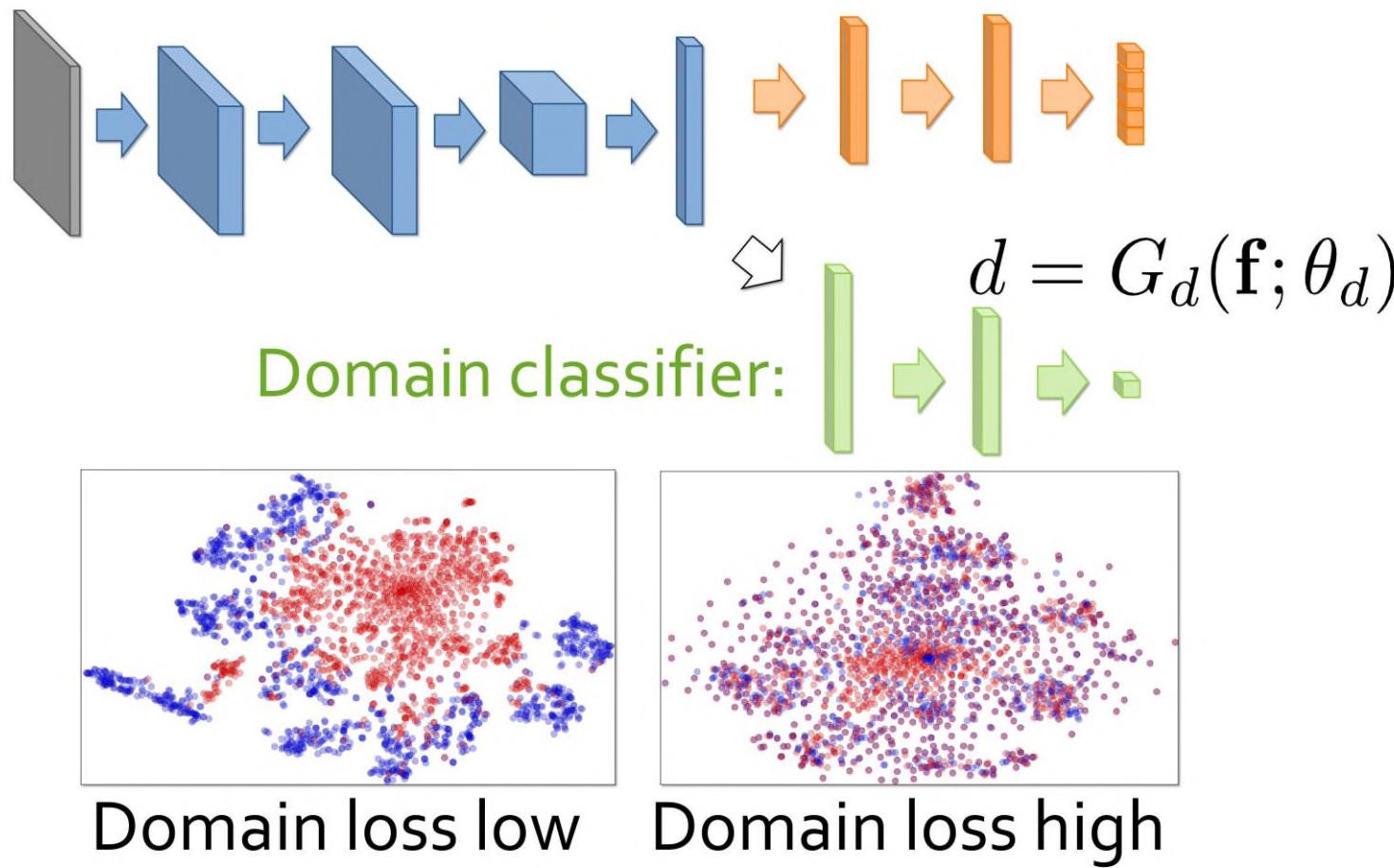


Выравнивание распределений доменов в пространстве признаков

Общий подход: добавить в функцию потерь элемент, который штрафует несоответствие двух распределений

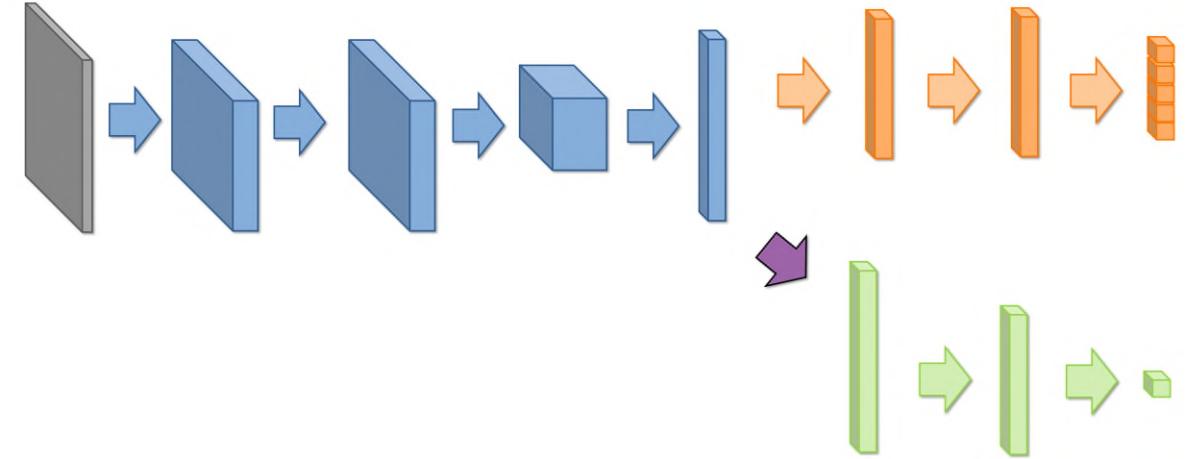
- Measuring second-order moments mismatch [*Sun and Saenko 2016*], ...
- Maximum mean discrepancy (MMD) [*Long and Wang ICML 2015*], ...
- *Alignment through adversarial learning [GANin and Lempitsky, ICML 2015] [GANin, Ustinova, Ajakan, Germain, Larochelle, Laviolette, Marchand, and Lempitsky, JMLR 2016] [Tseng, Hoffman, Darrell, Saenko, ICCV 2015]*

Adversarial подход: использование классификатора домена

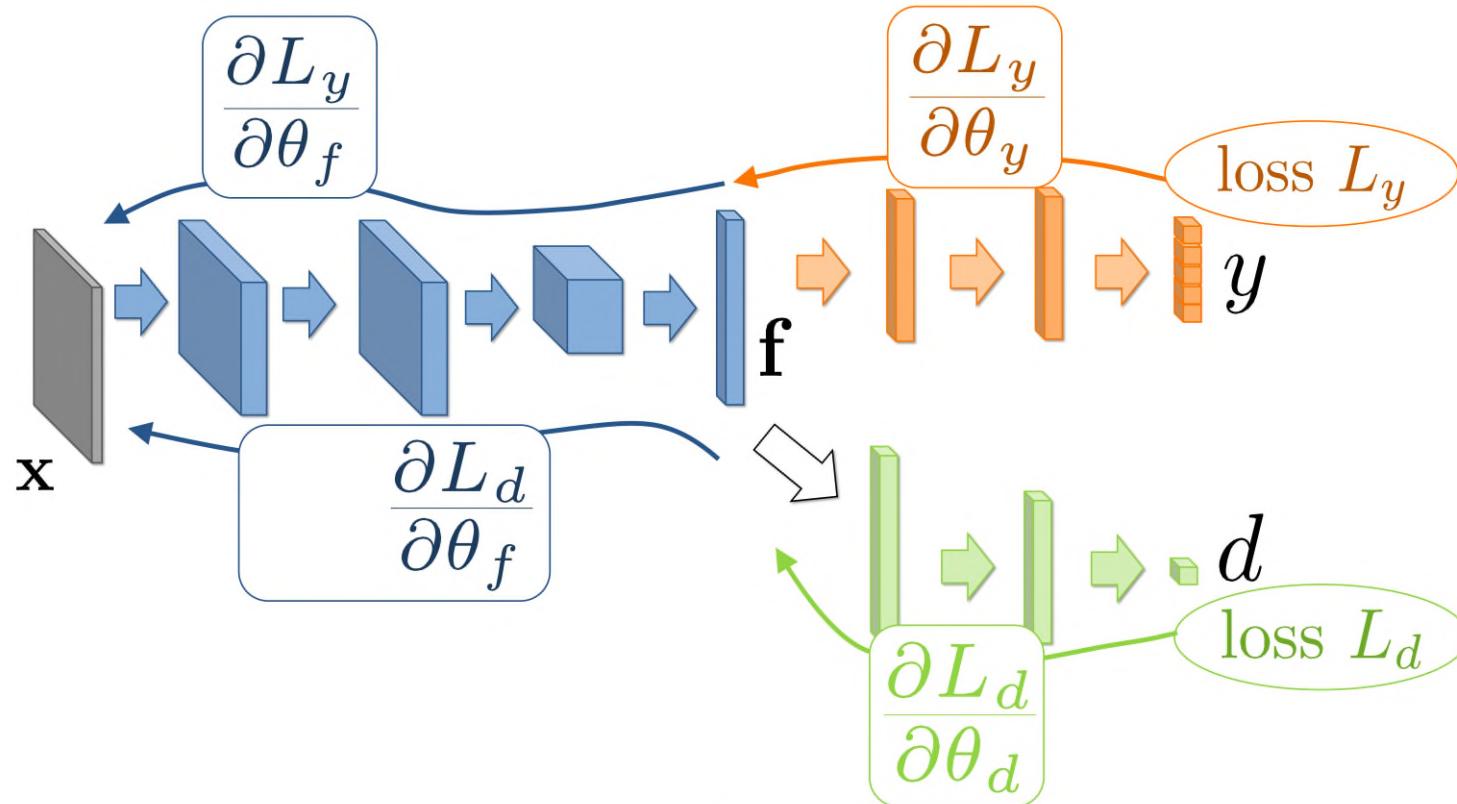


Adversarial подход: использование классификатора домена

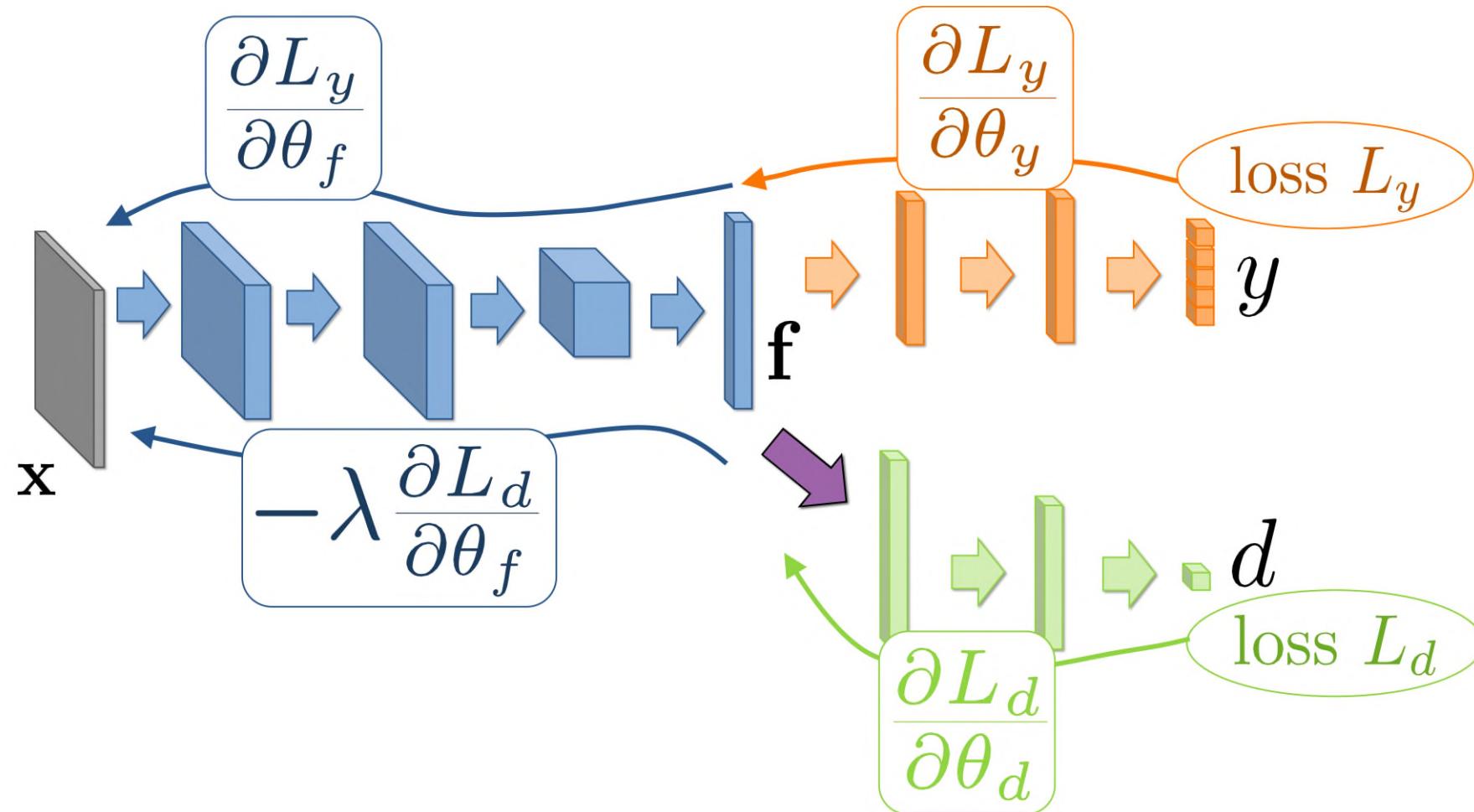
- Построить сеть такой архитектуры
- Обучать feature extractor + class predictor на исходных данных
- Обучать feature extractor + domain classifier на всех данных
- Использовать feature extractor + class predictor во время тестирования



Проблема: классификатор пытается максимизировать разницу между доменами, а feature extractor должен её минимизировать



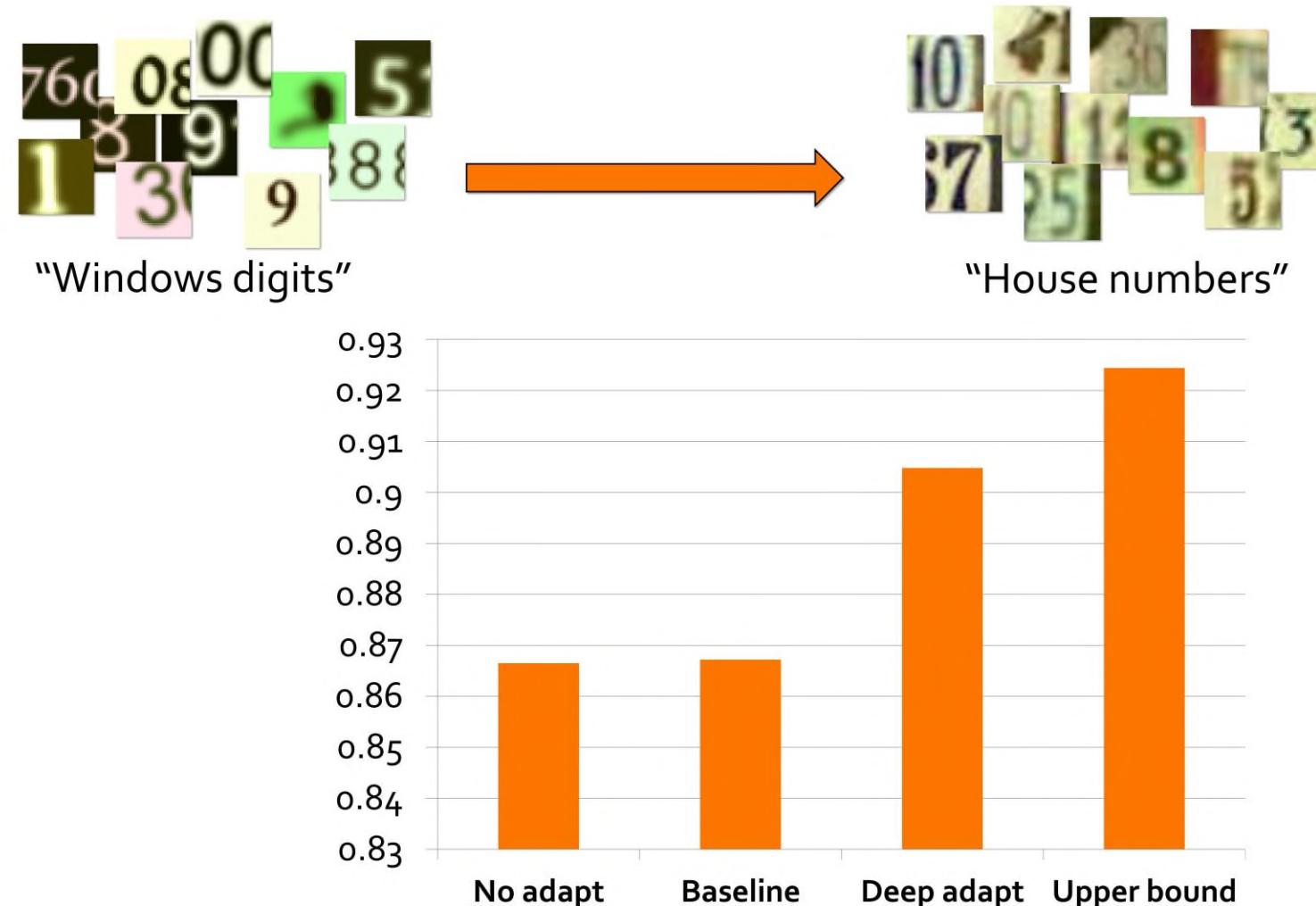
Решение: gradient reversal layer



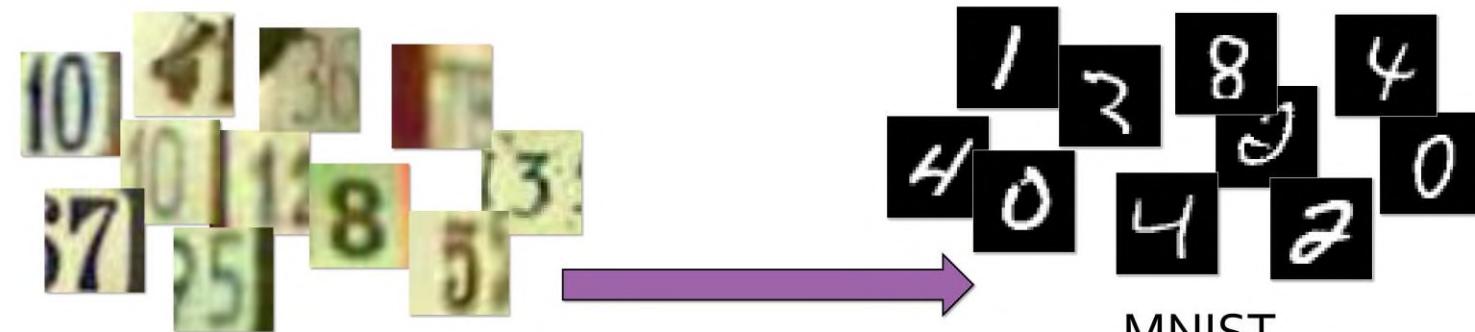
Решение: gradient reversal layer

```
class CGradientReversalLayer : public CCnnBaseLayer {  
...  
protected:  
    CCnnBlob RunOnce( CCnnBlob input ) {  
        return input;  
    }  
    CCnnBlob BackwardOnce( CCnnBlob diffBlob ) {  
        return -lambda * diffBlob;  
    }  
private:  
    float lambda;  
}
```

Результат работы: небольшой разрыв

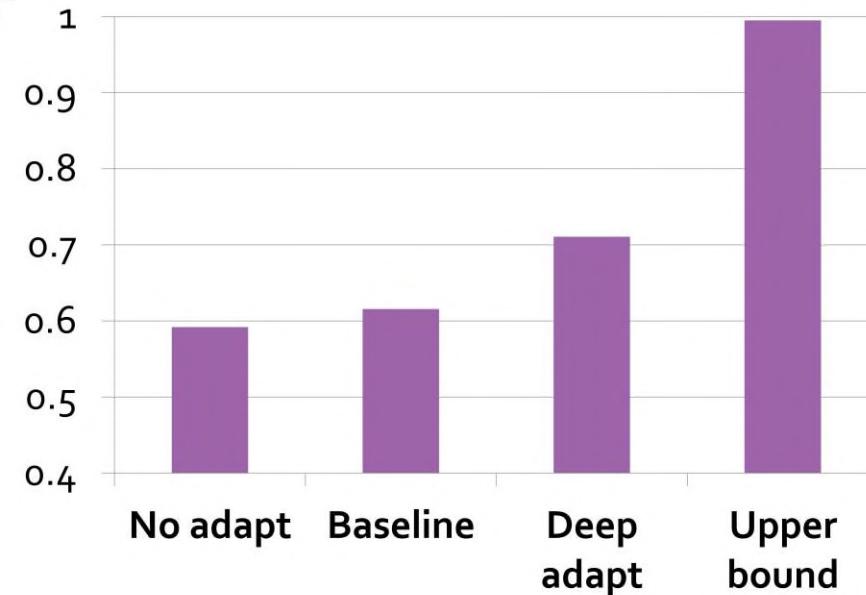


Результат работы: сильный разрыв

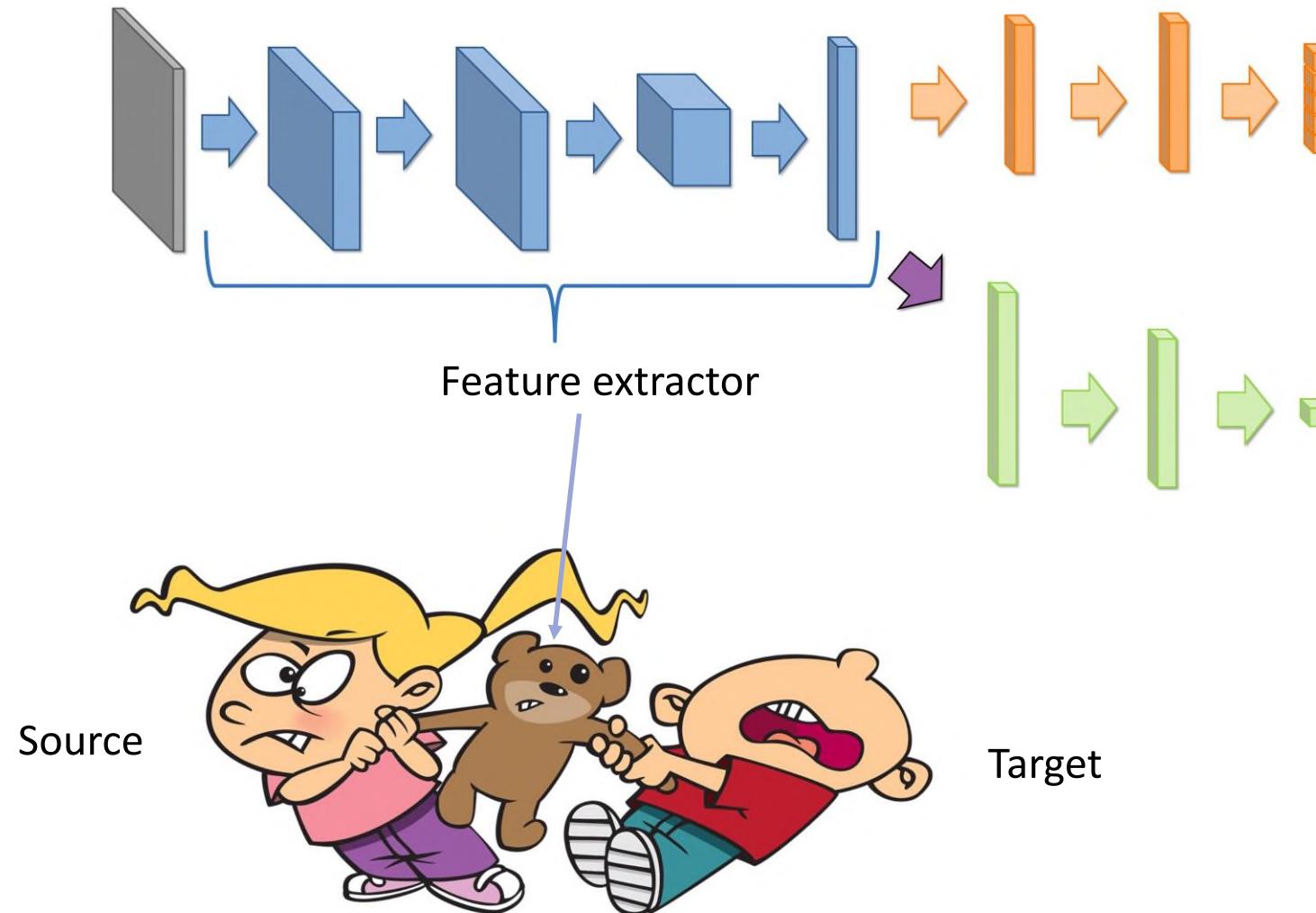


"House numbers"

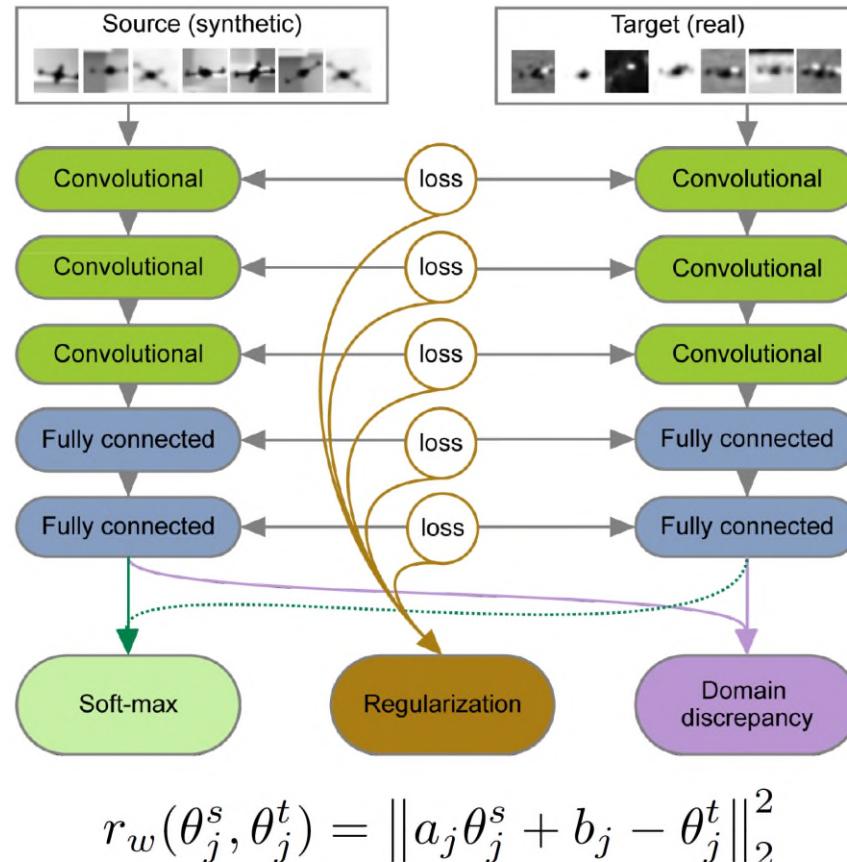
Reverse
direction does
not work ☹



Правильно ли разделять признаки между двумя доменами?



Нужно ли производить разделения весов?

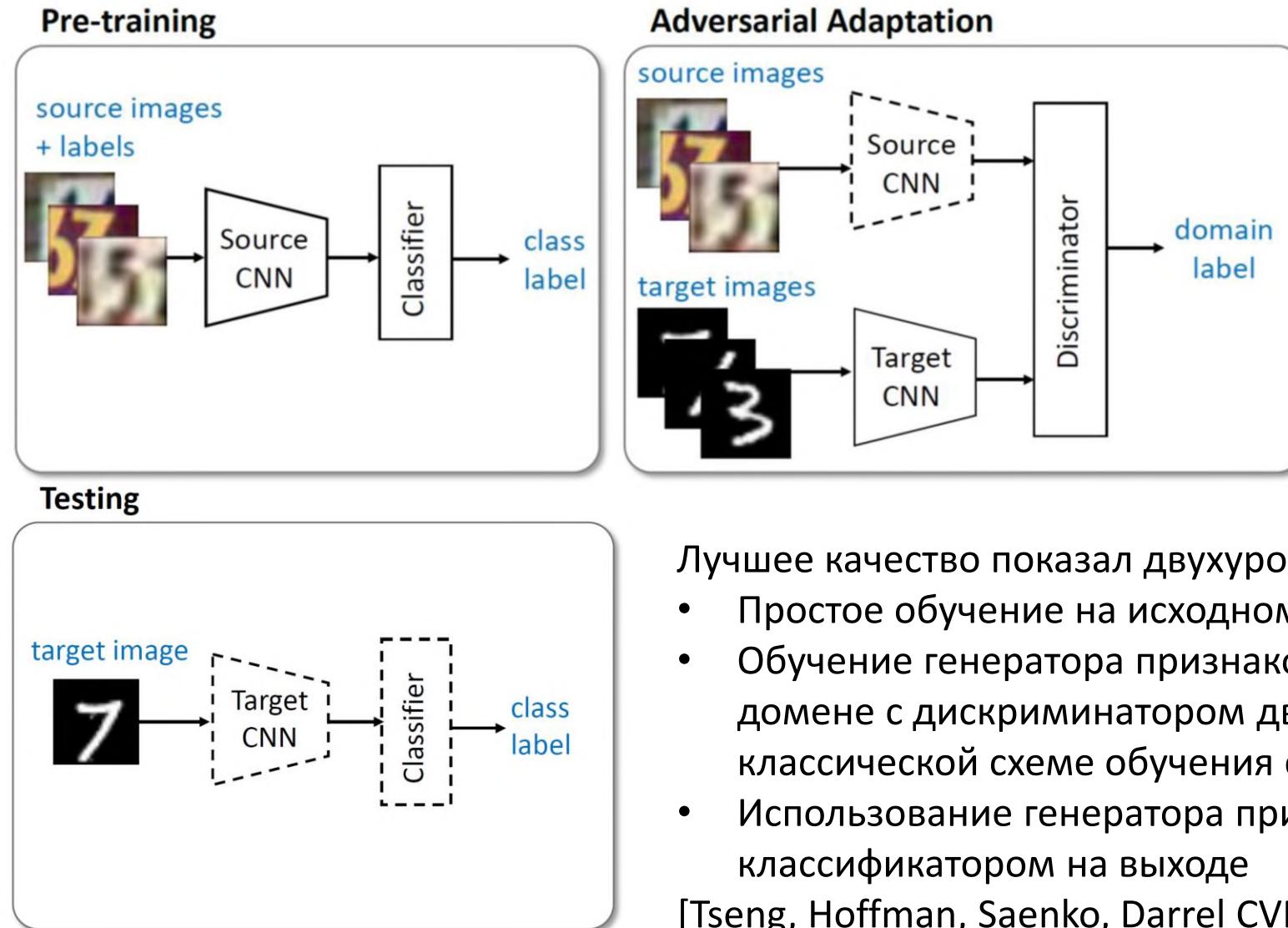


Idea: if you share some layers and not share others, results may get better

	A → W	D → W	W → D	Average
GFK [19]	0.214	0.691	0.650	0.518
DLID [7]	0.519	0.782	0.899	0.733
DDC [44]	0.605	0.948	0.985	0.846
DAN [34]	0.645	0.952	0.986	0.861
DRCN [16]	0.687	0.964	0.990	0.880
GRL [14]	0.730	0.964	0.992	0.895
Ours (+ DDC)	0.630	0.961	0.992	0.861
Ours (+ GRL)	0.760	0.967	0.996	0.908

[Rozantsev, Salzmann, Fua 2016]

Adversarial Discriminative Domain Adaptation

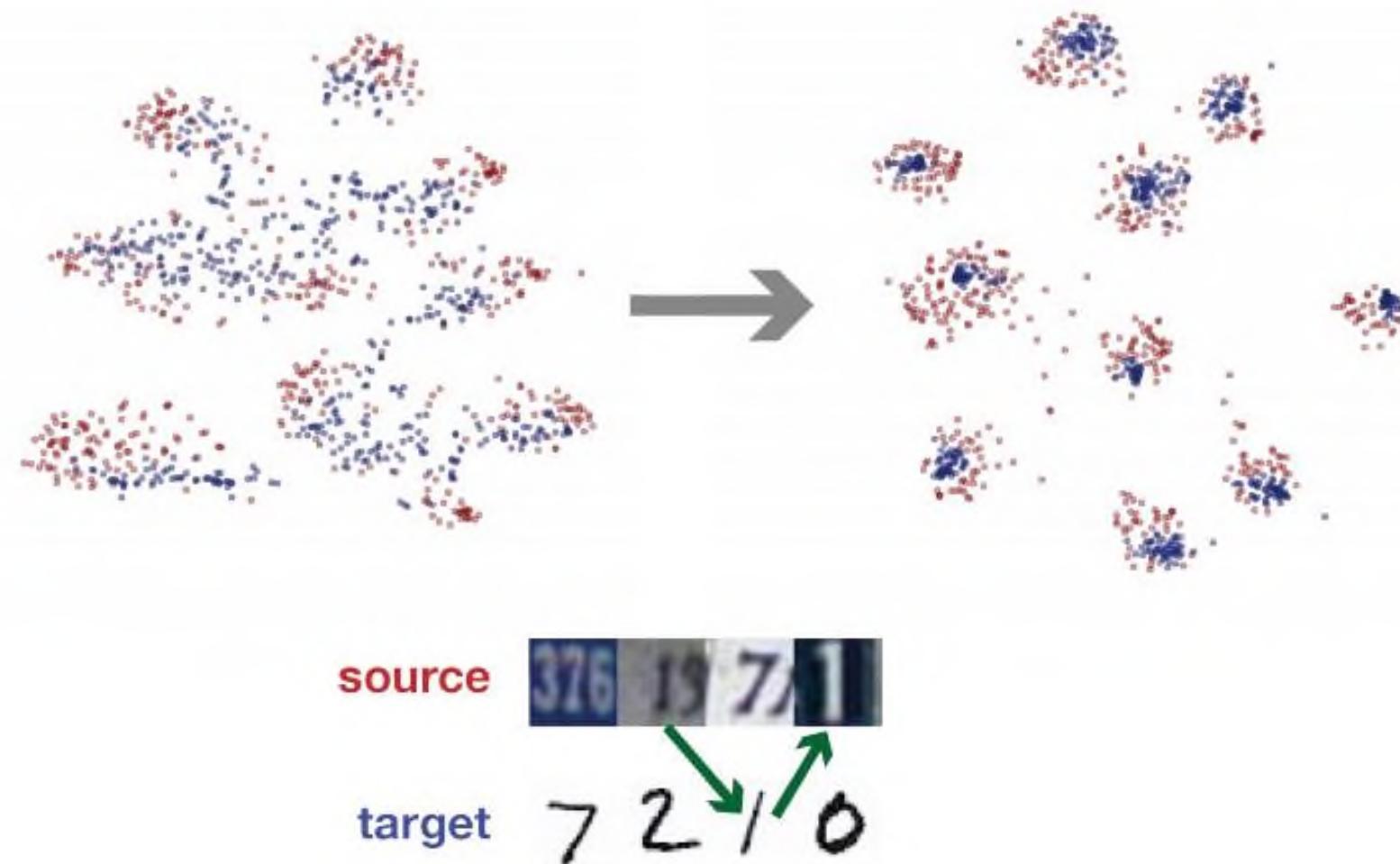


Лучшее качество показал двухуровневый подход:

- Простое обучение на исходном домене
- Обучение генератора признаков на целевом домене с дискриминатором двух доменов по классической схеме обучения cond. GAN
- Использование генератора признаков с классификатором на выходе

[Tseng, Hoffman, Saenko, Darrel CVPR17]

Associative Domain Adaptation

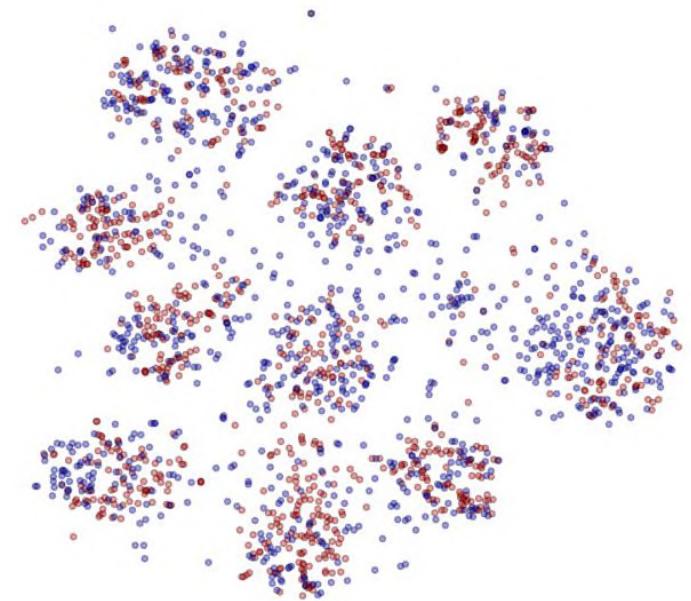
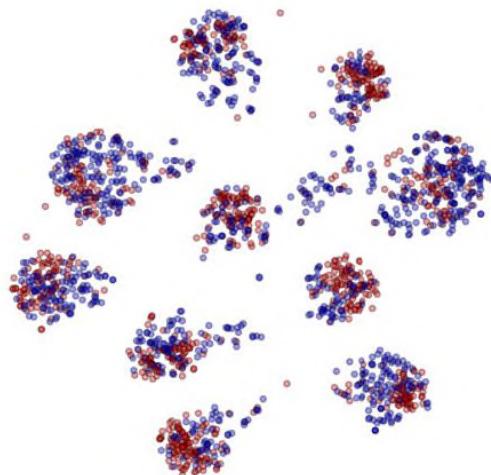
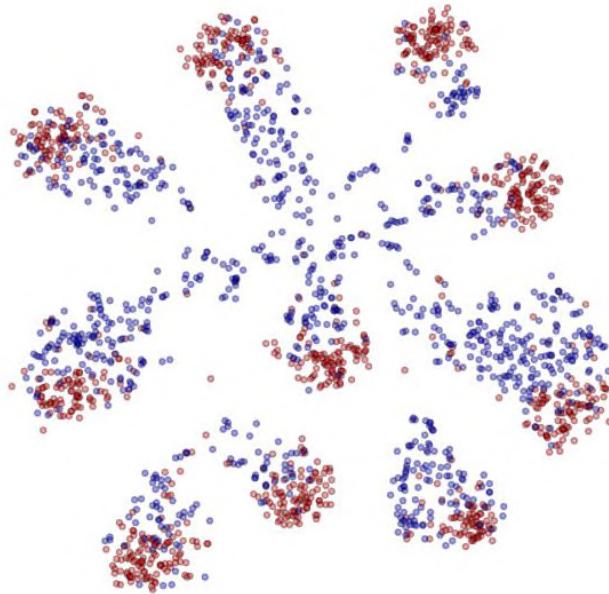


[Hausser, Frerix, ... ICCV17]

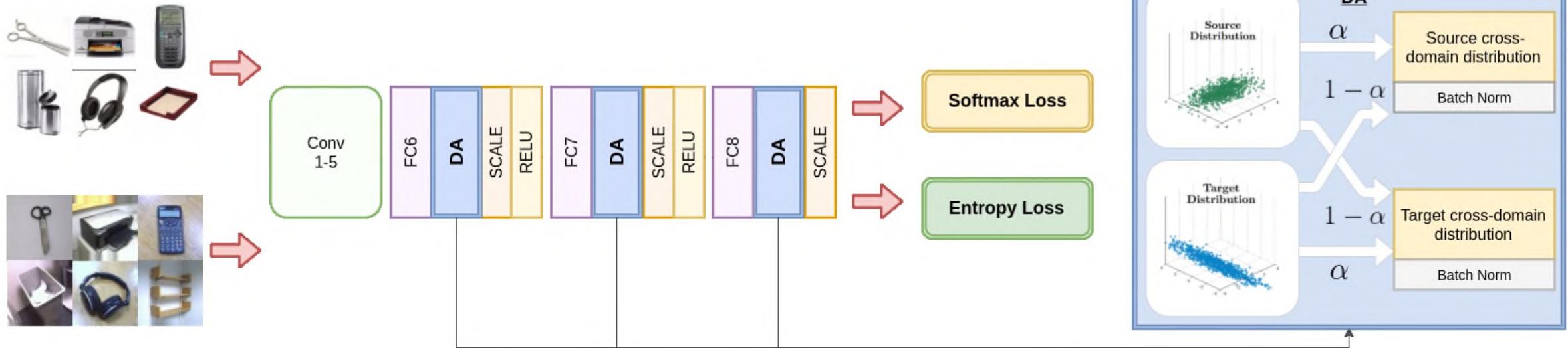
Associative Domain Adaptation – результаты

Method	Domains (source → target)			
	MNIST → MNIST-M	Syn. Digits → SVHN	SVHN → MNIST	Syn. Signs → GTSRB
Transf. Repr. [22]	13.30	-	21.20	-
SA [8]	43.10	13.56	40.68	18.35
CORAL [24]	42.30	14.80	36.90	13.10
ADDA [30]	-	-	24.00	-
DANN [9]	23.33 (55.87 %)	8.91 (79.67 %)	26.15 (42.57 %)	11.35 (46.39 %)
DSN w/ DANN [3]	16.80 (63.18 %)	8.80 (78.95 %)	17.30 (58.31 %)	6.90 (54.42 %)
DSN w/ MMD [3]	19.50 (56.77 %)	11.50 (31.58 %)	27.80 (32.26 %)	7.40 (51.02 %)
MMD [15]	23.10	12.00	28.90	8.90
DA _{MMD}	22.90	19.14	28.48	10.69
Ours (DA _{assoc} fixed params [†])	10.47 ± 0.28	8.70 ± 0.2	4.32 ± 1.54	17.20 ± 1.32
Ours (DA_{assoc})	10.53 (85.94 %)	8.14 (87.78 %)	2.40 (93.71 %)	2.34 (81.23)
Source only	35.96	15.68	30.71	4.59
Target only	6.37	7.09	0.50	1.82

Associative Loss vs MMD loss эмбеддинги



Automatic Domain Alignment Layers



$$q_{\alpha}^{st} = \alpha q^s + (1 - \alpha) q^t$$

$$q_{\alpha}^{ts} = \alpha q^t + (1 - \alpha) q^s$$

$$\text{DA}(x_s; \alpha) = \frac{x_s - \mu_{st,\alpha}}{\sqrt{\epsilon + \sigma_{st,\alpha}^2}}, \quad \text{DA}(x_t; \alpha) = \frac{x_t - \mu_{ts,\alpha}}{\sqrt{\epsilon + \sigma_{ts,\alpha}^2}}$$

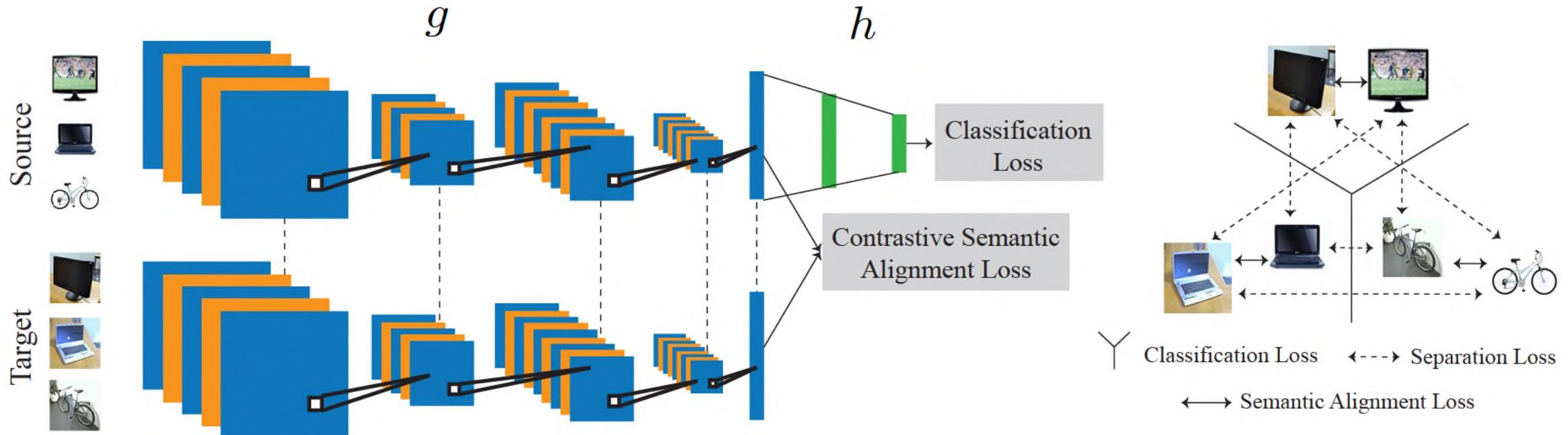
[Fabio, Lorenzo, ... ICCV17]

Automatic Domain Alignment Layers – результаты

ABBYY®

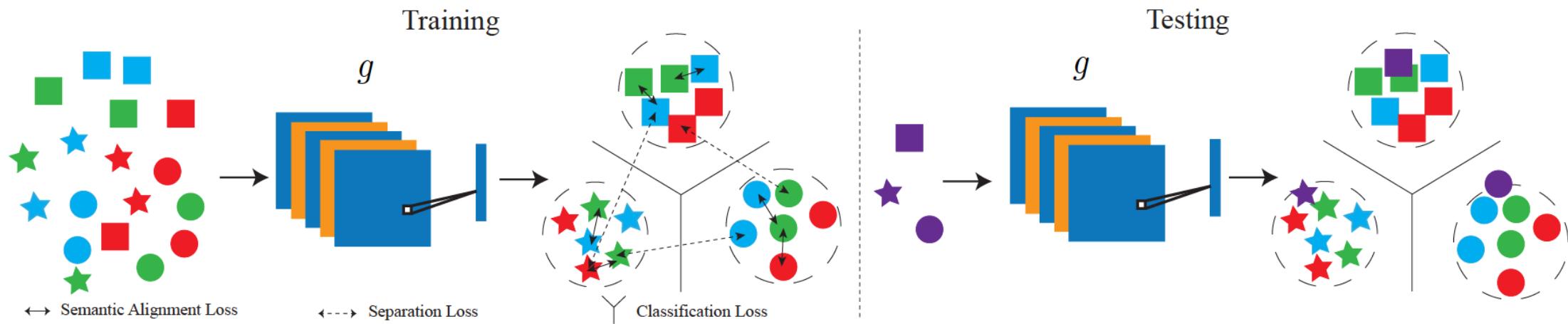
Method	Source Target	Amazon DSLR	Amazon Webcam	DSLR Amazon	DSLR Webcam	Webcam Amazon	Webcam DSLR	Average
AlexNet – source [18]		63.8	61.6	51.1	95.4	49.8	99.0	70.1
DDC [32]		64.4	61.8	52.1	95.0	52.2	98.5	70.6
DAN [21]		67.0	68.5	54.0	96.0	53.1	99.0	72.9
ReverseGrad [9]		67.1	72.6	54.5	96.4	52.7	99.2	72.7
DRCN [10]		66.8	68.7	56.0	96.4	54.9	99.0	73.6
RTN [23]		71.0	73.3	50.5	96.8	51.0	99.6	73.7
JAN [22]		71.8	74.9	58.3	96.6	55.0	99.5	76.0
AutoDIAL – AlexNet		73.6	75.5	58.1	96.6	59.4	99.5	77.1

Unified Deep Supervised Domain Adaptation and Generalization



[Motiian, Piccirilli, ... ICCV17]

Unified Deep Supervised Domain Adaptation and Generalization



Semantic Alignment

$$d(g(x_i^s), g(x_j^t)) = \frac{1}{2} \|g(x_i^s) - g(x_j^t)\|^2 ,$$

Separation

$$k(g(x_i^s), g(x_j^t)) = \frac{1}{2} \max(0, m - \|g(x_i^s) - g(x_j^t)\|)^2$$

Domain Generalization – результаты

	<i>CAE [49]</i>	<i>MTAE [24]</i>	<i>CCSA</i>
$M_{15^\circ}, M_{30^\circ}, M_{45^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M$	72.1	82.5	84.6
$M, M_{30^\circ}, M_{45^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M_{15^\circ}$	95.3	96.3	95.6
$M, M_{15^\circ}, M_{45^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M_{30^\circ}$	92.6	93.4	94.6
$M, M_{15^\circ}, M_{30^\circ}, M_{60^\circ}, M_{75^\circ} \rightarrow M_{45^\circ}$	81.5	78.6	82.9
$M, M_{15^\circ}, M_{30^\circ}, M_{45^\circ}, M_{75^\circ} \rightarrow M_{60^\circ}$	92.7	94.2	94.8
$M, M_{15^\circ}, M_{30^\circ}, M_{45^\circ}, M_{60^\circ} \rightarrow M_{75^\circ}$	79.3	80.5	82.1
Average	85.5	87.5	89.1

Выводы

- Стоит быть аккуратней с синтетическими и сгенерированными данными и пробовать методы Domain Adaptation для улучшения качества
- Стоимость неразмеченных целевых данных резко возрастает
- На каждый датасет свой state of the art, так что осмысленно пробовать разные подходы

Спасибо!
