Out-of-Distribution

Классический ML

Компоненты задачи в классическом ML:

- пространство объектов X
- множество ответов Y
- P(x, y) - истинное распределение данных на X×Y
- функция риска R
- обучающая выборка $D_{train} = \{x_i, y_i\}_{i=1..n1} \sim P(x, y)$
- тестовая выборка $D_{test} = \{x_i, y_i\}_{i=1..n2} \sim P(x, y)$

Допущение: на проде будут $D_{prod} \sim P \Rightarrow R_{prod} \approx \mathbf{E}R \approx R_{test}$

Реальность:на проде данные $D_{prod} \sim Q \neq P$ – **сдвиг распределения (distribution shift)**

Важно: часто Q не известно

Задача классификации

Note: далее рассматриваем на примере задачи многоклассовой классификации, где

- X - картинки
- Y - множество классов Y = {1, …, k}
- R - accuracy

Источники сдвига

Источник сдвига: изменение условий съемки

- обработанные студийные фотографии
- естественные фотографии пользователей

обучение

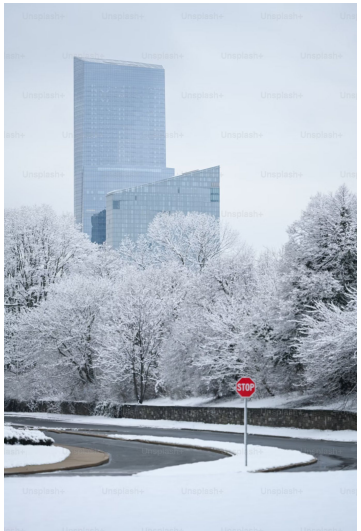
прод

Источники сдвига

Источник сдвига: изменение погодных условий

обучение

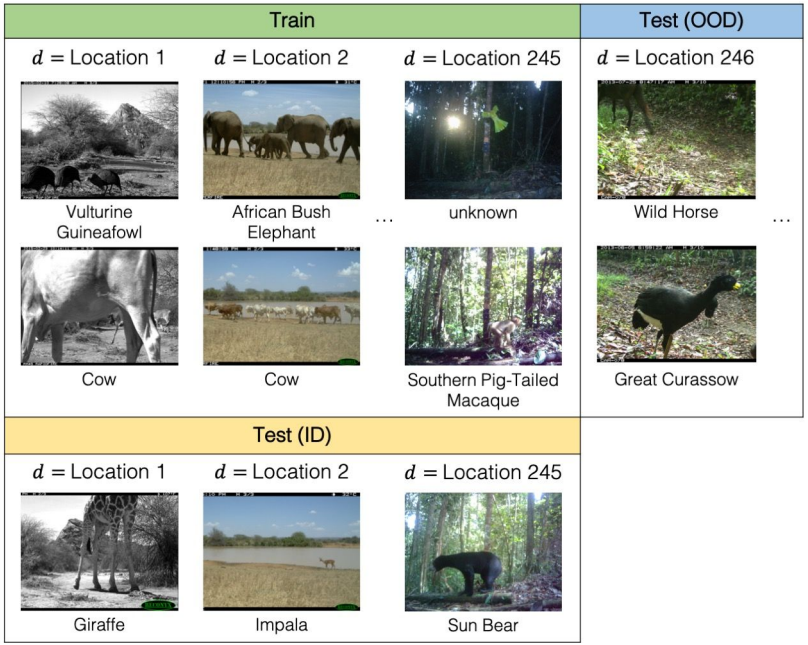
прод



Источники сдвига

Источник сдвига: изменение места съемки.
Пример (датасет WILDS iWildCam):

- обучение на старых фотоловушках
- распознавание на новых фотоловушках



Источники сдвига

Источник сдвига: прошло время.

Пример: собранные по примерно идентичной процедуре с разницей в 10 лет датасеты ImageNet (2010) и ImageNetV2 (2019).

(a) Test Set A

(b) Test Set B

Источники сдвига

Источник сдвига: сжатие данных (напр., JPEG)

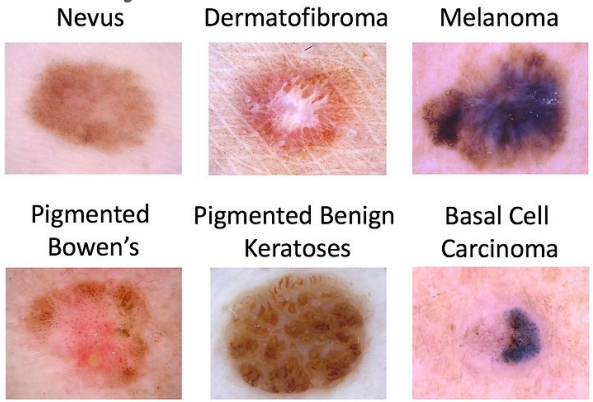
Источники сдвига

Источник сдвига: новый класс

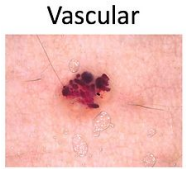
Пример:

- вы обучили модель распознавать 6 типов родинок
- пользователь по незнанию скормил ей иной, неподдерживаемый тип

обучение



прод



Виды сдвигов

Рассматривают разные виды сдвигов P(x, y) и Q(x, y):

вид сдвига	изменяется	сохраняется
covariate shift	$P(x) \neq Q(x)$	$P(y\|x) = Q(y\|x)$
concept shift	$P(y\|x) \neq Q(y\|x)$	$P(x) = Q(x)$
label (semantic) shift	$P(y) \neq Q(y)$	$P(x\|y) = Q(x\|y)$
conditional shift	$P(x\|y) \neq Q(x\|y)$	$P(y) = Q(y)$

Что делать?

Подходы к работе со сдвигами:

1. обеспечить устойчивость к сдвигам (OOD generalization)
2. детекция сдвига (OOD detection)
3. адаптация к сдвигу (domain adaptation)

Термины

- домен *D* = распределение данных $P_D(x,y)$.
- $P = P_s$ - исходное распределение (source domain)
- $Q = P_t$ - целевое распределение (target domain)

Обзоры

- OOD generalization: N. Ye et al., 2021 “OoD-Bench: Quantifying and Understanding Two Dimensions of Out-of-Distribution Generalization” https://arxiv.org/abs/2106.03721
- OOD detection: J. Yang et al., 2021 “Generalized Out-of-Distribution Detection: A Survey” https://arxiv.org/abs/2110.11334
- UDA: K. Musgrave et al., 2021 “Unsupervised Domain Adaptation: A Reality Check” https://arxiv.org/abs/2111.15672

OOD Generalization

OOD Generalization

OOD generalization (robustness): задача сделать модель устойчивую к сдвигам.

По режиму обучения:

- на 1 домене - single source DG (SSDG)
- на нескольких доменах - [multi-source] domain generalization (DG)

OOD Generalization: лучшие методы

Критерии выбора лучших:

1. sota
2. простота

SSDG: Pro-RandConv - sota 2023

MSDG: ERM, IRM – лидеры OoD-Bench’21

RandConv

Paper: Zh. Xu et al., 2020 "Robust and Generalizable Visual Representation Learning via Random Convolutions" arxiv

Цель: обеспечить робастность к изменению текстуры и небольшому шуму (к которому робастны люди)

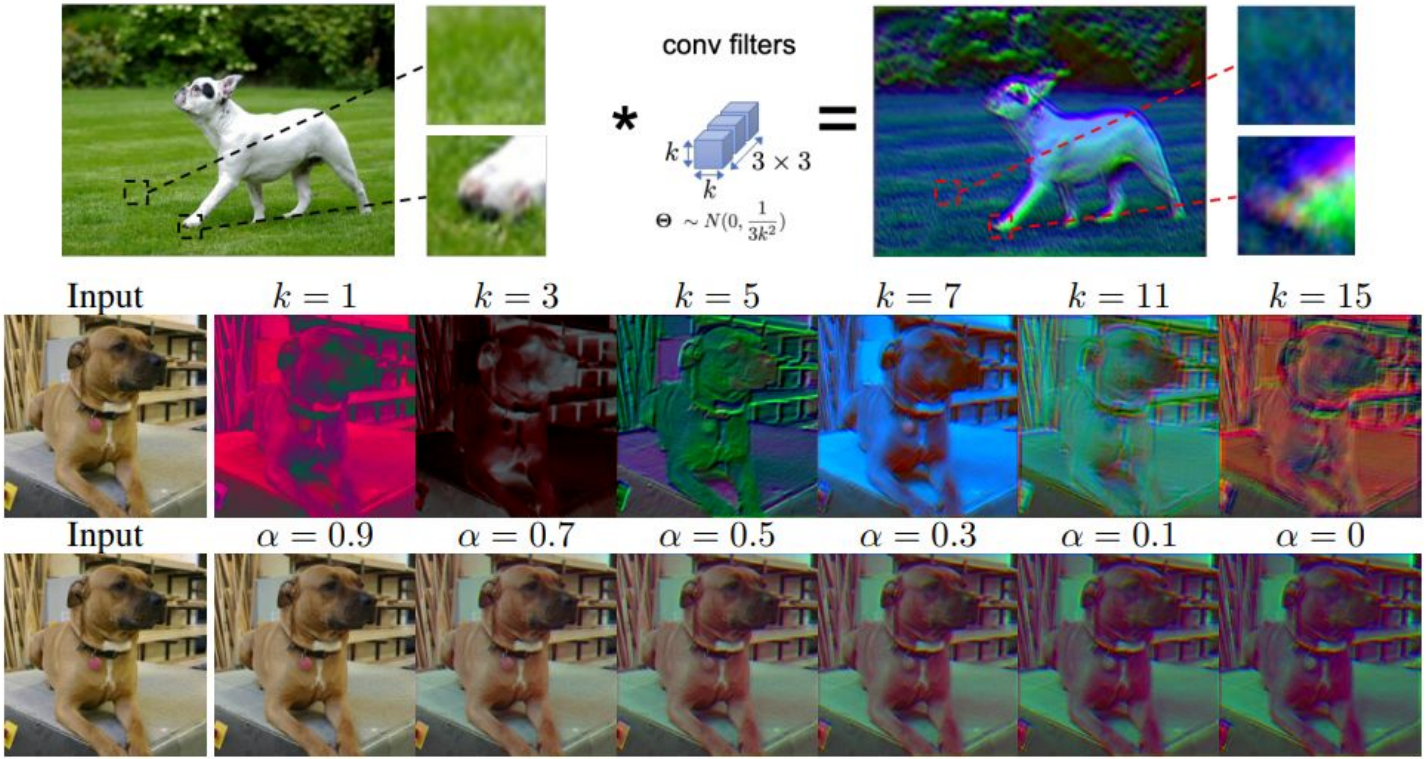
RandConv метод: аугментация данных через применение случайных сверток к исходным изображениям

Почему работает: случайные свертки сохраняют форму, но искажают локальные текстуры

Улучшалки:

- применяют свертки с разным размером ядра (multi-scale), для нарушения текстур разной величины
- использование mixup оригинальных изображений и RandConv изображений
- consistency loss

RandConv


conv filters
k
3×3
k
$\Theta \sim N(0, \frac{1}{3k^2})$
Input
$k = 1$
$k = 3$
$k = 5$
$k = 7$
$k = 11$
$k = 15$
Input
$\alpha = 0.9$
$\alpha = 0.7$
$\alpha = 0.5$
$\alpha = 0.3$
$\alpha = 0.1$
$\alpha = 0$

Pro-RandConv

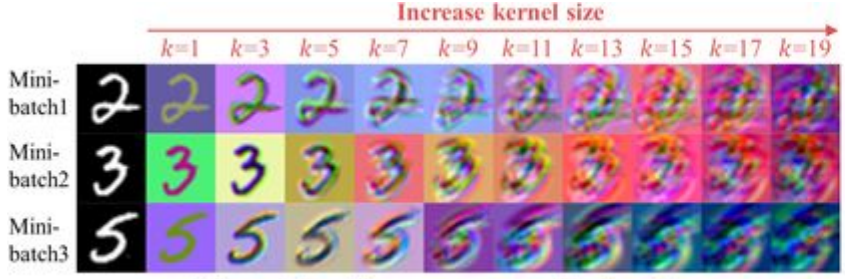
Paper: S. Choi et al., 2023 "Progressive Random Convolutions for Single Domain Generalization" arxiv

Проблемы RandConv:

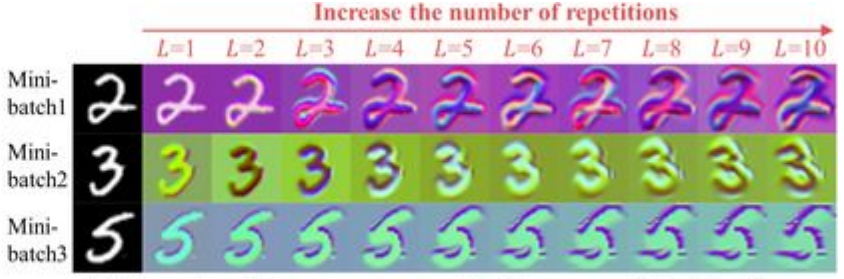
- маленькое ядро - не разбивает большие текстуры
- большое ядро - нарушает семантику

Решение: последовательное применение случайных сверток с маленьким ядром (progressive random convolutions = Pro-RandConv).

Почему работает: при увеличенном рецептивном поле, уменьшает влияние далеких пикселей на итоговое значение.



(a) Examples of images augmented by RandConv



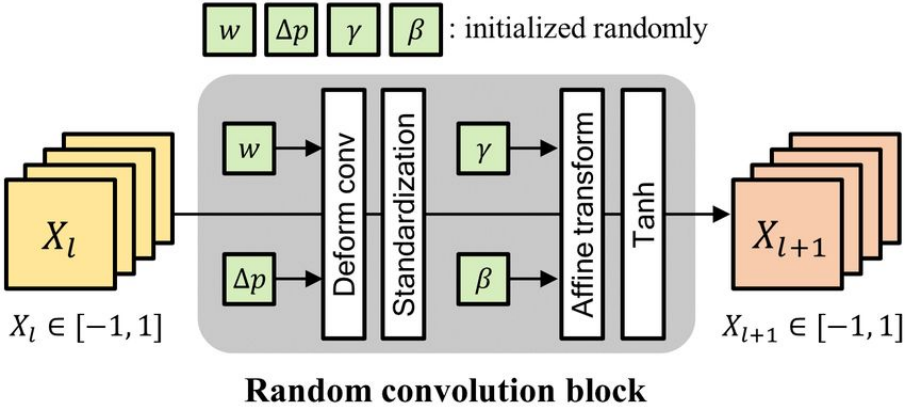
(b) Examples of images augmented by the proposed Pro-RandConv

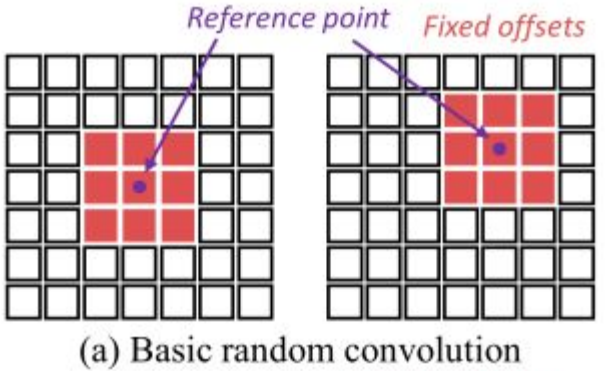
Pro-RandConv

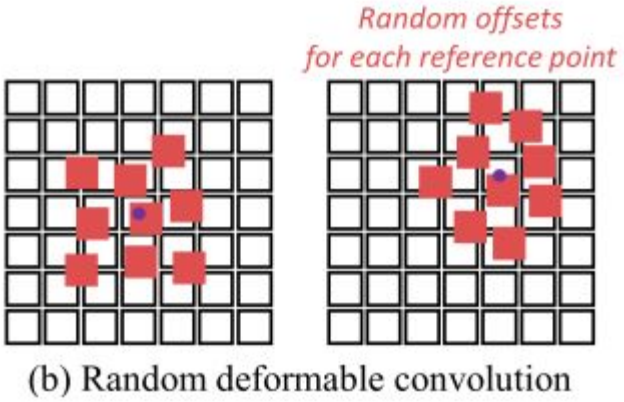
Улучшалка: замена RandConv слоя на RandConv блок.

Элементы RandConv блока:

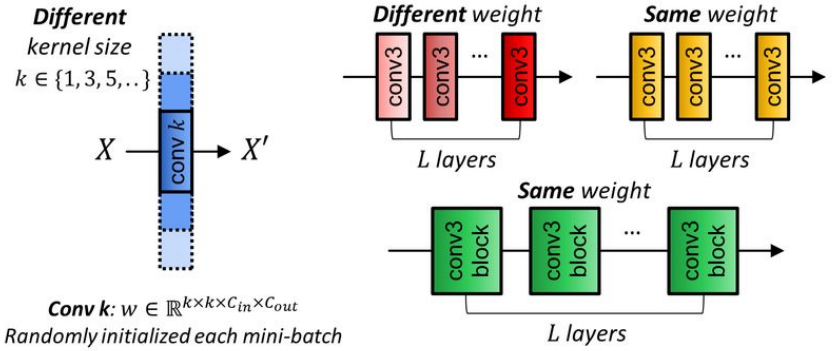
- random deformable covolutions – повышает разнообразие текстуры
- попиксельные нормализация + случайное линейное преобразование – повышает разнообразие контраста
- Tanh nonlinearity - возвращает значения пикселей в разумный диапазон



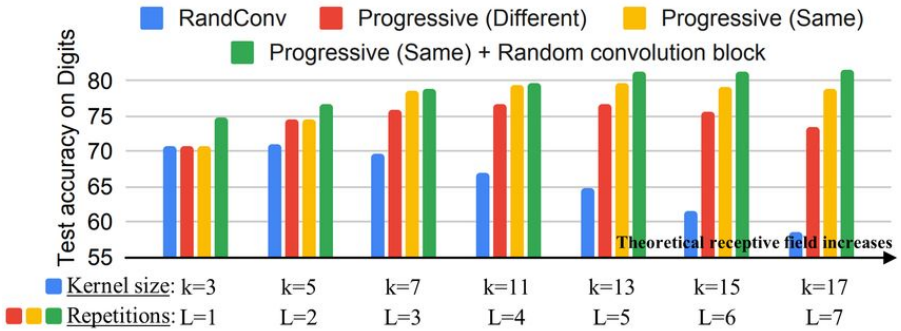




Pro-RandConv



(a) RandConv (single-layer) (b) Progressive approach (multi-layer)



(c) Performance with changing kernel size (k) or number of repetitions (L)

ERM

Paper: Vladimir Vapnik. Statistical Learning Theory. Wiley, 1998.

ERM (Empirical Risk Minimization): минимизация средней ошибки по всем сэмплам из всех доменов.

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i), y_i)$$

IRM

Paper: M. Arjovsky et al., 2019 "Invariant risk minimization" arxiv

Идея: строим модель f, делающую предсказания [только] по доменно-инвариантным фичам.

Соображение: разобъем модель f в композицию линейного классификатора w и энкодера Ф
Если Ф извлекает доменно-инвариантные фичи, то существует w, являющийся оптимальным классификатором на любом домене.

Целевая функция **IRM:**

$$\min_{\Phi, w} \sum_{e \in \mathcal{E}_{\text{tr}}} R^e(w \circ \Phi)$$

$$\text{subject to } w \in \arg\min_{\tilde{w}} R^e(\tilde{w} \circ \Phi) \quad \forall e \in \mathcal{E}_{\text{tr}}$$

IRMv1

Проблема: делать оптимизацию с ограничениями тяжело.

Решение - единая функция потерь **IRMv1:**

- зафиксируем единичный классификатор и постараемся его сделать искомым оптимальным
- для этого используем 2-компонентную функцию потерь
  - минимизация риска по всем доменам -> оптимальность модели вцелом
  - штраф на норму градиента по w -> оптимальность именно на w=1

$$\min_{\Phi} \sum_{e \in \mathcal{E}_{\text{tr}}} R^e(\Phi) + \lambda \cdot \left\| \nabla_{w|w=1.0} R^e(w \cdot \Phi) \right\|^2$$

OOD Detection

OOD Detection

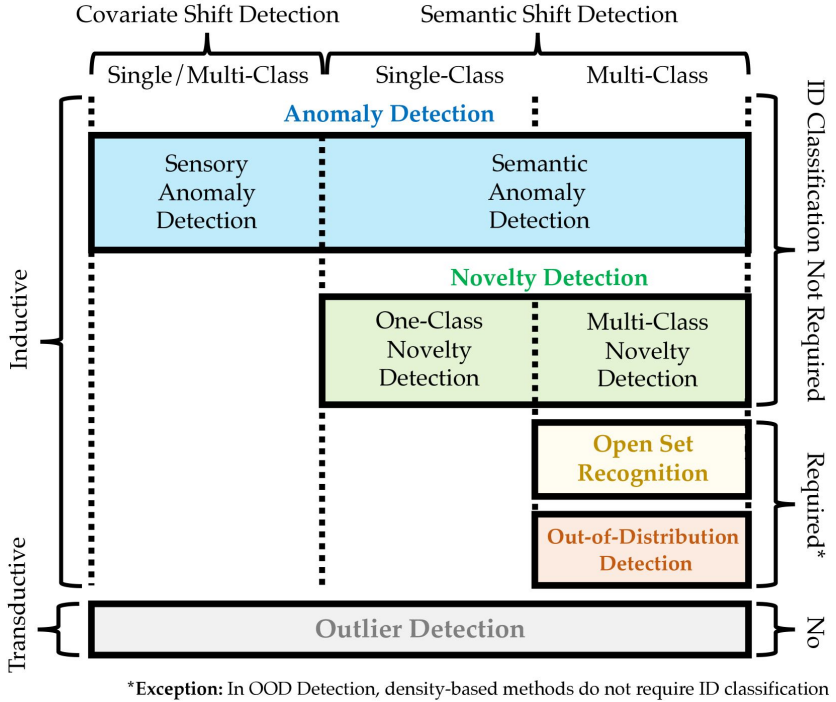
Задача обобщенной OOD детекции - выявить поступление сэмпла из другого распределения.

Разные модификации:

- OOD detection
- anomaly detection
- novelty detection
- outlier detection
- open-set recognition

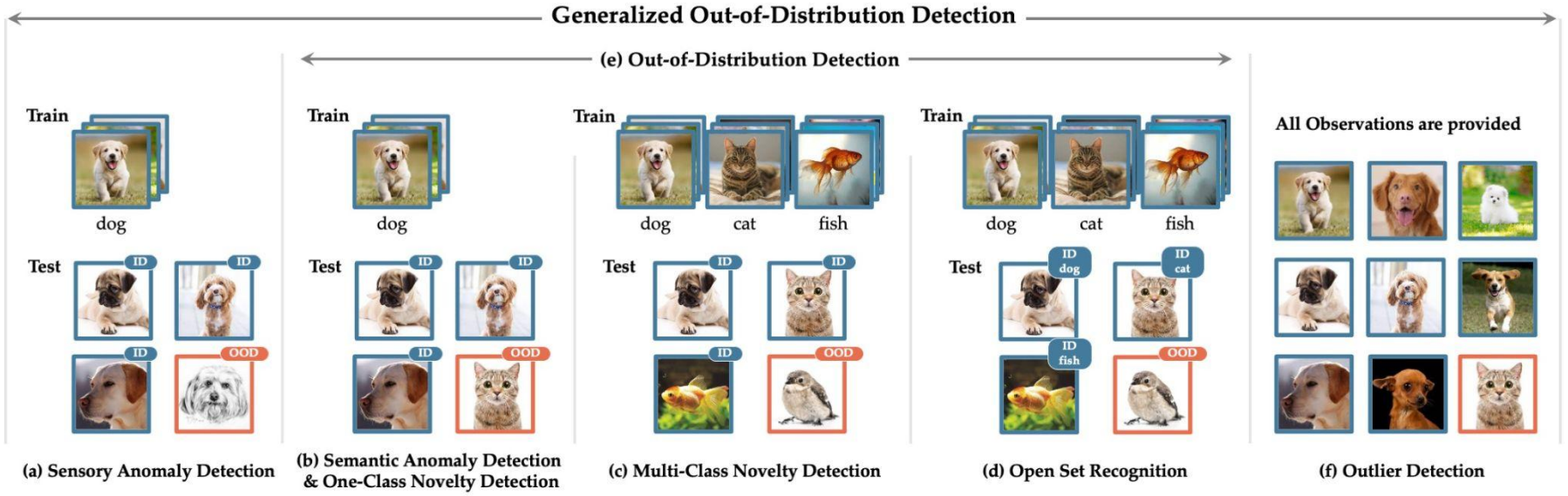
Классификация задач OOD детекции

- по носителю маргинального распределения лейблов
  - covariate shift - не содержит новых категорий
  - semantic shift - содержит новые категории
- по количеству классов, подлежащих классификации
  - single class
  - multiple class
- по необходимости распознавать ID классы
  - надо
  - не надо
- режим обучения
  - индуктивное обучение
  - трансдуктивное обучение



source: J. Yang et al., 2021 “Generalized Out-of-Distribution Detection: A Survey”
https://arxiv.org/pdf/2110.11334.pdf

Классификация задач OOD детекции



source: J. Yang et al., 2021 “Generalized Out-of-Distribution Detection: A Survey” https://arxiv.org/pdf/2110.11334.pdf

OOD Detection: лучшие методы

Критерии выбора лучших:

1. post-hoc метод
2. sota
3. простота

Выбираем: MSP, KNN, ReAct - лидеры OpenOOD

MSP

Paper: D. Hendrycks, K. Gimpel, 2016 "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks" arxiv

TLDR: детекция OOD по confidence

Подробнее:

- Детекция OOD осуществляется по confidence (= наибольшая вероятность в предсказательном распределении = maximum softmax probability (MSP))
- Сэмпл считается ID, если его confidence > λ
- Порог λ выбирается по confidence ID данных, напр., берется 5-й перцентиль валидационного множества

KNN

Paper: Y. Sun et al., 2022 “Out-of-Distribution Detection with Deep Nearest Neighbors” arxiv

TLDR: детекция OOD по расстоянию до k-го соседа

Метод:

- Дано: модель f и обучающая выборка X=[x1, x2, …, xn]
- Подготовка:
  - Разбиваем f на энкодер g и последний линейный слой с весами w и смещением b.
  - Собираем латентные представления на обучающей выборке $Z=[z_1, z_2, \ldots, z_n]$, где $z_i=g(x_i)/||g(x_i)||$
- Метод проверки на OOD для нового сэмпла x*:
  - Считаем его латентное представление $z^*=g(x^*)/||g(x^*)||$
  - Находим его k-го ближайшего соседа из обучающей выборки $z_{(k)}$
  - Принимается решение, что сэмпл OOD, если $||z^*-z_{(k)}|| > \lambda$
- Выбор порогового значения λ:
  - Порог λ выбирается по ID данным, напр., берется 95-й перцентиль

ReAct

Paper: Y. Sun et al., 2021 "ReAct: Out-of-distribution Detection With Rectified Activations" arxiv

TLDR: Способ понизить overconfidence модели на OOD сэмплах

ReAct: клиппинг выхода с penultimate слоя по порогу *c*

- пусть f(x) = W*g(x)
- => $f^{ReAct}(x)=W*ReAct(g(x), c)$, где
  - ReAct(g(x), c) = min(g(x), c)
  - порог *c* выбирается по ID данным, напр., берется 95-й перцентиль

Почему работает: эмпирическое наблюдение, что фичи на OOD данных имеют бОльший разброс значений

Note: совместим с разными OOD detection методами (KNN, энтропия, …)

Domain Adaptation

Domain Adaptation

Классификация задач DA

- По наличию разметки на новых данных:
  - есть -> supervised DA (SDA)
  - нет -> unsupervised DA (UDA)
- По доступности исходных данных:
  - обычная DA
  - source-free DA

UDA: лучшие методы

Критерии:

1. sota
2. простота

Методы:

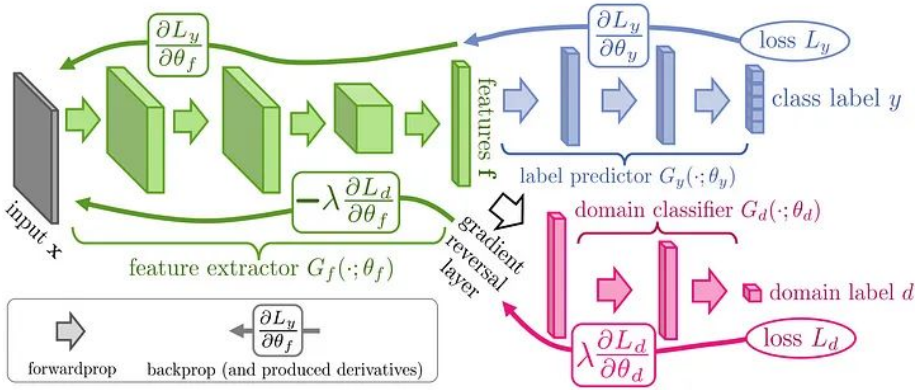
- MCC-DANN - sota согласно UDA: A Reality Check (комбинация 2 методов: MCC, DANN)

DANN

Paper: Y. Ganin et al., 2015 “Domain-Adversarial Training of Neural Networks“ arxiv

DANN:

- разбиваем обучаемую модель h в композицию feature extractor f и классификатора g
- учим h делать классификацию – на сорсе, по-обычному
- учим f выдавать доменно-инвариантные фичи – состязательно
  - обучаем доменный дискриминатор различать фичи сорса и таргета
  - обучаем f обманывать доменный дискриминатор



MCC

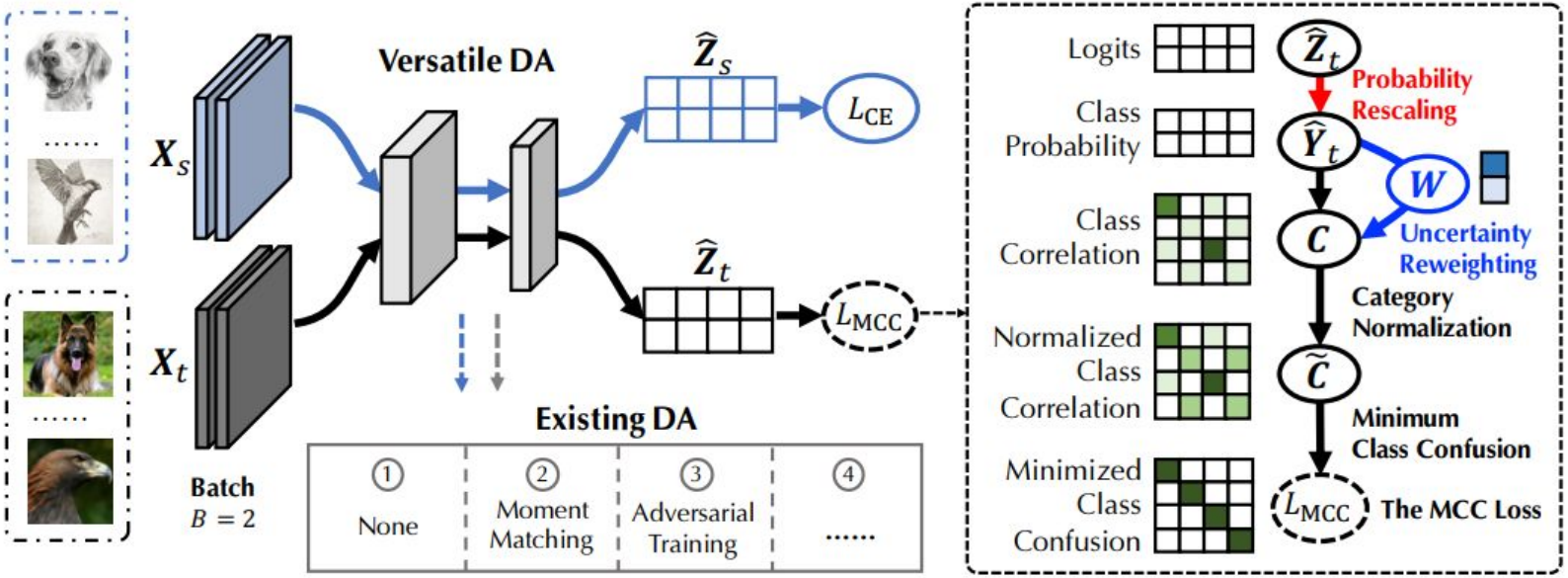
Paper: Y. Jin et al., 2019 “Minimum Class Confusion for Versatile Domain Adaptation” arxiv

Метод **MCC** построения модели h:

- учим h делать классификацию на сорсе, по-обычному
- учим h не сомневаться в выборе между классами на таргете с помощью MCC лосса

MCC лосс

MCC лосс штрафует за неуверенность при выборе из каждой пары классов



MCC лосс vs энтропия

Сравним [упрощенный] MCC лосс и энтропию (H) на 3 предсказательных распределениях:

распределение	MCC	H
[0.5, 0.3, 0.2]	0.68	1.03
[0.6, 0.2, 0.2]	0.56	0.95
[0.6, 0.4, 0]	0.48	0.67

Спасибо за внимание!