Мало данных

Мало данных

Маленький объем обучающей выборки - частая проблема.

Подходы к решению:

- захардкодить знания
  - inductive biases
- увеличить обучающую выборку
  - аугментации
  - active learning
  - weakly-supervised learning
- использовать знания из других задач
  - transfer learning (self-supervised learning)
  - синтетические данные
  - few-shot learning

Inductive bias

Inductive bias

Inductive bias — это априорные знания о природе данных, которые человек некоторым образом вкладывает в модель машинного обучения.

CNNs > трансформеров на малых данных благодаря inductive biases:

- локальность признаков;
- использование 2d структуры;
- трансляционная инвариантность;
- иерархичность признаков;

Можно привнести другие (напр., rotation invariance).

Аугментация данных

Аугментация данных

Опр. Аугментация данных - техника искусственного увеличения датасета путем модификации его элементов.

Почему улучшает обобщающую способность:

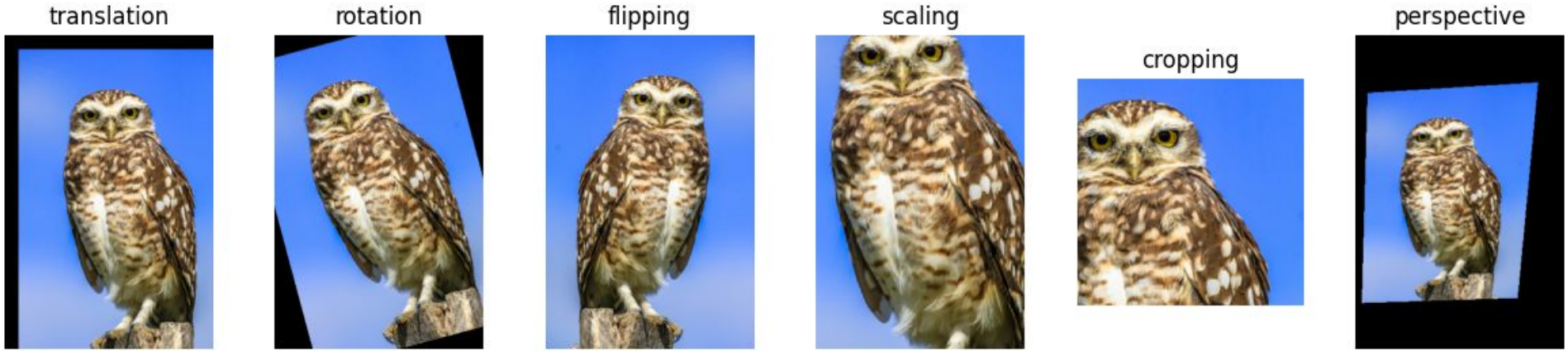
- увеличивает кол-во данных;
- обучает инвариантности к разумным модификациям.

Геометрические преобразования

Геометрические преобразования меняют расположение пикселей, оставляя *неизменными* значения.

Примеры:

- трансляция (translation)
- поворот (rotation)
- отражение (flipping)
- масштабирование (scaling)
- обрезка (cropping)
- изменение перспективы

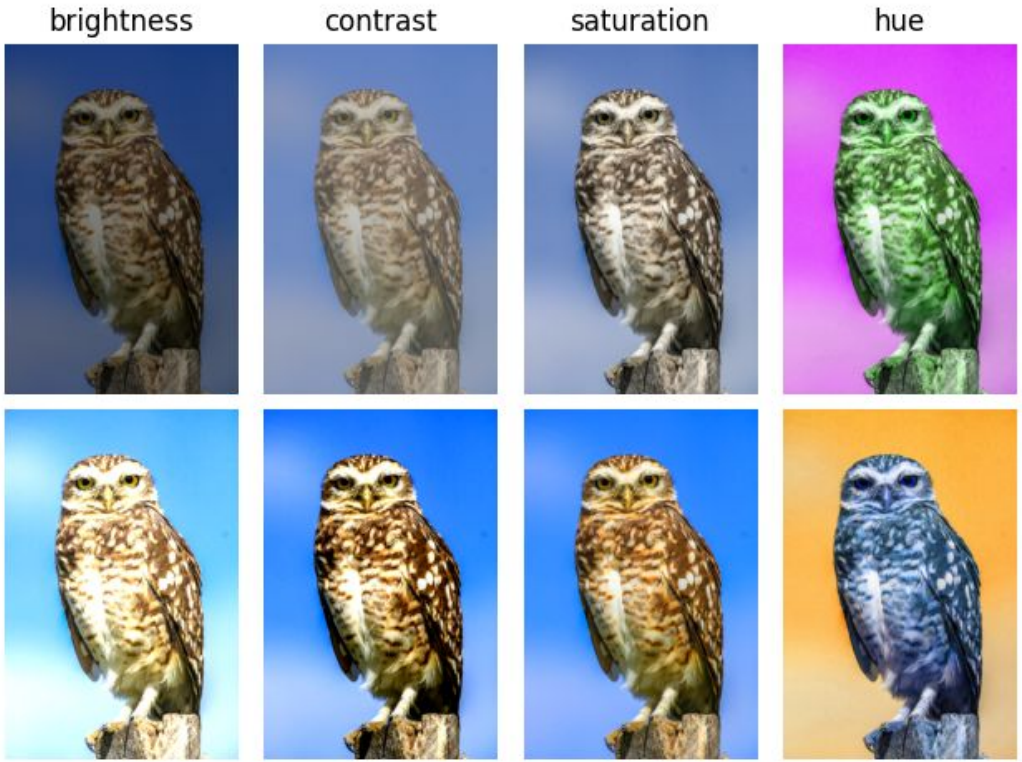


Цветовые преобразования

Цветовые преобразования меняют значения пикселей при неизменном расположении.

Примеры:

- яркость (brightness)
- контраст (contrast)
- насыщенность (saturation)
- оттенок (hue)



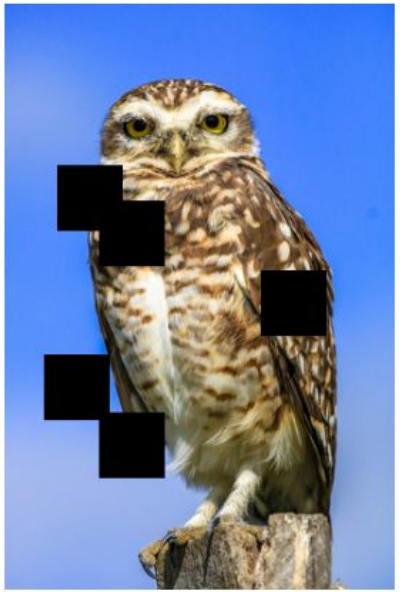
Другие

- точечные шумы
- блюры
- cutout
- …

Gaussian noise

Gaussian blur

cutout



Комбинация изображений

- mixup: линейная комбинация пары изображений
- CutMix: склейка кусков нескольких изображений

Автоматические подбор аугментации

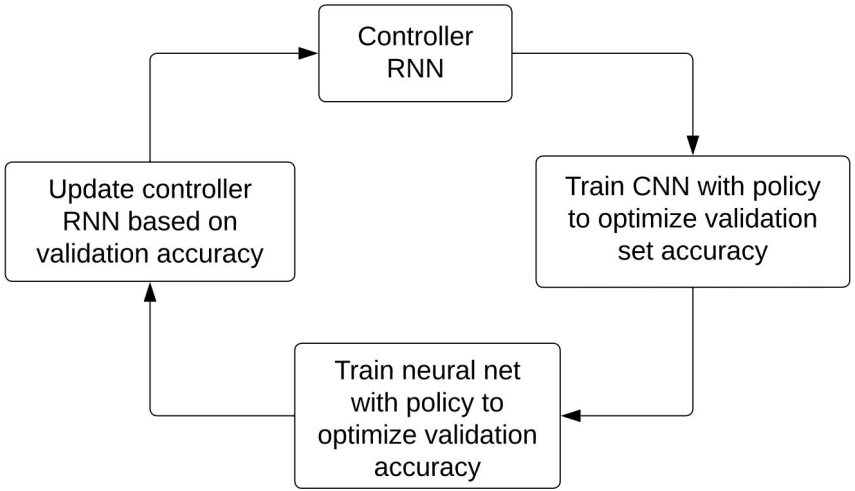
Подбирать вручную - искусство => нужна автоматизация. Напр.,

- AutoAugment
- RandAugment
- …

Сравнение by Sebastian Raschka: post

AutoAugment

AutoAugment - RL метод выбора оптимальной стратегии аугментации.



RandAugment

RandAugment - семейство аугментаций, задаваемое всего 2 гиперпараметрами:

- N - количество последовательных трансформаций;
- M - их магнитуда.

Каждая из N трансформаций выбирается равновероятно из фиксированного набора из K трансформаций.

В оригинальной статье (и имплементации Pytorch):

- N = 1..10
- M = 1..10
- K трансформаций: identity transformation, autoContrast, equalize, rotation, solarixation, colorjittering, posterizing, changing contrast, changing brightness, changing sharpness, shear-x, shear-y, translate-x, translate-y.

Выбор M и N: разные методы оптимизации ГП (напр., grid search).

псевдокод из статьи:

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

  Args:
    N: Number of augmentation transformations to
        apply sequentially.
    M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

Figure 2. Python code for RandAugment based on numpy.

Class-dependent augmentations

Note: одни и те же аугментации могут улучшать перформанс на одних классах и ухудшать на других => стоит делать class-dependent augmentations.

[2023] P. Kirichenko et al. Understanding the Detrimental Class-level Effects of Data Augmentation
https://arxiv.org/abs/2401.01764

Transfer Learning

Transfer Learning

Опр. Перенос обучения (transfer learning) - техника обучения модели, состоящая в использовании знания, выученного на вспомогательной задаче (source task, pretext task), при обучении целевой задаче (target task, downstream task).

TL позволяет

- достичь лучшей обобщающей способности;
- сэкономить ресурсы (сбор и разметка обучающей выборки, компьют).

TL на практике

На практике – через **переиспользование весов**:

1. предобучаем нейронку на вспомогательной задаче (pre-training);
2. тьюним на своем небольшом наборе данных (fine-tuning).

Варианты тьюнинга:

A. feature extraction: заморозить энкодер, обучать только голову
B. fine-tuning: обучать все слои с константным learning rate
C. discriminative fine-tuning (DFT, старое название differential learning rates): тьюнить все слои, но использовать разные значения learning rate на разных группах слоев (на нижних - меньше, на верхних - больше)

Качество: A < B < C

Ресурсы: A << B, C

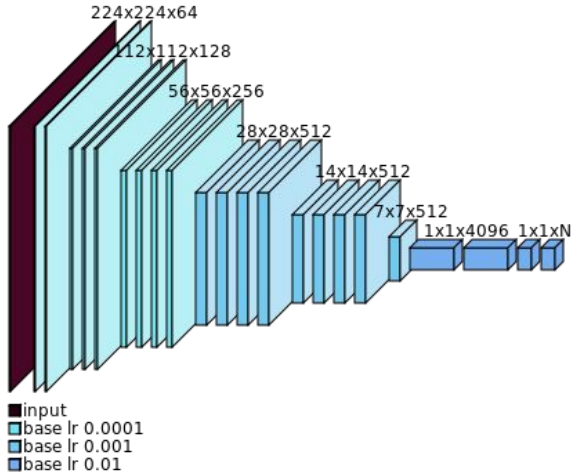
Подробнее про DFT

Мотивация:

- нижние слои выучивают low-level фичи
  -> универсально полезные
  -> их стоит сохранить для большей генерализуемости;
- верхние слои выучивают high-level фичи
  -> специфичные для задачи
  -> их необходимо сильнее менять для большей специфичности.

Эффекты:

- улучшение обобщающей способности;
- более быстрое обучение.



Пример использования DFT для тьюнинга VGG

Вспомогательные задачи для TL

Критерии хороших вспомогательных задач:

- большой размеченный датасет (или псевдо-размеченный);
- из схожего домена.

Примеры:

- классификация на ImageNet для дальнейшей работы с естественными изображениями;
- классификация рентгенов разных частей тела для дальнейшей работы с рентгенами стоп;
- self-supervised learning (SSL);
- …

Self-Supervised Learning

Self-supervised learning (SSL) - парадигма машинного обучения, в рамках которой производится обучение модели на неразмеченных данных по разметке, автоматически извлекаемой из самих данных.

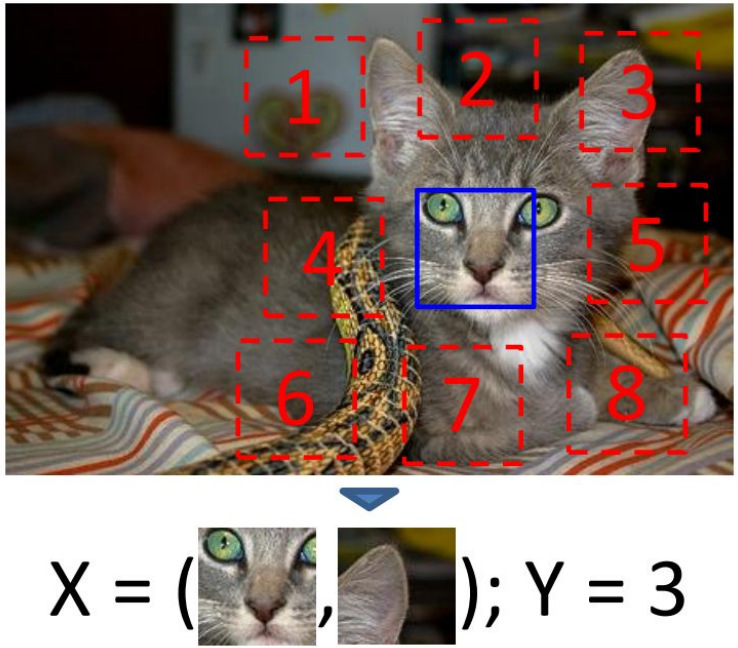
Подходы к извлечению разметки:

- дискриминативный (используется специфическая задача, формируемая на основании доменных знаний);
- генеративный;
- metric learning.

Дискриминативный подход

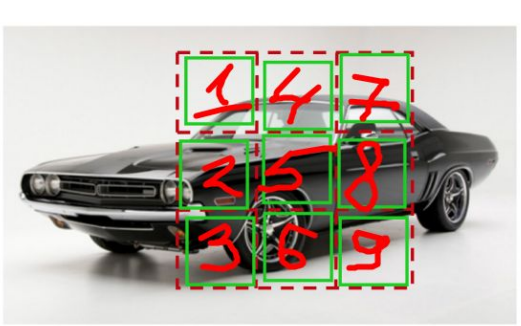
Doersh et al., 2015

Предсказание контекста куска изображения



Дискриминативный подход

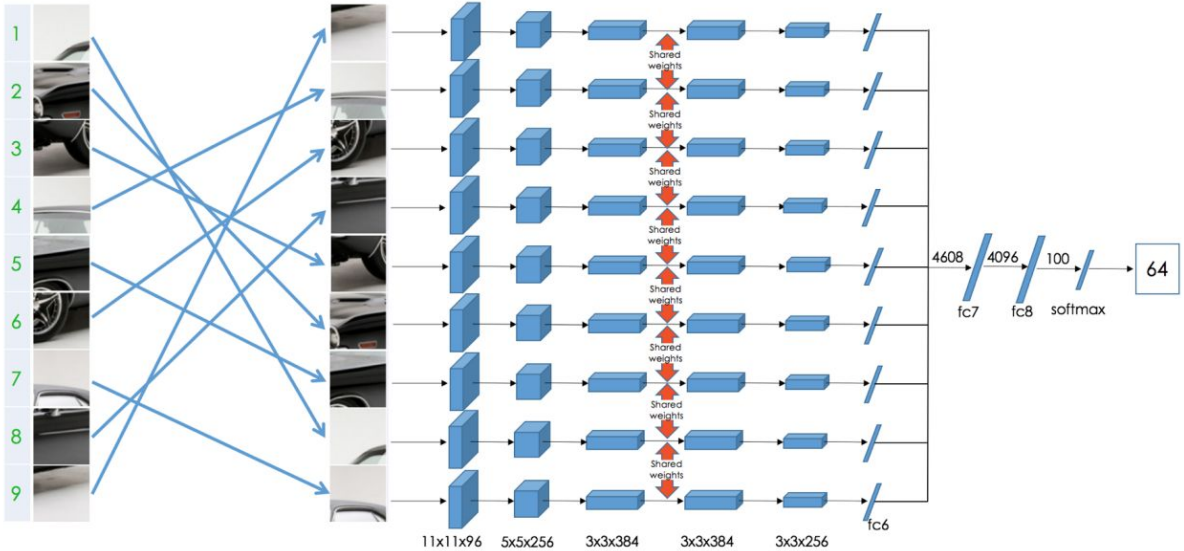
[2016] Noroozi, Favaro Jigsaw puzzle



Permutation Set

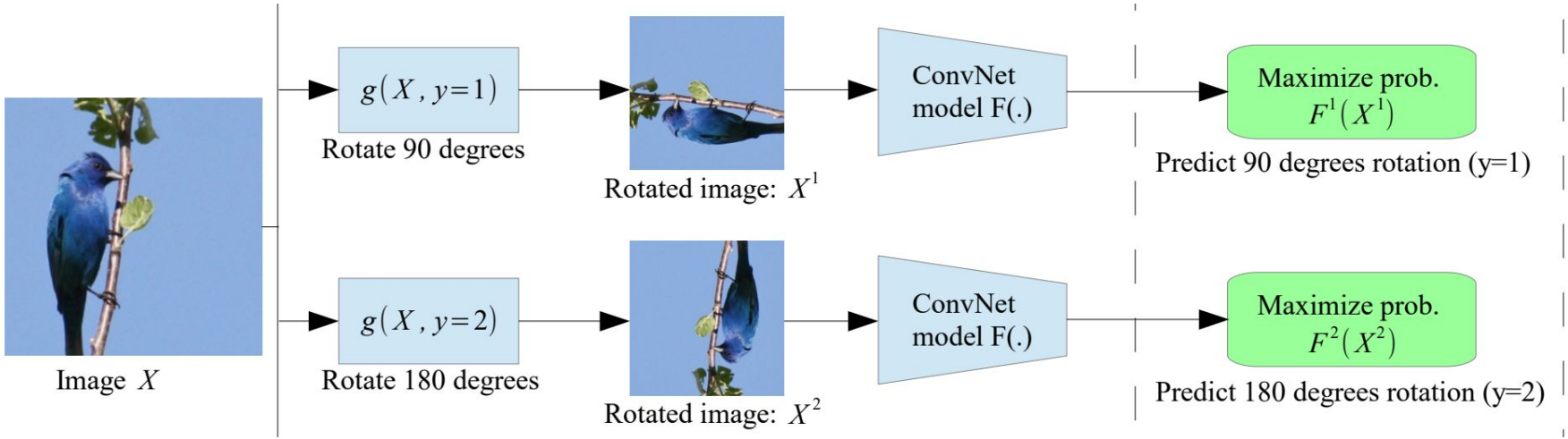
index	permutation
64	9,4,6,8,3,2,5,1,7

Reorder patches according to the selected permutation



Дискриминативный подход

[2018] Gidaris et al. предсказание угла поворота



Дискриминативный подход

Плюсы:

+ обычные задачи распознавания -> требуют не очень много ресурсов

Минусы:

- нужны труд и экспертиза чтобы придумать
- результаты уступают другим подходам

Генеративный подход

[2021] K. He et al. MAE (Masked AutoEncoder)

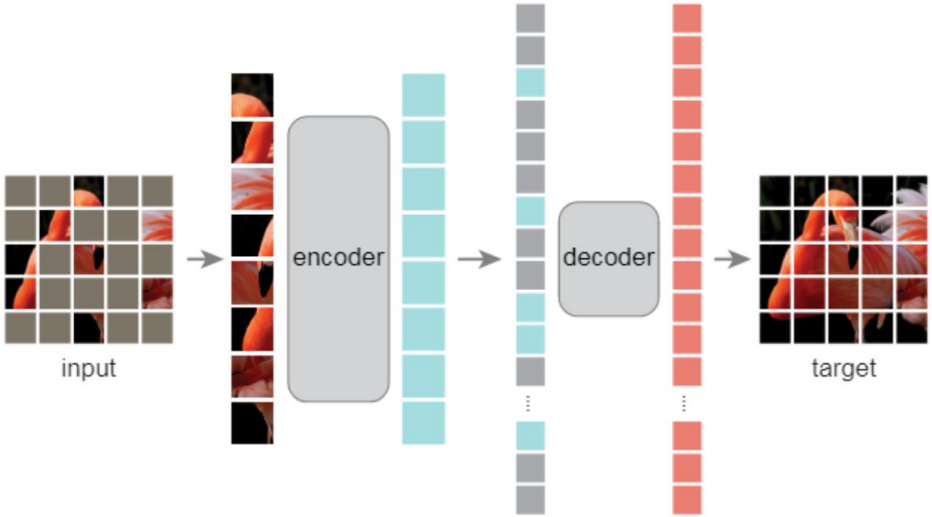
Плюсы:

+ экспрессивные латентные представления

Минусы:

- чувствительность к аутлаерам

source



Metric learning

Опр. Задачей **metric learning** является выучивание функции схожести экземпляров.

На практике: строим энкодер, такой что эмбеддинги похожих экземпляров лежат близко, а разных - далеко.

Контрастивное обучение

Штрафует, за большое расстояние между похожими сэмплами и маленькое между различными.

Примеры: MoCo, SimCLR, MoCo v2, SimCLR v2, BYOL

Контрастивное обучение

Как найти похожие сэмплы?

Набор данных $D = \{x\_1, \ldots, x\_N\}$

- якорь (anchor) $x = x_i$, $1 <= i <= N$
- положительный пример (positive sample) $x^+ = tfm(x)$
- отрицательный пример (negative sample) $x^- = x_j$, $1 <= j <= N, j \neq i$

Функции потерь

Выбирается функция схожести пары эмбеддингов, напр.,

- косинусный коэффициент (cosine similarity) cos_sim
- $||.||_2^2$

$$cos\_sim(A, B) = \frac{A.B}{\|A\|\|B\|}$$

Фукнции потерь:

- margin-based: triplet loss $L_{triplet}$
- Softmax-based:
  - NCE (Noise Contrastive Estimation) L_{NCE}
  - InfoNCE $L_{InfoNCE}$

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$

$$\mathcal{L}_{\text{NCE}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = -\log \frac{\exp(sim(\mathbf{x}, \mathbf{x}^+)/\tau)}{\exp(sim(\mathbf{x}, \mathbf{x}^+)/\tau) + \exp(sim(\mathbf{x}, \mathbf{x}^-)/\tau)}$$

$$\mathcal{L}_{\text{InfoNCE}}(\mathbf{x}, \mathbf{x}^+, \mathbf{y}_1, \ldots, \mathbf{y}_K) = -\log \frac{\exp(sim(\mathbf{x}, \mathbf{x}^+)/\tau)}{\exp(sim(\mathbf{x}, \mathbf{x}^+)/\tau) + \sum_{k=1}^{K} \exp(sim(\mathbf{x}, \mathbf{y}_k)/\tau)}$$

Обозначения:

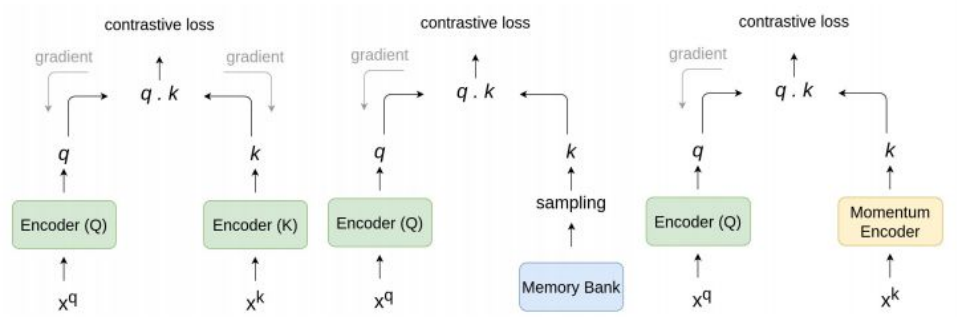
- sim - функция схожести
- τ - гиперпараметр температуры
- y_k - отрицательные примеры

Негативные образцы

Для хорошего перформанса нужно много негативных примеров.

Разные решения:

1. End-2-end (SimCLR, SimCLR v2)
2. memory bank (PIRL, Wu et al., 2018)
3. memory queue + momentum encoder (MoCo)

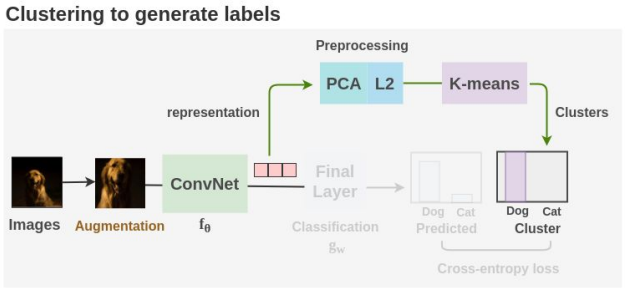


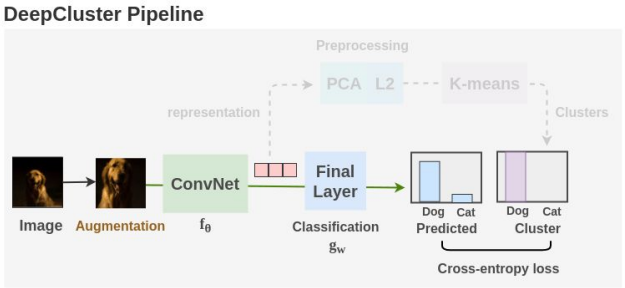
Cluster-based

Штрафует за неправильность кластеризации эмбеддингов.

Примеры: DeepCluster, SwAV.

Схема DeepCluster:





Few-shot Learning

Few-shot Learning

Опр Few-shot learning (FSL) - парадигма ML, позволяющая обучить модель распознавать новые классы по небольшому количеству примеров.

N-way K-shot модель:

- N - количество классов;
- K - количество сэмплов на класс.

Обычно $K <= 10$, чаще всего $K \in \{1, 5\}$.

Основные понятия:

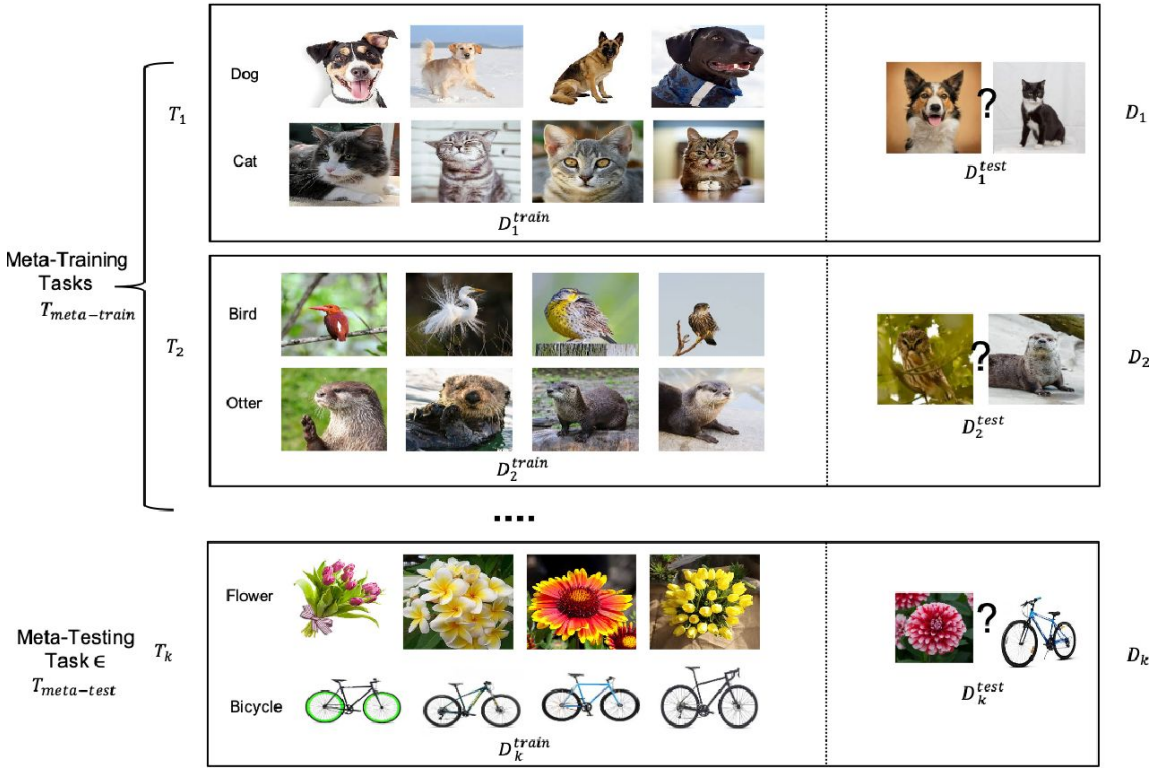
- множество образцов (support set) - множество образцов новых классов;
- множество запросов (query set) - множество сэмплов из новых классов, для которых надо выполнить предсказание.

Постановка задачи

Мета-обучение (meta-learning) - парадигма обучения модели, способной обучаться по новым примерам.

2 фазы:

1. meta-training: обучение на нескольких few-shot эпизодах;
2. meta-testing: применение к новым few-shot эпизодам.



Metric-based

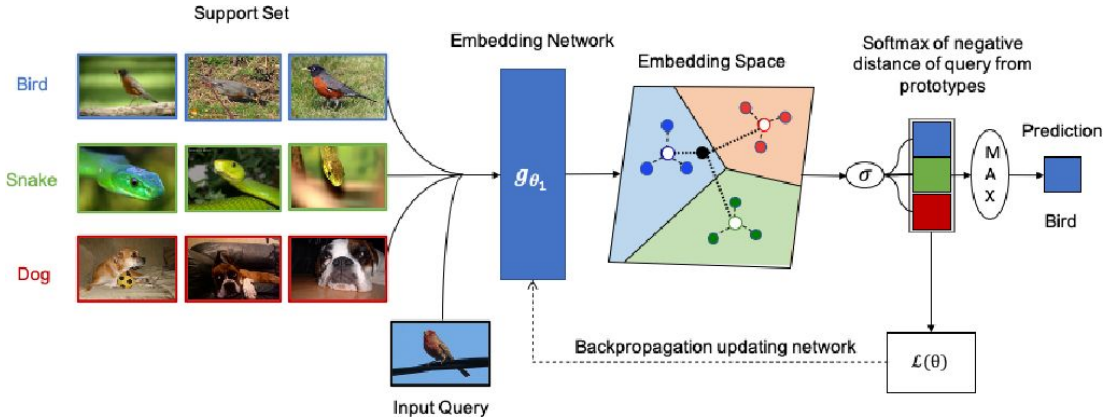
Метрический подход к классификации:

1. берем предобученный энкодер
2. получаем эмбеддинги образцов
3. по эмбеддингам образцов находим центроиды классов
4. получаем эмбеддинги запросов
5. для каждого запроса предсказываем класс ближайшего центроида

Metric-based meta-learning: на фазе мета-обучения выбираем энкодер, хорошо справляющийся с few-shot learning на новых классах.

Возможные вариации:

- не один центроид, а несколько;
- разные энкодеры на образцы и запросы;
- …



Optimization-based

Компоненты:

- лернеры f_θ решают конкретные эпизоды;
- мета-лернер g_φ учится обучать лернеры под задачи.

Сопоставление:

A. Обычное обучение: обучение f_θ на данных градиентным спуском $\theta_{i+1} = \theta_i - \alpha \nabla f(\theta_i)$
B. Optimization-based мета-обучение: апдейт f_θ делаем мета-лернер $\theta_{i+1} = g_i(\nabla f(\theta_i), \theta_i; \phi)$

Optimization-based

Algorithm 2: Train Meta-Learner

Input: Meta-training set $D_{meta-train}$, Learner f with parameters θ, Meta-Learner g with parameters ϕ

$\phi_0 \leftarrow$ random initialization
for $j \leftarrow 1$ **to** J **do**
  $D^{train}, D^{test} \leftarrow$ random dataset from $D_{meta-train}$
  $\theta_0 \leftarrow c_0$
  for $b \leftarrow 1$ **to** B **do**
    $\mathbf{X_b}, \mathbf{Y_b} \leftarrow$ random batch from D^{train}
    $\mathcal{L}_b \leftarrow \mathcal{L}(f(\mathbf{X_b}; \theta_{b-1}), \mathbf{Y_b})$ ▷ Get loss of learner on train batch
    $c_b \leftarrow g((\nabla_{\theta_{b-1}} \mathcal{L}_b, \mathcal{L}_b); \phi_{j-1})$ ▷ Get output of meta-learner
    $\theta_b \leftarrow c_b$ ▷ Update learner parameters
  end
  $\mathbf{X}, \mathbf{Y} \leftarrow D^{test}$
  $\mathcal{L}_{test} \leftarrow \mathcal{L}(f(\mathbf{X}; \theta_b), \mathbf{Y})$ ▷ Get loss of learner on test batch
  Update ϕ_j using $\nabla_{\phi_{j-1}} \mathcal{L}_{test}$ ▷ Update meta-learner parameters
end

Классификация подходов

- meta-learning
  - metric-based
  - optimization-based
  - model-based
- transfer learning
- c-VAE
- …

Обзор: https://arxiv.org/pdf/2203.04291

Другие подходы

Active Learning

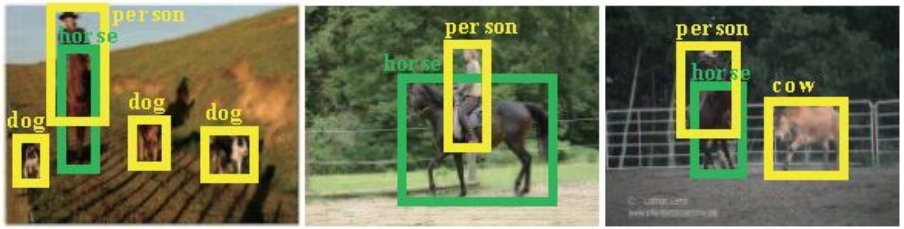
Активное обучение (active learning) - парадигма машинного обучения, при которой для обучения модели предоставляется множество размеченных и множество неразмеченных данных, с возможностью по мере обучения запрашивать доразметку отдельных неразмеченных экземпляров в рамках бюджета.

Что доразмечать: экземпляры, где модель наименее уверена.

Weak Supervision

Частичная разметка (weak supervision)

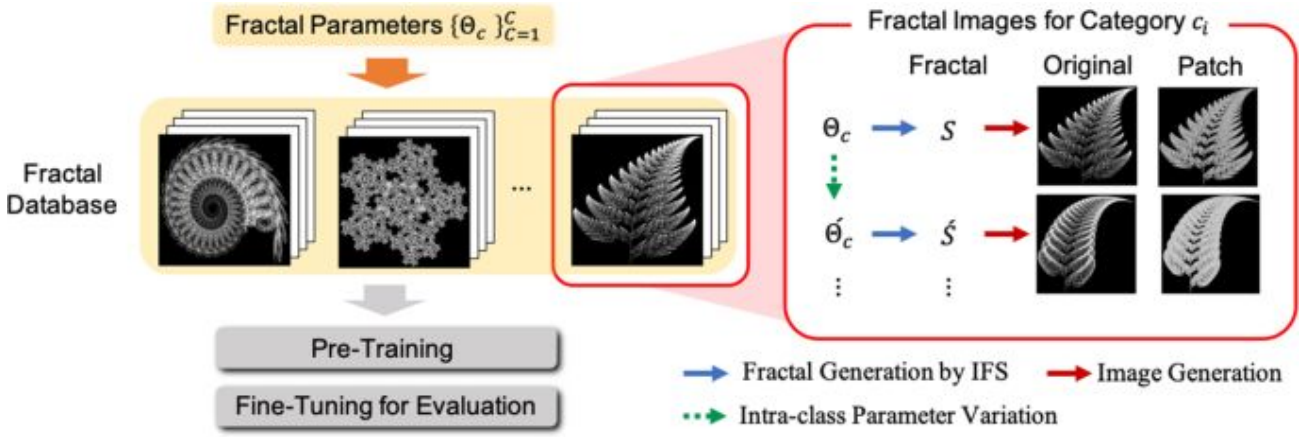
(a)



source: https://www.semanticscholar.org/paper/Weakly-supervised-object-detection-with-correlation-Wan-Wei/6ffe64fd47dff0758a1e22b826a5f63569f6e104

Синтетические данные

Использование синтетических данных (фракталов) может давать предобучение на уровне ImageNet (CNN, трансформеры).



source: https://link.springer.com/article/10.1007/s11263-021-01555-8

Спасибо за внимание!