Московский государственный университет
имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Выпускная квалификационная работа

на тему: «Семантическая сегментация изображений без учителя»

Выполнил:
студент гр. 316
Селезнёв М.В.

Научный руководитель:
д-р ф.-м. наук
Ульянов В.В.

Москва 2021

Оглавление

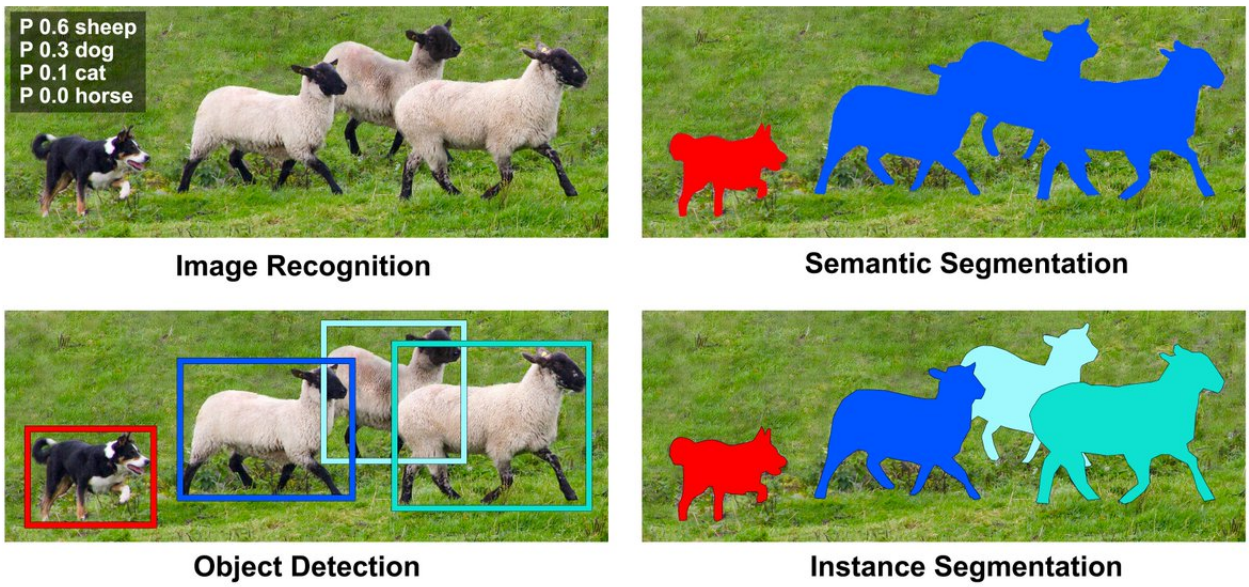
1 Введение

Работа посвящена задаче сегментации изображений без учителя (unsupervised image segmentation).

В начале будет объяснена мотивация выбора задачи сегментации без учителя, а также приведена некоторая формализация. Затем будут описаны способы её решения. Предложено также расширение одного из существующих методов.

2 Постановка задачи

Рис. 1: Семантическая и «объектная» сегментация



Сегментация — это задача разбиения изображения на области. Выделяют семантическую сегментацию и «объектную» сегментацию [1].

При семантической сегментации каждому пикселю сопоставляется класс объекта, в котором он находится. Пример можно видеть на рис. 1, в верхнем правом углу. Следует обратить внимание, что если на изображении есть несколько объектов одного класса, то они все обозначаются одним цветом.

При «объектной» сегментации (справа снизу на рис. 1) требуется не только определять класс, но и отличать разные объекты одного класса.

Потребность в сегментации изображений возникает в медицине, в распознавании лиц, в системах управления беспилотными автомобилями и многих других областях. Это актуальная задача, имеющая важное прикладное значение.

[1] Semantic и Instance segmentation соответственно. Для последней устоявшегося русскоязычного термина нет.

Обучение с учителем и без учителя

Задача сегментации сейчас решается методами машинного обучения. Поэтому для дальнейшего изложения понадобится ввести понятия обучения с учителем и без учителя.

Обучение с учителем (supervised learning) — это сеттинг, в котором мы предполагаем, что есть множество объектов $\mathcal{X}$, множество ответов $\mathcal{Y}$ и существует некоторое неизвестное отображение $f\colon \mathcal{X} \mapsto \mathcal{Y}$. Нам дана обучающая выборка $\big(\mathbb{X}, \mathbb{Y}\big) = \{(x_i, y_i)\colon x_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ f(x_i) = y_i,\ i = \overline{1,n}\}$, и требуется восстановить неизвестное отображение f.

В эту парадигму укладываются, например, задачи классификации и регрессии. В первом случае множество $\mathcal{Y}$ конечно. Можно без ограничения общности считать, что $\mathcal{Y} = \{1, \ldots, K\}$, где K - число классов. Во втором случае множество $\mathcal{Y}$ бесконечно. Чаще всего $\mathcal{Y} = \mathbb{R}$.

Множество $\mathcal{X}$, как правило, является некоторым подмножеством $\mathbb{R}^d$. Компоненты вектора $x = (x_1, \ldots, x_d)$ называются признаками. В некоторых случаях объекты имеют сложную структуру — например, если мы рассматриваем графы или временные ряды, и найти хорошее признаковое описание (т.е., представить объект в виде вектора чисел) само по себе является нетривиальной задачей. При работе с изображениями, к счастью, есть довольно естественный путь — использовать его RGB-представление.

При обучении с учителем обычно вводят некоторый параметрический класс функций $\mathcal{F} = \{f_\theta\colon \mathcal{X} \mapsto \mathcal{Y},\ \theta \in \Theta\}$, в котором будет искаться решение, а также функцию потерь $\mathcal{L}$.

Примерами множества $\mathcal{F}$ могут служить линейные комбинации признаков в случае линейной регрессии, кусочно-постоянные функции в случае решающих деревьев или более общие и сложно описываемые классы в случае нейронных сетей.

Функция потерь используется для формализации понятия качества модели $f_\theta \in \mathcal{F}$. Примерами функций потерь для задачи регрессии могут является среднеквадратичная ошибка и среднее абсолютное отклонение; для задачи классификации — кросс-энтропия.

Процесс обучения заключается в поиске такого $\theta^* \in \Theta$, что f_{θ^*} наилучшим образом приближает f в смысле минимизации выбранной функции потерь $\mathcal{L}$.

Обучение с учителем удобно тем, что допускает довольно стройное математическое описание, и по существу сводится к оптимизации некоторого функционала.

Обучение без учителя (unsupervised learning) описать несколько сложнее. К нему относят такие задачи, как кластеризация, оценка плотности, понижение размерности, визуализация, поиск аномалий. Можно сказать, что в этом

случае мы имеем некоторый набор данных $\mathbb{X} = \{x_1, \ldots, x_n\}$ объектов из множества $\mathcal{X}$ и пытаемся найти в нём некоторые структурные особенности.

Вернёмся к сегментации. Понятно, что если у нас есть набор картинок, для которых известна правильная разметка, то мы можем рассматривать нашу задачу как задачу обучения с учителем. А более конкретно, как классификацию пикселей.

Однако, специфика сегментации изображений заключается в том, что создание обучающей выборки крайне трудоёмко. Мы должны сопоставить класс каждому пикселю изображения, а на одной картинке разрешения 1024×1024 их уже более миллиона. В то же время для обучения качественной модели требуются, как правило, тысячи изображений. Более того, разметка, скажем, медицинских изображений не может быть выполнена случайным человеком — требуется привлечение специалиста.

Эти проблемы мотивируют желание интерпретировать сегментацию изображений как задачу обучения без учителя, или по крайней мере снизить объём размеченных данных, необходимых для обучения.

3 Методы

Наилучшие результаты в большинстве задач обработки изображений достигаются с использованием нейронных сетей. Нейросеть f — это отображение из некоторого параметризованного класса

$$\mathcal{F} = \{f_\theta \colon \mathcal{X} \mapsto \mathcal{Y}, \theta \in \Theta \subset \mathbb{R}^D\}. \tag{1}$$

Очень часто нейросеть можно представить в виде суперпозиции отображений $f = h \circ g,\ g \colon \mathcal{X} \mapsto \mathcal{Z},\ h \colon \mathcal{Z} \mapsto \mathcal{Y}$. Иными словами, объект x сперва преобразовывается в некоторое промежуточное (латентное, скрытое) представление $z = g(x)$, а затем уже по скрытому представлению z делается предсказание $\hat{y} = h(z)$. Таким образом, отображение g осуществляет преобразование признакового пространства. В англоязычной литературе функцию g обычно называют «feature extractor» или «encoder». Функцию h называют «head» или «decoder».

Сегментация с учителем может рассматриваться как классификация пикселей. В таком случае объекты $x \in \mathcal{X}$ — это векторные представления пикселей. Для каждого пикселя x строится его латентное представление $z = h(x)$, из которого decoder предсказывает семантический класс. Как отсюда перейти к unsupervised-сценарию?

Аналог классификации в режиме без учителя — это кластеризация. Если мы можем обучить encoder без доступа к меткам так, чтобы представления

пикселей образовывали семантические кластера, то можно заменить decoder на стандартный метод кластеризации (скажем, KMeans). По существу, на этом и основано большинство методов сегментации без учителя, и их отличия заключаются в том, каким образом обучается encoder.

Иногда у нас всё-таки есть небольшое количество размеченных изображений, и хочется их использовать. Поэтому интересно рассматривать также и методы Few-Shot сегментации. В них, как правило, тоже сначала обучается encoder без доступа к меткам (стадия *предобучения*), а потом происходит *дообучение* на имеющихся изображениях (либо всей сети $f = h \circ g$, либо обучается только лёгкий decoder h). На самом деле, предобученный encoder часто можно использовать для разных задач — детекции, image retrieval, copy detection и другие. На практике это часто является преимуществом, поэтому я буду упоминать об этом, говоря о методах, которые это позволяют.

Итак, вопрос сводится к тому, что надо придумать способ предобучения encoder-a, который не требует вручную размеченных данных. Давайте посмотрим, какие существуют решения.

3.1 IIC

В одно предложение: максимизация взаимной информации между предсказаниями сегментации для двух версий одного изображения.

Бóльшая часть алгоритмов сегментации без учителя на момент написания этой статьи базировалась на попытке использовать нейронные сети и классические алгоритмы кластеризации. Такие решения страдали от нестабильного обучения и вырожденных решений — когда всё обьединялось в один кластер, или в какие-то кластера не попадало ни одной точки. Авторы статьи Invariant Information Clustering предложили новую функцию потерь, которая естественным образом позволила избежать проблемы вырожденности. Этот подход также позволил хорошо решить задачу кластеризации изображений.

Функция потерь

Пусть $x, x' \in \mathcal{X}$ — это пара из совместного распределения $\mathcal{P}(x, x')$. Например, это могут быть разные изображения одного объекта. Цель IIC — выучить encoder $\Phi\colon \mathcal{X} \mapsto \mathcal{Y}$, который будет сохранять в латентном представлении $z = \Phi(x)$ существенные особенности объекта и отбрасывать незначительные детали. Это формализуется с использованием понятия взаимной информации:

$$\Phi^* = \underset{\Phi}{\operatorname{argmax}}\, I\left(\Phi(x), \Phi(x')\right) \tag{2}$$

В процессе решения такой задачи оптимизации представления $z = \Phi(x), z' = \Phi(x')$ становятся всё ближе друг к другу. Однако, в отличии от методов, основанных на KMeans, здесь не появляется вырождённых решений, так как взаимная информация включает в себя безусловную и условную энтропию. А именно,

$$I(z, z') = H(z) - H(z \,|\, z') \tag{3}$$

Максимальное значение для $H(z)$ равно $\ln C$, где C — число кластеров. Оно достигается, когда все кластеры равновероятны. Минимальное значение $H(z \,|\, z')$ — 0, достигается, когда z идеально восстанавливается по z'. Таким образом, значение взаимной информации не будет максимальным, если всё принадлежит одному кластеру. Можно рассматривать безусловную энтропию как регуляризацию.

Важно отметить, что отображение Φ должно иметь ограниченную выразительность. В противном случае задача максимизации решается тривиально — достаточно положить $\Phi(x) = x$. Часто можно избежать этого снижением размерности латентного пространства $\mathcal{Z}$. В случае кластеризации пикселей есть также и другой способ: на выходе для каждого пикселя мы ожидаем один из классов $y \in \mathcal{Y} = \{1, \ldots, C\}$. Авторы рассматривают нечёткую кластеризацию, поэтому для каждого пикселя его представление — это дискретное распределение по C кластерам. k-я компонента интерпретируется как оценка вероятности принадлежности кластеру с номером k: $\mathbb{P}(z = c \,|\, x) = \Phi_c(x)$. Это тоже не позволяет просто скопировать вход.

Пусть z, z' — это матрицы, в которых каждому пикселю соответствует метка определённого кластера. Чтобы посчитать взаимную информацию, нам нужно знать условное и безусловное распределения (z, z') относительно (x, x'). Будем полагать, что

$$\mathbb{P}\left(z = c, z' = c' | x, x'\right) = \Phi_c(x)\Phi_{c'}(x') \tag{4}$$

То есть, z и z' условно независимы при известных x, x'. При этом они не являются безусловно независимыми: например, если x, x' — это изображения одного и того же объекта со случайными смещениями, то z, z' будут сильно скоррелированы (в идеальном случае — совпадающими). Матрица $\boldsymbol{P} \in \mathbb{R}^{C \times C}$, описывающая совместное распределение z, z', будет оцениваться так:

$$\boldsymbol{P} = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i) \cdot \Phi(x_i')^T \tag{5}$$

Одномерные распределения $\boldsymbol{P}_c = \mathbb{P}(z = c), \boldsymbol{P}_{c'} = \mathbb{P}(z' = c')$ можно получить суммированием столбцов и строк соответственно. Наряду с парой (x, x') можно было рассмотреть и пару (x', x), поэтому матрица $\boldsymbol{P}$ симметризуется путём $\boldsymbol{P} = (\boldsymbol{P} + \boldsymbol{P}^T)/2$.

Теперь мы можем оценить взаимную информацию:

$$I(z, z') = \sum_{c=1}^{C} \sum_{c'=1}^{C} \boldsymbol{P}_{cc'} \ln \frac{\boldsymbol{P}_{cc'}}{\boldsymbol{P}_c \cdot \boldsymbol{P}_{c'}} \tag{6}$$

Чтобы получить функцию потерь, достаточно взять взаимную информацию с минусом.

Реализация

Ранее мы предположили, что у нас есть пара (x, x') из распределения $\mathcal{P}(x, x')$. Чтобы смоделировать это, мы можем использовать аугментации — случайные искажения исходного изображения, как то: масштабирование, отражение, поворот, изменение контраста или насыщенности цветов, и другие преобразования, которые не меняют семантику изображения. В таком случае IIC выучивает encoder, инвариантный к указанным трансформациям.

В некоторые датасетах (например, STL10) данные делятся на два типа: изображения первого типа содержат только релевантные семантические классы, а во втором могут быть неизвестные, или «отвлекающие» классы. Нам хотелось бы использовать для обучения и данные второго типа, так как их зачастую гораздо больше (например, 100 тысяч изображений вместо 13 тысяч в STL10). Авторы IIC предложили следующий приём: добавить к выходу свёрточной части нейросети ещё один линейный слой с большим количеством кластеров (overclusterization), и обучать эту ветвь нейросети на всех данных, а ветвь с основным линейным слоем (где число кластеров равно числу классов) — только на данных первого типа. Таким образом свёрточная часть нейронной сети (ResNet или VGG-11-like) обучается с использованием всех данных, и это даёт прирост качества (архитектуру можно видеть на рисунке 2).

Чтобы применить IIC к задаче сегментации, достаточно внести две модификации. Во-первых, кластеризация применяется к участкам изображения, размер которых определяется рецептивным полем нейронной сети для каждого пикселя. Во-вторых, следует учесть локальную пространственную инвариантность. Проще говоря, близлежащие пиксели должны иметь схожие метки. А формально — пусть $x \in \mathbb{R}^{3 \times H \times W}$ есть RGB-представление изображения, $u \in \Omega = \{1, \ldots, H\} \times \{1, \ldots, W\}$ – позиция пикселя, и x_u — участок с центром в u. Мы можем сформировать пару (x_u, x_{u+t}), где $t \in \mathbb{Z}^2$ — небольшие смещения. Оптимизируя функцию потерь IIC для такой пары, мы будем добиваться желаемой локальной пространственной инвариантности.

Вообще говоря, мы можем применять одновременно и фотометрические, и геометрические, и пространственные трансформации. Надо лишь следить за

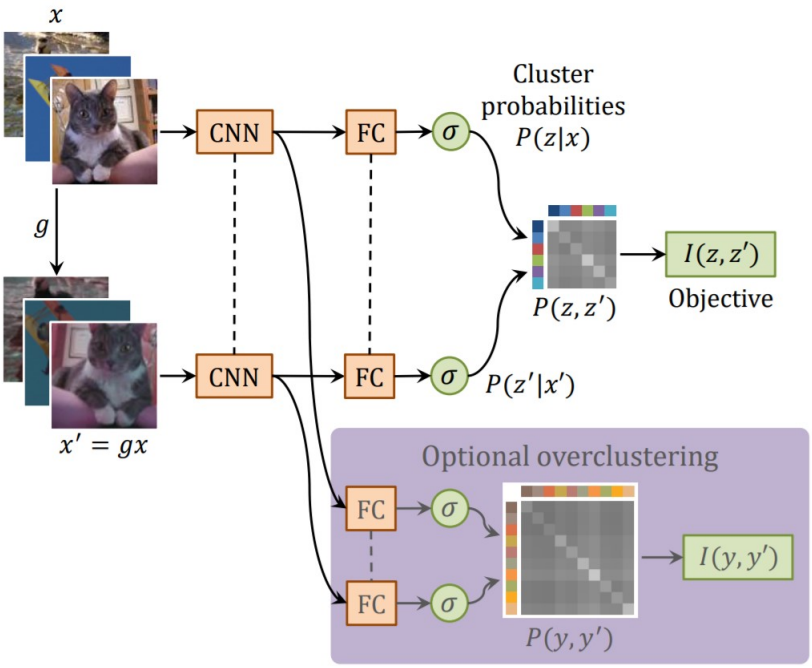


Рис. 2: IIC для кластеризации изображений. Пунктирные линии означают общие параметры, g — случайная аугментация, I — взаимная информация.

тем, что мы сравниваем соответствующие участки. Если g — геометрическое преобразование (например, отражение), то $\Phi_u(x)$ соответствует не $\Phi_u(gx)$, а $\Phi_{g(u)}(gx)$, так как x_u был отображён в $x_{g(u)}$. Однако, пользуясь тем, что $\left[g^{-1}\Phi(gx)\right]_u = \Phi_{g(u)}(gx)$, можно сопоставить все участки одновременно.

Итого, формулировка задачи для сегментации:

$$\max_{\Phi} \frac{1}{T} \sum_{t \in T} \boldsymbol{I}(\boldsymbol{P}_t), \tag{7}$$

$$\boldsymbol{P}_t = \frac{1}{n|G||\Omega|} \sum_{i=1}^{n} \sum_{g \in G} \sum_{u \in \Omega} \Phi_u(x_i) \cdot \left[g^{-1}\Phi(gx)\right]_{u+t}^{T} \tag{8}$$

Иными словами, мы максимизируем взаимную информацию между двумя соседними участками, один из которых дополнительно аугментирован, причём взаимная информация усредняется по соседям. Совместное распределение в уравнении 8 можно эффективно посчитать, заменяя внутреннюю сумму на операцию свёртки.

Результаты

Авторы используют датасеты COCO-Stuff и Postdam. COCO-Stuff — это датасет для сегментации, в котором размечен фон (трава, небо, река и т.д.). Авторы используют вариант с 15-ю классами, причём предварительно удаляют изображения, на который доля фоновых пикселей меньше 75% (видимо,

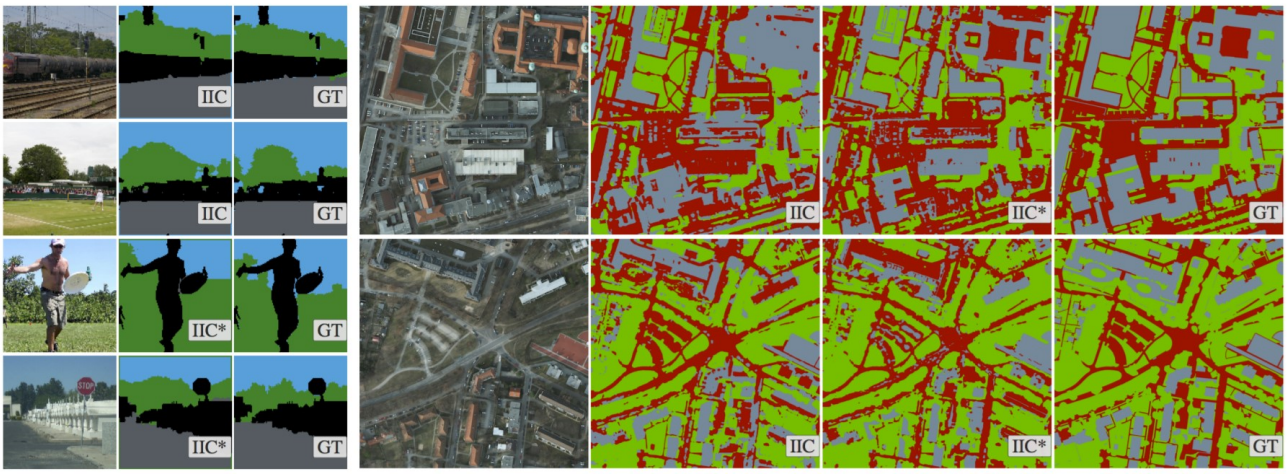


Рис. 3: Слева: COCO-Stuff-3 (объекты отмечены чёрным цветом), справа: Postdam-3. Исходные изображения, IIC, IIC*, GT (ground truth – истинная разметка)

метод хорошо работает именно для сегментации фона, и таким образом ему даётся преимущество). COCO-Stuff-3 — подмножество COCO-Stuff, где всего три класса: небо, земля и растения. При оценке качества предсказания для не фоновых пикселей игнорируются.

Postdam содержит 8550 снимков со спутников, из которых 3150 не размечены. Изначальная версия содержит 6 классов (дороги и машины, растительность и деревья, строения и беспорядок), но авторы также использовали Postdam-3, попарно объединив указанные классы.

На рисунке 3 можно видеть примеры сегментации с помощью IIC и IIC*. Первое — полностью unsupervised метод, второй предобучается без меток и с использованием overclusterization, а затем дообучается на небольшом числе изображений (например, в 10 раз меньше, чем нейросеть с учителем).

На рисунке 4 показаны результаты IIC и предшествующих методов. Измеряется метрика accuracy, усреднённая по пикселям (предсказания для пикселей объектов, не принадлежащих рассматриваемым семантическим классам, не учитываются). В случае IIC требуется сопоставить обнаруженные нейросетью кластеры с семантическими классами. Это делается с помощью венгерского алгоритма. Авторы подчёркивают, что хотя на этом этапе используются метки, это не является частью обучения, а просто гарантирует инвариантность результатов относительно порядка кластеров. Для IIC* кластеров больше, чем семантических классов, и набор кластеров может соответствовать одному классу. Чтобы найти отображение из кластеров в семантические классы, опять надо использовать метки. В этом случае для корректности нужно использовать лишь метки из train, так как в процессе уже существенно используется информация о метках.

	COCO-Stuff-3	COCO-Stuff	Potsdam-3	Potsdam
Random CNN	37.3	19.4	38.2	28.3
K-means [44]†	52.2	14.1	45.7	35.3
SIFT [39]‡	38.1	20.2	38.2	28.5
Doersch 2015 [17]‡	47.5	23.1	49.6	37.2
Isola 2016 [30]‡	54.0	24.3	63.9	44.9
DeepCluster 2018 [7]† ‡	41.6	19.9	41.7	29.2
IIC	**72.3**	**27.7**	**65.1**	**45.4**

Рис. 4: † – Методы, основанные на KMeans. ‡ – Методы, которые требуют применения KMeans.

Авторы IIC достигли заметного прогресса в задаче сегментации без учителя и реализовали интересную идею с максимизацией взаимной информации, что позволило избежать вырожденных решений. Такой подход, тем не менее, имеет свои недостатки — он хорошо работает лишь для фона и плохо справляется с мелкими деталями.

3.2 PiCIE

В одно предложение: попеременное обучение признаковых описаний пикселей и кластеризация, признаковые описания инвариантны относительно фотометрических и эквивариантны относительно геометрических преобразований.

Как уже было сказано, задачу семантической сегментации без учителя можно рассматривать как кластеризацию пикселей. В таком случае основная проблема заключается в построении качественного признакового описания. К сожалению, здесь возникает проблема курицы и яйца: чтобы получить семантически осмысленные (т.е. разнесённые по кластерам) описания пикселей,

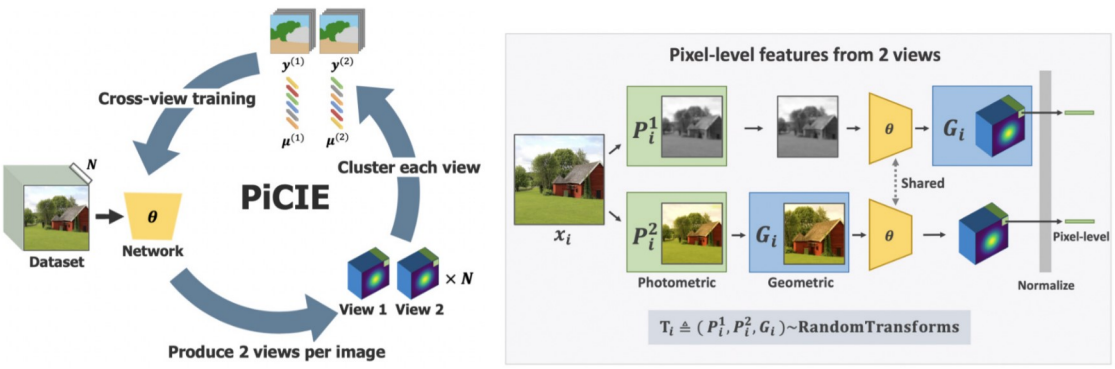


Рис. 5: Схема PiCIE

нужны метки семантических классов, которые мы и пытаемся получить — иначе нейросети неоткуда брать сигнал для обучения.

Решение этой проблемы было предложено в работе Deep Cluster: можно кластеризовать текущие представления и использовать кластера в качестве меток, чтобы обучить новые представления, и повторять этот процесс итеративно. Для обучения encoder-а f можно использовать следующую функцию потерь:

$$\mathcal{L}_{\text{clust}}(z_{ip}, y_{ip}, \boldsymbol{\mu}) = -\log\left(\frac{e^{-d(z_{ip},\mu_{y_{ip}})}}{\sum_l e^{-d(z_{ip},\mu_l)}}\right) \tag{9}$$

где $d(\cdot,\cdot)$ — косинусное расстояние, $z_{ip} = f(x_i)_p$ — признаковое описание p-го пикселя i-го изображения, y_{ip} — кластер этого пикселя, $\mu_{y_{ip}}$ — соответствующий центроид, $\boldsymbol{\mu} = (\ldots, \mu_l, \ldots)$ — все центроиды. По смыслу это кросс-энтропия, а косинусное расстояние с другим знаком используется в качестве логитов: чем меньше расстояние, тем больше логит и тем выше вероятность правильного кластера.

В процессе мы можем надеяться получить достаточно компактные кластера пикселей. Однако, мы никак не контролируем их семантическую осмысленность.

Авторы PiCIE предлагают использовать инвариантность относительно фотометрических преобразований и эквивариантность относительно геометрических преобразований. Иными словами, метка пикселя не должна изменяться при небольших колебаниях цвета, а если изображение, например, отражено относительно вертикальной оси, то метки также должны быть отражены. Более формально, если P_1, P_2 — некоторые фотометрические преобразования, G — геометрическое преобразование, f — нейронная сеть, осуществляющая сегментацию, то мы хотим, чтобы

$$f\big(G\big(P_1(x)\big)\big) = G\big(f\big(P_2(x)\big)\big). \tag{10}$$

Реализовать такое ограничение не так просто — псевдоразметка получается кластеризацией имеющихся представлений, которые, в свою очередь, зависят от фотометрических преобразований. Как же добиться инвариантности в кластеризации?

Пусть x_i — это i-е изображение. Для него случайно выбираются фотометрические преобразования $P_i^{(1)}, P_i^{(2)}$. Для каждого пикселя каждого изображения мы получаем по два векторных представления

$$z_{ip}^{(1)} = f(P_i^{(1)}(x_i))_p \tag{11}$$

$$z_{ip}^{(2)} = f(P_i^{(2)}(x_i))_p \tag{12}$$

Далее производится кластеризация

$$\boldsymbol{y}^{(1)}, \boldsymbol{\mu}^{(1)} = \underset{\boldsymbol{y},\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{i,p} \|z_{ip}^{(1)} - \mu_{y_{ip}}\|^2 \tag{13}$$

$$\boldsymbol{y}^{(2)}, \boldsymbol{\mu}^{(2)} = \underset{\boldsymbol{y},\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{i,p} \|z_{ip}^{(2)} - \mu_{y_{ip}}\|^2 \tag{14}$$

Имея два набора центроидов и меток, можно подсчитать значение функции потерь. Она состоит из двух слагаемых. Во-первых, мы хотим, чтобы пиксели хорошо разделялись на кластера:

$$\mathcal{L}_{\text{within}} = \sum_{i,p} \mathcal{L}_{\text{clust}}(z_{ip}^{(1)}, y_{ip}^{(1)}, \boldsymbol{\mu}^{(1)}) + \sum_{i,p} \mathcal{L}_{\text{clust}}(z_{ip}^{(2)}, y_{ip}^{(2)}, \boldsymbol{\mu}^{(2)}) \tag{15}$$

Во-вторых, мы хотим, чтобы кластеризация была инвариантна к фотометрическим преобразованиям, так что появляется ещё одно слагаемое:

$$\mathcal{L}_{\text{cross}} = \sum_{i,p} \mathcal{L}_{\text{clust}}(z_{ip}^{(1)}, y_{ip}^{(2)}, \boldsymbol{\mu}^{(2)}) + \sum_{i,p} \mathcal{L}_{\text{clust}}(z_{ip}^{(2)}, y_{ip}^{(1)}, \boldsymbol{\mu}^{(1)}) \tag{16}$$

Итоговая функция потерь получается суммированием:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cross}} + \mathcal{L}_{\text{within}} \tag{17}$$

Чтобы добиться эквивариантности к геометрическим преобразованиям, достаточно положить в вышеописанной схеме

$$z_{ip}^{(1)} = f\big(G(P_i^{(1)}(x_i))\big)_p \tag{18}$$

$$z_{ip}^{(2)} = G\big(f(P_i^{(2)}(x_i))_p\big) \tag{19}$$

где G — случайно выбранная геометрическая трансформация (отражение, приближение и т.д.).

Для стабилизации обучения оказалось полезно одновременно оптимизировать эту же функцию потерь с большим числом кластеров. Обычно при использовании нескольких функций потерь коэффициенты перед ними трактуются как гиперпараметры. Однако, в полностью unsupervised-режиме подбор гиперпараметров невозможен, так как за неимением меток мы не можем оценить качество разных гиперпараметров. Авторы балансируют вклад основного и вспомогательного лосса следующим образом:

$$\mathcal{L} = \frac{\log K_2}{\log K_1 + \log K_2}\mathcal{L}_{K_1} + \frac{\log K_1}{\log K_1 + \log K_2}\mathcal{L}_{K_2} \tag{20}$$

Интуиция — кросс-энтропия логарифмически зависит от числа кластеров, так что следует уменьшить вклад дополнительной функции потерь, дабы она не подавляла основную.

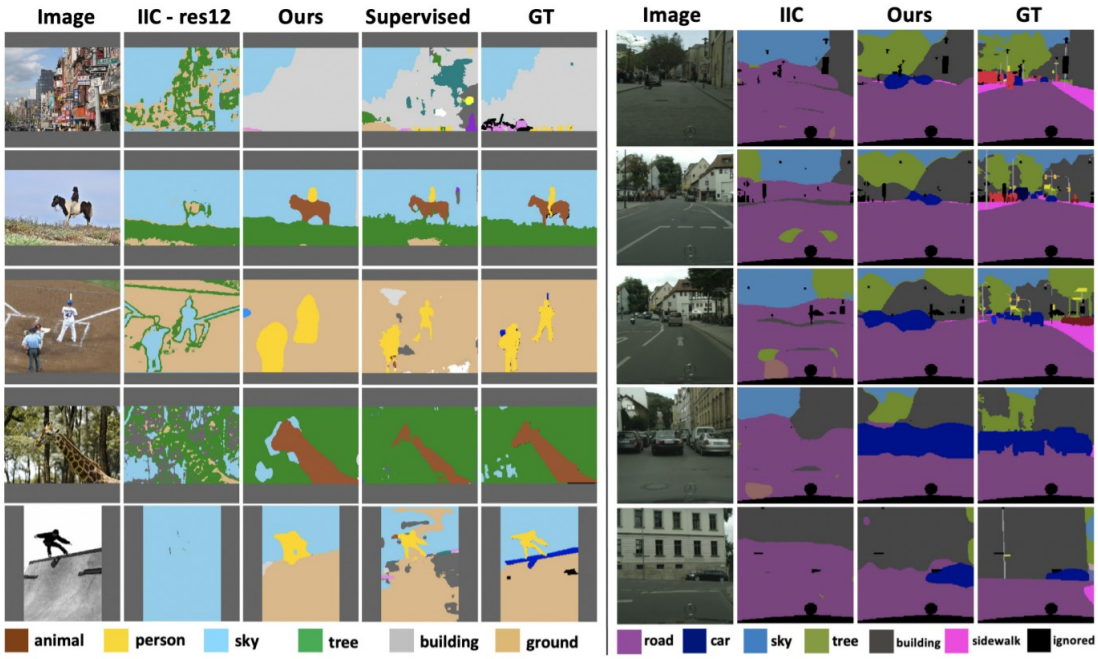


Рис. 6: Результаты на CoCo-All (слева) и Cityscapes (справа). Сравнение с Infomation Invariant Clustering (IIC, IIC-res12) и с сегментацией с учителем (supervised)

На рисунке 6 можно видеть примеры сегментации в сравнении с предыдущим наилучшим методом и с сегментацией с учителем. В таблицах 2, 3, 4 на рисунке 7 указаны метрики Accuracy и mean IoU (среднее IoU по всем классам). Стоит отметить, что PiCIE значительно лучше работает для категории Things. Авторы предполагают, что это следствие эквивариантности к геометрическим преобразованиям.

Интересно также взглянуть на то, какой прирост даёт каждая компонента алгоритма: это отражено в таблицах 5 и 6 на рисунке 8. Например, использование одного набора центроидов только с геометрическими преобразованиями, замена $\mathcal{L}_{\text{cross}}$ на MSE между $z^{(1)}, z^{(2)}$ и отсутствие балансирования функций потерь ведут к снижению качества.

Имея семантически осмысленное латентное представление пикселя, мы получаем возможность искать ближайших к нему соседей на других изображениях. На рисунке 9 можно видеть примеры. Слева показаны правильно найденные соседи (принадлежащие тому же семантическому классу — зебра, жираф, здание). Справа приведены случаи, где семантический класс ближайшего соседа оказался другим. Стоит отметить, что ошибки объяснимы — например, в первом ряду нейросеть спутала снег и облака, так как они довольно похожи.

Method	Partition	# Classes	Acc.	mIoU
Modified DC [4]	*Stuff*	15	44.28	**22.24**
IIC [23]			33.91	12.00
PiCIE + H.			**74.56**	17.32
Modified DC [4]	*Things*	12	67.06	11.55
IIC [23]			43.93	13.64
PiCIE + H.			**69.39**	**23.83**
Modified DC [4]	*All*	27	32.21	9.79
IIC [23]			21.79	6.71
PiCIE + H.			**49.99**	**14.36**

Table 2: Results on different partitions of the COCO dataset.

Method	# Classes	Accuracy	mIoU
IIC	27	47.88	6.35
IIC – res12		29.78	4.96
Modified DC		40.67	7.06
PiCIE		**65.50**	**12.31**

Table 3: Cityscapes results.

Method	COCO-*Stuff*
Random CNN	19.4
K-means [38]	14.1
SIFT [33]	20.2
Doersch 2015 [10]	23.1
Isola 2016 [21]	24.3
DeepCluster [4]	19.9
IIC [23]	27.7
AC [37]	30.8
Modified DC	25.26
IIC	27.97
IIC – res12	27.92
PiCIE	**31.48**

Table 4: COCO-*Stuff* results without ImageNet pretrained weight following [23, 37]. First section is from prior works [23, 37] and the last two sections are from our implementation.

Рис. 7: Результаты PiCIE

Nonpara-metric	Photo-metric	Geo-metric	Over-cluster	Accuracy	mIoU
				34.35	9.88
✓				39.25	9.82
✓	✓			42.55	9.84
✓		✓		46.97	12.04
✓	✓	✓		48.09	13.84
✓	✓	✓	✓	**49.99**	**14.36**

Table 5: Ablation study 1. Our method is decomposed to examine which components affect the performance the most.

Single	MSE eqv.	No inv.	No balance	Accuracy	mIoU
				48.09	**13.84**
			✓	40.56	11.46
✓				44.31	11.71
	✓			44.15	10.98
✓		✓		41.70	9.92

Table 6: Ablation study 2. One or more components in our method is replaced with alternative options.

Рис. 8: Важность компонент

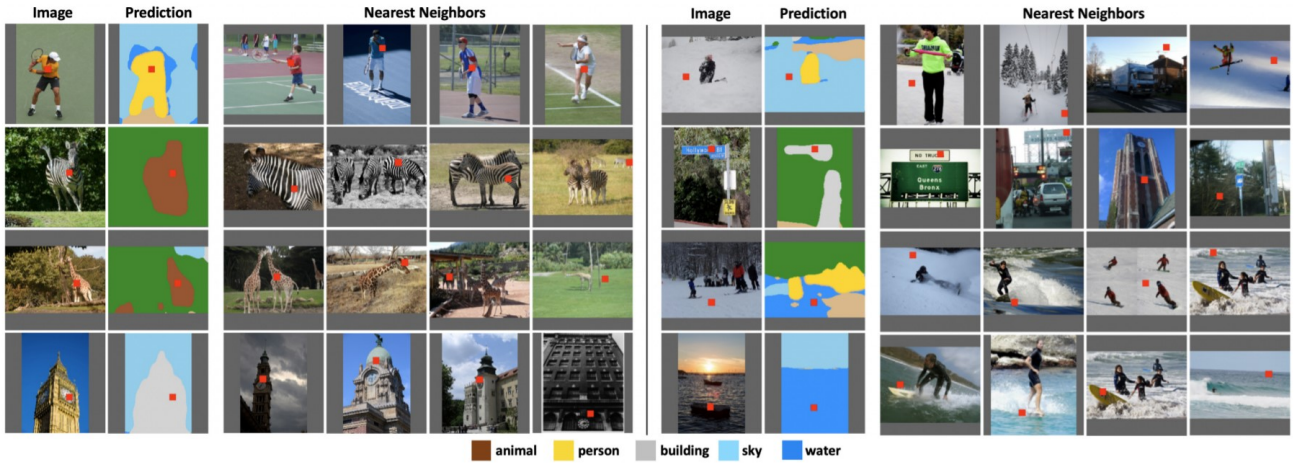


Рис. 9: Ближайшие соседи выделенных красным пикселей на других изображениях.

3.3 DINO

В одно предложение — внимание CLS-токена в DINO хорошо выделяет объекты.

Долгое время в компьютерном зрении доминирующую позицию занимали свёрточные нейронные сети. При этом в обработке естественного языка с 2017 года основной архитектурой является трансформер. Естественно, люди пытались применить их к задачам распознавания изображений, однако долгое время результаты были достаточно скромными — при обучении на Imagenet без сильной регуляризации трансформеры выдавали точность на несколько процентов ниже, чем ResNet с таким же числом параметров. Но в 2021 году была опубликована статья «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale», в которой было показано, что трансформеры, предобученные на очень больших объёмах данных ($\sim$ 300 млн изображений), не уступают CNN. Это дало сильный толчок развитию исследований в этой области.

Очевидно, требование большого количества размеченных данных для обучения — это существенный минус. В обработке естественного языка нашли изящный выход: оказывается, можно придумать задачу, для которой разметку можно получить автоматически и для любого количества данных. Такой задачей является, например, Masked Langauge Modeling: нейросети на вход подаётся текст, причём некоторые слова заменяются на пропуск (маскируются), и требуется предсказать, что же пропущено.

Задачи, для которых разметку можно получить автоматически из самих данных, относят к области Self-Supervised Learning. С приходом трансформеров в компьютерное зрение она стала значительно популярнее, так как позволяет избавиться от нужды в титанических размеченных датасетах. Авторы статьи «Emerging Properties in Self-Supervised Vision Transformers» предлагают новый метод self-supervised обучения для vision transformer-а, который не только показывает хорошие результаты на нескольких задачах, но и ведёт к возникновению некоторых интересных свойств.

Обучение

Метод обучения схематично изображён на рисунке 10. На вход студенту и учителю подаются две случайные трансформации входного изображения. Студент и учитель имеют одинаковые архитектуры, но разные наборы параметров. На выходе получаем некоторое векторное представление для исходных изображений. К ним применяется операция softmax, а затем между ними считается кросс-энтропия. Важная особенность: вектор от учителя перед взятием softmax-а центрируется, а потом все элементы делятся на малень-

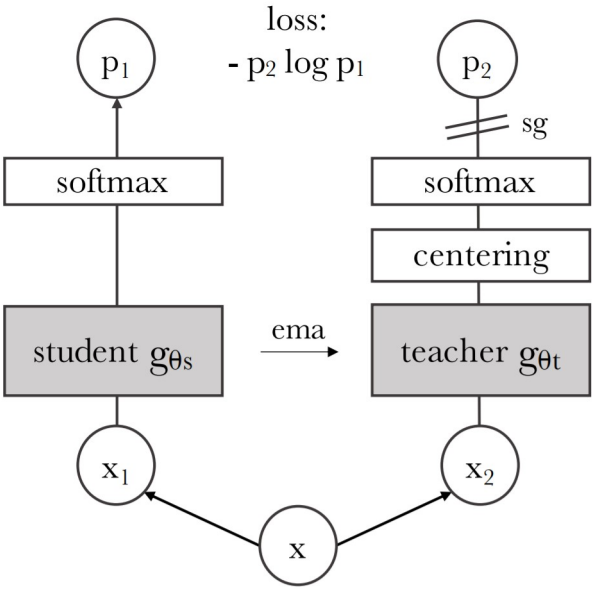


Рис. 10: Схема обучения DINO

кое число. Если оставить только центрирование, то учитель будет тяготеть к равномерному распределению в ответ на любое изображение; если оставить только масштабирование, то учитель станет предсказывать распределение, вырожденное в одной точке, так что для балансирования надо использовать и то, и другое.

Веса учителя обновляются не градиентным спуском (sg на рисунке означает stop-gradient), а считаются как экспоненциальное среднее весов студента. Т.е.

$$\theta_t^{(k+1)} = \lambda\theta_t^{(k)} + (1-\lambda)\theta_s^{(k+1)} \tag{21}$$

Усреднение весов здесь имеет смысл ансамблирования моделей. На протяжении всего обучения учитель производит более качественные эмбеддинги изображений, тем самым направляя обучение студента.

Центрирующее слагаемое c также обновляется с помощью экспоненциального сглаживания:

$$c^{(k+1)} = mc^{(k)} + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i) \tag{22}$$

Результаты

На рисунке 11 можно видеть численные результаты. Оценивание проведено для сегментации видео. Авторы говорят, что использовали метод ближайших соседей между рядом стоящими кадрами. *Не очень понятно, как именно произведена сегментация,* однако результаты превосходят многие сильные бейзлайны.

Method	Data	Arch.	$(\mathcal{J}\&\mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
Supervised					
ImageNet	INet	DeiT-S/8	66.0	63.9	68.1
STM [46]	I/D/Y	RN50	81.8	79.2	84.3
Self-supervised					
CT [68]	VLOG	RN50	48.7	46.4	50.0
MAST [38]	YT-VOS	RN18	65.5	63.3	67.6
STC [35]	Kinetics	RN18	67.6	64.8	70.2
DINO	INet	DeiT-S/16	61.8	60.2	63.4
DINO	INet	ViT-B/16	62.3	60.7	63.9
DINO	INet	DeiT-S/8	**69.9**	**66.6**	**73.1**
DINO	INet	ViT-B/8	**71.4**	**67.9**	**74.9**

Рис. 11: DAVIS 2017 Сегментация объектов на видео. $\mathcal{J}_m$ – это mIoU, $\mathcal{F}_m$ – метрика, оценивающая точность предсказания границ сегментации.

На рисунке 12 показаны примеры сегментации с использование карт внимания от разных голов токена CLS с последнего слоя трансформера. Маску образуют все патчи, которым присвоен вес больше некоторого порога, причём порог выбирается так, чтобы сохранить 60% от суммы всех весов.

Исследуя важность различных компонент обучения, авторы установили, что важно использовать аугментацию multicrop. Она заключается в том, что из изображения вырезаются небольшие кусочки, которые подаются студенту, а учитель видит либо всё изображение целиком, либо бóльший кусок. Студент должен предсказать такой же эмбеддинг, как и учитель, поэтому ему приходится восстанавливать семантику изображения по небольшому участку. Вероятно, поэтому он концентрирует своё внимание на объектах или его частях, потому что именно в них содержится основная информация на изображении. Можно предположить, что задача, решаемая DINO при предобучении, несёт более полезный сигнал для сегментации. В самом деле, на рисунке 13 авторы сравнивают нейросеть, обученную с помощью DINO, с нейросетью, предобученной с помощью supervised learning. У последней не возникает эффекта сегментации.

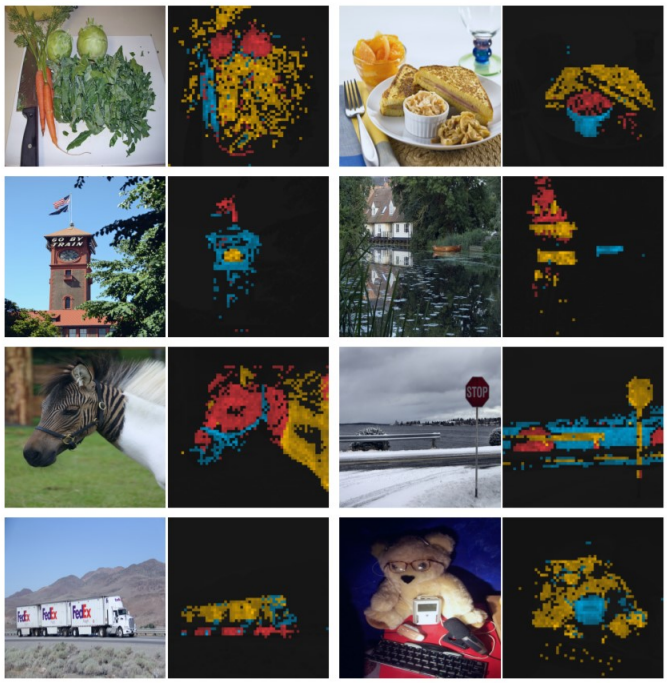


Рис. 12: Карты внимания разных голов для токена CLS. Разные головы, показанные разным цветом, фокусируются на разных объектах или их частях.

	Random	Supervised	DINO
DeiT-S/16	22.0	27.3	45.9
DeiT-S/8	21.8	23.7	44.7

Рис. 13: Сегментация с помощью карт внимания для DINO и supervised-модели. Показана лучшая голова для обоих моделей. Таблица сравнивает коэффициент Жаккара между правильной сегментаций и валидационными изображениями в PASCAL VOC12.

3.4 Stego

В одно предложение — дистилляция признаковых описаний пикселей от DINO с сохранением корреляции признаковых описаний.

Как стало известно в последние годы, промежуточные представления в нейронных сетях могут быть хорошим признаковым описанием или иными способами нести полезный сигнал. Например, в статье RAFT: Recurrent All-Pairs Field Transforms for Optical Flow удаётся использовать четырёхмерные тензоры корреляций между картами активаций. Они определяются следующим образом: пусть $A \in \mathbb{R}^{CHW}$, $B \in \mathbb{R}^{CIJ}$ — признаковые тензоры двух изображений, где C отвечает за размерность представления одного пикселя, а $(H, W), (I, J)$ отвечают за размеры изображений. Тогда тензор корреляций (или тензор соответствий признаков) F есть

$$F \in \mathbb{R}^{HWIJ}, \quad F_{hwij} = \sum_c \frac{A_{chw}}{|A_{hw}|} \frac{B_{cij}}{|B_{ij}|} \tag{23}$$

Его элементы — это косинусное расстояние между признаковым описанием пикселя на позиции (h, w) с первого изображения и на позиции (i, j) со второго. Если оно мало, то указанные пиксели схожи в признаковом пространстве. На рисунке 14 на Figure 2 можно видеть, как три избранных точки на изображении соотносятся с остальными точками этого же изображения, а также на ближайших соседях выбранного изображения. Признаковое описание здесь получено с помощью DINO (self-**di**stillation with **no** labels).

Оказывается, соответствия пикселей сильно скоррелированы с соответствиями меток при семантической сегментации. Если использовать косинусное сходство между пикселями как логит для предсказания того, совпадают ли метки этих пикселей, то беря признаковое описание от DINO без каких-либо изменений, можно получить recall 50% при precision 90%. Иными словами, мы найдём найдём половину всех пар пикселей с совпадающими метками, при этом лишь 10% предсказаний будут ложно-положительными. Учитывая то, что DINO обучается без разметки и на задачу, слабо связанную с сегментацией, это довольно впечатляющий результат. На Figure 3 можно увидеть precision-recall кривую для задачи нахождения пар пикселей с совпадающими метками для нескольких методов.

Убедившись, что соответствия признаковых описаний пикселей хорошо коррелируют с совпадением семантических меток, авторы решили обучить нейронную сеть, которая бы преобразовывала признаковое описание, полученное от DINO или другого encoder-a, в новое промежуточное представление, причём так, чтобы сохранить соответствие между пикселями.

Figure 2: Feature correspondences from DINO. Correspondences between the source image (left) and the target images (middle and right) are plotted over the target images in the respective color of the source point (crosses in the left image). Feature correspondences can highlight key aspects of shared semantics within a single image (middle) and across similar images such as KNNs (right)

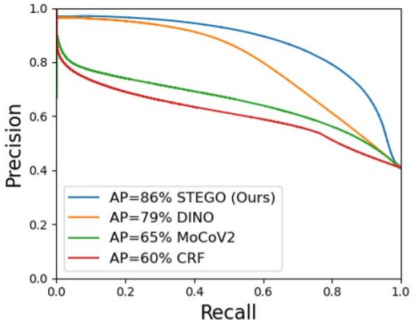


Figure 3: Precision recall curves show that feature self-correspondences strongly predict true label co-occurrence. DINO outperforms MoCoV2 and a CRF kernel, which shows its power as an unsupervised learning signal.

Рис. 14: Соответствия между пикселями при использовании DINO.

Более формально, пусть отображение $\mathcal{N}\colon \mathbb{R}^{C'H'W'} \mapsto \mathbb{R}^{CHW}$ — это исходный encoder, который отображает изображение с C' каналами размером (H', W') в признаковый тензор с C каналами размера (H, W). Мы хотим обучить легковесную «голову» $\mathcal{S}\colon \mathbb{R}^{CHW} \mapsto \mathbb{R}^{KHW}$, которая отображает пиксели из исходного пространства в новое, причём $K < C$. Цель $\mathcal{S}$ — описать такое нелинейное преобразование, чтобы в пространстве $\mathbb{R}^{KHW}$ образовывались плотные кластера и усиливались паттерны, присутствующие в тензоре корреляций исходных признаков.

Пусть x, y — изображения. Рассмотрим тензоры исходных признаков

$$f = \mathcal{N}(x) \in \mathbb{R}^{CHW},\ g = \mathcal{N}(y) \in \mathbb{R}^{CIJ} \tag{24}$$

и признаков после преобразования $\mathcal{S}$

$$s = \mathcal{S}(f) \in \mathbb{R}^{KHW},\ t = \mathcal{S}(g) \in \mathbb{R}^{KIJ} \tag{25}$$

Обозначим F, S — тензоры корреляций $(f, g), (s, t)$ соответственно. Если значение F_{hwij} велико, это значит, что пиксель на позиции (h, w) в изображении x похож на пиксель на позиции (i, j) в изображении y. Тогда мы хотим, чтобы S_{hwij} также было велико. Этого можно добиться, используя следующую функцию потерь:

$$\mathcal{L}_{\text{simple-corr}}(x, y, b) = -\sum_{h,w,i,j} (F_{hwij} - b) S_{hwij} \tag{26}$$

Параметр b интерпретируется как постоянное «негативное давление», которое не даёт решению сколлапсировать. По смыслу, такая фукция потерь за-

ставляет S_{hwij} расти, если $(F_{hwij} - b) > 0$ (т.е. пиксели достаточно схожи), и убывать иначе.

Эта версия функции потерь оказалось нестабильной. Авторы предполагают, что виной тому коллинеарность получающихся представлений. Кроме того, для маленьких объектов часто $(F_{hwij} - b) < 0$, и вместо того чтобы выравнивать признаковые описания пикселей этого объекта, сеть делает противоположное. Потому предлагается использовать следующую версию:

$$\hat{F}_{hwij} = F_{hwij} - \frac{1}{IJ} \sum_{k,m} F_{hwkm} \tag{27}$$

$$\mathcal{L}_{\text{corr}}(x, y, b) = - \sum_{h,w,i,j} (\hat{F}_{hwij} - b) \max(S_{hwij}, 0) \tag{28}$$

Центрирование корреляции помогает для небольших объектов, а обрезка S_{hwij} в нуле заставляет делать представления отличающихся пикселей ортогональными.

Данная функция потерь применяется трижды — для сравнения изображения с самим собой, с его ближайшими соседями и с случайными изображениями. Сравнение с собой и с ближайшими соседями даёт в основном положительный сигнал, а последнее — отрицательный. Итого получаем

$$\mathcal{L} = \lambda_{\text{self}} \mathcal{L}_{\text{corr}}(x, x, b_{\text{self}}) + \lambda_{\text{knn}} \mathcal{L}_{\text{corr}}(x, x_{\text{knn}}, b_{\text{knn}}) + \lambda_{\text{rand}} \mathcal{L}_{\text{corr}}(x, x_{\text{rand}}, b_{\text{rand}}) \tag{29}$$

Параметры b были подобраны так, чтобы средняя схожесть признаков с ближайшими соседями была ≈ 0.3, а со случайными изображениями ≈ 0.0. Соотношение между коэффициентами: $\lambda_{\text{self}} \approx \lambda_{\text{rand}} \approx 2\lambda_{\text{knn}}$ (подобрано эмпирически).

Чтобы хорошо сегментировать маленькие объекты, авторы вырезают из каждого изображения по 5 локальных участков и при дальнейшем обучении ищут ближайших соседей именно среди этих участков. Наконец, чтобы получить метки, надо кластеризовать признаковые описания пикселей, полученные с помощью $\mathcal{S}$. Авторы также используют Conditional Random Field, чтобы подправить сегментацию в соответствии с границами на изображении. Итоговая схема обучения представлена на рисунке 15.

Результаты можно видеть на рисунках 16, 17.

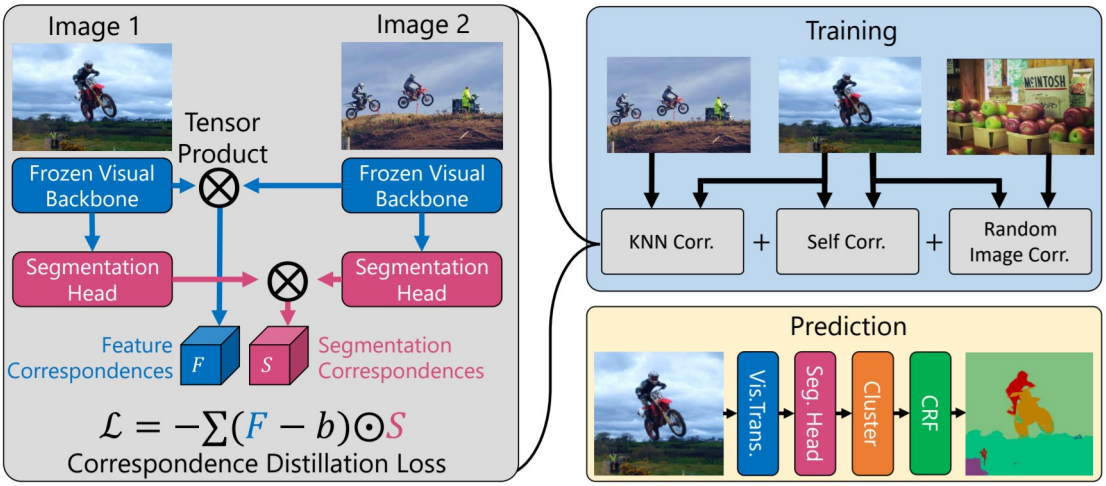


Рис. 15: Схема обучения Stego. Серые прямоугольники обозначают функцию потерь.

Table 1: Comparison of unsupervised segmentation architectures on 27 class CocoStuff validation set. STEGO significantly outperforms prior art in both unsupervised clustering and linear-probe style metrics.

	Unsupervised		Linear Probe	
Model	**Accuracy**	**mIoU**	**Accuracy**	**mIoU**
ResNet50 (He et al., 2016)	24.6	8.9	41.3	10.2
MoCoV2 (Chen et al., 2020c)	25.2	10.4	44.4	13.2
DINO (Caron et al., 2021)	30.5	9.6	66.8	29.4
Deep Cluster (Caron et al., 2018)	19.9	-	-	-
SIFT (Lowe, 1999)	20.2	-	-	-
Doersch et al. (2015)	23.1	-	-	-
Isola et al. (2015)	24.3	-	-	-
AC (Ouali et al., 2020)	30.8	-	-	-
InMARS (Mirsadeghi et al., 2021)	31.0	-	-	-
IIC (Ji et al., 2019)	21.8	6.7	44.5	8.4
MDC (Cho et al., 2021)	32.2	9.8	48.6	13.3
PiCIE (Cho et al., 2021)	48.1	13.8	54.2	13.9
PiCIE + H (Cho et al., 2021)	50.0	14.4	54.8	14.8
STEGO (Ours)	**56.9**	**28.2**	**76.1**	**41.0**

Рис. 16: Результаты на CoCo Stuff

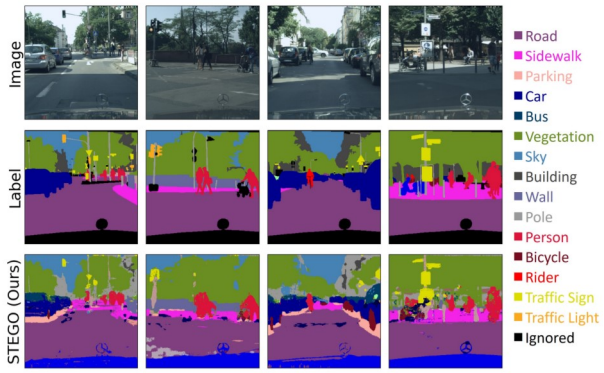


Figure 5: Comparison of ground truth labels (middle row) and cluster probe predictions for STEGO (bottom row) for images from the Cityscapes dataset.

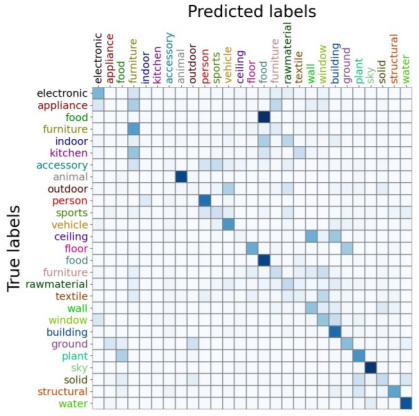


Figure 6: Confusion matrix of STEGO cluster probe predictions on CocoStuff. Classes after the "vehicle" class are "stuff" and classes before are "things". Rows are normalized to sum to 1.

Рис. 17: Пример сегментации. Матрица ошибок

3.5 Ours

В одно предложение: (пока) замена Selective Search в DETReg на генерацию синтетического датасета с помощью CutMix.

Пока удалось проверить только качество детекции.

Таблица 1: Дообучение на детекцию на PASCAL VOC

Метод	detection mAP50	detection mAP75	detection mAP
DETReg	**63.5**	83.3	70.3
Ours	**63.5**	**83.8**	**70.9**

Литература

[1] C. M. Bishop, Pattern Recognition and Machine Learning, 2006

[2] Non-Parametric Probabilistic Image Segmentation, M. Andreetto, L. Zelnik-Manor 2007

1 Contrastive learning

Как обучать encoder, не имея размеченных данных? Можно придумать такие задачи, для который разметка может быть получена автоматически. Например, восстановление повреждённого тем или иным способом изображения, восстановление вырезанного участка изображения, раскраска изображения, переведённого в чёрно-белый, предсказание относительной позиции двух участков изображения, предсказание угла, на который повёрнуто изображение, сравнительное обучение (contrastive learning). Подобные задачи выделяют в специальный раздел — self-supervised learning (самообучение).

Лучше всего (в смысле качества работы encoder-a после дообучения на конкретную задачу обучения с учителем) себя показывает contrastive learning, поэтому поговорим о нём поподробнее. Общая идея заключается в том, что мы хотим для похожих объектов выучить представления, которые будут близки в признаковом пространстве, а представления непохожих объектов должны быть далеко друг от друга.

Одним из первых примеров функции потерь, которая формализует это требование, является Contrastive loss:

$$\mathcal{L}_{\text{contr}}(x, x', \theta) = \begin{cases} \|f_\theta(x) - f_\theta(x')\|^2, & x' \text{ — позитивный пример} \\ \max\big(0, M - \|f_\theta(x) - f_\theta(x')\|^2\big), & x' \text{ — негативный пример} \end{cases}$$

Позитивным примером называется объект, который мы считаем похожим на x (например, в случае изображений можно взять горизонтально отражённый x, так как в большинстве случаев это не меняет семантику). Отрицательный (негативный) пример, соответственно, есть объект, непохожий на x. В качестве отрицательных примеров обычно берут случайное изображение из датасета.

Сейчас часто используется Noise Contrastive Estimation.

$$\mathcal{L}\big(\hat{z}_t, z_t, \{z_l\}\big) = \log\left(\frac{\exp(\hat{z}_t^T z_t)}{\exp(\hat{z}_t^T z_t) + \sum_l \exp(\hat{z}_t^T z_l)}\right)$$

Здесь $\hat{z}_t$ — описание исходного объекта, z_t — позитивный пример, $\{z_l\}$ — множество отрицательных примеров. На это можно смотреть как на классификацию из $N+1$ классов, где N — число негативных примеров.

Как оказалось, чтобы этот подход заработал, нужно большое количество негативных примеров — в статье SimCLR, в которой был предложен этот метод, использовались батчи из 2048 изображений (аугментированная версия i-го изображения рассматривалась как позитивный пример для него же, а все остальные — как негативные).