Содержание

Аннотация

Современные методы метрического глубинного обучения позволяют эффективно решать такие задачи, как поиск по изображению, идентификация личности по фотографии, сравнение изображений и определение степени их похожести. Для обучения нейросетевых моделей метрического обучения используется ряд особых функций потерь, не встречающихся в других областях машинного обучения. На протяжении последних лет было разработано множество различных архитектур функций потерь для достижения более высокого качества работы моделей. Однако некоторые статьи указывают на факт того, что последние достижения метрического обучения обусловлены не столько новыми функциями потерь, сколько более удачным подбором других гиперпараметров обучения. Данный курсовой проект посвящен исследованию давно придуманной классической триплет функции потерь, изучению существующих и созданию новых ее модификаций. В результате проделанной работы на задаче поиска похожих изображений удалось добиться качества, сравнимого с лучшими на данный момент результатами метрического обучения.

Ключевые слова

Глубинное обучение, метрическое обучение, контрастивное обучение, триплет лосс, нейронные сети, визуальные трансформеры, эмбеддинги, извлечение признаков, признаковое пространство

1 Введение

Метрическое обучение - не так давно появившееся направление машинного обучения. Его цель - сопоставить каждому объекту некоторое векторное представление таким образом, чтобы похожие объекты соответствовали близким векторам, а разные объекты - далеким друг от друга векторам.

Такая постановка задачи и существующие методы ее решения имеют множество применений и ряд преимуществ по сравнению, например, с обычной задачей классификации.

Далее я более подробно и формально расскажу о задаче метрического обучения.

1.1 Постановка задачи метрического обучения

В метрическом обучении обычно мы хотим обучить нейросеть $f(\theta, x)$, которая на вход принимает объект x (например, картинку), а на выходе возвращает числовой вектор. Этот вектор называют *векторным представлением* объекта x или его *эмбеддингом*. Пространство, которому принадлежат эмбеддинги называют *пространством эмбеддингов* или *признаковым пространством*. В признаковом пространстве в свою очередь вводится некоторая метрика расстояния $d(u, v)$ между любыми двумя векторами u и v. Чаще всего это либо евклидовое, либо косинусное расстояние. В классическом метрическом обучении также часто используют расстояние Махаланобиса $d(u, v) = (u - v)^t S(u - v)$, где S - обучаемая положительно определенная матрица. Таким образом, метрика d также может быть обучаемой.

При обучении параметров θ нейросети $f(\theta, x)$ мы хотим добиться, чтобы для похожих объектов x и y (например, картинок одного и того же автомобиля) величина $d(f(\theta, x), f(\theta, y))$ была маленькой, а для разных объектов x и y (например, фотографий совершенно разных предметов) та же величина имела большое значение.

1.2 Преимущества метрического обучения

Среди преимуществ такого подхода над обычными классификаторами можно выделить следующие:

1) Зачастую приходится решать задачу классификации с постоянно растущим числом классов. В таких случаях мы не хотим переучивать (или дообучать) модель каждый раз, когда число классов изменилось

2) Иногда мы заранее знаем, что при использовании в реальной жизни модель столкнется с объектами, не принадлежащим ни к одному классу из тех, которые модель видела

во время обучения. Но при этом мы все еще хотим, чтобы модель извлекала из объекта адекватные признаки

3) Считая расстояния между эмбеддингами, можно получить количественную характеристику «похожести» объектов из одного класса. Это может быть полезно, когда нужно не только определить класс объекта, но и проранжировать объекты из имеющейся у нас *галереи* по степени похожести на него (например, задача поиска по изображению)

4) Иногда может быть полезным получить более качественные признаки объекта для того, чтобы в дальнейшем передать их в другую модель и получить более высокое качество, чем было до этого

5) Очевидно, такой подход можно использовать также для решения задач снижения размерности и кластеризации объектов

6) Далее в отчете будет показано, что обучение модели метрического обучения не всегда относится к обучению с учителем, оно также может быть отнесено к *слабому* обучению с учителем (то есть когда у нас есть только информация об отдельных парах объектов, являются ли они объектами одного класса или нет)

1.3 Способы обучения модели

Обычно выделяют два основных подхода:

1) решение обычной задачи классификации с модифицированием функции потерь таким образом, чтобы выход предпоследнего слоя учился под задачу метрического обучения (пример такого решения изложен в статье [6]). Однако данный подход в последнее время уступает описанному в пункте 2

2) использование контрастивной функции потерь. Контрастивными называются функции потерь, которые явно направлены на сближение эмбеддингов похожих объектов и раздвигание эмбеддингов разных объектов. Одной из таких как раз и является триплет функция потерь

1.4 Описание триплет функции потерь

Триплет функция потерь всегда рассчитывается на основании 3 объектов: якоря, позитива и негатива. По определению этих объктов они выбираются так, чтобы якорь и позитив относились к одному классу, а якорь и негатив - к разным. Пусть мы применили нейросеть $f(\theta, x)$ к якорю, позитиву и негативу - получили эмбеддинги a, p, n соответственно. Тогда

формула триплет лосса:

$$\mathcal{L}_{tri}(a, p, n) = \max(0, m + d(a, p) - d(a, n))$$

где m - гиперпараметр.

1.5 Процесс обучения и валидации

1.5.1 Обучение

Процесс обучения хорошо иллюстрирует рисунок 1.1.

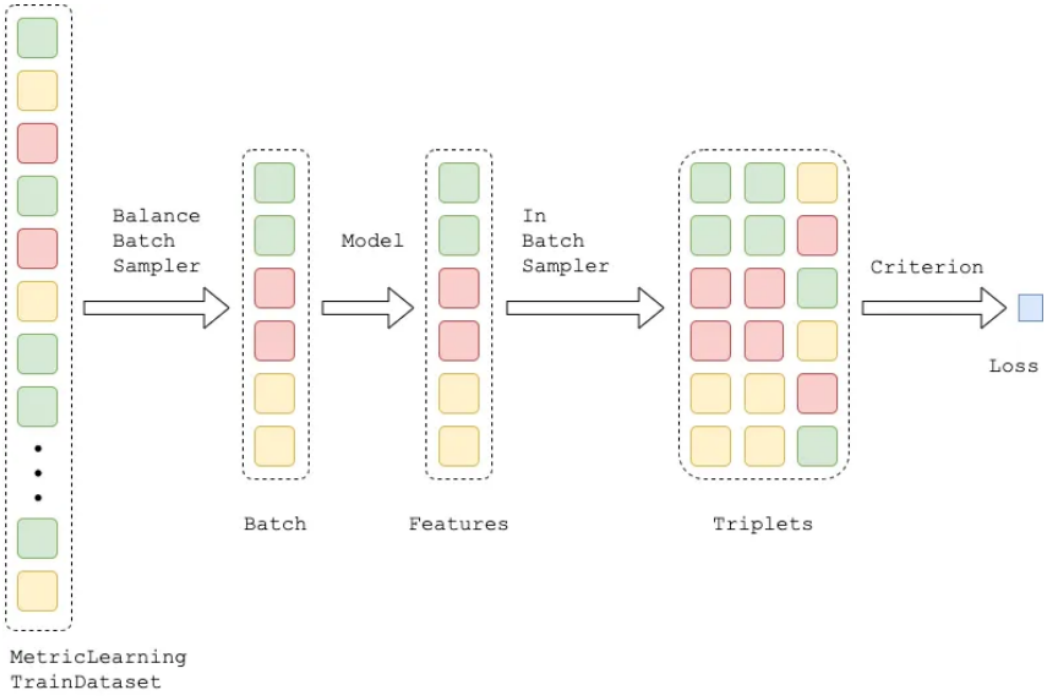

Рис. 1.1: Схема процесса обучения модели в метрическом обучении (иллюстрация из [21])

Одинаковыми цветами обозначены объекты одного класса. Во время обучения мы выбираем батч объектов, применяем к каждому из них нейросеть $f(\theta, x)$, затем формируем из полученных эмбеддингов тройки (триплеты), которые затем передаем в функцию потерь. Обучение обычно ведется градиентным спуском по параметрам θ неросети $f(\theta, x)$.

1.5.2 Валидация

Для начала введем терминологию, характерную для метрического обучения. В режиме инференса модели подается *объект-запрос* (*query*), для которого мы хотим найти топ похожих объектов из имеющейся у нас *галереи* (*gallery*).

Соответственно, на этапе валидации мы делим валидационную выборку на запросы и галерею (либо такое разделение может уже присутствовать в датасете). Для датасетов, у которых такое разделение отсутствует, часто используют стратегию, при которой каждый объект валидационной выборки по очереди рассматривается как запрос, а все остальные - как галерея.

Далее руководствуемся рисунком 1.2

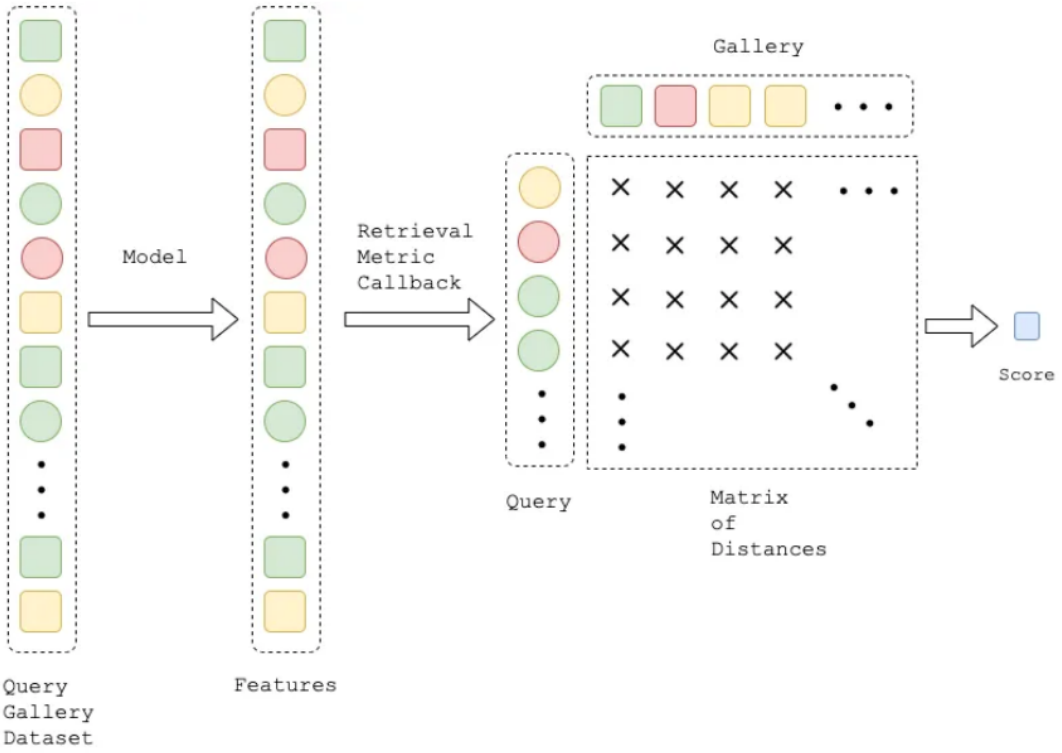

Рис. 1.2: Схема процесса валидации модели в метрическом обучении (иллюстрация из [21])

Все объекты валидационной выборки пропускаем через обученную нейросеть $f(\theta, x)$, после чего считаем все попарные расстояния между объектами-запросами и объектами из галереи (строим матрицу расстояний). Теперь для каждого объекта-запроса можно выбрать топ k ближайших объектов из галереи, после чего без проблем вычислить необходимые метрики (например, MAP@k или CMC@k).

1.6 Обзор метрик качества

Расскажем более подробно про метрики качества моделей метрического обучения. Основные из них: точность, AP и MAP, полнота и CMC. Про них хорошо рассказо в статье [25]. Будем оценивать качество *выдачи* модели. Выдачей модели будем называть k самых ближних объектов галереи к текущему запросу, упорядоченные по увеличению расстояния до этого запроса. Под расстоянием подразумевается метрика d между эмбеддингами, полученными

из нейросети $f(\theta, x)$ с текущими параметрами θ.

1.6.1 Точность

Формула для точности:

$$precision@k = \frac{true\ positives}{\min(k, all\ positives\ in\ gallery)}$$

Точноть показывает, какая доля из лучших k найденных в галерее картинок действительно являются релевантными (доля релевантных объектов в выдаче). Обратите внимание на взятие минимума в знаменателе: это нужно, на случай, если количество всех релевантных объектов в галерее меньше, чем k.

1.6.2 AP и MAP

AP (average precision) - метрика, учитывающая не только количество, но и положение релевантных объектов в выдаче:

$$AP@k = \frac{\sum_{i=1}^{k} r_i \cdot precision@i}{\sum_{i=1}^{k} r_i}$$

где $r_i = 1$, если объект на i-ой позиции выдачи релевантен объекту-запросу, и $r_i = 0$ иначе. По сути, метрика считает среднее значение $precision@i$ для всех позиций выдачи, на которых стоят релевантные объекты.

Если усреднить $AP@k$ по всем объектам-запросам, то получится $MAP@k$.

1.6.3 Полнота и CMC

Если определять полноту (обозначается *recall* или просто R) согласно общепринятому определению в машинном обучении, то это доля показанных в выдаче релевантных объектов среди всех релевантных объектов в галерее. Однако в статьях по метрическому обучению как правило придерживаются другого определения: $R@k = 1$, если среди первых k объектов есть хотя бы один релевантный, и $R@k = 0$ иначе. Далее мы будем использовать именно это определение. Как правило модели в метрическом обучении сравниваются именно по этой метрике.

Стоит заметить, что точно также определяется метрика CMC. Поэтому можно сказать, что для любого натурального k всегда верно $R@k = CMC@k$.

1.7 Цели данной работы

Целью данной работы является изучение триплет функции потерь, демонстрация ее результатов в сравнении с лучшими на данный момент результатми метрического обучения, исследование уже существующих ее модификаций и изобретение своих собственных модификаций. Более подробно:

1) Достижение хороших результатов с использованием классической триплет функции потерь на основных датасетах глубинного метрического обучения

2) Исследование влияния гиперпараметров моделей метрического обучения, использующих триплетную функцию потерь, на итоговое качество

3) Поиск и изучение статей и материалов с попытками улучшить архитектуру триплет функции потерь, определение текущих бенчмарков метрического обучения с ее использованием

4) Исследование своих собственных способов модификации триплет функции потерь

5) Постановка экспериментов на наиболее популярных датасетах метрического обучения, сравнение результатов наших моделей с текущими бенчмарками

6) Формулировка выводов и написание финального отчета

1.8 Используемые инструменты и датасеты

Для обучения моделей будет использоваться библиотека Open Metric Learning [16] для языка Python. Данная библиотека является оберткой над PyTorch, позволяющей проще реализовывать процессы обучения и валидации метрического обучения. Статья с примером использования OML [22].

Для того, чтобы ставить эксперименты с разными видами функций потерь, используя OML, мы вручную модифицировали код класса, соответствующего триплет функции потерь.

Изначально планировалось проводить эксперименты на двух относительно небольших датасетах: CARS196 (16185 фотографий 196 классов автомобилей) и CUB-200-2011 (11788 картинок 200 различных видов птиц), так как эксперименты на них не потребуют слишком больших вычислительных мощностей. А при получении значимых результатов лучшие архитектуры новой функции потерь планировалось испытать на более крупных базах данных. Однако с некоторого момента официальная страница CAR196 стала недоступна, поэтому все эксперименты проводились на CUB-200-2011.

2 Обзор литературы и существующих решений

Текущие бенчмарки для разных датасетов можно посмотреть на Papers With Code [18].

Одной из первых статей про триплет функцию потерь является [19]. Там же упоминается про необходимость особого формирования триплетов. Это означает, что нужно стараться выбирать негатив как можно близко к якорю, а позитив - как можно дальше (то есть выбирать так называемые *hard positive* и *hard negative*). Если так не делать, то нейросети зачастую будет очень просто отличить один класс от другого ($d(a,p) << d(a,n)$), из-за чего модель почти не будет учиться. Подбор тяжелых позитивов и негативов называют *майнингом* (*mining*). Майнинг чаще всего проводят в пределах одного мини-батча во время обучения.

Среди модификаций распространен center-based подход (когда измеряем расстояние от якоря до центров позитивов и негативов). Подобное, например, описано в статьях [1] [5]. Также есть попытки совмещения sample-based и center-based подходов [13].

Зачастую к триплет функции потерь добавляют дополнительные слагаемые. Например, в статье [10] в целях улучшения кластеризации к триплету добавляют слагаемые, направленные на независимое уменьшение $d(a,p)$ и увеличение $d(a,n)$. Кроме того, в статье [14] добавляют стандартную кросс-энтропийную функцию потерь.

Есть интересная статья, в которой вместо косинусного расстояния в признаковом пространстве используют просто угол между эмбеддингами [12]. За счет этого на последних стадиях обучения (при малом $d(a,p)$) градиент протекает лучше и не затухает, что улучшает обучение.

Также есть идея, похожая на одну из тех, что мы хотим реализовать: [3]. Здесь вместо классического триплета предлагают функцию потерь, которая иногда заменяет $d(a,n)$ на среднее по батчу расстояние между позитивами и негативами.

Иногда встречаются статьи, в которых пытаются сделать адаптивный гиперпараметр m, например, здесь [9].

Есть более теоретические подходы, например вещи по типу байесовской триплет функции потерь [23] (вместо конкретного эмбеддинга объекта рассматриваем распределение вероятности на пространстве эмбеддингов, а максимизируем мы вероятность того, что $d(a,p) < d(a,n) - m$) или триплет функции потерь с использованием дискриминанта Фишера [8].

В этой работе мы будем сравниваться с результатми из статьи про HypViT [7] (ТОП-2 результат на основных датасетах метрического обучения). Ее авторы вкладывали пространство эмбеддингов в гиперболическое пространство, аргументируя это тем, что данные имеют

древовидную иерархическую структуру, как и гиперболическое пространство.

Совсем недавно (уже во время написания данного отчета) появилась статья [2], в которой был достигнут существенный прорыв в качестве моделей метрического обучения. Ее авторы выбрали в качестве претренированной модели не ViT-S, претренированный на ImageNet1k, а ViT-L, претренированный особым образом на результате кластеризации огромного неразмеченного датасета LAION 400M на миллион псевдо классов.

Данная работа в значительной степени вдохновлена статьями [15] и [20]. В первой из них авторы утверждают, что последние достижения метрического обучения обусловлены не столько новыми функциями потерь, сколько более удачным подбором других гиперпараметров обучения. А самым главным фактором, влияющим на качество, является выбор претренированной модели. Во второй статье явно показано, что обычный ViT-S DINO, обученный триплет функцией потерь, (в статье эта архитектура называется ViT-Triplet) показывает результаты намного выше, чем уже упомянутый HypViT.

3 Подбор начальной конфигурации нейросети

В процессе обучения все результаты логировались в wandb. Поэтому все эксперименты, описанные ниже, и их код можно увидеть, пройдя по ссылке. Ноутбук, с помощью которого проводились эксперименты, можно найти там же, либо на гитхабе.

Перед тем, как проводить эксперименты с модификациями классического триплета, нужно подобрать гиперпараметры так, чтобы получить из него максимальное качество.

3.1 Гиперпараметры из готового решения библиотеки

В используемой мной библиотеке уже имелся пайплайн для обучения модели на датасете CUB-200-2011. Однако я выяснил, что в нем деление датасета на обучающую и валидационную выборку отличается от общепринятого в метрическом обучении. Как правило в задачах метрического обучения на валидации модель должна сталкиваться с теми классами, которые она не видела при обучении. Для этого в соответствии со стандартным протоколом обучения и валидации [11] первые 100 классов птиц датасета CUB были отнесены к обучающей выборке, а остальные 100 классов - к валидационной.

После корректировки этого момента было запущено первое обучение модели, все гиперпараметры которой были взяты из пайплайна библиотеки Open Metric Learning для датасета CUB. Перечисляем ниже все эти гиперпараметры:

1) В качестве аугментаций использовалась сложная последовательность из изменения размера, случайного горизонтального разворота, блюра, изменерия цветовой палитры, изменения ракурса изображения, прибавления случайного шума, канального дропаута и нормализации. Полный код можно посмотреть в самой библиотеке [17]

2) При выборе мини-батча случайно выбирались 8 классов, по 4 картинки из каждого класса. Важно уточнить, что в глубинном метрическом обучении каждый класс попадает в мини-батч ровно один раз за эпоху. То есть обычно 1 эпоха соответствует проходу по всем классам обучающей выборки, а не проходу по всем ее объектам

3) Использовался ViT-S DINO, претренированный без учителя на ImageNet1k. Стоит заметить, что в задачах глубинного метрического обучения почти всегда производится файнтьюнинг уже предобученной модели, а не полное обучение моделей со случайной начальной инициализацией параметров

4) эмбеддинги на выходе ViT не нормализуются и сразу передаются в майнер

5) используется довольно нестандартная версия майнера - с банком памяти. Данный майнер запоминает эмбеддинги, полученные на последних шагах, и классы, к которым они относятся, а в дальнейшем используех их для подбора сложных позитивов и негативов.

6) используется сглаженная версия триплет функции потерь:

$$\ell_{soft} = \log(1 + \exp(d(a, p) - d(a, n)))$$

7) используется оптимизатор Adam с постоянной длиной шага 10^{-5}

Так как 1 эпоха обучения длится довольно мало, то валидация происходит раз в 10 эпох. Наилучшее качество на валидации: $R@1 = 0,6911$. График $R@1$ ($CMC@1$) на валидации представлен на рисунке 3.1

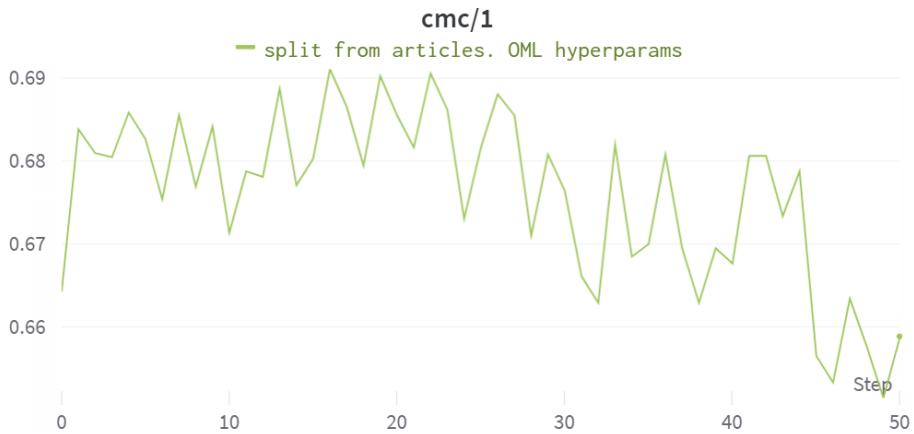

Рис. 3.1: График метрики $R@1$ на валидации. Единица на оси абцисс соответствует 10 эпохам обучения

3.2 Первые поправки гиперпараметров

В статье про HypViT [7] на датасете CUB себя лучше всего показал ViT-S претренированный на ImageNet1k, поэтому мы использовали этот претрейн. Максимальное $R@1$ выросло до 0.7105, однако между 90 и 100 эпохами обучения возникает устойчивый эффект, при котором $R@1$ падает до 0. Выяснилось, что во время подсчета градиента по параметрам нейросети в вычислениях возникают NaN'ы.

Тогда мы стали использовать оптимизатор AdamW с параметрами weight_decay=0.01, lr=0.00001 (как в статье про HypViT), нормализовывать эмбеддинги на выходе из трансформера, а также использовать классическую версию триплет функции потерь с гиперпараметром $m = 0.2$ (документация библиотеки OML указывает, что обычно оптимальное значение лежит в интервале от 0.1 до 0.2). Кроме того, начал производиться клиппинг градиента по норме 2 для более стабильного обучения. Наилучшее качество на валидации: $R@1 = 0,7122$. График $R@1$ ($CMC@1$) на валидации представлен на рисунке 3.2

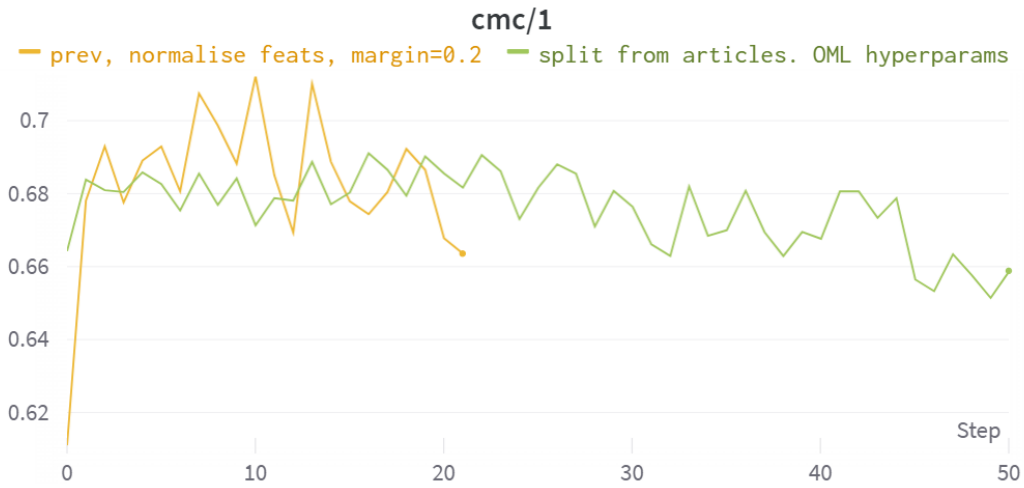

Рис. 3.2: График метрики $R@1$ на валидации. Желтым цветом показан эксперимент с первыми поправками гиперпараметров. Также, начиная с этого эксперимента, на шаге 0 (абцисса 0 на графике) указывается качество претренированной модели до начала файн-тьюнинга

3.3 Подбор майнера и аугментаций

Далее мы использовали триплет функцию потерь с гиперпараметром $m = 0.15$, а также применили майнер, который для каждого объекта батча ищет для него самый сложный позитив и самый сложный негатив внутри батча, формируя таким образом триплеты. Качество сильно выросло: лучший результат на валидации $R@1 = 0.7326$.

После чего мы стали использовать аугментации из той же статьи про HypViT. Стоит упомянуть, что данные аугментации чаще всего используются в статьях по метрическому

обучению, поэтому при их применении сравнение с результатами этих статей будет более корректным. Качество выросло очень сильно: лучший результат на валидации $R@1 = 0.8422$. Результаты эсперимента на рисунке 3.3

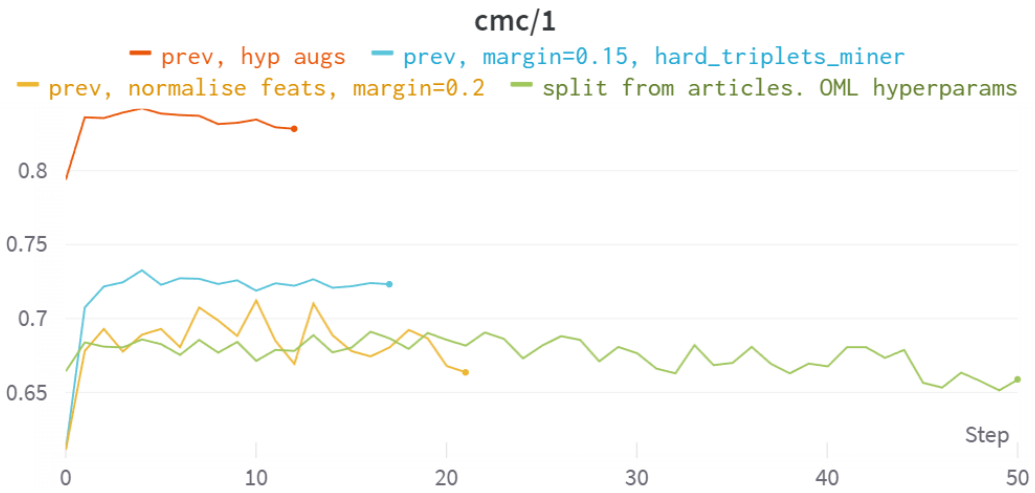

Рис. 3.3: График метрики $R@1$ на валидации. Синим цветом показан эксперимент с майнером, который для каждого объекта батча ищет для него самый сложный позитив и самый сложный негатив внутри батча. Красным цветом показан эксперимент с аугментациями из статьи про HypViT

Значительное улучшение качества при использовании новых аугментаций несложно объяснить. Дело в том, что в CUB птицы на картинках зачастую довольно маленькие и занимают лишь небольшую часть картинки. Поэтому на качество сильно влияет обрезание картинки. Новые преобразования как раз это и делают. При обучении производится случайное обрезание картинки таким образом, чтобы отношение сторон вырезанного фрагмента находилось в интервале от 0.75 до 1.33, а отношение площади вырезанного фрагмента к площади изначальной картинки находилось в интервале от 0.2 до 1. После чего производится изменение размера картинки до 224×224 и случайный горизонтальный разворот с вероятность 0.5. Во время валидации делается изменение размера картинки до 256×256 и вырез из нее центрального фрагмента размером 224×224.

3.4 Эксперименты с майнером и претренированными моделями

Далее проводились эксперименты:

1) с майнером, который выбирает из текущего батча очень большое количество триплетов случайным образом в надежде, что обучающий сигнал будет больше и некоторые из триплетов все же окажутся «сложными». Максимум на валидации $R@1 = 0.8371$

2) со старым майнером, но с претренированной моделью ViT-S DINO с размером патча

16×16 (данная модель также хорошо показала себя в статье про HypViT). Максимум на валидации $R@1 = 0.7757$

3) со старым майнером, но с претренированной моделью ViT-S DINO с размером патча 8×8. Эксперименты в статье [4] наглядно показали, что использование патча меньшего размера значительно улучшает качество в задачах обучения без учителя, а DINO как раз тренируется без учителя. Из-за этого время обучения сильно увеличивается, однако качество действительно растет - максимум на валидации $R@1 = 0.8304$

Результаты экспериментов на картинке 3.4

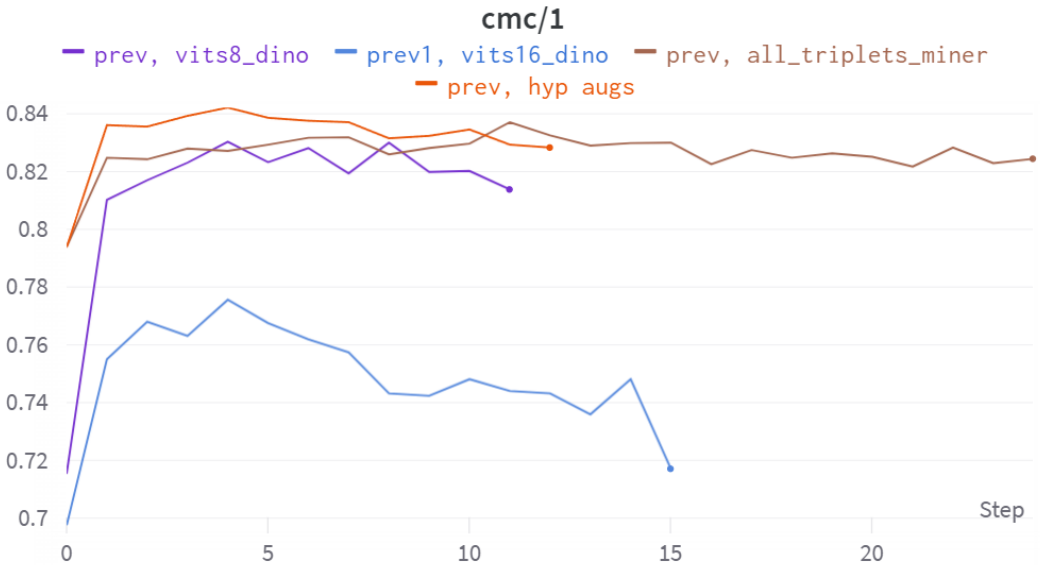

Рис. 3.4: График метрики $R@1$ на валидации. Коричневый, синий и фиолетовый цвета соответствуют экспериментам, описанным в пунках 1, 2, 3 секции 3.4

По графикам можно видеть, что отдельного внимания заслушивает модель ViT-S DINO 8×8, поскольку она вообще не требует никакой разметки данных для первичного обучения, но ее итоговое качество сравнимо с моделью, которая изначально училась на размеченном ImageNet.

3.5 Эксперименты с головной частью модели

Предыдущий эксперимент не дал прироста качества, поэтому дальнейшие исследования использовали лучшие гиперпараметры из пункта 3.3 и были направлены на изучение архитектуры головной части модели. Многие статьи после самого трансформера используют проекторы в пространства меньшей размерности или даже в более сложные пространства, не являющиеся линейными. При этом порой отбрасывание головной части после файн-тьюнинга еще больше увеличивает валидационные метрики.

У нашей претренированной модели изначальная голова - линейный слой с 384 входами и 1000 выходов. Ее мы убирали во всех следующих экспериментах:

1) голова состоит только из слоя нормализации эмбеддингов

2) голова состоит из линейного слоя с 384 входами и 128 выходами и слоя нормализации эмбеддингов

3) голова состоит из линейного слоя с 384 входами и 128 выходами и слоя нормализации эмбеддингов, но на валидации мы используем только выход из трансформера, не пропуская его через голову

4) голова состоит из линейного слоя с 384 входами и 128 выходами и слоя нормализации эмбеддингов, но на валидации мы используем только нормализованный выход из трансформера, не пропуская его через голову

В линейных слоях смещение инициализировалось 0, а матрица линейного преобразования - ортонормированными столбцами. Результаты экспериментов на рисунке 3.5

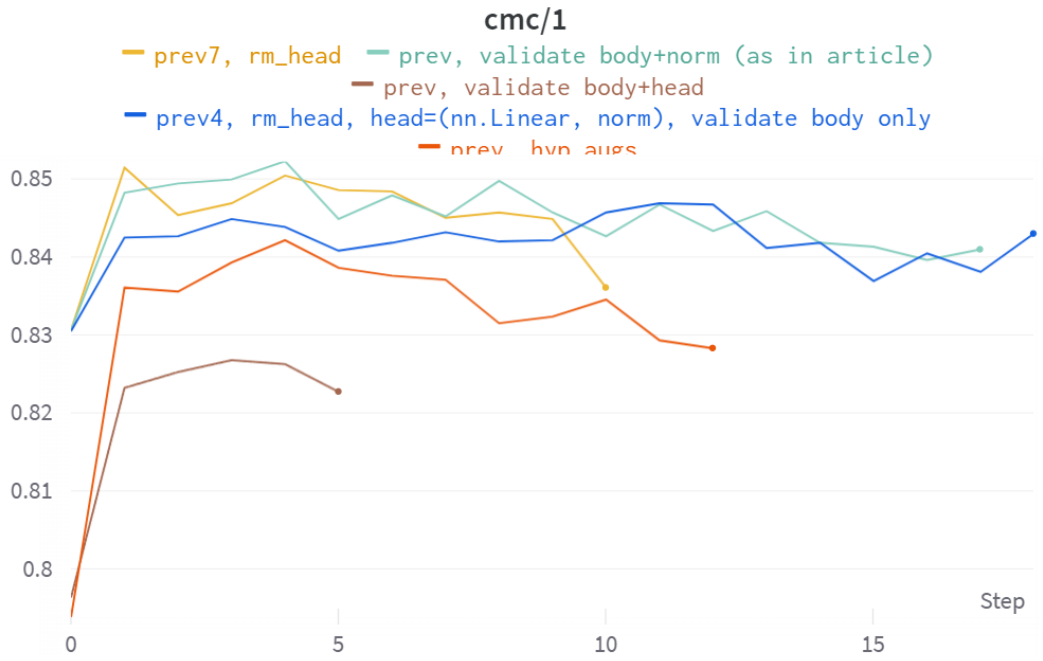

Рис. 3.5: График метрики $R@1$ на валидации. Желтый, коричневый, синий и бирюзовый цвета соответствуют экспериментам, описанным в пунках 1, 2, 3, 4 секции 3.5. Красным цветом показан лучший результат предыдущих секций

Внимания заслуживает эксперимент номер 4, однако дальнейшие эксперименты проще проводить с конфигурацией в пункте 1, поскольку она интуитивно понятней.

Итак, лучший результат первой конфигурации на валидации: $R@1 = 0.8515$

3.6 Подбор отступа m в триплетной функции потерь

С этого момента валидация проводилась 1 раз в 5 эпох (раньше было раз в 10 эпох).

Следующие эксперименты посвящены подбору лучшего гиперпараметра отступа m в триплет функции потерь 3.6

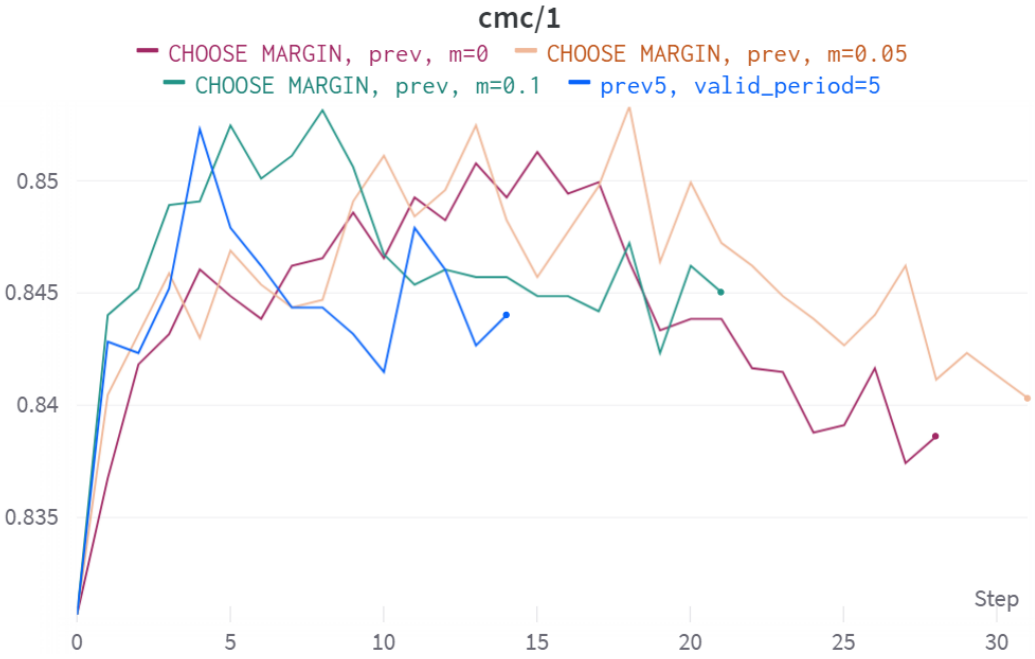

Рис. 3.6: График метрики $R@1$ на валидации. Соответствие цветов и величины m: синий - 0.15, зеленый - 0.1, бледно-оранжевый - 0.05, вишневый - 0

При $m = 0.1$ лучшее $R@1$ равно 0.8531

Заметим, что лучший показатель $R@1$ в статье про HypViT равен 0.856

4 Модификации триплетной функции потерь

4.1 Идеи модификаций из контрольной точки курсовой работы

В тексте контрольной точки курсовой работы был предложен ряд идей модификации классической архитектуры триплетной функции потерь. Разберем некоторые из них.

4.1.1 Добавление $d(p, n)$ в обычную формулу триплет лосса

Была предложена идея следующей функции потерь

$$\mathcal{L}(a, p, n) = \max(0, m + d(a, p) - d(a, n) - \lambda d(p, n))$$

где λ - гиперпараметр. Однако у такой формулы нашлись существенные недостатки:

1) разность $d(p,n)-d(a,p)$ понятно интерпретируема, и классическая триплет функция потерь старается сделать ее не меньше m. Теперь же выражение $d(a,p)-d(a,n)-\lambda d(p,n)$ не имеет простой интерпретации

2) непонятно, из каких логических соображений выбирать гиперпараметр m. Очевидно, что прежнее его значение совсем не подходит

Эксперименты с подбором m и λ привели лишь к ухудшению качества на валидации.

4.1.2 Взвешивание расстояний, идея фокального лосса

Было предложено использовать идею из focal loss и перевзвесить позитивные и негативные расстояния в триплет лоссе относительно радиуса или диаметра описанной вокруг них окружности. Например, можно рассмотреть такой лосс

$$\mathcal{L}(a,p,n)=\left(\frac{d(a,p)}{\max(d(a,p),d(a,n),d(p,n))}\right)^{\gamma_1}\cdot d(a,p)-\left(1-\frac{d(a,n)}{\max(d(a,p),d(a,n),d(p,n))}\right)^{\gamma_2}\cdot d(a,n)$$

Однако в этой формуле и подобным ей возникают явные примеры противоречивого поведения функции потерь. Например, для формулы выше рассмотрим ситуацию, при которой $0<d(a,n)<d(a,p)$ и $d(p,n)$ невелико (скажем, меньше $d(a,p)$). Завиксируем эмбеддинги a и p ($d(a,p)$ фиксированно) и начнем удалять n от a (увеличивать $d(a,n)$). Когда окажется, что $d(a,n)=d(a,p)$, заметим, что $\max(d(a,p),d(a,n),d(p,n))=d(a,p)$ не изменился, левое слагаемое не изменилось, а правое из отрицательного стало 0. То есть функция потерь увеличилась, в то время как ситуация улучшилась (эмбеддинги a и n удалились друг от друга).

4.2 Кластерный майнер

Майнер и триплетная функция потерь - неразделимые вещи, а потому усовершенствование майнера также является модификацией функции потерь.

Мы провели серию экспериментов с майнером, который для каждого класса в батче усредняет эмбеддинги всех его объектов - получает центры классов. После чего для каждого центра класса находит ближайший к нему объект другого класса (сложный негатив) и дальнейший от него объект этого же класса (сложный позитив), формируя таким образом триплеты (центр класса, сложный позитив, сложный негатив).

Однако эксперименты с данным майнером не привели ни к какому успеху, качество модели лишь значительно ухудшилось.

4.3 Майнинг полусложных негативов и позитивов

Нередко встречаются статьи, которые утверждают, что ислъзование лишь самых сложных позитивов и негативов порождает большой шум в градиенте и потому приводит к сходимости в плохие локальные оптимумы функции потерь [24]. Они как правило предлагают майнить как сложные, так и более простые позитивы и негативы.

У нас также есть этому обоснование. Дело в том, что используемые аугментации порой приводят к ситуации, изображенной на рисунке 4.1

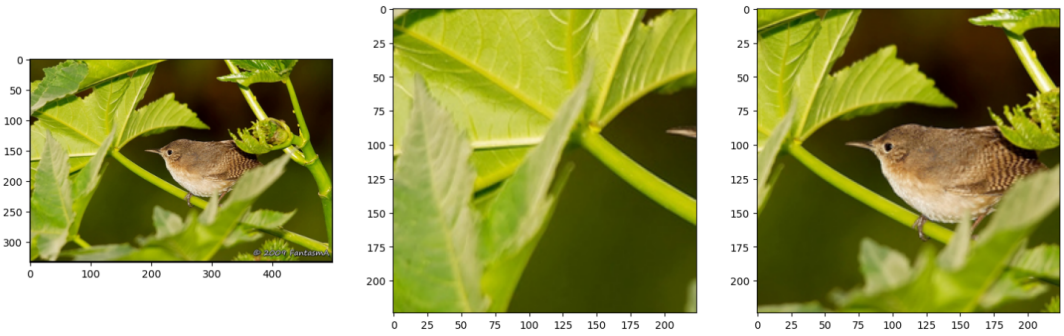

Рис. 4.1: Пример использования аугментаций. Слева - изначальная картинка из датасета, посередине - применение аугментации при обучении, справа - аугментации при валидации

Кажется, птичка не попала в кадр! Действительно, изображение посередине легко может оказаться сложным позитивом для другой фотографии этой же птички. Не стоит учить нейросеть на таких примерах. Это также служит мотивацией для использования полусложного майнинга.

Мы стали выбирать более простые негативы. Чаще всего это первый и второй по сложности негатив, так как использование более простых при формировании триплетов не дало особых результатов. Результаты полусложного майнинга негативов приведены на рисунке 4.2

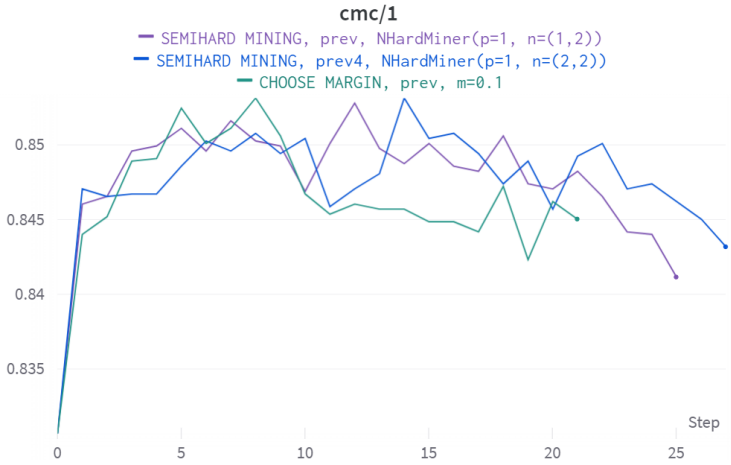

Рис. 4.2: График метрики R@1 на валидации. Нотация NHardMiner(p=1, n=(a, b)) означает, что выбираются негативы с a-ого по b-ый в порядке увеличения их расстояния до якоря

Здесь видно, что используемый метод немного снижает скорость переобучения.

Также мы поставили эксперименты с полусложным майнингом позитивов. Результаты на рисунке 4.3

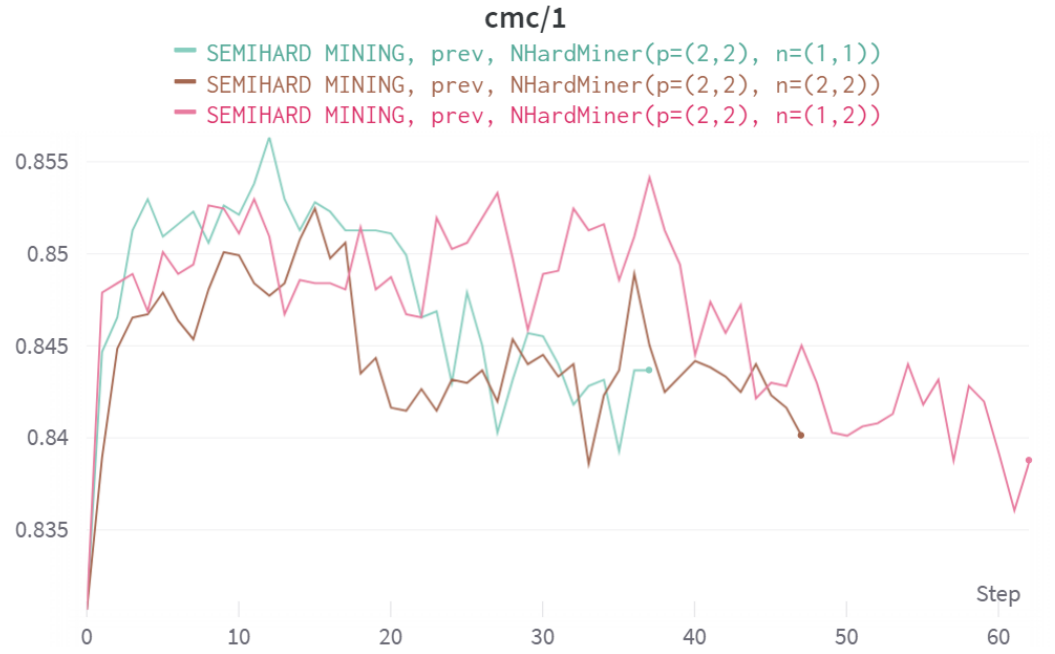

Рис. 4.3: График метрики $R@1$ на валидации. Нотация NHardMiner(p=(a, b), n=(c, d)) означает, что выбираются позитивы с a-ого по b-ый в порядке уменьшения их расстояния до якоря и негативы с c-ого по d-ый в порядке увеличения их расстояния до якоря

Наилучший результат показал майнер, выбирающий первый по сложности негатив и второй по сложности позитив. Его лучшее $R@1$ равно 0.8563, что уже чуть больше, чем у HypViT.

4.4 Увеличивающийся гиперпараметр отступа m

Мы также попробовали в процессе обучения постепенно увеличивать параметр отступа m. Однако все эти эксперименты не дали никакого улучшения, а лишь ухудшили прошлые результаты. Мы постарались выяснить причину этого.

Для этого мы стали измерять дополнительные метрики в процессе обучения:

1) валидационные метрики качества на обучающей выборке

2) минимальное, максимальное и среднее значение для $d(a,p)$, $d(a,n)$, $d(p,n)$, $d(a,n) - d(a,p)$ на обучении

3) минимальное, максимальное и среднее значение среднеквадратичного отклонения эмбеддингов каждого класса от среднего эмбеддинга класса (*размер класса*) на обучении и на валидации

4) минимальное, максимальное и среднее значение расстояния между средними эмбеддингами всех классов (*межклассовое расстояние*) на обучении и валидации

В идеале достичь такого состояния модели, при котором межклассовое расстояние велико, а размер классов в среднем маленький. Однако выяснилось, что при большом значении m происходит переобучение именно размеров классов. Это можно увидеть на рисунке 4.4

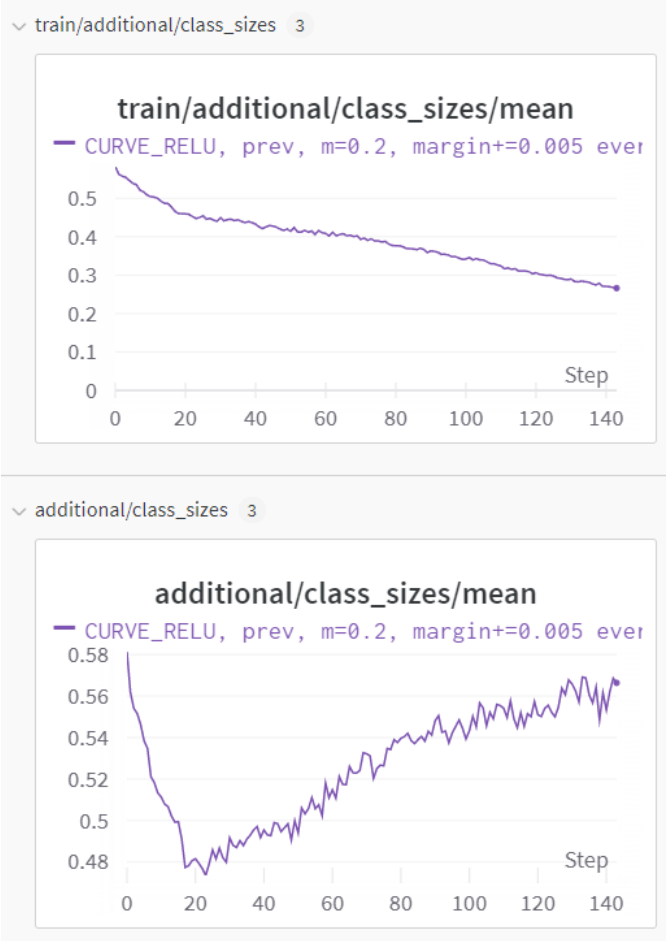

Рис. 4.4: Графики усредненного по классам значения среднеквадратичного отклонения эмбеддингов каждого класса от среднего эмбеддинга класса. Сверху график на обучающей выборке, снизу - на валидационной.

Из-за слишком большого отступа функция потерь начинает учитывать и без того хорошие триплеты (для которых $d(a,n) >> d(a,p)$). И потому, начиная с некоторого момента, размеры классов начинают увеличиваться.

4.5 Использование пороговой функции

По сути функцию $\max(0,\cdot)$ в формуле триплетной функции потерь можно записать как $x \cdot [x > 0]$, где в квадратных скобках указан индикатор. Мы попробовали использовать другую функцию: $x \cdot [x > 0] \cdot [x < t]$. Мотивация этого заключается в том, чтобы не учитывать при обучении триплеты, для которых $d(a,p) >> d(a,n)$. Вполне возможно, что на

этих триплетах плохо применились аугментации. Либо эти триплеты настолько сложны, что попытки учить нейросеть на них приведет только к ухудшению общего качества модели. Исходя из статистики по минимальным значениями $d(a,n) - d(a,p)$, мы подобрали примерный порог, по которому возможно стоит отсекать триплеты. Мы получили, что оптимальное $t \in [m + 0.4, m + 0.5]$. Однако наши эксперименты не увенчались успехом. Их результаты на рисунке 4.5

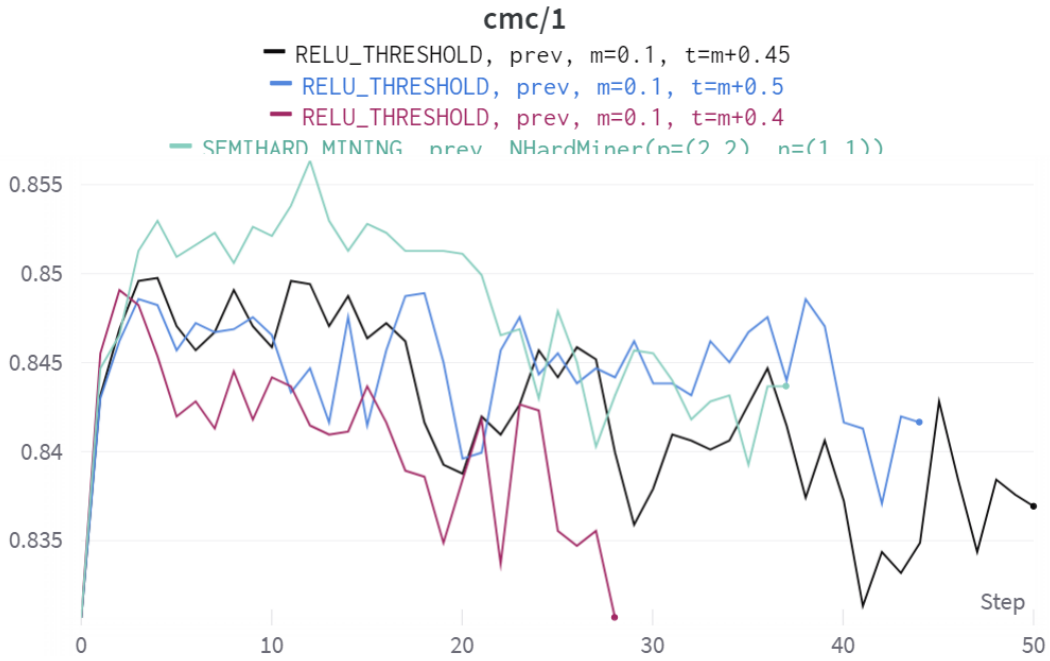

Рис. 4.5: График метрики R@1 на валидации. Гиперпараметры m и t указаны над графиками

И это в целом объяснимо. В некотором смысле наша цель как раз найти наиболее сложные триплеты и учиться на них. Видимо содержание «плохих» триплетов среди сложных довольно мало.

4.6 Сглаживание степенной функцией

Ту же функцию $\max(0, \cdot)$ можно попробовать сгладить следующим образом: $x^\gamma \cdot [x > 0]$, где γ - гиперпараметр, отвечающий за кривизну. Мотивация заключается в том, чтобы слабее учить те триплеты, которые и так хороши, а чем хуже триплет, тем сильнее от него будет протекать обучающий градиент.

Наилучшее качество показала конфигурация с гиперпараметрами $m = 0.2, \gamma = 2.0$, при которой лучшее значение R@1 равно 0.854.

Результаты наиболее удачных экспериментов на графике 4.6

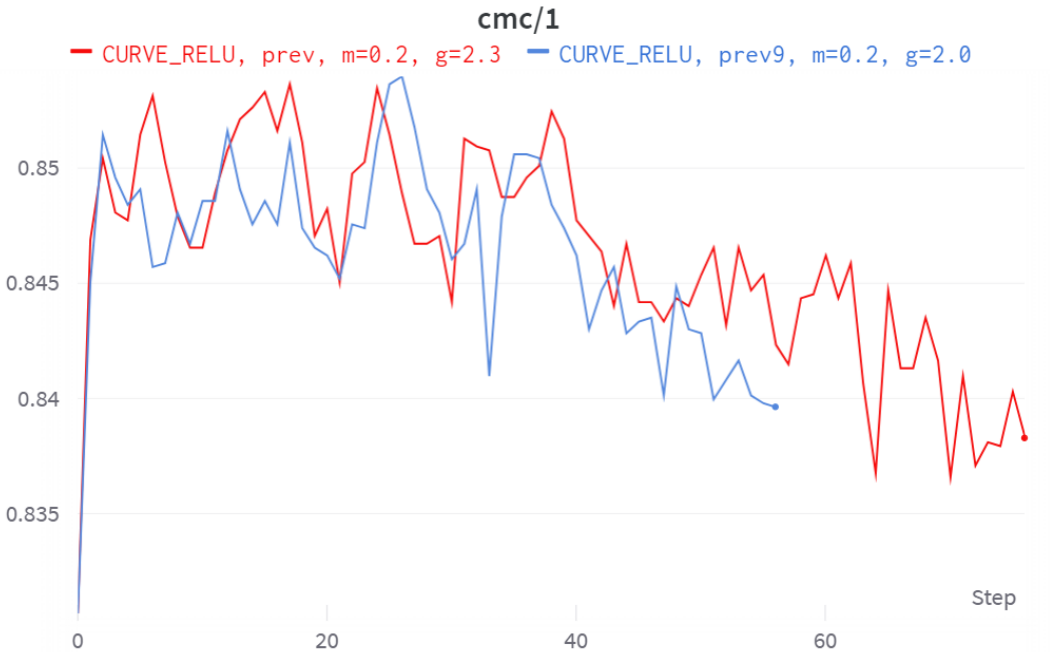

Рис. 4.6: График метрики $R@1$ на валидации. Гиперпараметры m и γ указаны над графиками

5 Заключение и дальнейшее направление исследований

Мы испробовали немало различных модификаций и каждая из них привела к хорошим или плохим результатам. Однако на некоторые эксперименты не хватило времени. Например, привлекательной выглядит идея совмещения сглаживания степенной функцией с полусложным майнингом, так как обе эти стратегии дали неплохие результаты.

Касательно полусложного майнинга, мы совсем не пробовали подбирать более удачный гиперпараметр m для каждого майнера по отдельности, а также не экспериментировали с размером батча.

Кроме того, совсем не исследовалась возможность сделать обучаемой метрику d между эмбеддингами в признаковом пространстве.

Таким образом, в результате проделанной работы можно заявить, что даже классическая версия триплетной функции потерь при хорошо подобранных гиперпараметрах дает качество, сравнимое с современными продвинутыми подходами глубинного метрического обучения. С другой стороны, некоторые модификации данной функции потерь также имеют определенный смысл.

Список литературы

[1] Alaa Alnissany и Yazan Dayoub. “Modied Centroid Triplet Loss for Person ReIdentication”. В: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2022. URL: https://assets.researchsquare.com/files/rs-1501673/v1_covered.pdf?c=1648739388.

[2] Xiang An, Jiankang Deng, Kaicheng Yang, Jaiwei Li, Ziyong Feng, Jia Guo, Jing Yang и Tongliang Liu. “Unicom: Universal and Compact Representation Learning for Image Retrieval”. В: *arXiv preprint arXiv:2304.05884* (2023).

[3] Fadi Boutros, Naser Damer, Florian Kirchbuchner и Arjan Kuijper. “Self-restrained Triplet Loss for Accurate Masked Face Recognition”. В: *Pattern Recognition* (2021). URL: https://arxiv.org/pdf/2103.01716.pdf.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski и Armand Joulin. “Emerging properties in self-supervised vision transformers”. В: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, с. 9650—9660.

[5] Kanghao Chen, Weixian Lei, Rong Zhang, Shen Zhao, Wei-shi Zheng и Ruixuan Wang. “PCCT: Progressive Class-Center Triplet Loss for Imbalanced Medical Image Classification”. В: *IEEE Journal of Biomedical and Health Informatics* (2017). URL: https://arxiv.org/pdf/2207.04793.pdf.

[6] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia и Stefanos Zafeiriou. “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”. В: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2015. URL: https://arxiv.org/pdf/1801.07698.pdf.

[7] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe и Ivan Oseledets. “Hyperbolic vision transformers: Combining improvements in metric learning”. В: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, с. 7409—7419.

[8] Benyamin Ghojogh, Milad Sikaroudi и Sobhan Shafiei. “Fisher Discriminant Triplet and Contrastive Losses for Training Siamese Networks”. В: *2020 international joint conference on neural networks (IJCNN)*. 2020. URL: https://arxiv.org/pdf/2004.04674.pdf.

[9] Mai Lan Ha и Volker Blanz. “Deep Ranking with Adaptive Margin Triplet Loss”. В: *arXiv preprint arXiv:2107.06187* (2021). URL: https://arxiv.org/pdf/2107.06187.pdf.

[10] Kalun Ho1, Janis Keuper, Franz-Josef Pfreundt и Margret Keuper. “Learning Embeddings for Image Clustering: An Empirical Study of Triplet Loss Approaches”. В: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2020. URL: https://arxiv.org/pdf/2007.03123.pdf.

[11] Sungyeon Kim, Dongwon Kim, Minsu Cho и Suha Kwak. “Proxy anchor loss for deep metric learning”. В: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, с. 3238—3247.

[12] Yitian Li, Ruini Xue, Mengmeng Zhu, Jing Xu и Zenglin Xu. “Angular Triplet Loss-based Camera Network for ReID”. В: *2021 International Joint Conference on Neural Networks (IJCNN)*. 2021. URL: https://arxiv.org/pdf/2005.05740.pdf.

[13] Haijun Liu, Yanxia Chai, Xiaoheng Tan, Dong Li и Xichuan Zhou. “Strong but Simple Baseline with Dual-Granularity Triplet Loss for Visible-Thermal Person Re-Identification”. В: *IEEE Signal Processing Letters* (2021). URL: https://arxiv.org/pdf/2012.05010.pdf.

[14] Ruan van der Merwe. “TRIPLET ENTROPY LOSS: IMPROVING THE GENERALISATION OF SHORT SPEECH LANGUAGE IDENTIFICATION SYSTEMS”. В: *arXiv preprint arXiv:2012.03775* (2020). URL: https://arxiv.org/pdf/2012.03775.pdf.

[15] Kevin Musgrave, Serge Belongie и Ser-Nam Lim. “A metric learning reality check”. В: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer. 2020, с. 681—699.

[16] “OML Github”. URL: https://github.com/OML-Team/open-metric-learning.

[17] “OML: code of default augmentations for CUB dataset”. URL: https://github.com/OML-Team/open-metric-learning/blob/5bc870223eaaf9a3a64fe8e473c59ada3cd378a1/oml/transforms/images/albumentations.py#LL106C7-L106C7.

[18] “Papers With Code | Metric Learning”. URL: https://paperswithcode.com/task/metric-learning.

[19] Florian Schroff, Dmitry Kalenichenko и James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”. В: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. URL: https://arxiv.org/pdf/1503.03832.pdf.

[20] Aleksei Shabanov, Aleksei Tarasov и Sergey Nikolenko. “STIR: Siamese Transformer for Image Retrieval Postprocessing”. В: *arXiv preprint arXiv:2304.13393* (2023).

[21] Aleksey Shabanov. “Metric Learning with Catalyst”. 2020. URL: https://medium.com/pytorch/metric-learning-with-catalyst-8c8337dfab1a.

[22] Aleksey Shabanov. “Practical Metric Learning”. 2022. URL: https://medium.com/@AlekseiShaba practical-metric-learning-b0410cda2201.

[23] Frederik Warburg, Martin Jørgensen, Javier Civera и Søren Hauberg. “Bayesian Triplet Loss: Uncertainty Quantification in Image Retrieval”. В: *Proceedings of the IEEE/CVF International conference on Computer Vision*. 2021. URL: https://arxiv.org/pdf/2011.12663.pdf.

[24] Chao-Yuan Wu, R Manmatha, Alexander J Smola и Philipp Krahenbuhl. “Sampling matters in deep embedding learning”. В: *Proceedings of the IEEE international conference on computer vision*. 2017, с. 2840—2848.

[25] “Метрики качества ранжирования”. URL: https://habr.com/ru/companies/econtenta/articles/303458/.