Академический лицей
«Физико-техническая школа»
им. Ж. И. Алфёрова

Отчет о практике

Few-Shot Learning

Ученик: Цветков Петр (11 А класс)

Научный руководитель: Шпильман Алексей Александрович

Санкт-Петербург, 2020 год

Аннотация

Few-Shot Learning (FSL) - активно исследуемая в наше время задача в области компьютерного зрения, подразумевающая классификацию изображений на основе малого числа примеров. В данной работе, помимо традиционного сценария FSL (5 классов, по 1 или 5 примеров на класс), рассматривается усложненная версия задачи, в котором количество классов много больше, чем число образцов для каждого класса (100 классов, по 5 примеров на класс), а так же задача применения опыта, полученного из одного датасета, к другому. Для этого был выбран метод решения, основанных на т. н. ProtoNet Classifier [ссылка]. Были проанализированы современные исследования в этой области, описанные в них модели реализованы, по-разному скомбинированы и протестированы в описанных сценариях. Для оценки использовался как популярный среди исследователей датасет miniImageNet [ссылка], так и собранный самостоятельно для этой задачи датасет, представляющий собой подмножество GoogleLandmarks [ссылка]. В результате работы были определены наиболее эффективные методы решения FSL для различных сценариев, что может быть полезно как при будущих исследованиях, так и при создании промышленных решений в области компьютерного зрения.

Оглавление

1. Введение **2**

2. Постановка задачи **3**

3. Методика **4**
3.1. Общие обозначения и определения 4
3.2. Алгоритмы решения 4
3.2.1. Функции потерь 5
3.2.2. Преобразование прототипов 7
3.2.3. Функции сходства 8
3.3. Данные 9
3.4. Сценарии и условия экспериментов 10

4. Результаты **12**

5. Вывод **13**

6. Приложение **14**
6.1. Использованные архитектуры нейросетей 14
6.2. Полные результаты экспериментов 14

1. Введение

Задача классификации изображений является базовой в компьютерном зрении. С появлением сверточных нейронных сетей в этой области произошел прорыв - в 2012 году нейросеть AlexNet достигла на датасете ImageNet (1.2 млн изображений, разделенных на 1000 классов) результата в 15.3% ошибок (приблизительно на 10% ниже, чем 2 место) [ссылка]. В последующие годы более сложные модели уже превзошли в этой задаче человека - в 2017 году ошибка на ImageNet составляла уже около 2.5% - 3% (у человека - 5%), в связи с чем соревнование по классификации изображений по данному сценарию перестало проводиться.

Однако для обучения нейросетей были использованы миллионы примеров, в то время как человек умеет различать объекты основываясь всего на нескольких примерах - например, один раз посмотрев на фотографию автомобиля, мы без труда сможем узнать такой же на улице. Упомянутым выше алгоритмам потребовалось бы проанализировать сотни фотографий, чтобы отличать этот автомобиль от других. Кроме того, традиционные модели для классификации изображений не могут быть легко расширены на новые классы - нейросеть придется доучивать на новом наборе данных.

Эти проблемы исследователи решают в рамках задачи Few-Shot Learning (в переводе - «обучение на малочисленных кадрах»), сокращенно FSL. Ее можно сформулировать так: необходимо для поданных на вход изображений определить, к какому классу они относятся, опираясь лишь на несколько (1 - 5 штук) примеров к каждому классу. Такая задача намного труднее, чем традиционная классификация, так как количество доступных данных уменьшается в десятки раз.

Исследования в области Few-Shot Learning позволяют создать более универсальные алгоритмы, не требующие перенастройки и сбора большого количество данных для адаптации к новым задачам.Такие модели могут применяться при обработке созданных людьми изображений (например, при анализе фотографий в социальных сетях) или при создании автономных роботов (например, дронов-беспилотников и самоуправляемых автомобилей).

2. Постановка задачи

Основными задачами данной работы являются:

- **Создание сценария для проверки моделей на большом количестве классов.** Традиционно используемые в работах на тему FSL наборы данных имеют не так много классов (miniImageNet [ссылка], к примеру, всего имеет 100 классов), а модели чаще всего тестируются в условиях, когда выбор нужно сделать из 5 классов. В реальной жизни необходимо зачастую различать десятки или сотни классов в рамках одной задачи. Для того, чтобы протестировать модели в таких усложненных условиях, необходимо подготовить новый набор данных.

- **Разработка программного обеспечения для обучения и тестирования моделей, реализация различных алгоритмов для FSL.** Для проведения большого числа экспериментов на различных наборах данных необходима программная среда, позволяющая по-разному конфигурировать модели, автоматически сохранять результаты экспериментов и обученные модели.

- **Выявление наиболее эффективных методов для различных случаев.** В результате работы будет установлено, какие комбинации моделей и алгоритмов лучше всего работают для того или иного подвида Few-Shot Learning.

3. Методика

3.1. Общие обозначения и определения

Для удобства введем некоторые обозначения и с их помощью сформулируем задачу более конкретно. Пусть *датасет* D - множество пар (x_i, y_i), где x_i - *изображение* из данного датасета, а y_i - *класс* этого изображения. *Множество всех классов датасета* $C = \{y_i\}$, а *количество различных классов* в датасете - $n_{classes} = |C|$. Общее количество элементов в датасете $N = |D|$.

Задача Few-Shot Learning характеризуется двумя параметрами: количеством классов n количеством примеров на класс k. Такая задача называется *n-way k-shot learning*.

Основной используемый датасет D_{base} делится на два непересекающихся по элементам и классам датасета D_{train} и D_{test} ($D_{train} \cup D_{test} = D_{base}$; $C_{train} \cap C_{test} = \varnothing$). На D_{train} модель обучается, выделяя признаки, необходимые для классификации новых изображений; D_{test} используется только на этапе тестирования решения.

В процессе обучения/оценки модели из датасетов случайным образом выбираются *задания*. Задание состоит из *набора образцов* S - датасета с n классами и k примерами на каждый класс (всего в нем $n \times k$ элементов) и *набора запросов* Q - набора изображений, для которых модель должна предсказать класс, основываясь на примерах из S. На этапе тренировки модели предъявляются задания из D_{train}, на этапе тестирования - задания из D_{test}.

3.2. Алгоритмы решения

В данной работе рассматриваются решения, основанные на т. н. ProtoNet [ссылка]. Основной частью такой модели является сверточная нейронная сеть - представим ее как функцию f_θ, где θ - обучаемые параметры сети. f_θ принимает на вход изображение x_i и преобразовывает его в вектор признаков: $f_\theta(x_i) = \hat{x}_i$. Необходимо заметить, что данные, возвращаемые f_θ представляют собой *карты признаков*, а значит имеют два пространственных измерения, помимо измерения признаков - для превращения

этого трехмерного тензора в одномерный вектор все его измерения объединяются в одно (с потерей пространственной информации), однако в описанном далее DFMN будут использоваться именно карты признаков, не превращенные в вектор. В задании T для каждого класса $c \in C_T$ вычисляется вектор-прототип $\langle \hat{x}_c \rangle$ - путем усреднения векторов признаков для изображений этого класса, входящих в S_T. Пусть $d(\bar{a}, \bar{b})$ - используемая в модели функция сходства двух векторов (чем слабее векторы различаются, тем больше значение функции; примером такой функции служит Евклидово расстояние, умноженное на -1: $d(\bar{a}, \bar{b}) = -\sqrt{(\bar{a} - \bar{b})^2}$). Для каждого $\hat{x}_q$ вектора признаков изображения - запроса $x_q \in Q_T$ выполняется поочередное сравнение со всеми прототипами $\langle \hat{x}_c \rangle : \forall c \in C_T$. Таким образом для каждого класса мы получаем число, характеризующее степень «похожести» запроса на этот класс. Для $T \leftarrow D_{train}$, используя функцию потерь (в данном случае - перекрестную энтропию) вычисляются градиенты и обновляются параметры сети θ. Для $T \subset D_{test}$ к полученным числам применяется функция *softmax*, в результате чего на выходе модели получаются вероятности принадлежности запроса каждому из классов.

Это базовый алгоритм решения может быть модифицирован в нескольких местах. В данной работе рассматриваются 3 вида модификаций: описание каждого вида приведено ниже.

3.2.1. Функции потерь

При обучении нейросетей на малом количестве данных может возникать проблема *переобучения* - ситуация, когда нейросеть «запоминает» тренировочные примеры, выделяет недостаточно обобщенные признаки, и на тестовых изображениях точность оказывается сильно ниже, чем на тренировочных. Для борьбы с переобучением используются т. н. *вспомогательные задачи* - на этапе тренировки та же нейронная сеть f_θ параллельно с основной задачей используется для решения дополнительных - так модель меньше «фокусируется» на малочисленных обучающих примерах и получается более универсальной. При использовании вспомогательных задач вычисляются дополнительные функций потерь, которые складываются с основной (возможно, с коэффициентами) перед вычислением градиентов и обновлением θ. Особенностью этого класса модификаций является то, что они влияют только на процесс обучения - при тестировании модель, обученная с вспомогательными задачами, работает по тому же алгоритму, что и обученная без них. Далее описаны

вспомогательные задачи, использованные в рамках данной работы.

Dense Feature-Matching network (DFMN) [ссылка]. Задания T, получаемые моделью при обучении содержат в себе мало классов и примеров на класс. В то же время, на этапе обучения нам доступен весь датасет D_{train}. Чтобы задействовать все данные, метод DFMN предполагает создания для каждого класса из D_{train} *глобального* обучаемого вектора-прототипа. Эти векторы обновляются вместе с другими параметрами модели, поэтому их значения доступны вне зависимости от конкретного задания в течение всего обучения. Получая задание T модель, помимо стандартной FSL классификации на основе S_T осуществляет в качестве вспомогательной задачи классификацию на основе этих глобальных прототипов (используя Евклидово сходство). Функции потерь для результатов двух классификаций складываются (при этом значение основной функции домножается на константу 0.2). Еще одной особенностью данного метода является то, что хотя карты признаков, полученные от нейросети f_θ имеют пространственное измерение, глобальный прототипы его не имеют. Вспомогательная классификация осуществляется для *векторов признаков от всех пикселей* карт признаков изображений - запросов. Например, если f_θ возвращает трехмерный тензор m размера $H \times W \times K$, то $m_{i,j}$ $(0 \leq i < H;\ 0 \leq j < W)$ - это вектор размера K. Глобальные прототипы - тоже векторы размера K, классификация происходит для всех $m_{i,j}$ $(0 \leq i < H;\ 0 \leq j < W)$. В остальных частях модели значения, полученные из f_θ все так же представляются в виде одномерного вектора размера $H \cdot W \cdot K$.

Распознавание поворота [ссылка]. Для этой вспомогательной задачи создается дополнительная нейронная сеть - классификатор (ее архитектура описана в приложении) g_ϕ, где ϕ - параметры. g_ϕ принимает на вход векторы признаков от f_θ и возвращает вектор из 4 чисел, предсказывающих, насколько вероятно, что картинка повернута на один из четырех углов - 0°, 90°, 180°, 270°. Таким образом, распознавание поворота - традиционная задача классификации изображений по 4 классам. Во время итерации обучения на задании T изображения этого задания x_i случайным образом поворачиваются на один из указанных углов, а затем подаются на вход композиции функций $g_\phi(f_\theta(x_i))$, от полученного на выходе вектора считается функция потерь (стандартная для классификации перекрестная энтропия), которая затем складывается с основной функцией потерь. По этой сумме вычисляются градиенты и обновляются θ и ϕ.

3.2.2. Преобразование прототипов

После получения прототипов по алгоритму, описанному выше, к полученным векторам можно применять различные преобразования. Такие преобразования должны быть типа «набор в набор», то есть принимать на вход и возвращать набор векторов-прототипов. В рамках данной работы рассматривается одно такое преобразование, его описание приведено ниже.

FEAT [ссылка]. Это преобразование основано на алгоритме *трансформер*. Трансформер - разработанная в 2017 году архитектура нейронных сетей, предназначенная для обработки наборов данных, в которых необходимо находить взаимосвязи между отдельными элементами. Модели на основе этой архитектуры активно применяются сейчас в задачах обработки естественных языков: например, на трансформере основана GPT-3 - самая продвинутая на данный момент модель для обработки естественного языка.

В моделях для решения FSL используется часть архитектуры трансформер под названием *механизм самовнимания*, который позволяет менять значения прототипа класса, учитывая информацию, полученную из прототипов других классов. Такой метод для Few-Shot Learning называется *FEAT - Few-Shot Embedding Adaptation with Transformer*. Для работы механизма необходимо добавить в обучаемые параметры модели три квадратные матрицы Q, K, V; причем размеры матриц должны совпадать с количеством элементов в векторе прототипа. Из вычисленного по описанным выше правилам прототипа класса $\langle \tilde{x}_c \rangle$ путем умножения на соответствующие матрицы получаются 3 вектора - $\langle \hat{x}_c \rangle \cdot Q = q_c$, $\langle \hat{x}_c \rangle \cdot K = k_c$ и $\langle \hat{x}_c \rangle \cdot V = v_c$ (т. н. запрос, ключ и значение). Вектор *внимания* для запроса q_c вычисляется как:

$$a_c = softmax\left(\frac{q_c \cdot k_j}{\sqrt{n_{dim}}} \mid \forall j \in C\right)$$

где n_{dim} - размер векторов прототипов. Новое значение прототипов вычисляется как произведение вектора внимания на матрицу, являющуюся вектором векторов - значений:

$$\langle \tilde{x}_c \rangle = a_c \cdot (v_j \mid \forall j \in C)$$

то есть новый прототип равен взвешенной сумме векторов - значений q_c, с коэффициентами из вектора внимания . Эти новые прототипы затем проходят через один обучаемый полносвязный линейный слой (т.

е. умножаются на обучаемую матрицу весов), после чего к ним применяется dropout. Для получения итоговых векторов-прототипов классов, которые будут использоваться далее в решении FSL новые прототипы $Dropout\left(Linear\left(\langle\tilde{x}_c\rangle\right)\right)$ складываются со старыми $\langle\tilde{x}_c\rangle$, полученные итоговые векторы-прототипы нормализуются.

3.2.3. Функции сходства

В описании алгоритма решения FSL была упомянута функция схожести $d(\bar{a}, \bar{b})$, характеризующая степени близости векторов $\bar{a}$ и $\bar{b}$. Эта функция может быть разной - в данной работе рассматриваются 3 ее варианта, которые описаны ниже.

Евклидово сходство. Уже упомянутая наиболее очевидная функция схожести, основанная на Евклидовом расстоянии (взятом со знаком -). Формула:

$$d_{euclidean}(\bar{a}, \bar{b}) = -\sqrt{(\bar{a} - \bar{b})^2}$$

.

Масштабированное нормированное Евклидово сходство [ссылка]. Описанная в той же работе, что и DFMN, данная функция сходства построена на базе все того же Евклидова сходства. Однако, в формулу внесены два дополнения: во-первых векторы нормируются по длине, во-вторых уже нормированные векторы делятся на коэффициент, вычисляемый отдельно для каждого вектора при помощи отдельной нейросети s_ψ (архитектура этой сети описана в приложении), где ψ - параметры. Таким образом, итоговая формула выглядит так:

$$d_{scaled\_euclidean}(\bar{a}, \bar{b}) = -\sqrt{\left(\frac{\bar{a}}{s_\psi(\bar{a}) \cdot |\bar{a}|} - \frac{\bar{b}}{s_\psi(\bar{b}) \cdot |\bar{b}|}\right)^2}$$

Параметры ψ обновляются в процессе обучения вместе с остальными параметрами модели.

SEN (Squared root of the Euclidean distance and the Norm distance) [ссылка]. Эта функция сходства так же основана на Евклидовом сходстве: в данном случае под знак корня в формулу обычного Евклидова расстояния добавляется квадрат разности длин векторов с коэффициентом ϵ. Итоговая формула выглядит так:

$$d_{SEN}(\bar{a}, \bar{b}) = -\sqrt{\left(\bar{a} - \bar{b}\right)^2 + \epsilon \cdot \left(|\bar{a}| - |\bar{b}|\right)^2}$$

Значение ϵ во время обучения этом равно:

$$\epsilon = \begin{cases} 1.0 & \text{при сравнении вектора признаков запроса с прототипом правильного класса} \\ -10^{-7} & \text{при сравнении вектора признаков запроса с другими прототипами} \end{cases}$$

Таким образом, данная функция стимулирует модель преобразовывать изображения одного класса в векторы, имеющие одинаковый модуль (т. е. расположенные на гиперсфере в многомерном пространстве), а изображения разных классов - в векторы, имеющие различный модуль. При тестировании, когда метки, определяющие для запросов правильный класс недоступны, $\epsilon = 1.0$

3.3. Данные

Для обучения и оценки моделей, решающих задачу классификации изображений, необходимы большие наборы размеченных данных - датасетов. В данной работе было использовано два датасета, описания которых приведены ниже.

MiniImageNet[ссылка]. Наиболее популярный у исследователей датасет для Few-Shot Learning. Является подмножеством ImageNet - основного датасета в классификации изображений. Состоит из 100 классов, по 600 примеров на каждый. В ходе обучения и тестирования выделяются 60 тренировочных классов, 20 классов для оценки во время тренировки и 20 классов для окончательной оценки модели. Как и в ImageNet, классы сильно различаются между собой: например, там представлены изображения птиц - юрков и в то же время тарелок для супа. Такой датасет требует от модели универсальности, но количество классов в нем ограничено, что стало причиной для поиска следующего датасета.

MiniGoogleLandmarks. Подмножество датасета GoogleLandmarks [ссылка], собранное самостоятельно в рамках работы. Были случайным образом выбраны 4979 классов, каждый из которых представлен 10-20 примерами. Разбивается на 3000 классов для непосредственного обучения, 979 классов для оценки во время тренировки и 1000 классов для окончательной оценки модели. Все классы в этом датасете представляют собой архитектурные достопримечательности (памятники, здания), изображения - фотографии, сделанные с разных ракурсов в разное время. Классы в этом датасете не столь разнообразны, но их количество

позволяет, например, реализовать сценарий, где при тестировании модель в каждом эпизоде классифицирует изображения - запросы по 100 классам. Особенностью этого набора данных, так же является то, что среди данных довольно часто встречаются нерелевантные изображения. Связано это с тем, что исходный датасет GoogleLandmarks во многом формировался на основе фотографий из Википедии, где в статье про архитектурные объекты могут так же встречаться фото табличек, планов и т. д. Такой «естественный» шум в данных приближает задачу к реальному миру.

3.4. Сценарии и условия экспериментов

Всего в данной работе рассматриваются 6 модификаций для модели ProtoNet. Методы из секций **Функции потерь** и **Преобразование прототипов** могут быть добавлены или не добавлены в модель, в то время как функцию сходства необходимо выбрать ровно одну. Число исследуемых моделей, таким образом $2^3 \cdot 3 = 24$.

Производительность каждой модели будет оценена в 5 сценариях:

- **1-shot** и **5-shot 5-way FSL на miniImageNet** - наиболее популярные сценарии в работах на данную тему при оценке общей производительности модели.
- **5-shot 100-way FSL на miniGoogleLandmarks** - сценарий для оценки производительности модели в условиях, когда необходимо решить задачу с большим количеством классов и малым количеством примеров. Так как показатели алгоритмов в этих условиях сильно хуже, чем в случае 5-way miniImageNet, сценарии с еще меньшим количеством примеров (например, 1-shot) на класс не использовались.
- **5-shot FSL с переносом опыта** - этот подвид FSL подразумевает обучение модели на одном датасете, а тестирование на другом - проверяется способность модели обобщать информацию и выделять признаки, неспецифичные для конкретного датасета. В данной работе рассматриваются 2 сценария из этого класса: обучение на miniImageNet, затем тестирование на miniGoogleLandmarks (100-way), и обучение на miniGoogleLandmarks, затем тестирование на miniImageNet (5-way). В первом случае, модель обученную на разнообразном датасете применяют к более однородному датасету, во втором - все происходит наоборот.

Вне зависимости от сценария обучение происходит на 15-way FSL. Это связано с тем, что использование большего числа классов при обучении, чем при тестировании дает лучший результат, однако обучение требует больше ресурсов, чем тестирование, следовательно невозможно обучать модель, например, на 100-way FSL. При выделении задания из датасета как во время тренировки, так и во время тестирования, в качестве Q берутся по 5 (или 3, если не хватает памяти) случайных изображений для каждого класса в S. При этом Q и S не пересекаются по изображениям.

Тестирование модели заключается в усреднении ее показателей на 1000 случайно сгенерированных заданий из D_{test}. В качестве показателей используются 2 метрики - *точность (accuracy)* и *топ-3 точность (top-3 accuracy)*. Топ-3 точность аналогична обычной точности за исключением того, что считается, что модель дает правильный ответ в том случае, если правильный класс изображения находится среди 3-х наиболее вероятных классов по «мнению» модели (такая метрика может быть полезна в случае, если мы можем предложить пользователю нашей программы вручную выбрать из нескольких классов, например в рекомендательных системах или системах поиска). Для метрик указаны доверительные интервалы, равные утроенному стандартному отклонению.

Следует отметить, что в большинстве работ, посвященных теме Few-Shot Learning, используются стандартные архитектуры сверточных нейронных сетей. В нашем случае была выбрана архитектура *Conv-4*, этот выбор был сделан исходя из доступной оперативной памяти видеокарты (4 GB на Nvidia 1050Ti). Так как выбор данной архитектуры не является предметом исследования, описание Conv-4, а так же архитектур вспомогательных нейросетей, подробное описание всех этих архитектур помещено в **Приложение**.

4. Результаты

5. Вывод

6. Приложение

6.1. Использованные архитектуры нейросетей

6.2. Полные результаты экспериментов

Здесь приведены полные таблицы со всеми результатами экспериментов.

	Rotation task	DFMN	FEAT	Distance	Accuracy	Top-3 accuracy
1	+	-	+	SEN	0.550 ± 0.013	0.894 ± 0.008
2	+	-	+	Euclidean	0.549 ± 0.012	0.889 ± 0.008
3	-	-	+	SEN	0.548 ± 0.012	0.895 ± 0.008
4	-	+	+	SEN	0.548 ± 0.012	0.894 ± 0.008
5	-	+	+	Euclidean	0.547 ± 0.013	0.889 ± 0.008
6	+	-	+	Scaled euclidean	0.546 ± 0.013	0.892 ± 0.008
7	-	-	+	Euclidean	0.546 ± 0.013	0.893 ± 0.008
8	-	-	+	Scaled euclidean	0.544 ± 0.012	0.891 ± 0.008
9	+	+	+	Euclidean	0.544 ± 0.012	0.887 ± 0.008
10	-	+	-	Scaled euclidean	0.542 ± 0.012	0.888 ± 0.008
11	+	+	+	SEN	0.530 ± 0.012	0.883 ± 0.008
12	-	+	+	Scaled euclidean	0.529 ± 0.012	0.881 ± 0.008
13	+	+	+	Scaled euclidean	0.529 ± 0.012	0.873 ± 0.008
14	-	+	-	Euclidean	0.526 ± 0.012	0.881 ± 0.008
15	+	-	-	Scaled euclidean	0.526 ± 0.012	0.878 ± 0.008
16	+	+	-	Scaled euclidean	0.525 ± 0.012	0.877 ± 0.008
17	-	-	-	Scaled euclidean	0.520 ± 0.012	0.875 ± 0.008
18	-	+	-	SEN	0.519 ± 0.012	0.876 ± 0.008
19	+	+	-	SEN	0.510 ± 0.012	0.871 ± 0.008
20	+	-	-	SEN	0.510 ± 0.012	0.877 ± 0.008
21	+	+	-	Euclidean	0.509 ± 0.012	0.869 ± 0.008
22	+	-	-	Euclidean	0.509 ± 0.012	0.870 ± 0.008
23	-	-	-	SEN	0.508 ± 0.012	0.872 ± 0.008
24	-	-	-	Euclidean	0.501 ± 0.012	0.871 ± 0.008

Таблица 6.1. 1-shot FSL на miniImageNet (5-way)

	Rotation task	DFMN	FEAT	Distance	Accuracy	Top-3 accuracy
1	-	+	-	Scaled euclidean	0.703 ± 0.011	0.955 ± 0.004
2	+	-	+	Euclidean	0.700 ± 0.011	0.954 ± 0.005
3	+	-	+	Scaled euclidean	0.700 ± 0.011	0.953 ± 0.005
4	-	+	-	SEN	0.697 ± 0.011	0.950 ± 0.005
5	-	+	-	Euclidean	0.696 ± 0.011	0.951 ± 0.005
6	-	-	+	Euclidean	0.694 ± 0.011	0.950 ± 0.005
7	+	-	+	SEN	0.693 ± 0.011	0.951 ± 0.005
8	-	+	+	Euclidean	0.692 ± 0.011	0.952 ± 0.005
9	-	+	+	SEN	0.690 ± 0.011	0.951 ± 0.005
10	+	+	-	Scaled euclidean	0.690 ± 0.011	0.950 ± 0.005
11	+	-	-	SEN	0.688 ± 0.011	0.947 ± 0.005
12	+	-	-	Euclidean	0.686 ± 0.011	0.947 ± 0.005
13	-	-	+	SEN	0.685 ± 0.011	0.949 ± 0.005
14	-	+	+	Scaled euclidean	0.685 ± 0.011	0.949 ± 0.005
15	+	+	+	SEN	0.684 ± 0.011	0.949 ± 0.005
16	+	+	+	Euclidean	0.683 ± 0.011	0.948 ± 0.005
17	+	+	-	SEN	0.680 ± 0.011	0.947 ± 0.005
18	-	-	-	SEN	0.679 ± 0.011	0.946 ± 0.005
19	+	+	-	Euclidean	0.678 ± 0.011	0.947 ± 0.005
20	+	-	-	Scaled euclidean	0.678 ± 0.011	0.945 ± 0.005
21	-	-	-	Scaled euclidean	0.677 ± 0.011	0.947 ± 0.005
22	-	-	-	Euclidean	0.677 ± 0.011	0.945 ± 0.005
23	-	-	+	Scaled euclidean	0.669 ± 0.012	0.943 ± 0.005
24	+	+	+	Scaled euclidean	0.654 ± 0.011	0.932 ± 0.006

Таблица 6.2. 5-shot FSL на miniImageNet (5-way)

	Rotation task	DFMN	FEAT	Distance	Accuracy	Top-3 accuracy
1	-	+	-	Scaled euclidean	0.298 ± 0.003	0.442 ± 0.003
2	+	+	-	Scaled euclidean	0.296 ± 0.003	0.441 ± 0.003
3	-	+	-	Euclidean	0.295 ± 0.003	0.436 ± 0.003
4	+	+	-	Euclidean	0.291 ± 0.003	0.434 ± 0.003
5	+	+	-	SEN	0.291 ± 0.003	0.433 ± 0.003
6	-	+	-	SEN	0.288 ± 0.003	0.432 ± 0.003
7	+	+	+	Euclidean	0.285 ± 0.003	0.428 ± 0.003
8	+	+	+	SEN	0.285 ± 0.003	0.429 ± 0.003
9	-	+	+	Euclidean	0.285 ± 0.003	0.427 ± 0.003
10	-	+	+	SEN	0.283 ± 0.003	0.427 ± 0.003
11	-	+	+	Scaled euclidean	0.275 ± 0.003	0.415 ± 0.003
12	+	+	+	Scaled euclidean	0.266 ± 0.003	0.406 ± 0.003
13	+	-	-	SEN	0.261 ± 0.003	0.399 ± 0.003
14	+	-	+	Scaled euclidean	0.259 ± 0.003	0.397 ± 0.003
15	+	-	-	Euclidean	0.258 ± 0.003	0.395 ± 0.003
16	+	-	-	Scaled euclidean	0.254 ± 0.003	0.390 ± 0.003
17	+	-	+	SEN	0.253 ± 0.003	0.391 ± 0.003
18	+	-	+	Euclidean	0.251 ± 0.003	0.388 ± 0.003
19	-	-	-	Euclidean	0.250 ± 0.003	0.388 ± 0.003
20	-	-	-	SEN	0.246 ± 0.003	0.382 ± 0.003
21	-	-	+	Euclidean	0.235 ± 0.003	0.371 ± 0.003
22	-	-	+	SEN	0.233 ± 0.003	0.367 ± 0.003
23	-	-	-	Scaled euclidean	0.221 ± 0.002	0.351 ± 0.003
24	-	-	+	Scaled euclidean	0.215 ± 0.002	0.347 ± 0.003

Таблица 6.3. 5-shot FSL; обучен на miniImageNet, протестирован на miniGoogleLandmarks (100-way)

	Rotation task	DFMN	FEAT	Distance	Accuracy	Top-3 accuracy
1	+	-	+	Euclidean	0.562 ± 0.011	0.897 ± 0.007
2	-	-	+	Euclidean	0.562 ± 0.011	0.896 ± 0.007
3	+	-	-	SEN	0.561 ± 0.012	0.897 ± 0.007
4	-	-	-	Euclidean	0.561 ± 0.011	0.895 ± 0.007
5	+	+	-	SEN	0.558 ± 0.012	0.894 ± 0.007
6	-	+	-	Euclidean	0.557 ± 0.012	0.891 ± 0.007
7	+	-	-	Euclidean	0.557 ± 0.012	0.898 ± 0.007
8	+	-	+	SEN	0.556 ± 0.011	0.891 ± 0.007
9	+	+	-	Euclidean	0.556 ± 0.012	0.890 ± 0.007
10	-	+	-	SEN	0.555 ± 0.011	0.892 ± 0.007
11	+	-	+	Scaled euclidean	0.555 ± 0.011	0.892 ± 0.007
12	+	-	-	Scaled euclidean	0.555 ± 0.011	0.892 ± 0.007
13	-	-	+	SEN	0.554 ± 0.012	0.894 ± 0.007
14	+	+	-	Scaled euclidean	0.550 ± 0.012	0.890 ± 0.007
15	+	+	+	Euclidean	0.550 ± 0.012	0.888 ± 0.007
16	-	-	-	SEN	0.549 ± 0.011	0.889 ± 0.007
17	-	+	+	SEN	0.546 ± 0.011	0.885 ± 0.007
18	-	-	+	Scaled euclidean	0.545 ± 0.011	0.890 ± 0.007
19	+	+	+	SEN	0.544 ± 0.012	0.885 ± 0.007
20	+	+	+	Scaled euclidean	0.541 ± 0.012	0.885 ± 0.007
21	-	+	-	Scaled euclidean	0.541 ± 0.012	0.888 ± 0.007
22	-	+	+	Euclidean	0.541 ± 0.011	0.884 ± 0.007
23	-	+	+	Scaled euclidean	0.541 ± 0.011	0.885 ± 0.007
24	-	-	-	Scaled euclidean	0.536 ± 0.011	0.881 ± 0.007

Таблица 6.4. 5-shot FSL; обучен на miniGoogleLandmarks, протестирован на miniImageNet (5-way)

	Rotation task	DFMN	FEAT	Distance	Accuracy	Top-3 accuracy
1	-	-	+	Euclidean	0.403 ± 0.003	0.565 ± 0.003
2	-	+	-	Euclidean	0.398 ± 0.003	0.555 ± 0.003
3	+	-	+	Euclidean	0.398 ± 0.003	0.559 ± 0.003
4	+	-	+	SEN	0.397 ± 0.003	0.558 ± 0.003
5	-	-	-	SEN	0.395 ± 0.003	0.556 ± 0.003
6	-	-	+	SEN	0.395 ± 0.003	0.558 ± 0.003
7	+	-	-	Euclidean	0.392 ± 0.003	0.551 ± 0.003
8	+	-	-	SEN	0.390 ± 0.003	0.549 ± 0.003
9	-	-	-	Euclidean	0.390 ± 0.003	0.549 ± 0.003
10	+	-	+	Scaled euclidean	0.389 ± 0.003	0.546 ± 0.003
11	-	+	-	SEN	0.389 ± 0.003	0.549 ± 0.003
12	+	+	-	SEN	0.386 ± 0.003	0.544 ± 0.003
13	+	+	-	Euclidean	0.386 ± 0.003	0.540 ± 0.003
14	-	-	+	Scaled euclidean	0.385 ± 0.003	0.544 ± 0.003
15	+	-	-	Scaled euclidean	0.382 ± 0.003	0.539 ± 0.003
16	-	-	-	Scaled euclidean	0.376 ± 0.003	0.532 ± 0.003
17	+	+	+	Euclidean	0.376 ± 0.003	0.537 ± 0.003
18	-	+	-	Scaled euclidean	0.375 ± 0.003	0.537 ± 0.003
19	-	+	+	SEN	0.372 ± 0.003	0.534 ± 0.003
20	-	+	+	Scaled euclidean	0.372 ± 0.003	0.533 ± 0.003
21	-	+	+	Euclidean	0.372 ± 0.003	0.533 ± 0.003
22	+	+	-	Scaled euclidean	0.369 ± 0.003	0.530 ± 0.003
23	+	+	+	SEN	0.365 ± 0.003	0.526 ± 0.003
24	+	+	+	Scaled euclidean	0.362 ± 0.003	0.523 ± 0.003

Таблица 6.5. 5-shot FSL на miniGoogleLandmarks (100-way)