

Аннотация

Целью работы было оценить способность моделей, схожих по архитектуре с нейросетью Трансформер, к мультимодальному предобучению при решении задачи распознавания действий в видео инструкциях (Action Segmentation). Требовалось сравнить одномодальную модель, обученную с нуля, с видео-частью предобученной мультимодальной модели. В качестве целевого был выбран датасет 50salads, на нём было разрешено дообучение (finetune) видео-части мультимодальной модели.

Как одномодальный baseline использовалась модель ASFormer. Модель была обучена с нуля и протестирована с применением кросс-валидации. Результаты отставали от авторских, но послужили сильным baseline для дальнейших исследований. В качестве мультимодальной модели был выбран VideoCLIP. После мультимодального предобучения и одномодального дообучения на целевом датасете удалось получить модель, которая превзошла baseline по качеству. Прирост качества по отношению к baseline оказался не слишком большим, поэтому для сравнения были проведены дополнительные замеры с использованием публично доступных весов модели. Код обучения и измерения качества можно найти в репозитории https://github.com/mira318/masters_degree

По итогам работы мультимодальное предобучение способно улучшить качество решения задачи Action Segmentation. Метод следует применять с осторожностью, расчитывая имеющиеся ресурсы. Дообучение общедоступных весов модели на целевом, специфическом для конкретной области, датасете может дать схожие результаты.

Содержание

1 Введение	4
2 Основная часть	6
2.1 Датасеты	7
2.1.1 Целевой датасет: 50salads	7
2.1.2 Датасет для мультиомодального предобучения: HowTo100M	8
2.2 Описание baseline	10
2.2.1 Сеть для извлечения векторов признаков: I3D	10
2.2.2 Основная модель: ASFormer	12
2.3 Результаты на baseline	14
2.3.1 Обучение на публично доступных векторах признаков	14
2.3.2 Самостоятельное получение векторов признаков датасета 50salads из видео	17
2.3.3 Обучение на самостоятельно извлечённых векторах признаков	20
2.4 Описание модели для мультиомодального pretrain	23
2.4.1 Сеть для извлечения векторов признаков: S3D	24
2.4.2 Основная мультиомодальная модель: VideoCLIP	26
2.5 Мультиомодальное предобучение на одном GPU	28
2.5.1 Возможные причины неудавшегося эксперимента	30
2.6 Мультиомодальное предобучение на семи GPU	31
2.7 Выделение видео-части мультиомодальной модели	32
2.8 Эксперименты без дообучения	34
2.8.1 Zero-shot перенос модели с самостоятельно обученными весами	34
2.8.2 Zero-shot перенос общедоступной модели	36
2.8.3 Анализ ошибок	38
2.9 Эксперименты по дообучению моделей	40
2.9.1 Дообучение модели с собственными весами	40
2.9.2 Дообучение модели с общедоступными весами	47
3 Заключение	54

1 Введение

Изначально нейронная сеть Трансформер была разработана для решения задачи машинного перевода. Первая и самая известная статья о ней "Attention Is All You Need" [1] была опубликована 7 лет назад. Модель оказалась настолько успешной, что на неё обратили внимание исследователи из смежных областей. Они попытались решить уже свои задачи машинного обучения, используя похожие архитектуры. Таким образом появилось множество моделей (и публикаций о них), которые используют идеи из Трансформер. Например, BERT [2], который адаптирован для проверки на перефразирование, проверки причинно-следственных связей в тексте, поиска ответа на вопрос в тексте, классификации и разбиения текстов по тегам. Для классификации изображений по типам используют ViT [3] и Convolution vision transformer [4], а для их генерации - Image Transformer [5]

То, что указанные модели успешно справляются со своими задачами, говорит о поразительной способности основных компонентов Трансформер, attention-блоков, к адаптации во многих сферах. Безусловно, архитектура Трансформеров может быть адаптирована и для задач, связанных с видео, но это требует больше усилий. Дело в том, что видео обычно обрабатывается по кадрам или группам соседних кадров. Таким образом из одного сравнительно небольшого видео-фрагмента получается огромное количество векторов признаков (эмбеддингов). Attention-блоки в свою очередь требуют перемножения входных векторов, что приводит к квадратичной сложности. Это легко увидеть из формулы 1 для одного блока.

$$\text{Attention}(Q, K, V) = \text{Softmax} \left(\frac{Q \cdot K^T}{\sqrt{d}} \right) \cdot V \quad (1)$$

где Q, K, V - векторы признаков query, key, value, получаются из входного вектора (или векторов в случае cross-attention) признаков домножением на обучаемые матрицы d - размерность, одинаковая для векторов Q и K

Ещё одной проблемой является необходимость учитывать положение кадров во времени. Это легко понять на примере двух различных действий: открытия и закрытия двери. С точки зрения картинки различается лишь последовательность кадров, но хотелось бы чтобы модели умели отличать эти действия друг от друга.

На данный момент ведущие исследователи предлагают разнообразные подходы к решению возникающих проблем. Например, можно использовать attention в пределах небольших фиксированных окон, а далее агрегировать полученную информацию как в Swin [6];

изменить операцию внутри attention-блока: пересчитывать attention лишь по 3м осям (при этом важно их сохранять) как в TimeSformer [7] или разделить входные эмбеддинги на независимо обрабатываемые группы, что было сделано в [8]. Во всём этом разнообразии можно найти и идею предобучения [9], однако обычно одномодального или использующего в качестве дополнительной модальности изображения.

Переходя к мультимодальным моделям стоит отметить возникающую дополнительную сложность: необходимо передавать информацию между модальностями. Самая простая идея предложена в первой успешной мультимодальной модели VideoBERT [10], здесь эмбеддинги из модальностей текста и видео просто сконкатенировали перед подачей в общий BERT [2], правда длина последовательностей токенизированного текста и кадров из видео была не постоянной и это мешало модели. Другой подход к совмещению модальностей предложили авторы ActBert [11]. Они использовали 3 отдельных BERT [2] для каждой из модальностей, а для связи добавили 2 дополнительных multi-head cross-attention между слоями. Ещё один способ продемонстрирован в STALE [12]. Модель использует декодер из оригинального Трансформер [1] и переходит в латентное пространство, где пытается сопоставить видео-запросу двоичную маску сегмента и дообучить извлечение векторов признаков из текста.

Из всего выше сказанного легко понять, что Трансформеры на данный момент активно развиваются и проникают почти во все области машинного обучения. Существует уже более сотни моделей на их основе. При этом использование Трансформеров для видео всё ещё остаётся достаточно сложной задачей. Идея данной работы - попробовать преодолеть возникающие проблемы с помощью мультимодального предобучения конкретно в задаче Action Segmentation для видео.

Безусловно, задача мультимодального предобучения сама по себе сложная и требует отдельного внимания. Более того: авторы мультимодальных моделей склонны рассматривать переход к одной видео модальности лишь как дополнительное приложение своих результатов и редко проводят сравнение качества с одномодальными моделями. В данной работе было решено действовать иначе. Был выбран конкретный достаточно специфический целевой датасет и поставлена задача Action Segmentation для видео. Далее были выбраны одномодальная и мультимодальная модели. В случае мультимодальной модели потребовалось выделить часть, ответственную именно за обработку видеоряда. После были предприняты попытки решить задачу несколькими способами:

- С помощью сильного одномодального baseline
 - используя авторские векторы признаков
 - используя самостоятельно извлечённые из видео векторы признаков
- С помощью видео-части мультимодальной модели
 - используя публично доступные веса параметров модели
 - обучив свою собственную версию модели

Все замеры качества проходили по сплитам с кросс-валидацией. В качестве исходных данных обе модели использовали эмбеддинги кадров из видео, извлечённые отдельными дополнительными нейросетями (backbone).

Для исследования использовалось отдельное conda-окружение, включающее язык python версии 3.10, библиотеки pytorch, opencv, для мультимодальной модели потребовалось так же использовать библиотеки fairseq [13] и faiss [14].

Одномодальная модель была обучена и измерена на одном GPU NVIDIA TITAN V с использованием операционной системы linux, для мультимодальной модели использовались NVIDIA TITAN RTX и NVIDIA QUADRO RTX 6000 для мультимодального предобучения на linux, а так же NVIDIA A40 для измерений и мультимодального предобучения на windows.

В результате было получено подтверждение гипотезы об улучшении качества за счёт использования мультимодального предобучения. Тем не менее полученные метрики не сильно отличаются от baseline. Это говорит о необходимости исследования доступных ресурсов перед применением метода. Возможным альтернативным решением будет использование общедоступной предобученной модели.

2 Основная часть

2.1 Датасеты

В исследовании используется 2 датасета: целевой датасет 50salads и датасет для мультимодального предобучения HowTo100M. На целевом датасете проводились все окончательные замеры, на базе которых делался окончательный вывод о применимости метода.

На датасете для мультимодального предобучения была предобучена своя версия мультимодальной модели. Известно, что именно этот датасет использовался авторами для обучения общедоступной предобученной версии модели. Разбиения на тренировочную и валидационную части в обоих случаях были случайными и, вероятно, различаются. Так же датасет использовался для предобучения одной из backbone-сетей для извлечения эмбеддингов видео.

2.1.1 Целевой датасет: 50salads

Датасет впервые описанный в статье [15] состоит из 50 видео, на которых 25 человек готовят 2 различных салата. Участникам не было дано конкретного рецепта, лишь список ингредиентов, таким образом авторы предотвратили заучивание нейросетью конкретных последовательностей действий (рецептов), вместо решения задачи их распознавания.

Датасет содержит около 4.5 часов видео от первого лица качества 30 fps с подробной посекундной разметкой классов действий, а так же покадровые карты глубины и данные акселерометров, прикреплённых к ножу, ложке, овощечистке, стакану и перечнице. В разметке задействованы 17 классов действий и 2 технических класса: action_start и action_end. Пример данных можно увидеть на рисунке 1

В силу небольшого размера и во избежание возможного переобучения, авторы разбили датасет на 5 различных сплитов для кросс-валидации по 40 видео для обучения и 10 видео для проверки результатов. Измерение качества всех описанных далее моделей использует это разбиение. Модели обучаются (дообучаются в случае finetune) на 40 тренировочных видео, а потом качество оценивается на оставшихся 10. Это действие повторяется на каждом из 5ти сплитов. Итоговая оценка качества получается усреднением.

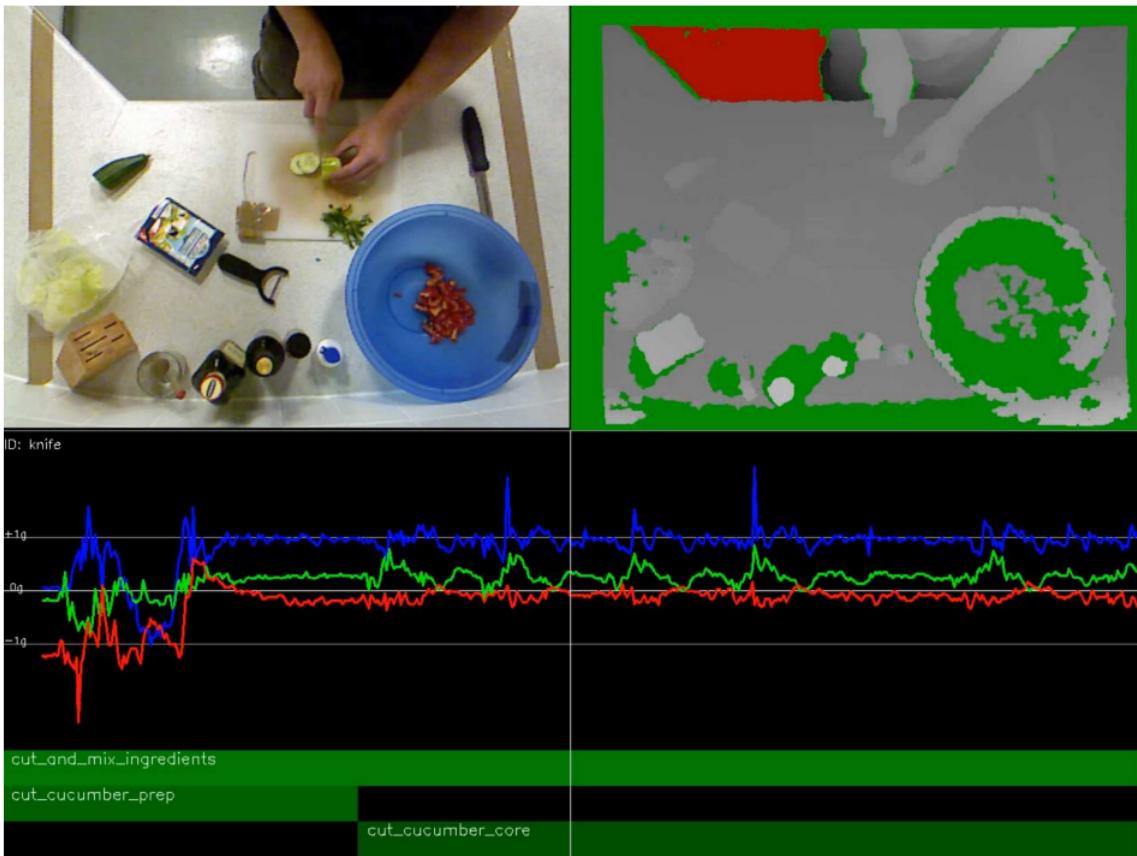


Рисунок 1: Пример данных из датасета 50salads

2.1.2 Датасет для мультимодального предобучения: HowTo100M

Крупный датасет, собранный с помощью YouTube. Для получения данных использовались запросы в YouTube вида "how to ...", где вместо многоточия указывалось конкретное действие. Авторы отдавали предпочтение действиям, которые легко понять визуально, например покраске, очистке, приготовлению пищи и т. д. Они отбирали видео по числу просмотров и наличию английских субтитров, а так же избавлялись от дубликатов. Полную и подробную процедуру можно найти в оригинальной статье [16].

После всех преобразований было отобрано 1.2 М видео, содержащих 136.6 М клипов с различными действиями. Действия были разбиты на 12 больших доменов (категорий 1-го уровня) и 129 подкатегорий (категорий 2-го уровня). В общей сложности в датасете присутствует 23611 классов действий, суммарная продолжительность которых составляет около 15 лет. При столь большом объеме не удивительно, что положение наблюдателя варьирует: есть как видео от первого лица, так и видео снятые из фиксированной точки. Примеры можно увидеть на рисунке 2

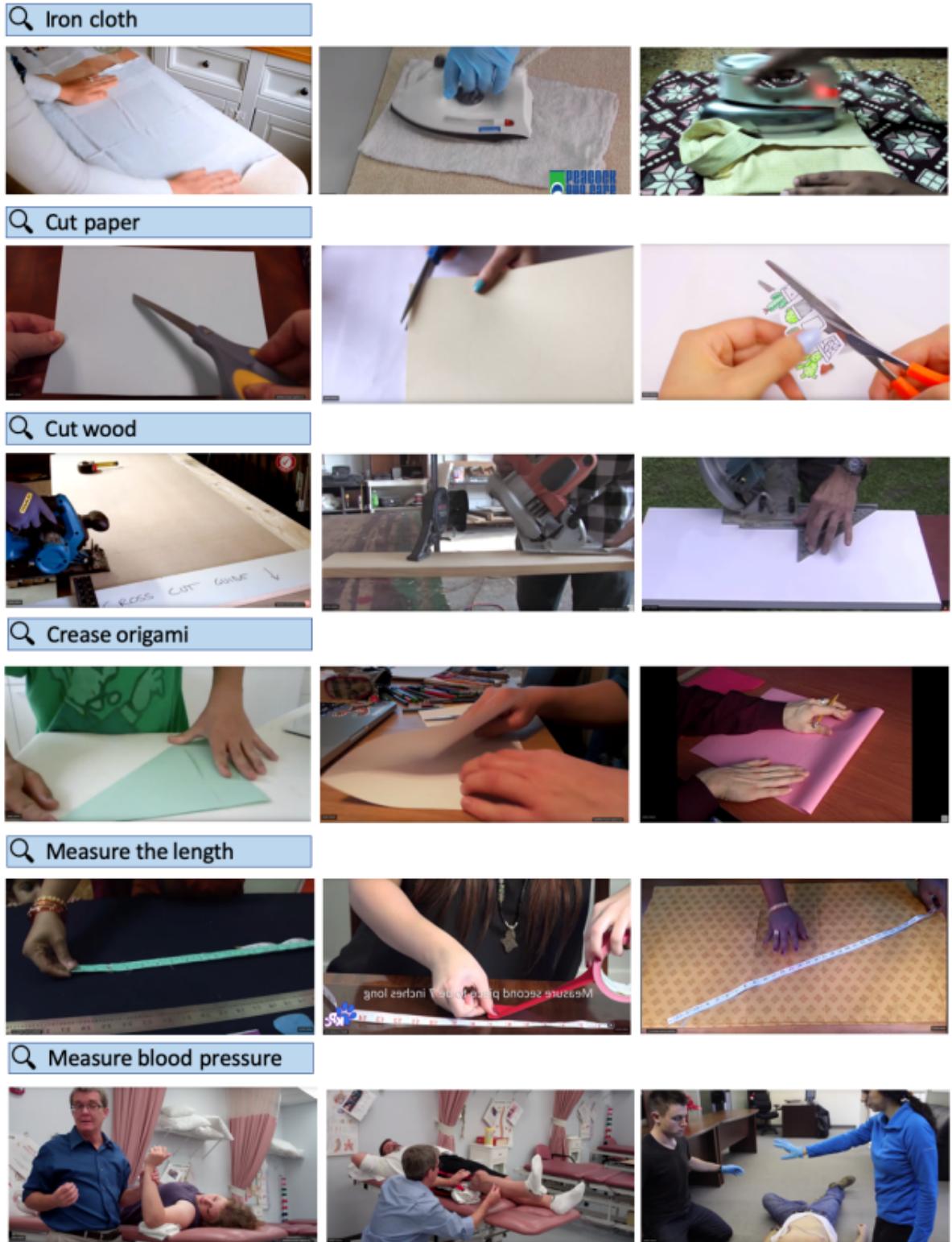


Рисунок 2: Примеры данных из датасета HowTo100M

Даже после тщательного отбора видео датасет остаётся зашумлённым. Тут есть 2 основные причины. Во-первых, люди часто объясняют свои действия до или после их демонстрации, что приводит к сдвигам субтитров относительно картинки. Более того, объяснения

простых с точки зрения человека действий могут быть опущены или заменены фразами вида "делайте, как показано далее". Во-вторых, возникает дополнительный шум из-за источника информации. Даже самые замечательные авторы, которые стремятся передать своей аудитории некоторые знания, часто просят подписаться, поставить лайк и включить уведомления в конце своих видео. К тому же существуют видео, которые создаются исключительно для набора просмотров и не содержат демонстрации нужного действия. В случае последних остаётся надеяться на сложную процедуру отбора, разработанную авторами датасета.

2.2 Описание baseline

В качестве одномодального baseline была выбрана модель ASFormer из [17]. Архитектура была предложена менее 3х лет назад и на момент своего появления стала одномодальной SOTA для целевого датасета. Она даже соревновалась с некоторыми мультимодальными подходами. Модель до сих пор удерживает достаточно высокую позицию на датасете 50salads, особенно при рассмотрении только модальности видео. Более того: многие новые, превзошедшие ASFormer, модели используют данную архитектуру как backbone или одномодальную подсеть для модальности видео.

Входными данными для ASFormer служат векторы, полученные из другой нейросети: I3D [18]. Т. е. у данной одномодальной модели существует свой собственный backbone для извлечения фичей. Далее части модели будут рассмотрены подробнее с указанием особенностей.

2.2.1 Сеть для извлечения векторов признаков: I3D

Авторы ASFormer выбрали I3D из статьи [18] как экстрактор векторов признаков из видео. Это было сделано вполне осознанно, поскольку данная нейросеть агрегирует векторы признаков по времени, а значит передаёт больше информации, чем любой двумерный экстрактор.

Модель была получена из нейросети Inception-v1 [19] для изображений путём "раздувания" весов. Т. е. авторы статьи [18] взяли уже предобученную на датасете ImageNet [20] версию модели и продублировали веса двумерных свёрток несколько раз, тем самым сделав их трёхмерными. При этом авторы не всегда делали из квадратных фильтров кубические, они использовали единичные фильтры, а так же $\text{stride} = 1$ в пулингах по размерности времени в начальных блоках нейросети. Интуитивно, это должно было помочь не потерять изменение положения объектов во времени слишком быстро. Получившуюся трёхмерную

архитектуру можно увидеть на рисунке 3

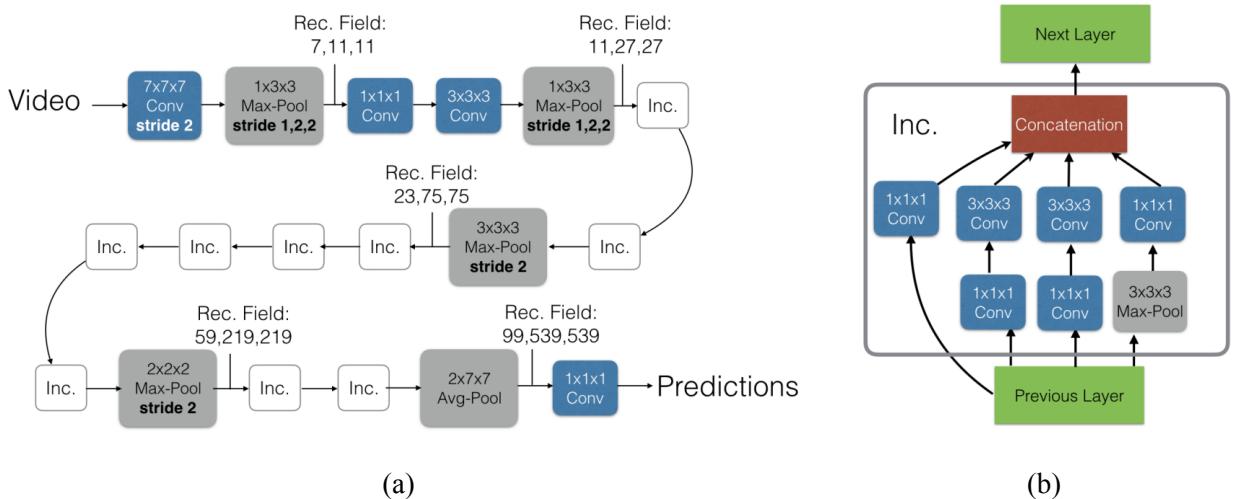


Рисунок 3: а) Общий вид архитектуры нейросети I3D б) Архитектура одного ”раздутого“ inception-блока

Описанная выше нейросеть дублируется. Таким образом получается 2 подсети: для кадров видео и для входного оптического потока. Их веса инициализируются одинаково, но обучаются по отдельности. Иллюстрацию полной архитектуры можно увидеть на рисунке 4. Итоговые векторы признаков получаются усреднением предсказаний 2x сетей.

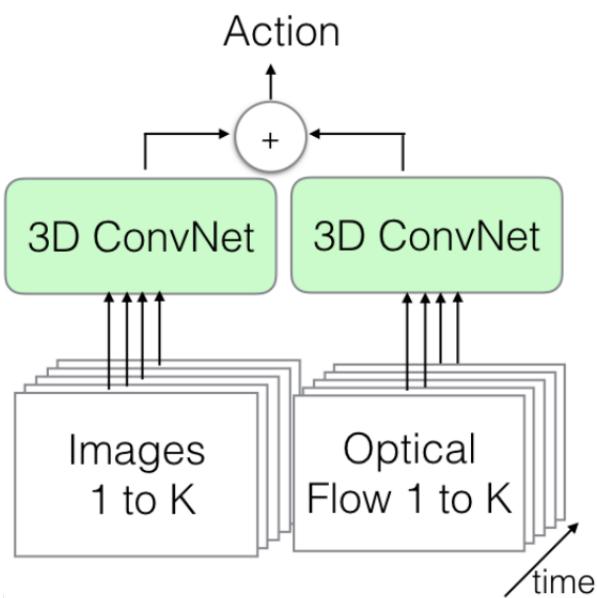


Рисунок 4: Конечная двупотоковая архитектура, содержит 25M параметров

2.2.2 Основная модель: ASFormer

ASFormer состоит из одного энкодера и нескольких декодеров, что можно увидеть на рисунке 5(b). В отличие от многих предшественников, модель использует декодер из оригинального Трансформер [1]. Более того, это делается несколько раз для уточнений предсказаний энкодера. Далее подробнее рассмотрим каждую из частей модели.

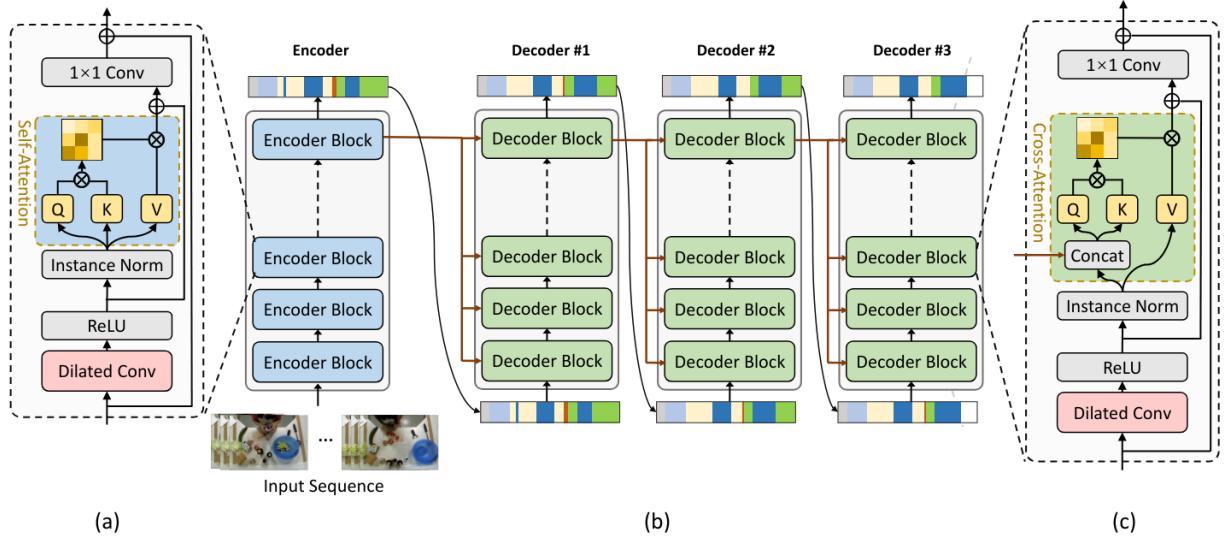


Рисунок 5: Архитектура ASFormer: а) Блок энкодера б) Архитектура целиком: 1 энкодер и 3 декодера с) Блок декодера

Изменения внутри **энкодера**, один блок которого можно увидеть на рисунке 5(a, существенны и помогают решить проблемы, упомянутые во введении. Авторы ASFormer переставили местами подблоки внутри блока: сначала идёт feed forward блок, а потом блок с self-attention. Вокруг этих подблоков, как и в классическом Трансформер, используются residual connections, правда сразу после них вместо обычной нормализации по батчу применяется instance norm. Т. е. нормализация происходит с учётом времени: для каждого момента считаются свои среднее и дисперсия.

Внутри feed forward и self-attention блоков так же были сделаны изменения. В feed forward блок вместо обычных линейных слоёв была добавлена свёртка по времени с dilation. В целом идея свёрток внутри feed forward блоков уже встречалась в статьях об обработке изображений, например в [4], но в данном случае свёртка происходит именно по времени - измерению, которое есть только у видео. Dilation свёртки растёт с номером блока энкодера: $2, 4, 8 \dots 2^i$ для блока i . Поскольку в энкодере всего 9 больших блоков из feed forward и attention подблоков, максимальным будет $dilation = 2^9 = 512$. Данный иерархический пат-

терн был так же использован и внутри self-attention блока. Как было указано в формуле 1, для расчёта attention используются 3 вектора: query, key, value, но здесь они ограничены окном 2^i при расчёте для слоя i . Т. е. вектор входных фичей намеренно обрезают, а уже после домножают на обучаемые матрицы и получают query, key и value. Таким образом удаётся преодолеть квадратичную сложность вычислений, создаваемую attention.

Идея иерархического паттерна исходит из комбинации двумерных свёрток и пулингов, которая часто используется при обработке изображений. Можно заметить, что при такой обработке receptive field увеличивается с ростом глубины сети, следуя некоторому схожему паттерну.

Оригинальный **декодер** тоже был серьёзно переработан, что можно увидеть на рисунке 5(c). Стоит сразу отметить, что декодеров стало 3 (рисунок 5(b). При этом первые предсказания действий получаются уже на выходе из энкодера, а далее ASFormer постепенно их уточняет.

Многие изменения в декодерах совпадают с изменениями из энкодера: feed forward подблок стоит перед подблоком с attention; вокруг feed forward и attention подблоков используется residual connection с последующим instance norm; feed forward блок представляет из себя свёртку по времени с иерархическим паттерном для dilation; внутри attention блока так же использованы окна с увеличивающимся согласно паттерну размером. Размеры декодеров тоже совпадают с размером энкодера: комбинация feed forward и attention блоков повторяется 9 раз.

Важные отличия видны в attention-блоке. Он представлен cross-attention, который использует выходной вектор из предыдущей части сети: энкодера или предыдущего декодера. Выходной вектор конкатенируется с выходом текущего feed forward блока внутри слоя, а затем домножается на обучаемые матрицы W_Q и W_K для получения query и key. Value использует только выход из feed forward блока и свою обучаемую матрицу W_V . Таким образом предыдущая часть сети (энкодер или декодер) может повлиять на входной вектор предсказаний V только с помощью линейных комбинаций. Так же, чтобы не испортить предсказания энкодера и предотвратить накопление ошибок, авторы архитектуры уменьшают вес cross-attention в каждом из последующих декодеров по формуле 2.

$$\begin{aligned} \text{out} &= \text{feed_forward}(x) \\ \text{out} &= \alpha \cdot \text{cross_attention}(\text{out}) + \text{out} \end{aligned} \tag{2}$$

где $\alpha = 1$ для первого декодера, а далее α уменьшается в экспоненциальное число раз для каждого следующего декодера.

Поскольку модель делает предсказания целых 4 раза (1 раз на выходе из энкодера и 3 раза на выходе из декодера), её функция потерь тоже изменилась: это сумма 4x функции потерь из формулы 3.

$$L = L_{cls} + \lambda L_{smo} = -\frac{1}{T} \sum_{t=1}^T \log(y_{t,c}) + \frac{\lambda}{TC} \sum_{t=1}^T \sum_{c=1}^C (y_{t-1,c} - y_{t,c})^2 \quad (3)$$

где L_{cls} - функция потерь для классов действий, представлена кросс-энтропией

L_{smo} - сглаживающая по времени квадратичная функция потерь

λ - коэффициент для балансировки функций потерь, во всех экспериментах описанных далее использовалась $\lambda = 0.25$

T - длительность видео в кадрах,

C - число классов действий,

$y_{t,c}$ - предсказанная вероятность класса c в момент времени t ,

\hat{c} - правильный класс действия, groundTruth в выбранный момент времени

Т. е. каждая из частей модели представлена отдельным слагаемым вида 3, что даёт некоторую независимость в момент подсчёта функции потерь, но при этом связь между частями модели сохраняется, благодаря cross-attention. Функция потерь для одной части достаточно простая и представляет собой комбинацию кросс-энтропии и сглаживания по времени с коэффициентом λ .

2.3 Результаты на baseline

Для получения сильного baseline необходимо обучить модель, которая смогла бы достаточно близко воспроизвести результат авторов на датасете 50salads. При этом обученная модель должна работать не только на каком-то ограниченном наборе видео, т. к. она может в последствии оказаться лучшей. Значит, нужно уметь не только обучать сам ASFormer, но и самостоятельно извлекать векторы признаков из любого видео.

2.3.1 Обучение на публично доступных векторах признаков

Для первого эксперимента воспользуемся публично доступными векторами признаков (покадровыми эмбеддингами), которые предлагают авторы модели. В статье [17] утверждается, что они получены с помощью I3D, предобученной на датасете Kinetics из статьи [21]. Графики обучения можно увидеть на рисунках 6 (train) и 7 (test), а итоговые метрики качества в таблице 1.

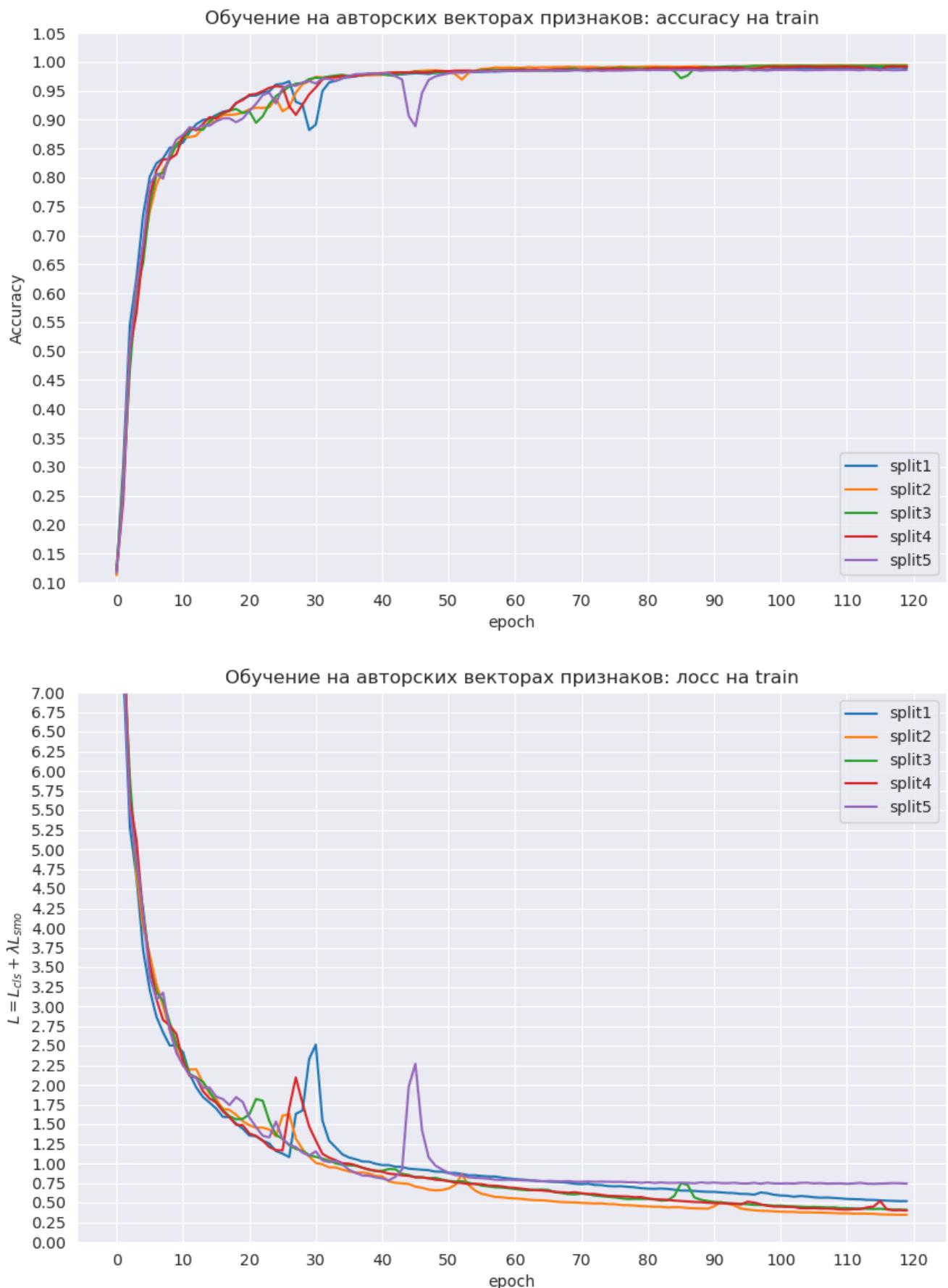


Рисунок 6: Обучение на публично доступных векторах признаков



Рисунок 7: Результаты на публично доступных векторах признаков

Таблица 1: Результаты на публично доступных векторах признаков

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучшая accuracy	эпоха с лучшей accuracy
1	0.809	75.4	0.807	0.794	0.700	0.809	120
2	0.885	74.6	0.816	0.794	0.739	0.893	100
3	0.849	78.9	0.840	0.835	0.769	0.858	110
4	0.862	80.1	0.859	0.835	0.772	0.866	100
5	0.865	80.0	0.866	0.856	0.761	0.865	110
среднее	0.854	77.8	0.838	0.823	0.748	0.858	-

Полученное качество после усреднения по сплитам соответствует метрикам из оригинальной статьи:

$$accuracy = 0.856 \quad edit.dist. = 79.6 \quad f1@10 = 0.851 \quad f1@25 = 0.834 \quad f1@50 = 0.760$$

Такие результаты говорят об удачном исходе эксперимента. Стоит всё же упомянуть несколько нестандартное обучение первого сплита. Скорее всего это связано с повреждениями в видео, с которыми пришлось столкнуться при дальнейших экспериментах с датасетом

50salads. Далее попытаемся воспроизвести весь pipeline обучения, для этого требуется самостоятельно получить покадровые эмбеддинги из видео.

2.3.2 Самостоятельное получение векторов признаков датасета 50salads из видео

Чтобы получить покадровые эмбеддинги из видео, нужна предобученная модель I3D. К несчастью, оригинальные веса модели, которые использовались для создания авторские векторов признаков, найти не удалось.

Уже достаточно давно в статьях, просвещённых задаче Action Segmentation, используют 3 небольших датасета: breakfast, GTEA и 50salads, пвекторы признаков из которых лежат по общей ссылке. Обычно авторы используют все данные и замеряют качество моделей на всех 3х датасетах сразу. В статье об ASFormer упоминаются лишь несколько предшествующих статей, использующих схожие данные. Самая ранняя из этих статей, которая использует признаки из I3D для всех 3х датасетов, рассказывает о модели MS-TCN [22] и не даёт дополнительных разъяснений о происхождении покадровых эмбеддингов. Последующие статьи попросту ссылаются на неё, поскольку в репозитории с имплементацией модели можно найти прямую ссылку на данные.

Несмотря на возникшую проблему, решить задачу всё же возможно: ASFormer обучается с нуля, поэтому изменение чекпоинта для извлечения векторов признаков, при соблюдении процедуры его подготовки, не должно сильно сказываться на качестве модели. Общедоступные веса модели I3D с такой же процедурой обучения есть например в списке моделей [model_zoo](#) библиотеки pytorchvideo [23].

При использовании I3D выяснилось, что некоторые видео из целевого датасета повреждены. Примеры повреждённых кадров можно увидеть на иллюстрации 8. Заметим, что все повреждённые видео попадают в данные для обучения сплита 1 и именно на нём обычно получается худшее качество. Единственным способом прочитать все файлы и не потерять кадры, стоящие после повреждений, оказалось использование VideoCapture из библиотеки opencv. При дальнейшей обработке будем пропускать Повреждённые кадры.

После чтения необходимо пройтись по видео окном размером в 21 кадр и пропустить полученные отрезки видео через модель I3D, как это сделали авторы ASFormer. Таким образом для каждого кадра получается вектор признаков, который равен выходу сети при запуске на отрезке, в котором этот кадр находится строго посередине. Для кадров в начале и в конце видео используется паддинг из нулей.

Результатом всех описанных операций являются самостоятельно извлечённые из ви-



rgv-14-1.avi



rgb-18-1.avi

Рисунок 8: Повреждённые видео в целевом датасете

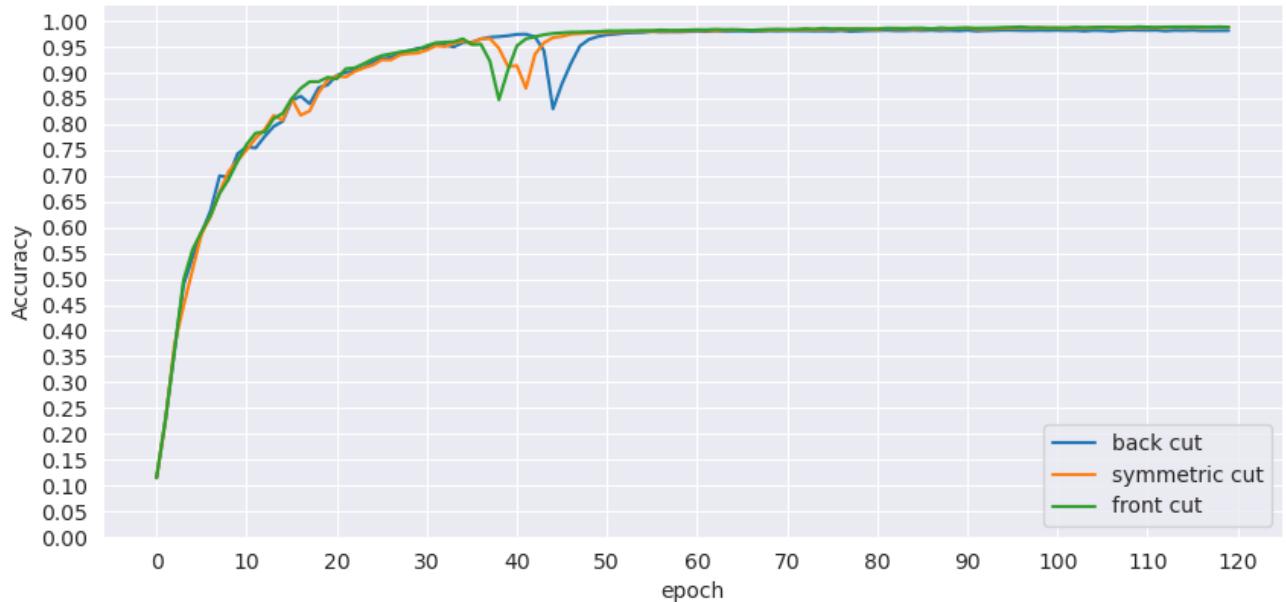
део покадровые эмбеддинги. Их размерность по времени всегда строго на 8 больше, чем у оригинальных покадровых эмбеддингов. Эта цифра не зависит от номера видео и от того, были ли в нём повреждённые кадры. Данный факт означает, что выбранная процедура извлечения покадровых эмбеддингов достаточно близка к авторской, но необходим дополнительный эксперимент для выбора правильного способа обрезки.

Для проведения эксперимента по обрезке векторов признаков зафиксируем первый сплит и обучим 3 тестовые модели с 3мя видами обрезки:

- с начала: удаляем первые 8 векторов признаков
- посередине (симметрично): удаляем по 4 вектора в начале и конце видео
- с конца: удаляем 8 векторов в конце видео

Графики обучения моделей можно увидеть на рисунке 9, а результаты на тестовой части данных на рисунке 10. Для удобства данные сведены в таблицу 2

Эксперимент с обрезкой: accuracy на train



Эксперимент с обрезкой: лосс на train

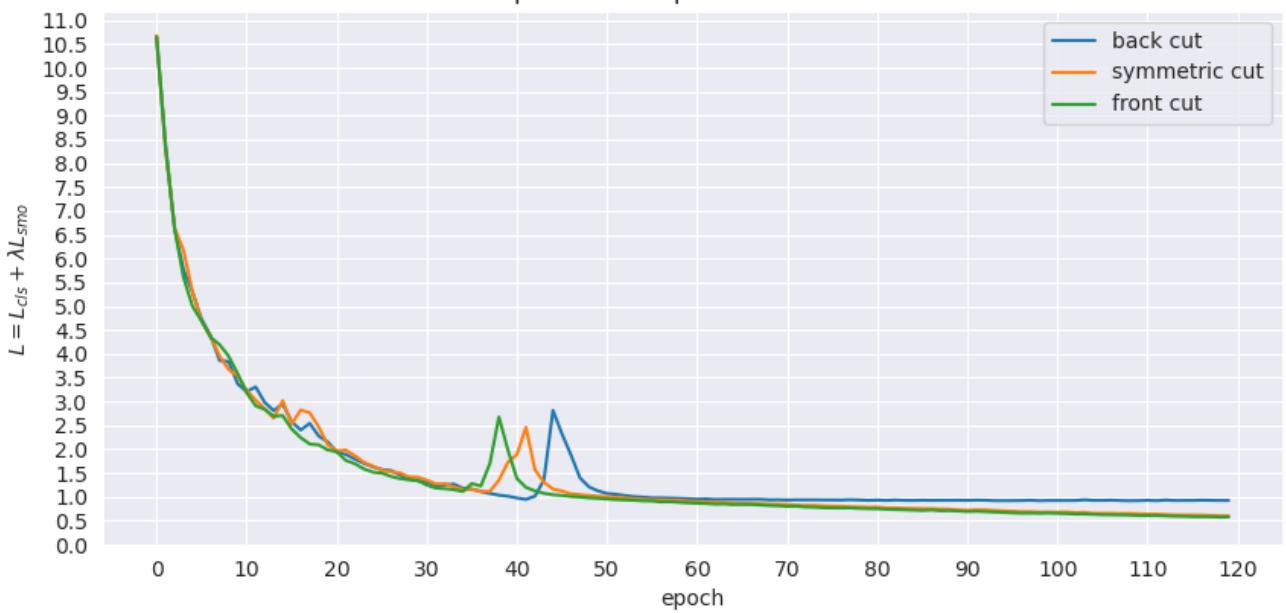


Рисунок 9: Обучение в эксперименте с обрезкой векторов признаков

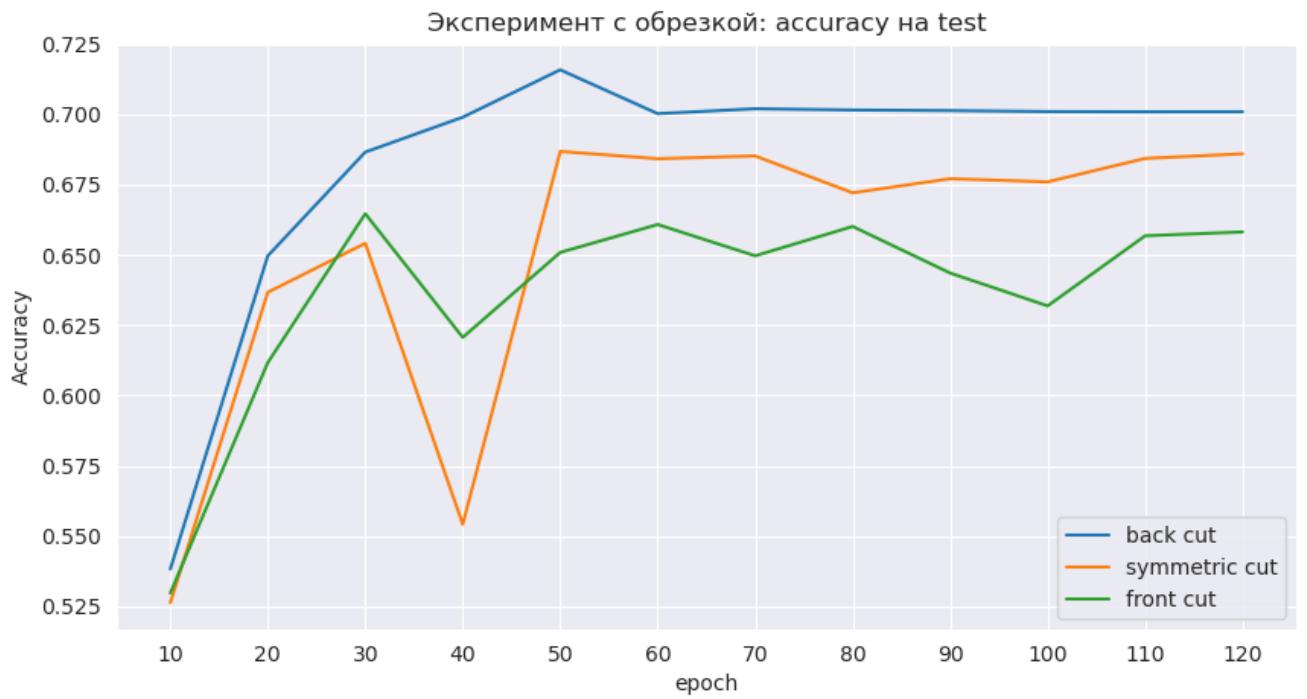


Рисунок 10: Результаты на тестовой части данных в эксперименте с обрезкой векторов признаков

Таблица 2: Эксперимент с обрезкой собственных векторов признаков - результаты

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучшая accuracy	эпоха с лучшей accuracy
с начала	0.658	62.0	0.681	0.651	0.527	0.665	30
посередине	0.686	63.5	0.702	0.667	0.566	0.687	50
с конца	0.701	65.9	0.725	0.698	0.586	0.716	50

Обрезка с конца более стабильная и даёт лучший результат, поэтому в экспериментах далее будем использовать её. На графике 10 можно отчётливо увидеть, что модель быстро усваивает информацию во всех 3х случаях, но чем дальше от верного способа обрезки среза, тем хуже получается качество. Вторая половина обучения нужна скорее для fine-tune: здесь декодеры учатся уточнять предсказания сегментов с действиями. Важно в процессе не получить сильной деградации модели.

2.3.3 Обучение на самостоятельно извлечённых векторах признаков

Наконец, перейдём к обучению модели на полученных покадровых эмбеддингах. Проведём обучение по всеми 5ти сплитам с целью дальнейшей кросс-валидации. Графики обу-

чения представлены на рисунках 11 (train) и 12 (test).

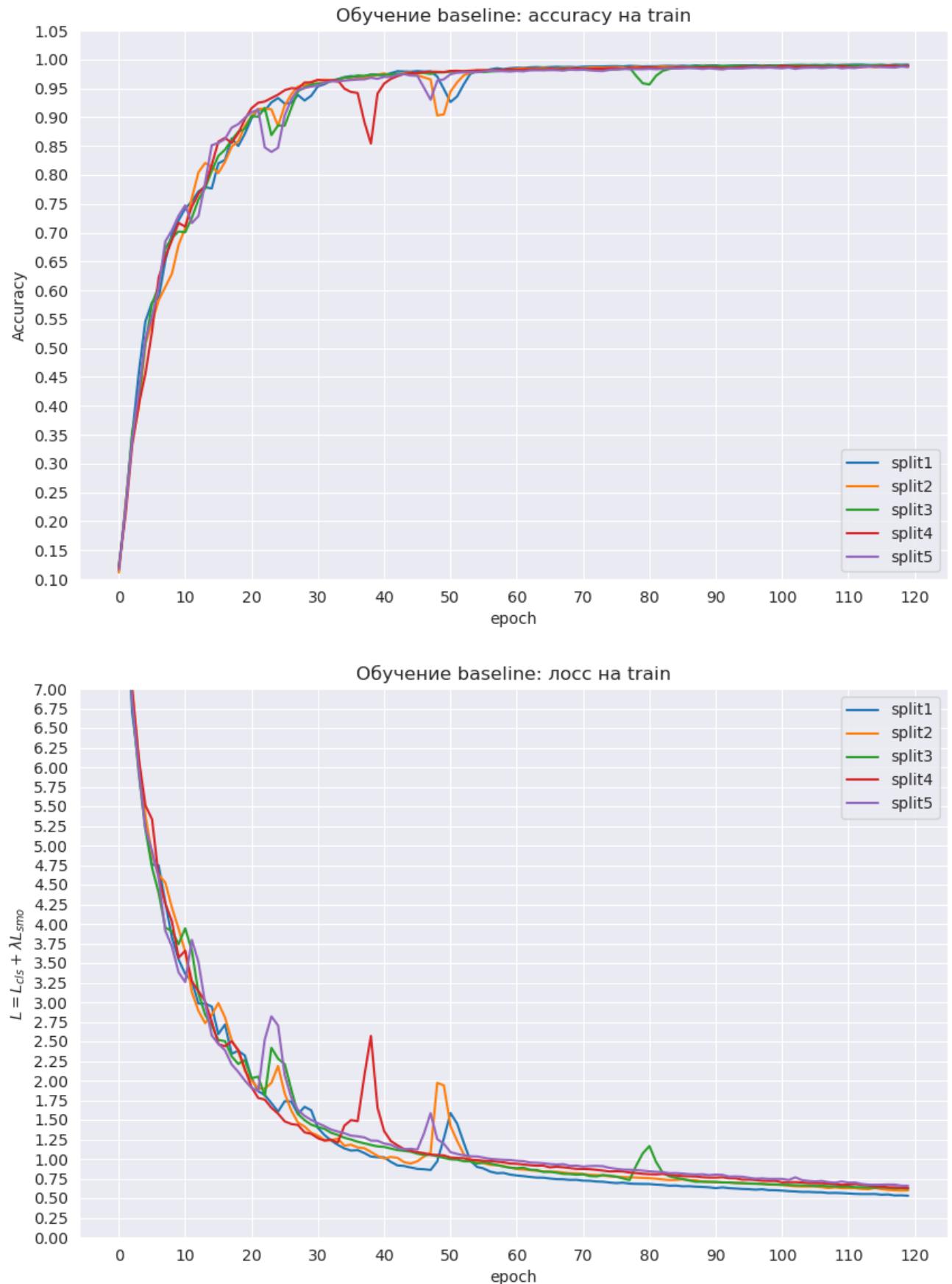


Рисунок 11: Обучение baseline на 5 сплитах целевого датасета

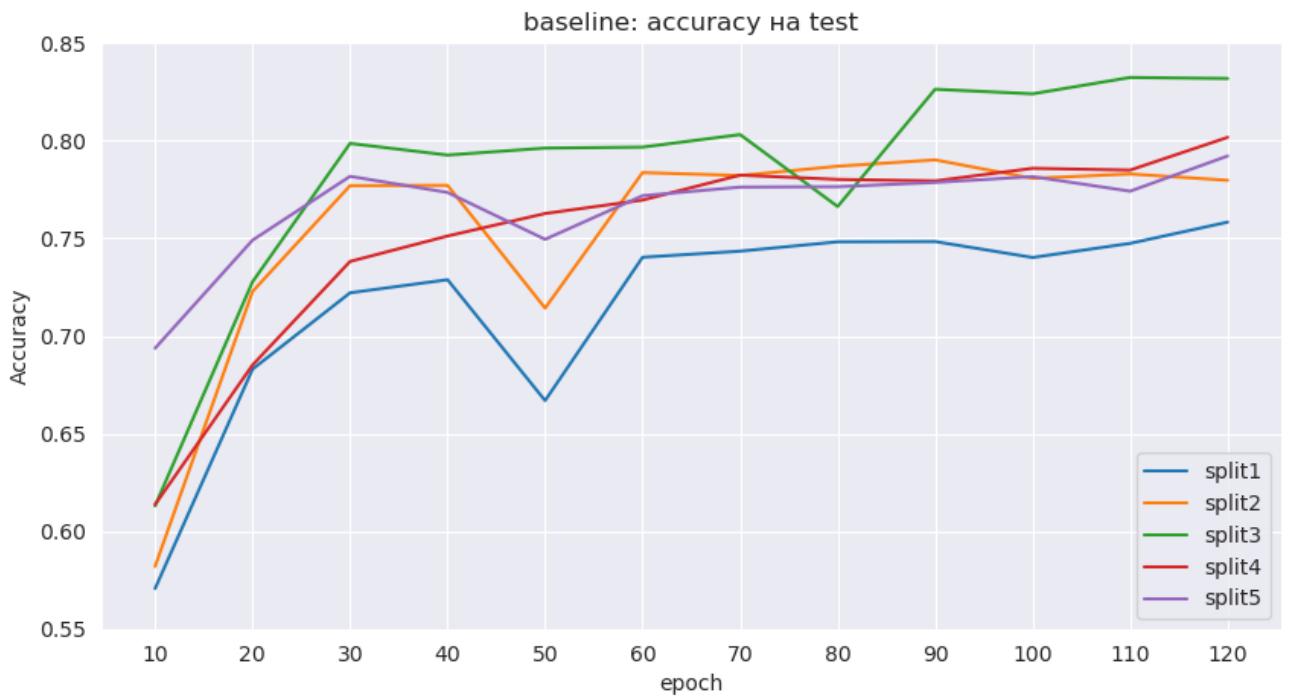


Рисунок 12: Результаты baseline

Из графиков видно, что обучение на каждом из 5 сплитов проходит достаточно успешно. При этом, несмотря на некоторые различия между сплитами, общий тренд в обучении сохраняется. Сначала модель быстро усваивает данные - учит энкодер, а потом медленно и последовательно улучшает предсказания за счёт декодеров. Падение качества на 50 эпохе на 1, 2 и 5 сплитах и на 80 эпохе на сплите 3 связано как раз с переходом к обучению декодеров. Вероятно, переходить к обучению декодеров чуть позже было бы лучше, т. к. третий сплит оказался самым удачным и показывает лучшее качество, а первый, наоборот - худшее. Применим кросс-валидацию, которая как раз предназначена для усреднения и обобщения подобных результатов. Далее полученные результаты модели будут упоминаться под общим названием *baseline*. За результат *baseline* примем средние значения, которые вместе с точными замерами качества можно найти в таблице 3. Из этой же таблицы можно сделать вывод, что деградации качества во время дообучения декодеров удалось избежать на всех сплитах, кроме второго. Деградацию на втором сплите можно назвать малой и приемлемой. Это показывает, что авторы ASFormer достаточно тщательно подошли к выбору процедуры обучения и её дальнейшее уточнение не требуется.

Таблица 3: Результаты обучения baseline на собственных эмбеддингах

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучшая accuracy	эпоха с лучшей accuracy
1	0.758	66.2	0.751	0.728	0.623	0.758	120
2	0.780	75.6	0.808	0.760	0.708	0.790	90
3	0.832	77.3	0.841	0.831	0.784	0.832	110
4	0.802	72.5	0.793	0.784	0.700	0.802	120
5	0.792	70.5	0.788	0.749	0.680	0.792	120
среднее	0.793	72.4	0.796	0.770	0.699	0.795	-

Итак, в данном исследовании baseline представляет из себя модель ASFormer с backbone I3D. С помощью общедоступных весов модели I3D из видео целевого датасета получены векторы признаков, на которых была обучена и протестирована основная модель ASFormer. Качество полученной модели всё же уступает авторскому, но является достаточным для использования в качестве baseline. Заметим, что авторское качество можно воспроизвести лишь на фиксированном наборе датасетов, а описанный pipeline можно в дальнейшем применить для новых данных.

2.4 Описание модели для мультимодального pretrain

Как можно понять из введения, существует достаточно много мультимодальных архитектур, которые можно было бы выбрать для исследования, например уже упомянутые ActBERT [11] и STALE [12] или совсем новые модели HierVL [24] и EgoVLP [25]. После длительного рассмотрения для мультимодального предобучения было решено использовать модель VideoCLIP из [26].

Основными причинами выбора именно VideoCLIP являются использование самого большого возможного датасета для предобучения - HowTo100M и наличие общедоступных эмбеддингов из него; использование contrastive learning для связи между модальностями, что позволяет легче выделить ответственную за видео часть модели; возможность адаптации к новым задачам, в том числе Action Segmentation, продемонстрированная в оригинальной статье на датасете COIN [27]. Так же стоит отметить, что в качестве backbone выступает нейросеть S3D, во многом схожая с I3D, которая является backbone для выбранного baseline. В дополнение к уже перечисленному в статье [26] продемонстрированы впечатляющие способности VideoCLIP к zero-shot переносу, что позволяет провести дополнительный экспе-

римент вовсе без дообучения на целевом датасете

2.4.1 Сеть для извлечения векторов признаков: S3D

При описании backbone авторы ASFormer ссылаются сразу на 2 другие статьи: [28] для архитектуры и [29] для процесса обучения.

В статье [28] архитектура S3D была получена путём некоторых преобразований I3D. Иллюстрацию для S3D на рисунке 13 можно сравнить с иллюстрацией архитектуры I3D на рисунке 3. Видно, что было сделано лишь 2 изменения в свёрточных блоках модели. Во-первых, первую и третью свёртки заменили на свёртку в пространстве с последующей свёрткой по времени, тем самым сделав их сепарабельными по времени. Во-вторых, внутри каждого inception блока так же заменили 2 трёхмерные свёртки сепарабельными по времени, после чего получился так называемый temporal separable inception block, который можно увидеть на рисунке 13b и сравнить с просто трёхмерным inception боком с рисунка 3b. Эти изменения трудно назвать значительными, но они дают прирост качества и уменьшают потребление ресурсов, возможно именно поэтому авторы VideoCLIP выбрали такой backbone.

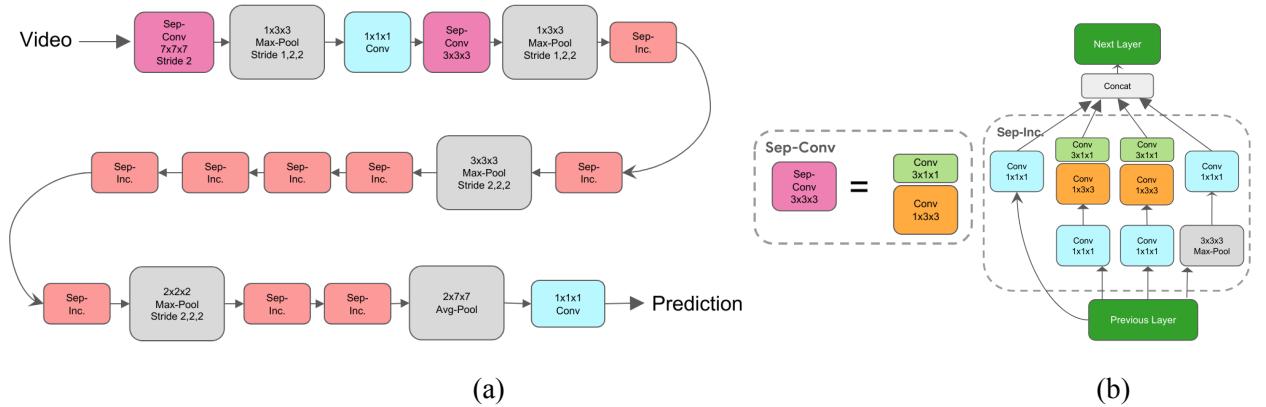


Рисунок 13: а) Общий вид архитектуры нейросети S3D б) Архитектура одного temporal separable inception-блока

Обучение используемой версии S3D описано в статье [29]. Авторы адаптировали процесс обучения к датасету HowTo100M. Как уже упоминалось, датасет шумный и к нему не существует разметки, поэтому было предложено учить совместные представления для видео и текста. В качестве представлений текста использовались эмбеддинги word2vec, обученного на датасете GoogleNews, из первой статьи о word2vec [30]. Схему предложенной архитектуры можно увидеть на рисунке 14, где векторы $f(x)$ и $g(x)$ обозначают эмбеддинги для видео и для текста соответственно.

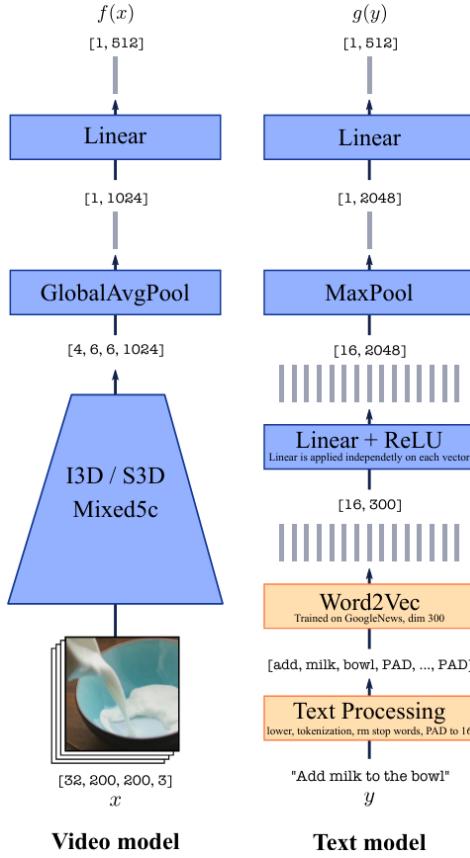


Рисунок 14: Pipeline обучения S3D

Для получения сильных векторов признаков в общем пространстве авторам S3D потребовалась нетривиальная функция потерь MIL-NCE, описанная формулой 4.

$$L = \max_{f,g} \sum_{i=1}^n \log \left(\frac{\sum_{(x,y) \in P_i} e^{f(x)^T \cdot g(y)}}{\sum_{(x,y) \in P_i} e^{f(x)^T \cdot g(y)} + \sum_{(x',y') \sim N_i} e^{f(x')^T \cdot g(y')}} \right) \quad (4)$$

где

n - общее число верных пар (фрагментов видео, субтитры)

x - фрагмент видео, в котором совершается одно действие

y - фрагмент субтитров видео, описывающий одно действие

f - функция для эмбеддинга видео, в данном случае S3D

g - функция для эмбеддинга текста, в данном случае word2vec с предварительным препроцессингом и несколькими дополнительными слоями

P_i - множество позитивных примеров, состоит из 5 ближайших к фрагменту видео по времени фрагментов субтитров.

N_i - множество из 512 негативных примеров, сэмплированных из того же батча, что и пара (x, y)

2.4.2 Основная мультиомодальная модель: VideoCLIP

Архитектура VideoCLIP, предложенная в [26] показана на рисунке 15. Векторы признаков видео, полученные из backbone S3D, являются двумерной последовательностью и обрабатываются с помощью 6 первых слоёв BERT. Признаки текста получаются из субтитров и используют 12 слоёв BERT. В обоих случаях слои проинициализированы весами, полученными в оригинальной статье [2] в архитектуре $BERT_{BASE-uncased}$. Сам факт использования именно этой нейросети с именно такой инициализацией никак не влияет на дальнейшую работу модели. Энкодеры для текста и видео могут быть любыми, в том числе различными, и могут быть инициализированы как угодно. Это делает модель более универсальной, но приводит к более сложной процедуре обучения.

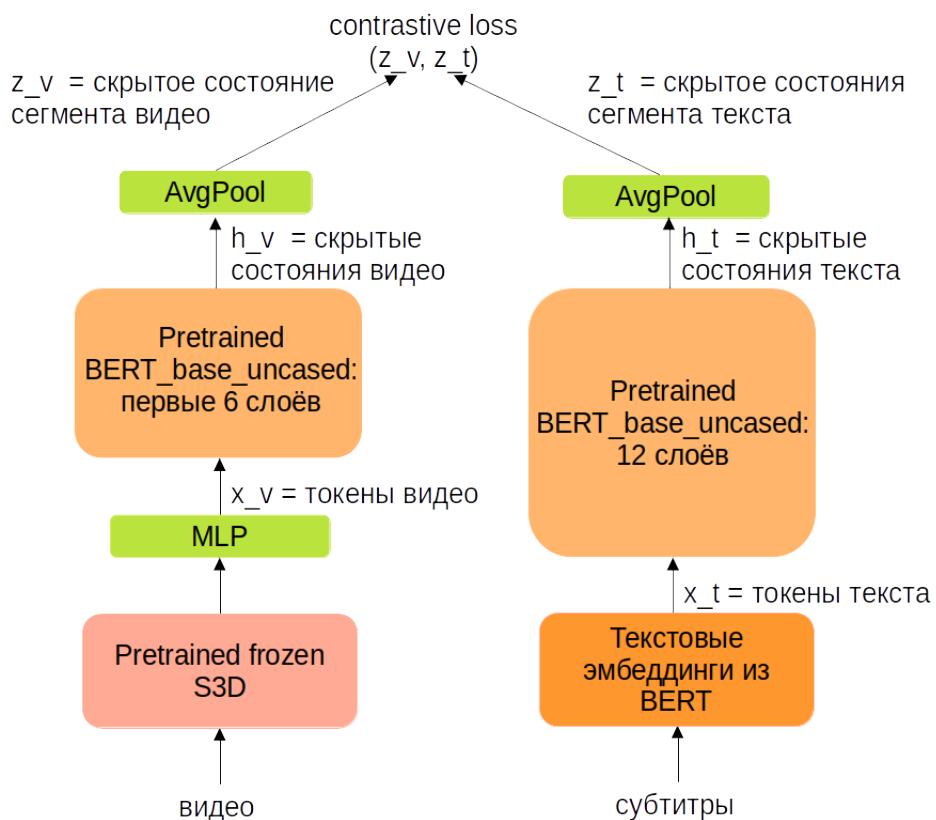


Рисунок 15: Архитектура VideoCLIP

VideoCLIP стремится выучить представления видео h_v и текста h_t в общем латентном пространстве на уровне токенов, а далее использует Average Pooling для усреднения в пределах видео-фрагмента с одним действием. Чтобы полученные векторы признаков для фрагмента видео z_v и векторы признаков для фрагмента текста z_t были как можно ближе друг к другу, используется contrastive loss из формулы 5

$$\begin{aligned}
L = & - \sum_{(v,t) \in B} (\log NCE(z_v, z_t) + \log NCE(z_t, z_v)) = \\
= & - \sum_{(v,t) \in B} \left(\log \left(\frac{\exp(\frac{z_v \cdot z_t^+}{\tau})}{\sum_{z \in (z_t^+, z_t^-)} \exp(\frac{z_v \cdot z}{\tau})} \right) + \log \left(\frac{\exp(\frac{z_t \cdot z_v^+}{\tau})}{\sum_{z \in (z_v^+, z_v^-)} \exp(\frac{z_t \cdot z}{\tau})} \right) \right) \quad (5)
\end{aligned}$$

где

- B - батч из пар (фрагмент видео, субтитры фрагмента)
- z_v - вектор признаков для субтитров фрагмента видео
- z_t - вектор признаков для фрагмента видео
- $\tau = 1$ - температура
- + и - обозначают предварительно сэмплированные позитивные и негативные примеры. Алгоритмы сэмплирования будут описаны далее.

Благодаря использованию предобученного backbone для видео, модель может потреблять более длинные видео сегменты, что помогает против главной проблемы датасета HowTo100M: шумных, не всегда точно расположенных субтитров. Вместо того, чтобы опираться на субтитры, совпадающие с видео-рядом по времени, авторы VideoCLIP создают пары из случайным образом удлинённых отрезков видео и текста, от которых требуется лишь пересечение по времени. Для **сэмплирования позитивных примеров** используется следующий алгоритм:

- Сэмплировать фрагмент субтитров с границами $[t_s, t_e]$ во времени
- Случайно выбрать некоторый момент времени $t_c \in [t_s, t_e]$ в качестве центра по времени для будущего фрагмента видео
- Выбрать случайную продолжительность от 0 до 32 секунд для видео-фрагмента. В этот момент фрагмент видео определён однозначно центром и продолжительностью.
- Сохранить результат как позитивный пример пары фрагментов.

Таким образом получаются положительные примеры разнообразной длины, которые должны лишь наполовину пересекаться по времени. При этом текст тоже может быть взят из удлинённого отрезка времени. Его выбор происходит первым, чтобы избежать сэмплирования отрезков видео, к которым нет субтитров.

В случае с HowTo100M негативные примеры требуется сэмплировать особенно аккуратно. Обычное сэмплирование из других позитивных пар внутри батча может привести к

попаданию пары, близкой по времени к изначальной паре (сегмент видео, субтитры сегмента), в число негативных. При этом она на самом деле может быть позитивной или схожей с позитивной, что запутает модель. В то же время негативные примеры должны оставаться сильными на протяжении всего времени обучения. Для решения перечисленных проблем авторами статьи [26] был разработан специальный алгоритм.

Алгоритм сэмплирования негативных примеров:

- Вычислить для каждого видео глобальное представление по формуле

$$z_V = \frac{1}{|2B_V|} \sum_{(v,t) \in B_V} (z_v + z_t) \quad (6)$$

где

V - видео из датасета

B_V - все пары (фрагмент видео, субтитры фрагмента), известные для данного видео

- Поместить полученные представления в базу данных FAISS [14] с плотным индексом.
- Выбрать $C =$ числу батчей случайных видео.
- Для каждого из случайно выбранных видео v найти $2k$ его ближайших соседей
- Из $2k$ ближайших соседей случайно выбрать k . Таким образом k видео образуют кластер из близких друг к другу видео. Это даёт возможность использовать их как сильные негативные примеры друг для друга.

Само обучение проводится по кластерам, получившимся в конце алгоритма сэмплирования негативных примеров, т. е. считаем, что B из формулы 5 - это один из C созданных кластеров. Далее для каждого из k видео в кластере случайно выбираем 16 пар (фрагмент видео, субтитры фрагмента), делаем сэмплирование позитивных примеров (удлинение сегментов) и проводим обучение. При авторском обучении использовалось $k = 32$, а больших батчей по $32 \cdot 16$ пар сегментов было $C = 8$ по одному на каждый из 8 NVIDIA Tesla V100 GPU, задействованных в обучении.

2.5 Мультимодальное предобучение на одном GPU

Для того, чтобы повторить впечатляющие результаты VideoCLIP и перенести их на целевой датасет 50salads, необходимо выполнить мультимодальное предобучение на датасете HowTo100M. Поскольку HowTo100M содержит колоссальное количество видео материалов, которое тяжело поместить на одну машину, будем использовать готовые, извлечённые

с помощью S3D, векторы признаков. К счастью, авторы HowTo100M предоставляют заранее подготовленные векторы признаков на своей [официальной странице](#). Нужно признать, что данные векторы признаков имеют размерность 1024, а не 512 и, вероятно, последний обученный линейный слой из архитектуры на рисунке 14 был опущен. Тем не менее за счёт входного линейного слоя (MLP в видео-части на рисунке 15) общая архитектура остаётся такой же и от обучения ожидаются схожие с оригиналом результаты.

График мультимодального предобучения модели можно увидеть на рисунке 16. Заметим, что в течение одной эпохи функция потерь несколько раз пересчитывается и на тренировочных и на валидационных данных. Подписи по оси x следует читать как "первая эпоха обучается от отметки 0 до отметки 1" и т. д.

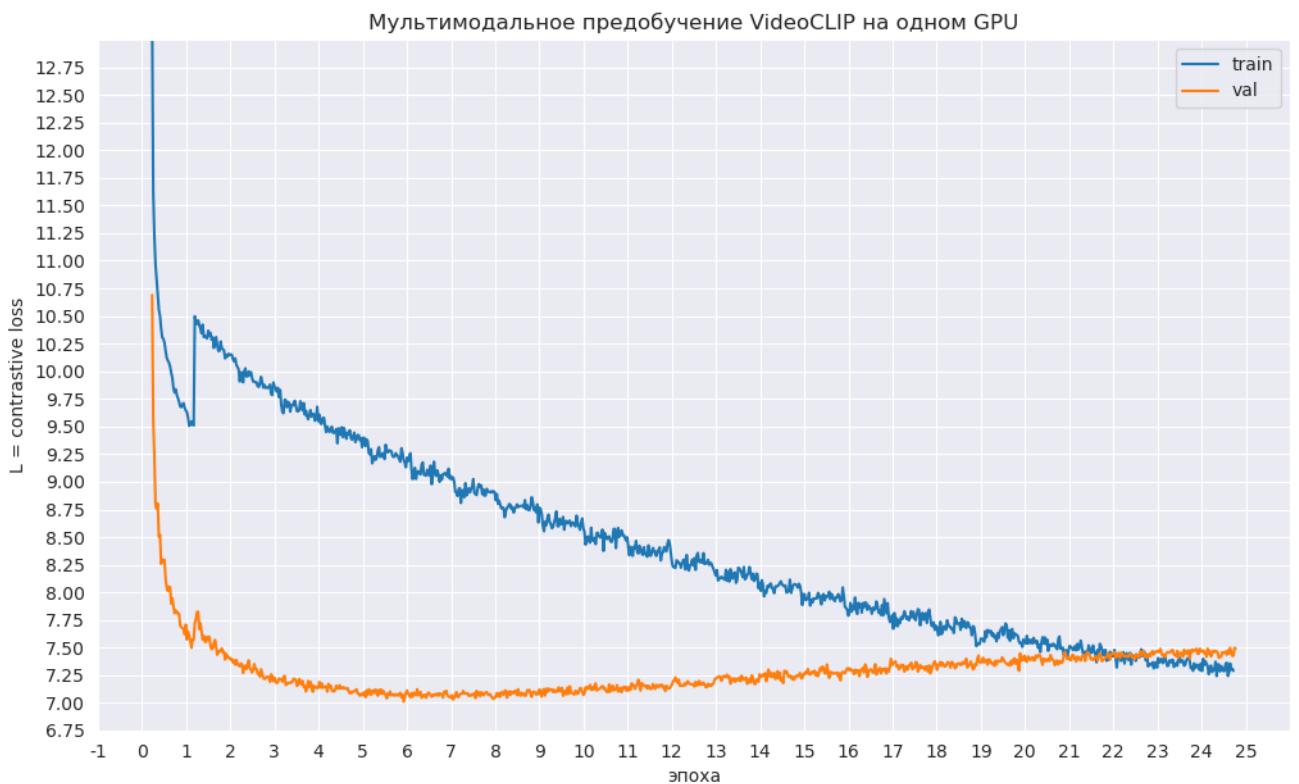


Рисунок 16: Мультимодальное предобучение на одном GPU. Подписи по оси x следует читать как "первая эпоха обучается от отметки 0 до отметки 1, вторая эпоха обучается от отметки 1 до отметки 2..." и так далее. Функция потерь L несколько раз пересчитывается внутри одной эпохи

Видно, что на тренировочной части данных функция потерь уменьшается на протяжении всего обучения. Кажется, что модель становится лучше, но достаточно обратить внимание на функцию потерь на валидации, чтобы понять, что что-то пошло не так. Здесь явно видно переобучение: модель настолько сосредоточена на обучающей части выборки,

что выучивает закономерности, которых на самом деле нет. Обучение стоило бы остановить в районе эпохи 8. Естественно, качество такой модели будет намного хуже, чем в оригинале.

2.5.1 Возможные причины неудавшегося эксперимента

Попробуем проанализировать, что именно пошло не так. Код модели взят из официального [репозитория VideoCLIP](#), а векторы признаков получены тем же способом, что в статье [26]. Более того: используются оригинальные авторские настройки для процедуры обучения. Оригинальную конфигурацию пришлось изменить, т. к. сложно получить доступ к 8 столь мощным GPU. График 16 получен при обучении на одном GPU под операционной системой windows, тогда как в оригинале использовалось 8 GPU и linux.

В данном случае количество GPU играет ключевую роль. Взглянем ещё раз на алгоритм сэмплирования негативных примеров из 2.4.2. На третьем шаге авторы VideoCLIP выбрали C случайных видео, по которым далее строили кластеры. Значит, самих кластеров получалось ровно C . При этом уже внутри кластера выбрались 32 видео и 16 фрагментов в каждом из них. Т. е. в одной итерации обучения было задействовано $C \cdot 32 \cdot 16$ позитивных пар сегментов. Теперь заметим, что в авторской процедуре было $C =$ числу доступных GPU. Это было сделано специально, чтобы на одном GPU обрабатывался один большой кластер (батч), а результаты соотносились внутри модели во время обратного прохода и обновления градиентов. Таким образом параллельно обрабатывалось C кластеров. Размер $32 \cdot 16$ так же был выбран не случайно: он позволял разумно использовать имеющиеся у авторов мощности.

К несчастью, при $C = 1$ идея начинает рушиться. На третьем шаге алгоритма сэмплирования негативных примеров теперь выбираем всего одно случайное видео и далее получаем 1 кластер вокруг него. При этом в силу случайности выбранное видео может оказаться сильно смещено в пространстве векторов z_V . Таким образом на каждой итерации обучения модель учится на случайном подпространстве. Из-за отсутствия усреднения по большим батчам (кластерам) изменения модели становятся слишком крупными и хаотичными.

Решить данную проблему достаточно тяжело. При попытке увеличить число кластеров на одном GPU приходится столкнуться с пределами возможностей современных видеокарт. При текущих мощностях можно попытаться уместить на одной видео-карте лишь 2 больших кластера. В случае уменьшения размера кластера произойдёт уменьшению числа негативных примеров в функции потерь 5 и как следствие деградация качества. Если же попытаться задействовать большее число GPU, возникает другая сложность. Дело в том, что

авторы VideoCLIP при написании своего кода усиленно использовали библиотеку fairseq [13], которая написана исключительно для операционной системы linux. Чтобы обучить модель на нескольких GPU на операционной системе windows, требуется перевести значительную часть многопоточного кода библиотеки.

2.6 Мультимодальное предобучение на семи GPU

К сожалению, идеальный сервер для воспроизведения предобучения VideoCLIP найти не удалось. Получилось найти лишь схожую конфигурацию с семью GPU, доступ к который был совместным. Из-за совместного доступа обучение пришлось несколько раз останавливать и запускать заново. Такая ситуация не удивительна, т. к. требования модели были очень высокими. Здорово, что нашёлся достаточно близкий по конфигурации сервер и на нём всё же получилось провести предобучение. Хотелось бы выразить благодарность Евгению Орлову и команде QA за предоставленные ресурсы.

График обучения на семи GPU представлен на рисунке 17, как и в случае с одним GPU функция потерь пересчитывается несколько раз внутри эпохи. Сразу можно выделить серьёзное отличие: график стал более гладким. Это означает, что усреднение по семи кластерам помогло в обучении и сгладило функцию потерь. Теперь модель изменяется плавно и улучшает результат на семи кластерах в пространстве векторов z_V , а не на одном.

Тем не менее на графике можно заметить несколько скачков функции потерь. Они происходят одновременно на обучающей и валидационной выборках и связаны с перезапусками модели. Несмотря на подобные отклонения от авторской процедуры обучения, удалось побороть главную проблему предыдущей модели - слабые намёки на переобучение можно увидеть лишь после эпохи 22. Вероятно, при использовании восьми кластеров график был бы ещё более гладким, а попыток переобучения не было бы вовсе. Всё же полученная модель значительно ближе к авторской, чем результат на одном GPU и её можно считать успешной. В дальнейших экспериментах в качестве мультимодального предобучения используется именно эта модель.

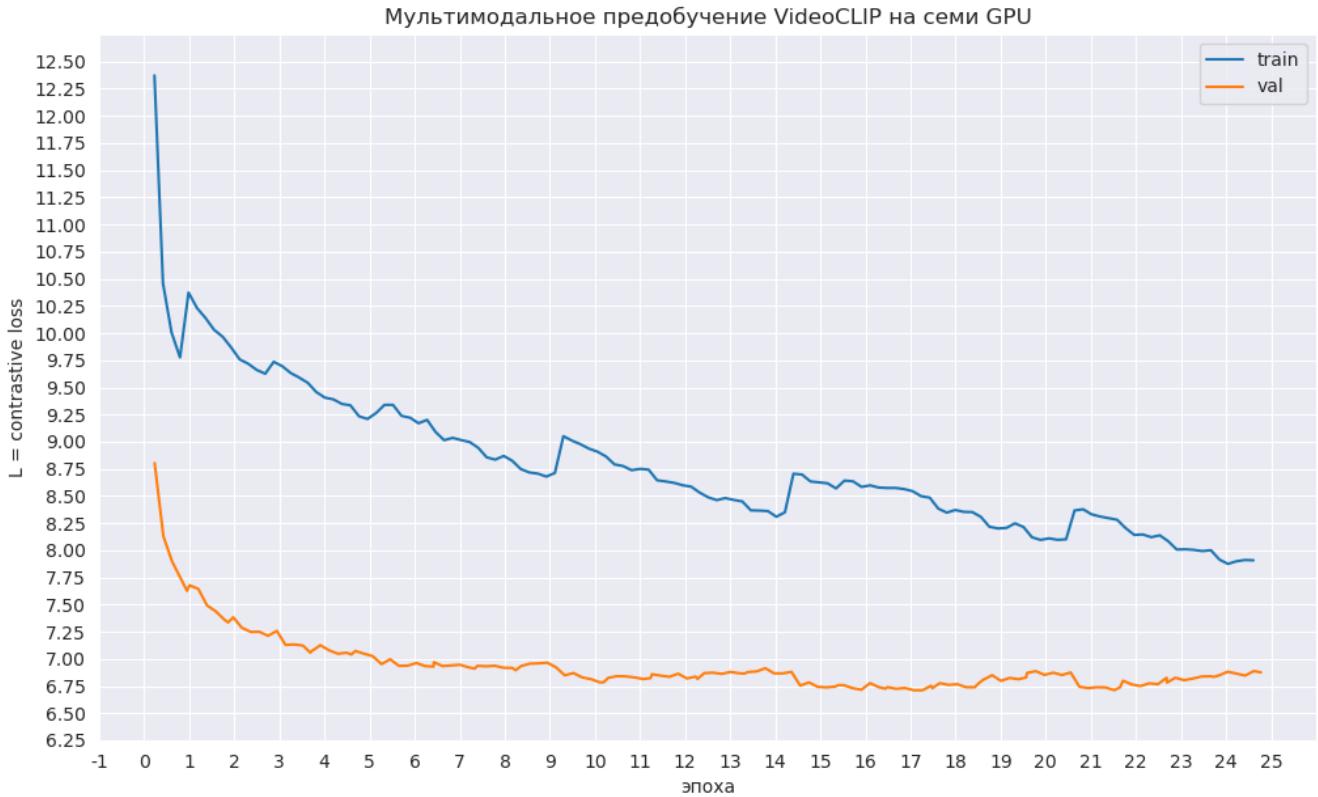


Рисунок 17: Обучение мультиомодального pretrain на семи GPU. Подписи по оси x следует читать как ”первая эпоха обучается от отметки 0 до отметки 1, вторая эпоха обучается от отметки 1 до отметки 2 ...” и так далее. Функция потерь L несколько раз пересчитывается внутри одной эпохи

Подводя промежуточный итог, после тщательного анализа в качестве мультиомодальной модели для исследования была выбрана нейросеть VideoCLIP. Для воспроизведения её обучения потребовались значительные ресурсы. Обучение было проведено на семи GPU с использованием операционной системы linux, что достаточно близко к авторской конфигурации. Несмотря на возникшие сложности, были получены собственные значения весов для модели, которые будут далее использованы для экспериментов на целевом датасете.

2.7 Выделение видео-части мультиомодальной модели

Итак, получены собственные веса для мультиомодальной модели VideoCLIP. Далее следует переходить к экспериментам на целевом датасете, содержащем лишь модальность видео. Благодаря contrastive learning, переход от сопоставления отрезков видео и их субтитров к задаче action segmentation достаточно прост. Поскольку, классы действий известны заранее, их словесные описания можно пропустить через текстовую часть модели. Таким образом получаются эмбеддинги для классов действий.

Обработку видео-последовательности тоже необходимо адаптировать. В предобучении модель оперирует сегментами видео и субтитрами к ним, поэтому на последнем слое происходит дополнительная агрегация скрытых состояний видео h_v в одно скрытое состояние сегмента z_v . Это делается с помощью Average Pooling, который можно заметить на рисунке 15. В случае задачи Action Segmentation такая агрегация не требуется. При этом видео длиною в несколько минут всё же нельзя подать в модель целиком, поэтому нельзя избежать деления на сегменты. Для действий в середине сегмента предсказания будут качественными благодаря трёхмерным свёрткам в backbone и блокам с attention внутри модели, а вот для действий по краям возникает проблема: модель получает дополнительную информацию из кадров только до или только после действия. Чтобы решить проблему, используем скользящие окно размером 32 с шагом 16, как это сделали авторы VideoCLIP для датасета COIN [27]. В данном случае важна не только одинаковая задача, но и характер видео внутри датасета: длинные видео по 10-15 минут с несколькими различными действиями. Таким образом эмбеддинг для конкретного кадра - это усреднение двух векторов h_v , полученных после применения модели к двум окнам, содержащим кадр, на соответствующих кадру местах.

Остаётся найти способ совместить эмбеддинги для кадров и для классов действий. Для этого используем их скалярное произведение. Чтобы предсказать действие по вектору признаков кадра, скалярно перемножим его со всеми известными векторами признаков действий, а затем из получившихся произведений выберем максимальное. Поскольку во время предобучения модель обучалась сопоставлять тексты и видео через contrastive loss, а скользящие окна можно воспринимать как искусственно созданные сегменты, векторы признаков у кадра и у названия действия на нём должны быть максимально близкими друг к другу.

В отличии от авторов статьи [26] не будем добавлять к предсказаниям фоновый класс. Как уже упоминалось, в датасете 50salads есть 2 технических класса: `action_start` и `action_end`. Они служат вполне логичной заменой фонового класса, поскольку являются общими и нейтральными: всё происходящее на видео можно назвать действиями. Более того, при введении фонового класса предсказания не получится улучшить, поскольку в разметке датасета каждой секунде видео соответствует одно из 19 заранее отобранных действий.

2.8 Эксперименты без дообучения

2.8.1 Zero-shot перенос модели с самостоятельно обученными весами

После мультимодального предобучения и выделения видео-части модели можно на конец перейти к экспериментам и протестировать полученную модель. Начнём с самого простого пути: zero-shot переноса. Для него не требуется дообучать или как-то ещё изменять модель, достаточно лишь применить процедуру, описанную в 2.7. Полученные результаты представлены в таблице 4 и на рисунке 18 в виде матриц ошибок. Их нельзя назвать случайными, но они значительно уступают baseline. Далее попробуем понять причины столь низкого качества модели.

Таблица 4: Zero-shot перенос модели с собственными весами

номер сплита	accuracy	редакционное расстояние	f1@10	f1@25	f1@50
1	0.296	278.1	0.127	0.075	0.042
2	0.287	273.9	0.125	0.079	0.029
3	0.323	261.3	0.142	0.093	0.049
4	0.300	269.6	0.151	0.084	0.041
5	0.264	267.9	0.138	0.091	0.035
среднее	0.294	270.2	0.137	0.084	0.039

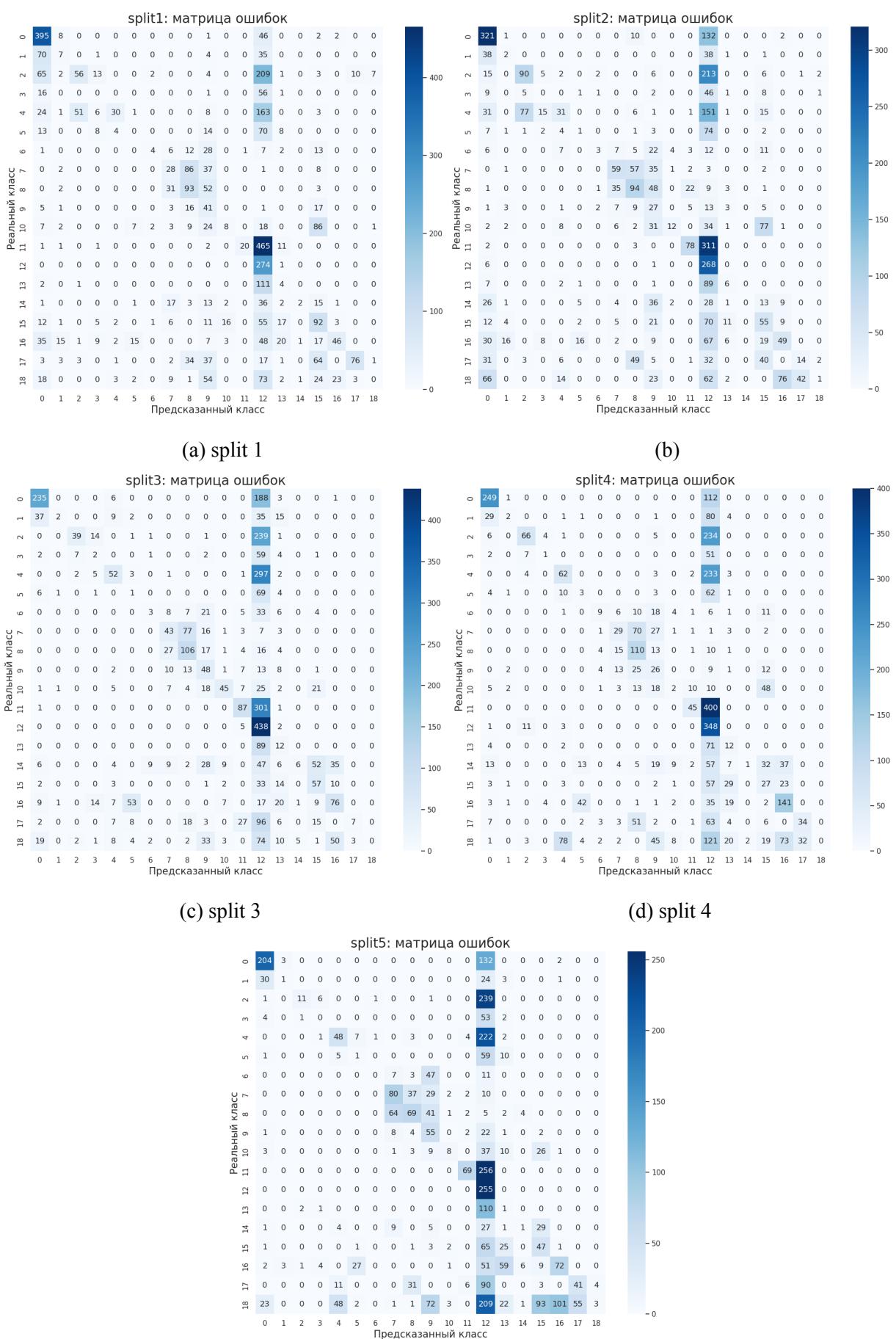


Рисунок 18: Матрицы ошибок при zero-shot переносе самостоятельно обученной модели

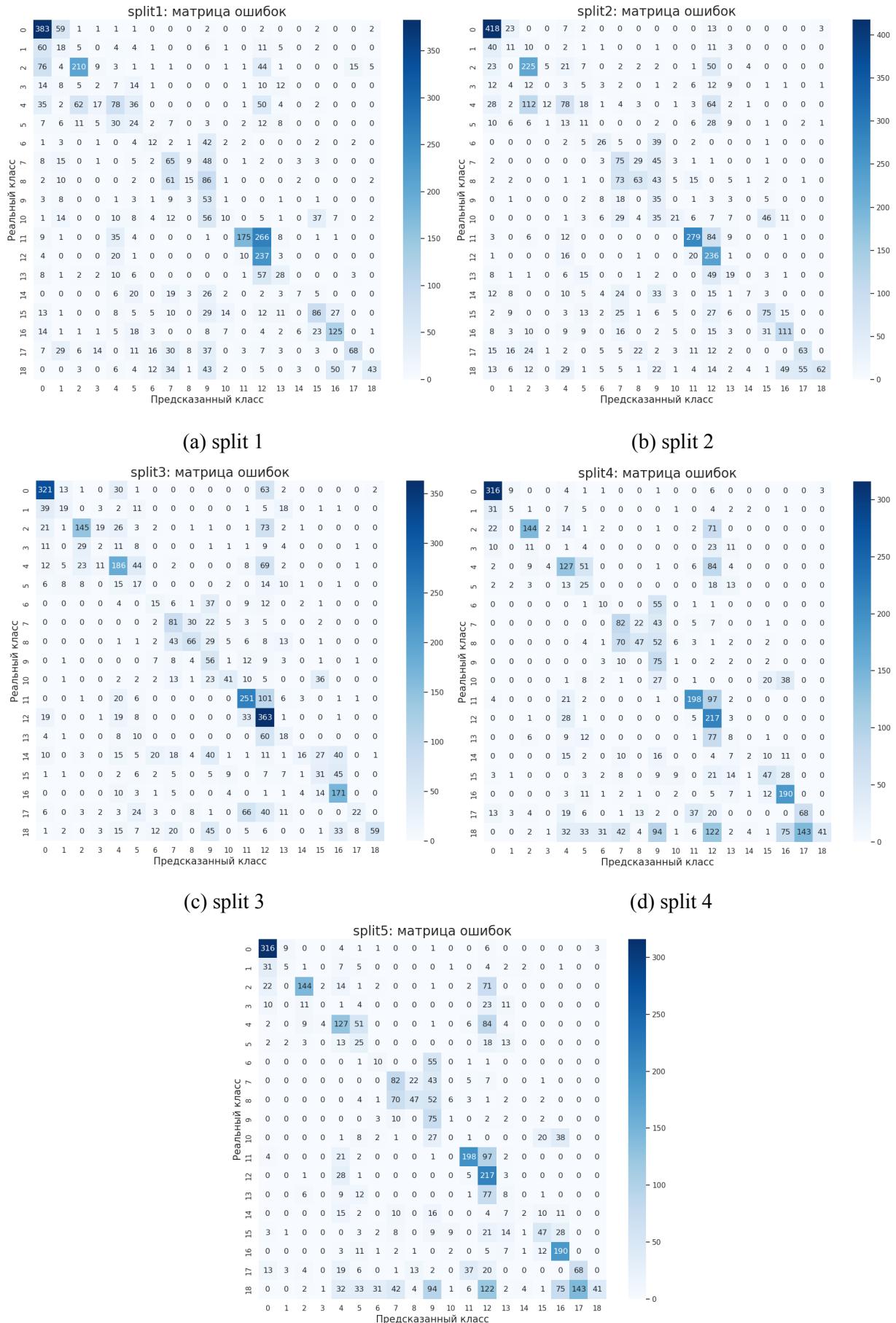
2.8.2 Zero-shot перенос общедоступной модели

Для сравнения проведём тот же эксперимент на модели с общедоступными весами. Единственным отличием будет размерность векторов признаков видео после прохождения S3D: как уже упоминалось, у оригинальной модели размерность входа для эмбеддингов видео = 512. Это связано с дополнительным линейным слоем сразу после S3D. Авторы VideoCLIP предоставили веса backbone и основной предобученной модели в своём официальном [репозитории](#), поэтому инференс модели не вызывает дополнительных трудностей. Результаты zero-shot переноса для авторской модели приведены в таблице. 5 и на рисунке 19

Общедоступная модель оказалась лучше, но и её результаты всё значительно хуже чем у baseline. Возможной причиной столь низкого качества является специфика датасета. Он небольшой и все действия в нём можно с лёгкостью отнести к одному из 12 больших доменов в датасете HowTo100M, а именно к Food and entertainig. С некоторыми допущениями можно даже выбрать 1 из 129 подкатегорий, Food preparation для всего датасета 50salads. Таким образом, весь целевой датасет лежит в некотором сравнительно маленьком подпространстве векторов z_V датасета для мультимодального предобучения. Более того, в подразделе с качественным анализом из оригинальной статьи авторы отмечают слабые результаты на датасете Youcook2 [31] из схожего домена на задаче сопоставления текста и видео. В качестве причин указаны не только различия между Youcook2 и HowTo100M в длительности сегментов, но и тенденция модели путать объекты схожих форм и цветов.

Таблица 5: Zero-shot перенос модели с общедоступными весами

номер сплита	accuracy	редакционное расстояние	f1@10	f1@25	f1@50
1	0.413	230.1	0.150	0.094	0.044
2	0.469	201.4	0.159	0.104	0.053
3	0.483	197.7	0.184	0.127	0.071
4	0.483	196.9	0.184	0.131	0.072
5	0.437	203.2	0.182	0.123	0.081
среднее	0.457	205.9	0.172	0.116	0.064



2.8.3 Анализ ошибок

Чтобы понять, где конкретно ошибаются модели, обратимся к матрицам ошибок 18 и 19, а так же приведём список всех классов в целевом датасете:

0. cut_tomato
1. place_tomato_into_bowl
2. cut_cheese
3. place_cheese_into_bowl
4. cut_lettuce
5. place_lettuce_into_bowl
6. add_salt
7. add_vinegar
8. add_oil
9. add_pepper
10. mix_dressing
11. peel_cucumber
12. cut_cucumber
13. place_cucumber_into_bowl
14. add_dressing
15. mix_ingredients
16. serve_salad_onto_plate
17. action_start
18. action_end

Как видно из рисунка 18 самостоятельно обученная модели делает больше всего ошибок на классе 12 `cut_cucumber`. Класс путается с 11 `peel_cucumber`, а так же с классами 0 `cut_tomato`, 2 `cut_cheese`, 4 `cut_lettuce` и техническим классом `action_end` на 4 и 5 сплитах. Это означает, что модель способна отличить основное действие `'cut'`, которое достаточно часто встречалось в предобучении, но пока не различает мелкие объекты, над которыми оно совершается. Заметим, что в случае с действиями 4 `cut_lettuce` и 2 `cut_cheese` спутанные объекты имеют схожий цвет и размер, а в случае с 11 `peel_cucumber` указанное действие специфично и в других видео может называться `'cut'`. Так же в матрицах ошибок можно заметить путаницу между действиями 7 `add_vinegar` и 8 `add_oil`. Эти ошибки тоже можно отнести к смешиванию объектов схожих форм и цветов, т. к. обе жидкости наливались из прозрачных бутылок. Последним стоит отметить класс 16 `serve_salad_onto_plate`, который путается с классом 18 `action_end`. Вероятно, тут влияет то, что готовое блюдо на тарелке обычно демонстрируется в конце видео-инструкций по приготовлению пищи.

На рисунке 19 для модели с общедоступными весами можно отметить, что проблема с классом 12 `cut_cucumber` выражена меньше. Он снова путается с 11 `peel_cucumber`. Смешивания с классами 2 `cut_cheese`, 4 `cut_lettuce` так же приступствуют, но выражены слабее. В целом классы 2 `cut_cheese` и 4 `cut_lettuce` начинают лучше распознаваться и выделяться на диагонали матрицы. Проблема с классами 7 `add_vinegar` и 8 `add_oil` актуальная и для общедоступной модели и серьёзных улучшений по сравнению с матрицами для самостоятельно обученной модели нет. Так же появляется новая проблема с классом 9 `add_pepper`: он путается с как с 7 `add_vinegar` и 8 `add_oil`, так и с 6 `add_salt` и 10 `add_dressing`. Вероятно, перечница похожа на солонку и на стакан для приготовления заправки, к тому же эти действия выполняются значительно быстрее нарезки ингредиентов и часто следуют друг за другом. Так же отметим проблему с техническими классами: 17 `action_start` достаточно часто принимается за 18 `action_end`. Возможно, причина в нейтральных названиях классов, использующих общее слово.

Различие результатов моделей с разными весами всё же нельзя отнести к специфике датасета. Следует признать, что собственные веса обучены хуже и уменьшение числа кластеров C всего на 1 влияет на конечный результат. Остаётся надеяться, что дообучение модели способно решить эту проблему.

2.9 Эксперименты по дообучению моделей

Переходя к дообучению стоит отметить, что его процедура была во многом позаимствована у авторов статьи о VideoCLIP, которые дообучали модель для решения задачи Action Segmentation на датасете COIN [27]. Основными заимствованиями являются длительность в 8 эпох и использование скользящих окон.

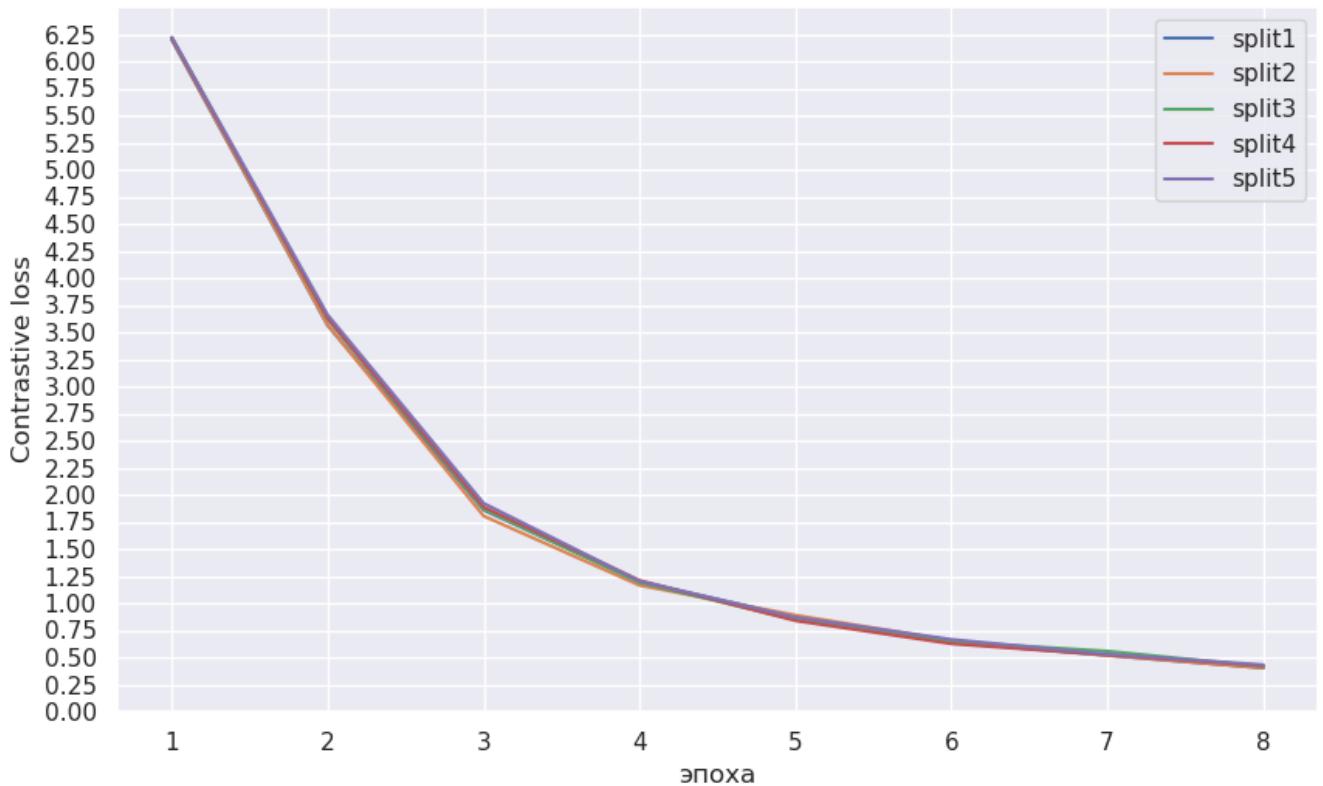
2.9.1 Дообучение модели с собственными весами

Как и в случае с baseline проведём обучением на пяти сплитах, а под общим результатом модели будем понимать средние значения. Графики дообучения представлены на рисунке 20, а числовые значения метрик качества модели в - таблице 6.

Таблица 6: Результаты собственной модели после дообучения

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучший лосс	эпоха с лучшим лоссом
1	0.837	58.6	0.486	0.455	0.381	0.888	8
2	0.791	79.2	0.465	0.429	0.346	0.774	8
3	0.753	94.2	0.496	0.446	0.347	0.944	7
4	0.799	77.1	0.503	0.463	0.384	0.739	8
5	0.816	65.7	0.498	0.443	0.388	0.612	7
среднее	0.799	75.0	0.490	0.447	0.369	0.791	-

Дообучение собственной модели на целевом датасете: train



Дообучение собственной модели на целевом датасете: val

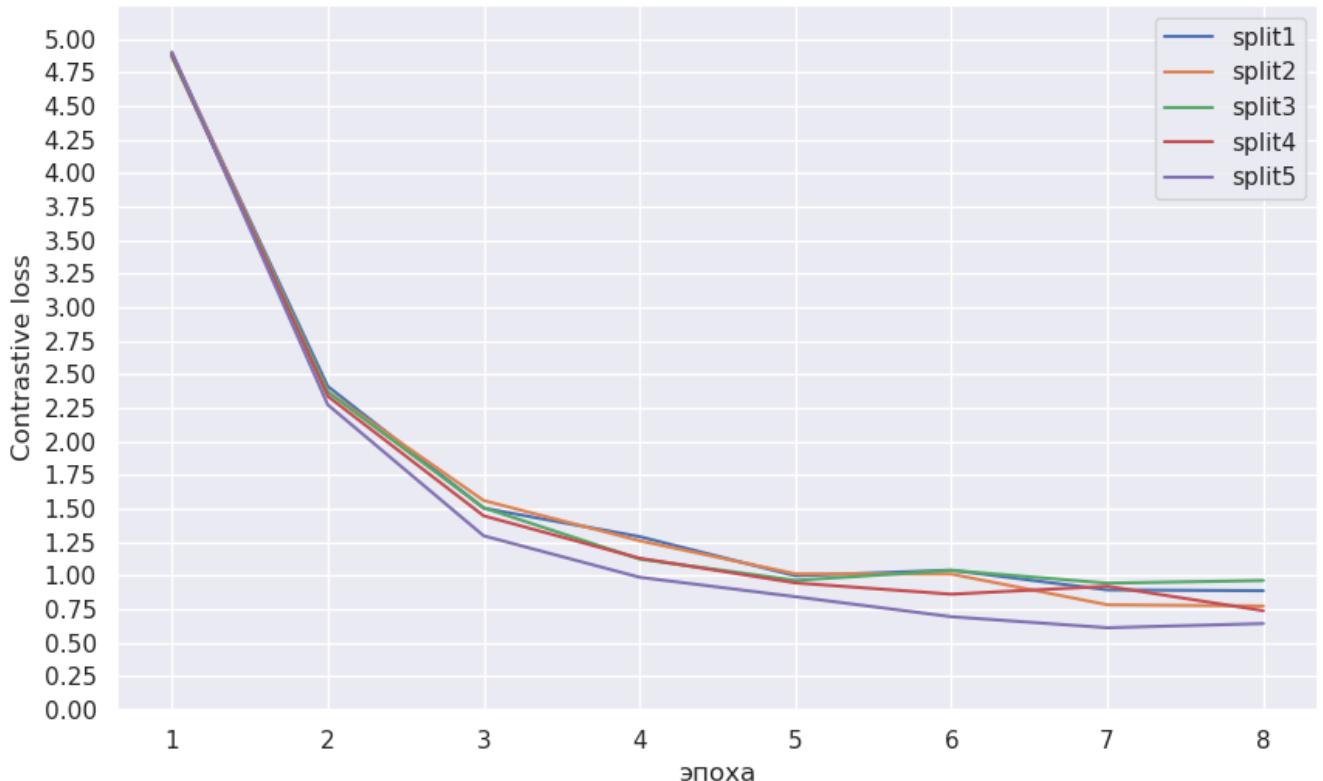


Рисунок 20: Дообучение собственной модели

На тренировочной выборке модель стабильно дообучается и значительно уменьшает функцию потерь на всех сплитах. На валидационных данных видно, что дообучение произошло успешно, хотя результаты всё же отличаются между сплитами. При анализе таблицы результатов становится понятно, что получилась достаточно сильная модель, которая способна соперничать с baseline. Результат всё же вышел слабым, особенно учитывая затраченные усилия. Далее попробуем проанализировать ошибки и улучшить его.

Посмотрим на последний столбец таблицы 6. Можно заметить, что лучшая модель для 1, 2 и 4 сплитов была получена лишь на последней эпохе. Так же заметим, что на графиках с рисунка 6 нет каких-либо поводов для окончания обучения: кривые остаются гладкими, функция потерь продолжает убывать, переобучения тоже нет. Возможно, модель следует дообучать некоторое большее количество эпох. Всё же процедура обучения была выбрана для датасета COIN [27], который значительно крупнее и содержит куда больше классов действий из датасета HowTo100M. При переходе к меньшему подпространству в рамках датасета 50salads модель могла бы сильнее адаптироваться к представленным классам.

Далее рассмотрим матрицы ошибок с рисунка 21. По сравнению с zero-shot ошибок стало значительно меньше, но именно оставшиеся неточности приводят к слабым результатам модели. Ситуация с классом 12 cut_cucumber стала значительно лучше, можно считать, что модель научилась отличать его от класса 11 peel_cucumber на всех сплитах, кроме 1 и 3. Больше всего данный класс теперь путают с 4 cut_lettuce. В данном случае объекты похожи и по цвету и по размеру, а действия одинаковы, поэтому окончательно разделить классы сложно. Ошибочное предсказание на классе 0 cut_tomato класса cut_cucumber происходит лишь в сплите 3, а на классе 2 cut_cheese - лишь в сплите 2. Это показывает способность модели дообучиться и начать различать мелкие объекты, специфичные для датасета. Модель продолжают путать классы 7 add_vinegar и 8 add_oil с добавлением класса 9 add_pepper на 2, 3 и 4 сплитах. Классы 6 add_salt и 10 add_dressing удалось почти полностью отделить от данной группы ошибок.

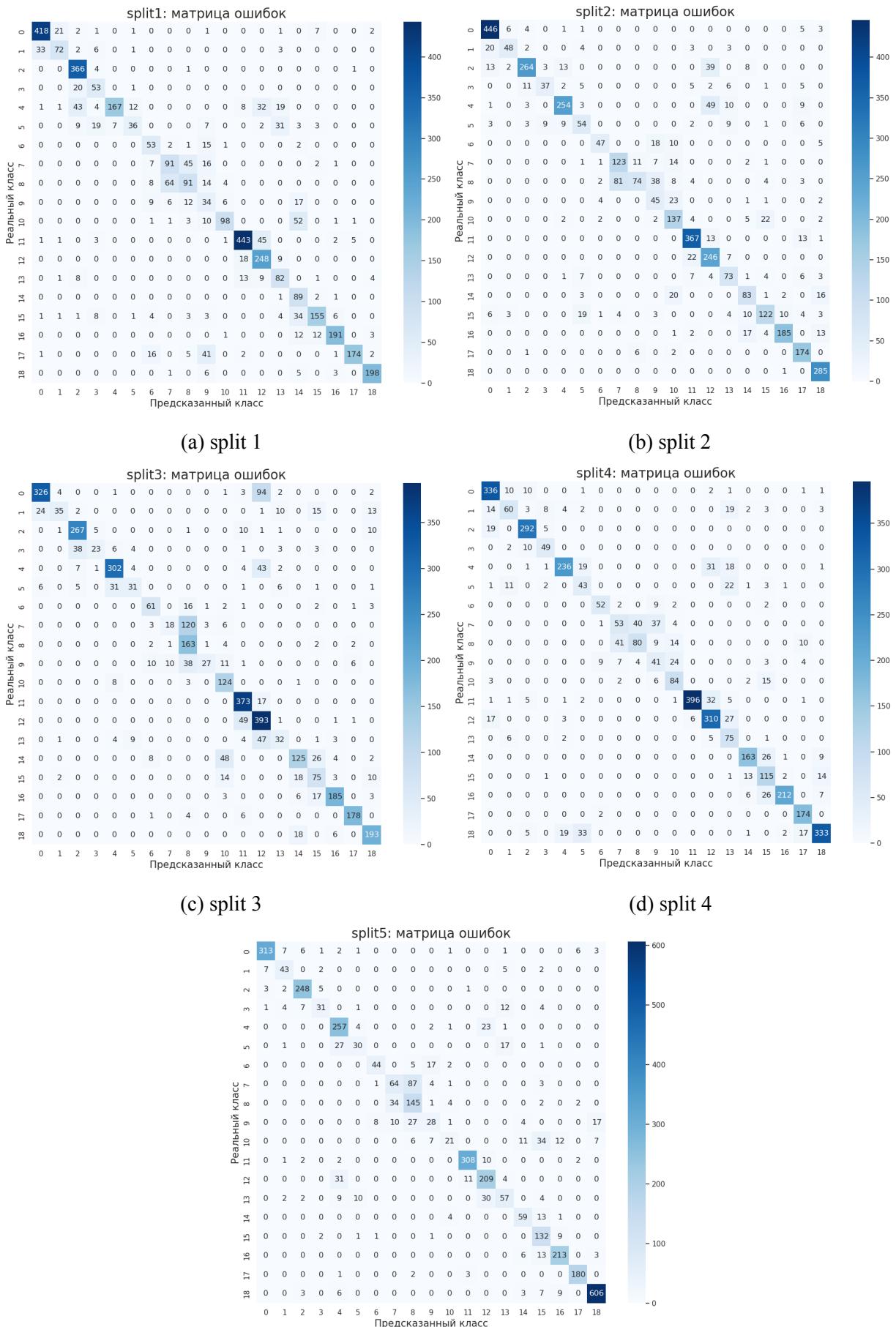


Рисунок 21: Матрицы ошибок при дообучении модели с собственными весами

Рассуждения выше наводят на мысль о возможности более долгого дообучения моделей: графики выглядят неоконченными, лучшие результаты получаются в том числе на последней эпохе, а смешиываемые классы различаются между сплитами. Далее попробуем обучить модель на большем числе эпох.

Графики более долгого или расширенного дообучения модели представлены на рисунке 22, метрики качества получившейся модели можно увидеть в таблице 7

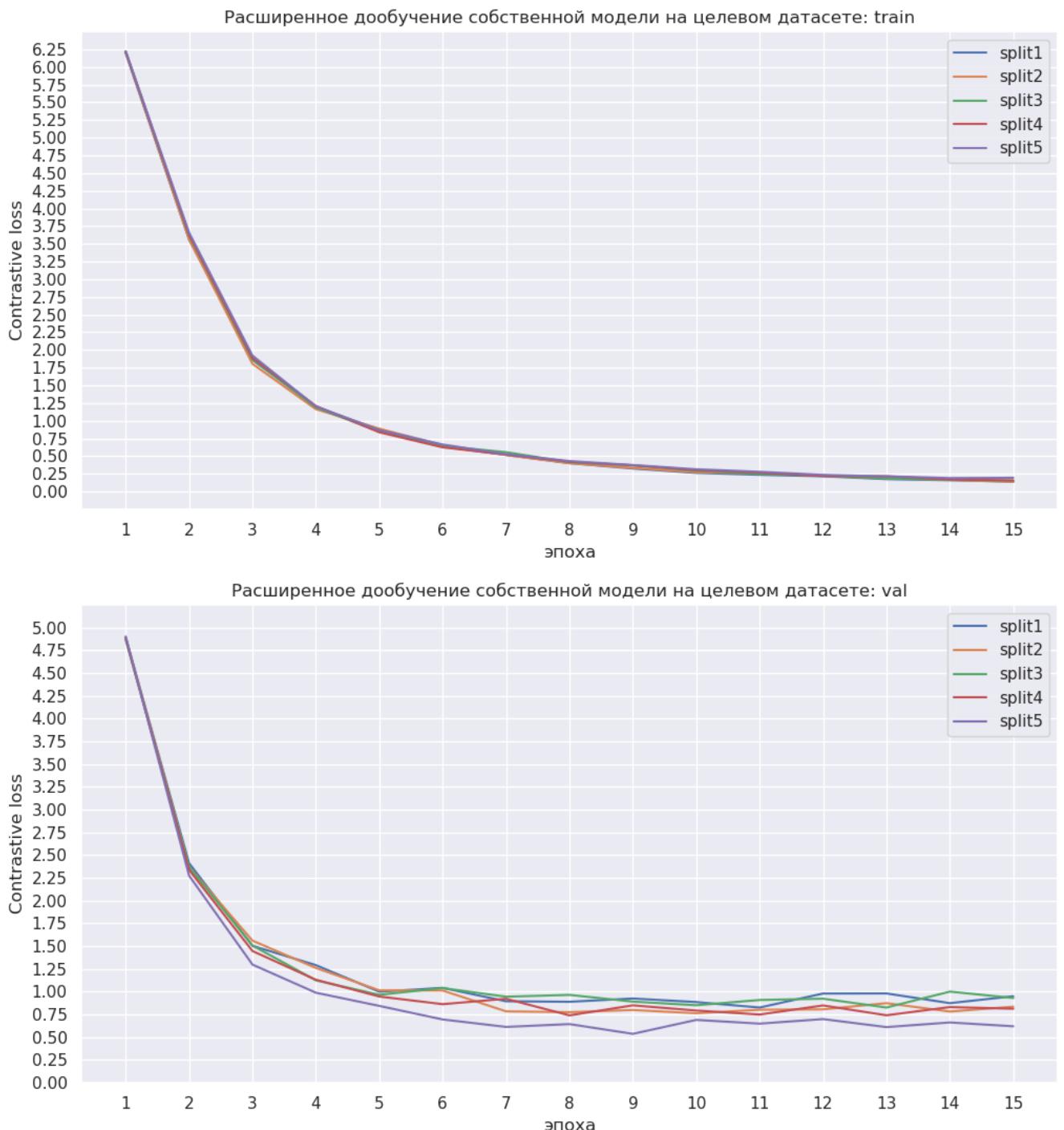


Рисунок 22: Расширенное дообучение собственной модели

Таблица 7: Результаты собственной модели после расширенного дообучения

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучший лосс	эпоха с лучшим лоссом
1	0.802	76.2	0.485	0.453	0.366	0.825	11
2	0.807	72.9	0.498	0.458	0.360	0.762	10
3	0.812	71.4	0.515	0.467	0.392	0.825	13
4	0.799	77.1	0.503	0.463	0.384	0.739	8
5	0.837	58.6	0.486	0.455	0.381	0.536	9
среднее	0.811	71.2	0.497	0.459	0.377	0.737	-

На графиках с рисунка 22 видно, что тенденция снижения функции потерь на тренировочных выборках продолжается на протяжении всех 15 эпох и проявляется одинаково для всех 5 сплитов. На валидационных подвыборках картина несколько иная: сплиты начинают всё больше отличаться друг от друга, а после 13 эпохи снижения функции потерь выглядит незначительным. Вероятно, при попытках дальнейшего дообучения модели начнёт происходить переобучение на тренировочные подвыборки.

По таблице 7 можно заметить, что итоговая accuracy увеличилась на всех сплитеах, кроме 1 и 4, поэтому и среднее качество модели улучшилось. В случае 4 сплита лучшая модель была найдена на эпохе 8 и не изменилась. На сплите 1 качество серьёзно упало, т. е. модель уже деградирует. Возможно, на это всё же повлияли повреждённые кадры. Напомним, что все повреждённые видео попадают в обучение для сплита 1. По последней колонке таблицы 7, становится понятно, что в предыдущем эксперименте моделям не хватило 2-3 эпох до лучшего качества. Это может быть связано с отклонениями от авторской процедуры во время мультимодального предобучения. Поскольку модель после предобучения оказывается несколько хуже, чем предполагалось, ей нужно больше выучить на этапе дообучения и 8 эпох становится недостаточно. Т. к. модель уже начала деградировать на первом сплите, дальнейшее дообучение может навредить итоговому качеству.

Далее рассмотрим получившиеся матрицы ошибок с рисунка 23.

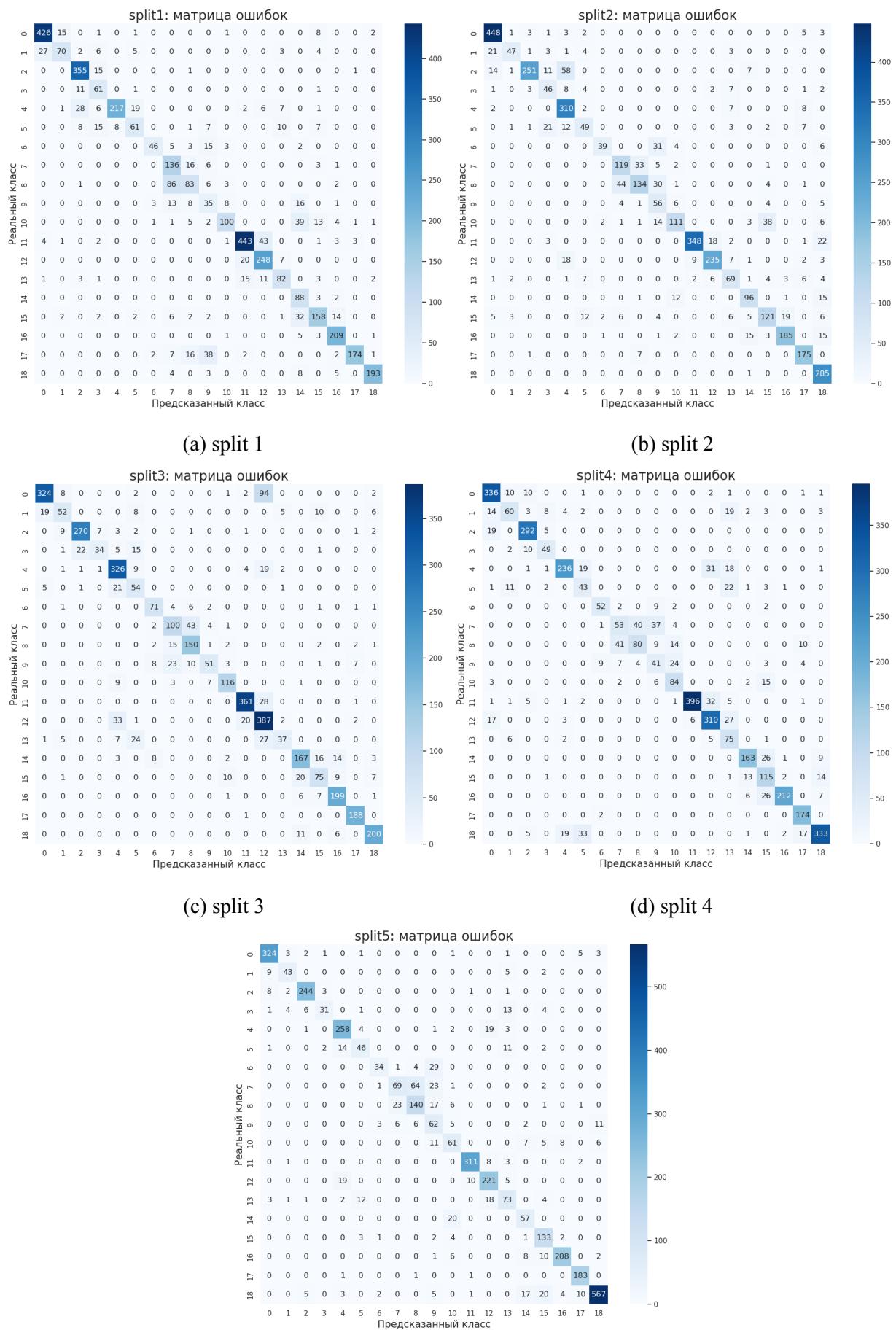


Рисунок 23: Матрицы ошибок при расширенном дообучении собственной модели

На матрицах ошибок после расширенного дообучения видны новые улучшения: классы 12 `cut_cucumber` и 11 `peel_cucumber` можно считать разделёнными. Так же модели почти полностью перестали путать класс 12 `cut_cucumber` и класс 4 `cut_lettuce`. Проблема с классами 12 `cut_cucumber` и 0 `cut_tomato` на сплите 4 осталась нерешённой. Смешение классов 7 `add_vinegar` и 8 `add_oil` незначительно уменьшилось. Класс 9 `add_pepper` стал чуть больше выделяться, но всё же смешивается с классом 7 `add_vinegar` на сплитах 4 и 5, с классом 8 `add_oil` на сплите 2 и с классом 6 `add_salt` на сплитах 2 и 5.

Таким образом получена модель, которая смогла улучшить `baseline` и подтвердить гипотезу о пользе мультимодального предобучения. Гипотеза всё же нуждается в уточнениях: исходя из таблицы 7 и матриц ошибок на рисунке 23, улучшение качества на практике может оказаться небольшим. При этом процесс мультимодального предобучения требует значительно больше ресурсов, чем обучение одномодальной модели или дообучение модели с публично доступными весами. Далее именно последнюю модель, обученную на 15 эпохах, будем считать итоговой для самостоятельного предобучения.

2.9.2 Дообучение модели с общедоступными весами

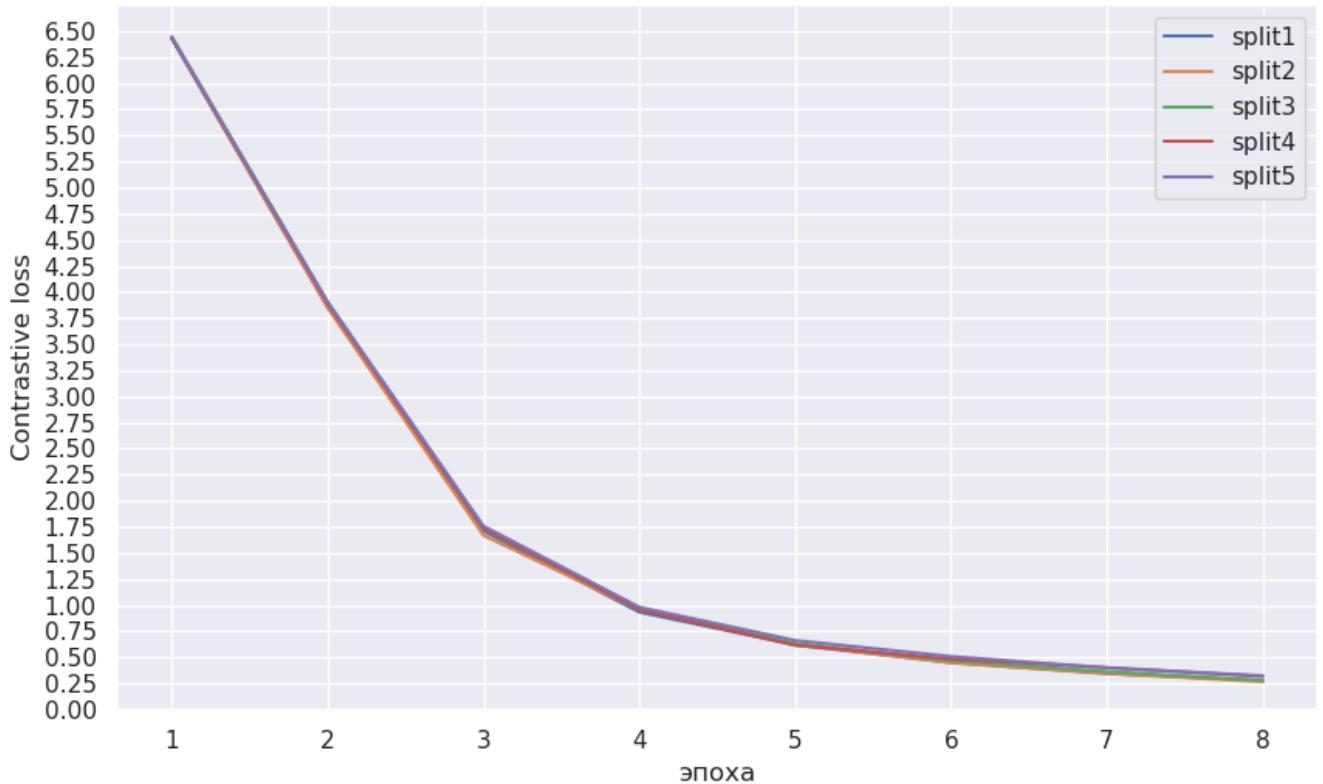
Проведём аналогичные эксперименты по дообучению и с общедоступной моделью. Это позволит лучше понять, как оказались отличия в процедуре мультимодального предобучения, и можно ли предлагать дообучение общедоступной модели как альтернативу для целевого датасета.

Графики дообучения общедоступной модели представлены на рисунке 24, а метрики качества - в таблице 8.

Таблица 8: Результаты дообучения общедоступной модели

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучший лосс	эпоха с лучшим лоссом
1	0.788	82.4	0.549	0.503	0.405	0.830	8
2	0.806	73.7	0.500	0.462	0.401	0.730	7
3	0.801	76.6	0.602	0.543	0.480	0.767	7
4	0.802	75.2	0.562	0.532	0.476	0.765	8
5	0.851	53.2	0.541	0.525	0.449	0.541	7
среднее	0.810	72.2	0.551	0.513	0.442	0.727	-

Дообучение общедоступной модели на целевом датасете: train



Дообучение общедоступной модели на целевом датасете: val

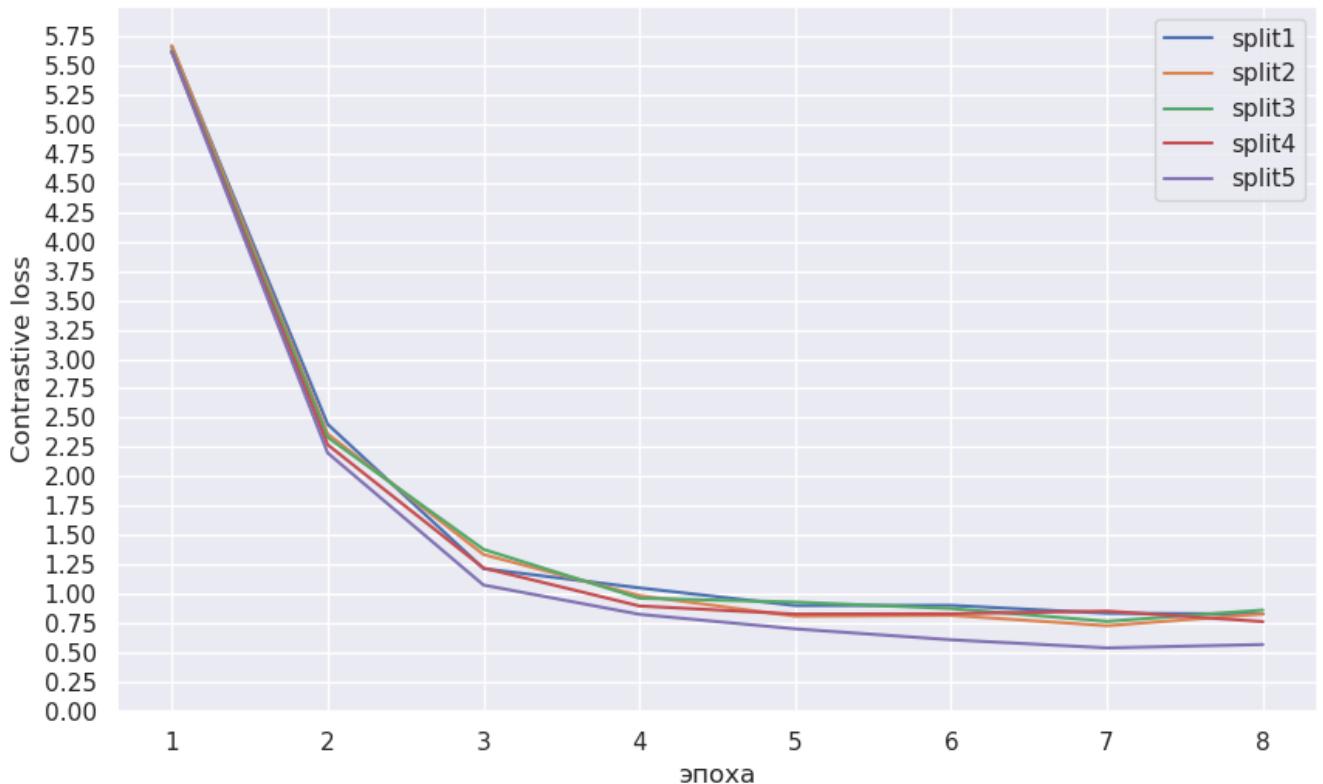


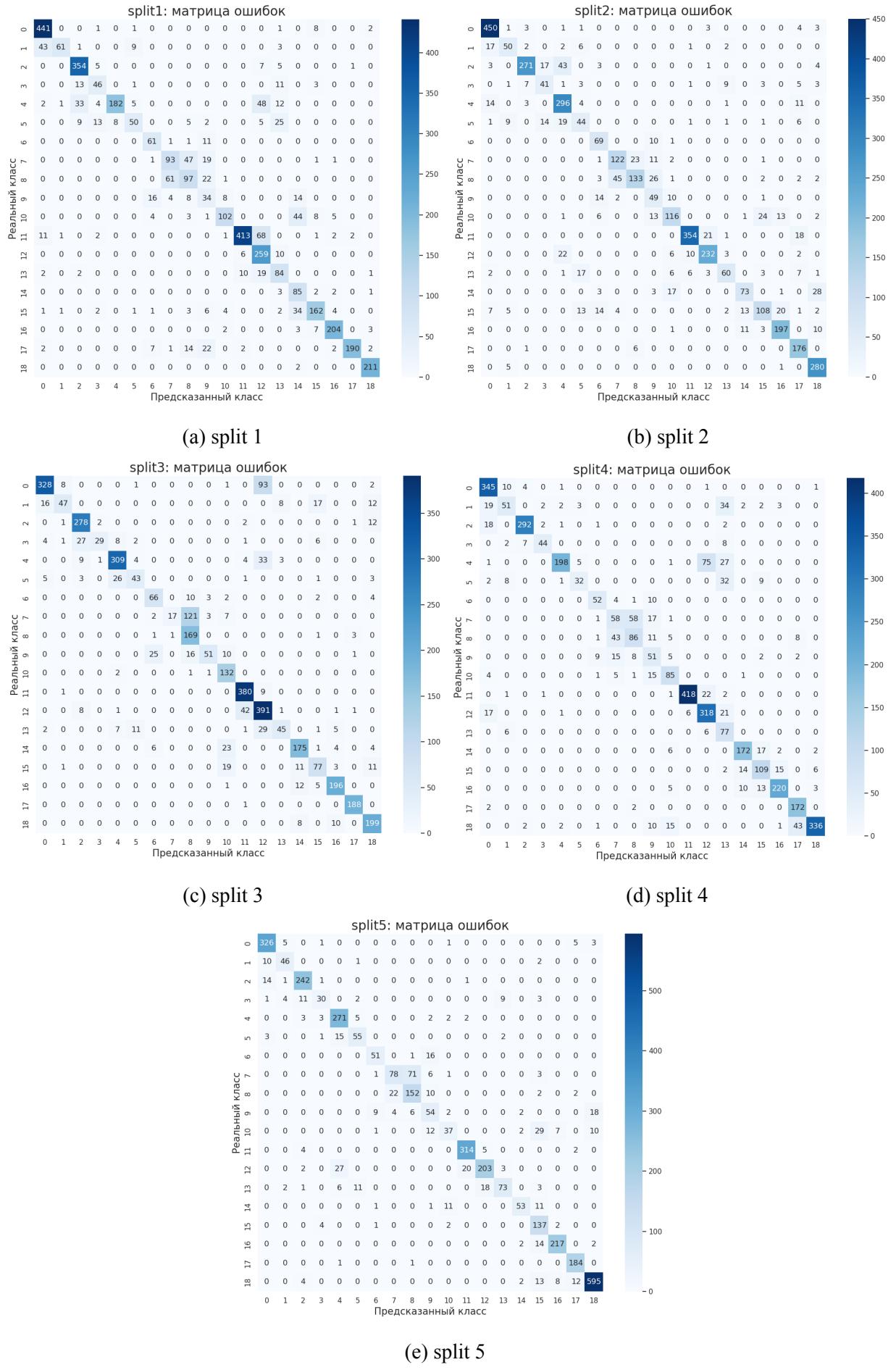
Рисунок 24: Дообучение общедоступной модели

Сравним получившиеся графики с такими же для модели с весами, полученными самостоятельно. Обучение на тренировочной выборке снова получилось стабильным и успеш-

ным для всех сплитов. Поведение модели на валидационных подвыборках стало лучше: единственный сплит, который расходится с остальными - это 5. При этом расхождение происходит в сторону меньшей функции потерь, что означает улучшение качества модели.

По таблице 8 видно, что получившаяся модель стоит наравне с самостоятельно пре- добученной моделью. При этом для её обучения потребовалось намного меньше ресурсов.

Теперь рассмотрим матрицы ошибок для получившейся модели с рисунка 25. В каче- стве проблемного снова можно выделить класс 12 cut_cucumber, который путается с класса- ми 11 peel_cucumber и 4 cut_lettuce. Классы 7 add_vinegar и 8 add_oil тоже смешиваются меж- ду собой. Так же выделяются ошибки в пределах одного сплита. Так класс 12 cut_cucumber снова путается с классом 0 cut_tomato на сплите 3, а класс 14 add_dressing - с классом 10 mix_dressing на сплите 1. Возможно, последнее происходит в силу похожих названий клас- сов и схожести действий, которые за этими названиями стоят. Класс 9 add_rapper отделяется достаточно плохо, существенные ошибки можно найти на сплите 1, где класс был принят за нейтральный 17 action_end, на сплите 2, где класс перепутали с классом 8 add_oil и на сплите 4, где класс смешался с классом 7 add_vinegar.



Далее повторим эксперимент с расширенным дообучением на 15 эпохах. Следует признать, что общедоступной модели оказалось достаточно 8 эпох, чтобы догнать модель, обученную самостоятельно, и побить baseline. Данный эксперимент нужен лишь для проверки, может ли общедоступная модель дообучиться до ещё большего качества. Таким образом для обоих моделей будут проведены одинаковые эксперименты и сравнение можно будет считать беспристрастным.

Расширенный график обучения общедоступной модели изображён на рисунке 26. Итоговые метрики качества находятся в таблице 9.

Таблица 9: Результаты расширенного дообучения общедоступной модели

номер сплита	итоговая accuracy	редакционное расстояние	f1@10	f1@25	f1@50	лучший лосс	эпоха с лучшим лоссом
1	0.813	72.3	0.593	0.552	0.463	0.803	11
2	0.810	72.2	0.540	0.540	0.430	0.693	9
3	0.814	71.1	0.565	0.531	0.463	0.765	9
4	0.802	75.2	0.562	0.532	0.476	0.765	8
5	0.842	57.3	0.579	0.559	0.486	0.502	9
среднее	0.816	69.6	0.568	0.543	0.464	0.706	-

Несмотря на то, что функция потерь на тренировочных данных продолжает падать, дополнительные 7 эпох почти не дают улучшений качества в среднем. На графике для валидационных подвыборки видно, что после 9 эпохи функция потерь на всех сплитах, кроме 1, перестаёт падать. Это можно понять и из последней колонки таблицы 9. Видимо, при полном воспроизведении авторской процедуры, дообучение более 8-9 эпох начинает вредить модели, поэтому авторы VideoCLIP [26] остановились на 8 эпохах в своей процедуре. При сравнении таблицы результатов с таблицей 8 можно заметить, что на 5 сплите модель уже начала деградировать, а на 1 продолжает учиться. Эти изменения нивелируют друг друга при усреднении.

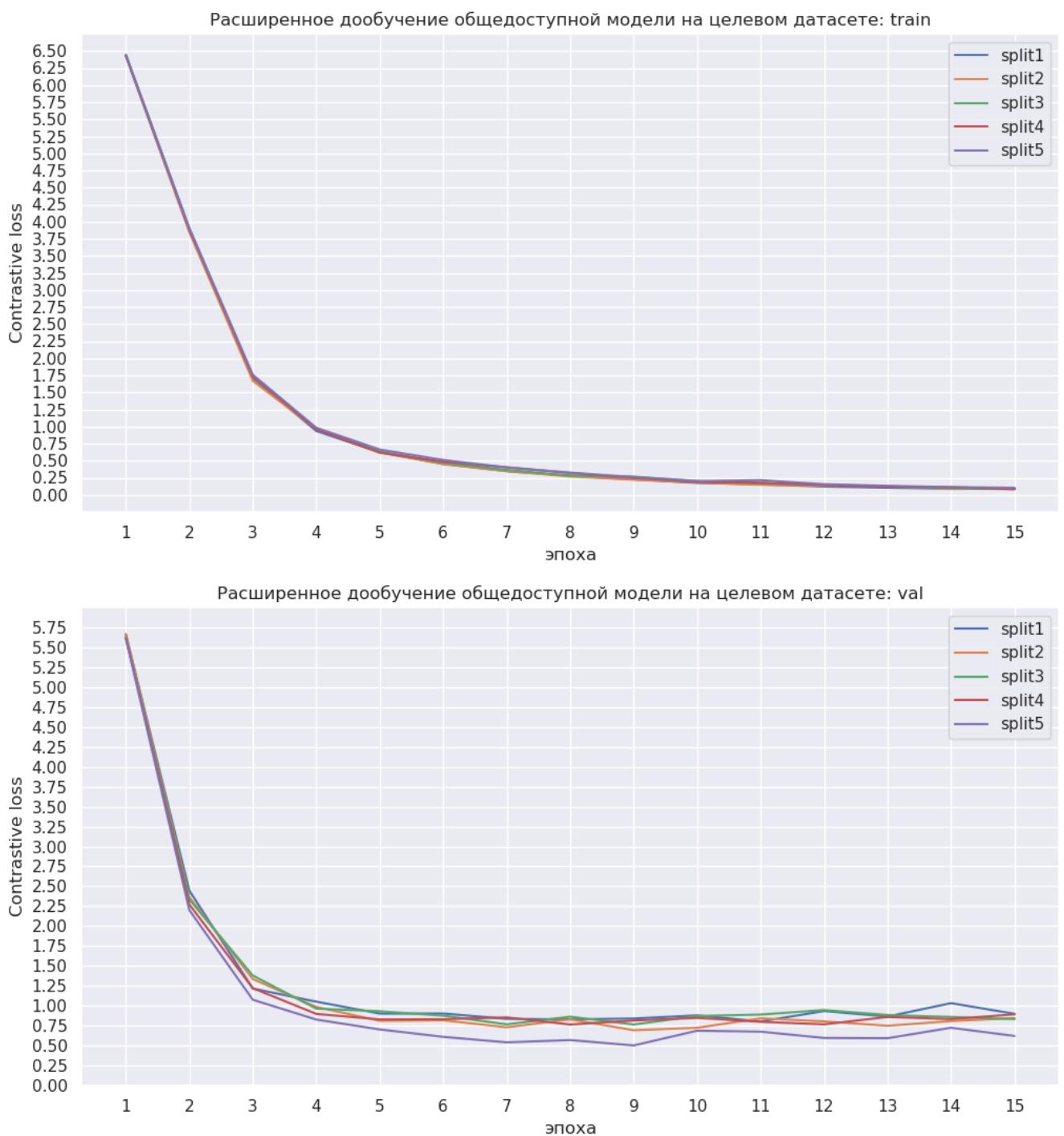
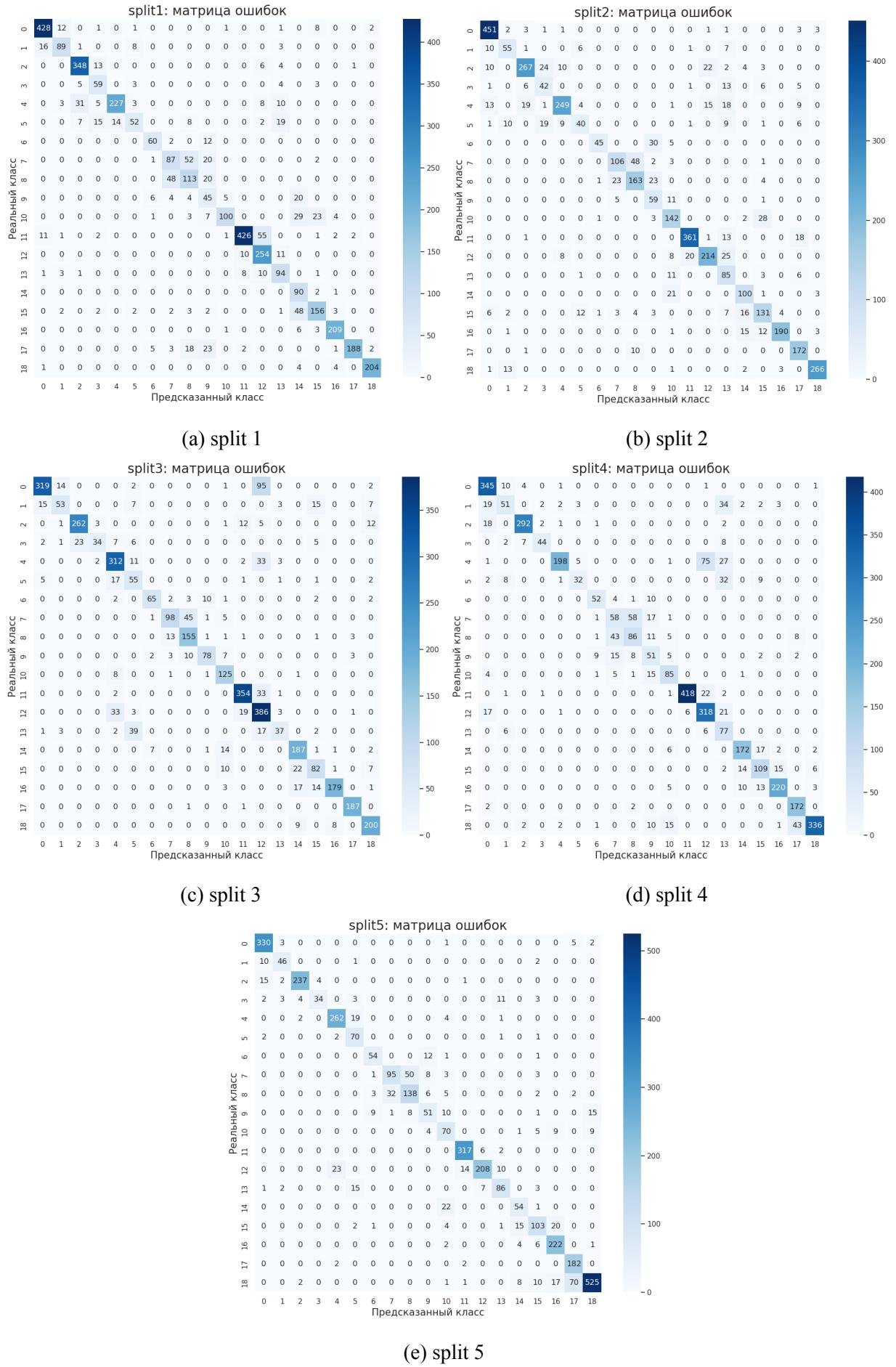


Рисунок 26: Расширенное дообучение общедоступной модели

Проанализируем матрицы ошибок с рисунка 27.



Даже после расширенного дообучения авторской модели не удаётся разделить классы 7 add_vinegar и 8 add_oil на всех сплитах. Надо признать, что дополнительные 7 эпох всё же уменьшили число соответствующих ошибок, но для получения полного разделения классов нужно что-то более серьёзное. Возможно смена процедуры дообучения или даже самой модели. Другой парой смешанных классов являются 12 cut_cucumber и 4 cut_lettuce. Модель путает их на сплитах 3 и 4, до расширенного дообучения к этим сплитам добавлялся ещё сплит 1. Модель научилась различать классы 12 cut_cucumber и 11 peel_cucumber на всех сплитах, кроме первого, что свидетельствует о некотором прогрессе.

3 Заключение

В процессе исследования на целевом датасете 50salads было опробовано несколько методов решения задачи Action Segmentation. Таблица 10 является итоговой сводной таблицей с усреднённым по сплитам качеством для всех подходов.

Таблица 10: Итоговая сводная таблица для всех методов

Метод	Усреднённая по сплитам accuracy
ASFormer по версии официальной статьи [17]	0.856
ASFormer на авторских векторах признаков	0.854
baseline: ASFormer на самостоятельно извлечённых векторах признаков	0.793
VideoCLIP: мультимодальное предобучение, затем zero-shot перенос	0.294
VideoCLIP: общедоступные веса мультимодально предобученной модели, затем zero-shot перенос	0.457
VideoCLIP: мультимодальное предобучение, затем дообучение 8 эпох	0.799
VideoCLIP: мультимодальное предобучение, затем дообучение 15 эпох	0.811
VideoCLIP: общедоступные веса мультимодально предобученной модели, затем дообучение 8 эпох	0.810
VideoCLIP: общедоступные веса мультимодально предобученной модели, затем дообучение 15 эпох	0.816

- В ходе исследований удалось приблизиться к авторскому качеству с помощью нейросети ASFormer. Это позволило получить сильный baseline для последующей работы.
- Воспроизведение полного предобучения мультимодальных моделей остаётся сложной ресурсоёмкой задачей. Даже небольшие отклонения от авторской процедуры могут привести к серьёзным последствиям.
- При сравнении с мультиомодальным VideoCLIP видно, что zero-shot перенос справляется с задачей ActionSegmentation значительно хуже. Вероятно, это связано с выбором достаточно специфического датасета, в котором есть близкие классы (cut cucumber и peel cucumber) и классы, которые можно условно отнести к background (action start и action end).

- После дообучения и на общедоступных весах и на весах, полученных самостоятельно в ходе мультимодального предобучения, получаются модели, способные соперничать с одномодальной специально обученной на целевой датасет архитектурой. Это говорит о способности Трансформер-подобных архитектур адаптироваться к новым задачам.
- Если по каким-то причинам авторское мультимодальное предобучение не удаётся повторить полностью, может помочь разумная адаптация процедуры дообучения.
- Гипотеза об улучшении качества модели за счёт мультимодального предобучения подтвердилась, но следует уточнить, что улучшения могут быть незначительными по сравнению с затраченными ресурсами, поэтому на практике следует применять данный метод с осторожностью.
- В случае нехватки ресурсов разумной альтернативой остаётся использование общедоступных предобученных весов.

Список литературы

1. Attention Is All You Need / A. Vaswani [и др.] // CoRR. — 2017. — T. abs/1706.03762. — DOI: <https://doi.org/10.48550/arXiv.1706.03762>. — arXiv: 1706.03762. — URL: <http://arxiv.org/abs/1706.03762>.
2. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin [и др.] // CoRR. — 2018. — T. abs/1810.04805. — DOI: <https://doi.org/10.48550/arXiv.1810.04805>. — arXiv: 1810.04805. — URL: <http://arxiv.org/abs/1810.04805>.
3. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale / A. Dosovitskiy [и др.] // CoRR. — 2020. — T. abs/2010.11929. — DOI: <https://doi.org/10.48550/arXiv.2010.11929>. — arXiv: 2010.11929. — URL: <https://arxiv.org/abs/2010.11929>.
4. CvT: Introducing Convolutions to Vision Transformers / H. Wu [и др.] // CoRR. — 2021. — T. abs/2103.15808. — DOI: <https://doi.org/10.48550/arXiv.2103.15808>. — arXiv: 2103.15808. — URL: <https://arxiv.org/abs/2103.15808>.
5. Image Transformer / N. Parmar [и др.] // CoRR. — 2018. — T. abs/1802.05751. — DOI: <https://doi.org/10.48550/arXiv.1802.05751>. — arXiv: 1802.05751. — URL: <http://arxiv.org/abs/1802.05751>.
6. Video Swin Transformer / Z. Liu [и др.] // CoRR. — 2021. — T. abs/2106.13230. — DOI: <https://doi.org/10.48550/arXiv.2106.13230>. — arXiv: 2106.13230. — URL: <https://arxiv.org/abs/2106.13230>.
7. Video Action Transformer Network / R. Girdhar [и др.] // CoRR. — 2018. — T. abs/1812.02707. — DOI: <https://doi.org/10.48550/arXiv.1812.02707>. — arXiv: 1812.02707. — URL: <http://arxiv.org/abs/1812.02707>.
8. ViViT: A Video Vision Transformer / A. Arnab [и др.] // CoRR. — 2021. — T. abs/2103.15691. — DOI: <https://doi.org/10.48550/arXiv.2103.15691>. — arXiv: 2103.15691. — URL: <https://arxiv.org/abs/2103.15691>.
9. Time Is MattEr: Temporal Self-supervision for Video Transformers / S. Yun [и др.]. — 2022. — DOI: <https://doi.org/10.48550/arXiv.2207.09067>. — arXiv: 2207.09067 [cs.CV]. — URL: <https://arxiv.org/abs/2207.09067>.

10. VideoBERT: A Joint Model for Video and Language Representation Learning / C. Sun [и др.] // CoRR. — 2019. — T. abs/1904.01766. — DOI: <https://doi.org/10.48550/arXiv.1904.01766>. — arXiv: 1904.01766. — URL: <http://arxiv.org/abs/1904.01766>.
11. Zhu L., Yang Y. ActBERT: Learning Global-Local Video-Text Representations // CoRR. — 2020. — T. abs/2011.07231. — DOI: <https://doi.org/10.48550/arXiv.2011.07231>. — arXiv: 2011.07231. — URL: <https://arxiv.org/abs/2011.07231>.
12. Zero-Shot Temporal Action Detection via Vision-Language Prompting / S. Nag [и др.]. — 2022. — DOI: <https://doi.org/10.48550/arXiv.2207.08184>. — arXiv: 2207.08184 [cs.CV]. — URL: <https://arxiv.org/abs/2207.08184>.
13. fairseq: A Fast, Extensible Toolkit for Sequence Modeling / M. Ott [и др.] // CoRR. — 2019. — T. abs/1904.01038. — DOI: <https://doi.org/10.48550/arXiv.1904.01038>. — arXiv: 1904.01038. — URL: <http://arxiv.org/abs/1904.01038>.
14. The Faiss library / M. Douze [и др.]. — 2024. — DOI: <https://doi.org/10.48550/arXiv.2401.08281>. — arXiv: 2401.08281 [cs.LG]. — URL: <https://arxiv.org/abs/2401.08281>.
15. Stein S., McKenna S. J. Combining embedded accelerometers with computer vision for recognizing food preparation activities // Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing. — 2013. — DOI: <https://doi.org/10.1145/2493432.2493482>. — URL: <https://api.semanticscholar.org/CorpusID:2333743>.
16. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips / A. Miech [и др.] // CoRR. — 2019. — T. abs/1906.03327. — DOI: <https://doi.org/10.48550/arXiv.1906.03327>. — arXiv: 1906.03327. — URL: <http://arxiv.org/abs/1906.03327>.
17. Yi F., Wen H., Jiang T. ASFormer: Transformer for Action Segmentation // CoRR. — 2021. — T. abs/2110.08568. — DOI: <https://doi.org/10.48550/arXiv.2110.08568>. — arXiv: 2110.08568. — URL: <https://arxiv.org/abs/2110.08568>.

18. *Carreira J., Zisserman A.* Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset // CoRR. — 2017. — T. abs/1705.07750. — DOI: <https://doi.org/10.48550/arXiv.1705.07750>. — arXiv: 1705.07750. — URL: <http://arxiv.org/abs/1705.07750>.
19. *Ioffe S., Szegedy C.* Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // CoRR. — 2015. — T. abs/1502.03167. — DOI: <https://doi.org/10.48550/arXiv.1502.03167>. — arXiv: 1502.03167. — URL: <http://arxiv.org/abs/1502.03167>.
20. ImageNet: a Large-Scale Hierarchical Image Database / J. Deng [и др.] //. — 06.2009. — С. 248—255. — DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
21. The Kinetics Human Action Video Dataset / W. Kay [и др.] // CoRR. — 2017. — T. abs/1705.06950. — DOI: <https://doi.org/10.48550/arXiv.1705.06950>. — arXiv: 1705.06950. — URL: <http://arxiv.org/abs/1705.06950>.
22. *Farha Y. A., Gall J.* MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation // CoRR. — 2019. — T. abs/1903.01945. — DOI: <https://doi.org/10.48550/arXiv.1903.01945>. — arXiv: 1903.01945. — URL: <http://arxiv.org/abs/1903.01945>.
23. PyTorchVideo: A Deep Learning Library for Video Understanding / H. Fan [и др.] // CoRR. — 2021. — T. abs/2111.09887. — DOI: <https://doi.org/10.48550/arXiv.2111.09887>. — arXiv: 2111.09887. — URL: <https://arxiv.org/abs/2111.09887>.
24. HierVL: Learning Hierarchical Video-Language Embeddings / K. Ashutosh [и др.]. — 2023. — DOI: <https://doi.org/10.48550/arXiv.2301.02311>. — arXiv: 2301.02311 [cs.CV]. — URL: <https://arxiv.org/abs/2301.02311>.
25. Egocentric Video-Language Pretraining / K. Q. Lin [и др.]. — 2022. — DOI: <https://doi.org/10.48550/arXiv.2206.01670>. — arXiv: 2206.01670 [cs.CV]. — URL: <https://arxiv.org/abs/2206.01670>.
26. VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding / H. Xu [и др.] // CoRR. — 2021. — T. abs/2109.14084. — DOI: <https://doi.org/10.48550/arXiv.2109.14084>. — arXiv: 2109.14084. — URL: <https://arxiv.org/abs/2109.14084>.

27. COIN: A Large-scale Dataset for Comprehensive Instructional Video Analysis / Y. Tang [и др.] // CoRR. — 2019. — T. abs/1903.02874. — DOI: <https://doi.org/10.48550/arXiv.1903.02874>. — arXiv: 1903.02874. — URL: <http://arxiv.org/abs/1903.02874>.
28. Rethinking Spatiotemporal Feature Learning For Video Understanding / S. Xie [и др.] // CoRR. — 2017. — T. abs/1712.04851. — DOI: <https://doi.org/10.48550/arXiv.1712.04851>. — arXiv: 1712.04851. — URL: <http://arxiv.org/abs/1712.04851>.
29. End-to-End Learning of Visual Representations from Uncurated Instructional Videos / A. Miech [и др.] // CoRR. — 2019. — T. abs/1912.06430. — DOI: <https://doi.org/10.48550/arXiv.1912.06430>. — arXiv: 1912.06430. — URL: <http://arxiv.org/abs/1912.06430>.
30. Efficient Estimation of Word Representations in Vector Space / T. Mikolov [и др.]. — 2013. — DOI: <https://doi.org/10.48550/arXiv.1301.3781>. — arXiv: 1301.3781 [cs.CL]. — URL: <https://arxiv.org/abs/1301.3781>.
31. Zhou L., Xu C., Corso J. J. ProcNets: Learning to Segment Procedures in Untrimmed and Unconstrained Videos // CoRR. — 2017. — T. abs/1703.09788. — DOI: <https://doi.org/10.48550/arXiv.1703.09788>. — arXiv: 1703.09788. — URL: <http://arxiv.org/abs/1703.09788>.