DOI: https://doi.org/10.30898/1684-1719.2021.11.15
УДК: 681.3, 004.8

ПРИМЕНЕНИЕ СВЁРТОЧНЫХ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ НЕКОТОРЫХ ЗАДАЧ АНАЛИЗА ТРАЕКТОРНЫХ ДАННЫХ

М. И. Костючек, А. В. Макаренко

**Институт проблем управления им. В. А. Трапезникова Российской академии наук,
117997, Москва, ул. Профсоюзная, д. 65**

Статья поступила в редакцию 30 октября 2021 г.

Аннотация. Рассматривается применение свёрточных глубоких нейронных сетей для решения задачи классификации и распутывания траекторий. Объекты классифицируются по кинематическим характеристикам. Для задачи классификации разработана модель типа Res-Net с функцией потерь перекрёстная энтропия. Нам удалось добиться качество классификации 0.883 по F_1-метрике. Показано, что лучше всего модель классифицирует объекты по скорости, при этом модель с высокой точностью отличает объекты от помех. Модель для классификации протестирована на устойчивость к шуму, который добавляется к данным в виде случайных точек, либо некоторые точки траектории убираются. Модель обучена на данных с десятью шумовыми точками, и протестирована на данных с большим числом шумовых точек. Показано, что качество классификации уменьшается «гладко» при увеличении количества шумовых точек, без резких скачков. Траектории объектов для задачи классификации могут иметь разрывы. Распутываются траектории двух объектов. Для задачи распутывания разработана модель типа UNet, с функцией потерь в виде комбинации перекрёстной энтропии и Tversky loss. Также разработанная модель возвращает оценку скорости объекта на основе скрытых признаков, для учёта динамики. Для задачи распутывания нам удалось добиться качество

классификации 0.911 по F_1-метрике. Модели обучены и протестированы на синтетическом наборе данных.

Ключевые слова: классификация объектов, распутывание траекторий, глубокое обучение, синтетический набор данных, свёрточная нейронная сеть.

Abstract. The application of convolutional deep neural networks for solving the problem of classification and untangling of trajectories is considered. Objects are classified by their cinematic characteristics. Res-Net type model with the cross-entropy loss function has been developed for the classification problem. We have achieved 0.883 classification quality according to the F_1-metric. The model classifies objects by speed better than by other cinematic characteristics. The model distinguishes objects from noise with high accuracy. The model for classification is tested for resistance to noise. Noise is added to the data in the form of random points, or some points of the trajectory are removed. The model is trained on data with ten noise points and tested on data with a large number of noise points. Classification quality decreases "smoothly" with an increase in the number of noise points, without sharp jumps. Object trajectories for classification can have break points. The trajectories of two objects are untangled. For the untangling problem, a UNet-type model has been developed. Loss function in the form of a combination of cross-entropy and Tversky loss has been used. Also, the developed model returns an estimate of the speed of an object based on hidden features. It was used to account for the dynamics. We have achieved 0.911 classification quality according to the F_1-metric for the untangling problem. The models are trained and tested on a synthetic dataset.

Key words: object classification, trajectory untangling, deep learning, synthetic dataset, convolutional neural network.

Введение.

На данный момент для обработки траекторных данных используется множество методов, наиболее популярным из которых является марковская теория оценивания и фильтрации [1, 2, 3]. Минусы этого метода в том, что он требует много информации об объекте и является очень ресурсоёмким, особенно

если задача является нелинейной. Большинство нелинейных задач данная теория решает путём ограничений или линеаризации, что существенно ограничивает общность задачи. Более того, описанные в данных монографиях алгоритмы хорошо показывают результаты только в условия гауссовского шума, что также ограничивает их применение. А в случае не гауссовского шума требуется использовать более сложные алгоритмы, которые очень ресурсоёмки. Чтобы избавится от многих проблем данных алгоритмов, мы решили воспользоваться методами глубокого обучения.

Идея использования нейронных сетей в задачах обработки траекторных данных не нова. В книге [3] описано применение полносвязных нейронных сетей без скрытого слоя или с одним скрытым слоем в задачах фильтрации. Показано, что нейронная сеть без скрытого слоя в задаче линейной фильтрации и нейронная сеть с одним скрытым слоем в задаче нелинейной фильтрации реализует работу фильтра Калмана. В книге [4] описано применение нейросетевых моделей многослойной полносвязной сети и ассоциативной памяти Хопфилда для решения задач радиолокации, таких как фильтрация сигнала, обнаружение сигналов и сопровождения групп целей. Показаны проблемы, с которыми сталкиваются данные подходы, и даны некоторые методы решения этих проблем. Но стоит отметить, что на данный момент существует большое количество различных архитектур нейронных сетей, которые работают намного эффективнее полносвязных сетей и моделей ассоциативной памяти Хопфилда. К тому же у новых архитектур не возникают некоторые проблемы (меньшая склонность к переобучению, инвариантность, относительно сдвигов), характерные для старых архитектур.

Глубокие нейронные сети высоко эффективны в обработке данных с ярко выраженными паттернами [5] и в отличии от многослойных персептронов они показывают хороший результат в обработке случайных данных [6, 7].

В статье [8] описан метод классификации летательных объектов по наблюдаемой траектории на основе каскада многослойных нейронных сетей прямого распространения (3 класса объектов и 1 класс неопознанных объектов).

Система распознавания состоит из двух блоков, первый блок осуществляет локальную во времени классификацию летательного аппарата на основе текущего значения входного вектора. Решения первого классификатора принимаются независимо в каждый момент времени. Для учета предыстории принятых в отношении данного летательного аппарата решений служит второй классификатор, он обеспечивает инерционность принятия решений. Предполагается, что данные зашумлены, причем шумы в различные моменты времени независимы.

Значительная часть современной литературы, связанной с анализом траекторных данных, посвящена прогнозированию движения транспортного средства в условиях окружающих его других транспортных средств. В работе [9] предложен метод прогнозирования траектории движения транспортного средства, основанный на архитектуре нейронной сети LSTM кодер-декодер. В предлагаемой системе используется кодер LSTM для анализа прошлых измерений датчика и декодер LSTM для формирования будущих образцов траекторий на основе выходных данных кодера. Эта структура производит k наиболее вероятных кандидатов траектории по карте сетки занятости. В исследовании [10] разрабатывается основа для классификации активности наблюдаемых на дорогах транспортных средств с использованием трехмерных траекторных сигналов (LIDAR, GPS, IMU) и модели с длительной кратковременной памятью (LSTM). Для оценки качества работы классификатора используется метрика качества accuracy. При использовании 2-х классов (две траектории) accuracy – 0.8, но при использовании 8 классов качество сильно падает, accuracy – 0.5. В работе [11] на основе рекуррентных и свёрточных нейронных сетей разработан метод, прогнозирующий движение объекта. В этом методе сверхточные нейронные сети извлекают визуальные особенности целевых объектов. Модель долговременной-кратковременной памяти (рекуррентной) использует аннотированные траектории, а также последовательную визуальную информацию, которая включает отслеживаемые

объекты и центральные местоположения целевого объекта, для прогнозирования движения.

Исследователи из Калифорнийского университета разработали метод [12], предсказывающий траекторию полета самолета в зависимости от погодных условий. На вход нейронной сети подается запланированная траектория самолета, а на выходе получаем фактическую траекторию. В этой работе используется рекуррентная нейросетевая структура кодера-декодера для прогнозирования траекторий воздушных судов.

Обычно временные последовательности анализируют с помощью рекуррентных нейронных сетей, но такие модели очень сложно оптимизировать [5]. В связи с этим возникает идея использовать свёрточную нейронную сеть для решения данной задачи. Такие нейронные сети имеют много достоинств: они легко обучаются, имеют хорошее быстродействие, хорошо обобщают и заточены на решение геометрических задач.

В нашей работе предлагаются новые методы решения задач классификации наблюдаемых объектов и распутывания траекторий, основанные на технологиях глубокого обучения. Возможность обработки случайных данных с помощью глубоких нейронных сетей [6, 7] позволяет использовать их как эффективный фильтр шумов без предварительной гипотезы о распределении шума. Такой подход, в отличии от классических методов [1, 2, 3] не требует знания математической модели наблюдаемых объектов, к тому же ожидается, что данный подход повысит точность решений и устойчивость, по сравнению с классическими методами.

В данной работе объекты классифицируются по кинематическим характеристикам на основе их наблюдаемых траекторий в трёхмерном пространстве. На вход классификатора подаётся массив точек траектории наблюдаемого объекта в трёхмерном пространстве. При этом траектория объекта может иметь разрыв, если объект наблюдения вышел из зоны действия радара и вернулся. Если в некоторый момент времени объект не наблюдался, то в массив

записываются нули. Объекты делятся на 12 классов, к тому же добавляется ещё один класс "не траекторий".

Объекты делятся по максимальным значениям скорости, центростремительному ускорению и по зависимости натурального параметра l от времени. Объекты делятся по скорости на низкоскоростные и высокоскоростные, по центростремительному ускорению на слабоманёвренные и сильноманёвренные. Объект считается низкоскоростным (высокоскоростным), если его максимальная норма вектора скорости V_m в области действия радара меньше $V_1 = 0.85$ (больше $V_2 = 0.95$), и слабоманёвренным (сильноманёвренным), если его максимальная норма вектора центростремительного ускорения a_m в области действия радара меньше $a_1 = 15.5$ (больше $a_2 = 16.5$). По параметризации на движущиеся равномерно, для которых $\ddot{l}(t) = 0$, движущиеся равноускорено, $\ddot{l}(t) = const \neq 0$, и движущиеся не равноускорено, $\ddot{l}(t) = f(t) \neq const$, $l(t)$ – зависимость натурального параметра траектории от времени. Получаем всего 12 классов объектов (2 метакласса по скорости, 2 метакласса по центростремительному ускорению, 3 метакласса по параметризации) и один класс не траекторий. Каждая не траектория представляет собой случайный набор точек в определённой области. Классы описаны в таблице 1.

Для тестирования модели классификации на устойчивость к шуму была использована та же модель. Она обучалась на данных, в которые добавляли до десяти шумовых точек. Количество шумовых точек для каждой траектории выбиралось случайным образом. В качестве шума в данные добавлялись случайные точки в трехмерном пространстве с равномерным распределением в случайный момент времени, если шум появился одновременно с объектом наблюдения, то точка траектории объекта заменяется на шум. Затем эта модель тестировалась на данных с шумовыми точками от 0 до 70.

Помимо классификации решалась задача распутывания траекторий. Здесь мы классифицируем точки по объектам, к которым они принадлежат.

Пусть нам дан набор точек нескольких объектов, но нам не известно к какому объекты принадлежит та или иная точка, но нам известно количество объектов и начальные точки этих объектов. В данной работе распутываются траектории двух объектов.

Таблица 1. Описание классов для задачи классификации. V_m, a_m – максимальное значение скорости и центростремительного ускорения наблюдаемой кривой, соответственно.

Класс	Описание
0	Равномерное движение, $V_m \leq V_1$, $a_m \leq a_1$
1	Равномерное движение, $V_m \leq V_1$, $a_m \geq a_2$
2	Равномерное движение, $V_m \geq V_2$, $a_m \leq a_1$
3	Равномерное движение, $V_m \geq V_2$, $a_m \geq a_2$
4	Равноускоренное движение, $V_m \leq V_1$, $a_m \leq a_1$
5	Равноускоренное движение, $V_m \leq V_1$, $a_m \geq a_2$
6	Равноускоренное движение, $V_m \geq V_2$, $a_m \leq a_1$
7	Равноускоренное движение, $V_m \geq V_2$, $a_m \geq a_2$
8	Не равноускоренное движение, $V_m \leq V_1$, $a_m \leq a_1$
9	Не равноускоренное движение, $V_m \leq V_1$, $a_m \geq a_2$
10	Не равноускоренное движение, $V_m \geq V_2$, $a_m \leq a_1$
11	Не равноускоренное движение, $V_m \geq V_2$, $a_m \geq a_2$
12	Помехи

Данная работа является продолжение работы [13], в которой решалась только задача классификации. По сравнению с работой [13], в этой работе изменена модель, улучшено качество классификации и рассмотрена устойчивость к шуму.

1. Задача классификации

1.1. Формирование набора данных

Для формирования обучающей и тестовой выборки для задачи классификации был использован алгоритм генерации случайных траекторий, основанный на гладкой интерполяции окружностями [14]. Во время генерации

траектории форма кривой и зависимость натурального параметра от времени формируются отдельно друг от друга. Данный алгоритм позволяет легко накладывать ограничения на форму генерируемых кривых в целом, а также на их кинематические характеристики. Получаемые траектории без труда могут быть параметризованны от натурального параметра и времени.

Для параметризации натурального параметра от времени была использована параметризация $l(t) = k_1 t, \quad k_1 = RND[0.78;\ 1.02]$, параметризация для равноускоренного движения – $l(t) = k_1 t + \frac{t^2}{2k_2}, \ k_1 = RND[0.08;\ 0.12], \ k_2 = RND[9;\ 11]$, не равноускоренное движение было параметризовано двумя способами: $l(t) = k_1 t + \frac{t^3}{3k_2}, k_1 = RND[0.78;\ 0.82], k_2 = RND[185;\ 215]$ и $l(t) = \frac{t}{k_1} - 0.9k_2 exp\left(-\frac{t}{k_2}\right) + 0.9k_2, k_1 = RND[1.3;\ 1.7], k_2 = RND[2.5;\ 3.5]$. Здесь $RND[a, b]$ – число, сгенерированное генератором псевдослучайных чисел из равномерного распределения на отрезке $[a, b]$.

После генерации траектории на ней определялись точки с шагом 0.01 по времени, затем строился куб с центром в случайной точке траектории, сдвинутой на случайный вектор с координатами $RND[-0.1, 0.1]$, длина ребра куба составляла 0.5. Наблюдаемыми точками траектории считались точки, которые попали в куб. Пример траектории из данных для обучения и тестирования показан на рис. 1. Помехи формируются из случайного набора точек в кубе с длиной ребра 0.5.

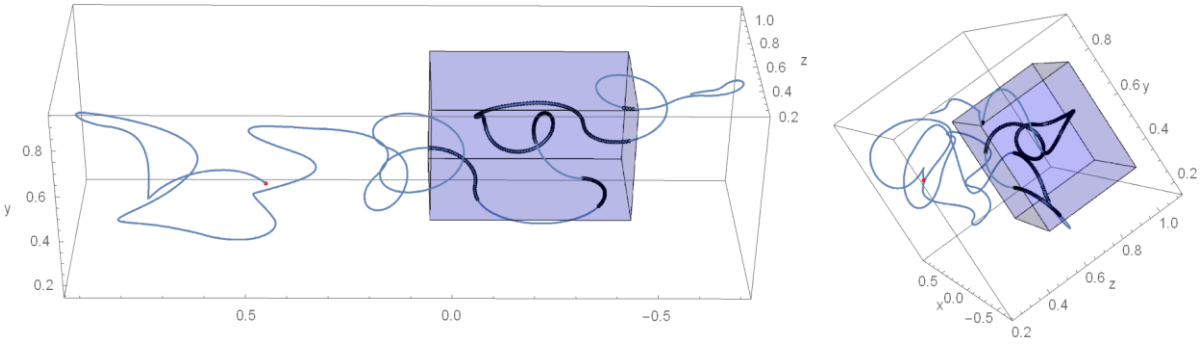

Рис. 1. Пример траектории для обучения и тестирования. Синей линией нарисован вся траектория, сиреневым кубом выделена область действия радара, чёрными точками отмечены точки траектории, которые попали в область действия радара, красной точкой отмечена начальная точка траектории. На обеих рисунках изображена одна траектория с разных ракурсов.

Для обучения было использовано по 15 000 траекторий каждого класса, всего 195 000 траекторий. Для тестирования – 6 500 траекторий каждого класса, всего 84 000 траектории. Множества траекторий для обучения и тестирования не пересекаются.

1.2.Архитектора нейронной сети

Для решения задачи классификации была использована свёрточная нейронная сеть типа ResNet [16] с одномерными свёрточными слоями и слоями объединения. Такая архитектура позволяет строить существенно более глубокие модели, что способствует лучшему извлечению скрытых признаков данных, и при этом модуль меньше склонна к переобучению, в отличии от моделей такой же глубины, но других архитектур. Также полученные модели имеют меньше обучаемых параметров, чем аналогичные модели других архитектур.

Сначала к входным данным применяется свёрточный слой с фильтром размера 7 и слой максимального объединения (англ. max pooling). Далее данные подаются на остаточные (англ. residual) слои. Всего 7 остаточных слоёв. Размер фильтра каждой свёртки в остаточном слое равен 3. В первом остаточном слое 16 фильтров. После каждого второго остаточного слоя применяется операция максимального объединения и количество фильтров последующего остаточного слоя увеличивается в 2 раза. После остаточных слоёв были использованы слой глобального объединения [16] и два полносвязных слоя. Всего в модели 184 653 весов. Для ускорения обучения была использована пакетная нормализация (англ.

batch norm) [17]. Для борьбы с переобучением была использована техника прореживания с параметром 0.5 (англ. dropout) [18]. После каждого свёрточного слоя использовалась функция активации гиперболический тангенс, после первого полносвязного слоя используется функция активации сигмоида, после второго – функция softmax.

На вход нейронной сети подавался массив точек траектории. Длина каждого массива равна 1415 измерений. Один элемент массива соответствует одной точки по времени. Если в заданный момент времени точка траектории не наблюдалась, то соответствующий элемент массива приравнивался нулю.

Модель возвращает вектор размера 13. Элемент вектора равен вероятности принадлежности траектории к соответствующему классу.

1.3.Обучение и тестирование

Для подбора значения весов модели использовалась функция потерь перекрёстная энтропия. В качестве оптимизатора был выбран стохастический градиентный спуск с импульсом [19], L_2-регуляризацией и экспоненциально уменьшающимся темпом обучения. Величина импульса составляла 0.9, коэффициент члена L_2-регуляризации равнялся 10^{-4}, начальный темп обучения 10^{-2}. Темп обучения уменьшался в 10 раз либо если прошло 100 эпох с момента прошлого изменения, либо если функция потерь не уменьшалась 20 эпох на величину больше 10^{-4}.

Веса модели подбираются за счёт минимизации перекрёстной энтропии

$$L = -\frac{1}{N}\sum_{j=1}^{M}\sum_{i=1}^{N} p_j^i \, lnq_j^i,$$

где M – размер обучающего множества, либо размер пакета; N – количество классов; p_j^i – маркер i-го класса j-го экземпляра данных, он равен 1, если j-й экземпляр имеет i-й класс, и 0 в противном случае; q_j^i – вероятность принадлежности j-го экземпляра к i-му класс, возвращаемый моделью.

Модель обучалась по пакетам размера 128, всего пакетов 1 524. Модель обучалась 500 эпох.

Архитектура нейронной сети написана с помощью библиотеки PyTorch на языке программирования Python.

График значений функции потерь для обучающих данных изображён на рис. 2. Наименьшего значения функция потерь достигла на 481-й эпохе, которое составило 0.245.

На рис. 3 изображён график значений функции потерь по эпохам для тестовых данных. Видно, что при большом темпе обучения значение функции потерь сильно осциллирует, но далее, когда темп обучения становится меньше, значение функции потерь стабилизируется и плавно уменьшается. Среднее значение функции потерь достигает минимального значения на 467-й эпохе, оно составляет 0.337, при этом на тренировочных данных среднее значение функции потерь равно 0.247. Это значение не сильно отличается от минимального среднего значения функции потерь для тренировочных данных.

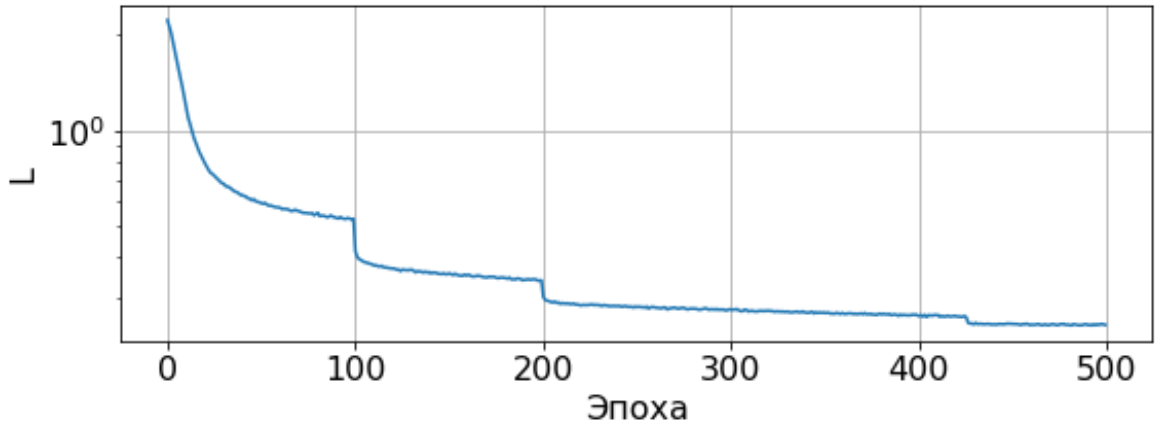

Рис. 2. График зависимости значения функции потерь для обучающих данных от номера эпохи. По оси ординат логарифмический масштаб.

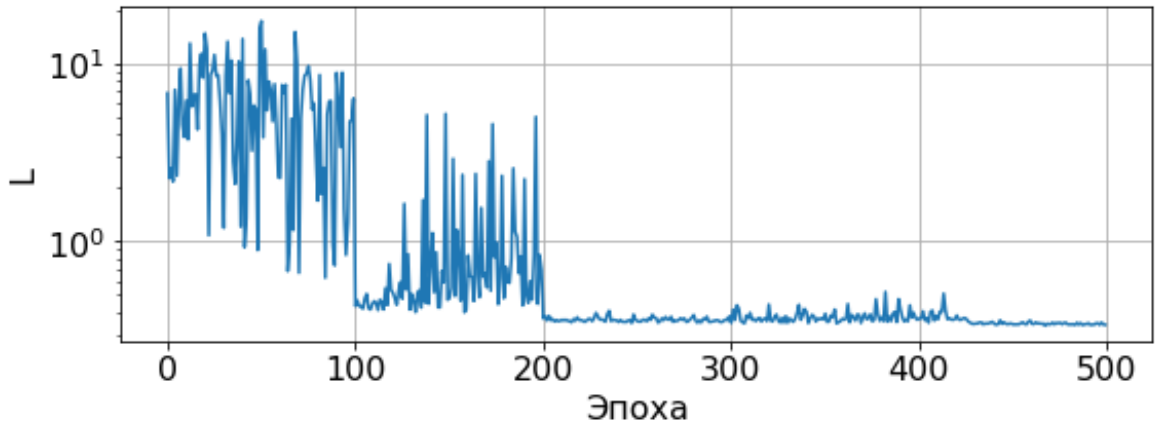

Рис. 3. График зависимости значения функции потерь для тестовых данных от номера эпохи. По оси ординат логарифмический масштаб.

Для проверки качества, с которой нейронная сеть классифицирует объекты, была использована F_1 метрика:

$$F_1^i = \frac{2\,TP_i}{2\,TP_i + FP_i + FN_i}$$

где TP_i – объекты i-го класса, которые были верно классифицированы; FP_i – объекты не i-го класса, которые были классифицированы как i-й класс; TN_i объекты i-го класса, которые были классифицированы, как не i-й класс. Эта мера является средним гармоническим величин $P_i = \frac{TP_i}{TP_i+FP_i}$ и $R_i = \frac{TP_i}{TP_i+FN_i}$, которые характеризуют ошибки первого и второго рода i-го класса соответственно.

На 467 эпохе обучения значение среднего геометрического по всем классам F_1-метрики для тестовых данных достигает максимального значения, равного 0.883. Поэтому рассмотрим модель после 467-й эпохи. В таблице 2 представлена F_1 метрика для каждого класса.

Таблица 2. Таблица значений F_1-метрики тестовых данных для каждого класса на 467-й эпохе.

Класс	0	1	2	3	4	5	6
F_1	0.798	0.879	0.908	0.931	0.936	0.941	0.835
Класс	7	8	9	10	11	12	–
F_1	0.816	0.799	0.879	0.815	0.897	0.998	–

Гистограмма распределения значения функции потерь для тестовых данных на 467-й эпохе представлена на рис. 4. Среднее значение равно 0.337, среднеквадратичное отклонение – 0.863. Квантили распределения значений ошибки представлены в таблице 3.

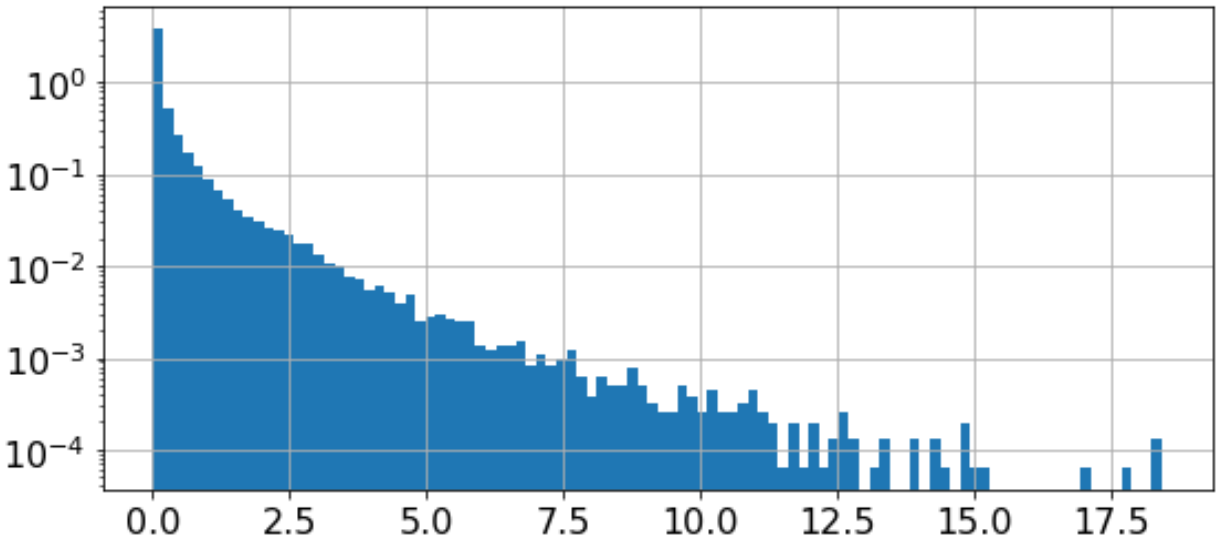

Рис. 4. Нормированная гистограмма распределения значений функции потерь для тестовых данных на 467-й эпохе. По оси ординат логарифмический масштаб.

Таблица 3. Таблица квантилей распределения значений функции потерь для тестовых данных на 467-й эпохе.

Уровень	0.005	0.05	0.25	0.5
Квантиль	$2.741 \cdot 10^{-6}$	$8.07 \cdot 10^{-5}$	$3.758 \cdot 10^{-3}$	$4.385 \cdot 10^{-2}$
Уровень	0.75	0.95	0.995	–
Квантиль	$2.582 \cdot 10^{-1}$	1.726	5.521	–

Для более глубокого анализа построим матрицу ошибок. Каждая строка соответствует одному истинному классу, а каждый столбец соответствует предсказанному классу. Элемент матрицы ошибок, находящийся на i-й строке и j-м столбце показывает сколько объектов, имеющих класс i, были отнесены к классу j. Если $i = j$, то объект классифицирован правильно. Если бы модель не ошибалась, то матрица ошибок была бы диагональная. Нормированная матрица ошибок для модели на 467-й эпохе представлена на рис. 5. Можно заметить, что наибольшие значения в строках имеют диагональные элементы, которые соответствуют верно классифицированным объектам. Также рядом с главной

диагональю вырисовываются матрицы 2 на 2, которые соответствуют одному метаклассу по скорости, но разным метаклассам по центростремительному ускорению внутри одного метакласса по параметризации. Заметим, что всю матрицу ошибок можно разделить на маленькие матрицы 2 на 2, лежащие на пересечение строк i, $i+1$ и столбцов j, $j+1$, где i, j – чётные числа от 0 до 10. Это матрицы соответствуют одному метаклассу по скорости по строке и столбцам, но разным метаклассам по центростремительному ускорению. Аналогично всю матрицу ошибок можно разделить на матрицы 4 на 4, лежащие на пересечение строк $4i,\ldots,4i+3$ и столбцов $4j,\ldots,4j+3$, где $i,j=\underline{0,2}$. Это матрицы соответствуют одному метаклассу по параметризации по строке и столбцам, но разным метаклассам по скорости и центростремительному ускорению.

Более наглядно ошибки для скорости, центростремительному ускорению и параметризации видны на рис. 6, 7. Из анализа матриц ошибок по метаклассам видно, что лучше всего модель классифицирует объекты по скорости, а хуже всего по центростремительному ускорению и параметризации.

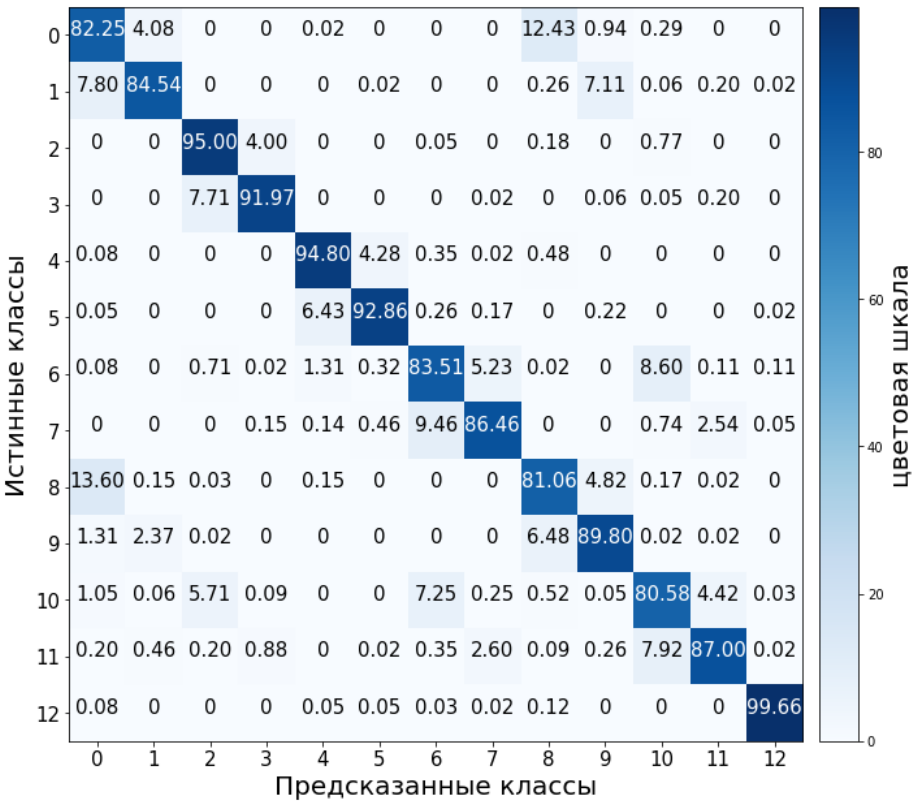

Рис. 5. Нормированная матрица ошибок для 467-й эпохи.

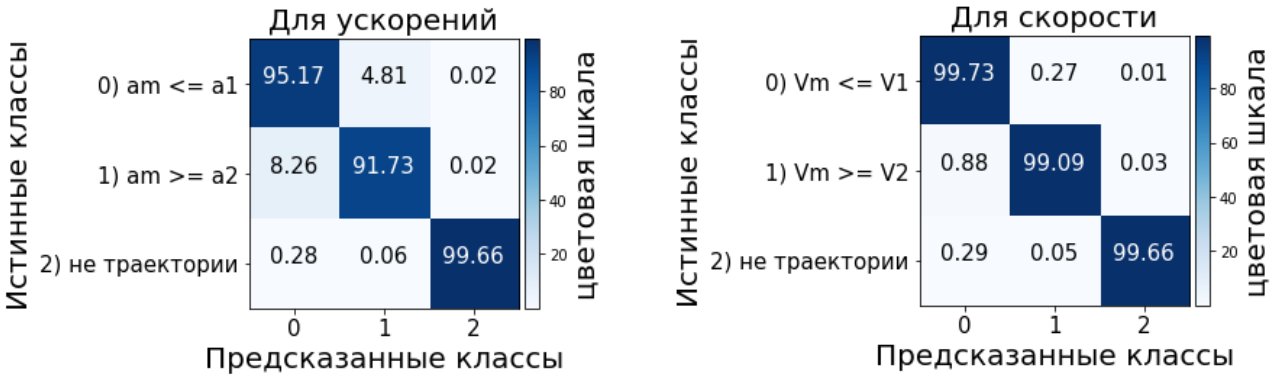

Рис. 6. Нормированные матрицы ошибок для метаклассов центростремительного ускорения и скорости.

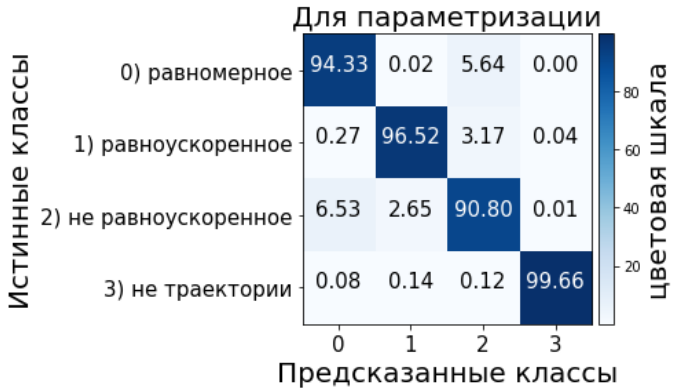

Рис. 7. Нормированная матрица ошибок для метаклассов параметризации.

1.4.Добавление шума

Из рис. 8 видно, что некоторые классы показывают хорошую устойчивость к шуму, а некоторые нет. Лучше всего устойчивы к шуму 3, 5, а также 1, 7 и 11 классы. Качество классификации этих классов падает медленнее, чем у других. Из рис. 8, что в целом модель показывает устойчивость к шуму. Значение F_1-метрики падает плавно при увеличении шума, а не резко. При этом значение F_1-метрики опускается меньше 0.8 только при шуме равном 33.

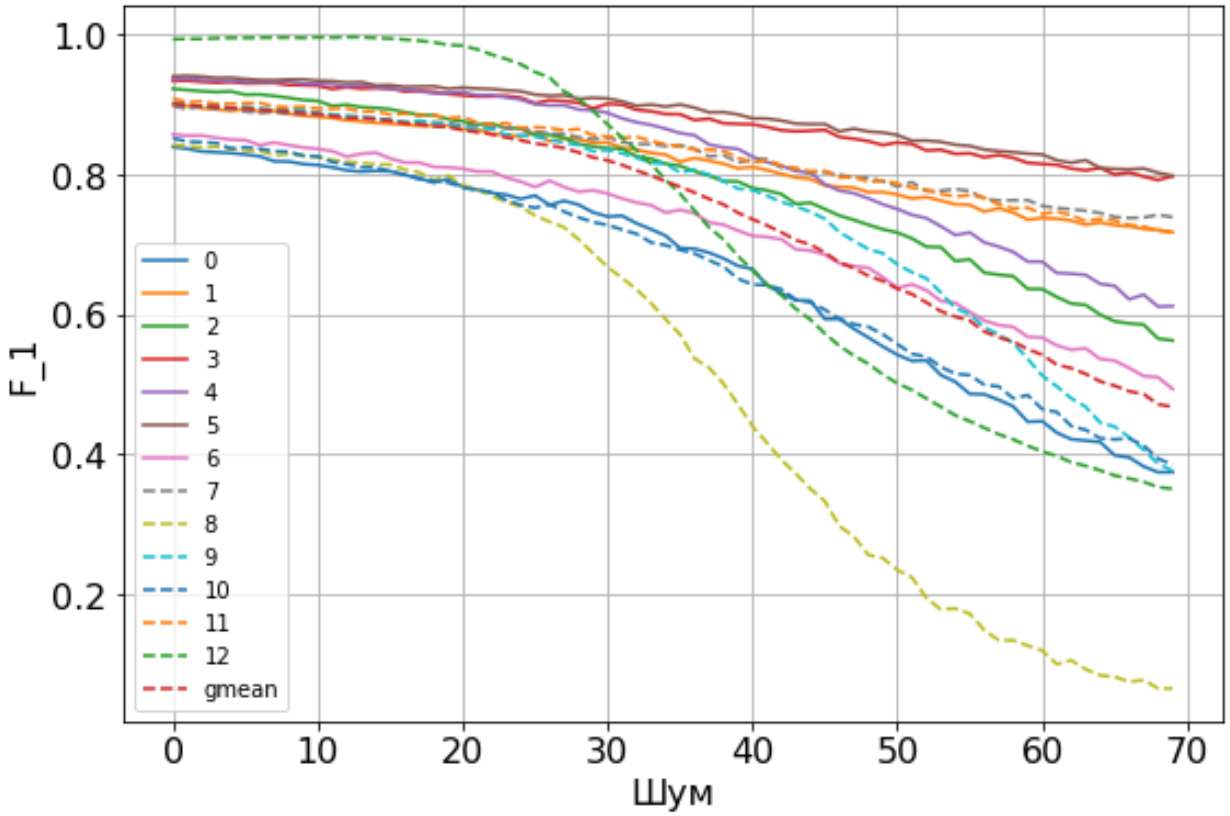

Рис. 8. Графики зависимости значения F_1-метрики для каждого класса и их среднее геометрическое от количество шумовых точек.

2. Задача распутывания

2.1. Формирование набора данных

В качестве траекторий объектов были использованы прямая, гипербола, парабола, кубическая парабола, окружность и винтовая линия, по 100 из этих кривых для обучения и по 65 для тестирования. При формировании данных брались две любые разные кривые. Всего 179 700 экземпляров для обучения и 75 855 для тестирования. Для каждой кривой были взяты 100 точек. Тестовое и обучающее множества не пересекаются.

Каждая траектория помещалась в куб с равномерной сеткой $[0, 31h]^3$, где $31h$ – длина ребра куба. Кубу ставится в соответствие массив M размера 31x31x31. Если в множество $K_{i_1,i_2,i_3} = [i_1h, (i_1+1)h) \times [i_2h, (i_2+1)h) \times [i_3h, (i_3+1)h)$, $i_1, i_2, i_3 = \underline{0,30}$ попадает точка траектории, то в элемент массива с номером $[i_1, i_2, i_3]$ записывается время, соответствующее этой точке. Если в

множество M_{i_1,i_2,i_3} попадает несколько точек, то записывается время самой поздней точки.

При формировании элемента обучающего и тестового множества из двух кривых массивы M этих кривых складываются, при этом если кривые пересекаются, то в новый массив записывается значение самой поздней точки.

2.2.Архитектура нейронной сети

Задача распутывания решалась подобно задаче сегментации изображений, но при этом учитывалась зависимость от времени. Для решения задачи распутывания была использована свёрточная нейронная сеть типа U-Net [20]. Данная архитектура показывает хорошие результаты в задаче сегментации изображений и трёхмерных снимков МРТ [21]. Она хорошо извлекает низкоуровневые признаки и не теряет высокоуровневые признаки за счёт дополнительных связей между элементами кодера и декодера. Архитектура U-Net представляет из себя кодер-декодер, при этом информация передаётся не только напрямую, но и между блоками. Кодер представляет из себя стандартную свёрточную нейронную сеть, которая преобразует и сжимает данные с помощью свёрточных слоёв и слоёв объединения. Декодер имеет архитектуру аналогичную декодеру, но данные направлены в другую сторону. По мере продвижения данных по декодеру от скрытого латентного пространства к выходу, их размерность увеличивается, и на выходе мы получаем массив такой же размерности, что и размерность входных данных. Вместо слоёв объединения в декодере используются слои разворачивания (англ. unpooling).

Для учёта временной зависимости был добавлен новый выход из латентного пространства, который возвращает максимальные значения скоростей траекторий.

На вход модели подавался трёхмерный массив, размером 31x31x31. Модель возвращает три трёхмерных массива, элементы которых лежат на отрезке $[0, 1]$. В первом массиве отмечены точки, которые не принадлежат ни одной из траекторий, во втором и третьем массивах отмечены точки первой и второй траектории соответственно.

2.3.Обучение и тестирование

В качестве функции потерь для данной модели была использована комбинация перекрёстной энтропии, Tversky loss [21], и отдельно Tversky loss для пересечения первого и второго классов.

Перекрёстная энтропия применяется поэлементно к массивам, возвращаемым моделью и массивам с истинными значениями, для каждого класса. Так как количество точек нулевого класса больше, чем количество элементов других классов, то были использованы следующие веса для перекрёстной энтропии для каждого класса: $w = \left(\frac{1}{301}, \frac{150}{301}, \frac{150}{301}\right)$.

Tversky loss имеет следующий вид

$$L(Y,Z) = \sum_{i=1}^{N} (1 - TI_i), \qquad TI_i = \frac{TP_i}{TP_i + \alpha_i\, FP_i + \beta_i\, FN_i},$$

$$TP_i = \sum_{j_1,j_2,j_3=1}^{s_1,s_2,s_3} Y^i_{j_1,j_2,j_3}\, Z^i_{j_1,j_2,j_3},$$

$$FP_i = \sum_{j_1,j_2,j_3=1}^{s_1,s_2,s_3} Y^i_{j_1,j_2,j_3}\left(1 - Z^i_{j_1,j_2,j_3}\right), \qquad TP_i = \sum_{j_1,j_2,j_3=1}^{s_1,s_2,s_3} \left(1 - Y^i_{j_1,j_2,j_3}\right) Z^i_{j_1,j_2,j_3},$$

где Y – выход модели, Y^i – выход модели для i-го класс; Z – истинные значения, Z^i – истинные значения для i-го класс; N – количество классов, $TI_i = TI\left(Y^i, Z^i\right)$ – Tversky индекс для i-го класса; $TP_i = TP\left(Y^i, Z^i\right)$ – количество правильно распознанных элементов i-го класса; $FP_i = TF\left(Y^i, Z^i\right)$ – количество элементов массива i-го класса, которые распознаны как элементы i-го класса, но таковыми не являются; $FN_i = TN\left(Y^i, Z^i\right)$ – количество элементов массива i-го класса, которые распознаны как элементы других классов, но являются элементами i-го класса; s_i – размерность выходных данных по i-му измерению; α_i, β_i – константы для i-го класса.

Tversky индекс обобщает F_1-метрику, и при $\alpha = \beta = 0.5$ совпадает с ней. В данном случае F_1-метрику можно трактовать, как отношение мер пересечения

двух множеств, к мере их объединения. Если множества не пересекаются, то F_1-метрика равна 0, а если совпадают, то равна 1.

Tversky индекс позволяет по-разному учесть вклад ошибок FP и FN за счёт подбора параметров α и β. Для нулевого класса нам важнее ошибка FP, чтобы точки траектории не распознавались, как точки, не принадлежащие траекториям. Для нулевого класса были взяты значения $\alpha = 1000,\ \beta = 1$. Для первого и второго класса нам важнее ошибка FN, чтобы минимизировать перепутывания точек первого и второго классов, поэтому для этих классов мы взяли $\alpha = 1, \beta = 3$.

Помимо этого, чтобы акцентировать больше внимания на перепутывании точек между первым и вторым классами, считалась Tversky loss для пересечения первого и второго классов:

$$L(Y^{1,2}, Z^{1,2}) = 1 - \frac{TP_{1,2} + 1}{TP_{1,2} + FP_{1,2} + FN_{1,2} + 1},$$

где $Y^{1,2}$ и $Z^{1,2}$ массивы, полученные поэлементным перемножением массивов первого и второго классов для выхода модели и истинных значений, соответственно. В числителе и в знаменателе добавляется 1, чтобы исключить деление на ноль, и чтобы функция не равнялась нулю всегда, когда нет пересечения между траекториями.

Для оптимизации параметров модели был использован стохастический градиентный спуск, L_2-регуляризация с коэффициентом 10^{-4} и экспоненциально уменьшающийся темп обучения. Темп обучения уменьшался в 10 раз либо после 65 эпох с момента прошлого изменения темпа обучения, либо если функция потерь не уменьшалась 20 эпох на величину больше, чем 10^{-4}.

Модель обучалась по пакетам размера 64, всего пакетов 2 808. Модель обучалась 150 эпох.

Архитектура нейронной сети написана с помощью библиотеки PyTorch. Код написан на языке программирования Python.

График значений функции потерь для обучающих данных представлен на рис. 9. Функция потерь достигает минимального значения 0.166 на 149-й эпохе.

На рис. 10 изображён график значений функции потерь по эпохам для тестовых данных. Минимальное значение, равное 0.327, достигается на 109-й эпохе.

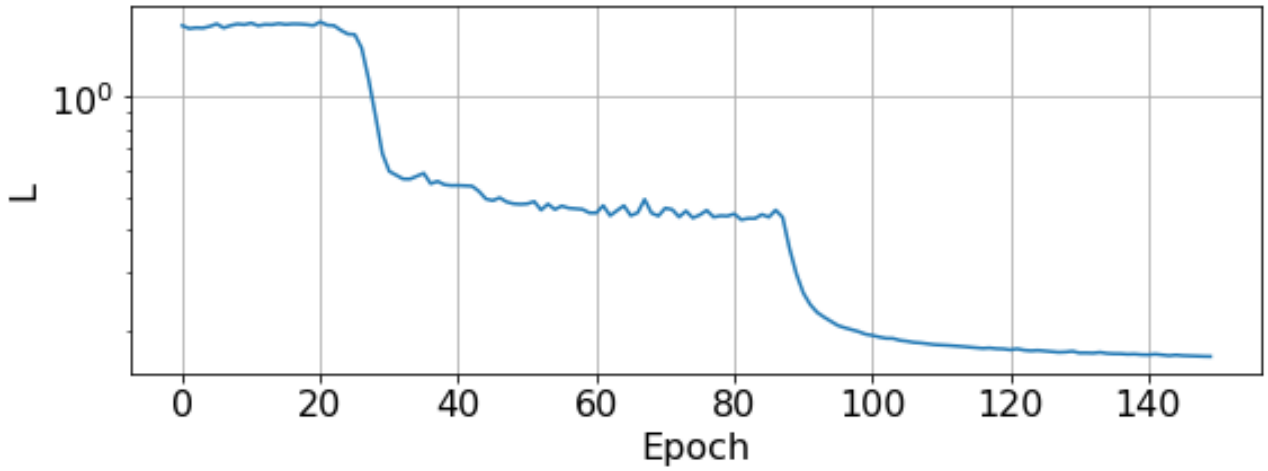

Рис. 9. График значений функции потерь для обучающих данных. По оси ординат логарифмический масштаб.

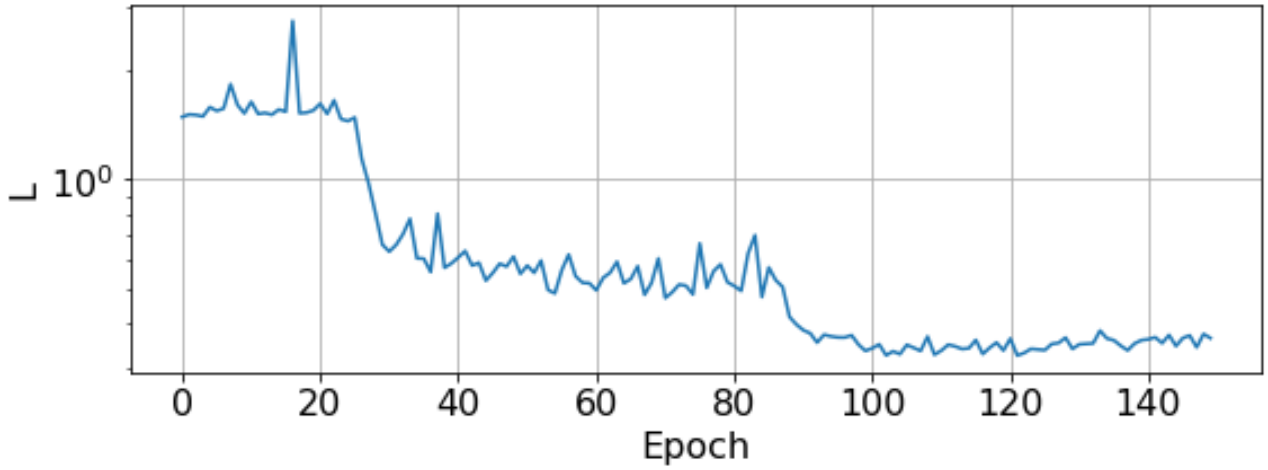

Рис. 10. График значений функции потерь для тестовых данных. По оси ординат логарифмический масштаб.

Нам важнее всего значение функции Tversky loss, так как она лучше остальных функций потерь отражает качество классификации. График среднего значения Tversky loss для тестовых данных представлен на рис. 11. Из графика видно, что в целом значение функция потерь падает примерно до 109, а после начинает расти. Этот эффект связан с переобучение. Минимального значения, равное 0.167, функция потерь достигает на 109-й эпохе.

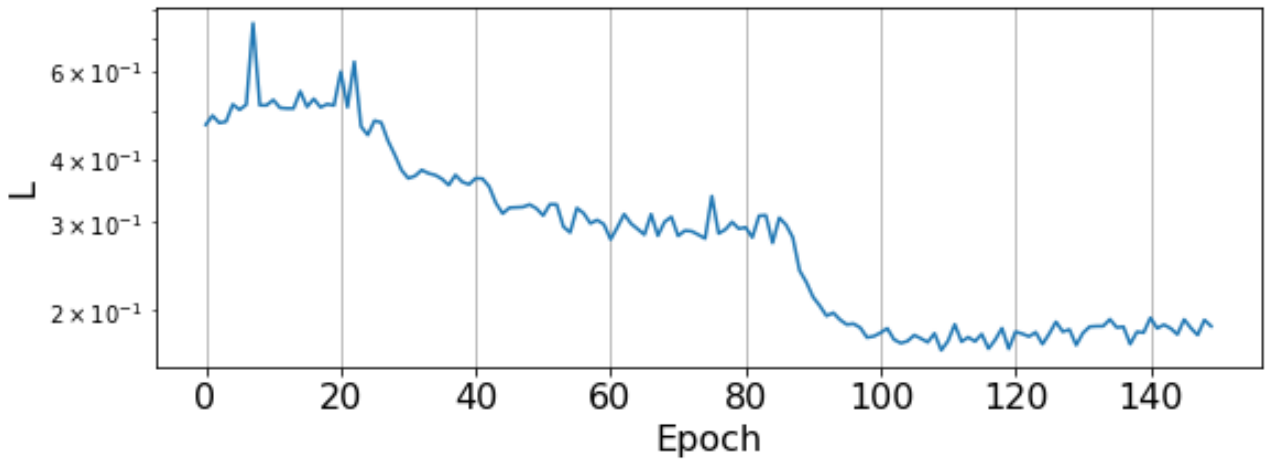

Рис. 11. График значений Tversky loss для тестовых данных. По оси ординат логарифмический масштаб.

Для проверки качества обучения был использован Tversky индекс с параметрами $\alpha = \beta = 0.5$. При таких параметрах Tversky индекс совпадает с F_1-метрикой. Максимальное значение среднего геометрического по всем классам данный Tversky индекс достигает на 109-й эпохе, и равняется 0.911. Значения Tversky индекс для каждого класса представлены в таблице 4.

Таблица 4. Значения Tversky индекс для каждого класса на 109-й эпохе.

Класс	Нет траектории	Траектория 1	Траектория 2
F_1	0.999	0.867	0.872

Гистограмма распределения значения Tversky loss с параметрами $\alpha = \beta = 0.5$ для тестовых данных на 109-й эпохе представлена на рис. 12. Среднее значение равно 0.089, среднеквадратичное отклонение – 0.13. Квантили распределения значений ошибки представлены в таблица 4.

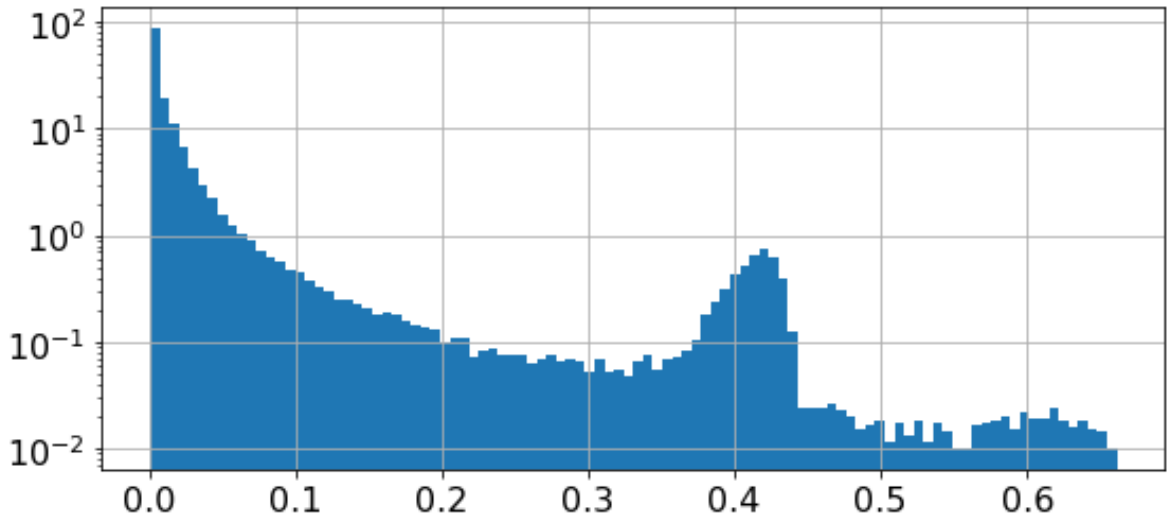

Рис. 12. Нормированная гистограмма распределения значений Tversky индекс для тестовых данных на 109-й эпохе. По оси ординат логарифмический масштаб.

Таблица 5. Квантили распределения Tversky loss для тестовых данных на 109-й эпохе.

Уровень	0.005	0.05	0.25	0.5
Квантиль	$1.257 \cdot 10^{-6}$	$4.389 \cdot 10^{-5}$	$3.843 \cdot 10^{-2}$	$2.662 \cdot 10^{-1}$
Уровень	0.75	0.95	0.995	–
Квантиль	0.121	0.402	0.61	–

Заключение.

Рассмотрено применение глубоких свёрточных нейронных сетей для задач классификации и распутывания траекторных данных. Предложены типы архитектур и модели для решения данных задач. Показано, что свёрточные нейронные сети справляются с данными задачами на синтетических наборах данных. Показано, что модель для задачи классификации устойчива к шуму.

Данная работа может быть основой для дальнейших исследований применения глубоких нейронных сетей для задач обработки траекторных данных.

Литература

1. Казаков И.Е., Мальчиков С.В. *Анализ стохастических систем в пространстве состояний*. Москва, Наука. 1983. 384 с.
2. Ярлыков М.С., Миронов М.А. *Марковская теория оценивания случайных процессов.* Москва, Радио и связь. 1993. 464 с.
3. *Марковская теория оценивания в радиотехнике,* под ред. М.С. Ярлыкова. Москва, Радиотехника. 2004. 504 с.
4. Татузов А.Л. *Нейронные сети в задачах радиолокации*. Москва, Радиотехника. 2009. 432 с.
5. Goodfellow I., Bengio Y., Courville A. *Deep Learning*. Cambridge, Massachusetts, USA, MIT Press. 2016. 800 p.
6. Portsev R.J., Makarenko A.V. Convolutional Neural Network for Noise Signal Recognition. *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*. Aalborg, Denmark. 2018. P.1-6. https://doi.org/10.1109/MLSP.2018.8516920
7. Makarenko A.V. Deep Convolutional Neural Networks for Chaos Identification in Signal Processing. *2018 26th European Signal Processing Conference (EUSIPCO)*. Rome, Italy. 2018. P.1481-1485. https://doi.org/10.23919/EUSIPCO.2018.8553098
8. Бобин А.В., Азаров В.А., Булгаков С.А., Савин Д.А. Методика распознавания летательных аппаратов и радиолокационных ловушек в контуре управления системы контроля воздушного пространства на основе нейросетевой технологии. *Известия МГТУ «МАМИ»*. 2013. Т.4 №1. С.123-130.
9. Park S.H., Kim B., Kang C.M., Chung C.C., Choi J.W. Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture, *2018 IEEE Intelligent Vehicles Symposium*. Changshu, China. 2018. P.1672-1678. https://doi.org/10.1109/IVS.2018.8500658
10. Khosroshahi A., Ohn-Bar E., Trivedi M.M. Surround Vehicles Trajectory Analysis with Recurrent Neural Networks. *2016 IEEE 19th International Conference on*

Intelligent Transportation Systems (ITSC). Rio de Janeiro, Brazil. 2016. P.2267-2272. https://doi.org/ 10.1109/ITSC.2016.7795922

11. Wang L., Zhang L, Yi Z. Trajectory Predictor by Using Recurrent Neural Networks in Visual Tracking. *IEEE Transactions on Cybernetics*. 2017. Vol.47. P.3172-3183. https://doi.org/10.1109/TCYB.2017.2705345
12. Liu Y., Hansen M. Predicting Aircraft Trajectories: A Deep Generative Convolutional Recurrent Neural Networks Approach. *CoRR*. 2018. https://arxiv.org/abs/1812.11670
13. Костючек М.И., Макаренко А.В. Classification of observable 3D moving object by their kinematic characteristics by deep convolution neural network. *Proceedings of the 5th International Conference on Stochastic Methods (ICSM-5, 2020)*. Москва. 2020. С.326-330.
14. Костючек М.И., Макаренко А.В. Метод генерации в кинематическом приближении синтетических трехмерных траекторий подвижных объектов. *Материалы 15-й Международной конференции «Устойчивость и колебания нелинейных систем управления» (конференция Пятницкого) (Москва, 2020)*. Москва. 2020. С.219-221.
15. He K., Zhang X., Ren S., Sun J., Deep residual learning for image recognition. *CoRR*. 2015. https://arxiv.org/abs/1512.03385
16. Lin M., Chen Q. Yan S. Network in network. *2nd International Conference on Learning Representations (ICLR) 2014*. Banff, AB, Canada. 2014. https://arxiv.org/abs/1312.4400
17. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning, PMLR*. Lille, France. 2015. Vol.37. P.448-456.
18. Hinton G.E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R., Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*. 2012. http://arxiv.org/abs/1207.0580

19. Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*. 1964. Vol.4. P.1–17. https://doi.org/10.1016/0041-5553(64)90137-5
20. Ronneberger O., Fischer P. Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*. 2015. http://arxiv.org/abs/1505.04597
21. Salehi S.S.M., Erdogmus D. Tversky loss function for image segmentation using 3D fully convolutional deep networks. 2017. http://arxiv.org/abs/1706.05721

Для цитирования:

Костючек М.И., Макаренко А.В. Применение свёрточных глубоких нейронных сетей для решения некоторых задач анализа траекторных данных. *Журнал радиоэлектроники* [электронный журнал]. 2021. №11. https://doi.org/10.30898/1684-1719.2021.11.15