Обзор статьи «Neural Ordinary Differential Equations»[1]

В [1] была предложена новая модель машинного обучения. Многие модели машинного обучения (ResNet, RNN, normalizing flows) используют последовательные модификации скрытого состояния:

$$h_{t+1} = h_t + f(h_t, \theta_t).$$

Авторы предлагают рассматривать вектор скрытых состояний как непрерывный вектор, динамика которого определяется нейросетью. Для получения выходов модели используются численные метода решения дифференциальных уравнений. Данная модель применима в множестве областей: классификация изображений, явные генеративные модели, модели, работающие с временными рядами. Предложенная модель имеет ряд достоинств: эффективная по памяти, имеет явный гиперпараметр регулирующий отношение точность / время вычисления, при применении в нормализующих потоках повышает вычислительную эффективность и выразительность модели, естественна работает с непрерывными временными рядами.

1 Описание модели

Пусть есть некоторый вектор $z(t)$, зависящий от времени, функция $f_\theta(z(t), t)$, заданная нейросетью с параметрами θ, $z(t_0) = z_0$ - вход нейросети, $z(t_1)$ - выход нейросети. Нейродиффуром называется следующая модель:

$$\frac{dz}{dt} = f_\theta(z(t), t), \;\; z(t_0) = z_0 \tag{1}$$

Выход модели $z(t_1)$ - решение дифференциального уравнения 1:

$$z(t_1) = z(t_0) + \int_{t_0}^{t_1} f_\theta(z(t), t)dt \tag{2}$$

Поскольку динамика $f_\theta(z(t), t)$ задается нейросетью, то аналитическое интегрирование (в случае произвольной сети) провести невозможно, поэтому для нахождения решения 2 (или выхода модели) необходимо использовать численные схемы решения уравнения 1. Таким образом, выходом модели является результат численной схемы решения дифференциальных уравнений $ODESolver$, получающий на вход начальную точку, функцию динамики (нейросеть), начальное и конечное время:

$$output = z(t_1) = ODESolver(z_0, f_\theta, t_0, t_1) \tag{3}$$

2 Обучение модели

Пусть задана некоторая функция потерь $L(z(t_1))$, обучение модели будет происходить градиентными методами, поэтому необходимо находить $\frac{\partial L}{\partial \theta}$.

2.1 Метод «в лоб»

Многие численные методы решения дифференциальных уравнений 3 могут быть продифференцированы по параметрам функции динамики f_θ, тогда возможно использовать обычный метод обратного распространения ошибки для получения $\frac{\partial L}{\partial \theta}$.

Рассмотрим пример для одномерного случая ($z(t) \in R$): пусть используется метод Эйлера для решения уравнения 1. Напомним, что для этого задается сетка:

$$t_0 = x_0 < \cdots < x_n = t_1$$

. А приближенное решение в узлах сетки z_i определяется по формуле:

$$z_i = z_{i-1} + (x_i - x_{i-1}) f_\theta(z_{i-1}, x_{i-1}),\ i = 1, , 2, 3, \ldots, n. \tag{4}$$

Тогда можно найти градиенты функции потерь по параметрам сети, но данный метод имеет существенные недостатки:

- Необходимо хранить градиенты по всем промежуточным узлам сетки z_i, что при большом количестве шагов дорого по памяти.
- Происходит накопление ошибки из-за того, что будет дифференцироваться приближенное решение. Авторами был предложен метод, в котором строится аппроксимация градиентов без дифференцирования приближенного решения.

В разделе 3.1 было экспериментально показано, что использование метода «в лоб» приводит к худшему результату, по сравнению с методом, предложенным авторами.

2.2 Метод, предложенный авторами

В статье предлагается рассматривать численную схему решения уравнения 1, как черный ящик. Рассматриваются функции $a_z(t) = \frac{\partial L}{\partial z(t)}$, $a_\theta(t) = \frac{\partial L}{\partial \theta(t)}$, называемые

сопряженными (adjoint). Стоит отметить, что в данной модели $\theta(t) = \theta$, то есть одинакова на различных шагах решения численной схемы 3, но для удобства выкладок она рассматривается, как функция от времени. Авторы статьи проводят дальнейшие выкладки по-другому, поэтому часть обозначений немного отличается. Все векторы рассматриваются, как векторы-строки ($z(t) \in R^{1\times M}$).

Теорема 1. *Динамики сопряженных функций* $\frac{da_\theta(t)}{dt}$, $\frac{da_z(t)}{dt}$ *задаются уравнениями:*

$$\frac{da_z(t)}{dt} = -a_z(t)\frac{\partial f_\theta(z(t),t)}{\partial z}$$

$$\frac{da_\theta(t)}{dt} = -a_z(t)\frac{\partial f_\theta(z(t),t)}{\partial \theta}$$

Доказательство. Пусть ϵ - некоторое положительное число. Функция потерь $L(z(t_1))$ неявно зависит от состояний $z(t)$ и параметров $\theta(t)$ для всех $t \le t_1$. Используя формулу производной функции, заданной неявно, а также формулу производной сложной функции можно записать:

$$a_z(t) = \frac{\partial L}{\partial z(t)} = \frac{\partial L}{\partial z(t+\epsilon)}\frac{\partial z(t+\epsilon)}{\partial z(t)} + \frac{\partial L}{\partial \theta(t+\epsilon)}\frac{\partial \theta(t+\epsilon)}{\partial z(t)} \tag{5}$$

$$a_\theta(t) = \frac{\partial L}{\partial \theta(t)} = \frac{\partial L}{\partial z(t+\epsilon)}\frac{\partial z(t+\epsilon)}{\partial \theta(t)} + \frac{\partial L}{\partial \theta(t+\epsilon)}\frac{\partial \theta(t+\epsilon)}{\partial \theta(t)} \tag{6}$$

В формулах выше $\frac{\partial \theta(t+\epsilon)}{\partial z(t)}$ - нулевая матрица ($\theta(t+\epsilon)$ не зависит от $z(t)$), а $\frac{\partial \theta(t+\epsilon)}{\partial \theta(t)}$ - единичная матрица ($\theta(t+\epsilon) = \theta(t)$). Используя формулу Тейлора для $z(t+\epsilon)$ в окрестности точки t:

$$z(t+\epsilon) = z(t) + \epsilon f_\theta(z(t),t) + \mathcal{O}(\epsilon^2) \implies$$

$$\frac{\partial z(t+\epsilon)}{\partial z(t)} = I + \epsilon\frac{\partial f_\theta(z(t),t)}{\partial z(t)} + \mathcal{O}(\epsilon^2),\ \frac{\partial z(t+\epsilon)}{\partial \theta(t)} = \epsilon\frac{\partial f_\theta(z(t),t)}{\partial \theta(t)} + \mathcal{O}(\epsilon^2) \tag{7}$$

Тогда подставляя 7 в 5 и 6, и учитывая $\frac{\partial L}{\partial z(t+\epsilon)} = a_z(t+\epsilon)$, $\frac{\partial L}{\partial \theta(t+\epsilon)} = a_\theta(t+\epsilon)$:

$$a_z(t) = a_z(t+\epsilon)(I + \epsilon\frac{\partial f_\theta(z(t),t)}{\partial z(t)} + \mathcal{O}(\epsilon^2)) \tag{8}$$

$$a_\theta(t) = a_z(t+\epsilon)(\epsilon\frac{\partial f_\theta(z(t),t)}{\partial\theta(t)} + \mathcal{O}(\epsilon^2)) + a_\theta(t+\epsilon) \tag{9}$$

Тогда используя 8, 9:

$$\frac{da_z}{dt} = \lim_{\epsilon\to 0}\frac{a_z(t+\epsilon)-a_z(t)}{\epsilon} = \lim_{\epsilon\to 0} -a_z(t+\epsilon)\frac{\epsilon\frac{\partial f_\theta(z(t),t)}{\partial z(t)} + \mathcal{O}(\epsilon^2)}{\epsilon} = -a_z(t)\frac{\partial f_\theta(z(t),t)}{\partial z(t)} \tag{10}$$

$$\frac{da_\theta}{dt} = \lim_{\epsilon\to 0}\frac{a_\theta(t+\epsilon)-a_\theta(t)}{\epsilon} = \lim_{\epsilon\to 0} -a_z(t+\epsilon)\frac{\epsilon\frac{\partial f_\theta(z(t),t)}{\partial \theta(t)} + \mathcal{O}(\epsilon^2)}{\epsilon} = -a_z(t)\frac{\partial f_\theta(z(t),t)}{\partial \theta(t)} \tag{11}$$

□

Таким образом, мы имеем два новых дифференциальных уравнения 10, 11. Стоит отметить, что величины $\frac{\partial f_\theta(z(t),t)}{\partial z(t)}$, $\frac{\partial f_\theta(z(t),t)}{\partial \theta(t)}$ могут быть легко вычислены, так как они являются матрицей производных выходов нейросети по входам, и по параметрам. Во всех популярных фреймворках, в случае использования в нейросети операций из этих фреймворков, вычисление этих матриц производных происходят автоматически. К дифференциальным уравнениям 10, 11 существуют начальные условия:

$$a_z(t_1) = \frac{\partial L}{\partial z(t_1)},\ a_\theta(t_1) = 0 \tag{12}$$

Первое условие является градиентом функции потерь в конечной точке, и может быть легко вычислено при наличии выхода модели $z(t_1)$. Второе условие следует из того, что при изменении весов в конечный момент времени, функция потерь никак не изменится, так как параметров в последний момент времени $z(t_1)$ «не успеет» поменяться.

Таким образом, во время прямого прохода получаем выход модели по формуле 3, используя численный метод решения ОДУ, на обратном шаге вычисляем сначала производную функции потерь по выходу $\frac{\partial L}{\partial z(t_1)}$, затем используя функции динамики 10, 11 и начальные условия 12 решаем вместе назад во времени уравнения 10, 11. То есть численная схема для решения этих уравнений использует одинаковые узлы сетки $\{x_i,\ i = 1, 2, 3, \ldots, n,\ x_1 = t_0,\ x_n = t_1\}$, так как для вычисления динамики 11 в n-ом узле сетки необходимо значение $a_z(x_n)$, полученное при численном решении уравнения 10.

Данный метод не имеет недостатков, описанных в разделе 2.1, так как напрямую аппроксимирует производную по параметрам и не хранит промежуточные активации (а перевычисляет их на обратном шаге).

В итоге решив уравнения 10, 11 назад во времени до момента времени t_0, будем иметь $\frac{\partial L}{\partial z(t_0)}$, $\frac{\partial L}{\partial \theta(t_0)}$. $\frac{\partial L}{\partial \theta(t_0)}$ используется для обучения нейросети. Стоит отметить, что несмотря на то, что $\theta(t) = \theta$, мы используем для обучения градиенты в начальный момент времени $\frac{\partial L}{\partial \theta(t_0)}$. Интуитивно это можно объяснить так: $\frac{\partial L}{\partial \theta(t)}$ показывает как изменится лосс при изменении весов в момент времени t, так как наши веса не меняются во времени, нас интересует, как изменится лосс при изменении весов в начальный момент времени. Величина $\frac{\partial L}{\partial z(t_0)}$ используется в случае, если $z(t_0)$ является выходом другой обучаемой модели (например, в эксперименте 3.1 $z(t_0)$ выход небольшой сверточной сети).

В статье была выведена формула динамики $\frac{\partial L}{\partial t}$ и формулы для градиентов $\frac{\partial L}{\partial t_0}$, $\frac{\partial L}{\partial t_1}$. Данные формулы получаются аналогично 10. Вывод данных формул не приводится в силу того, что данные градиенты не были использованы ни в одном из авторских экспериментов. $\frac{\partial L}{\partial t_0}$, $\frac{\partial L}{\partial t_1}$ позволяют обучать начальные и конечные точки, но примера такого обучения приведено не было.

3 Эксперименты из статьи

3.1 Эксперименты на MNIST-e

В данном разделе было исследовано качество на датасете MNIST [2] предложенной модели, обученной методами «в лоб», авторским методом, а также популярной для задачи классификации изображений модели ResNet [3].

Цель данного эксперимента: сравнить два метода обучения предложенной модели и сравнить предложенную модель с известными.

Авторы рассматривали три модели: ResNet, RK-Net, ODE-Net. В каждый из них в начале использовалась небольшая сверточная сеть, которая уменьшала размер входного изображения в 4 раза и состояла из одной свертки, увеличивающей число каналов до 64 и двух ResNet блоков, каждый из которых уменьшал пространственные размерности в два раза.

В ResNet сети далее использовались 6 ResNet блоков, не изменяющих размерности изображений, а в RK-Net и ODE-Net далее использовался численный метод Рунге-Кутта решения ОДУ ODESolver для функции динамики $f_\theta(z(t), t)$, заданной небольшой сверточной нейросетью с параметрами θ. Для моделирования явной зависимости динамики от времени к скрытому представлению $z(t)$ конкатенировался константный канал со значением t. Для RK-Net использовался метод обучения «в лоб», а для ODE-Net метод с использованием сопряженных функций.

Для получения предсказания класса картинки использовался Global Average Pooling на выходе модели, затем линейный слой.

	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$

Таблица 1: Результаты на датасете MNIST [2]

В таблице 1 приведены результаты моделей. Из сравнения RK-Net с ResNet и ODE-Net можно сделать вывод, что при обучении методом «в лоб» требуется значительно больше памяти ($\tilde{L}$ - это число оценки функции динамики, а L - число слоев в ResNet), при этом достигается худшее качество. Таким образом, было экспериментально показано наличие недостатков метода обучения «в лоб», описанных в разделе 2.1. Модели RK-Net и ODE-Net имеют сравнимое качество с ResNet, но имеет значительно меньшее количество параметров. В первой строке таблицы приведены результаты линейной модели, имеющей схожее количество параметров, но с значительно меньшим качеством. Данное сравнение не является показательным в силу различности архитектуры (было бы странно ожидать что линейный слой заработал бы лучше множества сверток на картинках).

Для любого численного метода решения ОДУ можно задавать число шагов (явно или используя ограничение отклонения аппроксимации от точного решения). Меняя число шагов, можно менять время работы модели. Это можно использовать для ускорения модели на тесте при незначительной потере качества. На рис. 1 изображе-

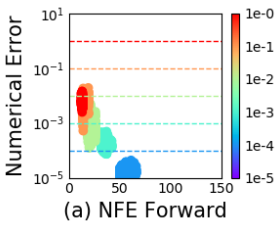

Рис. 1: a

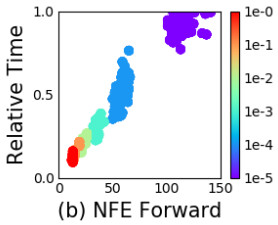

Рис. 2: b

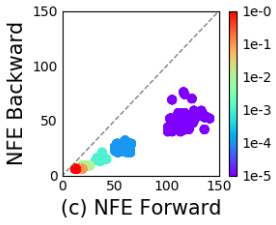

Рис. 3: c

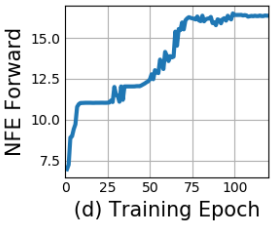

Рис. 4: d

на зависимость ошибки аппроксимации от количества шагов численного метода, а на рис. 2 зависимость времени работы от количества шагов. Таким образом, имеется некоторый гиперпараметр, который явно регулирует отношение качество / время вычисления.

На рис 3 показана зависимость количество шагов на обратном проходе от количества шагов на прямом проходе. Количество шагов на обратном проходе примерно вдвое меньше количества шагов на прямом проходе. Таким образом, метод сопряженных функций более эффективен не только по памяти, но и по количеству шагов, по сравнению с методом «в лоб».

На рис. 4 показана зависимость количества шагов численного метода на прямом проходе от номера эпохи. Количество шагов растет по мере обучения, постепенно повышая сложность модели. Можно сказать, что модель становиться более глубокой (если под глубиной понимать количество шагов численного метода).

3.2 Непрерывные нормализующие потоки

Рассмотрим сначала дискретные конечные нормализующие потоки - явные (то есть имеющие доступ к плотности) генеративные модели. Пусть имеется некоторое распределение $p_0(z_0)$, плотность которого известна и может быть вычислена, а также можно легко сэмплировать векторы из этого распределения $z_0 \sim p_0(z_0)$. Пусть $f_\theta(z_0)$ - непрерывно дифференцируемая и обратимая функция (нейросеть с параметрами θ, тогда справедлива формула замены переменных и плотность $z_1 = f_\theta(z_0)$ может быть найдена аналитически по формуле:

$$\log p_1(z_1) = \log p_0(z_0) - \log \det \frac{\partial f_\theta}{\partial z_0}.$$

На практике рассматривают последовательность функций $f_{\theta_k}(z_k)$, последовательно примененные к друг другу $z_{k+1} = f_{\theta_k}(z_k)$. Основная проблема - вычисление определителя $\det \frac{\partial f_\theta}{\partial z_0}$. В общем случае имеет сложность $\mathcal{O}(D^3)$, где $z_0,\ z_1 \in R^D$. Для решения этой проблема вводят ограничения на вид f_θ так, чтобы вычисление определителя выполнялось за меньшее число операций, но это приводит к снижению выразительной способности модели. Пример - плоские нормализующие потоки (planar normalizing flows):

$$z_{k+1} = f_{\theta_k}(z_k) = z_k + u_k h(w_k^T z_k + b_k), \theta_k = \{u_k, w_k, b_k\}$$

При таких f_θ вычисление одного определителя делается за $\mathcal{O}(D)$.

Рассмотрим теперь непрерывные нормализующие потоки. Имеется нейродиффур, вход которого является сэмплом из некоторого распределения $p_0(z(0))$, а выход $z(T)$ находится с помощью численных методов решения ОДУ:

$$\frac{dz}{dt} = f(z(t), t),\ z(0) \sim p_0(z(0)).$$

Пусть f равномерно удовлетворяет условию Липшица по z и непрерывна по t, тогда для плотности переменной $z(t)$ справедливо:

$$\frac{\partial \log p(z(t))}{\partial t} = -tr\left(\frac{df}{dz}(t)\right)$$

Доказательство данного утверждения может быть найдено в статье [1], в целом оно аналогично доказательству для динамик сопряженных функций, которое было приведено ранее.

Таким образом, для вычисление плотности происходит за $\mathcal{O}(D * costs(f))$,где $costs(f)$ - сложность вычисления функции f, так как в следе D частных производных, вычисление каждой из которых, в связи со спецификой автоматического дифференцирования происходит за время вычисления f. В случае непрерывных норм. потоков накладываются более слабые условия на f, и при этом плотность вычисляется за меньшее время. Стоит отметить, что если функция динамики задается как сумма других функций, то в силу линейности следа динамика плотности также задается как сумма:

$$\frac{dz(t)}{dt} = \sum_{n=1}^{M} f_n(z(t)) \implies \frac{d \log p(z(t))}{dt} = \sum_{n=1}^{M} tr\left(\frac{\partial f_n}{\partial z}\right).$$

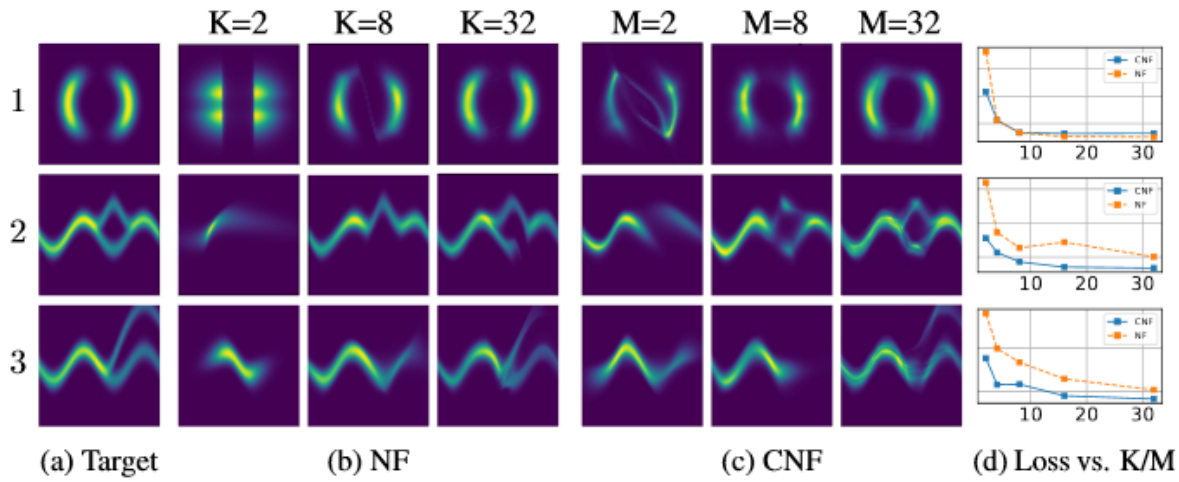

Рис. 5: Сравнение норм. потоков и непрерывных норм. потоков

Такое вычисление делается за $\mathcal{O}(M)$, а в случае аналогичного использования в дискретных потоках делалось бы за $\mathcal{O}(M^3)$. Авторы провели экспериментальное сравнение непрерывных и дискретных нормализующих потоков. Для этого провели ряд экспериментов на синтетических данных. В первом эксперименте было показано, что непрерывный норм. поток с динамикой, заданной суммой M функций не менее выразителен, чем дискретный норм. поток с $K = M$ последовательных преобразований, при этом непрерывный поток более вычислительно эффективен.

На рис. 5 приведены апостериорные плотности для норм. потоков (NF), K задает число функций преобразования, и для непрерывных норм. потоков, M задает количество слагаемых функций в динамике. В качестве лосса использовалась $KL(q(x)||p(x))$, где q - плотность из потока, а p - плотность данных. Непрерывные норм. потоки достигали меньшего значения лосса и были более выразительными.

Еще одним достоинством непрерывных норм. потоков является то, что сложность обратного преобразования (из данных в шум) примерно равна сложности прямого преобразования. Таким образом, можно обучать поток методом максимального правдоподобия, максимизируя $E_{p(x)}[\log q(x)]$, где p плотность данных, q - плотность потока. Далее используются набора данных, для которых p(x) не имеет простой аналитической формы , поэтому для вычисления матожидания используется среднее по батчу.

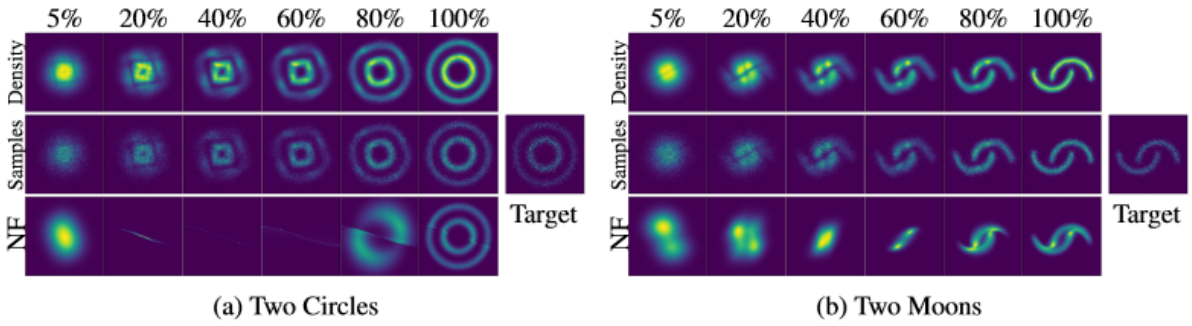

Рис. 6: Генерация данных из шума для норм. потоков и непрерывных норм. потоков

На рис. 6 показан процесс генерации данных из шума. Непрерывный норм. потоки показали себя более выразительными.

3.3 Эксперименты с временными рядами

Цель данного эксперимента: проверка способности обучения временным зависимостям и экстраполяции временных рядов, сравнение с известными методами работы с временными рядами (RNN).

Авторами был предложен подход для моделирования временных рядов. Модель представляет каждый временной ряд траекторией в некотором пространстве (называемым латентным). Каждая траектория определяется начальным состоянием z_{t_0} и набором глобальных динамик, задаваемых нейросетями и одинаковыми для всех временных рядов в датасете. Начальное состояние сэмплируется из некоторого распределения $z_{t_0} \sim p(z_{t_0})$. Имея наблюдения в моменты времени $t_0, \ldots, t_N$ координаты траектории в латентном пространстве получаются как численное решение ОДУ:

$$z_{t_1}, \ldots, z_{t_N} = ODESolve(z_{t_0}, f, \theta, t_0, \ldots, t_N).$$

А предсказания получаются сэмплированием из распределения с параметрами θ_x:

$$x_{t_i} \sim p(x|\ z_{t_i}, \theta_x).$$

Функция f не зависит от времени явно, то есть $\frac{\partial z(t)}{\partial t} = f_\theta(z(t))$, тогда ля любой точки траектории в латентном пространстве $z(t)$ существует единственная траектория в латентном пространстве. В качестве модели предлагается использовать вариационный автокодировщик [4]. В качестве энкодера используется RNN сеть, которая

# Наблюдения	30/100	50/100	100/100
RNN	0.3937	0.3202	0.1813
Latent ODE	**0.1642**	**0.1502**	**0.1346**

Таблица 2: Результаты метрики RMSE на тестовом датасете

последовательно обрабатывает наблюдения и выдает параметры распределения для начального состояния. Затем сэмплируется начальное состояние и находятся точки траектории, решая ОДУ назад во времени. Затем максимизируется нижняя оценка на обоснованность.

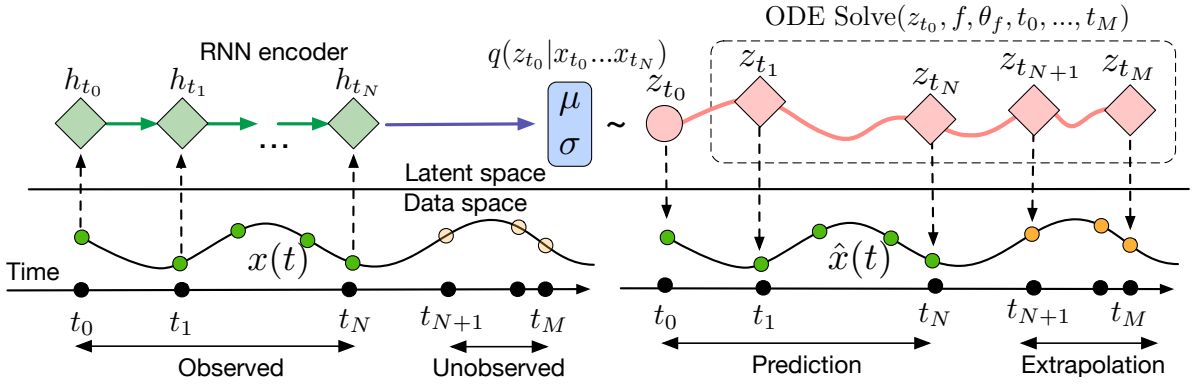

Рис. 7: Граф вычислений модели latent ODE.

Для проведения эксперимента был сгенерирован датасет из 1000 двумерных спиралей, начинающихся в разных точках, затем из каждой спирали сэмплировались 100 точек с равномерным по времени шагом. Половина спиралей шла по часовой стрелке, половина - против. К точкам был добавлен белый шум. Для получения точек с неравномерным временем случайно сэмплировались без возвращения 30, 50 или 100 точек из каждой траектории. В таблице 2 приведены результаты метрики RMSE. Для каждой спирали были сделаны предсказания для 100 новых точек, не входивших в обучающую выборку.

На рисунках 8, 9 показана реконструкция временного ряда RNN моделью, а на рисунках 10, 11 latent ODE моделью. Для реконструкции из latent ODE получены сэмплированием из апостериорного распределения латентных траекторий и декодирование их в точки в исходном пространстве.

Latent ODE построила более точные реконструкции и экстраполяции, независимо от числа наблюдаемых точек и наличия шума.

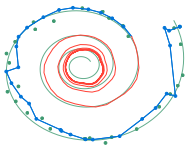

Рис. 8: a

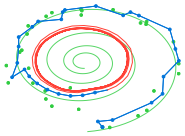

Рис. 9: b

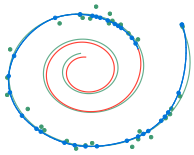

Рис. 10: c

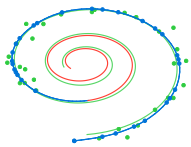

Рис. 11: d

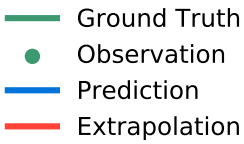

Рис. 12: e

4 Эксперимент: честный MNIST

Как уже отмечалось в разделе 3.1, предложенные модели имеют сравнимое, но худшее качество, но при этом втрое меньшее количество параметров, чем популярная модель ResNet. В качестве модели с близким количеством параметров авторы используют линейный слой и отмечают сильно худшее качество последнего. Сравнение линейной модели и сверточной сети для задачи классификации изображений не корректно, даже если количество параметров моделей одинаково. Оно было бы относительно корректно если число операций было бы сравнимо, но один параметр сверточной сети участвует в множестве сверток, а один параметр линейного слоя лишь однажды умножается. Поэтому была построена модель ResNet с близким числом параметров и была замерено ее качество. В таблице 3 показано, что модель ResNet(2) имеет сравнимое с авторскими моделями число параметров, и одинаковое качество.

	Test Error	# Params	Memory	Time
ResNet(2)	0.42%	0.28 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\boldsymbol{\mathcal{O}(1)}$	$\mathcal{O}(\tilde{L})$

Таблица 3: Результаты на датасете MNIST [2]

Для получения результатов для ResNet(2) можно запустить код honest_mnist.py без параметров. Этот код - модификация авторского примера odenet_mnist.py и от-

личается от него незначительно. Основная цель авторского эксперимента 3.1 не побить качество популярной модели на игрушечном датасете, а показать применимость к задаче классификации изображений и сравнение двух методов обучения.

5 Заключение

Авторами была исследована концепция использования численных методов решения ОДУ как часть модели машинного обучения, была разработана новая модель для работы с временными рядами, оценки плотности и обучения с учителем. Данные модели имеют возможность явно регулировать соотношение точность / время вычисления. Была выведена формула изменения плотности для случая непрерывных нормализующих потоков, которые являются более выразительными и вычислительно эффективными.

Список литературы

[1] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. *CoRR*, abs/1806.07366, 2018.

[2] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. cite arxiv:1312.6114.