Дата поступления рукописи в редакцию: 28.05.2026 г.
Дата принятия рукописи в печать: 19.06.2026 г.
DOI: 10.24412/2224-9133-2026-5-412-422

СЕМЕНОВ Александр Александрович,
научный сотрудник ФКУ НПО «СТиС» МВД России,
e-mail: mail@law-books.ru

ГОНЧАР Владимир Владимирович,
Доцент кафедры информационных технологий и организации расследования киберпреступлений Московская академия Следственного комитета Российской Федерации имени А.Я. Сухарева,
e-mail: vg0778@bk.ru

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ МАШИННОГО ОБУЧЕНИЯ: АРХИТЕКТУРА, ФОРМАТЫ И ПРОБЛЕМЫ СОВМЕСТИМОСТИ

Аннотация. *В статье рассматриваются особенности программного обеспечения для машинного обучения, включая фреймворки и библиотеки. Анализируются основные проблемы совместимости форматов данных, описания нейросетей и разметки датасетов. Описывается внутренняя структура фреймворков, их взаимодействие с библиотеками и особенности работы с различными форматами данных. Особое внимание уделяется проблемам совместимости при описании архитектур нейросетей и преобразовании данных.*

Ключевые слова: *машинное обучение, нейронные сети, фреймворки, библиотеки, форматы данных, совместимость, тензоры, инференс, программная архитектура, описание нейросетей.*

SEMENOV Alexander Alexandrovich,
Chief Researcher, Federal State Budgetary Institution "STIS" of the Ministry of Internal Affairs of Russia

GONCHAR Vladimir Vladimirovich,
Associate Professor, Department of Information Technology and Cybercrime Investigation Moscow Academy of the Investigative Committee of the Russian Federation named after A. Ya. Sukharev

MACHINE LEARNING SOFTWARE: ARCHITECTURE, FORMATS, AND COMPATIBILITY ISSUES

Annotation. *This article examines the features of machine learning software, including frameworks and libraries. It analyzes the main issues of data format compatibility, neural network description, and dataset labeling. It describes the internal structure of frameworks, their interaction with libraries, and the specifics of working with various data formats. Particular attention is paid to compatibility issues when describing neural network architectures and data transformations.*

Key words: *machine learning, neural networks, frameworks, libraries, data formats, compatibility, tensors, inference, software architecture, neural network description.*

Программное обеспечение (далее – ПО) для машинного обучения развивается и существует примерно по тем же законам, что и любое другое, т. е. не обладает каким-то экзотическими особенностями, хоть и имеет свои отличительные черты. Как правило, все программное обеспечение непосредственно для машинного обучения называют либо «фреймворками», либо «библиотеками».

Фреймворк машинного обучения – программное обеспечение, предоставляющее разработчику удобный интерфейс и функционал для составления, обучения, тестирования и инференса нейросетей.

Библиотека машинного обучения – программная библиотека, предоставляющая программисту программный интерфейс (API) для выполнения отдельных функций обучения или инференса нейросетей на конкретном устройстве.

Четкой границы между двумя категориями нет, зачастую одно и то же программное обеспечение можно назвать как фреймворком, так и библиотекой. Однако все же термин «фреймворк» чаще применяют именно для обозначения удобной среды, в которой может работать разработчик нейросетей, а термин «библиотека» наоборот чаще применяют просто к библиотеке с набором функций, играющую для разработчика нейросетей роль наподобие драйвера устройства. Основные вычислительные функции библиотеки машинного обучения, как правило, вызывают вычисления одного слоя, типовой пример функции приведен на *рисунке 1*. Помимо вычислительных функций, в библиотеках, как правило, представлен ряд технических функций для корректного вычисления размеров карт признаков, инициализации переменных и тензоров и т. д.

```
int Library_Convolution_Forward(
 tensor* input,
 tensor* output,
 tensor* filters,
 tensor* biases,
 int kernel_size,
 int stride,
 int padding);
```

Рисунок1. Типовой пример функции.

Наиболее распространенными фреймворками являются Pytorch и Tensorflow, примерами библиотек могут служить CudNN от Nvidia для обучения нейросетей на видеокартах (сокращенно «GPU». Строго говоря, аббревиатура относится к самому процессору, а не видеокарте, но для простоты используется в обоих смыслах) этого производителя, ее аналог MIOpen для видеокарт от AMD и библиотека OpenVINO от Intel для работы с нейросетями на устройствах Intel (главным образом на обычных центральных процессорах, сокращенно CPU). Как правило, библиотеки разрабатываются под конкретный тип устройств, чаще всего это делает сам разработчик устройств.

Отдельно от фреймворков и библиотек существуют также программы, упрощающие процесс создания разметки для датасетов.

Разметчик – программа, предоставляющая интерфейс для удобной и быстрой разметки данных и сохранения их в нужный формат.

В связи с тем, что машинное обучение, развивается децентрализовано, в области программного обеспечения для машинного обучения существует постоянно действующая проблема совместимости используемых форматов. Например, в разметке изображения для задачи детекции можно указывать координаты центра объекта с его размерами или координаты двух противоположных углов прямоугольника; в разметке изображения для задачи сегментации можно указывать принадлежность каждого пикселя к той или иной зоне, а можно – границы зон, и т. д. Разные форматы разметки могут использоваться разными разметчиками, фреймвор-

ками и, собственно, разработчиками нейросетей, из-за чего часто возникает необходимость произвести преобразование разметки из одного формата в другой, что не всегда возможно сделать однозначно.

Еще острее данная проблема проявляется в вопросе описания самих нейросетей. В общем случае описание обученной нейросети должно содержать следующую информацию:

1) взаимосвязи между слоями, т. е. выходы каких слоев являются входами каких слоев, какие слои принимают входные данные извне нейросети и какие наоборот выдают выходные результаты вовне;
2) тип каждого слоя и значения его параметров (под «параметрами» будем понимать все параметры, которые мы не называем весами, т. е. необучаемые параметры, например, размеры ядра свертки сверточного слоя);
3) значения всех весов каждого слоя (как правило, в бинарном формате, не текстовом).

Иногда описание нейросети может содержать не только слои, но и непосредственно массивы данных, возникающих между слоями.

С развитием сверточных нейросетей за массивами данных в машинном обучении закрепился термин «тензор».

Тензор – одномерный или многомерный массив. Термин «тензор» заимствован из традиционных разделов математики, однако без переноса исходного содержания, используется исключительно в качестве синонима слова «массив».

Рассмотрим, например, сверточный слой. Описание сверточного слоя должно содержать размеры ядра свертки, шаг свертки, размеры паддинга, число фильтров. Казалось бы, должны быть также указаны ширина, высота и глубина хотя бы входной карты признаков, но если в описании нейросети указаны тензор, подаваемый на вход слою, и размеры тензора также указаны, то в описании слоя необязательно указывать размеры карт признаков, достаточно сослаться на описание входного тензора. Число весов сверточного слоя не зависит от ширины и высоты входного тензора, и бывают фреймворки, в которых это свойство используется для универсального описания нейросети без указания размера входного изображения. Как следствие, в описании такой нейросети отсутствуют значения ширины и высоты тензоров и сверточных слоев. Данный подход одновременно и добавляет гибкость к описанию, и убавляет ее, поскольку добавление слоя нейросети, число весов которого зависит от размера входного тензора, в таком формате невозможно.

Представим себе теперь, что некоторый разработчик придумал новую модификацию сверточного слоя, которая показала свою эффективность на практике и теперь повсеместно применяется. Чтобы стало возможным описывать данную модификацию в прежнем формате описания нейросетей, необходимо его дополнить. Нужно при этом не трогать существующие параметры сверточного слоя, а только ввести новые, чтобы уже существующие описания нейросетей оставались корректными. Допустим, мы говорим об изобретении свертки с «раздвинутым» ядром. Можно ввести параметр «раздвинутости», указывая число пикселей, пропускаемых по ширине и высоте, и считать, что если эта величина не указана, то она равная нулю, как у обычной свертки (*рисунок 2*).

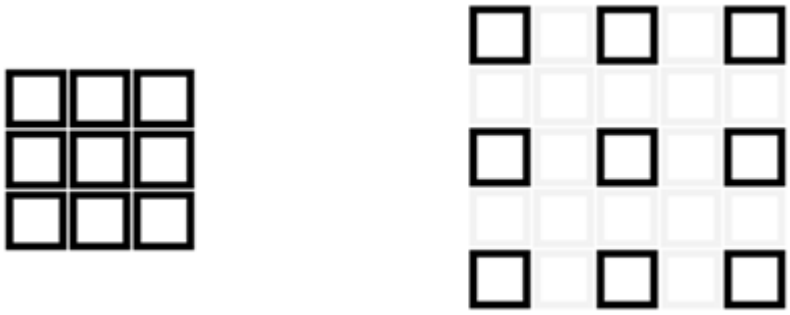

Рисунок 2. Параметр «раздвинутости»

Казалось бы, добавление такого параметра не нарушает никакую логику, но на самом деле оно порождает двусмысленность в том, как правильно читать описание слоя. Легко понять, что для свертки с нулевой «раздвинутостью» объем весов зависит от размеров ядра свертки, и от нее же, с учетом паддинга и шага, зависят размеры выходной карты признаков. Рассмотрим теперь свертку единичной «раздвинутостью», как на рисунке 2. Если отталкиваться от понимания того, что ядро свертки должно характеризовать число весов, то получается, размеры ядра равны 3. Если же смотреть с точки зрения того, как ядро свертки прикладывается к входной карте признаков и сколько раз в нее помещается, то получается, что размеры ядра такой свертки равны 5. В подобных случаях разные разработчики могут использовать разные интерпретации одного и того же описания, результатом чего становятся несовместимости фреймворков, библиотек и описаний.

Проблема несовместимости из-за разницы в описании или трактовки незначительных параметров встречается регулярно. Огромная масса таких проблем порождена одной только логикой понимания паддинга для сверток и пулинга. Например, существовало не менее трех разных способов указания паддинга:

1) указание величин паддиннга отдельно слева, справа, сверху и снизу;
2) указание величин паддинга отдельно для горизонтального (слева и справа) и вертикального (сверху и снизу);
3) указание режима подгона размера выходной карты признаков к четному или нечетному числу.

Представим себе теперь, что разработчик придумал принципиально новый слой, который также оказался востребован мировым сообществом. По идее, для этого слоя должно появиться свое описание в каждом существующем формате. Однако сам разработчик в процессе изобретения, очевидно, не обладал еще этим инструментом, и для него мог быть велик соблазн воспользоваться уже существующими слоями. Аналогично, если разработчик написал статью, где описал алгоритмические особенности нового слоя, но не предложил явного способа его представления в описании нейросети, другие разработчики, которые воспроизведут его эксперимент, могут реализовать такое описание разными способами.

Так, например, для поканальной свертки, существуют и используются два способа ее описания:

1) в виде отдельного слоя, похожего на пулинг, но с весами;
2) виде совокупности нескольких обычных сверточных слоев, каждый из которых принимает на вход карту признаков единичной глубины и производит с помощью единственного фильтра выходную карту признаков единичной глубины.

Обратим внимание на второй вариант. В нем несколько сверточных слоев порождают несколько выходных карт признаков и их, соответственно, перед подачей в следующий слой нужно собрать в единую с помощью слоя склейки, который в первом варианте не возникает. Более того, чтобы интерпретировать входную карту признаков поканальной свертки в качестве нескольких карт признаков соответствующих сверток нужно выполнить обратную склейке операцию, т. е. нарезать карту признаков на несколько карт признаков (в англ. источниках используется термин split). Необходимости подобного рода состыковки разных слоев порождают массу своего рода технических слоев, которые выполняют всевозможные умозрительные перекладывания, раскладывания, переставления (часто называемые слоем reshape) и прочие преобразования карт признаков. Особенностью этих технических слоев является то, что они не подразумевают никаких вычислений вообще, их как бы нет в оригинальном чисто алгоритмическом описании нейросети, но они вынуждено появляются в ее описании для того, чтобы его можно было корректно программно обработать.

В свою очередь, корректное описание отдельных таких преобразований требует четкого указания порядка размерностей тензора. Например, если сверточный слой выдал тензор с шириной и высотой, равными 5, и глубиной, равно 10, то для того, чтобы подать этот тензор в полносвязный слой, для которого все входы равнозначны, нужно четко условиться о том, в каком порядке значения этого слоя подаются. Программная реализация слоев будет, соответственно, подразумевать, что использован определенный порядок размерностей тензора и определенный порядок расположения значений тензора в памяти. Некоторым программистам может быть неочевидно, что по соображениям эффективности программы, выполняющие значительные математические вычисления, хранят любые большие массивы данных всегда плотно, одно значение за другим, какова бы ни была размерность массива. Например, матрицу A[5][10] хранят в виде массива из 50 значений a[50]. При этом возникает необходимость условиться о формате хранения, будут ли в массиве храниться значения по столбцам или по строкам матрицы. Соответственно, в зависимости от выбора формата значение матрицы A с индексами i и j может хра-

ниться либо по адресу a[i+5*j], либо по адресу [10*i+j]. Таким образом, возникает необходимость указания или неявного подразумевания для тензоров порядка хранения данных. Для сверточных нейросетей существует более или менее прижившийся способ обозначения этих порядков.

Рассмотрим карту признаков из трех каналов. На *рисунке 3* схематично изображено хранение этой карты признаков в форматах CHW (справа сверху) и HWC (справа снизу) соответственно.

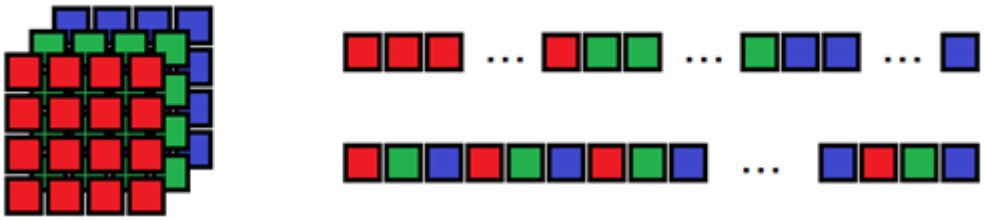

Рисунок 3. Схематичное изображение хранения карты признаков.

NHWC – аббревиатура, означающая порядок хранения данных, при котором значения разных каналов (буква «C» от слова «channel») одного пикселя карты признаков лежат друг за другом подряд, пиксели одной строки (буква «W» от слова «width») лежат друг за другом подряд, строки (буква «H» от слова «height») лежат друг за другом подряд и карты признаков, соответствующие разным входным изображениям одного батча (буква «N», выбранная без причины), лежат подряд. Для обозначения других форматов применяется аналогичная аббревиатура из тех же букв, расставленных в другом порядке, соответствующему порядку расположения данных также, как в данном определении. При единичном батче буква «N» опускается.

Представим себе теперь, что две такие карты признаков необходимо склеить в единую карту признаков с шестью каналами с сохранением формата. Для формата CHW такая операция достигается просто хранением двух массивов друг за другом. Однако для формата HWC потребуется после каждых трех значений первой карты признаков воткнуть три аналогичных значения второй карты признаков, т. е. выполнить существенные перемещения данных в памяти. Этот пример показывает, как из полностью умозрительного действия с нулевым алгоритмическим содержанием при программной реализации нейросетей появляется целая нетривиальная процедура, отнимающая время. В самых неудачных случаях время, затрачиваемое компьютером на подобные перекладывания, может составлять 99 % и более всего времени выполнения вычислений нейросети. Может показаться, что из приведенных соображений следует, что формат CHW лучше формата HWC. Однако такой однозначный вывод сделать нельзя, достаточно представить себе, как вычленяются значения из входной карты признаков для выполнения свертки 3×3 с глубиной 128: в формате HWC потребуется прочесть три массива по 384 значения, а в формате CHW – 384 массива по три пикселя, что компьютер сделает в несколько раз медленнее.

Вернемся к примеру, в котором разработчик изобрел новый слой. Среди разработчиков в области машинного обучения сложилась мода на использование языка Python, обусловленная его простотой. Интерфейс фреймворков Pytorch и Tensorflow для языка Python представляет собой библиотеку функций. В контексте программирования будем использовать общий термин «функция» для обозначения как функций C-подобных языков, так и методов, подпрограмм и т. д. других языков. Соответственно, регулярно возникают случаи, когда разработчик, придумавший новый слой, реализовал его вручную, т. е. просто запрограммировал соответствующие вычисления. При этом такой код может идти вперемешку с вызовами вычислений стандартных для фреймворка слоев. Сохранение подобной нейросети в лучшем случае возможно только в специальном формате, используемом фреймворком, и лишь им же может

быть воспроизведено. При этом о совместимости такого описания с другими существующими форматами описания нейросетей не может быть и речи. Соответственно, в данных ситуациях нейросеть в лучшем случае распространяется с авторским кодом, в худшем случае возникают разные реализации, откуда снова вытекают проблемы совместимости. В некоторых случаях, когда стандартные для фреймворка слои расположены единым куском, а авторские модификации приспособлены ко входу или выходу нейросети, нейросеть может распространяться без авторских модификаций. Так, нейросеть SSD, которая впервые решила задачу детекции. Фильтрация сгенерированных нейросетью баундин-боксов, существенно влияющая на конечный результат, с одной стороны, являлась неотъемлемой частью решения, предложенного разработчиками, но, с другой стороны, не могла быть описана в виде стандартных на тот момент форматов описания нейросетей и, соответственно, передана с ней в одном файле. В результате разные разработчики, повторявшие обучение нейросети по оригинальной статье, создавали отличающиеся реализации и, как следствие, получали разные результаты.

Формат описания нейросети неявно определяет алгоритмические ограничения на слои нейросети и особенности программного исполнения их вычислений. Несоответствие этим ограничениям или их неправильная трактовка приводят к проблемам совместимости, которые в некоторых случаях решаются правильным конвертированием из одного формата в другой, а в некоторых случаях могут и вовсе быть неразрешимыми. Подобного рода проблемы в практике машинного обучения возникают регулярно.

Наиболее широко известным и распространенным форматом описания нейросетей, не привязанным ни к какому фреймворку, является ONNX10F [16].

Из-за названных выше особенностей описание нейросети является ключевым объектом в любом программном обеспечении для машинного обучения, а разница во фреймворках и библиотеках как раз заключается в том, какие действия, как и в каком порядке эти фреймворки и библиотеки позволяют выполнять над нейросетью. Поэтому понимание устройства любого программного обеспечения для машинного обучения заключается в понимании того, что внутри него происходит с нейросетью.

Разберемся в этой логике в общем виде. Нет никакого строгого деления фреймворка на части, но можно условно говорить, что он разделен на интерфейсную и основную части. В общем случае фреймворк может также включать в себя одну или несколько библиотек, либо же быть стыкуемым с одной или несколькими внешними библиотеками. Общая схема, отражающая возможные действия с нейросетью с использованием фреймворка и библиотек, приведена на *рисунке 4*.

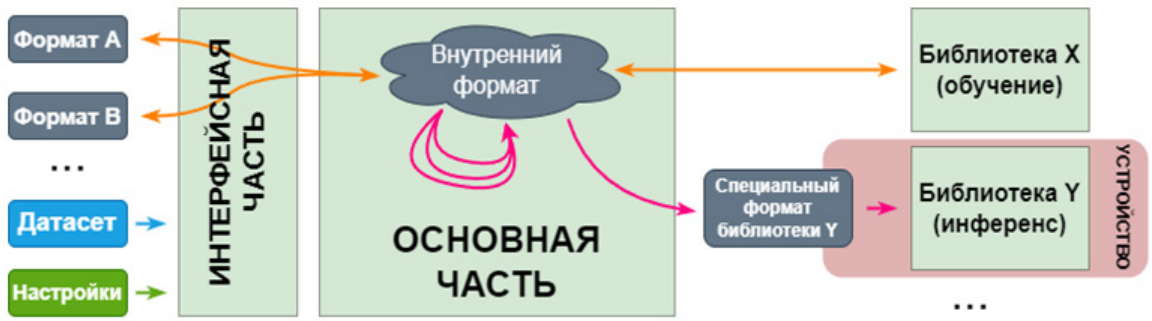

Рисунок 4. Общая схема, отражающая возможные действия с нейросетью с использованием фреймворка и библиотек.

Как было указано выше, использование определенного формата означает неявное использование некоторой логики нейросети. При этом во время разработки фреймворка в него самого закладывалась некоторая логика, и нейросеть внутри фреймворка хранится в его некотором внутреннем формате, который этой логике полностью соответствует. Внутренний формат может совпадать с одним из распространенных форматов описания нейросетей, быть уникальным или просто представлять собой совокупность разбросанных по массивам и переменным параметров и весов нейросети. В идеальном случае поддержка фреймворком некоторого формата означает, что этот формат полностью совместим с внутренним форматом фреймворка и их логики не противоречат друг другу.

В реальности противоречия все же встречаются, в особенности в связи с внесением изменений в версии фреймворков, библиотек или самих форматов, в лучшем случае они выявлены заранее и перечислены в документации к фреймворку. Вероятность этого увеличивается, если фреймворк поддерживает несколько форматов. Можно условно определить, что интерфейсная часть заканчивается и основная часть начинается там, где переданное во фреймворк из файла описание нейросети корректно преобразовано во внутренний формат. Интерфейсная часть, кроме того, предоставляет пользователю возможность передать датасет (как данные, так и разметку) и всяческие настройки, в том числе настройки машинного обучения. Фреймворки могут поддерживать разные форматы разметки, однако с ними нет такой путаницы, как с форматами описания нейросетей, поэтому мы не останавливаемся здесь на этом вопросе.

Для запуска машинного обучения фреймворк обращается к функциям библиотеки, приспособленной для обучения (на рисунке 4 это библиотека X). Опять же, каждая библиотека также подразумевает некоторую логику работы с нейросетью, которая может соответствовать или не соответствовать внутренней логике фреймворка; поддержка фреймворком библиотеки в идеале означает, что его внутренняя логика совместима с логикой библиотеки, но на практике противоречия встречаются как в виде заранее описанных границ совместимости, так и в виде ошибок. Характерным примером является создание AMD библиотеки MIOpen. Идея была в том, чтобы сделать точную копию библиотеки CudNN от Nvidia, поэтому все функции были абсолютно одинаковыми, отличались только названия. Однако запуск даже такой элементарной нейросети, как VGG, с помощью одного и того же кода приводил к кардинально разным результатам для двух библиотек. Причиной было то, что в функции для вычислений слоя пулинга в двух библиотеках использовались разные формулы подсчета высоты и ширины выходной карты признаков, и, таким образом, выходные карты признаков слоя пулинга для двух библиотек отличались не просто значениями, но даже и размерами. Как видно, данная ошибка могла бы быть исключена на уровне элементарного теста. То, что ошибка все же была допущена, хорошо иллюстрирует то, как ошибки совместимости могут появляться во фреймворках и библиотеках, даже если они разрабатываются крупнейшими мировыми компаниями.

В зависимости от библиотеки, базовые действия, выполняемые в связи с обучением нейросети, в той или иной степени реализуется внутри библиотеки или вне ее, т. е. внутри фреймворка (или вручную, если никакой фреймворк не используется). К ним относятся:

1) выделение массивов памяти для хранения промежуточных данных;
2) определение порядка выполнения вычислений слоев;
3) определение конкретных вариантов программной реализации каждого слоя, если таковых в библиотеке несколько.

Соответственно, интерфейс (API) библиотеки зависит от того, какие действия вынесены за ее границы, а также от функционала и поддержки совместимости с разными фреймворками и форматами.

Помимо функциональности и совместимости, ключевым качеством библиотеки является способность обеспечить выполнение вычислений как можно быстрее. Именно необходимость достижения высокой скорости служит причиной разработки отдельных библиотек для разных устройств. При этом обеспечение максимальной скорости вычислений при широчайшей поддержке всевозможных форматов и фреймворков потребует от библиотеки содержания огромного множества оптимизированных функций, разработать их все может быть трудоемко. Поэтому разработчики библиотек ограничивают функционал теми границами, в которых они способны оптимизировать все вычислительные функции.

После выполнения обучения нейросеть приводится к одному из поддерживаемых форматов описания и сохраняется в файл. Весь цикл из указанных выше преобразований проиллюстрирован на рисунке 4 оранжевыми стрелками, двунаправленность стрелок иллюстрирует «перетекание» описания нейросети из файла в некоторые данные библиотеки и обратно. Для инференса можно воспользоваться тем же маршрутом, но в одну сторону, т. е. прочесть нейросеть и вызы-

вать только функции, осуществляющие прямой ход. На первый взгляд, это естественно, и в простых случаях так и поступают. Однако инференс по числу и характеру действий проще обучения, что позволяет гораздо лучше оптимизировать вычисления для инференса, в том числе применить некоторые оптимизации, недоступные при обучении. В связи с этим выделяются отдельно библиотеки для инференса, их использование несколько усложняет логику обработки нейросети во фреймворке. Наиболее важные отличия инференса от обучения:

1) размер батча, который может существенно влиять на результат обучения, на инференсе уже не имеет никакого алгоритмического значения и может быть выбран по другим соображениям;
2) промежуточные значения карт признаков, которые необходимо хранить для применения метода обратного распространения ошибки, на инференсе хранить не нужно;
3) некоторые слои в инференсе оказывается возможным объединить для целей более эффективной программной реализации (самый распространенный пример – объединение сверточного слоя и ReLU);
4) некоторые слои, играющие важную роль во время обучения, в инференсе вырождаются в более простые (например, нормализация) или вовсе исчезают (например, дропаут). Во время обучения слой нормализации приводит все значения к заданному диапазону, в конце обучения последние использованные в слое нормализации коэффициенты замораживаются и далее используются в инференсе неизменно, поэтому вычисления слоя нормализации выражаются в применение к каждому значению операции вида a*x+b;
5) исключение вычислений обратного хода позволяет проще определиться с порядком хранения данных в тензорах, в том числе оптимизировать его под особенности конечного устройства;
6) веса, изменяемые во время обучения, в инференсе являются фиксированными значениями, которые можно заранее преобразовывать, раскладывать и т. д.;
7) зная значения весов, можно анализировать диапазоны возникающих в картах признаках значений и использовать более компактные типы данных, чем использовались при обучении. Обычно при обучении нейросетей используют плавающую точку одинарной точности (float/fp32). Мировое сообщество делало попытки перейти на использование плавающей точки половинной точности (half float/fp16), но обнаружило, что при огромном количестве чисел в современных нейросетях и разницей между ними процесс обучения чувствителен к точности используемого типа, и отказ от fp32 в пользу fp16 означает в некоторых ситуациях значительное снижение достижимой точности, а в отдельных случаях и вовсе делает обучение невозможным. Как следствие, нет никаких предпосылок к переходу к fp16, стандартом машинного обучения де-факто является fp32, хотя в частных случаях для обучения пробуют использовать другие форматы.

Перечисленные отличия делают возможными массу оптимизаций. Обратим внимание, что нейросеть никогда не учится для того, чтобы быть использованной единожды. Напротив, количество ее использований исчисляется тысячами или миллионами. С другой стороны, оптимизации применяются единственный раз и используются одинаково при каждом запуске. Отсюда вытекает достаточно естественная идея разделить подготовку нейросети к запуску в библиотеке и, собственно, сами вычисления в библиотеке. Поскольку инференс может выполняться не на том же устройстве, что и обучение, то возникает необходимость согласовать библиотеку, выполняющую вычисления на устройстве, с функциями фреймворка, подготавливающими нейросеть к исполнению в библиотеке. Это делается через еще одно описание нейросети в специальном формате, специфическом именно для используемой библиотеки.

В некоторых библиотеках подобное разделение на подготовку и запуск не видно программисту. Например, в библиотеке CudNN подготовка к вычислениям спрятана от программиста в функциях, выполняющих вычисления, при этом эта подготовка выполняется с первым запуском нейросети, в результате чего время первого запуска может быть в тысячи раз больше времени любого последующего. Nvidia это сделала для того, чтобы по содержимому описания нейросети в специальном формате нельзя было догадаться о каких-то особенностях библиотеки или даже самого устройства. Перед сохранением нейросети в специальном формате над ней выполняются преобразования, зачастую нетривиальные. Часть фреймворка, способную выполнять такие преобразования, иногда называют нейросетевым компилятором. Существуют в том числе разработчики, пытающиеся создать самостоятельные нейросетевые компиляторы, не привязанные к конкретным библиотекам и соответствующим им устройствам.

Специальным форматам библиотек для инференса характерны следующие свойства:

1) преобразование в специальный формат выполняется только в одну сторону и не предполагает возможности восстановления по нему описания нейросети;
2) специальный формат не содержит предназначенной для чтения человеком информации, поэтому информация в нем хранится не в текстовом, а бинарном виде;
3) все веса и параметры нейросети разложены так, как удобно функциям библиотеки, поэтому они могут быть частично перемешаны, продублированы, разбавлены нулями и т. д., при этом ложатся в память единым куском, адреса нужных каждому слою данных также заранее просчитаны и содержатся в описании.

Нейросеть в специальном формате передается на устройство, где предполагается ее использование. В библиотеке предусмотрен своего рода плеер, который умеет прочитывать из файла с описанием нейросети описания слоев один за другим и вызывать функции библиотеки, выполняющие соответствующие вычисления. Внешняя программа посредством функций библиотеки осуществляет прочтение нейросети из файла, передачу входных данных для нейросети, запуск инференса и получение его результатов.

Описанная выше схема справедлива для всех фреймворков и библиотек с той лишь поправкой, что многие фреймворки и библиотеки содержат не весь перечисленный функционал. Бывают, например, фреймворки только для инференса, библиотеки с нейросетевым компилятором, но без удобного пользовательского интерфейса и т. д. Следует понимать, что в нетривиальном исследовательском процессе, особенно при экспериментах с новыми слоями, разработчик нейросетей вынужден часть операций программировать вручную. Таким образом, если для обучения и инференса нейросетей под конкретную задачу, уже много раз решенную мировым сообществом, почти наверняка найдутся подходящие фреймворк и/или библиотека, то для разработки новых нейросетей и решения принципиально новых задач никакой фреймворк или библиотека никогда не будут достаточны.

Список литературы:

[1] Chen Zhang, Joohee Kim Video object detection with two-path convolutional LSTM pyramid. 2016. – URL: https://www.researchgate.net/publication /343702579_Video_Object_Detection_With_ Two-Path_ Convolutional_LSTM_ Pyramid/fulltext/5f3b2658928 51cd302013482/Video-Object-Detection-With-Two-Path-Convolutional-LSTM-Pyramid.pdf/ (дата обращения: 16.01.2026).

[2] Hanson A., Koutilya PNVR, Sanjukta Krishnagopal, Larry Davis Bidirectional Convolutional LSTM for the Detection of Violence in Videos. 2018. – URL: https://openaccess.thecvf.com/content_eccv_2018_workshops/w10 /html/Hanson_Bidirectional_Convolutional_LSTM_or_the_ Detection_of_Violence_in_ Videos_ ECCVW_2018_paper.html (дата обращения: 15.01.2026).

[3] Howard A.G., Menglong Zhu, Bo Chen, Kalenichenko D. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. – URL: https://arxiv.org/pdf/1704.04861/ (дата обращения: 11.01.2026).

[4] Iandola F.N., Han S., Moskewicz M.W., Ashraf K., Dally W.J., Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size Recognition. 2016. Published as a conference paper at ICLR. 2015. – URL: https://www. researchgate.net/publication/ 301878495_SqueezeNet_AlexNet-level_accuracy_ with_50x_fewer_ parameters_ and_05MB_model_size (дата обращения: 10.01.2026).

[5] Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015. – URL: https://arxiv.org/pdf/1502.03167/ (дата обращения: 13.01.2026).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition. 2015. – URL: https://arxiv.org/pdf/1512.03385 (дата обращения: 11.01.2026).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Spatial pyramid pooling in deep convolutional networks for visual recognition. 2015. – URL: https://arxiv.org/pdf/1406.4729/ (дата обращения: 11.01.2026).

[8] Krizhevsky I., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. 2012. – URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/
c399862d3b9d6b76c8436e 924a68c45b-Paper.pdf (дата обращения: 12.01.2026).

[9] Lavin A., Scott Gray Fast Algorithms for Convolutional Neural Networks. 2015. – URL: https://arxiv.org/abs/1509.09308 (дата обращения: 11.01.2026).

[10] Mikolov T., Kai Chen, Corrado G., Jeffrey Dean Efficient Estimation of Word Representations in Vector Space. 2013. – URL: https://arxiv.org/abs/1301.3781 (дата обращения: 15.01.2026).

[11] Norman P. Jouppi, Cliff Young, Nishant Patil and others In-Datacenter Performance Analysis of a Tensor Processing Unit. 2017. – URL: https://arxiv.org/pdf/1704.04760 (дата обращения: 11.01.2026).

[12] Ronneberger O., Fischer Ph., Brox Th. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. – URL: https://arxiv.org/pdf/1505.04597/ (дата обращения: 13.01.2026).

[13] Szegedy C., Wei Liu, Chapel Hill, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going Deeper with Convolutions. 2014. – URL: https://arxiv.org/pdf/1409.4842 (дата обращения: 12.01.2026)

[14] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015. Published as a conference paper at ICLR 2015. – URL: https://arxiv.org/pdf/1409.1556 (дата обращения: 10.01.2026).

[15] Wei Liu, Anguelov D., Erhan D., Szegedy C., Reed S., Cheng-Yang Fu, Berg A.C. SSD: Single Shot MultiBox Detector. 2016. – URL: https://arxiv.org/pdf/1512.02325 (дата обращения: 11.01.2026).

[16] Официальный сайт ONNX. – URL: https://onnx.ai/.

[17] Основы технологий искусственного интеллекта: учебное пособие/ А.А. Семенов, В.В. Гончар, А.С. Эрдниев. – Московский университет МВД России имени В.Я, Кикотя, 2025. – 190с.

[18] Финансовый мониторинг: учебное пособие / В.И. Глотов, А.У. Альбеков, О.Н. Тисен и др. – КноРус, 2022. – 198 с.

[19] Тисен О.Н. Противодействие современным приемам и способам вербовки в международные террористические организации // Российская юстиция. 2018, №9. С. 51-54.

[20] Овчинский А.С., Аветисян К.Р. Аудиальные воздействия на участников массовых мероприятий как объект административно-правового регулирования// Вестник Московского университета МВД. 2015. № 1-. С. 293-297.

[21] Зиборов О.В., Аветисян К.Р. Комплексный подход при противоборстве организации массовых беспорядков// Вестник Московского университета МВД. 2023. № 7. С. 100-104.

References:

[1] Chen Zhang, Joohee Kim Video object detection with two-path convolutional LSTM pyramid. 2016. – URL: https://www.researchgate.net/publication /343702579_Video_Object_Detection_With_ Two-Path_ Convolutional_LSTM_ Pyramid/fulltext/5f3b2658928 51cd302013482/Video-Object-Detection-With-Two-Path-Convolutional-LSTM-Pyramid.pdf/ (дата обращения: 16.01.2026).

[2] Hanson A., Koutilya PNVR, Sanjukta Krishnagopal, Larry Davis Bidirectional Convolutional LSTM for the Detection of Violence in Videos. 2018. – URL: https://openaccess.thecvf.com/content_eccv_2018_workshops/w10/html/Hanson_Bidirectional_Convolutional_LSTM_or_the_ Detection_of_ Violence_in_ Videos_ ECCVW_2018_paper.html (дата обращения: 15.01.2026).

[3] Howard A.G., Menglong Zhu, Bo Chen, Kalenichenko D. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. - URL: https://arxiv.org/pdf/1704.04861/ (access date: 11.01.2026).

[4] Iandola F.N., Han S., Moskewicz M.W., Ashraf K., Dally W.J., Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size Recognition. 2016. Published as a conference paper at ICLR. 2015. – URL: https://www. researchgate.net/publication/ 301878495_ SqueezeNet_AlexNet-level_accuracy_ with_50x_ fewer_ parameters_ and_05MB_model_size (дата обращения: 10.01.2026).

[5] Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015. – URL: https://arxiv.org/pdf/1502.03167/ (access date: 13.01.2026).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition. 2015. – URL: https://arxiv.org/pdf/1512.03385 (access date: 11.01.2026).

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Spatial pyramid pooling in deep convolutional networks for visual recognition. 2015. - URL: https://arxiv.org/pdf/1406.4729/ (access date: 11.01.2026).

[8] Krizhevsky I., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks. 2012. - URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d-6b76c8436e 924a68c45b-Paper.pdf (accessed on: 12.01.2026).

[9] Lavin A., Scott Gray Fast Algorithms for Convolutional Neural Networks. 2015. - URL: https://arxiv.org/abs/1509.09308 (access date: 11.01.2026).

[10] Mikolov T., Kai Chen, Corrado G., Jeffrey Dean Efficient Estimation of Word Representations in Vector Space. 2013. – URL: https://arxiv.org/abs/1301.3781 (access date: 15.01.2026).

[11] Norman P. Jouppi, Cliff Young, Nishant Patil and others In-Datacenter Performance Analysis of a Tensor Processing Unit. 2017. – URL: https://arxiv.org/pdf/1704.04760 (access date: 11.01.2026).

[12] Ronneberger O., Fischer Ph., Brox Th. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. – URL: https://arxiv.org/pdf/1505.04597/ (access date: 13.01.2026).

[13] Szegedy C., Wei Liu, Chapel Hill, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going Deeper with Convolutions. 2014. – URL: https://arxiv.org/pdf/1409.4842 (accessed on: 12.01.2026)

[14] Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015. Published as a conference paper at ICLR 2015. - URL: https://arxiv.org/pdf/1409.1556 (access date: 10.01.2026).

[15] Wei Liu, Anguelov D., Erhan D., Szegedy C., Reed S., Cheng-Yang Fu, Berg A.C. SSD: Single Shot MultiBox Detector. 2016. – URL: https://arxiv.org/pdf/1512.02325 (access date: 11.01.2026).

[16] ONNX official website. – URL: https://onnx.ai/.

[17] Fundamentals of artificial intelligence technologies: a textbook/A.A. Semenov, V.V. Gonchar, A.S. Erdniev. - Moscow University of the Ministry of Internal Affairs of Russia named after V.Ya, Kikotya, 2025. - 190s.

[18] Financial monitoring: training manual/V.I. Glotov, A.U. Albekov, O.N. Tisen et al. - KnoRus, 2022. - 198 s.

[19] Tisen O.N. Countering modern methods and methods of recruitment to international terrorist organizations//Russian Justice. 2018, №9. S. 51-54.

[20] Ovchinsky A.S., Avetisyan K.R. Audio effects on participants of mass events as an object of administrative and legal regulation//Bulletin of the Moscow University of the Ministry of Internal Affairs. 2015. № 1-. S. 293-297.

[21] Ziborov O.V., Avetisyan K.R. An integrated approach to confronting the organization of mass riots//Bulletin of the Moscow University of the Ministry of Internal Affairs. 2023. № 7. S. 100-104.