При использовании материалов статьи необходимо использовать данную ссылку:
Пчелинцев С.Ю. Сравнительный анализ фреймворков глубокого обучения // Информационно-экономические аспекты стандартизации и технического регулирования. 2020. № 1. (53). С. 41-51

УДК 004.93;

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ФРЕЙМВОРКОВ ГЛУБОКОГО ОБУЧЕНИЯ

Пчелинцев С.Ю.

В статье сравнивается несколько популярных фреймворков глубокого обучения в контексте их применения для решения задач в режиме реального времени, с высоким уровнем точности получаемых результатов, при ограниченных вычислительных ресурсах, а также с возможностью использования во встраиваемых и портативных платформах. Типичным примером такой задачи является распознавание дорожных знаков. Исследования проводились с использованием различных размеров входных изображений и включали в себя оценки скорости обучения и точности выходных данных для различных фреймворков при изменении размеров изображения. В качестве набора данных использовался популярный датасет GTSRB, содержащий изображения дорожных знаков. Результаты указываются как для вычислений, производимых с использованием графического ускорителя, так и без него. Также описана оптимизация, используемая для более точной подстройки фреймворков под решение конкретной задачи. В ходе исследования обнаружено, что наибольшая точность выходных данных получена при использовании фреймворка TensorFlow. При этом наиболее высокая скорость обучения в сочетании с высокой общей точностью выходных данных достигались при использовании фреймворков MXNet и Neon. В ходе исследования также установлено, что для используемого тестового набора данных размер изображения из выбранной области интересов не всегда влиял на точность выходных данных, кроме того, использование глубоких моделей с большим временем обучения, в частности ResNet-32, также не всегда способствовало увеличению точности вывода.

Ключевые слова: нейронные сети, искусственный интеллект, компьютерное зрение, глубокое обучение.

ВВЕДЕНИЕ.

Глубокое обучение лежит в основе многих задач в области вычислений, связанных с самоуправляемыми автомобилями. Данный метод обучения включает в себя модели глубоких сверточных нейронных сетей (СНС), которые обычно обучаются мощными компьютерами централизованно, а затем обученные модели передаются во встраиваемые системы, устанавливаемые на автомобили и работающие в режиме реального времени. Поскольку встроенные системы имеют ограниченные вычислительные ресурсы, а решаемые задачи требуют результатов в реальном времени с высокой точностью, чтобы обеспечить безопасность водителя, пассажиров и пешеходов, требуются системы, основанные на глубоком обучении, которые могут достичь высокоточных выходных результатов в реальном времени при ограниченных ресурсах. Чтобы эффективно использовать ограниченные вычислительные ресурсы встраиваемых систем для достижения результатов в реальном времени и с высокой точностью следует использовать механизм параллельных вычислений [1]. Существует несколько популярных фреймворков, позволяющих заниматься глубоким обучением моделей, используя возможности параллелизма. Чтобы выбрать наиболее подходящий фреймворк, необходимо произвести их сравнение, попутно исследовав возникающие в ходе сравнения компромиссы.

Новизна данного исследования заключается в сравнении пяти популярных фреймворков по трем моделям сверточных нейронных сетей с тремя различными размерами входных данных, комбинация которых не была изучена в предыдущих исследованиях. Исследование зависимости

Пчелинцев Сергей Юрьевич, аспирант, Тамбовский Государственный Университет имени Г.Р. Державина
г. Тамбов

времени и точности обучения от размеров изображения проводилось, поскольку в задаче распознавания дорожных знаков размер области интересов не имеет фиксированного значения. В ходе подготовки исследования для сравниваемых фреймворков были разработаны методы оптимизации, используемые в данной работе и ориентированные на применение в будущих исследованиях данных фреймворков. Результаты данной работы могут помочь ученым и инженерам в выборе и точной настройке наиболее подходящих из представленных фреймворков под нужды их исследований.

ОСНОВНАЯ ПРОБЛЕМАТИКА СТАТЬИ.

Важное ограничение сверточных нейронных сетей заключается в том, что размер входного изображения должен быть фиксирован после того, как сеть определена и обучена [2-3]. Однако в реальных приложениях на этапе обнаружения задачи распознавания дорожного знака, выполняемой до этапа классификации, обычно предлагаются области интереса с различными размерами. Крайне важно определиться с величиной, в пределах которой могут меняться размеры области интересов, а также определить размер входного изображения нейронной сети поскольку:

1. Обработка большого изображения может быть недоступна, ввиду ограниченности доступных ресурсов.
2. Низкоуровневая реализация и оптимизация вычислений фреймворка обычно неизвестны конечным пользователям.
3. Слишком большой или слишком маленький размер изображения может повлиять на точность вывода.

Кроме того, желательно минимизировать время, затрачиваемое на обучение модели. Поэтому при сравнивании точности выходных данных и производительности, то есть скорости обучения, основное внимание уделяется разным размерам изображений.

В рамках данного исследования результаты собираются при проведении вычислений фреймворками как с использованием исключительно центрального процессора (ЦП), так и с использованием графического процессора (ГП). Хотя в настоящее время использование графических ускорителей для обучения сверточных нейронных сетей предпочтительно, результаты о работе на ЦП представлены, поскольку в будущем обучение с использованием новых центральных процессоров (например, включающих специальные сопроцессоры, такие как Intel Xeon Phi) может быть предпочтительным, если они обеспечивают более энергоэффективные и высокопроизводительные вычисления, чем ГП.

Для проведения исследования были отобраны пять фреймворков: CNTK, Tensorflow, PyTorch, Neon и MXNet. Поскольку исследование сосредоточено на распознавании дорожных знаков, в ходе него использовался широко известный набор данных – немецкий бенчмарк для распознавания дорожных знаков GTSRB, содержащий собственно изображения дорожных знаков. Кроме того, были использованы три модели нейронных сетей. Это модель, именуемая по названию разработавшего её института IDSIA, а также две глубокие остаточные сверточные нейронные сети ResNet-20 и ResNet-32. Эти модели требуют небольшой размер входных изображений и обеспечивают высокую точность выходных данных [4].

Используемые в работе фреймворки в настоящее время являются популярными и применяются как в технологической среде, так и в академической.

1. Microsoft Cognitive Toolkit (CNTK) – это унифицированный фреймворк глубокого обучения, который позволяет создавать нейронные сети популярных типов, таких как сверточные нейронные сети, прямые нейронные сети, рекуррентные нейронные сети, а также сети типа долгая краткосрочная память через ориентированные вычислительные графы. CNTK считается фреймворком с высочайшей производительностью для обучения нейронных сетей с использованием нескольких процессоров.

2. Apache MXNet, разработанный Amazon, – это фреймворк глубокого обучения с высокой переносимостью между языками программирования. С MXNet можно сочетать декларативное и императивное программирование для максимизации эффективности и производительности, поскольку оба типа выполнения распараллелены и оптимизированы. В MXNet также заявлена отличная масштабируемость при выполнении на нескольких машинах.

3. Neon, базовый фреймворк нейронных сетей Intel Nervana, включает в себя оптимизацию для различного оборудования, но наиболее эффективно работает на процессорах Intel. Он тесно интегрирован с библиотекой машинного обучения Intel MKL, а также с новейшими библиотеками графического ядра. Для Neon заявлена самая высокая производительность среди библиотек глубокого обучения для проведения быстрой итерации и исследования моделей.

4. PyTorch, разработанный Facebook AI, представляет собой библиотеку Python для глубокого обучения, которая обеспечивает вычисление тензора с использованием графического ускорителя, а также имеет силь-ную нативную поддержку градиентного вычисления функций и перемен-ных, определенных внутри фреймворка. Такая техника, при которой гра-диенты автоматически рассчитываются для произвольных вычислений, называется автоматическим (иногда алгоритмическим) дифференцированием. Однако, в PyTorch, в отличие от многих других фреймворков, нет статически вычисляемых графов, поэтому отсутствует возможность добавлять градиентные узлы уже после того, как определены остальные вычисления. Вместо этого PyTorch приходится записывать или прослеживать поток значений через программу по мере их поступления, то есть строить расчетный граф динамически. Как только такой граф будет записан, у PyTorch будет информация, нужная для обратного обхода такого потока вычислений и расчета градиентов выходных значений на базе входных.

5. TensorFlow, изначально представленный как фреймворк Google Brain для машинного обучения и исследований глубоких нейронных сетей, вскоре стал широко используемой средой глубокого обучения и в настоящее время обладает одним из крупнейших сообществ среди фреймворков глубокого обучения. Он известен своей гибкой архитектурой и простым развертыванием на различном оборудовании [5].

Набор данных GTSRB является популярным набором данных о дорожных знаках. Он появился в результате работы Международной совместной конференции по нейронным сетям 2011 года и поддерживается Рурским университетом в Бохуме. В общей сложности он содержит более 50000 изображений, включающих 43 различных класса дорожных знаков. Изображения имеют размер от 15×15 до 250×250. Каждое изображение содержит дорожный знак. Тесно ограниченная область интересов каждого изображения описывается в наборе данных для дальнейшего кадрирования. GTSRB предоставляет изображения как в представлении RGB, так и в представлениях Haar, HueHist и HOG.

Обучаемая в ходе исследования модель IDSIA обеспечивает точность классификации более 99% при надлежащей предварительной обработке. Данной модель показала, что при соответствующей предварительной обработке сверточные нейронные сети могут достигать высокой точности даже при небольшом количестве слоев [6]. Структура модели IDSIA показана в таблице 1.

Таблица 1.

Структура IDSIA модели

Слой	Тип	Количество карт и нейронов	Размерность ядра
0	Входной	3 карты, 48×48 нейронов	
1	Сверточный	100 карт, 46×46 нейронов	3×3
2	Объединяющий	100 карт, 23×23 нейронов	2×2
3	Сверточный	150 карт, 20×20 нейронов	4×4
4	Объединяющий	150 карт, 10×10 нейронов	2×2
5	Сверточный	250 карт, 8×8 нейронов	3×3
6	Объединяющий	250 карт, 4×4 нейронов	2×2
7	Полносвязный	230 нейронов	
8	Полносвязный	43 нейрона	

Кроме того, в исследованиях использовалась более современная модель ResNet. Она предназначена для решения проблемы взрывного роста градиента и снимает со сверточных сетей ограничение глубины. Кроме того, ResNet имеет ряд особенностей. Так ее входные размеры являются гибкими. Это позволяет более широкое применение данной модели на практике. Количество параметров модели ResNet сравнительно мало. Например, ResNet32 с размером входного сигнала 32×32 имеет только 0,46 миллиона параметров, а модели AlexNet и VGG имеют 60 миллионов и 138 миллионов параметров соответственно [7]. ResNet состоит из повторяющихся базовых блоков, изображенных на рисунке 1.

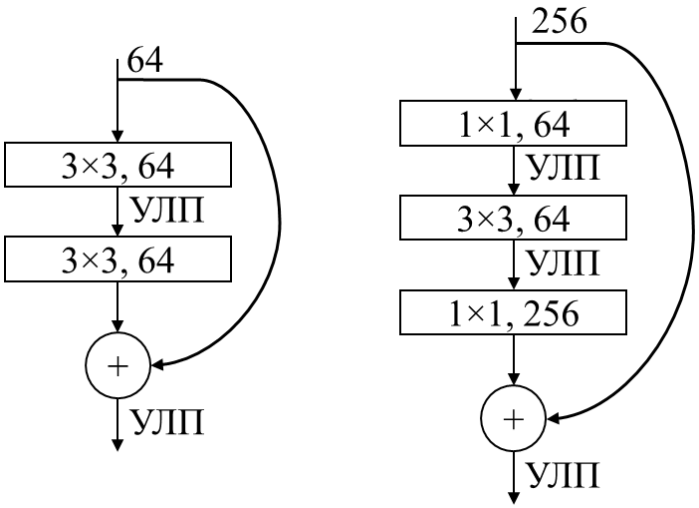

Рисунок 1. Базовые блоки построения модели ResNet, где УЛП – усеченное линейное преобразование

Все эксперименты проводились на персональном компьютере, оснащенном операционной системой семейства Linux Mint (версия 19.1), процессором Intel core i5 4570 3.2 ГГц, графическим ускорителем NVidia GeForce GTX1060 6 Гб, также 24 Гб оперативной памяти. Весь код написан на языке программирования Python 3.5 с использованием как стандартных, так и дополнительных библиотек, в частности MKL 2018.0.0, PyCuda 2017.1.1 и PyGPU 0.7.5. Использовались следующие версии исследуемых фреймворков: CNTK 2.7.0, Neon 2.6.0, MXNet 1.5.0, PyTorch 1.3, и Tensor flow 2.7.0.

Как правило, изображения, входящие в наборы данных для распознавания дорожных знаков, имеют небольшой, но различающийся размер, а также поделены на относительно небольшое количество классов, менее 100. Эти изображения обычно имеют различающиеся условия освещения, которые требуют проведения некоторой предварительной обработки. Для улучшения качества изображения в качестве предварительной обработки можно провести выравнивание гистограммы изображения. Сделать это необходимо прежде, чем изображения будут использованы в качестве исходных данных для обучения сверточной нейронной сети [8]. Исходные изображения сначала обрезаются, а затем преобразуются до размеров 32×32, 64×64 (только для сетей ResNet) и 48×48 (для сетей IDSIA и ResNet). Эти изображения дополнительно обрабатываются с помощью адаптивного выравнивания с ограниченной контрастностью (CLAHE). Применение CLAHE привело к снижению частоты появления ошибок. Более сложная предварительная обработка на цветовых каналах не проводилась, поскольку при выполнении логического вывода в режиме реального времени дополнительная предварительная обработка добавит нагрузку на производительность.

Общее количество слоев модели ResNet, l, можно представить, как:

$$l = 6n + 2, \qquad (1)$$

где n – это нечетное число, обычно $n > 3$.

В ходе экспериментов использовались n = 3 и n = 5, для построения моделей ResNet-20 и ResNet-32 соответственно. Поскольку сторона изображения для карты признаков будет уменьшена вдвое в модели ResNet, для нее требуется, чтобы размер входного сигнала был кратен 4. Целесообразно использовать 32, 48 и 64 в качестве размеров входных изображений после изменения их размера. Реализации ResNet перенесены из официальной модели «зоопарк» для каждого фреймворка. Необходимые изменения в код вносятся, чтобы гарантировать, что изображения разных размеров можно корректно подавать в каждую модель и что глобальный слой подвыборки в конце работает корректно.

ПРЕДВАРИТЕЛЬНАЯ НАСТРОЙКА ФРЕЙМВОРКОВ.

В экспериментах используется алгоритм стохастического градиентного спуска с шагом обучения 0,01 и нормой изменения 0,9 для оптимизатора, размер пакета данных составляет 64, число эпох равно 25. Также используется порядок каналов BGR и размер изображения (3, n, n), где n = 32, 48 и 64. Веса сверточных слоев инициализируются с помощью инициализатора

he_normal (также называемого kaiming_normal), обеспечивающего лучшую сходимость. Для получения максимальной производительности для каждого фреймворка, особенно с использованием только процессора, все они оптимизируются индивидуально, путем сборки из исходного кода и настройке с помощью Python. Обычно выполнение этих оптимизаций повышает производительность путем добавления поддержки MKL или использования преимуществ локальной машины (например, использования нескольких ЦП или расширенных наборов инструкций, таких как SSE и AVX). Для ускорения процесса, каждый фреймворк тестируется с одной эпохой обучения с помощью модели IDSIA в наборе данных GTSRB с размером обучающего набора 31367. Размер входного изображения, который используется в этих тестах, составляет 48×48, а размер пакета данных – 64. Для повышения производительности фреймворков, как с использованием исключительно процессора, так и с использованием графического ускорителя, применяется системная оптимизация. Для повышения скорости и точности этих фреймворков, используется также оптимизация кода. Более подробно оптимизации описаны ниже.

Скорость работы MXNet на центральном процессоре повышается с 219,9 секунд на эпоху до 114,4 секунд на эпоху, путем установки флагов USE_MKL=1 и USE_MKL_EXPERIMENTAL=1 перед сборкой или, в качестве альтернативы, установкой пакета mxnet-cu90mkl.

Производительность фреймворка Neon повышается со 198,8 секунд на эпоху до 141,5 секунд на эпоху установкой для OMP_NUM_THREADS количества физических ядер локальной машины (4 для тестируемой конфигурации), для KMP_AFFINITY – значений compact, 1, 0, для granularity – значения fine, как подсказывает Intel Nervana.

Скорость обучения TensorFlow на центральном процессоре улучшена с 298,6 секунд на эпоху до 167,8 секунд на эпоху после установки пакета Python, подготовленного TinyMind и оптимизированного с поддержкой SSE4.1, SSE4.2, AVX, AVX2, FMA и MKL, доступных на локальной машине. Стоит отметить, что для OMP_NUM_THREADS и KMP_AFFINITY также должны быть установлены те же значения, которые были установлены для Neon с целью достижения этого ускорения.

Поскольку CNTK и PyTorch по умолчанию поставляются с поддержкой MKL, таких оптимизаций для них не требуется.

Поскольку в настоящее время почти все основные фреймворки используют библиотеку cuDNN для вычислений на основе CUDA с использованием графического процессора, при условии, что cuDNN и CUDA правильно установлены и сконфигурированы, сборка фреймворков из исходного кода не повлияет на их производительность. Единственным исключением является PyTorch. Его производительность на графическом ядре была повышена с 22,4 секунд на эпоху до 21,1 секунды на эпоху после добавления поддержки LAPACK установкой библиотеки magma-cuda90. Стоит отметить, что упомянутые выше методы оптимизации могут повлиять на скорость сходимости обучения. Однако, поскольку количество используемых эпох гарантирует, что модели будут перегружены в проводимых тестах, нет необходимости учитывать влияние различных скоростей сходимости на точность выходных данных.

Также можно оптимизировать код для повышения скорости и точности. Для CNTK, перед передачей в конструктор источника минипакета, NumPy массивы данных преобразуются в непрерывные массивы, если они не предназначены для повышения эффективности вычислений. Поскольку точное измерение времени не может быть сделано просто с помощью обратных вызовов, передаваемых в общую обучающую функцию, необходимо создать цикл обучения для более гибкого измерения времени. Это может привести к потере некоторых преимуществ скорости CNTK (особенно с использованием графического процессора). На самом деле, предполагается, что вместо создания потребителя цикла обучения для каждой эпохи, функцию обучения следует использовать для более высокой скорости обучения. Другое отличие CNTK состоит в том, что он суммирует градиент каждого минипакета для обновлений параметров, а не усредняет минипакеты, как это делают другие фреймворки, например TensorFlow. CNTK также отличается от других фреймворков тем, что он нормализует выигрыш для обучаемого методом стохастического градиентного спуска с нормой изменения. Эти две особенности CNTK обычно снижают его точность по сравнению с другими фреймворками. Для решения этой проблемы устанавливаются значения параметров unit_gain=False и use_mean_gradient=True, что уравнивает общую точность CNTK и других фреймворков.

Также были изучены возможности оптимизации кода для PyTorch. Один

необязательный параметр функции Dataloader, num_working, может быть настроен на использование нескольких потоков для загрузки данных. Попытки увеличить число потоков от 0 (по умолчанию) до 4 и 16 привели к снижению скорости обучения с 21,1 секунды на эпоху до 22,3 секунд на эпоху и 23,4 секунд на эпоху, что противоречит ожиданиям. Причиной этому является то, что по сравнению с огромными наборами данных, такими как ImageNet, использованные в исследовании наборы данных имеют гораздо меньший размер, для которого накладные расходы на многопоточность преобладают во время выполнения кода. В результате в экспериментах использовалось значение по умолчанию.

РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ.

На рисунке 2 представлены графики, демонстрирующие время обучения модели ResNet-20 для всех пяти фреймворков с разными размерами входных данных и с использованием вычислительной мощности как графического (левый график), так и центрального (правый график) процессоров. Аналогичным образом представлены результаты экспериментов для модели ResNet-32 на рисунке 3. Время обучения всех трех моделей для фиксированного размера входных данных (48×48) представлено на рисунке 4.

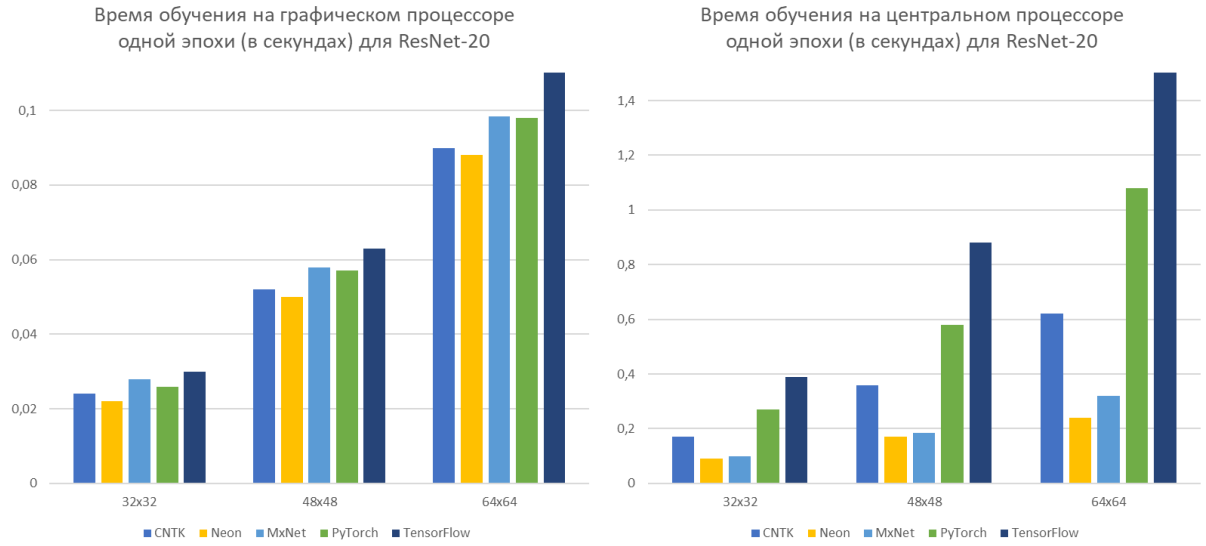

Рисунок 2. Время обучения модели ResNet-20

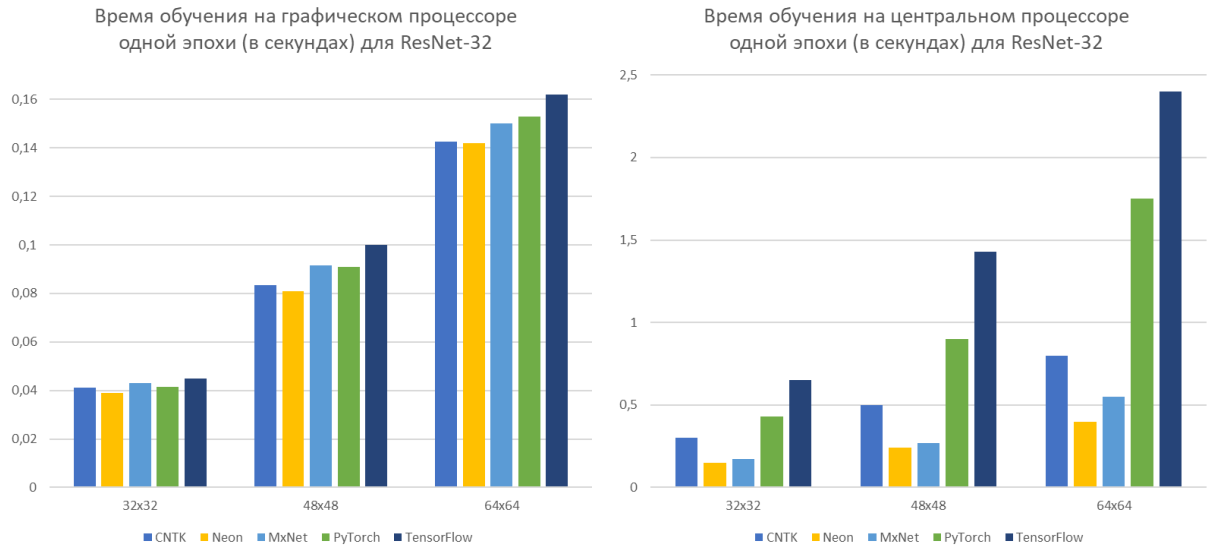

Рисунок 3. Время обучения модели ResNet-32

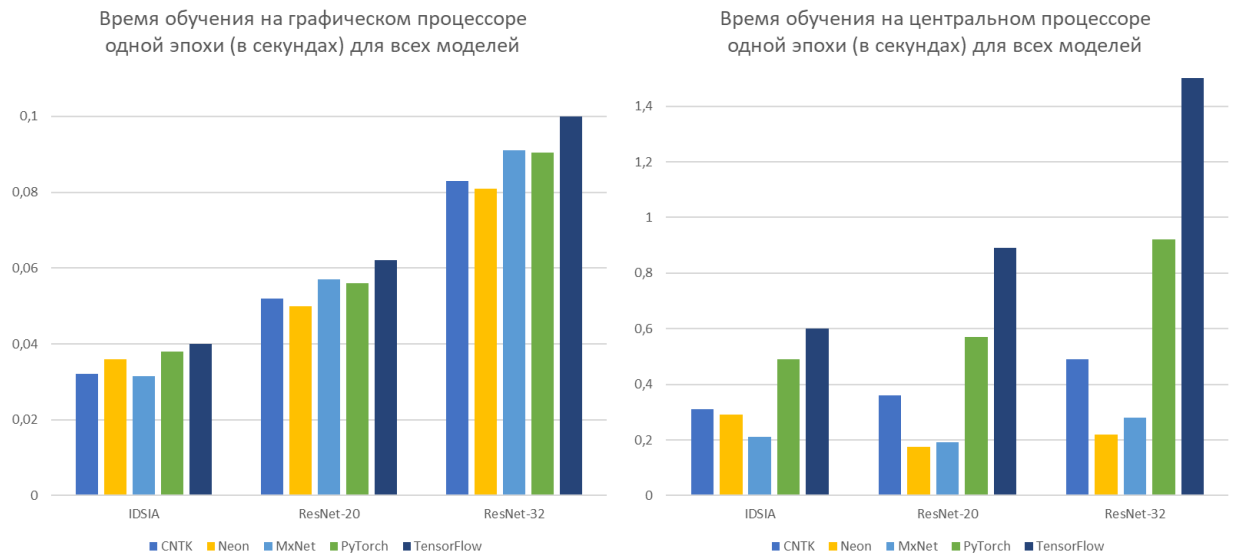

Рисунок 4. Время обучения всех моделей

Общее время обучения, а также точность выходных данных для двух моделей ResNet как при работе на центральном процессоре, так и на графическом представлены в таблице 2 для размера входных данных 32×32, в таблице 3 – для размера 64×64.

Общее время обучения, а также точность выходных данных для всех трех моделей при размере входных данных 48×48 представлены в таблице 4 для обучения на центральном процессоре, в таблице 5 – для обучения на графическом процессоре.

Таблица 2.

Время обучения и точность вывода для входного размера 32×32

Фреймворк	ЦП				ГП			
	ResNet-20		ResNet-32		ResNet-20		ResNet-32	
	вр. об., с	точн.	вр. об., с	точн.	вр. об., с	точн.	вр. об., с	точн.
CNTK	313.80	96.93%	502.60	97.24%	1731.30	97.63%	2582.35	97.12%
Neon	288.65	97.68%	461.50	97.72%	888.14	97.54%	1434.08	97.78%
MxNet	344.82	96.61%	531.36	96.26%	983.94	97.30%	1560.08	96.40%
PyTorch	319.53	94.00%	509.17	97.00%	3372.83	96.00%	5420.62	97.00%
TensorFlow	357.01	97.32%	558.48	97.60%	3964.77	97.63%	6260.26	97.85%

Таблица 3.

Время обучения и точность вывода для входного размера 64×64

Фреймворк	ЦП				ГП			
	ResNet-20		ResNet-32		ResNet-20		ResNet-32	
	вр. об., с	точн.	вр. об., с	точн.	вр. об., с	точн.	вр. об., с	точн.
CNTK	313.80	96.93%	502.60	97.24%	1731.30	97.63%	2582.35	97.12%
Neon	288.65	97.68%	461.50	97.72%	888.14	97.54%	1434.08	97.78%
MxNet	344.82	96.61%	531.36	96.26%	983.94	97.30%	1560.08	96.40%
PyTorch	319.53	94.00%	509.17	97.00%	3372.83	96.00%	5420.62	97.00%
TensorFlow	357.01	97.32%	558.48	97.60%	3964.77	97.63%	6260.26	97.85%

Таблица 4.

Время обучения и точность вывода при обучении на ЦП

Фреймворк	ЦП					
	IDSIA		ResNet-20		ResNet-32	
	вр. об., с	точн.	вр. об., с	точн.	вр. об., с	точн.
CNTK	393.40	96.78%	655.72	96.37%	1047.60	96.64%
Neon	432.20	95.87%	618.46	97.35%	1001.42	98.27%
MxNet	392.70	96.55%	712.59	96.80%	1116.57	97.32%
PyTorch	455.50	96.00%	697.94	97.00%	1114.56	96.00%
TensorFlow	491.92	96.37%	778.71	97.36%	1220.48	97.26%

Таблица 5.

Время обучения и точность вывода при обучении на ГП

Фреймворк	ГП					
	IDSIA		ResNet-20		ResNet-32	
	вр. об., с	точн.	вр. об., с	точн.	вр. об., с	точн.
CNTK	3657.14	96.45%	4041.97	96.63%	5994.42	96.77%
Neon	3557.70	96.07%	1877.23	97.49%	3057.35	98.02%
MxNet	2711.96	96.76%	2268.31	97.36%	3605.97	97.14%
PyTorch	6029.23	96.00%	7037.74	97.00%	11368.58	97.00%
TensorFlow	6355.15	96.29%	8086.93	97.77%	12268.18	97.49%

Результаты измерения скорости обучения на графическом процессоре показывают, что Neon является самым быстрым в большинстве случаев. Есть только одно исключение: MXNet выигрывает при работе с моделью IDSIA, когда размер входного изображения равен 48×48. В общем, время выполнения всех фреймворков довольно близко друг к другу. В большинстве тестовых случаев скорость обучения фреймворков имеет следующий нисходящий порядок: Neon, CNTK, PyTorch, MXNet и TensorFlow.

Результаты измерения скорости обучения на центральном процессоре показывают, что Neon и MXNet имеют самую высокую скорость. Neon занимает первое место в большинстве случаев, за исключением модели IDSIA, где его превосходит MXNet. Интересным открытием является то, что время обучения MXNet и Neon для модели IDSIA больше, чем для модели ResNet-20, которая является более глубокой, чем IDSIA. Причиной может быть то, что для этих фреймворков массивные вычисления на двух конечных полносвязных уровнях могут быть не оптимизированы для центральных процессоров так же эффективно, как для графических процессоров. По сравнению с результатами работы на графическом процессоре, где время выполнения CNTK очень близко к Neon и MXNet, в режиме работы на центральном процессоре CNTK не масштабируется так же хорошо, как Neon и MXNet, когда модель становится глубже, а размер входного сигнала увеличивается. Как и в случае CNTK, время выполнения PyTorch и TensorFlow на центральном процессоре не так близко к Neon и MXNet, как на графическом. В целом, за исключением случая с моделью IDSIA, скорость обучения фреймворков имеет следующий нисходящий порядок: Neon, MXNet, CNTK, PyTorch и TensorFlow.

Neon и TensorFlow обеспечивают наилучшуювыходных данных. Каждый из них лидирует в трех из семи исследованных случаях в то время, как CNTK лидирует в оставшемся случае. Тем не менее, разрыв между точностью их результатов и точностью результатов других фреймворков не очень велик. Для всех фреймворков две модели ResNet имеют явное преимущество перед моделью IDSIA, как и ожидалось. Однако ни одна модель ResNet не доминирует над другой. Среди всех трех размеров входных данных, которые тестировались, наилучшая точность в среднем была зафиксирована в эксперименте с размерностью 64×64 для модели ResNet-32. Тем не менее какого-то одного размера, постоянно приводящего к доминирующей точности результатов, не наблюдалось.

При обучении на центральном процессоре результаты аналогичны результатам работы графическом процессоре. Neon и TensorFlow по-прежнему обеспечивают лучшую точность выходных данных. Один или другой занимает первое место во всех случаях, за исключением случая с моделью IDSIA, где первое место занимает MXNet. Две модели ResNet также, как и ожидалось, превосходят модель IDSIA в то время, как между собой их точность почти одинакова. Кроме того, как и при работе с графическим процессором, выявлено отсутствие входного размера, имеющего доминирующую точность вывода.

Основываясь на этих результатах, можно сделать вывод, что среди пяти фреймворков Neon имеет самую высокую среднюю скорость обучения. Точность его выходных данных также является одной из лучших наряду с TensorFlow, который имеет более медленную скорость обучения на при работе, как с графическим, так и с центральным процессором. MXNet также обеспечивает выдающуюся производительность, сравнимую с результатами Neon. У CNTK, PyTorch и MXNet скорость обучения очень близка к скорости Neon, а точность вывода близка к точности TensorFlow с использованием графического процессора. Однако CNTK, PyTorch и TensorFlow страдают от снижения скорости обучения на центральном процессоре. В целом с точки зрения как скорости обучения, так и точности логического вывода, Neon и MX Net – наиболее подходящие фреймворки для обучения свёрточной нейронной сети как с использованием графического процессора, так и с использованием центрального процессора.

Как с использованием графического ускорителя, так и без него, в проводимых экспериментах две модели ResNet достигают более высокой точности, чем модель IDSIA, как и ожидалось. ResNet-32 не доминирует над ResNet-20, они обе имеют одинаковую точность. Это объясняется перегрузкой, так как ResNet-32 может быть излишне большим используемого набора данных. Таким образом, для наборов данных, аналогичных GTSRB, не имеет смысла использовать модели более глубокие, чем ResNet-20, поскольку они потребуют более длительного времени обучения и, в свою очередь, могут не привести к каким-либо улучшениям в точности вывода. Стоит также отметить, что изменение размера входных данных не обязательно влияет на точность вывода и масштабируемость обучающей среды ResNet-20 и ResNet-32 в наборе данных GTSRB. Следовательно, увеличение размера для области интересов может не привести к повышению точности вывода. Для проведенных экспериментов 32×32 – это оптимальный размер входных изображений для обучения набора данных о дорожных знаках без ущерба для точности.

Кроме того, для каждого фреймворка точность вывода при работе на центральном процессоре и на графическом процессоре различна при одинаковом размере входных данных и одинаковой архитектуре модели. Это связано с тем, что процесс обучения на центральном процессоре, как правило, является детерминированным (путем фиксирования случайного начального числа), а процесс обучения на графическом процессоре, как правило, таковым не является (по крайней мере, в связи с тем, что планирование потоковых блоков для процессоров не является детерминированным). Путем фиксирования случайного начального числа в обучении с центральным процессором можно воспроизводить результаты обучения на центральном процессоре, но нельзя делать то же самое на графическом процессоре, и, следовательно, нельзя генерировать идентичные предварительно обученные модели для разных устройств. Фактически, в проведенных экспериментах ни один фреймворк не может воспроизвести результаты обучения на графическом процессоре. CNTK является единственным фреймворком, который может воспроизвести результаты обучения начальных 20–30 эпох путем форсирования детерминированных алгоритмов, но в дальнейшем значения потерь слишком отклоняются. Форсирование детерминированных алгоритмов также снижает скорость обучения: при запуске теста длительностью в одну эпоху, можно обнаружить, что этот параметр увеличивает время обучения одной эпохи CNTK с 16,6 секунд до 29,4 секунд на графическом процессоре.

Наконец, некоторые фреймворки показали более длительное время обучения на центральном процессоре для модели IDSIA, чем при обучении для модели ResNet-20, возможно, из-за неоптимизированных размеров карты признаков и массивных вычислений на полносвязных уровнях. Это позволяет предположить, что сверточные нейронные сети, не участвовавшие в исследовании, но имеющие схожую архитектуру, при проведении аналогичного эксперимента могут показать схожие результаты, в особенности при обучении на центральном процессоре с использованием данных фреймворков.

ЗАКЛЮЧЕНИЕ.

В данном исследовании измеряется скорость обучения и точность выходных данных для пяти фреймворков глубокого обучения: CNTK, MXNet, Neon, PyTorch и TensorFlow. Обучение производится на трех моделях, которые подходят для получения высокоточных результатов в режиме реального времени при ограниченных вычислительных ресурсах: IDSIA, ResNet-20, и ResNet-32. Размеры входных данных варьируются. Также представлены методы оптимизации, которые были использованы для каждого фреймворка, и показаны результаты работы фреймворков как на центральном процессоре, так и с использованием графического процессора. В экспериментах исследовано влияние использования разных размеров изображений на результаты, что очень важно для моделей, которые могут быть использованы в задачах распознавания дорожных знаков с различными размерами области интересов. Данные исследования позволяют выявить и оптимизировать наиболее подходящий фреймворк с точки зрения времени выполнения и точности вывода для обучаемых моделей, предназначенных для распознавания дорожных знаков.

Таким образом, в целом Neon и MXNet имеют самые высокие скорость обучения и точность вывода во всех тестовых случаях. Кроме того, наблюдалось, что TensorFlow имеет одну из самых высоких степеней точности среди всех пяти фреймворков как при работе на центральном, так и на графическом процессорах. Поскольку изменение размера входных данных не обязательно влияет на точность вывода, при обучении ResNet-20 и ResNet-32 для набора данных GTSRB предлагается использовать 32×32 в качестве размера входного изображения для классификации дорожных знаков, что является вычислительно экономичным без ущерба для точности. Сравнение моделей также показало, что ResNet-32 имеет почти такую же точность вывода, что и ResNet-20. Это указывает на то, что, хотя ResNet32 и другие более глубокие модели имеют намного более длительное время обучения, чем более мелкие модели, в свою очередь они могут не обеспечить улучшения точности выходных данных для наборов данных, которые имеют характеристики, аналогичные GTSRB.

Дальнейший план исследований предполагает развертывание обучаемых моделей для обработки данных в режиме реального времени на устройствах, а также исследование производительности и точности вывода для развернутых моделей. Для лучшего обнаружения дорожных знаков планируется обучать сеть на наборах данных, больших, чем GTSRB. Расширение набора данных для обучения модели приведёт к новым сложностям, для решения которых предстоит исследовать возможности по предварительной обработке данных. Существует множество универсальных алгоритмов и решений для предварительной обработки изображений. Их сочетание вкупе с реализациями, встроенными в фреймворки глубоко обучения, позволит успешно обрабатывать большие наборы данных о дорожных знаках. iea

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ:

1. Тихонов А.А. Большие данные и глубокое машинное обучение в искусственных нейронных сетях. // Наука и образование сегодня. 2018. № 6. С. 35-38.
2. Петров С.П. Сверточная нейронная сеть для распознавания символов номерного знака автомобиля // Системный анализ в науке и образовании. 2013. № 3. С. 66-73.
3. Созыкин А.В. Обзор методов обучения глубоких нейронных сетей. // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 3. С. 28-59.
4. Ciresan D., Meier U., Schmidhuber J. Multicolumn Deep Neural Networks for Image Classification. In Proceedings of the 25th IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Washington. IEEE Computer Society. 2012. pp. 3642-3649.
5. Латкин И.И. Обзор возможностей TensorFlow для решения задач машинного обучения. // Молодежный научно-технический вестник. 2016. № 9. С. 27-32.
6. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe. NIPS. 2012. pp. 1097-1105.
7. Xing F., Yuan Y., Huo H., Fang T., Homeostasis-Based CNN-to-SNN Conversion of Inception and Residual Architectures. Neural Information Processing: 26th International Conference, ICONIP 2019. Sydney. Springer Nature. 2019, pp. 173-174.
8. Hoffman J., Guadarrama S., Tzeng E.S., Hu R., Donahue J., Girshick R., Darrell T, Saenko K. LSDA: Large scale detection through adaptation. Advances in Neural Information Processing Systems. Montreal. NIPS 2014. pp. 3536-3544.

COMPARATIVE ANALYSIS OF DEEP LEARNING FRAMEWORKS

Pchelintsev Sergey Y., applicant, Tambov State Technical University named after G.R. Derzhavin, Tambov

The article compares several popular deep learning frameworks in the context of their application for building a road sign recognition system. The size of the input data in the experiments varied from 32×32 to 64×64. Three different models of convolutional neural networks were used for comparison. Network training was performed separately on the CPU and on the GPU. As a result of the experiments Neon and MXNet demonstrated the highest training speed and best output data accuracy. In addition, TensorFlow reached very good results. It is also note that using a higher input data resolution did not lead to a significant increase in accuracy, but had a negative impact on performance. A similar conclusion can be made about the depth of the model. The future research plan includes training models on a larger data set and integration with the video capture system.

Key words: neural networks, artificial intelligence, computer vision, deep learning.

REFERENCES:

1. Tikhonov A.A. Bol'shie dannye i glubokoe mashinnoe obuchenie v iskus-stvennyh nejronnyh setyah. [*Big data and deep machine learning in artificial neural networks*]. Nauka i obrazovanie segodnya. 2018, no. 6, pp. 35-38.
2. Petrov S.P. Svertochnaya neyronnaya set’ dlya raspoznavaniya simvolov nomernogo znaka avtomobilya [*Convolutional neural network for recognition symbols of car license plate*]. Sistemnyj analiz v nauke i obrazovanii. 2012, no. 3, pp. 66-73.
3. Sozykin A.V. Obzor metodov obucheniya glubokih nejronnyh setej [*An Overview of Methods for Deep Learning in Neural Networks*]. Vestnik JuUrGU. Seriya: Vychislitel'naya matematika i informatika. 2017, vol. 6, no. 3. pp. 28-59.
4. Ciresan D., Meier U., Schmidhuber J. Multicolumn Deep Neural Networks for Image Classification. In Proceedings of the 25th IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Washington. IEEE Computer Society. 2012. pp. 3642-3649.
5. Latkin I.I. Obzor vozmozhnostej TensorFlow dlya resheniya zadach mashinnogo obucheniya [*The review of TensorFlow’s opportunities for solving machine learning tasks*]. Molodezhnyj nauchno-tekhnicheskij vestnik. 2016, no. 9, pp. 27-32.
6. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe. NIPS. 2012. pp. 1097-1105.
7. Xing F., Yuan Y., Huo H., Fang T., Homeostasis-Based CNN-to-SNN Conversion of Inception and Residual Architectures. Neural Information Processing: 26th International Conference, ICONIP 2019. Sydney, Springer Nature, 2019, pp. 173-174.
8. Hoffman J., Guadarrama S., Tzeng E.S., Hu R., Donahue J., GirshickR., Darrell T, Saenko K. LSDA: Large scale detection through adaptation. Advances in Neural Information Processing Systems. Montreal. NIPS 2014. pp. 3536-3544.