Московский Государственный Университет имени М. В. Ломоносова

Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

КУРСОВАЯ РАБОТА

Сравнительный анализ библиотек глубокого обучения Tensorflow и PyTorch

Выполнил:
студента 3 курса 323 группы
Васильев Семён Михайлович

Научный руководитель:
доцент, кандидат. физ.-мат. наук
Попова Нина Николаевна

Москва, 2019

Оглавление

Введение

За последние два десятилетия глубокие нейронные сети привлекли к себе много внимания, так как решают некоторые классы задач, такие как классификация изображений, машинные перевод и распознавание голоса значительно эффективнее традиционных алгоритмов. По этой причине в последнее время наблюдается значительное увеличение числа алгоритмов, фреймворков и платформ предназначенных для обучения глубоких нейронных сетей.

Остро стоит вопрос эффективности алгоритмов, так как для обучения современных сетей требуется обрабатывать огромные объемы данных, а число их параметров может измеряться в миллиардах.

Фреймворки, для построения и обучения глубоких нейронных сетей, по большей части не стандартизированы, что порождает препятствия при проведении их сравнительного анализа на эффективность. Например, между параметрами описания одного и того же слоя нейронной сети в разных фреймворках может не быть взаимно-однозначного соответствия, или один и тот же параметр может иметь различные множества значений.

Все это приводит к необходимости разработки бэнчмарка, который позволил бы выявлять сильные и слабые стороны архитектур нейронных сетей, алгоритмов, применяемых при обучении, и их реализаций в различных фреймворках.

Задачей данной работы является сбор базы данных, включающей набор значений метрик, описывающих процесс обучения нескольких конкретных архитектур нейронных сетей на кластерном комплексе Polus, реализованных на основе фреймворков Tensorflow и Pytorch.

Обзор

На данный момент существует несколько работ, ставящих своей целью проведение бэнчмаркинга и сравнительного анализа фреймворков глубоко обучения, а также реализацию средств помогающих в этом.

Статья « *Benchmarking and Analyzing Deep Neural Network Training*».

Сотрудники университета Торонто опубликовали статью *Benchmarking and Analyzing Deep Neural Network Training*, в которой описывают методологию бэнчмаркинга обучения глубоких нейронных сетей.

В своей работе они предлагают набор метрик, анализ которых позволит сделать выводы об эффективности реализации одного фреймворка относительно других. Набор состоит из следующих метрик: пропускная способность модели — количество сэмплов, которое модель обрабатывает за единицу времени, загруженность GPU - $\frac{GPU\ active\ time * 100}{Total\ time}$%, эффективность выполнения операций с числами с плавающей точкой - $\frac{actual\ flop\ count\ during\ T * 100}{FLOPS\ peak * T}$%, загруженность CPU - $\frac{\sum_c total\ active\ time\ of\ core\ c * 100}{CPU\ core\ count * total\ time}$%, потребление и распределение памяти, для сбора этой метрики авторами были разработаны профайлеры потребления памяти для фреймворков, которые они использовали для проведения эксперимента.

Для проведения эксперимента авторы реализовали модели, список которых приведен в таблице 1, каждая из них решает одну из следующих задач: классификация изображений, машинный перевод, обнаружение объектов, распознавание голоса, обучение с подкреплением, состязательное обучение. Для реализации нейронных сетей авторы использовали фреймворки Tensorflow, MXNet, CNTK.

Application	**Model**	**Implementations**	**Dataset**
Image classification	ResNet-50 Inception-v3	TensorFlow, MXNet, CNTK	ImageNet1K
Machine translation	Seq2Seq Transformer	TensorFlow, MXNet TensorFlow	IWSLT15 WMT-14
Object detection	Faster R-CNN	TensorFlow, MXNet	Pascal VOC 2007
Speech recognition	Deep Speech 2	MXNet	LibriSpeech
Adversarial learning	WGAN	TensorFlow	Downsampled ImageNet
Deep reinforcement learning	A3C	MXNet	Atari 2600

Табл. 1

На основе полученных данных авторами были сделаны некоторые выводы, например, при увеличении размера mini-batch увеличиваются пропускная способность модели и загруженность GPU, для рекуррентных нейронных сетей функция пропускной способности от размера mini-batch не сходить на имеющихся объемах памяти GPU, для сравнения фреймворков необходимо проводить исследования на разных задачах.

Deep500.

Сотрудники Швейцарской высшей технической школы Цюриха разработали мета-фреймворк Deep500 для бэнчмаркинга библиотек глубокого обучения. Deep500 интегрирован с такими фреймворками, как Tensorflow, PyTorch, Caffe2 и MXNet. Он даёт возможность использовать в процессе обучения модули из разных библиотек, поддерживается формат описания нейронных сетей ONNX, позволяет использовать собственные алгоритмы оптимизации и проверять их на корректность с помощью численного дифференцирования. Также Deep500 подсчитывает достаточно обширный набор метрик для различных этапов обучения, который включает в себя время выполнения, количество выполненных FLOPs объем пересылок, расходы, которые накладывает использование того или иного фреймворка.

Постановка задачи

Провести анализ производительности фреймворков построения и обучения нейронных сетей Tensorflow и Pytorch. Провести вычислительный эксперимент и собрать данные о процессе обучения с различными наборами значений гиперпараметров на основе системы IBM PowerAI на кластерном вычислительном комплексе IBM Polus. Разработать программное средство, позволяющий пользователю провести аналогичный эксперимент с возможностью выбора своего набора значений гиперпараметров.

Метод решения задачи

На основе каждого из двух фреймворков Tensorflow и Pytorch были реализованы две сверточные нейронные сети. Они представляют из себя адаптированные под задачи cifar-10 и cifar-100 сети AlexNet и VGG16 соответственно. В сети на основе AlexNet было увеличено количество фильтров в сверточных и количество нейронов в полносвязных слоях для усложнения сети и, как следствие, получения лучшей показательности эксперимента. В сети на основе VGG16 были убраны три последних сверточных слоя и один последний MaxPooling слой из-за слишком маленького размера изображений в выбранных наборах данных. Реализации одной и той же модели на разных фреймворках приведены к единообразному виду.

Эксперимент проводился на различных значениях гиперпараметров: размер mini-batch, оптимизатор, learning rate.

- Значения размера mini-batch: 250, 500, 1000, 2000, 5000 изображений 32x32 пикселя с тремя каналами цвета.
- Виды оптимизаторов: SGD, Adam, RMSProp.
- Значения learning rate: 0.01, 0.001, 0.0001.

Построены две таблицы. В первой таблице (табл. 2) приводятся данные о зависимости средней продолжительности эпохи обучения и пропускной способности модели от размера mini-batch. Во второй (табл. 3) собраны данные о зависимости точности предсказания модели на обучающей выборке на 5, 10, 15, 20, 25 и 30 эпохе обучения в зависимости от выбора оптимизатора и значения параметра learning rate. При проведение эксперимента значения размера mini-batch бралось 250. Пропуски в данных означают, что модель не обучается при заданных параметрах.

Экспериментальное исследование

Собранные данные.

Model	Framework	Optimizer	Batch size	Epoch time	Throughput
AlexNet	Tensorflow	SGD	250	6,83	7322
			500	6,1	8193
			1000	5,29	9451
			2000	5,02	9960
			5000	4,88	10255
		Adam	250	9,53	5242
			500	7,47	6693
			1000	5,96	8387
			2000	5,36	9328
			5000	5,1	9799
		RMSProp	250	8,7	5744
			500	7,01	7129
			1000	5,75	8692
			2000	5,28	9471
			5000	4,94	10116
	PyTorch	SGD	250	6,97	7120
			500	6,4	7801
			1000	5,68	8803
			2000	5,42	9210
			5000	5,15	9707
		Adam	250	8,83	5662
			500	7,34	6810
			1000	6,11	8170
			2000	5,63	8880
			5000	5,23	9558
		RMSProp	250	8,2	6093
			500	6,99	7154
			1000	5,97	8367
			2000	5,52	9045
			5000	5,18	9642
VGG16	Tensorflow	SGD	250	7,76	6441
			500	7,12	7020
			1000	6,64	7525
			2000	6,43	7770
			5000	6,56	7616
		Adam	250	8,23	6073
			500	7,32	6826
			1000	6,78	7366
			2000	6,47	7728
			5000	6,59	7582
		RMSProp	250	8,06	6199

			500	7,21	6929
			1000	6,73	7423
			2000	6,47	7723
			5000	6,61	7561
	PyTorch	SGD	250	11,9	4201
			500	10,9	4579
			1000	10,3	4845
			2000	9,96	5019
			5000	9,63	5192
		Adam	250	12,11	3420
			500	10,99	4546
			1000	10,34	4835
			2000	9,99	5005
			5000	9,65	5178
		RMSProp	250	12,03	4155
			500	10,96	4561
			1000	10,3	4851
			2000	9,93	5034
			5000	9,63	5192

Табл. 2

Model	**Framework**	**Optimizer**	**Learning rate**	**Epoch 5**	**Epoch 10**	**Epoch 15**	**Epoch 20**	**Epoch 25**	**Epoch 30**
AlexNet	Tensorflow	SGD	0,01	0,517	0,657	0,755	0,851	0,917	0.977
			0,001	0,335	0,424	0,479	0,503	0,551	0,578
			0,0001	0,139	0,211	0,264	0,278	0,291	0,307
		Adam	0,01						
			0,001						
			0,0001	0,703	0,823	0,906	0,945	0,97	0,981
		RMSProp	0,01						
			0,001						
			0,0001	0,641	0,85	0,944	0,964	0,975	0,988
	PyTorch	SGD	0,01	0,614	0,727	0,802	0,863	0,913	0,944
			0,001	0,399	0,49	0,539	0,574	0,603	0,628
			0,0001	0,249	0,265	0,289	0,307	0,327	0,346
		Adam	0,01						
			0,001	0,692	0,76	0,804	0,838	0,872	0,885
			0,0001	0,717	0,824	0,894	0,95	0,973	0,98
		RMSProp	0,01						
			0,001	0,478	0,576	0,67	0,775	0,832	0,884
			0,0001	0,709	0,844	0,928	0,958	0,971	0,977
VGG16	Tensorflow	SGD	0,01						
			0,001						
			0,0001						
		Adam	0,01						
			0,001	0,11	0,274	0,378	0,468	0,563	0,636
			0,0001	0,153	0,296	0,395	0,473	0,547	0,633

		RMSProp	0,01						
			0,001	0,068	0,286	0,495	0,651	0,729	0,748
			0,0001	0,098	0,205	0,312	0,438	0,575	0,704
	PyTorch	SGD	0,01	0,029	0,107	0,197	0,285	0,429	0,622
			0,001	0,016	0,024	0,028	0,031	0,034	0,036
			0,0001						
		Adam	0,01						
			0,001	0,211	0,371	0,462	0,529	0,61	0,641
			0,0001	0,29	0,418	0,539	0,685	0,817	0,892
		RMSProp	0,01						
			0,001	0,124	0,275	0,443	0,626	0,731	0,785
			0,0001	0,225	0,373	0,484	0,586	0,685	0,773

Табл. 3

Анализ.

Из данных можно сделать следующие выводы.

Производительность модели зависит от размера mini-batch. Для всех приведенных реализаций нейронных сетей и всех оптимизаторов с увеличением размера mini-batch увеличивается пропускная способность модели. Это связано с меньшим количеством пересылок между основной памятью и GPU. На рис. 1 изображена зависимость пропускной способности от размера mini-batch двух реализаций модели AlexNet с оптимизатором SGD.

Обе реализации AlexNet показали примерно одинаковую производительность, в то время как, реализация VGG16 на Tensorflow имеет пропускную способность примерно в полтора раза большую, чем на PyTorch. В VGG .значительно больше сверточных слоёв, чем в AlexNet. Возможно, такая разница в производительности появилась из-за того, что операция свертки в PyTorch реализована менее эффективно, чем в Tensorflow. На рис. 2 изображена зависимость пропускной способности от размера mini-batch двух реализаций модели AlexNet и VGG16 с оптимизатором SGD.

В обоих фреймворках реализация алгоритма SGD имеет большую производительность, чем RMSProp, в среднем на 5%, а RMSProp в свою очередь быстрее Adam, в среднем на 4%. На рис. 3 изображена зависимость пропускной способности от размера mini-batch двух реализаций модели AlexNet с оптимизаторами SGD, Adam, RMSProp.

Модели, реализованные на PyTorch, в среднем сходятся за меньшее число эпох. Также существуют пары оптимизатор — значение learning rate, при которых модели, реализованные на PyTorch обучаются, а на Tensorflow нет. На рис. 4 изображена зависимость точности на обучающей выборке от номера эпохи двух реализаций модели VGG16 с оптимизатором RMSProp и значением learning rate 0.0001.

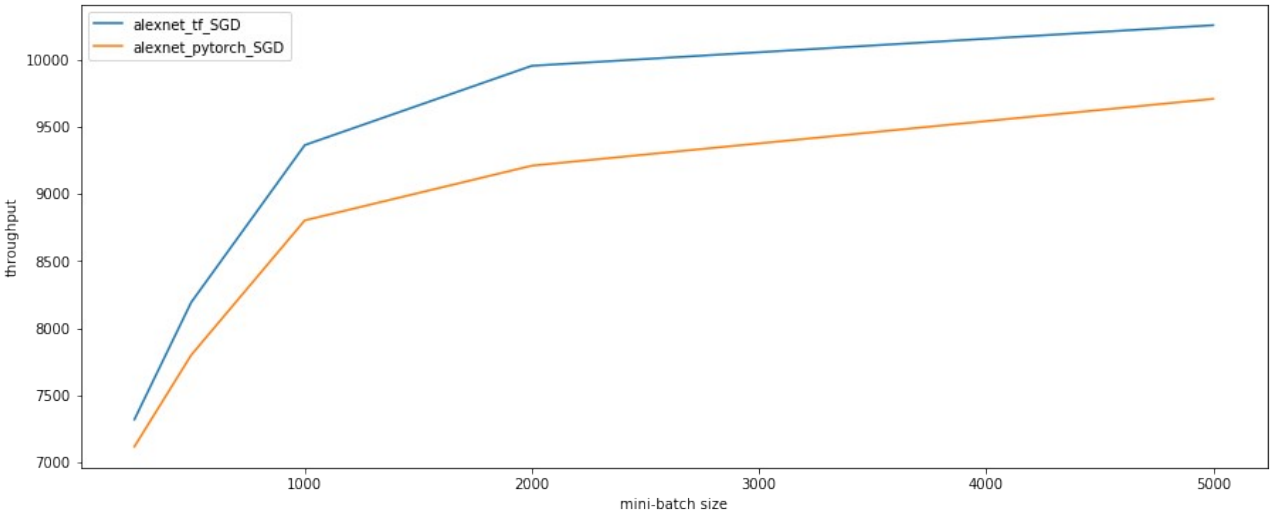

Рис. 1

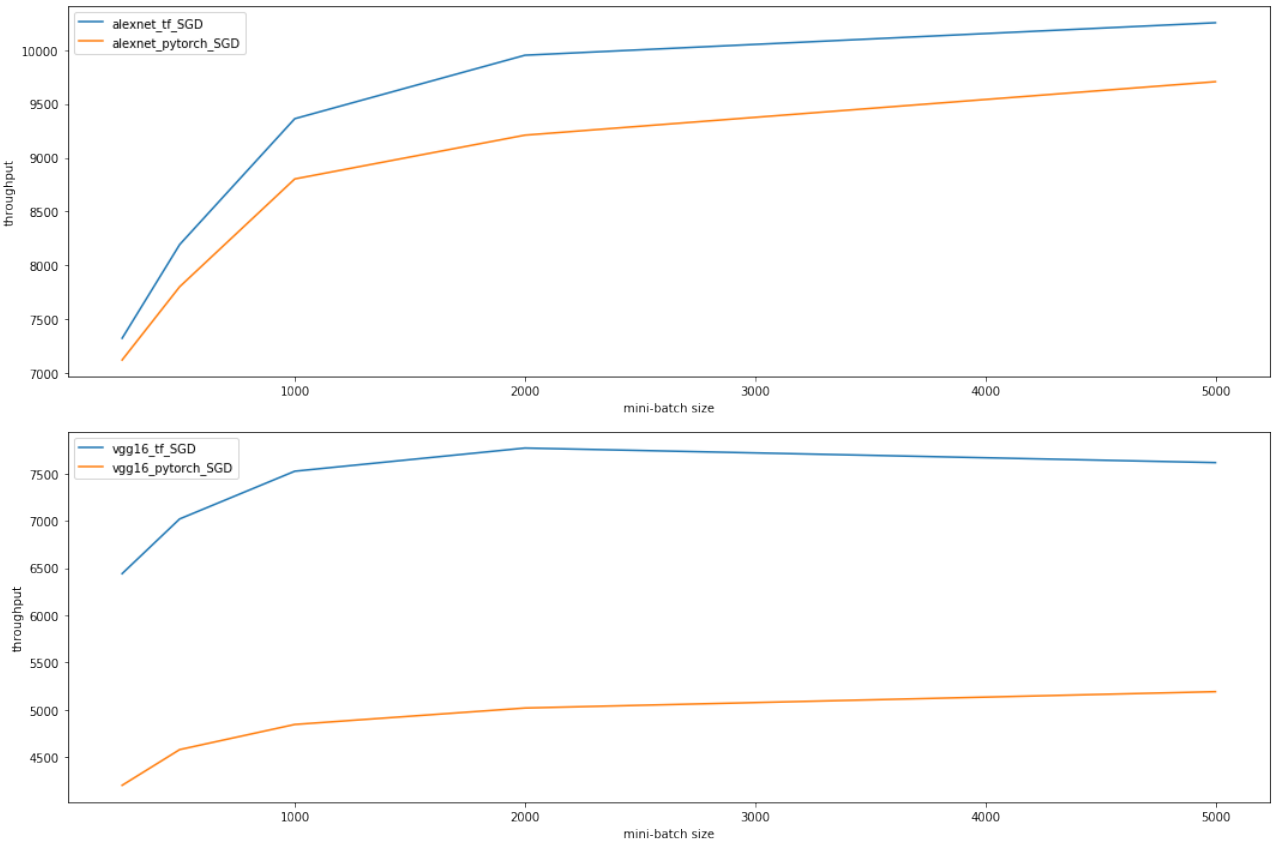

Рис. 2

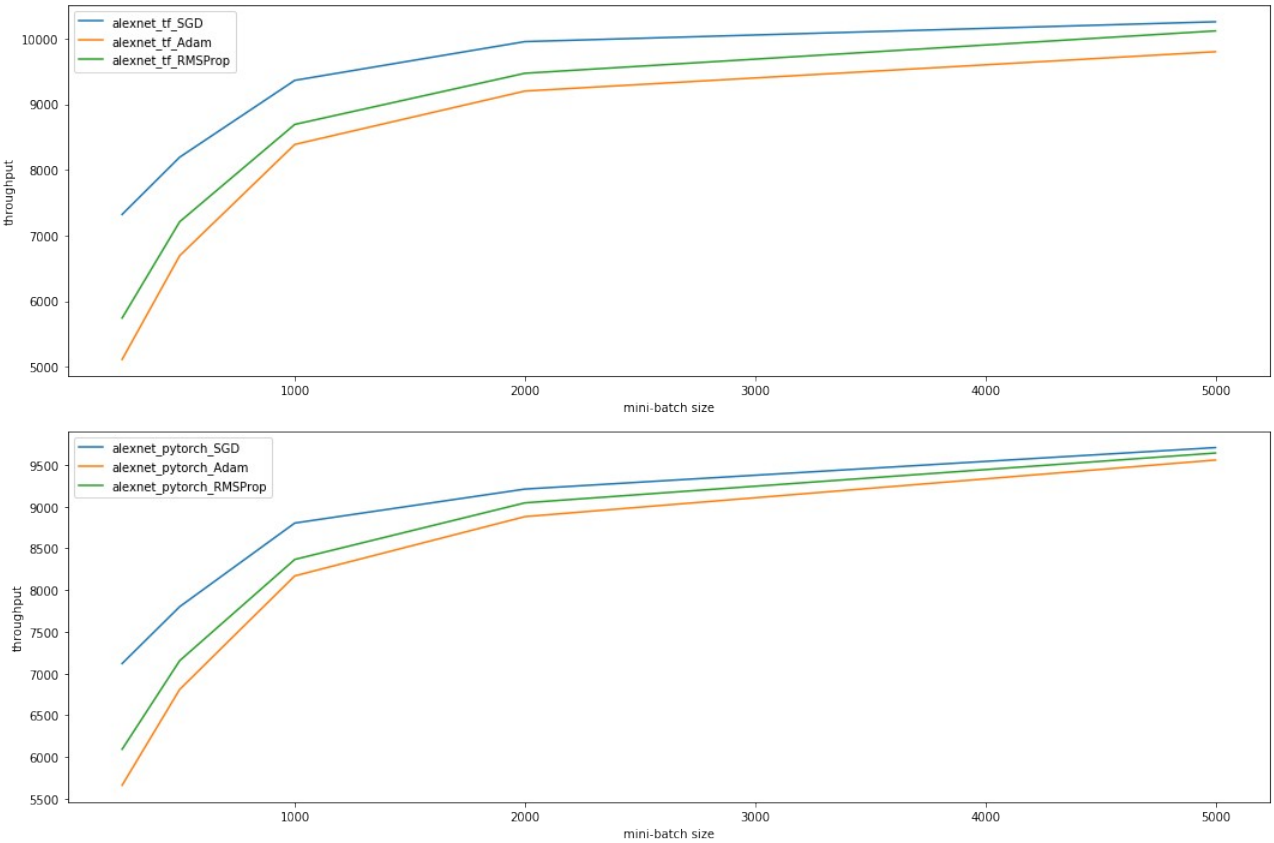
alexnet_tf_SGD
alexnet_tf_Adam
alexnet_tf_RMSProp
throughput
mini-batch size
alexnet_pytorch_SGD
alexnet_pytorch_Adam
alexnet_pytorch_RMSProp
throughput
mini-batch size

Рис. 3

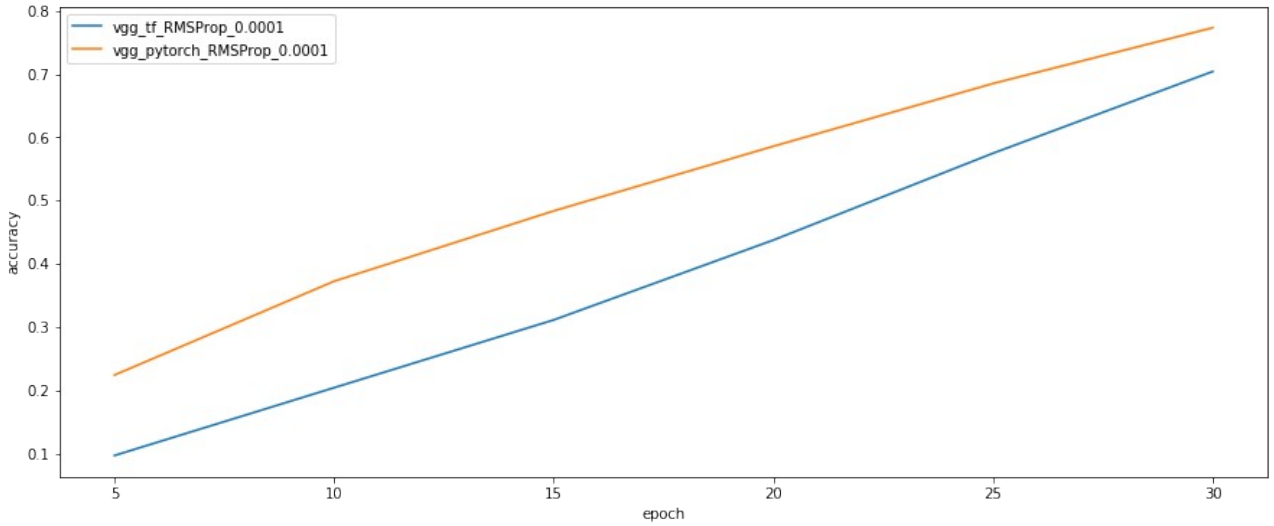
vgg_tf_RMSProp_0.0001
vgg_pytorch_RMSProp_0.0001
accuracy
epoch

Рис. 4

Описание программной реализации

Описание скрипта.

Были реализованы два модуля models.py и train.py. Models.py содержит определения нейронных сетей, train.py цикл обучения.

Вызов скрипта выполняется следующим образом: python3 train.py [options]

Номер параметра	Возможные значения	Описание
1	cpu, cuda	cpu — вычисления будут производиться на центральном процессоре, cuda — на GPU
2	tf, pytorch	Фреймворк, реализация на основе которого будете запущена
3	alexnet, vgg	Выбор модели
4	Целое число	Количество эпох обучения
5	Целое число	Размер mini-batch
6	sgd, adam, rmsprop	Выбор алгоритма оптимизации
7	Вещественное число	Значение параметра оптимизатора learning rate

Табл. 4

Описание реализованных нейронных сетей.

AlexNet

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param
================================================================
            Conv2d          [-1, 192, 15, 15]           5,376
              ReLU          [-1, 192, 15, 15]               0
         ZeroPad2d          [-1, 192, 17, 17]               0
         MaxPool2d            [-1, 192, 8, 8]               0
            Conv2d            [-1, 384, 6, 6]         663,936
              ReLU            [-1, 384, 6, 6]               0
         ZeroPad2d            [-1, 384, 8, 8]               0
         MaxPool2d            [-1, 384, 4, 4]               0
            Conv2d            [-1, 256, 2, 2]         884,992
              ReLU            [-1, 256, 2, 2]               0
         ZeroPad2d            [-1, 256, 4, 4]               0
            Conv2d            [-1, 256, 2, 2]         590,080
              ReLU            [-1, 256, 2, 2]               0
         ZeroPad2d            [-1, 256, 4, 4]               0
            Conv2d            [-1, 256, 2, 2]         590,080
              ReLU            [-1, 256, 2, 2]               0
         ZeroPad2d            [-1, 256, 4, 4]               0
         MaxPool2d            [-1, 256, 2, 2]               0
            Linear                 [-1, 8192]       8,396,800
              ReLU                 [-1, 8192]               0
            Linear                 [-1, 8192]      67,117,056
              ReLU                 [-1, 8192]               0
            Linear                   [-1, 10]          81,930
================================================================
Total params: 78,330,250
```

VGG16

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param
================================================================
            Conv2d           [-1, 64, 30, 30]           1,792
              ReLU           [-1, 64, 30, 30]               0
         ZeroPad2d           [-1, 64, 32, 32]               0
            Conv2d           [-1, 64, 30, 30]          36,928
              ReLU           [-1, 64, 30, 30]               0
         ZeroPad2d           [-1, 64, 32, 32]               0
         MaxPool2d           [-1, 64, 16, 16]               0
            Conv2d          [-1, 128, 14, 14]          73,856
              ReLU          [-1, 128, 14, 14]               0
         ZeroPad2d          [-1, 128, 16, 16]               0
            Conv2d          [-1, 128, 14, 14]         147,584
              ReLU          [-1, 128, 14, 14]               0
         ZeroPad2d          [-1, 128, 16, 16]               0
         MaxPool2d            [-1, 128, 8, 8]               0
            Conv2d            [-1, 256, 6, 6]         295,168
              ReLU            [-1, 256, 6, 6]               0
         ZeroPad2d            [-1, 256, 8, 8]               0
            Conv2d            [-1, 256, 6, 6]         590,080
              ReLU            [-1, 256, 6, 6]               0
         ZeroPad2d            [-1, 256, 8, 8]               0
            Conv2d            [-1, 256, 6, 6]         590,080
              ReLU            [-1, 256, 6, 6]               0
         ZeroPad2d            [-1, 256, 8, 8]               0
         MaxPool2d            [-1, 256, 4, 4]               0
            Conv2d            [-1, 512, 2, 2]       1,180,160
              ReLU            [-1, 512, 2, 2]               0
         ZeroPad2d            [-1, 512, 4, 4]               0
            Conv2d            [-1, 512, 2, 2]       2,359,808
              ReLU            [-1, 512, 2, 2]               0
         ZeroPad2d            [-1, 512, 4, 4]               0
            Conv2d            [-1, 512, 2, 2]       2,359,808
              ReLU            [-1, 512, 2, 2]               0
         ZeroPad2d            [-1, 512, 4, 4]               0
         MaxPool2d            [-1, 512, 2, 2]               0
            Linear                  [-1, 512]       1,049,088
              ReLU                  [-1, 512]               0
            Linear                  [-1, 100]          51,300
================================================================
Total params: 8,735,652
----------------------------------------------------------------
```

Заключение

В ходе выполнения курсовой работы было реализовано программное средство, позволяющее провести процесс обучения одной из двух моделей нейронных сетей, реализованных средствами фреймворков PyTorch и Tensorflow. Данное программное средство даёт возможность выбора значения таких гиперпараметров, как оптимизатор, learning rate и размер mini-batch. По мере выполнения скрипт выводит значения следующих метрик: время каждой эпохи, точность и ошибка на каждой эпохе, среднее время эпохи и средняя пропускная способность.

Была собрана таблица значений метрик средняя продолжительность эпохи и средняя пропускная способность модели в зависимости от модели нейронной сети, фреймворка средствами которого эта модель реализовывалась и оптимизатора, который использовался в процессе обучения. Был проведён анализ собранных данных.

В дальнейшем планируется сделать следующее. Расширить набор метрик, которые подсчитывает реализованная программа. Добавить возможность запуска программы не только с предопределенными моделями, но и моделями спроектированными пользователем. Расширить набор поддерживаемых фреймворков.

Список литературы

[1] «Benchmarking and Analyzing Deep Neural Network Training» http://www.cs.toronto.edu/~serailhydra/publications/tbd-iiswc18.pdf

[2] «A Modular Benchmarking Infrastructure for High-Performance and Reproducible Deep Learning» http://unixer.de/publications/img/deep500_ipdps19.pdf