Проблема аугментации при подготовке трехмерных медицинских датасетов

Е.П. Васильев [1]

[1] Нижегородский государственный университет им. Н.И. Лобачевского, пр. Гагарина, 23, Нижний Новгород, 603022, Россия

Аннотация

Задача трехмерной аугментации данных является вычислительно сложной и может требовать большое количество вычислений и сильно замедлять процесс обучения глубоких моделей. Уменьшить время обучения можно, оптимизировав применение аугментаций во время обучения. В статье рассматриваются различные инструменты для трехмерной аугментации данных: BatchGenerators, TorchIO, Rising. Приводится время работы распространенных трехмерных аугментаций: поворота, отражения, обрезки данных, добавления шума, размытия, применения аффинных преобразований и изменения размера трехмерных данных на центральном процессоре (CPU) и графическом ускорителе (GPU). Продемонстрировано преимущество выполнения аугментаций на GPU в сравнении с CPU, получаемое ускорение составляет 1-2 порядка для большинства аугментаций. Представлено сравнение времени различных аугментаций относительно каждого фреймворка для более наглядного выбора аугментаций с упором на их время выполнения.

Ключевые слова

Аугментация данных, компьютерная томография, глубокое обучение, бенчмаркинг.

The problem of Data Augmentation in the Preparing of Three-Dimensional Medical Datasets

E.P. Vasiliev [1]

[1] Lobachevsky State University of Nizhny Novgorod, Gagarina av. 23, N.Novgorod, 603022, Russian Federation

Abstract

The task of three-dimensional data augmentation is computingly coslty and may require a large number of calculations and slow down the process of teaching deep models. You can reduce learning time by optimizing the use of augmentations during training. The article discusses various tools for threedimensional data augmentation: BatchGenerators, Torchio, Rising. The working time of the common three-dimensional augmentations is given: rotation, flip, data crop, blur, application of affine transformations and changes in the size of three-dimensional data on the central processor (CPU) and graphic accelerator (GPU). The advantage of performing augmentation on the GPU in comparison with the CPU is demonstrated, the resulting acceleration is 1-2 orders for most augmentations. A comparison of the time of various augmentations relative to each framework is presented for a more visual choice of augmentations with an emphasis at their time.

Keywords

Data augmentation, computed tomography, deep learning, benchmarking.

ГрафиКон 2023: 33-я Международная конференция по компьютерной графике и машинному зрению, 19-21 сентября 2023 г., Институт проблем управления им. В.А. Трапезникова Российской академии наук, г. Москва, Россия
EMAIL: evgeny.vasiliev@itmm.unn.ru (Е.П.Васильев)
ORCID: 0000-0002-7949-1919 (Е.П.Васильев)

1. Введение

Методы машинного обучения, такие как нейронные сети, машины опорных векторов, случайные деревья требуют значительного количества примеров для обучения и настройки параметров. В задачах компьютерного зрения средством увеличения разнообразия в обучающем наборе данных является аугментация данных - метод искусственного увеличения размера обучающего набора данных с помощью применения различных преобразований к существующим данным. В настоящее время использование аугментаций является повсеместным, в качестве одной из первых работ, популяризировавшей данный подход, можно выделить [1], в которой для обучения модели AlexNet использовались случайное кадрирование и изменение яркости. В настоящее время список применяемых аугментаций значительно расширился, а популярные библиотеки глубокого обучения имеют встроенные методы для аугментации данных [2,3,4]. В то же время, при обеспечении заданного коэффициента увеличения объема датасета остается проблема выбора типов преобразований, которые внесут минимум искажений в природу целевого объекта. Время обучения при прочих равных также остается важным фактором. То есть интересно установить приоритеты по качеству и времени обучения для типовых методов в аугментации трехмерных медицинских данных.

С переходом от обработки двумерных данных к обработке трехмерных данных количество доступных размеченных наборов данных кардинально уменьшается, что требует значительно большего применения аугментаций для качественного обучения глубоких моделей. При этом применение трехмерных аугментаций является вычислительно затратной операцией.

В фреймворках глубокого обучения нет встроенных алгоритмов трехмерных аугментаций, но научным сообществом разработаны несколько библиотек, позволяющих использовать трехмерные аугментации.

BatchGenerators [5]. Открытая библиотека, реализованная на языке Python. Включает в себя пространственные преобразования (масштабирование, поворот, перенос, передискретизация в 3D), поканальные (цветовые) преобразования, добавление шума, кадрирование.

TorchIO [6]. Открытая библиотека, также поддерживающая пространственные и поканальные преобразования. Также в библиотеку включены преобразования, имитирующие артефакты магнитно-резонансной томографии: размытие движения, искажения поля.

Rising [7]. Открытая библиотека, поддерживающая различные трехмерные преобразования. Преимуществом библиотеки является возможность использования графических ускорителей Nvidia для выполнения трехмерных преобразований, что теоретически должно положительно сказаться на времени обучения модели.

В данном исследовании будет рассмотрено применение каждой из библиотек трехмерной аугментации данных.

2. Параметры эксперимента

В качестве вычислительного устройства используется следующая вычислительная система: CPU AMD Ryzen 5600X (6 cores; 12 threads; 3,7GHz), GPU Nvidia RTX 3090Ti; 24GB GDDR6X; RAM 64Gb DDR4 - 3200MHz.

Для экспериментов использовано подмножество открытого набора данных Multi-Modality Whole Heart Segmentation [A4]. Данный набор данных содержит в себе 20 компьютерных томограмм (КТ) сердца с контрастированием и 20 магнитно-резонансных томограмм (МРТ) сердца.

3. Алгоритмы аугментации

В таблице 1 показаны главные характеристики библиотек аугментации трехмерных данных, используемые в обучении глубоких моделей. Мы видим, что существующие фреймворки являются сильно отличающимися между собой в плане их базовой математической библиотеки (бэкенда), с связи с чем ожидается очень сильное различие производительности между ними.

Таблица 1 – Сравнение фреймворков аугментации трехмерных данных

	Albumentations	Rising	BatchGenerators	TorchIO
Бэкенд	Numpy	Pytorch	Numpy	PyTorch / SITK
Поддержка GPU	-	+	-	-
3D аугментации	-	+	+	+

Для проведения сравнения фреймворков были выбраны 6 наиболее часто используемых алгоритмов аугментации, на которых будет производиться сравнение. Список анализируемых аугментаций ниже (рисунок 1):

- Поворот (Rotation). Одна из самых распространенных и вычислительно затратных операций в трехмерных данных. Двумерные реализации выполняются быстро и на CPU, и на GPU, но для трехмерных вариантов выполнение на CPU является не эффективным, так как весь массив данных как правило не помещается в кэш процессора, и приходится выполнять множество операций чтения из оперативной памяти.
- Добавление шума (Noise). Очень часто используемая аугментация, потому что медицинские данные часто бывают зашумлены устройством их получения, и обучаемая модель должна быть к нему устойчива. Данная операция одинаково эффективно выполняется и для 2D и для 3D.
- Изменение размера (Resize). Данная операция выполняется крайне долго при выполнении средствами стандартных библиотек. Однако, если данную операцию перенести на графический ускоритель, то она будет выполняться значительно быстрее за счет аппаратной трилинейной интерполяции и архитектуры доступа к данным.
- Обрезка (Crop). Операция обрезки данных не требует множественных операций доступа к памяти и быстро выполняется и в 2D, и в 3D.
- Размытие (Smoothing). Производительность операции размытия на CPU значительно зависит от того, насколько оптимизирован ее исходный код. На GPU как правило выполняется в виде трехмерной свертки, которая очень хорошо оптимизирована в каждом фреймворке с поддержкой GPU.
- Зеркальное отражение (Flip). Операция обрезки данных не требует множественных операций доступа к памяти и быстро выполняется и в 2D, и в 3D.

Помимо стандартных алгоритмов аугментации, некоторые библиотеки реализуют нестандартные алгоритмы аугментации, используемые в специализированных областях. Среди таких алгоритмов можно выделить алгоритм RandomGhosting из библиотеки TorchIO [A9], реализующий добавление случайных фантомных ореолов - артефактов, возникающих вдоль направления фазового кодирования во время МРТ сканирования, или алгоритм RandomBiasField, имитирующий неоднородность магнитного поля МРТ через вариации интенсивности очень низкой частоты по всему изображению (рисунок 2).

Отдельного упоминания заслуживают условия применения трансформаций к пачке данных. Например, API библиотеки Rising позволяет применять трансформации как на CPU, так и на GPU, применять их, для каждого трехмерного изображения в пачке, с одинаковыми или с разными параметрами. Также имеется возможность применять аугментацию с заданной вероятностью, а также случайно применять только одну трансформацию из предложенного списка. Если GPU или CPU являются узким местом в процессе обучения, то можно изменить устройство выполнения аугментации для балансировки нагрузки между ними. Все эти возможности позволяют максимизировать разнообразие данных для обучения.

Исходный код, демонстрирующий работу с указанными библиотеками, результаты применения аугментаций и подсчет времени выполнения аугментаций, доступен в открытом репозитории на GitHub: https://github.com/FenixFly/3d_augmentations_execute_time.

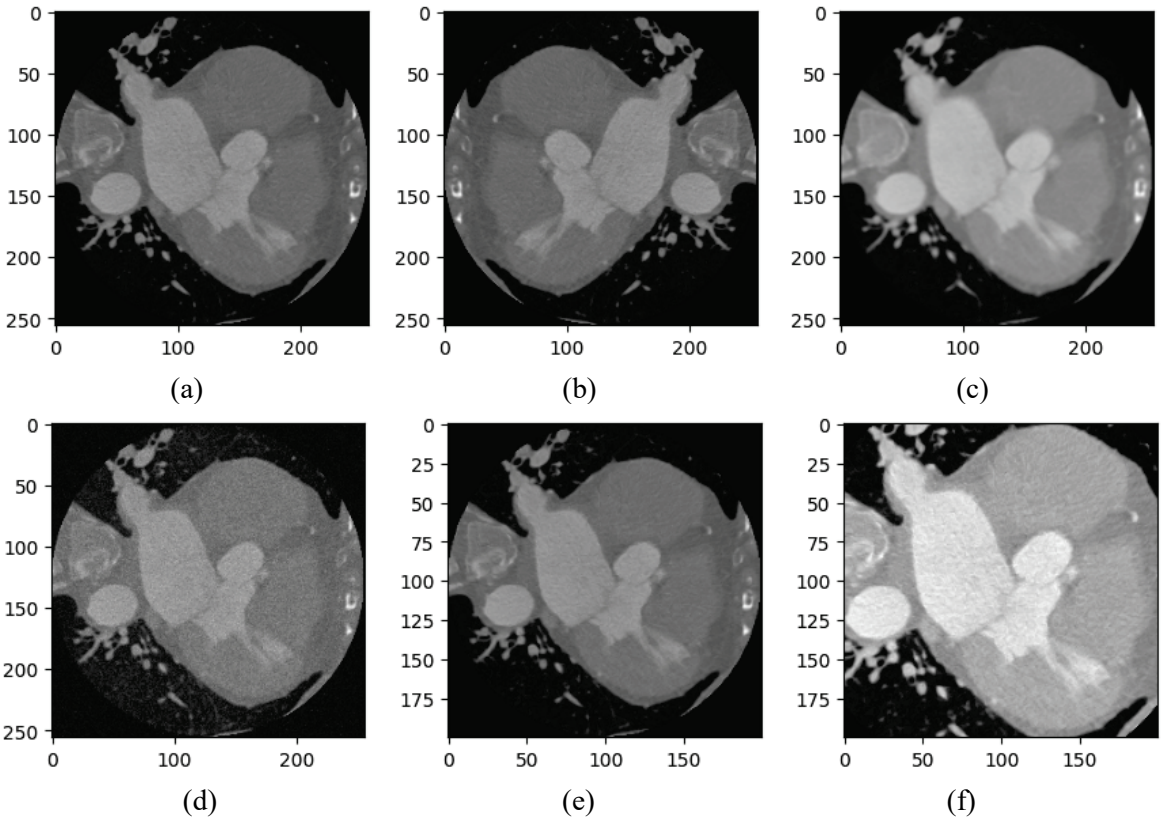

Рисунок 1 – Примеры рассматриваемых алгоритмов аугментации: (a) слой оригинального трехмерного изображения; (b) зеркальное отражение; (c) размытие изображения; (d) добавление шума; (e) изменение размера; (f) обрезка изображения

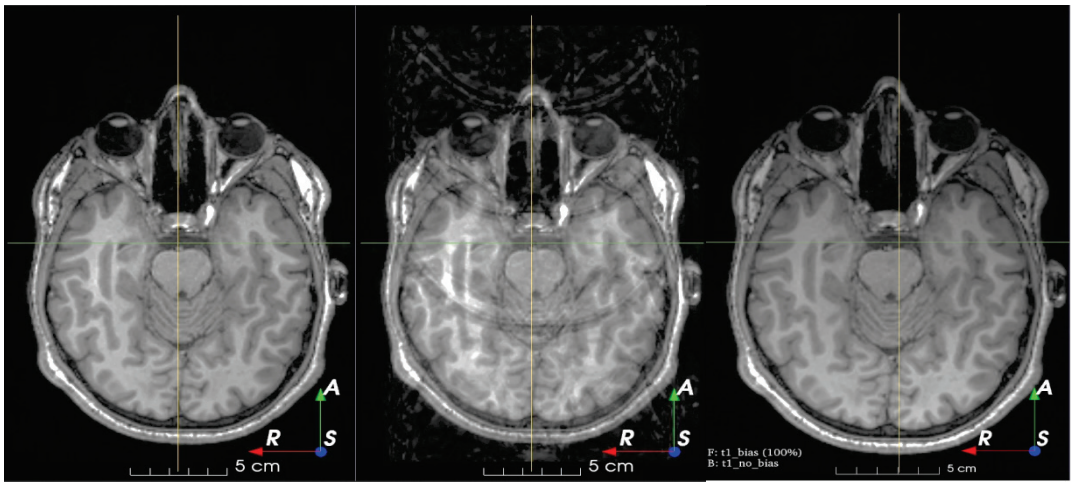

Рисунок 2 – Примеры нестандартных алгоритмов аугментации фреймворка TorchIO: (a) нормальное изображение; (b) аугментация RandomGhosting; (c) аугментация RandomBiasField [8]

4. Сравнение производительности

В таблице 2 приведены времена выполнения распространенных операций 3D аугментаций с использованием различных библиотек. Время выполнения аугментаций вычислялось для трехмерного изображения 256*256*128. Все операции циклически повторялись, количество повторений было выбрано таким образом, чтобы выполнение занимало порядка минуты.

Таблица 2 – Время выполнения операций 3D аугментации

Тип операции	Время выполнения 1 итерации, миллисекунд			
	Numpy/Scipy	Rising (CPU)	Rising (GPU)	TorchIO
Noise	166.8	174.34	0.988	72.58
Smoothing	136.6	25.92	0.225	190.9
Crop (200,200,100)	0.000371	0.0868	0.0974	5.151
Flip (axis 2)	0.002002	0.0106	0.1493	15.43
Аффинные преобразования (scale, rotate)	-	60.55	1.687	155.96
Resize 3D (200,200,100)	1049.8	16.58	0.4030	104.74

Сравнивая скорость выполнения аналогичных аугментаций в разных фреймворках, можно увидеть, что использование GPU дает значительное преимущество по скорости при выполнении аугментаций. При этом, некоторые аугментации, которые можно реализовать средствами библиотеки Numpy, на CPU можно выполнить быстрее.

Также большой интерес вызывает сравнение времени выполнения каждой операции в рамках каждого из фреймворков отдельно. Данная информация дает нам понять, насколько выгодно часто применять ту или иную аугментацию. Например, чтобы понизить вероятность применения более медленной аугментации с целью ускорения процесса обучения.

Средствами самых распространенных библиотек Numpy / Scipy аугментации, требующие множественных математических операций, работают катастрофически медленнее операций переупорядочивания массивов (Crop, Flip), поэтому предпочтительнее их использовать чаще. Данное утверждение относится только к трехмерным аугментациям, двумерные аугментации, намного более оптимизированы и не имеют такого разрыва в производительности.

Реализация аугментаций в фреймворке Rising основана на быстром и оптимизированном ядре библиотеки Pytorch, что дает высокую скорость выполнения аугментаций трехмерных данных. В CPU варианте между различными аугментациями существует значительной разброс по времени выполнения, и имеет смысл реже использовать зашумление данных. В GPU варианте происходит значительное сокращение разницы времени работы, поэтому можно использовать различные аугментации равновероятно, тем более что и само время выполнения аугментации очень маленькое. Замедление времени аугментации в CPU варианте представлено в таблице 3.

Таблица 3 – Относительные времена выполнения аугментаций

	Замедление аугментации относительно самой быстрой, раз					
Фреймворк	Crop	Flip	Smoothing	Noise	Resize	Affine
Numpy / Scipy	1.0	5.39	368194	449595	2829649	-
Rising (CPU)	1.0	0.12	298.6	2008.5	191.0	697.6
Rising (GPU)	1.0	1.53	2.31	10.1	4.1	17.3
TorchIO	1.0	2.99	37.1	14.1	20.3	30.3

5. Заключение

В данной работе рассмотрены алгоритмы и фреймворки для аугментации трехмерных данных, которые позволяют производить различные варианты преобразования медицинских данных для тренировки моделей глубокого обучения. Рассмотрены различные библиотеки для создания трехмерных аугментаций: Numpy, Scipy, TorchIO, Rising. В данной работе приведено сравнение не всех библиотек, поддерживающих трехмерные аугментации, в дальнейшем список сравниваемых приложений будет расширен.

Дальнейшим направлением данной работы является тестирование данных алгоритмов аугментации в задаче тренировки моделей глубокого обучения для сегментации трехмерных медицинских данных с целью сравнения библиотек аугментации и их влияния на качество обучения, на время обучения, на характеристики процесса сходимости во время обучения.

6. Благодарности

Работа выполнена при поддержке Проекта № 0729-2021-013, выполняемого в рамках Государственного задания на выполнение научно-исследовательских работ лабораториями, прошедших конкурсный отбор в рамках национального проекта «Наука и университеты», в отношении которых принято решение Бюджетной комиссии Минобрнауки России (от 14.09.2021 № БК-П/23) о предоставлении из федерального бюджета субсидии на финансовое обеспечение государственного задания на выполнение научно-исследовательских работ.

7. Список источников

[1] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks // Neural Information Processing Systems. 25. 10.1145/3065386.

[2] Pytorch. Transforming and augmenting images [Электронный ресурс]. URL: https://pytorch.org/vision/stable/transforms.html (дата обращения 14.07.2023).

[3] Keras. API for preprocessing images [электронный ресурс]. URL: https://keras.io/api/preprocessing/image/ (дата обращения 14.07.2023).

[4] Designing Efficient Data Loaders for Deep Learning [электронный ресурс]. URL: https://mxnet.apache.org/versions/master/api/architecture/note_data_loading (дата обращения 14.07.2023).

[5] Isensee Fabian et al. (2020). batchgenerators - a python framework for data augmentation. doi:10.5281/zenodo.3632567.

[6] F. Pérez-García, R. Sparks, and S. Ourselin. TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. Computer Methods and Programs in Biomedicine (June 2021), p. 106236. ISSN: 0169-2607.doi:10.1016/j.cmpb.2021.106236.

[7] Rising [электронный ресурс]. URL: https://github.com/PhoenixDL/rising (дата обращения: 14.07.2023).

[8] Augmentation in TorchIO [электронный ресурс]. URL: https://torchio.readthedocs.io/transforms/augmentation.html (дата обращения 14.07.2023).