infinitybatch

Axel Ország-Krisz Dr.

Richárd Ádám Vécsey Dr.

Table of Contents

Summary....8

What is infinitybatch?....9

Background....10

The problem and need....10

Big batch vs. bigger batch....14

Science of deep learning and papers....15

Optimization techniques....18

Number of layers (model.structure)....21

Type of layers (model.structure)....21

Number of nodes (model.structure)....21

Bias (model.structure)....21

Activation functions (model.structure)....21

Batch normalization (model.structure)....21

Parameters (model.train)....21

Loss function (model.train)....21

Optimizer (model.train)....22

Dropout (model.train)....22

Manual learning rate (model.train)....22

Learning Rate Schedule (model.train)....22

Momentum (model.train)....22

Weight Regularization (model.train)....22

Epoch number (model.length)....22

Early Stopping (model.length)....23
Nodes' initialization (model.init)....23
Bias' initialization (model.init)....23
Pretrained network with retrain (model.init)....23
Pretrained network without retrain (model.init)....23
Retrain network: Transfer learning (model.init)....23
Retrain network: Continue from checkpoint (model.init)....24
Constant batch size (batch.batch_size)....24
Various batch sizes (batch.batch_size)....24
Constant batch (batch.batch_selection)....24
Managed batch selection (batch.batch_selection)....24
Randomized batch selection (batch.batch_selection)....24
Subsample majority class (data.dataset_balancing)....24
Oversample minority class (data.dataset_balancing)....24
Collecting new data (data.size_of_dataset)....25
Generating new data (data.size_of_dataset)....25
Data augmentation (data.data_selection)....25
Adding noise (data.data_selection)....25
Feature selection (other)....25
Validation dataset selection by real life cases (other)....25
Market....26
Survey....27
Short description about survey....29
Questions....30

Method....35
Question 1....35
Question 2....35
Question 3....35
Question 4....36
Question 5....36
Question 6....36
Question 7....36
Question 8....36
Question 9....36
Question 10....37
Question 11....37
Question 12....37
Question 13....37
Question 14....37
Question 15....37
Result....39
Infinitybatch....52
License....52
Formulas....52
Limitation....53
Perfect batch size based on CUDA cores....53
Structure....54
Dataflow....55

Intended use....56
Competitors....56
Benefits for PyTorch....56
Integration into PyTorch....56
UX....58
The gauntlet....59
Slow....59
Cloud 9....59
Big batch blah....60
Unprovable....60
A need that cause mislead....60
Import distort....61
Unintended extended....61
Countably infinite....61
Cheap, sweep, weep....61
Code reference....62
infinitybatch....62
class InfinityBatch....62
class InfinityBatchError()....69
class InfinityBatchWarning()....70
tools....70
classes....70
class CudaMemorySnapShot....70

class CudaHistory....71
functions....73
determinize()....73
is_equal_network()....73
is_equal_with_error()....74
compare_with_error_()....75
constants....75
ERROR_TOLERANCE....75
Statistics....76
Video card popularity....76
VRAM on Steam....77
System RAM on Steam....78
PyTorch on Github....79
Open source licenses on Github....80
Tesla index....82
Unique and special solutions in infinitybatch....84
Future plans....85
eval()....85
PyPi....85
Event monitor....85
Improvement of warnings....85
Improvement of the containers....86
Counter of learnable parameters....86

Memory usage monitor....86
Improving memory usage....86
Improving UI....87
Change of model....87
“In case if needed” plans....87
Callbacks....87
Improvement of tools.CudaHistory....88
Complexity....88
Sources....89
Papers....89
Links....92
About us....94
cheXrad....95
DOF – Deep Model Core Output Framework....98
MGP – Medical Gateway Platform....102
PyTessy....106
COVID-19 dataset....107
COVID-19 map....107
COVID19CasesWorldwideDatasetFromECDC....109

Summary

This book is about infinitybatch and the whole related project. First of all, there are some important notes that can be useful during reading.

This document contains different colors to highlight some type of content such as headings, hyperlinks and cross-links.

We use “GPU RAM” instead of VRAM since VRAM can be placed in system RAM. However, we use “CPU RAM” instead of system RAM or RAM if it seems to necessary.

However, in the scientific discourse the batch and mini-batch are separated, the official documentation of PyTorch speaks about only “batch”. The description of Batch class contains the following part: “Defines a batch of examples along with its Fields”. That’s why we decided to use “batch” instead of “mini-batch”. In this term batch can mean the situation when the gradient is computed to the whole training dataset in one iteration and the situate when only a subset of training dataset is the basis of computation.

Two fonts are used in this book with different meaning. Roboto is used generally, `IBM Plex Mono` is used for code samples and formulas.

The book has two major parts. Background of infinitybatch is the first one, that covers the description of solved problem, the scientific explanations, and some other interesting stuff such as our survey and other statistics. It can be skipped, it is not necessary to understand our project. It just draws a frame to the concept of infinitybatch. The second part is called: Infinitybatch. It contains explanations, use cases, source code, etc. You should jump there if you want to know only the dry facts.

In the chapter called Science of deep learning and papers we collect the most important statements. The sources and proper citations are in the end of this book. Those are easily available by clicking the [Papers] link at the end of every paragraph.

What is infinitybatch?

Infinitybatch is an open source solution that helps deep learning developers to train with bigger batch size than it could be loaded into GPU RAM.

The core concept of the idea comes from the fact that GPU time is expensive and the usage of own GPU cluster or a cloud based GPU service has to be optimized to be cost efficient. Furthermore, developers and researchers regularly has limited access to GPU. However, CPU based training mostly allows higher batches than a normal GPU could provide, it is much slower. Infinitybatch helps to use GPU during training with bigger batch size thanks to the special unloading and uploading process that manages the GPU RAM to avoid memory overrun.

The name of infinitybatch is a little deceptive for the first catch. It does not mean we could really use batch size that is infinite. The highest limit of batch size always will be the length of training dataset. Secondly, infinite batch cannot exist since the resources are always limited, such as GPU RAM, CPU RAM, GPU speed, CPU speed, storage, FSB bandwidth or the bandwidth of a network.

We only provide to use bigger batch size than GPU could be provide during a normal training process. That's why inifinitybatch does not necessary mean big batch. For example, when a special training process on a common GPU can be trained with batch size 4, the developer could be train with batch size 32 with infinitybatch. However, if the developer want use really big batch, it can be done with our solution.

We does not want to compete with GPU cloud services, but we can help the users to use services smart and well prepared. Infinitybatch promises only to expand the limits. This is a great and yet simple idea that helps researchers and developers to test a lot of ideas with different hyperparameters on machine with limited resources and train the finalized idea on a dedicated machine or cloud service. Also it can be a good solution for students, who have mid range GPU hardware, to try different batch sizes and learn and understand the concept of deep learning model development and the behavior of training process.

Background

The problem and need

Nowadays computers at home or at companies usually contain GPU. However, these GPUs have very poor computational resources compared to dedicated deep learning GPUs like Tesla or Quadro series. Despite of limitation, the home or company GPU, it can be used for experimenting, testing new theories, trying new approaches and learning. Using cloud based GPU services can be expensive especially when the whole concept of model is made on the cloud. A properly trained model is based on a lot of dead ends. A huge amount of model architectures are in the trash after the training phase. A developer or researcher has to try different batch sizes with different batch-creation technology and tune other hyperparameters as well. If any of these steps can be used on own GPU, it saves a lot of money and internet bandwidth compered with a cloud based service. Facebook supports deep learning courses by collaborate with Udacity such as Intro to Deep Learning with PyTorch or Secure and Private AI. Students comes from all over the world with different financial background.

Seeing this problem closer paints the whole picture really dark. The income inequality creates a great gap between different programmers. Some of us work with 2 or 4 monitors simultaneously or use a huge LED TV as monitor. Some of us has to use old CRT monitor that we dropped out 15 years ago. Installing PyTorch can be finished within 1 minute or 7 hours. The availability of electricity to us is 24/7, but there are regions where the power cuts are part of a regular day. Infinitybatch cannot solve these problems, but we have to remember, there are programmers colleagues in very bad life situation. We could help to programmers who has a some resource, a GPU at least.

During the analysis of problems and needs, we knew this is a hot topic, but we wanted to measure how painful the problem is. That's why we started on two different paths. The first idea was a survey, that is able to show what other programmers and developers think about problem. The second way was slightly different that popped up when we did a

research about why people have very different, but usually low end or mid range GPUs. The income inequality is a real problem that cannot be measured properly. There is a very famous Big Mac Index, but it focuses on the purchasing power parity and national market and goods, but programmers are living in a globalized world where every items can be purchased from the global market. That's why we create a new index called: Tesla index.

At this point we have identified a lot of pain points, but there are more. Choosing the perfect computational architecture is a profession that requires a lot of knowledge and experience. Programmers or researchers do not have the necessary skills to solve this problem. We usually try to build a universal machine with a GPU that we can afford. Small companies struggle with these symptoms as well. With infinitybatch they can estimate the necessary resources to make a right decision about infrastructure expansion.

As we have seen, problem can be physical, like income and lack of resources or knowledge-based, like lack of experience. The third part of problems comes from the deep learning method. Theoretically there are 3 possible ways to define the batch size. The easiest way when the gradient is computed to the whole training dataset in one iteration. The other side of scale when every element is forwarded through the gradient calculation alone. The third when the batch size is somewhere between 1 and the length of training dataset. When a developer or researcher try to choose an optimal batch, they will face with some limitation:

- Using CPU usually provides bigger batch size since RAM are cheaper than video cards with high GPU-RAM. However, CPU-based trains are usually slower than GPU-based trains.
- The model and the batch size have to fit into GPU memory during GPU training.
- There is no a magical batch size that can be used very well in different tasks.
- There is no magical batch size, but there are common values that are worth trying.

Problems above are identified like this process was the part of a research as we were social scientist at a university. What could say a programmer, how could they grab the

problem of deep learning training. What is the cause from the perspective of forum topics and google searches? We know, there are a lot of questions, bug and error reports. When we want to solve one of them, we should dig deeper about the cause of the error and try to figure it out why those errors come out. According to Python's official documentation, there are two different types of error may occur: syntax error and exception that can be interpreted as logical error. These are similar to Java language according to David Matuszek from University of Pennsylvania. Seeing the common questions and problems from field of deep learning especially PyTorch, there is an other way to group error:

- Syntactical
- Structural
- Architectural
- Dataset-based
- Hyperparameters

Syntactical error means the code or the part of it is incorrect that prevents successfully compiling and running. Usually this type of error comes from lack of programming skills and knowledge. It can be avoided if anybody knows the programming framework and programming language syntax properly.

Structural error occurs when the error is in the deep learning part of the code. This error comes from lack of knowledge about a used module or library such as PyTorch. For example, the backpropagation is missing from train loop. The Python compiler can compile and run the code, but it will raise an error during running, or learning for the model will be impossible.

Architectural error means the model's architecture does not fit for the problem. Maybe it contains too little or too much nodes, wrong activations or bad layers. The code can run, but the model learns nothing or the result will be far away from the optimal solution. This error can be avoided with practicing and getting deep knowledge about deep learning architectures and problems.

Dataset-based error occurs when the developer or programmer don't understand the problem and dataset. That's why the dataset may contain unnecessary biases, unwanted columns, preventable noise. This error can be avoided for example with exploratory data analysis (EDA).

Error in hyperparameter causes too long training or bad result after training process. This is a really hard problem since it can be avoided with a lot of practice and learning. Reading papers and finishing good online or offline courses can give solid basics, but there are an other factor: resources. Knowledge is only the first part of a good and effective approach for finetuning hyperparameters. The real limitations are the available resources such as RAM, number of CUDA cores, etc. Our solution tries to provide a little extra resources for this kind of tasks. We cannot make the bed bigger, but we can sew and make the sheet a little longer.

Big batch vs. bigger batch

First of all, we have to separate to different things: big batch and bigger batch. When we talk about Infinitybatch, we always think about the 'bigger' adjective. There are no consensus in science of deep learning about what is the limit that separates the batch size into small or big size. Since we expand the train limits, developers could use bigger batch size compered to the PyTorch's normal train loop. We do not want to tell what is the ideal batch size, since tuning hyperparameters is one of the key moment of deep learning developing process. If somebody has a video card that can handle a batch that contains 8 elements, they will be able to train with for example 16, 32. These batch sizes are not so big. However, if somebody owns a Tesla, they can reproduce famous researches easily and set the batch size to 512, 1024, 2048 or 4096. The choice is free within the limitation. In a very extreme example to use 2GB GPU-RAM and set the batch size somewhere around 10k. It can be possible with Infinitybatch, but it is not cost- and time effective, so it is not be advised.

That's why we speak about bigger batch size instead of big batch. Still, we know there is a big discourse between developers and researches about big batch and it's effect on train process. In this context, 'bigger batch' can mean 'big batch' as well. That recognition leads us to write about the scientific background of big batch training on the next chapter. We would like to show various intended uses, scientific discourses to prove that Infinitybatch is a good and useful tool for developers, programmers and researches.

Science of deep learning and papers

In this section we show the most quoted or latest papers that are connected to out project. This link can be weak or a really strong. Our goal is to take a snapshot about this field to prove that there is place in the world for our solution. There are a lot of research relied on or connected to batch size where infinitybatch can help to prove or to refute previous statements.

The biggest debate about batch size is about generalization. The answer comes from Intel. Using SGD or any variants method, large batch tends to converge to sharp minimizers. This sharping effect affects both training and testing functions. This causes poorer generalization. However, they defined small batch as the batch contains 32 – 512 data points, there is no consensus about what is the limit between small or large batches. The batch between this range helps flat minimizers and better generalization, due to the noise. [Paper 01]

Ian Goodfellow has other important statement about speed of converge. Using large batch consumes more computation than small batch. The author gave a very good example: let us suppose, we use two gradients, the first includes 100 elements and the second includes 10,000 samples. To the large batch needs 100 times more computation than the small one. However, the large batch computation reduces the standard error of mean only by a factor of 10. When we identify the speed as the number of total computation instead of total computational time or the number of updates, most optimization algorithms converge faster by allowing to compute approximate estimates of the gradient. [Paper 02]

One of the most strict statement is the batch size should be between 2 and 32, since it provides the best performance. Experiment confirms that is said by Ian Goodfellow, small batch size causes more computation, that leads to more stable and better training. Batch size also connected to learning rate according to this paper. Increasing the batch size reduces the range of optimal learning rate. [Paper 03]

There are papers from the other side of debate and they reveal other conclusions. As it was mentioned above, there is a connection between learning rate and batch size. There is

a good technique for reducing training time to increase the batch size instead of learning rate decay. [Paper 04]

Somebody may think, using large batch is like playing with fire, Facebook gave a very strong argument to the camp of large batch fanatics. They used distributed synchronous SGD to divide SGD mini batches over a pool of parallel workers. There was no loss of accuracy during training on ImageNet with large batch size up to 8,192 images. [Paper 05]

The generalization gap during big batch training can be reduced or eliminated by increasing the learning rate. However, Kevin Shen observed a disturbing phenomenon: when models are initialed with the same average Euclidean norm distance, larger batch size have larger variance compared to smaller batch size, epoch by epoch. [Paper 06]

Many groups are satisfied with mathematical deductions, others are not. Researchers in the last category created theoretical and empirical evidence to prove their statement. 1,600 models was trained to prove the negative correlation between the generalization ability of networks with the ratio of batch size to learning rate. [Paper 07]

Sometimes new algorithmic solution is born to solve a problem in different way. Such technique is the Ghost Batch Normalization that tries to answer to the problem of generalization gap. [Paper 08]

One of the most mind-blowing ideas is a 32K batch size, that means 32,768 batch size with ImageNet dataset and ResNet50 model. However, the concept is brilliant, they ended the task successful. The trick is the Layer-wise Adaptive Rate Scaling method, that use different learning rate for different layers. Using larger batch causes significant speedup over small batch training. Infinitybatch can help to reproduce this experiment on a mid-range GPU at home computer. [Paper 09]

There is other researchers on the same path as the previous group. 16K batch size can function quite well. It is good to know, there are researchers who try to experiment not only with large batches, but different hyperparameters, own framework or Hessian based method. There are new way to achieve significantly reduced train time. [Paper 10]

Most researchers agree with the statement, that flat minima leads to good generalization in stochastic gradient based methods and sharp minima leads poorer generalization. This is one of the base concepts of small batch training. However, there are experiments, that shows sharp minima can cause acceptable generalization as well. Of course, it has to exist specific terms. [Paper 11]

Since the world is not black or white, there are more options than small or large batch. Nvidia, the giant GPU supplier, made a research with adaptive batch size. With adaptive batch can get better train time compared to fix batch size. Infinitybatch can help researchers to use adaptive batch as well. In the paper there is written about the benefits of large batch. It helps to use better the modern architectures like GPU memory bandwidth. [Paper 12]

Researchers and developers usually speak about ideal batch. However, there is a method to esteem the ideal batch size. It is an interesting fact, that their arguments against large batch is that it may not fit into GPU-RAM. Well, this situation is handled, infinitybatch eliminates this kind of problem. There are no thumb rule, but they say, the square root of the size of the dataset is a good starting point. [Paper 13]

It is worth to mention that there is a method to research to batch problem through the glass of economic feasibility. The parameters depends on two factors: compute-cost and training time. Former is represented as the budget, latter is represented as the deadline. There is a very unique idea to pull theories from economic into the science of deep learning. Pareto efficiency is named after Italian economist Vilfredo Pareto and it says a situation is called Pareto optimal if no change could lead to improved or higher satisfaction for all parties at the same time. This kind of thinking opens new ways in model development. Infinitybatch can help to get closer or to reach a Pareto optimal situation. [Paper 14]

Batch normalization is a very popular method that can help to reduce training time according to Google's research. [Paper 15]

As you can see, there are a lot of researches, scientific publications or articles that are related to batch size. Cited sources above are just the top of a huge iceberg.

Optimization techniques

In this chapter we show the most popular optimization techniques in a structured format. This method cannot be found in any paper in this form. We created and it represents the way we think about theory of model performance optimization. However, it really rhymes to papers, since there are no new things under the Sun. Infinitybatch-related areas are marked.

I. **Model**

1. Structure
   - Number of layers
   - Type of layers
   - Number of nodes
   - Bias
   - Activation functions
   - Batch normalization
2. Train
   - Parameters
   - Loss function
   - Optimizer
   - Dropout
   - Learning rate
     - ↳ Manual
     - ↳ Learning rate schedule
   - Momentum
   - Weight regularization

3. Length
    - Epoch number
    - Early stopping
4. Init
    - Nodes' initialization
    - Bias' initialization
    - Pretrained network
        - ↳ with retrain
        - ↳ without retrain
    - Retrain network
        - ↳ Transfer learning
        - ↳ Continue from an early stopped checkpoint

II. **Batch**

1. Batch size
    - Constant (manual)
    - Various
2. Batch selection
    - Constant batch
    - Managed batch selection
    - Randomized batch selection

III. **Data**

1. Dataset balancing
    - subsample majority class
    - oversample minority class

2. Size of dataset
    - Collecting new data
    - Generating new data
3. Data selection
    - Data augmentation
    - Adding noise

IV. **Other**

1. Feature selection
2. Validation dataset selection by real life cases

Choosing one or more techniques from above always goes together with choosing an optimization target. Some of them can live parallel, some of them excludes any other target.

- learning
    - better: higher train accuracy, lower train loss
    - faster: reduced training time
- generalization
    - better: focus on real life cases
    - specialization: focus on some cases
- prediction
    - better: higher validation accuracy, lower validation loss
    - faster: reduced inference time

Number of layers (model.structure)

Adding or removing layers has great influence on the performance and size of a model and on the train time as well. An epoch cycle is faster on small networks. However, big models might learn deeper contexts. Dead nodes can decrease the model's performance.

Type of layers (model.structure)

Different layers can learn different things. Adding new type of layer can improve learning skills.

Number of nodes (model.structure)

Adding nodes can increase the learning ability without a significant size increasing compared to adding new layer into the model.

Bias (model.structure)

Bias is an option for the layer to learn or not learn additive bias during training.

Activation functions (model.structure)

Activation functions give the non-linearity factor to the deep learning models.

Batch normalization (model.structure)

Batch normalization is a technique to make model learn faster and be more stable. Normalization is a process where the data of input layers are re-centered and re-scaled.

Parameters (model.train)

Parameters give a lot of freedom to teach a model.

Loss function (model.train)

Loss function is a necessary part of train loop. It have to be fit for the problem.

Optimizer (model.train)

Optimizer is a required part of train loops like loss function. Different optimizers are needed to specific problems. It usually determines the next steps of optimization.

Dropout (model.train)

Dropout is one of the most popular methods to avoid stucking in a local minima and to avoid overfitting.

Manual learning rate (model.train)

Manual option means the user sets up the value. It can be the default value or any other constant.

Learning Rate Schedule (model.train)

Learning rate schedule means a way to use different learning rate during the training based on any property of training like epoch number, loss value.

Momentum (model.train)

Momentum factor helps model to converge better.

Weight Regularization (model.train)

Weight regularization is a method to avoid overfitting. There are multiple types of weight regularization like L1 and L2 norms.

Epoch number (model.length)

Epoch number represents the length of training. Usually it comes with early stopping.

Early Stopping (model.length)

Early stopping means jump out from a train loop before we go through the whole epoch cycle. It can help to avoid overfitting. Condition of stopping usually are loss and accuracy.

Nodes' initialization (model.init)

Nodes' initialization means create the node's weight with a specific value, that can be a constant, a random number or values based on any distribution method.

Bias' initialization (model.init)

Bias' initialization means create the layer's bias with a specific value, that can be a constant, a random number or values based on any distribution method.

Pretrained network with retrain (model.init)

Pretrained network with retrain is a very popular technique. It is similar like the initialization methods, but the start values comes from a pretrained network such as VGG, ResNet, etc. The pretrained model's wrights and biases will be modified after a retrain.

Pretrained network without retrain (model.init)

Pretrained network without retrain is a very popular technique. It is similar like the initialization methods, but the start values comes from a pretrained network such as VGG, ResNet, etc. The pretrained model's wrights and biases will be freezed and used unchangedly.

Retrain network: Transfer learning (model.init)

Transfer learning is a common technique where a trained model is retrained with the new train dataset.

Retrain network: Continue from checkpoint (model.init)

Continue from an early stopped checkpoint is the part of a retrain process. It helps to train with different hyperparameters or different batches.

Constant batch size (batch.batch_size)

Constant batch size means the batch size is static during training. However, the exact value has a huge impact on the result such as prediction accuracy and train time as well.

Various batch sizes (batch.batch_size)

Various batch sizes means the batch size is different over epochs. It can helps to avoid some error that can occur when using constant batch size.

Constant batch (batch.batch_selection)

Constant batch means the content of every batches are the same over epochs.

Managed batch selection (batch.batch_selection)

The managed batch selection means the content of a batch is sampled.

Randomized batch selection (batch.batch_selection)

Randomized batch selection means the content of a batch is randomly sampled.

Subsample majority class (data.dataset_balancing)

The first method of dataset balancing is subsample majority class that means data from majority class will be removed until every subset's size will be equal.

Oversample minority class (data.dataset_balancing)

The second method of dataset balancing is oversample minority class that means data from minority class will be copied multiple times until every subset's size will be equal.

Collecting new data (data.size_of_dataset)

This is the most advised tip to get a better result based on social media posts. In spite of the simplicity of idea, most of time it is not a real option.

Generating new data (data.size_of_dataset)

Generating new data could help to use bigger dataset.

Data augmentation (data.data_selection)

A good data augmentation causes better generalization. It can be used to make the dataset bigger.

Adding noise (data.data_selection)

Noise data helps better generalization and can avoid getting stuck on a local minima.

Feature selection (other)

Feature selection comes from machine learning, but it can improve the learning ability in deep learning too. Feature with very low influence on result makes the model size bigger and causes slower prediction. Using unnecessary features might add extra noise to the train process, that has bad influence on the accuracy of prediction.

Validation dataset selection by real life cases (other)

During training an optimal solution is to use subsamples with equal sizes. When we want to measure the performance on real life scenario, the validation dataset must be proportioned as real life cases. For example training a model with 2 classes, train dataset must be sliced 50-50%. However, if this is a medical situation where 80% of results are negative, the validation dataset have to contain 80% negative and 20% positive case ratio. This helps to measure the performance in real life scenario.

Market

When we focus about the problem of batch size during training, we should check how the great companies are thinking about this question.

Nvidia's GPUs are the base of deep learning. Since the profit of Nvidia depends on the demands, their products always refers to the conditions of the market. Tesla V100, the first tensor core GPU, comes in 16 or 32GB configurations. Besides the high computational capacity and fast memory modules, it offers huge GPU-RAM.

Intel is great multiprocessor supplier. It researches about batch size and generalization. However, researchers advised to use smaller batches, they had made the research, since batch size optimization is a very popular method to upgrade the learning process.

Facebook made research with large batch size.

Google gives limited computational resources with huge GPU-RAM in Google Colab to everyone for free. This offer contains a K80 GPU with 12GB GPU-RAM. Customers from USA can pay or Colab Pro. For a small amount of money they give access to TPUs such as T4 and P100. The top limit of GPU-RAM is not defined, but it depends on the load of resources. So resources are not guaranteed.

AWS is a giant cloud service provider. It supports hackathons worldwide. Its offer contains enormously huge RAM such as 256GB, 512GB, 712GB or 768GB.

Lambda GPU cloud offers two different instances. 4×11GB or 8×16GB seems quite enough.

Exoscale created solutions up to 225GB RAM.

As it is seen, the greatest companies offer huge RAM or make researches with large batches. The market shows, people and companies are under the impact of huge RAM and large batch or at least the batch size modification as an optimization method.

Survey

We decided to take a survey about GPU-RAM and Batch size with some added extras such as deep learning frameworks and source of knowledge. The main targets were programmers, developers and researchers. In the public call, we speak about the survey with the focuse on connection between deep learning and GPU utilization, we didn't speak about infinitybatch. The cause behind this behavior is we don't want to influence the responders. There are questions that effect negatively on a survey accuracy. We made this survey via internet with the cooperation of a very small population. That's why we tried to avoid any biases, manipulation or influence as possible to get the most accurate result.

We published the public call on the following sites under own real name or under own personal account:

[18. 08. 2020.]

- Facebook
  - ↳ Artificial Intelligence and Machine Learning [private group]
  - ↳ Udacity Alumni Network [private group]
  - ↳ PyTorch Scholarship Challenge Program from Facebook [private group]
  - ↳ Udacity Community - Europe [private group]
  - ↳ Udacity Scholarship Participants [private group]
  - ↳ Udacity AI Hungary [private group]
  - ↳ Developer Circles: Machine Learning [private group]
- Slack
  - ↳ Udacity Alumni > nd-deep-learning

[19. 08. 2020.]

- Slack
  - ↪ Udacity Alumni > random

[20. 08. 2020.]

- Reddit
  - ↪ artificial
  - ↪ MachineLearning
  - ↪ pytorch
  - ↪ tensorflow

[23. 08. 2020.]

- Reddit
  - ↪ deeplearning

This was the full version of public call:

Hi my fellow programmer,

I am programmer who work in the field of deep learning. Me and my colleagues usually make lectures and statistics about fields that we are interested in. That's why we created a survey about the connection between deep learning and GPU utilization. If you have 3-4 minutes, please help us by answering our questions.

https://forms.gle/2L6nUfxSN5KyVo786

Thank you for your time and answers,
Axel / Richard

Short description about survey

The survey is started with a short description. Its goal is to hide the survey's real focus. Maybe it seems deceptive that it is the only way to minimize the distorting effect. When people know we create a batch based research, it is a great chance that their responses adjust to their supposition about hypothetically perfect answers to the problem of batch size instead of the real life scenario.

Dear Responder,

We created a survey about the connection between deep learning and GPU utilization. We are programmers who work in the field of deep learning. Beside the work, we are learning. This small project is a part of a big journey. We usually make lectures and statistics about fields that we are interested in. Please help us by answering our questions. We would like to know what our fellow colleagues from around the world think about this topic. There are no right or wrong answers. If you help us, we will be very-very happy.

Thank you for your time and answers.

Cheers,
Axel and Richard

Questions

1. **What describes you better? You are a... [O]***
   - programmer
   - developer
   - researcher
   - gamer
   - student
   - hobbist
   - none of the above
2. **Do you have your own GPU? [O]***
   - Yes
   - No
3. **What is the memory size of your GPU (in Gigabytes)? [O]***
   - I don't have own GPU
   - 2 or less
   - 4 (more than 2 but no more than 4)
   - 6 (more than 4 but no more than 6)
   - 8 (more than 6 but no more than 8)
   - 10 (more than 8 but no more than 10)
   - 12 (more than 10 but no more than 12)
   - more than 12

4. **How do you usually train your models at home? [O]***

- I don’t use my personal computer to train.
- CPU
- single GPU
- GPU cluster
- cloud service
- other

5. **How do you usually train your models at workplace / school? [O]***

- I don’t use my workstation / school computer to train.
- CPU
- single GPU
- GPU cluster
- cloud service
- other

6. **Do you try to different batch size during training process? [O]***

- Always
- Usually
- Rarely
- Sometimes
- Never

7. **How often do you get memory error (such as out of CUDA memory or any other memory overrun) message during training? [O]***

- Always
- Usually
- Rarely
- Sometimes
- Never

8. **Which framework do you use? [M]***

- Caffe
- Chainer
- DL4J
- Keras
- MATLAB
- Microsoft CNTK
- MXNet
- ONNX
- OpenVINO
- PaddlePaddle
- PyTorch
- TensorFlow
- Theano
- other

9. **How familiar with these libraries? [R]***
   never heard / never used it / just tested it / use it rarely / use it regularly

   - Aequitas
   - Seaborn
   - Scikit-Learn
   - SciPy
   - Matplotlib
   - NLTK
   - NiBabel
   - NumPy
   - OpenCV
   - Pandas
   - PyBigData
   - Pydicom
   - PyTessy
   - PySpark
   - Tensorboard

10. **Do you think finetuning the hyperparameters matters for the performance of a model? [O]***

    - Yes
    - No

11. **Do you finetune hyperparameters? [O]***

    - Yes with manual settings
    - Yes with automated way
    - No

12. Where are you from? [O]*

- Europe
- North-America
- Mid- and South-America
- Africa
- Australia and New Zealand
- Middle-East
- South and West Asia
- North and Middle Asia
- I prefer not to say it.

13. Where did you learn about machine learning, deep learning? [M]*

- College / University
- Course at your workplace
- Online course
- alone with DIY learning (eg: book, Youtube)

14. Where did you meet first with machine learning, deep learning? [O]*

- College / University
- Course at your workplace
- Online course
- Online video sharing site
- Book
- none of the above

15. What are your favourite techniques to optimize the performance of a model? Please, describe them. [F]

Method

The questions can be different type like only choice [O], multiple choice [M], a rating matrix [R] or free-response questions [F]. Most of them are mandatory that a "*" sign represents after the question type marker.

First of all, our survey isn't representative, since the filling is voluntary and we made a public call via internet. It seems a good assumption that people who are committed to the community is overpopulated in the responders. This is not a bad situation, but we must to remember, there are lonely wolf programmers around the world who does not like using social media or any other community channel for programmers. We did not have the resources to get demographically absolutely representative data.

However, we are developers, data science is the part of our daily routine. That's why we cannot miss to talk about the lacks or weakness of survey. The goal of this survey to show the trends in the world. We think despite of limitation are mentioned above, this little survey is capable of filling this position.

Question 1

We are curious how people describe themselves. This question means nothing standalone, but it can refine the whole picture. If a lot of none of above occur, we can remove those answers for avoiding distortion.

Question 2

Our project's persona has own GPU at home or at the company / school. This question measure the first target group.

Question 3

Size of GPU-RAM might be interesting. If a lot of user has limited resources, it is good for tools like infinitybatch. However, a user with huge GPU-RAM can also profit from our solution.

Question 4

Infinitybatch has very different use cases. We can add something into training. The biggest advantages pop up when user uses own GPU or GPU cloud. The only group lagging behind is the CPU-based trainers. This question is about home computer.

Question 5

Infinitybatch has very different use cases. We can add something into training. The biggest advantages pop up when user uses own GPU or GPU cloud. The only group lagging behind is the CPU-based trainers. This question is about work or school computer.

Question 6

It seems unnecessary to ask about the frequency of batch size modification, we are very curious the portion of “never” answers.

Question 7

Infinitybatch is very useful who get CUDA memory overrun messages usually at least since it helps to avoid this type of error.

Question 8

We want to know what is the ratio between PyTorch and every other popular framework. We made an other statistic about this topic that is based on github searches.

Question 9

This question focuses on the popular libraries. However, it contains two tricky answers. First is “PyBigData” that does not exist. There are pip packages with similar names, but this is a fiction that can help filter responders who want to make shiny and proudable responses. The second tricky answer is PyTessy. It is one of our projects and we put it just for fun.

Question 10

The purpose of this questions is to lead up the next question.

Question 11

Manual finetuners are the member of our target audience. Who use automatic method might find infinitybatch interesting as well.

Question 12

The location question is always sensitive. However, most of people live in freedom, there are a lot of causes while people do not want to share their location. That's why we made answers that embraces huge areas and a unique answer for people who do not want to answer this question at all.

Question 13

It reveals the correlation between the source institute of learning and the habits. This question is not related to our project but we are curious about answers, since we learned from online courses and DIY.

Question 14

The previous question is a multichoice one, so the first source should be concretized

Question 15

At the beginning we wanted to create a question about responder's favourite optimization techniques. The responders could answer on a scale. In the end, we dropped the idea and switched the question into a non mandatory, free response question. Collecting and ordering answers are harder, but the data is more valuable. Offering a lot of options could confuse the responder and force him into a very strange situation where they could rate higher the less used techniques. The unique answers of every techniques could be correlated even if there is no real correlation. Nobody wants to see that they use mostly

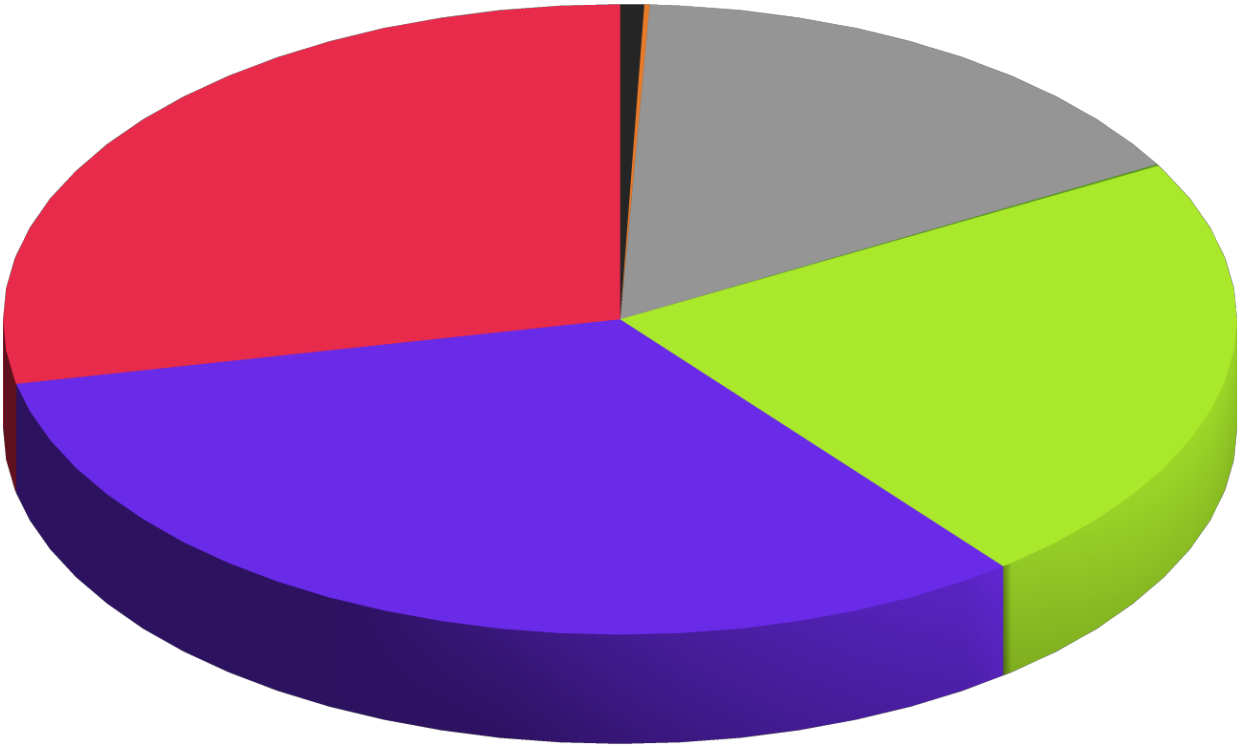
one or two techniques. Or maybe they are deceptived by scale and made them connect a higher frequency of usage with the given techniques than the method occurs in real life. We want to avoid any type of conflicts like these and try to get accurate answers as possible.

Survey closed: **25. 08. 2020. 12:00 CEST**

Total number of filled survey: **1,439**

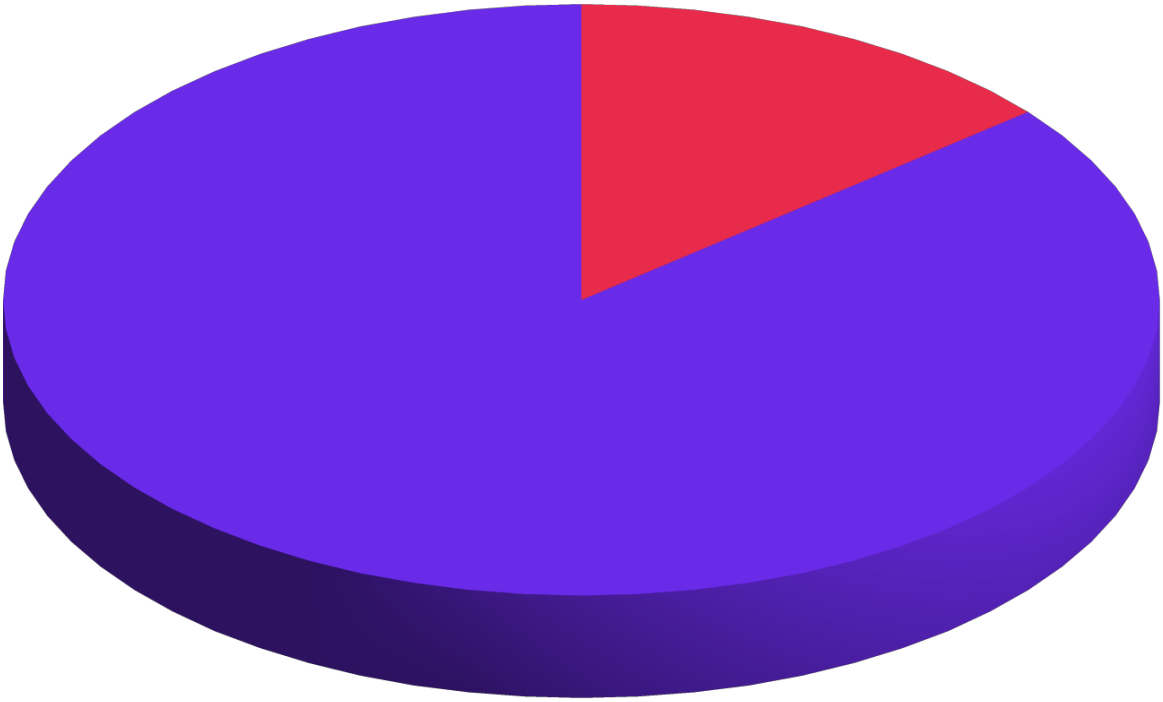
Result

What describes you better? You are a...



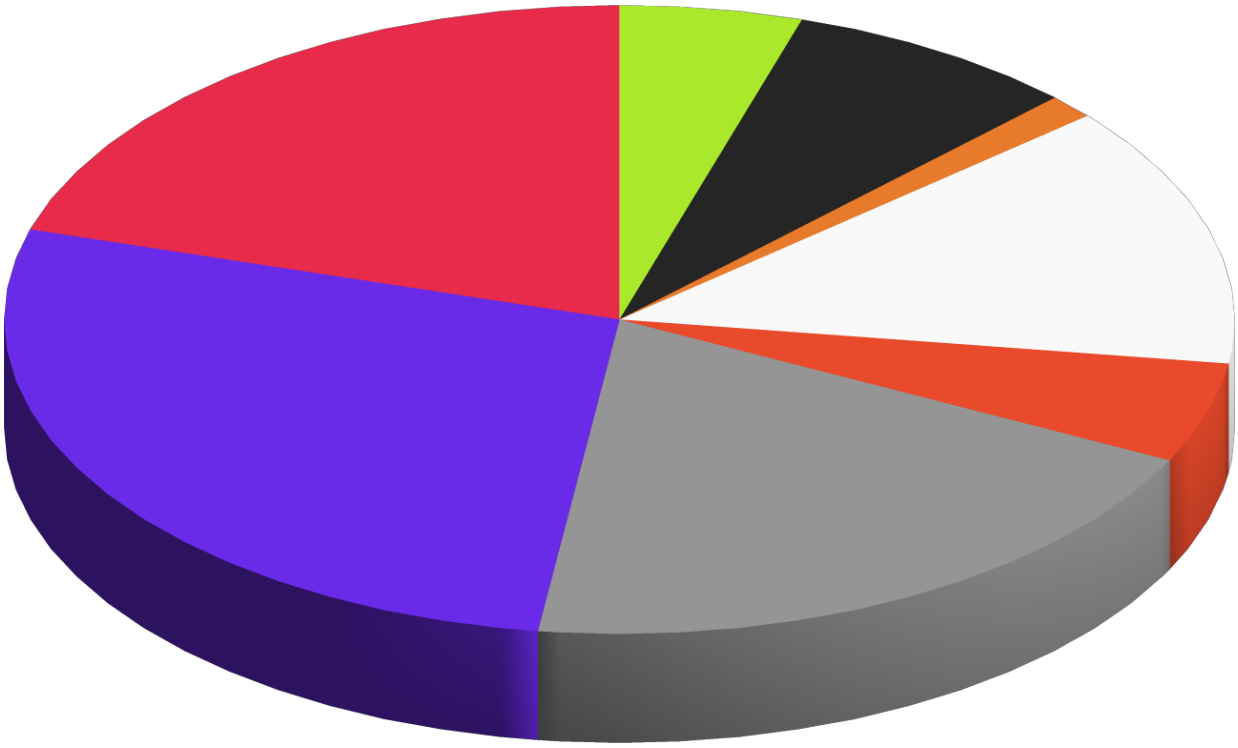
	answer	count	ratio
	programmer	407	28,28%
	developer	466	32,38%
	researcher	323	22,45%
	gamer	1	0,07%
	student	231	16,05%
	hobbist	2	0,14%
	none of the above	9	0,63%

Do you have your own GPU?



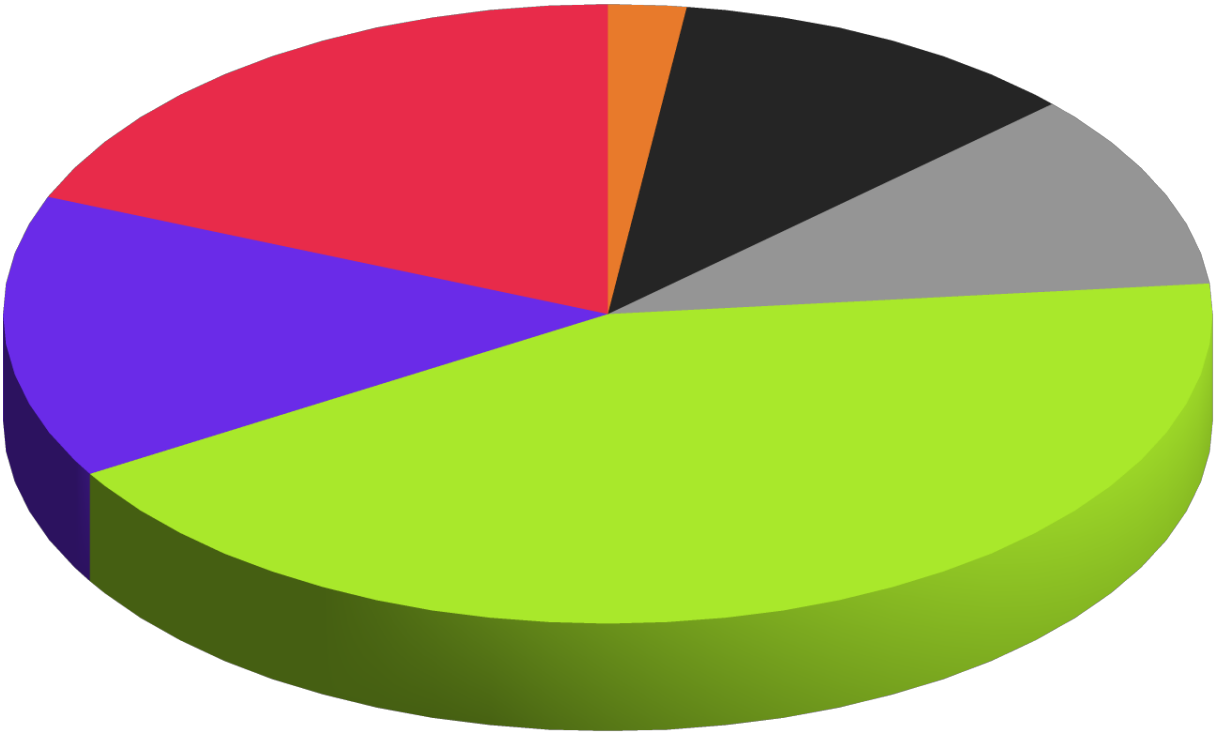
	answer	count	ratio
	Yes	1237	85,96%
	No	202	14,04%

What is the memory size of your GPU (in Gigabytes)?



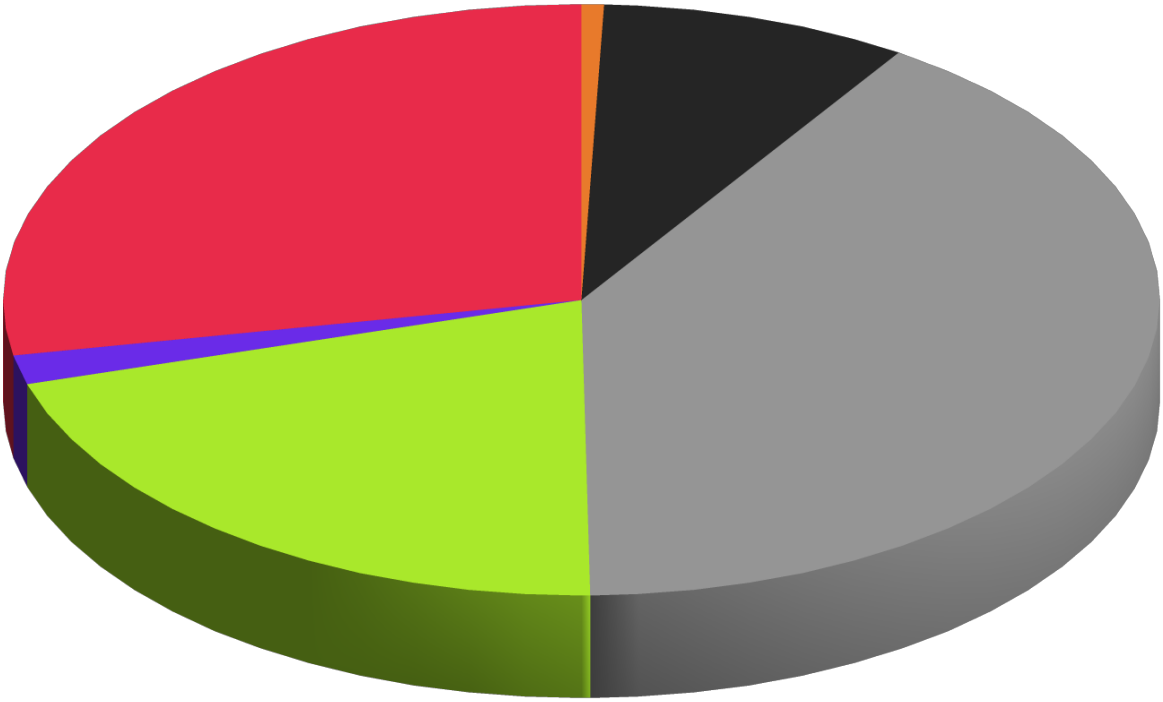
	answer	count	ratio
	I don't have own GPU	293	20,36%
	2 or less	396	27,52%
	4 (more than 2 but no more than 4)	284	19,74%
	6 (more than 4 but no more than 6)	74	5,14%
	8 (more than 6 but no more than 8)	194	13,48%
	10 (more than 8 but no more than 10)	18	1,25%
	12 (more than 10 but no more than 12)	111	7,71%
	more than 12	69	4,79%

How do you usually train your models at home?



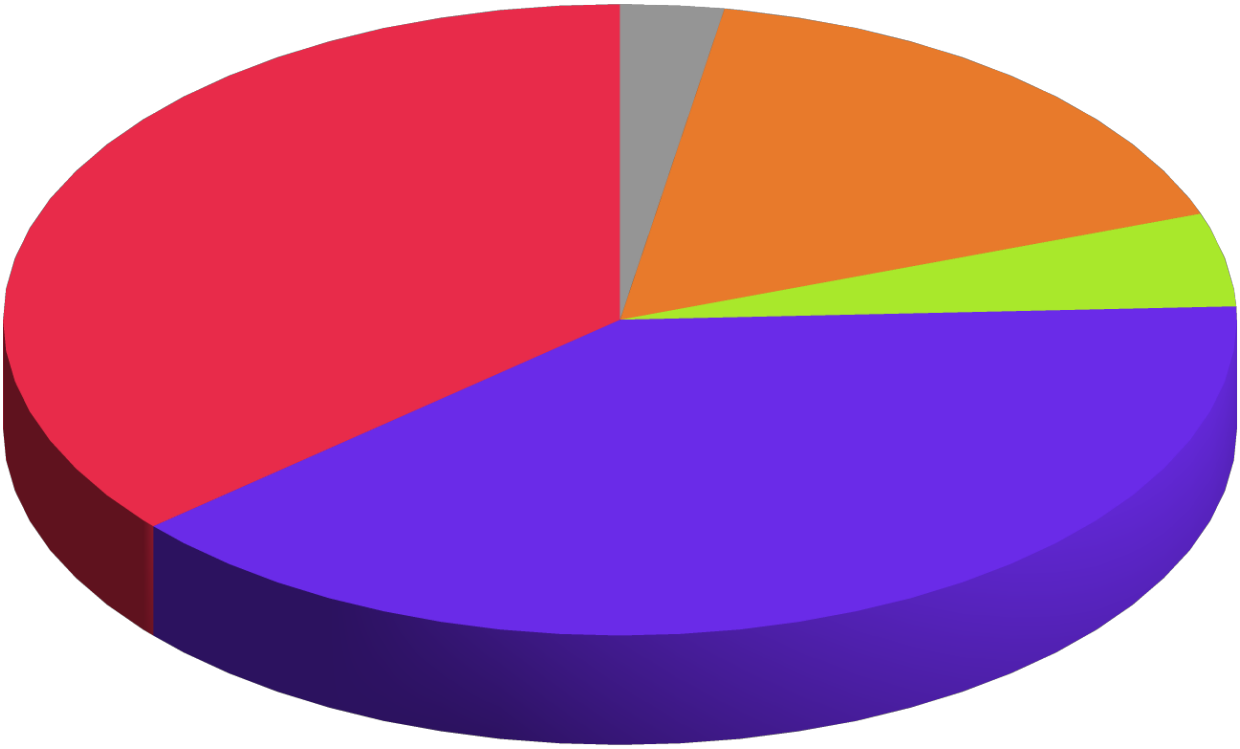
	answer	count	ratio
	I don’t use my personal computer to train.	271	18,83%
	CPU	213	14,80%
	single GPU	618	42,95%
	GPU cluster	148	10,28%
	cloud service	159	11,05%
	other	30	2,08%

How do you usually train your models at workplace / school?



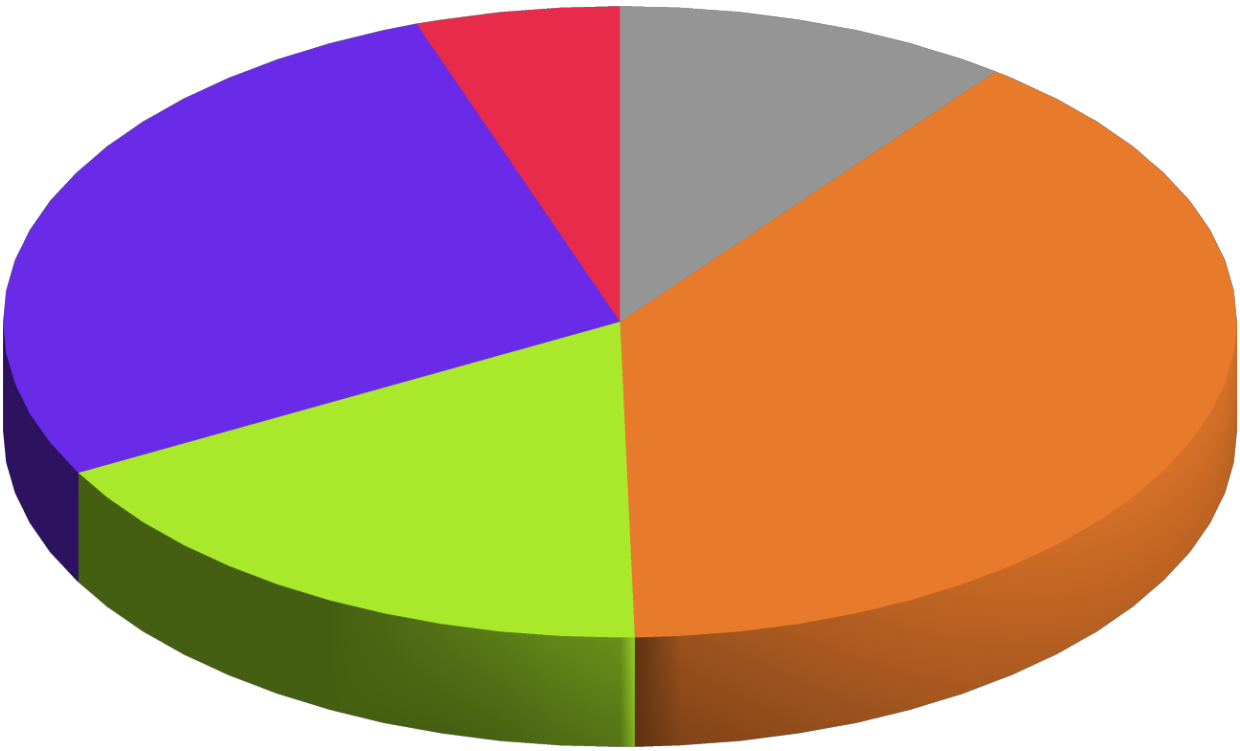
	answer	count	ratio
	I don't use my workstation / school computer to train.	403	28,01%
	CPU	23	1,60%
	single GPU	297	20,64%
	GPU cluster	583	40,51%
	cloud service	124	8,62%
	other	9	0,63%

Do you try to different batch size during training process?

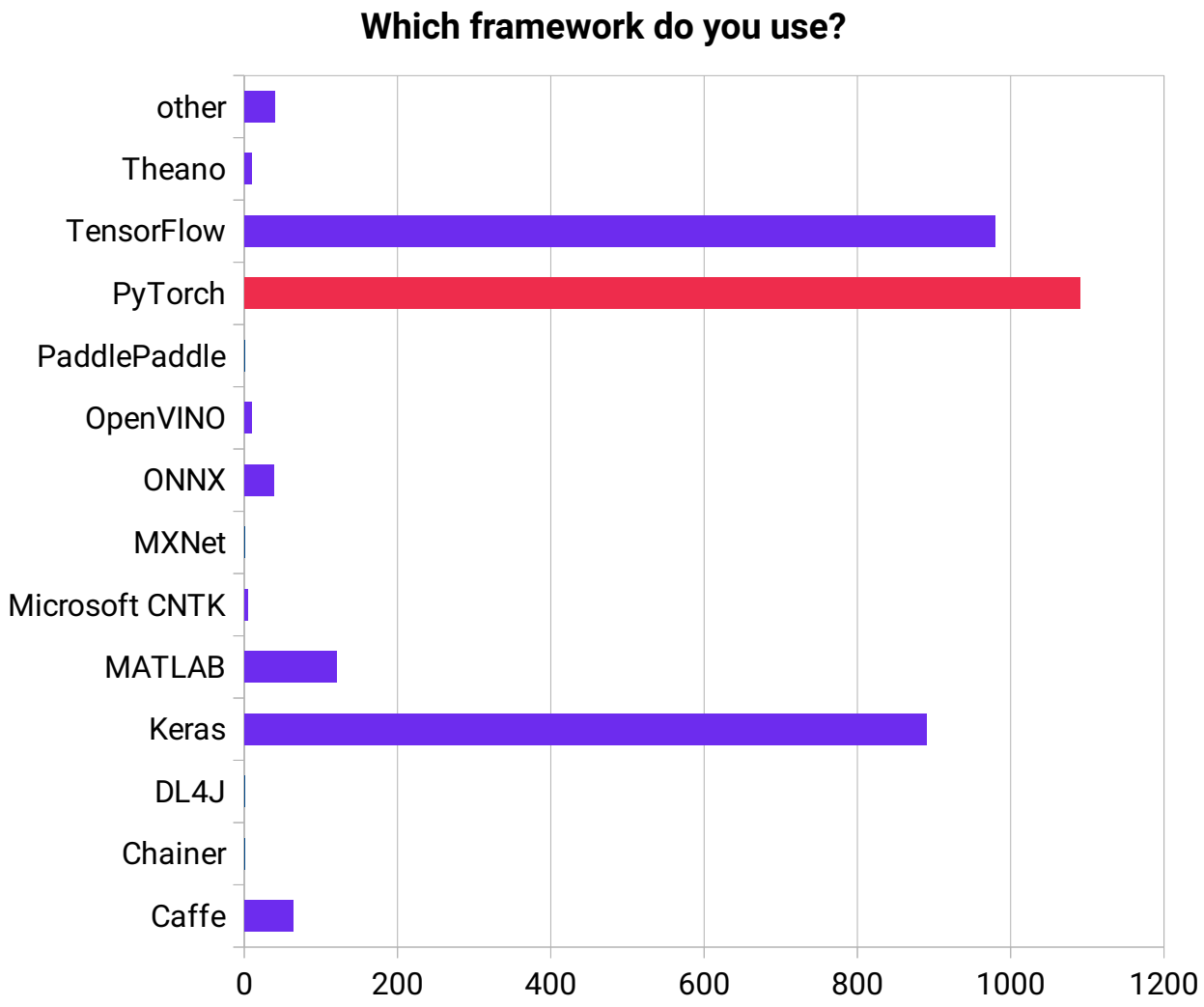


	answer	count	ratio
	Always	523	36,34%
	Usually	566	39,33%
	Rarely	69	4,79%
	Sometimes	242	16,82%
	Never	39	2,71%

How often do you get memory error (such as out of CUDA memory or any other memory overrun) message during training?



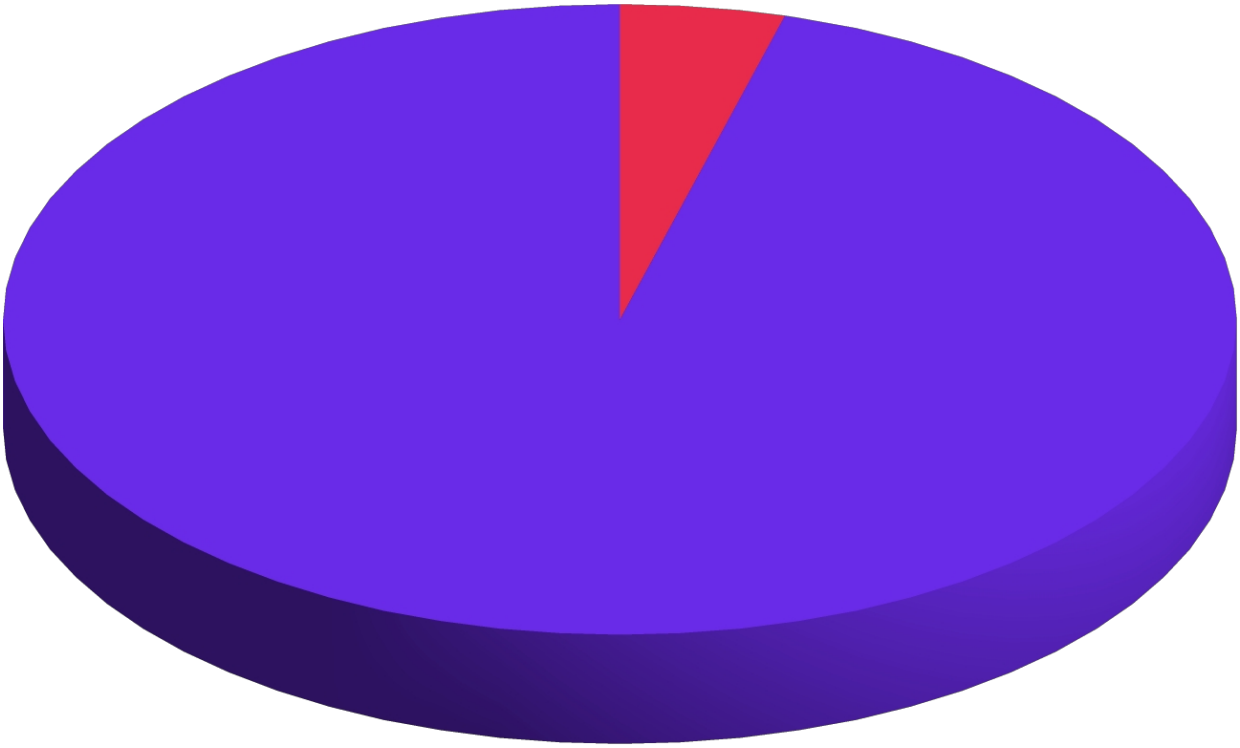
	answer	count	ratio
	Always	77	5,35%
	Usually	397	27,59%
	Rarely	251	17,44%
	Sometimes	564	39,19%
	Never	150	10,42%



	answer	count	ratio
	Caffe	64	4,45%
	Chainer	0	0,00%
	DL4J	0	0,00%
	Keras	890	61,85%
	MATLAB	120	8,34%
	Microsoft CNTK	5	0,35%
	MXNet	0	0,00%

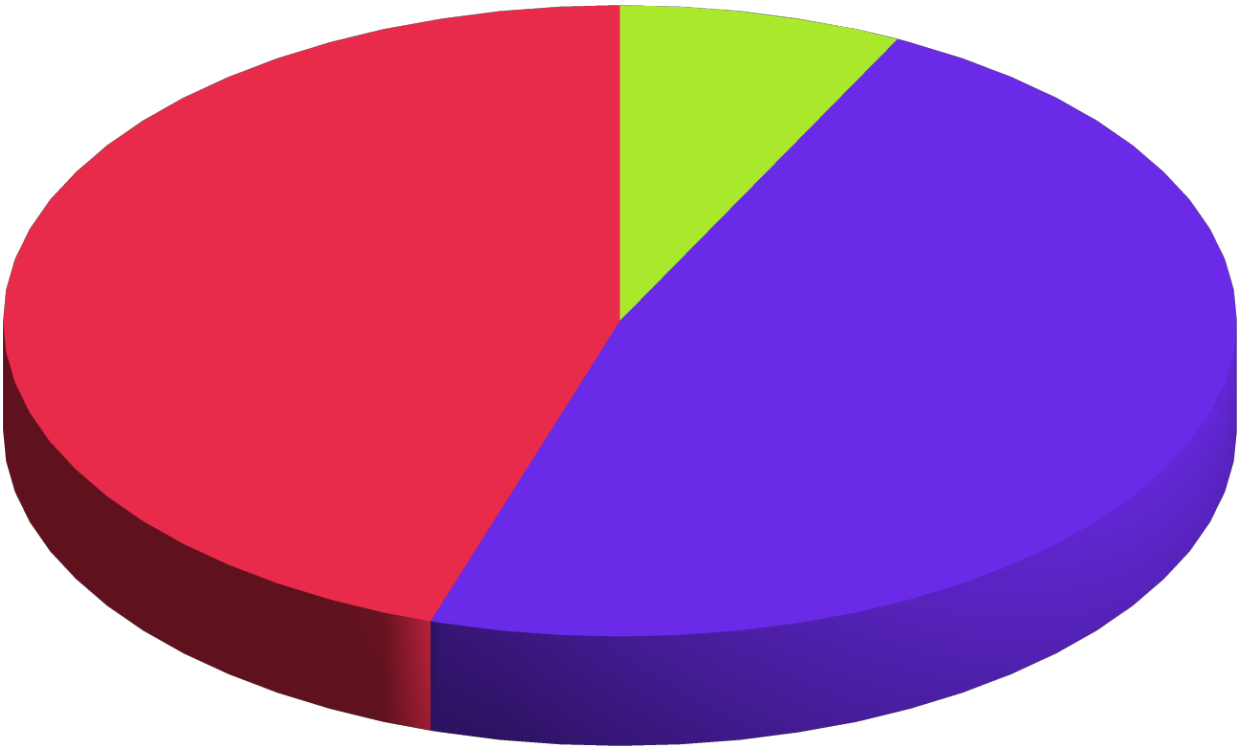
	answer	count	ratio
	ONNX	39	2,71%
	OpenVINO	9	0,63%
	PaddlePaddle	0	0,00%
	PyTorch	1090	75,75%
	TensorFlow	979	68,03%
	Theano	10	0,69%
	other	40	2,78%

Do you think finetuning the hyperparameters matters for the performance of a model?



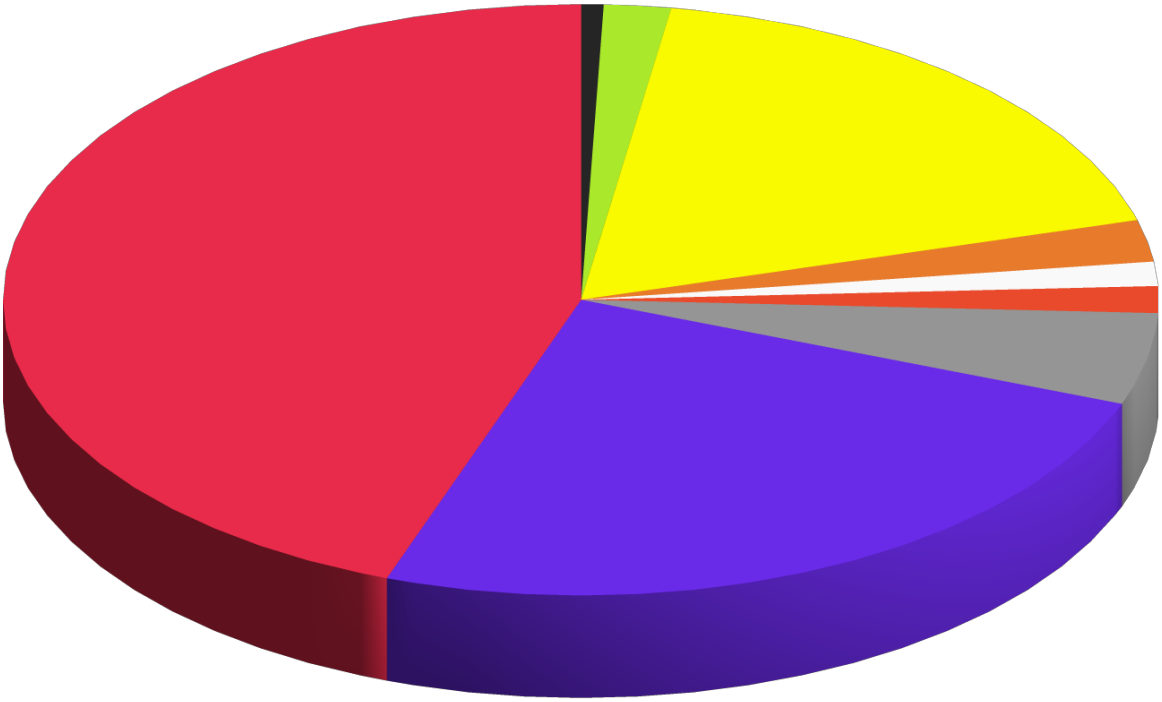
	answer	count	ratio
	Yes	1377	95,69%
	No	62	4,31%

Do you finetune hyperparameters?



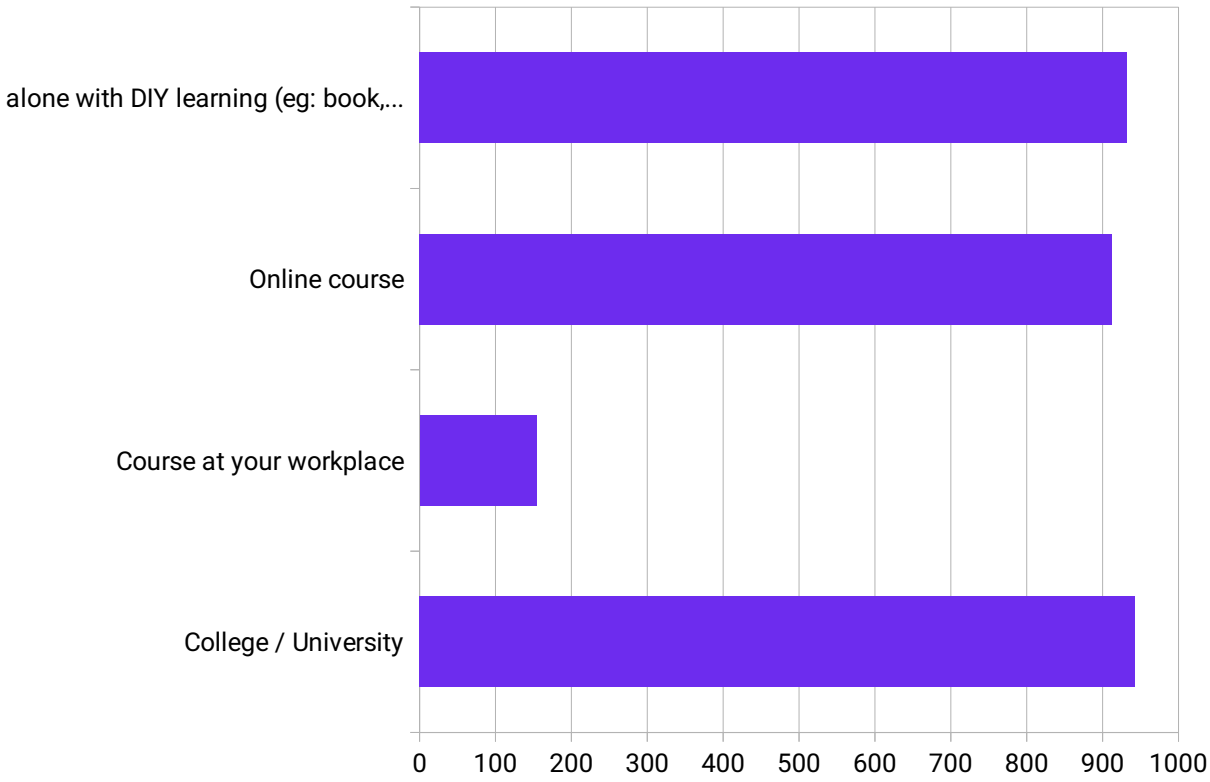
	answer	count	ratio
	Yes with manual settings	648	45,03%
	Yes with automated way	684	47,53%
	No	107	7,44%

Where are you from?



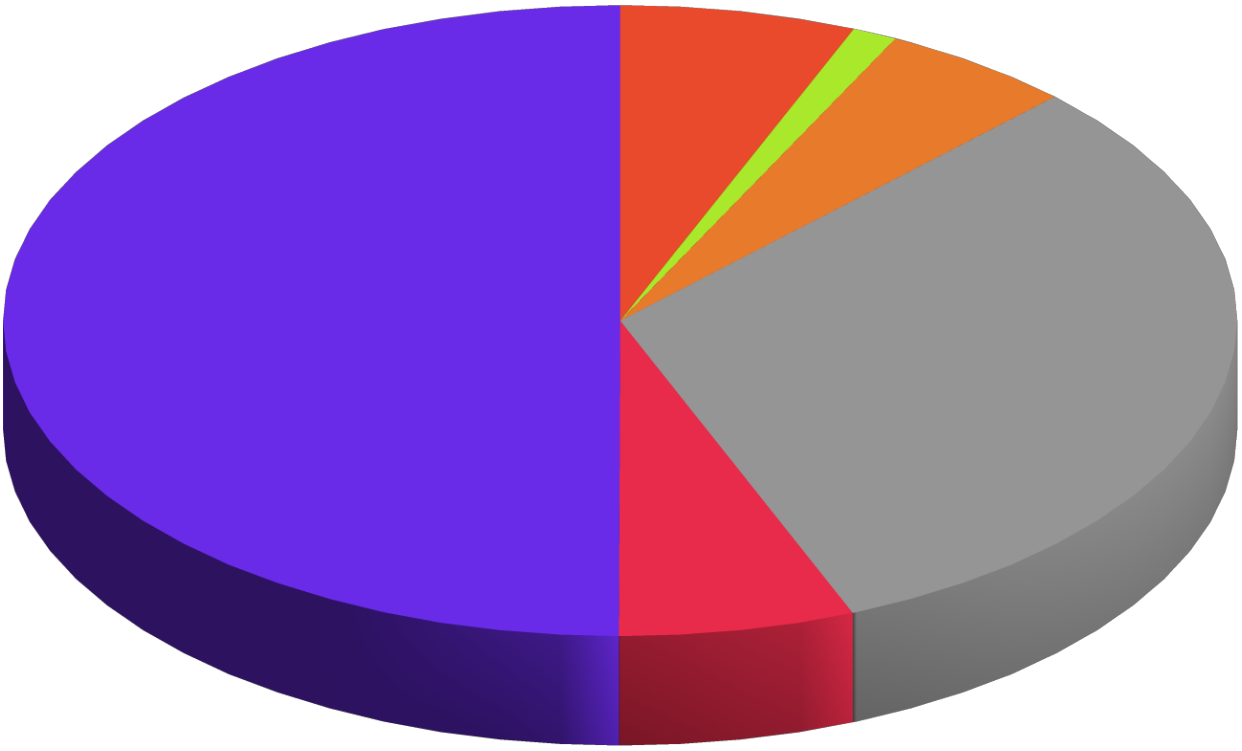
	answer	count	ratio
	Europe	641	44,54%
	North-America	356	24,74%
	Mid- and South-America	72	5,00%
	Africa	21	1,46%
	Australia and New Zealand	19	1,32%
	Middle-East	33	2,29%
	South and West Asia	261	18,14%
	North and Middle Asia	27	1,88%
	I prefer not to say it.	9	0,63%

Where did you learn about machine learning, deep learning?



	answer	count	ratio
	College / University	943	65,53%
	Course at your workplace	154	10,70%
	Online course	913	63,45%
	alone with DIY learning (eg: book, Youtube)	932	64,77%

Where did you meet first with machine learning, deep learning?



	answer	count	ratio
	College / University	719	49,97%
	Course at your workplace	89	6,18%
	Online course	452	31,41%
	Online video sharing site	73	5,07%
	Book	17	1,18%
	none of the above	89	6,18%

Infinitybatch

Infinitybatch is an open source solution that helps deep learning developers to train with bigger batch size than it could be loaded into GPU RAM. In this chapter we show the workflows, dataflows and how our code integrate into PyTorch's system.

We will prove that our code works similar the normal PyTorch train loop but it helps to train with bigger batch size than can fit into GPU-RAM at the same time.

License

Infinitybatch is open source software under MIT license. The core concept behind choosing a proper license is following the daily practice in IT and especially in deep learning field and be guided by PyTorch. PyTorch is under a modified 4-clause BSD license.

An open source solution does not generate money income, but it can help to monetize other ideas, software or solution. It depends on the decision of platform and its supporter, for example PyTorch and Facebook.

Formulas

We use some easy formulas to show how infinitybatch works and what benefit can it takes to the user.

Limitation

The only one limitation with Infinitybatch is that at least the model and one element at the same time must be fit into GPU-RAM.

$$LIM: MM + ES \leq GR$$

LIM = Limitation

MM = Model's memory consumption

ES = Memory consumption of the biggest element from the dataset. This equals the highest memory consumption that can reach any element from dataset

GR = Available and allocatable GPU-RAM for PyTorch

Perfect batch size based on CUDA cores

The ideal batch size is when each batch fits perfectly well into GPU memory and loads full the CUDA cores. The second condition is the train dataset is sliced without remainder, so there are no incomplete batch.

$$CC \% BS = 0 \;\&\&\; TL \% BS = 0$$

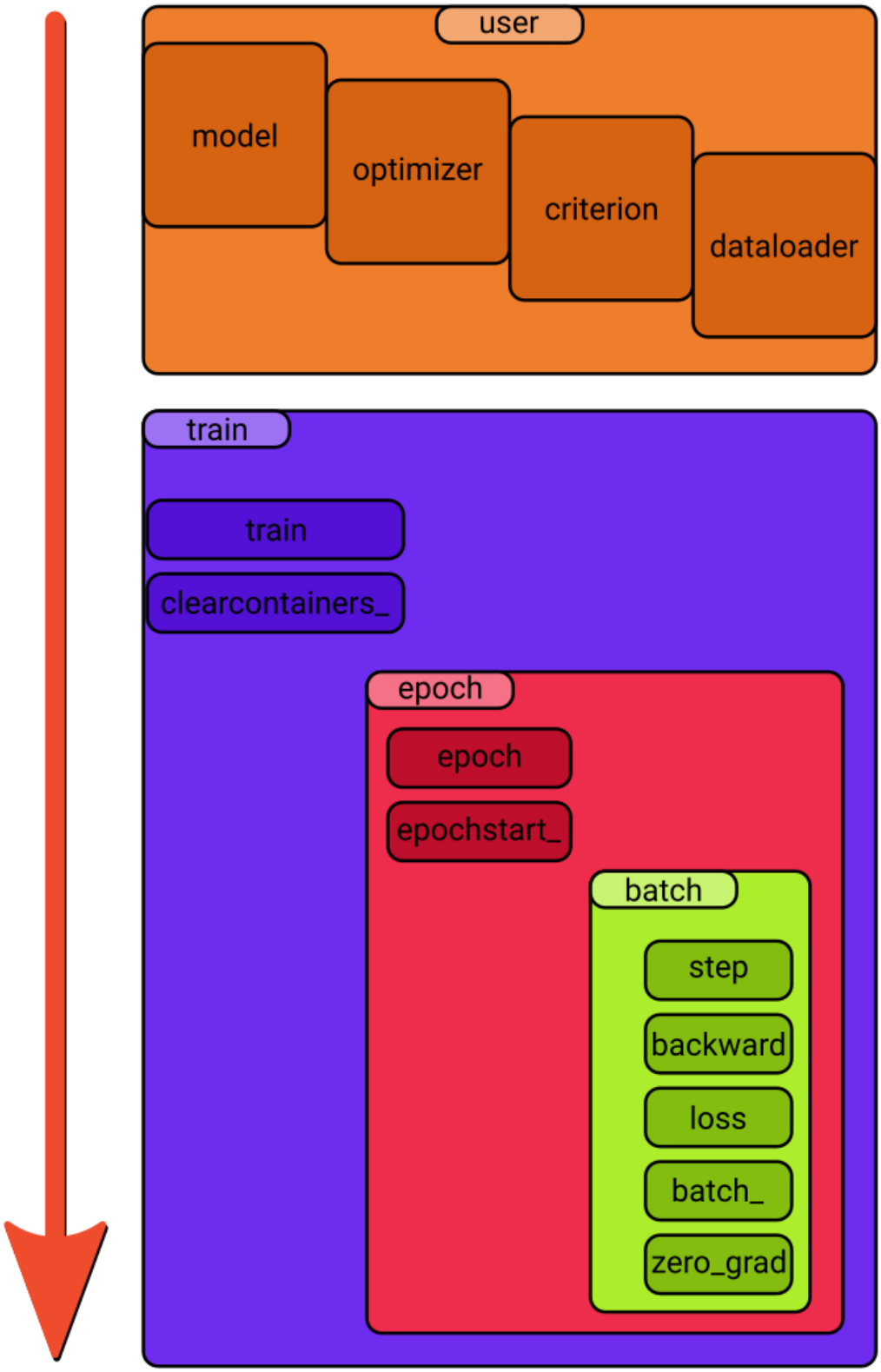
BS = Batch size

CC = Number of CUDA cores

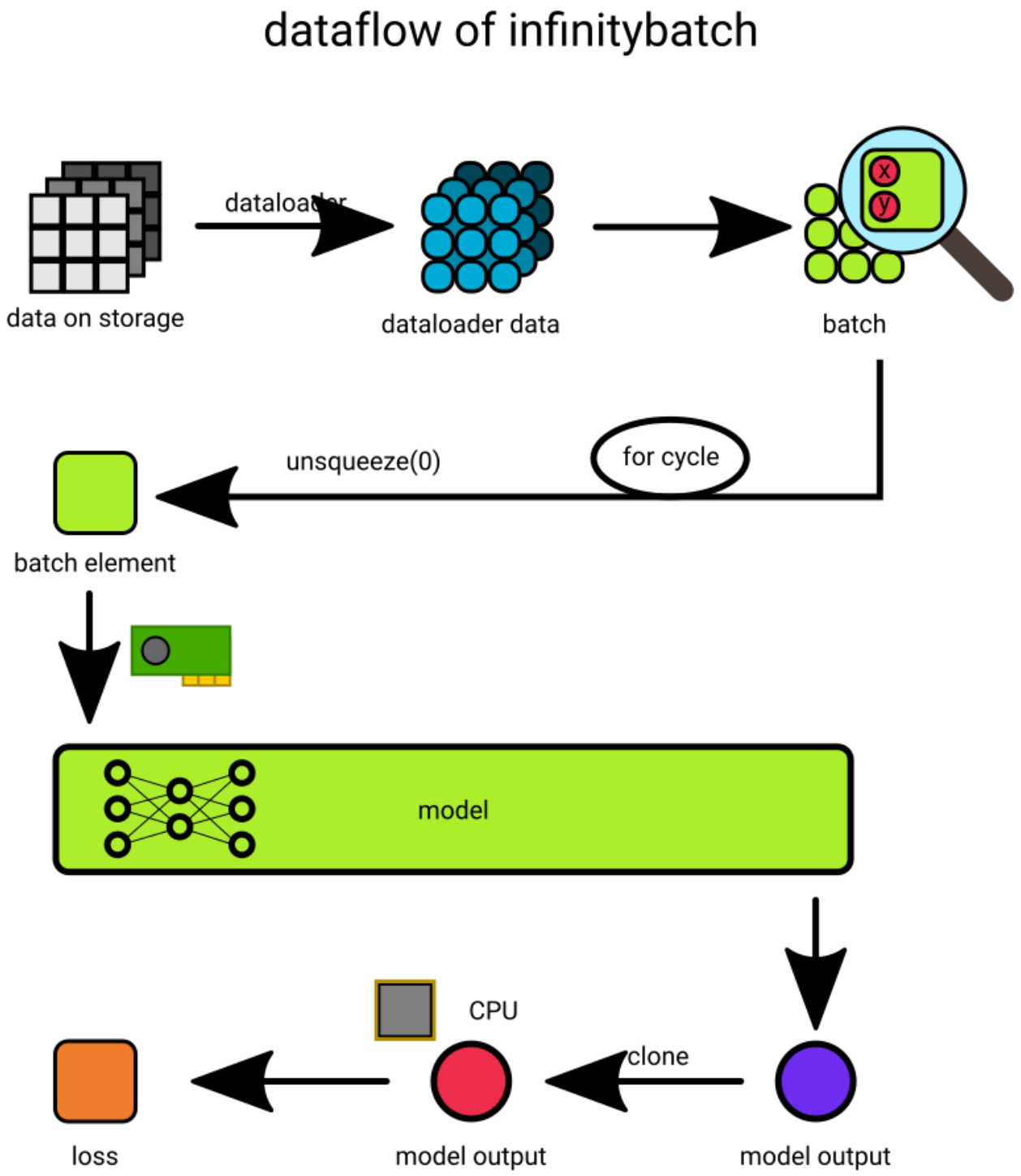
TL = Length of train dataset

Structure

workflow of infinitybatch



Dataflow


dataflow of infinitybatch
dataloader
data on storage
dataloader data
x
y
batch
unsqueeze(0)
for cycle
batch element
model
CPU
clone
loss
model output
model output

Intended use

Infinitybatch is an open source solution that helps deep learning developers to train with bigger batch size than it could be loaded into GPU RAM.

Competitors

There are a lot of forum topics about possible solutions. Using the same batch size, compared to a PyTorch normal train loop, the result with the existing solutions will be slightly different. However the results should be the same. There are no real competitors, since earlier solutions get back bad results.

Benefits for PyTorch

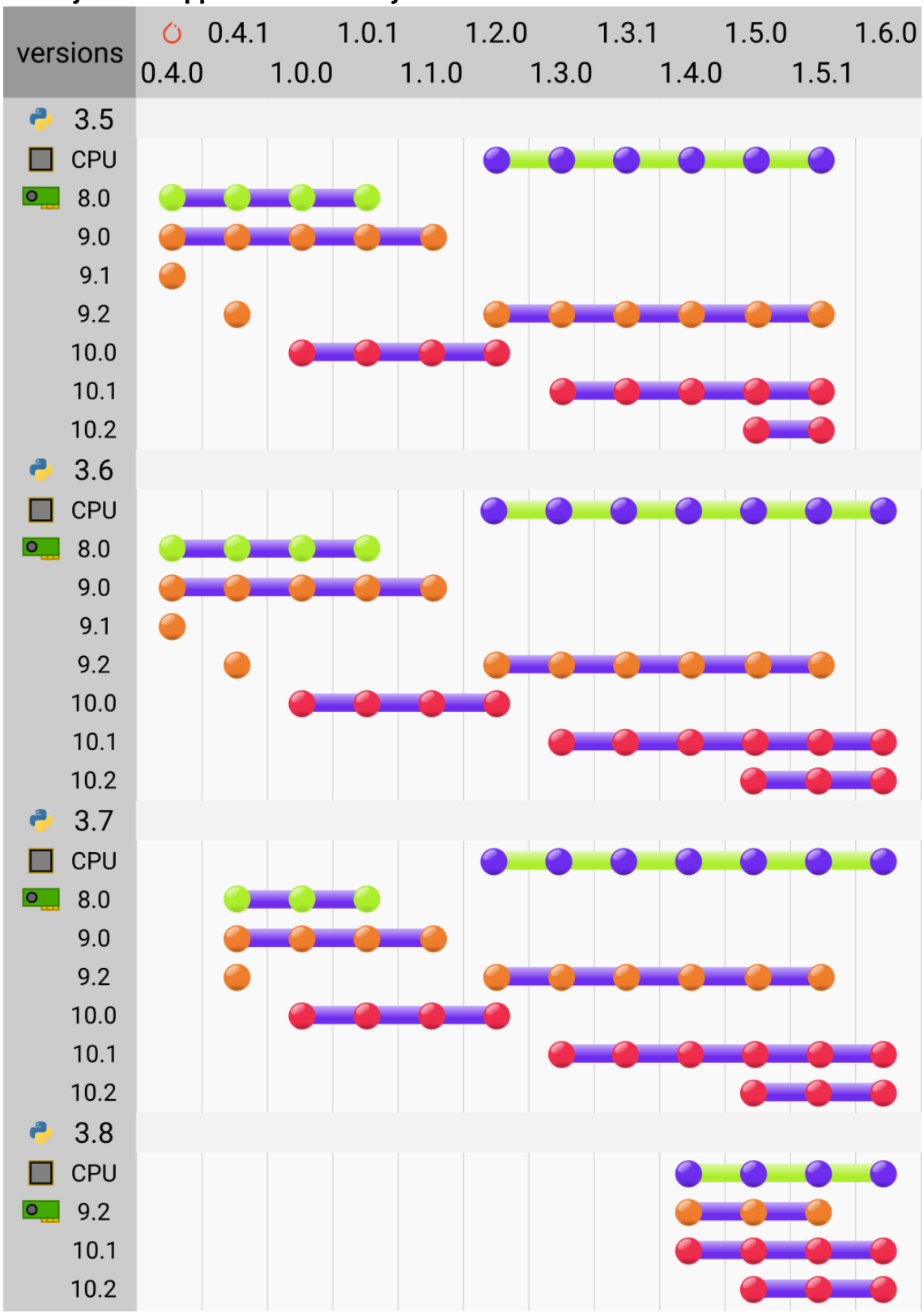
Infinitybatch is good for PyTorch since it terminates a real and live gap in training process. With this solution PyTorch can be satisfy users' demands that other major deep learning frameworks aren't able to handle. Our solution helps PyTorch to expand for new group of users. Small businesses and developers with low or mid-range GPU will be able to test new approaches, methods or train models with own limited resources. When a developer are familiar with one of framework, they will pursue to use that framework during at workplace. This opens gate to new businesses for PyTorch.

Integration into PyTorch

Infinitybatch fits perfectly into PyTorch and PyTorch's workflow. It is not override or exploit any PyTorch's function or class. A train loop with infinitybatch suits smoothly into a logic of a common train loop. The user initializes a model, an optimizer, a criterion and a dataloader as it is seen on the picture in the chapter named "Structure".

This kind of workflow guarantees the compatibility through different PyTorch version in the future as well. If the **torch.stack** and **torch.clone** will be the same and the core concept of autograd function will remain, there are no harm to this compatiblity.

How PyTorch supports different Python versions based on CPU or CUDA version



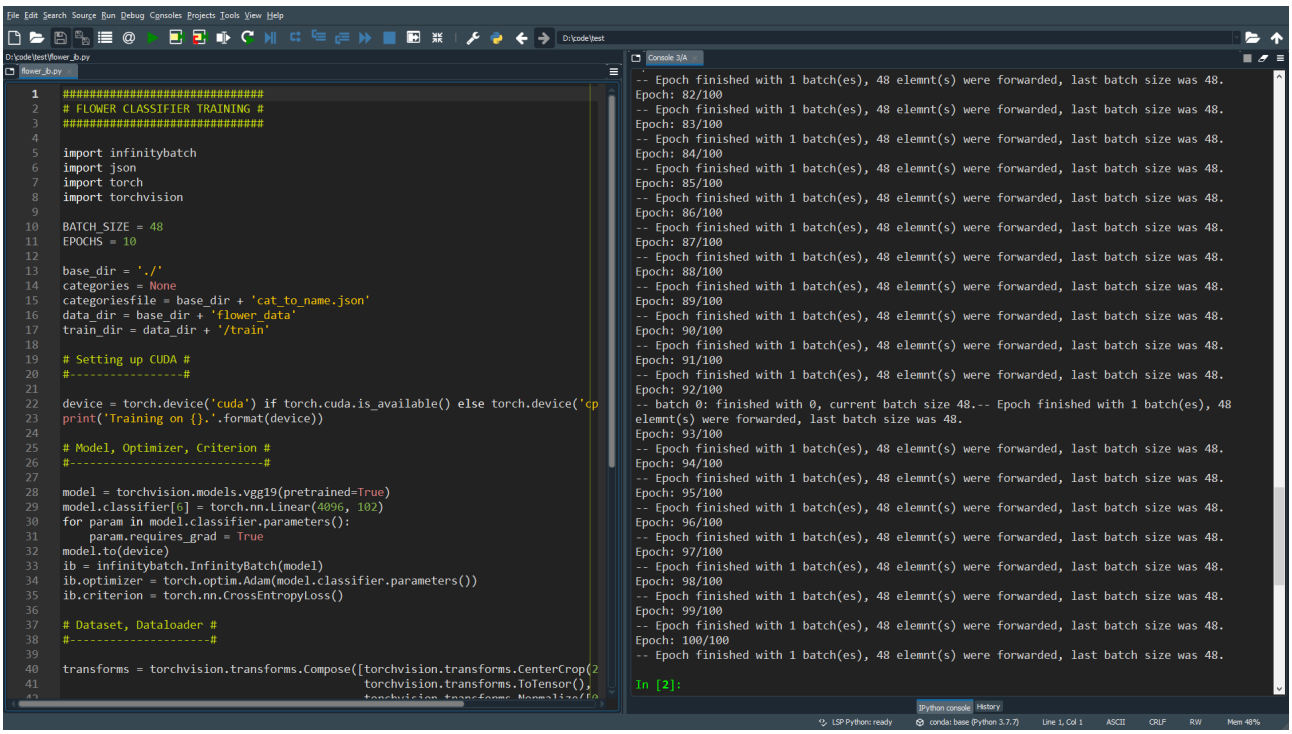
source: PyTorch's Anaconda channel [03-08-2020 14:15:00 CEST]

UX

The persona of infinitybatch is a deep learning programmer, developer or researcher who has GPU or train any GPU related architecture. The latter condition can mean multiple GPUs in one computer or a local cloud service as well. Our persona like to play with hyperparameters, especially batch size, to get an optimal result. This is a very important criterion, since infinitybatch is useful only for those, who try to solve a problem with different parameters that includes different batch size too.

The user get a complex solution that can be used as a Keras-like train manager or built the whole process brick by brick to get a quite personalized train loop. Beside the main core of infinitybatch we give a handy tool, that helps to check our statements and promises about infinitybatch. However, this tool can be used without the main program, since it contains a lot of function, for example a function that sets up random seed or CUDA-related functions.

The code is remarked and documented. We take useful examples that help to understand the workflow of infinitybatch.



The gauntlet[1]

In this chapter we try to collect those arguments that can be pop up in the mind of audience when they are hearing about infinitybatch. Some of them comes from our real life examples. Since the first battle is the hardest, we tried to speak about infinitybatch with our friends. We know already, the family dinner is not the best place and time for talking about infinitybatch. The most voiced arguments are listed below with appropriate reply.

Slow

Frankly, we never promised that infinitybatch provides a super fast training. So yes, it is slower in batch and epoch level. When we measure the whole training time consumption it the performance depends on two factors. Depending on the problem, using infinitybatch might provide time saving. It is occur when the model could learn faster with bigger batch size. The second factor is when the normal batch size does not fill fully the available CUDA cores. Choosing a proper batch size could give benefit over normal training cycle even if it has to use multiple GPU upload and unload process with infinitybatch.

The most of time infinitybatch performs a little slower than a normal train loop, but it is the cost of using bigger batch size. If you want to save time, try to use managed batch selection or other data augmentation that can help the model to converge faster.

Cloud 9

One of the most used argument was that cloud service is better, since there are no need to buy GPUs, providers take very wide resource range for relatively cheap. Cloud service are good. It is cheap when the programmer or developer knows what and how they want to achieve. When the full testing phase is on cloud service, they can easily run out of the budget. We help people to figure it out what they want to do on cloud service.

1 The title of chapter compliments one of the greatest economist, Richard H. Thaler on his work. He used this title in his bestseller book that is called Misbehaving: The Making of Behavioral Economics. As he wrote, talking about his work with his colleagues felt like running a medieval gauntlet. As Thaler replied on every arguments that he had got, we are trying to do some similar.

Big batch blah

Some people says, supporting big batch size training is a silly idea, since most developer does not use big batches. The most popular batch sizes are up to 256. We know that a lot of developer and researcher use relatively low size. Our practice is the common size of batch size is under 1024. However, there are a lot of problem and GPU where developer cannot use 32, since it causes memory overrun. That's why we does not speak about big batch, we prefer the "bigger batch" term. The new world will open with infinitybatch for those people, who cannot use for example 32 batch size due to GPU-RAM limitation.

Unprovable

This statement is false. We provide tool that can help to reproduce our tests. It reveals that our solution results perfectly equals as the normal train loop. There were solutions in forums that promised something similar as infinitybatch, but the concept of each idea was bad. That's why they could not be able to prove or repeat the tests. Our solution is unique and takes what it promises: train with bigger batch size than can fit into a GPU memory and with the perfectly equal result than it would be if the train could run in a normal train loop.

A need that cause mislead

One of the creepiest argument, that if there is not in PyTorch, it is not necessary. This way of thinking is the obstacle of any kind of progress. Yesterday was the problems only, today we give a solution, tomorrow it will be the part of common knowledge.

Interesting fact, that our solutions seems to fit very well not in PyTorch's structure only, but in the way of PyTorch's thinking. Seeing the release notes it is very clean, the world and PyTorch goes on a path that is similar like the vision of our solution.

Import distort

No, there are no distortion in infinitybatch. If you train your model with a normal train loop and infinitybatch with the same batch size, results will be equal. We provide a test tool where anybody can check it.

Unintended extended

We got an interesting objection: it modifies the workflow of PyTorch. Yes, it is right. There are no other option to make a new functionality just with modify the basic framework. However, we like to think about out solution like it does not just modify, but it integrates into the PyTorch's workflow and logic. The last part is very important to make a user friendly and easy-use software.

Countably infinite

One of the most funny comment is: "Your program isn't infinite". Well yes, it has limitations. The top limit of batch size is always the size of training dataset. The name represents only the batch size can be stretched into infinity when the resources are support it. You can add an other layer to the meaning of "infinity". Using the human's imagination, there are infinite ways to use infinitybatch to solve deep learning problems.

Cheap, sweep, weep

This is a very rude argument. It says, people, who cannot buy proper GPU, do not make great businesses. There are a lot of poor programmer and students around the world. Infinitybatch can help them to use their infrastructure better. Furthermore, everybody has the chance to flourish and be a programmer who has a great job. This image fits into every companies' strategy.

Code reference

infinitybatch

class InfinityBatch

```
class InfinityBatch(model: Optional[torch.nn.Module] = None, optimizer: Optional[torch.optim.Optimizer] = None, criterion: Optional[torch.nn.Module] = None, dataloader: Optional[Any] = None, backward_device: Union[torch.device, str, int, None] = None, verbose: bool = True)
```

Provides the whole functionality of infinitybatch.

```
model: Optional[torch.nn.Module] = None
```

PyTorch model that is used for training. The given model is stored as InfinityBatch.model property. This property can be set to other than None at once only. The only requirement against a model is that it should accept the input in batch_first form. At best it can handle different batch sizes as well.

```
optimizer: Optional[torch.optim.Optimizer] = None
```

PyTorch optimizer that is instantiated to the given model. The given optimizer is stored as InfinityBatch.optimizer property. This property can be set to other than None at once only.

```
criterion: Optional[torch.nn.Module] = None
```

PyTorch loss function that is used for training. The given criterion is stored as InfinityBatch.criterion property. This property can be set to other than None at once only.

`dataloader: Optional[Any] = None`

PyTorch alike data loader. The given dataloader is stored as InfinityBatch.dataloader property. Because PyTorch does not demand too much needs against dataloaders InfinityBatch follows the same way too. However InfinityBatch has three essential requirements. The output of the iteration of dataloaders should be a pair of inputs (x) and targets (y) with the same batch size. The first dimension of the output should be the dimension of the batch. The number of further dimensions does not matter. The batch level output of the dataloader, both inputs and targets, should be type of torch.Tensor. They should fit the needs of the model and the criterion respectively.

`backward_device: Union[torch.device, str, int, None] = None`

Optional parameter to set the device of the backward processes. If the parameter is omitted or set to None, InfinityBatch sets the device to “cpu”. The setting of InfinityBatch.backward_device calls a check about the equality of forward and backward devices which can lead to InfinityBatchWarning about having the same device also as forward and backward. This warning does not break the run of the code and it can be managed with the change of the InfinityBatch.backward_device or by calling InfinityBatch.updateforwarddevice() if the model is moved to another device yet. Moving the model to another device with InfinityBatch.changemodeldevice() solves this issue as well.

`verbose: bool = True`

Whether to print progress of training or not. By default it is set to True.

The instantiation of an InfinityBatch object warns if torch.cuda is not available since InfinityBatch is useless without CUDA device.

`InfinityBatch.backward() -> None`

Handles loss.backward() at batch level in a training loop. In case of a totally DIY training loop it can be called from outside to ensure the highest flexibility of InfinityBatch.

This function does not return anything.

This function warns if trying to backward on non existing loss.

`InfinityBatch.backward_device`

Property with type of torch.device, that stores the identifier of the backward device. InfinityBatch sets the device to “cpu” by default at instantiation. The setting of InfinityBatch.backward_device calls a check about the equality of forward and backward devices which can lead to InfinityBatchWarning about having the same device also as forward and backward. This warning does not break the run of the code and it can be managed with the change of the InfinityBatch.backward_device or by calling InfinityBatch.updateforwarddevice() if the model is moved to another device yet. Moving the model to another device with InfinityBatch.changemodeldevice() solves this issue as well.

`InfinityBatch.changemodeldevice(device: Union[torch.device, str, int]) -> None`

Moves the model to the given device. The call of InfinityBatch.changemodeldevice() calls a check about the equality of forward and backward devices which can lead to InfinityBatchWarning about having the same device also as forward and backward. This warning does not break the run of the code and it can be managed with the change of the InfinityBatch.backward_device or by calling InfinityBatch.updateforwarddevice() if the model is moved to another device yet. Moving the model to another device with InfinityBatch.changemodeldevice() solves this issue as well.

device: Union[torch.device, str, int]

Device to move the model to.

This function does not return anything.

`InfinityBatch.criterion`

Property with type of torch.nn.Module or NoneType. It contains a PyTorch loss function that is used for training. This property can be set to other than None at once only.

This attribute warns if trying to change the attribute when already set to other than None.

`InfinityBatch.dataloader`

Property with type of anything that fits the requirements of being used as dataloader. Because PyTorch does not demand too much needs against dataloaders InfinityBatch follows the same way too. However InfinityBatch has three essential requirements. The output of the iteration of dataloaders should be a pair of inputs (x) and targets (y) with the same batch size. The first dimension of the output should be the dimension of the batch. The number of further dimensions does not matter. The batch level output of the dataloader, both inputs and targets, should be type of torch.Tensor. They should fit the needs of the model and the criterion respectively.

`InfinityBatch.epoch() -> None`

Manages an epoch. It can be called from outside to train a solo epoch only.

This function does not return anything.

`InfinityBatch.forward_device`

Read-only property with the type of torch.device that stores the identifier of the forward device. This property is set by changing the InfinityBatch.model property or calling InfinityBatch.changemodeldevice() or InfinityBatch.updateforwarddevice() functions.

`InfinityBatch.inputs`

Container-like property that holds the inputs of the training. This container has the shape: [epoch] [batch] [batch element] [data [...]]. This is a read-only property. It cannot be cleared

alone. All the containers can be cleared with the inner function InfinityBatch.clearcontainers_() only.

`InfinityBatch.loss() -> None`

Handles the calculation of the loss in the training loop and saves batch-level loss values to its container. In case of a totally DIY training loop it can be called from outside to ensure the highest flexibility of InfinityBatch.

This function does not return anything.

`InfinityBatch.losses`

Container-like property that holds the loss.item() values of the training at batch level. This container has the shape: [epoch] [batch] (loss.item(), size_of_the_batch)]. This is a read-only property. It cannot be cleared alone. All the containers can be cleared with the inner function InfinityBatch.clearcontainers_() only.

`InfinityBatch.model`

Property with type of torch.nn.Module or NoneType. It contains a PyTorch model that is used for training. This property can be set to other than None at once only. The only requirement against a model is that it should accept the input in batch_first form. At best it can handle different batch sizes as well.

This attribute warns if trying to change the attribute when already set to other than None.

`InfinityBatch.optimizer`

Property with type of torch.optim.Optimizer or NoneType. It contains a PyTorch optimizer that is used for training. This property can be set to other than None at once only.

This attribute warns if trying to change the attribute when already set to other than None.

`InfinityBatch.outputs`

Container-like property that holds the model outputs of the training. This container has the shape: [epoch] [batch] [batch element] [data [...]]. This is a read-only property. It cannot be cleared alone. All the containers can be cleared with the inner function InfinityBatch.clearcontainers_() only.

`InfinityBatch.step() -> None`

Handles the step of the optimizer in the training loop. In case of a totally DIY training loop it can be called from outside to ensure the highest flexibility of InfinityBatch.

This function does not return anything.

`InfinityBatch.targets`

Container-like property that holds the targets of the training. This container has the shape: [epoch] [batch] [batch element] [data [...]]. This is a read-only property. It cannot be cleared alone. All the containers can be cleared with the inner function InfinityBatch.clearcontainers_() only.

`InfinityBatch.train(num_epochs: int) -> None`

Handles a training. This is the default use-case of InfinityBatch.

num_epochs: int

The number of epochs to train for.

This function does not return anything.

`InfinityBatch.updateforwarddevice() -> None`

Updates forward device from the model parameters. It is called automatically when a model is added to the InfinityBatch object or when InfinityBatch.changemodeldevice() is

called. Call of this function is reasonable when something occurred with the device location of the model outside of the scope of InfinityBatch.

This function does not return anything.

`InfinityBatch.verbose_mode`

Boolean property that stores the verbosity state of the InfinityBatch object.

`InfinityBatch.zero_grad() -> None`

Handles the zero_grad of the optimizer and of the model respectively in the training loop. In case of a totally DIY training loop it can be called from outside to ensure the highest flexibility of InfinityBatch.

This function does not return anything.

`InfinityBatch.batch_(inputs: torch.Tensor) -> torch.Tensor`

Handles the forwarding of a batch. This is an inner function. Call it from outside only if you make sure you need to call this function anyway and you fully understand the effect of the call of this function.

inputs: torch.Tensor

Inputs for the model to forward.

This function returns the output of the model respectively to the given batch in the form of torch.Tensor.

`InfinityBatch.checkdevices_() -> None`

Checks the forward and backward devices and warns if they are the same. This is an inner function. Call it from outside only if you make sure you need to call this function anyway and you fully understand the effect of the call of this function.

This function does not return anything.

`InfinityBatch.clearcontainers_() -> None`

Clears each container. This is an inner function. Call it from outside only if you make sure you need to call this function anyway and you fully understand the effect of the call of this function.

This function does not return anything.

`InfinityBatch.epochstart_() -> None`

Adds new entries to each container to be used in the epoch that begins. This is an inner function. Call it from outside only if you make sure you need to call this function anyway and you fully understand the effect of the call of this function.

This function does not return anything.

`InfinityBatch.__call__(num_epochs: int) -> None`

Provides direct instance level call. This is an alias of InfinityBatch.train() function. PyTorch models have the ability to be called at instance level for forwarding; this function adds a similar feature to InfinityBatch. InfinityBatch's main focus is to provide a whole training manager and this function provides it.

num_epochs: int

The number of epochs to train for.

This function does not return anything.

class InfinityBatchError()

Used when an error occurs. It is an empty subclass of Exception. At the moment it is not used yet.

class InfinityBatchWarning()

Used to warn the user about important things. It is an empty subclass of Warning.

tools

classes

class CudaMemorySnapShot

`class CudaMemorySnapShot(device_id: Union[torch.device, str, int, None] = None, synchronize: bool = True)`

Provides a simple read-only container for memory snapshot of a torch.cuda device.

device_id: Union[torch.device, str, int, None] = None

The device where the snapshot should be taken. If the device is not given it is set to None. The effect of the setting of this parameter fully accords to the behavior of the concerning PyTorch functions.

`synchronize: bool = True`

Whether or not to apply torch.cuda.synchronize() at the given device before taking the snapshot.

`CudaMemorySnapShot.allocated`

Property with type of int. It contains the size of the allocated memory in bytes at the moment when the snapshot was taken. In fact this property is read-only since it can be set at once only, and it is set at instantiation. Due to the experimental character of this snapshot, trying to give a new value does not raise error however it won't be successful.

`CudaMemorySnapShot.device`

Property with type of torch.device, str, int or NoneType. It contains the device of the snapshot without converting to torch.device. In fact this property is read-only since it can be set at once only, and it is set at instantiation. Due to the experimental character of this snapshot, trying to give a new value does not raise error however it won't be successful.

`CudaMemorySnapShot.free_cache`

Property with type of int. This is a calculated value of free cache size in bytes at the moment when the snapshot was taken. Since this property is calculated it is true read-only property.

`CudaMemorySnapShot.reserved`

Property with type of int. It contains the size of the reserved memory in bytes at the moment when the snapshot was taken. In fact this property is read-only since it can be set at once only, and it is set at instantiation. Due to the experimental character of this snapshot, trying to give a new value does not raise error however it won't be successful.

`CudaMemorySnapShot.total`

Property with type of int. It contains the size of the total available device memory in bytes at the moment when the snapshot was taken. In fact this property is read-only since it can be set at once only, and it is set at instantiation. Due to the experimental character of this snapshot, trying to give a new value does not raise error however it won't be successful.

class CudaHistory

`class CudaHistory()`

Provides a simple container for CudaMemorySnapShot objects with labels. Entries of this object are tuples in the form of (label, CudaMemorySnapShot).

This is a subclass of list but it cannot be instantiated with addition of existing elements like lists can be.

`CudaHistory.append(name: str, device_id: Union[torch.device, str, int, None] = None, synchronize: bool = True) -> None`

Appends items to the container. Every add-in leads to instantiation of a new CudaMemorySnapShot object.

name: str

The label of the new CudaMemorySnapShot entry.

device_id: Union[torch.device, str, int, None] = None

The device where a new CudaMemorySnapShot should be taken. If the device is not given it is set to None. The effect of the setting of this parameter fully accords to the behavior of the concerning PyTorch functions.

synchronize: bool = True

Whether or not to apply torch.cuda.synchronize() at the given device before taking the snapshot.

This function does not return anything.

`CudaHistory.__str__() -> str`

Provides human readable representation of the CudaHistory object.

This function returns a string.

functions

determinize()

```
determinize(seed: Optional[int] = None, be_deterministic: bool = True) -> None
```

Seeds more random sources and cares about the environment to be or don't be deterministic.

seed: Optional[int] = None,

The number to use as seed for manual seeding. Setting this number to a constant value highly increases the chance of the reproducibility of a training. On the contrary leaving this number empty highly increases the randomness of each training individually.

be_deterministic: bool = True

A boolean that specifies whether to switch torch.backends.cudnn into deterministic mode or not.

This function imports random from the standard library and tries to import NumPy however it won't fail if NumPy is not installed.

This function does not return anything.

is_equal_network()

```
is_equal_network(one: torch.nn.Module, another: torch.nn.Module) -> bool
```

Compares two models based on their state_dicts and returns whether their weights and biases are exactly equal or not. The bases of the comparison are the same layers and same sizes as well.

one: torch.nn.Module

The one model to compare.

another: torch.nn.Module

The other model to compare.

This function returns a bool whether the two networks are the same by weights and biases or not.

is_equal_with_error()

```
is_equal_with_error(one: torch.Tensor, another: torch.Tensor, detach: bool=True) -> bool
```

Compares two tensors if they have values in the given calculation error tolerance or not. The base of the comparison is the same dimensionality.

one: torch.Tensor,

One tensor to compare.

another: torch.Tensor,

Another tensor to compare.

detach: bool=True

Whether to detach tensors before comparison or not. This settings effects on tensors with requires_grad = True only.

This function returns a bool whether the two tensors are the same respectively to the calculation error tolerance or not.

compare_with_error_()

compare_with_error_(one: float, another: float) -> bool

Compares two floats whether they are in the range of the calculation error tolerance or not.

one: float

One float to compare.

another: float

Another float to compare.

This function returns the result of the comparison.

constants

ERROR_TOLERANCE

`ERROR_TOLERANCE = 1e-7`

The range of the tolerance of calculation errors in comparison-functions of the tools.

Statistics

This section contains relevant or interesting statistics. Every table or chart is linked to the main content of book.

Video card popularity

Video cards ratio between Steam gamers in may and june:

VIDEO CARDS	MAY	JUN
NVIDIA GeForce GTX 1060	11,8	11,73
NVIDIA GeForce GTX 1050 Ti	8,22	8,34
NVIDIA GeForce GTX 1050	5,31	5,31
NVIDIA GeForce GTX 1070	4,13	4,05
NVIDIA GeForce GTX 1650	2,6	2,99
NVIDIA GeForce GTX 1660 Ti	2,51	2,71
NVIDIA GeForce RTX 2060	2,38	2,53
NVIDIA GeForce GTX 1080	2,52	2,48
AMD Radeon RX 580	2,03	2,06
NVIDIA GeForce GTX 970	2,09	2,04
NVIDIA GeForce RTX 2070	1,81	1,85
NVIDIA GeForce GTX 1660	1,51	1,64
NVIDIA GeForce GTX 960	1,71	1,62
NVIDIA GeForce RTX 2070 SUPER	1,4	1,6
NVIDIA GeForce GTX 1080 Ti	1,6	1,58

VIDEO CARDS	MAY	JUN
NVIDIA GeForce GTX 750 Ti	1,48	1,51
AMD Radeon RX 570	1,26	1,34
NVIDIA GeForce GTX 960M	1,19	1,12
Intel UHD Graphics 620	1,18	1,11
NVIDIA GeForce GTX 1070 Ti	1,12	1,1
NVIDIA GeForce RTX 2080	0,99	0,98
NVIDIA GeForce RTX 2060 SUPER	0,81	0,97
NVIDIA GeForce 940M	1,01	0,95
NVIDIA GeForce GTX 1660 SUPER	0,71	0,94
Intel HD Graphics 620	0,98	0,91
NVIDIA GeForce RTX 2080 Ti	0,83	0,89
Intel HD Graphics 4000	0,86	0,75
AMD Radeon RX 5700 XT	0,67	0,74
Intel Haswell	0,74	0,68
AMD Radeon Vega 8 Graphics	0,6	0,68

Source: Steam statistic [25.07.2020.]

VRAM on Steam

GPU RAM ratio between Steam gamers in june:

VRAM	JUN
512 MB	1,52
1024 MB	11,94
2047 MB	13,79
3071 MB	5,46
4095 MB	17,46
6143 MB	20,79
8191 MB	21,19
11263 MB	4,14
Other	3,72

Source: Steam statistic [25.07.2020.]

System RAM on Steam

CPU RAM ratio between Steam gamers in june:

System RAM	JUN
Less than 2 GB	0,08
2 GB	0,49
3 GB	1,82
4 GB	6,67
5 GB	0,34
6 GB	2,41
7 GB	2,38
8 GB	33,44
9 GB	0,03
10 GB	0,32
11 GB	0,22
12 GB	3,14
13 GB	0,02
14 GB	0,33
15 GB	0,59
16 GB	39,53
More than 16 GB	8,18

Source: Steam statistic [25.07.2020.]

PyTorch on Github

The table shows how many times are mentioned pytorch in different contexts on GitHub.

Search	Repositories	Code	Commits	Issues	Issues(closed)	Issues (open)	Discussions	Packages	Marketplace	Topics	Wikis	Users
pytorch	47580	1353638	370071	147747	115695	32055	4	6	0	353	1893	378
pytorch batch	144	529941	6562	19944	14196	5748	1	0	0	0	297	1
pytorch batch size	7	478985	2000	14153	9862	4291	0	0	0	0	185	0
pytorch memory error	0	91454	1134	7109	5067	2038	0	0	0	0	87	0
pytorch cuda out of memory	0	60047	346	4314	3007	1307	0	0	0	0	87	0
pytorch GPU memory	10	97783	1062	7159	4922	2237	1	0	0	0	185	0
pytorch GPU memory error	0	51819	331	4594	3156	1438	0	0	0	0	66	0
pytorch batch memory error	0	61125	169	3333	2325	1008	0	0	0	0	47	0
pytorch GPU batch	8	203337	673	8840	6112	2728	1	0	0	0	171	0

Source: Github's search engine [30.07.2020.]

Note: Results are only approximation since every search returns with slightly different value.

Open source licenses on Github

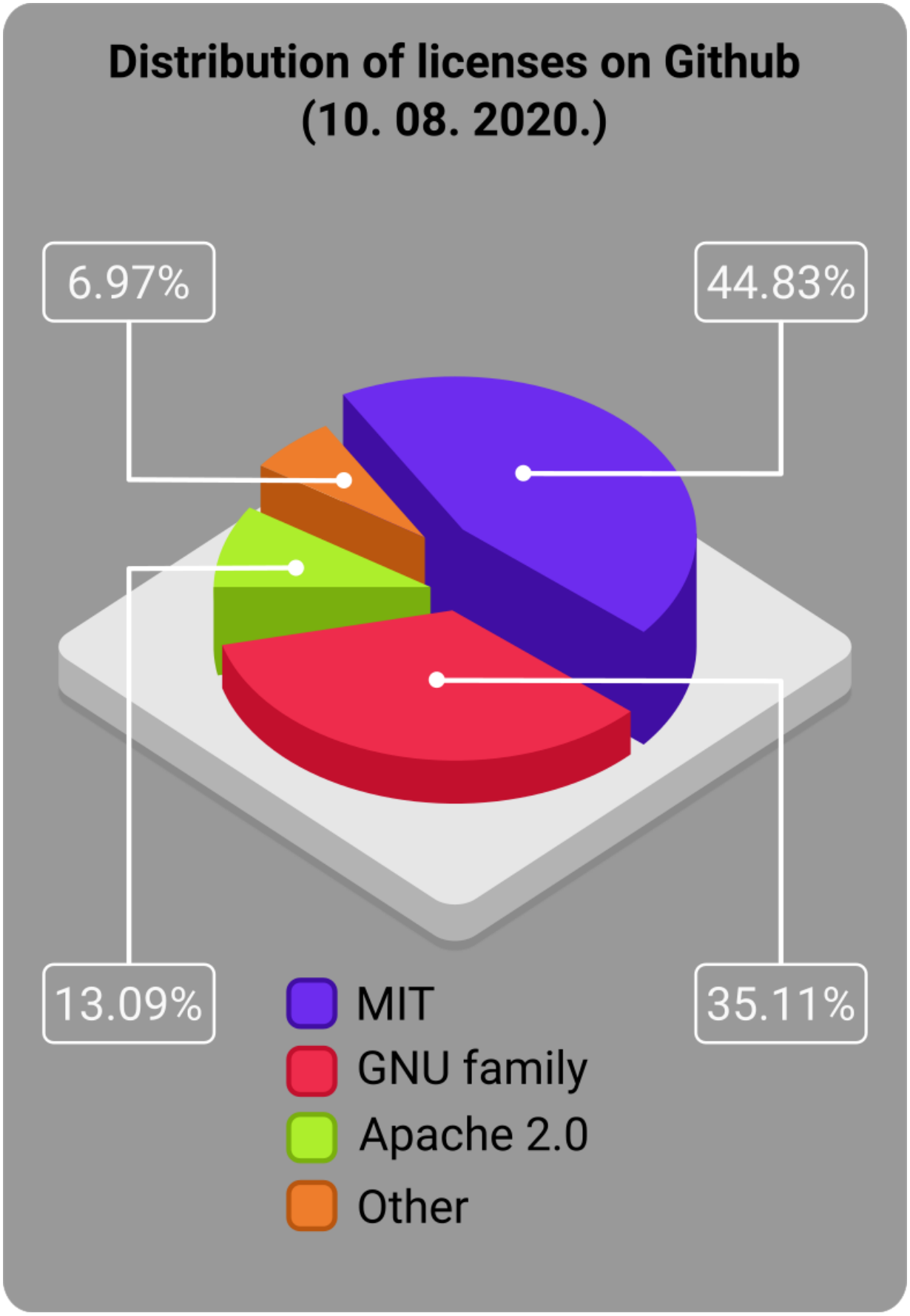
These licenses are available for repository licensing on GitHub ordered by frequently.

License	Count
MIT	4 800 806
GNU General Public License family	1 718 287
Apache license 2.0	1 401 561
GNU General Public License v3.0	1 301 332
GNU General Public License v2.0	405 052
BSD 3-clause "New" or "Revised" license	187 291
The Unlicense	145 469
GNU Lesser General Public License family	117 141
GNU Affero General Public License v3.0	100 767
GNU Lesser General Public License v3.0	87 681
BSD 2-clause "Simplified" license	80 097
Creative Commons license family	79 785
Creative Commons Zero v1.0 Universal	60 043
Mozilla Public License 2.0	53 758
Eclipse Public License 1.0	38 089
ISC	35 216
GNU Lesser General Public License v2.1	29 460

License	Count
Do What The F*ck You Want To Public License	18 032
Creative Commons Attribution Share Alike 4.0	10 024
Artistic license 2.0	10 017
Creative Commons Attribution 4.0	9 718
Boost Software License 1.0	4 302
zLib License	4 113
SIL Open Font License 1.1	3 747
Open Software License 3.0	2 132
Microsoft Public License	1 606
Academic Free License v3.0	893
European Union Public License 1.1	703
LaTeX Project Public License v1.3c	669
BSD 3-clause Clear license	598
Educational Community License v2.0	374
PostgreSQL License	98
University of Illinois/NCSA Open Source License	60

Source: Github's search engine with own customized search terms "license:"[10.08.2020.]

Note: Results are only approximation since every search returns with slightly different value.


Distribution of licenses on Github
(10. 08. 2020.)
6.97%
44.83%
13.09%
35.11%
MIT
GNU family
Apache 2.0
Other

Tesla index

The idea of Tesla index comes from The Economist's Big Mac Index. That index focuses on the purchasing power parity and national market and goods, but programmers are living in a globalized world where every items can be purchased from the global market. Tesla Index compares the local salaries through the most expensive Tesla card that is available at the moment. Data of salary comes from PayScale, data of Tesla card's price comes from Amazon, data of currencies exchange rate comes from Google finance.

Source of data:

Salary: PayScale

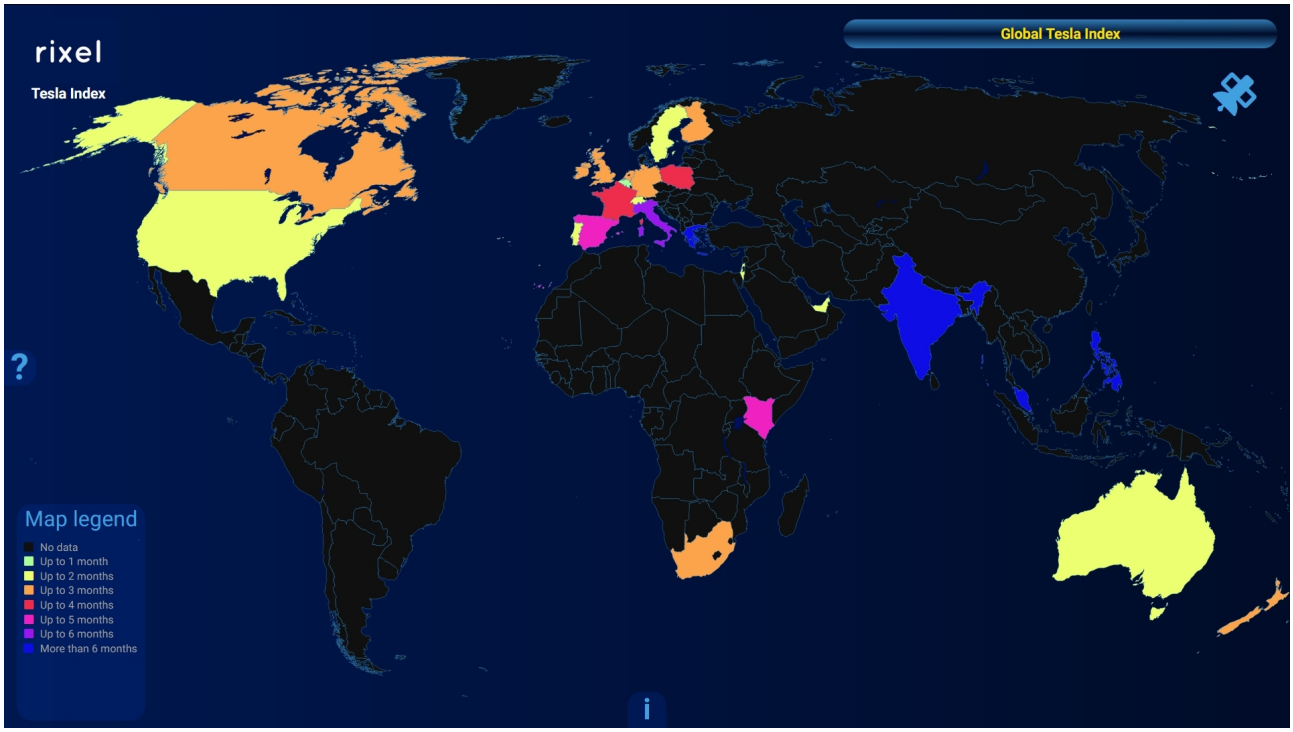
Tesla card: Amazon

Currencies exchange rate: Google finance

Chosen card: NVIDIA Tesla V100 Volta GPU Accelerator 32GB Graphics Card by NvidiaCorporation

Price of card: 8,209 USD

Date of data checking: 21. 08. 2020



Country	Median salary in local money	Dollar exchange rate	Median salary in USD	Hours to Tesla	Months to Tesla
Australia	40,4100	0,7200	29,0952	283	2
Belgium	71,5000	1,1800	84,3700	98	1
Canada	33,5300	0,7600	25,4828	323	3
Finland	15,2600	1,1800	18,0068	456	3
France	14,0000	1,1800	16,5200	497	4
Germany	15,0000	1,1800	17,7000	464	3
Greece	7,0000	1,0000	7,0000	1173	8
India	494,3500	0,0130	6,4266	1278	8
Ireland	17,5000	1,1800	20,6500	398	3
Israel	139,2000	0,2900	40,3680	204	2
Italy	7,6800	1,1800	9,0624	906	6
Kenya	1 106,2500	0,0093	10,2881	798	5
Malaysia	25,3100	0,2400	6,0744	1352	9
Netherlands	18,0800	1,1800	21,3344	385	3
New Zealand	34,2400	0,6600	22,5984	364	3
Philippines	375,0000	0,0210	7,8750	1043	7
Poland	50,5000	0,2700	13,6350	603	4
Portugal	29,0000	1,1800	34,2200	240	2
Singapore	25,0000	0,7300	18,2500	450	3
South Africa	422,6800	0,0580	24,5154	335	3
Spain	10,5000	1,1800	12,3900	663	5
Sweden	295,0000	0,1100	32,4500	253	2
Switzerland	36,0000	1,1000	39,6000	208	2
United Arab Emirates	157,5000	0,2700	42,5250	194	2
United Kingdom	15,1300	1,3100	19,8203	415	3
United States	35,4600	1,0000	35,4600	232	2

Unique and special solutions in infinitybatch

There are two main concepts in infitybatch that can be considered as unique or special. One conception is the way how we treat the output of the forward pass of a model and the other conception is to separate forward-pass and backpropagation devices.

The output of a model should be treated carefully since it contains a computational graph which is essential to be preserved for the backpropagation process at least till the calculation of the loss. PyTorch doesn't let developers to access autograd directly on the purpose to ensure its proper workflow at any circumstances. This limit slightly encumbers to find working solutions for a couple of problems like this which is solved by infinitybatch now.

The ability to separate forward and backward devices hardly hangs together with the ability to preserve computational graphs. Graphs are essential for the loss to backpropagate. If the graph is broken for any reason the backpropagation will lead to bad results. This is the reason why common solutions for increasing batch size on GPU RAM don't work.

Infinitybatch clones the output of the model with its computational graph and this clone is moved away from the device of the model to the device with more memory – usually CPU RAM – where the backpropagation will occur. As the whole batch is forwarded the outputs get stacked and together with targets the loss is calculated. The computation of the loss happens on the backward device however it affects the data on the forward device as well. All of the rest of training is just like any other PyTorch model training and this means real freedom for a developer who tries out infinitybatch.

Future plans

The future improvement plans below are ordered by their importance and schedule.

eval()

However making evaluation with infinitybatch is very easy for user who like to have “canonical way” for everything we plan to implement .eval() function separately. The hardest task of this implementation is to relate to dataloaders because of the freedom use that we can provide at this point.

PyPi

Since infinitybatch is a production ready module it could be right now loaded into PyPi.org to be usable as a Python package globally. We plan to wait till the outcome of this hackathon gets clear and then we will upload this package together with a ReadTheDocs documentation and with a lot of examples and tests as well.

Event monitor

To support do-it-yourself train loops even more we plan add an event monitor to infinitybatch. This feature could warn the user if they leaves out something important from the forward-backward process.

Improvement of warnings

A good and highly configurable warning system is needed for infinitybatch in the future to be able to serve experimenters and real experts at the same time. We plan to add warning plans and warning levels and more safety functions.

Improvement of the containers

The containers of infinitybatch are very simple constructions at the moment. In the future we plan to add sophisticated storage strategies and container-related functions too.

Counter of learnable parameters

We plan to add sophisticated counter function to make a summary about the characteristics of learnable parameters of a model. We plan to show a detailed overview across the whole model including the number of elements and their aspects to memory and speed as well. This feature is far away from a level where pure Python could be enough.

Memory usage monitor

To be able to have a better overview of what happens in the memory we have to dig much deeper therefore we plan to make a detailed memory usage monitor. To develop this we have to leave the level of Python but we plan to build something that is usable in the pure Python level as well.

Improving memory usage

If we have a better overview of what happens in the memory why not to improve the usage of the memory. At least the precise and up-to-date calculation of active and inactive memory blocks would be very welcome and the lifecycle of temporary variables could be also examined.

Improving UI

At the moment our user interface is far away from perfect but at least we have a verbose property to decide about the appearance of very simple prints during the training. In the future we plan to have more configurable training prints. We plan to offer saving stats into file and we plan to create some functionality to be able to connect with graphical interfaces.

Change of model

We plan to add observer to the model, optimizer and criterion properties of infinitybatch to be able to monitor important changes of these attributes without the need to implement a wrapper of the PyTorch method. This ways the use of infinitybatch would be much more convenient.

“In case if needed” plans

The improvements below are a question of need. Because of different reasons we don’t consider those ones that much important right now but we are open and curious about the real-world needs of the users.

Callbacks

Based on the first experiences we plan to add the ability to place callbacks in the worklfow of infinitybatch. We think the ability of calling forward-backward stages separately should be enough to build very different train loops but there could be draw up some real needs.

Improvement of tools.CudaHistory

Classes tools.CudaHistory and tools.CudaMemorySnapshot began their lives like experimental classes. The importance of the management of snapshots like memory state at a given moment or anything else is obvious therefore we plan improve history based containers.

Complexity

We plan to add the ability of using more models, criterions or optimizers together. This leads to much more complex epochs and functionality. Though it would be nice to have that level of functionality yet tomorrow it seems it's not that easy. To give all the needed energy for this improvement we need to have a better view to the real-world use-cases of our users.

Sources

This chapter contains all sources that are written, mentioned or cited in the book. In the link section every source has a title, a website link, the author name, brand name or the webpage name and the date when the content was checked last time.

Papers

Scientific sources, like paper, book or article are, listed below in the order of appearance.

[Paper 01]
title: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima
authors: Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, Ping Tak Peter Tang
arXiv: 1609.04836v2

[Paper 02]
title: Deep Learning
authors: Ian Goodfellow and Yoshua Bengio and Aaron Courville
publisher: MIT Press
year: 2016
links: book, cited chapter

[Paper 03]
title: Revisiting Small Batch Training for Deep Neural Networks
authors: Dominic Masters, Carlo Luschi
arXiv: 1804.07612v1

[Paper 04]
title: Don't Decay the Learning Rate, Increase the Batch Size
authors: Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, Quoc V. Le
arXiv: 1711.00489v2

[Paper 05]
title: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour
authors: Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, Kaiming He
arXiv: 1706.02677v2

[Paper 06]

title: Effect of batch size on training dynamics
authors: Kevin Shen
date: 19. 06. 2018.
link: medium.com

[Paper 07]
title: Control Batch Size and Learning Rate to Generalize Well: Theoretical and Empirical Evidence
authors: Fengxiang He, Tongliang Liu, Dacheng Tao
link: Neural Information Processing Systems (NIPS)

[Paper 08]
title: Train longer, generalize better: closing the generalization gap in large batch training of neural networks
authors: Elad Hoffer, Itay Hubara, Daniel Soudry
arXiv: 1705.08741v2

[Paper 09]
title: Scaling SGD Batch Size to 32K for ImageNet Training
authors: Yang You, Igor Gitman, Boris Ginsburg
link: Berkeley

[Paper 10]
title: Large batch size training of neural networks with adversarial training and second-order information
authors: Zhewei Yao, Amir Gholami, Daiyaan Arfeen, Richard Liaw, Joseph Gonzalez, Kurt Keutzer, Michael Mahoney
arXiv: 1810.01021v3

[Paper 11]
title: Sharp Minima Can Generalize For Deep Nets
authors: Laurent Dinh, Razvan Pascanu, Samy Bengio, Yoshua Bengio
arXiv: 1703.04933v2

[Paper 12]
title: AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks
authors: Aditya Devarakonda, Maxim Naumov, Michael Garland
arXiv: 1712.02029v2

[Paper 13]
title: Epochs, Batch Size, & Iterations
authors: AI Wiki
link: Paperspace

[Paper 14]
title: An Empirical Model of Large-Batch Training
authors: Sam McCandlish, Jared Kaplan, Dario Amodei, OpenAI Dota Team
arXiv: 1812.06162v1

[Paper 15]
title: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
authors: Sergey Ioffe, Christian Szegedy
arXiv: 1502.03167v3

Links

AWS: Amazon EC2 Instance Types [05.08.2020]:
https://aws.amazon.com/ec2/instance-types/

Exoscale: Pricing [24.08.2020]:
https://www.exoscale.com/pricing/

Facebook: Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour [24.08.2020]:
https://research.fb.com/wp-content/uploads/2017/06/imagenet1kin1h5.pdf

Google Colab Pro: FAQ [24.08.2020]:
https://colab.research.google.com/signup

Google Publications: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [24.08.2020]:
https://research.google/pubs/pub43442/

ImageNet: ImageNet main page [24.08.2020]:
http://www.image-net.org/

Intel: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima [20.08.2020]:
https://www.intel.com/content/www/us/en/artificial-intelligence/researches/large-batch-training-deep-learning-generalization-gap-sharp-minima.html

Lambda: Lambda GPU Cloud for Deep Learning [24.08.2020]:
https://lambdalabs.com/service/gpu-cloud

Matuszek, David: Java Syntax [30.07.2020]:
https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/errors.html

Nvidia Publications: AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks [24.08.2020]:
https://research.nvidia.com/publication/2017-12_AdaBatch%3A-Adaptive-Batch

Nvidia: Tesla V 100 [21.08.2020]:
https://www.nvidia.com/en-gb/data-center/tesla-v100/

Udacity: Intro to Deep Learning with PyTorch [25.07.2020]:
https://www.udacity.com/course/deep-learning-pytorch--ud188

Udacity: Secure and Private AI [25.07.2020]:
https://www.udacity.com/course/secure-and-private-ai--ud185

Programiz: Python Errors and Built-in Exceptions [31.07.2020]:
https://www.programiz.com/python-programming/exceptions

Python: 8. Errors and Exceptions [31.07.2020]:
https://docs.python.org/3/tutorial/errors.html

PyTorch's Github channel: PyTorch license [07.08.2020]:
https://github.com/pytorch/pytorch/blob/master/LICENSE

PyTorch's documentation: batch [16.08.2020]:
https://pytorch.org/text/_modules/torchtext/data/batch.html

The Economist's Github channel: big mac data [24.08.2020]:
https://github.com/TheEconomist/big-mac-data

Wikipedia: Misbehaving: The Making of Behavioral Economics [19.08.2020]:
https://en.wikipedia.org/wiki/Misbehaving:_The_Making_of_Behavioral_Economics

About us

Axel ORSZÁG-KRISZ Dr.

https://hyperrixel.com/en/axel

Richárd Ádám VÉCSEY Dr.

https://hyperrixel.com/en/richard

We are two deep learning developers and data scientists. We work a lot of different field of AI developing, but our main objective is a computer vision in healthcare. Nowadays, most companies like to hire developers who are outsiders, who have other skills and education. We are lawyers who have jumped from law to programming and deep learning. We had a lo of opportunity from Facebook and Udacity. These companies maintain scholarship programs to get knowledge people around the world. We were two of them. Computer Vision Nanodegree or Deep Learning Nanodegree were supported by Facebook. Besides the knowledge, we got a lot of new viewpoint and the possibility to help others. We are thankful and during the COVID-19 epidemic, we are trying to get back something to the community and to the people around the Globe. Under these sign, we created the following projects.

cheXrad

CheXrad is pulmonary application with 4 different artificial intelligence models. It can be trained very fast with CoCoLo technology. The software's goal is to support doctors' decisions with analyzing X-ray images.

Each artificial intelligence network is trained separately so their knowledge can be specialized or generalized as well. The neural network analyze the content of an image independently from each other too. The result can be a simple decision about the X-ray picture: no findings – not healthy. In case of illness the concrete disease is also communicated.

The four separated neural networks work like healthcare counselor group of several specialists which only purpose is to support the doctor. It gives not only a second opinion but also third and fourth opinion of the actual case. The result of each network will be evaluated and visualized after the decision process. Multi-level hierarchy is the key so the computer could decide even if it did not see the actual problem during the training process.

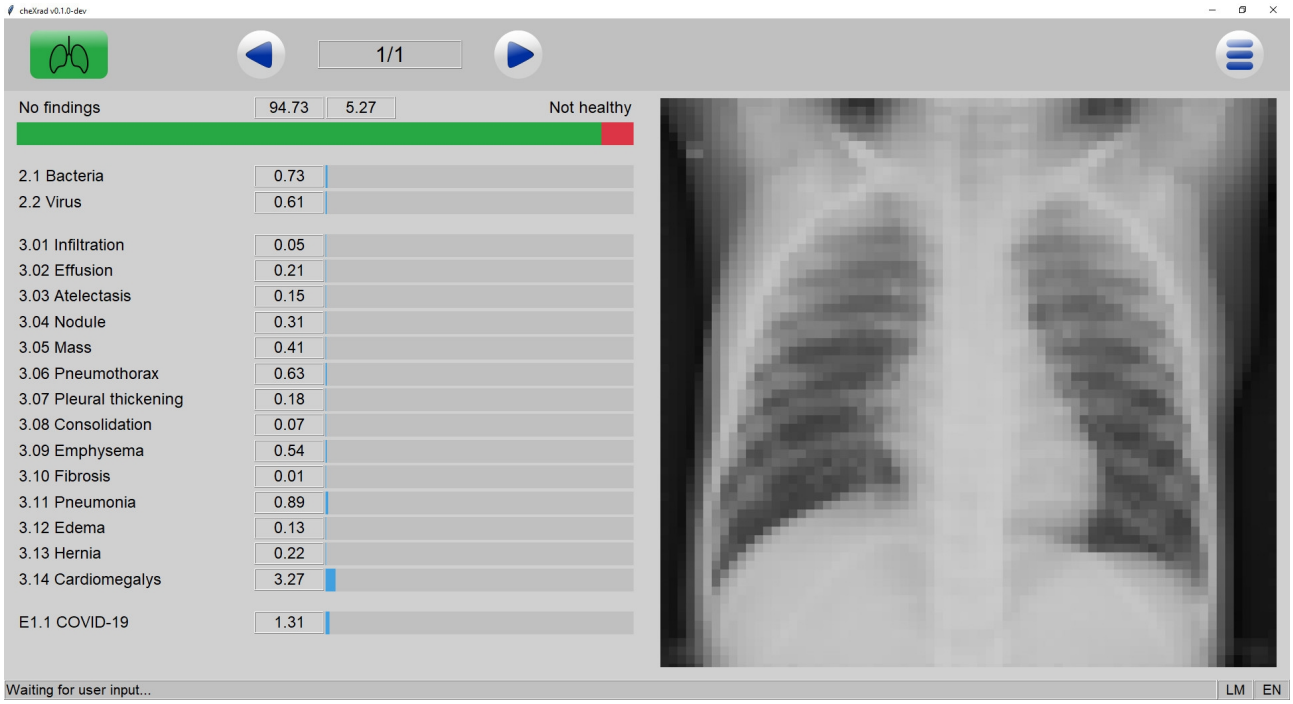
Repository is available here:

https://github.com/hyperrixel/cheXrad

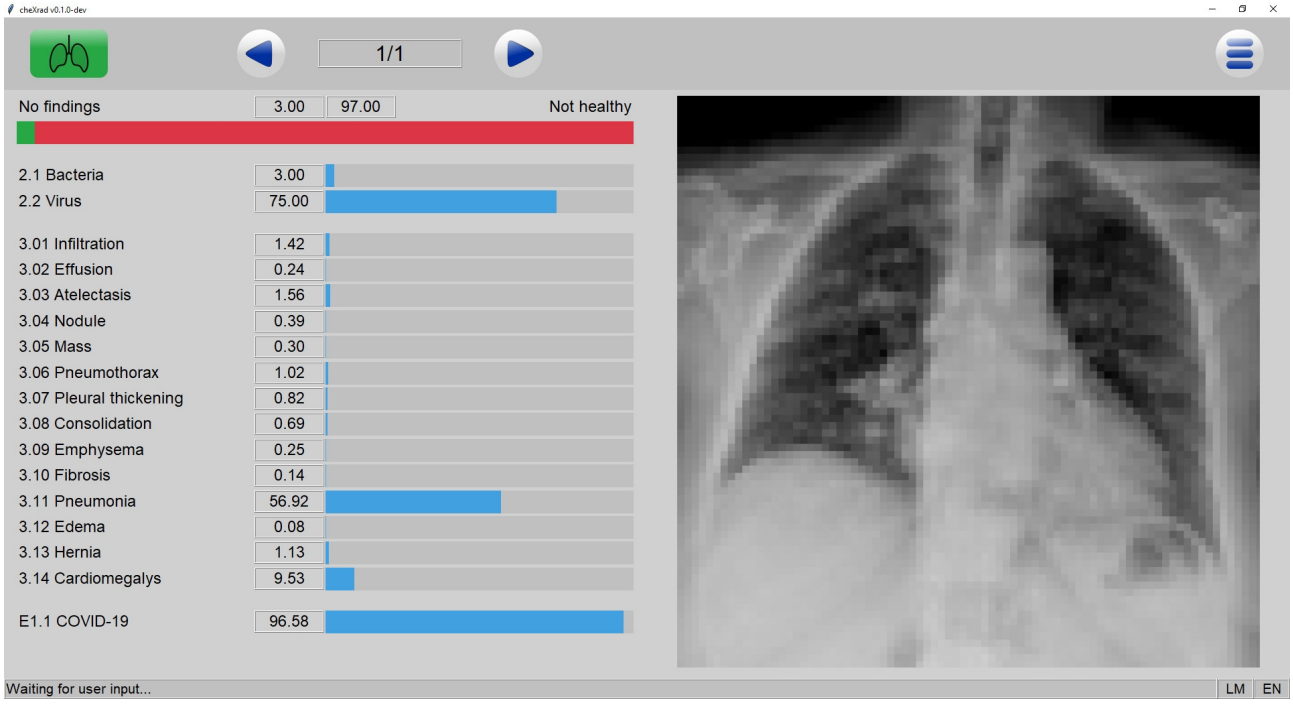
Screenshots:

X-ray images on screenshots are pixelized due to regulation about usage of health data. Images on "not healthy" pictures is made from a patient in COVID-19 disease according to https://github.com/ieee8023/covid-chestxray-dataset GitHub repository. There is two available visual format, light and dark theme.

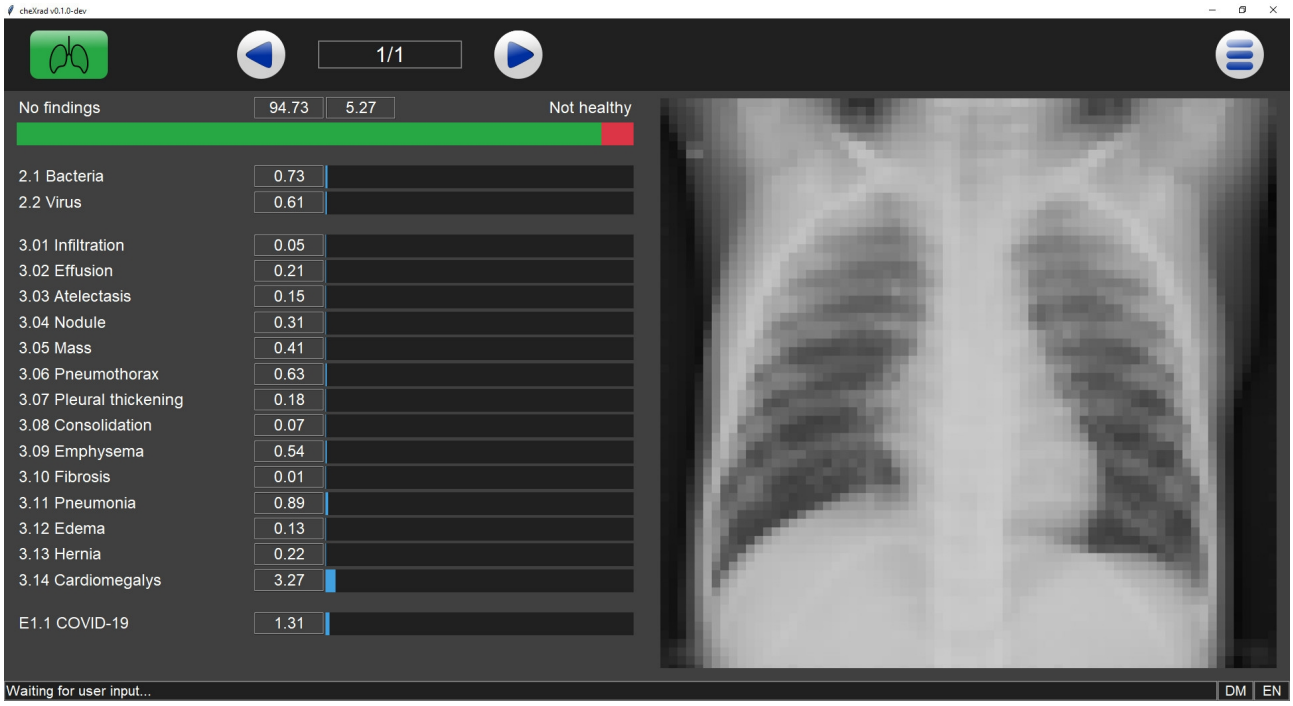
No findings in light theme



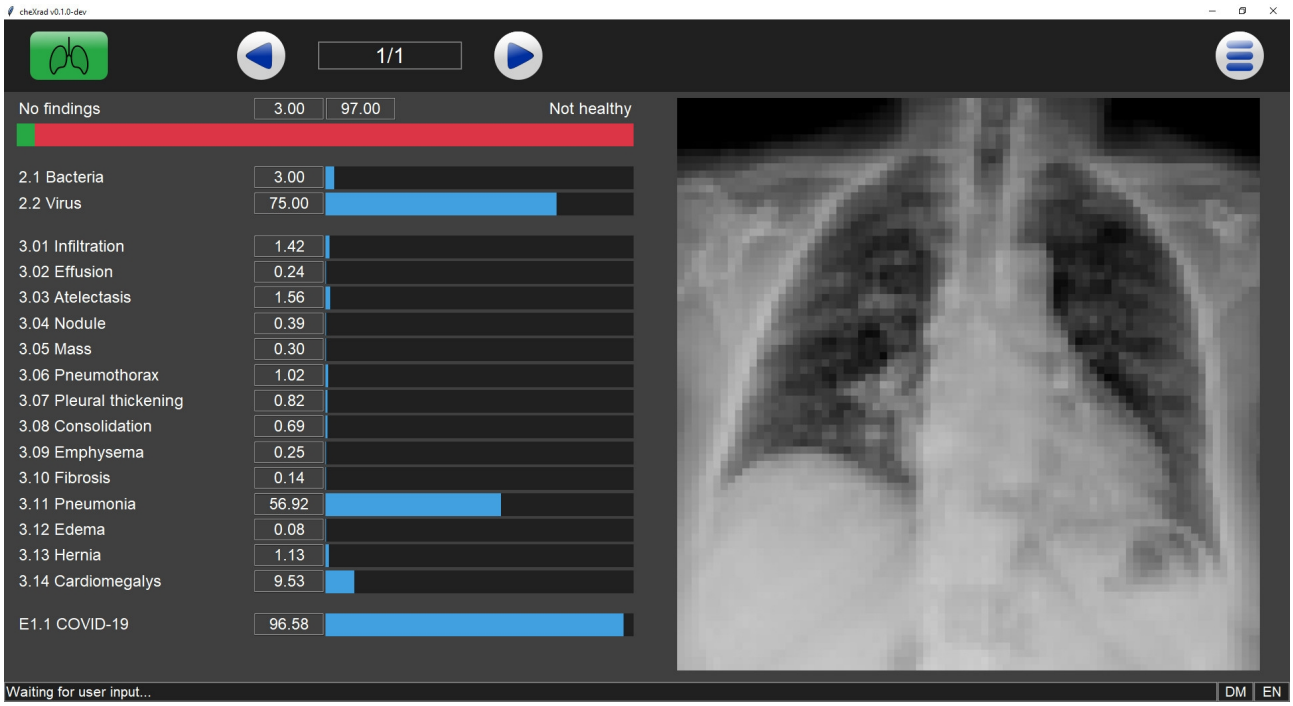
Not healthy in light theme



No findings in dark theme



Not healthy in dark theme



DOF – Deep Model Core Output Framework

DoF is a highly scalable dataset format which helps deep learning scientists to work with foreign or sensitive data. DoF provides fast dataset sharing and data-secure at the same time.

It is fast because during training process it's enough to run forward the dataset through the pretrained core models only once. It's not necessary to calculate these values again at the next epoch. It saves computational time. These values will be always the same because the core part of the model is no more trained.

It is secure since headless deep learning networks anonymize the unique element of the original dataset. The original data cannot be restored from the saved result, layers make reversibility proof destructions.

The DoF framework is not just a container file. It is away of thinking and sharing datasets to help each others work.

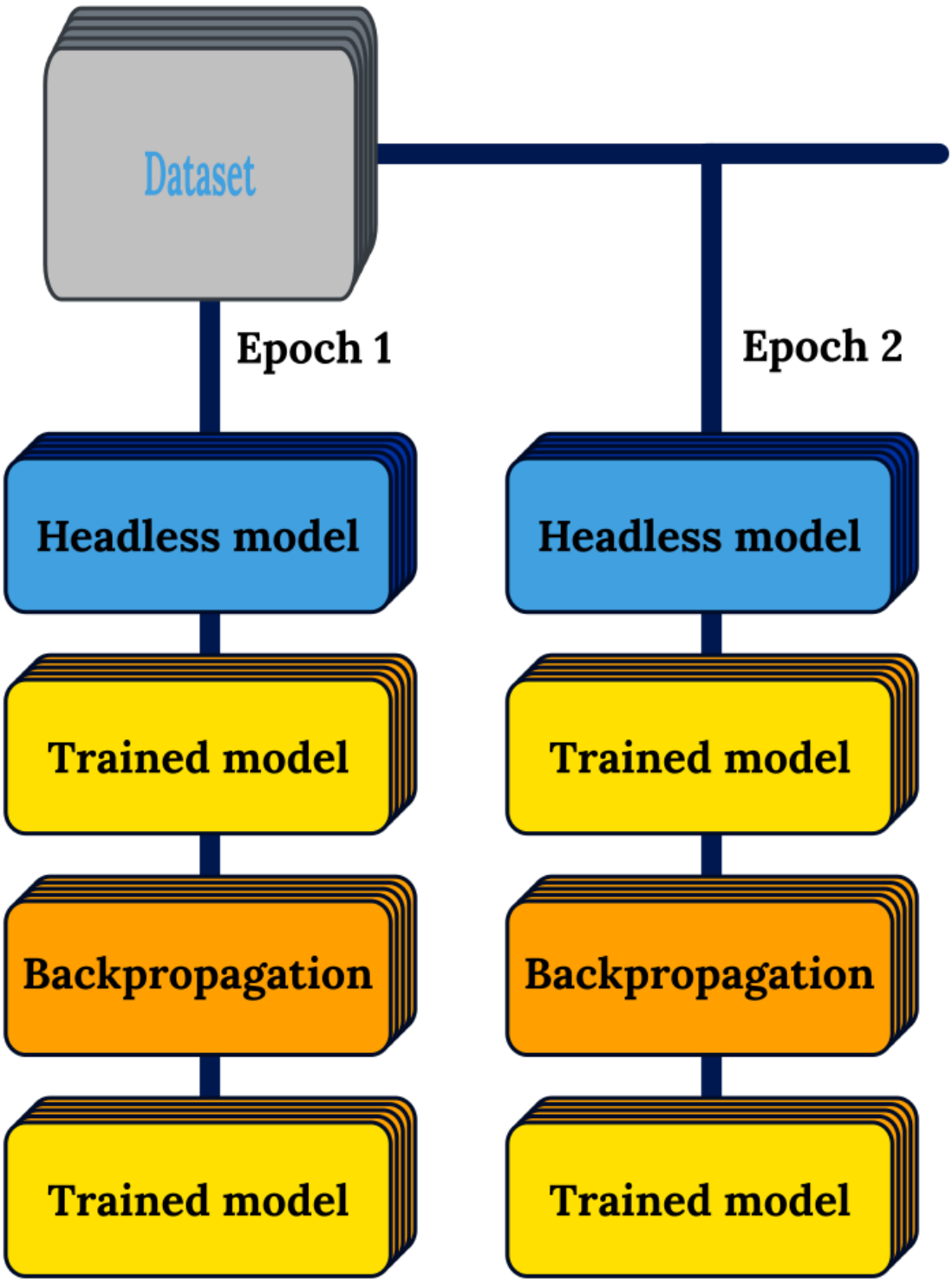
The 4 leverage of DoF:

1. faster deep learning training process
2. consume less storage and bandwidth capacity
3. possibility of addition notes
4. GDPR-, regulation- and legal friendly data-handling

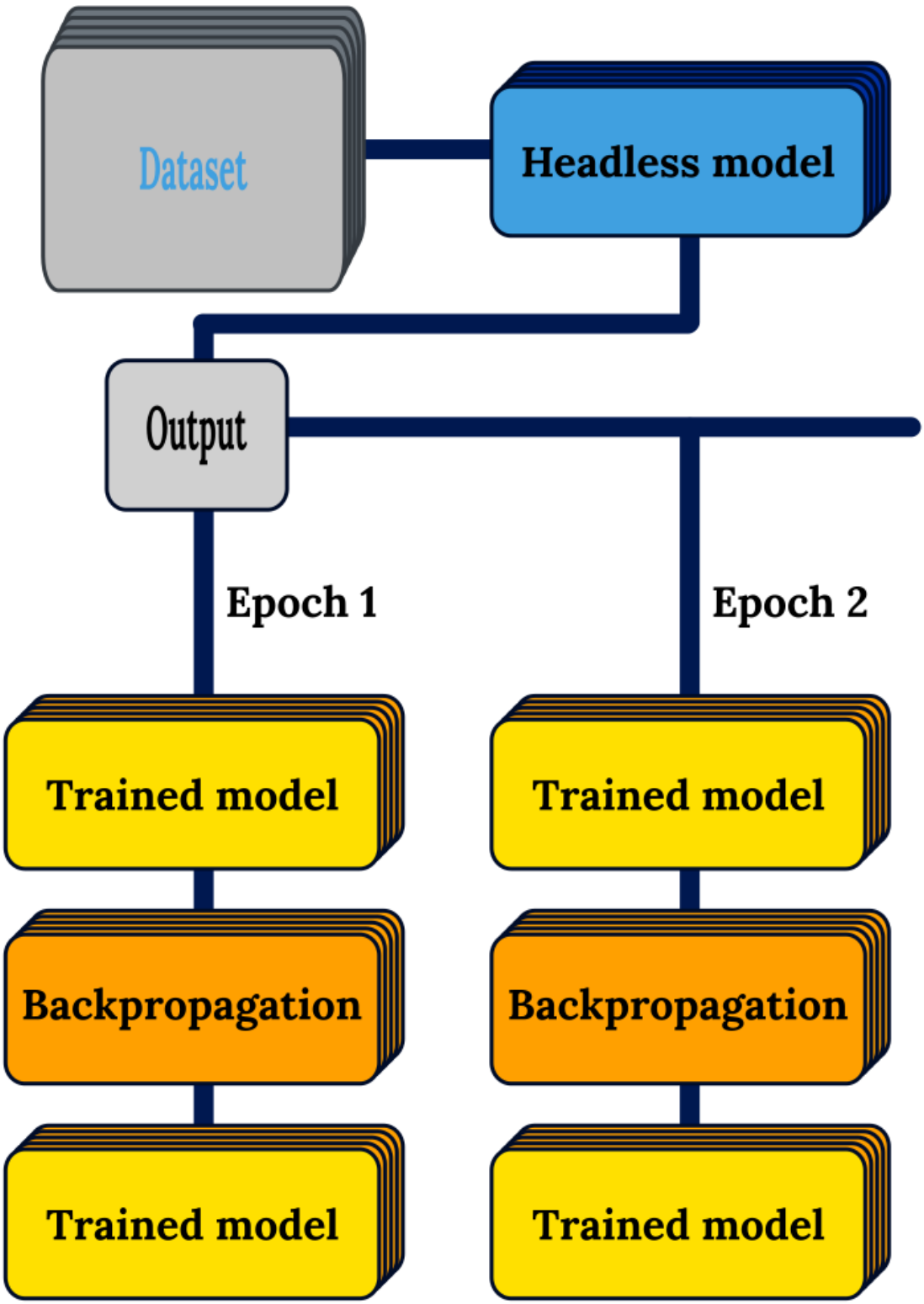
Repository is available here:

https://github.com/hyperrixel/dof

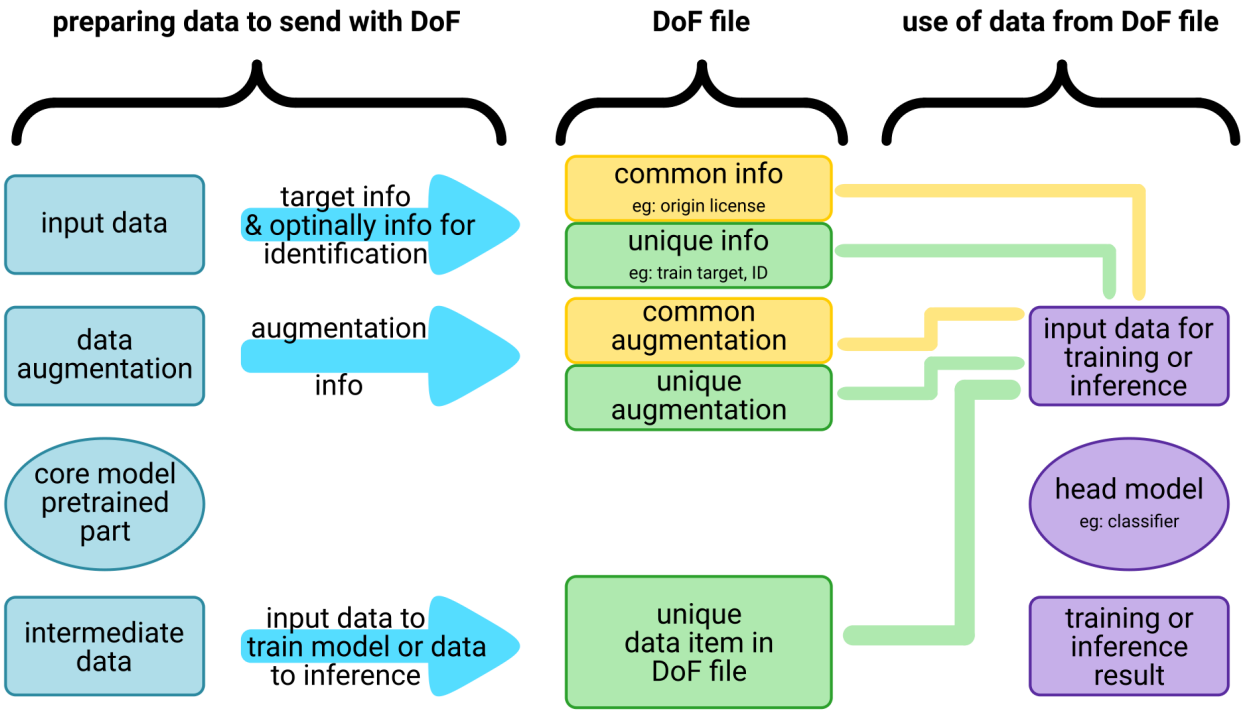
normal train process



DoF train process



Dataflow of DoF



Formulas

`DIS` = **D**oF **I**nfo **S**ize

`EC` = **E**poch **C**ount

`IC` = **I**mage **C**ount

`IS` = **I**mage **S**ize

`HNCT` = **H**eadless **N**etwork **C**omputation **T**ime

`HOLT` = **H**eadless **O**utput **L**oading **T**ime

`HOS` =**H**eadless **O**utput **S**ize

`TNCT` = **T**rained **N**etwork **C**omputation **T**ime

Total saved time

`(HNCT – HOLT) * EC - HNCT`

Benefit ratio

`(TNCT + HOLT) / (HNCT + TNCT)`

Total benefit ratio

`((TNCT + HOLT) * (EC - 1) + (HNCT + TNCT)) / ((HNCT + TNCT) * EC)`

Total saved space

`(IS * IC) - (HOS * IC) - DIS`

MGP – Medical Gateway Platform

Medical Gateway Platform is a data driven solution to connect medical hardware or equipment and to support real-time secure and private data sharing. It helps to developers, medical staff, patients and data scientists at the same time.

Doctors, nurses or operators are able to monitorize the working process. since it handles and contains data to operate medical equipment. This platform is a general solution that means it can be used not only for ventilators, that was the primary target when the idea was born, but for any other type of medical devices. We provide a unique filter solution to handle the data that comes in the system.

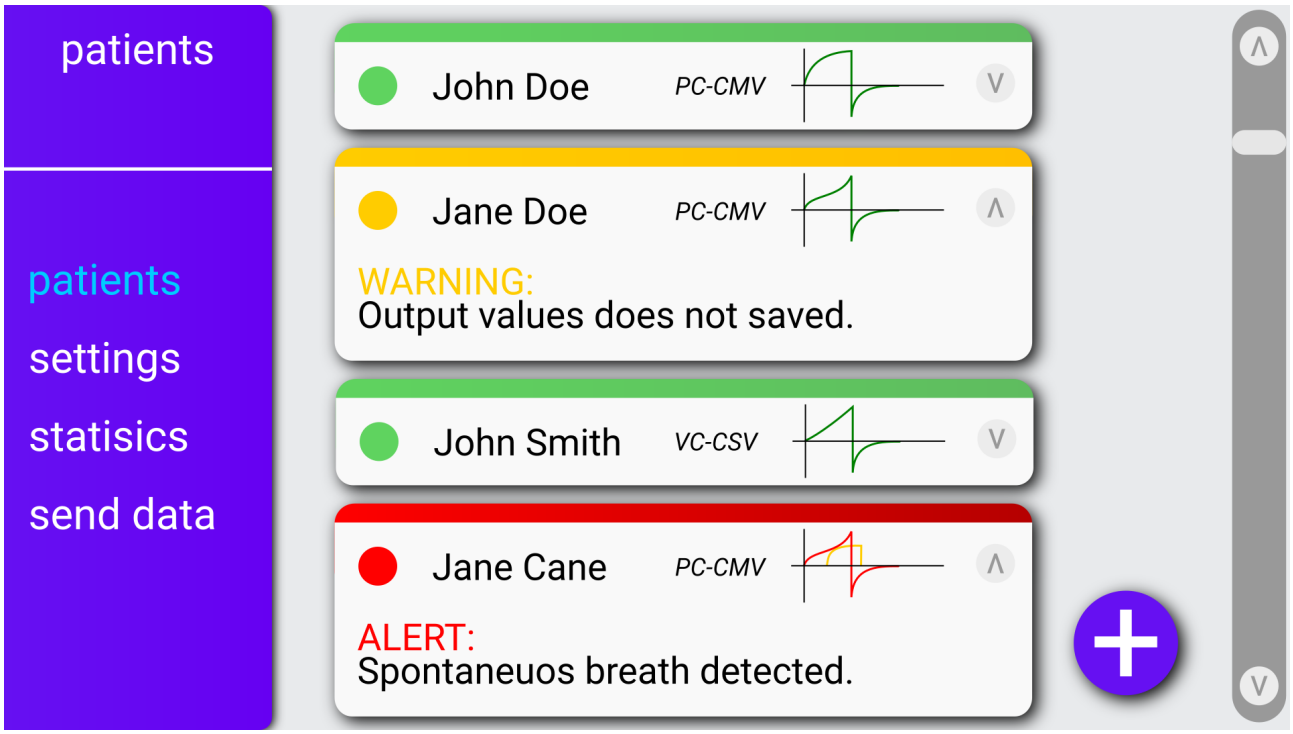
Handling can be conversion, encryption or any possible way of modification. These rules are written by developer but there is a chance to edit or tune later, even if during a real life operation.

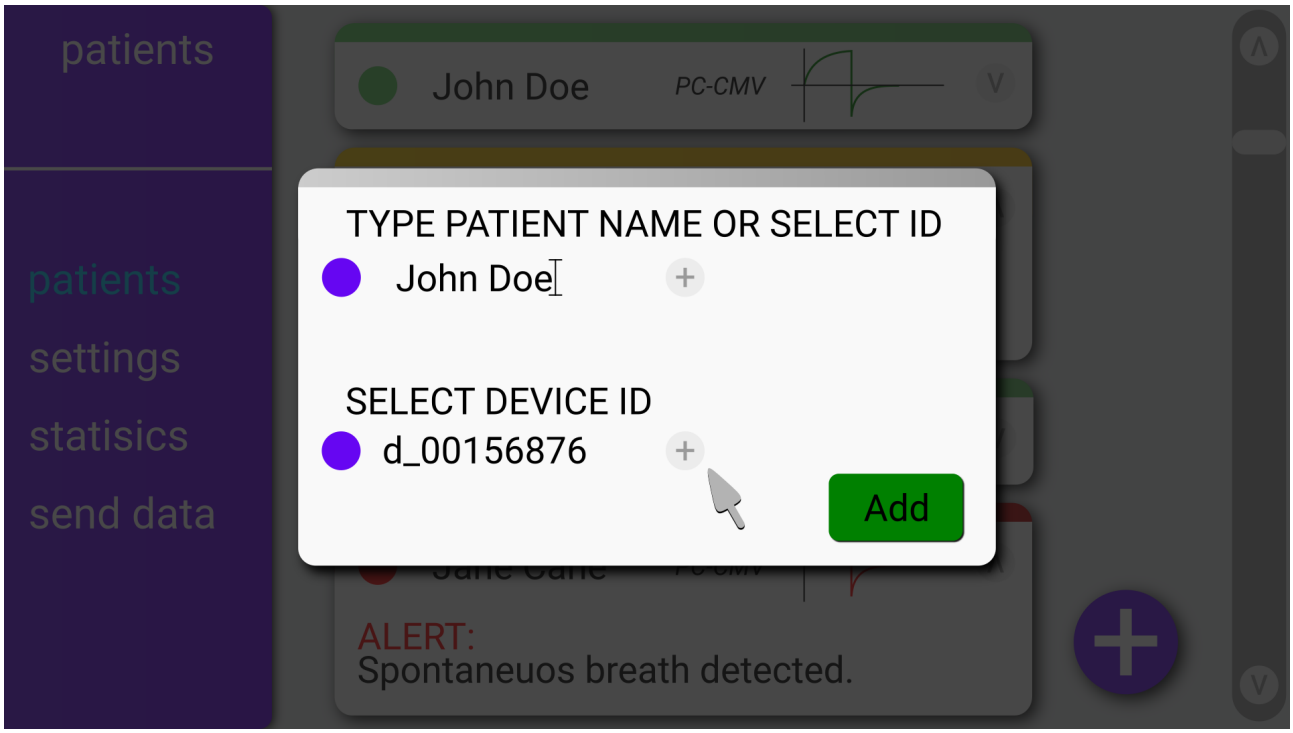
Due to health-data is a sensitive and protected data, we provide the possibility of placing legal and encryption filters. Legal filter grants to export the stored data without breaking any kind of legal regulation such as GDPR or act of countries about usage of health or personal-related data.

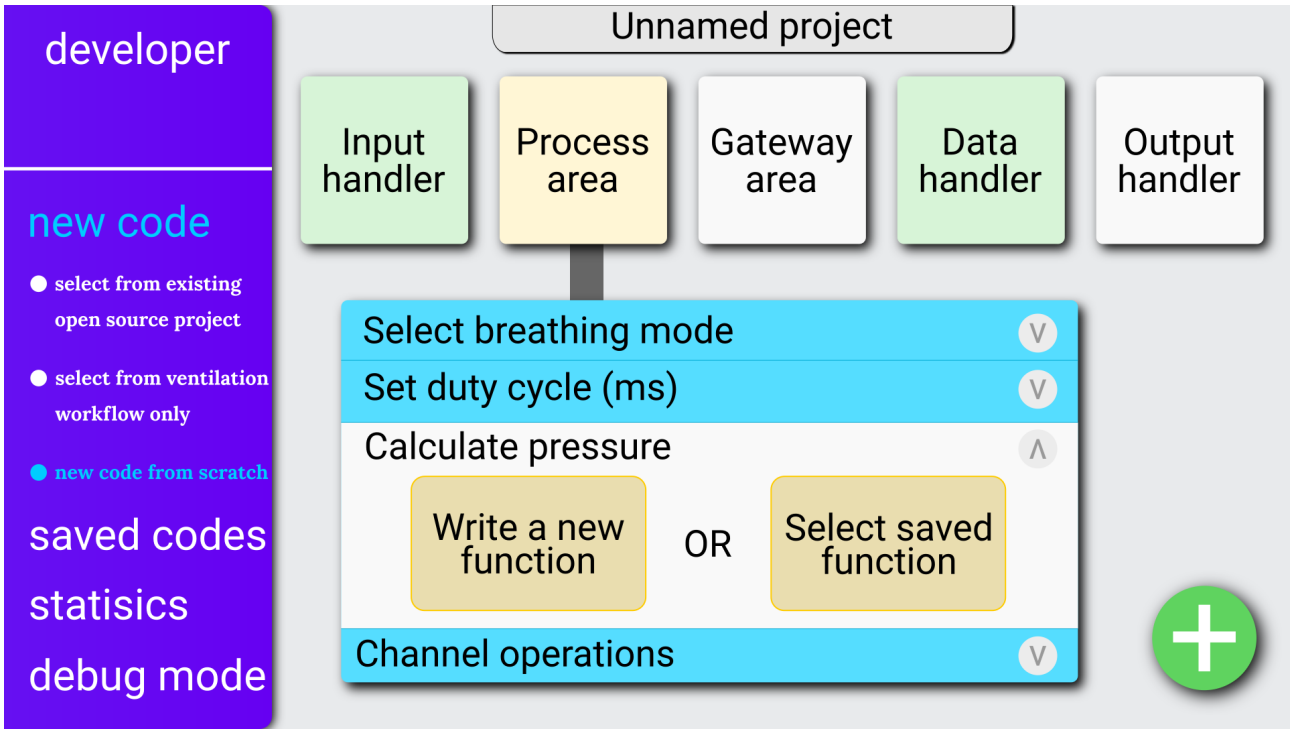
The fundamentals of Medical Gateway Platform are solo machines, they can be computers or micro computers which maintain a medical device like a ventilator or they can be computers or cellphones of doctors or of other medic who controls patient's care remotely. Moreover MGP stations can provide data to the government or to research institutes to be well informed for statistic or for research.

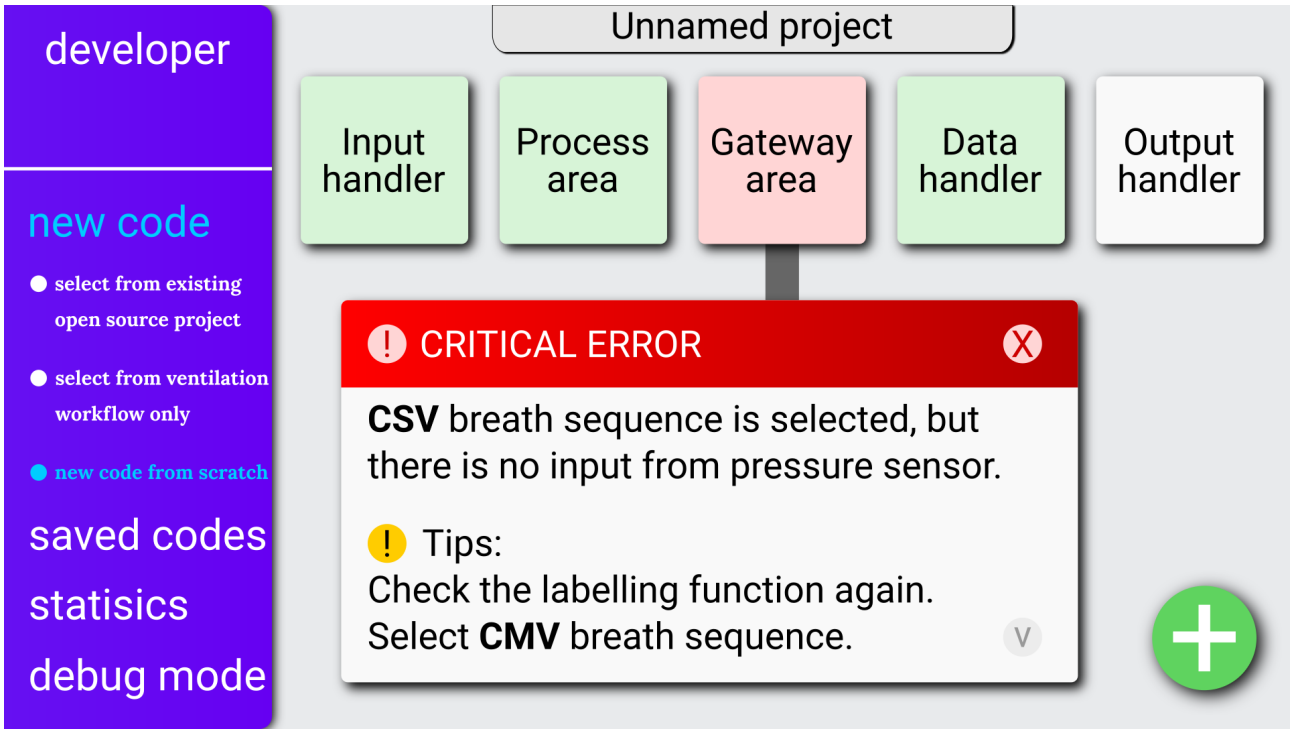
Repository is available here:

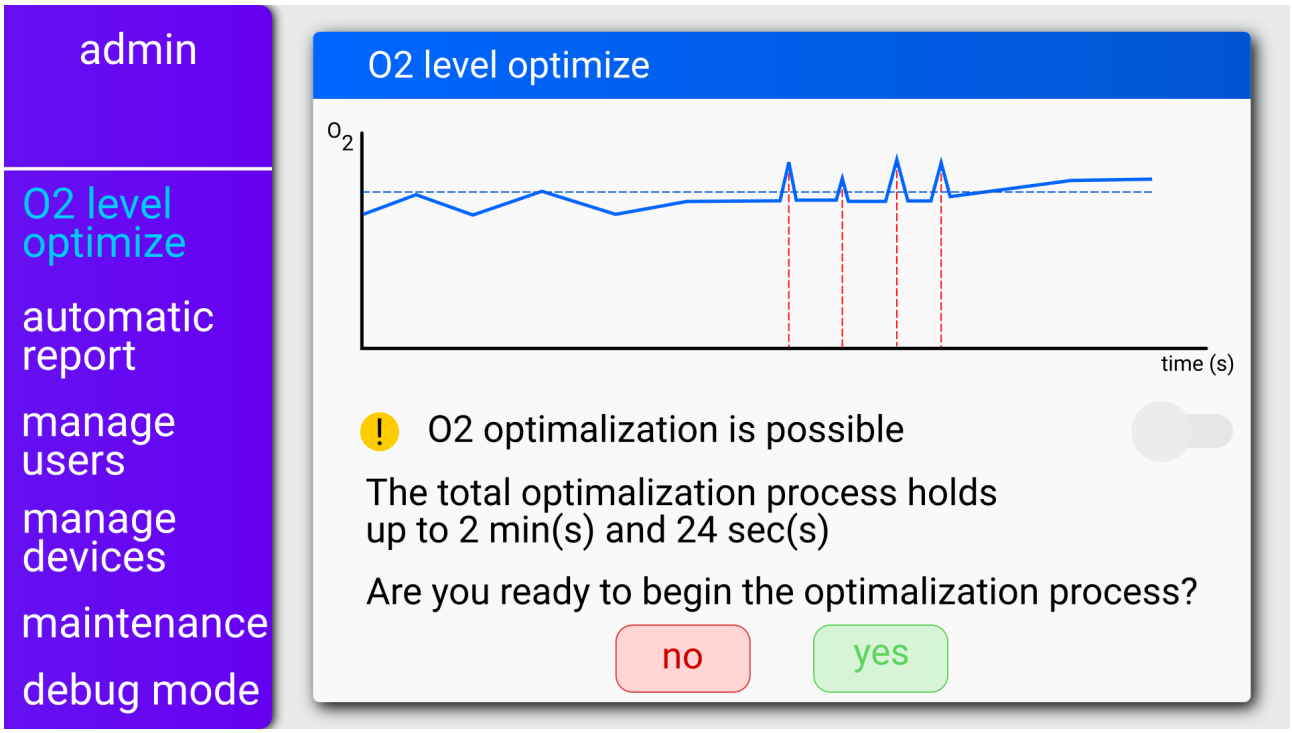
https://github.com/hyperrixel/MedicalGatewayPlatform

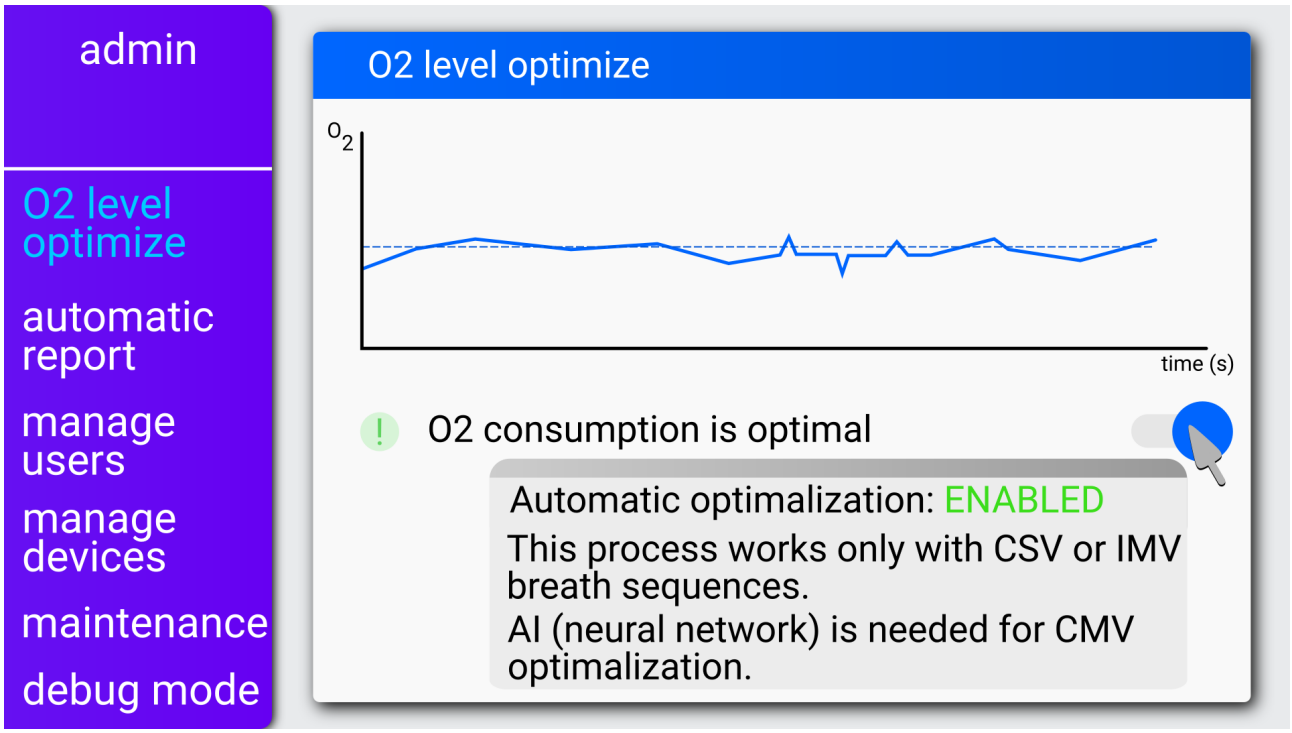

patients
patients
settings
statisics
send data
John Doe
PC-CMV
Jane Doe
PC-CMV
WARNING:
Output values does not saved.
John Smith
VC-CSV
Jane Cane
PC-CMV
ALERT:
Spontaneuos breath detected.


patients
patients
settings
statisics
send data
John Doe
PC-CMV
TYPE PATIENT NAME OR SELECT ID
John Doe
SELECT DEVICE ID
d_00156876
Add
ALERT:
Spontaneuos breath detected.


developer
new code
select from existing open source project
select from ventilation workflow only
new code from scratch
saved codes
statisics
debug mode
Unnamed project
Input handler
Process area
Gateway area
Data handler
Output handler
Select breathing mode
Set duty cycle (ms)
Calculate pressure
Write a new function
OR
Select saved function
Channel operations


developer
new code
select from existing open source project
select from ventilation workflow only
new code from scratch
saved codes
statisics
debug mode
Unnamed project
Input handler
Process area
Gateway area
Data handler
Output handler
CRITICAL ERROR
CSV breath sequence is selected, but there is no input from pressure sensor.
Tips:
Check the labelling function again.
Select CMV breath sequence.


admin
O2 level optimize
automatic report
manage users
manage devices
maintenance
debug mode
O2 level optimize
O2
time (s)
O2 optimalization is possible
The total optimalization process holds up to 2 min(s) and 24 sec(s)
Are you ready to begin the optimalization process?
no
yes


admin
O2 level optimize
automatic report
manage users
manage devices
maintenance
debug mode
O2 level optimize
O2
time (s)
O2 consumption is optimal
Automatic optimalization: ENABLED
This process works only with CSV or IMV breath sequences.
AI (neural network) is needed for CMV optimalization.

PyTessy

PyTessy is a modul that allows faster access to Tesseract-OCR from Python scripts. It uses direct library-level access to Tesseract-OCR's core library.

OCR means optical character recognition or optical character reader when we talk about software, instead of the method. Tesseract-OCR is one of the most current open source OCR software built in C++.

PyTessy is an operating system independent solution, but library search process is implemented on Windows in the current build. PyTessy uses only modules from the Standard Library only. Python version must be >= 3.6. Tesseract-OCR is have to be installed or used in a portable version.

Repository is available here:

https://github.com/hyperrixel/pytessy

PyPi link:

https://pypi.org/project/pytessy/

COVID-19 dataset

We download the daily dataset from ECDC (European Centre for Disease Prevention and Control) and make a standardized csv from this dataset. It contains the additional and cumulated confirmed cases and death number per days and countries. The standardized structure supports the fast processing. It can be filtered fast and easily. Every common programs and modules for datascientists or journalists be able to handle csv format from pandas in python through R till Excel.

Dataset contains 6 columns in the following order:
date | areacode | cumulated cases | cumulated death | additional cases | additional death

Example:

date	areacode	cumulated cases	cumulated death	additional cases	additional death
2020-03-29	20	536	30	41	6
2020-03-29	27	1187	2	17	0
2020-03-29	30	1061	32	95	4
2020-03-29	31	9762	639	1159	93
2020-03-29	32	9134	353	1850	64
2020-03-29	33	37575	2314	4611	319
2020-03-29	34	72248	5690	8189	832

Repository is available here:

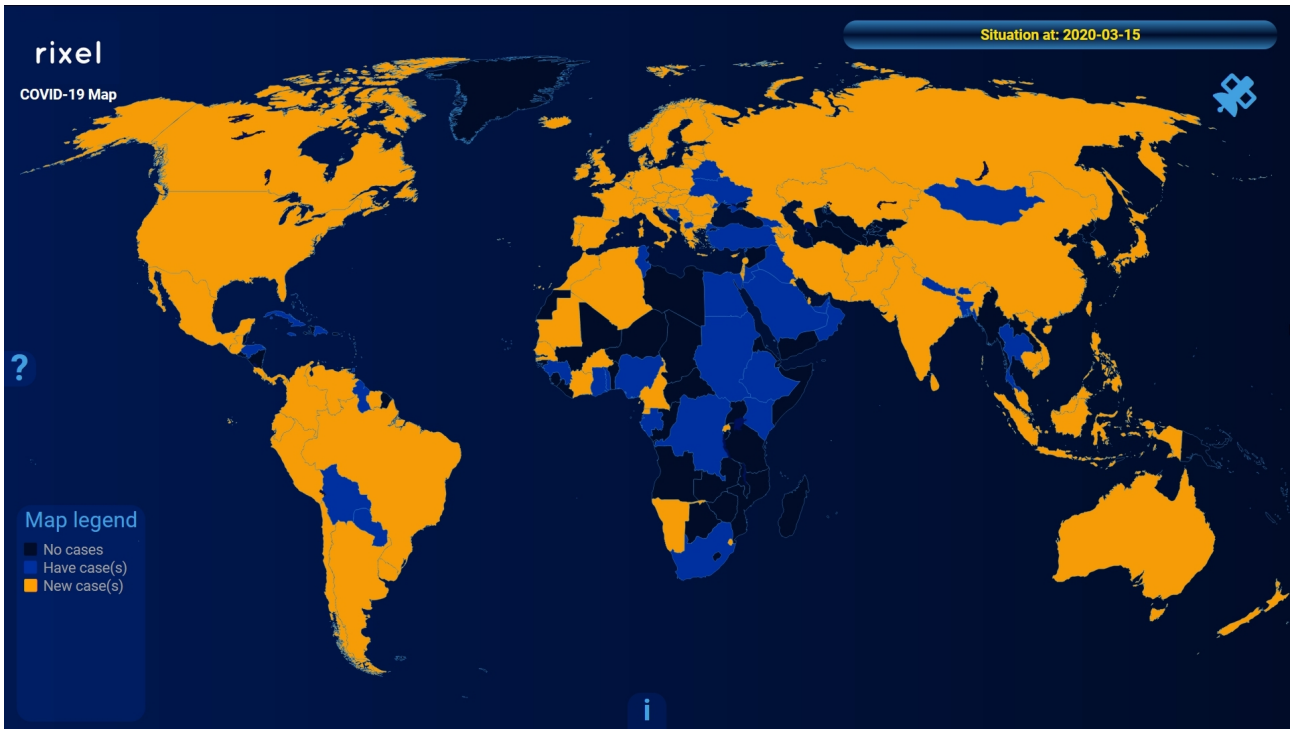
https://github.com/hyperrixel/covid19map-dataset

COVID-19 map

Besides the daily dataset-making routine, we wanted to create a simple visualization tool that helps people to understands the raw numbers and add visualization ideas to other

datascientists. There are a lot of techniques to process an epidemic dataset, we show an example one of them.

Screenshot:



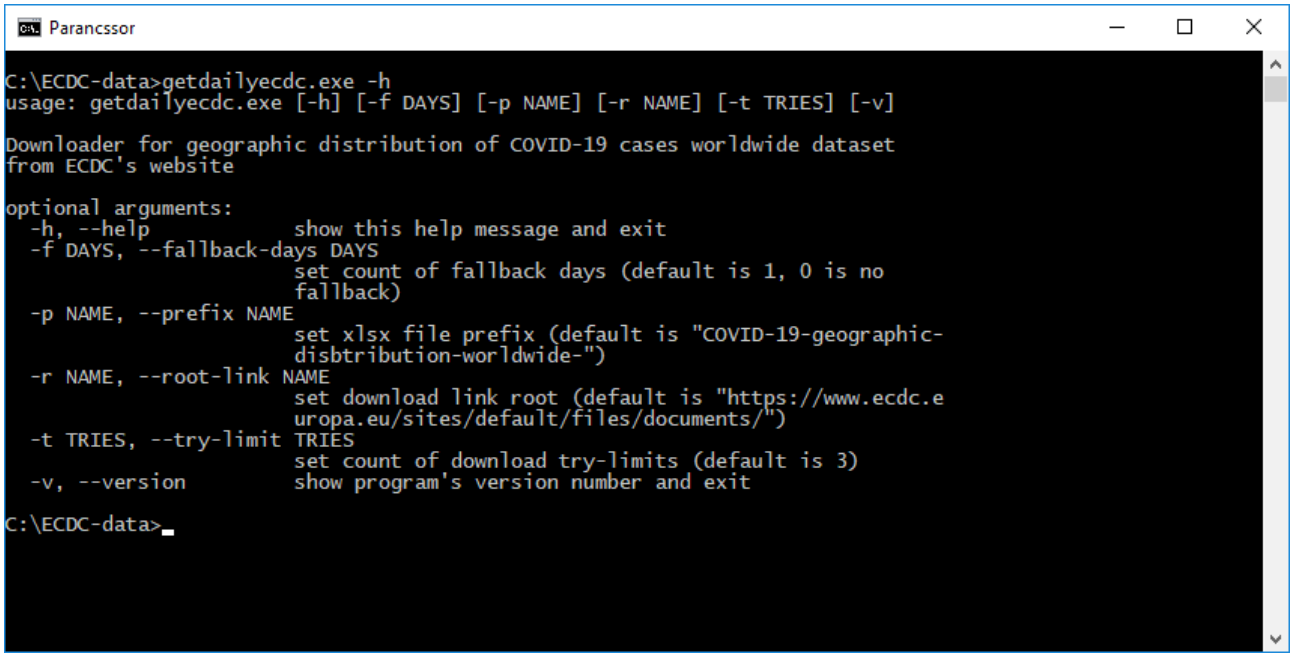
Visualization is available here:

https://www.hyperrixel.com/covid19map/

COVID19CasesWorldwideDatasetFromECDC

The goal of this project is to provide downloader and loader for geographic distribution of COVID-19 cases worldwide dataset from ECDC's website. This project includes multiple files. These can be used as module or as script based on the user's need. As a module ecdcloader.py downloads the latest dataset and loads it into pandas dataframe. As a script it just downloads the latest dataset. It is possible to finetune the parameters with argparser. This method works with the executable file too. For researchers who are not familiar with python, we provide an executable version of script.

Screenshot:



Repository is available here:

https://github.com/hyperrixel/COVID19CasesWorldwideDatasetFromECDC