A Comparative Study on Algorithms of Computer Vision and Deep Learning for Facial Expressions Analysis

Vladyslav Kuznetsov [1], Iurii Krak [1,2], Olexander Barmak [3], Anatolii Kulias [1] and Valentyna Petrovich [1]

[1] *Glushkov Cybernetics Institute, 40, Glushkov avenue, Kyiv, 03187, Ukraine*
[2] *Taras Shevchenko National University of Kyiv, 60, Volodymyrska str., Kyiv, 01033, Ukraine*
[3] *Khmelnytskyi National University, 11, Instytutska str., Khmelnytskyi, 29016, Ukraine*

Abstract

This article discusses two different approaches to study facial expressions on the human face: using a computer vision approaches and deep convolution neural networks. The similarities and difference between these two approaches, in particular the computation devices needed to perform these specific tasks was discussed in detail. The study proposes a technique to estimate the computation power needed to perform facial expressions recognition based on open data on performance of CUDA and OpenCL libraries on different computational devices. The best-case and worst-case scenarios are studied in order to find the performance bottlenecks that happen in deep learning using graphic processing units. This involved studying pre-processed facial expression datasets including positions of facial landmarks on human face using deep unsupervised autoencoders with different levels of constraints to compare them with singular value decomposition. It also included in-depth study of convolution neural networks on images of facial expressions (FER-2013 dataset) using tensorflow-directml library. A set of tests was conducted in order to infer relative performance such as time of processing (for input images) as well as processed data. According to the experimental study, our tests of real performance on particular task stand together with theoretical estimates. A few solutions to overcome the computation bottlenecks in deep learning problems was suggested.

Keywords

Deep learning, facial expressions, computer vision, recognition, Tensorflow, DirectML

1. Introduction

Recent advances in human-computer interaction allow usage of more user-oriented means of communication; such means include voice commands, gesture input systems [1], haptic control, facial recognition [2] and many others. The main outcome of these advances is that they give users new ways to interact with computers and each other; using new means of human-computer interaction give not only the economic but social effect since they widen the area of interest for people with hearing, speech and vision disabilities what is very important in modern society.

Among these new means of communication, one is drawing big interest – facial recognition which also incorporates lip-reading [3] (recognition of words in speech), emotion recognition and user authentication. Based upon recent advances in information technology [4] which incorporates powerful means of data acquisition, data processing as well as new algorithms that can process this data, now it is possible on nearly every desktop computer nowadays.

The development of new means of communication comes with development of new algorithms and new computer computation devices since analysis of speech, volumetric sensing and computer vision

COLINS-2022: 6th International Conference on Computational Linguistics and Intelligent Systems, May 12–13, 2022, Gliwice, Poland
EMAIL: kuznetsow.wlad@gmail.com (V. Kuznetsov); yuri.krak@gmail.com (I. Krak); alexander.barmak@gmail.com (O. Barmak); anatoly016@gmail.com(A. Kulias); filonval63@gmail.com (V. Petrovich)
ORCID: 0000-0002-1068-769X (V. Kuznetsov); 0000-0002-8043-0785 (I. Krak); 0000-0003-0739-9678 (O. Barmak); 0000-0003-3715-1454 (A. Kulias); 0000-0002-5982-8983 (V. Petrovich)

are quite demanding on computational power – processors (CPUs) and graphic processing units (GPUs), which raises a certain list of problems and questions how to solve them properly.

2. Background and problem statement

According to preliminary studies, the software and hardware has to be available for most customers on the market, when introduction of such means is plausible in both economical and technical views. It means that if the hardware is cheap to produce for end customer in relatively high volumes, the software development on these types of hardware also becomes cheaper since the technology can be incorporated in every operating system on state-of-art computers.

Although, during recent two years (2020-2021), the hardware and software production shown high decline. It was caused mostly because of recent pandemic (COVID-19) [5]; thus operation revenue, market share and other economic factors were affected too. According to such high negative effects on global economy, the electronic industry raised high demand of personal computers, laptops and other devices on one hand and on other hand created great decline in availability due to supply chain problems [6]. Due to this, every item that has the highest percentage of IP included in product created the effect, where electronic devices increased in price, in some cases, two to three times (complete devices) to nearly tenfold for various structural elements - microcontrollers, voltage controlling units and many others, which highly influenced the price of graphic processing units (Hardware- und Nachrichten-Links des 18./19. September 2021 | 3DCenter.org) in last year.

Since the graphic processing units are widely known for machine learning tasks in general, their ***prices affected the cost of AI research***. Because of this, one may ask: ***how big is the computation cost, and what is the lowest possible configuration can be used to fulfill such task****?*

If we discuss quite narrow and specific task what – facial recognition what is a topic of interest of this article – we can discuss more in detail, but according to our study in area, most of the tasks related to human-computer interaction – such as gesture recognition, voice recognition, facial recognition and others in one or another way are made using relatively similar data processing algorithms, which are adopted to shape of data structures and are less impacted by type of data itself.

In order to answer the question above, we suggest solving the following problems:

- to conduct a literature survey on facial emotion recognition methods;
- to prepare the test data for experimental tests (facial expressions' datasets);
- to perform an experiment on computer vision methods for emotion recognition;
- to suggest an estimate for performance using standardized computer benchmarks;
- to benchmark a baseline performance of landmark detection algorithms;
- to benchmark a performance of a computer systems on machine learning tasks;
- to suggest ways to optimize network architectures to gain performance;
- to compare performance of GPUs and CPUs on optimized network architectures;
- to discover the edge situations where CPUs outperform GPUs and vice versa;
- to study the edge situations and suggest solutions to increase the performance;
- to test the estimate for performance and computation cost on real machine learning problems.

3. Approaches for facial expression analysis: an overview

3.1. Motion capture and computer vision

Performance motion capture [7] is often used in computer animation in order to obtain precise movements of the body to create an animation model that imitates the movement of a real human [8]. These models utilize different optical, kinematic and magnetic sensors to calculate magnitude of movements of a movement in certain points known as points of interest. In case of optical sensors, these movements can be measured by means of computer vision [9] that distinguishes the image of an optical sensor from background or other parts of the body. This makes the methods used to be quite simple and can be programmed on relatively low-end machines and can be used for real-time tracking of body movements [10]. As a drawback, these systems are suitable only for static systems and very unlikely to be used for end user, except for research purposes (studying biomechanics of the body) or for computer

animators. In our recent studies we have proven that these are suitable for research purposes, and we had a few iterations of facial motion capture technologies, that focus more on sensors, the algorithms and their implementation, what is discussed more in detail in [11, 12].

3.2. Markerless motion capture and deep learning

Markerless capture [13], in contrary to performance motion capture is quite scalable to different devices such as mobile phones, desktop computers or laptops. The main requirements are: availability of built-in or external web camera and availability of sufficient processing power. The recent methods that process input image such as active appearance models or convolution neural networks utilize the so-called activation (or in other words) of the pre-trained neural network in order to obtain characteristics of the image that display certain areas of interest on the human face [14].

Even if there is no need to train weights for specific case, the process of localization of every feature on face, using convolutions has relatively high requirements to memory bandwidth and performance of computing units, so as the computations are made either in onboard SOC video chip (in mobile phones) or on dedicated GPU, which are best suited to process high number of similar operations (such as convolutions) in parallel. In case of using low-end CPUs without dedicated GPU, some solutions can be done sacrificing the image size or frame rate, what is in most cases makes these applications not yet suitable for real-time applications on regular web camera input. This can be applied only for facial recognition to authorize a user; therefore, computation time is not the case [15].

4. Experimental tests of CPU-based solutions in real conditions

4.1. Experimental tests of optical flow algorithms

According to our recent studies, the CPU-based solutions work quite fine for computer vision, in particularly optical flow method implementation. For instance, the performance motion capture solution, discussed in [12] was achieving nearly 60 frames per second on our dataset [16] representing facial expressions filmed at YUV 720p on a test system with A10-9620P CPU using EMGU computer vision library [17]. The main outcome of these experiments is that optical flow method performs well even on low-end on CPU on even outdated desktop machines of similar performance. According to our experimental tests, using these data to classify emotion expressions in separate classes shows us that this video resolution is high enough [12]. Our estimation is that using videos of higher resolution (1080p, 2K and higher) and higher frame rate may not affect quality of data, unless this data is long conversations (acting) and used in computer graphics or film-making. Only in this case, they may benefit from professional video editing software and high-end computers.

4.2. Experimental tests of markerless motion capture

In order to obtain the relative performance of different CPU-based systems, we studied three systems based on three platforms: AMD A10-9620P, Intel Core i5-6600k and AMD Ryzen 5 3600X (no overclocking was used). Every system had similar volume of system memory – 8 GB DDR 4 SDRAM and type of OS – Windows 10. These platforms were tested due to availability and their relative low age. The test systems included benchmarking software was written on Python [18] using external library Dlib [19] – to track the positions of landmarks on face. This implementation was based upon a pre-trained convolution neural network and was tested on 720p test video sequences that contained facial expressions. According to series of tests, the average performance was following – 0.31 FPS for AMD A10-9620P, 0.96 FPS for Intel Core i5-6600k and 2.7 FPS for AMD Ryzen 5 3600X. According to these tests, it becomes very clear that even high-end CPU is not suitable to perform facial expressions' detection using convolution network inference, even though being capable to make similar task with optical flow and permanent markers. The problem lays behind the following: the image (or pattern) of a certain facial landmark can be distorted in many ways and the number of “true” images of same features is stored within a certain neural network detector is much higher in contrary to image of a permanent marker in about 2 orders of magnitude (50-100 times). This leads to the statement that the

computational unit has to contain either 50-100 times more logical cores or have the higher clock speed, what in both cases seem unrealistic in general purpose desktops.

4.3. Performance estimation for certain device using benchmarks

In some typical cases, the performance of one system in a specific task can be compared with the performance of other ones using a standard test and normalizing score that represents performance of each given system. In some cases, these benchmarks are available, and the test system can be compared to a known one. For instance, according to our previous tests (facial landmarks detection), one can calculate performance of one device and compare it with others for this given task. The same can be done comparing performance of general purpose machine learning; for instance, the site Geekbench.com provides results of benchmarking computer systems using Geekbench software [20].

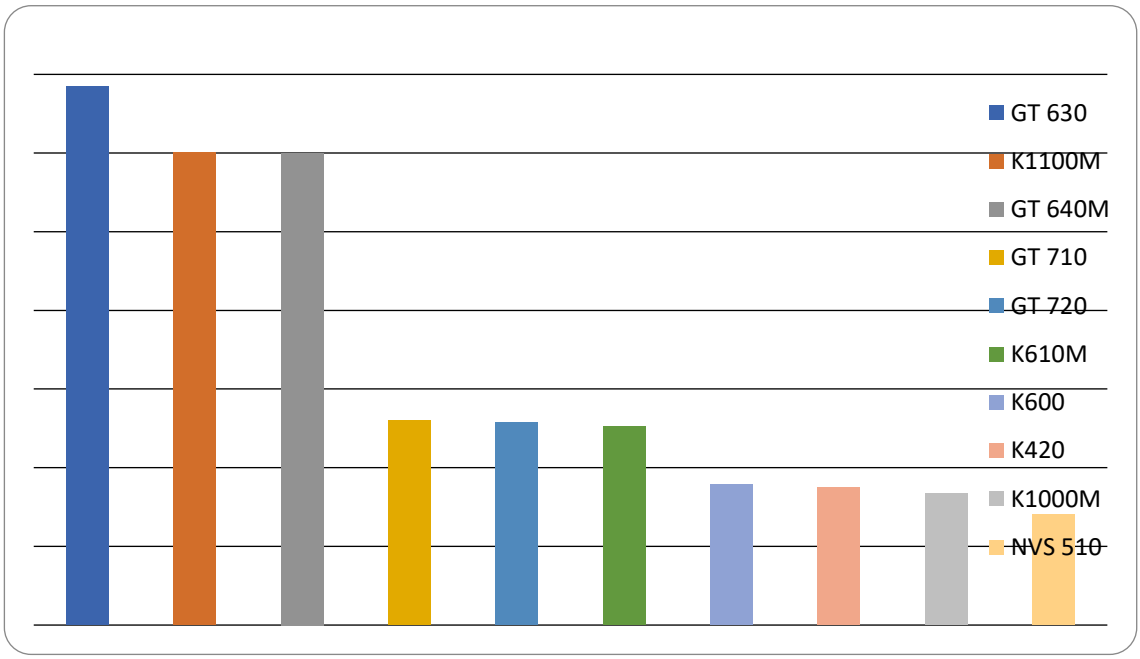


Figure 1: CUDA benchmarks of various low-end GPU devices. Data courtesy of geekbench.com site

Using one of benchmark lists on Geekbench.com site on CUDA technology, we can compare performance of specific GPUs based on known performance of for Intel Core i3-8100, which is taken as 1000 [21]. Thus, if Intel Core i5-6600k is equal to i3-8100 in other tasks, we can infer performance from other tasks (facial landmark recognition): e.g., AMD Ryzen 5 3600X is 2,81 times faster than i3-8100 and around 2x faster than NVIDIA GeForce GT 710 (Fig. 1).

This assumption tends to the following statement. *In order to perform specific task with certain performance*, one can assume, that the desired performance of a specific system with unknown component can be calculated in similar way. Hence, in case of facial landmark detection, we want to achieve certain performance rate based upon our assumptions about desired performance and known performance of baseline system in specific tasks.

It means following: if expected performance of certain CPU-based system is equals 2812 score points (let's assume the baseline AMD Ryzen 5 3600x CPU) with approximate performance of 2.7 operations per second in real benchmark in target task (*frames per second*) and desired performance is 60 operations (frames) in second on 720p video sequences led us to estimated performance of 56240 in specific benchmark, based upon known one, on, for instance OpenCL computing tasks [22]. This gives us an estimated level of performance as of Nvidia Quadro P6000 GPU or similar GPU, for instance AMD Radeon RX 5700 XT (Fig.2).

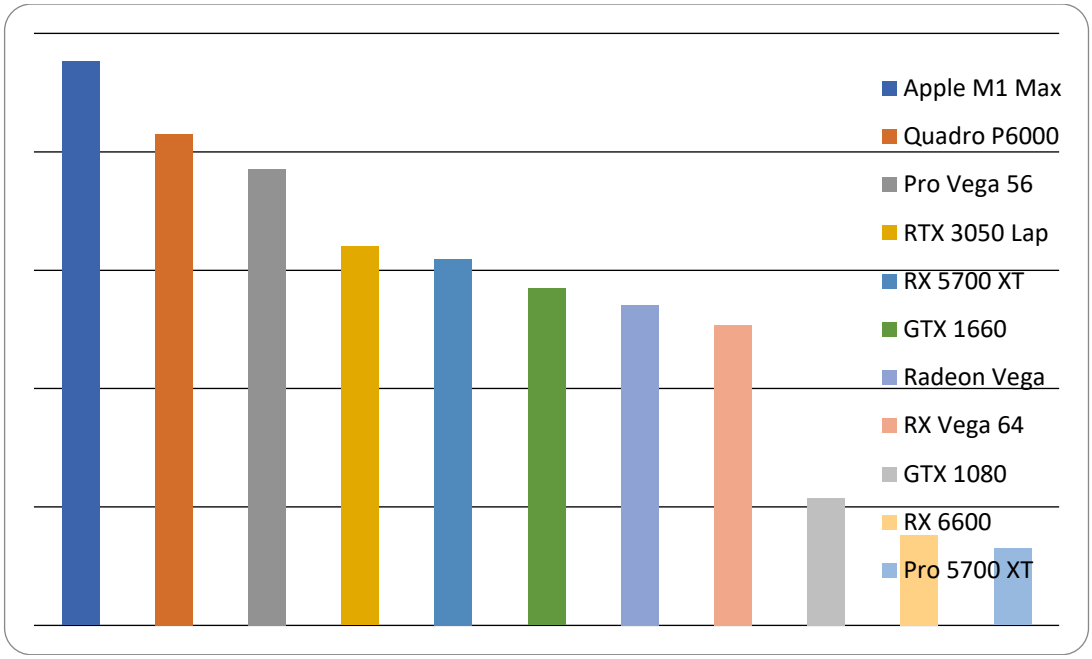


Figure 2: OpenCL benchmarks of various mid-end GPU devices. Data courtesy of geekbench.com site

4.4. Experimental test of general-purpose machine learning

In order to test our hypotheses, an additional study of existing computing tools was conducted in order to assume the computational procedures that require the most computational time and, accordingly, potential ways to optimize these procedures on the same systems, previously tested in computer vision tasks and also extended to two other systems based on AMD Athlon X2 and AMD Ryzen 7 4800H processor with same memory specs (refer to paragraph 4.2.). This may give us a clue how the general-purpose ML is compared to CV tasks.

The following computational procedures were analyzed in the experiment:

- Data read time (from disk) into machine learning environment
- Changing the dimensionality of a data array
- Reducing the dimensionality of data (using singular value decomposition [23])
- Grouping of features (using T-stochastic grouping of nearest neighbors [24])
- Data clustering (K-means clustering) [25]
- Deep artificial neural network (ANN) initialization, training and testing [26]
- Classifier training based on decision trees [25].

The experiments involved computers with different architecture (AMD and Intel processors) on Ubuntu 18 and Windows 10 operating systems.

Testing was performed using Anaconda environment for execution of applications written in Python 3. The results are shown in the Table 1.

Based on the experiment, the following was also found:

- Reading large arrays of data from disk, using different types of processors and different media types can vary up to 10 times (among the least powerful and most powerful computational tools at our disposal), so when processing large data sets media types can play an important role.
- The execution time of machine learning procedures indirectly depends on the number of computing cores and processor threads (logical CPU cores) - to a greater extent affects the implementation of a particular algorithm; for example, the difference in performance between the most powerful and the least powerful processor for grouping of features was 6 times, and for singular decomposition up to 16 times.

- The difference between the mobility and the instruction set may impact the performance of similar operations - such as desktop-based solutions showed better performance procedures for training neural networks in contrary to laptop ones. Note that these indicators are associated with a specific set of instructions for a specific generation of processors and, accordingly, certain procedures will be performed faster due to the availability of library optimization when compiling for these processors.

Table 1
Performance indicators of machine learning procedures for different architectures

Procedure name	Max, sec	Min, sec	Diff, sec	Avg, sec
Read (from disk)	1.4502	0.1398	10.37339	0.6081
Changing the size of the matrix	0.0060	0.0010	6.006006	0.0029
Singular value decomposition	2.1730	0.0751	28.92936	0.6348
Grouping of features	65.5700	10.8194	6.06041	39.0700
Clustering of features	0.6000	0.0728	8.241758	0.2387
Sample preparation	0.0070	0.0026	2.681992	0.0051
Deep ANN initialization	0.2693	0.0558	4.829627	0.1585
Deep ANN training	117.6430	24.1080	4.879832	71.0883
Deep ANN inference	2.2210	0.5830	3.809802	1.2514
Learning decision trees	0.8269	0.1480	5.587162	0.3859

4.5. Network architecture optimization for machine learning speedup

One of the ways to decrease time of learning and inference of a neural network on data is to optimize architecture of the network in such way that decreases computation time [27]. In most cases it depends on width and depth of a neural network; decreasing size may decrease computation power.

Let's discuss it on typical architecture – an autoencoder. The main outcome of an autoencoder is a possibility to pre-train multilayer network or to get optimal non-linear feature space representation that can compact features in clusters and allow applying machine learning methods on data.

According to this approach, autoencoder is a deep learning method with two inputs and two outputs; the first output creates an encoding of data and puts it in the inputs of a decoder that calculates a reconstructed data with regard to reconstruction error and level of detail. Obviously – more layers are presented – more of the features are excluded from the final reconstruction. In contrary, it can be done with so-called wide autoencoder, which expands current data representation. This approach allows preserving most features, but increases the computation costs.

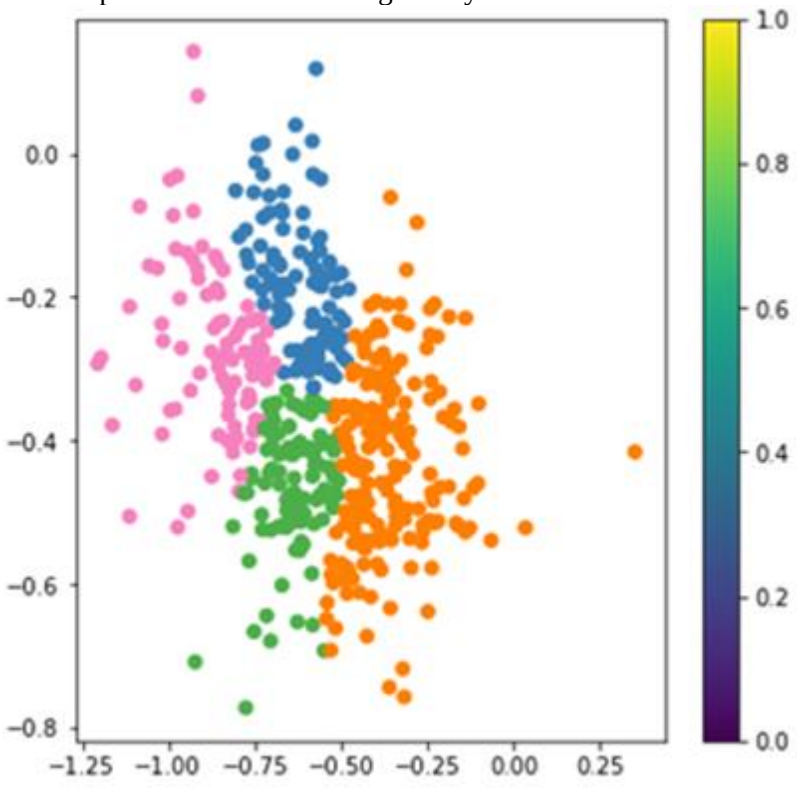
In order to overcome this, an approach is proposed that combines methods of synthesis of linear systems for classification and neural network approaches to learning [28], [29]. The following procedure is proposed: a set of data arrives at the encoder input, which forms an encoding for all data samples; in turn, the decoder layers receive encoding for the data set in the space of reduced dimension; the results of the transformation are subject to the requirements:

- the transformed code allows to obtain approximately direct and inverse transformations for which true $(X^TX)^{-1}A=Y^{\sim}$ i $(A^TA)=E$, $(B^TB)=E$ where X is the output matrix, $Y^{\sim}$ – coding, A - encoder weight, B - decoder weight, E – identity matrix);
- the weights of the matrix of direct and inverse transformation are in a given range of values (data at the input and output and the coding layer have the same dimension);
- the coefficients of direct and inverse transformation are linearly independent.

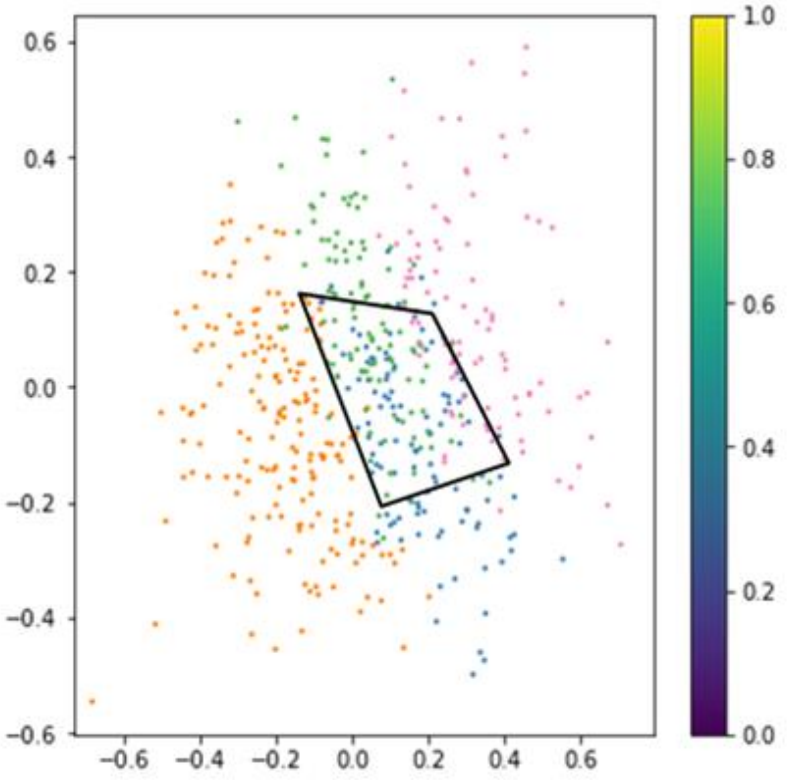
Imposing such constraints on the procedure of autocoder training allows obtaining an approximate solution and implementing the synthesis of linear systems for classification within neural network.

In order to test this approach, an experimental implementation in Python was created, which involves Tensorflow machine learning library. The experimental implementation was tested on a test data set,

which was divided into training and test samples. To train the algorithm, constraints were imposed, which were passed to the procedure for initializing the layers of the autoencoder.



a



b

Figure 3: Constrained autoencoder's labels (a) and SVD with autoencoder data labels applied (b)

To verify the efficiency of the transformation, the data encoding was transmitted to the input of the clustering method. Data labels were used to assess the similarity of the singular value decomposition (SVD) and the proposed approach (Fig. 3).

As a result of testing this approach, it was found that the images of data elements on the hidden layer for the auto encoder and the singular schedule, despite the imposed restrictions, differ. An area with weakly separated data elements is formed in the center (Fig. 3).

As a result of the conducted researches, in general, satisfactory results were obtained - the root-mean-square error of reconstruction and orthogonality of the weights did not exceed 2%.

To improve the results, we also tried to imply other limitations on the optimization procedure and hyper parameters. We hope that this can be improved more, thus the autoencoder encoded feature space representation will behave likely like singular value decomposition in terms of its own feature space being uncorrelated and compact.

We tried this scenario in GPU vs CPU tests, which will be discussed more in detail below.

5. Experimental tests of GPU vs CPU-based solutions vs CPU-based solutions

5.1. Dataset and test system

We decided to study two different scenarios where the GPUs are will be used – facial expressions dataset FER-2013 [30] and our dataset containing processed facial expression (landmarks) [12]. Both datasets represent two different tasks – feature detection on input images and feature dimensionality reduction. The neural networks structures were studied are – deep convolution network (image processing) [31] and constrained linear autoencoder (feature-space dimensionality reduction).

Our test system consists of Windows desktop PC with AMD Ryzen 5 3600X desktop CPU [32] and AMD Radeon RX 6500 XT desktop GPU [33].

All our test scripts were tested within Python 3 (Anaconda) environment with pre-installed tensorflow-directml [34] (TensorFlow version 1.15.5) machine learning library, which is fully compatible with our test system.

5.2. Autencoder training using pre-processed facial expressions dataset

GPUs mostly because of their possibility to accelerate AI tasks are known to be good at performing multiple similar task in parallel because of utilization computing, tensor or shader cores. Thus, there is a misconception that the GPUs are **always** good at performing general machine learning tasks, if comparing them to CPUs.

According to our assumptions, this can't be always true and GPUs should perform better in long memory-consuming tasks that allow storing all intermediate data within GPU memory (video RAM). In order to verify our hypotheses, we ran a task of learning a relatively simple architecture – an autoencoder (AE) on our dataset (refer to chapter 4.5 for more details). Every test had to increase the computation cost (involving more operations) and, hence, time of computation. It would give us estimation where CPU can compete against GPU in computation and where GPU gets better in terms of performance (Table 2).

The Table 2 represents the change in complexity of tasks. The less complicated task is to calculate weights of an autoencoder without any constraints.

Thus, it affects overall number of iterations and time to perform such a task; the end rows in the table contain the task, with respect to constraints of orthogonality, linear independence etc., so number of iterations is highly affected.

According to our tests, we see, that performance issue lies within both complexity of task (e.g. doing one or more operation N times) and number of iterations.

It becomes clear that some time is lost because of PCI Express bandwidth - because the CPU has to transfer the processed data to GPU memory and then perform certain task. Just because the PCI Express to 4 lanes [33], the task may have performed longer due to time needed to read data from system RAM to video RAM.

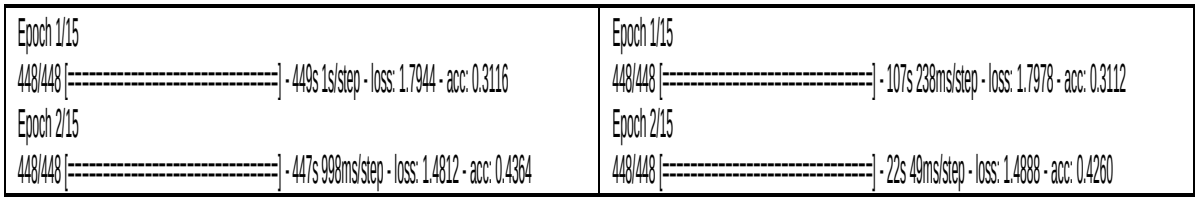
Table 2
Performance ratio of autoencoder training in different tasks

Task name	CPU task time, sec	GPU task time, sec	Performance ratio
Unconstrained AE, small number of iterations	0,46	0,61	0,754098
Slightly constrained AE, small number of iterations	2,19	6,97	0,314204
Slightly constrained AE, moderate number of iterations	27,55	7,36	3,743207
Moderate constrained AE, moderate number of iterations	32,98	7,68	4,294271
Highly constrained AE, high number of iterations	41,11	7,85	5,236943

5.3. Experiments on facial expressions image processing

Since our previous test was carried out on a relatively small dataset, and it had small footprint in terms of number of layers, number of hidden parameters, we decided to study the algorithms on a bigger dataset, which was in our case FER-2013 human emotions' dataset. Main outcome of this test is to verify our hypotheses, which is actually to estimate a performance of a certain computing unit using a certain relative performance, provided online on sites related to computer benchmarking.

According to this, we conducted a series of tests that included usage both of CPU and GPU on our test system. This can help us to estimate the computation bottlenecks and also performance ratio. According to preliminary test, we have got a view how the data affects the performance; the dataset and weights of a neural network has to be transferred in GPU (see Fig. 4).



a b

Figure 4: Performance of deep convolution network training on a CPU device (a) and GPU device (b)

According to the test, the overall time that is needed to transfer the 1.2 GB of data from disk and RAM to video memory is approximately 80 seconds, that is 4 times more than a performance on a second epoch (Fig. 4, b); in worst-case scenario GPU/CPU performance ratio is 4.5 to 1 and best-case is slightly less than 20 to 1 (if the number of iterations is more than 100).

In order to estimate overall time needed to achieve desired performance, we ran a test, which shows us mean–square error (MSE) and accuracy on a test dataset (Fig. 5).

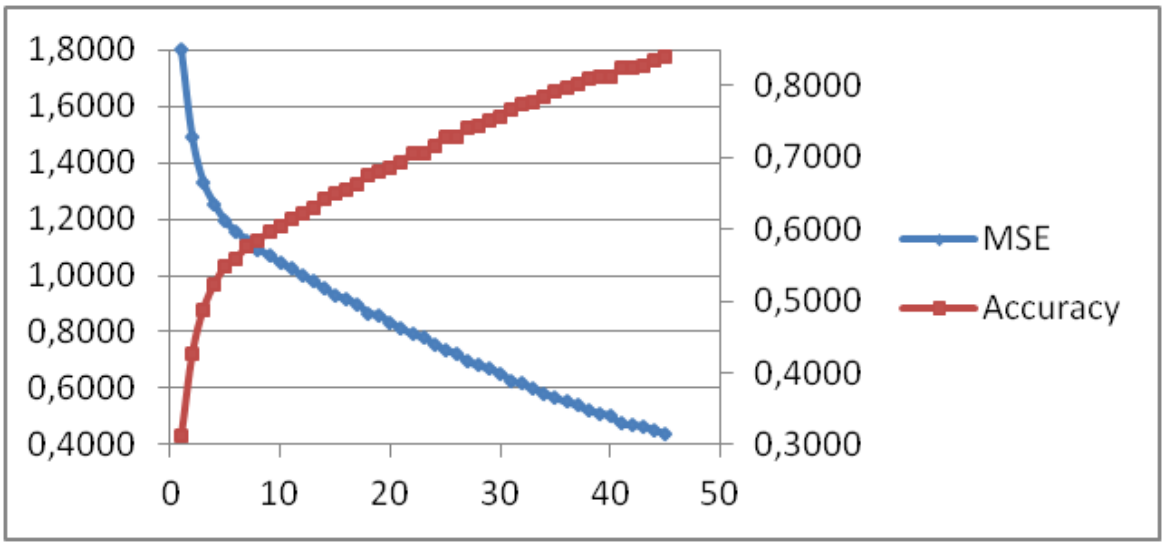


Figure 5: Mean-square error (left axis) and accuracy (right axis) plot on 50 iterations (bottom axis)

According to Fig. 5 it becomes very clear that achieving high (>75%) values of accuracy needs much more iterations and thus, computation time. For instance, time needed to perform 50 iterations for a given computation system, dataset and neural network architecture on a CPU lays behind 6 hours, in contrary the overall time needed to perform the exact same task on GPU is 20 minutes. The performance ratio for such number of iterations for a specific computation system is 18.9 to 1.

6. Discussion

Let's discuss the main outcomes and general thoughts about our experimental tests.

Foremost, we need to discuss the necessity of powerful computation devices. In modern world which nowadays has severe problems in production of electronic devices due to transportation issues, big demand and recent pandemic each research in AI faces the question "*what is the price of a certain computation task*". And this task, besides the price of human time and electricity, is affected by the price of a certain electronic device. Thus, one can ask "*how I can overcome this difficulty*" and the answer is – to model certain big problem in smaller scope. For instance, training a certain natural language processing algorithm can be done on a smaller dataset that involves small computation costs. In one of our experiments, shared in paragraph 4.2 of this article, certainly visible that general purpose machine learning task (not involving neural networks training) and even computer vision can be done even on outdated CPU devices like AMD Athlon X2. In contrary, in further parts of this article (paragraph 4.4 and 5.2) it is visible that tasks involving deep learning have a great need of GPU accelerators since time of computation on CPUs on a certain task is out of range.

On the other hand, one may ask another question, "*do we really need that high computation power?*" According to our research, the computation cost can be estimated given desired performance, baseline performance and known performance metrics on a specific task. In earlier chapters of this article, we raised a question – "*what a computation device is needed to detect landmarks on images of a human face?*". We estimated baseline performance of our system using AMD Ryzen 5 3600X (without the GPU) on this task using Dlib library to detect the landmarks is around 3 frames per second on 720p video. Using the rule of thumb, we can project the estimated performance (60 fps) on a specific device – for instance, Nvidia GeForce 1660Ti and then compare the estimation with our results. According to our tests on convolution neural networks, we can estimate the performance on this task given our GPU device – AMD Radeon RX 6500 XT – and compare it to one we can estimate via benchmarks (Fig. 6).

According to our calculations, if the estimated performance ratio is 18.9 (performance ratio against AMD Ryzen 5 3600 X) multiplied by 2812 (theoretical performance of AMD Ryzen 5 3600 X against Intel i3-8100U) it gives us performance benchmark value of 53156, which is slightly less than estimated 53342 (by 0.35%) that is a good result. This proves that this type of performance estimation can be used to infer performance on an unknown task based on known task performance.

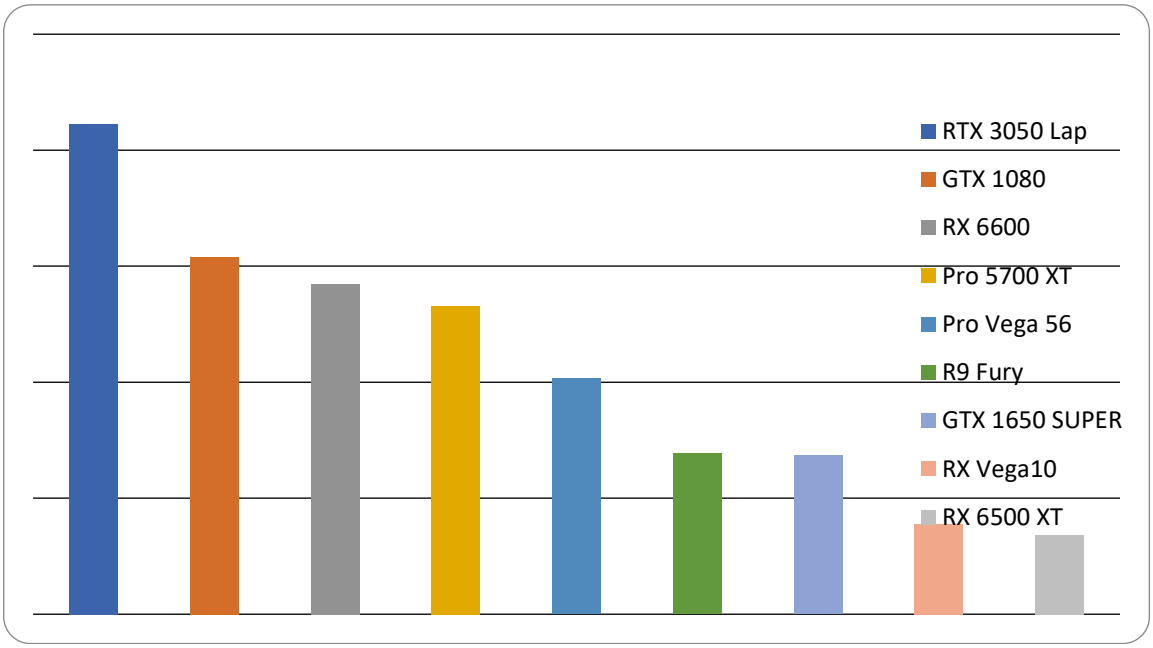


Figure 6: OpenCL benchmarks of various mid-end GPU devices. Data courtesy of geekbench.com site

However, our performance estimations, according to other tests, in less complicated tasks may not always be true (Table 2). The outcome of abundant computation power isn't always useful when studying "toy" datasets like MNIST since most of the computation certainly can be done on a CPU.

It means that cost of computation involves only price of CPU and a computation system; however, if taking in account the price of a specific CPU and GPU in price of computations done on this system, it means that actually on simple tasks, price of computation per certain amount of time can be actually twice as higher than of a baseline system if taking in account the manufacturer's suggested retail price (MSRP) from [33], [34].

In contrary, in opposite case, where the computation power is of high importance (for instance image classification or landmark detection), the actual price of a system is overshadowed by a human time needed to wait the results to perform next tasks such as making assumptions and reasoning about the data, reprogramming the architecture due performance issues or changing the data due to lack of consistency of results or confidence about them due to low performance.

The high-performance solutions may be useful if one have to make these solutions often or to process big amounts of data on certain computer system.

Other significant outcome of our experiments is performance issues. Actually, it means that not every task can be done on a given computation system, even having enough computation power. In our case, it means that the computation system is restricted to a video RAM size and PCIe bandwidth.

When we studied convolution networks (paragraph 5.3), our test system already involved 1.2 GB of video memory, and we saw that each GB restricts the performance drastically.

One may assume that at the moment where the overall volume of data exceeds the size of a video RAM, here become another issues – using a shared memory. Since the memory speed of system RAM (DDR4) and video RAM (GDDR6) differ significantly, the computation of such task may become too expensive in terms of time.

So the solution of this task can be done in literally two ways – either using GPU with higher amount of video RAM or scaling down the task that helps to solve it without exceeding the memory restrictions. The same can be said about the datasets themselves.

Using data augmentation and data slicing one can get smaller dataset which is quite viable and plausible solution to solve this task.

7. Conclusions and future work

We discussed two approaches in facial emotion recognition which involves computer vision algorithms and convolution neural networks, in particular how the type of computation unit affects the performance on certain computer vision and machine learning task. The study involves open data on performance rate of a certain computation device using OpenCL and CUDA libraries on standard benchmarks from Geekbench.com website. According to our tests in two machine learning tasks – training a restricted autoencoder on facial expressions temporal data and training a convolution neural network on facial expression image dataset (FER-2013) we have got a performance ratio that agree with our previous estimates with around 0.35% deviation of performance rate. Using the same data, we investigated bottlenecks in GPU machine learning, which are influenced by hardware and size of a dataset. In further research, we plan to study other types of deep learning methods such as recurrent neural networks and study them on real data.

8. References

[1] G. Devineau, W. Xi, F. Moutarde, J. Yang, Deep Learning for Hand Gesture Recognition on Skeletal Data, 13th IEEE Conference on Automatic Face and Gesture Recognition (2018) 106-113. doi:10.1109/FG.2018.00025.

[2] A. Ghulam, A. Amjad, A. Farman, D. Umar, M. Fiaz, Y. Sana, A. Tariq, H. Noman, Artificial Neural Network Based Ensemble Approach for Multicultural Facial Expressions Analysis, vol. 8, IEEE Access (2020): 134950-134963. doi: 10.1109/ACCESS.2020.3009908

[3] M.C. Poonam, Review and Analysis of Various Lip Reading System Techniques, volume 6 Issue IV of "International Journal for Research in Applied Science and Engineering Technology", IJRASET, Sonipat, (2018): 4094–4098. doi: 10.22214/ijraset.2018.4675

[4] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al., Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8(53) (2021) 1-74. doi:10.1186/s40537-021-00444-8

[5] J. Nayak, M. Mishra, B. Naik, H. Swapnarekha, K. Cengiz and V. Shanmuganathan, An impact study of COVID-19 on six different industries: Automobile, energy and power, agriculture, education, travel and tourism and consumer electronics, Expert systems, 39 (2021): doi:10.1111/exsy.12677

[6] P. Chowdhury, S. K. Paul, S. Kaisar and Md. A. Moktadir. COVID-19 pandemic related supply chain studies: A systematic review, Transp Res E Logist Transp Rev, 148 (2021). doi:10.1016/j.tre.2021.102271

[7] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, M. J. Black, AMASS: Archive of Motion Capture As Surface Shapes, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 5442-5451. arXiv:1904.03278

[8] J. Zhao, Research on 3D Animation Processing Technology in Modern Art Design System, 2021 International Conference on Computer Network Security and Software Engineering (CNSSE 2021), vol. 1856 (2021). doi:10.1088/1742-6596/1856/1/012052

[9] X. Feng, Y. Jiang, X. Yang, M. Du, X. Li, Computer vision algorithms and hardware implementations: A survey, The VLSI Journal, vol. 69 (2019) 309-320. doi: /10.1016/j.vlsi.2019.07.005

[10] K. Vougioukas, S. Petridis, M. Pantic, Realistic Speech-Driven Facial Animation with GANs, International Journal of Computer Vision, vol.128 (2020) 1398–1413. doi:10.1007/s11263-019-01251-8.

[11] I.G. Kryvonos, I.V. Krak, Modeling Human Hand Movement, Facial Expressions, and articulations to Synthesize and Visualize gesture information, Cybernetics and Systems Analysis, 47(4) (2011) 501-505. doi: 10.1007/s10559-011-9332-4.

[12] I.G. Kryvonos, I.V. Krak, O.V. Barmak, A.S. Ternov, V.O. Kuznetsov, Information Technology for the Analysis of Mimic Expressions of Human Emotional States, Cybernetics and Systems Analysis, 51(1) (2015) 25-33. doi: 10.1007/s10559-015-9693-1

[13] Y. Wu, Q. Ji, Facial Landmark Detection: a Literature Survey, International Journal of Computer Vision. (2018): 115–142. doi: 10.1007/s11263-018-1097-z
[14] D. Mehta, M. Faridul, H. Siddiqui, Facial Emotion Recognition: A Survey and Real-World User Experiences in Mixed Reality, Sensors, 18(2) (2018). doi: 10.3390/s18020416.
[15] M. Wang, W. Deng, Deep face recognition: A survey, Neurocomputing, vol. 429 (2021) 215-244. doi: 10.1016/j.neucom.2020.10.081.
[16] O. Barmak, O. Kalyta, I. Krak, E. Manziuk, V. Kuznetsov, Model of the facial emotions expressions based on grouping classes of feature vectors, Advances in Intelligence Systems and Computing, 1246 (2021) 65-76. doi: 10.1007/978-3-030-54215-3_5
[17] EmguCV: a cross platform .Net wrapper to the OpenCV image processing library, 2022. URL: https://www.emgu.com/wiki/index.php/Main_Page
[18] Python software foundation, Python: an open-source programming language, environment and interpreter, 2022. URL: https://www.python.org/about/
[19] Dlib.net, Dlib: modern open-source C++ library containing machine learning algorithms and tools, 2022. URL: http://dlib.net/
[20] GeekBench5: a cross-platform performance benchmark for computer hardware, 2022. URL: https://www.geekbench.com/
[21] GeekBench: a list of performance benchmarks on Nvidia CUDA technology, 2022. URL: https://browser.geekbench.com/cuda-benchmarks
[22] GeekBench: a list of performance benchmarks on OpenCL technology, 2022. URL: https://browser.geekbench.com/opencl-benchmarks
[23] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale, SIAM REVIEW, 60(4) (2018) 808-865. doi:10.1137/17M1117732
[24] D. M. Chan, R. Rao, F. Huang, J. F. Canny, T-SNE-CUDA: GPU-Accelerated T-SNE and its Applications to Modern Data, 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), (2018) 330-338, doi: 10.1109/CAHPC.2018.8645912. arXiv:1807.11824v1 [cs.LG]
[25] S. Dasgupta, N. Frost, M. Moshkovitz, C. Rashtchian, Explainable k-Means and k-Medians Clustering, 37 th International Conference on Machine Learning (PMLR) 119 (2020) arXiv:2002.12538 [cs.LG]
[26] S. Arora, N. Cohen, E. Hazan, 35th International Conference on Machine Learning (PMLR), vol. 80 (2018) 244-253. arXiv:1802.06509 [cs.LG],
[27] Y. Xue, T. Tangб A. X. Liu, Large-Scale Feedforward Neural Network Optimization by a Self-Adaptive Strategy and Parameter Based Particle Swarm Optimization, IEEE Access, vol.7 (2019) 52473-52483. doi: 10.1109/ACCESS.2019.2911530.
[28] A. Vahdat, J. Kautz, NVAE: A Deep Hierarchical Variational Autoencoder, 34th Conference on Neural Information Processing Systems (NeurIPS 2020), 2020. arXiv:2007.03898 [stat.ML]
[29] TensorFlow: A System for Large-Scale Machine Learning, 2022. URL: https://www.tensorflow.org/about/
[30] Kaggle.com, FER-2013 facial expression database: learn facial expressions from an image, 2022. https://www.kaggle.com/msambare/fer2013
[31] F. Sultana, A. Sufian, P. Dutta, Advancements in Image Classification using Convolutional Neural Network, Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), (2018) 122-129, doi: 10.1109/ICRCICN.2018.8718718.
[32] TechPowerUp Central Processing Units Database: AMD Ryzen 5 3600X specifications, 2022. URL: https://www.techpowerup.com/cpu-specs/ryzen-5-3600x.c2131
[33] TechPowerUp Graphic Processing Units Database: AMD Radeon RX 6500 XT specifications, 2022. URL: https://www.techpowerup.com/gpu-specs/radeon-rx-6500-xt.c3850
[34] Microsoft Corporation, GitHub repository for Tensorflow fork accelerated by DirectML, 2022. URL: https://github.com/microsoft/tensorflow-directml.