Journal of Real-Time Image Processing (2024) 21:113
https://doi.org/10.1007/s11554-024-01493-x

RESEARCH

GPU-based key-frame selection of pulmonary ultrasound images to detect COVID-19

Emanuele Torti[1] · Marco Gazzoni[1] · Elisa Marenzi[1] · Francesco Leporati[1]

Received: 18 April 2024 / Accepted: 5 June 2024 / Published online: 15 June 2024

Abstract
In the last decades, technological advances have led to a considerable increase in computing power constraints to simulate complex phenomena in various application fields, among which are climate, physics, genomics and medical diagnosis. Often, accurate results in real time, or quasi real time, are needed, especially if related to a process requiring rapid interventions. To deal with such demands, more sophisticated approaches have been designed, including GPUs, multicore processors and hardware accelerators. Supercomputers manage high amounts of data at a very high speed; however, despite their considerable performance, their limitations are due to maintenance costs, rapid obsolescence and notable energy consumption. New processing architectures and GPUs in the medical field can provide diagnostic and therapeutic support whenever the patient is subject to risk. In this context, image processing as an aid to diagnosis, in particular pulmonary ultrasound to detect COVID-19, represents a promising diagnostic tool with the ability to discriminate between different degrees of disease. This technique has several advantages, such as no radiation exposure, low costs, the availability of follow-up tests and the ease of use even with limited resources. This work aims to identify the best approach to optimize and parallelize the selection of the most significant frames of a video which is given as the input to the classification network that will differentiate between healthy and COVID patients. Three approaches have been evaluated: histogram, entropy and ResNet-50, followed by a K-means clustering. Results highlight the third approach as the most accurate, simultaneously showing GPUs significantly lowering all processing times.

Keywords ResNet · K-means · High-performance computing · Key-frame selection · Artificial intelligence · Machine learning

1 Introduction

The SARS-CoV-2 virus belongs to the coronavirus family and is responsible for the COVID-19 infection, which involves the respiratory apparatus and is transmitted through drops produced when coughing, sneezing or simply talking and breathing. The incubation period lasts usually 4–5 days [1].

If symptoms develop, after about a week the illness can either heal or worsen into a serious condition, developing dyspnoea due to hypoxaemia, quickly followed by respiratory insufficiency that evolves into an acute respiratory distress syndrome. This can be observed as a pulmonary lesion characterized by inflammation and loss of pulmonary tissue [1]. In certain cases, pneumonia is present, producing a decrease in oxygen saturation and alterations, such as anomalies of frosted glass, stain stabilization and interlobular involvement made visible thanks to radiations and other imaging techniques [2].

This illness has induced scientists to develop rapidly reliable diagnostic approaches to reduce both the spreading of the infection and serious complications. The traditional diagnostic methodology to detect and monitor pneumonia is computed tomography (CT) of the thorax; COVID-19

✉ Elisa Marenzi
elisa.marenzi@unipv.it

Emanuele Torti
emanuele.torti@unipv.it

Marco Gazzoni
marco.gazzoni01@universitadipavia.it

Francesco Leporati
francesco.leporati@unipv.it

1 Department of Computer, Electrical and Biomedical Engineering, University of Pavia, Via Ferrata 5, 27100 Pavia, Italy

patients in particular show specific features: frosted or thin reticular opacities, reticulation, vascular thickening, traction bronchiectasis, bilateral involvement, predominant inferior lung involvement and multifocal distribution [3]. CT scans show high sensitivity and specificity in detecting COVID-19 pneumonia; however in multiple studies, both symptomatic and asymptomatic patients did not show identifiable anomalies. Besides, in a considerable number of children undergoing CT, no pathological signs could be detected [4]. Therefore, new diagnostic tools are required, since many patients with COVID-19 pneumonia do not show alterations, especially in the first stages of the disease [5].

Pulmonary ultrasound is a non-invasive technique to diagnose and follow up pulmonary interstitial syndrome, since it is able to identify and discriminate between healthy portions of pleura from irregularities, nodules and thickenings [6]. This kind of approach has shown high levels of precision in the diagnosis of pneumonia, even compared to traditional and consolidated methods like CT scans. Therefore, nowadays it is considered an alternative to CT for the diagnosis of interstitial diseases and pneumonia, as well as acute respiratory distress syndrome, with accuracies always higher than 90% and higher values of sensitivity and specificity also [7–9].

Hence, ultrasound imaging represents a relevant technique both in the diagnostic and therapeutic fields, thanks to the opportunity to detect in real time the dynamics of organs' movements and details of blood flow, as well as their low costs without the use of radiation. In this work, a database of pulmonary ultrasound videos has been made available by the IRCCS San Matteo General Hospital of Pavia, Italy, where recordings can vary greatly in terms of the number of frames, duration (due to the specific examination and the subject under test) and resolution (because of the type of instrument in use). Nonetheless, such data represents a fundamental resource for a better understanding of pulmonary issues and improvement of diagnostic approaches. The aim of this work is to provide an innovative way to diagnose the presence of COVID-19 and determine its severity through a non-invasive method, based on pulmonary ultrasound. Key-frame selection is used to select the most informative frames of acquired videos, followed by a ResNet-50 and K-means to highlight diagnostic patterns and then group such frames in terms of their similarity. This methodology has been compared with consolidated approaches to determine the most appropriate frames and severity parameters. The results confirm the ability to correctly detect diagnostic evidence and severity of COVID-19.

1.1 Related works

Imaging modalities such as chest X-ray, computed tomography scans and ultrasound are used for rapid and precise COVID-19 diagnoses; however, processing such images is time-consuming and susceptible to human error. Therefore, artificial intelligence (AI) methods and in particular deep learning (DL) models provide high-performance results, since they automate all stages of feature extraction, selection and classification [10]. More specifically, numerous studies have demonstrated that both CT and lung ultrasound represent the most appropriate diagnostic tools for the early detection of the presence of COVID-19, with ultrasound being portable, safe and with better real-time characteristics compared to CT. In this context, different deep learning (DL) models have been studied and tested to automatically expose the presence of SARS-CoV-2 from medical images, more specifically involving ultrasound technology [11, 12].

However, most of them assess single frames, manually extracted from a video by an expert physician to ensure that the main patterns were present in the image, thus limiting its applicability [13]. In fact, DL models are able to learn data representations autonomously, avoiding manual feature extraction and the limitations imposed by dependency on expert physicians [14]. These models also excel in handling large datasets, inherently addressing scalability issues that often plague conventional ML models. As an example, Two-Stream Inflated 3D ConvNet (I3D) has been tested for end-to-end video classification to perform an indirect SARS-CoV-2 diagnosis [15], while a network based on convolutional neural network (CNN) and long short-term memory (LSTM) cells has been evaluated, but it only differentiates between viral and bacterial cases of pneumonia from healthy patients [16]. Moreover, a combination of multi-layer perceptron (MLP), EfficientNet and vision transformer (ViT) has been used, observing that the second method outperforms the other techniques [17]. However, no information is provided about the criteria chosen to select the most appropriate frame.

In recent years, key-frame selection has emerged as an additional innovative approach also in the field of medical image processing. As an example, considering ultrasound imaging for breast cancer prevention, key-frame selection has been applied to effectively detect the diagnostic features associated with malignant lesions through an automatic reinforcement learning-based framework using videos of unfixed length [18]. In the same context, echocardiography and colonoscopy could benefit from this kind of procedure, thanks to the removal of both less informative and highly correlated frames [19, 20]. Another application concerns endoscopic videos that are useful not only for diagnostic purposes, but also for surgical training and quality assurance. In this case, video summarization in the form of a weighted dictionary selection model was implemented to extract frames with the highest image quality [21]. Moreover, laparoscopic videos represent a critical source, since their frames are very rich in details, thus requiring an efficient approach to extract only the relevant ones and then extrapolate significant characteristics [22, 23]. Even in other types of imaging techniques, results can be improved, like in the CT of the pelvis for bone segmentation to aid in the early diagnosis of injuries, in planning and study the effects of surgeries [24] and to highlight the most important frames during gastroscopy.

Another critical issue when dealing with image processing in the medical field is related to the need of obtaining accurate results in real time or, depending on the specific application, at least in a timely manner, to speed as much as possible the diagnostic process.

In this context, DL model adoption and acceleration have been significantly impacted by the use of high-performance computing technologies, which allow to parallelize and thus optimize code execution with the specific goal of maximizing the speedup. In this way, it is possible to best exploit the characteristics of the computing architecture at disposal and greatly reduce the computational burden, since parallelizing the same operation executed on different data allows a directly proportional gain, in terms of seconds or even minutes saved according to the computational load [25]. Numerous examples in the healthcare field, more specifically as an aid in the diagnostic procedure, are available: from the real-time detection of skin tumours through FPGAs for the design and development of portable and low-consumption systems [26], to the graphics processing units (GPUs) acceleration of super-resolution techniques to enhance image quality when high-resolution instrumentation is not available, thus resulting in a massively parallelized algorithm that compensates for the original slow execution with consistent speedup [27]. Even quantum machine learning could provide accurate predictions of disease progression by identifying ulterior pattern correlations and biomarkers [28]. Such technologies are fundamental also in applications where it is necessary to extract and reconstruct relevant diagnostic information in real time: as an example, to reproduce three-dimensional models of tissue surface and successively mosaic these models, where GPU parallelization dramatically reduces computational time [29]. Moreover, models like CNNs can be highly parallelized by exploiting most matrix operations when using GPUs and some preliminary implementations have been proposed, even for the COVID-19 diagnosis process, such as ResNet-18 application using a desktop GPU from X-ray images [30].

In this work, two classical methods and a deep learning approach to perform key-frame selection of medical video, particularly targeting lung ultrasound (LUS), were studied both in terms of detection accuracy and processing times. The aim is to evaluate three significantly different methodologies to compare the behaviour of the new deep learning one with respect to a simpler, but strongly consolidated, key-frame selection procedure and also a more complex but equally widespread technique. The performance of these methods regarding processing times was measured on different computing infrastructures, considering both serial and parallel processing.

The application of deep learning in the context of LUS for COVID-19 detection and progress monitoring is emerging and could show promising results, but the high computational demands of processing entire videos of varied durations could prevent its widespread usage. As a consequence, the research gap that this work aims at eliminating is to introduce an innovative AI-based methodology to assess COVID-19 stage through the use of HPC resources.

The results highlight that, on one hand, the DL architecture represents the most suitable combination for key-frame selection for the purposes of precise diagnosis and computational demands. Conversely, the need to consider high-performance computing implementations to sustain the impact required by such innovative approaches in terms of computational requirements has led to demonstrate the feasibility and appropriateness of a GPU cluster version of the deep learning method considered as a good compromise between complexity and accuracy.

The paper is organized as follows: firstly, the three key-frame selection methodologies under evaluation are presented, followed by their parallelization approaches on the different technologies used. Then, the results of the optimization procedure are discussed and, lastly, conclusions and potential future developments are stated.

2 Methodology

The aim of this work is to optimize the pre-processing phase by evaluating different key-frame selection methods and then identifying the most appropriate approach to obtain accurate results while parallelizing and accelerating the process. Among the possible methodologies, many studies have been focused on single frame classification, since ultrasound videos are computationally demanding [13]. Such frames are manually extracted by an expert physician who visually evaluates the presence of all relevant characteristics (i.e., B lines and their convergence or pleural thickenings), thus limiting the applicability of these procedures. Here, an efficient and automatic process is proposed to detect and extract the most significant frames from an entire video that represents the input to deep learning classification models. Specifically, different *key-frame selection* strategies are evaluated to perform an ultrasound clip synthesis to extrapolate the most important contents, hence reducing the amount of data to be processed and consequently leading to a faster diagnosis. To accomplish this, a series of methodologies can be adopted: selection of the most appropriate frames; reduction of the memory required to process and archive the videos; simplification of their information structure.

More specifically, three methods of *key-frame selection* are implemented and compared, all belonging to the first above-mentioned methodology: two metrics that represent well-established parameters in diagnosis evaluations, together with a deep learning model, combined with a clustering technique.

2.1 Histogram

The histogram key-frame selection method has found applications in healthcare image processing when thresholding techniques can be of aid, for example in the case of wireless capsule endoscopy [31] or for human video activity recognition [32]. Here, its difference between two consecutive frames is compared using a threshold value [33]. In this work, frames have a depth of 8 bits (for a total of 256 possible values in each pixel); therefore, 128 bins were chosen; after that, for every two frames a score value is computed, followed by a normalization procedure. The resulting vector, of length equal to the number of bins, has each element corresponding to a difference: the overall score value is obtained by summing all vector elements. Lastly, the score is compared to the threshold (that is the sum between the score vector's mean and the absolute value of its standard deviation): all the calculated distances are then listed in a descending order and the N most distant frames are kept. The value N depends on how many frames should be chosen for the classification algorithm. Typically, it is a trade-off between memory occupancy and computational complexity. For the COVID-19 classification task targeted in this work, the value of N is equal to 32.

2.2 Entropy

The entropy is a widely used technique, especially for multimodal medical image fusion [34] and in CT, MRI and X-ray image segmentation [35]. Here, the distance between probability distributions is used for neighbouring frames to partition video sequences [36]; since the source is a discrete aleatory variable, the expected value is a mean of the information contained in each event and weighted with its probability to happen. In this field, it is helpful to evaluate the complexity and the type of information of biological images, with the aim of finding the intensity distribution per pixel among the various frames and detect the most relevant ones through the identification of the most "distant" frames in terms of entropy. At first, for every frame, the frequency of occurrence of each pixel is calculated, followed by their probability evaluation. After that, for non-null probabilities, the base 2 logarithm is computed to obtain the entropy value. Successively, for every couple of consecutive frames, a score parameter is calculated as the entropy difference: such value is then compared to a threshold obtained in a similar way to the previous methodology (sum of mean and standard deviation of the scores vector). Finally, the resulting distances are listed in a descending order to select the $N = 32$ most distant frames.

2.3 ResNet-50 combined with K-means

In this procedure, the neural network is used to code the relevant information present in each frame and is followed by the clustering technique to group images based on the similarity criterion [37]. The ResNet-50 is a convolutional neural network (CNN) that is part of the residual networks (ResNet) family; it is composed of 50 convolutional layers adopting the residual architecture and grouped into blocks. It is used to superimpose a kernel (or mask) function to an input image and then calculate the integral of their product between their overlapping parts. This is done for every possible position of the input image, thus producing a convolutional map that can emphasize specific details like boundaries and prominent characteristics. The key feature of this network is the introduction of residual blocks, made up of two paths: while in the main one, one or more convolutional, normalizing and activation operations are performed, in the residual path the output of these layers is added to the original input, creating a skip connection. This is done because it is believed that the input contains relevant information for the training of the successive layers. In this way, such networks can learn deep and complex representations producing high-quality results [38]. The general ResNet structure makes use of a combination of four different types of layers:

- convolutional layers, to extract principal features through the application of convolutional filters;
- pooling layers, to reduce the dimensions of both feature maps and redundant information, at the same time keeping the most relevant characteristics;
- normalization layers, applied to the feature map to stabilize and improve the training;
- ReLU activation layers, to set to zero the detected negative values while maintaining the positive ones. This kind of layer introduces a nonlinearity in neural networks that is helpful in overcoming the problem of gradients disappearing, thus fostering a more stable training and a more accurate learning.

The ResNet-50 in particular is composed of the following layers (Fig. 1):

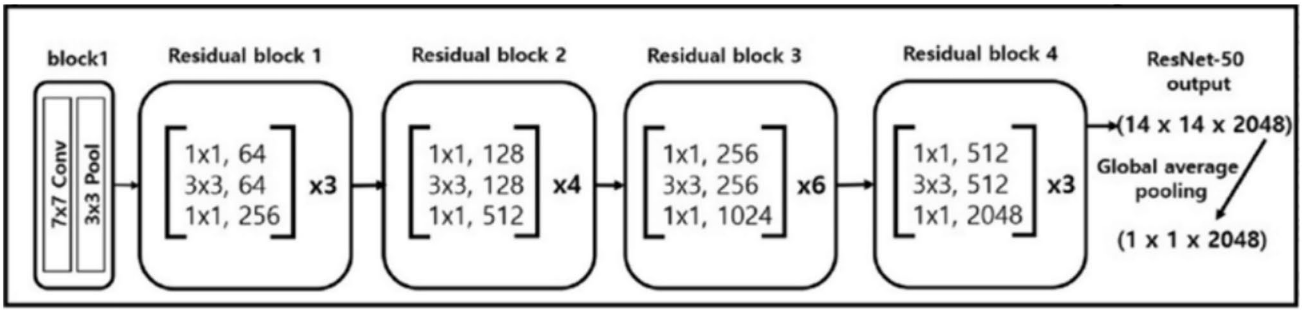

Fig. 1 General structure of the ResNet-50

- The first convolutional layer operating on 3 input feature maps representing the corresponding RGB colour channels to obtain 64 output feature maps. This and all the successive convolutional layers are always followed by a normalization and an activation ReLU layer.
- Max pooling layer.
- In the first residual block (repeated 3 times to produce 9 convolutional layers), a convolution is done followed by a similar one.
- In the second residual block, a convolutional layer producing feature maps is again followed by two successive convolutions. These three layers are repeated four times to perform a total of 12 convolutions.
- The third residual block is performed six times, resulting in 18 layers.
- in the last residual block, repeated three times, convolutions are performed obtaining 2048 output feature maps.
- Average pooling layer, followed by fully connected and softmax layers (for the classification of the various classes). In particular, the last one converts the features in probability values representing the network confidence in assigning an image to a specific class.

The K-means algorithm is used in combination with this network: its input corresponds to the output of the ResNet-50; in particular, the output of the last layer of the ResNet, except for the classification layer, is given as input to the K-means as a set of features. The agglomerative clustering can be divided into three sequential phases (Fig. 2).

- Assignment of the centroid role to k selected frames, where in this specific context $k=32$, since it is considered the most suitable value to adequately partition the frames in a likewise number of clusters, so that it is possible to extract a frame from a cluster in a pseudorandom way.
- Distance computation between vectors of all the frames' features and each centroid, to associate frames with the closest cluster. At the end, the centre of each cluster is calculated again through the mean among all frames; the computation is stopped when the error, defined as the mean of the absolute values of the differences between the centroids' vectors of the current and previous frame at each iteration, is below a predefined threshold or after a preset maximum number of iterations.

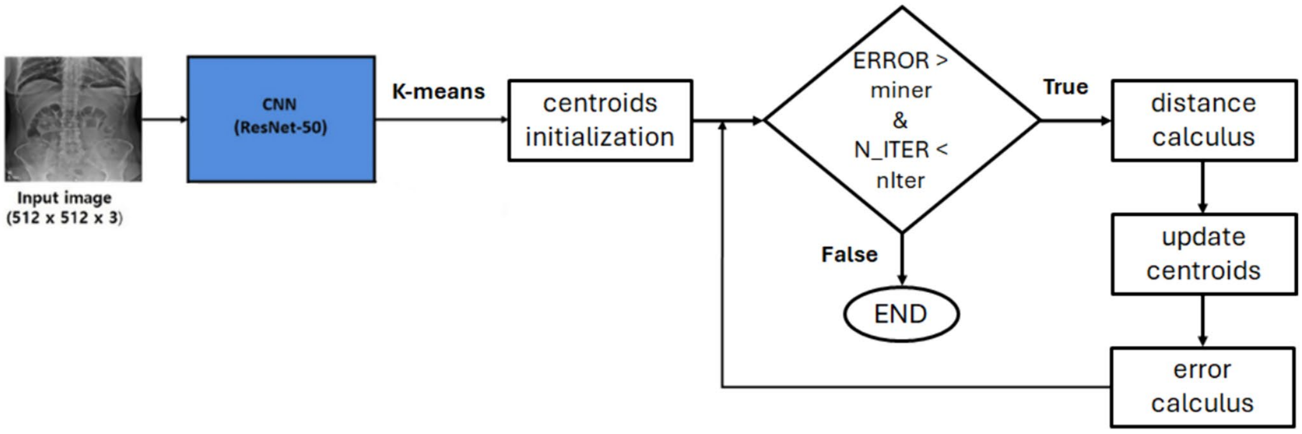

Fig. 2 Main sequential steps of the clustering procedure, through the application of the ResNet-50 and successively the K-means

- All the calculations needed to obtain the above-mentioned error.

The selected frames are grouped to create summary clips, each composed of 32 frames, to train ad hoc neural networks to classify COVID-19 hallmark features.

3 Key-frame selection algorithms

In this work, a series of key-frame selection algorithms has been developed and implemented, starting with a C++ serial version where the focus has been on frame extraction and consequently the creation of the new video, composed only of the most informative frames. Such implementation has been also optimized through a parallelization in CUDA language and then tested on several models of NVIDIA GPUs, considering both consumer boards and devices optimized for scientific computations.

The first step of the procedure is to extract all relevant frames from the ultrasound video; therefore, at first, all frames are considered and evaluated separately and after that, the most significant ones are incorporated into a new video of reduced dimensions, more specifically composed of 32 frames. To do this, the OpenCV library has been employed to efficiently examine the video to extract each frame: for normalization and readability purposes, a series of parameters are recorded for all frames, like the height and the width of each frame, as well as the total number of frames. Every one of them is read and pixel values are stored in a vector, followed by a conversion of the colour parameters into a grey scale.

Once all frames have been individually extracted and stored in their corresponding vectors, it is possible to proceed to the application of the key-frame selection algorithms.

3.1 Histogram

Each frame is evaluated pixelwise; a conditional control is performed on every single intensity value to determine whether it is an even or an odd number: in the first case, its value is added to the bin corresponding to the following formula, otherwise such value is decremented by a unit:

$$\text{hist1}: j + i \times \text{nbin}, \tag{1}$$

$$\text{hist1}: j + (i - 1) \times \text{nbin}, \tag{2}$$

where i and j are the coordinates of each pixel. After all pixels of the entire video have been evaluated, a histogram for every frame is built and it is used to calculate the score parameter as the sum of the differences between consecutive histograms in the sequence. For each couple of consecutive histograms, their maximum value is evaluated and then a normalization procedure is performed to adjust the original values in the new range [0;1]; in this way, the difference between the two histograms per bin is obtained. The *score* value is then updated by adding the previously calculated difference to the intermediate parameter. The successive step involves the computation of the threshold (that is the sum of mean and standard deviation values of the score vector), the distances listing following a descending order and the final selection of the 32 most informative frames, representing the ones showing the highest distances.

3.2 Entropy

This method starts with calculating the probability of occurrence for the diverse pixels' intensity levels inside each frame. To obtain such a result, a probability distribution for all pixels' values is created, in a similar fashion to the previous approach based on the histogram. In fact, once all such distributions are present, the entropy score, which is based on the differences in entropy among consecutive frames, is calculated. In addition, for all non-null probability values, the base 2 logarithm is computed to determine the entropy value.

The next step is to calculate the entropy score between consecutive frames: the absolute difference between the current and the previous frame is computed.

In the same way as the histogram algorithm, in this phase the threshold, the descending listing of the distances vector and the consequent selection of the most distant frames are processed.

3.3 ResNet-50 combined with K-means

The third method examines each extracted frame through a deep learning algorithm, the ResNet-50; to do so, an RGB tensor has to be initialized using PyTorch CPU C library to contain the data coming from the frame. For this application, it is necessary to use the 32-bit floating point representation and also to normalize the values in the [0;1] range. Moreover, to simulate an RGB image, the concatenation of three copies of the tensor along the first dimension is applied. The ResNet-50 implements a convolution and a normalization, followed by a ReLU activation function and a 2D max pooling function on the input data, by specifying a pooling window, a stride and a padding value.

Initially, appropriate weights are loaded for each convolutional layer, where a series of other fundamental parameters are also stored: the number of input and output channels, the kernel size, stride, padding, as well as the bias flag which is initialized to "false" to indicate that the corresponding layer has no bias.

After that, the layer is set on inference mode to deactivate the gradients' update, since no backpropagation is done while in inference mode. A tensor to incorporate the weights and a vector for the weights read from an input file are created.

The same procedure is performed for the normalization layers: while in the previous case only one tensor was necessary, here four different tensors are used, representing mean, standard deviation, bias and scale terms.

At this point, the frame processing is ended, and the output features are derived and can be stored into a tensor in output from the last layer of the ResNet-50.

The key-means approach makes use of this output; in particular, a series of parameters need to be initialized: the number of clusters, the error threshold employed to determine whether the algorithm converges or must be interrupted and also the maximum number of iterations allowed. This last parameter is set to 32, since this is the quantity of frames needed.

The algorithm begins by initializing the centroids through a pseudo-random approach; each frame is then assigned to a centroid, at the same time avoiding associating different frames to the same or adjacent centroid, compared to the previous iteration. This is done because adjacent frames could be identical, thus generating errors in the successive clustering phase. The output is the cluster ID, the number of frames belonging to such cluster and the pointer where the cluster's features are stored.

The iterative part of the algorithm ends either when the error goes below the predefined threshold or the maximum number of iterations has been reached. For each iteration, the Euclidean distance between the centroids and the frames' features is computed, with the consequent association (or reassociation) of each frame with the closest centroid.

Based on this calculus and on the frames assigned to the various clusters, new centroids are computed: more specifically, if the frame's cluster coincides with the cluster considered, a counter is incremented to quantify the number of frames per cluster. A similar procedure is exploited to determine the mean value of all clusters' features, so that the new centroids can be obtained; however, data concerning the previous centroids are temporarily kept. Finally, the error between the previous and the current centroids is calculated and then summed to obtain the total Euclidean distance. The mean error is given by the ratio between the total Euclidean distance and the product of number of clusters and number of features' dimensions.

The output of the K-means is represented by the indices of the frames selected for each cluster: the first 32 elements will be the most significant frames.

For all the three algorithms, once the 32 most relevant frames have been identified, they need to be incorporated into a new video.

4 GPU parallelization through CUDA programming language

To accelerate processing speed and reduce storage for the classification, a series of parallel and distributed-based approaches have been proposed. NVIDIA's Compute Unified Device Architecture (CUDA) is a hardware–software architecture for the development of massively parallel programs with GPUs, providing a development environment using C programming language. Threads run on streaming processors, each executing the same portion of code in single-instruction–multiple-thread fashion [39].

GPUs can be understood as an array of independent processors, where each of them corresponds to an independent thread of execution. The parallelism that can be inferred in this architecture totally differs from the FPGA architecture parallelism [26]. Hence, the parallelism inferred in the GPU can be efficiently exploited when there is a situation in which a set of operations is to be independently carried out on many different elements of a dataset [40].

The three algorithms considered undergo a parallelization using GPU's proprietary development environment and language, CUDA: all of them follow the same general procedure, although with important differences, especially concerning the ResNet-50 with the K-means methodology.

Data regarding each frame are stored in the device memory, where the histograms or the entropies are calculated for all frames and successively also their scores. Therefore, the parallelization here affects the bins population.

Through a series of *cuBLAS* (CUDA Basic Linear Algebra Subprograms) functions, it is possible to accelerate the execution of specific computations in a more compact and optimized way; in particular, it is applied to determine the histogram score.

In fact, both the histogram and the entropy algorithms perform their calculation through ad hoc CUDA kernels specifically designed to best exploit the GPU's characteristics and thus optimize calculations. Once this portion is completed, the reciprocal values and successively the score are determined using functions of the *cuBLAS* library, since they are highly optimized for arithmetic operations that represent the best fit for this application. As a notable example, since the absolute differences are all positive values, their sum can be determined efficiently by means of the *cublaSasum* function.

When approaching the parallelization of the ResNet-50, the first step is to create a 3D tensor with three channels using the *cuDNN* (CUDA Deep Neural Network) library, since its functions are tailored for operations involving tensors and neural networks; as a consequence, here their extensive use will be made. The configured tensor is used to store data from each frame in the format required by the neural network; hence also a descriptor is needed (provided with the ad hoc *cuDNN* function *cudnnCreateTensorDescriptor*) and equipped with its corresponding parameters: data format, data type, dimensions and information peculiar of the specific resource. The next step is to convert the data format into the floating point and then to perform the normalization when all the proper network computation can start. The complexity of the algorithm, particularly the high number of layers, requires performing a series of functional steps for each layer; in addition, for every block of code, attention needs to be paid to correctly set the network's layer parameters.

The majority of the computations can be provided with *cuBLAS* and *cuDNN* functions that show optimized performance compared to designing ad hoc CUDA kernels. These steps are necessary before performing each convolution, because the output's dimensions can vary depending on the convolution's parameters (number of filters, kernel dimensions and so on).

For each layer of the neural network, a convolution and a normalization function are performed; a notable feature of the *cuDNN* library is the opportunity to provide multi-operation fusion patterns useful for further optimizations, allowing the user to express a computation by defining an operation graph and not selecting from a fixed set of API calls. In fact, *cuDNN* has been designed to accelerate deep neural networks supporting functions such as convolutions, pooling, normalizations and activations. *cuBLAS* routines instead are optimized for vectorial and matrixial linear algebra operations that can be performed in floating point, double precision or even with complex numbers.

They have been implemented in this work through the use of appropriate *cuDNN* sublibraries, in particular, the *cudnn_graph* library (for example, the CUDNN_CONVOLUTION_FWD_ALGO_GEMM in the convolutional layer function). At the end of the 50th layer, the examined frame undergoes the K-means algorithm.

The parallelization is not employed for the centroids' initialization, since it is best exploited in the clustering portion of the algorithm, where the various intermediate computations can be efficiently done using *cuBLAS* functions. This same library is fundamental for the successive portion of the algorithm, where the centroids' processing and the error calculation are performed.

5 Results

To evaluate performance in terms of both disease detection and execution time for all the three methods previously presented, a series of tests was performed on ultrasound videos of different dimensions and on different architectures. In particular:

- the CPU is an Intel Core i7 processor with 2.90 GHz working frequency, Windows 10 64-bits OS and 16 GB of RAM memory;
- two different models of GPUs, more specifically the NVIDIA GeForce GT 1030 (Pascal architecture) and the RTX 4090 (Ada Lovelace architecture), which are both developed for advanced computer graphics but belong to diverse performance ranges;
- a university cluster composed of four processing units, or so-called nodes, equipped with NVIDIA A16 GPUs (Ampere architecture), as well as FPGAs and CPUs; however only the GPUs were employed in this work since they show the best performance for this specific application, due to their being optimized for scientific computing.

The CPU processor was adopted not only for the simulations, but also for the tests involving the GeForce GT 1030 GPU model, where the CUDA environment 11.4 has been installed together with the *cuBLAS* and *cuDNN* libraries.

The processor employed with the GeForce RTX 4090 is an Intel Core i9, 13th generation equipped with 64 GB of RAM, Windows 11 64-bits OS and CUDA 12.0 environment.

Only a single node of the university cluster was used. It is composed of two Intel Xeon Silver processors, each with 16 nodes, working at a frequency of 2.4 GHz and with 768 GB of RAM, three NVIDIA A16 GPUs and eight TB of memory mounted on an SSD NVMe.

Visual Studio 2019 and 2022 were adopted to compile both the serial and parallel versions, since the CUDA development environment is optimized for its graphical interface.

The dataset is composed of 246 ultrasound videos of two main typologies: the first is made of 334 frames of 600×800 pixels and the second has 120 frames with 864×1152 pixels.

A possible limitation of this work is due to the need of keeping a limited number of frames, since this could confine the detection and classification capabilities of the algorithm. However, this work could pave the way for a better understanding of the most important frames in an LUS video and thus help also in identifying better clusters of frames, even in terms of numerosity.

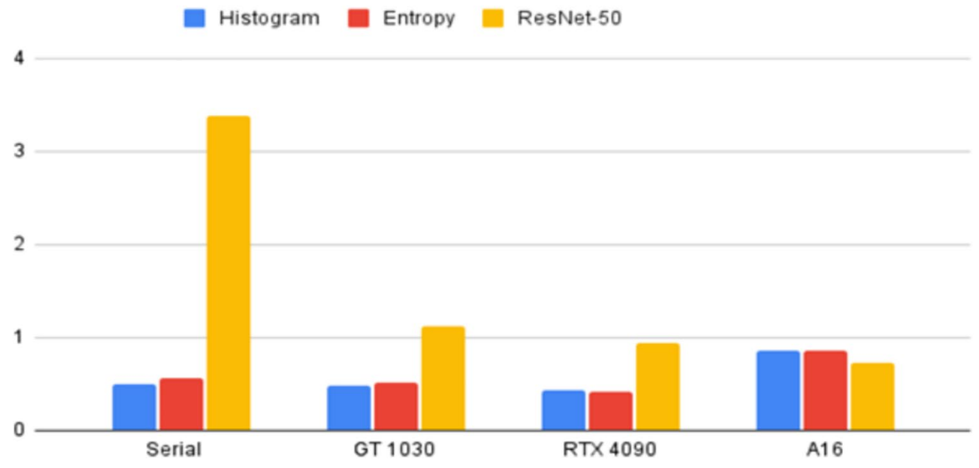

Fig. 3 Comparison of execution times for a single frame of the video, in logarithmic scale

The most significant contributions in speeding up the process and move closer to real-time performance are those of the various algorithms, while inputs acquisition and output visualization do not represent relevant percentages. Moreover, processing time is not influenced by each frame's resolution, but only by the number of frames in the entire video, in particular in the case of ResNet-50 in combination with K-means. Figure 3 shows the results in logarithmic scale for one of the videos of bigger dimensions, where it is possible to observe the best results in terms of speedup after the parallelization, as well as a convergence of the algorithm after only three iterations.

In fact, in the cases of histogram and entropy, even though the algorithms' times undergo a reduction once the CUDA parallelization is done, it is quite small compared to the third method.

In addition, considering the different GPU architectures, the GeForce 4090 shows better results compared to the GeForce 1030 model, as suggested by their different configurations and performance designs.

Similar considerations can be made when observing a single frame and successively the entire video using the ResNet-50 + K-means approach. Here, the speed of the execution is reduced dramatically, shrinking from some days in the case of the serial version run on the CPU to a few minutes employing the GPUs; in particular, the best performance is given by the RTX 4090.

The cluster represents the only difference in this context, since independently of the dataset used, the performance is worse for the histogram and for the entropy calculus rather than the ResNet-50 with the K-means. This is due to a series of factors: the parallelized code can only exploit 25% of the total computing power, since only the GPU A16 is equipped with the quad-GPU architecture. Moreover, the hardware configuration is different from the desktop GPUs evaluated and it is a rigid architecture that cannot be adapted to specific needs of certain applications (that may benefit more from other kinds of architectures in terms of optimization and thus performance). The third approach was able to provide the best optimization and the quickest execution of just about 10 min (Table 1).

Table 1 execution time [s] to process the entire video, composed of 120 frames

	Histogram	Entropy	ResNet-50	K-means (single iteration)
Serial C code	3.150	3.663	286,581.720	1.826
GT 1030	3.093	3.228	1616.400	0.178
RTX 4090	2.704	2.621	1049.460	0.176
A16	7.268	7.212	634.320	0.070

It is worth noticing that since video durations vary between 4 and 9 s, histogram and entropy algorithms are performed in real time. Considering the combination of ResNet-50 and K-means, a serial processing takes too much time to summarize the video, making the final diagnosis too slow. On the other hand, the parallelization of this approach on the A16 GPU reduces the processing time to about 10 min, which is compatible with clinical diagnostic procedures. Therefore, given such application, where a complete diagnosis scoring the severity of the infection takes several minutes and requires the evaluation by an expert medical doctor, the parallelization allows to consider this specific case as real-time compliant.

6 Discussion

Considering with more detail the performance of the ResNet-50, multiple experiments were conducted specifically on the convolutional layer with the aim of analysing the behaviour of the total channels with regard to the kernels' dimensions. The presented models have been trained in [13], where overfitting was prevented by training the network for the ResNet-based key-frame selection with images coming from a dataset different to the one considered for the performance metrics. In this way, a further training is not necessary. Moreover, the classification metrics of [13] were maintained, due to the fact that no modifications were made to the original methods and the authors validated the parallel algorithms against the original ones.

The results suggest that the stride and the padding do not significantly influence the execution time of the convolution step. It emerged that its duration depends not on the number of a single type of channel (either input or output), but only on their overall number (both input and output ones). This leads to the conclusion that the two main factors influencing the execution time are the kernel dimensions and the total number of channels (as shown in the following scatter plots in Fig. 4).

In fact, the execution time increases proportionally with regard to the number of channels and the kernel dimensions, but more specifically there is a difference of two orders of magnitude between the serial and the parallel implementations. Moreover, the first convolutional layer is always the most computationally demanding, even compared to the other convolutional layers of the ResNet in use, since it requires to firstly initialize the descriptors and allocate the memory, while the subsequent layers do not need this step to be repeated. Lastly, the normalization layer was evaluated compared to the number of input filters (Table 2). In the serial implementation, the execution times increase in a directly proportional way with the rise in the number of input channels; instead, employing the parallelized version using the GPU, after a limited transient time interval, the trend is similar but with a much smaller increase, as clearly visible from Fig. 5.

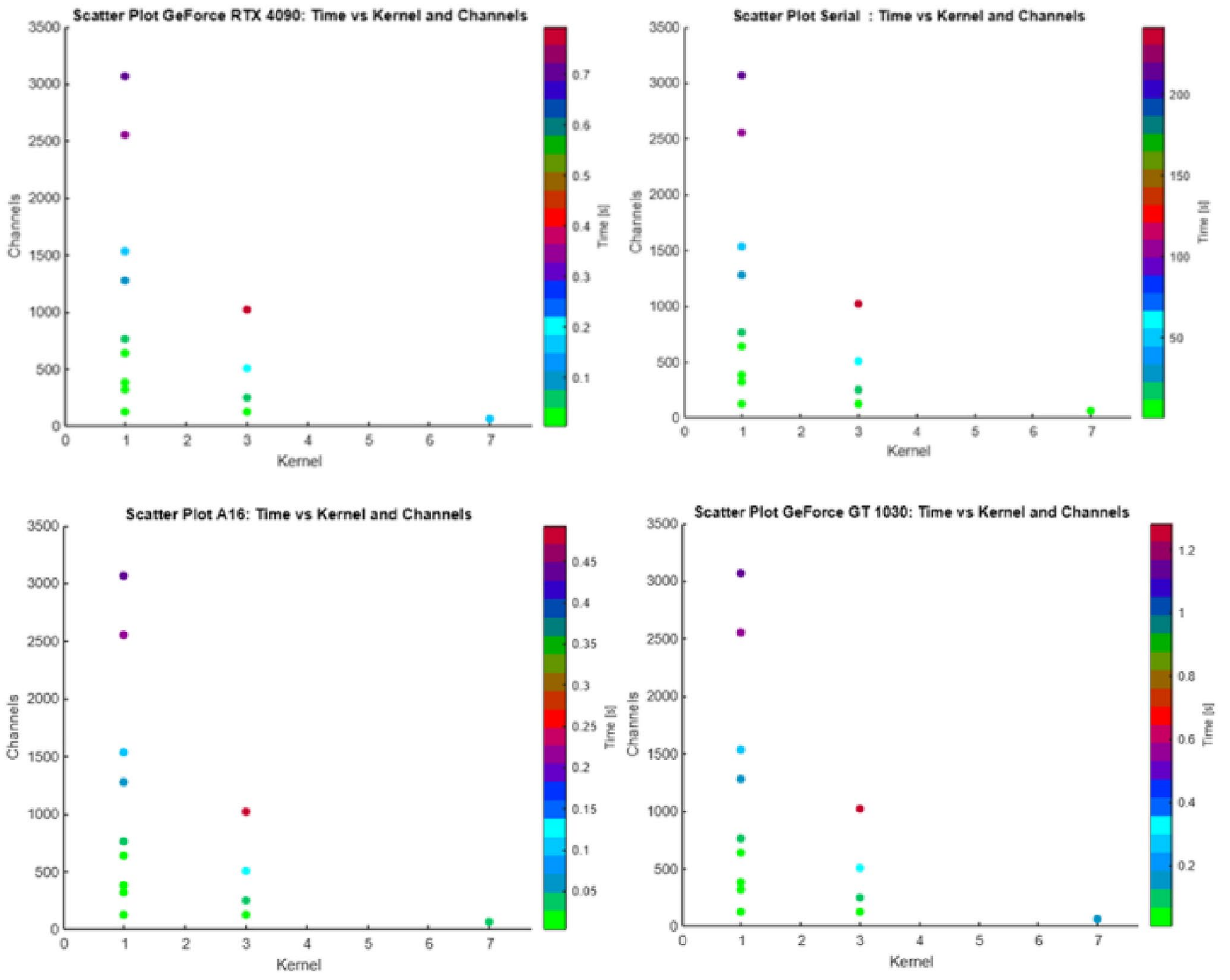

Fig. 4 Scatter plot of all the implementations

This is due to the usage, in this work, of only those layers with the same number of channels available in a traditional ResNet-50 and the mean value of their execution time was calculated. Consequently, the weight of the normalization layer is higher when a smaller number of channels is present, thus slowing the execution (similarly to what happens in the first convolutional layer of the network).

In addition, an important consideration must be made also concerning the diagnosing ability of the three approaches evaluated. In fact, they have been tested to detect the presence of disease, as well as three progressively worsening degrees. This is very relevant when considering the most appropriate drug and its dose that need to be tailored to the specific patient. With this aim, four scores were defined (no disease, mild, severe, dangerous) and the performance of a deep network proposed in [13] for this classification problem was measured giving as input to the network the summarized videos obtained by each approach and the best results were achieved by the ResNet + K-means.

Table 3 presents the values for the three methods concerning binary, three- and four-way disease classification: the results clearly show that the third approach is the most suitable, not only in terms of computational performance but also of diagnostic purposes. This is true and the deep

Table 2 Execution times [s] of the normalization layer batch

Number of filters	Serial code	NVIDIA GeForce GT 1030	NVIDIA GeForce RTX 4090	NVIDIA A16
64	0.025285	0.003285	0.003428	0.002424
128	0.038500	0.002375	0.001625	0.001931
256	0.068250	0.004062	0.003625	0.002103
512	0.122636	0.003818	0.002636	0.002295
1024	0.252428	0.004857	0.003285	0.002562
2048	0.481000	0.005750	0.004000	0.003223

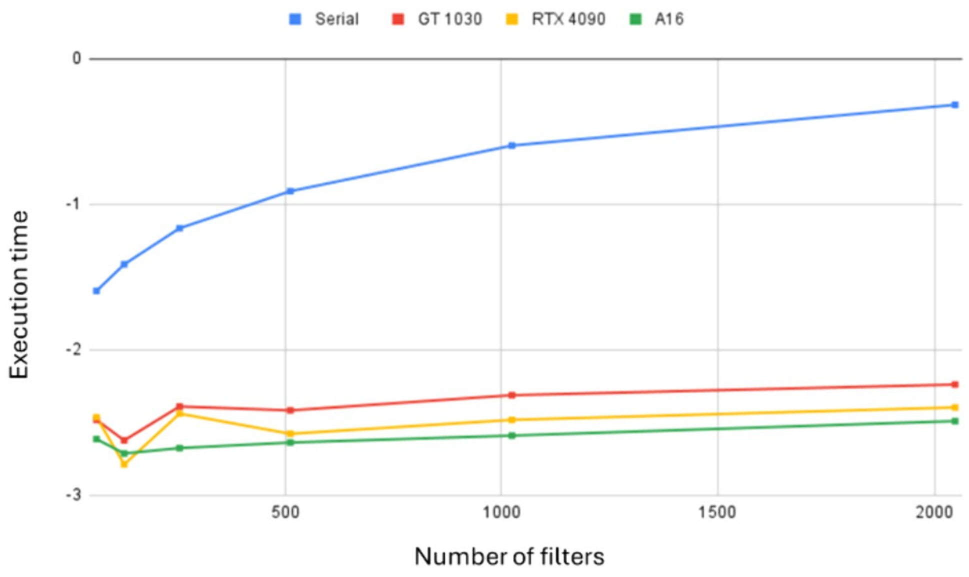

Fig. 5 Normalization layer execution times in logarithmic scale against the number of channels

Table 3 Detection accuracy and precision considering binary, three- and four-way disease classification for the three key-frame selection methods [13]

Key-frame selection method	Metric	Binary classification	Three-way classification	Four-way classification
entropy	Accuracy	0.773	0.739	0.687
	Precision	0.773	0.739	0.687
histogram	Accuracy	0.770	0.845	0.654
	Precision	0.770	0.845	0.654
ResNet + K-means	Accuracy	0.862	0.843	0.743
	Precision	0.862	0.843	0.743

learning solution outperforms the other ones, especially when considering the four-way classification, where accurately estimating the severity of the disease is crucial for medical personnel in the identification of the most appropriate and personalized treatment for each patient. Moreover, when the goal is solely to detect the presence of the pathology, the binary classification could be employed and the same methodology increases its accuracy and precision of a ten factor, compared to the entropy and the histogram.

7 Conclusions

In this work, three key-frame selection approaches were proposed and evaluated to detect the most informative frames, which are then extrapolated to create a new and reduced version of the original video. This sequence of frames is less computationally demanding and represents the input to a classification network to discriminate between healthy subjects and patients.

In particular, the parallelization procedure using NVIDIA GPUs allowed to guarantee the same level of accuracy in the detection phase, at the same time significantly reducing their execution, and thus leading to a faster diagnosis. Therefore, based on the obtained results, the ResNet-50 deep learning approach proved to be the most suitable solution concerning the combination of accuracy of detection and computational demands, even compared with more consolidated methodologies. Moreover, the aim to evaluate different high-performance computing solutions for computational sustainability purposes shows that the best performance in terms of execution times is achieved through the implementation using a GPU cluster, still providing the same output compared to the other accelerating architectures considered that make use of both serial and parallel processing. With respect to the capabilities of single desktop GPUs, in a context where precise and accurate diagnosis is fundamental, the opportunity to exploit the available resources in a cluster represents an additional advantage.

Moreover, the classification capability of the three approaches was evaluated considering four scores that correspond to the healthy condition and three progressively worsening degrees. The combination of ResNet and K-means outperforms the results obtained with the entropy and the histogram, laying the foundations for a better identification of personalized treatments, and at the same time considerably improving the accuracy in detecting the disease.

As further developments, the key-frames selection methodologies can be further optimized and new approaches also could be envisaged to enlarge the representativeness of the dataset; in addition, high-performance computing approaches like GPU parallelizations and implementations, either on desktop computers or in clusters with adequate architectures that can best exploit the algorithms' performance, could produce fast, precise and consistent results. The next step will be the introduction of an extended set of pulmonary diseases to be evaluated with this technique, since all interstitial pathologies could greatly benefit from this innovative approach. This could also consolidate the clinical validity of such a method and bring a new platform for diagnosis and monitoring.

Author contributions All authors wrote and reviewed the manuscript. ET and EM performed formal analysis and data curation. ET and MG performed the conceptualization, the methodology and the validation. FL supervised the work.

Funding Open access funding provided by Università degli Studi di Pavia within the CRUI-CARE Agreement.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Lamers, M.M., Haagmans, B.L.: SARS-CoV-2 pathogenesis. Nat. Rev. Microbiol. **20**, 270–284 (2022). https://doi.org/10.1038/s41579-022-00713-0
2. Velavan, T.P., Meyer, C.G.: The COVID-19 epidemic. Trop. Med. Int. Health **25**, 278–280 (2020). https://doi.org/10.1111/tmi.13383
3. Zhao, W., Zhong, Z., Xie, X., Yu, Q., Liu, J.: Relation between chest CT findings and clinical conditions of coronavirus disease (COVID-19) pneumonia: a multicenter study. Am. J. Roentgenol. **214**, 1072–1077 (2020). https://doi.org/10.2214/AJR.20.22976
4. Liguoro, I., Pilotto, C., Bonanni, M., Ferrari, M.E., Pusiol, A., Nocerino, A., Vidal, E., Cogo, P.: SARS-COV-2 infection in children and newborns: a systematic review. Eur. J. Pediatr. **179**, 1029–1046 (2020). https://doi.org/10.1007/s00431-020-03684-7
5. Yoon, S.H., Lee, K.H., Kim, J.Y., Lee, Y.K., Ko, H., Kim, K.H., Park, C.M., Kim, Y.-H.: Chest radiographic and CT findings of the 2019 novel coronavirus disease (COVID-19): analysis of nine patients treated in Korea. Korean J. Radiol. **21**, 494 (2020). https://doi.org/10.3348/kjr.2020.0132

6. Volpicelli, G., Elbarbary, M., Blaivas, M., Lichtenstein, D.A., Mathis, G., Kirkpatrick, A.W., Melniker, L., Gargani, L., Noble, V.E., Via, G., Dean, A., Tsung, J.W., Soldati, G., Copetti, R., Bouhemad, B., Reissig, A., Agricola, E., Rouby, J.-J., Arbelot, C., Liteplo, A., Sargsyan, A., Silva, F., Hoppmann, R., Breitkreutz, R., Seibel, A., Neri, L., Storti, E., Petrovic, T.: International evidence-based recommendations for point-of-care lung ultrasound. Intensive Care Med. **38**, 577–591 (2012). https://doi.org/10.1007/s00134-012-2513-4
7. Long, L., Zhao, H.-T., Zhang, Z.-Y., Wang, G.-Y., Zhao, H.-L.: Lung ultrasound for the diagnosis of pneumonia in adults. Medicine **96**, e5713 (2017). https://doi.org/10.1097/MD.0000000000005713
8. Lichtenstein, D., Goldstein, I., Mourgeon, E., Cluzel, P., Grenier, P., Rouby, J.-J.: Comparative diagnostic performances of auscultation, chest radiography, and lung ultrasonography in acute respiratory distress syndrome. Anesthesiology **100**, 9–15 (2004). https://doi.org/10.1097/00000542-200401000-00006
9. Lu, W., Zhang, S., Chen, B., Chen, J., Xian, J., Lin, Y., Shan, H., Su, Z.Z.: A clinical study of noninvasive assessment of lung lesions in patients with coronavirus disease-19 (COVID-19) by bedside ultrasound. Ultraschall Med. Eur. J. Ultrasound. **41**, 300–307 (2020). https://doi.org/10.1055/a-1154-8795
10. Shoeibi, A., Khodatars, M., Jafari, M., Ghassemi, N., Sadeghi, D., Moridian, P., Khadem, A., Alizadehsani, R., Hussain, S., Zare, A., Sani, Z.A., Khozeimeh, F., Nahavandi, S., Acharya, U.R., Gorriz, J.M.: Automated detection and forecasting of COVID-19 using deep learning techniques: a review. Neurocomputing **577**, 127317 (2024). https://doi.org/10.1016/j.neucom.2024.127317
11. La Salvia, M., Secco, G., Torti, E., Florimbi, G., Guido, L., Lago, P., Salinaro, F., Perlini, S., Leporati, F.: Deep learning and lung ultrasound for Covid-19 pneumonia detection and severity classification. Comput. Biol. Med. **136**, 104742 (2021). https://doi.org/10.1016/j.compbiomed.2021.104742
12. Manoj Kumar, M.V., Atalla, S., Almuraqab, N., Moonesar, I.A.: Detection of COVID-19 using deep learning techniques and cost effectiveness evaluation: a survey. Front. Artif. Intell. (2022). https://doi.org/10.3389/frai.2022.912022
13. Gazzoni, M., La Salvia, M., Torti, E., Secco, G., Perlini, S., Leporati, F.: Perceptive SARS-CoV-2 end-to-end ultrasound video classification through X3D and key-frames selection. Bioengineering **10**, 282 (2023). https://doi.org/10.3390/bioengineering10030282
14. Khan, A., Khan, S.H., Saif, M., Batool, A., Sohail, A., Waleed Khan, M.: A survey of deep learning techniques for the analysis of COVID-19 and their usability for detecting omicron. J. Exp. Theor. Artif. Intell. (2023). https://doi.org/10.1080/0952813X.2023.2165724
15. Erfanian Ebadi, S., Krishnaswamy, D., Bolouri, S.E.S., Zonoobi, D., Greiner, R., Meuser-Herr, N., Jaremko, J.L., Kapur, J., Noga, M., Punithakumar, K.: Automated detection of pneumonia in lung ultrasound using deep video classification for COVID-19. Inform. Med. Unlock. **25**, 100687 (2021). https://doi.org/10.1016/j.imu.2021.100687
16. Barros, B., Lacerda, P., Albuquerque, C., Conci, A.: Pulmonary COVID-19: learning spatiotemporal features combining CNN and LSTM networks for lung ultrasound video classification. Sensors. **21**, 5486 (2021). https://doi.org/10.3390/s21165486
17. Al Rahhal, M.M., Bazi, Y., Jomaa, R.M., Zuair, M., Melgani, F.: Contrasting EfficientNet, ViT, and gMLP for COVID-19 detection in ultrasound imagery. J. Pers. Med. **12**, 1707 (2022). https://doi.org/10.3390/jpm12101707
18. Huang, R., Ying, Q., Lin, Z., Zheng, Z., Tan, L., Tang, G., Zhang, Q., Luo, M., Yi, X., Liu, P., Pan, W., Wu, J., Luo, B., Ni, D.: Extracting keyframes of breast ultrasound video using deep reinforcement learning. Med. Image Anal. **80**, 102490 (2022). https://doi.org/10.1016/j.media.2022.102490
19. Dhane, D.M., Deokar, C.S.: Key frame abstraction, extraction, and browsing of echocardiogram videos. In: 2010 International Conference on Industrial Electronics, Control and Robotics. pp. 220–224. IEEE (2010)
20. Sharma, V., Sasmal, P., Bhuyan, M.K., Das, P.K., Iwahori, Y., Kasugai, K.: A multi-scale attention framework for automated polyp localization and keyframe extraction from colonoscopy videos. IEEE Trans. Autom. Sci. Eng. (2024). https://doi.org/10.1109/TASE.2023.3315518
21. Ma, M., Mei, S., Wan, S., Wang, Z., Ge, Z., Lam, V., Feng, D.: Keyframe extraction from laparoscopic videos via diverse and weighted dictionary selection. IEEE J. Biomed. Health Inform. **25**, 1686–1698 (2021). https://doi.org/10.1109/JBHI.2020.3019198
22. Schoeffmann, K., Del Fabro, M., Szkaliczki, T., Böszörmenyi, L., Keckstein, J.: Keyframe extraction in endoscopic video. Multimed Tools Appl. **74**, 11187–11206 (2015). https://doi.org/10.1007/s11042-014-2224-7
23. Muhammad, K., Sajjad, M., Lee, M.Y., Baik, S.W.: Efficient visual attention driven framework for key frames extraction from hysteroscopy videos. Biomed. Signal Process. Control **33**, 161–168 (2017). https://doi.org/10.1016/j.bspc.2016.11.011
24. Yu, H., Wang, H., Shi, Y., Xu, K., Yu, X., Cao, Y.: The segmentation of bones in pelvic CT images based on extraction of key frames. BMC Med. Imaging **18**, 18 (2018). https://doi.org/10.1186/s12880-018-0260-x
25. Marenzi, E., Torti, E., Leporati, F., Quevedo, E., Callicò, G.M.: Block matching super-resolution parallel GPU implementation for computational imaging. IEEE Trans. Consumer Electron. (2017). https://doi.org/10.1109/TCE.2017.015077
26. Marenzi, E., Torti, E., Danese, G., Leporati, F.: FPGA High Level Synthesis for the classification of skin tumors with hyperspectral images. In: 2022 11th Mediterranean Conference on Embedded Computing (MECO). pp. 1–4. IEEE (2022)
27. Marenzi, E., Carrus, A., Danese, G., Leporati, F., Callico, G.M.: Efficient parallelization of motion estimation for super-resolution. In: Proceedings—2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2017 (2017)
28. Ullah, U., Garcia-Zapirain, B.: Quantum machine learning revolution in healthcare: a systematic review of emerging perspectives and applications. IEEE Access. **12**, 11423–11450 (2024). https://doi.org/10.1109/ACCESS.2024.3353461
29. Zhou, H., Jagadeesan, J.: Real-time dense reconstruction of tissue surface from stereo optical video. IEEE Trans. Med. Imaging **39**, 400–412 (2020). https://doi.org/10.1109/TMI.2019.2927436
30. Gu, J., Qian, X., Zhang, Q., Zhang, H., Wu, F.: Unsupervised domain adaptation for Covid-19 classification based on balanced slice Wasserstein distance. Comput. Biol. Med. **164**, 107207 (2023). https://doi.org/10.1016/j.compbiomed.2023.107207
31. Raut, V., Gunjan, R.: Video summarization approaches in wireless capsule endoscopy: a review. E3S Web Conf. **170**, 03005 (2020). https://doi.org/10.1051/e3sconf/202017003005
32. Rodríguez-Moreno, I., Martínez-Otzeta, J.M., Sierra, B., Rodriguez, I., Jauregi, E.: Video Activity Recognition: State-of-the-Art. Sensors. **19**, 3160 (2019). https://doi.org/10.3390/s19143160
33. Sheena, C.V., Narayanan, N.K.: Key-frame extraction by analysis of histograms of video frames using statistical methods. Procedia Comput Sci. **70**, 36–40 (2015). https://doi.org/10.1016/j.procs.2015.10.021
34. Hermessi, H., Mourali, O., Zagrouba, E.: Multimodal medical image fusion review: theoretical background and recent advances. Signal Process. **183**, 108036 (2021). https://doi.org/10.1016/j.sigpro.2021.108036

35. Siddique, N., Paheding, S., Elkin, C.P., Devabhaktuni, V.: U-Net and its variants for medical image segmentation: a review of theory and applications. IEEE Access. **9**, 82031–82057 (2021). https://doi.org/10.1109/ACCESS.2021.3086020
36. Guo, Y., Xu, Q., Sun, S., Luo, X., Sbert, M.: Selecting video key frames based on relative entropy and the extreme studentized deviate test. Entropy **18**, 73 (2016). https://doi.org/10.3390/e18030073
37. Yang, S., Lin, X.: Key frame extraction using unsupervised clustering based on a statistical model. Tsinghua Sci Technol. **10**, 169–173 (2005). https://doi.org/10.1016/S1007-0214(05)70050-X
38. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. IEEE (2016)
39. NVIDIA CUDA Programming Guide 3.1
40. Torti, E., Marenzi, E., Danese, G., Plaza, A.J., Leporati, F.: Spatial-spectral feature extraction with local covariance matrix from hyperspectral images through hybrid parallelization. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **16**, 7412–7421 (2023). https://doi.org/10.1109/JSTARS.2023.3301721