The Journal of Supercomputing
https://doi.org/10.1007/s11227-024-06076-y

Edge and cloud computing approaches in the early diagnosis of skin cancer with attention-based vision transformer through hyperspectral imaging

Marco La Salvia[1] · **Emanuele Torti**[1] · **Elisa Marenzi**[1] · **Giovanni Danese**[1] · **Francesco Leporati**[1]

Accepted: 14 March 2024

Abstract

Hyperspectral imaging is applied in the medical field for automated diagnosis of diseases, especially cancer. Among the various classification algorithms, the most suitable ones are machine and deep learning techniques. In particular, Vision Transformers represent an innovative deep architecture to classify skin cancers through hyperspectral images. However, such methodologies are computationally intensive, requiring parallel solutions to ensure fast classification. In this paper, a parallel Vision Transformer is evaluated exploiting technologies in the context of Edge and Cloud Computing, envisioning portable instruments' development through the analysis of significant parameters, like processing times, power consumption and communication latency, where applicable. A low-power GPU, different models of desktop GPUs and a GPU for scientific computing were used. Cloud solutions show lower processing times, while Edge boards based on GPU feature the lowest energy consumption, thus resulting as the optimal choice regarding portable instrumentation with no compelling time constraints.

Keywords Skin cancer · Attention-based algorithm · Vision transformer · Hyperspectral images · Edge computing · Cloud computing

✉ Elisa Marenzi
elisa.marenzi@unipv.it

Marco La Salvia
marco.lasalvia01@universitadipavia.it

Emanuele Torti
emanuele.torti@unipv.it

Giovanni Danese
giovanni.danese@unipv.it

Francesco Leporati
francesco.leporati@unipv.it

[1] Custom Computing and Programmable Systems Laboratory, Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Via Ferrata 5, 27100 Pavia, Italy

1 Introduction

The healthcare research field requires hardware able to process huge amounts of data and mathematical models as fast as possible to provide prompt answers to aid medical diagnoses according to the highest standards [1]. Automated medical image diagnosis is one of the most flourishing disciplines of artificial intelligence (AI) in health care. Many specialties including radiology, ophthalmology and dermatology rely on image-based diagnoses through AI [1–5]. The main drawback of this approach is related to the processing power needed both to train the models and to perform the inference. On the other hand, AI algorithms are based on vector and matrix operations, which are intrinsically parallel. Thus, both the training and inference portions of AI models are typically performed using parallel architectures. In this context, high-performance computing (HPC) technologies such as multi- and many-core processors play a significant role [6, 7], where HPC is a broad term comprising a multitude of solutions that encompass both Edge and Cloud Computing approaches.

Cloud Computing can be defined as a model of service delivery and access where dynamically scalable and virtualized resources are provided as a service over the Internet [8]. In particular, computing power, data storage and services of Cloud systems are usually, but not always, outsourced to third parties and made available to both enterprises and customers [9]. On the other hand, Edge Computing is a paradigm where resources are located at the edge of the Internet, in close proximity to the devices that generate data (IoT and mobile devices, sensors and end users) or at the nearest edge (i.e., gateway, micro data center), and allow computation on downstream data on behalf of Cloud services and upstream data on behalf of IoT services [10–12]. They represent counterpoints to massive data centers that dominate in Cloud Computing. In fact, the term "Edge" defines any computing and network resource along the path between data sources and Cloud data centers, as a continuum.

These approaches led academic investigators and scientists to design so-called human-sensible applications for personalized medicine. In fact, the number of works on this subject greatly increased in recent years thanks to HPC advancements [2, 4]. Among the various HPC technologies, portable instruments represent a fundamental aid in the diagnostic process, oftentimes producing real-time results through specific computing devices, like FPGAs and GPUs [13, 14]. GPUs allow for faster training and inference times compared to CPUs, since GPUs host many cores that perform parallel computations. This is a critical issue when dealing with large datasets and complex models. Additionally, in medical contexts where the privacy of patients is of crucial importance a single-GPU or a multi-GPU system is better than Cloud clusters. In fact, one of the main advantages of GPUs is the ability to store and process data on-site, eliminating the need to transfer sensitive data to the Cloud. Furthermore, a local GPU system is often more cost-effective than Cloud clusters, especially when working with limited research budgets.

The need of HPC hardware is fundamental, since medical AI applications present several challenges [1, 4]. Clinical contexts usually provide poor datasets, making it hard to exploit elaborated deep learning (DL) algorithms. Moreover, the more complex the model, the greater the need to employ highly performant hardware. In addition, healthcare experts may give contrasting opinions regarding a diagnosis. Lastly, from a regulatory perspective, clinical AI systems must be certified before commercialization [1, 4, 5]. Consequently, hierarchically structured and standardized evaluations are mandatory to enable AI-based diagnosis. Therefore, this work targets approaches employing portable hardware, desktop NVIDIA GPUs and a cluster of GPUs, together with programming frameworks to accelerate the algorithms and enable the design of blueprints, to comply with the time-sensitive criteria required for industry translation.

Among the different imaging methods adopted in the medical field, hyperspectral imaging (HSI) gained relevance in the last decade thanks to its increased processing power which enabled faster analysis. HSI is a noninvasive, non-contact, non-ionizing and label-free technique, which measures the reflected and transmitted light, gathering relations values associated with several bands of the electromagnetic spectrum [7, 15]. Therefore, the outcome is a three-dimensional dataset comprising a two-dimensional image for each wavelength: this is named "hyperspectral cube," because it retains both the spatial and spectral information of the analyzed sample. Such cube contains the fraction of incident electromagnetic radiation reflected by a surface. Typically, the bands cover the range of both the visible and near-infrared wavelengths, allowing a precise discrimination of different materials [16, 17]. Each presents a specific variation of reflectance values concerning wavelengths, called spectral signature, allowing for precise discrimination [16, 18–21]. HSI classification focuses on the recognition of the material contained in each pixel and comprises several supervised and unsupervised algorithms whose elaboration could be computationally intensive [22, 23].

This feature of hyperspectral imaging has been adopted in the medical field to discriminate between healthy and pathological tissues [21]. Biochemical and morphological changes associated with lesions modify the optical characteristics of tissues, providing valuable information for automatic cancer detection. In fact, compared to other imaging techniques, hyperspectral cameras can capture larger areas and deliver more accurate results [7, 24]. Technological progress allowed their widespread use in many fields, making them popular, especially in medicine, for cancer detection [6, 18–20, 22, 25–27]. Several studies highlighted that tumor cells exhibit peculiar reflectance values and molecular spectral signatures [27–30]. In cancer research, the focus is on the light–matter physical interaction causing different reactions from each material or tissue to the beam radiation on its surface, thus allowing the discrimination between healthy and tumor tissue [27].

During the last decade, machine and deep learning (ML and DL, respectively) solutions emerged as tools to analyze and cluster different cancer types in HSIs [18, 31–35]. In particular, the literature focused on brain, skin, colon and esophageal cancer [7, 27, 28, 36].

More specifically, one of the most common types is skin cancer that is categorizable as non-melanoma skin cancer (NMSC) and melanoma skin cancer (MSC). NMSC is the fifth most common tumor worldwide [37]. Moreover, progression of melanocytes causes pigmented skin lesions (PSLs) classifiable as malignant or benign. Even atypical moles or dysplastic nevi are benign PSLs that are associated with an increased risk of evolving to melanoma. On the other hand, melanoma is one of the rarest skin cancers, but it has associated the highest mortality rate, due to the lack of early detection. Thus, research aims to provide AI solutions to strengthen current diagnostic performances [38]. The first adopted approach concerned typical ML methods such as artificial neural networks (ANNs), support vector machines (SVMs) and K-means clustering. More complex solutions combined different algorithms to enhance the detection quality. An example is given by the K-means clustering applied together with SVM classification [22]. However, in recent years, DL networks emerged as an ideal solution for end-to-end classification tasks, even though they are not yet adopted in the medical field [39]. Among the different DL methods, Vision Transformers (ViTs) have recently been proposed [40]. These networks rely on the self-attention mechanism, firstly designed for natural language processing (NLP) applications [15, 40]. More recently, ViT has been used to detect skin cancer [15, 41], in particular addressing the evaluation of the most appropriate algorithmic solution, without considering the development of the whole system.

This paper proposes different solutions to develop a portable system to detect skin cancer through HSI adopting ViT. Two main approaches are evaluated and compared, namely Edge and Cloud Computing. In this second case, also data transfer technology and latency have been discussed and taken into account. The evaluated solutions are then compared in terms of both processing times and power consumption.

The paper is organized as follows: Section II shows the main computing frameworks, in particular regarding Edge and Cloud Computing, while Section III presents the technologies and algorithms used in the specific case of skin cancer detection with HSIs, together with experimental results, followed by the conclusions.

2 Skin cancer detection through HSI and Vision Transformers

2.1 Related works

Skin cancer affects the body's largest organ. MSC develops from any cell capable of forming melanin and includes three subtypes: superficial extension, lentigo maligna and nodular tumor [24, 42–44]. Some present genetic modifications that, if left untreated, can spread over the body, potentially yielding metastases. NMSC lesions include more than 98% of the known skin lesions, of which 75–80% are basal cell carcinoma (BCC) and 15–20% are squamous cell carcinoma (SCC) [22, 24, 42, 45]. Therefore, distinguishing epidermal tumors into benign and malignant categories is fundamental.

Clinical dermatological diagnosis follows the so-called ABCDE rule, where A stands for asymmetry, B for border irregularity, C for color, D for diameter and E for evolution [29, 42, 46]. Nevertheless, this procedure introduces false positives; consequently, the gold standard is performing a biopsy. However, this process is painful, time-consuming, slow and expensive [24, 42, 46]. Hence, in this domain, the increase of skin cancers and the lack of expertise and innovative methodologies leads to the demand for systems based on HPC techniques making use of AI and novel optical technologies [24].

In this context, HSI systems aim at capturing information coming from chromophores (melanin and/or hemoglobin), which characterize epidermal lesions' spectral properties and vary among different skin lesions [22, 24].

AI literature also reports the presence of standard stages in most workflows for medical image processing (Fig. 1) [1, 2]. Present solutions differ mainly in the model of camera employed, its cost and weight, as well as on the materials and the existence of customized graphical user interfaces (GUIs).

The main methods used for skin cancer detection are ANNs and convolutional neural networks (CNNs). However, ANNs are not able to detect small lesions, compared to CNNs. This is because the convolution takes into account not only the spectral features but also the spatial domain [42, 46, 47]. Current AI algorithms are in the early stages of clinical application, but they can be scalable to multiple devices [1]. Modern solutions carry out a concurrent elaboration of the pixels to reduce execution time. For this level of classification, parallel technologies are suitable, since each processing core elaborates on a single or a group of pixels.

Concerning skin cancer, the output of these classification systems is typically a thematic map, where researchers assign a color to each pixel to represent a specific tissue type or lesion condition.

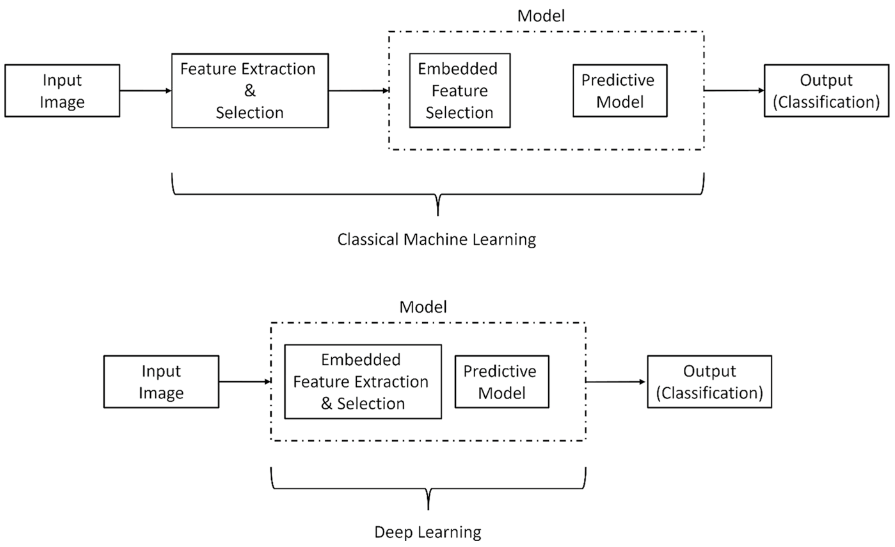

Fig. 1 The standard workflow for medical image processing through AI solutions. In the classical ML framework, feature extraction and selection are typically separated from the prediction performed by the model. During the inference process, the model can automatically pull out other features from the received ones. On the other hand, in the DL paradigm, feature extraction and selection are always embedded into the model, which automatically extracts and chooses the most suitable feature to perform the prediction. The cost is a more complicated model, resulting in higher processing times

2.2 Dataset of skin cancer hyperspectral images

Images of skin lesions of various body parts have been acquired from 116 subjects at the Doctor Negrin University Hospital and the Materno Infantil University Hospital Complex, both located in Las Palmas de Gran Canaria (Spain).

The database has been obtained via a custom solution [30]: a snapshot camera (Cubert UHD 185, Cubert GmbH, Ulm, Germany) capable of capturing the visual and near-infrared (VNIR) spectrum, with a spectral range from 450 to 950 nm, resulting in a spectral resolution of 8 nm and a spatial resolution of 50 × 50 pixels, whose pixel size was 240 × 240 μm^2. The camera has a Cinegon 1.9/10 lens (Schneider Optics Inc., Hauppauge, NY, USA) with a 10.4 nm focal length. The acquisition system employed a Dolan-Jenner halogen source light (Dolan-Jenner, Boxborough, MA, USA) with a 150 watts quartz–tungsten bulb lamp. A fiber optic ring light guides the camera to illuminate the skin surface with cold light, avoiding the high temperature of a halogen lamp on the subject's skin [30]. A dermoscopic lens with human skin refraction index in a 3D printed contact structure was attached to the system. This allows an HSI acquisition time of 250 ms [30]. Fifty-five images have been removed from the initial database because they derived from challenging areas (e.g., shoulders, nose, chin and other face parts) from the acquisition point of view, preventing total lens contact with the skin surface. Hence, the final database is made up of 76 images, 40 benign and 36 malignant skin lesions, belonging to 61 subjects [30, 48].

Pathologists and dermatologists diagnosed suspected malignant lesions through histological assessment to evaluate its etiology. Professionals clustered the images and assigned each pixel a specific label from one of the etiologies described in Section II. Images were also manually segmented, generating the ground truth. Each hyperspectral image has 50 × 50 × 125 hyperspectral cubes, and a single hypercube has a 50 × 50 ground truth. Figure 2 shows the ground truths of the entire database, with each type of label associated with a color: black for skin, yellow and red for malignant skin lesions, green and blue for benign skin lesions. 53% of them are benign, while 47% are malignant. The most prevalent tumors are the benign melanocytic (45%) and malignant epithelial ones (32%). The distribution of lesions within the database reflects the actual distribution in the population.

The dataset has a tree structure with two root nodes for, respectively, benign and malignant lesions. The root node then splits into melanocytic and epidermal tumors. This taxonomy represents a trade-off between other classification approaches, introduced as medically relevant, complete and well suited to ML classifiers [24]. The

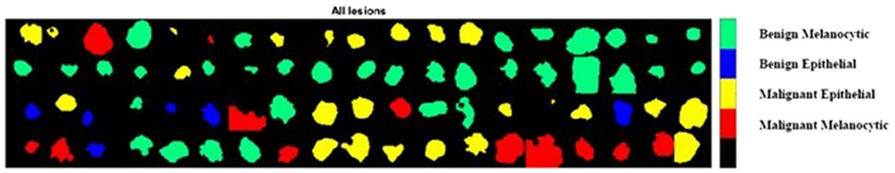

Fig. 2 Ground truth of the dataset

first validation uses the primary layer nodes and represents the broadest partition, whereas the children layers are disease classes sharing similar clinical treatment strategies. Pathologists assigned a category from the taxonomy proposed to each epidermal lesion, and they produced a mask highlighting tumor borders [22, 24, 30].

2.3 The vision transformer architecture

Among the different AI solutions proposed in the literature, Vision Transformers (ViTs) are a recent method which achieved interesting results in hyperspectral skin cancer classification [15]. It is based on the *attention mechanism*, aiming at the imitation of the human behavior of focusing on some parts of an image to make a decision. Before the images are given as input to the ViT, preprocessing and bands reduction steps are applied. The preprocessing aims at reducing the noise and standardizing the spectral signature. First, the image is calibrated according to Eq. (1):

$$CI = \frac{RI - DI}{WI - DI} \tag{1}$$

where DI is the calibrated image, RI is the raw one and WI and DI are the white and dark reference images, respectively. DI was acquired with the camera shutter closed and the room lights turned off, while WI was derived targeting a reference tile capable of reflecting 99% of the incident light. The first step is to remove the first four and the last five bands of the calibrated image, thus moving from 125 to 116 hyperspectral bands, since the sensor spectral response at these wavelengths is poor. To further reduce spectral noise, the remaining bands go through a smooth filter with a sample window of 5. The spectral values are then normalized in the range [0,1] with the min–max technique since [49] demonstrated that it ensures the best classification performance.

The bands reduction step aims at reducing the computational complexity of the ViT architecture. It relies on three convolutional blocks connected in cascade. Each of them is based on: 3×3 filters, normalization and ReLU activation function. This step further reduced the bands from 116 to 14, obtaining an image with sizes $50\times50\times14$, which represents the input to the ViT architecture. The aim is to extract high-level features and to reduce the training and inference times ensuring suitable classification performance.

The first operation of the ViT is the *patching*. It is based on the convolution operation and transforms the data shape from $H\times W\times C$ into $H\times P\times P\times C$ where H is the height, W is the width, C denotes the channels and $P\times P$ is the patch resolution. Thus, the number of patches given as input to the ViT is equal to $\mathrm{N}=\mathrm{HW}/\mathrm{P}^2$. Each patch is unrolled into a 1-D array and projected into a latent space by a Q–D array. After this projection, *position embedding* and *class token* are integrated into the patches to preserve the information of their original position. This new data is the input to the transformer encoder which is made up of three main components: multi-head self-attention (MSA), multilayer perceptron (MLP) and normalization.

The heart of the processing is performed by the MSA, since it implements the attention mechanism. It projects the input data to generate three vectors, namely

Key (K), Query (Q) and Value (V), by multiplying each patch by three matrices W_K, W_Q and W_V whose values are defined during the training phase. The projections are performed by eqs. (2) - (4):

$$K = X\,W_K \tag{2}$$

$$Q = X\,W_Q \tag{3}$$

$$K = X\,W_V \tag{4}$$

where X is a patch with $P \times P$ resolution. The K, Q and V vectors are used by the heads of the MSA to compute the attention maps. Each of the heads computes its attention map which is then concatenated with the outputs of the other heads to generate the input to the next stage. Each attention map is computed according to Eq. (5):

$$\text{attention map}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{5}$$

where $\sqrt{d}$ is a normalization term.

A ViT architecture includes L transformer encoders whose outputs are fed as input to an MLP which produces the final classification. The MLP inside each Transformer encoder is used to extract the high-level features from the attention maps, while the MLP outside the ViT combines the output of all the transformer encoder to produce the final classification.

The main steps of the whole architecture adopted for the classification are summarized in Fig. 3, while a pseudo-code in Table 1 shows the main steps of the algorithm.

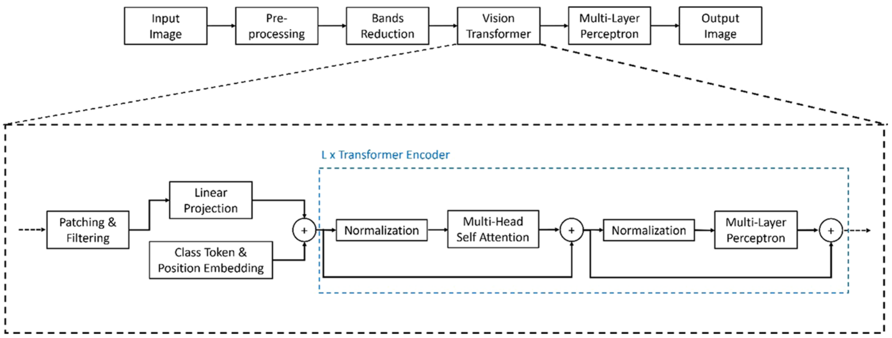

Fig. 3 Processing workflow adopted in this paper. The hyperspectral image is acquired and pre-processed. Then, the number of bands is reduced adopting three convolutional layers. The resulting image is the input to the Vision Transformer, whose operations are detailed in the dashed box. The output of the Vision Transformer is then given as input to a Multi-Layer Perceptron to produce the final classification

Table 1 Pseudo-code of the ViT methodology

```
Input: Image of size (H, W, C)

Parameters:
  - Patch size: P
  - Number of patches: N
  - Embedding dimension: DIM
  - Number of heads: H
  - Number of transformer encoder: L
  - Output classes: K

# Step 1: Image to Patches
patches = extract_patches(image, patch_size=P)

# Step 2: Linear Projection of Patches
embedded_patches = linear_projection(patches, output_dimension=d)

# Step 3: Positional Encoding
position_encoding = generate_positional_encoding(N, DIM)

# Step 4: Combine Embedded Patches and Positional Encoding
input_embeddings = embedded_patches + position_encoding

# Step 5: Transformer Encoder Blocks
for i in range(L):
    input_embeddings = transformer_encoder_block(input_embeddings, num_heads=H)

# Step 6: Global Average Pooling
global_avg_pooling = average_pooling(input_embeddings)

# Step 7: Fully Connected Layer
output_logits = fully_connected(global_avg_pooling, output_classes=K)

# Step 8: Softmax Activation
output_probabilities = softmax(output_logits)

Output: Class probabilities for each class
```

3 Cloud vs edge computing

The US National Institute for Standards and Technologies (NIST) defines Cloud Computing as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [50]. This definition comprehends five essential characteristics, three service models and four deployment models [51]. Concerning the main features of such solutions, these are: on-demand self-service access to computing resources, which include processing power, storage and virtual machines; broad network access from different devices like laptops or mobiles telephones; resource pooling among numerous clients (multi-tenure management); elasticity (resources scaling out and scaling back); and resources utilization monitoring (storage, CPU hours, bandwidth). Each Cloud provides different levels of abstraction.

Cloud services can be provided through three main models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In the SaaS model, the user accesses services from the web browser and does not have knowledge nor control of the underlying framework. PaaS allows control only over applications and correlated environment arrangements, providing the ability to deploy customized applications, produced using programming languages and tools supported by the provider. Users in IaaS models can manage processing, storage, networks and other computing resources and also deploy and run any software (even operating systems, services and applications). The client has control potentially over every aspect of the system.

There are four Cloud deployment models, varying in scalability, reliability, security and cost: private (used exclusively by a single user, offering increased security at higher costs), public (used by the general public), community (shared by two or more users for specific purposes, e.g., a university) and hybrid Cloud (a combination of the previous types)). Since public Cloud is potentially open to everyone, security issues could arise; however, it offers the advantages of a Cloud solution at the lowest costs. Examples of public Cloud vendors are Microsoft Azure, Google Cloud or Amazon AWS.

Cloud Computing could be a convenient alternative when moving applications from expensive internal data centers to economic and resourceful solutions. In fact, it comes with several advantages, among which scalability (that can be manual as well as automatic) and virtualization.

According to the International Data Corporation (IDC) prediction, by 2025 global data will reach 180 zettabytes (ZB) and 70% of those generated by IoT will be processed on the edge of the network [10, 52]. Therefore, the centralized processing model based on Cloud Computing would not be efficient enough. Edge Computing is a new paradigm in which its resources are at the edge of the Internet, on downstream data on behalf of Cloud services and upstream data on behalf of IoT services, in close proximity to devices, sensors and users [11]. It is characterized by low latency to support applications in the fields of augmented reality, public safety, smart manufacturing and health care. New challenges and opportunities arise as the consolidation of Cloud Computing meets the dispersion of Edge Computing. In Edge Computing, end devices are both data consumers and data producers. Devices can not only request service and content from the Cloud but they can also perform computing tasks from it. Computing offloading, data storage, caching, processing, distributing requests and delivering services from the Cloud are all activities that can be performed in an Edge Computing system. Hence, crucial requirements to be met are efficiency, security and privacy protection.

From this scenario, a three-tier Edge Computing model has been defined: IoT, Edge and Cloud [10]. Several communication protocols are used to connect IoT to the Edge tier, with low power consumption and short-distance features. For example, drones can connect to a cellular tower by 5G/4G/LTE, and sensors in the smart home can communicate with the home gateway through Wi-Fi. Instead, since this second layer requires Cloud Computing and storage capabilities, its protocols with the Cloud are characterized by a large throughput and a high speed. Ethernet, optical fibers and 5G are the preferred communication protocols between the Edge and the Cloud.

Edge Computing and Cloud Computing are complementary paradigms. The huge distribution of smart devices and rapid development of Cloud technologies have brought Edge Computing to the foreground.

Edge servers are both physically and logically closer to the end user compared to Cloud servers. As a consequence, even if the processing speed of an Edge server is slower, it will provide a better quality of service [53]. In fact, Edge Computing has several advantages: large amounts of temporary data are processed at the edge of the network, but only a portion is uploaded to the Cloud, which greatly reduces data center power consumption and the requirements in terms of network bandwidth. Moreover, this kind of data processing greatly reduces system latency and improves service response capability, since Cloud Computing resources are not involved. In addition, Edge Computing keeps sensitive information on the Edge devices, thus assuring better security and privacy [53].

Various technologies can be used to deploy an Edge or a Cloud Computing system. In certain cases, the same family of devices could be part of both system types; on the contrary, specific tools can be used in only one of the two contexts.

GPUs are many-core architectures that represent the dominant type of device for parallel computing purposes in both Edge Computing (as portable devices or desktop solutions) and Cloud Computing (as single elements of a bigger cluster). Many GPU architectures are produced, compared to CPUs. The latter maximize sequential programs' performance and distribute the workload maintaining a serial setting, while GPUs increase memory bandwidth, speeding up data and instruction transfers and maximizing computational throughput. To program GPUs, NVIDIA's proprietary language CUDA (Computer Unified Device Architecture) has been used.

The technological HPC growth drives also healthcare innovation, in particular for experimenting with new approaches for gathering, managing and transmitting data [6]. Two main factors are necessary to their continuous improvement, namely fast data availability and technological infrastructure. Multi-GPU systems or supercomputers are also employed to increase the number of available cores, thus reducing execution time. Fast or even real-time reaction is essential in healthcare and computational complexity demands efficient many-core technologies to grow and manage big data in constrained computational times.

4 The parallel algorithm

The ViT architecture described in Sect. 2.2 mainly relies on matrix–matrix operations. The most intensive operations are those described by Eqs. (2)–(5), together with the convolutions related to bands reduction. All these operations feature an intrinsically parallel nature that can be well mapped on a GPU device to lower the processing times. Nowadays, GPU computing exploits both Edge and Cloud solutions. Among the different possibilities, NVIDIA GPUs represent an interesting solution since they adopt CUDA as a programming language. Indeed, a CUDA program can run on both a low-power GPU and a board designed for a Cloud solution. Besides, the CUDA ecosystem includes optimized libraries for linear algebra operations (*cuBLAS*) and for deep learning layers (*cuDNN*).

The algorithm described in Sect. 2.2 can be developed using the *cuDNN* library for the bands' reduction and the *cuBLAS* routines for Eqs. (2)–(5). The other parts can be developed using handwritten CUDA code.

The parallel algorithm flow starts by allocating the GPU memory to store the input raw hyperspectral image, which is then transferred onto this memory area. The hypercube is stored as a linearized 1D array, adopting the NCHW data layout where N is the number of images, C is the channels number (i.e., the bands) and H and W are the image height and width, respectively. It is worth noting that N is equal to 1 since a single image at a time is acquired. This data layout has been chosen for two main reasons. The first is that it makes some preprocessing steps efficient concerning memory access patterns, while the latter is that it is supported by both the linear algebra and deep learning libraries (*cuBLAS* and *cuDNN*, respectively).

Together with the raw image, also the white and dark references are transferred to the GPU memory. After that, each thread computes Eq. (1) for a single spectral pixel. The result overwrites the original input image in the GPU memory space. The first four and the last five noisy bands are then removed by a simple memory transfer into another GPU memory location, through a proprietary optimized function (the *cudamemcpy* routine, specifying as copy direction *cudaMemcpyDeviceToDevice*). The next steps are the smoothing filtering and the normalization, which are performed in parallel, like the image calibration.

To develop the bands reduction algorithm, based on the convolutional operator, the *cuDNN* library is exploited, which is initialized by a proper handle (through the *cudnnCreate* routine) and requires enriching the input data with special variables, called descriptors. At this first stage of the parallel algorithm, both the *cuDNN* context creation and the *cuBLAS* library are initialized before activating all the necessary routines.

The *cuDNN* descriptors store different information, such as the data layout, the numeric representation, the stride, the padding and the convolution algorithm to use. Finally, the *cuDNN* routines require to specify a workspace, which is a GPU memory location used to store temporary data during the computation. The workspace size is computed by a custom function which should be activated before the convolutional one. It is worth noticing that also this routine is included within the *cuDNN* library. This workspace needs to be allocated on the GPU memory before executing the forward convolution (by activating the corresponding *cudnnConvolutionForward* routine). Once this operation is completed, the descriptors are destroyed and the workspace must be deallocated. These steps, except for the *cuDNN* context creation, are repeated three times with different descriptors' parameters, according to the values reported in [15].

The output of the third convolutional layer is given as input to the ViT architecture. At this stage, linear algebra operations are performed exploiting the signature optimization features of the *cuBLAS* library, like the adoption of the column-major format to store the array, together with the 1-based indexing. It is worth noticing that this is the opposite of the C language standard, which states that the array should be stored in the row-major format adopting the 0-based indexing. In particular, the matrix–matrix multiplications of Eqs. (2)–(4) have been parallelized through the *cublasSgemm* routine, which also solves the different data layout issue by appropriately setting its parameters to consider the two input matrices as transposed. Indeed, by exploiting the matrix–matrix multiplication properties, the resulting matrix stored in the row-major format can be easily obtained.

Moreover, the same kind of routines allow the computation of the argument of the *softmax* nonlinear function, as well as the implementation of all the MLP layers, since they can be written as matrix–matrix multiplications. The final MLP stage produces the final classification. In this case, the class to which the image is assigned corresponds to the perceptron with the maximum values. This is computed using an ad hoc function (the *cublasIsamax* routine) that maximizes computing performances and, at the same time, makes the code more compact, which finds the index of the maximum element within an array. This value is copied back to the host and decremented by 1 to convert it into the 0-based indexing. The obtained value corresponds to the label assigned to the class by the classifier. After that, GPU memory is deallocated, the descriptors, the *cuDNN*, the *cuBLAS* contexts are destroyed and the program ends. Figure 4 summarizes the flow of the parallel algorithm.

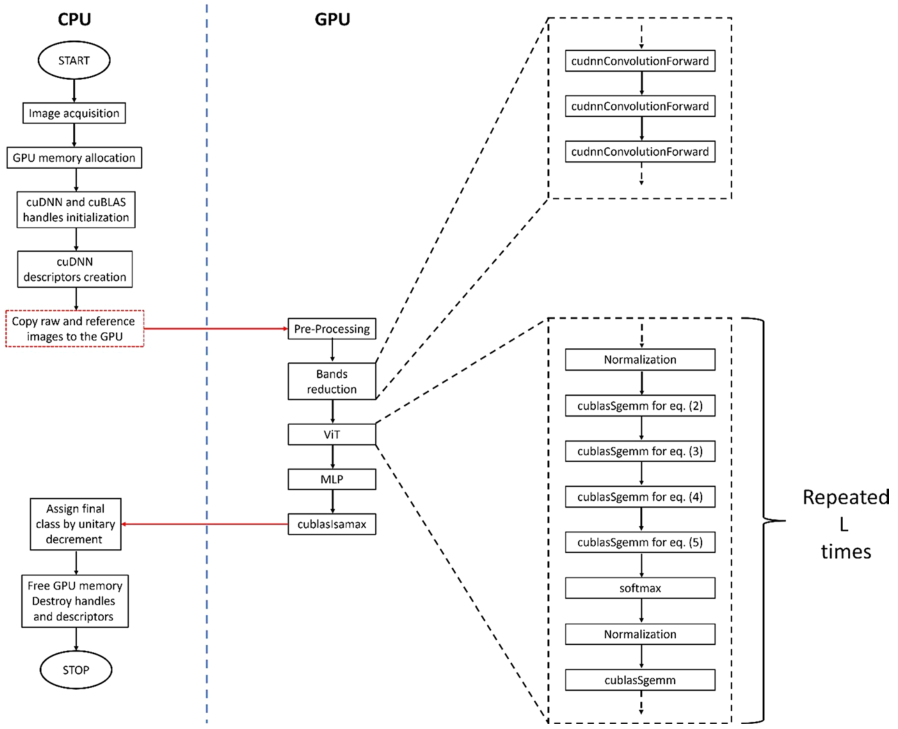

Fig. 4 The flow of the parallel algorithm. The red arrows denote data transfers from CPU to GPU memory spaces and vice versa. The most important steps of the bands reduction and Vision Transformer are detailed in the dashed boxes

5 Experimental setup

The ViT architecture described in Sect. 2.3 has been validated on different architectures, covering a wide range of serial and parallel architectures. In particular, the architectures considered in the experiments are:

1. NVIDIA Jetson Nano Developer Kit featuring a 128 CUDA cores Maxwell GPU, a Quad core ARM A57 running at 1.43 GHz CPU and 4 GB 64-bit LPDDR4 RAM;
2. An Avenger 96 board equipped with an ARM A7 CPU working at 650 MHz and equipped with 1 GB of RAM;
3. Intel-i5 CPU for Mac Pro (2.3 GHz, 8 GB of RAM);
4. A desktop system equipped with an Intel-i9-9900X CPU (3.5 GHz, 128 GB of RAM) and two 2944 CUDA cores NVIDIA RTX 2080 GPUs;
5. On-premise university cluster composed of five computing nodes, each equipped with three NVIDIA A16 boards and two Intel Xeon Silver CPUs working at 2.4 GHz with 768 GB of RAM;
6. A desktop system equipped with an Intel-i9-13900 K CPU (3.0 GHz, 64 GB of RAM) and a 16,348 CUDA cores NVIDIA RTX 4090 GPU.

The algorithm has been tested on different systems, characterized by specific configurations and designed for various purposes. In particular, the most important GPU models are covered. This kind of architecture also inherits the enhancements introduced in the Volta family of devices: independent thread scheduling, hardware accelerated multi-process service (MPS) with address space isolation for multiple applications and cooperative groups.

CPUs like Intel's i5 for Mac Pro, Intel-i9-9900X and ARM processors have been used to characterize different behaviors among various serial architectures regarding both processing time and power consumption, related to their diverse characteristics. Furthermore, these solutions serve as reference to evaluate GPU acceleration performances.

All these elements concur to the evaluation of the applicability of such methodologies using various technologies and paradigms. The portability, but also the performance in terms of processing times, power and energy consumption, among the possible parameters are in fact crucial factors in determining the most appropriate architecture for a specific context and goal.

5.1 GPU Turing architecture

This technology provides advances in efficiency and performance in the contexts of PC gaming, professional graphics applications and deep learning through an innovative streaming multiprocessor (SM) architecture with improved shader execution efficiency and a new memory system architecture that includes support for the latest GDDR6 memory technology [54]. Moreover, its tensor cores (TC) power a suite of new DL-based neural services for graphics effects, in addition to fast AI inferencing for Cloud-based systems.

The Turing SM is a new streaming multiprocessor architecture that increases shading efficiency, incrementing by 50% the delivered performance per CUDA core, compared to the previous generation. This is possible by introducing a new independent integer data path that can execute instructions concurrently with the floating-point math data path. In addition, SM memory path has been redesigned to unify shared memory, texture caching and memory load caching into one unit. This brings twice more bandwidth and more than double the capacity available for L1 cache for common workloads [54].

Tensor cores are new specialized execution units designed specifically for tensor and matrix operations (core functions in DL). Turing GPUs include a new version of the design that has been enhanced for inferencing workloads that can tolerate quantization and do not require 16-bit floating-point (FP16) precision. The Deep Learning Super-Sampling (DLSS) technique is powered by these cores, which leverages a deep neural network to extract multi-dimensional features of a scene and combine details from multiple frames for higher-quality images. Indeed, this architecture is specifically designed to improve inference performance, by integrating TCs, NVIDIA's run-time inferencing framework TensorRT, CUDA and CuDNN libraries. TCs also support fast 8-bit integer (INT8) matrix operations to accelerate inference throughput with minimal accuracy loss.

The fourth system listed makes use of two GPUs with this architecture.

5.2 GPU Ampere architecture

Released in 2020, the Ampere architecture runs 1.7 times faster than Turing GPUs [55]. It has been developed for AI training and inference, HPC workloads and data analytics applications and it is twice more power efficient than the previous generation. Three GPUs belonging to this family are used in the fifth experimental system used in this paper. GPUs of the Ampere family have 28.3 billion transistors with a processor that contains three types of compute resources: programmable shading cores (NVIDIA CUDA cores); RT cores, accelerating Bounding Volume Hierarchy (BVH) and third-generation tensor cores, for speedup in AI neural network training and inferencing. They incorporate 10,752 CUDA cores, 84 s-generation RT cores and 336 third-generation tensor cores. New Tensor Float 32 (TF32) precision provides five times the training to accelerate AI and data science code, specifically for AI denoising, super-resolution, enhanced video and voice communications.

5.3 GPU Ada Lovelace architecture

This is the newest and fastest architecture developed, with focus on graphics, AI and compute workloads [56]. The sixth experimental setup used is equipped with an RTX 4090 model. These GPUs have 76.3 billion transistors and 18,432 CUDA cores capable of running at clocks over 2.5 GHz. They support Shader Execution Reordering to dynamically organize shading workloads and deliver frame rates over

100 frames/s at 4 K resolution. This architecture doubles the raw FP32 compute performance and almost triples the rendering performance, allowing also to execute 724 TFLOPs of tensor operation performance at 300W power envelope. This is combined with the lowest power consumption for AI, graphics and video workloads in datacenters [56]. Therefore, it is the ideal solution for applications in the fields of generative AI, autonomous vehicles, High-Performance Computing (HPC), virtual workstations and single-GPU AI training and inferencing and can be used in regional datacenters, Edge Computing systems and outdoor locations.

5.4 GPU Maxwell architecture

This architecture differs from the previous ones, since it has been designed with the aim of maximizing power and energy efficiency, as well as performance per watt consumed, making them ideal for power-limited environments like embedded systems [57]. It introduces a new design for the streaming multiprocessor that also improves control logic partitioning, workload balancing, clock-gating granularity, compiler-based scheduling and number of instructions issued per clock cycle. Moreover, it enhances CUDA programming compared to previous architectures, thus producing speedups without modifying original codes designed for Fermi and Kepler devices.

In this paper, the first system listed above has been considered using two different development board functioning modes, respectively: FULL-POWER™ and ECO-MODE™.

6 Experimental results

The technologies described in Sect. 4 can be used to implement the Edge or the Cloud paradigms. In the Edge paradigm, images are acquired and locally processed by a parallel device. On the other hand, considering the Cloud environment, images are locally acquired by a hyperspectral camera and sent to a processing center adopting 5G communication. This protocol ensures low latency and a bandwidth suitable to upload big data amounts. Moreover, this technology has already been used in the medical field for remote data transmission [58].

Concerning the GPUs, the parallel code described in Sect. 2 has been compiled enabling all the code optimization flags and specifying the compute capability of the target board. Also, for the ARM processors the code has been compiled enabling all the code optimization flags.

For both approaches, the input is given by a single hyperspectral image, which is acquired by the snapshot camera described in [15]. The system targets the processing of a single image since the portable device was designed in [30, 48] to acquire and process one image at a time. All the results have been validated against the original code developed in [15] considering all the 76 hyperspectral images included in the dataset.

Table 2 shows the classification performance of the ViT, which are also reported in [15]. It is worth noticing that the classification performance is the same for all the proposed devices. Indeed, the outputs of all serial and parallel codes have been validated against the original results of [15].

The processing times have been obtained as the mean value of the execution on the dataset described in Sect. 2.2. Table 3 summarizes the processing time obtained by all the systems.

Concerning the data transfer time in the Edge Computing paradigm, the literature reports that the typical bandwidth of the 5G technology is 75 Mb/s, while the work in [58] measured a bandwidth of about 50 Mb/s. Thus, we evaluated the data transmission time adopting not only these two values, but also a lower one of 30 Mb/s, to show the impact of the communication on the Cloud approach.

Table 4 shows the data transfer time, assuming that a single image is transferred. The input image has a size of 50×50 pixels $\times$ 125 bands and is represented using the IEEE 754 single-precision floating-point standard. Thus, the image features a size of 1.22 MB. Moreover, we analyzed transfer times between host and device, always showing values in the range of only 7–10% of the total processing time, considering the sum of all the CPU–GPU and GPU–CPU transfers.

Table 2 ViT classification performance

Class	Accuracy (%)	Precision (%)
Benign epithelia	63	90
Benign melanocytic	84	100
Malignant epithelia	41	5
Malignant melanocytic	91	99

Table 3 Processing time of all the test systems

Board	Processing time (s)
CPU Intel i5	207.000
Avenger 96	868.324
Jetson Nano ARM A57	354.569
Jetson Nano Full Power Mode	2.415
Jetson Nano Eco Mode	3.340
RTX 2080	0.956
RTX 4090	0.552
A 16	0.751

Table 4 Transfer time for a single image vs the upload band

Upload band (Mb/s)	Transfer time (s)
75	0.130
50	0.195
30	0.325

Table 5 Impact on the transfer time of the total processing time for the Cloud solutions

Board	Total processing time (s)		
	75 Mb/s	50 Mb/s	30 Mb/s
CPU Intel i5	207.130	207.195	207.325
RTX 2080	1.086	1.151	1.281
RTX 4090	0.682	0.747	0.877
A 16	0.881	0.946	1.076

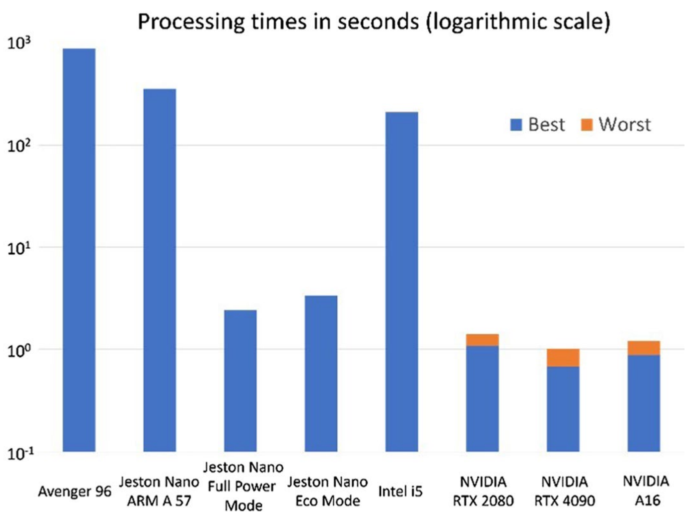

Fig. 5 Processing time for all the proposed systems. The processing times are reported in seconds and the logarithmic scale is adopted. For the cloud solutions, the best and worst cases processing times are shown

To evaluate the impact of the transfer times, they should be added to the processing times of Table 2. For the Edge Computing devices, namely the Jetson and the Avenger 96 boards, the total processing times are related only to the elaboration of the ViT algorithm (i.e., the values reported in Table 1). On the other hand, for the Cloud solution, i.e., the RTX 2080, the RTX 4090, the A16 GPUs and the Intel i5 processor, the total processing times should also consider the time taken by the data transfer from the acquisition point to the elaboration facility. Thus, the values reported in Table 4 should be added to the one reported in Table 3. The impact of the different upload bands on the total processing time of the Cloud solution is reported in Table 5.

The results obtained by all the systems are also compared in Fig. 5 where the best- and worst-case scenarios are considered for the Cloud-based solutions. Moreover, the energy dissipated by the computation is shown in Table 6.

The analysis of the processing times and of the dissipated energy clearly highlights that the parallel processing is the most suitable to develop the targeting

Table 6 Dissipated energy

Architecture	Board	Dissipated energy (J)	Time (s)	Energy/time (mJ/s)	Time/energy (s/mJ)	Percentage of time (%/mJ)
CPU	CPU Intel i5	4.887	207.325	23.57168696	0.042423777	0.02
Arm Cortex	Avenger 96	4.342	868.324	5.000437625	0.199982497	0.023
NVIDIA MAXWELL GPU	Jetson Nano ARM A57	0.985	354.569	2.778020639	0.359968528	0.1
NVIDIA AMPERE GPU	Jetson Nano Full Power Mode	$6.708 * 10^{-3}$	2.415	2.777639752	0.360017889	14.91
NVIDIA AMPERE GPU	Jetson Nano Eco Mode	$4.639 * 10^{-3}$	3.34	1.388922156	0.719982755	21.56
NVIDIA TURING GPU	RTX 2080	0.040	1.281	31.225605	0.032025	2.5
NVIDIA ADA LOVELACE GPU	RTX 4090	0.069	0.877	78.67730901	0.012710145	1.45
NVIDIA AMPERE GPU	A 16	0.052	1.076	48.32713755	0.020692308	1.92

application. Indeed, the Intel i5 and the ARM-based solutions feature the highest computational times. On the other hand, among the parallel solutions, the Cloud GPUs obtain the lowest processing times with the RTX 4090 as the most performant board. The reason is that the RTX 4090 is the GPU featuring the highest number of SMXs, thus capable of executing more thread blocks in parallel than the other GPUs. Indeed, the A16 GPU is a quad-chip device, meaning that a standard CUDA code written for a single GPU only exploits one-fourth of the total computational power. On the other hand, the data dimensionality of the images considered in this work and the algorithm structure does not justify the development of a parallel code targeting multi-GPU systems.

The GPU-based Edge solutions have processing times which are six times slower than the Cloud GPUs but two orders of magnitude faster than the ARM and Intel i5 processors. Moreover, considering the dissipated energy, the Edge boards based on GPU achieve the best performance. The worst performance is obtained by the Edge ARM and the Cloud Intel, which perform a serial processing. Finally, Cloud GPUs have an energy dissipation which is an order of magnitude greater than the Edge solution, even if the contribution of the data transfer is not considered. Jetson Nano boards are more effective in terms of energy consumption with a range of processing times between 2.4 and 3.3 s, which take much longer than the RTX/A16 boards.

Therefore, Edge solutions based on GPUs can be considered a good trade-off between the energy dissipation and the processing time. Since skin cancer has a temporal evolution that does not imply the need for real-time processing, it can be stated that Jetson GPUs represent an innovative solution regarding energy dissipation. Indeed, both the Full Power and Eco Mode feature the best performances. Moreover, the value obtained with the Jetson boards for skin cancer classification is one order of magnitude lower than the best cloud-based solutions while maintaining adequate processing times.

A direct comparison between the proposed work and the state of the art is a challenging task since there are some critical issues that are addressed in the following.

In particular, the work in [59] targets hyperspectral images classification on low-power GPUs through CNN. To reduce data dimensionality, the authors adopted the principal component analysis (PCA) as a preprocessing step, similarly to the channels reduction block proposed in this paper. However, it is important to highlight that PCA is a classical data reduction technique, which is not based on the convolutional operator as the channel reduction block adopted in this research. Moreover, CNNs have already been exploited for hyperspectral data classification and cannot be considered a novel approach as the ViT.

In [60], the authors adopted a spatial–spectral CNN to improve classification performance with respect to standard CNN. They also proposed a parallel implementation on high-end GPUs, achieving reduced processing times. Again, a direct comparison would not be fair since the algorithm structure is very different and they focused on reducing processing times without considering power consumption constraints.

Another interesting work is [61], where a novel U-shaped pansharpening network with channel cross-concatenation and spatial–spectral attention mechanism for hyperspectral image super-resolution is proposed. The authors developed this network using PyTorch and exploiting an NVIDIA RTX 3090 GPU. They classified three hyperspectral images with processing times of about 47 s, which is higher than the ones obtained by the parallel technology proposed in this paper. However, a direct comparison would not be fair since the hyperspectral images feature a diverse number of bands from the skin cancer dataset and the classification network structure is also different.

To the best of the authors' knowledge, this is the first time that a work evaluates the Cloud paradigm to analyze hyperspectral medical images. On the other hand, parallel implementations based on GPU boards have already been proposed in the literature [60, 62–65]. However, these algorithms have different structures and computational complexities, making direct comparisons not feasible.

7 Conclusions and future developments

This paper described and compared the adoption of Edge and Cloud paradigm to analyze medical hyperspectral images, targeting skin cancer detection.

The proposed work uses a novel ViT architecture which has been trained on a small-size dataset. Even if data augmentation techniques were adopted in the original ViT development, the model can be further enhanced by acquiring additional data. Indeed, a larger dataset can be used to fine tune the model and improve its generalization capabilities. Another possibility is to consider a pre-trained architecture and to apply the transfer learning methodology to partially retrain the network on the actual problem. However, this method cannot be applied at the time of writing, since, to the best of the authors' knowledge, no previous ViT architecture was trained on medical hyperspectral images.

Concerning possible future research lines, a step that has not been covered by the proposed work is data visualization. Indeed, medical doctors should have at their disposal a user-friendly interface to start data acquisition and to clearly visualize the results, for example with a false color image representing the classification of the acquired lesion.

A wide range of Edge and Cloud devices have been considered, ranging from serial ARM and Intel processors to many-core GPUs.

A parallel version of the ViT architecture has been developed using CUDA language targeting GPU boards. The parallel algorithm relies on both custom CUDA code and highly optimized libraries, such as *cuBLAS* and *cuDNN*. Moreover, a serial version of the ViT algorithm has been developed to exploit the serial CPUs and to serve as reference for the parallel implementation.

These code versions have been tested on different Edge and Cloud devices, recording the processing time and the power consumption. The comparison showed that Cloud GPU solutions perform better than the other devices from the processing times perspective. However, they featured an energy consumption which is an order of magnitude greater than Edge GPUs. Indeed, Edge solutions based on GPUs emerged as the best trade-off between the processing time and the energy consumption. Focusing on the development of a portable instrument for skin cancer detection, the most important metric is the energy dissipated by the device. In this context, the best choice is represented by the Jetson boards, since their values are one order of magnitude lower than the best cloud-based solutions.

Possible future research lines are related to developing other classification algorithms and to evaluate their different performance in the Cloud and Edge paradigms, with data dimensionality playing a critical role in the choice of the best trade-off.

Author contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by M.L.S., E.T. and E.M.. The first draft of the manuscript was written by E.M. and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

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

Declarations

Competing interests The authors declare no competing interests.

References

1. Yu K-H, Beam AL, Kohane IS (2018) Artificial intelligence in healthcare. Nat Biomed Eng 2:719–731. https://doi.org/10.1038/s41551-018-0305-z
2. Barragán-Montero A, Javaid U, Valdés G et al (2021) Artificial intelligence and machine learning for medical imaging: a technology review. Physica Med 83:242–256. https://doi.org/10.1016/j.ejmp.2021.04.016
3. Castiglioni I, Rundo L, Codari M et al (2021) AI applications to medical images: from machine learning to deep learning. Phys Med 83:9–24. https://doi.org/10.1016/j.ejmp.2021.02.006
4. Rong G, Mendez A, Bou Assi E et al (2020) Artificial intelligence in healthcare: review and prediction case studies. Engineering 6:291–301. https://doi.org/10.1016/j.eng.2019.08.015
5. Rajpurkar P, Chen E, Banerjee O, Topol EJ (2022) AI in health and medicine. Nat Med 28:31–38. https://doi.org/10.1038/s41591-021-01614-0
6. Florimbi G, Fabelo H, Torti E et al (2020) Towards real-time computing of intraoperative hyperspectral imaging for brain cancer detection using multi-GPU platforms. IEEE Access 8:8485–8501. https://doi.org/10.1109/ACCESS.2020.2963939
7. Lin B, Wu S (2022) Digital transformation in personalized medicine with artificial intelligence and the internet of medical things. OMICS 26:77–81. https://doi.org/10.1089/omi.2021.0037
8. Amsel N, Tomlinson B (2010) Green tracker: a tool for estimating the energy consumption of software. In: CHI '10 extended abstracts on human factors in computing systems. ACM, New York, NY, USA, pp 3337–3342
9. Antonopoulos N, Gillam L (2017) Cloud computing. Springer, Cham
10. Shi W, Pallis G, Xu Z (2019) Edge computing. Proc IEEE 107:1474–1481. https://doi.org/10.1109/JPROC.2019.2928287

11. Shi W, Cao J, Zhang Q et al (2016) Edge computing: vision and challenges. IEEE Internet Things J 3:637–646. https://doi.org/10.1109/JIOT.2016.2579198
12. Verma P, Kumar U (2023) Analyzing the application of edge computing in smart healthcare. In: Convergence of cloud with AI for big data analytics. Wiley, pp 121–155
13. Marenzi E, Torti E, Danese G, Leporati F (2022) FPGA High level synthesis for the classification of skin tumors with hyperspectral images. In: 2022 11th mediterranean conference on embedded computing (MECO). IEEE, pp 1–4. https://doi.org/10.1109/MECO55406.2022.9797211
14. Fontanella A, Marenzi E, Torti E et al (2018) A suite of parallel algorithms for efficient band selection from hyperspectral images. J Real Time Image Process. https://doi.org/10.1007/s11554-018-0765-0
15. Salvia M La, Torti E, Gazzoni M et al (2022) Attention-based skin cancer classification through hyperspectral imaging. In: 2022 25th euromicro conference on digital system design (DSD). IEEE, pp 871–876. https://doi.org/10.1109/DSD57027.2022.00122
16. Meyer JM, Kokaly RF, Holley E (2022) Hyperspectral remote sensing of white mica: A review of imaging and point-based spectrometer studies for mineral resources, with spectrometer design considerations. Remote Sens Environ 275:113000. https://doi.org/10.1016/j.rse.2022.113000
17. Torti E, Gazzoni M, Marenzi E et al (2023) An attention-based parallel algorithm for hyperspectral skin cancer classification on low-power GPUs. In: 2023 26th Euromicro conference on digital system design (DSD), pp 111–116. https://doi.org/10.1109/DSD60849.2023.00025
18. Signoroni A, Savardi M, Baronio A, Benini S (2019) Deep learning meets hyperspectral image analysis: a multidisciplinary review. J Imaging 5:52. https://doi.org/10.3390/jimaging5050052
19. Ozdemir A, Polat K (2020) Deep learning applications for hyperspectral imaging: a systematic review. J Inst Electron Comput 2:39–56. https://doi.org/10.33969/JIEC.2020.21004
20. Kumar D, Kumar D (2021) Hyperspectral image classification using deep learning models: a review. J Phys Conf Ser 1950:012087. https://doi.org/10.1088/1742-6596/1950/1/012087
21. Rehman A, ul Qureshi SA (2021) A review of the medical hyperspectral imaging systems and unmixing algorithms' in biological tissues. Photodiagnosis Photodyn Ther 33:102165. https://doi.org/10.1016/j.pdpdt.2020.102165
22. Torti E, Leon R, La Salvia M et al (2020) Parallel classification pipelines for skin cancer detection exploiting hyperspectral imaging on hybrid systems. Electronics 9:1503. https://doi.org/10.3390/electronics9091503
23. Torti E, Fontanella A, Plaza A et al (2018) Hyperspectral image classification using parallel autoencoding diabolo networks on multi-core and many-core architectures. Electronics 7:411. https://doi.org/10.3390/electronics7120411
24. La Salvia M, Torti E, Leon R et al (2022) Neural networks-based on-site dermatologic diagnosis through hyperspectral epidermal images. Sensors 22:7139. https://doi.org/10.3390/s22197139
25. Reshef ER, Miller JB, Vavvas DG (2020) Hyperspectral imaging of the retina: a review. Int Ophthalmol Clin 60:85–96. https://doi.org/10.1097/IIO.0000000000000293
26. Barberio M, Benedicenti S, Pizzicannella M et al (2021) Intraoperative guidance using hyperspectral imaging: a review for surgeons. Diagnostics 11:2066. https://doi.org/10.3390/diagnostics11112066
27. Lu G, Fei B (2014) Medical hyperspectral imaging: a review. J Biomed Opt 19:010901. https://doi.org/10.1117/1.JBO.19.1.010901
28. Khan U, Paheding S, Elkin CP, Devabhaktuni VK (2021) Trends in deep learning for medical hyperspectral image analysis. IEEE Access 9:79534–79548. https://doi.org/10.1109/ACCESS.2021.3068392
29. Johansen TH, Møllersen K, Ortega S et al (2020) Recent advances in hyperspectral imaging for melanoma detection. WIREs Comput Stat. https://doi.org/10.1002/wics.1465
30. Leon R, Martinez-Vega B, Fabelo H et al (2020) Non-invasive skin cancer diagnosis using hyperspectral imaging for in-situ clinical support. J Clin Med 9:1662. https://doi.org/10.3390/jcm9061662
31. Li S, Song W, Fang L et al (2019) Deep learning for hyperspectral image classification: an overview. IEEE Trans Geosci Remote Sens 57:6690–6709. https://doi.org/10.1109/TGRS.2019.2907932
32. Yang X, Ye Y, Li X et al (2018) Hyperspectral Image Classification With Deep Learning Models. IEEE Trans Geosci Remote Sens 56:5408–5423. https://doi.org/10.1109/TGRS.2018.2815613
33. Lazcano R, Salvador R, Marrero-Martin M et al (2019) Parallel implementations assessment of a spatial-spectral classifier for hyperspectral clinical applications. IEEE Access 7:152316–152333. https://doi.org/10.1109/ACCESS.2019.2938708

34. Petropoulos GP, Arvanitis K, Sigrimis N (2012) Hyperion hyperspectral imagery analysis combined with machine learning classifiers for land use/cover mapping. Expert Syst Appl 39:3800–3809. https://doi.org/10.1016/j.eswa.2011.09.083
35. Saha D, Manickavasagan A (2021) Machine learning techniques for analysis of hyperspectral images to determine quality of food products: a review. Curr Res Food Sci 4:28–44. https://doi.org/10.1016/j.crfs.2021.01.002
36. Fabelo H, Ortega S, Kabwama S, et al (2016) HELICoiD project: a new use of hyperspectral imaging for brain cancer detection in real-time during neurosurgical operations. In: Bannon DP (ed), p 986002
37. Bray F, Ferlay J, Soerjomataram I et al (2018) Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA Cancer J Clin 68:394–424. https://doi.org/10.3322/caac.21492
38. Liopyris K, Gregoriou S, Dias J, Stratigos AJ (2022) Artificial intelligence in dermatology: challenges and perspectives. Dermatol Ther 12:2637–2651. https://doi.org/10.1007/s13555-022-00833-8
39. Hu X, Xie C, Fan Z et al (2022) Hyperspectral anomaly detection using deep learning: a review. Remote Sens 14:1973. https://doi.org/10.3390/rs14091973
40. Dosovitskiy A, Beyer L, Kolesnikov A, et al (2021) An Image is Worth 16x16 words: transformers for image recognition at scale. In: International conference on learning representations
41. Petracchi B, Gazzoni M, Torti E et al (2023) Machine learning-based classification of skin cancer hyperspectral images. Procedia Comput Sci 225:2856–2865. https://doi.org/10.1016/j.procs.2023.10.278
42. Dildar M, Akram S, Irfan M et al (2021) Skin cancer detection: a review using deep learning techniques. Int J Environ Res Public Health 18:5479. https://doi.org/10.3390/ijerph18105479
43. Ferlay J, Colombet M, Soerjomataram I et al (2021) Cancer statistics for the year 2020: AN overview. Int J Cancer 149:778–789. https://doi.org/10.1002/ijc.33588
44. Siegel RL, Miller KD, Fuchs HE, Jemal A (2022) Cancer statistics, 2022. CA Cancer J Clin 72:7–33. https://doi.org/10.3322/caac.21708
45. Esteva A, Kuprel B, Novoa RA et al (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542:115–118. https://doi.org/10.1038/nature21056
46. Rey-Barroso L, Peña-Gutiérrez S, Yáñez C et al (2021) Optical technologies for the improvement of skin cancer diagnosis: a review. Sensors 21:252. https://doi.org/10.3390/s21010252
47. Haggenmüller S, Maron RC, Hekler A et al (2021) Skin cancer classification via convolutional neural networks: systematic review of studies involving human experts. Eur J Cancer 156:202–216. https://doi.org/10.1016/j.ejca.2021.06.049
48. Fabelo H, Melian V, Martinez B, et al (2019) Dermatologic hyperspectral imaging system for skin cancer diagnosis assistance. In: 2019 XXXIV conference on design of circuits and integrated systems (DCIS). IEEE, pp 1–6
49. Martinez-Vega B, Tkachenko M, Matkabi M et al (2022) Evaluation of preprocessing methods on independent medical hyperspectral databases to improve analysis. Sensors 22:8917. https://doi.org/10.3390/s22228917
50. Puthal D, Sahoo BPS, Mishra S, Swain S (2015) Cloud computing features, issues, and challenges: a big picture. In: 2015 international conference on computational intelligence and networks. IEEE, pp 116–123
51. Sasikala P (2013) Research challenges and potential green technological applications in cloud computing. Int J Cloud Comput 2:1. https://doi.org/10.1504/IJCC.2013.050953
52. Zwolenski M, Weatherill L (2020) The digital universe. J Telecommun Digit Econ 2:9. https://doi.org/10.18080/jtde.v2n3.285
53. Kumar U, Verma P, Qamar Abbas S (2021) Bringing edge computing into IoT architecture to improve IoT network performance. In: 2021 international conference on computer communication and informatics (ICCCI). IEEE, pp 1–5
54. (2018) NVIDIA TURING GPU ARCHITECTURE. https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf. Accessed 20 Sep 2023
55. (2020) NVIDIA AMPERE GA102 GPU ARCHITECTURE. https://www.nvidia.com/content/PDF/nvidia-ampere-ga-102-gpu-architecture-whitepaper-v2.pdf. Accessed 20 Sep 2023
56. (2022) NVIDIA ADA GPU ARCHITECTURE. https://images.nvidia.com/aem-dam/Solutions/Data-Center/l4/nvidia-ada-gpu-architecture-whitepaper-v2.1.pdf. Accessed 20 Sep 2023

57. (2014) NVIDIA Maxwell GM204 Architecture. https://www.microway.com/download/whitepaper/NVIDIA_Maxwell_GM204_Architecture_Whitepaper.pdf. Accessed 20 Sep 2023
58. Spicher N, Schweins M, Thielecke L, et al (2021) Feasibility analysis of fifth-generation (5G) mobile networks for transmission of medical imaging data. In: 2021 43rd annual international conference of the IEEE engineering in medicine & biology society (EMBC). IEEE, pp 1791–1795
59. De Lucia G, Lapegna M, Romano D (2023) A GPU accelerated hyperspectral 3D convolutional neural network classification at the edge with principal component analysis preprocessing, pp 127–138
60. Torti E, Marenzi E, Danese G et al (2023) 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. https://doi.org/10.1109/JSTARS.2023.3301721
61. Liu Z, Han G, Yang H et al (2023) CCC-SSA-UNet: U-shaped pansharpening network with channel cross-concatenation and spatial-spectral attention mechanism for hyperspectral image super-resolution. Remote Sens 15:4328. https://doi.org/10.3390/rs15174328
62. Marenzi E, Carrus A, Danese G et al (2017) Efficient parallelization of motion estimation for super-resolution. In: Proceedings—2017 25th Euromicro international conference on parallel, distributed and network-based processing, PDP 2017. https://doi.org/10.1109/PDP.2017.64
63. Marenzi E, Torti E, Leporati F et al (2017) Block matching super-resolution parallel GPU implementation for computational imaging. IEEE Trans Consum Electron. https://doi.org/10.1109/TCE.2017.015077
64. Lu Y, Xie K, Xu G et al (2021) MTFC: A Multi-GPU training framework for cube-CNN-based hyperspectral image classification. IEEE Trans Emerg Top Comput 9:1738–1752. https://doi.org/10.1109/TETC.2020.3016978
65. Ordonez A, Heras DB, Arguello F (2022) Multi-GPU registration of high-resolution multispectral images using HSI-KAZE in a cluster system. In: IGARSS 2022—2022 IEEE international geoscience and remote sensing symposium. IEEE, pp 5527–5530