Fast Information Retrieval from Web Pages

HAZEM M. EL-BAKRY
Faculty of Computer Science & Information Systems,
Mansoura University, EGYPT
E-mail: helbakry20@yahoo.com

NIKOS MASTORAKIS
Department of Computer Science,
Military Institutions of University Education (MIUE)
-Hellenic Naval Academy, Greece

Abstract: In this paper, a new fast algorithm for information retrieval is presented. Such algorithm relies on performing cross correlation in the frequency domain between input data and the input weights of fast neural networks (FNNs). It is proved mathematically and practically that the number of computation steps required for the presented FNNs is less than that needed by conventional neural networks (CNNs). The main objective of Internet users is to find the required information with high efficiency and effectiveness. Finding information on an object's visual features is useful when specific keywords for the object are not known. Since intelligent mobile agent technology is expected to be a promising technology for information retrieval, there is a number of intelligent mobile agent based-information retrieval approaches have been proposed in recent years. Here, the work presented in [25] for image-based information retrieval using mobile agents is greatly enhanced. Multiple information agents continuously traverse the Internet and collect images that are subsequently indexed based on image information such as the URL location, size, type and the date of indexation. In the search phase, the intelligent mobile agent receives the image of object as a query and searches the set of web pages that contain information about the object. This is done by matching the query to images on web pages faster than the work presented in [25]. Furthermore, by applying cross correlation, object detection becomes position independent. Moreover, by using neural networks, the object can be detected even with rotation, scaling, noise, distortion or deformation in shape.

Keywords: Fast information retrieval, Content-based image retrieval, Image clustering, and Intelligent mobile agent

I. Introduction

With the development of Internet technology, the fast growing World Wide Web has become one of the most important sources of information and knowledge. When searching the Web, a user can be overwhelmed by thousands of results retrieved by a search engine, of which few are valuable. The problem for search engines is not only to find topic relevant results, but results consistent with the user's information need. How to retrieve desired information from the Internet with high efficiency and good effectiveness is become the main concern of internet user-based [1]. Users generally find information using search engines. The input to these search engines often consists of keywords or other written text like a question. But the Web contains not only text, but also information in other modalities such as images. They are however not yet being used as input for general Web searches. When the user wants to query about an object that has unique visual features, the image of the object that represents the visual properties can be used as a query. By enabling searches on object appearance provide users with a new, more convenient and direct means for finding information about objects encountered in everyday life. Many existing world wide search engines, for example Google and Yahoo, rely on meta-data (annotations) or the context in which the image is found and the query is performed using text-based information retrieval method, the matching is performed based on image captions or file names, thus the performance of image retrieval is based on the similarity between the user's text query and image text annotation not on the image content itself [2]. Content Based Image Retrieval (CBIR) searches for images by using asset of visual features such as color, shape and texture, are extracted from the images that characterize the image content [3]. These techniques have been used in many areas, such as geographic information system, biomedical image processing and digital libraries [4]. One of the main advantages of the CBIR approach is the possibility of an automatic retrieval process, instead of the traditional keyword-based approach, which usually requires very difficult and time-consuming previous annotation of database images. Comparatively speaking, CBIR more objectively reflects the content of images [6]. Most of CBIR systems are only operate on a local demo database of a few thousand images stored at the host web site [7]. Unlike image retrieval from a fixed database, where each image is treated as an independent object, for Searching for image on the Web basically comes down to locating an appropriate Web page and to retrieve relevant information about the image from that site[8,9].

Intelligent mobile agent is a program that traverses the Internet, moves to different sites with different characteristics, searches for the desired information, and returns the search report to the user query [10,11]. When large quantities of data are stored at

distributed sites, moving the computations to the data is a more realistic and feasible approach, compared with migrating data to the computations. In other words; instead of gathering information from distributed sites to be processed at a central site, users can dispatch mobile agents to a destination site to perform information retrieval and to return to a user the results of the analysis. Thus, the data transmitted over the network are minimized. This is especially important when using a low-bandwidth access network. Features of mobile agent, such as cross-platform migrating, dynamic and distributed computing make it possible to solve the problems of low bandwidth and unstable connection in the internet and to provide the support of searching in the internet via inconsistent connected devices (e.g. laptop, PDA, mobile phone, etc). Mobile agents already have been used to build many distributed applications, including distributed information retrieval [12], and network management [13]. Furthermore, they provide an attractive paradigm for design and implementation of scalable, flexible, and extensible system for image retrieval from multiple, distributed heterogeneous image sources [14].

In this paper, an image-based information retrieval using intelligent mobile agent is presented. Image crawler continuously traverses the Internet and collect images that are subsequently indexed based on integrated feature vectors. These features along with additional information such as the URL location and the date of index procedure are stored in a database. The user can access and search these indexed contents through the Web with an advanced and user friendly interface. The output of the system is a set of links to the content available in the WWW, ranked according to their similarity to the submitted image by the user.

There are many systems for Content –based image retrieval (CBIR) using mobile agent are presented e.g. [5, 15, 16].None of these systems provides a web search engine; they are based on local image database. Comparing to image databases, the Web is an unlimited, immense repository of images, covering much broader resources, and is increasing continuously at high speed rate. In [17, 18] mobile agent is used for web information retrieval based on text query not image query. Unfortunately, keywords may not accurately or completely describe image content.

Two approaches for information retrieval from distributed database based on mobile agents are presented in [12]. The first approach utilizes the mobility of agent for moving the query to the desired site where the data resided while the second one is based on reduction of the number of migrating agents. The work in this paper presents a new approach for finding information related to an image from the World Wide Web by matching image itself and retrieving the web page which contains information related to the submitted image. This is called image-based information retrieval. The information retrieved can be in any format or media, but it's found using an image as a query.

The paper is organized as follows: Section II presents the architecture of the proposed system. Section III illustrates experiments and evaluation of the proposed system. The proposed alogrithm for fast information retrieval is presented in section IV. Conclusion and future work are presented in section V.

II. System Architecture

The overall system is split into two parts: (i) the off-line part and (ii) the on-line or user part.

In the off-line part, several *Information Crawlers,* continuously traverse the WWW, collect images and store them to a database for further processing. Then the *image indexing* module processes the image in order to extract descriptive feature using color and shape features. These features along with information of the images such as URL, date of process, size and a thumbnail are stored in the *database.*

In the on-line part, a *user* connects to the system through a common Web Browser using the HTTP protocol. The user can then submit queries either by example images or by sketch. The query then is processed by the server and the *retrieval* phase begins; the indexing module is repeated again for the submitted image and then the extracted features are matched against those stored in the database. The results containing the URL, as well as, the thumbnail of the similar images are transmitted to the user and the results are ranked according to their similarity to the submitted image.

A) Information Crawler

The image collection process is conducted by a different autonomous Web agent or Crawler run on a network of computers. The agent traverses the Web by parsing the hyperlinks, detects images, retrieves and transfers them for processing to the system server. The system server extracts features from the images and then stores it in the database.

The overall collection process is illustrated in Fig. 1; it is carried out using several distinct modules:

The Traversal Crawler - assembles lists of candidate Web pages that may include images or hyperlinks to them.

The Hyperlink Parser - extracts the URLs of the images.

The Retrieval Crawler - retrieves and transfers the image to the system server for further processing.

This section gives a description of the information crawler as shown in Fig. 1; Traversal -crawler browses the WWW to collect images. The web-sites listed in the categories Shopping and Recreation of the DMOZ directory [21] are chosen as a starting point. In the first phase, the Traversal Crawler follows a breadth first search across the Web. It retrieves

pages via Hypertext Transfer Protocol (HTTP) and passes the Hypertext Markup Language (HTML) code to the Hyperlink Parser. In turn, the Hyperlink Parser detects new URLs, encoded as HTML hyperlinks, and adds them back to the queue of Web pages to be retrieved by the Traversal Crawler. The Hyperlink Parser detects the hyperlinks in the Web documents and converts the relative URLs to absolute addresses. By examining the types of the hyperlinks and the filename extensions of the URLs, the Hyperlink Parser extracts the URLs of the images. In the second phase, the list of image URLs from the Hyperlink Parser is passed to the Retrieval Crawler. The Retrieval Crawler retrieves the images and provides them as input to the indexing module. After the indexing procedure, the extracted features are added to the database. Another important function of the Retrieval Crawler is to extract attributes associated with the image such as URL, date of processing, size, width, height, file size, type of visual data, and so forth, and also generate a thumbnail icon, that sufficiently compacts and represents the visual information.

B) Indexing Module

After the image detection module collects and transfers images to the server, image indexing module processes them in order to extract low level feature of image. This module consists of two steps; feature extraction and image clustering.

1. Feature Extraction

The prototype implementation uses *Color Coherence Vectors* [19] as a feature extraction and comparison algorithm. Feature vectors consist of 128 float values; each vector is computed as follows: the image is blurred using a simple 3 x 3 convolution filter which averages the color values of all horizontal and vertical neighbors of the filtered pixel. The blurred image is then quantized to a color space of 64 colors. In the last step, the pixels of the image are classified into coherent and incoherent pixels. Coherent pixels are pixels which are part of a horizontally and vertically connected pixel area of the same color whose size exceeds a certain threshold. Incoherent pixels are pixels which are not coherent pixels. For each of the 64 colors, the coherent and incoherent pixel counts are summed up separately and normalized with regard to the total image area. This results in a 128 dimensional vector. The Color Coherence Vector algorithm has the advantage that it is easy to implement, reasonably fast, and achieves a high compression rate. Images are reduced to a vector whose encoding is less than 600 bytes.

2. Clustering Similar Image

The increase of number of images in the database, the increase in the time required to retrieve results. To avoid search over the whole images database, the images with similar features vector should be grouped together in the same class. K-means clustering algorithm [20] is used for this operation because it is quite fast, simple and has been applied and shown to be useful in many applications. The basic step of k-means clustering is simple. At the beginning, numbers of clusters K are determined and random feature vectors are selected as the initial centroids of these clusters. The grouping is done by minimizing the sum of squares of distances between feature vector and the corresponding cluster centroid.

C) Image Query Module.

The image query module includes methods for extracting features from query images, classifying query image, and determining similarity between features extracted from the query and those stored in the database.

1. Query Feature Extraction

When a user initiates a query for information about the image, the *Feature Extractor* module performs feature extraction on the example image and sends the feature vector, contained in a query message, to the broker server.

2. Query Classification

Once the broker server receives the query, it will calculate the distances from the Query-image feature vector to the centroids of the set of each cluster to determine which clusters are closest to the query. The minimum square error is used to obtain the distance between a Query Image feature vector and the centroid vector of the cluster, which can be described as:

$$Sim\ (\vec{C}, \vec{Q}) = Min\left(\sum_{i=1}^{k} \left\| \vec{Q}_i - \vec{C}_i \right\| \right) \quad (1)$$

3. Distance Measure

In content-based image retrieval, a distance measure is usually used to check similarity or dissimilarity between two images. The L1 distance is taken as a measure of similarity between two color coherence vectors.

Let $\langle (h_1, \bar{h}_1), ..., (h_n, \bar{h}_n) \rangle$ be a color coherence vector where h_i is the percentage of coherent pixel of color *i* and $\bar{h}_i$ is the percentage of incoherent pixels of color *i,* then the L_1 distance is defined as:

$$\left\| h - h' \right\| = \sum_{i=1}^{n} (|\, h_i - h_i' \,| + |\, \bar{h}_i - \bar{h}_i' \,|) \quad (2)$$

III. Experimental Results

According to the experiments results, it has been observed that a single-threaded information crawler can traverse the web gathering images at an average rate of one image every 82 seconds. Gathering the images has the potential to take long time with a single-threaded information crawler on a single computer. This problem is address by employing a

parallel information crawler distributed across many machines (image servers). Experiments indicate that the framework can allow using 16 information crawlers to collect over one million images monthly.

The system was developed using c#.net on Pentium 4 3.2GHz CPU (2GB RAM) running on Windows vista platform. For image related operations, the Matlab image processing toolbox was used to assist in extracting visual features. During experiments, different port numbers are used to simulate different image servers. Fig.2, shows an overview and example result from the system: the image of the object is selected. The object image is used to query the database to find the most relevant object, which in turn retrieves the relevant web pages for the user.

A) Implementation Models

1. *Centralized-Broker Approach:*

The architecture is capable of gathering, indexing, caching, replicating, and accessing Internet images in a distributed way, which assures scalability. This model consists of a central image broker, several image servers, *index agents, search agents,* and *fetch agents* (all agents are mobile and relocate during their life cycle). The image broker dispatches index agents which transport feature extraction algorithms to one or more image servers. On these servers, the index agents extract relevant feature vectors from local images and send or take image entries back to the image broker. At the image broker, all image entries are merged into the central index. Each image entry consists of a feature vector, the URL pointing to the host where the image was retrieved, an image ID that uniquely identifies the image at the image server, an optional thumbnail, and optional further information on the image such as its size. The globally unique ID of an image consists of the globally unique URL plus the locally unique image ID. Based on the index data, image brokers can either serve requests in a client/server fashion, or they support intelligent mobile agent queries as follows; A client sends a search agent to the broker which queries information related to a specific image, a sketch, or a feature vector which is extracted from either of these query images. The query result consists of extended image entries which contain the normalized distance between the query and the entry's feature vector, in addition to, the entry itself. The search agent transports the result set back to the client who selects images for retrieval based on the included thumbnails. Once an image is selected for retrieval, the client sends a fetch agent that migrates between image servers to collect information about the query image. An advantage is that this information can then be sent to Brokers in a compressed form, which reduces the network load. Another advantage is that one indexer agent can send the collected data to many Brokers, saving indexing costs. Finally, a third advantage is that the server load is reduced since indexer agents run at the image server, which contributes to the scalability of the system. A Broker can retrieve indexing information from many indexers to build an index of widely distributed data. The Broker can also retrieve indexing information from other Brokers, which gives the Broker an indexed view of the other Brokers.

2. *The Broadcasting Approach*

In this model, one index agent per image server is dispatched and takes residence at the image server to extract features from the images and sets up an index directly at the image server. The index agent may also monitor the local image repository for changes, and it can update its index accordingly and incrementally. The only communication between the broker and the index agent is a short notice that the computation of the index is completed and the index agent is ready to provide service to search agents. The image broker is still a central point of access but it resembles more a yellow page server. It refers search agents to the image servers where index agents reside.

Once the client selects an image for retrieval, the process continues as in the gatherer model.

Based on its index the index agent serves queries of search agents which visit the image server. On each image server, the search agent merges previously collected results with the results of its local search, and prunes the overall number of results.

B) Clustering Approach

In this model, each index Agent creates *metadata (centroid of clusters)* describing its own image collection and sends it to broker server. The broker collects the metadata from each index agent. This broker can route every query to the appropriate image server which contains images belongs to the category of query image. At each image server, the search agent matches the query image with the image collection on the appropriate image server and sends related web pages which contain the related information back to the user.

C) Performance Measure

In order to decide the best model for designing the framework, a comparative analysis is performed using two important parameters that influence the performance of the model. These parameters are; size of intermediate data transfer and number of comparisons required answering the query. The response time, i.e., time elapsed between a user initiating a request and receiving the results, as the metric for performance measure. This includes the time taken for agent creation, time taken to visit mediator server to determine list of image servers to be visited and processing time at each image server to obtain the required images.

Response Time: time elapsed between a user initiating a request and receiving the results,

$$RT = Y + X/DBR \quad (3)$$

Where: *RT*: Response Time (sec.), Y: Communication Initialization Time and Client Processing Time (sec.),
DBR: Data Bit Rate (Network Data Transmission speed: bytes/sec.), and X: Amount of transmitted data (bytes).
X = (Mobile Agents Transfer) + (images descriptor Transfer).
Each image descriptor consists a thumbnail, the ID of that image (unique within the domain of the image server), a measure of similarity to the original query image, and the URL of the image server from which the full size of image can be retrieved.
The Response Time can be written as

RT = time (migrating mobile agents) + time (Total Access Delay) + time (Transferred images descriptors)

$$RT = C + (S * N)/DBR + \sum D + \sum (Mn*Ms)/DBR \quad (4)$$

Where: N: number of servers will be visited, S: size of mobile agent, Mn: the number of retrieved images descriptor, Ms: image descriptor size, D: access delay at each server, and C: processing time at *mediator server.*
Client processing time and intelligent mobile agent creation time will be excluded from calculation, because it will be equal in three approaches. Image size, images numbers are the same at the three approaches. Other parameters have been used for comparison such as:

1) *Size of Intermediate Data Transfer*

In the centralized approach, the size of data transfer is equal to the summation of image descriptors (feature vector+ thumbnail image +image ID +server URL). In the clustering approach size of transfer data is equal to the summation of centroid of image clusters in all image servers. In the Broadcasting approach, the size of intermediate data transported over the network is in complexity *O* ($\sum_{i=2}^{N}$ ((*i-1*)***Xi*), where *Xi* denote to the number of cluster on the *i-th* image server and *N* denote to the total number of image servers in the system. Fig. 3 shows the amounts of intermediate data transported of the network in the centralized approach vs. the clustering approach.

The graph is plotted based on the size of all feature vectors of images from all servers. It is observed that; the amount of data transported in indexing phase in clustering approach is too small compared to the centralized gatherer approach due to two reasons: Firstly, because each image server in the indexing phase of the clustering approach has to communicate only its local centroid for each cluster. Thus, the communication load of clustering per image server is of the order *O*(*NClusters*), *i.e.* for a large number of images per server, the price of distributed clustering is still low compared to centralized gatherer approach that would require sending over the network a large sample of all data stored within the image server. The second reason is that there is no need to update the cluster centroids with each new image because slightly outdated cluster centroids do not significantly affect the retrieval process.

2) *Number of Comparisons*

One of the main factors which affects the response time is the number of comparison performed to retrieve images and location of processing. For the clustering approach, number of comparisons that performed to response query is in order *O [(NClusters) performed on mediator + (Number of images respective clusters) performed on image servers],* where *NClusters* denote to number of clusters in the network. In broadcasting and centralized approaches, intelligent mobile agent perform comparisons in order of *O [(the summation of images stored in all servers) performed on mediator only].* The increase in the number of comparisons which are required to be performed to response query will increase the response time. Moreover, distributing this task between different image servers overcome problems of centralized server and make all hosts participating in the system act as servers. Fig. 4, shows the number of comparison in broadcasting and clustering approaches.

3) *The Response Time*

Number of visited servers during query processing varies at Broadcasting and Clustering approaches. For the broadcasting approach, the search agent will visit all servers in the network in order to find the results for the query so number of servers that will be visited is equal to the total number of servers in the network. But in clustering approach, number of visited servers is only that having the same category as the query image. By calculating total access delay of the system which is used to implement these approaches, it is clear that by allowing *search agent* to visit all servers, total access delay increase proportionally. Fig. 5, shows the effect of increasing number of image servers on access delay in the two cases. Total access delay is the summation of access delay time of all image servers while access delay time in clustering is the access delay of servers only that maintain images belonging to the category of the query. This can be concluded as fewer numbers of migrations of mobile agents used to execute the tasks will cause lower network traffic and consume less bandwidth, and the total time taken to retrieve the query results is minimized.

IV. Fast Information Retrieval by using High Speed Neural Networks

Finding a certain information in the input one dimensional matrix is a searching problem. Each position in the input matrix is tested for the presence or absence of the required information. At each position in the input matrix, each sub-matrix is multiplied by a window of weights, which has the same size as the sub-matrix. The outputs of neurons in the hidden layer are multiplied by the weights of the output layer. When the final output is high, this means that the sub-matrix under test contains the required information and vice versa. Thus, we may conclude that this searching problem is a cross correlation between the matrix under test and the weights of the hidden neurons.

The convolution theorem in mathematical analysis says that a convolution of f with h is identical to the result of the following steps: let F and H be the results of the Fourier Transformation of f and h in the frequency domain. Multiply F and H in the frequency domain point by point and then transform this product into the spatial domain via the inverse Fourier Transform. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the frequency domain, speed up in an order of magnitude can be achieved during the detection process [22]. In information retrieval phase, a sub matrix I of size 1xn (sliding window) is extracted from the tested matrix, which has a size 1xN, and fed to the neural network. Let W_i be the matrix of weights between the input sub-matrix and the hidden layer. This vector has a size of 1xn and can be represented as 1xn matrix. The output of hidden neurons h_i can be calculated as follows:

$$h_i = g\left(\sum_{k=1}^{n} W_i(k) I(k) + b_i\right) \tag{5}$$

where g is the activation function and b(i) is the bias of each hidden neuron (i). Eq. 5 represents the output of each hidden neuron for a particular sub-matrix I. It can be obtained to the whole input matrix Z as follows:

$$h_i(u) = g\left(\sum_{k=-n/2}^{n/2} W_i(k)\, Z(u+k) + b_i\right) \tag{6}$$

Eq.6 represents a cross correlation operation. Given any two functions f and d, their cross correlation can be obtained by:

$$f(x) \otimes d(x) = \left(\sum_{y=-\infty}^{\infty} f(x+y) d(y)\right) \tag{7}$$

Therefore, Eq. 7 may be written as follows [22]:

$$h_i = g\left(Z \otimes W_i + b_i\right) \tag{8}$$

where h_i is the output of the hidden neuron (i) and h_i (u) is the activity of the hidden unit (i) when the sliding window is located at position (u) and (u) $\in$ [N-n+1].

Now, the above cross correlation can be expressed in terms of one dimensional Fast Fourier Transform as follows [22]:

$$Z \otimes W_i = F^{-1}\left(F(Z) \bullet F^*\left(W_i\right)\right) \tag{9}$$

F: is the Fast Fourier Transform.

F*: is the conjugate Fast Fourier Transform.

F^{-1}: is the Inverse Fast Fourier Transform.

$\otimes$: is the cross correlation operator.

$\bullet$: is the dot product (element by element) operator.

Hence, by evaluating this cross correlation, a speed up ratio can be obtained comparable to CNNs. Also, the final output of the neural network can be evaluated as follows:

$$O(u) = g\left(\sum_{i=1}^{q} W_o(i)\, h_i(u) + b_o\right) \tag{10}$$

where q is the number of neurons in the hidden layer. O(u) is the output of the neural network when the sliding window located at the position (u) in the input matrix Z. W_o is the weight matrix between hidden and output layer.

The complexity of cross correlation in the frequency domain can be analyzed as follows:

1- For a tested matrix of 1xN elements, the 1D-FFT requires a number equal to $N\log_2 N$ of complex computation steps [23]. Also, the same number of complex computation steps is required for computing the 1D-FFT of the weight matrix at each neuron in the hidden layer.

2- At each neuron in the hidden layer, the inverse 1D-FFT is computed. Therefore, q backward and (1+q) forward transforms have to be computed. Therefore, for a given matrix under test, the total number of operations required to compute the 1D-FFT is $(2q+1)N\log_2 N$.

3- The number of computation steps required by FTDNN is complex and must be converted into a real version. It is known that, the one dimensional Fast Fourier Transform requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ complex additions [23]. Every complex multiplication is realized by six real floating point operations and every complex addition is implemented by two real floating point operations. Therefore, the total number of computation steps required to obtain the 1D-FFT of a 1xN matrix is:

$$\rho = 6(N/2)\log_2 N + 2N\log_2 N \tag{11}$$

which may be simplified to:

$$\rho=5N\log_2 N \quad (12)$$

4- Both the input and the weight matrices should be dot multiplied in the frequency domain. Thus, a number of complex computation steps equal to qN should be considered. This means 6qN real operations will be added to the number of computation steps required by FNNs.

5- In order to perform cross correlation in the frequency domain, the weight matrix must be extended to have the same size as the input matrix. So, a number of zeros = (N-n) must be added to the weight matrix. This requires a total real number of computation steps = q(N-n) for all neurons. Moreover, after computing the FFT for the weight matrix, the conjugate of this matrix must be obtained. As a result, a real number of computation steps = qN should be added in order to obtain the conjugate of the weight matrix for all neurons. Also, a number of real computation steps equal to (N) is required to create butterflies complex numbers ($e^{-jk(2\Pi n/N)}$), where 0<K<N. These (N/2) complex numbers are multiplied by the elements of the input matrix or by previous complex numbers during the computation of FFT. To create a complex number requires two real floating point operations. Thus, the total number of computation steps required for FNNs becomes:

$$\sigma=(2q+1)(5N\log_2 N) +6qN+q(N-n)+qN+N \quad (13)$$

which can be reformulated as:

$$\sigma=(2q+1)(5N\log_2 N)+q(8N-n)+N \quad (14)$$

6- Using sliding window of size 1xn for the same matrix of 1xN pixels, q(2n-1)(N-n+1) computation steps are required when using CNNs for certain information detection or processing (n) input data. The theoretical speed up factor η can be evaluated as follows:

$$\eta=\frac{q(2n-1)(N-n+1)}{(2q+1)(5N\log_2 N)+q(8N-n)+N} \quad (15)$$

FNNs accepts serial input data with fixed size (n). Therefore, the number of input neurons equals to (n). Instead of treating (n) inputs, our new approach is to collect all the input data together in a long vector (for example 100xn). Then the input data is tested by FNNs as a single pattern with length L (L=100xn). Such a test is performed in the frequency domain. Complex-valued neural networks have many applications in fields dealing with complex numbers such as telecommunications, speech recognition and image processing with the Fourier Transform [24]. Complex-valued neural networks mean that the inputs, weights, thresholds and the activation function have complex values. In this section, formulas for the speed up ratio with different types of inputs will be presented. The special case of only real input values (i.e. imaginary part=0) will be considered. Also, the speed up ratio in the case of a one and two dimensional input matrix will be concluded. The operation of FNNs depends on computing the Fast Fourier Transform for both the input and weight matrices and obtaining the resulting two matrices. After performing dot multiplication for the resulting two matrices in the frequency domain, the Inverse Fast Fourier Transform is calculated for the final matrix. Here, there is an excellent advantage with FNNs that should be mentioned. The Fast Fourier Transform is already dealing with complex numbers, so there is no change in the number of computation steps required for FNNs. Therefore, the speed up ratio in the case of FNNs can be evaluated as follows:

1) In case of real inputs

A) For a one dimensional input matrix

Multiplication of (n) complex-valued weights by (n) real inputs requires (2n) real operations. This produces (n) real numbers and (n) imaginary numbers. The addition of these numbers requires (2n-2) real operations. Therefore, the number of computation steps required by CNNs can be calculated as:

$$\theta=q(2n-1)(N-n+1) \quad (16)$$

The speed up ratio in this case can be computed as follows:

$$\eta=\frac{2q(2n-1)(N-n+1)}{(2q+1)(5N\log_2 N)+q(8N-n)+N} \quad (17)$$

The theoretical speed up ratio for searching short successive (n) data in a long input vector (L) using FNNs is shown in Figures 6, 7, and 8. Also, the practical speed up ratio for manipulating matrices of different sizes (L) and different sized weight matrices (n) using a 2.7 GHz processor and MATLAB is shown in Figure 9.

B) For a two dimensional input matrix

Multiplication of (n^2) complex-valued weights by (n^2) real inputs requires ($2n^2$) real operations. This produces (n^2) real numbers and (n^2) imaginary numbers. The addition of these numbers requires ($2n^2$-2) real operations. Therefore, the number of computation steps required by CNNs can be calculated as:

$$\theta=2q(2n^2-1)(N-n+1)^2 \quad (18)$$

The speed up ratio in this case can be computed as follows:

$$\eta=\frac{2q(2n^2-1)(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2)+q(8N^2-n^2)+N} \quad (19)$$

The theoretical speed up ratio for detecting (nxn) real valued submatrix in a large real valued matrix (NxN) using FNNs is shown in Figures 10, 11, 12. Also, the practical speed up ratio for manipulating matrices of different sizes (NxN) and different sized weight

matrices (n) using a 2.7 GHz processor and MATLAB is shown in Figure 13.

2) In case of complex inputs

A) For a one dimensional input matrix

Multiplication of (n) complex-valued weights by (n) complex inputs requires (6n) real operations. This produces (n) real numbers and (n) imaginary numbers. The addition of these numbers requires (2n-2) real operations. Therefore, the number of computation steps required by CNNs can be calculated as:

$$\theta=2q(4n-1)(N-n+1) \quad (20)$$

The speed up ratio in this case can be computed as follows:

$$\eta=\frac{2q(4n-1)(N-n+1)}{(2q+1)(5N\log_2 N)+q(8N-n)+N} \quad (21)$$

The theoretical speed up ratio for searching short complex successive (n) data in a long complex-valued input vector (L) using FNNs is shown in Figures 14, 15, and 16. Also, the practical speed up ratio for manipulating matrices of different sizes (L) and different sized weight matrices (n) using a 2.7 GHz processor and MATLAB is shown in Figure 17.

B) For a two dimensional input matrix

Multiplication of (n^2) complex-valued weights by (n^2) real inputs requires ($6n^2$) real operations. This produces (n^2) real numbers and (n^2) imaginary numbers. The addition of these numbers requires ($2n^2$-2) real operations. Therefore, the number of computation steps required by CNNs can be calculated as:

$$\theta=2q(4n^2-1)(N-n+1)^2 \quad (22)$$

The speed up ratio in this case can be computed as follows:

$$\eta=\frac{2q(4n^2-1)(N-n+1)^2}{(2q+1)(5N^2\log_2 N^2)+q(8N^2-n^2)+N} \quad (23)$$

The theoretical speed up ratio for detecting (nxn) complex-valued submatrix in a large complex-valued matrix (NxN) using FNNs is shown in Figures 18, 19, and 20. Also, the practical speed up ratio for manipulating matrices of different sizes (NxN) and different sized weight matrices (n) using a 2.7 GHz processor and MATLAB is shown in Figure 21.

In practical implementation, the multiplication process consumes more time than the addition one. The effect of the number of multiplications required for CNNs in the speed up ratio is more than the number of multiplication steps required by the FNNs. Also, the variations in Pc clock have an effect on practical computations.

For a one dimensional matrix, from Figures 6,7,8,9,10,11,12, and 13, we can conclude that the response time for vectors with short lengths are faster than those which have longer lengths. For example, the speed up ratio for the vector of length 10000 is faster that of length 1000000. The number of computation steps required for a vector of length 10000 is much less than that required for a vector of length 40000. So, if the vector of length 40000 is divided into 4 shorter vectors of length 10000, the number of computation steps will be less than that required for the vector of length 40000. Therefore, for each application, it is useful at the first to calculate the optimum length of the input vector. The same conclusion can be drawn in case of processing the two dimensional input matrix as shown in Figures 10,11,12,13,18,19,20, and 21. From these Figures, it is clear that the maximum speed up ratio is achieved at image size (N=200) when n=20, then at image size (N=300) when n=25, and at image size (N=400) when n=30. This confirms our previous results presented in [26] on fast subimage detection based on neural networks and image decomposition. Using this technique, it was proved that the speed up ratio of neural networks becomes faster when the input image is divided into many subimages and each subimage is processed in the frequency domain separately using a single fast neural processor. Another point of interest should be noted. In CNNs, if the whole input data (N) is available, then there is a waiting time for each group of (n) input data so that CNNs can release their output for the previous group of (n) data. In contrast, FNNs can process the total N data directly with zero waiting time. For example, if the total (N) input data is appeared at the input neurons, then:

1- CNNs can process only data of size (n) as the number of input neurons = (n).
2- The first group of (n) data is processed by CNNs.
3- The second group of (n) data must wait for a waiting time = τ, where τ is the response time consumed by CNNs for treating each group of (n) input data.
4- The third group of (n) data must wait for a waiting time = 2τ corresponding to the total waiting time required by CNNs for treating the previous two groups.
5- The fourth (n) data must wait for a waiting time = 3τ.
6- The last group of (n) data must wait for a waiting time = $(N-n)\tau$.

As a result, the wasted waiting time in the case of CNNs is $(N-n)\tau$. In the case of FNNs, there is no waiting time as the whole input data (Z) of length (N) will be processed directly and the time consumed is the only time required by FNNs themselves to produce their output.

V. Conclusion

A new fast algorithm for information retrieval has been presented. This has been achieved by performing cross correlation in the frequency domain

between input image and the input weights of FNNs. It has been proved mathematically and practically that the number of computation steps required for the presented FNNs is less than that needed by CNNs. Simulation results using MATLAB has confirmed the theoretical computations. Furthermore, by applying cross correlation, object detection has been become position independent. Moreover, by using neural networks, the object has been detected even with rotation, scaling, noise, distortion or deformation in shape. An approach for image-based information retrieval using intelligent mobile agent has been introduced. The proposed approach relies on using different web crawlers to collect images from the World Wide Web located on different servers; consequently similar images at each image server are clustered together. The mobile agents travel across the image servers and analyze image databases at their resource to perform feature extraction, clustering similar images at each node together to speed up the retrieving process. The advantages of the image-based information retrieval using mobile agents are a natural parallelization of the computation and the reduction of bandwidth by avoiding the transfer of huge intermediate data through the network.

The image based information retrieval system has been tested by three approaches; central indexing, broadcasting and clustering approach. These have been subsequently simulated to quantify their effectiveness in the information retrieval function. It has been concluded that the three approaches did not differ much in discovering the requisite information, but differ in the size of intermediate data transfer and the response time to the user's query. Future work will optimize the routing path of search agent in the system. Search agent will not have to visit image servers by their order list; instead, it will determine its route using an optimization technique. The future work also will concern on investigating the performance of the proposed system using other clustering algorithms and other distance measures.

References

[1] W. Qu, M. Kitsuregawa, and K.Li, "Performance analysis on mobile agent-based parallel information retrieval approaches," ICPADS13th International Conference on Parallel and Distributed Systems, Hsinchu , Vol.1, pp. 1-8, 5-7 Dec 2007.

[2] Y. Lu, P. Gao, R. Lv, and Z. Su**, "**Study of content-based image retrieval using parallel computing technique**,"** Conference on High Performance Networking and Computing Proceedings of the 2007 Asian technology information program's (ATIP's) 3rd workshop on High performance computing in China: solution approaches to impediments for high performance computing**.** pp. 186-191, 2007.

[3] A. W. M. Smeulders, "Content-based image retrieval: the end of the early years". IEEE transaction on Pattern Analysis and Machine Intelligence, vol.22, pp. 1349- 1380, 2000.

[4] M.L. Kherfi, and D. Ziou, "Image Retrieval From the World Wide Web Issues, Techniques, and Systems," *ACM Computing Surveys*, vol. 36, pp. 35–67, 2004.

[5] V. Roth, U. Pinsdorf, and J. Peters, "A Distributed Content Based Search Engine Based on Mobile Code," *Proceedings of the 2005 ACM symposium on Applied computing (session: Agents, interactions, mobility and systems (AIMS)), New Mexico*, pp 66–73, 2005.

[6] H. Liu, G. He, "Shape Feature Extraction of High Resolution Remote Sensing Image Based on SUSAN and Moment Invariant," *cisp*, , 2008 Congress on Image and Signal Processing, Vol. 2, pp. 801-807, 2008.

[7] I. King, C.H. Ng, and K.C. Sia, "Distributed content-based visual information retrieval system on peer-to-peer networks," *ACM TOIS*, vol. 22(3), pp. 477–501, 2004.

[8] B. Luo, X. G. Wang, and X. O. Tang. "A World Wide Web Based Image Search Engine Using Text and Image Content Features." Proceedings of the SPIE Electronic Imaging 2003, Internet Imaging IV, Chinese Univ. of Hong Kong, vol. 5018, pp.123-130 2003.

[9] I. Kompatsiaris, E. Triantafylluo, and M. G. Strintzis, "A World Wide Web Region-Based Image Search Engine,"International Conference on Image Analysis and Processing, IEEE 2001, Palermo , Itale, pp.392-397, 2001.

[10] W.Qu, K.li, and C.zhang , "Efficient Information Retrieval by Dispatching Mobile Agents in Parallel," mue, 2008 International Conference on Multimedia and Ubiquitous Engineering, Busan**,** Korea, vol.00, pp. 73-76, 2008.

[11] W. Qu, H. Shen, and J. Sum, "New Analysis on Mobile Agents Based Network Routing", *Applied Soft Computing Journal*, vol.6(1), pp.108-118, 2005.

[12] A.M. Riad, and H.A. Ali, " New Approach for Information Retrieval based-on Mobile Agent," *Egyptian Computer Journal*, *ISSR*, vol. 10, no.1, pp. 110-123, *Cairo Univ.,* 2002.

[13] C. Timon, Y. Eldon, and A. Chang, "Mobile Agents in Distributed Network Management," July 2003, vol. 46, no. 7, *Communications Of The ACM*.

[14] J. Guan, and B. Zhao, "Distributed Geographic Information Querying Approach based on Mobile Agent," *Proceedings of the Fourth International Conference on Computer and Information Technology (CIT'04) 2004 IEEE,* Wuhan, China, pp. 782-787,2004.

[15] D. Picard, M. Cord, and A. Revel. "CBIR in distributed databases using a multi-agent system,"

In IEEE International Conference on Image Processing (ICIP'06), Atlanta, GA, USA,pp. 3205-3208,2006.

[16] D. Picard, A.Revel and M. Cord . "Performances of mobile-agents for interactive image retrieval," *In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 06)*, IEEE Computer Society, pp. 581–586, 2006.

[17] A. Revel, "Web-agents inspired by ethology: a population of "ant"-like agents to help finding user-oriented information.," in *IEEE WIC'2003 : International Conference on Web Intelligence.*, Halifax, Canada, October 2003, IEEE Computer Society, pp. 482–485, 2003.

[18] A. Revel, "From robots to web-agents: Building cognitive software agents for web-information retrieval by taking inspiration from experience in robotics," in *ACM International conference on Web Intelligence*, France, pp. 434–437, 2005.

[19] G. Pass, R. Zabih, and J. Miller. "Comparing images using color coherence vectors". In *Proc. ACM Conference on Multimedia*, Boston, Massachusetts, U. S. A., November 1996.

[20] J. A. Hartigan, and M.A. Wong, "Algorithm AS136: A k means clustering Algorithm", Applied Statistics, Vol. 28, pp. 100-108, 1979.

[21] Dmoz open directory project, http://www.dmoz.org, downloaded 16 June 2008.

[22] H. M. El-Bakry, "New Faster Normalized Neural Networks for Sub-Matrix Detection using Cross Correlation in the Frequency Domain and Matrix Decomposition," Applied Soft Computing journal, vol. 8, issue 2, March 2008, pp. 1131-1149.

[23] J. W. Cooley, and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput. 19, 1965, pp. 297–301.

[24] A. Hirose, "Complex-Valued Neural Networks Theories and Applications", Series on innovative Intelligence, vol.5. Nov. 2003.

[25] A. M. Riad, A. Atwan, and S. Abd El-Ghany, "Analysis of Performance of Mobile Agents in Distributed Content Based Image Retrieval," The 2008 International Conference on Computer Engineering & Systems (ICCES'08), November 25 - 27, 2008.

[26] H. M. El-Bakry, "Face detection using fast neural networks and image decomposition," *Neurocomputing Journal*, vol. 48, 1039-1046, 2002.

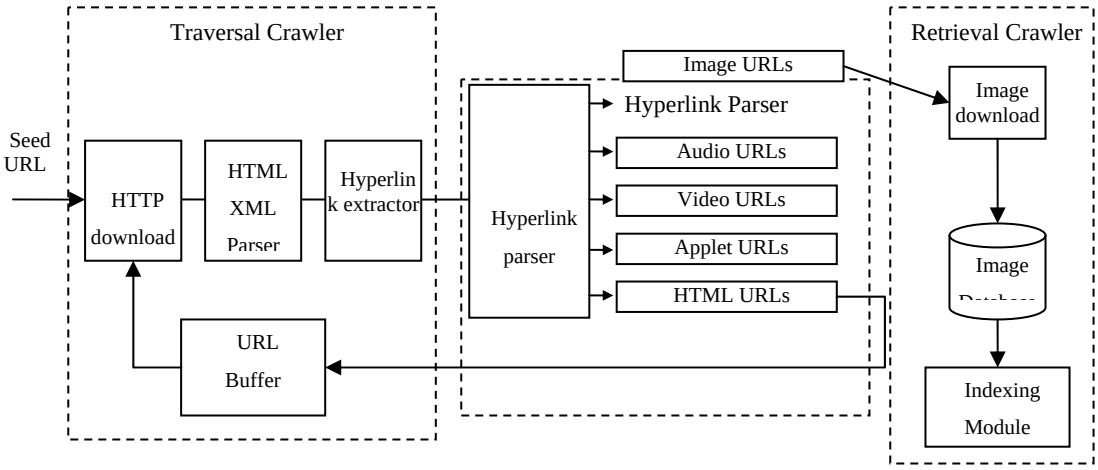

Fig. 1. Information Crawler Components.

Fig. 2. Image-based information retrieval.

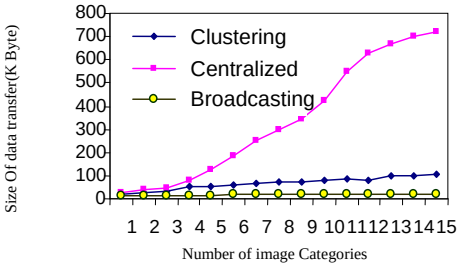

Fig. 3. Size of intermediate data vs. Increase in image categories

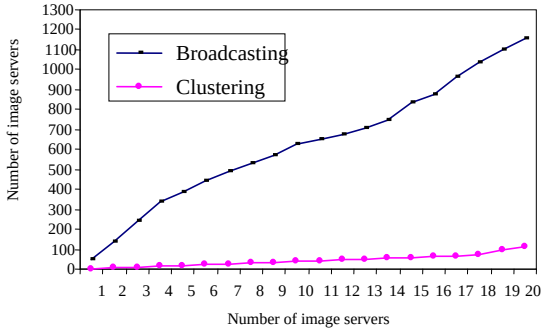

Fig. 4. Number of comparison in clustering and broadcasting approaches

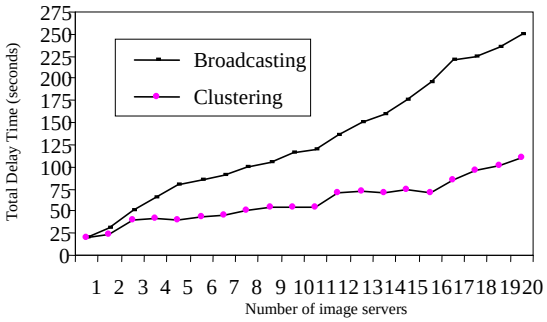

Fig. 5. The total access delay vs. number of image servers

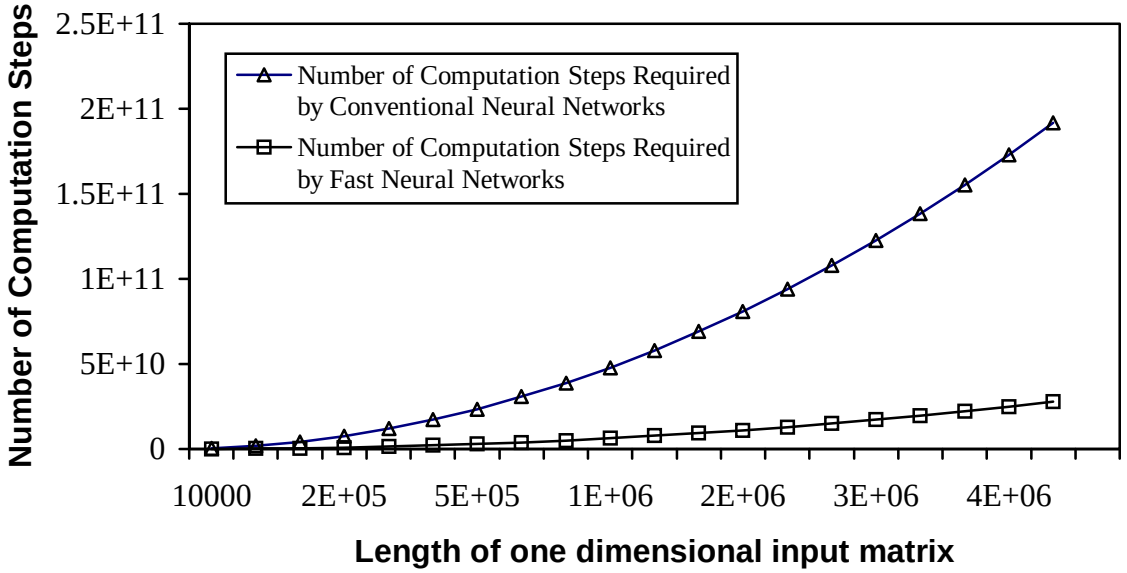

Fig. 6. A comparison between the number of computation steps required by FNNs and CNNs in case of real-valued one dimensional input matrix and complex-valued weight matrix (n=400).

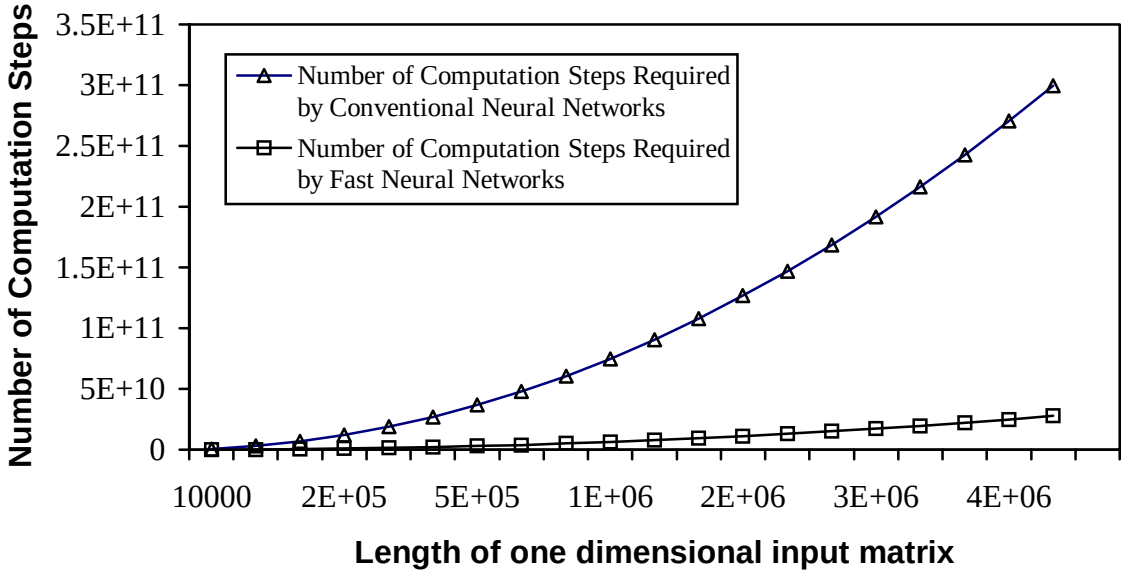

Fig. 7. A comparison between the number of computation steps required by FNNs and CNNs in the case of real-valued one dimensional input matrix and complex-valued weight matrix (n=625).

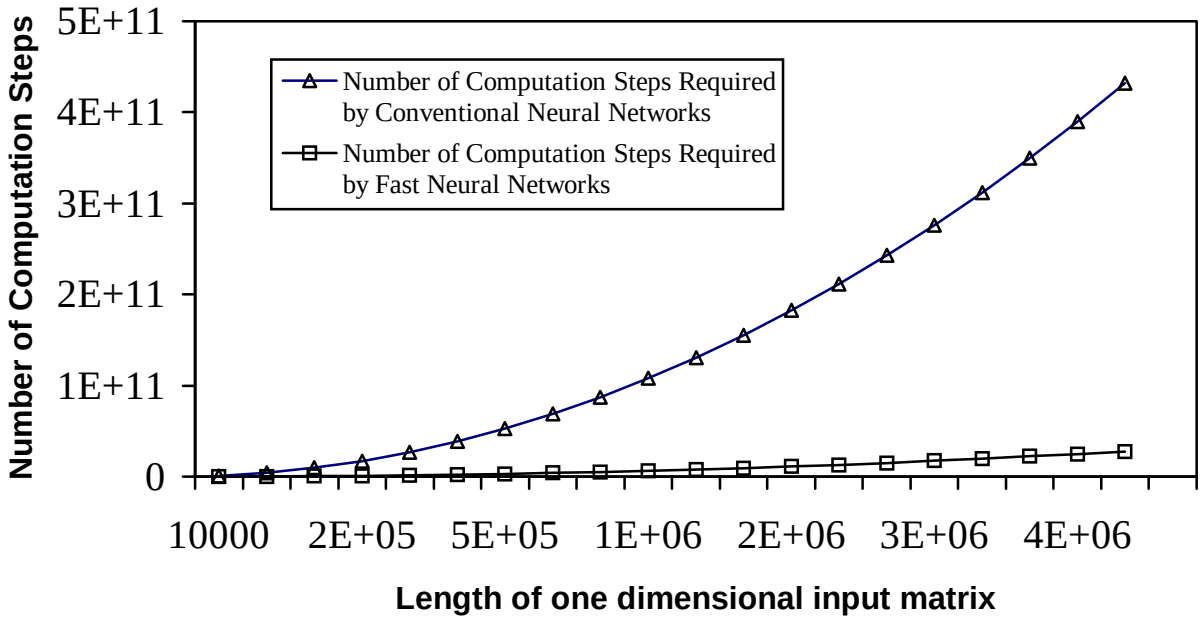

Fig. 8. A comparison between the number of computation steps required by FNNs and CNNs in the case of real-valued one dimensional input matrix and complex-valued weight matrix (n=900).

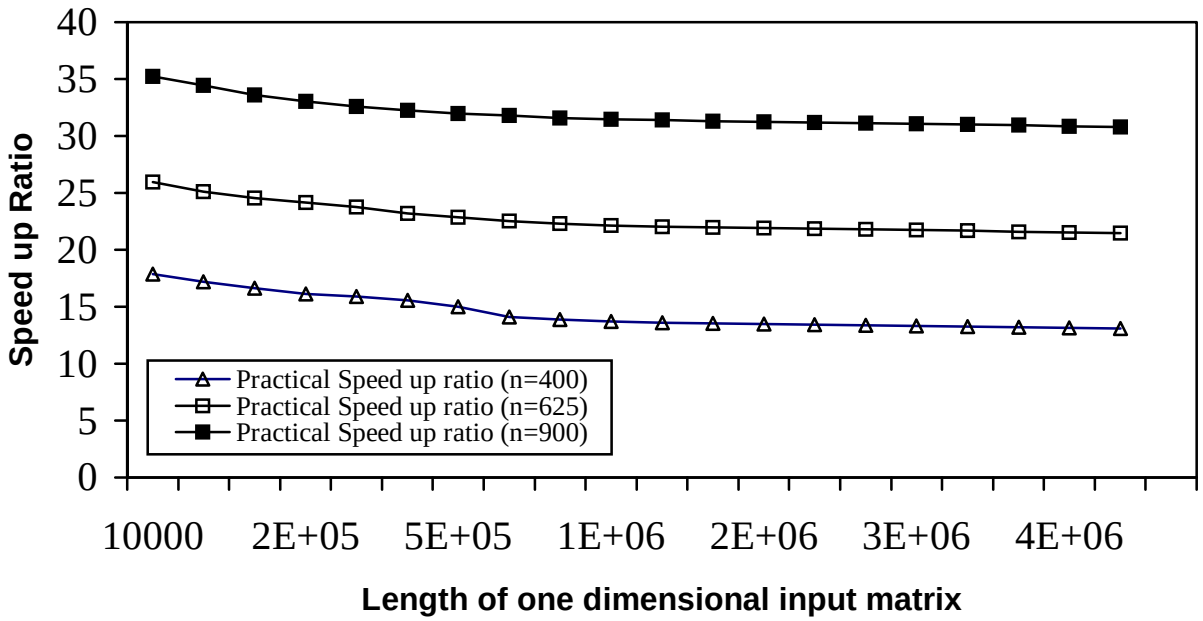

Fig. 9. Practical speed up ratio for neural networks in case of one dimensional real-valued input matrix and complex-valued weights.

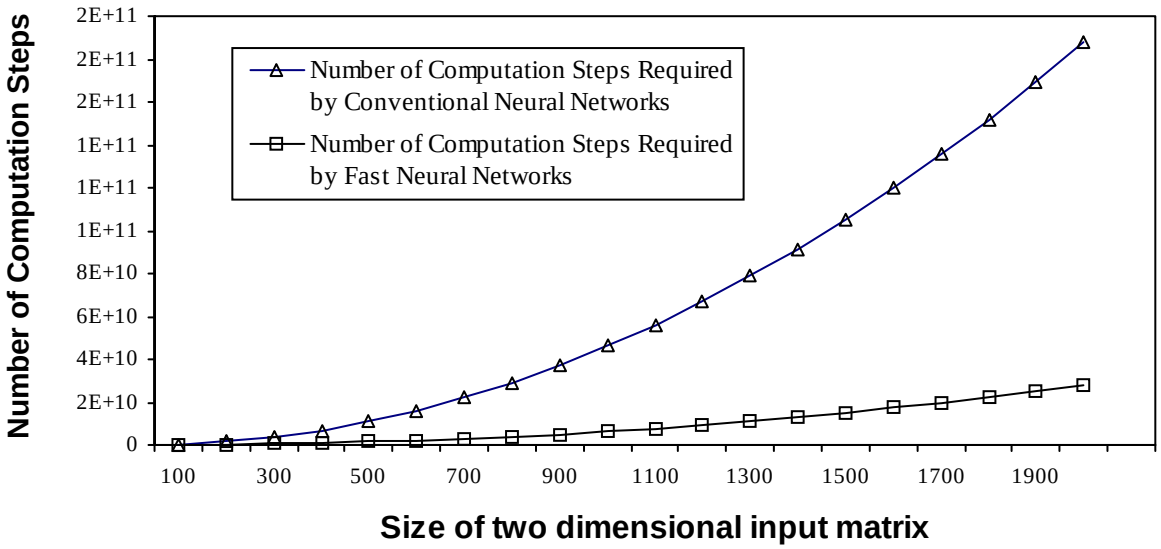

Fig. 10. A comparison between the number of computation steps required by FNNs and CNNs in the case of real-valued two dimensional input matrix and complex-valued weight matrix (n=20).

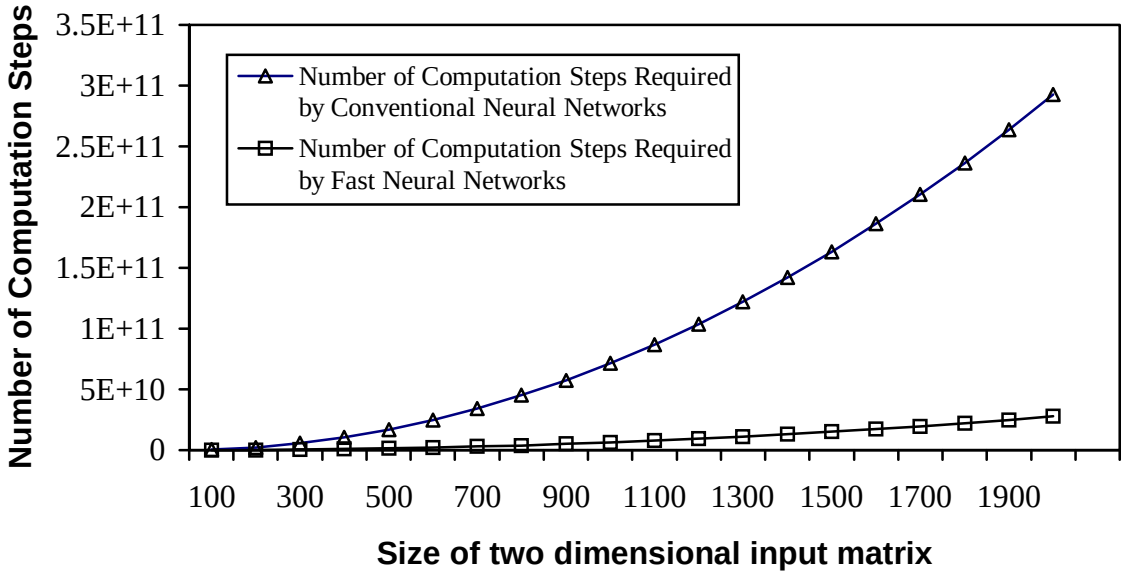

Fig. 11. A comparison between the number of computation steps required by FNNs and CNNs in the case of real-valued two dimensional input matrix and complex-valued weight matrix (n=25).

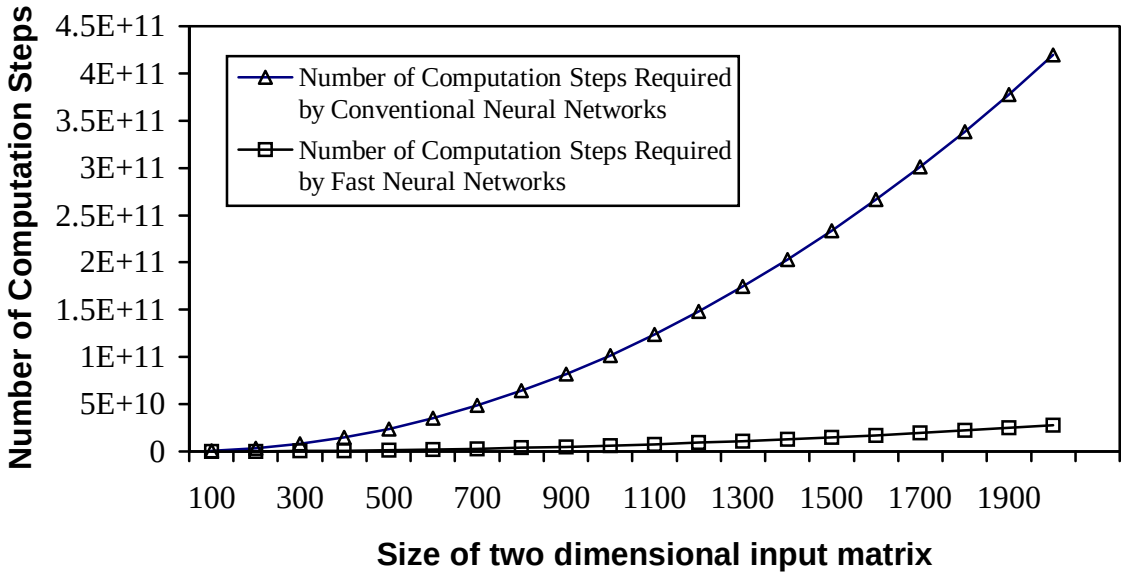

Fig. 12. A comparison between the number of computation steps required by FNNs and CNNs in the case of real-valued two dimensional input matrix and complex-valued weight matrix (n=30).

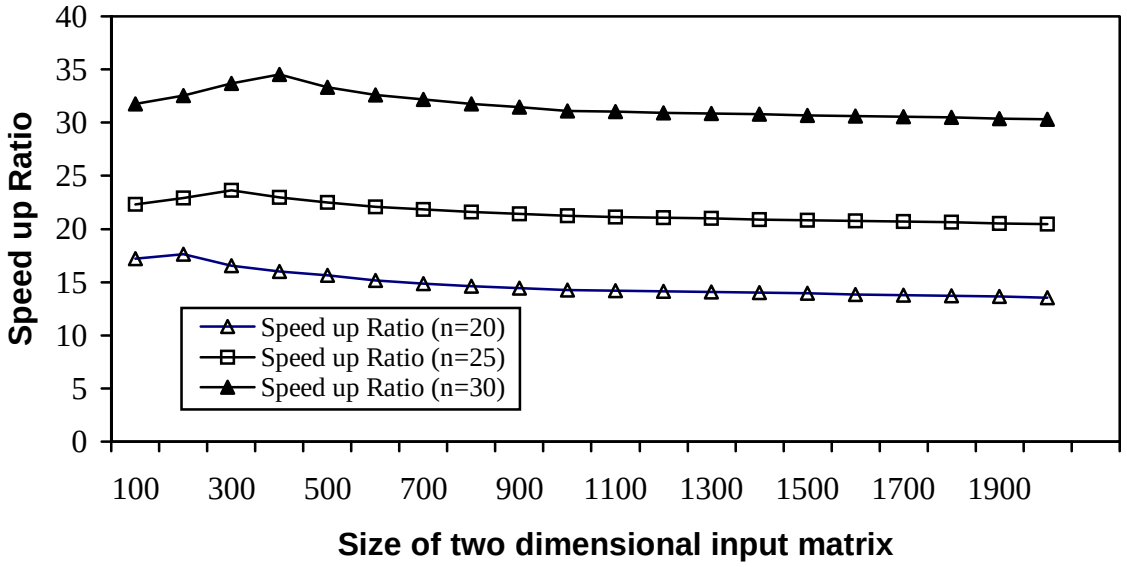

Fig. 13. Practical speed up ratio for neural networks in case of two dimensional real-valued input matrix and complex-valued weights.

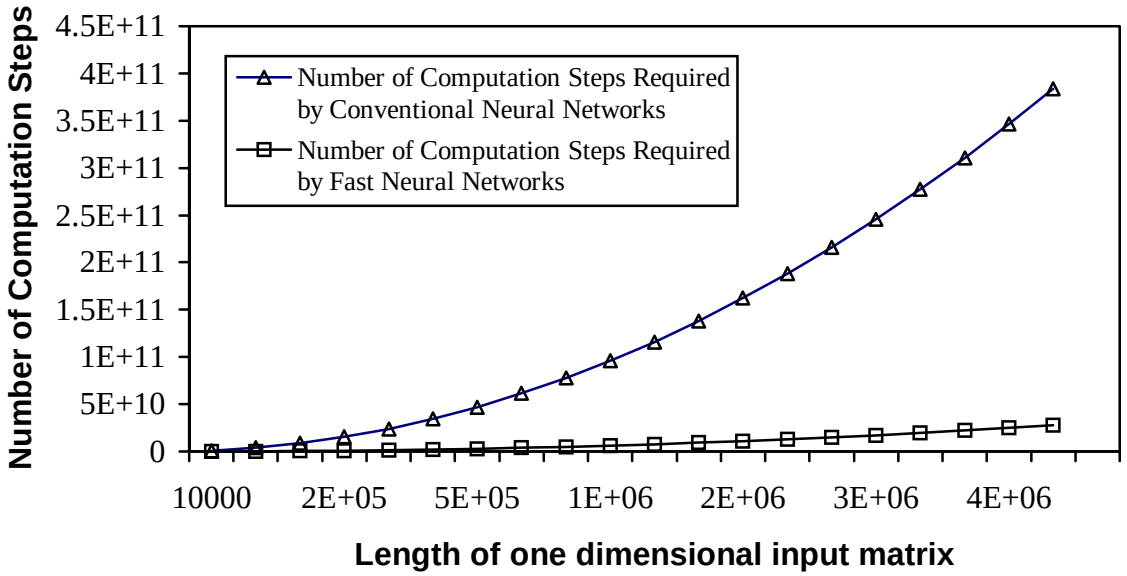

Fig. 14. A comparison between the number of computation steps required by FNNs and CNNs in the case of complex-valued one dimensional input matrix and complex-valued weight matrix (n=400).

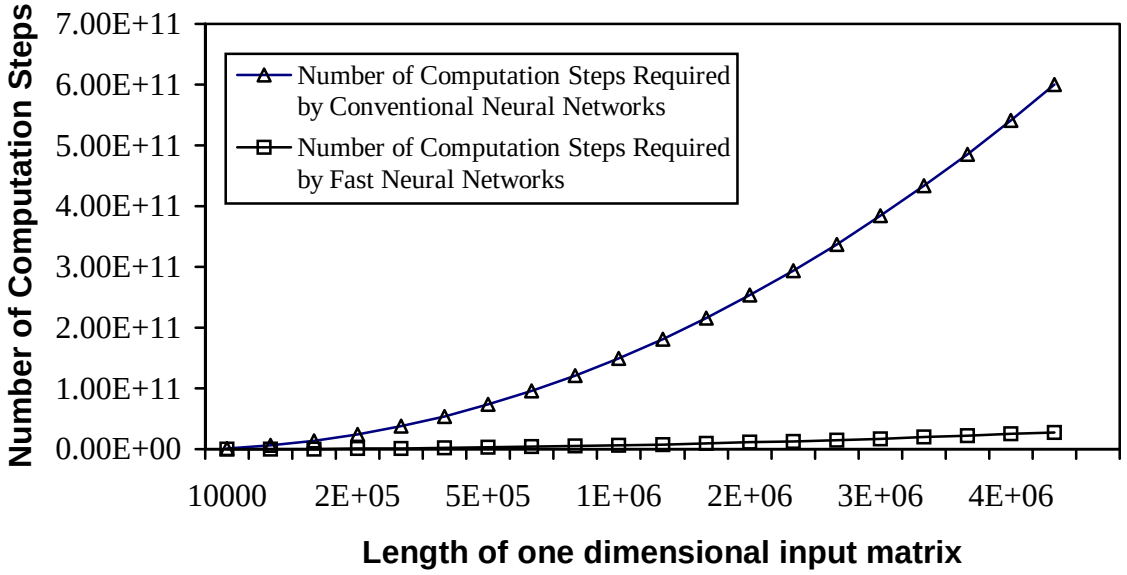

Fig. 15. A comparison between the number of computation steps required by FNNs and CNNs in the case of complex-valued one dimensional input matrix and complex-valued weight matrix (n=625).

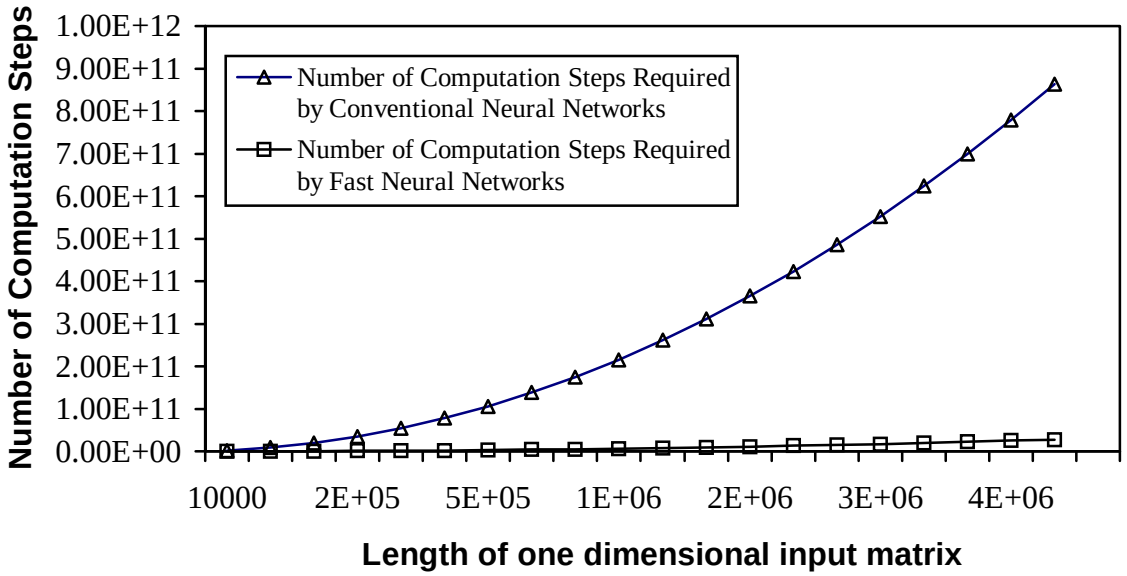

Fig. 16. A comparison between the number of computation steps required by FNNs and conventional neural networks in the case of complex-valued one dimensional input matrix and complex-valued weight matrix (n=900).

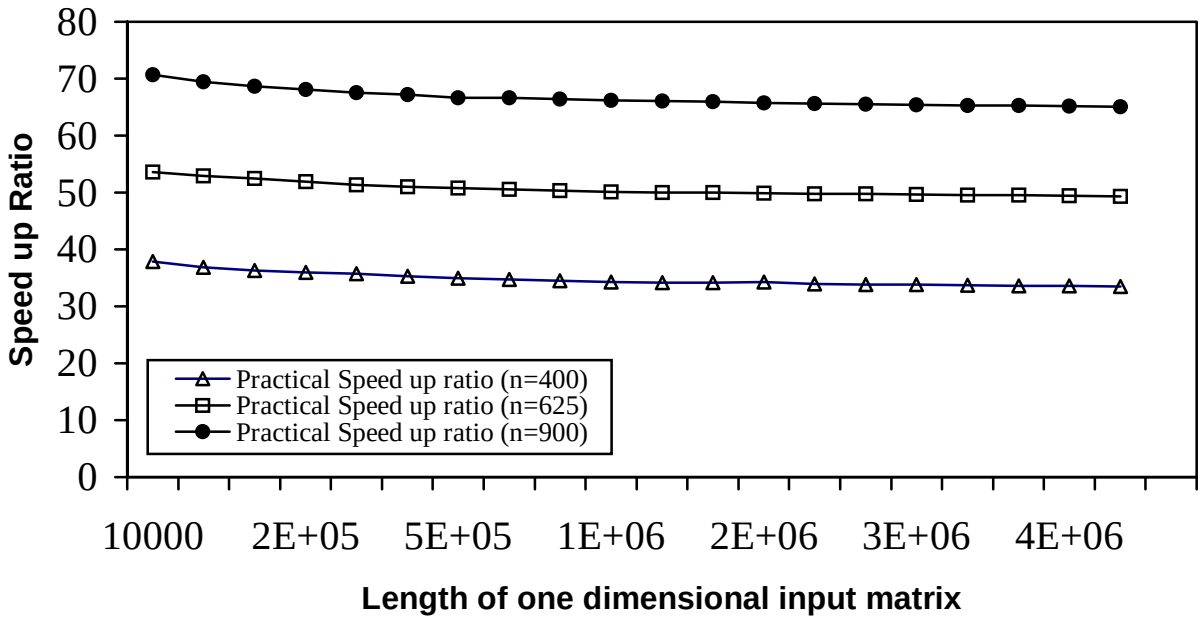

Fig. 17. Practical speed up ratio for neural networks in case of one dimensional complex-valued input matrix and complex-valued weights.

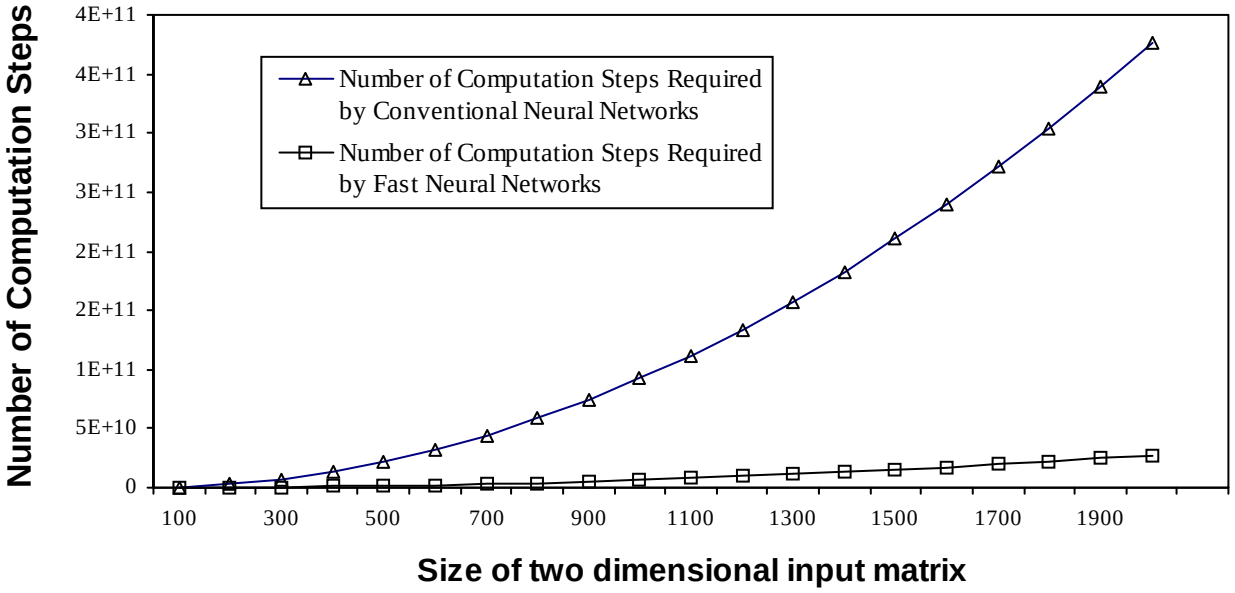

Fig. 18. A comparison between the number of computation steps required by FNNs and CNNs in the case of complex-valued two dimensional input matrix and complex-valued weight matrix (n=20).

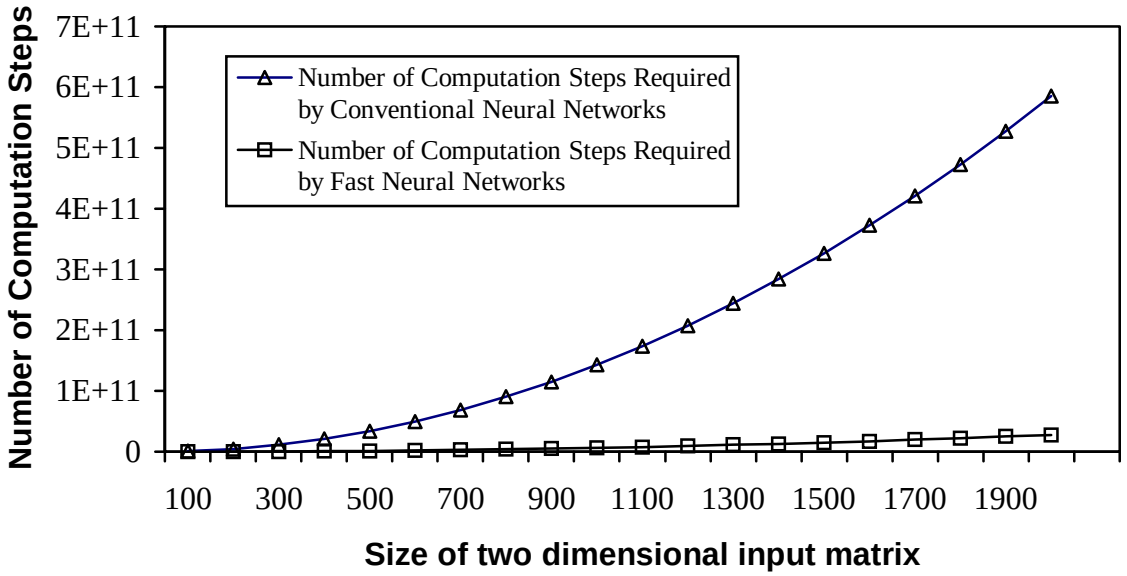

Fig. 19. A comparison between the number of computation steps required by FNNs and CNNs in the case of complex-valued two dimensional input matrix and complex-valued weight matrix (n=25).

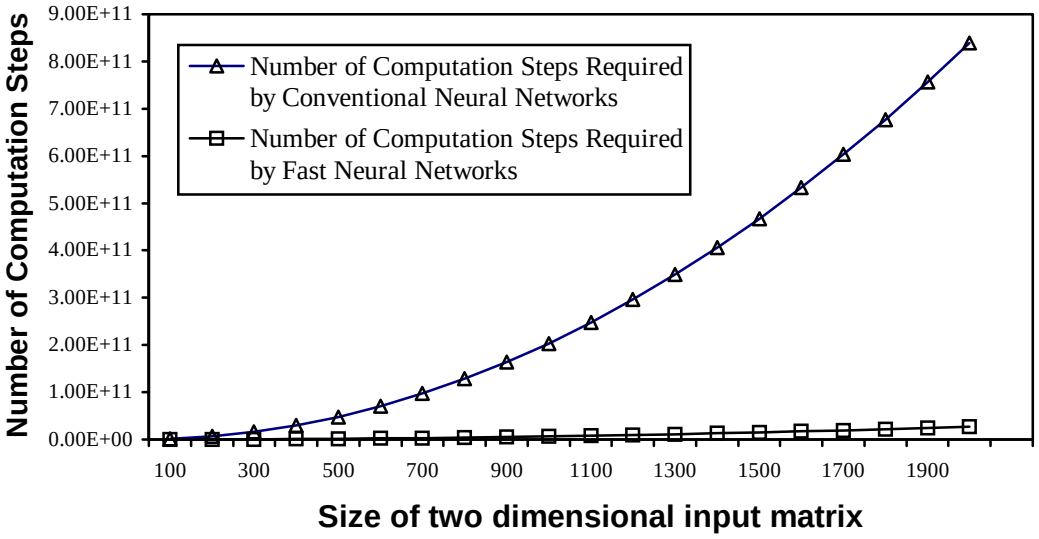

Fig. 20. A comparison between the number of computation steps required by FNNs and CNNs in the case of complex-valued two dimensional input matrix and complex-valued weight matrix (n=30).

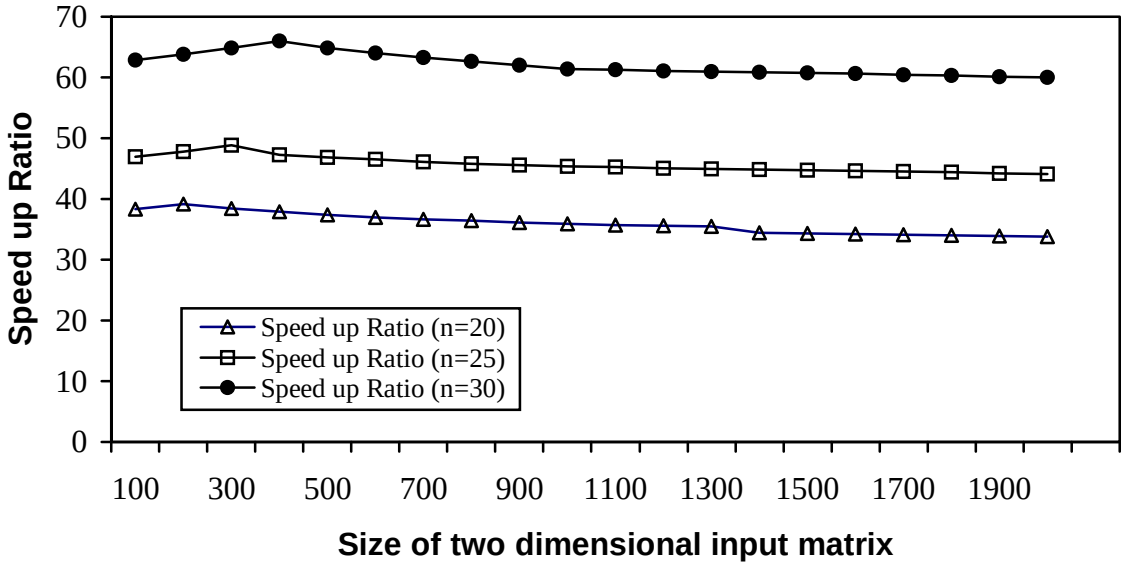

Fig. 21. Practical speed up ratio for neural networks in case of two dimensional complex-valued input matrix in and complex-valued weights.