In the domain of visual question answering (VQA), the computational task involves presenting a computer with an image along with a relevant question, eliciting a response that adequately addresses the inquiry. A longstanding objective in the sphere of artificial intelligence research has been the development of machines capable of comprehending visual content and providing responses akin to human understanding. Notably, visual question answering (VQA) has emerged as a prominent academic discipline. Within the specific domain of medical visual question answering (Med-VQA), clinical queries are accompanied by radiological images, aiming to devise a system that can accurately generate responses based on the visual data contained in the images.

3.1 Description of the research design and methodology used

The system architecture for a Visual QA (VQA) system in the context of skeletal images typically involves several key components designed to process and understand both skeletal image and textual knowledge in the current era. Here's an overview of a generalized architecture with Input module their Image and question as the input, This module handles the medical images. It may utilize various techniques like Convolutional NN (CNNs) for visual feature extraction. Pre-trained models like ResNet or VGG might be employed for this purpose. The textual input includes questions related to the medical images. Natural Language Processing (NLP) approaches are often used to preprocess and encode the textual data. Image and text features are extracted separately. For images, CNNs are employed to capture visual patterns and features. For text, embedding layers and recurrent networks like Long Short-Term Memory (LSTM) or bidirectional LSTMs may be used to understand the context and relationships in the question.

Multimodal fusion pertains to the amalgamation of extracted features from both modalities, namely images and text. This fusion can occur at various levels, encompassing concatenative fusion, which requires integrating features at a basic level, and subsequent fusion, which integrates features at a higher, more abstract level. Techniques for multimodal fusion include concatenation, element-wise multiplication, and attention processes. The resultant fused features may undergo further processing, potentially within additional neural network layers. This stage facilitates the comprehension of intricate correlations between visual and textual

information. Ultimately, the final layer is tasked with making predictions or guesses, with a softmax layer commonly employed in classification challenges. In regression tasks, a linear layer is commonly employed. The training of the entire system involves utilizing a dataset consisting of paired medical images, associated questions, and corresponding responses. During training, the network's parameters are optimized to minimize the disparity between expected and actual answers. Following training, the system undergoes evaluation on a separate dataset to assess its effectiveness, frequently assessed using measures like accuracy, precision, recall, and F1 score.

Post-processing techniques may be implemented based on specific needs, which could involve refining the answer or providing additional context.

This design is adaptable and can be tailored to unique needs and the nature of the medical VQA assignment at hand. Various model topologies, pre-processing processes, and fusion procedures can be investigated to optimize performance.

This section underscores the diverse visual feature extraction methods available for distinct datasets, with a specific emphasis on medical radiological images. The primary objective of the research is to proficiently train models using a variety of medical images, queries, and associated responses. To accomplish this, two distinct feature extraction methodologies were employed for the dataset.

The feature extraction process involves two pivotal components—visual and textual. In the context of this proposal, the focus is directed towards medical radiological images. The datasets are divided into a 70:30 training-to-testing ratio, which means that 70% of the dataset is allocated for training, while the remaining 30% is allocated for testing.

There are various models of processing mentioned in this chapter, which are highlighted below.

User Interaction: The user initiates the process by inputting a medical image into the model along with relevant questions.

Image Feature Extraction: The system meticulously extracts features from the input images, diligently searching for distinctive patterns.

Text Feature Extraction: Simultaneously, the text entered as a question undergoes feature extraction, ensuring that relevant textual features are identified.

Question Classification and Prediction: The loaded question is subjected to classification, and the model employs predictive analysis to furnish accurate responses.

This formal description elucidates the systematic approach adopted in the research, encompassing the selection of feature extraction methods, dataset distribution, and the intricate processes involved in model training and prediction.

3.1.1 Morphology of a Radiology Image

Radiology is a specialized field within the medical profession that employs various imaging techniques to detect, diagnose, and address a spectrum of disorders [63]. Within radiology, there are two key subspecialties: diagnostic radiology and interventional radiology [64].

Diagnostic Radiology:

Diagnostic radiology enables radiologists to meticulously examine internal bodily structures, facilitating the identification of the root causes of symptoms, screening for health issues, and monitoring the body's response to treatment. Common modalities in diagnostic radiology encompass plain radiographic images, computed axial tomography (CT), magnetic resonance tomography, positron emission tomography (PET), and ultrasound imaging [65]. These modalities are adept at visualizing a diverse range of ailments, including but not limited to breast cancer, colon cancer, and heart disease.

Diagnostic Imaging Modalities:

CT (Computerised Tomography): Often referred to as CAT (Computerised Axial Tomography), CT is a widely utilized diagnostic radiology examination. It includes various applications such as CT angiography, fluoroscopy with upper gastrointestinal (GI) studies, MRI (Magnetic Resonance Imaging) and MRA (Magnetic Resonance Angiography) scans, mammography, bone scans, thyroid scans, plain x-rays, PET (Positron Emission Tomography) images, PET scans, PET-CT scans, and ultrasound [64, 31].

System Functionality:

When a user submits a medical image to the system, the system conducts a comprehensive comparison with its database. Subsequently, the system provides the user with pertinent information crucial for guiding the next phase of medical analysis and decision-making. This formal elucidation underscores the significance of diagnostic radiology in medical practice, elucidating its various modalities and their essential roles in disease detection and characterization.

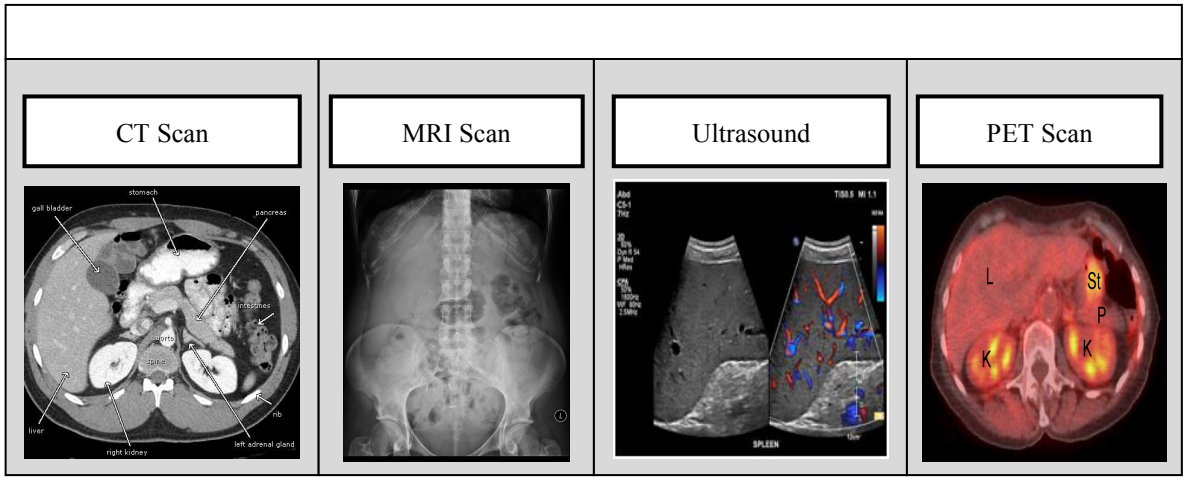

Fig 6 : Types of Radiology Image

The above figure indicates the different types of radiology images used for VQA systems. Each image describes a different description in a different angle. Based on the image the modality of question and answer will change. Fig (b) is from [66] which has lots of information about it, same for other images too. So collect a huge amount of information regarding images, question and answer. Remaining chapters elaborate the significance and methodology of current vqa system, architecture, uses, its error rate, accuracy. Then finally our proposed system projects the limitations of other models and introduces the new model to answer the question with high ratings.

Specifically, many Visual or imaginary QA models have focused on how they incorporate question and skeletal inputs into the model. Various Visual or image QA techniques were reviewed in the previous chapters. Different datasets used for various methodology, based on the problem state, the models will differ.

3.2 Explanation of data collection methods, tools, and procedures

As of the last knowledge update in September 2022, several Visual or imaginary QA (VQA) models have been proposed and applied in the healthcare sector. However, the field is dynamic, and new models may have been developed since then. Here are some notable VQA models that were relevant in healthcare. MedVQA is specifically designed for radiology visual question answering. The system employs convolutional neural networks (CNNs) to analyze images and recurrent neural networks (RNNs) to handle textual queries. Inspired by BERT (Bidirectional Encoder Representations from Transformers), Medical-BERT utilizes transformer-based models to respond to queries regarding both textual and visual content in medical images. The MQR-VQA (Multi-Modal Quality-Aware Relevance) model aims to assess the relevance and quality of regions in medical images to answer questions. It combines visual attention mechanisms with textual information. The MedVQA-Transformer model employs transformer architectures for processing both visual and textual information. Recognized for their effectiveness in capturing long-range dependencies in sequences. DeepMedQA uses a combination of deep learning techniques, including CNNs for image processing and recurrent networks for text. It is designed to answer questions related to medical images. The Attention-Gated CNN-LSTM model integrates attention mechanisms into a network composed of convolutional neural networks (CNNs) and long short-term memory (LSTM) units. The attention mechanism directs the model's attention to relevant sections of medical imagery. Healthcare Image QA is a framework developed for skeletal visual or image question answering systems. It utilizes deep learning models to process both visual and textual data, enabling the model to answer questions about medical images. Remember to check for recent publications, conferences, or preprint archives for the latest advancements in the field. Additionally, the specific requirements of a healthcare application may lead to the development of specialized models tailored to particular tasks or medical domains.

In this section, we are looking forward to current methodologies such as Linear classifier, k-nearest neighbors algorithm, Softmax classifier, support vector machine, Convolutional Neural Network (CNN), regions with convolutional neural networks (R-CNN), fast regions with convolutional neural networks (FR-CNN), Faster regions

with convolutional neural networks (FR-CNN), Deep Belief Networks (DBN), Long short-term memory (LSTM) AND Bidirectional Long short-term memory (BiLSTM).

3.3 Current Methodology on Both Visual and Textual Feature Extraction Techniques

3.3.1 Linear Classifier

The purpose of this document is to investigate and analyze the application of linear classifiers in the domain of skeletal imaginary questions and answers specifically tailored to medical images. The focus will be on understanding how linear classifiers, known for their simplicity and interpretability, can contribute to answering questions related to complex medical visuals. Through a thorough exploration, we aim to uncover the strengths, limitations, and potential advancements that linear classifiers bring to the challenging task of VQA within the medical context. This exploration encompasses the entire pipeline, from the extraction of features from healthcare related images to the design, training, and optimization of linear classifiers, with a keen eye on their practical applications and performance evaluation metrics in the medical domain. Ultimately, this document seeks to provide insights into the role of linear classifiers as a valuable tool in enhancing the interpretability and accuracy of VQA systems when dealing with medical imagery.

Linear classifiers are a category of machine learning models designed to categorize input data points into discrete groups or classes. The fundamental concept behind linear classifiers is based on the idea of drawing a decision boundary in the feature space, which separates the data instances belonging to various classes. This decision boundary is a hyperplane, a subspace with one dimension less than the input space, and is determined by a set of parameters.

The linear classifier's decision-making process involves categorizing an input based on which side of the decision boundary the point falls on. The decision boundary is defined by a linear combination of the input features, each multiplied by a weight, plus an additional bias term. Mathematically, it can be represented as:

$$f(x) = sign(\sum_{i=1}^{n} w_i . x_i + b) \quad (1)$$

Here:

- f(x) is the decision function.
- x_i represents the input features.
- w_i represents the weight associated with each feature.
- B represents the intercept or bias term
- Sign is the sign function, determining the class label based on the sign of the expression.

Training a linear classifier entails determining the best values for the weights and bias. This is typically achieved through optimization algorithms that aim to minimize a certain objective function, often associated with the misclassification of training data. Key characteristics of linear classifiers include simplicity, interpretability, and efficiency. They work well when the relationship between features and classes is approximately linear. However, they may struggle with more complex, non-linear relationships in the data. Linear classifiers are frequently employed across diverse applications, including picture classification, text categorization, and, depending on your environment, visual or picture question answering (VQA). In the healthcare domain, linear classifiers can be useful for tasks such as disease diagnosis using visual data.

3.3.2 Traditional Neural Network

Suppose if the image pixel size is 1000 × 1000, then their will be a total of 3 (rgb size) × 1000 × 1000 features, which means 3 million input features are their. The first layer of the neural network will have 1000 input characteristics that reflect the neural network's weight, or a connected layer of two layers. So the whole amount of weight is given below.

No. of weights = $3 \times 10^6 \times 10^3$ which means 3×10^9

So, there are 3 billion weight parameters in a single layer.

The size is too large to load into the system and model. Also, it is very difficult to manage using laptops or our PCs because of this drawback. The time requirement to train the model is very high. The traditional neural network is overfitting for the

object detection concept. To overcome this drawback, a new technique arrived on the market that we called the Convolutional Neural Network (CNN).

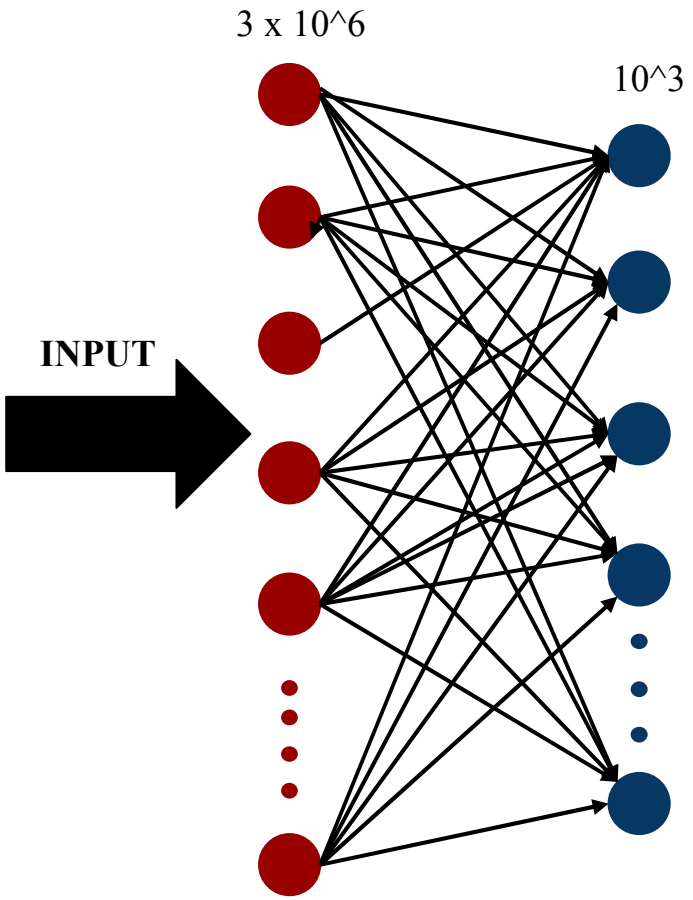

Fig 7 : Traditional Neural Network with a Single Layer

3.3.3 The prime Idea behind a Convolutional Neural Network for image feature extraction

The primary idea behind a neural network that uses convolution is to use filters. These filters are located on sliding windows. This filter is tasked with identifying the characteristics of objects and patterns in the image

Fig: 8 : RGB Image and Gray scale Image to find features of the image

The features of the above image are Shape, Size, Edges, Colors etc. We can use the images with vertical edge detector and horizontal edge detector, when combine then it will be 3 × 3 pixel filter which means 9 pixel size, Thus we reduce the number of parameters from the objects.

1	0	-1
1	0	-1
1	0	-1

Fig 9 : 3 × 3 pixel object

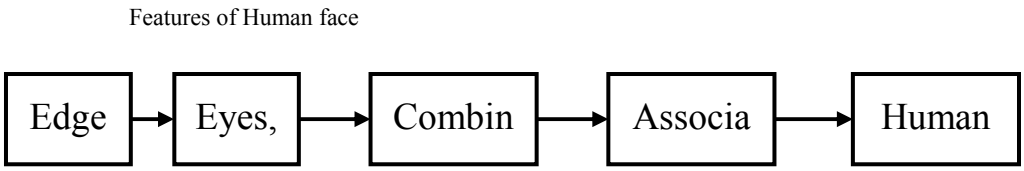

Fig 10 : Structure of CNN process from Input object to final output object with feature extraction

The RGB image converts to a black and white image; every pixel value starts at 0, which means black color, and ends with 255, which means white.

Fig 11 : The value of Black and white colors 0 to 255

Example:

6 ✕ 6 Pixel images convolution operation with 3 ✕ 3 (RGB), then the Output Image size will be 4 ✕ 4 Pixel.

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

*

1	0	-1
1	0	-1
1	0	-1

=

-8	-9	-2	14
6	-3	-13	9
4	-4	-8	5
2	2	1	-3

Fig 12 : How the convolution operation take place with 6 ✕ 6 Pixel images and 3 ✕ 3 image

The output result calculated as for first pixel is (1 ✕ 1) + (2 ✕ 1) + (3 ✕ 1) + (6 ✕ 0) + (5 ✕ 0) + (7 ✕ 0) + (9 ✕ (-1)) + (1 ✕ (-1)) + (4 ✕ (-1)) = -8

$(n \times n) * (f \times f) = (n - f + 1) \times (n - f + 1)$.

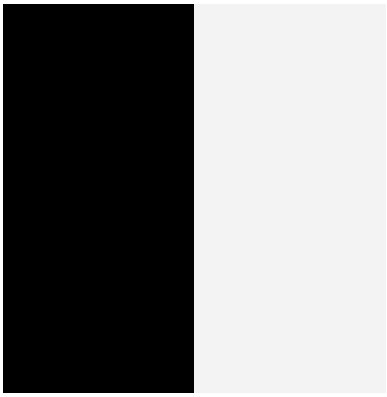

Fig 13 : Black and white input image for convolution operation to identify the vertical edge

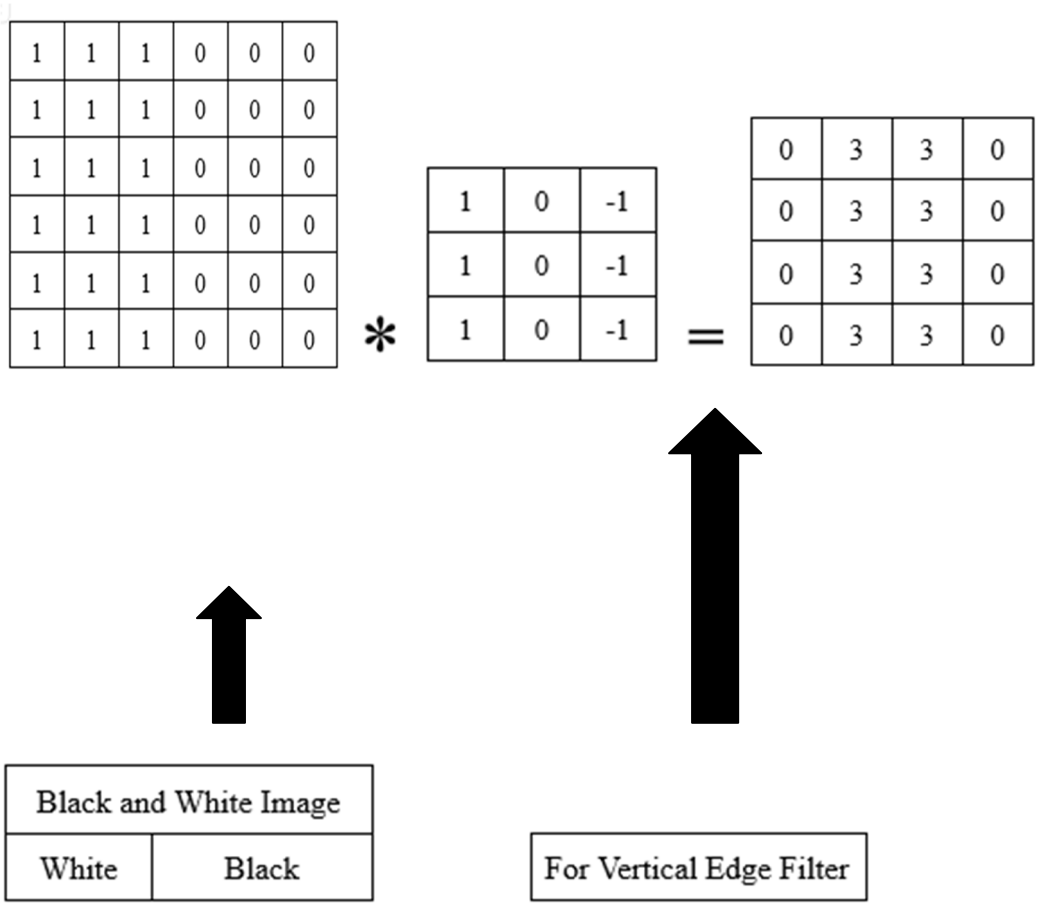

Fig 14 : Black and White image with 3 ✕ 3 px vertical edge filter the output is in 4 ✕ 4px

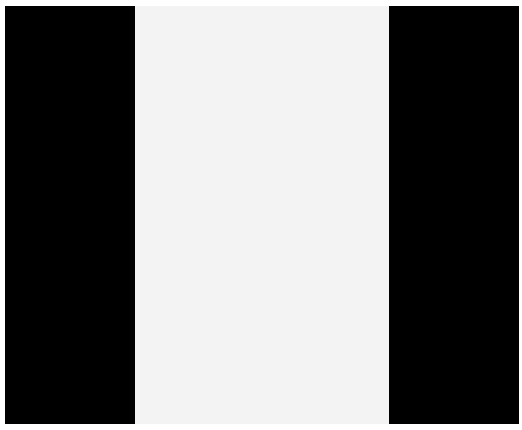

Fig 15 : Output image from above figure after 3 ✕ 3 px convolution with 6 ✕ 6 px

The output result calculated as for first pixel is (1 ✕ 1) + (1 ✕ 1) + (1 ✕ 1) + (1 ✕ 0) + (1 ✕ 0) + (1 ✕ (-1)) + (1 ✕ (-1)) + (1 ✕ (-1)) = 0

The remaining calculation will be done the same way for other pixels.

1	1	1
0	0	0
-1	-1	-1

Fig 16 : Image 3 ✕ 3 px horizontal edge filter

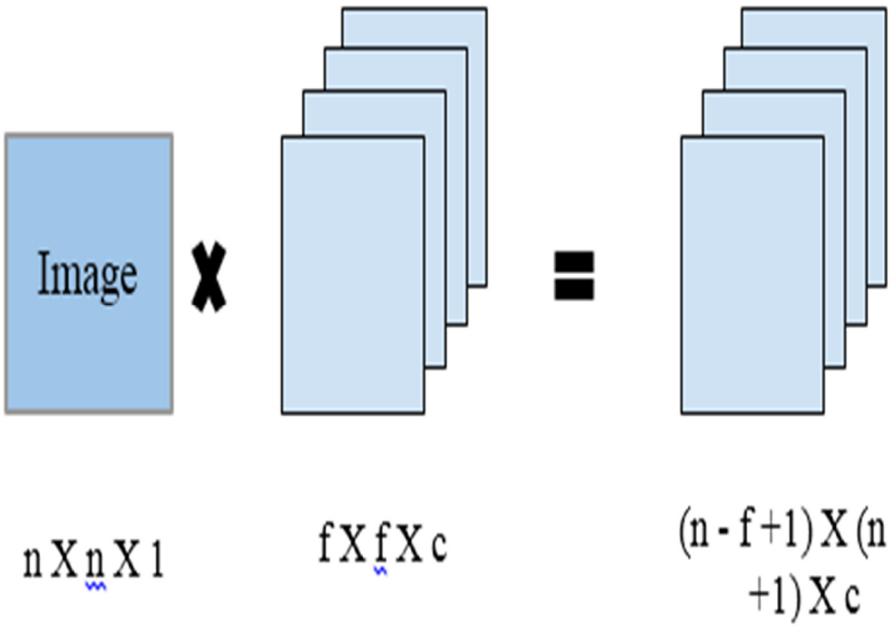

Fig: 17. n ✕ n ✕ 1 for Gray scale image with f ✕ f ✕ c number of filters the output will be multiple images.

When dealing with a colored or RGB image with dimensions n×n×3, the convolution process involves using a three-channel filter with dimensions f×f×3. Each channel of

the filter aligns with the corresponding channel of the input image. The filter, being a 3D cube, consists of a total of 27 values (3 values per channel, totaling 9 values per layer). To perform convolution between the input image and the filter, we overlay the two images and multiply the values in each corresponding cell. These products are then added together to form a single output value, which is displayed in the first cell of the output image. This operation is repeated, shifting the filter one pixel to the right until the entire input image is covered.

When converging an n×n×3 image with a f×f×3 filter, the resulting output image will have dimensions (n−f+1)×(n−f+1). This output represents a single image resulting from the convolution operation. In a convolutional neural network (CNN), multiple such filters are utilized in a single layer. Each filter may have different parameter values, which are acquired throughout the training procedure. For instance, in the example provided, a filter with hardcoded values such as (1, 1, 1), (0, 0, 0), and (-1, -1, -1) may behave as a vertical filter.

However, during training, these parameter values will adjust autonomously, allowing the filters to recognize relevant features in the images. The process of parameter adjustment during training enables the filters to effectively identify and extract meaningful patterns from the input data.

In convolutional neural networks (CNNs), the parameters of the filters are adjusted during the training process, enabling them to adapt and identify specific features within input images. This capacity allows CNNs to accomplish tasks like image categorization, object identification, and image segmentation.

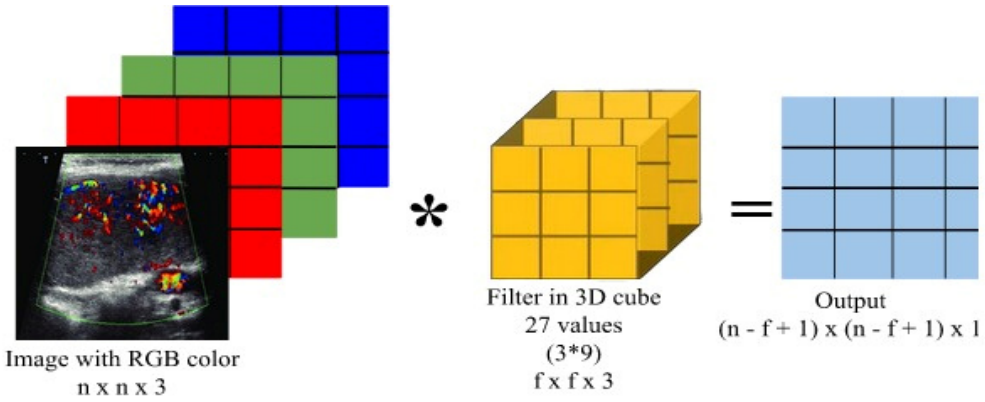

Fig: 18. Single filter convolution with single output

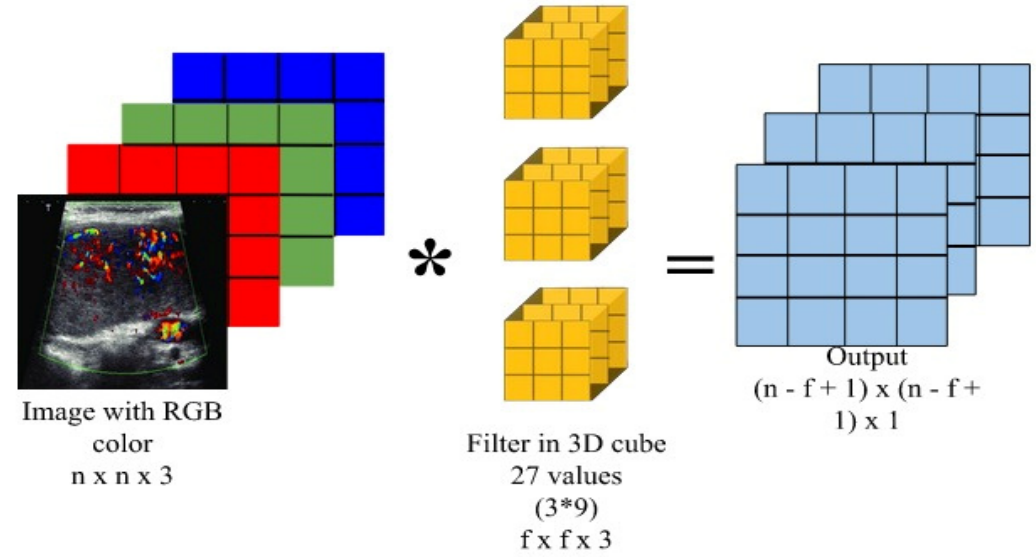

Fig: 19. Multiple filter convolution with multiple output

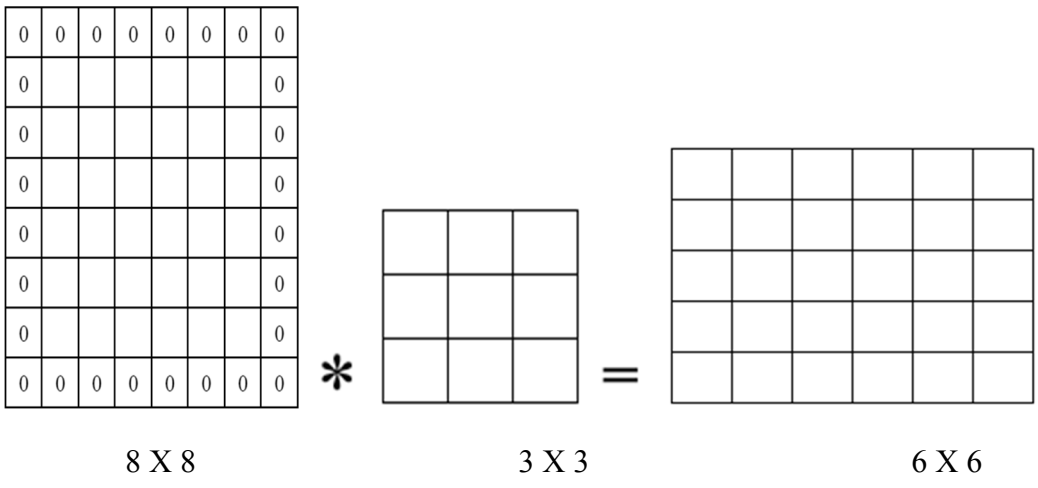

Fig 20 : Padding in Convolutional Neural Network

There are two main choices when it comes to padding VALID and SAME. When VALID performs the convolution operation with no padding at all,. After performing the convolution operation, the output image should be the same size as the input image. The size of new image will be

n' = n +2p, where p is amount of padding that we are applying

(n' - f + 1) = n that is (n + 2p - f + 1) = n the value of p is given below

$$p = (\frac{f - 1}{2})$$

If we use the Tensor Flow framework and apply a convolution layer, we just specify what kind of padding we want.

tf.nn.conv2D(X, W1, strides = [1, 1, 1, 1], padding = 'VALID'). Here 'f' is usually taken as an odd number. If we take an even number, then we will need to apply uneven or asymmetric padding.

For example, we might need 1 pixel of padding on one side of the image, and on the other side, we might need 2 pixels.

Max Pooling is the most powerful operation on CNN. There can be many types of pooling layers. In CNN, the function pooling layer is to reduce the size of the dimension of the image by preserving the feature on it. For the max pooling operation, we need to fix a filter for a particular size and stride a particular amount. This filter can be of any size or amount. Usually, any amount of stride is taken from the same size as the side length of the filter. For example, the filter size is 2 × 2 and the stride is 2. After the performance, the output will be an image with a 2 × 2 matrix. Performing the max pooling operation, we will see the filter image in the top left column of the input image, then we will extract the maximum value of the window where the filter image is as per stride size for the next maximum value. This will be repeated till the last pixel. Why might we want to do this max pool?

The first reason is that it reduces the image size and, thus, the computational cost. So that we can get the output faster and train the model faster. We want to do our operations quickly. Even though it reduces the size of the image, it still preserves the features of that image, and max pooling works in such a way that it sharpens the

features or enhances the features. Suppose the input image has a vertical edge; this vertical edge is still preserved but actually enhanced in the output image. As shown in one window of an image, we can say that the maximum value holds the maximum intensity of the features. When they extract the maximum value, it sharpens the features. Now where do we need max pooling in CNN?

The standard procedure in convolutional neural networks (CNNs) involves applying a max pooling layer after the convolutional layer. This step effectively reduces the size of the output image from the convolutional layer while also enhancing the extracted features. The convolution layer itself executes the convolution process using multiple filters. Suppose if we use 5 numbers of filters in one convolution layer then we know that it will generate 5 numbers of output images. Thus after performing the max pooling operation we will again get the same number of output images. Which means 5 numbers of images in the output of the max pooling layer. From the entire CNN we will use many convolution layers and max pooling layers. Here there is a small side note that it does not always use max pooling layer after every convolution layer. We may use convolution layers and may skip max pooling layers or just fewer numbers of max pooling layers than the number of convolution layers. The reason for that is the max pooling layer reduces the size of the image but we might not reduce the size too much, if we are using a very big convolutional neural network. The use of max pooling will improve the performance of CNN. There are no parameters in it or no training in it. Another pooling method is average pooling, which takes the average of values filter size and image size pixel.

To perform the convolutional layer we use the convolutional operation of the input radiology image to get the value of the input skeletal image which leads to it being scaled properly. So, we add a ‘bias’ which denotes ‘b’ and assign it to the nonlinear function. This non linear function can be tanh or ReLU but most commonly ReLU is used after that we will get the output as size as compared with input image. So the entire convolution is one applying the convolution operation and one applying ReLU operation or nonlinear activation function. These combain we called convolution layers.

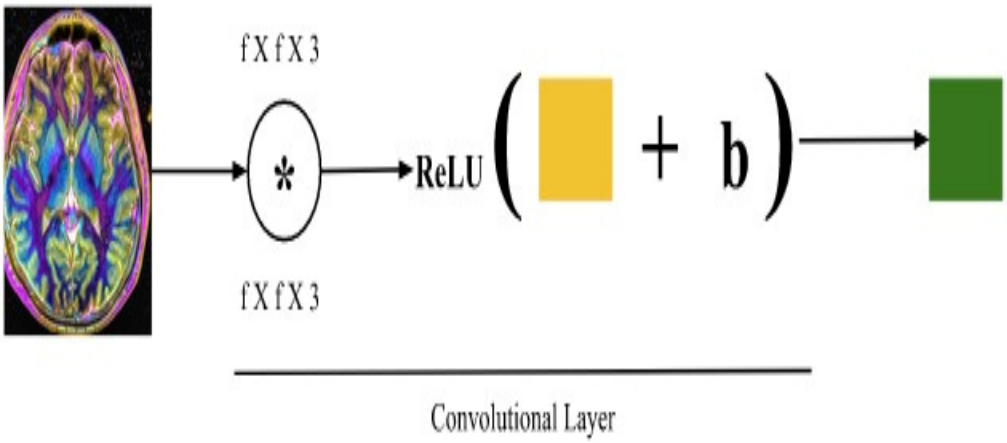

Fig: 21. Single Filter Convolution layer using non linear function and bias with single image as output

If we use multiple layers, the bias will also be changed, and we will get multiple outputs, as shown in the below figure.

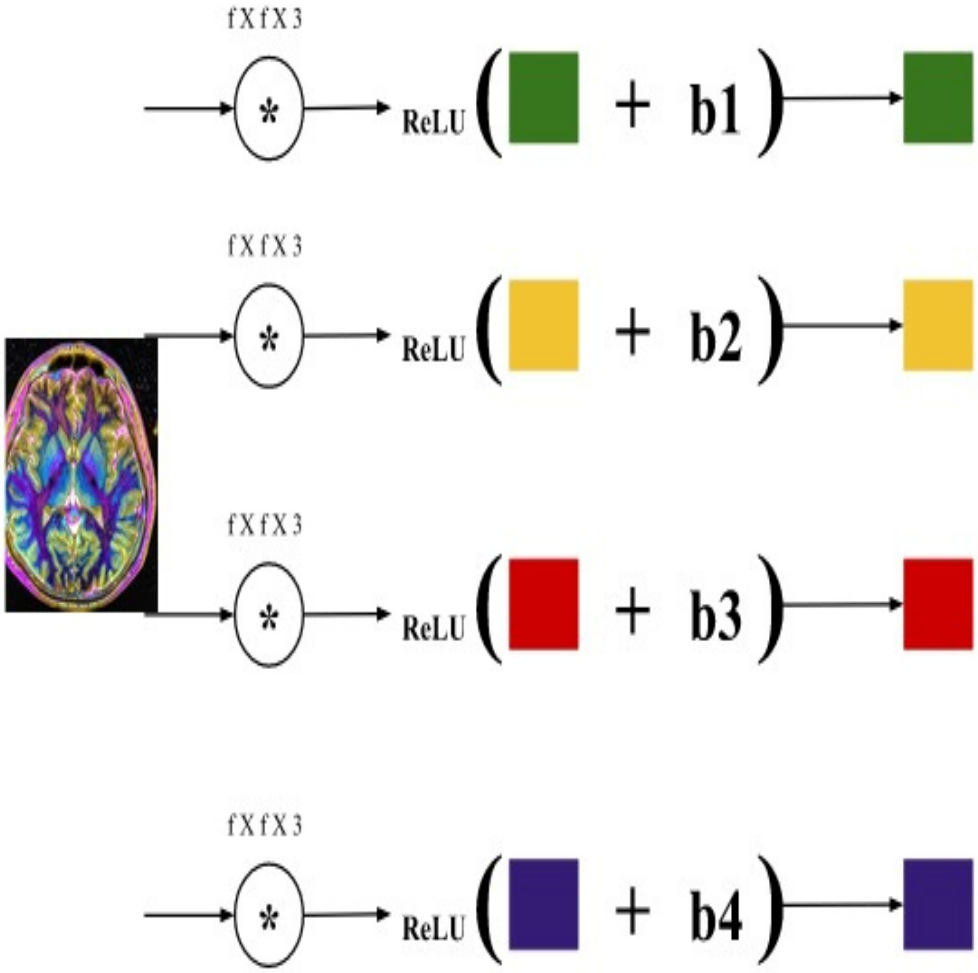

Fig. 22. Multiple Filter Convolution layer using non linear function and bias with multiple output

So, in short, we convert it as,

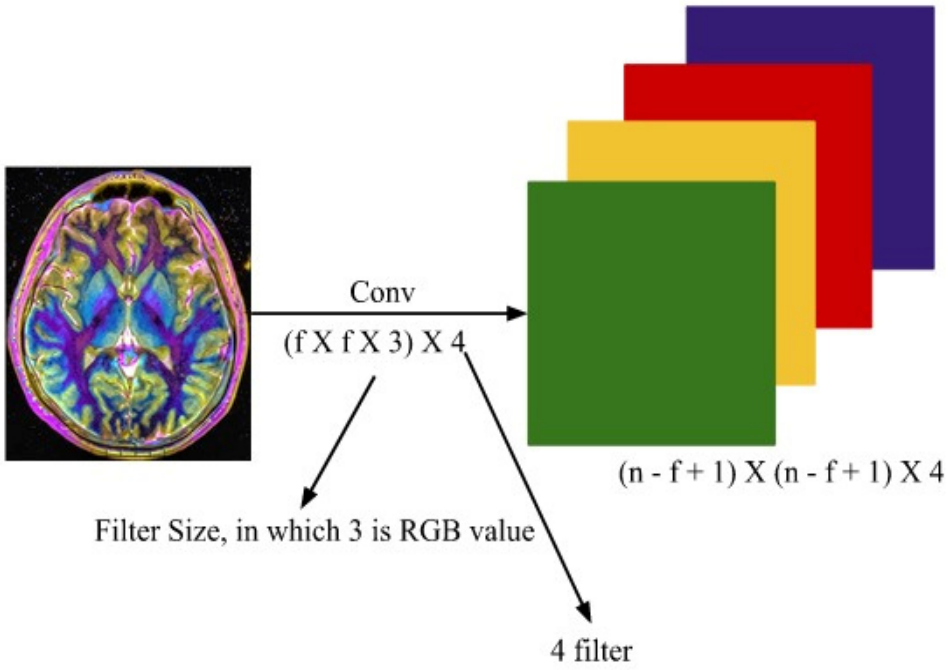

Fig: 23. Overall Convolution layer using non linear function and bias with multiple output in one frame

Let we see one image of MRI Brain as input which has the size 32 X 32 pixel with RGB value 3. So the image size is 32 x 32 x 3. First we will convolve the input image with the particular filter size 5 x 5 x 3 and use strider 1 with 4 filters so the output size will be 28 x 28 x 4. Here the 28 is n - f + 1 = 28. As we use 4 numbers of filters we will get 4 dimensional output. After convolving this image we pass this to a max pooling layer. So here we are using max polling 1 with the filter size 2 x 2 and slider 2. Thus the dimension image will become half, which means 28 x 28 will become 14 x 14 x 4 and the number of channels will remain the same. In the traditional CNN both convolutional layer and max pooling layer will be considered in one layer. So, now this output 14 x 14x 4 will pass another convolutional layer which we name Conv2. Here we will use another filter size 3 x 3 but the 4 will remain the same. In the Conv 2 layer we used a filter so the output has 8 channels, again it passes through the max pooling layer and we will get the output half of the input image size. In this we consider layer 2. So, we can continue these layers as many times as we want. It depends on the type of application. If we are performing a completed application like healthcare then we might need our complex or large convolutional architecture. Also

the another thing is that we can completely skip this max pooling operation if we are building big CNN architecture because maxpooling will reduced the size of images that we are dealing with, and we might not want to reduce the size of images that we are dealing with, and we might not want to reduce size too much.

So once we are done with all convolutional layers and max pooling layers then it's time to add fully connected layers, before we apply these connected layers we need to flatten the final output image. So, the 6 x 6 x 8 will be flatten only into the 1D factor which is 288 units in one flatten. Now once flattened it is now connected with a fully connected layer. This will represent FC3(Fully Connected layer), which indicates it as 3 layers in our CNN architecture. Let we keep 120 neurons in FC3 so it will connect with all nodes of flatten output. So the number of weight parameters is 288 x 120. Now we add multiple fully connected layers as much as we want but, it is not fair adding fully connected layers highly increases the amount of parameters we need to deal with. Then in the final layer we will apply the sigmoid or softmax activation function depending on the types of applications we are making. For example if we are using binary classification then we will be using sigmoid or we will be using softmax activation. So, the final layer is called sigmoid or softmax or FC4 layer. In the softmax layer, there are huge numbers of neurons that must correspond to the number of output categories. This final layer is responsible for generating the predicted value. Prior to making predictions, it is imperative to train the system to ensure the factors are appropriately adjusted. Subsequently, the final predictor is utilized to compute the cost function, which quantifies the extent of inaccuracies in the model's predictions.

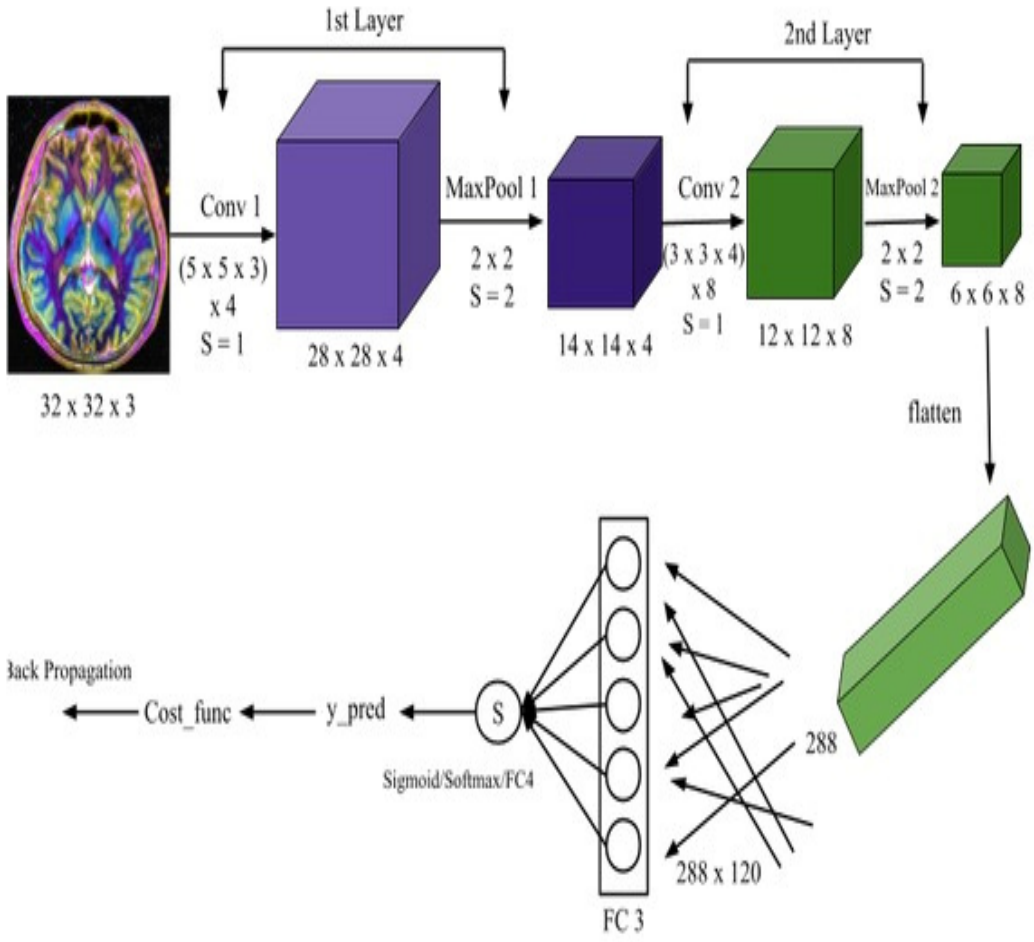

Fig: 24. The Architecture of Convolutional Neural Network

Now the question is which cost function is used? The answer is if we are using binary classification then it will be 'Bn' binary entropy cost function. If we are performing multiclass classification then we can use categorical class entropy classification. The goal is to lower the cost function so that we can train and improve our model's accuracy. So, how do we conduct the training? We employed the same old back propagation model as in our Artificial Neural Network or Neural Network.

In Artificial Neural Networks we use dense networks. There is lots of redundancy. The parameters are very high but we can reduce the amount of factors by using convolutional and max pooling. So we can reduce the size by convolutional layer and max pooling layer. The implementation of the back propagation algorithm in CNN is actually very complicated because here we also need to calculate the δ cost/ δ w. The

challenge is that the 'w' parameters are actually inside the filters used for the convolution operation.

Thus calculating this $\frac{\delta cost}{\delta w}$ and applying the greadent descent is very complicated in CNN. To make this job easy we use the Tensorflow framework in python. It makes our implementation very quick and easy. We do not need to deal with the details in the integrity of the complex technique but we can just mention the name. Example just mention what cost function we need or the name of back propagation we want to use Adam, or Momentum or greadent descent. So, tensorflow automatically implements for us. The advantage is to make CNN and find that the accuracy is not pretty good so you may want to add or remove some layer or modify the network. Whenever we modify the network layer we also need to do the same modification in back propagation as well. There are chances to lead to errors while doing back propagation. Thus all these will be very complicated and very time consuming and we won't be able to focus on the architecture properly but in tensorflow we just mention the type of layer like conv layer and parameters like stride, padding etc. If we want to use max pooling then mention it in there and so on.

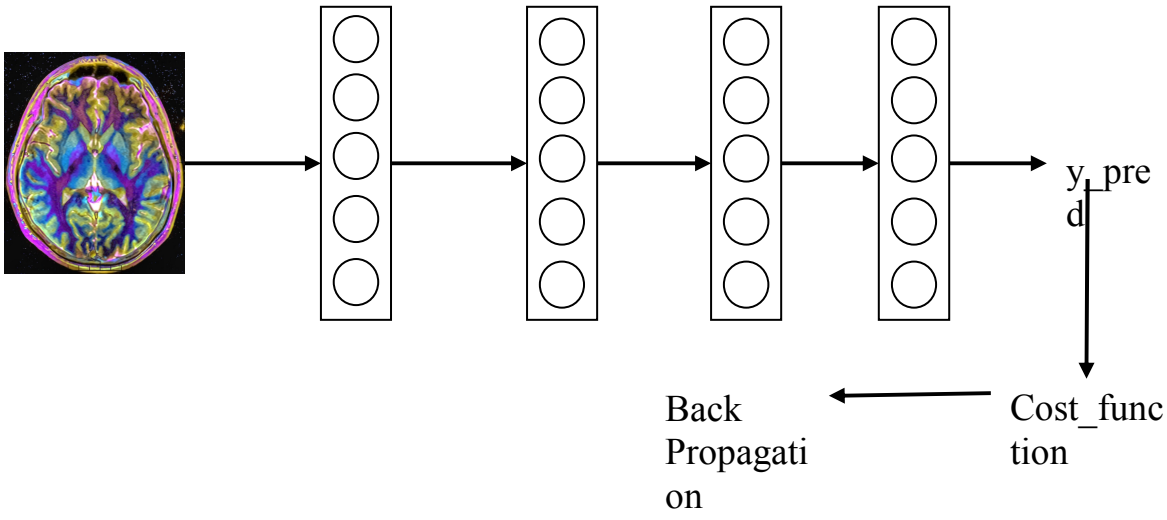

Fig: 25 : The Traditional Architecture of Artificial Neural Network

3.3.4 Deep Belief Network model, the feature extraction of skeletal image

A Deep Belief Network (DBN) is a form of artificial neural network made up of several layers of stochastic, latent variables known as hidden units. DBNs function as

generative models and are Well-suited for unsupervised learning tasks such as feature learning, dimensionality reduction, and data generation.

A DBN typically consists of several layers of neurons systematized in a hierarchical pattern. The network's initial layer is the visible layer, and it interacts directly with input data. The following layers are made up of hidden layers, with each layer having a unique number of neurons. DBNs frequently feature a symmetric architecture, with connections between each layer and its neighboring layers but not inside the same layer. Restricted Boltzmann Machines (RBMs) are probabilistic, generative neural networks that serve as the foundation of Deep Belief Networks. RBMs comprise two layers: visible and hidden, connected by symmetric connections. The architecture follows a bipartite graph structure, where neurons in the visible layer connect to neurons in the hidden layer, with no connections within each layer. RBMs adopt a probabilistic approach, employing binary units to represent the presence or absence of features. During training, RBMs learn to reconstruct input data by altering connection weights with approaches like Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD). DBNs are often trained using a greedy layer-wise strategy, which involves training each layer of the network independently before fine-tuning the entire network. The first layer (visible layer) is learned directly on the input data using unsupervised learning techniques like RBMs. After the first layer is trained, its output is utilized as input to train the next layer, and so on until all layers are trained. After training each layer independently, the complete network is fine-tuned using supervised learning techniques like backpropagation. Fine-tuning changes the weights of connections across the network to improve performance on a specific job, such as classification or regression. DBNs have been effectively used for a variety of applications, including image identification, audio recognition, natural language processing, and collaborative filtering. They are especially good at learning hierarchical representations of complex data and extracting significant features from high-dimensional input spaces. In general, a Deep Belief Network is a hierarchical, generative neural network design made up of numerous layers of RB Machine. Using a greedy layer-wise training technique, Deep BNs may learn hierarchical data representations and extract valuable features for a variety of machine learning tasks.

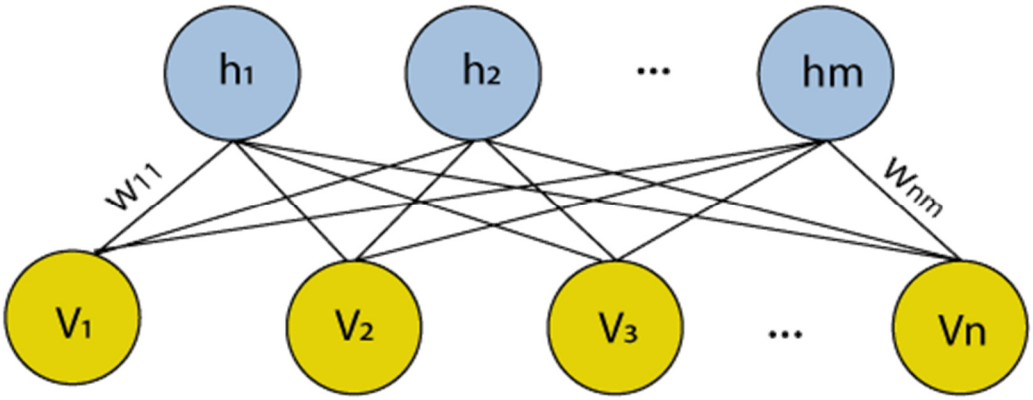

Fig 26 : Two-layer probabilistic neural network, RBM Architecture

Deep Belief Networks (DBNs) can be used in visual question answering (VQA) systems to extract meaningful features from both the visual and textual modalities of input data. In a VQA system, the input consists of both images and textual questions. The initial phase involves extracting features from both modalities. Regarding the visual modality, the DBN can be used to extract hierarchical representations of image features. Each DBN layer learns more abstract and complicated features from the image's raw pixel values. Similarly, for the textual modality, the DBN can process the text input (questions) to learn hierarchical representations of word embeddings or semantic features. Once features are extracted from both modalities, they must be joined or fused to form a single representation that captures the relationship between the image and the question. The extracted features from the visual and textual modalities can be concatenated, merged, or processed using additional layers of neural networks to create a joint representation. The composite representation should include relevant inputs derived from both the image and the question, allowing the VQA system to generate precise predictions. The DBN-based VQA system is trained using a massive dataset of matched images and questions, as well as their responses. During training, the DBN layer parameters, as well as any additional layers used for fusion and prediction, are optimized using backpropagation and stochastic gradient descent algorithms. The goal is to reduce the prediction error between the forecasted and ground-truth responses in the training data. Once trained, the DBN-based VQA system can be used to answer questions about unseen images. Given a new image and a question, the system first extracts features from both modalities using the trained

DBN. The retrieved characteristics are then integrated to form a joint representation, which is used by the prediction layer to generate the answer. The projected answer is often the result of a softmax layer, which represents the probability distribution of probable replies.

The performance of the DBN-based VQA system is examined using performance metrics such as accuracy, precision, recall, and F1 score. The system's ability to correctly answer questions about unseen images is assessed using a separate test dataset, and its performance is compared to other VQA systems using benchmark datasets. DBNs can be integrated into VQA systems to extract the features from both skeletal image and question-answer modalities, which are then fused to generate a joint representation for answering questions about images. By leveraging hierarchical feature learning, DBNs can capture complex relationships between visual and text, leading to improved performance in visual question-answering tasks.

A neural network of beliefs is a deep learning model made up of numerous layers of probabilistic latent variables that are often placed in a stack of restricted Boltzmann machines (RBMs). A DBM design consists of alternating levels of visible and hidden units, with each layer fully connected to the next. An RBM is a generative probabilistic neural network that represents the joint probability distribution of visible and hidden units. On an RBM, each visible unit is linked to every hidden unit, Nevertheless, there are no inter-unit connections within the same layer.. The connections between visible and hidden units are weighted, and each connection has its own weight parameter. RB Machines (RBMs) comprise a visible layer that represents input data, such as pixel values in an image, and one or more hidden layers that capture latent features or representations of the input. Deep Belief Networks (DBNs) are constructed by stacking multiple RBMs, creating a deep architecture. Each RBM's visible layer is connected to the hidden layer of the subsequent RBM, forming a feedforward chain. This enables the network to acquire hierarchical representations of the input data. Typically, DBNs are trained using layer-wise pretraining followed by fine-tuning via backpropagation. The pretraining step in a DBN initializes the weights of the RBMs layer by layer, allowing each layer to acquire informative data representations before fine-tuning the entire network. After layer-wise pretraining, the complete DBN undergoes fine-tuning using supervised

learning techniques such as backpropagation. This process involves training the DBN on a labeled dataset using gradient-based optimization algorithms to minimize a predetermined loss function, such as cross-entropy loss. Fine-tuning adjusts the weights of the entire network, enhancing its capability for classification or data generation. Each unit, whether visible or hidden, typically applies an activation function to its input to produce an output. The logistic sigmoid function, hyperbolic tangent function, and rectified linear unit (ReLU) function are all examples of common activation functions used in deep neural networks. The output layer of a DBN is determined by the individual task being handled. For example, in a classification assignment, the output layer could be made up of softmax units that represent different classes.

Overall, the architecture of a DB network consists of stacked RBMs with alternating layers of visible and hidden units. By learning hierarchical representations of leveraging the input data, DBNs have the capacity to apprehend complex patterns and dependencies, thereby proving effective in tasks such as feature acquisition, categorization, and generation.

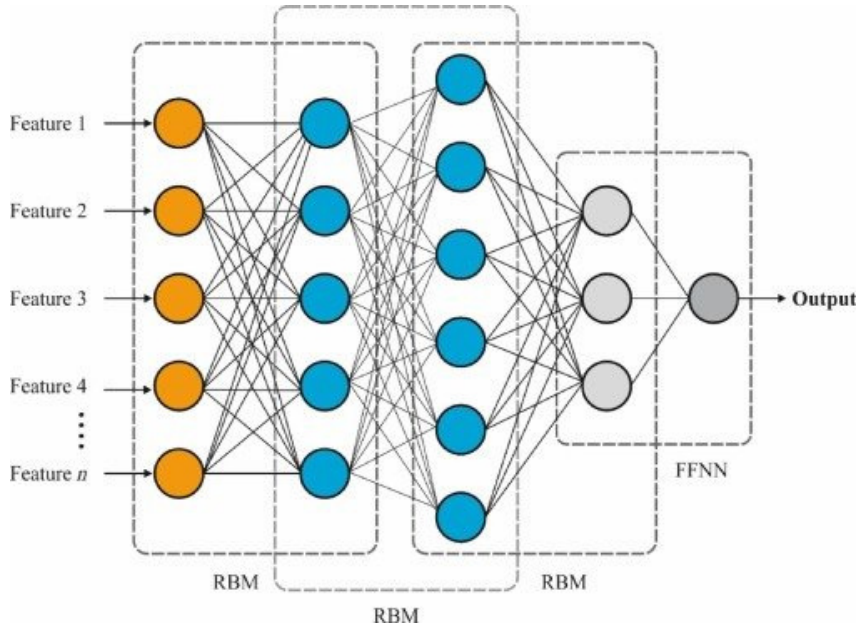

Fig: 27 : Overall architecture of a Deep Belief Network

A restricted Boltzmann machine is trained in a fundamentally different way from neural networks, which use stochastic gradient descent. There are two main steps to

train our dataset: the first one is Gibbs sampling, and the other one is the Contrastive Divergence Step.

Gibbs collection is the initial stage of training. When we are provided an input vector, we use the following $p(h|v))$ to forecast the hidden values h. However, if we know the hidden values h, we may utilize $p(v|h)$ to forecast the new input values.

$$p(v_i = 1 \mid h) = \frac{1}{1+e^{(-(a_i+w_i h_j))}} = o'(a_i + \Sigma h_j w_{ij}) \quad (1)$$

This action is performed k times, generating a new input vector v_k that is derived from the initial input value v_0.

$$p(v_i = 1 \mid v) = \frac{1}{1+e^{(-(b_i+w_i v_j))}} = \sigma(b_j + \sum_i \quad v_j w_{ij}) \quad (2)$$

The contrastive divergence stage modifies the weight matrix. The vectors v_0 and v_k are utilized to compute the activation probability of the hidden variables h_0 and h_k.

$$p(v_i = 1 \mid h) = \frac{1}{1+e^{(-(a_i+w_i h_j))}} = o'(a_i + \Sigma h_j w_{ij}) \quad (3)$$

The update matrix is calculated as the difference between the outer products of the probabilities with input vectors v_0 and v_k, as shown in the matrix below.

$$\Delta W = v_o \otimes P(h_o|v_o) - v_k \otimes P(h_k|v_k) - v_k \quad (4)$$

We can now utilize this updated weight matrix to calculate new weight using gradient descent, as stated in the equation below.

$$W_{new} = W_{old} + \Delta W \quad (5)$$

To apply the architecture of a DB Network (DBN) to the task of visual question answering (VQA) using medical images, we need to adapt the network to handle both visual and textual input. The input layer of the DBN will consist of two components:

Visual input: This component represents the medical images. Each image is represented as a vector of pixel values or a higher-level feature representation extracted using techniques such as convolutional neural networks (CNNs).

Textual input: This component represents the textual questions associated with each image. Each question is represented as a vector using techniques such as word embeddings or one-hot encoding.

The DBN architecture will consist of alternating layers of visible and hidden units, similar to the traditional DBN architecture. However, each visible layer will now have two components: one for visual input and one for textual input. Each RBM in the stack will learn joint representations of the visual and textual inputs, capturing the relationships between the image content and the corresponding questions. During the layer-wise pretraining phase, each RBM in the stack will be trained independently using both visual and textual input. The pretraining phase initializes the weights of the RBMs, allowing them to learn hierarchical representations of both visual and textual features. After layer-wise pretraining, the entire DBN will be fine-tuned using supervised learning techniques.The network will undergo training using a dataset comprising labeled pairs of images and corresponding questions, aiming to predict accurate answers for each question based on the associated image. The output layer of the DBN will consist of units representing potential responses to inquiries, potentially employing a softmax activation function to produce probability distributions across potential answers. Integration of visual and linguistic components throughout the network will facilitate capturing the relationships between image content and textual queries. Through the hidden layers, the DBN will acquire composite representations of visual and linguistic attributes, enabling comprehension of the correlations between questions and images. By adapting the architecture of a DBN in this way, we can leverage its ability to learn hierarchical representations of complex data to effectively tackle the task of visual question answering in the medical domain. The network will be capable of understanding both the visual content of medical images and the textual queries associated with them, enabling it to provide accurate answers to a wide range of medical questions.

3.3.5 The deep insights of Region-based Convolutional Neural Network (R-CNN) for visual classification

The Region-based CNN (R-CNN) represents a seminal advancement in object recognition methodology, seamlessly integrating deep learning with conventional computer vision methodologies. Introduced by Ross Girshick et al. in 2014, it has

emerged as a foundational model in the realm of object detection. R-CNN, along with its subsequent iterations, has laid the groundwork for contemporary object detection architectures such as Faster R-CNN and Mask R-CNN. Unlike traditional object detection methods which relied on handcrafted features and disparate algorithms for localization and classification, R-CNN endeavors to unify these tasks within a unified deep learning framework, facilitating end-to-end training and enhanced performance. R-CNN begins by generating region recommendations via a selective search method. This algorithm looks for probable object regions in an image using color, texture, and other low-level information. Each region suggestion is then sent into a pre-trained CNN, such as AlexNet or VGG-16, which produces fixed-length feature vectors. The collected characteristics are then utilized to train a linear SVM classifier for object categorization. Each SVM is taught to distinguish between various object categories (for example, person, automobile, and dog). R-CNN also adds a bounding box regression step to refine the location of detected objects. This phase teaches how to change the bounding box coordinates predicted by the selective search method. R-CNN employs selective search to generate boundary recommendations from an user's input medical image. Each region suggestion is warped to a specific size and sent into a pre-trained CNN to extract features. The features are then sent via distinct SVM classifiers for object classification. Finally, the bounding box regression step adjusts the predicted bounding boxes to improve localization accuracy.

- R-CNN is trained in multiple stages:
    - Pre-training: The CNN is trained in advance on an extensive dataset. (e.g., ImageNet) for feature extraction.
    - Fine-tuning: The CNN is adapted on the target dataset for object detection.
    - SVM training: Separate SVM classifiers are trained for each object category using the extracted features.
    - Bounding box regression: Another model is trained to learn the adjustment needed for bounding box coordinates.

R-CNN has several limitations, including its slow inference speed due to the sequential processing of region proposals and the need to extract features for each proposal independently. This inefficiency makes it impractical for real-time

applications. Another limitation is the difficulty of end-to-end training, as the SVM classifiers and bounding box regressors are trained independently of the CNN. Several variants of R-CNN have been offered to address its limitations, including Fast R-CNN, Faster R-CNN, and Mask R-CNN. These variations strive to increase object detection speed and accuracy by using innovations like region proposal networks (RPNs) and shared feature extraction.

R-CNN takes a skeletal image as input and detects and localizes items in it. R-CNN starts by creating region proposals, which are candidate bounding boxes that may contain objects. These recommendations are often created with selective search, a standard computer vision technique that finds regions of interest based on color, texture, and other low-level properties. Selective search generates a huge number of area recommendations with different scales and aspect ratios. Each region proposal is cut from the input image and scaled to a predefined size, which typically corresponds to the input size predicted by a pre-trained convolutional neural network (CNN). The pre-trained CNN functions as a feature extractor and is often loaded with weights learned from a large-scale image classification challenge like ImageNet. The region proposals are fed through the CNN to produce fixed-length feature vectors. These feature vectors carry detailed semantic information about the content of each region proposal. The extracted feature vectors are subsequently fed into a sequence of fully connected layers, followed by a softmax layer for object categorization. Each completely linked layer serves as a classifier for a certain item category (such as a human, automobile, or dog). The softmax layer generates a probability distribution over the object categories, showing the likelihood that each region suggestion belongs to a specific category. In addition to object classification, R-CNN incorporates a bounding box regression phase to improve the localization of discovered objects. This phase teaches how to alter the coordinates of the bounding boxes predicted by the region proposal algorithm, improving their accuracy. Bounding box regression is often accomplished using a separate set of fully connected layers that forecast the offsets required to change the bounding box positions. The ultimate output of R-CNN comprises identified objects, their corresponding bounding boxes, and class labels. Each detected object is associated with a bounding box that precisely delineates its location within the input image, alongside a class label denoting its category.

- Training Stages of R-CNN in various steps:
    - Pre-training: The Convolutional NN is pre-trained on a large dataset (e.g., ImageNet) for feature extraction.
    - Fine-tuning: The Convolutional NN is optimized on the target dataset for object detection.
    - Object classification and bounding box regression: The fully connected layers in charge of object categorization and bounding box regression are trained by backpropagation and gradient descent.

R-CNN has inspired several variants, including Fast R-CNN, Faster R-CNN, and Mask R-CNN, which enhance upon its speed and accuracy by introducing innovations such as region proposal networks (RPNs) and shared feature extraction.

Region-based Convolutional Neural Network (R-CNN) can be used in a Visual Question Answering (VQA) system for image feature extraction by first selecting regions of interest within the image, and then extract features from these regions to answer image-related questions.

R-CNN starts by generating region proposals within the input image. These region proposals are possible bounding boxes that may contain objects of interest. These region suggestions can be generated using selective search or another region proposal algorithm that takes into account low-level image properties like color, texture, and shape. Once the region proposals have been developed, R-CNN crops and warps each one from the original image to a defined size. The cropped regions are then fed into a pre-trained CNN, such as VGG, ResNet, or Inception, to extract features. The CNN functions as a feature extractor and is often trained on a large-scale picture dataset such as ImageNet. The output of the CNN is a fixed-length feature vector that encodes rich semantic information about the content within each region proposal. In parallel to extracting image features, the input question is processed using natural language processing (NLP) techniques to standardize it into a fixed-length vector representation, known as a question embedding. Techniques like word embedding (e.g., Word2Vec, GloVe) and recurrent neural networks (RNNs) or transformer models (e.g., BERT) can be used to encode the question into a numerical vector. The feature vector obtained from the image regions and the question embedding are then fused or concatenated to create a joint representation that combines both visual and

textual information. Various fusion methods can be employed, such as element-wise addition, concatenation, or attention mechanisms, to successfully integrate image characteristics and question embeddings effectively. The joint representation is then passed through a classifier, such as a multi-layer perceptron (MLP) or a softmax layer, to predict the answer to the input question. The classifier learns to map the joint representation to a probability distribution over a predefined set of answer options. The entire VQA system, including the R-CNN for image feature extraction, is trained end-to-end using a large dataset of image-question-answer triplets. During training, the parameters of the CNN, question embedding model, fusion method, and classifier are optimized using backpropagation and gradient descent to minimize the prediction error. During inference, given a new image and question, the trained VQA system predicts the most likely answer by passing the image through the R-CNN for feature extraction, processing the question to obtain its embedding, fusing the image features and question embedding, and finally predicting the answer using the trained classifier.

By leveraging R-CNN for image feature extraction in a VQA system, the model can effectively extract visual features from specific regions of interest within the radiology image, enabling it to accurately answer questions about the content of the image.

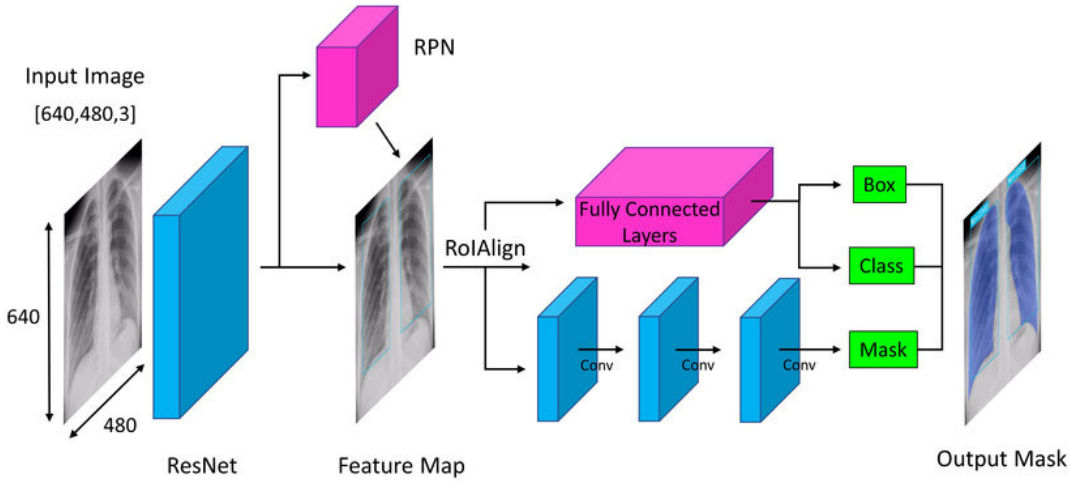

Fig: 28: Faster Region-based Convolutional Neural Network (R-CNN)

Faster R-CNN enhances the original R-CNN by integrating a Region Proposal Network (RPN) directly into the architecture. This integration facilitates end-to-end training, leading to improved efficiency and accuracy in object detection tasks. The

RPN operates as a fully convolutional network, leveraging feature maps extracted from input images to generate region proposals, which represent potential bounding boxes containing objects. By sliding a tiny network across the feature map, the RPN anticipates the presence of objects ("foreground") within each anchor box and refines their positions. The Region Proposal Network (RPN) in Faster R-CNN generates region proposals by integrating expected bounding box coordinates with anchor boxes of different scales and aspect ratios. This process aims to efficiently cover potential object locations across the image. Additionally, Faster R-CNN utilizes a deep convolutional neural network (CNN) as a feature extraction backbone to extract high-level features from the input image. Commonly used backbone networks include VGG, ResNet, and MobileNet, which are pretrained on extensive image classification datasets like ImageNet. These backbone networks process the input image to produce a feature map containing detailed spatial information. After generating region proposals, Faster R-CNN uses RoI pooling to extract fixed-size feature vectors from each proposal. RoI pooling partitions each proposed region into a predetermined number of spatial bins, subsequently performing max pooling within each bin to yield a feature vector of fixed dimensions.

This ensures that the features extracted from different-sized region proposals have the same spatial dimensions, which is essential for feeding them into subsequent layers of the network. RoI pooling generates fixed-size feature vectors, which are used in fully connected layers for object classification and bounding box regression. The classification branch assigns a class label to each proposal based on its likelihood to contain an item. The regression branch refines the coordinates of the bounding box to better localize the object within the proposal. Both the classification and regression branches are typically implemented as fully connected layers on top of the shared feature representation. After generating region proposals, Faster R-CNN employs Region of Interest (RoI) pooling to derive fixed-size feature vectors from individual proposals. This process entails segmenting each area proposal into a predefined number of spatial bins and subsequently applying max pooling within each bin to produce a fixed-size feature vector. This process ensures that features extracted from region proposals of different sizes have consistent spatial dimensions, which is necessary for subsequent layers in the network.The resulting fixed-size feature vectors

are then input into fully connected layers for object classification and bounding box regression. The classification branch determines the likelihood that each proposal contains an object and assigns a class label.

Faster R-CNN can help with feature extraction for visual or skeletal image question answering (VQA) in radiology images by efficiently detecting and localizing relevant objects or regions within the images. Faster R-CNN can accurately localize and identify regions of interest (ROIs) within radiology images that contain anatomical structures, abnormalities, or other medically significant features. These ROIs can serve as the basis for answering questions related to specific areas or abnormalities within the image. Once the relevant regions are identified, Faster R-CNN can extract high-level features from these regions using RoI pooling. This process involves pooling characteristics from the convolutional layers of the network within the identified regions, allowing for the extraction of rich and discriminative feature representations. By detecting and localizing objects within radiology images, Faster R-CNN helps in understanding the semantic content of the images. This semantic understanding can be crucial for answering questions that require interpretation of the visual content, such as identifying anatomical structures, pathologies, or medical devices.

Faster R-CNN offers cutting-edge performance in object detection tasks, providing accurate localization of objects with high precision and recall. This improved accuracy ensures that the extracted features are relevant to the task at hand, thereby enhancing the performance of the VQA system. Faster R-CNN can be seamlessly integrated into the VQA pipeline, allowing for end-to-end training of the system. This integration enables joint optimization of the object detection and question answering components, leading to better alignment between the visual and textual modalities and improved overall performance.

Overall, Faster R-CNN serves as a powerful tool for feature extraction in VQA systems for radiology images, facilitating the identification of relevant visual content and enhancing the system's ability to provide accurate and informative answers to medical-related questions.

Multiple CNN-based object detection algorithms have been developed to address various challenges in computer vision tasks. These algorithms include R-CNN [87], SPPnet[88], Fast R-CNN [89], Faster R-CNN [90], You Only Look Once (YOLO) [91], and Single Shot Multibox Detector (SSD) [92]. Among these, Faster R-CNN stands out for its cutting-edge performance in object detection tasks.

Faster R-CNN is renowned for its accuracy and ability to precisely localize objects within images. While it may have a slightly slower processing speed compared to newer approaches like YOLO and SSD, its performance and versatility make it an ideal choice for many applications, including visual or skeletal image question answering in the healthcare domain.

One of the principal advantages of Faster R-CNN lies in its capacity to efficiently execute object identification tasks. It achieves this by employing a region proposal network (RPN) to create region recommendations directly from convolutional feature maps. These region proposals are then used to localize objects within the image, allowing for precise object detection.

Another important characteristic of Faster R-CNN is its flexibility in handling images of varying sizes. Unlike some other object detection algorithms that require fixed input image sizes, Faster R-CNN can efficiently process images of different sizes using a sliding window approach. This makes it particularly useful for dealing with large medical images commonly encountered in radiology.

Furthermore, Faster R-CNN does not compromise on accuracy despite its efficiency. It leverages deep convolutional neural networks (CNNs) to extract rich feature representations from images, enabling robust object detection and classification. Additionally, it incorporates advanced techniques such as region of interest (RoI) pooling to further enhance its performance.

Faster R-CNN consists of two main modules: the region proposal network (RPN) and the object detection network. The RPN is responsible for generating region proposals, while the object detection network classifies the objects within these proposed regions.

The workflow of Faster R-CNN begins with the radiology input image being processed by a feature extractor, typically a convolutional neural network (CNN) with

convolutional and pooling layers. This feature extractor generates feature maps, which contain high-level representations of the image that capture important visual information.

The feature maps extracted from the input image are subsequently fed into the region proposal network (RPN) which examines them to determine whether regions are likely to contain objects. The RPN achieves this by traversing a small window (typically 3x3) across the feature maps, predicting the presence of an object within each window, and generating bounding box proposals for these regions. This process allows the network to propose candidate regions of interest for further analysis and classification.

Once the region proposals are generated by the RPN, they are passed to the object detection network, which further processes each region to classify the objects within them and refine their bounding box coordinates if necessary. This network typically consists of additional convolutional and fully connected layers, followed by classification and regression heads.

In the original Faster R-CNN architecture, the feature extractor often relies on established CNN architectures such as VGG-16. However, alternative models may offer better performance. For instance, Inception-ResNet, which merges the Inception and ResNet architectures, surpasses VGG-16 in certain applications.

The region proposal network (RPN) of Faster R-CNN examines feature maps using a sliding window approach. However, utilizing a fixed-size window could be limiting, especially when dealing with objects of diverse shapes and sizes. Faster R-CNN mitigates this concern by introducing the notion of anchor boxes.

Anchor boxes are predetermined bounding boxes with diverse scales and aspect ratios that are distributed across the image. These anchor boxes provide reference points for spotting items of various shapes and sizes.By using multiple anchor boxes, the RPN can better capture the variability in object shapes within the image.

During the sliding window operation, each anchor box generates two scores: one indicating whether the window contains an object (foreground) or not (background), and the other indicating the class of the object if it is present. Specifically, for each anchor box, the RPN produces $4 \times k$ pieces of information corresponding to the

bounding box coordinates (i.e., the coordinates of the box's top-left corner and bottom-right corner) and 2 × k pieces of information corresponding to the objectness score and the class score.

Using anchor boxes with multiple scales and aspect ratios, the RPN can successfully handle objects of various forms and sizes, improving object detection accuracy in Faster R-CNN. This approach allows Faster R-CNN to perform consistently over a wide range of object types and combinations in input images. R-CNN's region proposal network (RPN) produces 4k × W × H pieces of area information and 2k × W × H pieces of class information, where k is the number of anchor boxes used and W × H is the size of the feature map.

Initially, Faster R-CNN typically utilizes 9 anchor boxes (k = 9), each with a combination of scales and aspect ratios. However, to improve the detection of specific types of objects, such as glomeruli in medical images, the number of anchor boxes can be increased. In this case, 12 anchor boxes (k = 12) were employed, including variations in scale and aspect ratio to better capture the diversity of object shapes and sizes.

Since the RPN may propose multiple candidate regions for the same object, it introduces redundancy in the detection process. To address this issue, non-maximum suppression (NMS) is applied based on the class scores associated with the proposed regions. NMS is a technique used to remove redundant bounding boxes by selecting the most confident detection and discarding overlapping detections with lower confidence scores. By applying NMS, the number of proposed regions of interest (ROIs) is typically reduced to a more manageable number, typically around 300, ensuring that only the most relevant and confident detections are retained for further processing.

In Faster R-CNN, the region of interest (ROI) pooling technique is employed to convert ROIs of varying sizes into fixed-sized feature vectors. ROI pooling ensures that the input to subsequent layers remains consistent in size, regardless of the size or shape of the original ROIs. This process is crucial for maintaining spatial information while enabling the network to handle inputs of different dimensions.

Following ROI pooling [87], the fixed-sized feature vectors are sent through a fully linked layer. This layer is responsible for two major tasks: bounding box regression and object classification.

Bounding box regression predicts the coordinates of the bounding boxes that surround the identified items. These coordinates usually contain the upper-left and lower-right corners of the bounding box.

Object categorization seeks to assign a class label to every identified object. In the case of Faster R-CNN, the classification problem is distinguishing between different object classes, with an additional class for the background. The network is trained to classify objects into n + 1 classes, where n represents the number of distinct object categories, and the additional class represents the background.

Both bounding box refinement and object classification are supervised tasks, meaning that the network's predictions are compared to ground truth annotations during training to optimize the network parameters and improve its performance.

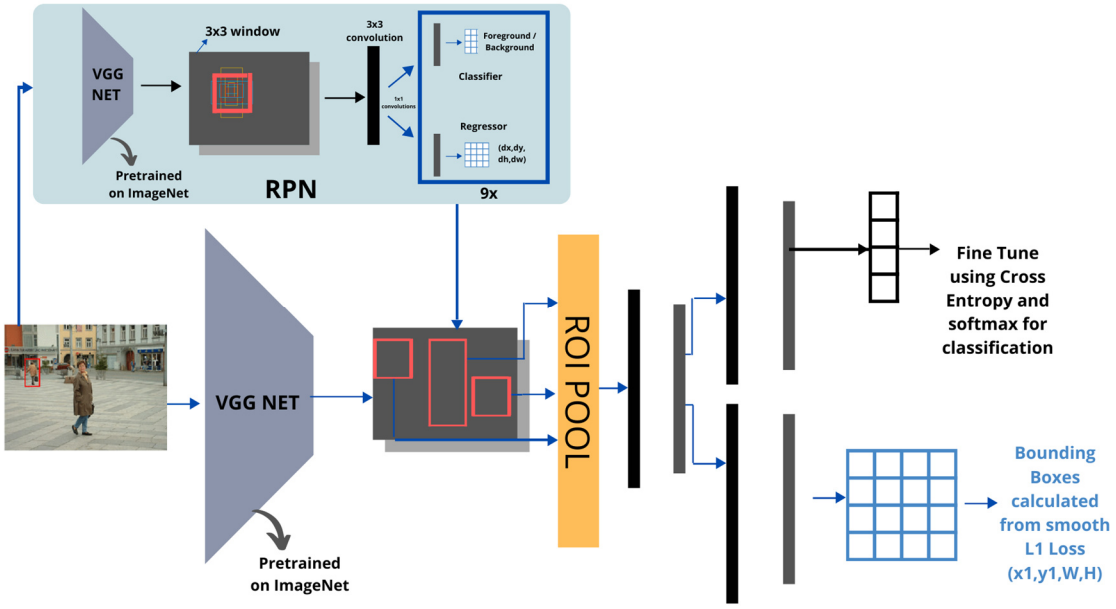

Fig 29 : Understanding the Fast RCNN and Faster RCNN Architecture

3.3.7 Long Short-Term Memory networks for question answering system

Long Short-Term Memory Networks (LSTMs) are a sort of a recurrent neural network (RNN) architecture developed to solve the vanishing gradient problem., which impedes the training of standard RNNs on long sequences. LSTMs have

emerged as powerful tools for modeling sequences, with applications in natural language processing, speech recognition, and time series analysis. LSTMs are designed to capture long-term dependencies in sequential data while limiting the influence of disappearing gradients. They use a memory cell and a set of gates to regulate information flow, allowing them to selectively keep or discard information over time. The memory cell is the central component of an LSTM, temporally persistent storage The memory cell has a self-connected recurrent connection, allowing it to maintain its state over multiple time steps. The cell state, denoted as C_t is updated at each time step based on the input, previous cell state, and gate activations.

- LSTMs use three types of gates to control the flow of information: input gate (i_t), forget gate (f_t), and output gate (o_t).
- Each gate is made up of a logistic function or the normal logistic function, which produces values ranging from 0 to 1 that indicate how much information should be allowed through.
- The input gate regulates the flow of new information into the cell state.
- The forget gate controls the extent to which previous information should be forgotten from the cell state.
- The output gate controls how much of the cell state is revealed to the network output.

The gates are computed using following equations

$$i_t = \sigma(W_i \,.\, [h_{t-1,} x_t] \; + \; b_i) \tag{6}$$

$$f_t = \sigma(W_f \,.\, [h_{t-1,} x_t] \; + \; b_f) \tag{7}$$

$$o_t = \sigma(W_o \,.\, [h_{t-1,} x_t] \; + \; b_o) \tag{8}$$

σ represents the sigmoid activation function, W denotes weight matrices, h_{t-1} is the previous hidden state, and $x_t$is the current input.

The cell state is updated using following equations

$$\underline{C_t} = tanh(W_c \,.\, [h_{t-1,} x_t] \; + \; b_c \tag{9}$$

$$C_t = f_i \,.\, C_{t-1} + i_t \,.\, \underline{C_t} \quad (10)$$

$\underline{C_t}$ represents the new candidate values to be added to the cell state.

Finally, the output of the LSTM at time step t, denoted as h_t, is computed as:

$$h_t = o_t \,.\, \tanh(C_t)$$

The output gate regulates the exposure of the cell state $C_t$to produce the final hidden state h_t for the current time step.

LSTMs are commonly trained using optimization techniques such as backpropagation through time (BPTT) or gradient descent, with gradients computed using backpropagation through time (BPTT) or more sophisticated methods like Long Short-Term Memory networks in deep learning. Throughout the training process, the LSTM's parameters, including gate weights and biases, are iteratively adjusted to minimize a specified loss function, such as cross-entropy loss or mean squared error.

Several strategies and versions of Long Short-Term Memory (LSTM) networks were created to address certain difficulties or suit particular use cases. The conventional LSTM design is made up of memory cells, input gates, forget gates, and output gates, as indicated in the previous detailed perspective. BiLSTMs analyze input sequences in both forward and backward orientations, allowing the network to collect data from both past and future contexts concurrently. This leads to a better understanding of context and enhanced performance in tasks such as sequence labeling and sentiment analysis.. The placed LSTMs are made up of many LSTM layers placed on top of one another. Each layer gets input from the preceding layer and produces output for the subsequent layer. Stacking LSTMs enables the model to acquire hierarchical representations of sequential data, facilitating the capture of intricate patterns and dependencies. Alternatively, GRUs offer a simpler architecture with fewer parameters compared to LSTMs. GRUs consolidate the functionalities of input and forget gates into a single "update gate," enhancing computational efficiency while preserving long-term dependencies. Additionally, attention mechanisms enhance LSTM networks by enabling them to direct attention towards specific segments of the input sequence during output generation. Attention mechanisms enable the model to adaptively allocate its focus to salient information, dynamically assigning varying degrees of importance to different segments of the input sequence. This dynamic

weighting facilitates improved performance across various tasks such as machine translation, image captioning, and summarization. Several LSTM variations have been created to solve distinct issues in different domains. For example, medical LSTM variants may incorporate domain-specific features or pre-training on medical data to improve performance in healthcare-related tasks. Researchers and practitioners often develop customized LSTM architectures tailored to specific tasks or datasets. These customized architectures may include additional layers, connections, or modifications to the standard LSTM architecture to optimize performance for the given task.

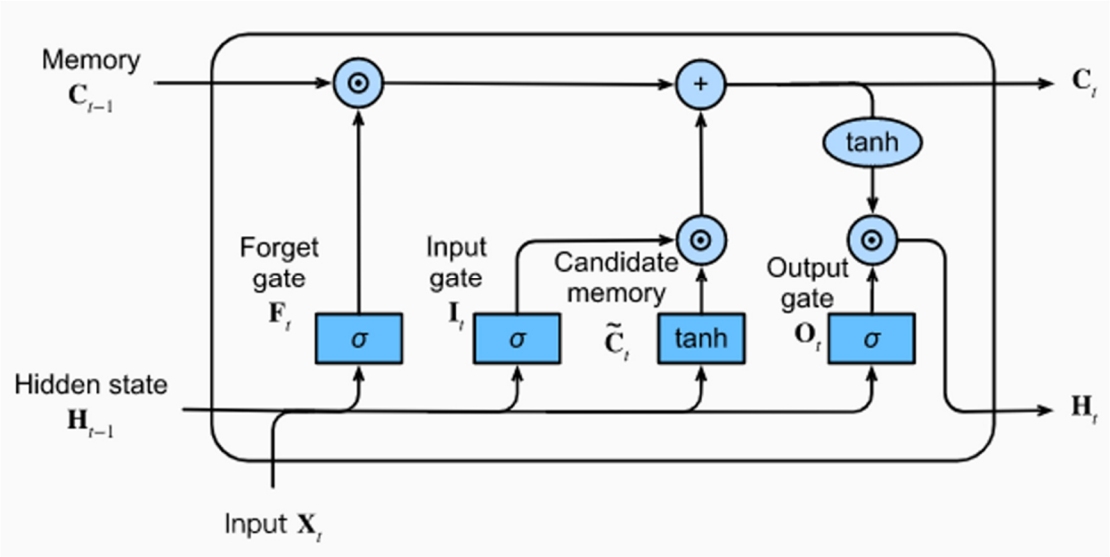

Fig 30 : Architecture of Long Short-Term Memory (LSTM) networks

3.4 The proposed Approach for Visual Question Answering System (VQAS) for skeletal Images

The Visual Textual System or VQA model represents a significant advancement in artificial intelligence research, aiming to equip machines with the ability to grasp visual content and effectively respond to questions posed about it. In the domain of medical visual question answering (Med-VQA), this technology takes on added importance as it endeavors to provide accurate responses to clinical queries based on radiological images.

In Med-VQA, the task involves presenting the computer with a radiological image alongside a relevant clinical question. The objective is to develop a sophisticated system capable of analyzing the visual information within the image and generating

appropriate answers to the posed questions. This demands a comprehensive understanding of both medical imaging and natural language processing, making Med-VQA a challenging yet promising field of study.

The ultimate goal of Med-VQA is to enhance medical diagnostics, decision-making, and patient care by leveraging the capabilities of artificial intelligence to assist healthcare professionals in interpreting radiological images and extracting valuable insights from them. By seamlessly integrating image analysis with question answering capabilities, Med-VQA has the capability to innovate medical imaging interpretation and make a substantial contribution to the advancement of healthcare practices.

Through rigorous research and development efforts in Med-VQA, we aim to achieve breakthroughs that not only augment the efficiency and accuracy of clinical decision-making but also enhance patient outcomes and overall healthcare delivery. With its formalized methodologies and interdisciplinary approach, Med-VQA represents a pivotal area of exploration at the intersection of artificial intelligence and healthcare, with far-reaching implications for the future of medical diagnostics and treatment.

3.4.1 Skeletal image Feature Extraction using Block_12_add Faster R-CNN (B12- FRCNN) algorithm

Block_12_add Faster R-CNN (B12-FRCNN) extends the Faster R-CNN (Region-based Convolutional Neural Network) approach, which is widely used for object detection. B12-FRCNN improves the performance of the Faster R-CNN architecture by including an additional block known as Block 12.

B12-FRCNN has two basic components: The region proposal network (RPN), as well as the object detection network. The RPN creates region proposals, which are candidate bounding boxes that may include objects of interest. The object detection network refines and classifies these proposals before producing the final detection findings.

B12-FRCNN inserts Block 12 into the Faster R-CNN network's feature extraction backbone. This block is often made up of numerous convolutional layers, followed by activation functions and pooling layers. Block 12's goal is to extract additional discriminative features from the input skeletal images, increasing the accuracy of object detection.

Block 12's design may change depending on the application and task requirements. It can be tailored to the complexity of the dataset, the diversity of the items to be detected, and the availability of computer resources.

Overall, B12-FRCNN extends the capabilities of the original Faster R-CNN algorithm by incorporating additional layers for feature extraction, thereby enhancing its ability to detect objects accurately and efficiently. This makes it a valuable tool for a wide range of computer vision applications, including object detection in images and videos, surveillance, autonomous driving, and medical imaging.

Block_12_add Faster R-CNN (B12-FRCNN) represents an enhanced iteration of the Faster R-CNN approach, renowned for its state-of-the-art capabilities in object detection within images. B12-FRCNN elevates the efficacy of Faster R-CNN by integrating a novel component known as Block 12 into its architecture. This pivotal block is seamlessly integrated into the feature extraction backbone of the network and is specifically designed to extract more discriminative features from input images. Faster R-CNN, a two-stage object detection methodology, comprises the region proposal network (RPN) and the object detection network. The RPN is responsible for generating region proposals, which serve as candidate bounding boxes potentially containing objects of interest. Subsequently, the object detection network refines and categorizes these proposals, ultimately yielding the definitive detection outcomes. B12-FRCNN adds an additional block, known as Block 12, to the Faster R-CNN network's feature extraction backbone. Block 12 is made up of many convolutional layers, subsequent to activation functions and pooling layers. The goal of Block 12 is to extract additional discriminative features from the input images, allowing the network to catch finer details and subtleties in the visual information. In the feature extraction process, the input image is transmitted via Block 12's convolutional layers. Each convolutional layer uses a sequence of learnable filters to extract features from the input image, including edges, textures, and forms. Activation functions, such as ReLU (Rectified Linear Unit), are then used to add nonlinearity to the network and allow it to learn complex patterns.Pooling layers play a crucial role in feature map downsampling, effectively reducing spatial dimensions while retaining essential information. In the case of Block 12, retrieved feature maps undergo integration into the Faster R-CNN framework. These enhanced feature maps serve as the foundation

for both the region proposal network (RPN) and the object detection network. The RPN utilizes these feature maps to generate region ideas, subsequently refined and categorized by the object detection network to discern objects within the image. Through the incorporation of Block 12 into the Faster R-CNN architecture, B12-FRCNN achieves notable enhancements in object detection accuracy and efficiency. The additional layers within Block 12 enable the network to extract more intricate and nuanced characteristics from input images, thereby augmenting detection performance. B12-FRCNN is particularly effective in scenarios where detecting small or intricate objects is challenging, as it can extract finer-grained features from the images.

In the proposed approach system, the feature extraction process plays an indispensable role in extracting discriminative features from radiology images to enable accurate question answering. Choosing the best feature extraction layer is critical for obtaining good performance in the visual question answering (VQA) assignment. Input data for the training phase includes radiological images and question-and-answer pairs. The goal is to train the VQA system to correctly answer questions concerning radiological images. The first stage of the training process involves extracting features from radiological images using the suggested Block_12_add Faster R-CNN (B12-FRCNN) method. B12-FRCNN is used to extract image characteristics because it can capture finer-grained visual information than the current Faster R-CNN method. In B12-FRCNN, the "block_12_add" layer is used as the visual feature extraction layer instead of the conventional "activation_40_relu" layer used in Faster R-CNN. This modification addresses the gradient vanishing problem associated with the "activation_40_relu" layer and improves the quality of the extracted features from medical images. The selection of the optimal feature extraction layer, such as "block_12_add," is a critical aspect of the training phase. Empirical analysis is conducted to evaluate the performance of different feature extraction layers and determine which layer yields the best results for the VQA task. This analysis involves training the VQA system using different feature extraction layers, including "block_12_add" and other candidate layers. Each configuration's performance is appraised using measures such as accuracy, precision, recall, and F1-score on a validation dataset. The layer that achieves the highest performance metrics

is selected as the optimal feature extraction layer for the VQA system. After selecting the optimal feature extraction layer, the VQA system is validated on a separate validation dataset to ensure its generalization performance. Fine-tuning may be performed to further optimize the VQA system's performance in accordance with the validation results. In the testing phase, the trained VQA system is evaluated on unseen radiology images and question-answer pairs to assess its performance in real-world scenarios. By empirically analyzing and selecting the optimal feature extraction layer, the proposed system ensures that the VQA system can effectively leverage visual information from radiology images to generate accurate answers to clinical questions.

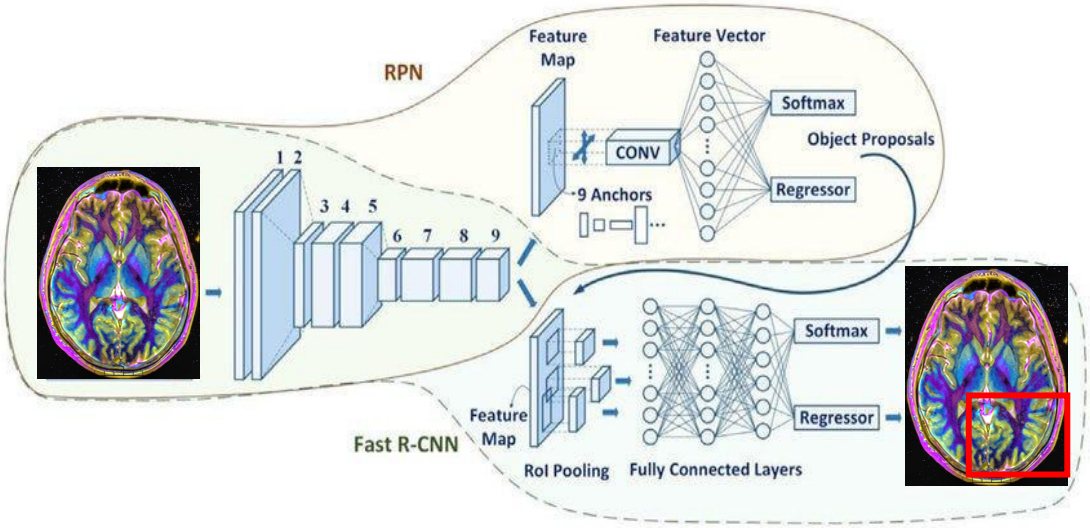

Fig 31 : B12-FRCNN, the "block_12_add" layer

B12-FRCNN begins by extracting features from medical images using a CNN backbone. Let's denote the input image as II, and the feature extraction function as $CNN(I)$. The output of this process is a set of feature maps denoted as $Feat = \{feat_1, feat_2, ..., feat_N\}$, where each $feat_i$ represents a feature map produced by the CNN backbone.

The RPN generates candidate regions of interest (RoIs) by scanning the feature maps Feat. Let p_i denote the position and scale information of the i^{th} anchor box, and f_i denote the feature vector extracted from the corresponding region of the feature map. The RPN predicts the probability $p(object)$ and adjusts the coordinates p_i to refine the proposed bounding boxes. RoI pooling extracts fixed-size feature vectors from each RoI to facilitate subsequent processing. Let RoI_i denote the i^{th} RoI, and

$feat_{RoI_i}$, denote the feature vector extracted from RoI_i. The RoI pooling operation transforms variable-sized RoIs into fixed-size feature vectors by dividing the RoI into a grid and applying max pooling within each grid cell. The extracted RoI features are passed through classification and regression heads. Let f_{RoI_i} denote the feature vector of RoI_i after RoI pooling. The classification head predicts the probability distribution over object classes $p(class|RoI_i, Feat)$, while the regression head refines the bounding box coordinates p_i based on the extracted features Feat. Finally, B12-FRCNN integrates with a question answering (QA) module to answer clinical queries based on the detected objects and their contextual information. The QA module processes textual questions Q related to the medical images and utilizes the detected objects' features $Feat$ to generate appropriate answers. The integration of object detection and question answering can be represented as follows: $Answer = QA(Feat)$ This equation indicates that the answer generated by the QA module is a function of the textual question Q and the extracted features $Feat$ from the medical images.

Overall, B12-FRCNN combines advanced object detection techniques with question answering capabilities, leveraging both skeletal image and question answer information to provide accurate and meaningful responses to clinical queries related to medical images.

3.4.2 The proposed approach Kai-Bi-LSTM for Question Answering feature extraction

The Question-Answer (QA) pairs undergo preprocessing, starting with the splitting of the question text into individual words. Subsequently, stop word removal eliminates frequently occurring words like "a", "an", and "the", which do not contribute to the text's meaning. The root forms of words are then extracted via stemming, which condenses words with different suffixes under the same root word. This process minimizes the index size and improves computational efficiency.

After preprocessing, keywords are extracted from the question, representing highly relevant information. Keyword extraction entails obtaining relevant information from a text. For the response, text-only splitting is used. This step's output is in string

format, which is then translated to numerical format for effective categorization. This conversion is based on the word representation hypothesis.

LogishBERT is used for this purpose, a variation of Bidirectional Encoder Representations from Transformers (BERT) that employs a Logarithmic Swish activation function to handle complex data more effectively than the traditional GELU activation function. It is an activation function commonly used in deep learning models, significantly in neural networks. GELU is designed to approximate the Gaussian cumulative distribution function and has been found to perform well in various tasks, including natural language processing and computer vision. It is defined mathematically as:

$$GELU(x) = x.P(X \leq x) = x.\Phi(x) \quad (10)$$

where $\Phi(x)$ represents the cumulative distribution function of the standard normal distribution.

LogishBERT converts the output from its fully connected layer into numerical data, referred to as a score value.

The radiological imagine feature and LogishBERT score value are merged and used to train a Kai-Bi-LSTM classifier to predict answers. This classifier employs Bidirectional Long Short-Term Memory (BiLSTM) networks to address dependencies and different timelines. The Kaiming Initialization function is used to set the activation value for the forget gate.

During the testing phase, the system receives a sample image and question, initiating feature extraction and preprocessing procedures akin to those employed during training. The extracted features and corresponding scores are then fed into the classifier, which predicts a score value corresponding to the answer. Subsequently, this score value and the question are cross-referenced with the LogishBERT lexicon to determine the appropriate response.

In the results analysis section, the performance of the proposed system is assessed by juxtaposing it against existing methodologies, scrutinizing its innovative aspects, and evaluating various performance metrics including accuracy, precision, recall, and F-measure. Figure 31 illustrates a block diagram delineating the proposed technique.

LogishBERT is an adaptation of Bidirectional Encoder Representations from Transformers (BERT), a pre-trained Linguistic-focused model understanding tasks. BERT has gained significant attention in the field of textual feature extraction processing due to its effectiveness in capturing contextual information and semantic relationships in text data. LogishBERT enhances the traditional BERT model by introducing a logarithmic swish activation function, which aims to handle complex data more effectively. BERT, an acronym for the same, adopts a transformer-based methodology to glean contextual insights from input text. Its architecture comprises multiple layers of bidirectional encoders, facilitating the extraction of nuanced contextual information. BERT undergoes pre-training on extensive text corpora, engaging in two unsupervised learning tasks: masked language modeling (MLM) and next sentence prediction. Renowned for its exceptional performance, BERT has showcased state-of-the-art results across a spectrum of natural language processing tasks, encompassing text classification, named entity recognition, and question answering. The traditional BERT model uses the GELU (Gaussian Error Linear Unit) activation function in its fully connected layers. LogishBERT replaces the GELU activation function with a logarithmic swish activation function. The logarithmic swish function is defined as.

$$logish(h) = \ln(1 + e\text{^}x \quad (11)$$

Compared to the GELU function, the logarithmic swish function introduces a logarithmic term in the activation function, which helps in handling complex data distributions more effectively. The logarithmic swish function is believed to provide smoother gradients and better performance on tasks with complex data distributions. After processing the input text data through the LogishBERT model, the output from the fully connected layers is converted into numerical scores. These numerical scores represent the likelihood or confidence of different answers or predictions. The scores are typically obtained by applying a softmax function to the output vector, which normalizes the values to probabilities, ensuring that they sum up to one. LogishBERT is particularly useful in question answering systems, where it can effectively process and understand textual data to generate accurate answers to user queries. It is integrated into the proposed system architecture described earlier, where it plays a

crucial role in converting textual input (questions and answers) into numerical representations for further processing and classification.

The Kaiming Initialization, also referred to as He Initialization, is a technique utilized to instigate the parameters of deep neural networks, including RN networks (RNNs) such as the Bidirectional LSTM (BiLSTM) model. This initialization method determines the initial weights of the neural network layers, encompassing the recurrent connections within the BiLSTM model. Its primary objective is to set the weights' initial values in a manner that prevents them from being excessively small or large, thereby mitigating issues related to vanishing or exploding gradients during the training process. Kaiming Initialization takes into consideration the activation function employed by the network, such as hyperbolic tangent (tanh) or rectified linear unit (ReLU), to ensure optimal performance. It adjusts the scale of initialization based on the characteristics of the activation function to ensure that the activations do not saturate too quickly, leading to gradient convergence/divergence issues. In the case of the BiLSTM model, which consists of multiple recurrent layers with forward and backward connections, the Kaiming Initialization function initializes the weights of both the forward and backward connections. It ensures that the weights of the recurrent connections are initialized in a way that allows information to flow effectively in both directions through the network during training. By initializing the weights appropriately, the Kaiming Initialization function helps in stabilizing the learning dynamics of the neural network. It facilitates smoother and more efficient training by preventing the gradients from becoming too small or too large, which can hinder the convergence of the optimization process. The Kaiming Initialization function combined with the BiLSTM algorithm can increase learning performance, accelerate convergence, and improve generalization to previously unknown material. It aids in overcoming the difficulties involved with training deep recurrent neural networks, particularly in tasks that require sequential data processing, such as natural language processing and time series analysis.

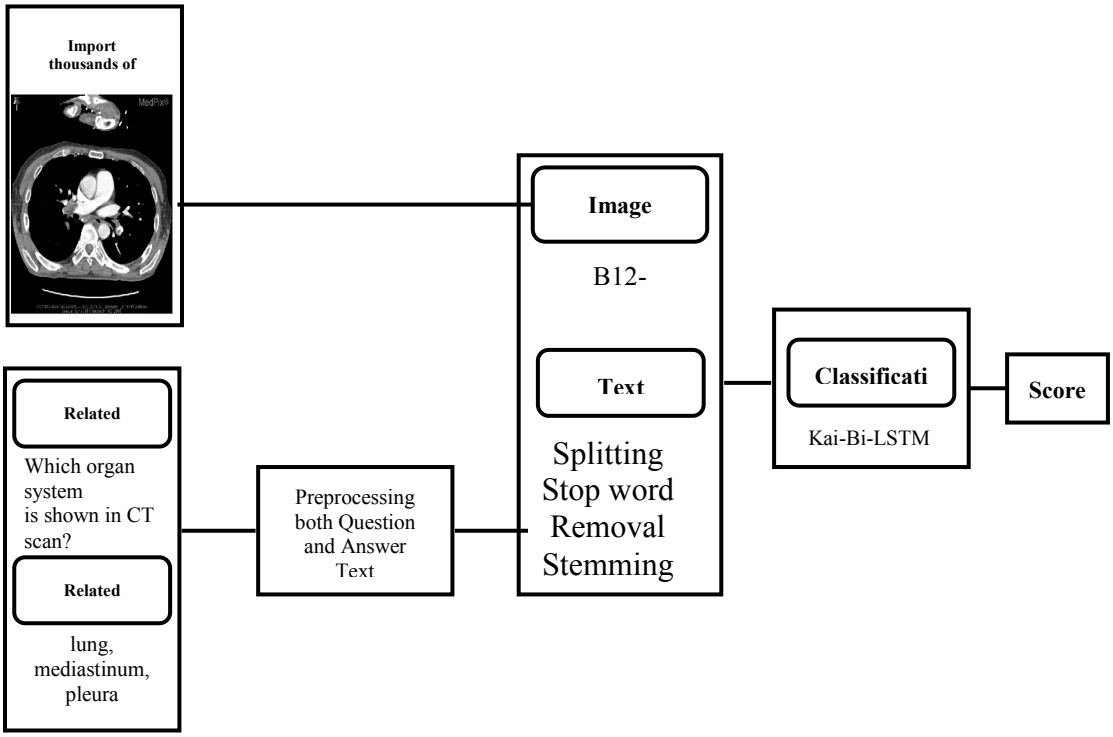

Fig 32 : Block diagram of QA System in the Training phase

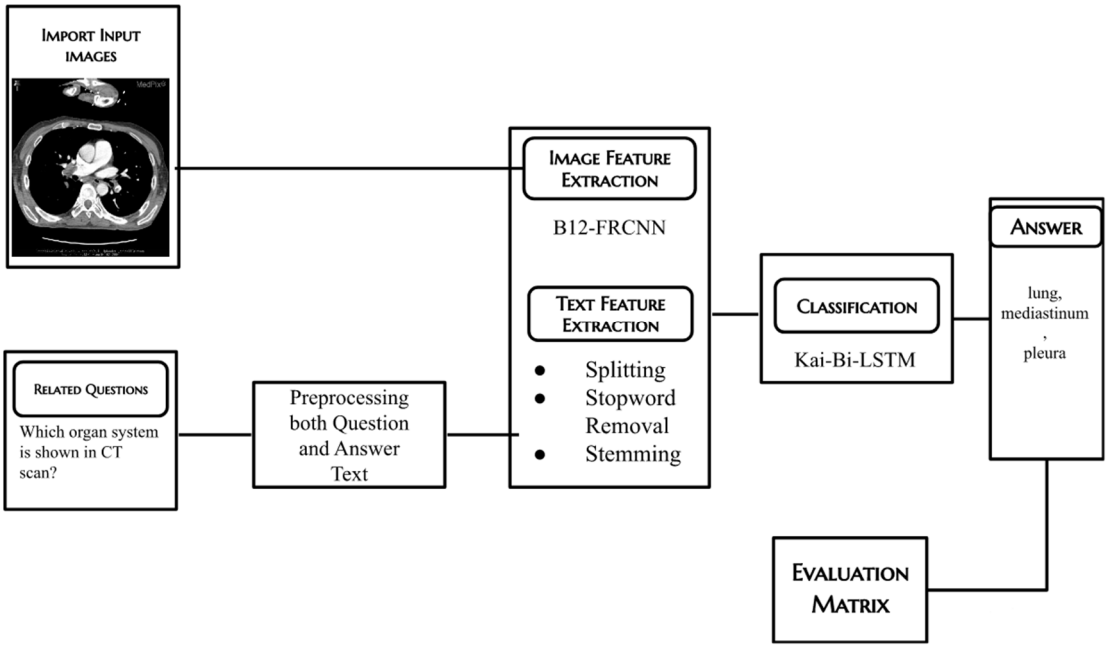

Fig 33 : Block diagram of QA System on Testing Phase