ACTIVE LEARNING METHODS FOR INTERACTIVE IMAGE RETRIEVAL

Prepared by:

G.E.M.LATHA(07241A1203)
CH.HARITHA(07241A1205)
D.PRATHYUSH(07241A1211)
M.SWACHITHA(07241A1220)
INTRODUCTION
Purpose:

Human interactive systems have attracted a lot of research interest in recent years, especially for content-based image retrieval systems. Contrary to the early systems, which focused on fully automatic strategies, recent approaches have introduced human-computer interaction. Here focus on the retrieval of concepts within a large image collection. We assume that a user is looking for a set of images, the *query concept*, within a database. The aim is to build a fast and efficient strategy to retrieve the query concept. In content-based image retrieval (CBIR), the search may be initiated using a query as an example. The top rank similar images are then presented to the user.
• **Project Scope:**

This project is mainly designed considering the scenarios for comparing the images dynamically. Active learning methods are used to interact with the user. Although it is used for image comparison which fulfills exact requirements according to the user. But it is not used in case of video applications. Finally, we compare our method to existing ones using real scenario on large databases.
Intended Audience and Reading Suggestions:

- The intended audience for this document is the internal guides of the organization where the team has developed the project. Further modifications and reviewing will be done by the organization and deliver a final version. The final version of this document is reviewed by the Internal Guides and Head of the Department of the college.

- The sequence to follow better understanding is here Purpose, Scope, Operating requirements, Advantages, Requirements etc. In the rest of the part of this SRS is mentioned with what are our product benefits, how to use this product, how the product was developed, what are the major things we have taken into the consideration all are mentioned in this rest of the part of the SRS. In the document at first we have discussed the importance of our product and functionality of the product and the software we used and all the way how we can get utilized this software is mentioned in this document.
**Design and Implementation Constraints:**

- The software is designed in such a way that the user can easily interact with the screen because they are GUI. The customer has to enter the image dynamically and select any folder then start comparing the images one by one. So the set of similar images are displayed.
User Documentation:

- In our user manual we are going to perform image comparison in a very interactive way, which can be understandable by a new person who is going to use it. If a new person is using it, necessary information will be provided. We are going to explain each and every step clearly about our product so that any user can easily understand it.
SYSTEM ANALYSIS AND DESCRIPTION
Existing System:

- In the existing system the CBIR method faced a lot of disadvantage in case of the image retrieval. The following are the main disadvantage faced in case of the medical field - Medical image description is an important problem in content-based medical image retrieval. Hierarchical medical image semantic features description model is proposed according to the main sources to get semantic features currently. Hence we propose the new algorithm to over come the existing system. In existing system, Images were first annotated with text and then searched using a text-based approach from traditional database management systems.
Proposed System:

- In the proposed system we use the retin technique. In this technique the user gives his input in the form of an image. We then check for the images in the training set rather than going to the database to search the images. The training set contains the images that are frequently searched by the user. Then all the relevant images are compared and the top rank similar images are displayed as the output. Here the efficiency and accuracy are increased and the drawbacks of the existing system are overcome.
Project Description:

- The project entitled as “Active Learning Methods for Interactive Image Retrieval” developed using .NET using C#. Modules display as follows.

- Comparable image
  - Image Utility
    - RGB Projections

- Similarity measure

- Graph
Comparable Image:

- From a method called as Process Image in the main method we come to the comparable images module. It acts as a loop. It has one sub method in it. That method is called as Image Utility.

- **Image Utility**

- The Image Utility method senses each and the red blue and green colors in every pixel and also calculates the luminosity and adds it to the horizontal and vertical projections. It also locks the bits of the image so that other users do not have the right to modify the images being used. After done with all the calculations it unlocks the bits of the image. It has a sub method called as the RGB Projections.

- **RGB Projections**

- The RGB Projection senses each and every pixel in both horizontal and vertical directions. Basically it calculates the horizontal and vertical projections.
Similarity Measure:

The Similarity Measures module calculates the similarity between the source image and destination image. It also sorts the images in descending order of their similarity and displays it in the form of a grid.

Graph:

The Graph module displays the graphs for both source and destination images in the form of histograms. The graph for each image has two curves one for horizontal projection and the other for vertical projection. We even calculate the co ordinates for each curve in this module.
Input/Output

- The image will take the relevant image what the user search. One can see that we have selected concepts of different levels of complexities. The performances go from few percentages of Mean average precision to 89%. The concepts that are the most difficult to retrieve are very small and/or have a very diversified visual content.
Image Retrieval:

- An **image retrieval** system is a computer system for browsing, searching and retrieving images from a large **database** of **digital images**. Most traditional and common methods of image retrieval utilize some method of adding **metadata** such as captioning, **keywords**, or descriptions to the images so that retrieval can be performed over the annotation words. Manual image annotation is time-consuming, laborious and expensive; to address this, there has been a large amount of research done on **automatic image annotation**. Additionally, the increase in social **web applications** and the **semantic web** have inspired the development of several web-based image annotation tools.
Search Methods

- Image search is a specialized data search used to find images. To search for images, a user may provide query terms such as keyword, image file/link, or click on some image, and the system will return images "similar" to the query. The similarity used for search criteria could be meta tags, color distribution in images, region/shape attributes, etc.

- **Image meta search** - search of images based on associated metadata such as keywords, text, etc.

- **Content-based image retrieval** (CBIR) – the application of computer vision to the image retrieval. CBIR aims at avoiding the use of textual descriptions and instead retrieves images based on their visual similarity to a user-supplied query image or user-specified image features.
CBIR:

- Content-based image retrieval (CBIR), also known as query by image content (QBIC) and content-based visual information retrieval (CBVIR) is the application of computer vision to the image retrieval problem, that is, the problem of searching for digital images in large databases. (see this survey[1] for a recent scientific overview of the CBIR field).

- "Content-based" means that the search will analyze the actual contents of the image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. Without the ability to examine image content, searches must rely on metadata such as captions or keywords, which may be laborious or expensive to produce.
History

- The term CBIR seems to have originated in 1992, when it was used by T. Kato to describe experiments into automatic retrieval of images from a database, based on the colors and shapes present. Since then, the term has been used to describe the process of retrieving desired images from a large collection on the basis of syntactical image features. The techniques, tools and algorithms that are used originate from fields such as statistics, pattern recognition, signal processing, and computer vision.
Technical Progress

• There is a growing interest in CBIR because of the limitations inherent in metadata-based systems, as well as the large range of possible uses for efficient image retrieval.

• Textual information about images can be easily searched using existing technology, but requires humans to personally describe every image in the database.

• This is impractical for very large databases, or for images that are generated automatically, e.g. from surveillance cameras. It is also possible to miss images that use different synonyms in their descriptions.

• Systems based on categorizing images in semantic classes like "cat" as a subclass of "animal" avoid this problem but still face the same scaling issues.
Potential uses for CBIR include:

- Art collections
- Photograph archives
- Retail catalogs
- Medical diagnosis
- Crime prevention
- The military
- Intellectual property
- Architectural and engineering design
- Geographical information and remote sensing systems
CBIR Software Systems and Techniques

- Many CBIR systems have been developed, but the problem of retrieving images on the basis of their pixel content remains largely unsolved.

Public CBIR Systems

- See CBIR engines for examples of publicly available and accessible CBIR systems.

Query Techniques

- Different implementations of CBIR make use of different types of user queries.
Query by Example

- Query by example is a query technique that involves providing the CBIR system with an example image that it will then base its search upon. The underlying search algorithms may vary depending on the application, but result images should all share common elements with the provided example.

- Options for providing example images to the system include:
  - A preexisting image may be supplied by the user or chosen from a random set.
  - The user draws a rough approximation of the image they are looking for, for example with blobs of color or general shapes.
  - This query technique removes the difficulties that can arise when trying to describe images with words.
Semantic Retrieval

- The ideal CBIR system from a user perspective would involve what is referred to as *semantic* retrieval, where the user makes a request like "find pictures of dogs" or even "find pictures of Abraham Lincoln".

- This type of open-ended task is very difficult for computers to perform - pictures of chihuahuas and Great Danes look very different, and Lincoln may not always be facing the camera or in the same pose.

- Current CBIR systems therefore generally make use of lower-level features like texture, color, and shape, although some systems take advantage of very common higher-level features like faces (see facial recognition system).

- Not every CBIR system is generic. Some systems are designed for a specific domain, e.g. shape matching can be used for finding parts inside a CAD-CAM database.
Other Query Methods

- Other query methods include browsing for example images, navigating customized/hierarchical categories, querying by image region (rather than the entire image), querying by multiple example images, querying by visual sketch, querying by direct specification of image features, and multimodal queries (e.g. combining touch, voice, etc.)

- CBIR systems can also make use of *relevance feedback*, where the user progressively refines the search results by marking images in the results as "relevant", "not relevant", or "neutral" to the search query, then repeating the search with the new information.
Content Comparison Techniques

- The sections below describe common methods for extracting content from images so that they can be easily compared. The methods outlined are not specific to any particular application domain.

Color
- Retrieving images based on color similarity is achieved by computing a color histogram for each image that identifies the proportion of pixels within an image holding specific values (that humans express as colors). Current research is attempting to segment color proportion by region and by spatial relationship among several color regions. Examining images based on the colors they contain is one of the most widely used techniques because it does not depend on image size or orientation. Color searches will usually involve comparing color histograms, though this is not the only technique in practice.
Texture

- Texture measures look for visual patterns in images and how they are spatially defined. Textures are represented by texels which are then placed into a number of sets, depending on how many textures are detected in the image.

- These sets not only define the texture, but also where in the image the texture is located. Texture is a difficult concept to represent.

- The identification of specific textures in an image is achieved primarily by modeling texture as a two-dimensional gray level variation.

- The relative brightness of pairs of pixels is computed such that degree of contrast, regularity, coarseness and directionality may be estimated (Tamura, Mori & Yamawaki, 1978).

- However, the problem is in identifying patterns of co-pixel variation and associating them with particular classes of textures such as "silky, or "rough."
Shape

- Shape does not refer to the shape of an image but to the shape of a particular region that is being sought out.
- Shapes will often be determined first by applying [segmentation](#) or [edge detection](#) to an image.
- Other methods like Tushabe and Wilkinson 2008 use shape filters to identify given shapes of an image.
- In some cases, accurate shape detection will require human intervention because methods like segmentation are very difficult to completely automate.
Applications

- Some software producers are trying to push CBIR based applications into the filtering and law enforcement markets for the purpose of identifying and censoring images with skin-tones and shapes that could indicate the presence of nudity, with controversial results.

- The CBIR technology has been used in several applications such as:
  - fingerprint identification
  - biodiversity information systems
  - digital libraries
  - crime prevention
  - medicine
  - historical research
Medical Applications

- The use of CBIR can result in powerful services that can benefit biomedical information systems.
- Three large domains can instantly take advantage of CBIR techniques: teaching, research, and diagnostics [73]. From the teaching perspective, searching tools can be used to find important cases to present to students. Research also can be enhanced by using services combining image content information with different kinds of data.
- For example, scientists can use mining tools to discover unusual patterns among textual (e.g., treatments reports, and patient records) and image content information.
- Similarity queries based on image content descriptors can also help the diagnostic process. Clinicians usually use similar cases for case-based reasoning in their clinical decision-making process. In this sense, while textual data can be used to find images of interest, visual features can be used to retrieve relevant information for a clinical case (e.g., comments, related literature, HTML pages, etc.).
Biodiversity Information Systems

- Biologists gather many kinds of data for biodiversity studies, including spatial data, and images of living beings.
- Ideally, Biodiversity Information Systems (BIS) should help researchers to enhance or complete their knowledge and understanding about species and their habitats by combining textual, image content-based, and geographical queries.
- An example of such a query might start by providing an image as input (e.g., a photo of a fish) and then asking the system to “Retrieve all database images containing fish whose fins are shaped like those of the fish in this photo”.
- A combination of this query with textual and spatial predicates would consist of “Show the drainages where the fish species with ‘large eyes’ coexists with fish whose fins are shaped like those of the fish in the photo”. Examples of initiatives in this area include [55, 74].
Digital Libraries

- There are several digital libraries that support services based on image content [74–79]. One example is the digital museum of butterflies [74], aimed at building a digital collection of Taiwanese butterflies.

- This digital library includes a module responsible for content-based image retrieval based on color, texture, and patterns. In a different image context, Zhu et al. [76] present a content-based image retrieval digital library that supports geographical image retrieval.

- The system manages air photos which can be retrieved through texture descriptors. Place names associated with retrieved images can be displayed by cross referencing with a Geographical Name Information System (GNIS) gazetter.

- In this same domain, Bergman et al. describe an architecture for storage and retrieval of satellite images and video data from a collection of heterogeneous archives. Other initiatives cover different concepts of the CBIR area. For example, while research presented in [77,78] concentrates on new searching strategies for improving the effectiveness of CBIR systems, another popular focus is on proposing image descriptors [79].
Digital Crime Scene Images

- The digital image databases maintain large collection of images. They can be used to store, archive, browse and search interested data.
- Nowadays, fingerprint images have been used in law enforcement and become increasingly popular for access control to secure information and identity.
- The relevance of the digital image databases for crime scene applications is to organize collections of scene pictures, evidence pictures, trace evidence images, vehicle number images, identification documents and investigation documents, and so on.
- In CBIR, we provide a retrieval method for digital image databases of crime scene photos.
SDLC Model:

Life Cycle Model

- The life cycle model, which we use in our application, is RAD (Rapid Application Development) model. RAD is a concept that products can be developed faster with higher quality through,
- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that defers design improvements to the next product version.
- Less formality in reviews and other team communication.
SYSTEM STUDY
Feasibility Study:

- The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates.
- During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company.
- For feasibility analysis, some understanding of the major requirements for the system is essential.
- Three key considerations involved in the feasibility analysis are:
  - ECONOMICAL FEASIBILITY
  - TECHNICAL FEASIBILITY
  - SOCIAL FEASIBILITY
Economical Feasibility:

- This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility:

- This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.
Social Feasibility:

- The aspect of study is to check the level of acceptance of the system by the user.
- This includes the process of training the user to use the system efficiently.
- The user must not feel threatened by the system, instead must accept it as a necessity.
- The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.
- His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.
SYSTEM SPECIFICATION
Hardware Requirements:

- **SYSTEM**: Pentium IV 2.4 GHz
- **HARD DISK**: 40 GB
- **FLOPPY DRIVE**: 1.44 MB
- **MONITOR**: 15 VGA colour
- **MOUSE**: Logitech.
- **RAM**: 256 MB
- **KEYBOARD**: 110 keys enhanced.
Software Requirements:

- Operating system : Windows XP Professional
- Coding Language : C# 2.0
NON - FUNCTIONAL REQUIREMENTS
Performance Requirements

Usability

- A very interactive GUI is provided which helps the user to easily interact with the system. This application allows users to operate with little or no learning.

Portability

- This application is developed on dot net platform, which runs under CLR (Common Language Runtime) of that particular operating system like Windows, UNIX, and Linux. At present we have CLR working on windows platform in the market. So this application is portable to work under CLR environment imported on windows OS.
**Response Time**
- It is the time a generic system or functional unit takes to react to a given input.
- Responsiveness, how quickly an interactive system responds to user input. This will take a second or two to respond for the files with more than 100 classes.

**Size**
- For better usage of RAM chip memory in our application constraints are followed like re-usage of the variables and objects, garbage collection of unused objects etc.

**Safety Requirements**
- No harm is expected from the use of the product either to the OS or any data that resides on the client system.
PROJECT DESIGN

SOFTWARE DESIGN
The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.
The Nine UML diagrams are:

**Use case diagram:**
- The use case diagram is used to identify the primary elements and processes that form the system.
- The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

**Class diagram:**
- The class diagram is used to refine the use case diagram and define a detailed design of the system.
- The class diagram classifies the actors defined in the use case diagram into a set of interrelate classes.
- The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities.
- These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.
Object diagram:
- The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.

State diagram:
- A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

Activity diagram:
- The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.
Sequence diagram:

- A sequence diagram represents the interaction between different objects in the system.
- The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step.
- Different objects in the sequence diagram interact with each other by passing "messages".

Collaboration diagram:

- A collaboration diagram groups together the interactions between different objects.
- The interactions are listed as numbered interactions that help to trace the sequence of the interactions.
- The collaboration diagram helps to identify all the possible interactions that each object has with other objects.
**Component diagram:**

- The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

**Deployment diagram:**

- The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.
SYSTEM
DEVELOPMENT
ENVIRONMENT
The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components:

- The common language runtime and the .NET Framework class library.
- The common language runtime is the foundation of the .NET Framework.
You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.
For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable ASP.NET applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.
Features of .NET:

- **MICROSOFT .NET IS A SET OF MICROSOFT SOFTWARE TECHNOLOGIES FOR RAPIDLY BUILDING AND INTEGRATING XML WEB SERVICES, MICROSOFT WINDOWS-BASED APPLICATIONS, AND WEB SOLUTIONS. THE .NET FRAMEWORK IS A LANGUAGE-NEUTRAL PLATFORM FOR WRITING PROGRAMS THAT CAN EASILY AND SECURELY INTEROPERATE. THERE’S NO LANGUAGE BARRIER WITH .NET: THERE ARE NUMEROUS LANGUAGES AVAILABLE TO THE DEVELOPER INCLUDING MANAGED C++, C#, VISUAL BASIC AND JAVA SCRIPT. THE .NET FRAMEWORK PROVIDES THE FOUNDATION FOR COMPONENTS TO INTERACT SEAMlessly, WHETHER LOCALLY OR REMOTELY ON DIFFERENT PLATFORMS. IT STANDARDIZES COMMON DATA TYPES AND COMMUNICATIONS PROTOCOLS SO THAT COMPONENTS CREATED IN DIFFERENT LANGUAGES CAN EASILY INTEROPERATE.**

- “.NET” is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).
The .NET Framework

- The .NET Framework has two main parts:
- 1. The Common Language Runtime (CLR).
- 2. A hierarchical set of class libraries.
- The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run.
- The most important features are:
  - Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
  - Memory management, notably including garbage collection.
  - Checking and enforcing security restrictions on the running code.
  - Loading and executing programs, with version control and other such features.
- The following features of the .NET framework are also worth description:
Managed Code:

- The code that targets .NET, and which contains certain extra information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.
Managed Data:

- With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.
Common Type System:

- The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling.

- As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.
The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.
The Class Library:

- .NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

- The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity.

- The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.
The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.
Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

Other languages for which .NET compilers are available include:
- FORTRAN
- COBOL
- Eiffel
Features of C#.NET:

- C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

- C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET. The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.