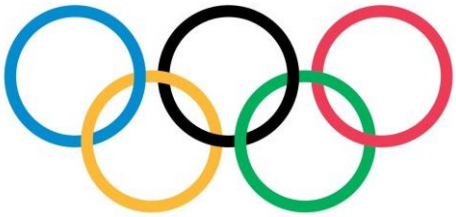
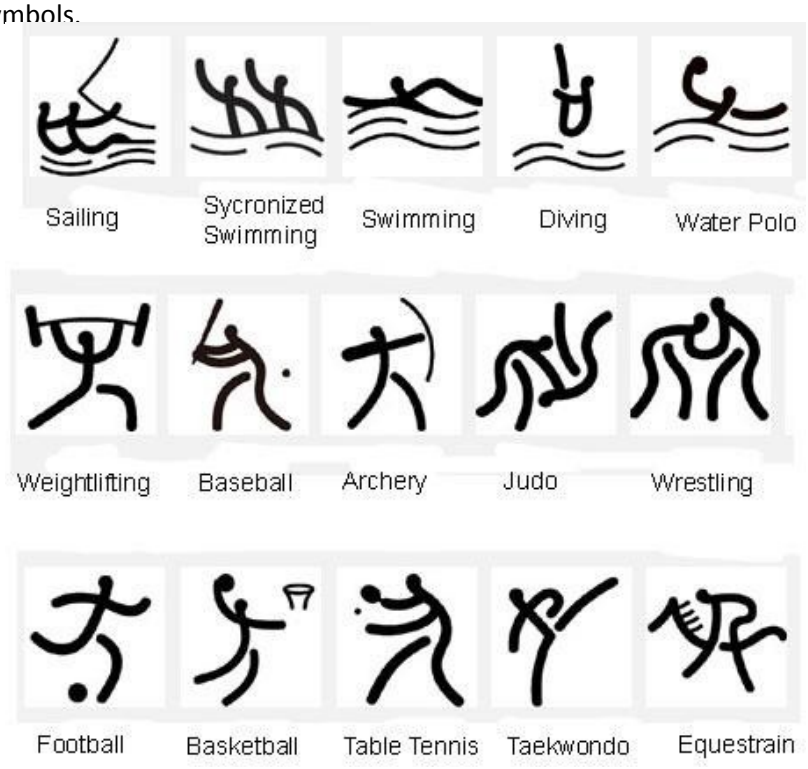
Pattern classification – Recognition of Olympic games symbols

The International Olympic Committee (IOC) uses icons, flags and symbols to elevate the Olympic Games.



These symbols include those commonly used during Olympic competition—such as the flame, fanfare and theme—as well as those used throughout the years, such as the Olympic flag.

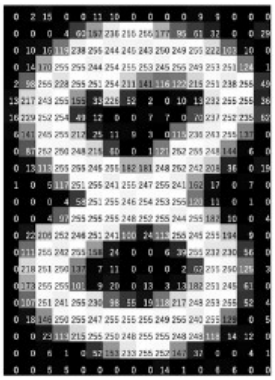
Recognizing Olympic Games symbols can be approached as a pattern classification problem. The first step in solving this problem is to obtain a dataset of Olympic Games symbols.

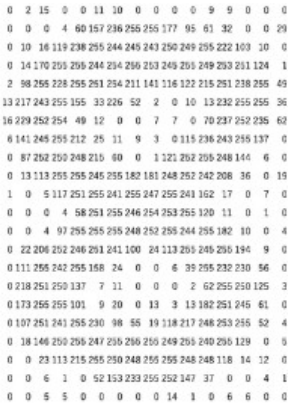


This dataset can be created by gathering images of Olympic Games symbols from various sources, such as official Olympic websites, news articles, and social media.

Once the dataset is obtained, the next step is to preprocess the images.
This can involve resizing the images to a standard size, converting them to grayscale or binary format, and removing any noise or artifacts present in the images.

After preprocessing, the next step is to **extract features from the images** .
This can involve using techniques such as edge detection, corner detection, texture analysis, and color analysis to identify key features of the symbols.





Once the features are extracted, a machine learning algorithm can be trained to recognize the symbols. This can involve using techniques such as support vector machines (SVMs), random forests, or neural networks to classify the symbols based on their features.

To evaluate the performance of the classification algorithm, the dataset can be split into training and testing sets.
The algorithm can be trained on the training set and then tested on the testing set to measure its accuracy in recognizing the symbols.

Why Pattern classification is required in Recognition of Olympic Games symbols?

Pattern classification is required in the recognition of Olympic Games symbols because it allows us to automatically identify and categorize images based on their visual features. In the case of Olympic Games symbols, there are many different symbols that represent different sports, countries, and events, and they can vary in size, color, and style.

Without pattern classification, it would be very difficult to manually identify and classify each symbol, especially if the dataset is large. However, by using machine learning techniques such as support vector machines (SVMs), random forests, or neural networks, we can train a computer to recognize patterns in the images and classify them into different categories automatically.

Pattern classification also allows for scalability and adaptability. As the dataset of Olympic Games symbols grows over time, the pattern classification algorithm can be retrained with new data to improve its accuracy. Additionally, the same techniques used to recognize Olympic Games symbols can be applied to other domains where image recognition is needed, such as medical imaging, surveillance, and robotics.

Recognition of printed Characters

It is a process to perform electronic conversion of the text on a physical paper. The text on the document can be either machine print or handwritten.

> Note:
> If the physical paper has typed text then chances are high for an exact match.
> But if it has handwritten text, it will use artificial intelligence algorithm to find a corresponding character.

Machine printed character recognition uses intelligent document recognition technology to read the pattern of the text on the physical paper and convert them accurately in digital format. It increases efficiency and work productivity like never before. It plays the major part in digitization of companies and making workplaces more efficient.

Image preprocessing involves techniques such as noise reduction, contrast enhancement, and image normalization, to improve the quality of the image and make it suitable for further analysis.

Here are the steps involved in the recognition of printed characters using AI:

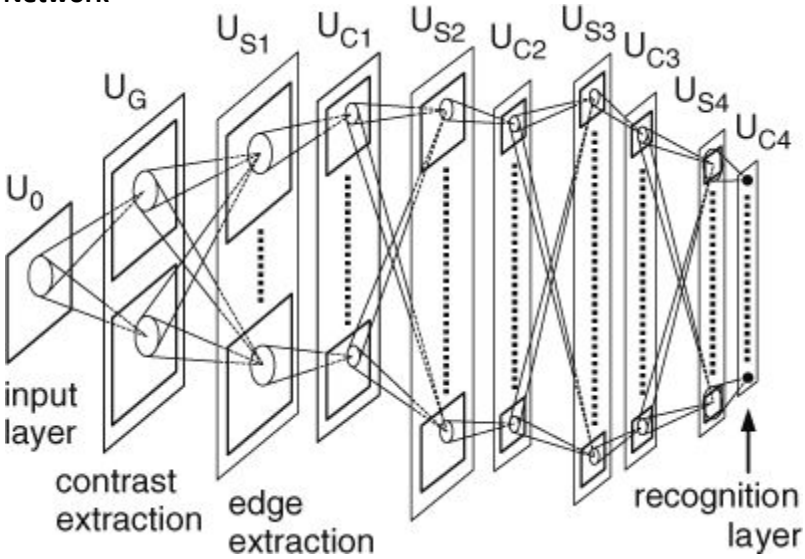
1. **Image acquisition:** The first step is to obtain a digital image of the printed document or text. This can be done using a scanner or a camera.
2. **Preprocessing:** The acquired image may contain noise or other artifacts that can interfere with character recognition. Preprocessing techniques such as filtering, binarization, and noise reduction can help improve the quality of the image and make it more suitable for analysis.
3. **Segmentation:** In this step, the image is divided into smaller regions, each containing a single character. This is done to make it easier to recognize each character individually. Techniques such as thresholding and contour detection can be used for segmentation.
4. **Feature extraction:** Each segmented character is analyzed to identify its unique features, such as shape, size, and orientation. Feature extraction can be done using techniques such as edge detection, morphological operations, and texture analysis.
5. **Classification:** Once the features of each character have been extracted, a machine learning algorithm is used to match the features to known characters and recognize them. Popular classification algorithms used in OCR include Support Vector Machines (SVMs), Random Forests, and Convolutional Neural Networks (CNNs).
6. Post-processing: The recognized characters may contain errors, such as misrecognitions or false positives. Post-processing techniques such as spell-checking and dictionary lookup can be used to correct these errors and improve the accuracy of the recognition results.

Neocognitron – Recognition of handwritten characters

The neocognitron is a hierarchical, multilayered artificial neural network proposed by Kunihiko Fukushima in 1979.
It has been used for Japanese handwritten character recognition and other pattern recognition tasks.

Neocognitron	**Cognitron**
Developed by Kunihiko Fukushima in the 1980s	Developed by Kunihiko Fukushima and Walter Freeman in the 1970s
A hierarchical, multilayered neural network	A single-layer neural network
Designed for pattern recognition, specifically handwritten characters	Designed for visual processing, specifically edge detection
Consists of alternating S-layers and C-layers	Consists of a single layer of processing units
Each layer extracts increasingly complex features from the input	Performs simple linear and non-linear transformations on the input
Uses backpropagation algorithm for training	Trained using a simple Hebbian learning rule
Has been used in applications such as handwriting recognition, speech recognition, and image recognition	Has been used in applications such as visual perception and edge detection

Architecture: Neocognitron Neural Network



The lowest stage is the input layer consisting of two-dimensional array of cells.
Each cell receives input connections that lead from cells situated in a limited area on the preceding layer.

Layers of "S-cells" and "C-cells" are arranged alternately in the hierarchical network.
(In the network shown in Figure , a contrast-extracting layer is inserted between the input layer and the S-cell layer of the first stage).

S-cells (Simple cells) work as feature-extracting cells.
They resemble **simple cells** of the primary visual cortex in their response. Their input connections are variable and are modified through learning.
After having finished learning, each S-cell come to respond selectively to a particular feature presented in its receptive field.
The features extracted by S-cells are determined during the learning process.
Generally speaking, local features, such as edges or lines in particular orientations, are extracted in lower stages.
More global features, such as parts of learning patterns, are extracted in higher stages.

C-cells (Complex cells) , which resembles **complex cells** in the visual cortex, are inserted in the network to allow for positional errors in the features of the stimulus.
The input connections of C-cells, which come from S-cells of the preceding layer, are fixed and invariable.
Each C-cell receives excitatory input connections from a group of S-cells that extract the same feature, but from slightly different positions.
The C-cell responds if at least one of these S-cells yield an output.
Even if the stimulus feature shifts in position and another S-cell comes to respond instead of the first one, the same C-cell keeps responding.
Thus, the C-cell's response is less sensitive to shift in position of the input pattern.
We can also express that C-cells make a blurring operation, because the response of a layer of S-cells is spatially blurred in the response of the succeeding layer of C-cells.

Each layer of S-cells or C-cells is divided into sub-layers, called "cell-planes", according to the features to which the cells responds.
The cells in each cell-plane are arranged in a two-dimensional array.
A cell-plane is a group of cells that are arranged retinotopically and share the same set of input connections.
In other words, the connections to a cell-plane have a translational symmetry.
As a result, all the cells in a cell-plane have receptive fields of an identical characteristic, but the locations of the receptive fields differ from cell to cell.
The modification of variable connections during the learning progresses also under the restriction of shared connections.

Here are the steps involved in the recognition of handwritten characters using Neocognitron:

1. Preprocessing: The first step is to preprocess the image of the handwritten character. This involves techniques such as noise reduction, normalization, and smoothing to improve the quality of the image and make it more suitable for analysis.
2. Feature extraction: In this step, the image is analyzed to extract features that are relevant for character recognition. For example, features such as stroke direction, curvature, and line thickness can be extracted using techniques such as edge detection, contour analysis, and Hough transforms.
3. Training: Once the features have been extracted, the Neocognitron network is trained using a set of training examples. During training, the network adjusts its weights and biases to minimize the difference between its output and the desired output.
4. Testing: After training, the network is tested using a set of test examples that it has not seen before. The performance of the network is evaluated based on its ability to correctly recognize the characters in the test set.
5. Post-processing: The recognized characters may contain errors, such as misrecognitions or false positives. Post-processing techniques such as spell-checking and dictionary lookup can be used to correct these errors and improve the accuracy of the recognition results.

NET Talk: to convert English text to speech

Text to speech is also called as Neural Text to Speech.

How general Text to Speech works?

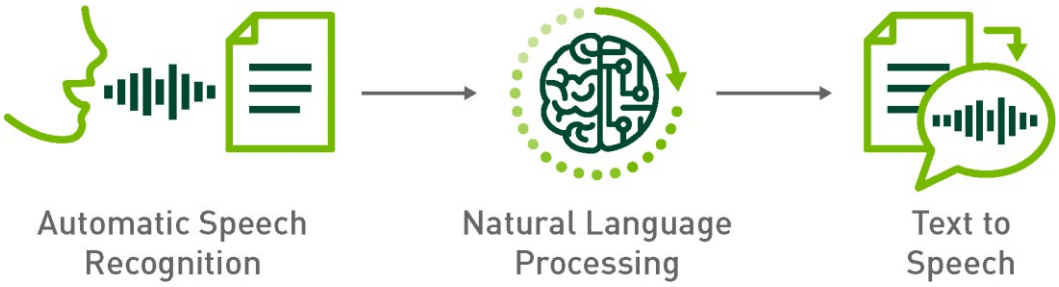
- First, the speech engines take the audio input and recognize sound waves produced by a human voice.
- This information is then translated into language data, which is called automatic speech recognition (ASR).
- After acquiring this data, it must then analyze it to understand the meaning of the words it has collected, which is called natural-language generation (NLG).
- AI has advanced to the point where it can understand human communication to the point where it
  can determine the appropriate response. AI does this by analyzing a large volume of human speech.
- After understanding the context of the text response, it produces the necessary speech sounds or phonemes.

NET Talk:

NET Talk is a neural network-based speech synthesizer that was developed in the 1980s by Terry Sejnowski and Charles Rosenberg. It is designed to convert English text into speech, and it is based on a type of neural network called a time-delay neural network (TDNN).

The NET Talk system consists of two main components: a training module and a synthesis module.

1. **During the training phase, the system is trained using a large corpus of English text and corresponding speech recordings.**
2. **The training data is used to adjust the weights and biases of the TDNN (time-delay neural network), so that it can accurately predict the acoustic features of the speech given the input text.**
3. **Once the TDNN has been trained, it can be used to synthesize speech from new input text.**
4. **The synthesis module takes a text input and converts it into a sequence of acoustic features, which are then used to generate the corresponding speech waveform.**



What is the best TTS voice, and how can you create one?

There are several areas you should be concerned about when conducting your search for the best TTS voice. You'll want to determine:

- **How accurate sounding is it, especially if you have language needs outside of English?**
- **How easy is it to use, especially if you have a large amount of content you need to translate into a TTS voice?**
- **Does it cost more to use additional voices or to use different languages?**

Recognition of consonant vowel (CV) segments

Segment: section or part of something

Combination of Consonant and Vowel creates letters in English (or any other language)

Vowel: A, E, I, O, U

In text or speech recognition, why differently recognition of consonant and vowel is required?

- Text and speech recognition systems differ in the way they recognize consonants and vowels, because consonants and vowels have different acoustic properties that affect their perceptual distinctiveness and recognition.
- Due to differences in acoustic properties, recognizing consonants and vowels requires different signal processing techniques and machine learning models.
- In automatic speech recognition systems, different feature extraction techniques and acoustic models are used for recognizing consonants and vowels, because of their different acoustic properties.

1. Consonants

- Consonants are produced by obstructing or constricting the airflow in the vocal tract, creating noise or turbulence in the sound signal.
- Consonants are generally shorter in duration and have a faster changing spectrum compared to vowels.
- Moreover, consonants are produced by different articulators in the vocal tract, such as lips, tongue, and teeth, which affects their acoustic properties.

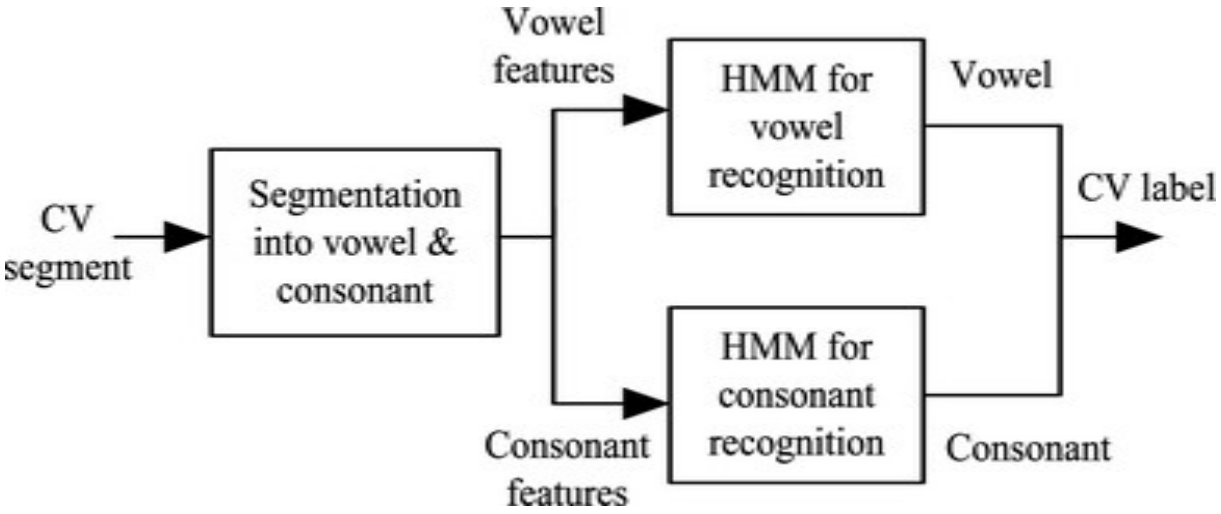
Consonants: B, C, D, F, G, J, K, L, M, N, P, Q, S, T, V, X, Z and often H, R, W, Y

2. Vowels

- Vowels are produced by maintaining a relatively stable and unobstructed airflow in the vocal tract, resulting in a more tonal or harmonic sound signal.
- Vowels are generally longer in duration and have a slower changing spectrum compared to consonants.
- Moreover, vowels are produced by the position of the tongue and the shape of the lips, which affects their acoustic properties.

Block Diagram for CV recognition system:

(HMM: Hidden Markov Model)



Aspect	HMM for Vowel Recognition	HMM for Consonant Recognition
Acoustic Model	Typically uses a single Gaussian distribution per state due to the simplicity of vowel sounds.	Uses multiple Gaussian distributions per state to model the more complex and diverse range of consonant sounds.
Training Data	Requires a large amount of training data for each vowel sound, as the acoustic features that distinguish vowels from each other are subtle.	Requires a smaller amount of training data for each consonant sound, as the acoustic features that distinguish consonants from each other are more pronounced.
State Topology	Typically uses a simple state topology with fewer states due to the relatively uniform spectral properties of vowels.	Uses a more complex state topology with more states to capture the rapid spectral changes in consonant sounds.
Recognition Accuracy	Achieves high recognition accuracy for vowel sounds due to the clear acoustic differences between vowels.	Achieves lower recognition accuracy for consonant sounds due to the more complex and diverse range of consonant sounds.
Application	Used in speech recognition tasks where the focus is on recognizing spoken words or phrases.	Used in tasks that require recognition of individual phonemes in spoken language, such as in phonetic transcription or speech therapy.

Hidden Markov Models (HMMs) are commonly used for speech recognition tasks, including vowel recognition and consonant recognition. Here's a general overview of how HMMs work for each task:

1. HMM for Vowel Recognition: The first step in using an HMM for vowel recognition is to train the model with a large set of labeled training data, consisting of audio recordings of different vowel sounds. The HMM is trained to learn the statistical properties of each vowel sound, by estimating the probability distribution of the spectral features of each vowel sound.

Once the HMM is trained, it can be used to recognize vowel sounds in new speech signals. This involves breaking the speech signal into small time frames and extracting the spectral features of each frame.
The HMM then calculates the probability that each frame of the speech signal belongs to each vowel sound. By combining the probabilities of each frame, the HMM produces a final estimate of the most likely vowel sound.

2. HMM for Consonant Recognition: HMMs for consonant recognition are similar to those for vowel recognition, but with some key differences. The main difference is that consonant sounds are more complex and diverse than vowel sounds, and require a more complex model to capture the variability of the acoustic features that distinguish different consonants.

To train an HMM for consonant recognition, a large set of labeled training data is used, consisting of audio recordings of different consonant sounds. The HMM is trained to learn the statistical properties of each consonant sound, by estimating the probability distribution of the spectral features of each consonant sound. This requires a more complex acoustic model, with multiple Gaussian distributions per state.

Once the HMM is trained, it can be used to recognize consonant sounds in new speech signals. This involves breaking the speech signal into small time frames and extracting the spectral features of each frame. The HMM then calculates the probability that each frame of the speech signal belongs to each consonant sound. By combining the probabilities of each frame, the HMM produces a final estimate of the most likely consonant sound.

Recognizing consonant-vowel (CV) segments in artificial neural networks (ANNs) can be done using multiple approaches, two popular are

1. Convolutional Neural Networks
2. combination of feature extraction and classification algorithms

recognition of CVs are mostly rely on large and diverse training dataset that includes a variety of speakers, accents, and languages to ensure the system can generalize well to new speech signals.

Texture classification and segmentation

1. Texture Classification:

- Texture classification is the task of identifying the underlying texture pattern in an image and categorizing it into one or more predefined classes.
- Texture refers to the visual properties of the surface of an object, such as roughness, smoothness, regularity, or randomness. In texture classification, a feature vector is extracted from the image, which represents the texture properties of the image.
- This feature vector is then used to train a machine learning model, such as a Support Vector Machine (SVM) or a Convolutional Neural Network (CNN), to classify the image into one of several predefined texture classes.
- Texture classification has applications in areas such as material identification, medical imaging, and image retrieval.

2. Texture Segmentation:

- Texture segmentation is the task of dividing an image into distinct regions or segments based on the underlying texture patterns in the image.
- Texture segmentation can be performed using a variety of methods, including edge detection, thresholding, clustering, and region growing.
- The goal of texture segmentation is to identify regions in the image that share similar texture properties, and to group them together into segments.
- Texture segmentation has applications in areas such as object recognition, scene analysis, and image editing.