iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** X **Month of publication:** October 2023

DOI: https://doi.org/10.22214/ijraset.2023.55256

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

Computer Vision Mouse for Tetraplegic Individuals to Access Computers Using Facial Expressions

Amogh Naik[1], Kush Parihar[2]
MIT ADT University, Loni Kalbhor

Abstract: ***Controlling the mouse by a physically challenged person is really a tough one. To find a solution for the people who cannot use the Mouse physically, we have proposed this mouse cursor control using Eye Movements. Eye gaze is an alternative way of accessing a computer using eye movements to control the mouse. For someone who fine touchscreens, mouse inaccessible, eye gaze is an alternative method to allow a user to operate their computer, using the movement of their eyes. Eye movement can be regarded as a pivotal real-time input medium for human-computer communication, which is especially important for people with physical disability. In order to improve the reliability, mobility, and usability of eye tracking technique in user-computer dialogue, a novel eye control system is proposed in this system using Webcam and without using any extra hardware. The proposed system focuses on providing a simple and convenient interactive mode by only using user's eye. The usage flow of the proposed system is designed to perfectly follow human natural habits. The proposed system describes the implementation of both iris and movement of cursor according to iris position which can be used to control the cursor on the screen using webcam and implemented using Python.***

I. INTRODUCTION

A. Deep Learning

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the back propagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input. Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-

purpose learning procedure. Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition and speech recognition, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules, analysing particle accelerator data, reconstructing brain circuits, and predicting the effects of mutations in non-coding DNA on gene expression and disease. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding, particularly topic classification, sentiment analysis, question answering and language translation. We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

B. Supervised Learning

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labelled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as 'knobs' that define the input–output function of the machine. In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labelled examples with which to train the machine. To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector. The objective function, averaged over all the training examples, can be seen as a kind of hilly landscape in the high-dimensional space of weight values. The negative gradient vector indicates the direction of steepest descent in this landscape, taking it closer to a minimum, where the output error is low on average. In practice, most practitioners use a procedure called stochastic gradient descent (SGD). This consists of showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly. The process is repeated for many small sets of examples from the training set until the average of the objective function stops decreasing. It is called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples. This simple procedure usually finds a good set of weights surprisingly quickly when compared with far more elaborate optimization techniques. After training, the performance of the system is measured on a different set of examples called a test set. This serves to test the generalization ability of the machine — its ability to produce sensible answers on new inputs that it has never seen during training.

Many of the current practical applications of machine learning use linear classifiers on top of hand-engineered features. A two-class linear classifier computes a weighted sum of the feature vector components. If the weighted sum is above a threshold, the input is classified as belonging to a particular category. Since the 1960s we have known that linear classifiers can only carve their input space into very simple regions, namely half-spaces separated by a hyperplane. But problems such as image and speech recognition require the input–output function to be insensitive to irrelevant variations of the input, such as variations in position, orientation or illumination of an object, or variations in the pitch or accent of speech, while being very sensitive to particular minute variations (for example, the difference between a white wolf and a breed of wolf-like white dog called a Samoyed). At the pixel level, images of two Samoyeds in different poses and in different environments may be very different from each other, whereas two images of a Samoyed and a wolf in the same position and on similar backgrounds may be very similar to each other. A linear classifier, or any other 'shallow' classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category. This is why shallow classifiers require a good feature extractor that solves the selectivity–invariance dilemma — one that produces representations that are selective to the aspects of the image that are important for discrimination, but that are invariant to irrelevant aspects such as the pose of the animal. To make classifiers more powerful, one can use generic non-linear features, as with kernel methods, but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples. The conventional option is to hand design good feature extractors, which requires a considerable amount of engineering skill and domain expertise. But this can all be avoided if good features can be learned automatically using a general-purpose learning procedure.

This is the key advantage of deep learning. A deep-learning architecture is a multilayer stack of simple modules, all (or most) of which are subject to learning, and many of which compute non-linear input–output mappings. Each module in the stack transforms its input to increase both the selectivity and the invariance of the representation. With multiple non-linear layers, say a depth of 5 to 20, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details — distinguishing Samoyeds from white wolves — and insensitive to large irrelevant variations such as the background, pose, lighting and surrounding objects.

C. Back Propagation to Train Multilayer Architectures

From the earliest days of pattern recognition, the aim of researchers has been to replace hand-engineered features with trainable multilayer networks, but despite its simplicity, the solution was not widely understood until the mid 1980s. As it turns out, multilayer architectures can be trained by simple stochastic gradient descent. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure. The idea that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s. The backpropagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing more than a practical application of the chain rule for derivatives. The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module). The back propagation equation can be applied repeatedly to propagate gradients through all modules, starting from the output at the top (where the network produces its prediction) all the way to the bottom (where the external input is fed). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each module. Many applications of deep learning use feedforward neural network architectures (Fig. 1), which learn to map a fixed-size input (for example, an image) to a fixed-size output (for example, a probability for each of several categories). To go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear function. At present, the most popular non-linear function is the rectified linear unit (ReLU), which is simply the half-wave rectifier $f(z)= \max(z, 0)$. In past decades, neural nets used smoother non-linearities, such as $\tanh(z)$ or $1/(1+\exp(-z))$, but the ReLU typically learns much faster in networks with many layers, allowing training of a deep supervised network without unsupervised pre-training. Units that are not in the input or output layer are conventionally called hidden units. The hidden layers can be seen as distorting the input in a non-linear way so that categories become linearly separable by the last layer (Fig. 1). In the late 1990s, neural nets and backpropagation were largely forsaken by the machine-learning community and ignored by the computer-vision and speech-recognition communities. It was widely thought that learning useful, multistage, feature extractors with little prior knowledge was infeasible. In particular, it was commonly thought that simple gradient descent would get trapped in poor local minima — weight configurations for which no small change would reduce the average error. In practice, poor local minima are rarely a problem with large networks. Regardless of the initial conditions, the system nearly always reaches solutions of very similar quality. Recent theoretical and empirical results strongly suggest that local minima are not a serious issue in general. Instead, the landscape is packed with a combinatorially large number of saddle points where the gradient is zero, and the surface curves up in most dimensions and curves down in the remainder. The analysis seems to show that saddle points with only a few downward curving directions are present in very large numbers, but almost all of them have very similar values of the objective function. Hence, it does not much matter which of these saddle points the algorithm gets stuck at. Interest in deep feed forward networks was revived around 2006 by a group of researchers brought together by the Canadian Institute for Advanced Research (CIFAR). The researchers introduced unsupervised learning procedures that could create layers of feature detectors without requiring labelled data. The objective in learning each layer of feature detectors was to be able to reconstruct or model the activities of feature detectors (or raw inputs) in the layer below. By ‘pre-training’ several layers of progressively more complex feature detectors using this reconstruction objective, the weights of a deep network could be initialized to sensible values. A final layer of output units could then be added to the top of the network and the whole deep system could be fine-tuned using standard backpropagation. This worked remarkably well for recognizing handwritten digits or for detecting pedestrians, especially when the amount of labelled data was very limited. The first major application of this pre-training approach was in speech recognition, and it was made possible by the advent of fast graphics processing units (GPUs) that were convenient to program and allowed researchers to train networks 10 or 20 times faster. In 2009, the approach was used to map short temporal windows of coefficients extracted from a sound wave to a set of probabilities for the various fragments of speech that might be represented by the frame in the centre of the window. It achieved record-breaking results on a standard speech recognition benchmark that used a small vocabulary38 and was quickly developed to give record-breaking results on a large vocabulary task39.

By 2012, versions of the deep net from 2009 were being developed by many of the major speech groups6 and were already being deployed in Android phones. For smaller data sets, unsupervised pre-training helps to prevent overfitting, leading to significantly better generalization when the number of labelled examples is small, or in a transfer setting where we have lots of examples for some 'source' tasks but very few for some 'target' tasks. Once deep learning had been rehabilitated, it turned out that the pre-training stage was only needed for small data sets. There was, however, one particular type of deep, feedforward network that was much easier to train and generalized much better than networks with full connectivity between adjacent layers. This was the convolutional neural network (ConvNet). It achieved many practical successes during the period when neural networks were out of favour and it has recently been widely adopted by the computer vision community.

D. Convolutional Neural Networks

ConvNets are designed to process data that come in the form of multiple arrays, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers. The architecture of a typical ConvNet is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a ReLU. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The reason for this architecture is twofold. First, in array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected. Second, the local statistics of images and other signals are invariant to location. In other words, if a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array. Mathematically, the filtering operation performed by a feature map is a discrete convolution, hence the name. Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighbouring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating an invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked, followed by more convolutional and fully-connected layers.

Backpropagating gradients through a ConvNet is as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained. Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones. In images, local combinations of edges form motifs, motifs assemble into parts, and parts form objects. Similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences.

The pooling allows representations to vary very little when elements in the previous layer vary in position and appearance. The convolutional and pooling layers in ConvNets are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience, and the overall architecture is reminiscent of the LGN–V1–V2–V4–IT hierarchy in the visual cortex ventral pathway. When ConvNet models and monkeys are shown the same picture, the activations of high-level units in the ConvNet explains half of the variance of random sets of 160 neurons in the monkey's inferotemporal cortex. ConvNets have their roots in the neocognitron46, the architecture of which was somewhat similar, but did not have an end-to-end supervised-learning algorithm such as backpropagation. A primitive 1D ConvNet called a time-delay neural net was used for the recognition of phonemes and simple words. There have been numerous applications of convolutional networks going back to the early 1990s, starting with time-delay neural networks for speech recognition and document reading. The document reading system used a ConvNet trained jointly with a probabilistic model that implemented language constraints. By the late 1990s this system was reading over 10% of all the cheques in the United States. A number of ConvNet-based optical character recognition and handwriting recognition systems were later deployed by Microsoft49. ConvNets were also experimented with in the early 1990s for object detection in natural images, including faces and hands, and for face recognition.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

E. Image Understanding with Deep Convolutional Networks

Since the early 2000s, ConvNets have been applied with great success to the detection, segmentation and recognition of objects and regions in images. These were all tasks in which labelled data was relatively abundant, such as traffic sign recognition, the segmentation of biological images particularly for connectomics, and the detection of faces, text, pedestrians and human bodies in natural images. A major recent practical success of ConvNets is face recognition. Importantly, images can be labelled at the pixel level, which will have applications in technology, including autonomous mobile robots and self-driving cars. Companies such as Mobileye and NVIDIA are using such ConvNet-based methods in their upcoming vision systems for cars. Other applications gaining importance involve natural language understanding and speech recognition . Despite these successes, ConvNets were largely forsaken by the mainstream computer-vision and machine-learning communities until the ImageNet competition in 2012. When deep convolutional networks were applied to a data set of about a million images from the web that contained 1,000 different classes, they achieved spectacular results, almost halving the error rates of the best competing approaches. This success came from the efficient use of GPUs, ReLUs, a new regularization technique called dropout, and techniques to generate more training examples by deforming the existing ones. This success has brought about a revolution in computer vision; ConvNets are now the dominant approach for almost all recognition and detection tasks and approach human performance on some tasks. A recent stunning demonstration combines ConvNets and recurrent net modules for the generation of image captions. Recent ConvNet architectures have 10 to 20 layers of ReLUs, hundreds of millions of weights, and billions of connections between units. Whereas training such large networks could have taken weeks only two years ago, progress in hardware, software and algorithm parallelization have reduced training times to a few hours. The performance of ConvNet-based vision systems has caused most major technology companies, including Google, Facebook, Microsoft, IBM, Yahoo!, Twitter and Adobe, as well as a quickly growing number of start-ups to initiate research and development projects and to deploy ConvNet-based image understanding products and services. ConvNets are easily amenable to efficient hardware implementations in chips or field-programmable gate arrays. A number of companies such as NVIDIA, Mobileye, Intel, Qualcomm and Samsung are developing ConvNet chips to enable real-time vision applications in smartphones, cameras, robots and self-driving cars. Distributed representations and language processing Deep-learning theory shows that deep nets have two different exponential advantages over classic learning algorithms that do not use distributed representations. Both of these advantages arise from the power of composition and depend on the underlying data-generating distribution having an appropriate componential structure. First, learning distributed representations enable generalization to new combinations of the values of learned features beyond those seen during training (for example, 2n combinations are possible with n binary features). Second, composing layers of representation in a deep net brings the potential for another exponential advantage (exponential in the depth). The hidden layers of a multilayer neural network learn to represent the network's inputs in a way that makes it easy to predict the target outputs. This is nicely demonstrated by training a multilayer neural network to predict the next word in a sequence from a local context of earlier words. Each word in the context is presented to the network as a one-of-N vector, that is, one component has a value of 1 and the rest are 0. In the first layer, each word creates a different pattern of activations, or word vectors. In a language model, the other layers of the network learn to convert the input word vectors into an output word vector for the predicted next word, which can be used to predict the probability for any word in the vocabulary to appear as the next word. The network learns word vectors that contain many active components each of which can be interpreted as a separate feature of the word, as was first demonstrated in the context of learning distributed representations for symbols. These semantic features were not explicitly present in the input. They were discovered by the learning procedure as a good way of factorizing the structured relationships between the input and output symbols into multiple 'micro-rules'. Learning word vectors turned out to also work very well when the word sequences come from a large corpus of real text and the individual micro-rules are unreliable. When trained to predict the next word in a news story, for example, the learned word vectors for Tuesday and Wednesday are very similar, as are the word vectors for Sweden and Norway. Such representations are called distributed representations because their elements (the features) are not mutually exclusive and their many configurations correspond to the variations seen in the observed data. These word vectors are composed of learned features that were not determined ahead of time by experts, but automatically discovered by the neural network. Vector representations of words learned from text are now very widely used in natural language applications. The issue of representation lies at the heart of the debate between the logic-inspired and the neural-network-inspired paradigms for cognition. In the logic-inspired paradigm, an instance of a symbol is something for which the only property is that it is either identical or non-identical to other symbol instances. It has no internal structure that is relevant to its use; and to reason with symbols, they must be bound to the variables in judiciously chosen rules of inference.

By contrast, neural networks just use big activity vectors, big weight matrices and scalar non-linearities to perform the type of fast 'intuitive' inference that underpins effortless commonsense reasoning. Before the introduction of neural language models, the standard approach to statistical modelling of language did not exploit distributed representations: it was based on counting frequencies of occurrences of short symbol sequences of length up to N (called N-grams). The number of possible N-grams is on the order of VN, where V is the vocabulary size, so taking into account a context of more than a handful of words would require very large training corpora. N-grams treat each word as an atomic unit, so they cannot generalize across semantically related sequences of words, whereas neural language models can because they associate each word with a vector of real valued features, and semantically related words end up close to each other in that vector space.

F. Recurrent Neural Networks

When backpropagation was first introduced, its most exciting use was for training recurrent neural networks (RNNs). For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs. RNNs process an input sequence one element at a time, maintaining in their hidden units a 'state vector' that implicitly contains information about the history of all the past elements of the sequence. When we consider the outputs of the hidden units at different discrete time steps as if they were the outputs of different neurons in a deep multilayer network, it becomes clear how we can apply backpropagation to train RNNs. RNNs are very powerful dynamic systems, but training them has proved to be problematic because the backpropagated gradients either grow or shrink at each time step, so over many time steps they typically explode or vanish. Thanks to advances in their architecture and ways of training them, RNNs have been found to be very good at predicting the next character in the text83 or the next word in a sequence, but they can also be used for more complex tasks. For example, after reading an English sentence one word at a time, an English 'encoder' network can be trained so that the final state vector of its hidden units is a good representation of the thought expressed by the sentence. This thought vector can then be used as the initial hidden state of (or as extra input to) a jointly trained French 'decoder' network, which outputs a probability distribution for the first word of the French translation. If a particular first word is chosen from this distribution and provided as input to the decoder network it will then output a probability distribution for the second word of the translation and so on until a full stop is chosen. Overall, this process generates sequences of French words according to a probability distribution that depends on the English sentence. This rather naive way of performing machine translation has quickly become competitive with the state-of-the-art, and this raises serious doubts about whether understanding a sentence requires anything like the internal symbolic expressions that are manipulated by using inference rules. It is more compatible with the view that everyday reasoning involves many simultaneous analogies that each contribute plausibility to a conclusion. Instead of translating the meaning of a French sentence into an English sentence, one can learn to 'translate' the meaning of an image into an English sentence. The encoder here is a deep ConvNet that converts the pixels into an activity vector in its last hidden layer. The decoder is an RNN similar to the ones used for machine translation and neural language modelling. There has been a surge of interest in such systems recently. RNNs, once unfolded in time (Fig. 5), can be seen as very deep feedforward networks in which all the layers share the same weights. Although their main purpose is to learn long-term dependencies, theoretical and empirical evidence shows that it is difficult to learn to store information for very long. To correct for that, one idea is to augment the network with an explicit memory. The first proposal of this kind is the long short-term memory (LSTM) networks that use special hidden units, the natural behaviour of which is to remember inputs for a long time. A special unit called the memory cell acts like an accumulator or a gated leaky neuron: it has a connection to itself at the next time step that has a weight of one, so it copies its own real-valued state and accumulates the external signal, but this self-connection is multiplicatively gated by another unit that learns to decide when to clear the content of the memory. LSTM networks have subsequently proved to be more effective than conventional RNNs, especially when they have several layers for each time step, enabling an entire speech recognition system that goes all the way from acoustics to the sequence of characters in the transcription. LSTM networks or related forms of gated units are also currently used for the encoder and decoder networks that perform so well at machine translation. Over the past year, several authors have made different proposals to augment RNNs with a memory module. Proposals include the Neural Turing Machine in which the network is augmented by a 'tape-like' memory that the RNN can choose to read from or write to88, and memory networks, in which a regular network is augmented by a kind of associative memory89. Memory networks have yielded excellent performance on standard question-answering benchmarks. The memory is used to remember the story about which the network is later asked to answer questions. Beyond simple memorization, neural Turing machines and memory networks are being used for tasks that would normally require reasoning and symbol manipulation. Neural Turing machines can be taught 'algorithms'. Among other things, they can learn to output a sorted list of symbols when their input consists of an unsorted sequence in which each symbol is accompanied by a real value that indicates its priority in the list.

Memory networks can be trained to keep track of the state of the world in a setting similar to a text adventure game and after reading a story, they can answer questions that require complex inference. In one test example, the network is shown a 15-sentence version of the The Lord of the Rings and correctly answers questions such as "where is Frodo now?".

G. The Future of Deep Learning

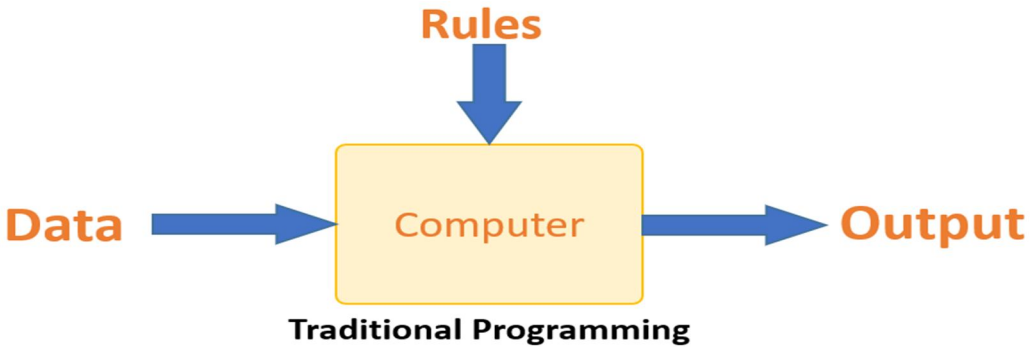
Unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely supervised learning. Although we have not focused on it in this Review, we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object. Human vision is an active process that sequentially samples the optic array in an intelligent, task-specific way using a small, high-resolution fovea with a large, low-resolution surround. We expect much of the future progress in vision to come from systems that are trained end-to-end and combine ConvNets with RNNs that use reinforcement learning to decide where to look. Systems combining deep learning and reinforcement learning are in their infancy, but they already outperform passive vision systems at classification tasks and produce impressive results in learning to play many different video games. Natural language understanding is another area in which deep learning is poised to make a large impact over the next few years. We expect systems that use RNNs to understand sentences or whole documents will become much better when they learn strategies for selectively attending to one part at a time. Ultimately, major progress in artificial intelligence will come about through systems that combine representation learning with complex reasoning. Although deep learning and simple reasoning have been used for speech and handwriting recognition for a long time, new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors

II. WHAT IS MACHINE LEARNING?

Machine Learning is a system of computer algorithms that can learn from example through self-improvement without being explicitly coded by a programmer. Machine learning is a part of artificial Intelligence which combines data with statistical tools to predict an output which can be used to make actionable insights. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input and uses an algorithm to formulate answers. A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation. Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

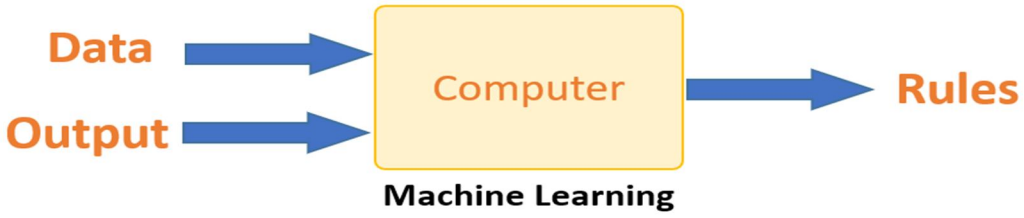
A. Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain. Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.



Traditional Programming

Machine learning is supposed to overcome this issue. The machine learns how the input and output data are correlated and it writes a rule. The programmers do not need to write new rules each time there is new data. The algorithms adapt in response to new data and experiences to improve efficacy over time.



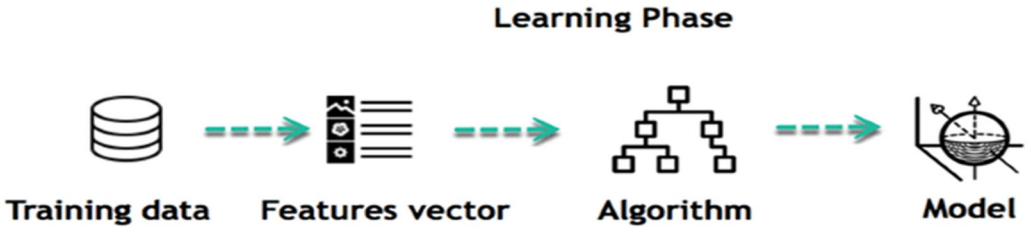
Machine Learning

B. How Does Machine Learning Work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

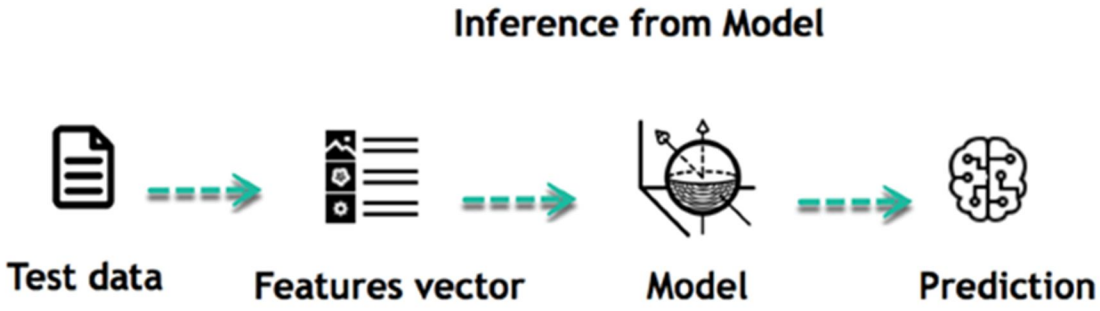
The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

C. Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

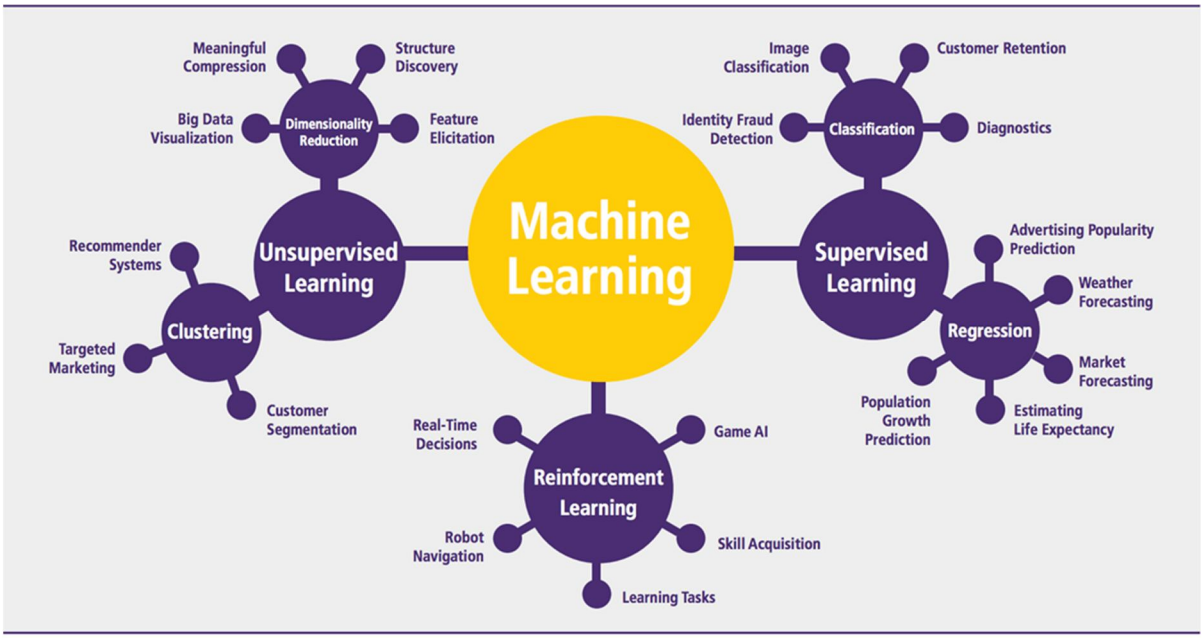


The life of Machine Learning programs is straightforward and can be summarized in the following points:

1) Define a question
2) Collect data
3) Visualize data
4) Train algorithm
5) Test the Algorithm
6) Collect feedback
7) Refine the algorithm
8) Loop 4-7 until the results are satisfying
9) Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

D. Machine Learning Algorithms and where they are used?



Machine learning Algorithms

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

E. Supervised Learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

F. Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected).

When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.
The label can be of two or more classes. The above Machine learning example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

G. Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

Algorithm Name	Description	Type
Linear regression	Finds a way to correlate each feature to the output to help predict future values.	Regression
Logistic regression	Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g only black or white) rather than continuous (e.g., an infinite list of potential colors)	Classification
Decision tree	Highly interpretable classification or regression model that splits data-feature values into branches decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a fin decision output is made	Regression Classification
Naive Bayes	The Bayesian method is a classification method that makes use of the Bayesian theorem. The theore updates the prior knowledge of an event with the independent probability of each feature that can affe the event.	Regression Classification
Support vector machine	Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds hyperplane that optimally divided the classes. It is best used with a non-linear solver.	Regression (not ver common) Classificatio
Random forest	The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generat many times simple decision trees and uses the 'majority vote' method to decide on which label to retur For the classification task, the final prediction will be the one with the most vote; while for the regressi task, the average prediction of all the trees is the final prediction.	Regression Classification
AdaBoost	Classification or regression technique that uses a multitude of models to come up with a decision b weighs them based on their accuracy in predicting the outcome	Regression Classification
Gradient-boosting trees	Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the err committed by the previous trees and tries to correct it.	Regression Classification

H. Unsupervised Learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)
You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

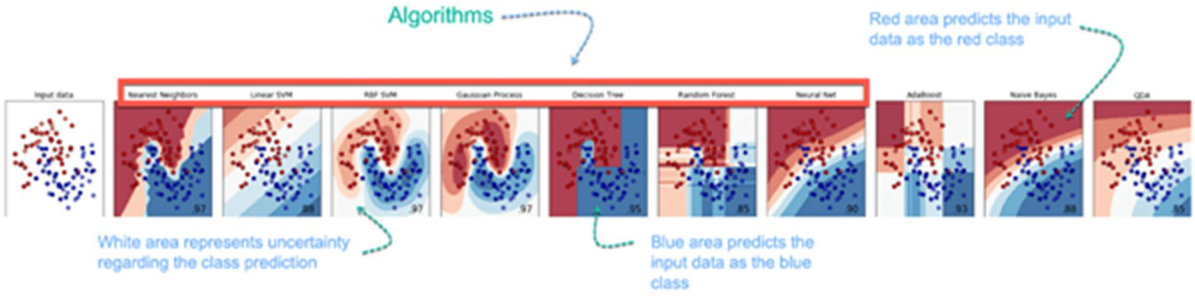
Algorithm	Description	Type
K-means clustering	Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans)	Clustering
Gaussian mixture model	A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters)	Clustering
Hierarchical clustering	Splits clusters along a hierarchical tree to form a classification system. Can be used for Cluster loyalty-card customer	Clustering
Recommender system	Help to define the relevant data for making a recommendation.	Clustering
PCA/T-SNE	Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances.	Dimension Reduction

III. HOW TO CHOOSE MACHINE LEARNING ALGORITHM

A. *Machine Learning (ML) Algorithm*

There are plenty of machine learning algorithms. The choice of the algorithm is based on the objective.

In the Machine learning example below, the task is to predict the type of flower among the three varieties. The predictions are based on the length and the width of the petal. The picture depicts the results of ten different algorithms. The picture on the top left is the dataset. The data is classified into three categories: red, light blue and dark blue. There are some groupings. For instance, from the second image, everything in the upper left belongs to the red category, in the middle part, there is a mixture of uncertainty and light blue while the bottom corresponds to the dark category. The other images show different algorithms and how they try to classified the data.



B. *Challenges and Limitations of Machine Learning*

The primary challenge of machine learning is the lack of data or the diversity in the dataset. A machine cannot learn if there is no data available. Besides, a dataset with a lack of diversity gives the machine a hard time. A machine needs to have heterogeneity to learn meaningful insight. It is rare that an algorithm can extract information when there are no or few variations. It is recommended to have at least 20 observations per group to help the machine learn. This constraint leads to poor evaluation and prediction.

C. *Application of Machine Learning*

1) *Augmentation:* Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.
2) *Automation:* Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.
3) *Finance Industry:* Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.
4) *Government Organization:* The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.
5) *Healthcare Industry:* Healthcare was one of the first industry to use machine learning with image detection.
6) *Marketing:* Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

D. *Example of Application of Machine Learning in Supply Chain*

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

E. Example of Machine Learning Google Car

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

IV. WHY IS MACHINE LEARNING IMPORTANT?

Machine learning is the best tool so far to analyze, understand and identify a pattern in the data. One of the main ideas behind machine learning is that the computer can be trained to automate tasks that would be exhaustive or impossible for a human being. The clear breach from the traditional analysis is that machine learning can take decisions with minimal human intervention.

Take the following example for this ML tutorial; a retail agent can estimate the price of a house based on his own experience and his knowledge of the market.

A machine can be trained to translate the knowledge of an expert into features. The features are all the characteristics of a house, neighborhood, economic environment, etc. that make the price difference. For the expert, it took him probably some years to master the art of estimate the price of a house. His expertise is getting better and better after each sale.

For the machine, it takes millions of data, (i.e., example) to master this art. At the very beginning of its learning, the machine makes a mistake, somehow like the junior salesman. Once the machine sees all the example, it got enough knowledge to make its estimation. At the same time, with incredible accuracy. The machine is also able to adjust its mistake accordingly.

Most of the big company have understood the value of machine learning and holding data. McKinsey have estimated that the value of analytics ranges from $9.5 trillion to $15.4 trillion while $5 to 7 trillion can be attributed to the most advanced AI techniques.

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

A. Overview

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.

B. Machine Learning Approaches

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

1) *Supervised Learning:* The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
2) *Unsupervised Learning:* No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

3) *Reinforcement Learning:* A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches have been developed which don't fit neatly into this three-fold categorisation, and sometimes more than one is used by the same machine learning system. For example topic modeling, dimensionality reduction or meta learning.
As of 2020, deep learning has become the dominant approach for much ongoing work in the field of machine learning.

C. History and Relationships to other Fields

The term machine learning was coined in 1959 by Arthur Samuel, an American IBMer and pioneer in the field of computer gaming and artificial intelligence. A representative book of the machine learning research during the 1960s was the Nilsson's book on Learning Machines, dealing mostly with machine learning for pattern classification. Interest related to pattern recognition continued into the 1970s, as described by Duda and Hart in 1973. In 1981 a report was given on using teaching strategies so that a neural network learns to recognize 40 characters (26 letters, 10 digits, and 4 special symbols) from a computer terminal.
Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?".
Modern day machine learning has two objectives, one is to classify data based on models which have been developed, the other purpose is to make predictions for future outcomes based on these models. A hypothetical algorithm specific to classifying data may use computer vision of moles coupled with supervised learning in order to train it to classify the cancerous moles. Where as, a machine learning algorithm for stock trading may inform the trader of future potential predictions.

D. Artificial intelligence

Machine Learning as subfield of AI
Part of Machine Learning as subfield of AI or part of AI as subfield of Machine Learning
As a scientific endeavor, machine learning grew out of the quest for artificial intelligence. In the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what was then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.
However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation. By 1980, expert systems had come to dominate AI, and statistics was out of favor. Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval. Neural networks research had been abandoned by AI and computer science around the same time.
This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.
Machine learning (ML), reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory.
As of 2020, many sources continue to assert that machine learning remains a subfield of AI. The main disagreement is whether all of ML is part of AI, as this would mean that anyone using ML could claim they are using AI. Others have the view that not all of ML is part of AI where only an 'intelligent' subset of ML is part of AI.
The question to what is the difference between ML and AI is answered by Judea Pearl in The Book of Why. Accordingly ML learns and predicts based on passive observations, whereas AI implies an agent interacting with the environment to learn and take actions that maximize its chance of successfully achieving its goals.

E. Data Mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

F. Optimization

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples).

G. Generalization

The difference between optimization and machine learning arises from the goal of generalization: while optimization algorithms can minimize the loss on a training set, machine learning is concerned with minimizing the loss on unseen samples. Characterizing the generalization of various learning algorithms is an active topic of current research, especially for deep learning algorithms.

H. Statistics

Machine learning and statistics are closely related fields in terms of methods, but distinct in their principal goal: statistics draws population inferences from a sample, while machine learning finds generalizable predictive patterns. According to Michael I. Jordan, the ideas of machine learning, from methodological principles to theoretical tools, have had a long pre-history in statistics. He also suggested the term data science as a placeholder to call the overall field.

Leo Breiman distinguished two statistical modeling paradigms: data model and algorithmic model, wherein "algorithmic model" means more or less the machine learning algorithms like Random forest.

Some statisticians have adopted methods from machine learning, leading to a combined field that they call statistical learning.

I. Theory

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has under fitted the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer.

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

V. APPROACHES

A. Types of Learning Algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

B. Supervised Learning

A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white. Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task. Types of supervised learning algorithms include active learning, classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. As an example, for a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email.

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

C. Unsupervised Learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features. Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

D. Semi-supervised Learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

E. Reinforcement Learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov decision process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

F. Self-learning

Self-learning as a machine learning paradigm was introduced in 1982 along with a neural network capable of self-learning named crossbar adaptive array (CAA). It is a learning with no external rewards and no external teacher advice. The CAA self-learning algorithm computes, in a crossbar fashion, both decisions about actions and emotions (feelings) about consequence situations. The system is driven by the interaction between cognition and emotion. The self-learning algorithm updates a memory matrix W =||w(a,s)|| such that in each iteration executes the following machine learning routine:

In situation s perform an action a;

Receive consequence situation s';

Compute emotion of being in consequence situation v(s');

Update crossbar memory w'(a,s) = w(a,s) + v(s').

It is a system with only one input, situation s, and only one output, action (or behavior) a. There is neither a separate reinforcement input nor an advice input from the environment. The backpropagated value (secondary reinforcement) is the emotion toward the consequence situation. The CAA exists in two environments, one is the behavioral environment where it behaves, and the other is the genetic environment, wherefrom it initially and only once receives initial emotions about situations to be encountered in the behavioral environment. After receiving the genome (species) vector from the genetic environment, the CAA learns a goal-seeking behavior, in an environment that contains both desirable and undesirable situations.

G. Feature Learning

Several learning algorithms aim at discovering better representations of the inputs provided during training. Classic examples include principal components analysis and cluster analysis. Feature learning algorithms, also called representation learning algorithms, often attempt to preserve the information in their input but also transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions. This technique allows reconstruction of the inputs coming from the unknown data-generating distribution, while not being necessarily faithful to configurations that are implausible under that distribution. This replaces manual feature engineering, and allows a machine to both learn the features and use them to perform a specific task.

Feature learning can be either supervised or unsupervised. In supervised feature learning, features are learned using labeled input data. Examples include artificial neural networks, multilayer perceptrons, and supervised dictionary learning. In unsupervised feature learning, features are learned with unlabeled input data. Examples include dictionary learning, independent component analysis, autoencoders, matrix factorization and various forms of clustering.

Manifold learning algorithms attempt to do so under the constraint that the learned representation is low-dimensional. Sparse coding algorithms attempt to do so under the constraint that the learned representation is sparse, meaning that the mathematical model has many zeros. Multilinear subspace learning algorithms aim to learn low-dimensional representations directly from tensor representations for multidimensional data, without reshaping them into higher-dimensional vectors. Deep learning algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or generating) lower-level features. It has been argued that an intelligent machine is one that learns a representation that disentangles the underlying factors of variation that explain the observed data.

Feature learning is motivated by the fact that machine learning tasks such as classification often require input that is mathematically and computationally convenient to process. However, real-world data such as images, video, and sensory data has not yielded to attempts to algorithmically define specific features. An alternative is to discover such features or representations thorough examination, without relying on explicit algorithms.

H. Sparse Dictionary Learning

Sparse dictionary learning is a feature learning method where a training example is represented as a linear combination of basis functions, and is assumed to be a sparse matrix. The method is strongly NP-hard and difficult to solve approximately. A popular heuristic method for sparse dictionary learning is the K-SVD algorithm. Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine the class to which a previously unseen training example belongs. For a dictionary where each class has already been built, a new training example is associated with the class that is best sparsely represented by the corresponding dictionary. Sparse dictionary learning has also been applied in image de-noising. The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

VI. ANOMALY DETECTION

In data mining, anomaly detection, also known as outlier detection, is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. Typically, the anomalous items represent an issue such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are referred to as outliers, novelties, noise, deviations and exceptions.

In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts of inactivity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular, unsupervised algorithms) will fail on such data unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro-clusters formed by these patterns.

Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal, by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherently unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set and then test the likelihood of a test instance to be generated by the model.

A. Robot Learning

In developmental robotics, robot learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies and imitation.

B. Association Rules

Association rule learning is a rule-based machine learning method for discovering relationships between variables in large databases. It is intended to identify strong rules discovered in databases using some measure of "interestingness".

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply knowledge. The defining characteristic of a rule-based machine learning algorithm is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learning algorithms that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

Based on the concept of strong rules, Rakesh Agrawal, Tomasz Imieliński and Arun Swami introduced association rules for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets. For example, the rule {\displaystyle \{\mathrm {onions,potatoes} \}\Rightarrow \{\mathrm {burger} \}}\{{\mathrm {onions,potatoes}}\}\Rightarrow \{{\mathrm {burger}}\} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as promotional pricing or product placements. In addition to market basket analysis, association rules are employed today in application areas including Web usage mining, intrusion detection, continuous production, and bioinformatics. In contrast with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

Learning classifier systems (LCS) are a family of rule-based machine learning algorithms that combine a discovery component, typically a genetic algorithm, with a learning component, performing either supervised learning, reinforcement learning, or unsupervised learning. They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a piecewise manner in order to make predictions.

Inductive logic programming (ILP) is an approach to rule-learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming language for representing hypotheses (and not only logic programming), such as functional programs.

Inductive logic programming is particularly useful in bioinformatics and natural language processing. Gordon Plotkin and Ehud Shapiro laid the initial theoretical foundation for inductive machine learning in a logical setting. Shapiro built their first implementation (Model Inference System) in 1981: a Prolog program that inductively inferred logic programs from positive and negative examples. The term inductive here refers to philosophical induction, suggesting a theory to explain observed facts, rather than mathematical induction, proving a property for all members of a well-ordered set.

C. Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

D. Artificial Neural Networks

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another. Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

E. Decision Trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.

F. Support Vector Machines

Support vector machines (SVMs), also known as support vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

G. Regression Analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization (mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel, logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

H. Bayesian Networks

A simple Bayesian network. Rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

I. Genetic Algorithms

A genetic algorithm (GA) is a search algorithm and heuristic technique that mimics the process of natural selection, using methods such as mutation and crossover to generate new genotypes in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

J. Training Models

Usually, machine learning models require a lot of data in order for them to perform well. Usually, when training a machine learning model, one needs to collect a large, representative sample of data from a training set. Data from the training set can be as varied as a corpus of text, a collection of images, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from biased data can result in skewed or undesired predictions. Algorithmic bias is a potential result from data not fully prepared for training.

K. Federated Learning

Federated learning is an adapted form of distributed artificial intelligence to training machine learning models that decentralizes the training process, allowing for users' privacy to be maintained by not needing to send their data to a centralized server. This also increases efficiency by decentralizing the training process to many devices. For example, Gboard uses federated machine learning to train search query prediction models on users' mobile phones without having to send individual searches back to Google

VII. LITERATURE SURVEY

1) Assessing drivers' visual-motor coordination using eye tracking, GNSS and GIS: a spatial turn in driving psychology

AUTHORS: Q. Sun, J. Xia, N. Nadarajah, T. Falkmer, J. Foster, and H.Lee

Vehicle-driving in real traffic can be considered as a human-machine system involving not only the attribute of the vehicle movement but also the human visual perception, cognition and motion of the driver. The study of driving behaviours, therefore, would integrate information related to driver psychology, vehicle dynamics and road information in order to tackle research questions concerning driving safety. This paper describes a conceptual framework and an integrated GIS data model of a visual-motor coordination model (VMCM) to investigate drivers' driving behaviour via the combination of vision tracking and vehicle positioning. The eye tracker recorded eye fixations and duration on video images to exhibit the driver's visual search pattern and the traffic scenes. Real-time kinematic (RTK) post-processing of multi-GNSS (global navigation satellite system) tracking generated the vehicle movement trajectory at centimeter-level accuracy, which encompasses precise lateral positioning and speed control parameters of driving behaviours. The eye fixation data were then geocoded and linked to the vehicle movement trajectory to

represent the VMCM on the GIS platform. An implementation prototype of the framework and the VMCM for a study of older drivers is presented in this paper. The spatial-temporal visualisation and statistical analysis based on the VMCM data-set allow for a greater insight into the inherent variability of older drivers' visual search and motor behaviours. The research framework has demonstrated a discriminant and ecologically valid approach in driving behaviour assessment, which can also be used in studies for other cohort populations with modified driving scenarios or experiment designs.

2) Investigation of the use of eye tracking to examine tourism advertising effectiveness

AUTHORS: N. Scott, C. Green, and S. Fairley

Previous studies of printed marketing stimuli have used self-report measures to determine the relative preference for one advertisement among several different versions. This study uses TobiiTM eye-tracking hardware and software along with self-report measures to compare the relative effectiveness of two versions of a tourism magazine advertisement. Data were collected from 25 respondents in a laboratory-based study. Analysis of data shows significant differences between the two advertisements tested with agreement between the eye-tracking and self-report results. These results indicate that eye-tracking methods are useful for analysis of tourist advertising.

3) Estimating 3-D point-of-regard in a real environment using a head-mounted eye-tracking system

AUTHORS: K. Takemura, K. Takahashi, J. Takamatsu, and T. Ogasawara

Unlike conventional portable eye-tracking methods that estimate the position of the mounted camera using 2-D image coordinates, the techniques that are proposed here present richer information about person's gaze when moving over a wide area. They also include visualizing scanpaths when the user with a head-mounted device makes natural head movements. We employ a Visual SLAM technique to estimate the head pose and extract environmental information. When the person's head moves, the proposed method obtains a 3-D point-of-regard. Furthermore, scanpaths can be appropriately overlaid on image sequences to support quantitative analysis. Additionally, a 3-D environment is employed to detect objects of focus and to visualize an attention map.

4) Eye Tracking in human computer interaction and usability research: ready to deliver the promises

AUTHORS: R. J. K. Jacob and K. S. Karn

The objective of the tutorial is to give an overview on how eye tracking is currently used and how it can be used as a method in human computer interaction research and especially in usability research. An eye tracking system records how the eyes move while a subject is completing a task for example on a web site. By analyzing these eye movements we are able to gain an objective insight into the behavior of that person.

5) Low cost eye tracking: the current panorama

AUTHORS: O. Ferhat and F. Vilarino

Despite the availability of accurate, commercial gaze tracker devices working with infrared (IR) technology, visible light gaze tracking constitutes an interesting alternative by allowing scalability and removing hardware requirements. Over the last years, this field has seen examples of research showing performance comparable to the IR alternatives. In this work, we survey the previous work on remote, visible light gaze trackers and analyze the explored techniques from various perspectives such as calibration strategies, head pose invariance, and gaze estimation techniques. We also provide information on related aspects of research such as public datasets to test against, open source projects to build upon, and gaze tracking services to directly use in applications. With all this information, we aim to provide the contemporary and future researchers with a map detailing previously explored ideas and the required tools.

VIII. PROBLEM DESCRIPTION AND OVERVIEW

The proposed AI virtual mouse system can be used to overcome problems in the real world such as situations where there is no space to use a physical mouse and also for the persons who have problems in their hands and are not able to control a physical mouse. Also, amidst of the COVID-19 situation, it is not safe to use the devices by touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to overcome these problems since hand gesture and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera.

A. Input Design And Output Design

1) Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

2) Objectives

a) Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

b) It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

c) When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

3) Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

a) Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

b) Select methods for presenting information.

c) Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

IX. IMPLEMENTATION

A. Modules

1) Importing the necessary libraries

2) Open cv

3) Eye detection Using Dlib

4) How It Works

5) Eye-Aspect-Ratio (EAR)

6) Mouth-Aspect-Ratio (MAR)

7) Prebuilt Model Details

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

X. MODULES DESCSRIPTION:

A. *Importing the Necessary Libraries*

We will be using Python language for this. First we will import the necessary libraries such as open cv and dlib for building the main model

B. *Open CV*

Open CV (Open Source Computer Vision Library) is an open library of python that is mainly aimed at real-time computer-vision. It is also available in C++ and Java. It is an open source machine learning software library. It makes use of Numpy, which is a python library that is used for implementing multi-dimensional arrays and matrices along with high-level mathematical operations on these arrays. OpenCV is mainly used to capture data from a live video hence it is mainly focuses on image processing and video capture. In this paper OpenCV is focused mainly on video capture. OpenCV is also used for applications such as face detection, OCR, Vision-guided robotics surgery, 3D human organ reconstruction, QR code Scanner etc., Using OpenCV we can perform detection of specific objects such as eyes, faces etc., we can also analyze videos such as estimating the motion in the video or subtracting the background from the video, and tracking objects [8] in it. OpenCV covers the basic data structures such as Scalar, point etc. that are used for building OpenCV applications. OpenCV library is imported into the python using the code 'import cv2'.

C. *Eye Detection Using Dlib*

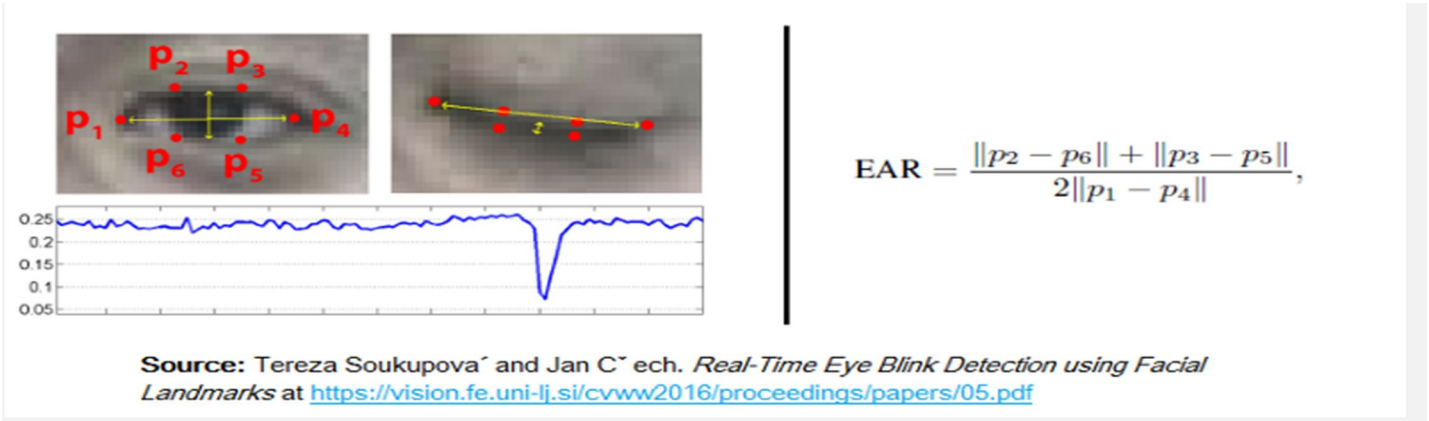
The first thing to do is to find eyes before we can move on to image processing and to find the eyes we need to find a face. The facial keypoint detector takes a rectangular object *of the dlib module* as input which is simply the coordinates of a face. To find faces we can use the inbuilt frontal face detector of dlib. You can use any classifier for this task. If you want high accuracy and speed is not an issue for you then I would suggest you use a CNN as it will give much better accuracy especially for non-frontal facing faces

D. *How It Works:*

This project is deeply centered around predicting the facial landmarks of a given face. We can accomplish a lot of things using these landmarks. From detecting eye-blinks in a video to predicting emotions of the subject. The applications, outcomes, and possibilities of facial landmarks are immense and intriguing. Dlib's prebuilt model, which is essentially an implementation of , not only does a fast face-detection but also allows us to accurately predict 68 2D facial landmarks. Very handy. Using these predicted landmarks of the face, we can build appropriate features that will further allow us to detect certain actions, like using the eye-aspect-ratio (more on this below) to detect a blink or a wink, using the mouth-aspect-ratio to detect a yawn etc or maybe even a pout. In this project, these actions are programmed as triggers to control the mouse cursor. PyAutoGUI library was used to move the cursor around.

E. *Eye-Aspect-Ratio (EAR)*

You will see that Eye-Aspect-Ratio is the simplest and the most elegant feature that takes good advantage of the facial landmarks. EAR helps us in detecting blinks and winks etc.



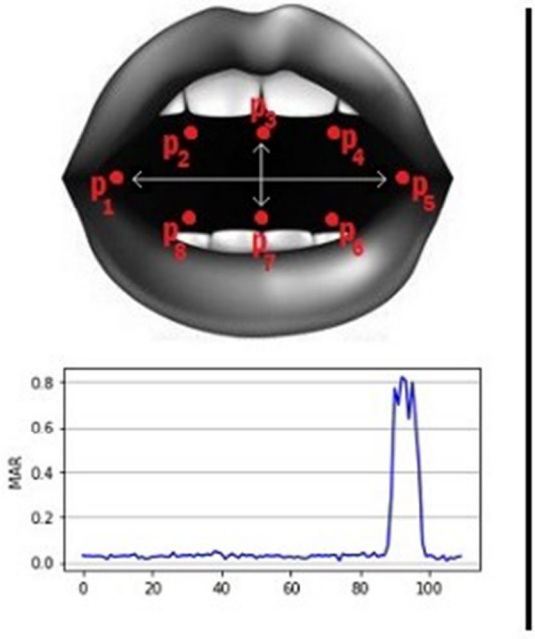
Source: Tereza Soukupova´ and Jan C˘ech. *Real-Time Eye Blink Detection using Facial Landmarks* at https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

You can see that the EAR value drops whenever the eye closes. We can train a simple classifier to detect the drop. However, a normal *if* condition works just fine. Something like this:

if EAR <= SOME_THRESHOLD:
EYE_STATUS = 'CLOSE'

F. Mouth-Aspect-Ratio (MAR)

Highly inspired by the EAR feature, I tweaked the formula a little bit to get a metric that can detect opened/closed mouth. Unoriginal but it works.



$$\mathbf{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\,\|p_1 - p_5\|}$$

Similar to EAR, MAR value goes up when the mouth opens. Similar intuitions hold true for this metric as well.

G. Prebuilt Model Details

The model offers two important functions. A detector to detect the face and a predictor to predict the landmarks. The face detector used is made using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme

You can get the trained model file from http://dlib.net/files, click on **shape_predictor_68_face_landmarks.dat.bz2**. The model, .dat file has to be in the project folder

XI. SYSTEM REQUIREMENTS

A. Hardware Requirements

- System : Pentium i3 Processor.
- Hard Disk : 500 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram : 4 GB

B. Software Requirements

- Operating system : Windows 10.
- Coding Language : Python
- Web Framework : Flask

XII. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue X Oct 2023- Available at www.ijraset.com

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

A. Types Of Tests

1) Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2) Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3) Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.
Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4) System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5) White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6) Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7) Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8) Test Strategy and Approach

Field testing will be performed manually and functional tests will be written in detail.

a) Test Objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

b) Features to be Tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

9) Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

10) Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

XIII. CONCLUSION

In order to make user interact with computer naturally and conveniently by only using their eye, we provide an eye tracking based control system. The system combines both the mouse functions and keyboard functions, so that users can use our system to achieve almost all of the inputs to the computer without traditional input equipment. The system not only enables the disabled users to operate the computer the same as the normal users do but also provides normal users with a novel choice to operate computer. According to our experiments, it is considered our eye movement system to be easy to learn. Meanwhile, participants show their interest in using the proposed eye control system to search and browse information. They are looking forward to see more of our research results on the use of eye tracking technique to interact with the computer.

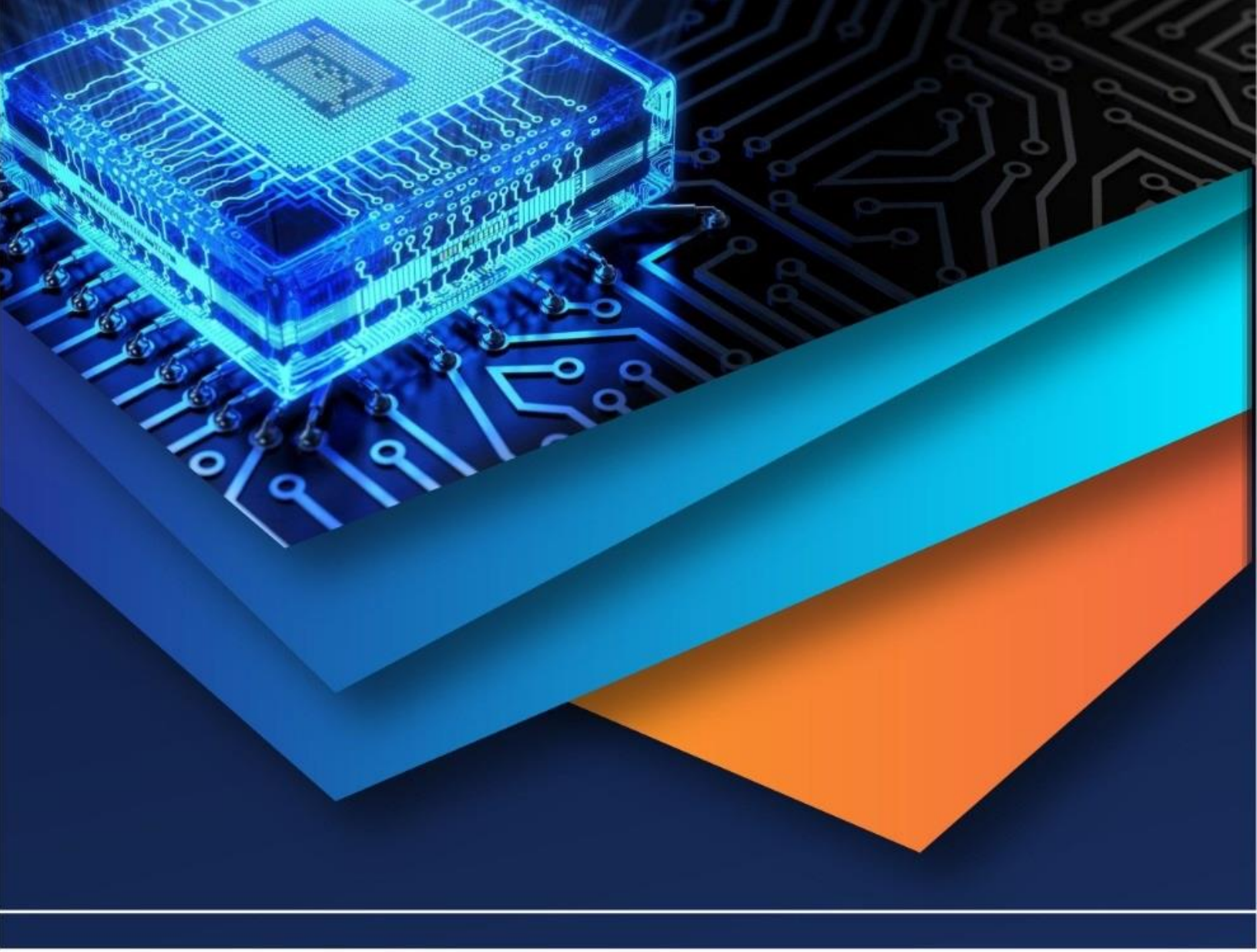
Future Work:

Currently, this system is applied for the general operating behavior to interact with computer by simulating mouse. In future, we will try to add new operation functions for more usage situations for users to communicate with media and adjust our system on new platform, such as tablet or phone. We will also develop series operation modules in order to achieve a complete operating experience for users from turning on to turning off the computer

REFERENCES

[1] Q. Sun, J. Xia, N. Nadarajah, T. Falkmer, J. Foster, and H.Lee, “Assessing drivers’ visual-motor coordination using eye tracking, GNSS and GIS: a spatial turn in driving psychology,” Journal of Spatial Science, vol. 61, no. 2, pp. 299–316, 2016.
[2] N. Scott, C. Green, and S. Fairley, “Investigation of the use of eye tracking to examine tourism advertising effectiveness,” Current Issues in Tourism, vol. 19, no. 7, pp. 634–642, 2016.
[3] K. Takemura, K. Takahashi, J. Takamatsu, and T. Ogasawara, “Estimating 3-D point-of-regard in a real environment using a head-mounted eye-tracking system,” IEEE Transactions on Human-Machine Systems, vol. 44, no. 4, pp. 531–536, 2014.
[4] R. J. K. Jacob and K. S. Karn, “Eye Tracking in humancomputer interaction and usability research: ready to deliver the promises,” Minds Eye, vol. 2, no. 3, pp. 573–605, 2003.
[5] O. Ferhat and F. Vilarino, “Low cost eye tracking: the current panorama,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 8680541, pp. 1–14, 2016.
[6] Tobii EyeX, “EyeX,” 2014, http://www.tobii.com/eyex.
[7] GazePoint, “Gazept,” 2013, http://www.gazept.com/category/gp3-eye-tracker.
[8] The eyeTribe, “EyeTribe,” 2014, http://www.theeyetribe.com.

[9] M. A. Eid, N. Giakoumidis, and A. El Saddik, "A novel eyegaze-controlled wheelchair system for navigating unknown environments: case study with a person with ALS," IEEE Access, vol. 4, pp. 558–573, 2016.
[10] L. Sun, Z. Liu, and M.-T. Sun, "Real time gaze estimation with a consumer depth camera," Information Sciences, vol. 320, pp. 346–360, 2015.
[11] Y.-M. Cheung and Q. Peng, "Eye gaze tracking with a web camera in a desktop environment," IEEE Transactions on HumanMachine Systems, vol. 45, no. 4, pp. 419–430, 2015.
[12] P. S. Holzman, L. R. Proctor, and D. W. Hughes, "Eye-tracking patterns in schizophrenia," Science, vol. 181, no. 4095, pp. 179–181, 1973.
[13] C. Donaghy, M. J. Thurtell, E. P. Pioro, J. M. Gibson, and R. J. Leigh, "Eye movements in amyotrophic lateral sclerosis and its mimics: a review with illustrative cases," Journal of Neurology, Neurosurgery & Psychiatry, vol. 82, no. 1, pp. 110–116, 2011.
[14] M. Nehete, M. Lokhande, and K. Ahire, "Design an eye tracking mouse," International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 2, 2013.
[15] E. Missimer and M. Betke, "Blink and wink detection for mouse pointer control," in Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '10), Samos, Greece, June 2010.
[16] GitHub, "MasterLomaster/bkb," 2015, https://github.com/MastaLomaster/bkb.
[17] H. Singh and J. Singh, "Human eye tracking and related issues: a review," International Journal of Scientific and Research Publication, vol. 2, no. 9, pp. 146–154, 2012.
[18] C. A. Chin, A. Barreto, J. G. Cremades, and M. Adjouadi, "Integrated electromyogram and eye-gaze tracking cursor control system for computer users with motor disabilities," Journal of Rehabilitation Research and Development , vol. 45, no. 1, pp. 161–174, 2008.
[19] R. G. Lupu, F. Ungureanu, and R. G. Bozomitu, "Mobile embedded system for human computer communication in assistive technology," in Proceedings of the 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing (ICCP '12), pp. 209–212, Cluj-Napoca, Romania, September 2012.
[20] R. G. Lupu, F. Ungureanu, and V. Siriteanu, "Eye tracking mouse for human computer interaction," in Proceedings of the 4th IEEE International Conference on E-Health and Bioengineering (EHB '13), Iasi, Romania, November 2013.
[21] S. Wankhede and S. Chhabria, "Controlling Mouse Cursor Using Eye Movement," International Journal of Application or Innovation in Engineering & Management, no. 36, pp. 1–7, 2013.
[22] S. M. A. Meghna, K. L. Kachan, and A. Baviskar, "Head tracking virtual mouse system based on ad boost face detection algorithm," International Journal on Recent and Innovation Trends in Computing and Communication, vol. 4, no. 4, pp. 921–923, 2016.
[23] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," MIS Quarterly, vol. 13, no. 3, pp. 319–339, 1989.
[24] F. D. Davis, R. P. Bagozzi, and P. R.Warshaw, "User acceptance of computer technology: a comparison of two theoretical models," Management Science, vol. 35, no. 8, pp. 982–1003, 1989.
[25] V. Venkatesh and F. D. Davis, "A theoretical extension of the technology acceptance model: four longitudinal field studies," Management Science, vol. 46, no. 2, pp. 186–204, 2000.
[26] T. S. H. Teo, V. K. G. Lim, and R. Y. C. Lai, "Intrinsic and extrinsic motivation in Internet usage," Omega , vol. 27, no. 1, pp. 25–37, 1999.
[27] V. Venkatesh, C. Speier, and M. G. Morris, "User acceptance enablers in individual decision making about technology: towards an integrated model," Decision Sciences, vol. 33, no. 2, pp. 297–316, 2002.
[28] A. Bangor, K. Philip, and M. James, "Determining what individual SUS scores mean: adding an adjective rating scale," Journal of Usability Studies, vol. 4, no. 3, pp. 114–123, 2009.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY