AGORA International Journal of Economical Sciences, http://univagora.ro/jour/index.php/aijes
ISSN 2067-3310, E-ISSN 2067-7669
Vol. 18, No. 2 (2024), pp. 320-333

FROM ORIGINS TO INNOVATIONS: AI'S ROLE AND THE COST IMPACT ON COMPUTER VISION

K. TIRANA, E. BEJLERI

Kledia Tirana[1], Endri Bejleri[2]
University Metropolitan Tirana, Albania
[1] https://orcid.org/0009-0008-0772-4243, E-mail: ktirana@umt.edu.al
[2] https://orcid.org/0009-0006-7586-4171, E-mail: endri.bejleri24@umt.edu.al

Abstract: *The field of Computer Vision, a pivotal subdomain of Artificial Intelligence (AI), has seen extraordinary advancements since its emergence in the 1960s. This paper examines the historical development of Computer Vision technologies, tracing the journey from early foundational models, such as Frank Rosenblatt's Perceptron, to contemporary breakthroughs driven by Deep Learning. Key milestones are explored, including the development of algorithms like Scale-Invariant Feature Transform (SIFT), Viola-Jones for face detection, and Eigenfaces, which paved the way for modern solutions such as Convolutional Neural Networks (CNNs), YOLO and FaceNet. The paper highlights the evolution of face detection and recognition techniques, contrasting traditional methods with the transformative capabilities of Deep Learning-driven approaches. Additionally, we analyze the growing computational demands of modern algorithms, discussing the trade-offs between accuracy and efficiency and their implications for practical applications. This study underscores the rapid progression of Computer Vision, its challenges, and its role as a cornerstone in shaping the future of Artificial Intelligence.*

Keywords: *Computer Vision, Algorithms, Deep Learning, Artificial Intelligence, Convolutional Neural Networks.*

INTRODUCTION

Computer Vision, a critical branch of Artificial Intelligence (AI), enables machines to interpret and process visual data with minimal human involvement. What began as a set of basic algorithms in the 1960s has grown into a sophisticated field that powers applications across industries such as healthcare, automotive, and security. This paper explores the historical trajectory of Computer Vision, starting with early milestones like the Perceptron, which showcased the promise of neural-inspired machine learning algorithms. Subsequent breakthroughs, including feature-detection techniques like SIFT and early face detection models like Viola-Jones, set the stage for today's sophisticated methods. The advent of Machine Learning and Deep Learning marked a turning point, introducing transformative technologies such as Convolutional Neural Networks (CNNs), YOLO, and FaceNet. These models deliver remarkable accuracy and efficiency in object detection and recognition.

Through this analysis, the paper provides an in-depth perspective on the evolution of Computer Vision—from its experimental origins to its current status as a pivotal field shaping the future of AI. Particular focus is given to the interplay between accuracy, computational efficiency, and practical usability, highlighting both the achievements and the ongoing challenges that define this dynamic domain.

The Rise of Computer Vision

Computer Vision, a specialized subfield of Artificial Intelligence (AI), empowers machines to "see" and interpret visual data. By leveraging Computer Vision techniques, machines can process, analyze, and extract meaningful insights from images and other visual inputs without human intervention (IBM, n.d.).

The origins of Computer Vision date back to the 1960s, when researchers began experimenting with algorithms to enable computers to interpret and analyze visual data, laying the foundation for this transformative field. In this discussion, we will explore the evolution of Computer Vision from its early experimental stages in the 1960s to its remarkable advancements in the 21st century (Minsky, 1966).

Frank Rosenblatt's Perceptron

Before we step into the origins of Computer Vision, first we need to understand Frank Rosenblatt's Perceptron. Introduced in 1958 by Frank Rosenblatt, the Perceptron was the earliest model that could demonstrate that machines could effectively learn and act from data. Frank's design was inspired by human's neurons and the Perceptron in itself was a simplified version of it. Perceptron showcased that a simple algorithm can be further improved by exposing the model to batches of data, laying the foundations for the creation of the more complex algorithms such as the modern Deep Learning models and Neural Networks used in today's era (Rosenblatt, 1958).

The Perceptron is a binary classifier, meaning it decides whether an input or stimuli belongs to one or two binary classes. It included 3 crucial components which we can also recognize in today's algorithms: Inputs & Weights, Activation Function and the Learning Algorithm.

The way the Perceptron works is that it takes one or more inputs, each assigned with an adaptable weight resembling the input's contribution to the output. The weighted sum of the inputs is that ran through an activation function which determines if the output is classified as a 0 or 1 based on a set threshold. To get the desired output, the weights are adjusted according to the error in the output compared to the desired output (weight = current weight + (error $*$ input)), a form of a learning algorithm. Despite its limitations of the time, the Perceptron laid the groundwork for ongoing advancements in Neural Networks and Machine Learning.

MIT's Summer Vision Project

During the 1960's researchers had tried to develop ways for computers to recognize and understand visual data. On 1963, Lawrence Gilman Roberts, known as Larry Roberts, presented his thesis "Machine Perception of Three-Dimensional Solids" where he presented methods to reconstruct 3D objects from 2D images, paving the path for 3D Computer Vision. He also introduced the idea of edge detection to identify the boundaries of objects in an image as well as mathematical models, which all contributed to future developments of Computer Vision technologies (Roberts, 1963).

During the summer of 1966, the "Summer Vision Project" at MIT's Artificial Intelligence Laboratory led by Marvin Minsky and Seymour Papert was built to develop a computer system that could recognize objects in images. The project was quite ambitious for the time and marked the beginning for further research in the area of Computer Vision. Its goal

was to develop algorithms that could make machines able to "see" and interpret data from real environments (Minsky, 1966).

The project focused on:

- Image Processing: Techniques for controlling and evaluating digital images, including filtering, enhancement and feature extraction.
- Object Detection: Development of algorithms that could capture and localize objects or faces within images or videos.
- Pattern Recognition: Development of methods for recognizing and differentiating patterns in visual information.

While the project didn't meet its initial objectives due to the difficulties of the time, including lack of processing power, sophisticated algorithms and computational models, it greatly influenced and inspired further research and innovations in the field of Computer Vision.

David Marr's Vision

David Marr was a British innovator and neuroscientist who made important contributions to the field of Computer Science throughout the years. During the late 1970s and early 1980s, Marr developed a theoretical framework for understanding vision systems, changing the way researchers thought about vision systems. In the book named "Vision" released on 1982, Marr proposed a general framework of how the human brain processes visuals and how machines could replicate this process (Marr, 1982). Marr's framework is divided into 3 levels to better understand how vision overall works:

- The Computational Level: The first level showcases the problem of what the vision is trying to accomplish. At the base, the computational level is about identifying this problem, specifically, the initial analysis is about trying to identify how the brain is able to create 3D objects from 2D images.
- The Algorithmic Level: The second level of analysis covers how to solve the problem the Computational Level puts on the board. It describes the specific methods used to tackle the issue. Marr concluded that the human brain must perform specific complex calculations to extract the needed information. As the name of the level, he theorized in the face that these calculations could be implemented/transformed into an algorithm. Marr proposed that detecting edges (contrast) is crucial for recognizing objects in a scene.
- Implementation/Physical Level: The final level of the framework explains how the system or the human body can achieve the goal set in the computational level. In the case of human body, neural structures and neuronal activities present in the brain and eye.

Marr also put forward the "Theory of Visual Perception" where he described the human vision as a hierarchical process, where the visual processing system includes several stages, from getting input as a two-dimensional visual array (on the retina) and progressing through different stages we will cover next to produce a three-dimensional description of the world as output (Marr, 1982).

- Primal Sketch: The initial stage of the process describes how the first step to visual processing is to create a low-level image by extracting basic visual features like edges and textures without any context.
- 2.5D Sketch: Once a low-level image has been created, the visual system tries to convert the image into 3D by assigning depth, orientation and spatial relationship of edges.
- 3D Model: In the final stage, a full 3D model is created, with every object identified and positioned independent of the observer's viewpoint.

David Marr's work in the theory of vision, to this day is highly regarded and serves as a cornerstone In the Computer Vision field.

Emergence of Machine Learning, Computer Vision Machine Learning Algorithms

During the 1980s and 1990s, with the increase of the computational power and advancements in the field of Artificial Intelligence technology, researchers started to experiment, develop and deploy advanced algorithms, Machine Learning algorithms to solve Computer Vision problems. In this section we will feature some important algorithms used throughout the years before the emergence of Deep Learning models which we will cover later on.

Scale-Invariant Feature Transform

Scale-Invariant Feature Transform (SIFT) is a feature detection algorithm introduced in 1999 by David Lowe, a researcher in computer vision. SIFT is designed to identify distinctive features within an image that are invariant to changes in scale and rotation. This makes SIFT a valuable tool for various computer vision tasks, including object recognition, image matching, and scene analysis. The algorithm begins by detecting keypoints/features, which are distinctive and scale-invariant within an image, such as corners or edges. To achieve this, SIFT constructs a scale-space representation of the image by applying Gaussian filters with varying levels of blur (Lowe, 1999).

Mathematically can be described as:

$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y)$

where $L(x,y,\sigma)$ is the blurred image at scale σ, $G(x,y,\sigma)$ is the Gaussian kernel, (x,y) is the original image and σ is the scale parameter. The image is also downsampled (reduced in size) after each octave, allowing features to be detected at smaller resolutions (or sizes) as well (Lowe, 1999).

Differences between successive Gaussian-blurred images, known as the Difference of Gaussians (DoG), are used to identify keypoints. Mathematically, the DoG is represented as:

$DoG(x,y,\sigma)=L(x,y,k\sigma)-L(x,y,\sigma)$

where k is a constant (typically around square root of 2) that controls the scale between successive Gaussian images. This difference highlights the edges and key points in the image at various scales and then the algorithm detects keypoints by finding the maxima and the minima which are selected only if the specific pixel value is larger than all of these neighbors or smaller than them all (Lowe, 1999). These keypoints are further refined by eliminating low-contrast or poorly localized points to ensure that only the most stable keypoints are retained (Lowe, 1999).

Once keypoints are identified, SIFT generates descriptors that describe the local appearance around each keypoint. This process involves analyzing the image gradients within a neighborhood surrounding the keypoint and creating a histogram of gradient orientations. The descriptor is constructed as a 128-dimensional vector summarizing these gradients, ensuring invariance to rotation and minor changes in lighting. These descriptors act as unique signatures for each keypoint, enabling robust comparisons between different images.

To match keypoints between two images, SIFT compares their descriptors, by identifying and matching keypoints, SIFT enables computers to perform complex tasks such as recognizing objects, combining overlapping images into seamless panoramas, reconstructing 3D structures from multiple views, and analyzing spatial relationships within scenes. By finding and matching these keypoints, SIFT can help machines perform tasks like:

- Object Detection
- Image Stitching
- 3D Reconstruction

SIFT is a key part of many modern computer vision systems. It is applied in different fields of AI like Object Recognition where SIFT features can be used to identify objects in images or videos, Image Stitching where SIFT features can help stitch multiple images together to create panoramas, 3D Reconstruction where SIFT features can be used to establish correspondences between images, enabling 3D reconstruction and Augmented Reality (AR) where SIFT features can be used to track objects or scenes in real-time for augmented reality applications (Lowe, 1999).

Viola Jones, Face Detection Algorithm

In 2001 Paul Viola and Michael Jones published a paper called "Rapid Object Detection using A Boosted Cascade of Simple Features" where they first introduced the Viola-Jones algorithm. The Viola Jones algorithm revolutionized real-time face detection and despite being an older method, due to its low computational needs and fast speed, it still finds use as face detection model in the current day (Viola, 2001).

There are three main steps in face detection by Viola Jones Algorithm:

- Integral Image: The first step creates a new image representation, known as the integral image, which allows for fast feature evaluation.
- AdaBoost: The second step utilizes AdaBoost, an algorithm to iteratively train weak classifiers on different parts of the training data.
- Cascade Classifier: The third step is to combine more complex classifiers in cascade structure which dramatically increases the speed of the detector by focusing attention on promising regions of the image. Viola and Jones used Haar-like features (more complex) to detect faces in this algorithm. Haar-like features help in detecting faces by capturing differences in intensity between rectangular regions of an image. These differences, often between contrasting areas, are key to identifying facial structures (Viola, 2001).

The Viola Jones algorithm maximizes speed while achieving high detection accuracy, making it suitable for live-video systems like surveillance systems. The algorithm, however is susceptible to image variations such as changes in lighting, non-regular frontal poses or

different expressions, lowering its accuracy significantly. Newer Deep Learning models can also outperform and outpace the Viola Jones algorithm.

Eigenfaces

Eigenfaces is a method used in face recognition, based on the idea of representing faces as a combination of principal components (or eigenvectors) of a set of training face images. It was introduced in the 1990s and is one of the earlier methods for face recognition (Turk, 1991). The method uses Principal Component Analysis (PCA) to reduce the dimensionality of face images while preserving the key features required for distinguishing between faces. The Eigenfaces method deconstructs face images into a small set of characteristic feature images, which form the basis of the initial training set. Recognition occurs by projecting a new image into the "face space" defined by these Eigenfaces. The new image is then classified by comparing its position to those of known Eigenfaces. Eigenfaces are a set of eigenvectors used in computer vision to address human face recognition. This approach focuses on capturing the variations within a collection of face images and uses this information to encode and compare individual faces in a holistic manner.

The primary objective of the Eigenfaces algorithm is to represent facial features in a lower-dimensional space by capturing the principal components of facial images. However, a notable limitation of PCA is its sensitivity to variations in lighting conditions. Since it captures overall variance, changes in illumination can significantly impact its effectiveness (Tirana, 2024).

Fisherfaces

Another famous algorithm used for face recognition is Fisherfaces, primarily of its ability to maximize the class distinguishing in the training process (Belhumeur, 1997). Fisherfaces combines PCA (Principal Component Analysis) to reduce face space dimension and LDA (Linear Discriminant Analysis) to obtain feature of image characteristic (Martinez, 2001). Fisherfaces is an extension of the Eigenfaces method, designed to improve face recognition performance, especially when dealing with interchanging conditions. The key difference between Eigenfaces and Fisherfaces is in the way how they handle variations (Belhumeur, 1997).

Eigenfaces focuses on the overall variation in the face dataset, while Fisherfaces seeks to enhance the class separability by focusing on the variation between classes rather than just the overall variance (Martinez, 2001). This approach utilizes LDA to identify a projection that enhances the separation between classes (individuals) while minimizing variations within the same class. LDA aims to maximize the ratio of between-class scatter to within-class scatter, emphasizing the discriminative power of features. The representation generated by Fisherfaces is specifically designed to highlight differences between individuals, making the selected features particularly effective for classification (Belhumeur, 1997).

However, Fisherfaces can be sensitive to certain challenges, especially in cases with small datasets or high within-class scatter. Significant variability in the images of the same individual—such as differences in facial expressions, poses, or lighting conditions—can limit its ability to capture the distinguishing features necessary for accurate recognition. The method can also become computationally expensive, especially with large datasets (Martinez, 2001).

Support Vector Machines (SVM)

Support Vector Machines (SVM) are supervised machine learning classifying algorithm designed to classify data into two distinct groups. They are widely used for their effectiveness in handling high-dimensional feature spaces, making them a popular choice for tasks like face recognition after the features are extracted. SVMs work by mapping data into a high-dimensional space where separating different categories becomes easier, even when the data is not initially separable. The algorithm identifies the best separator, known as a hyperplane, which divides the data into distinct classes. Once the data is transformed into this space, the hyperplane serves as the decision boundary (Dasgupta, Ray, & Talukdar, 2018). When new data points are introduced, the model determines their classification based on their position relative to the hyperplane (Cortes, 1995).

SVMs are split into two main types: Linear and Non-Linear SVMs. A Linear SVM is applied when the data can be perfectly separated by a straight line which happens rarely in real world applications, while a Non-Linear SVM is used for scenarios where the data points cannot be linearly separated. In these cases, techniques like kernel tricks are employed to facilitate classification (Cristianini, 2000).

SVM performs faster and more accurately when the data is linearly separable, and the kernel trick enables the algorithm to solve more complex problems. However, certain challenges in applications like face recognition must be addressed, such as selecting an appropriate kernel, managing inaccuracies with large datasets, and fine-tuning hyperparameters. While a classifier algorithm was essential in the past to perform facial recognition, the modern Deep Learning models often integrate built-in classifiers as part of their architecture (Tirana, 2024).

Modern Deep Learning Computer Vision Algorithms

Deep Learning algorithms have immensely changed the field of Artificial Intelligence, especially the field of Computer Vision. By leveraging multi-layered neural networks, these algorithms are exceptional at finding complicated patterns within large datasets. Convolutional Neural Networks (CNNs) are in the center of this revolution and they are great identifying features in images, including facial characteristics. These algorithms automatically learn hierarchical representations of data, getting rid of the need for manual feature extraction (O'Shea & Nash, 2015). Additionally different techniques have been created such as transfer learning which have further enhanced their performance, allowing pre-trained models to be well-tuned for specific applications. This chapter will explore the evolution of Computer Vision algorithms, focusing on Deep Learning's impact on facial recognition and detection technology. We will also compare the differences between the old Machine Learning algorithms and modern cutting-edge CNN-utilizing algorithms, showcasing the progress made in the 21st century in the field of Computer Vision.

Convolutional Neural Networks (CNN)

Before we get into the algorithms used in the modern era, first we need to understand how Convolutional Neural Networks (CNNs) are designed and how they work. CNNs are at the core of every modern Deep Learning algorithm's architecture. CNNs are the extended

version of Artificial Neural Networks (ANN) which are predominantly used to extract the feature from the grid-like matrix dataset. Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers (O'Shea & Nash, 2015).

- Input Layer: The first layer of the architecture, will take the raw input, prepare it for processing and hold the pixel values of the image.
- Convolutional Layer: This layer is key as it performs feature extraction. Convolutional Layers apply a convolution filter or kernel in which these filters are responsible for detecting low-level features like edges, textures, or gradients, which are then combined into higher-level features in subsequent layers followed by a non-linear activation function, such the Rectified Linear Unit (ReLU).

To further understand the structure, we must discuss ReLU. A non-linear activation function that has a simple derivative. (It eradicates negative values and only outputs positive values, the derivative of ReLU is either 0 or 1). ReLU has become the standard activation function in modern Deep Learning networks because it has an improved gradient propagation and it helps the network converge faster and mitigates the vanishing gradient problem (Agarap, 2018).

- Pooling Layer: The layer will then perform downsampling along the spatial dimensionality of the given input, reducing their dimensions while retaining the most important information.
- Fully Connected Layer: The last layer in the architecture, connects every neuron in the layer to every neuron in the previous layer (Han, Mao, & Dally, 2015). It acts as a decision-making layer as it combines the extracted features into a prediction for the final output (Taherdoost & Madanchian, 2023).

When these layers are stacked, a CNN architecture is formed (O'Shea & Nash, 2015).

Modern Face Detection Algorithms, One Stage vs Two Stage Detection Models

Modern face detection algorithms are split into two categories: One Stage and Two Stage Detection Models.

- One Stage Detection Models: Object classification and bounding-box regression are done without using pre-generated region proposals (candidate object bounding-boxes) (Redmon, 2016).
- Two Stage Detection Models: Contrary to One Stage Detection Models, Two Stage Detection Models utilize two steps, generation of region proposals either by selective search or Region Proposal Network (RPN) and then object classification is done for each region proposal. Two Stage Detection Models are more accurate but require more computing power and are quite slower than their counterpart (Ren, 2015).

We will cover one state-of-the-art algorithm for each of these, respectively: YOLO (You Only Look Once) and Faster R-CNN.

YOLO (You Only Look Once)

YOLO is a very fast, accurate object detector, making it ideal for Computer Vision applications. As a One Stage Detection Model, it is an approach of object detection that simplifies the process by treating it as a single regression problem rather than a traditional

classification task and unlike many other detectors that focus on region proposals, YOLO performs detection in one step, predicting both the class and location of objects in an image simultaneously hence the name You Only Look Once. The core concept behind YOLO is its efficiency: it looks at the image only once to detect what objects are present and where they are located (Redmon, 2016).

YOLO's architecture is composed of 24 convolutional layers followed by 2 fully connected layers. Convolutional layers are responsible for feature extraction, while the fully connected layers handle the final predictions.

The way YOLO works is as follow: the system splits the image into a grid of cells, with each cell predicting B bounding boxes and confidence scores for those boxes. The confidence score reflects both the probability that the box contains an object and the accuracy of the predicted box. Each grid cell in YOLO predicts C conditional class probabilities, generating a single set of class probabilities per grid cell, regardless of the number of predicted bounding boxes B. During the testing phase, these class probabilities are multiplied by the individual box confidence scores, resulting in class-specific confidence values for each bounding box. These scores indicate represent the likelihood that the object in the cell belongs to a specific class, assuming an object exists and how the box fits the object.

The algorithm also uses the Intersection over Union (IoU) metric to assess the accuracy of bounding box predictions. By default, an IoU threshold of 0.5 is used, which can be adjusted to control false positives and false negatives. A higher IoU threshold reduces false positives but may increase false negatives, as only boxes with an IoU greater than the threshold are considered valid detections. The output is then encoded as an $S \times S \times (B * 5 + C)$ tensor, where $S \times S$ are the grid dimensions, B is the number bounding boxes, 5 represents the 4 coordinate values for each bounding box + 1 confidence score and C stands for the number of classes.

A key component of YOLO is Non-Maximum Suppression (NMS), which ensures that each object is detected only once, even if multiple bounding boxes are predicted for the same object. NMS keeps only the bounding box with the highest confidence score for each object and removes the others. After NMS, YOLO outputs the final set of bounding boxes, each associated with a class label and a confidence score showing the probability of the class and how well the bounding box fits the object.

As YOLO processes each image in a single pass, it is very fast (YOLO processes images in real-time at 45 frames per second), doesn't require the highest computational power but may lose in accuracy, especially when detecting small objects in the image compared to Two Stage Detection Models such as R-CNN which we will be covering next. YOLO has been significantly improved since release, with each variation improving the issues of the latter, up to version YOLOv9 which is the latest release.

R-CNN

R-CNN is a Two-Stage Detection Model which operates by first generating potential object proposals and then classifying each proposal in the second stage. Unlike one-stage models like YOLO, R-CNN takes a more methodical approach by dividing the process into two steps, which contributes to its higher accuracy, especially in complex scenarios (Girshick, 2014).

R-CNN's architecture begins with a region proposal step, where potential object regions are identified using selective search. Selective search generates a set of candidate object regions (bounding boxes) by grouping similar regions from the image. In the second stage, each of these proposed regions is then passed through a Convolutional Neural Network (CNN), which extracts features for classification and bounding box regression.

The network uses CNN features to perform a classification task where each region proposal is classified into one of the predefined object categories or marked as background. Additionally, the algorithm predicts bounding box corrections to improve the accuracy of the initial proposals. This two-step approach ensures that R-CNN can focus on the most likely object areas, leading to more accurate predictions compared to one-stage models like YOLO.

While R-CNN achieves high accuracy, especially in detecting objects of various sizes and complexities, its main drawback is slowness and its very high computing power cost. This is due to the fact that R-CNN requires the CNN to process each individual region proposal separately, which is computationally expensive and time-consuming (Liu, Hu, Weng, & Yang, 2017). For every image, hundreds or thousands of proposals are generated, and each one is forwarded through the CNN for feature extraction, followed by classification. This multiple-step process makes R-CNN much slower than single-shot detectors like YOLO.

To improve efficiency, Fast R-CNN was introduced, which shares convolutional features across all region proposals to reduce redundant computations, but it still maintains the two-stage approach. Further optimization led to Faster R-CNN, which integrated Region Proposal Networks (RPN) into the pipeline, eliminating the need for selective search and improving both speed and accuracy.

R-CNN is effective in detecting objects in a wide range of images, but its two-stage process makes it more suited for tasks where accuracy is prioritized over real-time speed, such as in applications where detection quality is critical, and computational resources are available.

In summary, R-CNN's two-stage approach allows it to achieve high accuracy by focusing on region proposals and using a CNN for classification and bounding box regression. However, the method is slower and require more computation power compared to single-shot detectors like YOLO, making it less suitable for real-time applications. Despite this, R-CNN and its improved versions (Fast R-CNN and Faster R-CNN) remain vital in the evolution of deep learning-based object detection.

Modern Face Recognition Algorithms, FaceNet

FaceNet is a Deep Learning face recognition model introduced in 2015 by Google researchers. The idea behind FaceNet is to transform images of faces into a fixed-length Euclidean space, known as embeddings, where the distances reflect the similarity or dissimilarity between faces. This enables the system to perform both face recognition, verification and face clustering, using a process known as one-shot learning (Schroff, 2015).

FaceNet uses a Deep Convolutional Neural Network (CNN) architecture to learn a 128-dimensional embedding for each face. This embedding is a compact representation of a face’s features, trained so similar faces are closer together in the embeddings and faces of different people are farther apart. Its architecture follows the following schema: an input layer where each image is resized to 160x160 for standardization, followed by 22 convolutional layers, structured in 13 blocks.

Each block consists of convolutional layers followed by batch normalization and ReLU activation functions (Huang, Liu, & Weinberger, 2016; Lim, Kim, Choo, & Choi, 2023; Dubey & Jain, 2019). Multiple max pooling layers, placed after every few convolutional layers perform downsampling along the spatial dimensionality which help decrease the computational cost while maintaining the key features.

To optimize FaceNet for face recognition, the network uses a triplet loss function. In this loss function, the network is trained using triplets of images: one anchor image, one positive image (same person as the anchor), and one negative image (different person). The goal is to minimize the distance between the anchor and positive image embeddings, while maximizing the distance between the anchor and negative image embeddings. This helps the network learn the similarity relationships between different faces.

Once the features are extracted, 3 fully connected layers containing 1024 neurons followed by batch normalization and ReLU activation, 512 and 128-dimensional embedding vector respectively process the output and give as output the 128-dimensional embedding vector which is then used to perform facial recognition or identification on the output layer. To verify if two faces belong to the same person, the embeddings of both faces are compared using the Euclidean distance. If the distance is below a set threshold (typically 0.6), the faces are considered a match.

Due to its well-designed deep CNNs architecture and the triplet loss function, FaceNet is extremely accurate as shown with its accuracy of 99.63% in the Labeled Faces in the Wild (LFW) dataset. Due to it utilizing the one-shot learning technique, FaceNet is also very fast. However, FaceNet requires large amounts of computational power due to its deep CNNs architecture and the triplet loss function can be difficult to optimize. Overall FaceNet is a massive improvement compared with its predecessors and comes inherently with a classifier so an algorithm like SVM isn't needed for classification.

Cost Perspective: From Perception to Deep Learning

The journey of Computer Vision from its inception to the modern age has not only been marked by technical advancements but also by evolving cost structures. Understanding this progression provides insights into the technological and financial investments required to achieve breakthroughs.

The Perceptron hardware was relatively inexpensive for its time, as it relied on simple circuitry to implement neural-like computations. However, the computational capabilities were limited (Rosenblatt, 1958):

- Costs: Low due to the simplicity of components and reliance on existing computing technologies.
- Performance vs. Cost: Inefficient for complex tasks due to hardware limitations and inability to solve nonlinear problems.
- Use Cases: Minimal, mostly experimental and limited to binary classification tasks.

Algorithms like Scale-Invariant Feature Transform (SIFT), Viola-Jones, and Eigenfaces brought about practical applications but at varying costs:

- SIFT:

    - Computationally expensive due to the need for Gaussian blurring and keypoint detection at multiple scales.
    - Costs were tied to the processing power of the time, which was growing but still limited for large datasets.
    - Applications: Object detection, image stitching.
- Viola-Jones (Viola, 2001):
    - Lower computational cost due to simplified features and the use of AdaBoost.
    - Suited for real-time face detection with modest hardware, making it accessible for security and surveillance systems.
- Eigenfaces (Belhumeur, 1997):
    - High sensitivity to variations in lighting and pose.
    - Cost tied to dataset preparation and PCA computation.
    - Effective for small-scale, controlled environments.

With the introduction of Support Vector Machines (SVMs) and Fisherfaces, costs began to shift:

- SVMs:
    - Computational costs increased due to the need for hyperparameter tuning and kernel tricks.
    - Effective but resource-intensive for large datasets.
- Fisherfaces (Belhumeur, 1997):
    - Combined PCA and LDA, increasing computational requirements while improving robustness.
    - Performance depended heavily on dataset quality and size, adding costs for data preprocessing.

The rise of Convolutional Neural Networks (CNNs) and models like YOLO, R-CNN, and FaceNet has revolutionized vision capabilities but at a steep cost:

- Hardware:
    - Requires GPUs or TPUs for training and inference, significantly increasing infrastructure costs.
    - Cloud computing services (e.g., AWS, Google Cloud) have democratized access but introduced recurring expenses.
- Energy Consumption:
    - High during training due to large datasets and deep architectures.
    - YOLO is optimized for real-time applications, lowering energy costs but with slightly reduced accuracy compared to R-CNN.
- Data Preparation:
    - Costs tied to collecting and labeling large datasets.
    - Techniques like transfer learning help reduce training time and data needs, lowering costs slightly.
- Model Complexity:
    - R-CNN variants achieve high accuracy but are resource-intensive.
    - YOLO offers a trade-off, being faster and cheaper for real-time applications.

The following tables summarizes the cost comparisons:

Technology	Cost (Relative)	Performance/Accuracy	Computational Needs	Use Case Scope
Perceptron	Low	Minimal	Very Low	Experimental
SIFT	Medium	Good	Medium	Object Detection
Viola-Jones	Low	Moderate	Low	Real-Time Detection
Eigenfaces	Medium	Moderate	Medium	Face Recognition
SVM	High	Good	High	Classification
CNN (YOLO)	High	Very Good	Medium	Real-Time Applications
CNN (R-CNN)	Very High	Excellent	Very High	Accuracy-Critical

Source: Author's own elaboration. (2024)

CONCLUSIONS

The progression of Artificial Intelligence in Computer Vision showcases an extraordinary transformation from its early days of handcrafted feature-based methods to the advanced, Deep Learning-driven models we see today. In the realm of face detection, modern algorithms like YOLO and R-CNN have significantly outperformed traditional approaches such as the Viola-Jones algorithm, offering higher accuracy and greater adaptability to varying conditions. Similarly, advancements like FaceNet have surpassed earlier models, achieving exceptional precision that rival human vision. However, a trade-off exists: while these contemporary methods provide unparalleled accuracy, they demand substantial computational power and often operate slower than their predecessors, except for optimized models like YOLO. Despite the widespread adoption of cloud computing, the balance between accuracy and computational efficiency doesn't meet the needs of every business or use case.

The evolution of Computer Vision algorithms underscores the rapid pace of innovation, suggesting that ongoing research and development will continue to optimize and refine these technologies as we advance into a promising era of Artificial Intelligence.

REFERENCES

1. Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU).
2. Belhumeur, P. N. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
3. Cortes, C. &. (1995). Support-vector networks. *Machine Learning*.
4. Cristianini, N. &.-T. (2000). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press.
5. Dasgupta, S., Ray, S. N., & Talukdar, P. (2018). HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding. *Conference on Empirical Methods in Natural Language Processing.*
6. Dubey, A., & Jain, V. (2019). Comparative Study of Convolution Neural Network's Relu and Leaky-Relu Activation Functions. In *Lecture Notes in Electrical Engineering.*
7. Girshick, R. D. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*
8. Han, S., Mao, H., & Dally, W. (2015). Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations.*
9. Huang, G., Liu, Z., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

10. IBM. (n.d.). *Computer vision*. Retrieved from IBM: https://www.ibm.com/topics/computer-vision
11. Lim, H., Kim, B., Choo, J., & Choi, S. (2023). A Domain-Shift Aware Batch Normalization in Test-Time Adaptation. *International Conference on Learning Representations.*
12. Liu, Z., Hu, J., Weng, L., & Yang, Y. (n.d.). Rotated region based CNN for ship detection. *International Conference on Information Photonics, 2017 IEEE International Conference on Image Processing (ICIP).*
13. Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision.* IEEE.
14. Marr, D. (1982, September). *Vision: A computational investigation into the human representation and processing of visual information.* MIT Press. Retrieved from https://www.researchgate.net/publication/235626691_The_Vision_of_David_Marr
15. Martinez, A. M. (2001). PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
16. Medium.com. (2023, November 1). *Frank Rosenblatt's Perceptron, Birth of The Neural Network*. Retrieved from https://medium.com/@robdelacruz/frank-rosenblatts-perceptron-19fcce9d627f
17. Minsky, M. a. (1966). *Summer Vision Project.* MIT Artificial Intelligence Laboratory.
18. O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks.
19. Osowski, S., & Siwek, K. (2020). *CNN application in face recognition* (Vol. 96).
20. Princeton. (2023). *Human Cognitive*. Retrieved from https://pni.princeton.edu/research-areas/human-cognitive#:~:text=Human%20cognitive%20neuroscience%20is%20a,%2C%20language%20and%20problem%2Dsolving.
21. Redmon, J. D. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*
22. Ren, S. H. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems.*
23. Roberts, L. G. (1963). *Machine Perception of Three-Dimensional Solids.* MIT.
24. Rosenblatt, F. (1958). *The Perceptron: A probabilistic model for information storage and organization in the brain.*
25. Schroff, F. K. (2015). FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*
26. Sciotex. (2023). *A Brief History of AI in Vision Systems*. Retrieved from https://pni.princeton.edu/research-areas/human-cognitive#:~:text=Human%20cognitive%20neuroscience%20is%20a,%2C%20language%20and%20problem%2Dsolving.
27. sciotex. (2024, January). *A Brief History of AI in Vision Systems*. Retrieved from https://sciotex.com/a-brief-history-of-ai-in-vision-systems/#:~:text=Artificial%20Intelligence%20(AI)%20has%20revolutionized,incredible%20technologies%20we%20have%20today.)
28. Taherdoost, H., & Madanchian, M. (2023). Multi-Criteria Decision Making (MCDM) Methods and Concepts.
29. Tirana, K. (2024). A Comprehensive Evaluation of Face Recognition Algorithms. *AGORA INTERNATIONAL JOURNAL*.
30. Turk, M. a. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*.
31. Viola, P. &. (2001). Rapid Object Detection using A Boosted Cascade of Simple Features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*
32. Živanović, S., Abramović, N., Živanović, M., & Smolović, S. (2023, 4). *INNOVATION MANAGEMENT ON THE WAY TO BUSINESS EXCELLENCE* (Vol. 17).