IEEE Open Journal on
Immersive Displays

Received 27 July 2024; revised 15 September 2024; accepted 25 September 2024. Date of publication 30 September 2024;
date of current version 18 October 2024. The review of this article was arranged by Associate Editor Kai Wang.

Digital Object Identifier 10.1109/OJID.2024.3470711

Artificial Intelligence (AI) Fundamentals for the Display Industry: A Review (August 2024)

EUNKYUNG KOH, HYEON-DEUK KIM, SEUNGIN BAEK, AND CHANGHEE LEE (Senior Member, IEEE)

(Invited Paper)

Display Research Center, Samsung Display Company Ltd., Asan 31454, South Korea

CORRESPONDING AUTHOR: EUNKYUNG KOH (e-mail: eunkyung.koh@samsung.com).

ABSTRACT Artificial intelligence (AI) has been implemented into the display industry in recent years. This paper introduces AI theories applied to the display industry, covering concepts of AI, machine learning, and deep learning. It reviews optimization techniques for developing machine learning models and explains various deep learning architectures and learning paradigms applicable to the display industry. Additionally, it introduces explainable AI techniques for model analysis. Practical application cases will be reviewed in a separate paper.

INDEX TERMS Artificial intelligence, machine learning, deep learning, algorithms, displays.

I. INTRODUCTION

Machine learning has been evolving since the mid-20th century, starting with early developments in algorithms like perceptrons and decision trees. In the 2010s, deep learning significantly advanced the field, particularly through neural networks such as convolutional neural networks (CNNs) [1]. These deep learning algorithms and related methodologies have led to breakthroughs in AI applications across various industries.

In the display industry, it has been widely applied to areas such as research and development (R&D) and manufacturing. Application cases in the display industry will be reviewed in a separate paper [2]. This review paper aims to explain essential AI theories to understand these application cases. For a more comprehensive review of AI, one may refer to [1], [3] and the references therein.

This paper is organized as follows. Section II reviews the concepts of AI, machine learning (ML), and deep learning (DL). Section III introduces a typical workflow of ML and optimization techniques used along this workflow. Section IV classifies ML tasks and introduces primary metrics and models for each task. Section V provides architectures of DL models frequently encountered in display industry-related literature. Section VI examines learning paradigms for efficiently training models in various scenarios. Section VII introduces explainable AI (XAI) techniques that can be used for the model analysis. Finally, Section VIII presents the conclusion.

II. AI, MACHINE LEARNING, AND DEEP LEARNING

AI, ML and DL form a hierarchical relationship, as depicted in Fig. 1. Hard-coded knowledge-based automation and optimization are subfields of AI. ML is a subfield of AI that acquires knowledge from raw data by automatically extracting patterns from the given data [1]. Intrinsically, ML is a data-driven method. If an ML model uses a feedforward deep network or multilayer perceptron as its algorithm, it is referred to as deep learning.

A. MACHINE LEARNING

The process related to machine learning can be divided into the training and inference, as shown in Fig. 2.

1) TRAINING

During training, an ML model learns from raw data iteratively by optimizing model parameters to minimize the loss. The loss quantifies the difference between the model's prediction and the ground truth of the data, known as the label.

Validation involves evaluating and selecting the optimal model from many trained models. A validation set, separated from the training set, is used to ensure that the model has not been overfitted to the training data.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

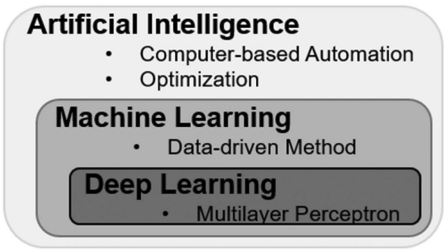

FIGURE 1. Hierarchical relationship between AI, machine learning, and deep learning.

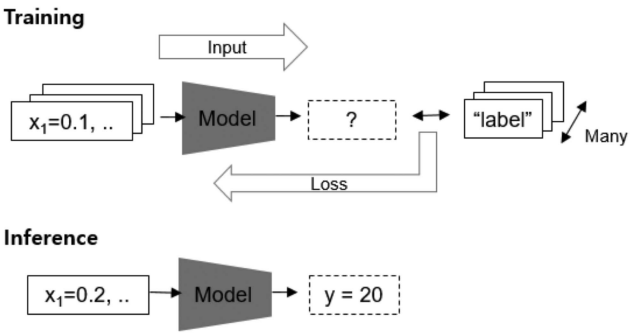

FIGURE 2. The machine learning process, consisting of training (above) and inference (below).

The test process follows validation and involves evaluating the final model's performance on completely unseen data that was not encountered during either the training or validation stages. This ensures that the performance assessed during testing reflects the model's ability to generalize to new data.

2) INFERENCE

The inference refers to the deployment of a trained model to make predictions based on new observations.

B. DEEP LEARNING

DL is a type of ML algorithm that employs a feedforward deep network, which is also referred to as multi-layered artificial neural networks (ANNs).

1) ARTIFICIAL NEURAL NETWORKS

An ANN consists of interconnected nodes (neurons) organized into layers: an input layer, one or more hidden layers, and an output layer (Fig. 3, above). Each node processes its input through weighted connections and applies a non-linear activation function (Fig. 3, below). This activation function introduces non-linearity to the network, enabling it to model complex patterns. Weights and biases are trainable parameters within an ANN.

The fully connected, multi-layered ANN shown in Fig. 3 is also referred to as a multi-layered perceptron (MLP) in the literature. When the architecture comprises only one hidden layer, the network is termed a shallow model, to distinguish it from DL models. Shallow models are often used for tasks such as word embedding, where they effectively capture semantic relationships while maintaining computational efficiency.

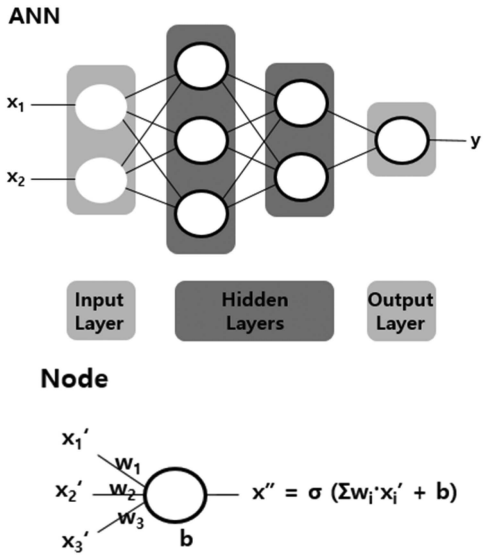

FIGURE 3. (Above) A fully connected ANN with two hidden layers. (Below) A node processes its inputs, denoted as x', with the weight w_i and bias b, and then applies a non-linear activation function σ.

2) UNIVERSAL APPROXIMATION THEOREM

The universal approximation theorem states that there exists a sequence of neural networks capable of approximating any continuous function on a compact subset of R^n to any desired degree of accuracy [4]. Although the theorem does not provide a practical guide to find the exact sequence, it offers theoretical support for using DL models.

3) BACK PROPAGATION

Backpropagation is an algorithm used in training ANNs that calculates the gradient of the loss function with respect to each weight using the chain rule, and then updates the weights according to these gradients [5]. This algorithm enables the analytical calculation of parameter updates, as opposed to numerical methods, facilitating the training of DL models with a large number of parameters.

The explicit form of parameter updates can vary depending on the method used, known as optimizers. Frequently used optimizers in DL include Adaptive Momentum Estimation (Adam) [6], and Root Mean Square Propagation (RMSprop) [7], which enhance stochastic gradient descent (SGD) by adaptively adjusting the learning rate and using the momentum to avoid local minima.

III. MACHINE LEARNING WORKFLOW

A typical machine learning workflow includes defining the task, preprocessing the data, building machine learning models, and deployment.

A. DEFINING THE TASK

The first step in ML is to define the task. This involves clarifying the problem to solve and identifying the type of machine learning task, such as classification, regression, or clustering. Exploratory data analysis (EDA) is performed in this step

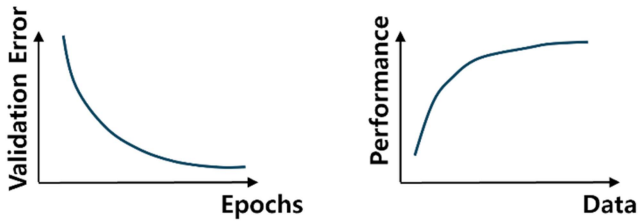

FIGURE 4. Two types of learning curves: (left) The epochs versus error curve, and (right) the amount of data versus performance curve.

to understand the data distribution and identify relevant data from various sources.

B. DATA REPROCESSING

Data need to be preprocessed before building models. Typical preprocessing steps for tabular data include: (1) handling missing values, (2) normalization or standardization, (3) encoding categorical values, (4) adjusting class imbalance, and (5) removing outliers.

For images, preprocessing techniques such as contrast enhancement, de-noising, and histogram normalization can be employed [8]. In the display industry, where defect sample images are often rare, data augmentation methods like geometric transformations and color space adjustments are frequently used. Data augmentation or synthesis can also help overcome challenges associated with data collection, with generative AI models used to synthesize data.

In industrial environments, tabular data, such as sensor values, are often imbalanced. To adjust data imbalance, methods like SMOTE [9] have been suggested.

C. TRAINING, EVALUATION, AND OPTIMIZATION

In this step, it is essential to choose an appropriate ML algorithm based on the task and data. Publicly available tools can compare the performance of various algorithms on a given dataset [10]. Subsequently, training optimization techniques can be applied to enhance model performance and prevent overfitting.

1) LEARNING CURVES

In ML, there are two primary types of learning curves (Fig. 4) that provide valuable information for optimizing learning.

Firstly, the epochs versus error learning curve plots the number of training epochs (i.e., iterations) on the x-axis against the training and validation error on the y-axis. This curve helps identify issues such as overfitting and convergence problems and can be used to adjust the initial learning rate.

Secondly, the amount of data versus performance curve plots the size of the training dataset on the x-axis against the model's performance metric, such as accuracy, on the y-axis. This curve is useful for assessing the efficiency of different learning methods and for predicting performance improvements from additional data, helping to determine whether further data collection is necessary.

2) LEARNING TECHNIQUES

Early stopping is a regularization technique used in training ML models, where the training process is halted before completion if the model's performance on a validation set begins to degrade, indicating potential overfitting. The metric used to decide when to halt training may differ from the loss function used during the training process.

Adjusting the learning rate at a plateau of the learning curve can enhance performance. This technique, known as learning rate scheduling or learning rate annealing, involves decreasing the learning rate to make more fine-grained adjustments to the trainable parameters, potentially overcoming the plateau and leading to further performance improvements. When scheduling involves increasing the learning rate, it is often referred to as annealing, which helps the model avoid being trapped in local minima during training.

3) ENSEMBLE METHODS

Ensemble methods combine the predictions of multiple models to improve overall performance and generalization. By leveraging the strengths of diverse models, ensemble methods often yield higher accuracy and robustness compared to individual models.

K-fold cross-validation, which involves splitting the training set into K subsets, is commonly used to build ensemble models. Each subset, or fold, serves as a validation set while the remaining K-1 folds are used for training. As a result, K different models are acquired, from which an ensemble can be formed using the high-performing models.

Voting and stacking are two methods for combining predictions from individual models. Voting uses the majority of predictions for classification tasks and the average of predictions for regression tasks. Stacking involves combining the outputs of individual models using a meta-learner. The meta-learner, a higher-level model, uses the outputs of the individual models as its input features.

4) HYPERPARAMETER OPTIMIZATION

Hyperparameters are the parameters of an ML model that are not updated during the training, as well as the external parameters related to the learning process. Examples include the number of neurons in a neural network and the learning rate.

To adjust hyperparameters, one can observe the model's bias and variance [3]. High bias indicates that the model does not perform well even on the training set, suggesting that features may need to be added or the model architecture adjusted. Low bias but high variance indicates overfitting. In this case, adding regularizations such as dropout, using feature selection methods, or reducing the number of trainable parameters may help. When both bias and variance are low, the model performs well on both the training and validation sets; increasing the number of trainable parameters may further enhance performance.

For automatic hyperparameter optimization (HPO), commonly used algorithms include random search, grid search, and Bayesian optimization. Hyperband [11] is a more recent algorithm that accelerates random search through adaptive resource allocation and early stopping. Neural architecture search (NAS) [12] utilizes reinforcement learning methods for HPO.

D. MODEL DEPLOYMENT

At this step, the model is prepared for deployment. A common practice involves retraining the model on the entire dataset using the finalized hyperparameters and training processes. It is also essential to establish environments for monitoring, retraining, and updating the model. MLOps (Machine Learning Operations) encompasses the practices and tools used to streamline the deployment, monitoring, and management of ML models in production environments.

For certain applications, reducing latency, the time taken for a model to process a data point, can be desirable. Techniques such as quantization [13] and pruning [14] can be employed to achieve this goal.

IV. ML TASKS

ML tasks can be broadly categorized into supervised learning, unsupervised learning, and reinforcement learning.

A. SUPERVISED LEARNING

Supervised learning involves training the model on labeled data, where each training example is paired with an output label. Through this process, an ML model learns a mapping from inputs to outputs, which can then be used to predict the labels of new, unseen data.

1) REGRESSION

Regression is a type of supervised learning that predicts continuous values.

Key metrics for regression include the followings. (1) Mean Absolute value Error (MAE) is the average of the absolute differences between the predicted and actual values, measuring the magnitude of errors without considering their direction. (2) Mean Squared Error (MSE) is the average of the squared differences between the predicted and actual values. (3) Root Mean Squared Error (RMSE) is the square root of MSE, providing an error metric in the same units as the target variable. (4) R-squared (R^2) coefficient is the proportion of variance in the dependent variable that is predictable from the independent variables, indicating the goodness of fit of the model. (5) Mean Absolute Percentage Error (MAPE) is the average of the absolute percentage differences between the predicted and actual values, providing a normalized measure of prediction accuracy as a percentage.

Several ML models are commonly used for regressions tasks. The vanilla linear regression can be combined with L1 or L2 regularization which penalize large coefficients to prevent overfitting. Ridge regression uses L2 regularization, Lasso regression uses L1 regularization, and Elastic Net with combines both L1 and L2 regularizations. Lasso regression is used as a feature selection method because L1 regularization tends to enforce some coefficients in a linear model to vanish.

Examples for non-linear models are as follows: (1) decision trees, (2) Random Forests which is an ensemble method using multiple decision trees where different trees are trained either on a subset of the training data or using a subset of features, (3) Gradient Boosting Machines (GBM), an ensemble method that builds models sequentially with each new model predicts the errors made by the previous models, (4) Support Vector Regression (SVR) which utilizes a hyperplane to minimize errors, (5) K-nearest neighbors (KNNs), a non-parametric method that predicts the target variable based on the average of the k-nearest training examples, (6) Extreme Gradient Boosting (XGBoost), an optimized implementation of GBM [15], (7) Light Gradient Boosting Machine (LightGBM), a gradient boosting model that allows asymmetric trees, which is known for its speed and efficiency [16], (8) Categorical Boosting (CatBoost), a GBM that implemented native handling of categorical features [17].

Despite recent advancements in DL, models based on gradient boosting and decision trees are often observed to outperform DL models for tabular data, particularly for datasets with irregularities [18], [19]. Examples of deep learning-based regression models with tree-like architectures, designed for tabular data, can be found in references [20] and [21].

For image data, tasks such as depth estimation or segmentation are regarded as regression tasks, where the goal is to predict a continuous-valued output for each pixel or region in an image.

2) CLASSIFICATION

Classification is a task that predicts discrete labels or categories. For instance, automated optical inspection (AOI) systems can use classification to differentiate between repairable and non-repairable defects in images. The results of classification can be summarized using a confusion matrix.

Metrics commonly used in classification tasks include precision, recall, and F1 score. Precision is the proportion of true positive predictions out of the total predicted positives. Recall is the proportion of true positive predictions out of the total actual positives. The F1 score is the harmonic mean of precision and recall. The Area Under the Curve of the Receiver Operating Characteristic (AUC-ROC) is another often used metric to evaluate a classification model's performance.

For classification, loss functions such as cross-entropy and focal loss are commonly used. Focal loss is an extension of cross-entropy loss designed to address class imbalance by adding parameters that increase the effect of minor class samples.

Various models can be employed for classification tasks. Logistic regression is a linear model for binary classification and can serve as a simple baseline model. Decision tree-based models which were discussed in the context of regression, can

also be used for tabular data classification, along with support vector machines (SVMs) and KNNs.

For image data, the ResNet model won the ImageNet classification competition in 2015 [22] with a top-5 error rate of 3.57%, surpassing human-level performance estimated at around 5%. Other notable DL models for classification will be reviewed in Section V.

3) DETECTION AND SEGMENTATION

Detection and segmentation are image related tasks.

Detection involves identifying and classifying local objects within an image. This typically results in outputs that include the coordinates defining the bounding box of an object and class scores for each object category. YOLO (You Only Look Once) models, as developed in [23], are widely used for various detection applications. The Slicing Aided Hyper Inference (SAHI) framework, proposed in [24], enhances detection performance, especially for small objects, by systematically slicing and resizing the original image before applying detection models.

Segmentation involves assigning a class label to each pixel in an image, a process known as semantic segmentation. When the task also includes identifying and differentiating each individual object within the image, it is referred to as instance segmentation.

4) SEMI-SUPERVISED LEARNING

Semi-supervised learning combines a small amount of labeled data with a large amount of unlabeled data to improve learning accuracy [25]. Active learning, which will be reviewed in Section VI, can be regarded as a type of semi-supervised learning.

5) SELF-SUPERVISED LEARNING

In self-supervised learning, a model generates labels from surrogate tasks using data without explicit labels. For instance, the masking technique is used to train language models like BERT [26]. In this approach, parts of the input text are randomly masked, and the model is then trained to predict these masked parts, allowing it to learn contextual representations of words.

B. UNSUPERVISED LEARNING

Unsupervised learning trains an ML model on unlabeled data, with the goal of identifying underlying patterns, structures, or relationships within the data. Techniques such as clustering, dimensionality reduction, and anomaly detection are examples of unsupervised learning.

1) CLUSTERING

Clustering involves grouping a set of data points into clusters. Clustering methods can be categorized into hierarchical, partitioning, density-based, and model-based approaches.

The hierarchical methods create a hierarchy of clusters using a tree-like structure, which can be agglomerative or divisive. Partitioning methods divide the data into a predefined number of clusters, then optimize a specific objective function related to the clustering performance. An example is the K-Means algorithm, which minimizes the sum of squared distances between points and their respective cluster centroids. Density-based methods create clusters on areas of high density. An example is DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. Model-based methods assume that the data is a mixture of underlying probability distributions and estimate the parameters of these distributions, Gaussian Mixture Models (GMM) as an example.

Various metrics are used to evaluate clustering algorithms for a given dataset. Frequently used metrics include the Davies-Bouldin index, Calinski-Harabasz index, and the Silhouette coefficient.

2) DIMENSIONALITY REDUCTION

In ML, reducing the dimensionality of features can be beneficial for improving model performance and interpretability. Two primary techniques for dimensionality reduction are Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (T-SNE).

PCA transforms a dataset by projecting it onto principal components (PCs), which are the eigenvectors of the covariance matrix of the dataset. The eigenvalues indicate the magnitude of variance along the corresponding eigenvectors. To effectively reduce dimensionality, one can retain only PCs with the highest eigenvalues, thereby preserving the most significant features of the data while reducing the number of dimensions.

T-SNE is primarily used for visualizing high-dimensional data by mapping it to a lower dimensional space. In this lower dimensional space, it uses Student's t-distribution with one degree of freedom to compute the pairwise similarities. Compared to the Gaussian distribution, the t-distribution has heavier tails which helps t-SNE prevent the crowding problem and ensures that dissimilar points are not placed too close together.

3) ANOMALY DETECTION

Anomaly detection is a task aimed at identifying anomalous data points. There are three primary methods, statistical, proximity-based, and deviation-based [27].

Statistical methods assume a specific distribution for the data points, such as a Gaussian distribution, and measure deviations from this distribution. An example is the Z-score.

Proximity-based methods use the proximity of a data point to other data points to detect anomalies, assuming that anomalous data are isolated from the majority of the data. An example is the Isolation Forest algorithm [28].

Deviation based methods use reconstruction errors as anomaly scores, where the reconstruction error of a data point is defined as the difference between the original data and its reconstruction from a low dimensional representation. PCA

components or a latent space of a Variational Autoencoder (VAE) can be used as the low dimensional representation [27].

C. REINFORCEMENT LEARNING

Reinforcement learning (RL) is a method where an agent learns optimal actions through trial-and-error interactions with an environment [29].

1) TERMINOLOGY OF RL

The agent is defined as the entity that interacts with the environment. The actions are choices made by the agent to interact with the environment. The states represent the current configuration of the environment. The reward is the feedback on the actions taken in a given state. In RL, a scalar function is used as the reward function.

The policy determines the agent's behavior for the given state by specifying each action's probabilities. The value of a policy, denoted as $V^{\pi}(s)$ in Eq. (1), is the expected cumulative reward that an agent can achieve starting from state s and following policy π thereafter. When calculating the cumulative reward, future rewards are discounted by a factor known as the discount factor (γ), which ranges between 0 and 1.

2) MARKOV DECISION PROCESS

To define an ML task as an RL problem, it needs to be framed as a Markov Decision Process (MDP). An MDP is defined as a process where the transition probability (P) between states is determined by the states (S), actions (A), and rewards (R), while satisfying the Markov property of the states. The Markov property is that the future state depends only on the present state and action, not on the sequence of events that preceded it. In other words, the state representation should provide sufficient information for agent's decision.

3) BELLMAN EQUATION

The learning of an agent relies on the Bellman equation, which provides a recursive decomposition for the value of a policy, given as follows:

$$V^{\pi}(s) = \mathrm{Exp}_{\pi}\left[R_{t+1} + \gamma V^{\pi}(S_{t+1}) \,|S_t = s\right]. \qquad (1)$$

4) REINFORCEMENT LEARNING ALGORITHMS

RL algorithms can be classified into three primary categories, model-free, model-based, and policy gradient.

The model-free methods do not require a model of the environment and learn the value function directly. An example of this is Deep Q-learning [30]. The model-based methods are applicable when a model of the environment is provided. Policy gradient methods optimize the policy by adjusting the policy parameters to maximize the value. Examples include REINFORCE [31], actor-critic [32], and PPO (Proximal Policy Optimization) [33].

V. DEEP LEARNING MODELS

Deep learning models, based on neural networks, can be classified by their distinct interconnected structures between neurons. Primary examples include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and transformers. Generative models focus on learning the distribution of the raw data, while discriminative models learn to differentiate between different data points. Representative examples of generative models are Variational Auto-Encoders (VAEs), Generative Adversarial Network (GAN), and diffusion models.

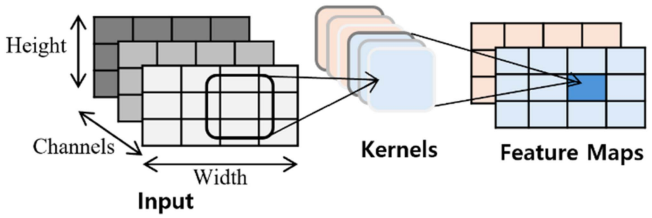

FIGURE 5. Schematic diagram illustrating a convolutional layer, where the kernel weights are trainable parameters.

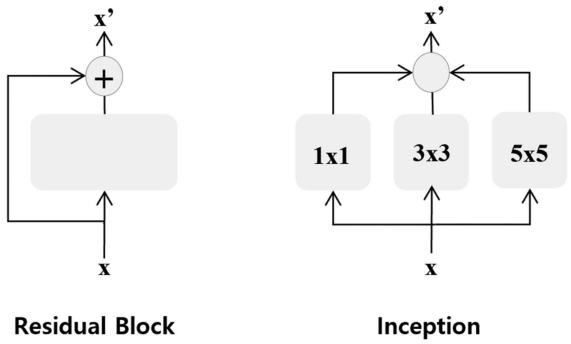

FIGURE 6. Schematic diagrams of a residual block (left) and an Inception block (right). The gray boxes represent convolutional layers, with the numbers indicating kernel sizes.

A. FULLY CONNECTED NEURAL NETWORK

A Fully Connected Neural network (FCN), also referred to as dense network or dense ANN, is a type of neural network where each node is connected to every node in the previous layer, forming a dense matrix of connections (Fig. 3, above).

B. CONVOLUTIONAL NEURAL NETWORK (CNN)

A Convolutional Neural Network (CNN) is a DL model that utilizes convolutional layers. In a convolutional layer, each node is connected to a subset of nodes in the previous layer, determined by the size of the kernel (Fig. 5). This localized connectivity allows CNNs to efficiently capture hierarchies of features in structured data like images or time series.

1) RESIDUAL BLOCK

The residual block, introduced in [22], uses a skip connection that adds the input directly to the output of intermediate layers (Fig. 6, left). This technique enables very deep neural networks by preventing the vanishing gradient problem.

2) INCEPTION

The inception architecture, introduced by the Inception Network [34], incorporates multiple convolutions of different

kernel sizes simultaneously and concatenates their outputs (Fig. 6, right).

3) OTHER VARIATIONS

There are variations from the vanilla CNN architecture for different purposes.

Depthwise separable convolution, introduced in [35], splits the standard convolution into a depthwise convolution, processing each channel separately, and a pointwise convolution. This technique aims to reduce the number of parameters compared to conventional CNNs while achieving similar performance. In [36], the authors introduced the channel shuffle operation which rearranges feature maps to increase performance. The paper [37] discussed scaling the depth, width, and resolution of the network to achieve better performance within resource constraints.

Traditional CNN kernels are applied uniformly across the entire feature map. However, for some applications, it is desirable to have spatially variant kernels. Spatial attention, incorporated in [38], enhances the spatial variability and adaptability of kernels.

C. GRAPH NEURAL NETWORK (GNN)

A GNN is a neural network model designed to operate on graph-structured data. Graph-structured data consist of a collection of nodes connected by edges [39]. Node vectors encode features associated with each node, while the adjacency matrix captures pairwise relationships between nodes, indicating the connections and the strength or weight of those connections. In the display industry, graph representation is useful for modeling the product flow in manufacturing lines and modeling the molecular structure of materials.

1) MESSAGE PASSING NEURAL NETWORKS (MPNN)

MPNN is a term used equivalently to GNN. The term describes how a GNN utilizes message passing schemes to iteratively update node representations based on messages exchanged between neighboring nodes [40].

2) GRAPH CONVOLUTIONAL NETWORK (GCN)

When a GNN model employs convolutional layers for processing nodes, the model is referred to as a GCN [41]. In other words, GCNs apply convolutional operations over the adjacency matrix.

3) GRAPH ATTENTION NETWORK (GAT)

A GAT, introduced in [42], uses attention mechanisms to assign different weights to neighboring nodes' contributions.

D. RECURRENT NEURAL NETWORK (RNN)

An RNN is a type of ANN designed to perform sequential processing by incorporating feedback loops [43]. RNNs are effective for maintaining long-term dependencies in data by avoiding the vanishing gradient problem using cell structures like Long Short-Term Memory (LSTM) [44] and Gated Recurrent Units (GRU) [45]. These cells achieve this purpose through gating mechanisms that regulate the flow of information, ensuring important information is preserved over long sequences.

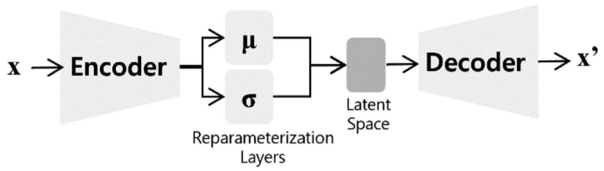

FIGURE 7. Schematic representation of the variational autoencoder (VAE) architecture.

E. VARIATIONAL AUTOENCODER (VAE)

An autoencoder is a model designed to reconstruct input data. Autoencoders typically have an encoder-decoder structure, where the encoder maps the input into a latent space representation, and the decoder reconstructs the original input from this encoded representation.

The VAE, suggested in [46], has numerous applications in the display industry. Compared to vanilla autoencoders, the VAE imposes Gaussian distributions on the latent variables using reparameterization layers and a loss function that involves a Kullback-Leibler (KL) divergence term for the latent variables (Fig. 7). KL divergence measures the difference between two probability distributions, in this case, the difference between the latent variable distributions and Gaussian distributions.

F. GENERATIVE ADVERSARIAL NETWORK (GAN)

GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously. The generator aims to create data similar to the training data, while the discriminator distinguishes whether a sample is generated or real [47]. There are many variants of GANs, and some frequently encountered models will be reviewed here, categorized by their tasks.

1) GENERATIVE MODELING

Once trained, the generator of a GAN model can be used to generate new samples. Deep Convolutional GAN (DCGAN), suggested in [48], employed convolutional neural networks, while StyleGAN [49] provided controlling parameters for synthesizing images. The Adversarial Autoencoder (AAE) [50] combined autoencoders with GANs to regularize the distribution of latent variables.

2) IMAGE-TO-IMAGE TRANSLATION

Pix2Pix [51] converts paired images from one domain to another. CycleGAN [52] performs a similar task, however, does not require paired datasets for training. Super-Resolution GAN, given in [53], aims to achieve higher resolution images from lower resolution inputs.

G. TRANSFORMER

The transformer model utilizes an attention layer to capture global dependencies between elements in a sequence [54]. In many applications, transformers are increasingly preferred over RNNs, especially in Natural Language Processing (NLP) tasks [26], [55]. They have also been used for applied beyond NLP, such as in predicting molecular reactions [56].

Transformers can be classified into three categories based on their architectures: encoder-only, decoder-only, and encoder-decoder transformers. A primary example of encoder-only transformers is BERT [26], which is primarily used for contextualized representation of the input, enabling various downstream tasks based on the representation. An example of decoder-only transformers is Generative Pre-trained Transformer (GPT) [55]. The Text-To-Text Transfer Transformer (T5) [57], and the original transformer model [54] are examples of encoder-decoder architectures, suitable for sequence-to-sequence tasks such as machine translation.

H. DIFFUSION MODEL

Diffusion models (DMs) generate data by progressively denoising a variable that initially starts as pure noise [58]. Currently, they are used for applications such as synthesizing high-resolution images [59] and text-to-image generation [60]. A comparative study between GANs and DMs can be found in [61]. While DMs are noted for their advantages in terms of image quality, GANs typically generate images faster than DMs because GANs require only a single forward pass through the network, whereas DMs involve multiple steps.

VI. LEARNING PARADIGMS

A learning paradigm refers to a framework or approach that dictates how learning is structured and executed. The choice of a learning paradigm can depend on factors such as the availability and quality of data, computational resources, and the specific goals of the task. In this section, we will review several examples of learning paradigms.

A. TRANSFER LEARNING

Transfer learning begins with a pretrained model, often trained on a large dataset, and then fine-tunes this model on a new dataset. Typically, only the later layers of the pretrained model are updated, while the weights of earlier layers, assumed to capture generic features, remain fixed. This approach is widely used in computer vision and NLP tasks involving deep learning models.

B. ACTIVE LEARNING

Active learning is a learning paradigm where a model interacts with an annotation system to intelligently select the most informative data samples for labeling.

Active learning strategies can be categorized into density-based, representativeness-based, and uncertainty-based approaches. Density-based methods select data to be labeled based on their proximity to existing labeled instances. Representativeness-based methods aim to choose data that are most representative of the overall data distribution, often using clustering techniques. Uncertainty-based methods select data based on the uncertainty of the model's predictions [62].

Various metrics can evaluate the uncertainty of a data point. For classification tasks, examples include the entropy of predicted class probabilities, and the absolute value of the difference between the two highest class probabilities. For regression tasks, the standard deviation of predicted values from an ensemble model is commonly used. Metrics for uncertainty can be either model-based [63] or model-agnostic [64].

C. KNOWLEDGE DISTILLATION

Knowledge distillation [65] is a technique that a smaller, computationally more efficient model, known as the student model, is trained to mimic the behavior of a larger, more complex model, known as the teacher model.

D. CURRICULUM LEARNING

Curriculum learning is a learning paradigm that organizes training data in a meaningful sequence, or curriculum, starting with simpler examples and gradually increasing in complexity [66]. Training on easier examples is considered beneficial for improving the final model's performance, as it helps the model learn generalizable features early in training. This early learning can help prevent the model from becoming trapped in local minima.

E. FEDERATED LEARNING

In federated learning, multiple devices collaboratively train a shared model while keeping their data stored locally [67]. This paradigm is useful in scenarios where data privacy is paramount and can also be advantageous when dealing with large datasets, such as AOI images in the display industry.

F. ONLINE LEARNING

Online learning, known as incremental learning, is a paradigm where models are continuously updated as new data arrives, allowing them to adapt to changes in the data distribution over time [68]. In this approach, learning is conducted incrementally, typically on one data point or mini-batch at a time.

VII. EXPLAINABLE AI (XAI)

XAI refers to methods in AI that aim to make AI models and their decisions understandable and interpretable.

A. XAI FOR TABULAR DATA

1) FEATURE IMPORTANCE

Feature importance for tabular data can be calculated using various methods. For tree-based models, a common metric is Gini importance, which measures the total reduction of Gini impurity brought by a feature across all trees in the model. For linear models, the coefficients can be used as an importance metric, assuming proper normalization between features. Another metric is permutation importance, which evaluates the decrease in model accuracy when the values of a feature are randomly permuted.

2) SHAPLEY INDEX

The Shapley index is a concept from cooperative game theory that calculates the distribution of the total payoff among players by considering the marginal contributions of each player to every possible coalition [69]. In ML, this index is used to quantify the contribution of each feature to the model's output for a given data point. This method can also be extended to images to explain the importance of each pixel in the prediction [70].

3) LIME

Local Interpretable Model-agnostic Explanations (LIME) perturbs input data around a prediction locally and evaluates the model's response to these perturbations [71]. It can be applied regardless of the underlying model and has been explicitly discussed for use with images.

4) PARTIAL DEPENDECE PLOT (PDP)

Partial Dependence Plots (PDPs) visualize the relationship between a feature and the predicted outcome while marginalizing over the other features. They show how changes in a feature affect the model's prediction [72].

B. XAI FOR IMAGE DATA

Gradient-weighted Class Activation Mapping (Grad-CAM) is a technique in deep learning that visualizes the importance of different regions of an input image for making a specific prediction. It computes the gradients of the target class score with respect to the feature map from the last convolutional layer, and then overlays the result on the input image. The resulting heat map highlights which parts of the image were important for the specific class prediction [73]. Methods such as Guided Grad-CAM [73], SmoothGrad [74], and Integrated Gradients [75] are also available and may be preferred in certain cases.

VIII. CONCLUSION

In this review, we have introduced the essential AI concepts relevant to applications in the display industry, which will be explored in detail in a separate paper [2]. Understanding these fundamentals will facilitate the identification of new potential applications of AI within the display industry.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[2] E. Koh, H.-D. Kim, S. Baek, and C. Lee, "A review on the applications of artificial intelligence (AI) in the display industry," *IEEE Open J. Immersive Displays*, vol. 1, pp. 165–172, 2024, doi: 10.1109/OJID.2024.3458904.

[3] A. Ng, "Machine learning yearning: Technical strategy for AI engineers, in the era of deep learning," USA, self-published, 2018. [Online]. Available: https://info.deeplearning.ai/machine-learning-yearning-book

[4] K. Hornik et al., "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989, doi: 10.1016/0893-6080(89)90020-8.

[5] D. E. Rumelhart et al., "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[7] G. Hinton, "Lecture 6e, RMSProp: Divide the gradient by a running average of its recent magnitude," 2012. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_1ec6.pdf

[8] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, 2019, doi: 10.1186/s40537-019-0197-0.

[9] N. V. Chawala et al., "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.

[10] M. Ali et al., "PyCaret: An open source, low-code machine learning library in Python," PyCaret version 1.0.0, 2020. [Online]. Available: https://www.pycaret.org

[11] L. Li et al., "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 185, pp. 1–52, 2018.

[12] B. Zoph et al., "Neural architecture search with reinforcement learning," 2016. [Online]. Available: https://arxiv.org/abs/1611.01578

[13] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[14] S. Han et al., "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, vol. 28, pp. 1135–1143.

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd Assoc. Comput. Machinery Sigkdd Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[16] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 3146–3154.

[17] L. Prokhorenkova et al., "CatBoost: Unbiased boosting with categorical features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 6638–6648.

[18] L. Grinsztajn et al., "Why do tree-based models still outperform deep learning on typical tabular data?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 507–520.

[19] D. McElfresh et al., "When do neural nets outperform boosted trees on tabular data?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 76336–76369.

[20] S. O. Arik et al., "TabNet: Attentive interpretable tabular learning," in *Proc. Assoc. Advance. Artif. Intell. Conf. Artif. Intell.*, 2021, vol. 35, pp. 6679–6687, doi: 10.1609/aaai.v35i8.16826.

[21] X. Huang et al., "TabTransformer: Tabular data modeling using contextual embeddings," 2020. [Online]. Available: https://arxiv.org/abs/2012.06678

[22] K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[23] P. Jiang et al., "A review of Yolo algorithm developments," *Procedia Comput. Sci.*, vol. 199, pp. 1066–1073, 2022, doi: 10.1016/j.procs.2022.01.135.

[24] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," in *Proc. IEEE Int. Conf. Image Process.*, 2022, pp. 966–970, doi: 10.1109/ICIP46576.2022.9897990.

[25] O. Chapelle et al., *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006, doi: 10.7551/mitpress/9780262033589.001.0001.

[26] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[27] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture IE*, vol. 2, no. 1, pp. 1–18, 2015.

[28] F. T. Liu, "Isolation forest," in *Proc. IEEE 8th Int. Conf. Data Mining*, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[30] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[31] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992, doi: 10.1007/BF00992696.

[32] V. Konda et al., "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, vol. 12, pp. 1008–1014.

[33] J. Schulman et al., "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[34] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[35] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[36] X. Zhang et al., "An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[37] M. Tan et al., "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[38] M. Jaderberg et al., "Spatial transformer networks," in *Proc. Adv. Nueral Inf. Process. Syst.*, 2015, vol. 28, pp. 2017–2025.

[39] J. Zhou et al., "Graph neural networks: A review of methods and applications," *Artif. Intell. Open*, vol. 1, pp. 57–81, 2020, doi: 10.1016/j.aiopen.2021.01.001.

[40] J. Gilmer et al., "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[41] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02907

[42] P. Velickovic et al., "Graph attention networks," 2017. [Online]. Available: https://arxiv.org/abs/1710.10903

[43] D. Zipser and R. J. Williams, "Gradient-based learning algorithms for recurrent networks and their computational complexity," in *Backpropagation*. London, U.K.: Psychology Press, 2013, pp. 433–486.

[44] A. Graves et al., "Long short-term memory," *Supervised Sequence Labelling Recurrent Neural Netw.*, pp. 37–45, 2012.

[45] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014. [Online]. Available: https://arxiv.org/abs/1406.1078

[46] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Representations*, 2014. [Online]. Available: https://arxiv.org/abs/1312.6114

[47] I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 2672–2680.

[48] A. Radford et al., "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[49] T. Karras et al., "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[50] A. Makhzani et al., "Adversarial autoencoders," 2015. [Online]. Available: https://arxiv.org/abs/1511.05644

[51] P. Isola et al., "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1125–1134.

[52] J. Y. Zhu et al., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2223–2232.

[53] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4681–4690.

[54] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 5998–6008.

[55] T. B. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.

[56] P. Schwaller et al., "Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction," *Amer. Chem. Soc. Central Sci.*, vol. 5, no. 9, pp. 1572–1583, 2019, doi: 10.1021/acscentsci.9b00576.

[57] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[58] J. Ho et al., "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 6840–6851.

[59] R. Rombach et al., "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10684–10695.

[60] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 36479–36494.

[61] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 8780–8794.

[62] S. Tong, "Active learning: Theory and applications," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2001.

[63] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, Univ. Oxford, New York, NY, USA, 2016.

[64] B. Park et al., "Data uncertainty without prediction models," 2022. [Online]. Available: https://arxiv.org/abs/2204.11858

[65] J. Gou et al., "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, 2021, doi: 10.1007/s11263-021-01453-z.

[66] Y. Bengio et al., "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[67] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," *Artif. Intell. Statist.*, vol. 54 pp. 1273–1282, 2017.

[68] S. C. Hoi et al., "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[69] L. Shapley, "A value for n-person games," in *Classics in Game Theory*. Princeton, NJ, USA: Princeton Univ. Press, 1997, doi: 10.1515/9781400829156-012.

[70] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 4765–4774.

[71] M. T. Ribeiro, "'Why should I trust you?':Explaining the predictions of any classifier," in *Proc. 22nd Assoc. Comput. Machinery SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[72] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[73] R. Selvaraju et al., "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 618–626.

[74] D. Smilkov et al., "Smoothgrad: Removing noise by adding noise," 2017. [Online]. Available: https://arxiv.org/abs/1706.03825

[75] M. Sundararajan et al., "Axiomatic attribution for deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 3319–3328.