NAVIGATING THE WAVES: CONVOLUTIONAL NEURAL NETWORKS FOR COMPREHENSIVE NAVAL VESSEL SURVEILLANCE AND CLASSIFICATION TITLE

Vaibhav Chadha[1], Vikas Chumber[2], Palkin Sharma[3], Aishwarya Dhara[4]

[1]*Assistant Professor, Aerospace Engineering, GNA University,*

[2]*Assistant Professor, Aerospace Engineering, GNA University,*

[3]*Assistant Professor, Electronics and Communication Engineering, GNA University, India*

[4]*Assistant Professor, Aerospace Engineering, GNA University, India*

Email:- vaibhavchadha29@gmail.com
vikaschumber92@gmail.com
palkin3120@gmail.com
dharaaishwarya@gmail.com

Abstract

In an era characterized by global connectivity through diverse transportation modes, naval vessels play a pivotal role in facilitating the movement of goods and people across continents via waterways. The heterogeneity of these vessels, spanning varying displacements and intended purposes, necessitates a comprehensive classification system for effective surveillance. This review paper addresses the critical need for vessel classification in waterway channels and ports, employing image classification techniques. Focusing on object recognition, the paper proposes the application of Convolutional Neural Networks (CNNs) to classify diverse vessel types. The significance lies in the management of waterborne traffic, where a multitude of vessel types coexist. The paper emphasizes the development of a CNN model trained on a dataset comprising labelled vessel images, catering to the nuanced requirements of model training. The envisioned outcome is a robust model capable of accurate vessel classification, contributing to enhanced naval surveillance and maritime security.

Keywords: tourism industry, growth, development, foreign exchange earnings, India

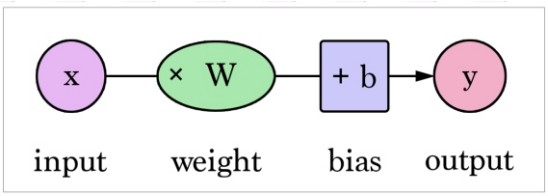
Introduction

The world has shrunken down connecting one point to another through the modes of transportation such as roadways, airways and waterways. The majority of the freight and population that is moved between the continents is done by the means of waterways, inculcating various types of vessels in the operations. These vessels vary in displacement, payload and their intended purposes, for said a small kayak would be suitable for a maximum of 2 people while displacing the smallest amount of water (Albawi et al., 2017). These vessels need to be classified into various types using image classification. This has significance as object recognition since many of the waterway channels and associated ports face a plethora of vessels as traffic which consists of all the types in it. The goal of the development of

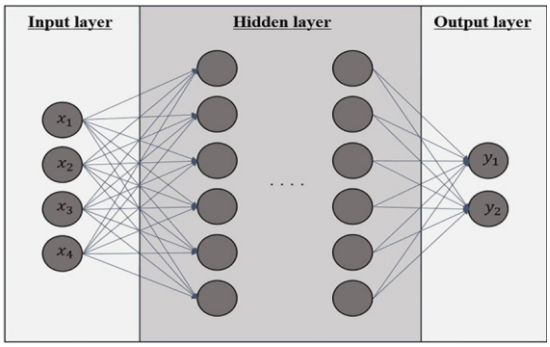
this system is to provide a model capable of classifying the vessels by making predictions. Thus, a Convolutional Neural Network could be used for the problem statement. CNN could be used for classifying the images into distinguished classes or labels. In this application, the dataset that has been used consists of multiple images of vessels distinguished into vessel types (Li et al., 2021). This data set could be used to satisfy the various prerequisites of the CNN model training, such as curating a knowledgeable training set. Your contribution should be prepared in Microsoft Word.

Methodology

Provided the application of convolutional neural networks is designated for object detection and classification, multiple layers of neural networks could be stacked to form a trainable model. The layer is a collection of several neurons which are responsible for the transformation of the data fed into it using activation functions (Lo et al., 1995). Once the application of the neural network has been decided, a model is created using these layers, connecting each node or the neuron involved in a layer is associated with the layer weights known as layer weights. The consecutive nodes in the succeeding layer are fed with a weighted sum of the output of the preceding node with the associated weights. It could be explained as the following diagram:



A model could be overviewed as the following diagram:

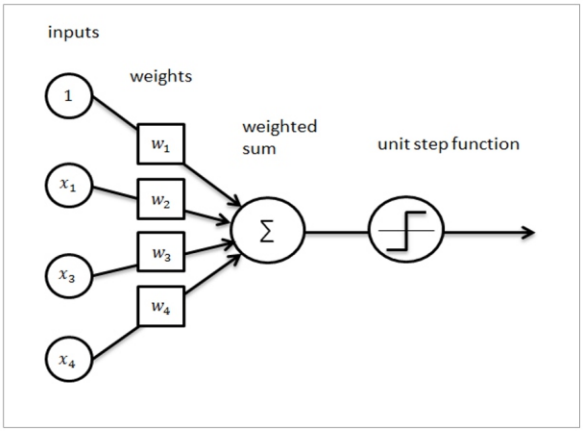


A Convolutional Neural Network model is proposedly comprised up of triple layer setup as briefly described below:

Input layer: In the above depiction of a CNN model, all the input nodes for said x1 to x4 (artificial neurons) represent the input layer. All the artificial neurons responsible for the input are termed to be *f*Passive Neurons≈. These neurons are said to be passive, sans the backpropagation (Kamath et al., 2019). An input layer is responsible for taking in the data and passing it on to the subsequent layers,

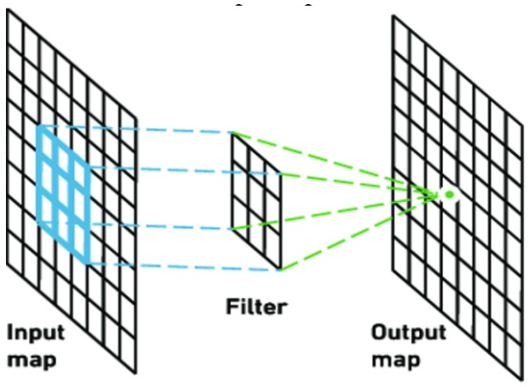
artificial neurons incorporated in the input layer lack predefined layer weight or randomly associate the layer weights with data.

Hidden layer(s): Hidden layers are intermediary layers among the input and output layers, responsible for the computational operations. Each connected node in the hidden layer accepts the inputs from the preceding layer, followed by the multiplication with the layer weights providing the weighted sum for the activation function. Hidden layers are intermediary layers among the input and output layers, responsible for the computational operations. Each connected node in the hidden layer accepts the inputs from the preceding layer, followed by the multiplication with the layer weights providing the weighted sum for the activation function.

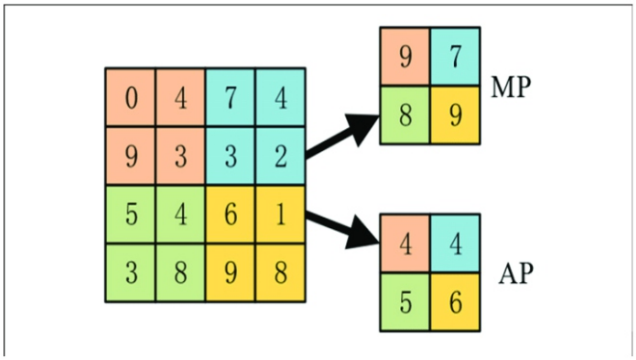


These hidden layers function together, fully or partially connected, an image classification model consists of the following essential layers:

Convolutional Layer: A convolution layer contains an ensemble of filters or sometimes referred to as kernels, whose parameters are meant to be learned, the filters' peak and weight are both lower than the recessed volume, and each filter is convoluted with the input data to generate an activation map from the fake neurons. To put it another way, the filter is moved across the dimension and height of the input in proposed strides and also the purpose product between the input and the filter is calculated at each abstraction position, for the spatial position. The convolutional layer's output volume is calculated by stacking the activation cards of all the filters on the depth (Bayar & Stamm, 2016; Ahmed & Islam, 2023; Birajdar & Mankar, 2013)). Because each filter's width and height are less than the input volume, each artificial neuron inside the activation map is merely linked to the local area of the input volume. In other words, the scale of each neuron's receptive field is small and matches the size of the filter. The following figure explains the implicit function of the generated filter across the input to provide what is known as output map:



Pooling Layer: A pooling layer is operated for down sampling the output of a convolutional layer generated as a feature map by providing the crux of the prevalent features of the feature map, referred to as *f*local translational invariance≈. The below-given figure explains the example for pooling of stride 2*2, MP represents the Maximum Pooling and AP represents the Average Pooling:

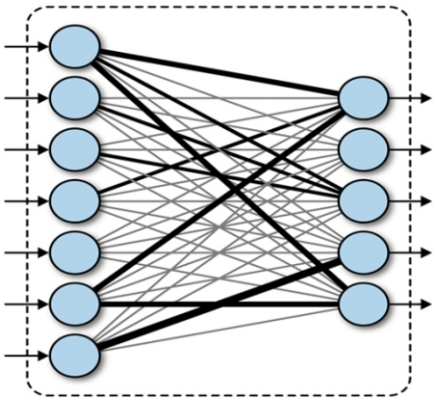


These two approaches for down sampling are briefly discussed below:

Maximum Pooling: Maximum pooling or also referred to as "Max Pooling≈ operates by selecting the maximal activation from the feature map, presumably providing the most prevalent features.

Average Pooling: Average Pooling operates by computing the average of the activations present in the feature map.

Fully Connected Layer: This layer in the CNN model has all inputs of the preceding layer connected to each activation unit of the succeeding layers. These layers prepare data for the output layer by compiling the extracted data from the previous layers (Lu et al., 2016). The following representation shows a fully connected layer:



Output layer: This layer is responsible for producing the result, neurons are bespoke for generating a streamlined output from the unordered data received through the iterative operations of training by coalescing the inputs received.

Research simulation

By definition, Simulation is a study or teaching approach in industry, science, and education that replicates actual events and processes under test settings. Creating a simulation is frequently a difficult mathematical task. Initially, rules, relationships, and operational processes, as well as other factors, are described. The combination of these phenomena results in new circumstances, and even new laws, which evolve further as the simulation progresses.

This simulation provides the solution to the problem statement, as to predict the vessel type by the means of image classification. This follows a structured approach first to satisfy the prerequisites of the problem statement followed by the problem-solving and implementation.

Prerequisites

To perform the standard problem-solving methodology, some dependencies need to be satisfied before proceeding. These dependencies are essential for the successful onset of the classification, referred to as program dependencies. In this study, python 3 being open source is used as the choice of programming language to implement various operations. Python provides extensive support for object-oriented programming as well as functional programming as a high-level programming language. To perform the several operations for the simulation, python libraries are used. A library is a collection of various classes supporting several functions intensively, providing extensive implementation into the simulation. There are multiple libraries which are required as dependencies for the operability of the simulation, provided to be imported into the simulation using *f*import≈, are briefly discussed below:

NumPy: Numerical Python said NumPy provides the support for multidimensional arrays and matrices in python, created as a community project in 1995 (Guide to Numpy, n.d.). All the NumPy objects are operable under advanced mathematical operations for matrix manipulation, providing the

data structure known as an "array≈, for an n-dimensional array(matrix).

Matplotlib: Matplotlib provides visualization support for the Python programming language with the initial release (Droettboom et al., 2003; Rosner, 1978). It provided an object-oriented application programming interface for incorporating the visualizations. Support for various visualizations is provided by this library as bar graphs, histograms, scatter plots, and 3-Dimensional visualizations are also supported.

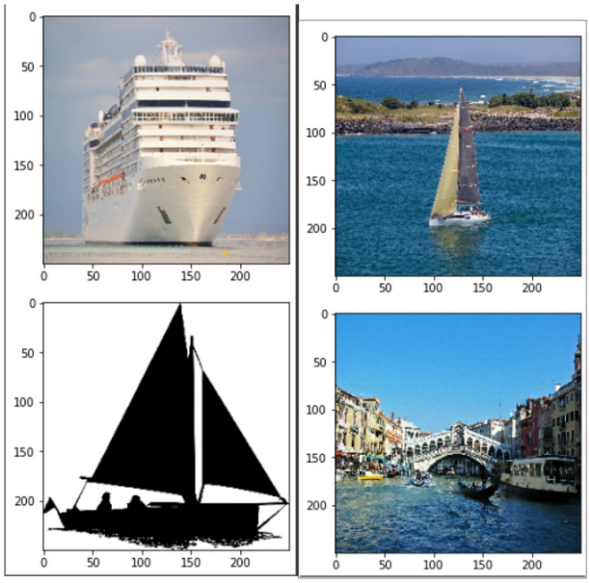
TensorFlow: TensorFlow is an open-source library, developed by Google Brain Team. It provides intensive support for machine learning and artificial intelligence, training support for neural networks, and operating on differential programming.

Keras: Keras is an open-source library, It provides the requisites of neural network modelling such as neural network layers, optimizers, and image and text processing tools for deep learning while acting as an interface for TensorFlow (Parisi et al., 2021; Jacot et al., 2020).

OpenCV: Open Source Computer Vision Library, referred to as OpenCV was developed by the Intel Corporation, It provides real-time image processing, with hardware acceleration using CUDA and OpenCL-based Graphics Processing Units.

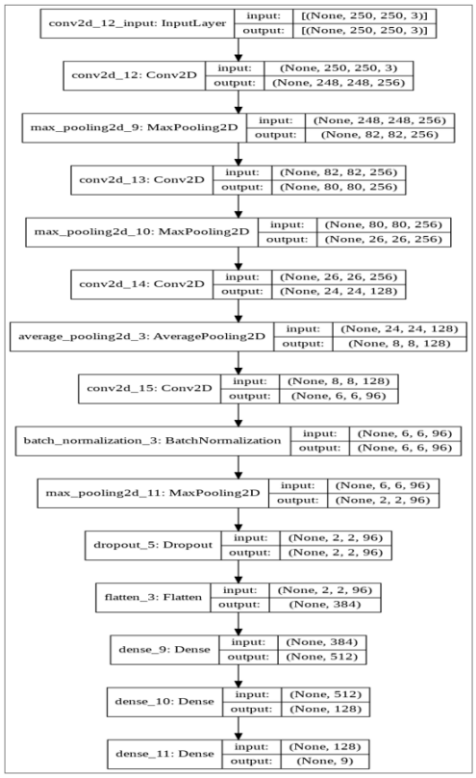
Data Preparation

Respecting the problem statement the data set is loaded into the main memory for the operations required to perform the task of image classification. The data is loaded using the Image Data Generator from the "preprocessing" class of the "Keras" library. The loaded data was reported to have 1462 images belonging to 9 classes namely "buoy", "gondola", "inflatable boat", "kayak","cruise ship", "ferry boat", "freight boat", "paper boat" and "sailboat". In this case, the use of the ImageDataGenrator() function is used to build training data (Guo et al., 2016; Ciresan et al., 2012). To make the best use of the available computational power, training images are required to be scaled to the size by passing the target size to a size of about 250 pixels by 250 pixels. This works on top of all the images found in the indices resizes them regardless of their default size, and uploads images in batches of 32 objects. Image filtering is used to sort the pixel list of numbers from 1 to 255; as default, each pixel is represented by 3 channels R for the "Red", G for the "Green" and B for the "Blue". In matrix representation, the size of each channel pixel is measured between values 0 to 255. Therefore, to further reduce computational requirements, pixel matrices are redeemed by a factor of 1/255 per channel. The posting of a few sample images is shown below:



Architectural development

The onset of image classification commences from the development of a convolutional neural network, by importing layers from the "layers" class of the "Keras" library. The layers are arranged sequentially, meaning one after another subsequently. This paradigm results in 1,267,113 trainable parameters and 192 non-trainable parameters for the model training. The following stack explains the sequential arrangement of the developed model:



Layer 0, Input Layer, Convolutional Layer 1: The convolutional layer for processing 2-dimensional images is referred to as Conv2D, also used as the input layer for this model with the input dimensions provided as 250 pixels by 250 pixels with 3 channels as kernel size, feature extraction was done using 256 filters and activation function is used as "ReLU", accounting for 7168 parameters.

Layer 1, Hidden Layer 1, Pooling Layer 1: This is the pooling layer used for Maximum Pooling with a pool size of 3*3 and stride size of 3, accounting for 0 parameters.

Layer 2, Hidden Layer 2, Convolutional Layer 2: This is a convolutional layer for image processing with subsequently 256 filters with a kernel size of 3 and activation function as "ReLU", accounting for 590080 parameters.

Layer 4, Hidden Layer 4, Convolutional Layer 3: This is a convolutional layer for image processing with subsequently 128 filters with a kernel size of 3 and activation function as "ReLU", accounting for 295040 parameters.

Layer 5, Hidden Layer 5, Pooling Layer 3: This is the pooling layer used for Average Pooling with a pool size of 3*3 and stride size of 3, accounting for 0 parameters.

Layer 6, Hidden Layer 6, Convolutional Layer 4: This is a convolutional layer for image processing with subsequently 256 filters with a kernel size of 3 and activation function as "ReLU", accounting for 110688 parameters.

Layer 7, Hidden Layer 7, Normalization Layer 1: Batch normalization performed using this layer for standardizing the inputs in mini-batches for the subsequent layers, helps in the stabilization of the learning process and reduces the number of required epochs, accounting for 384 parameters.

Layer 8, Hidden Layer 8, Pooling Layer 4: This is the pooling layer used for Maximum Pooling with a pool size of 3*3 and stride size of 3, accounting for 0 parameters.

Layer 9, Hidden Layer 9, Dropout Layer 1: For classification error has to be reduced, for which the dropout layer is added before the output layer with a rate of 0.1, to improve the generalization by deactivating the neurons. This layer is implemented after the pooling layers for the generation of image noise augmentation, accounting for 0 parameters.

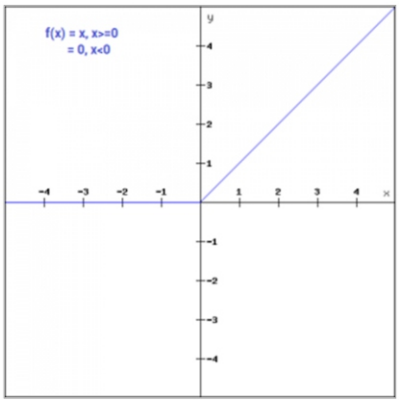
Layer 10, Hidden Layer 10, Flatten Layer 1: The flattening layer is used to reshape the preceding tensor to a single dimension, removing all the other dimensions present in the output from the preceding layers, accounting for 0 parameters.

Layer 11, Hidden Layer 11, Fully Connected Layer 1: This is a "Dense" layer responsible for compiling the data from the proceeding layers, provided with 512 units and "ReLU" as an activation function, accounting for 197120 parameters.

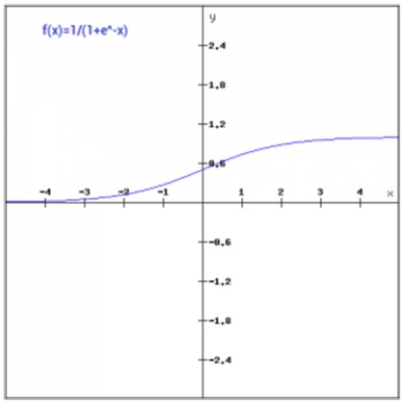
Layer 12, Hidden Layer 12, Fully Connected Layer 2: This is a "Dense" layer responsible for compiling the data from the proceeding layers, provided with 128 units and "ReLU" as an activation function, accounting for 65664 parameters.

Layer 13, Output Layer, Fully Connected Layer 3: This is a "Dense" layer responsible for compiling the data from the proceeding layers, provided with 9 units and "Sigmoid" as an activation function. This is functions as an output layer with each unit responsible for each class, provided with 9 classes for classification, accounting for 1161 parameters. The activation functions used in several layers are as follows:

ReLU: A nonlinear activation function which provides output greater than 0, if less than 0 it transforms it into 0 resulting in the deactivation of the neuron in the network. It can be said as f(x)=max(0,x)



Sigmoid: This function performs the transformation over the input providing the output between the range of 0 and 1. It can be said as f(x) = 1/(1+e^-x).

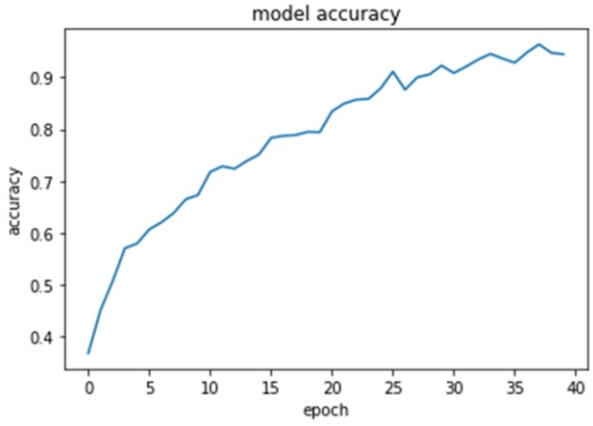


Model Compilation

Upon successfully adding the sequential layers, the model is compiled with an optimizer known as "Adam" and the loss function as "categorrical_crossentropy". Optimizers are implemented here in order to alter the attributes like the learning rate of the neural network for a reduction in loss. The optimizer implemented here, Adam, learns adaptively for the learning rate using the squared gradients for the learning rate and the moving average of the gradient. It computes the learning rates individually for various parameters, thus named adaptive moment estimation or adam. The Cross-Entropy loss function measures the model's performance for the output, given the probability varies between 0 and 1.

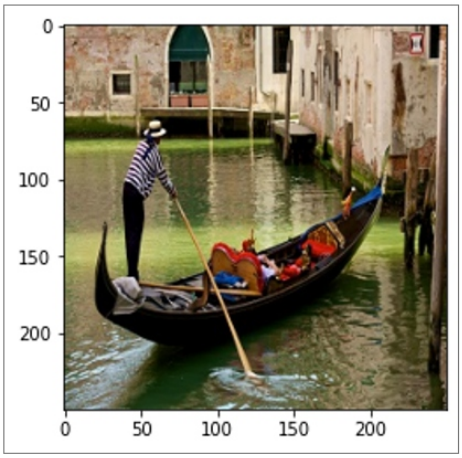
(Result) Model Training

The model is then trained using the training data for 40 epochs using fit_generator() resulting in 94.46% of accuracy over the training data, defining the number of total predictions made correctly. It is the ratio of the correct predictions to the total number of predictions made. The following graph describes the growth for accuracy:



Prediction

The trained model is later on used to predict a test image to check the relevance of the prediction made over an RGB image as shown below:



Then, successfully predicted the test image as “Gondola”.

The prediction is : gondola

Interference: The inference from this simulation is drawn that the convolutional neural network could be used for image classification tasks provided the input data is noise free. Noise in the input data could result in a degraded performance in classification tasks as well as failure to train properly in real-time.

Conclusion

By means of extensive research simulation, a classification task for the provided problem statement was successfully performed. The provided data consisted of intensive background noise as a backdrop which was tackled using batch normalization in the model training resulting in better operability and accurate predictions. The training of the model resulted in an accuracy score of 94.46% for the .jpg images present in the dataset in the RGB colour channel. Using the GPUs for computational requirements and stacking multiple convolution and pooling layers resulted in better accuracy, aided further by the dropout layer to reduce computational requirements."

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET), pp. 1-6. Antalya, Turkey. doi: 10.1109/ICEngTechnol.2017.8308186.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. IEEE Transactions on Neural Networks and Learning Systems, 33(12), 1-21.

Lo, S.-C. B., Chan, H.-P., Lin, J.-S., Li, H., Freedman, M. T., & Mun, S. K. (1995). Artificial convolution neural network for medical image pattern recognition. Neural Networks, 8(7-8), 1201-1214.

Traore, B. B., Kamsu-Foguem, B., & Tangara, F. (2018). Deep convolution neural network for image recognition. Ecological Informatics, 48, 257-268.

Kamath, U., Liu, J., & Whitaker, J. (2019). Convolutional Neural Networks. Springer Ebooks, 263-314.

Bayar, B., & Stamm, M. C. (2016). A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security.

Lu, Y., Zhu, S.-C., & Wu, Y. (2016). Learning FRAME Models Using CNN Filters. Proceedings of the ... AAAI Conference on Artificial Intelligence, 30(1).

Oliphant, T. E. (2006). Guide to numpy (Vol. 1, p. 85). USA: Trelgol Publishing.

Droettboom, M., MacMillan, K., & Fujinaga, I. (2003, April). The Gamera framework for building custom recognition systems. In Symposium on Document Image Understanding Technologies (pp. 275-286).

Rosner, R. A. (1978). Computer software. Nature, 274(5670), 516-516.

Gehler, P., & Nowozin, S. (2009, September). On feature combination for multiclass object classification. In 2009 IEEE 12th International Conference on Computer Vision (pp. 221-228). IEEE.

Bi, J., Zhang, T., & Bennett, K. P. (2004). Column-generation boosting methods for mixture of kernels.

Kandola, J., Shawe-Taylor, J., Cristianini, N., Kandola, J., Shawe-Taylor, J., & Cristianini, N. (2002). Optimizing Kernel Alignment over Combinations of Kernel. Eprints.soton.ac.uk.

Parisi, L., Ma, R., RaviChandran, N., & Lanzillotta, M. (2021). hyper-sinh: An accurate and reliable function from shallow to deep learning in TensorFlow and Keras. Machine Learning with Applications, 6, 100112.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. Neurocomputing, 187, 27-48.

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. 2012 IEEE Conference on Computer Vision and Pattern Recognition.

Parisi, L., Ma, R., RaviChandran, N., & Lanzillotta, M. (2021). hyper-sinh: An accurate and reliable function from shallow to deep learning in TensorFlow and Keras. Machine Learning with Applications, 6, 100112.

Jacot, A., Gabriel, F., & Hongler, C. (2020). Neural Tangent Kernel: Convergence and Generalization in Neural Networks. ArXiv:1806.07572 [Cs, Math, Stat].

Ahmed, S., & Islam, S. (2023). Methods in detection of median filtering in digital images: a survey. Multimedia Tools and Applications, 82(28), 43945-43965.

Birajdar, G. K., & Mankar, V. H. (2013). Digital image forgery detection using passive techniques: A survey. Digital Investigation, 10(3), 226-245.