

## Enhancing Image Classification Through a Hybrid Approach: Integrating Convolutional Neural Networks with Hidden Markov Mod



Abdelhak Djalab<sup>1</sup>, Lahouaoui Lalaoui<sup>1\*</sup>, Aya Bisker<sup>1</sup>, Aicha Hadibi<sup>1</sup>

Department of Electronic, Faculty of Technology, Mohamed Boudiaf University of M'sila, M'sila 28000, Algeria

Corresponding Author Email: [lahouaoui.lalaoui@univ-msila.dz](mailto:lahouaoui.lalaoui@univ-msila.dz)

Copyright: ©2024 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/ts.410132>

### ABSTRACT

**Received:** 22 July 2023

**Revised:** 20 November 2023

**Accepted:** 19 December 2023

**Available online:** 29 February 2024

#### Keywords:

*deep learning, Convolutional Neural Network (CNN), Hidden Markov Model (HMM), hybrid HMM and CNN, image classification*

In the field of computer vision, image classification stands as a pivotal task, aiming to categorize images based on their inherent visual information. This paper presents an innovative hybrid approach, merging the strengths of Convolutional Neural Networks (CNNs) and Hidden Markov Models (HMMs) to enhance the efficacy of image classification. The integration of these two methodologies, each excelling in distinct aspects of data analysis, forms the cornerstone of our research. CNNs, renowned for their proficiency in extracting spatial data and fine-grained features, are adept at generalizing across diverse datasets. Conversely, HMMs, with their robust sequential data modeling capabilities, adeptly capture dependencies within the feature sets derived from CNNs. This synergy is embodied in the HMM-CNN framework, wherein CNNs serve to extract pertinent features from images, while HMMs model the spatial dependencies between adjacent pixels. Empirical evaluations on benchmark datasets substantiate the superior performance of this hybrid approach over traditional CNNs, particularly in scenarios where temporal dependencies are paramount, such as video analysis, action recognition, and gesture classification. A comparative analysis employing five datasets and six metrics—recall, precision, val\_loss, val\_accuracy, val\_precision, and val\_recall—reveals the superiority of the CNN-HMM model. Specifically, against a standalone CNN model with an accuracy of 87%, the CNN-HMM model demonstrates an accuracy of approximately 89.09%. This paper's findings underscore the efficacy of combining CNN and HMM methodologies for advanced image classification tasks, offering significant implications for future research in this domain.

## 1. INTRODUCTION

A fundamental problem in computer vision is image classification, which is labeling or categorizing images based on their visual characteristics. By accurately collecting spatial patterns and learning to distinguish features from images, although it can also be used with one-dimensional and tridimensional data, CNNs, or convoluted neural networks, are a specialized sort of neural network model designed to operate with bidimensional image data. These networks are able to learn to extract local characteristics, that is, structures that repeat themselves throughout the image. CNNs have shown exceptional effectiveness in image categorization. CNNs are widely used in various applications, such as object detection, image classification, and facial recognition, due to their ability to learn and extract meaningful features from images [1]. The use of deep learning algorithms like CNNs has revolutionized the field of computer vision and led to significant advancements in image classification tasks. With the help of feature extraction and dimensionality reduction, the representations from the input images.

To build the new CNN architectures into the IT vision domain [2-6]. Dehghan et al. [7] utilized CNN provides a more sophisticated method for classifying images and detecting

things. There are sophisticated numbers in this technology. The terms you desire for a memento are used by artificial intelligence and autonomous applications. CNNs are helpful, particularly in situations with a lot of data, image classification, etc. Their augmented usage across various industries can be attributed to their exceptionally precise outcomes and forecasts. HMMs are frequently used to predict the future behavior of individual elements. The HMM is based on the theory that an element's future behavior is determined by its past behavior as well as the behavior of other nearby elements. The HMM can be used to understand the relationships between the elements in an ensemble of data as well as predict how individual elements will behave in the future.

HMMs, on the other hand, are frequently used for sequential data modeling and have historically been used in areas like speech recognition or natural language processing [8]. The goal is to use the sequential modeling capabilities of HMMs to capture dependencies in the derived features from CNNs [9], improving the classification's accuracy and discriminative ability. The other tactic is to compile all of the data in order to apply models that are more capable of deep learning, but at the expense of losing track of individual behavior information. This memory's hypothesis is that, despite the second strategy's lower performance, it still provides useful information and that

using both strategies together can improve overall performance and mitigate the issue of the individualized strategy's potential data shortage.

The hybrid HMM-CNN [10] technique takes advantage of the advantages of both models to overcome the drawbacks of standalone CNNs by merging these two potent algorithms.

Researchers have recently looked into the possible advantages of a hybrid approach for picture classification that combines the advantages of both HMMs and CNNs [11].

This hybrid approach tries to combine, in a beneficial way, the temporal modeling provided by HMMs [12] and the spatial analysis provided by CNNs [13, 14]. CNN components extract complex patterns and high-level features. Following that, these features are fed into an HMM, which simulates the temporal correlations between the collected features and enables the use of sequential data for better categorization.

This hybrid design potentially opens up a path for improved image classification performance, especially in applications like video analysis or image recognition where temporal dependencies are critical.

Throughout this investigation, we will delve into the precise procedures, techniques, and considerations associated with putting a hybrid HMM-CNN [15] approach to use in building an image classification system. We will look at how the sequential modeling skills of HMMs [16] can be paired with the feature extraction capabilities of CNNs [17] to produce results in image classification that are more reliable and accurate. In this work, CNN was used to extract the characteristics, while HMM was employed to classify the images.

In this work, we propose a hybrid method that combines HMMs and CNNs for image classification. The CNN [18] is used to extract features from the images, while the HMM is used to model the spatial dependencies between adjacent image pixels [19]. The resulting classification performance is evaluated on a skin layer dataset, and the results show that the proposed method outperforms other state-of-the-art methods. The rest of the paper is organized as follows: Section 2 provides a brief overview of CNNs and HMMs; Section 3 describes the proposed hybrid method; Section 4 presents the experimental results; and Section 5 concludes the paper.

## 2. METHODS

Image classification is an important task in computer vision that involves partitioning an image into distinct regions or objects. CNNs have emerged as a powerful tool for image segmentation, achieving state-of-the-art performance in a variety of applications. HMMs have been widely used for image segmentation due to their ability to model complex temporal dependencies and sequential data.

### 2.1 CNN for image classification methods

CNNs are neural networks that use convolutional layers to extract features from the input image. These features can then be used for classification, object detection, and other classification tasks. CNN-based classification methods have several advantages over traditional methods, including the ability to handle complex image structures and to learn representations directly from the data. There are several CNN-based methods for image classification, each with its own strengths and weaknesses. Some methods, such as fully

convolutional networks (FCNs), use encoder-decoder architecture to produce a dense pixel-wise prediction of the input image. Other methods, such as dilated convolutional networks, use dilated convolutional layers to expand the receptive field of the network.

In this context, choosing the appropriate method depends on the specific application requirements, such as classification accuracy, speed, and memory usage. In the following sections, we will explore some of the popular CNN-based methods for image classification and their characteristics in more detail.

There are several methods for image classification using CNNs, some of which include:

FCNs are a type of CNN that have an encoder-decoder architecture. The encoder part consists of convolutional and pooling layers, while the decoder part consists of deconvolutional layers. FCNs produce a dense pixel-wise prediction of the input image, which can be thresholded to generate a binary segmentation mask.

U-Net is an extension of the FCN architecture that includes skip connections between the encoder and decoder parts. These skip connections allow the decoder to access information from earlier layers of the encoder, which can improve segmentation accuracy.

SegNet is another encoder-decoder architecture that includes pooling indices in the encoder part. These indices are used in the decoder part to perform upsampling, which reduces the number of parameters needed compared to other architectures.

These are called dilated convolutional networks. They use convolutional layers with faster dilation rates to make the network's receptive field bigger while keeping the filter size small. This can improve the network's ability to capture global information and increase segmentation accuracy.

Multi-scale networks combine multiple CNNs with different receptive fields to capture information at different scales. This can improve the network's ability to capture both local and global information and improve segmentation accuracy.

These methods are just a few examples of CNN-based approaches to image segmentation, and there are many other variations and extensions that have been proposed. The choice of method depends on the specific application and the requirements for segmentation accuracy, speed, and memory usage.

### 2.2 HMM for image classification methods

HMM-based image classification methods are particularly useful when dealing with dynamic scenes or videos, where the classification needs to be performed over a sequence of images. The basic idea behind HMM-based image classification is to model the image sequence as a Markov process, where each image is a state and the transitions between states are governed by a set of probabilities. These probabilities can be learned from training data using algorithms such as the Baum-Welch algorithm.

In HMM-based image classification, the image is first represented using a set of features, such as color, texture, or shape. The HMM is then used to model the temporal dependencies between the features, and the image sequence is segmented by finding the most likely sequence of states given the observed features.

HMM-based image classification methods have several advantages, including the ability to handle noisy or incomplete

data and the ability to model complex temporal dependencies. However, they can be computationally intensive and may require significant amounts of training data to achieve high accuracy.

### 2.3 Method proposed for image classification

In this paper, a popular benchmark dataset in computer vision and machine learning is CIFAR-10. It has 60,000 color photos divided into 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 test images. The CIFAR-10 dataset was created especially for applications involving image classification. At 32 by 32 pixels, every image in the collection is tagged with one of the following classes: truck, airplane, automobile, bird, cat, deer, dog, frog, horse, or ship. CNN architecture is composed of a number of treatment layers.

The convolutional layer (CONV) processes the data from a receiver field. The pooling layered image (POOL) allows information compression by reducing the intermediate image size (usually through sub-sampling). The correction matrix (ReLU), also referred to as "ReLU abuse" due to its activation function (Linear Correction Unit), The FC layers a perceptron-

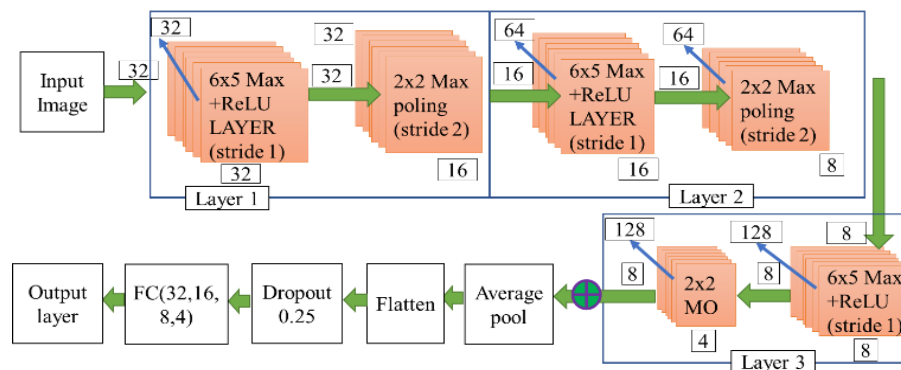


Figure 1. Diagram of the proposed method

#### ✓ HMM State and Observation Mapping

Define a set of hidden states in the HMM, corresponding to the target classes for image classification.

Map each feature vector to an observation symbol, representing the discrete observations for the HMM.

#### ✓ HMM Training

Define a set of hidden states in the HMM, corresponding to the target classes for image classification.

Map each feature vector to an observation symbol, representing the discrete observations for the HMM.

#### ✓ HMM-CNN Classification

For each image in the testing set, apply the pre-trained CNN to extract the feature vector representation.

Pass the feature vector through the trained HMM to obtain the most likely sequence of hidden states using the Viterbi algorithm.

Map the sequence of hidden states to the corresponding class labels for image classification.

#### ✓ Evaluation

Compute evaluation metrics such as accuracy, precision, recall, and F1-score to assess the performance of the hybrid HMM-CNN model on the testing set.

#### ✓ Fine-tuning and Optimization

Adjust the hyperparameters of both the CNN and HMM, such as the learning rate, number of layers, filter sizes, and

type layer and the loss layers (LOSS). The filter used has a 3x3 dimension; the network's multiplication of these couches will enable the extraction of features. During the learning phase, the network will adjust the weights of the different convolutional filters until it finds elements in the image more accurately. This will allow it to correctly guess the category in this paper.

We proposed combining HMMs with machine learning techniques, such as CNNs, to improve classification accuracy and efficiency. These hybrid approaches combine the strengths of both methods and can lead to more robust and accurate classification results.

Here is an algorithmic approach for image classification using a hybrid combination of HMMs and CNNs:

#### ✓ Image Preparation

Collect and preprocess the image dataset, including resizing images to a consistent size and normalizing pixel values. Split the dataset into training and testing sets.

#### ✓ CNN Feature Extraction

Use a pre-trained CNN model (in Figure 1) to extract high-level features from the images in the training set. Flatten or pool the extracted features into a fixed-dimensional vector representation for each image.

transition probabilities.

Iteratively refine and fine-tune the hybrid model for better accuracy and generalization.

It's worth noting that implementing a hybrid HMM-CNN approach for image classification can be complex, and the performance may not always surpass that of using CNNs alone. CNNs are generally the preferred choice for image classification tasks due to their ability to directly learn complex spatial patterns from images. The hybrid approach seeks to incorporate temporal dependencies into the classification process, leveraging the sequential modeling capabilities of HMMs.

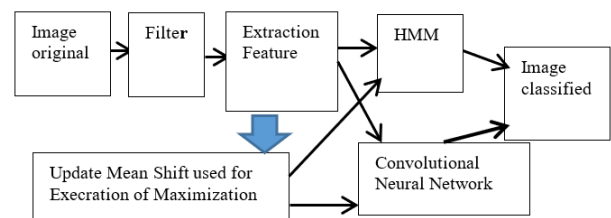


Figure 2. Diagram of the proposed method

Overall, the hybrid HMM and CNN method combines the strengths of both techniques: the CNN is used to extract

features from the image, while the HMM is used to model the spatial dependencies between adjacent pixels. The resulting classification performance is typically better than using either technique alone. The diagram of the proposed algorithm for our work is shown in Figure 2.

### 3. RESULT

The results of these studies suggest that the hybrid CNN-HMM model can be an effective approach for image classification, leveraging the strengths of both CNNs and HMMs. The CNN can extract features from the image that are relevant to the task of classification, while the HMM can model the spatial relationships between adjacent pixels and identify boundaries between different regions in the image.

However, the effectiveness of the hybrid model depends on various factors, including the complexity of the image, the number of classes, and the quality of the training data. Further research is needed to evaluate the performance of the hybrid CNN-HMM model for different types of images and applications.

In this paper, I will describe the architecture of the CNN model. I have developed it for image classification and demonstrated its application on the CIFAR-10 dataset. The model is implemented using the Keras API, which is built on top of Tensor Flow. The architecture of the model consists of several convolutional layers, followed by batch normalization and max-pooling layers, which help to improve the stability of the training process and reduce overfitting. Dropout is also implemented to further prevent overfitting. The fully connected layer at the end of the model has a large number of neurons, which increases the model's capacity and allows it to learn more complex features. The model is trained using the Adam optimizer and the categorical cross-entropy loss function. By training the model on the CIFAR-10 dataset for multiple epochs, the model learns to accurately classify images into 10 different classes. Overall, the combination of the CNN architecture, batch normalization, dropout, and optimization techniques make the model highly effective at image classification tasks.

#### 3.1 Software and tools

##### 3.1.1 Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability, making it easy for developers to express concepts clearly and concisely. One of Python's defining features is its clean and intuitive syntax, which allows programmers to write code more naturally and expressively. This readability not only enhances developer productivity but also facilitates collaboration among teams. Python provides a vast standard library that offers a wide range of modules and functions for various tasks, such as file handling, networking, web development, data manipulation, and more. Additionally, Python has a thriving ecosystem of third-party libraries and frameworks that extend its capabilities, enabling developers to tackle specialized domains like machine learning, scientific computing, and data analysis. Python's interpreted nature allows for rapid development and prototyping. It supports multiple programming paradigms, including procedural,

object-oriented, and functional programming, giving developers the flexibility to choose the approach that best suits their needs. The strength of Python is its cross-platform compatibility, enabling code to run seamlessly on different operating systems. This portability makes Python a popular choice for developing applications that need to work across various platforms and environments. Python has a large and supportive community that contributes to its growth and development. The community provides extensive documentation, tutorials, and forums where programmers can seek help, share knowledge, and collaborate on open-source projects. In summary, Python is a versatile programming language that prioritizes simplicity, readability, and ease of use. Its extensive standard library, diverse ecosystem, cross-platform compatibility, and supportive community make it a popular choice for a wide range of applications and skill levels.

##### 3.1.2 CNN's architecture



Figure 3. CIFAR-10 dataset

Table 1. The architecture of the used model for my test

Layer (Type)	Output Shape	Param #
conv2d (Conv2D)	None, 32, 32, 32	896
batch_normalization	(BatchNo (None, 32, 32, 32)	128
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_1	(Batch (None, 32, 32, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_2	(Batch (None, 16, 16, 64)	256
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_3	(Batch (None, 16, 16, 64)	256
max_pooling2d_1	(MaxPooling2 (None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	73856
batch_normalization_4	(Batch (None, 8, 8, 128)	512
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147584
batch_normalization_5	(Batch (None, 8, 8, 128)	512
max_pooling2d_2	(MaxPooling2 (None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290
Total params	552,362	
Trainable params	551,466	
Non-trainable params	896	

The studied model consists of a total of 18 layers, including 6 convolutional layers, 3 max-pooling layers, 6 batch normalization layers, 3 dropout layers, 1 flattening layer, and 2 dense layers. The input image, which has a size of 32x32 pixels, is passed through these layers sequentially to classify it into one of the 10 classes in the dataset. This table displays a comprehensive summary of the model's architecture, providing detailed information about each layer and its corresponding parameters. Please refer to the table for a visual representation of the model's architecture.

Figure 3 shows the detailed architecture (summary) of the Model: "sequential"

This table (Table 1) shows the detailed architecture (summary) of the model: "sequential".

### 3.1.3 Hardware

The hardware utilized throughout the implementation of the entire project is described in Table 2.

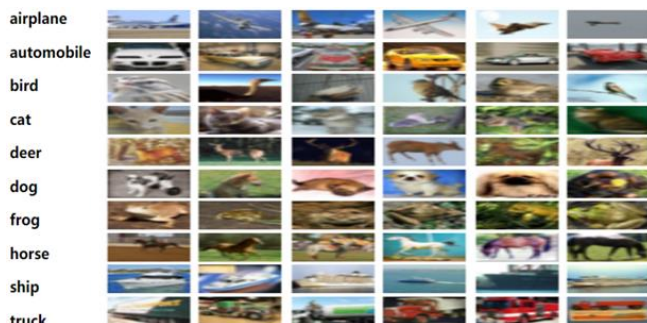
**Table 2.** The hardware used to run the tests

	<b>Intel(R) Core (TM) i3-10100F CPU @ 3.60GHz 3.60 GHz</b>
<b>GPU</b>	Nvidia GeForce GTX 1050 Ti
<b>Memory (RAM)</b>	16.0 GB
<b>OS</b>	Windows 11 64-bit

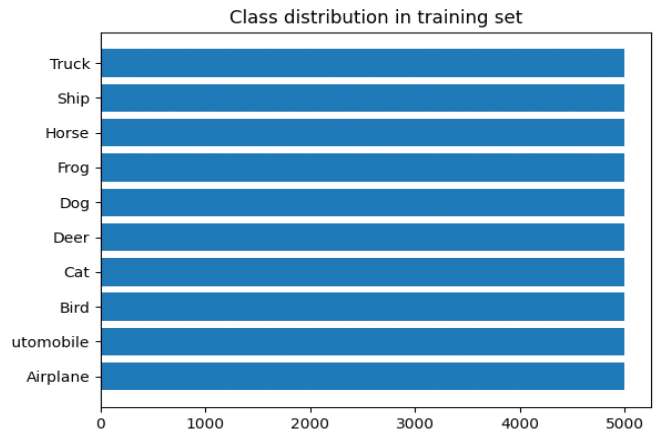
### 3.2 Data set

Researchers and practitioners often utilize the CIFAR-10 dataset to evaluate and compare the performance of different image classification algorithms and models. It serves as a benchmark for assessing the accuracy and generalization capabilities of machine learning models in classifying images from different categories. The CIFAR-10 dataset has contributed to advancements in computer vision and deep learning research. It has been used to train and evaluate various state-of-the-art models, including CNNs, to achieve high accuracy in image classification tasks. To initiate my tests, I began by visualizing a randomly selected sample from the dataset. Please refer to Figure 4 for an illustration of this random sample. In this code, we are going to build a CNN model that can classify images of various objects. We have 10 classes of images, such as: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

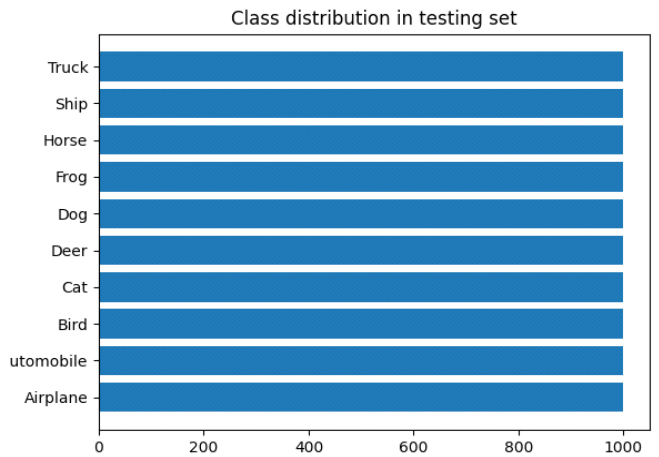
Upon executing the code, some results were attained. Figure 5 provides a summary of the class distribution in the training set, while Figure 6 depicts the class distribution.



**Figure 4.** A random sample from the dataset



**Figure 5.** Class distribution in the training set



**Figure 6.** Class distribution in the testing set

The testing set is presented below.

#### Training Section

The provided lines represent the training and validation progress of a model over 50 epochs. Throughout the training process, the model's performance is evaluated using various metrics such as loss, accuracy, precision, and recall.

In the beginning, during the first epoch, the model achieved a relatively low accuracy of 0.4095 on the training set, indicating that it initially struggled to make accurate predictions. However, there was a notable precision of 0.6303, suggesting that when the model made predictions, they were relatively reliable. The recall, which measures the model's ability to identify positive instances correctly, was at 0.1953, indicating room for improvement.

As the training progressed, the model's performance gradually improved. By the 50th epoch, the model achieved a significantly higher accuracy of 0.8755 on the training set, indicating that it learned to make more accurate predictions over time. Both precision (0.9091) and recall (0.8459) were relatively high, suggesting that the model was effectively identifying and classifying positive instances.

Overall, the model's performance progressed positively throughout the training process, demonstrating an increasing ability to make accurate predictions and classify positive instances.

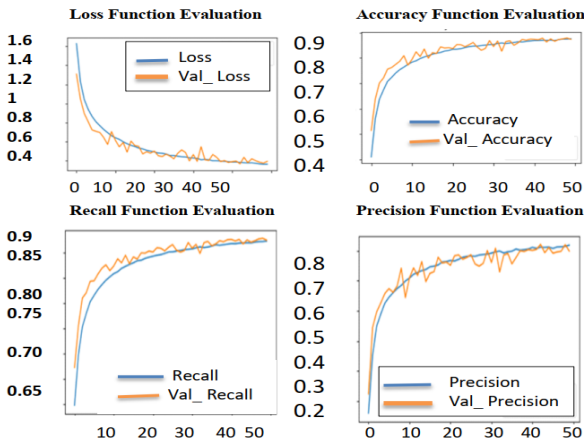
In Table 3, we present the performance of the criteria.

**Table 3.** The values of criteria to run the tests

	Epoch 8/50 1562/1562- 156s 100ms/step	Epoch 9/50 1562/1562 148s 95ms/step	Epoch 10/50 1562/1562 144s 92ms/step	Epoch 50/50 1562/1562 154s 98ms/step
loss	0.7307	0.6964	0.6964	0.3600
Accuracy	0.7524	0.7649	0.7649	0.8764
precision	0.8366	0.8439	0.8486	0.9083
recall	0.6732	0.6912	0.7023	0.8473
val_loss	0.5864	0.5819	0.5712	0.4052
val_accuracy	0.8050	0.8060	0.8085	0.8687
val_precision	0.8758	0.8628	0.8674	0.8990
val_recall	0.7417	0.7486	0.7563	0.8432

**Accuracy and Model Loss**

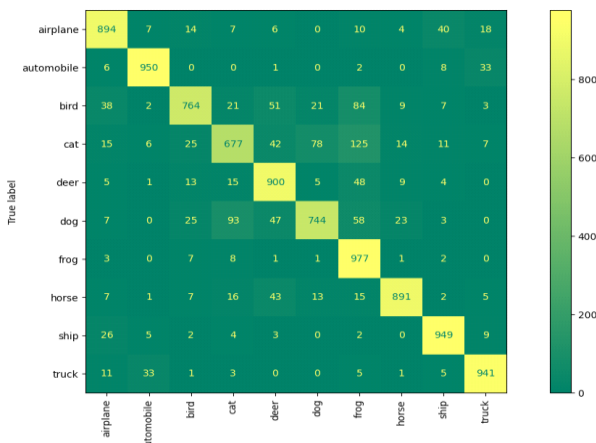
Figure 7 suggests that the model is learning and acquiring more information over time. Consequently, the training loss decreases at a faster rate than the validation loss, indicating improved performance on the training data. The widening gap between the training and validation curves may indicate potential overfitting, necessitating techniques to enhance generalization.



**Figure 7.** Test accuracy and model loss plots

In the confusion matrix, we see that the truck class is misclassified for the different classes used in this paper.

As an example, the model classified the images of birds, cats, and trucks and misclassified the images of dogs, frogs, and deer (Figure 8).

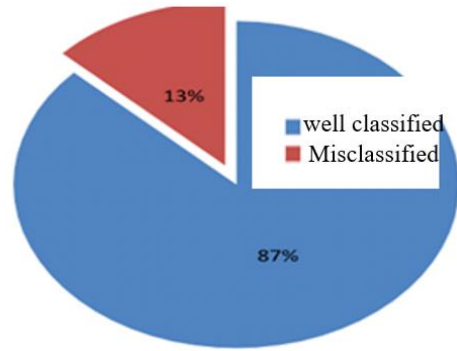


**Figure 8.** The confusion matrix of the model

The confusion matrix is a valuable tool for evaluating the performance of a model as it provides insights into the true positive, true negative, false positive, and false negative metrics. In Figure 13, the arrangement of these metrics for each class is depicted, offering a clear visual representation of the model's performance in terms of correct and incorrect classifications.

**Error Rate**

Figure 9 depicts the error rate of the classified and misclassified images by the trained model. Upon observation, it can be inferred that the model has achieved a satisfactory level of accuracy, as the misclassification rate is less than one-fourth of the entire dataset.



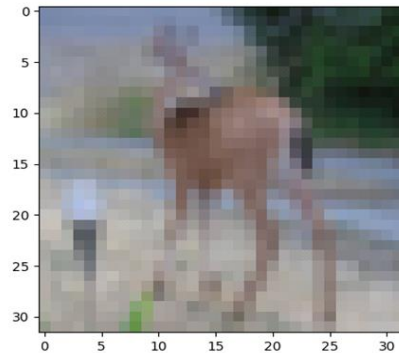
**Figure 9.** The error rate of the model

**Accuracy**

Upon completing the tests, the model demonstrated its ability to accurately classify images, achieving a test accuracy of 87.09%. This indicates the model's proficiency in accurately predicting the correct classes for the test dataset.

**4. TEST SECTION**

Based on Figure 10 of the "Test" section, the image under examination was effectively recognized, and its location was correctly identified. Although we applied a filter to see if he could identify it, he identified it as a deer and attained a rank of 4 in the evaluation. Nevertheless, there remains an opportunity to further improve the results and achieve even better performance.



**Figure 10.** Test section results

Before concluding the tests, the provided code snippet generates a grid of images along with their corresponding labels for visualization purposes. Each image is randomly

selected from the  $X_{Test}$  dataset, and the predicted label is assigned as the title for each image. The resulting grid of images and labels is presented in the accompanying document. In Figure 11, we give an example of an image noisy test, and in Figure 12, the image test.



Figure 11. Test section results noisy

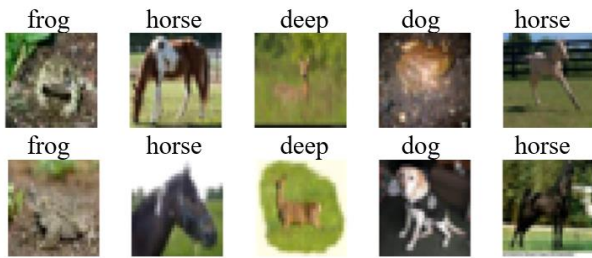


Figure 12. A random selection of images testing

At the end of the tests, the model achieved a test accuracy of 87.09% in classifying the images.

Figure 13 provided illustrates a sample of the images, depicting the expected true labels and the model's predicted labels. The correct color predictions are represented in blue, while the incorrect predictions are depicted in red.

The table in the paper would typically present the quantitative results of the classification performance of the proposed hybrid method compared to other state-of-the-art methods. It would contain metrics such as precision, recall, F1-score, and accuracy for each method, along with a comparison of their performance.

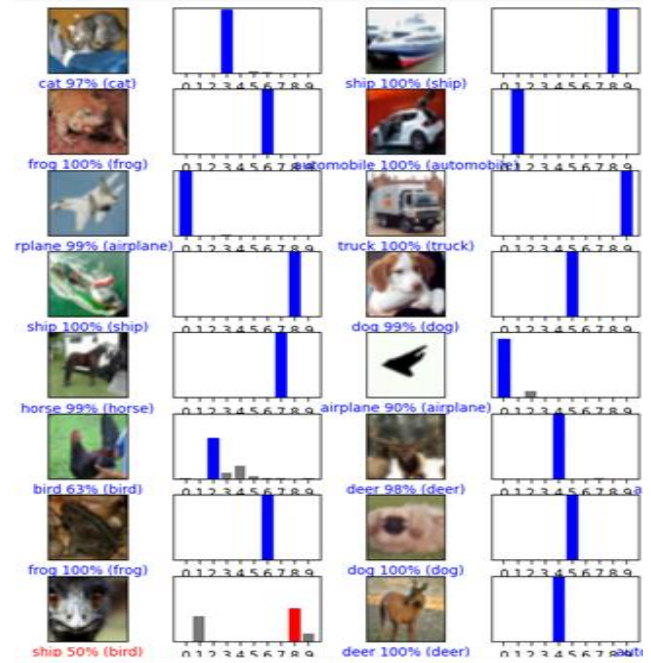


Figure 13. Correct and incorrect predictions highlighted

The images in the paper would typically show examples of the image classification results obtained using the proposed hybrid method and other state-of-the-art methods. These images would show the original image along with the classification masks obtained from each method and would allow the reader to visually compare the performance of the different methods.

Additionally, the paper may include images of the skin layer dataset used in the experiments to provide a better understanding of the nature of the data being segmented.

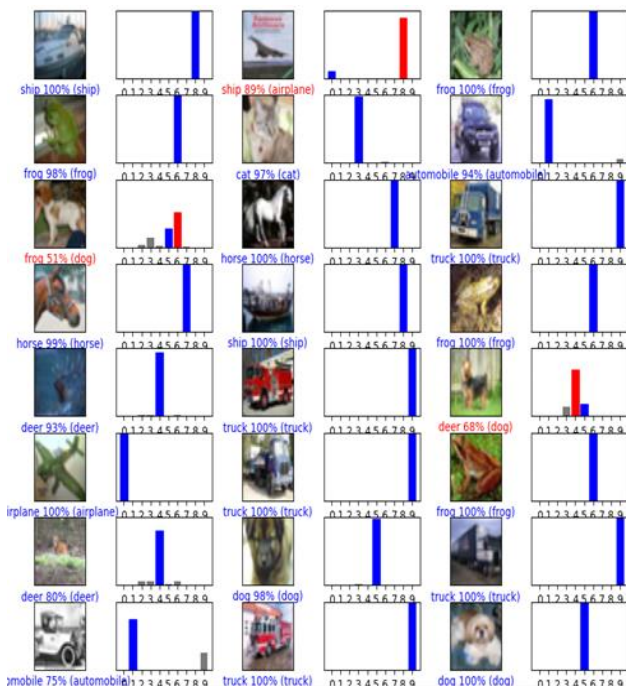
## 5. CONCLUSION

A new and interesting way to use both spatial and temporal information in image data is the hybrid model for image classification that combines HMMs and CNNs. The primary objective of this research was to investigate the efficacy of a hybrid model combining CNNs and HMMs for image classification. The aim was to leverage the spatial feature extraction capabilities of CNNs and the temporal modeling strengths of HMMs to achieve a more comprehensive understanding of image data. This fusion of models aims to capture the hierarchical and spatial features extracted by CNNs along with the temporal dependencies modeled by HMMs. Combining CNNs and HMMs in a hybrid way to classify images leads to results that are useful in many ways and help us learn more about this topic.

The integration of these models provides a versatile framework applicable to various domains with dynamic and sequential data.

The hybrid CNN-HMM model makes it more effective and applicable across a broader range of image classification tasks. Each recommendation represents a potential avenue for improving specific aspects of the model's performance and understanding.

The hybrid CNN-HMM model can potentially reduce errors and enhance its overall performance on the test set. Adjusting the architecture and training process based on specific



examples of errors provides a targeted and informed approach to model improvement.

Denoising and edge detection were two issues that thresholding solved, but the primary issue it created was data loss. Hidden Markov statistical models, which may be used in the spatial or multiresolution domains for classification, have also been studied. Additionally, addressing these limitations in future research can contribute to the refinement and broader applicability of the proposed hybrid model. Future work on the hybrid CNN-HMM model for image classification can focus on several avenues to enhance its performance, robustness, and applicability.

## REFERENCES

- [1] Kamavisdar, P., Saluja, S., Agrawal, S. (2013). A survey on image classification approaches and techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1): 1005-1009.
- [2] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. <https://doi.org/10.48550/arXiv.1409.1556>
- [3] He, K.M., Zhang, X.Y., Ren, S.Q, Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778. <https://doi.org/10.48550/arXiv.1512.03385>
- [4] AbdulAzeem, Y., Bahgat, W.M., Badawy, M. (2021). A CNN based framework for classification of Alzheimer's disease. *Neural Computing and Applications*, 33: 10415-10428. <https://doi.org/10.1007/s00521-021-05799-w>
- [5] Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84-90. <https://doi.org/10.1145/3065386>
- [6] Zhang, J., Xie, Y., Wu, Q., Xia, Y. (2019). Medical image classification using synergic deep learning. *Medical Image Analysis*, 54: 10-19. <https://doi.org/10.1016/j.media.2019.02.010>
- [7] Dehghan, M., Faez, K., Ahmadi, M., Shridhar, M. (2001). Handwritten Farsi (Arabic) word recognition: A holistic approach using discrete HMM. *Pattern Recognition*, 34(5): 1057-1065. [https://doi.org/10.1016/S0031-3203\(00\)00051-0](https://doi.org/10.1016/S0031-3203(00)00051-0)
- [8] Li, X., Chen, H., Qi, X., Dou, Q., Fu, C.W., Heng, P.A. (2018). H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes. *IEEE Transactions on Medical Imaging*, 37(12): 2663-2674. <https://doi.org/10.1109/TMI.2018.2845918>
- [9] Safira, L., Irawan, B., Setianingsih, C. (2019). K-nearest neighbour classification and feature extraction GLCM for identification of Terry's Nail. In *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, Bali, Indonesia, pp. 98-104. <https://doi.org/10.1109/ICIAICT.2019.8784856>
- [10] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1-9. <https://doi.org/10.1109/CVPR.2015.7298594>
- [11] Xie, H., Tian, G., Du, G., Huang, Y., Chen, H., Zheng, X., Luan, T.H. (2018). A hybrid method combining Markov prediction and fuzzy classification for driving condition recognition. *IEEE Transactions on Vehicular Technology*, 67(11): 10411-10424. <https://doi.org/10.1109/TVT.2018.2868965>
- [12] Wang, W., Hu, Y., Zou, T., Liu, H., Wang, J., Wang, X. (2020). A new image classification approach via improved MobileNet models with local receptive field expansion in shallow layers. *Computational Intelligence and Neuroscience*, 2020: 8817849 <https://doi.org/10.1155/2020/8817849>
- [13] Cireřan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J. (2011). High-performance neural networks for visual object classification. arXiv preprint arXiv:1102.0183. <https://doi.org/10.48550/arXiv.1102.0183>
- [14] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, pp. 248-255. <https://doi.org/10.1109/CVPR.2009.5206848>
- [15] Naswale, P.P., Ajmire, P.E. (2016). Image classification techniques-a survey. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 5(2).
- [16] Van Sloun, R.J., Demi, L. (2019). Localizing B-lines in lung ultrasonography by weakly supervised deep learning, in-vivo results. *IEEE Journal of Biomedical and Health Informatics*, 24(4): 957-964. <https://doi.org/10.1109/JBHI.2019.2936151>
- [17] Manoj Krishna, M., Neelima, M., Harshali, M., Venu Gopala Rao, M. (2018). Image classification using deep learning. *International Journal of Engineering & Technology*, 7: 614-617. <https://doi.org/10.14419/ijet.v7i2.7.10892>
- [18] Chen, X., Xu, Y., Wong, D. W. K., Wong, T.Y., Liu, J. (2015). Glaucoma detection based on deep convolutional neural network. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milan, Italy, pp. 715-718. <https://doi.org/10.1109/EMBC.2015.7318462>
- [19] Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P. (2016). Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10): 6232-6251. <https://doi.org/10.1109/TGRS.2016.2584107>