Neural Computing and Applications (2023) 35:17987–18002
https://doi.org/10.1007/s00521-023-08644-4

ORIGINAL ARTICLE

A new hybrid model of convolutional neural networks and hidden Markov chains for image classification

Soumia Goumiri[1,2] · Dalila Benboudjema[1] · Wojciech Pieczynski[3]

Received: 10 November 2022 / Accepted: 2 May 2023 / Published online: 31 May 2023

Abstract
Convolutional neural networks (CNNs) have lately proven to be extremely effective in image recognition. Besides CNN, hidden Markov chains (HMCs) are probabilistic models widely used in image processing. This paper presents a new hybrid model composed of both CNNs and HMCs. The CNN model is used for feature extraction and dimensionality reduction and the HMC model for classification. In the new model, named CNN-HMC, convolutional and pooling layers of the CNN model are applied to extract features maps. Also a Peano scan is applied to obtain several HMCs. Expectation–Maximization (EM) algorithm is used to estimate HMC's parameters and to make the Bayesian Maximum Posterior Mode (MPM) classification method used unsupervised. The objective is to enhance the performances of the CNN models for the image classification task. To evaluate the performance of our proposal, it is compared to six models in two series of experiments. In the first series, we consider two CNN-HMC and compare them to two CNNs, 4Conv and Mini AlexNet, respectively. The results show that CNN-HMC model outperforms the classical CNN model, and significantly improves the accuracy of the Mini AlexNet. In the second series, it is compared to four models CNN-SVMs, CNN-LSTMs, CNN-RFs, and CNN-gcForests, which only differ from CNN-HMC by the second classification step. Based on five datasets and four metrics recall, precision, F1-score, and accuracy, results of these comparisons show again the interest of the proposed CNN-HMC. In particular, with a CNN model of 71% of accuracy, the CNN-HMC gives an accuracy ranging between 81.63% and 92.5%.

Keywords Convolutional neural networks (CNNs) · Hidden Markov chains (HMCs) · Deep learning · Image classification

✉ Wojciech Pieczynski
Wojciech.Pieczynski@telecom-sudparis.eu

Soumia Goumiri
s_goumiri@esi.dz

Dalila Benboudjema
d_benboudjema@esi.dz

[1] Laboratoire des Méthodes de Conception de Systèmes (LMCS), Ecole nationale Supérieure d'Informatique (ESI), BP, 68M Oued-Smar, 16270 Alger, Algeria

[2] CERIST, Centre de Recherche sur l'Information Scientifique et Technique, Ben Aknoun 16030, Algeria

[3] SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France

1 Introduction

A Deep Neural Network is composed of a set of neurons grouped in layers connected to each other. There are three types of layers based on their functions: input layer, hidden layers, and output layer. The input layer is connected to the first hidden layer, and the last hidden layer is connected to the output layer. Each neuron applies the following equation when receiving an input x to produce an output y: $y = f(xW + b)$, where W is called the weights matrix and b is called the bias. The objective is to tune some of the parameters to minimize the error between the produced output and the expected value.

In neural networks there is a category called "deep learning model" in which the network is combined by more than three layers, i.e., it contains more than one hidden layer. Convolutional Neural Network (CNN) is a deep learning model used for image classification. In this model we apply filters to extract the features of the images then classify those features. CNN is widely used for image and video classification in several fields such as medical applications, transportation systems, agriculture, manufacturing, etc. Some examples are the diagnosis of breast cancer using mammogram images [1], the annotation of breast cancer images [2], brain tumor segmentation [3], COVID-19 diagnosis using X-Ray images [4], Alzheimer's disease detection [5], patterns of cystic fibrosis [6], pedestrians' detection [7], moving object detection [8], deep fakes in videos [9], face recognition [10], parking occupancy detection [11–13], and recognition of fire base on video [14]. CNN is also used for scene classification using a deep attention CNN [15], semantic correspondence [16, 17], and select of interest [18]. A fundamental stage in such algorithms is feature extraction. Feature extraction from pictures entails extracting a small number of features from low-level image pixel values that include many items or scene information therefore, capturing the differences between the object categories [19].

Markov chains are probabilistic models that proved their interest in image processing. Various models based on Markov chains have been proposed, among them is the hidden Markov chains (HMCs) known be to very efficient in signal processing, examples are speech recognition ([20–23]) or image processing ([24–28], among others). Other applications, such as genome analysis, prediction in economics and finance, environment, and meteorology, are also commonly used. Their success is due to their ability of processing "large" amounts of data. We assume we have access to a noisy version of a signal modeled by a Markov chain, and the challenge is to estimate the chain's unobservable realization.

CNN models are the preferred models for image classification tasks. But they need a large amount of data to be trained and provide height accuracy. In addition to the huge number of parameters generated by these models, the training of this amount of data needs a powerful GPU and RAM. A commonly used solution is transfer learning, where models are trained and weights are saved for later use; however, the problem still persists. Hybrid models using pre-trained CNN for feature extraction can reduce training time and provide more accurate results than a single model. CNNs have been combined with several machine learning models such as Long short-term memory (LSTM), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Random Forest (RF), and multi-Grained Cascade Forest (gcForest). Such combinations have improved the accuracy of the prediction but have increased the number of parameters. To overcome this, the idea is to use pre-trained CNN models for feature extraction and another model for classification with a reduced number of parameters.

We propose a new model based on the hybridization of the CNN model and the HMC one. We call this model CNN-HMC. It uses CNN for feature extraction and HMC for classification. The objective is to enhance the performances of the CNN models for the image classification task, while reducing considerably the number of model parameters. To assess the performance of the proposed model we apply it to a classification problem of cats/dogs, and show its interest with respect to two classic CNN models. The first one is composed of four convolutions (4Conv) and the second one is mAlexNet [29]. To evaluate the performance of our proposal, we have carried out two series of experiments on five datasets [30–34].

The remainder of this paper is organized as follows. In Sect. 2, we present related work. In Sect. 3, we recall some definitions related to the CNNs, the Hilbert-Peano curve, and the HMCs. Section 4 describes our approach. We provide two study cases in Sect. 5 and 6, respectively. In Sect. 7, we report the experiments and discuss the results. Finally, we conclude our paper in Sect. 8.

2 Related work

Features are parameters or characteristics that enable recognition of different items of an image or a video. Feature extraction is an important task for image classification. Traditional feature extraction methods are exhausting and time consuming. Convolutional neural networks (CNNs) can replace traditional feature extractors since they are significantly more effective and have a great ability to retrieve complicated characteristics that express the image in a deeper level. The use of CNN for feature extraction has been coupled with other machine learning models such as SVM, and RNN.

The authors in [35] and [36] have proposed the use of CNN for feature extraction and Support Vector Machine (SVM) for classification. The hybrid model was named CNN-SVM. The SVM classifier is applied to the last layer of the fully connected layers of the CNN model instead of the activation function. The SVM model was used with the "rbf" kernel. The CNN-SVM was tested on the MNIST dataset for handwritten digits recognition.

The CNN-SVM model was also used in different applications such as: recognizing patterns in knee movement using mechanomyography data [37], Brain tumors and MRI image classification [38], grapevine leaves classification [39], detection of cervical cancer cells [40], classification for Remote Sensing Data [41], human Activity Recognition [42] and classification for weed recognition in winter rape field [43].

CNN was hybridized with RNN models such as LSTM and GRU. Karimi et al. [44] propose the use of CNN-

LSTM model to classify nuclear atypia in breast cancer images. In this work, the results of the fully connected layer by the CNN is feed to the two-layer LSTM model. Precision, specificity, recall and F-score were used to evaluate the performance of the proposed model. The overall accuracy of the system is computed by varying the patch size of the input images between 64, 128, 227, and 344. The CNN accuracy was 84.72% against 86.67% of the CNN-LSTM model.

CNN-LSTM was also applied in different fields such as sentiment Analysis [45], predicting residential energy consumption [46], gold price time-series forecasting [47], speech emotion recognition using deep 1D & 2D CNN LSTM [48], human Activity Recognition [49], detection of diabetes using CNN and CNN-LSTM network and heart rate signals [50], and forecasting monthly gas field production [51].

CNN-GRU, and CNN-RNN work the same way as the CNN-LSTM model. CNN-GRU was applied to forecast short-term electricity consumption in [52], water level [53], activity recognition [54] and [55], soil moisture [56], PM2.5 concentration in urban environment [57], traffic speed prediction [58], ship motion [59], etc. CNN-RNN was used to classify fruit in [60], diagnosis of COVID-19 [61], medical recommendation system [62], sentiment analysis [63], emotion recognition [64], Fake news detection [65], multiple people tracking [66], crop yield prediction [67], etc.

Hamidi et al. [68] proposed a multi-stage architecture that uses CNN, Beta-Elliptic Model (BEM) for features extraction, and Deep Bidirectional Long Short Term Memory (DBLSTM) and SVM networks for classification. CNN works on offline data, while BEM extracts visual characteristics of online data. Since the features extracted by CNN are inexpressive, the K-means algorithm clustered them into k groups. The k groups are classified using fuzzy classification. The previous step results are fed to two DBLSTMs networks for training. The final output was obtained by applying the SVM classifier to the results of the two DBLSTMs. The proposed multi-stage architecture is applied to multilingual online handwriting recognition.

In [69], the authors proposed a multi-level fusion classifier framework. It involves five steps: data collection, features preparation, training of multiple classifiers, primary fusion, and final fusion. In the features preparation stage, the LeNet-5 CNN model is used for feature extraction and a collection-based algorithm to reduce feature size. The two feature sets are trained using KNN and multiple decision trees. The results of the decision trees are gathered in a random forest, and the outputs are fused with KNN results to get a secondary ensemble. The outcome is obtained by combining the secondary ensemble of the two sets of features. The proposed ensemble learning approach was tested on the MNIST dataset.

Xu et al. combined CNN and RF in [70]. The CNN architecture is composed of two convolutional layers, two pooling layers and a fully connected layer. Three different RF are applied after both of the pooling layers and the fully connected layer. The final output is obtained by combining the outcomes of the three RF using ensemble learning. In [71], the authors use a pre-trained CNN model for feature extraction. For each feature map they applied four classifiers: random forest, gcforest, SVM, and LSTM. The comparison of the four hybrid models reveal that CNN-SVM and CNN-RF give higher accuracy for bearing fault classification.

All the works mentioned above have improved the prediction accuracy. However, the hybridization models need to be trained sequentially to tune the parameters of the combined models to get the best performance. In addition, combining two or more models increases the number of parameters, hence the need for compute resources to manage them.

3 Background

3.1 The convolutional neural networks (CNNs)

Convolutional neural networks are among the most used models for image classification. In numerous situations, they predict the image class with a great precision. CNN is an operational class of models for better comprehension of the information contained in an image, leading in improved image identification, segmentation, detection, and retrieval [19]. A CNN model is composed of an input layer, convolution layers, pooling layers, and fully connected layer.

- Input layer: It represents the first layer of a CNN model. Images in this layer must have the same size. They are passed to a convolutional layer of feature extraction.
- Convolution layer: The following layers are "Convolution layers" which function as image filters, allowing to extract features from pictures and calculate match feature points during testing.
- Pooling layer: After that, the extracted feature sets are sent to the "pooling layer." This layer reduces the size of large images while keeping the most critical information. It maximizes the value of each window by preserving the optimum fit of each feature within the window.
- Fully connected layer: This is the last layer in the CNN model. It takes the high-level filtered pictures and transforms them into categories with labels.

CNN models are also composed of an activation function and a loss function. The former is applied to convolution layers and to the fully connected layer. While the latter is applied to the output to measure how much the predicted value match with the real one. Rectified Linear Units (ReLU) is the most used activation function. It replaces every negative value in the pooling layer with 0. This keeps learned values from becoming stuck at 0 or ballooning out toward infinity, allowing the CNN to remain mathematically stable.

3.2 The Hilbert-Peano Curve

A space-filling scan allows converting a 2D or 3D matrix to a one-dimensional vector, which is needed to use Markov chain models. Among the space-filling scans, we pick the Hilbert-Peano scan. This scan is constructed using four patterns as shown in Fig. 1. The interest of Markov chain methods for image segmentation with respect to 2D Markov field models is that, being based on 1D modeling, they lead to significantly lower computational cost. However, consideration of contextual information is less satisfactory: two neighbors in the chain are neighbors in the grid, but two neighbors in the grid can be distant in the chain. In fact, the Hilbert-Peano path preserves the neighborhood in the 1D vector, as well as possible [26]. This characteristic makes it useful in multidimensional signal processing. In particular, with the rapid development of digital image processing, the Hilbert curve, as a scanning technique, is widely applied in digital image processing [72].

The Hilbert-Peano curve is adapted to images of size 2^p whereas in our case the volumes on which we apply the Peano path are of arbitrary size. To overcome this problem we use the solution proposed in [72]. Given a rectangular matrix of size $n \times m$. This solution aims at finding the curve that corresponds to a matrix of size 2^{order} such as:

2^{order} is the smallest value $> max(n, m)$.

Then we eliminate the $2^{\text{order}} - (n \text{ x } m)$ extra cells. In Fig. 2, we illustrate the Peano scan applied to matrices of size 2 × 2, 4 × 4, and 8 × 8, respectively. In Fig. 3, we show the adopted scan used for matrices of arbitrary size.

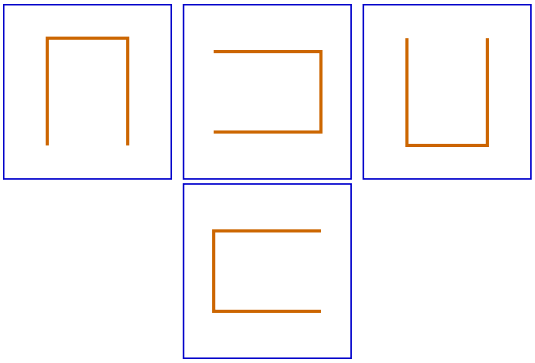

Fig. 1 The four patterns for the construction of the Peano curve, the first on the left is the base pattern and the others are obtained by rotation $(\pi, \frac{\pi}{2} and \frac{3\pi}{2})$, respectively, of the latter

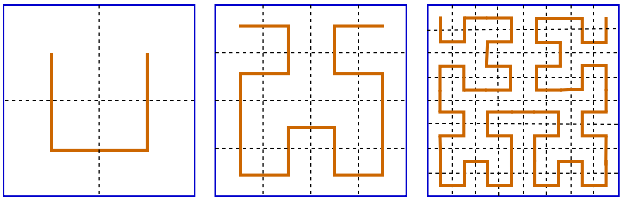

Fig. 2 The Peano scan for matrices of size two, four, and eight, respectively

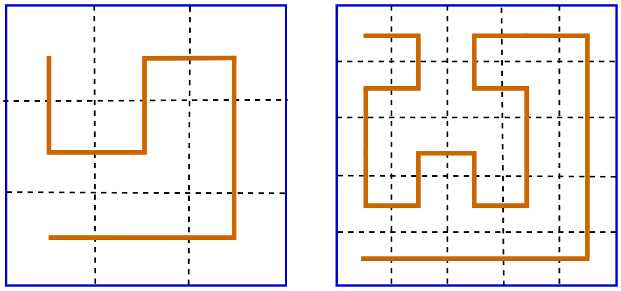

Fig. 3 The Peano scan used when the size of the matrix is not a power of two, the first example on the left is for matrix of size three, and the second example on the right is for matrix of size five

In the figure, we give two examples for matrices of size 3 × 3, and 5 × 5. We note that this scan is also applied for non-squared images. In our case, all the manipulated images are square, in other words, they are of size $n \times n$.

3.3 The hidden Markov chains (HMCs)

We consider that we have access to a noisy version of the signal modeled by a Markov chain, and the general problem is that of estimating the unobservable realization of the chain.

We consider stochastic processes $\mathrm{X} = X_{1:N}$ and $\mathrm{Y} = Y_{1:N}$. Y is observed and X is not. Each X_n takes its values in the finite set of K classes $\Omega = \{w_1, ..., w_K\}$ and each Y_n takes its values in the set of real numbers $\mathbb{R}$. Dependence oriented graph of HMC with $N = 3$ is given in Fig. 4.

In classic HMC (X, Y), we consider X as Markov with the distribution

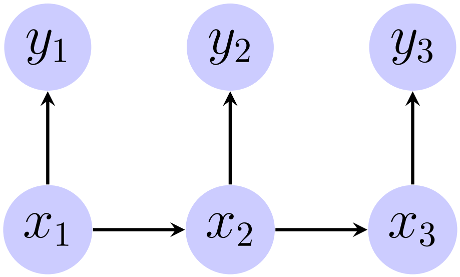

Fig. 4 Dependence oriented graph of Hidden Markov Chain

$$p(x) = p(x_1)\prod_{n=1}^{N-1} p(x_{n+1}|x_n), \tag{1}$$

and the distribution $p(y|x)$ is defined with

$$p(y|x) = \prod_{n=1}^{N} p(y_n|x_n), \tag{2}$$

where $p(y_n|x_n)$ are assumed Gaussian. The Bayesian Marginal Posterior Mode (MPM) we use for estimation $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_N)$ of X from Y is defined with

$$\begin{aligned} &For\ each\, n = 1, \ldots, N, \\ &\hat{x}_n = \underset{w_k}{argmax}\, p(x_n = w_k|y) = \underset{w_k}{argmax}\, \alpha_n(k)\beta_n(k), \end{aligned} \tag{3}$$

where the forward probabilities $\alpha_n(k) = p(x_n = w_k, y_1, \ldots, y_n)$ and the backward ones $\beta_n(k) = p(y_{n+1}, \ldots, y_N|x_n = w_k)$ are computed recursively with following forward and backward recursions:

$$\begin{aligned} &\alpha_1(k) = p(x_1 = w_k, y_1); \\ &\alpha_{n+1}(k) = \sum_k p(x_{n+1}|x_n)p(y_{n+1}|x_{n+1})\alpha_n(k); \end{aligned} \tag{4}$$

$$\begin{aligned} &\beta_N(k) = 1; \\ &\beta_n(k) = \sum_k p(x_{n+1}|x_n)p(y_{n+1}|x_{n+1})\beta_{n+1}(k); \end{aligned} \tag{5}$$

In the homogeneous case that we will consider in this paper, the distributions $p(x_{n+1}|x_n)$ and $p(y_n|x_n)$ don't depend on n. Then the parameters defining $p(x, y) = p(x)p(y|x)$ are the parameters defining $p(x_1)$, $p(x_2|x_1)$ and $p(y_1|x_1)$. For $k = 1, \ldots, K$, they will be denoted with $\pi_k = p(x_1 = w_k), a_{ij} = p(x_2 = w_j|x_1 = w_i)$. Gaussian $p(y_1|x_1 = w_k)$ are of means μ_k and variances σ_k^2. To make MPM (3) unsupervised, we estimate all parameters from $Y = y$ using the classic "Expectation-Maximization" (EM) method. EM produces a sequence of parameters in the following way. For $k, i, j = 1, \ldots, K$, let $\pi_k^q, a_{ij}^q, \mu_k^q, \sigma_k^{2,q}$ be the current parameters. Setting

$$\begin{aligned} &\Psi_n^q(i,j) = \\ &\frac{\alpha_n^q(i)a_{ij}^q p^q(y_{n+1}|x_{n+1} = w_j)\beta_{n+1}^q(j)}{\sum_{i=1}^K \alpha_n^q(i)[\sum_{j=1}^K a_{ij}^q p^q(y_{n+1}|x_{n+1} = w_j)\beta_{n+1}^q(j)]} \end{aligned} \tag{6}$$

$$\xi_n^q(i) = \sum_{j=1}^{K} \Psi_n^q(i,j), \tag{7}$$

parameters are updated with

$$\pi_k^{q+1} = \frac{1}{N}\sum_{n=1}^{N} \xi_n^q(k); \qquad a_{ij}^{q+1} = \frac{\sum_{n=1}^{N-1} \Psi_n^q(i,j)}{\sum_{n=1}^{N-1} \xi_n^q(i)}; \tag{8}$$

$$\mu_k^{q+1} = \frac{\sum_{n=1}^{N} \xi_n^q(k)y_n}{\sum_{n=1}^{N} \xi_n^q(k)}; \tag{9}$$

$$\sigma_k^{2,q+1} = \frac{\sum_{n=1}^{N} \xi_n^q(k)(y_n - \mu_k^{q+1})^2}{\sum_{n=1}^{N} \xi_n^q(k)} \tag{10}$$

Initialization and criterion for stopping iterations depend on the particular case studied.

4 New hybrid CNN-HMC approach

We place ourselves in a classification task where the problem is to find the class of the input image. We consider the case of two possible classes $\Omega = \{w_1, w_2\} = \{0, 1\}$.

The new model we propose, called CNN-HMC, uses CNN for feature extraction, and HMC for classification. It extends any CNN. The input of CNN-HMC is an RGB image of the same size as the input of the CNN model. We first apply a combination of convolution layers, pooling layers, and possibly dropout layers to extract the features of the image. The output of this operation is a volume of size (x, y, h), where x and y are the dimensions of each feature and h is the number of features. Then we apply the Peano scan to obtain h hidden Markov chains. We consider that the features are independent of each other, so that the h HMCs obtained are considered independently. After having h HMCs, we use the MPM to estimate each value of the h hidden chains. We choose the dominant class in each of the h chains, then we take the dominant class in the h classes obtained. We estimated the parameters of each of the h chains with EM described in the previous section. In Fig. 5, we illustrate the architecture of the CNN-HMC model.

The classification algorithm based on CNN-HMC is given in Algorithm 1 below.

Algorithm 1 The algorithm of the CNN-HMC model

```
Ensure: Classification of the input image
Extracting features with CNN
while number of features do
    Apply the Peano scan
    Estimate parameters with EM
    Find feature classification with MPM
    Find the dominant class of each feature
end while
Find the dominant class of the input
Return the result.
```

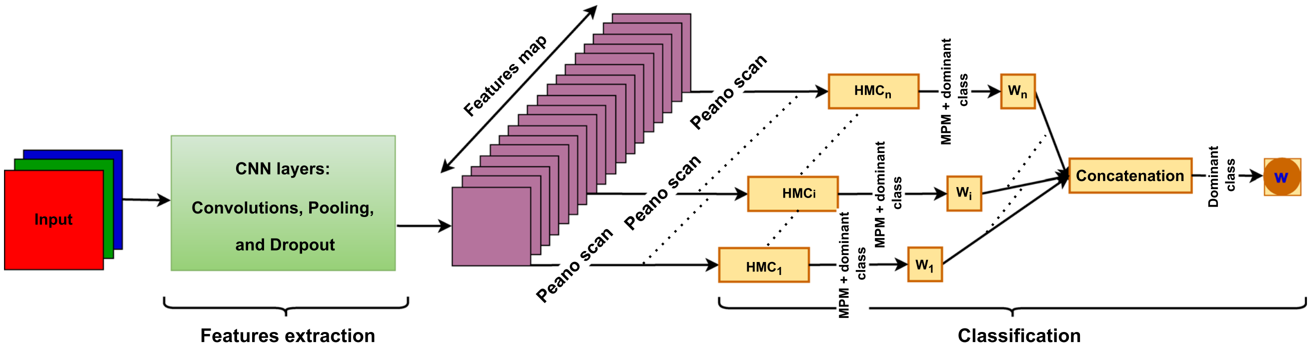

Fig. 5 The architecture of the CNN-HMC model

5 Models description

5.1 4Conv description

We consider the following CNN model, which we will refer to as 4Conv. This model takes as input, RGB images of size 64×64. It is composed of four blocks of convolutions, max-pooling, and dropout layers. Each convolution uses a kernel of size 3×3 and filters of size 16, 32, 32, and 64, respectively. We apply to each convolution layer the function ReLU as an activation function. The Max-pooling layer uses windows of size 2×2. The dropout takes as a parameter the value 0.5 which means that 50% of neurons will be eliminated from this layer and only the 50% of the neurons that remain will send their values to the next layer. The result of the four blocks is a volume of size 2×2×64 representing the features map where each sub-matrix of size 2×2 is a characteristic of the input image. The features map is flattened using a sequential scan to have a column vector. The latter squeezed the entrance to a classical neural network for classification. This network takes 256 neurons as input, has two hidden layers, and produces two outputs. The first hidden layer has 128 neurons and the second hidden layer has 64 neurons. The hidden layers use the activation function ReLU, while the output layer uses the softmax function.

We illustrate in Fig. 6 the convolutional neural network architecture, 4Conv, which we have proposed with its two parts: feature extraction and classification. In Fig. 7, we show summary of the 4Conv model.

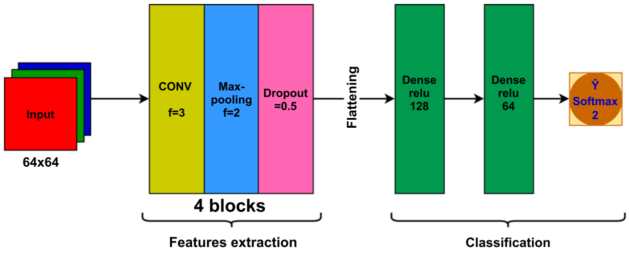

Fig. 6 The architecture of the considered CNN model, 4Conv

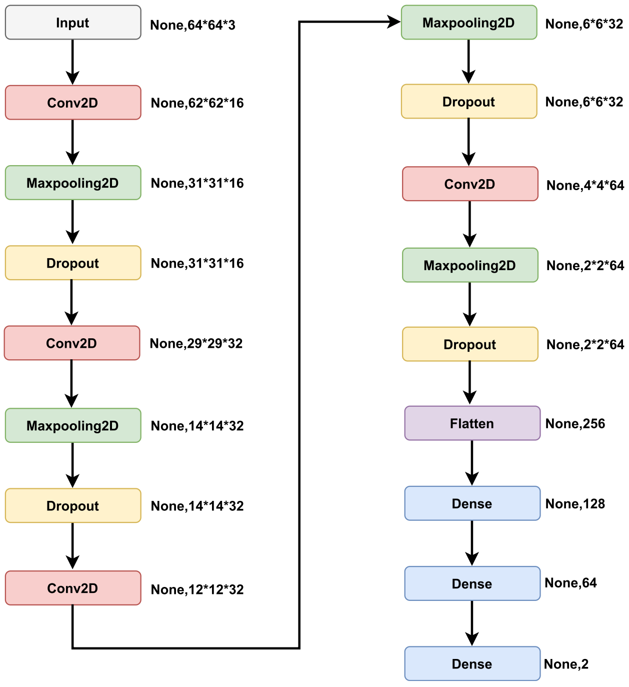

Fig. 7 Summary of the considered CNN model, 4Conv, with input and output size of each layer

5.2 CNN-HMC for 4Conv

The CNN-HMC we propose takes the feature extraction part of CNN networks then applies the Peano path to obtain a hidden Markov chain and at the end finishes the classification with the MPM.

We use the 4Conv model described in the previous section for feature extraction. After that, we take the output just before applying the flattening that is the matrix of characteristics and which is a volume of size (2, 2, 64). This means that we have 64 characteristics where each is a square matrix of size 2. For each matrix of size (2, 2), we apply the Peano path to obtain a hidden Markov chain of 4 elements. To each chain, we apply the MPM to have a vector composed of values “0” and “1“. We used the EM algorithm to estimate the parameters of the MPM that are: the probability of appearance of each class $p(0)$ and $p(1)$, the transition matrix A, the mean of the first class μ_1, the variance of the first class σ_1^2, the mean of the second class μ_2, and the variance of the second class σ_2^2. We run the EM algorithm for three iterations. To choose the number of iterations we have tested different values from one to ten. We observed that after three iterations the value of the variance became zero which results in an error in the computation of the Gaussian function. This problem arises because the inputs of the EM algorithm are small, in the order of 10^{-6}. We initialized the parameters as follows: for the cat class: $p(0) = 0.25, p(1) = 0.75$, $A = \begin{bmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix}$, $m_1 = 1$, ${\sigma_1}^2 = 1$, $m_2 = 0.9$ and ${\sigma_2}^2 = 2$. For the dog class: $p(0) = 0.2, p(1) = 0.8$, $A = \begin{bmatrix} 0.2 & 0.8 \\ 0.2 & 0.8 \end{bmatrix}$, $m_1 = 3$, ${\sigma_1}^2 = 2$, $m_2 = 0.8$, and ${\sigma_2}^2 = 1$. We note that we have done extensive experiments to find the appropriate initialization for parameters that maximized the accuracy of the CNN-HMC model. In Fig. 8 and 9, we illustrate the initial transition graphs of the two classes, respectively.

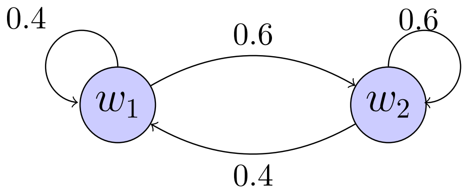

Fig. 8 Initial transition graph for the cat class

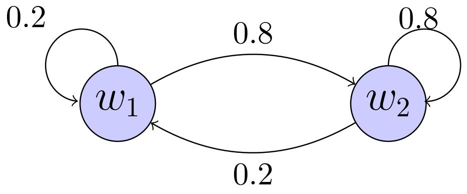

Fig. 9 Initial transition graph for the dog class

We repeat the same process for all the 64 matrices of size (2×2). At the end, all the vectors of size 4 obtained in the previous step are concatenated and the dominant class is computed. This class represents the class of the input image. In this architecture, we suppose that the characteristics are independent and that is why we transform each characteristic of size (2, 2) into a Markov chain, which gives us 64 Markov chains. If we assume the opposite then all the feature matrices which are of size (2, 2, 64) will be transformed into a single Markov chain of size 256. We have tested this possibility but the results were not satisfactory.

We illustrate the architecture of the CNN-HMC for the 4Conv model in Fig. 10.

5.3 mAlexNet description

We will now compare our proposal to a model called mAlexNet [29] for mini AlexNet. This model takes RGB images of size (224,224) as input. It is composed of three convolutional layers and each of them is followed by a Max-pooling layer. The first convolutional layer takes the parameters: 16 for filters, (11, 11) for kernel size, and 4 for the value of stride. The second convolutional layer takes the parameters: 20 for filters, (5, 5) for kernel size, and 1 for the value of stride. And finally, the last convolutional layer takes the parameters: 30 for filters, (3, 3) for kernel size, and 1 for the value of stride. All those layers use the ReLU activation function. The Max-Pooling layers use a pool size of (3, 3) and stride of size 2. The features map extracted by the convolutional and the pooling layers is of size 3×3×30. This means that we have 30 features with each of them having a size of 3×3. Afterward, the classification is done by a regular neuronal network with one layer having 48 neurons and an activation function, ReLU. We obtain the output of the model by applying the ”softmax“ function which is used to normalize the output, transforming it from weighted sum values to single-sum probabilities. Each value in the softmax function is

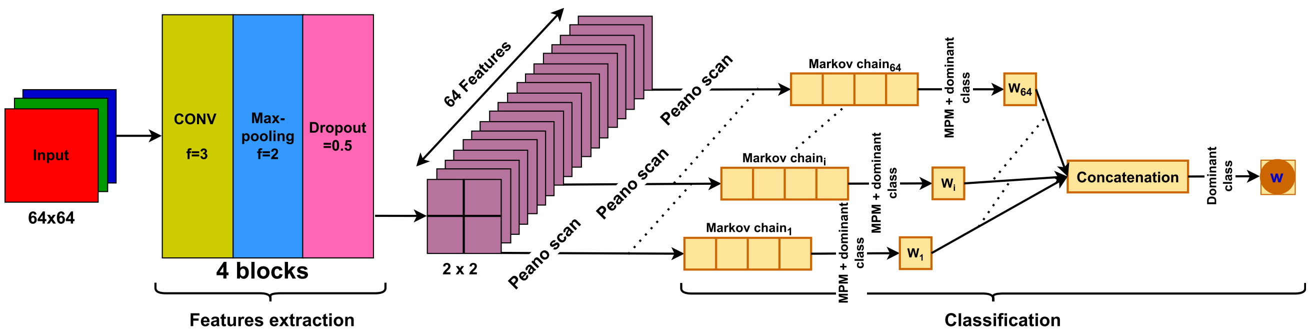

Fig. 10 The architecture of CNN-HMC for 4Conv

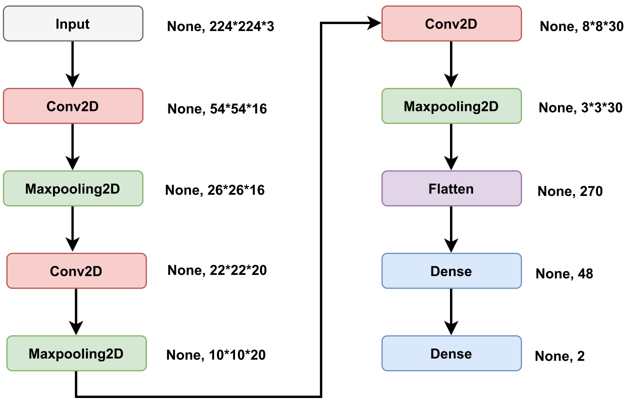

Fig. 11 mAlexNet model summary with input and output size of each layer

interpreted as the probability of belonging to each category. In Fig. 11, we show a summary of the mAlexNet model with input and the output size of each layer.

5.4 CNN-HMC for mAlexNet

During this stage we apply the same process applied to 4Conv to mAlexNet. We take the output of the model before applying the flattening that is a volume of size 3×3×30, representing the features map of the model. This means that mAlexNet has extracted 30 features, each of size 3 by 3. To each feature or matrix of size 3×3 we apply the Peano scan to obtain a hidden Markov chain of size 9. After that, we estimate the parameters for the MPM method using the EM algorithm. We run the EM algorithm for three iterations. We initialized its parameters as follow: for the cat class: $p(0) = 0.25, p(1) = 0.75$, $A = \begin{bmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix}$, $m_1 = 1$, ${\sigma_1}^2 = 0.9$, $m_2 = 0.9$ and ${\sigma_2}^2 = 2$. For the dog class: $p(0) = 0.2, p(1) = 0.8$, $A = \begin{bmatrix} 0.2 & 0.8 \\ 0.2 & 0.8 \end{bmatrix}$, $m_1 = 3$, ${\sigma_1}^2 = 2$, $m_2 = 0.8$, and ${\sigma_2}^2 = 1$. In Fig. 12, we show the architecture of the CNN-HMC applied to the mAlexNet model.

6 Models training

We trained all the models with “Adam” optimizer [73]. The parameters used for this algorithm are by default as follows: learning rate $alpha = 0.001$, $beta1 = 0.9$, $beta2 = 0.999$, and $epsilon = 1exp^{-07}$. This parameters have shown a significant performance for models learning and are rarely modified. We have tested other parameters such as learning rate = 0.01, and 0.0001, $beta1$ = 0.1, and 0.01, and $beta_2$ = 0.1, and 0.01, but the best results were achieved by the default parameters mentioned above. For the loss function, we used Categorical CrossEntropy (CCE) [74]. This function is utilized in multi-classification problems, which is our case with the two classes: cat and dog. It is defined as:

$$\text{Loss} = -\sum_{i=1}^{N} y_i * log(\hat{y_i}), \tag{11}$$

where:

y_i: is the target output corresponding to the i^{th} class it is equal to 1 or 0;

$\hat{y_i}$: is the output predicted by the model for the i^{th} class, $\hat{y}\epsilon[0, 1]$;

N: is the output size, or the number of output classes in the model.

This loss function is a useful indicator of how easily two discrete probability distributions can be distinguished from one another. In this case, y_i represents the likelihood that event I will occur, and the total of all y_i equals 1, implying that only one event will occur. When the distributions become closer to one another, the negative sign ensures that the loss gets lower.

To evaluated the performances of our models we used recall, precision, F1-score, and accuracy metrics defined as follows:

$$\text{Recall} = \frac{T_p}{T_p + F_n}, \tag{12}$$

$$\text{Precision} = \frac{T_p}{T_p + F_p}, \tag{13}$$

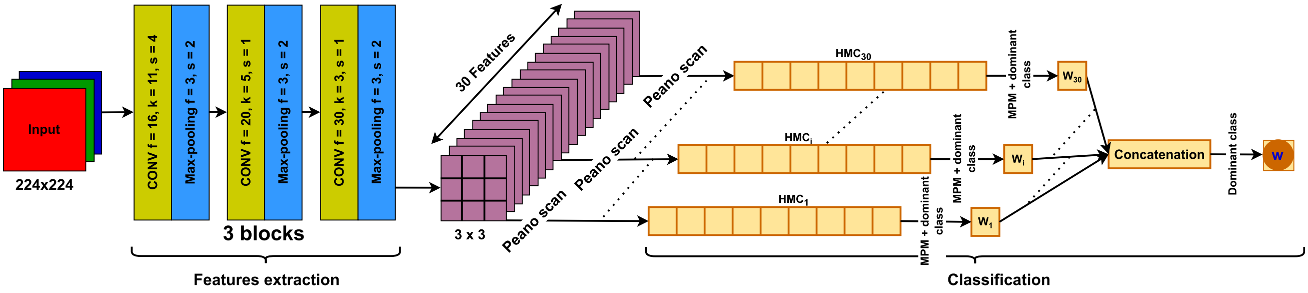

Fig. 12 The architecture of CNN-HMC for mAlexNet

$$\mathrm{F1-score} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \quad (14)$$

$$\mathrm{Accuracy} = \frac{T_\mathrm{p} + T_\mathrm{n}}{T_\mathrm{p} + T_\mathrm{n} + F_\mathrm{p} + F_\mathrm{n}}, \quad (15)$$

where, T_p and T_n are true positives, and true negatives, F_p and F_n are false positives and false negatives. The objective of the training is to minimize the loss and maximize the accuracy.

6.1 4Conv training

We have run our models 15 times. In Fig. 13, we illustrate the graph of the accuracy and the loss of the proposed 4Conv model. The training accuracy of this model is 68.21%, and the loss is 59.05%. As we can see from the figure, the model does not suffer from the overfitting problem [75], since the gap between the training and the testing accuracy is small.

6.2 mAlexNet training

We have run mAlexNet 15 times. In Fig. 14, we illustrate the graphs of the accuracy and the loss of the mAlexNet model. The training accuracy of this model is 71.77%, while the loss is 56.66%. Again this model also does not overfit the data.

6.3 Datasets

To evaluate the performance of our model, we opted for a classification of an image as a cat or a dog. For this problem we used a dataset made up of 10,028 images of cats and dogs [30]. We divided this data into two subsets: the first for training and the second for testing. The training set represents 80% of the data and the test set represents the remaining 20%. We refer to the test dataset as “Test1” Fig. 15 illustrates two images belonging to the two classes of cat and dog, respectively. Usually, models perform well in the training and test datasets because most of the data are from the same source, and they have almost the same backgrounds and the same light degradation. That’s why we used two other datasets for the test, the first is downloaded from [31], we refer to it by “Test2”, and the second was created by us, and we refer to it as “Test3” [32]. Test2 contains 4747 images of cats and 4724 images of dogs. For Test3, we downloaded 20 random images of each class from the internet. In Table 1, we show the details of the datasets used for the training and the test of the model with the number of images of each class.

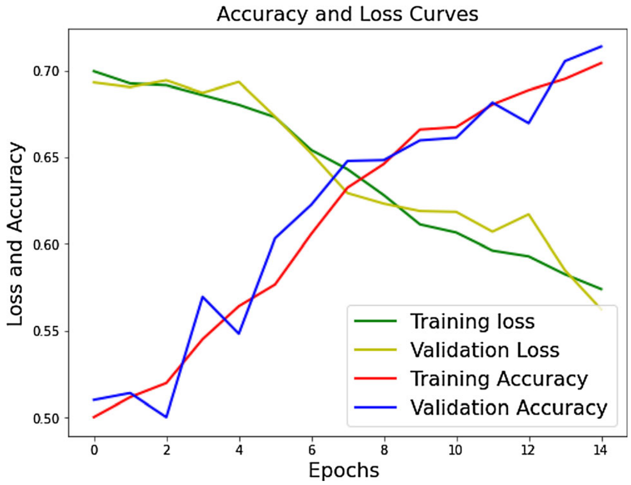

Fig. 13 Training and validation accuracy and loss of the 4Conv model

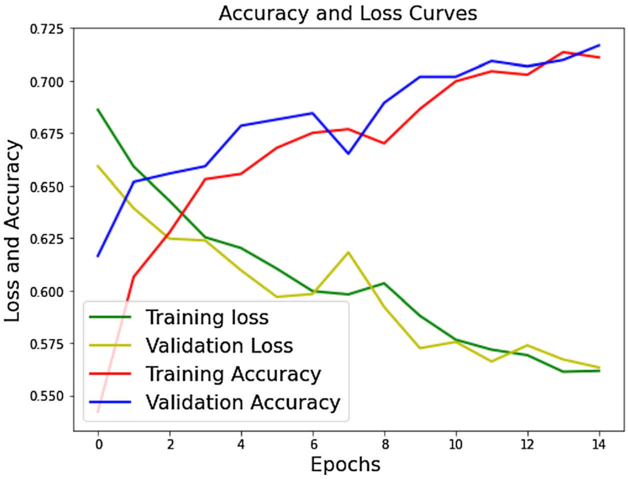

Fig. 14 Training and validation accuracy of the mAlexNet model

In addition, in order to evaluate the performance of our model with literature work, we used two other datasets. The first is a Car-Bike dataset [33], and the second is an Elephant dataset [34]. The Car-Bike dataset contains 2000 images in each class, while the elephant dataset has 840 images divided equally between the African and the Asian elephant. Table 2 illustrates the details of the two datasets.

Fig. 15 Illustration of two dataset images, on the left an image of a cat and on the right an image of a dog

Table 1 The datasets used for models training and testing

	Cat	Dog	Total
Training	4000	4005	8005
Test1	1011	1012	2023
Test2	4747	4724	9471
Test3	20	20	40

7 Evaluation experiments

7.1 Implementation details

To evaluate the performances of our models we used Keras [76] which is a framework based on TensorFlow [77]. Keras is a high-level neural network library that provides a lot of predefined algorithms, methods, and functions. We run our solution on Google Colaboratory or Colab which is a free online service based on Jupyter Notebook and designed for deploying Machine Learning solutions. It can be used without any hardware requirement and installation. It offers a RAM of 13GB and a disk of size 78.19GB.

7.2 Results and discussion

In Table 3, we illustrate the results obtained from the tests of 4Conv against CNN-HMC. We executed tests on three datasets: Test1, Test2, and Test3. They are made up of 2023, 9471, and 40 images, respectively, divided equally among the two classes. The table shows the number of cats and dogs predicted by two classic CNN models, 4Conv, against its corresponding CNN-HMC model. The models are compared using the recall, the precision, the F1-score and accuracy of the prediction. Test1 belongs to the same dataset as the training dataset, which is the traditional method used to evaluate any deep learning model. We notice that the images of Test1 are unseen images by the model. The recall of our model is 73.91% against only 45.55% of the 4Conv model. The CNN-HMC reached a precision equal to 99.86% while the CNN model gives 83.36%. The values of the F1-score are 84.95% and 58.91% for the CNN-HMC and the 4Conv, respectively.

With this dataset, CNN-HMC reached an accuracy of 86.90% against only 68.21% for 4Conv. We also notice that CNN-HMC has miss-classified only one image of a cat while 1010 images were well classified. As mentioned above, some deep learning models such as CNN works well when testing them in the same dataset used for the training as we did with Test1, but do not work well with other datasets. This problem is due to the resolution of the images. To prove the robustness of our model we test it with two other datasets, Test2, and Test3.

Table 2 The datasets used for comparison with related work

	Class 1	Class 2
Car-Bike	Bike: 2000	Car: 2000
Elephant	African: 420	Asian: 420

The value of the recall is equal to 71.72% of the CNN-HMC model compared to only 47.44% of the 4Conv model in the Test2 dataset. 99.88% is the precision of our model against 84.06% of the 4Conv model. The CNN-HMC model gives an F1-score equal to 83.49% compared to 60.65% for its corresponding CNN with an improvement of 13%.

In Test2 CNN-HMC reached an accuracy of 85.85% against only 69.29% for 4Conv, with an improvement of 16%. We notice that CNN-HMC was able to predict correctly 4743 images of cats from a total of 4747 images of cats, which means that it has missed only four images. Each one of the recall, the precision, and the F1-score are equal to 100% for the Test3 dataset. While these metrics are equal to 50%, 66.67%, and 57.14%, respectively, for the 4Conv model.

In Test3 the CNN-HMC model was able to predict all cats and dogs correctly unlike the 4Conv model, which makes the accuracy of the CNN-HMC 100%, while the accuracy of 4Conv is only 62.5%.

In Fig. 16, we plot the accuracies results of 4Conv and its CNN-HMC version using the three datasets: Test1, Test2, and Test3. We will refer to CNN-HMC in this figure as CNN-HMC1. The accuracy of CNN-HMC1 is shown in purple striped color while that of 4Conv is shown in yellow. As we can see in the graph, the accuracy of CNN-HMC is always higher than that of 4Conv within the three datasets.

In Table 4, we illustrate the comparison results of mAlexNet and its CNN-HMC version using the same datasets: Test1, Test2, and Test3. In the Test1 dataset, the recall is 98.81% for our model against 75.39% of the CNN model. The CNN-HMC gives a precision equal to 74.40% while the mAlexNet model gives 70.32%. The F1-score is 84.88% for the CNN-HMC and 72.77% for the mAlexNet.

The mAlexNet gives an accuracy of 71.77%, while CNN-HMC's accuracy is 82.40% which shows an increase of 10%. We notice here that the CNN-HMC has well classified 1000 images of dogs from a total of 1012.

The CNN-HMC model reached a recall value equal to 98.92% compared to only 77.90% of the mAlexNet model in the Test2 dataset. The CNN-HMC improved the recall metric by 21% compared to mAlexNet. The precision is 73.46% for our model against 71.61% of the mAlexNet model. Using the above values of the recall and the precision, we computed the F1-score metric. The CNN-HMC model gives an F1-score equal to 84.31% compared to

Table 3 Comparison of test results of 4Conv model with the CNN-HMC model using three lats vs dogs datasets

Datasets	Models	TN	TP	Recall (%)	Precision (%)	F1-score (%)	Accuracy (%)
Test1	CNN-HMC	1010	748	73.91	99.86	84.95	86.90
	4Conv	919	461	45.55	83.36	58.91	68.21
Test2	CNN-HMC	4743	3388	71.72	99.88	83.49	85.85
	4Conv	4322	2241	47.44	84.06	60.65	69.29
Test3	CNN-HMC	20	20	100	100	100	100
	4Conv	15	10	50.00	66.67	57.14	62.50

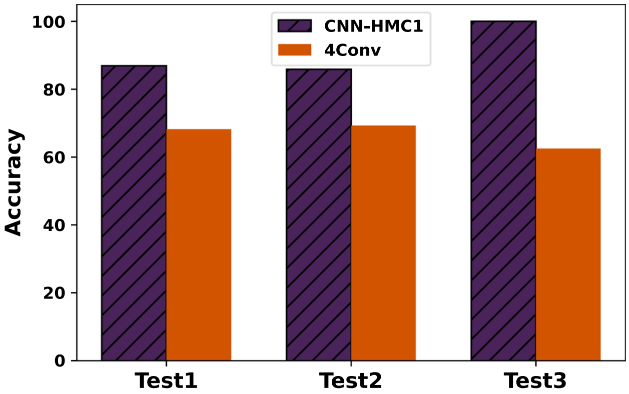

Fig. 16 Comparison of CNN-HMC and 4Conv accuracy's for prediction using the three datasets

74.62% for its corresponding CNN with an improvement of about 10%.

mAlexNet gives an accuracy of 73.57% compared to 81.63% for CNN-HMC, which means there was an improvement of 8%. The last dataset used for the test is Test3, that we collected ourselves.

In the Test3 dataset, 100%, 86.96%, and 93.02% are the values of the recall, the precision, and the F1-score, respectively, given by the CNN-HMC model. The recall of the mAlexNet model is 85%, and the precision is 60.71%, they are lower than those obtained using our model by about 15%. 70.83% is the value of the F1-score given by the mAlexNet model in the same dataset. For this last metric, our model has shown an improvement of almost 23%.

The CNN-HMC model was able to predict correctly all dogs' images unlike the mAlexNet model, which has miss-classified three dogs images. For the cat class, the number predicted by mAlexNet is lower than the number predicted by the CNN-HMC model. This last model was able to predict correctly almost double of the images classified by mAlexNet. These results have led to an accuracy of 92.50% for the CNN-HMC model and 65% for the mAlexNet model with the same datasets.

In Fig. 17, we plot the accuracies of mAlexNet and its corresponding CNN-HMC model according to the three used datasets. To make difference between CNN-HMC for 4Conv and the other for mAlexNet, we will refer to CNN-HMC in this figure as CNN-HMC2. The accuracies of the CNN-HMC2 are shown in purple striped color and those of mAlexNet in yellow color. As we can see from the bar chart, in the three datasets, the accuracy of mAlexNet is always lower than that of CNN-HMC2.

From Figs. 16 and 17 we conclude that our proposed model using convolutional neural networks for feature extraction and hidden Markov chains for classification gives significant results compared to a classic CNN model in terms of accuracy.

We will use mAlexNet as a feature extractor since it has the highest training accuracy compared to the 4Conv model. We employ transfer learning to a far comparison with similar related work; this means that we use the same features map extracted by mAlexNet, and we vary the classification algorithm between HMC, SVM, LSTM, RF, and gcForest. All the models are tested on five datasets, Test1, Test2, Test3, Car-Bike, and Elephant. Table 5 represents the results of comparing CNN-HMC, CNN-SVM [35], CNN-LSTM [44], CNN-RF [71], and gcForest [71] models using the recall, the precision, the F1-score, and the accuracy for a classification task. The recall of the CNN-HMC range from 98.81% to 100%. The CNN-SVM recall

Table 4 Comparison of test results of mAlexNet model with the CNN-HMC model using three cats vs dogs datasets

Datasets	Models	TN	TP	Recall (%)	Precision (%)	F1-score (%)	Accuracy (%)
Test1	CNN-HMC	667	1000	98.81	74.40	84.88	82.40
	mAlexNet	689	763	75.39	70.32	72.77	71.77
Test2	CNN-HMC	3059	4673	98.92	73.46	84.31	81.63
	mAlexNet	3288	3680	77.90	71.61	74.62	73.57
Test3	CNN-HMC	20	17	100	86.96	93.02	92.5
	mAlexNet	17	9	85.00	60.71	70.83	65.00

is equal to 87.45%, 91.26%, 95.0%, 92.90%, and 89.76% in the Test1, the Test2, the Test3, the Car-Bike, and the Elephant datasets, respectively. The recall of the CNN-LSTM was less than the CNN-HMC and the CNN-SVM, with a value ranging between 68.97% and 84.15%. The worst value of the CNN-RF recall is 65.0% obtained in the Test3 dataset, while the better value is 80.82% in the Test2 dataset. In the Test1 dataset, the recall of the gcForest is 62.35%, while its highest value is 76.52% obtained in the Test2 dataset. In the remaining three datasets, the recall of the gcForest is near 55.0%.

The precision of the CNN-HMC model is 74.40% compared to 57.80%, 71.96%, 72.67%, and 64.65% for the CNN-SVM, the CNN-LSTM, the CNN-RF, CNN-gcForest, respectively, in the Test1 dataset. In the Test2 dataset, the five models' precisions were equal to 73.46%, 68.24%, 74.79%, 81.23%, and 76.30%, respectively. In the Test3 dataset, the precision of our model is 86.96%, while the other models give a value ranging between 51.35% and 55.55%. In the Car-Bike dataset, of the CNN-HMC model precision is 83.23%, while the rest of the models have a value less than 50.0%. In the Elephant dataset, our model has the highest precision value with 73.46% against an average of 50.0% for the compared models.

The F1-score of our model, in the Test1 dataset, is equal to 84.88% against almost 70.0% of the CNN-SVM, the CNN-LSTM, and the CNN-RF, while the low value is 63.48% given by the CNN-gcForest model. In the Test2 dataset, the CNN-HMC gives the highest F1-score value equal to 84.31%, followed by the CNN-RF with 81.02%, then the CNN-SVM and the CNN-gcForest with 78.09% and 76.41%, respectively, and finally CNN-LSTM with 73.40%. In the Test3 dataset, the F1-score of our model is 93.02% against 66.0% and 63.83% of the CNN-SVM, and the CNN-LSTM, respectively, while the two based forest models give a value of 57.7% and 53.66%. In the Car-Bike dataset, the CNN-HMC, the CNN-SVM, the CNN-LSTM, the CNN-RF, and the CNN-gcForest models give an F1-score values equal to 90.76%, 63.82%, 62.73%, 58.27%, and 49.96%, respectively. In the Elephant dataset, the

Table 5 Comparison of test results of CNN-HMC model against CNN-SVM, CNN-LSTM, CNN-RF, and CNN-gcForest using the same features map extracted by mAlexNet

Datasets	Models	TP	TN	Recall(%)	Precision (%)	F1-score (%)	Accuracy (%)
Test1	CNN-HMC	1000	667	98.81	74.40	84.88	82.40
	CNN-SVM [35]	885	365	87.45	57.80	69.60	61.789
	CNN-LSTM [44]	698	739	68.97	71.96	70.43	71.03
	CNN-RF [71]	718	741	70.95	72.67	71.80	72.12
	CNN-gcForest [71]	631	666	62.35	64.65	63.48	64.11
Test2	CNN-HMC	4673	3059	98.92	73.46	84.31	81.63
	CNN-SVM [35]	4311	2741	91.26	68.24	78.09	74.46
	CNN-LSTM [44]	3404	3600	72.06	74.79	73.40	73.95
	CNN-RF [71]	3818	3865	80.82	81.23	81.02	81.12
	CNN-gcForest [71]	3615	3624	76.52	76.30	76.41	76.43
Test3	CNN-HMC	17	20	100	86.96	93.02	92.50
	CNN-SVM [35]	19	2	95.0	51.35	66.0	52.50
	CNN-LSTM [44]	15	8	70.5	55.55	63.83	57.50
	CNN-RF [71]	13	8	65.0	52.0	57.7	52.50
	CNN-gcForest [71]	11	10	55.0	52.38	53.66	52.50
Car-Bike	CNN-HMC	1996	1598	99.80	83.23	90.76	89.85
	CNN-SVM [35]	1858	36	92.90	48.61	63.82	47.35
	CNN-LSTM [44]	1683	317	84.15	50.0	62.73	50.0
	CNN-RF [71]	1532	274	76.60	47.02	58.27	45.15
	CNN-gcForest [71]	1088	733	54.40	46.20	49.96	45.52
Elephant	CNN-HMC	418	269	99.52	73.46	85.0	81.79
	CNN-SVM [35]	377	45	89.76	50.13	64.33	50.24
	CNN-LSTM [44]	319	98	75.95	49.76	60.13	49.64
	CNN-RF [71]	319	109	75.95	50.63	60.76	50.95
	CNN-gcForest [71]	229	195	54.52	50.44	52.40	50.47

worst F1-score value is equal to 52.40% given by the CNN-gcForest followed by the CNN-LSTM, and the CNN-RF with a value near 60.0%, the CNN-SVM with an F1-score equal to 64.33%, and finally the CNN-HMC with a value of 85.0%.

The accuracy of our model is 82.40% in the Test1 test dataset, while the accuracy of the CNN-LSTM and the CNN-RF is around 70% and the accuracy of the CNN-SVM and the CNN-gcForest is around 60%. In the Test2 test set, the accuracy of the CNN-HMC and the CNN-RF are almost equal, while the accuracy of the other models ranged from 73% to 76%. The CNN-HMC model reached an accuracy of 92.5%, while the CNN-LSTM accuracy is the highest among all the tested models with a 57.5%, against 52.5% of the CNN-SVM, the CNN-RF, and the gcForest models in the Test3 dataset. In the Car-Bike dataset, the accuracy of our model is 89.85% against a value less than 50% of the other compared to models. The Elephant dataset yields the same outcomes as above with an accuracy of up to 81.79% of our model compared to a value near 50% of the other models.

From Table 5, we observe that the recall of the CNN-HMC is the highest compared to CNN-SVM, the CNN-LSTM, the CNN-RF, and the CNN-gcForest in the five tested datasets. The precision of our model is better than all the compared models in the five datasets except in the Test2 dataset. The F1-score of the CNN-HMC is ranged from 84% to 93% in the five datasets, while the other models give fewer values. Finally, the accuracy of our model is highest in the five test dataset, while the CNN-SVM, the CNN-LSTM, the CNN-RF, and the CNN-gcForest accuracies' are near 50% in the Test3, the Car-Bike, and the Elephant dataset.

These results are explained as follows. The Test1 dataset is used for both the training and the testing of HMC, SVM, LSTM, RF, and gcForest, and that is why the accuracy is high while this dataset contains 2003 images. The Test2 dataset has more than 9000 images which explains the good accuracy obtained by the CNN-SVM, the CNN-LSTM, the CNN-RF, and the CNN-gcForest compared to the CNN-HMC. The low performance of the machine learning models are probably caused by the lack of data in the Test3, the Car-Bike, and the Elephant datasets. We also notice that the CNN-SVM, the CNN-LSTM, the CNN-RF, and the CNN-gcForest parameters were adjusted to the datasets used in the original work.

In summary, the CNN-HMC model outperforms the CNN-SVM, the CNN-LSTM, the CNN-RF, and the CNN-gcForest in terms of recall, precision, F1-score, and accuracy using five different test datasets.

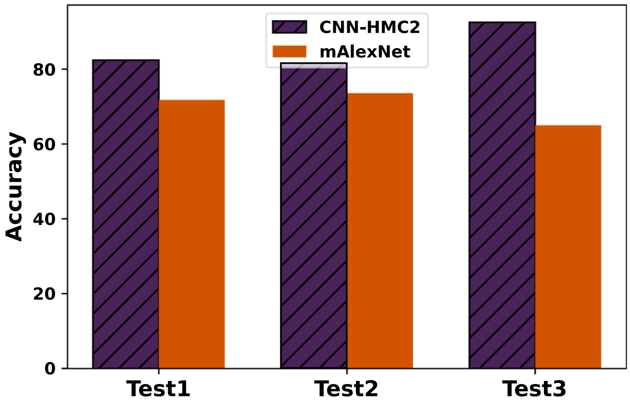

Fig. 17 Comparison of CNN-HMC and mAlexNet accuracy's for prediction using the three datasets

8 Conclusion

We have proposed a new model, called CNN-HMC, for convolutional neural networks (CNNs)-hidden Markov chains (HMCs). Our model uses CNN for feature extractions and dimensionality reduction, and uses HMC for classification. The objective is to enhance the performances of CNN models for the image classification task. To evaluate the performance of our proposal, it is compared to six models in two series of experiments. In the first series, we applied it to the problem of cats/dogs classification. We have compared CNN-HMC to two CNN models. The first one is composed of four convolutions, named 4Conv, and the second is a minimal version of the well-known AlexNet model, called mAlexNet. The results show that CNN-HMC model outperforms the classical CNN model, and significantly improves the accuracy of the Mini AlexNet.

In the second series, we have compared our solution to four models CNN-SVM [35], CNN-LSTMs [44], CNN-RF [71], and CNN-gcForests [71], which only differ from CNN-HMC by the second classification step. Based on five datasets and four metrics (i.e., recall, precision, F1-score, and accuracy), results of these comparisons show again the interest of the proposed CNN-HMC model.

Let us mention one perspective for further works. HMCs considered in the paper are very basic ones, and different extensions have been proposed since their introduction. In particular, they have been extended to "pairwise" and "triplet" Markov chains [78], and to hidden Markov chains with copulas, which model non-Gaussian correlated noise [79]. Using such extensions instead of HMC are likely to improve the CNN-HMC classifier proposed in the paper.

Data availability The five datasets used in this work can be accessible via the links: Test1: https://www.kaggle.com/tongpython/cat-and-dog Test2: https://www.kaggle.com/trishalsingh/dogs-vs-cats Test3: https://drive.google.com/file/d/

1vlVpP3ZtLSWCxUs0e9oypV857Bnia8Sv/view?usp=sharing Car-Bike dataset: https://www.kaggle.com/datasets/utkarshsaxenadn/car-vs-bike-classification-dataset Elephant dataset: https://www.kaggle.com/datasets/vivmankar/asian-vs-african-elephant-image-classification.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Melekoodappattu JG, Dhas AS, Kandathil BK, Adarsh K (2022) Breast cancer detection in mammogram: combining modified cnn and texture feature based approach. J Ambient Intell Humaniz Comput pp. 1–10
2. Beeravolu AR, Azam S, Jonkman M, Shanmugam B, Kannoorpatti K, Anwar A (2021) Preprocessing of breast cancer images to create datasets for deep-cnn. IEEE Access 9:33438–33463
3. Xia H, Cai N, Wang H, Mao Y, Wang H, Li J, Wang P (2021) Brain mr image super-resolution via a deep convolutional neural network with multi-unit upsampling learning. Signal, Image Video Process 15(5):931–939
4. Reshi AA, Rustam F, Mehmood A, Alhossan A, Alrabiah Z, Ahmad A, Alsuwailem H, Choi GS (2021) An efficient cnn model for covid-19 disease detection based on x-ray image classification. Complexity
5. AbdulAzeem Y, Bahgat WM, Badawy M (2021) A cnn based framework for classification of alzheimer's disease. Neural Comput Appl 33(16):10415–10428
6. Nezamabadi K, Naseri Z, Moghaddam HA, Modarresi M, Pak N, Mahdizade M (2019) Lung hrct pattern classification for cystic fibrosis using convolutional neural network. Signal, Image Video Process 13(6):1225–1232
7. Innocenti E, Giuliano R (2021) in *2021 AEIT International Conference on Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)* (IEEE), pp. 1–6
8. Chen Y, Wang H, Li W, Sakaridis C, Dai D, Van Gool L (2021) Scale-aware domain adaptive faster r-cnn. Int J Comput Vis 129(7):2223–2243
9. Ajoy A, Mahindrakar CU, Gowrish D, Vinay A (2021) In:2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)
10. Han C, Shan S, Kan M, Wu S, Chen X (2022) Personalized convolution for face recognition. Int J Comput Vis 130(2):344–362
11. Goumiri S, Benboudjema D, Pieczynski W (2021) In: 2021 IEEE International Smart Cities Conference (ISC2)
12. Farley A, Ham H et al (2021) Real time ip camera parking occupancy detection using deep learning. Procedia Comput Sci 179:606–614
13. Dhope T, Chitale P, Rampure S, Ghane S (2021) In: 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)
14. Saponara S, Elhanashi A, Gagliardi A (2021) Real-time video fire/smoke detection based on cnn in antifire surveillance systems. J Real-Time Image Process 18(3):889–900
15. Alhichri H, Alswayed AS, Bazi Y, Ammour N, Alajlan NA (2021) Classification of remote sensing images using efficientnet-b3 cnn model with attention. IEEE Access 9:14078–14094
16. Lee J, Kim D, Ponce J, Ham B (2019) In:Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2278–2287
17. Han K, Rezende RS, Ham B, Wong KYK, Cho M, Schmid C, Ponce J (2017) In: Proceedings of the IEEE international conference on computer vision, pp. 1831–1840
18. Song R, Zhang W, Zhao Y, Liu Y (2022) Unsupervised multi-view cnn for salient view selection and 3d interest point detection. Int J Comput Vis 130(5):1210–1227
19. Sharma N, Jain V, Mishra A (2018) An analysis of convolutional neural networks for image classification. Procedia comput Sci 132:377–384
20. Rabiner L (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286. https://doi.org/10.1109/5.18626
21. Juang BH, Rabiner LR (1991) Hidden Markov models for speech recognition. Technometrics 33(3):251–272
22. Gales M, Young S, et al (2008) The application of Hidden Markov Models in Speech Recognition. Foundat Trends® Signal Process 1(3), 195–304
23. Rabiner L, Juang B (1986) An introduction to hidden Markov models. IEEE ASSP Magaz 3(1):4–16. https://doi.org/10.1109/MASSP.1986.1165342
24. Fjortoft R, Delignon Y, Pieczynski W, Sigelle M, Tupin F (2003) Unsupervised classification of radar images using hidden Markov chains and hidden Markov random fields. IEEE Trans Geosci Remote Sens 41(3):675–686
25. Carincotte C, Derrode S, Bourennane S (2006) Unsupervised change detection on SAR images using fuzzy hidden Markov chains. IEEE Trans Geosci Remot Sens 44(2):432–441
26. Bricq S, Collet C, Armspach JP (2008) Unifying framework for multimodal brain MRI segmentation based on Hidden Markov Chains. Medi Image Analy 12(6):639–652
27. Nefian AV, Hayes MH (1998) In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181), vol. 5 (IEEE), pp. 2721–2724
28. Li J, Najmi A, Gray RM (2000) Image classification by a two-dimensional hidden Markov model. IEEE Trans Signal Process 48(2):517–533
29. Amato G, Carrara F, Falchi F, Gennaro C, Meghini C, Vairo C (2017) Deep learning for decentralized parking lot occupancy detection. Expert Systh Appl 72:327–334
30. Cat and dog (2018). https://www.kaggle.com/tongpython/cat-and-dog. (Last accessed: 26/12/2021)
31. Dogs vs cats (2021). https://www.kaggle.com/trishalsingh/dogs-vs-cats. (Last accessed: 26/12/2021)
32. Cats/Dogs (2020). https://drive.google.com/file/d/1vlVpP3ZtLSWCxUs0e9oypV857Bnia8Sv/view?usp=sharing, (Last accessed: 26/12/2021)
33. Car vs bike classification dataset (2021). https://www.kaggle.com/datasets/utkarshsaxenadn/car-vs-bike-classification-dataset. (Last accessed: 26/03/2023)
34. Asian vs african elephants (2022). https://www.kaggle.com/datasets/vivmankar/asian-vs-african-elephant-image-classification. (Last accessed: 26/03/2023)
35. Niu XX, Suen CY (2012) A novel hybrid cnn-svm classifier for recognizing handwritten digits. Pattern Recogn 45(4):1318–1325
36. Ahlawat S, Choudhary A (2020) Hybrid cnn-svm classifier for handwritten digit recognition. Procedia Comput Sci 167:2554–2560
37. Wu H, Huang Q, Wang D, Gao L (2018) A cnn-svm combined model for pattern recognition of knee motion using

mechanomyography signals. J Electromyogr Kinesiol 42:136–142

38. Khairandish MO, Sharma M, Jain V, Chatterjee JM, Jhanjhi N (2022) A hybrid cnn-svm threshold segmentation approach for tumor detection and classification of mri brain images. Irbm 43(4):290–299
39. Koklu M, Unlersen MF, Ozkan IA, Aslan MF, Sabanci K (2022) A cnn-svm study based on selected deep features for grapevine leaves classification. Measurement 188, 110,425
40. Jia AD, Li BZ, Zhang CC (2020) Detection of cervical cancer cells based on strong feature cnn-svm network. Neurocomputing 411:112–127
41. Sun X, Liu L, Li C, Yin J, Zhao J, Si W (2019) Classification for remote sensing data with improved cnn-svm method. IEEE Access 7, 164,507–164,516
42. Basly H, Ouarda W, Sayadi FE, Ouni B, Alimi A.M(2020) In:Image and Signal Processing: 9th International Conference, ICISP 2020, Marrakesh, Morocco, June 4–6, Proceedings 9 (Springer, 2020), pp. 271–281
43. Tao T, Wei X (2022) A hybrid cnn-svm classifier for weed recognition in winter rape field. Plant Methods 18(1):29
44. Karimi Jafarbigloo S, Danyali H (2021) Nuclear atypia grading in breast cancer histopathological images based on cnn feature extraction and lstm classification. CAAI Trans Intell Technol 6(4), 426–439
45. Wang J, Yu L.C, Lai K.R, Zhang X (2016) In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers), pp. 225–230
46. Kim TY, Cho SB (2019) Predicting residential energy consumption using cnn-lstm neural networks. Energy 182:72–81
47. Livieris IE, Pintelas E, Pintelas P (2020) A cnn-lstm model for gold price time-series forecasting. Neural Comput Applicat 32:17351–17360
48. Zhao J, Mao X, Chen L (2019) Speech emotion recognition using deep 1d & 2d cnn lstm networks. Biomedi Signal Process Control 47:312–323
49. Mutegeki R, Han DS (2020) In: 2020 international conference on artificial intelligence in information and communication (ICAIIC)
50. Swapna G, Kp S, Vinayakumar R (2018) Automated detection of diabetes using cnn and cnn-lstm network and heart rate signals. Procedia Comput Sci 132:1253–1262
51. Zha W, Liu Y, Wan Y, Luo R, Li D, Yang S, Xu Y(2022) Forecasting monthly gas field production based on the cnn-lstm model. Energy p. 124889
52. Sajjad M, Khan Z.A, Ullah A, Hussain T, Ullah W, Lee MY, Baik SW (2020) A novel cnn-gru-based hybrid approach for short-term residential load forecasting. Ieee Access 8, 143, 759–143,768
53. Pan M, Zhou H, Cao J, Liu Y, Hao J, Li S, Chen CH (2020) Water level prediction model based on gru and cnn. Ieee Access 8:60090–60100
54. Dua N, Singh SN, Semwal VB (2021) Multi-input cnn-gru based human activity recognition using wearable sensors. Computing 103:1461–1478
55. Dua N, Singh SN, Semwal VB, Challa SK (2022) Inception inspired cnn-gru hybrid network for human activity recognition. Multim Tools Appl 1–35
56. Yu J, Zhang X, Xu L, Dong J, Zhangzhong L (2021) A hybrid cnn-gru model for predicting soil moisture in maize root zone. Agric Water Manag 245: 106,649
57. Faraji M, Nadi, Ghaffarpasand O, Homayoni S, Downey K (2022) An integrated 3d cnn-gru deep learning method for short-term prediction of pm2. 5 concentration in urban environment. Sci Total Environ 834: 155,324
58. Ma C, Zhao Y, Dai G, Xu X, Wong SC (2022) A novel stfsa-cnn-gru hybrid model for short-term raffic speed prediction. IEEE Trans Intell Transport Syst
59. Li MW, Xu DY, Geng J, Hong WC (2022) A hybrid approach for forecasting ship motion using cnn–gru–am and gcwoa. Appl Soft Comput 114: 108,084
60. Gill HS, Khalaf OI, Alotaibi Y, Alghamdi, S Alassery (2022) Multi-model cnn-rnn-lstm based fruit recognition and classification. Intell Autom Soft Comput 33:(1)
61. Al-Rakhami MS, Islam MM, Islam MZ, Asraf A, A.H. Sodhro, W. Ding, Diagnosis of covid-19 from x-rays using combined cnn-rnn architecture with transfer learning. MedRxiv pp. 2020–08 (2020)
62. Zhou X, Li Y, Liang W (2020) Cnn-rnn based intelligent recommendation for online medical pre-diagnosis support. IEEE/ACM Trans Computat Biol Bioinf 18(3):912–921
63. Basiri ME, Nemati S, Abdar M, Cambria E, Acharya UR (2021) Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. Fut Generat Comput Syst 115:279–294
64. Kollias D, Zafeiriou S (2020) Exploiting multi-cnn features in cnn-rnn based dimensional emotion recognition on the omg in-the-wild dataset. IEEE Trans Affect Comput 12(3):595–606
65. Nasir JA, Khan OS, Varlamis I (2021) Fake news detection: a hybrid cnn-rnn based deep learning approach. Int J Inf Manag Data Insights 1(1):100,007
66. Babaee M, Li Z, Rigoll G (2019) A dual cnn-rnn for multiple people tracking. Neurocomputing 368:69–83
67. Khaki S, Wang L, Archontoulis SV (2020) A cnn-rnn framework for crop yield prediction. Front Plant Sci 10:1750
68. Hamdi Y, Boubaker H, Rabhi B, Ouarda W, Alimi A (2021) Hybrid architecture based on rnn-svm for multilingual online handwriting recognition using beta-elliptic and cnn models. TechRxiv
69. Zhao Hh, Liu H (2020) Multiple classifiers fusion and cnn feature extraction for handwritten digits recognition. Granular Comput 5:411–418
70. Xu G, Liu M, Jiang Z, Söffker D, Shen W (2019) Bearing fault diagnosis method based on deep convolutional neural network and random forest ensemble learning. Sensors 19(5):1088
71. Xie W, Li Z, Xu Y, Gardoni P, Li W (2022) Evaluation of different bearing fault classifiers in utilizing cnn feature extraction ability. Sensors 22(9):3314
72. Zhang J, Kamata Si, Ueshige Y (2006) In: International Workshop on Intelligent Computing in Pattern Analysis and Synthesis (Springer, 2006), pp. 290–299
73. Kingma DP, Ba J, (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
74. Categorical crossentropy (2004). https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy. (Last accessed: 29/10/2021)
75. Hawkins DM (2004) The problem of overfitting. J Chem Inf Comput Sci 44(1):1–12
76. Keras (2015). https://keras.io/. (Last accessed: 03/11/2021)
77. Tensorflow (2015). https://www.tensorflow.org/. (Last accessed: 03/11/2021)
78. Gorynin I, Gangloff H, Monfrini E, Pieczynski W (2018) Assessing the segmentation performance of pairwise and triplet Markov models. Signal Process 145:183–192
79. Derrode S, Pieczynski W (2016) Unsupervised classification using hidden Markov chain with unknown noise copulas and margins. Signal Process 128:8–17

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.