DOI 10.15622/ia.2020.19.6.4

C. SARMIENTO , J. SAVAGE

COMPARISON OF TWO OBJECTS CLASSIFICATION TECHNIQUES USING HIDDEN MARKOV MODELS AND CONVOLUTIONAL NEURAL NETWORKS

Sarmiento C., Savage J. **Comparison of Two Objects Classification Techniques using Hidden Markov Models and Convolutional Neural Networks.**

Abstract. This paper presents a comparison between discrete Hidden Markov Models and Convolutional Neural Networks for the image classification task. By fragmenting an image into sections, it is feasible to obtain vectors that represent visual features locally, but if a spatial sequence is established in a fixed way, it is possible to represent an image as a sequence of vectors. Using clustering techniques, we obtain an alphabet from said vectors and then symbol sequences are constructed to obtain a statistical model that represents a class of images. Hidden Markov Models, combined with quantization methods, can treat noise and distortions in observations for computer vision problems such as the classification of images with lighting and perspective changes.

We have tested architectures based on three, six and nine hidden states favoring the detection speed and low memory usage. Also, two types of ensemble models were tested. We evaluated the precision of the proposed methods using a public domain data set, obtaining competitive results with respect to fine-tuned Convolutional Neural Networks, but using significantly less computing resources. This is of interest in the development of mobile robots with computers with limited battery life, but requiring the ability to detect and add new objects to their classification systems.

Keywords: Hidden Markov Models, Image Classification, Computer Vision, Pattern Recognition

1. Introduction. In the field of computer vision, it has always been required to interpret visual content captured in sensors, providing information to the systems to carry out tasks that are useful. One of these tasks is to recognize what types of objects are in a work environment.

By knowing the meaning of the data that comes from cameras, it's possible to determine the current state of the environment such as knowing where an object is or not in a certain region of space. With this, it's possible to carry out planning, navigation, and manipulation tasks in robotic systems.

Automatic planning or briefly called planning can be described as a set of techniques to represent knowledge, calculate a temporal sequence of actions, and obtain a final configuration to complete one or more tasks. It's common to establish a representation of the internal state of a robotic system and a representation for the environment, by calculating a sequence of actions, a transition between said states is achieved until reaching a terminal state, this last configuration represents the execution of one of several goals.

Autonomous navigation is the ability of a robotic system to estimate its position, calculate a trajectory to a given target in the environment, and

follow a dynamically generated path in order to avoid moving obstacles. To achieve this, it's necessary to build a representation of the environment such as a map, recognize distinctive markings to self-locate through the use of sensors or vision systems and finally estimate a trajectory to the place that has been selected as a destination.

Robotic manipulation, in the context of robots operating in service tasks, refers to the design of control systems that operate mobile manipulators in spaces shared with humans. These places have a certain degree of structure such as a kitchen, a living room, or a bedroom but to carry out manipulation of objects in these environments it's necessary to have computer vision systems. Computer vision can facilitate the control of actions on the manipulator, obtaining descriptions of the shape or texture that would be difficult to do using other types of perception.

Systems based on computer vision can perform tasks such as creating a representation of the environment, self-locate and navigate autonomously, estimate the best way to take an object, all this through the use of digital cameras. This has helped the integration of planning, navigation and manipulation systems in mobile robots to service to people.

The first computer vision systems were based on the extraction of visual features, these had to be robust to changes in lighting, perspective, and scale which were designed manually. These representations were later classified using supervised or unsupervised machine learning techniques, or in variants and compositions of these two options [1].

At present, the most accurate image classification systems are based on what is called Artificial Neural Networks (ANNs). The artificial neurons are used as fundamental building blocks to create new structures, hierarchically and with locality properties in their arrangement, in what is called Convolutional Neural Networks (CNNs) [2].

CNNs are the image detection systems with highest performance [3], with varied and specialized architectures depending on the problem where they are applied. A disadvantage of these detection systems is that the mathematical theory on which they are based is the gradient descent optimization theory [4,5], which requires highly dimensional differentiable functions. This is a problem in terms of computing requirements since it's necessary to store millions of parameters, perform calculations on each of them, and update their values. Due to a large number of parameters very small learning rates are used to maintain convergence and stability, this is a disadvantage since many examples are required to achieve a balance between the generalization of the system and it's precision.

Service robots have gained relevance in recent years [6, 7]. This is because with current technology it's already possible to assemble robots for common tasks such as surveillance, customer service, disinfection of spaces or even home delivery [8]. As a result of research in this branch of robotics, competitive communities have emerged such as RoboCup, which is an international scientific initiative that encourages research and communication of discoveries and its applications.

Mobile service robots must be autonomous, the use of energy and computing must be more efficient to extend the service time that is provided. Currently, these systems use specialized computers with low power consumption and with architectures that allow CNNs to be executed in short intervals of time, some systems, even use laptops as a common rule [9]. In this context, this research shows that it's possible to use discrete Hidden Markov Models (HMMs) as an inference method for classifying images using a data set with a reduced number of classes, achieving similar results to CNNs but using generic computers such as a laptop, with the advantage of executing the classification faster and therefore consuming fewer resources.

The remaining of the paper is divided as follows. In Section 2 we present a summary of HMMs applications for computer vision. Then, in Section 3 we introduce our probabilistic approach to solve the image recognition task and in Section 4 we describe experimental results. Finally, the main findings are discussed in Section 5.

2. Related Work. Hidden Markov Models (HMMs) are described in the writings of L. E. Baum [10], in them the theory is described to use Markov chains as a tool to analyze time series. Later its use becomes popular due to the applications found by L.R. Rabiner [11] for speech recognition. In his work, numerical stability techniques for algorithms, preprocessing techniques, and details about the implementation are documented. In [12], a great variety of applications and variations to the original method is compiled. However, HMMs have mainly been used for recognition of symbol sequences in the area of bioinformatics, handwriting recognition, and as a method of pattern recognition for electrical signals in biomedicine.

In the case of recognition from digital images, there are applications such as face recognition [13], in which a number of different architectures are described, such as using one-dimensional HMMs but allowing transitions between various stacked models. This represents different types of sequences as they are in order from top to bottom and from left to right. The symbols are obtained by applying a fixed-size sliding window and calculating the discrete cosine transform over pixel values. Clustering techniques were used to obtain a

discrete alphabet, which is a common practice to discretize observation vectors, in this case, obtained by the coefficients.

The application of HMMs for thermal image classification is shown in [14], which describes its application to classify images of breast cancer. This work merges grayscale images and thermal images represented in the RGB color space. Through binarization of constant size windows on the image, a unique symbol of each section is obtained. Finally, an HMM with two hidden states classifies a sequence as diseased or healthy tissue.

Another application in the field of medical imaging is written in [15]. This work describes the application of HMMs for the detection of blood vessels, where the HMMs have the function of estimating the next most probable state in the images to remove pixel occlusions in binarized retinal images.

In a previous work [16], we expanded the application of HMMs for point clouds detection. To carry out the classification of objects and places, 3D keypoints are detected, and using 3D descriptors it's possible to obtain observation vectors. One HMM was trained with these sequences for each class of location or object, and a pattern was detected. A disadvantage is that 3D descriptors are usually vectors with many dimensions, slowing down the quantization process to obtain a discrete alphabet. This can be solved by using a more compact representation.

The year 2010, an image classification challenge arises as a proposal to improve the performance of classification systems. Today this is known as the ImageNet challenge [17]. This is a benchmark to compare detection systems on millions of images with hundreds of different categories.

In the case of CNNs, they are based on the paradigm of having groups of neurons emulating the convolution operation by applying filter banks. The year 2012 training process using graphics processors was implemented, resulting in an inference design highly dependent on parallelization with excellent results such as AlexNet [18], which was the architecture with the best recognition rate in the ImageNet data set. For this reason, most of the image classifiers are of the CNN type.

Over time, variants with more parameters emerged, such as the so-called VGGNet in 2014 [19], an architecture that manages to reduce the percentage of detection error to less than 8%, obtaining a first place that year in the ImageNet challenge. With the increase in the number of parameters, new problems appear such as the vanishing of gradient values due to the number of stacked layers. To solve this, a new architecture is proposed, such as residual blocks [20] and therefore, the authors manage to add a greater number of layers without degrading the accuracy of the classification system, obtaining an error

close to 3 % in the ImageNet challenge of the year 2015. This architecture was called ResNet.

Today, there are many variations to the architecture. Some have improvements such as being optimized to require fewer parameters and computing time thinking of mobile devices [21]. In this work, we will use VGGNet and ResNet as a reference to compare HMMs in the same task, classification of images, in a domestic service context.

3. HMM for Image Recognition. A Hidden Markov Model (HMM) is a set of discrete or continuous variables, representing a time-series observed $\{o_1, o_2, ..., o_T\} = o_{1:T}$. These variables $o_{1:T}$ are assumed to be generated by an internal configuration that changes successively over time $\{s_1, s_2, ..., s_t\} = s_{1:T}$ but which we cannot observe, so they are assumed to be hidden variables. The probability that an observation o_t is presented at a certain time t only depends on the current state, that is

$$p(o_t \mid o_{1:T}, s_{1:T}) = p(o_t \mid s_t). \tag{1}$$

The stochastic process associated with the states is causal and stationary with a finite number of states, so it's represented as a finite automaton with edges labeled with the probability of transiting from one vertex to another. This process only depends on state s_t at time t, and the previous state s_{t-1}, expressed as follows:

$$p(s_t \mid s_{1:T}) = p(s_t \mid s_{t-1}). \tag{2}$$

The probability of observing a sequence of variables $o_{1:T}$ in a fixed period T is given as

$$p(o_{1:T}, s_{1:T}) = p(o_1 \mid s_1) \prod_{t=2}^{T} p(o_t \mid s_t) p(s_t \mid s_{t-1}). \tag{3}$$

Where the probability of transit between internal states s_{t-1} to s_t is defined as

$$p(s_{t=j} \mid s_{t-1=i}) = a_{ij}. \tag{4}$$

The probability of observing a variable o_t depends on hidden state s_t, and is defined as:

$$p(o_{t=j} \mid s_{t=i}) = b_{ij}. \tag{5}$$

The parameters of the model are listed below:

– a finite set of N states $S = \{s_1, s_2, ..., s_N\}$;
– a finite alphabet M, with symbols $V = \{v_1, v_2, ..., v_M\}$;
– an initial probability distribution for each state $\pi = [\pi_1, \pi_2, ..., \pi_N]$;
– a transition matrix $\mathbf{A} = \{a_{ij}\}$, with $i, j \in [1, N]$;
– and an emission matrix $\mathbf{B} = \{b_{nm}\}$ with $m \in V$, in some state $n \in S$.

Briefly the parameter set of an HMM is denoted as $\lambda = (S, V, \mathbf{A}, \mathbf{B}, \pi)$.

In order to be able to use an HMM for image classification, we can establish a temporal order that corresponds uniquely with visual features located in the image. This is possible if we fragment each image into regions of constant size and always generate observations with the same trajectory. Since each image will be different in brightness and perspective, elevation angle, and rotation angle (using as reference the object captured in each image), an HMM is ideal for finding a statistical description of the image.

3.1. Image Feature Extraction. We propose to find a representation automatically using clustering techniques. For this, fixed-size windows $w^t_{n,m}$ are established on the image, and from these, vectors are obtained by concatenating the three channels of the image, forming an observation vector o_t of size $1 \times 3 \times n \times m$. Figure 1 shows these sections.

A visual alphabet is built in an unsupervised way using the mini-batch k-means clustering method [22] for all vectors o_t that are selected as training samples. The clusters obtained assign an index to each vector o_t according to a similarity metric (euclidean distance is chosen). The quantization process has the objective of eliminating noise, obtaining centroids representing the average of the content in each window, which eliminates disturbances.

Fig. 1. Example of the sections created to obtain visual features (images from [23]). a) Three windows with size $n = 74$, $m = 224$; b) Six windows with size $n = 74$, $m = 112$; c) Nine windows with size $n = 74$, $m = 74$

This method enables an automatic but distinctive visual content representation between different objects, quickly and with compact representation because only clusters centroid list is stored. Then, each vector can be statistically treated using an HMM to generate a probabilistic model for each image

class. The trained classes generate a higher probability, and observations of other classes or untrained examples generate a lower probability or zero tendencies. The next section explains this procedure.

3.2. HMMs for Feature Detection. In computer vision, Hidden Markov Models (HMMs) are a useful tool because they model the appearance of symbols $o_{1:T}$ that we want to use as comparison patterns. These patterns can be built by associating a sequence of windows $w_{n,m}^{t:T}$ on a image at each instant t, so we can say that the sequence $\{o_1, o_2, ..., o_T\} = \{w_{n,m}^{t=1}, w_{n,m}^{t=2}, ..., w_{n,m}^{t=T}\}$ represents a complete unit. Where m, n are the window dimensions, and the variable t assign a fixed path on the image as time increases.

The architecture selected to model the sequences is the so-called ergodic HMM. This structure is used since it allows to obtain the probability of transition between all states, avoiding constraints on model dynamics. The Figure 2 shows this configuration between states. The number of states T is associated with the number of symbols that describe each image, and this number can be arbitrarily long, but to reduce processor usage time in an autonomous robotic system, it's preferable to use a small number of symbols in each sequence. This also makes it easy to scale a detection system to new classes without degrading response time.

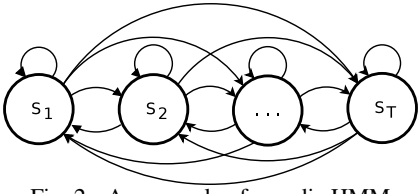

Fig. 2. An example of ergodic HMM

The detection system used is based on a set of HMMs, where each one model λ_k has been trained with examples of sequences that correspond to images of a single class k. This allows learning a distribution of these sequences and assigning a single model. In the end, given a test sequence $o_{1:T}$, each model λ_k is evaluated, and the one with the highest probability is the one that best explains the dynamics of the sequence. This is defined as

$$_k\{p(o_{1:T} \mid \lambda_k)\}. \tag{6}$$

Figure 3 illustrates this procedure. This method is efficient since inference can be made in parallel as well as training.

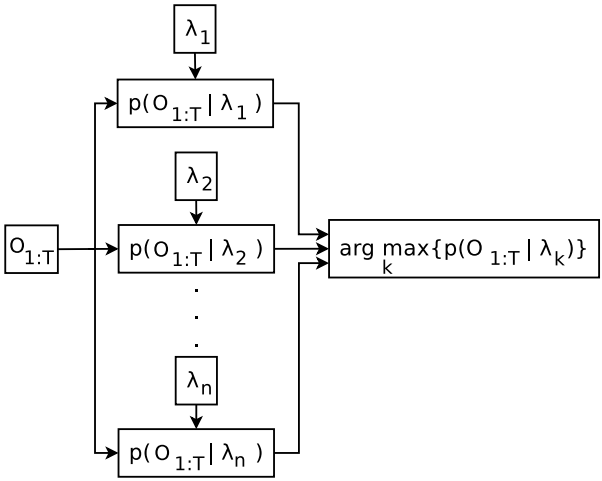

Fig. 3. Block diagram of the inference system using HMMs

The following section shows how it's possible to obtain a classification with enough precision to compete with pre-trained and fine-tuned CNNs, maintaining equally competitive training and inference times, using only a laptop processor.

4. Experiments and Results. For experiments, the Hidden Markov Model (HMM) shown in Figure 2 has been used. This architecture has been chosen since there is no predefined model that describes transitions between each window $w^t_{n,m}$ created in each image, so a transition between all the states is allowed. This architecture is used with $T = 3$, $T = 6$ and $T = 9$.

The data set published in [23] is used since the objective is to test our method in a context of a mobile service robot, where objects to train and detect are those that could be found inside a house, distributed in different classes.

This data set is made up of 51 classes of objects and each class has a variable number of instances, but only one instance of each class is used in this work.

If necessary, a detection system based on HMMs can be scaled to new classes without the need for retraining but simply by adding new models λ_k to the structure shown in the Figure 3. This applies equally to instances of the same objects.

The data set images were captured with a fixed RGB-D camera, a rotating base and each class present an unbalanced number of images. Images are captured with an object rotation between 0 and up to 720 degrees, so

they are sub-sampled in half at regular intervals to obtain an object coverage of approximately 0 to 360 degrees. This subset also contains images with an elevation angle of 30 and 60 degrees with respect to the object, different lighting and perspective. Figure 4 shows an example of the training images used.

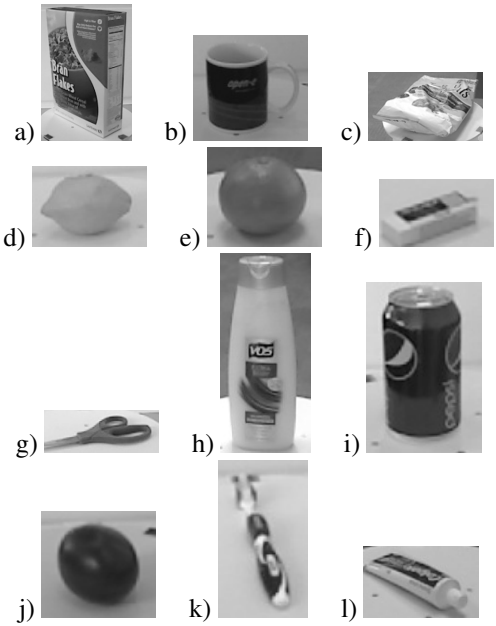

Fig. 4. Twelve examples of objects contained in the data set for testing [23]: (a cereal_box; b) coffe_mug; c) food_bag; d) lemon; f) orange; g) rubber_eraser; h) scissors; i) shampoo; j) soda_can; k) tomato; k) toothbrush; l) toothpaste. All classes are listed in the first column of Table 9 to Table 13

For the tests three recognition system architectures have been used. The first architecture is based on work of Rabiner [11] and is shown in Figure 5. The second and third architectures are shown respectively in the Figure 14 and Figure 15, the latter are a composition of the first architecture in what is known as an ensemble method. This is described in Section 4.3.

For the first architecture, a rescaling is applied to each image in the training set, and this new scale is 224 by 224 pixels. Then each image is transformed into the CIE Lab color space [24] and three, six, and nine sections of the image were created.

For the case where there are three windows in the image, vectors of size $1 \times (74 \times 224 \times 3)$ are obtained, where $w^t_{74,224}$ are the size of the window. The factor $\times 3$ appears since there are three channels in the image corresponding to channels L, a y b. In the case of extracting six windows with size $w^t_{74,112}$, there are vectors of size $1 \times (74 \times 112 \times 3)$. For nine windows with size $w^t_{74,74}$ the vectors are of dimension $1 \times (74 \times 74 \times 3)$.

To quantize all vectors scikit-learn library was used [25], it incorporates optimizations based on openMP, the use of AVX2 instructions (Advanced Vector Extensions 2) and LaPACK (Linear Algebra PACKage) library support. Mini batch k-means algorithm [22] is used, and it's initialized with k-means++ for faster convergence [26]. The number of centroids was predefined with values 64, 128, 256 and 512. Furthermore, mini-batch k-means is implemented using the Elkan algorithm [27], this algorithm uses triangular inequalities, so it's not necessary to search for a new vector through all centroids and mini-batch k-means avoids doing searches using all data (it samples mini-batches speeding up calculations). The quantization process is performed offline.

The symbol sequences are obtained by associating an identifier index of the centroids with a vector to be compared, which generates a sequence $o_{1:T}$ for each image. These sequences are generated to train each model λ_k or during the inference process. Each model is trained by each category of objects in the data set and subsequently stored as a later use file. In the end, the model λ_k that has the highest probability calculated given the sequence $o_{1:T}$ is the one that assigns the detected class. The architecture of Figure 5 shows the proposed detection system.

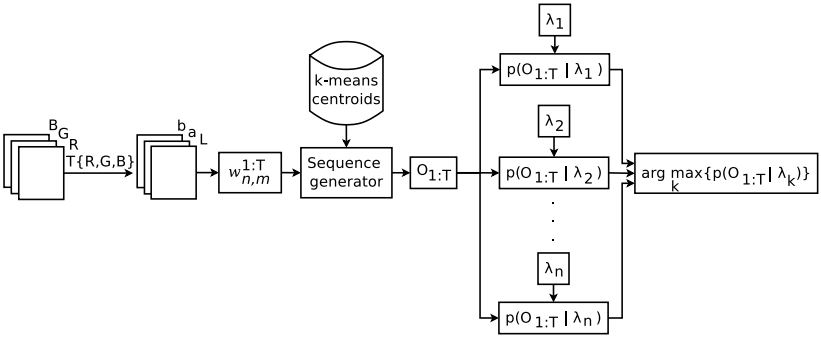

Fig. 5. Block diagram of the recognition system. A transformation from RGB to CIE Lab color space, construction of sections in the image, and construction of symbol sequences to be subsequently detected by a set of previously trained HMMs

The experiments were organized as follows:

– an HMM with three states for each class of images. A quantizer for 64, 128, 256, and 512 centroids;

– an HMM with six states for each class of images. A quantizer for 64, 128, 256, and 512 centroids;

– an HMM with nine states for each class of images. A quantizer for 64, 128, 256, and 512 centroids.

The experiments were performed five times, using the subset of images that cover the object from approximately 0 to 360 degrees, partitioned into 75% for training and 25% for testing. In the case of CNNs, 10 % of the data set is taken for training validation but from the section for testing.

For the training set, all images are randomly preprocessed with one of the following transformations: rotations of up to 25 degrees, horizontal and vertical displacement of 10% with respect to the center of the original image, skew transformation of 20% on the horizontal and vertical axis, reflections in the horizontal and vertical axis and zoom of up to 10%. The validation set (only for CNNs) and testing set (HMMs and CNNs) does not include transformations. All this in order to compare performance against CNNs typically trained with data augmentation techniques.

A laptop with 32 GB of RAM, a model GeForce 1070 with 8 GB GPU, and a 4-core seventh-generation CPU was used for testing. The results are reported as weighted Average Precision (wAP) and weighted Average Recall (wAR) due to imbalance in the number of images between classes. Accuracy, precision, recall, and F1-score are also reported [28].

Weighted Average Precision is defined as follows:

$$\frac{1}{\sum_{i=1}^{K}\hat{y}_i}\sum_{i=1}^{K}\hat{y}_i P_i. \tag{7}$$

Weighted Average Recall is defined as:

$$\frac{1}{\sum_{i=1}^{K}\hat{y}_i}\sum_{i=1}^{K}\hat{y}_i R_i. \tag{8}$$

The index K is the total number of classes and $\hat{y}_i$ is the total number of images labeled as class i.

Precision P_i and Recall R_i are defined for each class i as:

$$P_i = \frac{t_p}{t_p + f_p}. \tag{9}$$

$$R_i = \frac{t_p}{t_p + f_n}. \tag{10}$$

Where t_p means true positives, f_p false positives and f_n false negatives.

4.1. HMMs Results. Figure 6 shows the qualification result for 51 object classes. Of 5 experiments carried out, weighted Average Precision (wAP) is reported, this since each class of objects contains a different number of images, so imbalance must be taken into account, weighting the precision for each class with respect to the number of classified images. In the same way, Figure 7 shows the detection average for weighted Average Recall (wAR).

The highest percentage of recognition is carried out by HMMs using 512 symbols as an alphabet, as is shown in Figure 6. The use of three states to form models λ_k results in a wAP value equal to 0,97, but wAR does not increase above one hundredth from models with six and nine states. Therefore, using three states increases the detection of positive cases but does not increase for negative cases, as shown in Figure 7.

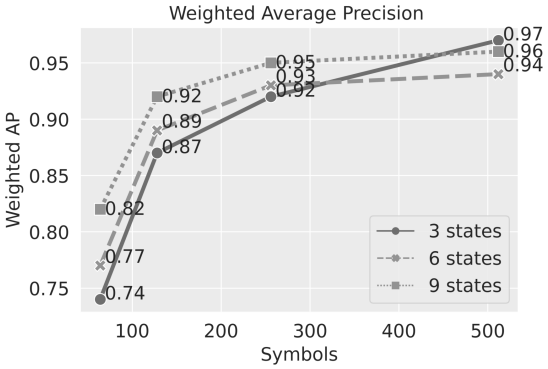

Fig. 6. Weighted Average Precision for all HMMs

Figure 8 shows the processing time used to obtain an alphabet using the mini-batch k-means method. For the case of 512 symbols and three states, the peak processing time is shown close to 220 seconds, this is due to the fact that the chains with three states use the largest vector sizes, so the clustering process is slower.

Table 1 in row one shows each vector's average query time obtained from each window. The average time used to obtain the sequence of symbols of each image is included in row two. Only the case for 512 symbols is shown, since it's the case with the slowest access of all the symbol variants. The query

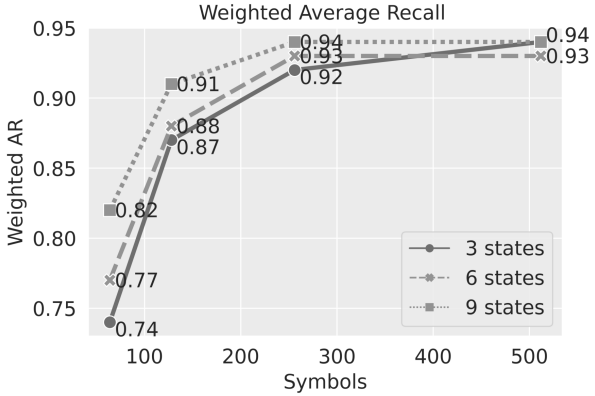

Fig. 7. Weighted Average Recall of all HMMs

process reuses the Elkan method, so on average, the time of 2.7 milliseconds in the worst case is never exceeded.

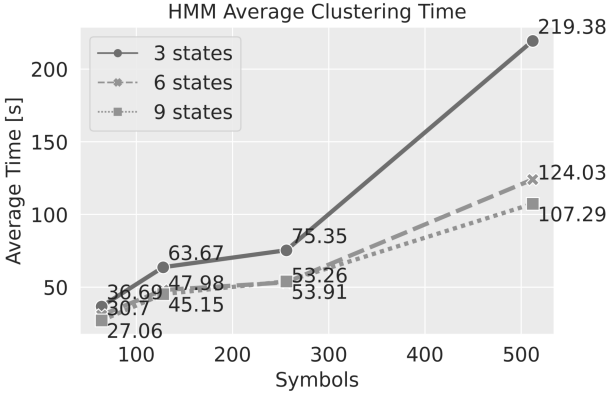

Fig. 8. Average clustering time for HMMs

Table 1. Average Query Time for 512 symbols

	3 windows	6 windows	9 windows
single vector	0,00088 s	0,000425 s	0,000288 s
single image	0,002639 s	0,002551 s	0,00259 s

In the case of training time, HMMs with the highest number of states, such as those with six and nine states, are the ones that take the longest, this is shown in Figure 9. Even so, training time does not exceed 0,200 seconds in the worst case, on average for each HMM.

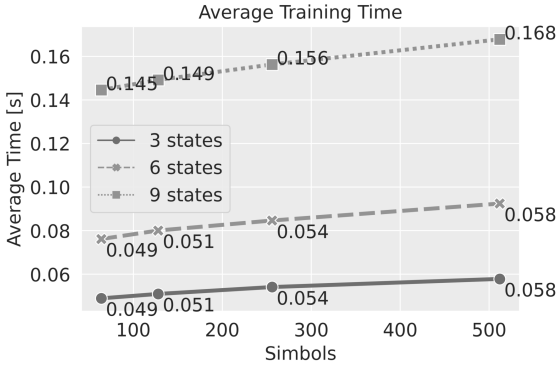

Fig. 9. Average training time for HMMs

Inference time depends on the number of states and the number of transitions between them. Times range from approximately $2.54x10^{-6}$ seconds to $16.4x10^{-6}$ seconds as shown in Figure 10.

Less number of symbols generates similarity between learned distributions, so it's more likely to go through many states until the end of a complete sequence, this increases inference time for 64 and 128 symbols. On the contrary, more symbols generate more distinctive observations, with more differentiated statistics, so that when evaluating an input sequence, fewer states are passed, resulting in a shorter inference time for 256 and 512 symbols. It's useful to see how fast an image can be classified using HMMs. The inference time do not exceed $20x10^{-6}$ seconds on average in any HMM variant and number of symbols.

The amount of memory used in critical parts of the system was monitored. Creating an alphabet using larger image sections (as was done in the case of three states in each HMM) has as a consequence a high consumption of system memory because all the data must be stored for the queries reaching a peak of approximately eight giga bytes of occupation as shown in Figure 11.

In comparison, HMMs with six and nine states use smaller vectors that do not exceed the size of two point one gigabytes of memory.

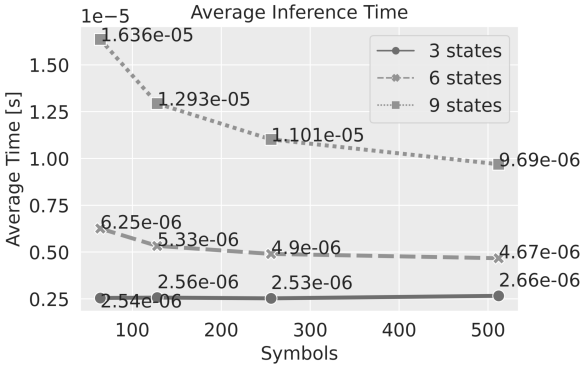

Fig. 10. Average inference time for HMMs

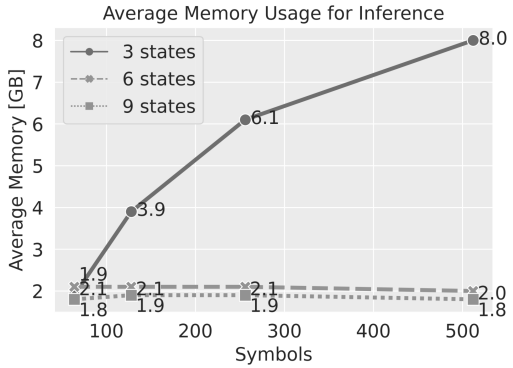

Fig. 11. Average memory usage for inference with HMMs

For the memory used in the clustering process, again the HMMs with three states demand storage to build the corresponding alphabet as seen in Figure 12. It reaches the memory occupancy peak of up to twenty-two point eight gigabytes, which is a disadvantage. The Figure 6 shows that this three-state model with 512 symbols is the one with the highest wAP but requires excess memory for both inference and clustering.

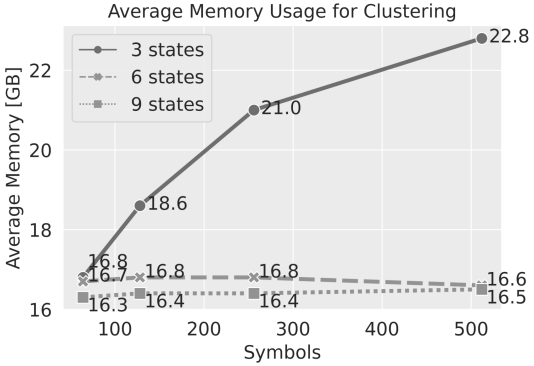

Fig. 12. Average memory usage for clustering

In Section 4.3, we will explain how to solve the memory consumption problem by discarding models with high memory occupancy and high computation time to form an HMMs ensemble.

4.2. CNNs Results. In the case of Convolutional Neural Networks (CNNs), two variants were used. The first called VGG-19 [19] and the second ResNet-50 [20]. These are chosen due to their high level of accuracy in tests carried out in the ImageNet challenge, in addition to the fact that previously trained coefficients are in the public domain included in the latest version of the library named TensorFlow [29].

VGG-19 and ResNet-50 were tested by freezing pre-trained coefficients obtained from ImageNet data set. Only classification layers were removed, and two hidden and fully connected layers were added with 256 neurons. A softmax classification layer for 51 classes is also added and data augmentation is used as described in Section 4.

These architectures were designed to classify millions of images, but they perform well in this compact data set. Table 2 shows the weighted Average Precision (wAP) in first row, weighted Average Recall (wAR) is reported in second row. VGG-19 achieves a wAP of 1,0 and a wAR of 1,0. On the other hand, Resnet-50 achieves 0,96 in wAP and 0,94 in wAR through all five experiments.

Table 2. CNNs Average Results

	VGG-19	ResNet-50
Weighted Average Precision	1,0	0,96
Weighted Average Recall	1,0	0,94

For the case of training time and inference time, due to the massive amount of parameters, they turn out to be slower than any HMM. Table 3 shows in row one that VGG-19 has an average inference time per image close to 0,35 seconds, and an average training time of 1882.6 seconds is reported in row two. For ResNet-50 inference, time is 0,183 seconds, and an average training time of 2846.8 seconds is shown in the second column.

Table 3. Time for Training and Inference for CNNs

	VGG-19	ResNet-50
Average Inference Time	0,3498 s	0,183 s
Average Training Time	1882.6 s	2846.8 s

The memory consumption of these classifiers is composed of occupation in GPU and system memory. The two architectures use the same amount of system and GPU memory during training and inference, as reported in Table 4.

Table 4. Memory usage for CNNs

	VGG-19	ResNet-50
GPU	7.8 GB	7.8 GB
System	2.8 GB	2.8 GB

The Figure 13 shows a wAP and wAR comparison of the best HMMs with respect to VGG-19 and ResNet-50 at the end of the experiments. In the case of wAP, VGG-19 gets the best test performance, scores 1,0 in 5 experiments. Behind this classifier is the HMM with three states and 512 symbols. The HMM with nine states and 512 symbols and ResNet-50, the latter with similar capacity to recognize positive and negative images of each class. The six-state, 512-symbol HMM lags behind all with a wAP equal to 0,94.

In the case of wAR, Figure 13 shows each value with a triangular marker. ResNet-50 and HMMs with three and nine states obtain a value equal to 0,94, and the HMM with six states obtain 0,93. From these results, it can be concluded that CNNs perform well in detecting positive and negative examples. However, HMMs are more likely to make mistakes in positive examples classified as negative, which decreases the recall value.

4.3. HMMs Ensemble. During the experiments, 51 HMMs were trained, these also in four varieties of symbols and three types of architectures, giving a total of 612 trained HMMs (51x4x3 models). So the question arises: Is it possible to reuse some of these HMMs to create a more robust classification system? The answer is developed below.

First, we must establish that not all HMMs are useful for this purpose since models with three states make extensive use of memory and processing

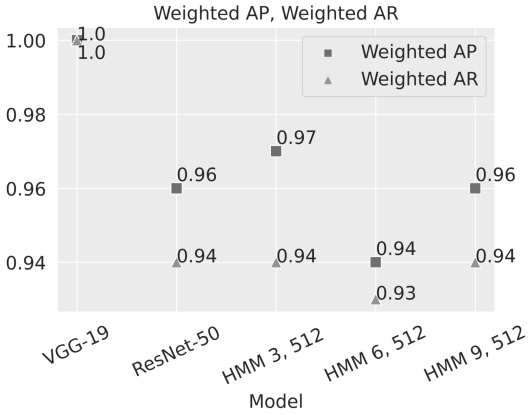

Fig. 13. Weighted Average Precision and Weighted Average Recall for best models. HMM 3, 512 for three states and 512 symbols. HMM 6, 512 for six states and 512 symbols. HMM 9, 512 for nine states and 512 symbols

time in their variants of 512 symbols. However, models with nine states and 512 symbols are the most accurate. Thus, HMMs with less memory use and, if possible, with greater precision and recall were selected. The only HMMs that simultaneously fulfill these two characteristics are the chains of short length or small observation vectors, the set of models with three states and 64 symbols, and the models with nine states and 512 symbols.

This results in using three types of HMMs, two types of symbols (64 and 512), and 51 models for classification, so it's proposed to make an ensemble of 306 HMMs. This might seem implausible, but because HMMs are compact in memory during the inference process in addition to a speed in the order of microseconds, 306 sequentially evaluated models must have a lower execution time than a CNN like VGG-19 or ResNet-50. The number of parameters for VGG-19 is approximately 143.6 million and for ResNet-50 it's 25.6 million for the ImageNet data set. For the case of our HMMs ensemble, it can be calculated taking into account the following: the size of each vector (having three, six, and nine windows) and the number of symbols to store (64 and 512 symbols). This is [74x224x3x(64+512)] + [74x112x3x(64+512)] + [74x74x3(64+512)] which results in approximately 52.4 million parameters to store the centroids. In the case of 306 HMMs, we have less than 0,55 million parameters counting each matrix $\mathbf{A}_k$, $\mathbf{B}_k$ and π_k in each model λ_k.

It's important to note that the data set to obtain the extraction of visual features in CNNs is enormous, close to 1,3 million images (ImageNet data set), which makes its direct application difficult in systems that do not have such a large amount of data. The HMMs allow training with a compact data set, and in a modular way since the performance of a classification system is increased by adding or removing models λ_k, adding or removing lists of centroids obtained in the clustering process, and there is no dependence between models during their training. On the contrary, in the case of a CNN, retraining is necessary to add or remove classes.

To construct the ensembles, the approach of a soft and hard classifier was used. The soft classifier calculates a joint probability on all models already trained and selects a class with the highest joint probability. This system is shown in Figure 14. The hard classifier works as HMMs do, selecting the model with the highest probability but with the difference that now one vote per class is accumulated. The most voted class is the selected class, Figure 15 shows this system.

Table 5 shows in the first row the wAP result of the ensembles. These two new classifiers achieve a wAP value of 0,98. For the case of wAR which is shown in the second row, the classifier with soft-decision lags behind with a wAR of 0,95.

Thus, it's confirmed that it's possible to ensemble HMMs with a different number of hidden states to increase the overall performance. Ensemble 2 obtains a wAP equal to 0,98 and a wAR equal to 0,98, lagging behind VGG-19 with scores equal to 1,0.

Table 5. Results for HMMs Ensemble

	HMM Ensemble 1	HMM Ensemble 2
Weighted Average Precision	0,98	0,98
Weighted Average Recall	0,95	0,98

Inference time when assembling the models is triggered, reaching values of up to 0,0022 seconds for the best ensemble case. This is reported in Table 6. The HMM ensemble, even evaluated sequentially, is faster than an average of 0,183 seconds of inference per image in ResNet-50.

Table 6. Average Inference Time

	HMM Ensemble 1	HMM Ensemble 2
Average Inference Time	4.410^{-5} s	0,002237 s

The memory usage increases because it's necessary to load all the centroids used by each HMM, but even with this, it only increases the memory

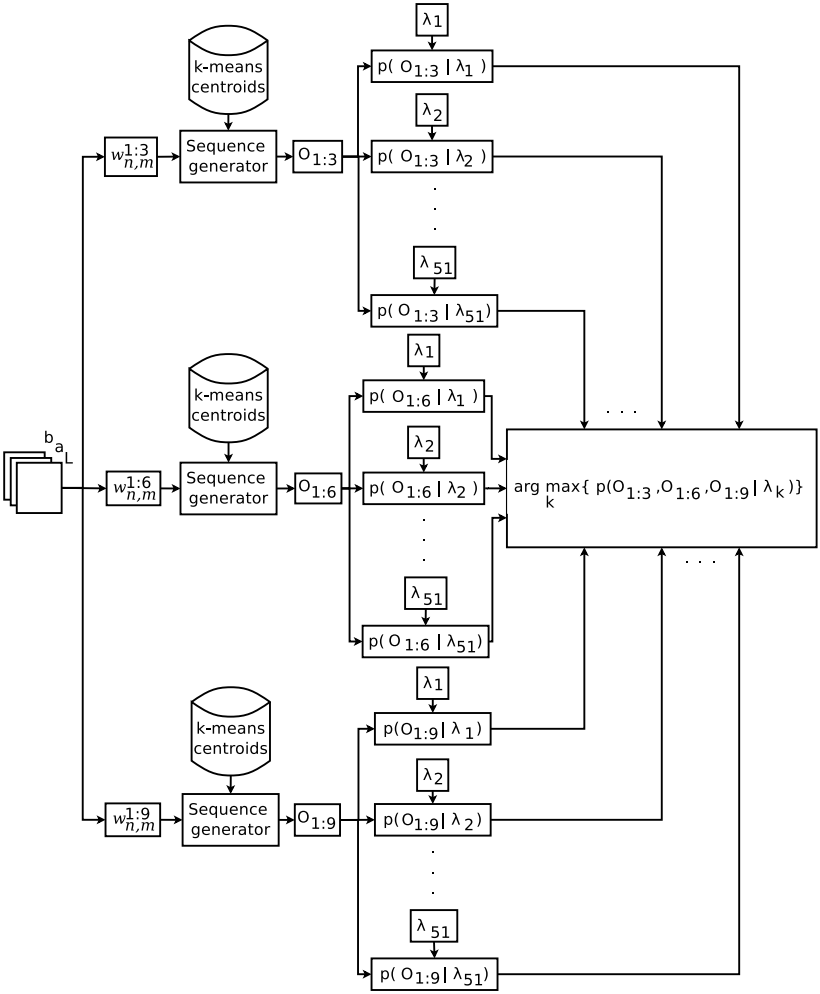

Fig. 14. Ensemble 1. A soft classifier based on HMMs

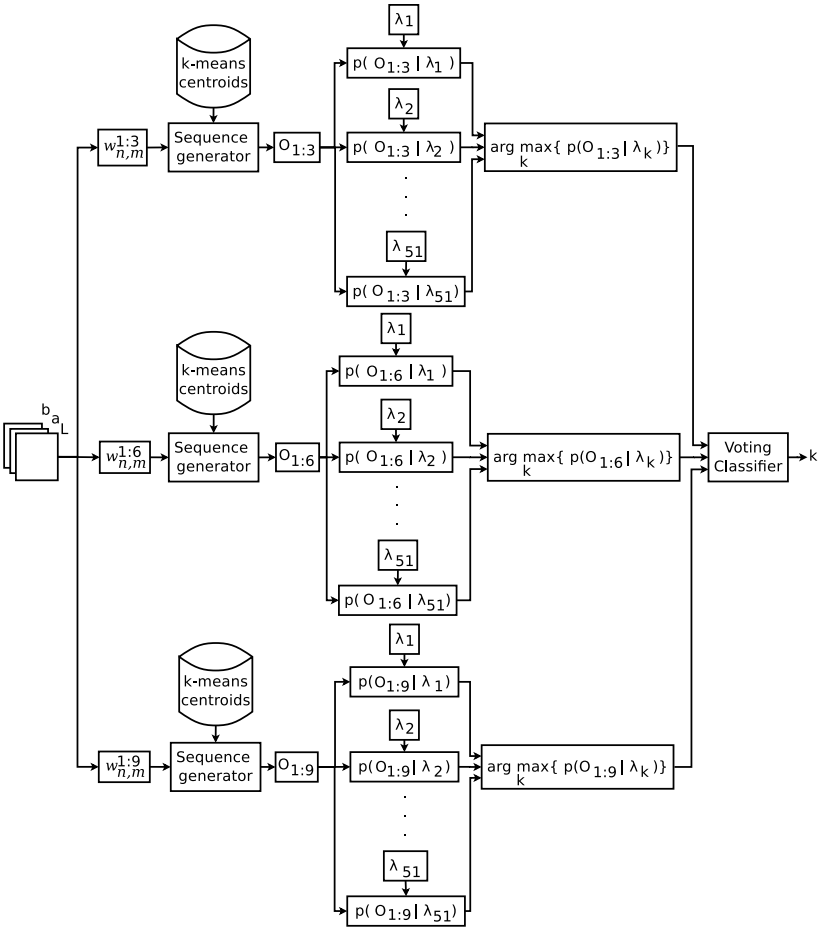

Fig. 15. Ensemble 2. A hard classifier based on HMMs

usage up to a limit of two point eight gigabytes on average, as reported in the Table 7 (a 32-bit representation is used as well as CNNs parameters).

Table 7. Average Memory Usage

	HMM Ensemble 1	HMM Ensemble 2
Average Memory Usage	2.8 GB	2.8 GB

Finally, a summary of wAP and wAR is shown in Figure 16 for the best models evaluated in this work. VGG-19 is the system with the most accuracy, precision and best recall, scoring 1,0 in all metrics. Table 9 presents the full report.

HMMs Ensemble 2 ranks second among the best classifiers, but this by far uses generic resources such as memory and CPU and turns out to be faster than VGG-19 and ResNet-50, verifying that HMMs are an effective tool for image classification (see Table 13 for full report).

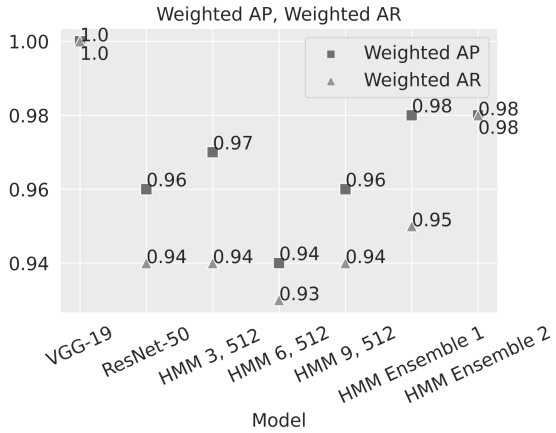

Fig. 16. Final metrics of best classifiers. HMM 3, 512 for three states and 512 symbols. HMM 6, 512 for six states and 512 symbols. HMM 9, 512 for nine states and 512 symbols

The other image classifiers developed have similar performance in the task. ResNet-50 has a wAP with 0,96 value, Table 10 shows the complete classification report.

The HMM of nine states and 512 symbols is reported with detail in Table 11. This is the model with less resource consumption and better detection metrics simultaneously. Also, the Table 12 shows the performance for ensemble 1. The HMM report with three states and 512 symbols is skipped

due to its high memory consumption, and HMMs with six states and 512 symbols due to its low precision.

5. Conclusions. In this paper, a practical comparison was presented using publicly available data. It has been demonstrated experimentally that the application of the theory of HMMs is useful to create a classification system for images containing objects. This task has been widely developed with the use of deep convolutional neural networks but these models, even with great precision, are complex to integrate into systems that require high energy efficiency due to the millions of parameters that must be adjusted and then transported to a final program.

Table 8 summarizes the three best models tested, their values of wAP, wAR, inference time for an image, and the number of parameters. Row three for the case of HMM Ensemble 2 is the time required to obtain the complete inference on a 224x224 size image. This time is the sum of 6 average queries with 512 symbols, which correspond to three different architectures of HMMs and two different types of symbols.

Table 8. Results comparison

	HMM Ensemble 2	VGG-19	ResNet-50
wAP	0,98	1,0	0,96
wAR	0,98	1,0	0,94
Avg. Time per Image	0,018071 s[1]	0,3498 s	0,183 s
Number of Parameters	52.95 millions[2]	143.6 millions	25.6 millions

Weighted Average Precision (wAP). Weighted Average Recall (wAR). [1] Query time for an image in 6 centroid lists with 512 symbols, plus run all 306 models. [2] All centroids for 64 and 512 symbols and 306 HMMs

The HMMs ensemble 2 had the best value of wAP and wAR of all proven HMMs, but without reaching the performance of VGG-19. Its main advantage is that although with the increase of parameters, it still achieves an execution speed ten times higher than ResNet-50 using a moderate performance CPU that only has four cores.

All the models achieve a good recognition capacity, equal to or greater than 0,94 in wAP also for wAR, but it's possible to build an ensemble with half of these models to reach up to 0,98 for both wAP and wAR. This result means a reasonable ability to distinguish between positive and negative examples. Note that tests demonstrate the power of the method using data augmentation and an unbalanced data set per class. Of course, a disadvantage of the method is that it requires the construction of a visual vocabulary using unsupervised methods, so it's not possible to modify the representation online. Using a more compact and distinctive visual representation, in addition to a segmentation subsystem, will be the reason for future work.

Table 9. Classification report for VGG-19

Class	Precision	Recall	F1-score	Support
apple	1,00	1,00	1,00	380
ball	1,00	1,00	1,00	515
banana	1,00	1,00	1,00	460
bell_pepper	1,00	1,00	1,00	395
binder	1,00	1,00	1,00	445
bowl	1,00	1,00	1,00	365
calculator	1,00	1,00	1,00	365
camera	1,00	1,00	1,00	395
cap	1,00	1,00	1,00	395
cell_phone	1,00	1,00	1,00	345
cereal_box	1,00	1,00	1,00	360
coffee_mug	1,00	1,00	1,00	345
comb	1,00	1,00	1,00	355
dry_battery	1,00	1,00	1,00	355
flashlight	1,00	1,00	1,00	380
food_bag	1,00	1,00	1,00	495
food_box	1,00	1,00	1,00	495
food_can	1,00	1,00	1,00	515
food_cup	1,00	1,00	1,00	490
food_jar	1,00	1,00	1,00	485
garlic	1,00	1,00	1,00	490
gluestick	1,00	1,00	1,00	505
greens	1,00	1,00	1,00	455
hand_towel	1,00	1,00	1,00	490
instant_noodles	1,00	1,00	1,00	485
keyboard	1,00	1,00	1,00	445
kleenex	1,00	1,00	1,00	490
lemon	1,00	1,00	1,00	370
lightbulb	1,00	1,00	1,00	390
lime	1,00	1,00	1,00	400
marker	1,00	1,00	1,00	505
mushroom	1,00	1,00	1,00	490
notebook	1,00	1,00	1,00	500
onion	1,00	1,00	1,00	485
orange	1,00	1,00	1,00	450
peach	1,00	1,00	1,00	425
pear	1,00	1,00	1,00	425
pitcher	1,00	1,00	1,00	355
plate	1,00	1,00	1,00	450
pliers	1,00	1,00	1,00	370
potato	1,00	1,00	1,00	380
rubber_eraser	1,00	1,00	1,00	495
scissors	1,00	1,00	1,00	390
shampoo	1,00	1,00	1,00	495
soda_can	1,00	1,00	1,00	395
sponge	1,00	1,00	1,00	400
stapler	1,00	1,00	1,00	370
tomato	1,00	1,00	1,00	425
toothbrush	1,00	1,00	1,00	375
toothpaste	1,00	1,00	1,00	515
water_bottle	1,00	1,00	1,00	350
Average	1,00	1,00	1,00	21900
Weighted Avg	1,00	1,00	1,00	21900
Accuracy			1,00	21900

Table 10. Classification report for ResNet-50

Class	Precision	Recall	F1-score	Support
apple	1,00	0,93	0,97	380
ball	1,00	0,63	0,77	515
banana	1,00	0,96	0,98	460
bell_pepper	1,00	1,00	1,00	395
binder	1,00	0,80	0,89	445
bowl	1,00	0,79	0,89	365
calculator	0,97	1,00	0,99	365
camera	1,00	1,00	1,00	395
cap	1,00	1,00	1,00	395
cell_phone	0,95	1,00	0,97	345
cereal_box	1,00	0,93	0,96	360
coffee_mug	1,00	1,00	1,00	345
comb	1,00	0,93	0,96	355
dry_battery	1,00	1,00	1,00	355
flashlight	1,00	1,00	1,00	380
food_bag	1,00	1,00	1,00	495
food_box	1,00	0,99	0,99	495
food_can	0,95	1,00	0,98	515
food_cup	0,96	0,89	0,92	490
food_jar	1,00	1,00	1,00	485
garlic	1,00	0,89	0,94	490
glue_stick	0,99	1,00	1,00	505
greens	1,00	1,00	1,00	455
hand_towel	0,99	0,99	0,99	490
instant_noodles	1,00	1,00	1,00	485
keyboard	1,00	0,98	0,99	445
kleenex	0,99	1,00	0,99	490
lemon	0,96	0,96	0,96	370
lightbulb	0,91	1,00	0,95	390
lime	0,96	0,95	0,96	400
marker	1,00	0,86	0,93	505
mushroom	0,80	1,00	0,89	490
notebook	1,00	1,00	1,00	500
onion	1,00	0,46	0,63	485
orange	0,66	1,00	0,80	450
peach	1,00	0,72	0,84	425
pear	1,00	0,79	0,88	425
pitcher	1,00	0,94	0,97	355
plate	1,00	1,00	1,00	450
pliers	1,00	1,00	1,00	370
potato	1,00	0,58	0,73	380
rubber_eraser	1,00	1,00	1,00	495
scissors	0,90	1,00	0,95	390
shampoo	1,00	0,98	0,99	495
soda_can	0,83	1,00	0,91	395
sponge	0,81	1,00	0,89	400
stapler	0,97	1,00	0,99	370
tomato	0,44	1,00	0,61	425
toothbrush	0,85	1,00	0,92	375
toothpaste	1,00	0,99	1,00	515
water_bottle	1,00	1,00	1,00	350
Average	0,96	0,94	0,94	21900
Weighted Avg	0,96	0,94	0,94	21900
Accuracy			0,94	21900

Table 11. Classification report for HMM with 9 states and 512 symbols

Class	Precision	Recall	F1-score	Support
apple	0,33	0,99	0,50	380
ball	0,99	0,95	0,97	515
banana	0,98	0,91	0,95	460
bell_pepper	0,99	0,97	0,98	395
binder	0,99	0,97	0,98	445
bowl	1,00	0,99	1,00	365
calculator	0,85	0,81	0,83	365
camera	0,87	0,89	0,88	395
cap	0,82	0,91	0,86	395
cell_phone	0,96	0,94	0,95	345
cereal_box	0,95	0,88	0,91	360
coffee_mug	0,99	0,90	0,94	345
comb	0,94	0,86	0,90	355
dry_battery	0,99	0,86	0,92	355
flashlight	0,88	0,82	0,85	380
food_bag	0,97	0,94	0,96	495
food_box	1,00	0,93	0,96	495
food_can	1,00	0,97	0,98	515
food_cup	0,99	0,98	0,99	490
food_jar	1,00	0,96	0,98	485
garlic	0,96	0,98	0,97	490
glue_stick	1,00	0,98	0,99	505
greens	0,99	0,92	0,96	455
hand_towel	0,99	0,94	0,96	490
instant_noodles	0,95	0,94	0,94	485
keyboard	0,99	0,84	0,91	445
kleenex	1,00	0,92	0,96	490
lemon	1,00	0,98	0,99	370
lightbulb	0,98	0,91	0,94	390
lime	1,00	0,99	1,00	400
marker	0,95	0,89	0,92	505
mushroom	0,99	0,92	0,95	490
notebook	0,98	0,93	0,95	500
onion	0,99	1,00	0,99	485
orange	1,00	0,99	1,00	450
peach	1,00	0,98	0,99	425
pear	1,00	0,99	1,00	425
pitcher	0,97	0,94	0,96	355
plate	1,00	1,00	1,00	450
pliers	0,96	0,85	0,90	370
potato	1,00	0,98	0,99	380
rubber_eraser	0,96	0,96	0,96	495
scissors	0,90	0,83	0,86	390
shampoo	0,99	0,96	0,98	495
soda_can	1,00	0,95	0,98	395
sponge	1,00	0,99	1,00	400
stapler	0,85	0,84	0,84	370
tomato	1,00	0,98	0,99	425
toothbrush	0,95	0,93	0,94	375
toothpaste	0,92	0,89	0,90	515
water_bottle	0,98	0,99	0,99	350
Average	0,96	0,93	0,94	21900
Weighted Avg	0,96	0,94	0,94	21900
Accuracy			0,94	21900

Table 12. Classification report for HMMs Ensemble 1

Class	Precision	Recall	F1-score	Support
apple	0,27	1,00	0,43	380
ball	1,00	0,97	0,98	515
banana	1,00	0,92	0,96	460
bell_pepper	1,00	0,98	0,99	395
binder	1,00	0,98	0,99	445
bowl	1,00	0,99	1,00	365
calculator	1,00	0,80	0,89	365
camera	0,97	0,90	0,93	395
cap	0,87	0,93	0,90	395
cell_phone	1,00	0,93	0,96	345
cereal_box	0,99	0,91	0,95	360
coffee_mug	1,00	0,91	0,95	345
comb	0,99	0,86	0,92	355
dry_battery	1,00	0,86	0,93	355
flashlight	0,97	0,84	0,90	380
food_bag	1,00	0,96	0,98	495
food_box	1,00	0,94	0,97	495
food_can	1,00	0,97	0,99	515
food_cup	1,00	0,99	0,99	490
food_jar	1,00	0,96	0,98	485
garlic	1,00	0,97	0,99	490
glue_stick	1,00	0,99	1,00	505
greens	1,00	0,92	0,96	455
hand_towel	1,00	0,97	0,98	490
instant_noodles	1,00	0,93	0,96	485
keyboard	1,00	0,88	0,94	445
kleenex	1,00	0,96	0,98	490
lemon	1,00	0,99	1,00	370
lightbulb	0,99	0,95	0,97	390
lime	1,00	0,99	1,00	400
marker	1,00	0,92	0,96	505
mushroom	1,00	0,97	0,98	490
notebook	0,99	0,93	0,96	500
onion	1,00	0,99	1,00	485
orange	1,00	0,99	1,00	450
peach	1,00	0,98	0,99	425
pear	1,00	0,99	0,99	425
pitcher	1,00	0,99	0,99	355
plate	1,00	1,00	1,00	450
pliers	1,00	0,84	0,91	370
potato	1,00	0,98	0,99	380
rubber_eraser	1,00	0,96	0,98	495
scissors	0,98	0,83	0,90	390
shampoo	1,00	0,97	0,98	495
soda_can	1,00	0,96	0,98	395
sponge	1,00	1,00	1,00	400
stapler	0,94	0,90	0,92	370
tomato	1,00	0,99	1,00	425
toothbrush	0,98	0,94	0,96	375
toothpaste	0,99	0,93	0,96	515
water_bottle	1,00	0,99	0,99	350
Average	0,98	0,95	0,96	21900
Weighted Avg	0,98	0,95	0,96	21900
Accuracy			0,95	21900

Table 13. Classification report for HMMs Ensemble 2

Class	Precision	Recall	F1-score	Support
apple	0,97	1,00	0,98	380
ball	1,00	1,00	1,00	515
banana	0,99	1,00	0,99	460
bell_pepper	1,00	1,00	1,00	395
binder	1,00	1,00	1,00	445
bowl	1,00	1,00	1,00	365
calculator	0,98	0,98	0,98	365
camera	0,88	0,93	0,90	395
cap	0,73	0,86	0,79	395
cell_phone	0,97	1,00	0,98	345
cereal_box	0,97	0,97	0,97	360
coffee_mug	1,00	0,97	0,98	345
comb	0,99	0,96	0,97	355
dry_battery	1,00	0,95	0,97	355
flashlight	0,94	0,88	0,91	380
food_bag	0,99	1,00	0,99	495
food_box	1,00	0,99	1,00	495
food_can	1,00	1,00	1,00	515
food_cup	0,99	1,00	1,00	490
food_jar	1,00	1,00	1,00	485
garlic	0,91	1,00	0,95	490
glue_stick	1,00	1,00	1,00	505
greens	1,00	1,00	1,00	455
hand_towel	1,00	1,00	1,00	490
instant_noodles	0,99	1,00	1,00	485
keyboard	1,00	0,75	0,85	445
kleenex	1,00	0,98	0,99	490
lemon	1,00	1,00	1,00	370
lightbulb	0,90	0,95	0,92	390
lime	0,99	1,00	0,99	400
marker	0,99	0,99	0,99	505
mushroom	0,99	0,91	0,95	490
notebook	1,00	1,00	1,00	500
onion	1,00	1,00	1,00	485
orange	1,00	1,00	1,00	450
peach	1,00	1,00	1,00	425
pear	0,98	1,00	0,99	425
pitcher	0,99	0,98	0,99	355
plate	1,00	1,00	1,00	450
pliers	0,99	0,98	0,99	370
potato	1,00	0,99	0,99	380
rubber_eraser	0,98	1,00	0,99	495
scissors	0,97	0,98	0,98	390
shampoo	1,00	1,00	1,00	495
soda_can	1,00	1,00	1,00	395
sponge	1,00	1,00	1,00	400
stapler	0,88	0,88	0,88	370
tomato	1,00	1,00	1,00	425
toothbrush	0,94	0,98	0,96	375
toothpaste	0,98	0,98	0,98	515
water_bottle	0,99	1,00	1,00	350
Average	0,98	0,98	0,98	21900
Weighted Avg	0,98	0,98	0,98	21900
Accuracy			0,98	21900

References

1. Dhall D., Kaur R., Juneja M. Machine Learning: A Review of the Algorithms and Its Applications. Proceedings of International Conference on Recent Inno-vations in Computing (ICRIC'2019). 2019. pp. 47–63.
2. Dhruv P., Naskar S. Image Classification Using Convolutional Neural Net-work (CNN) and Recurrent Neural Network (RNN): A Review. International Conference on Machine Learning and Information Processing (ICMLIP'2019). 2020. pp. 367–381,
3. Wang W et al. Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*. 2019. vol. 58. no. 4. pp. 1–19.
4. Gambella C., Ghaddar B., Naoum-Sawaya J. Optimization problems for ma-chine learning: A survey. *European Journal of Operational Research.* Available at: https://doi.org/10.1016/j.ejor.2020.08.045 (accessed: 20.09.2020).
5. 5. Dogo E.M. et al. A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks. International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS'2018). 2018. pp. 92–99.
6. Belanche D., Casal ´o L.V., Flavi´an C., Schepers J. Service robot implementation: a theoretical framework and research agenda. *The Service Industries Journal.* 2020. vol. 40 no. 3-4. pp. 203–225.
7. Torras C. Service Robots for Citizens of the Future. *European Review*. 2016. vol. 24. no. 1. pp. 17–30.
8. Matamoros M. et al. Robocup at home 2019: Rules and regulations. Available at: http://www.robocupathome.org/rules/2019rulebook.pdf (accessed: 20.09.2020).
9. Matamoros M., Rascon C., Wachsmuth S., Moriarty A.W., Kummert J., Hart J, Pfeiffer S., van der Brugh M., St-Pierre M. Robocup at home 2019: Rules and regulations. Available at: http://www.robocupathome.org/rules/2019_rulebook.pdf. (accessed 20.09.2020).
10. Baum L.E., Petrie T., Soules G., Weiss N. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*. 1970. vol. 41. no.1. pp. 164–171.
11. Rabiner L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Readings in Speech Recognition. Morgan Kaufmann Publishers Inc. 1990. pp. 267–296.
12. Mor B., Garhwal S., Kumar A. A Systematic Review of Hidden Markov Models and Their Applications. Archives of Computational Methods in Engineering. Available at: https://doi.org/10.1007/s11831-020-09422-4. (accessed: 20.09.2020).
13. 13. Corcoran P., Iancu C. Hidden Markov Models in Automatic Face Recognition – A Review. *Reviews, Refinements and New Ideas in Face Recognition.* Available at: https://doi.org/10.5772/17664 (accessed: 20.09.2020).
14. Rastghalam R., Pourghassem H. Breast cancer detection using MRF-based probable texture feature and decision-level fusion-based classification using HMM on thermography images. *Pattern Recognition*. 2016. vol. 51. pp. 176–186.
15. Hassan M. et al. Robust Hidden Markov Model based intelligent blood vessel detection of fundus images. *Computer Methods and Programs in Biomedicine.* 2017. vol. 151. pp. 193–201.
16. Sarmiento C. et al. Feature detection using Hidden Markov Models for 3D-visual recognition. IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC'2019). 2019. pp. 1–6.
17. Russakovsky O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*. 2015. vol. 115. no. 3. pp. 211–252.

18. Krizhevsky A., Sutskever I., Hinton G. Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems (NIPS'2012). 2012. pp. 1097–1105.
19. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. International Conference on Learning Representations (ILCR'2015). 2015
20. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2016). 2016. pp. 770–778.
21. Howard A.G. et al. MobileNets:Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*. Available at: http://arxiv.org/abs/1704.04861 (accessed: 20.09.2020).
22. Sculley D. Web-scale k-means clustering. Proceedings of the 19th International Conference on World Wide Web (WWW '10). 2010. pp. 1177–1178.
23. Lai K., Bo L., Ren X., Fox D. A large-scale hierarchical multi-view RGB-D object dataset. IEEE International Conference on Robotics and Automation (ICRA'2011). 2011. pp. 1817–1824.
24. Sharma G., Wu W., Dalal E.N. The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations. *Color Research and Application*. 2005. vol. 30. no. 1. pp. 21–30.
25. Pedregosa F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research.* 2011. vol. 12. no. 85. pp. 2825–2830.
26. Arthur D., Vassilvitskii S. K-means++: The advantages of careful seeding. ACM-SIAM Symposium on Discrete Algorithms (SODA'07). 2007. pp. 1027–1035.
27. Elkan C. Using the triangle inequality to accelerate k-means. Proceedings of the 20th International Conference on Machine Learning (ICML'03). 2003. pp 147–153.
28. Marina S.M., Lapalme G. A systematic analysis of performance measures for classification tasks. *Information Processing Management.* 2009. vol. 45. no. 4 pp. 427–437.
29. Abadi M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems. Available at: www.tensorflow.org (accessed: 20.09.2020).

Sarmiento Carlos — PhD student, Laboratory of Biorobotics, National Autonomous University of Mexico (UNAM). Research interests: computer vision, reinforcement learning, autonomous navigation. The number of publications -– 2. ing.adriansarmiento@comunidad.unam.mx; Circuito Exterior S/N, Ciudad Universitar-ia, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Savage Jesus — Ph.D., Professor, Professor, Electrical Engineering Department, National Autonomous University of Mexico (UNAM). Research interests: rch interests: autonomous mobile robots, digital signal processing, computer architectures. The number of publications — 100. robotssavage@gmail.com; Circuito Exterior S/N, Ciudad Universitaria, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Acknowledgements. This research is supported by UNAM-CONACYT.

УДК 004.9 **DOI** 10.15622/ia.2020.19.6.4

К. Сармьенто , Х. Саваж

СРАВНЕНИЕ ДВУХ МЕТОДОВ КЛАССИФИКАЦИИ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ СКРЫТЫХ МАРКОВСКИХ МОДЕЛЕЙ И СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Сармьенто К., Саваж Х. **Сравнение двух методов классификации объектов с использованием скрытых марковских моделей и сверточных нейронных сетей.**

Аннотация. Представлено сравнение дискретных скрытых марковских моделей и свёрточных нейронных сетей для классификации изображений. После разбивки изображений на части целесообразно получить векторы, которые представляют локальные визуальные структуры, одновременно определяющие изображения глобально через пространственную последовательность. С использованием методов кластеризации создается алфавит из указанных векторов, а затем конструируются последовательности символов, которые описывают статистические модели, соответствующие классам изображений. Скрытые марковские модели в сочетании с методами квантования могут обрабатывать шум и искажения в наблюдениях для решения проблем компьютерного зрения, таких как классификация изображений с изменением освещения и перспективы.

Протестированы архитектуры, основанные на трех, шести и девяти скрытых состояниях, в пользу скорости обнаружения и низкого использования памяти. Также были протестированы два типа ансамблевых моделей. Точность предлагаемого метода была оценена с помощью общедоступных данных; полученные результаты оказались сравнимы с известными оценками при использовании тонко настроенных свёрточных нейронных сетей, но требовали значительно меньших вычислительных ресурсов. Результат представляет интерес при разработке мобильных роботов с вычислительными устройствами, имеющими ограниченное время автономной работы, но требующими способности обнаруживать и добавлять новые объекты в свои системы классификации.

Ключевые слова: скрытые марковские модели, классификация изображений, компьютерное зрение, распознавание образов

Сармьенто Карлос — аспирант, лаборатория биороботики, Национальный автономный университет Мексики (НАУМ). Область научных интересов: компьютерное зрение, машинное обучение с подкреплением, автономная навигация. Число научных публикаций — 2. ing.adriansarmiento@comunidad.unam.mx; Циркуито Экстериор б/н, Университетский город, 04510, Мехико, Мексика; р.т.: +52(55)56223041.

Саваж Хесус — канд. техн. наук, профессор, профессор, электротехнический отдел, Национальный автономный университет Мексики (НАУМ). Область научных интересов: интересов: автономные мобильные роботы, цифровая обработка сигналов, архитектуры компьютеров. Число научных публикаций — 100. robotssavage@gmail.com; Циркуито Экстериор б/н, Университетский город, 04510, Мехико, Мексика; р.т.: +52(55)56223041.

Поддержка исследований. Данное исследование поддержано UNAM-CONACYT.

Литература

1. *Dhall D., Kaur R., Juneja M.* Machine Learning: A Review of the Algorithms and Its Applications // Proceedings of International Conference on Recent Innovations in Computing (ICRIC'2019). 2019. pp. 47–63.

2. *Dhruv P., Naskar S.* Image Classification Using Convolutional Neural Net-work (CNN) and Recurrent Neural Network (RNN): A Review // International Conference on Machine Learning and Information Processing (ICMLIP'2019). 2020. pp. 367–381.
3. *Wang W et al.* Development of convolutional neural network and its application in image classification: a survey // Optical Engineering. 2019. vol. 58. no. 4. pp. 1–19..
4. *Gambella C., Ghaddar B., Naoum-Sawaya J.* Optimization problems for ma-chine learning: A survey // European Journal of Operational Research. Availa-ble at: https://doi.org/10.1016/j.ejor.2020.08.045 (accessed: 20.09.2020).
5. *Dogo E.M. et al.* A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks // International Confer-ence on Computational Techniques, Electronics and Mechanical Systems (CTEMS'2018). 2018. pp. 92–99.
6. *Belanche D., Casal´o L.V., Flavi´an C., Schepers J.* Service robot implementation: a theoretical framework and research agenda // The Service Industries Journal. 2020. vol. 40 no. 3-4. pp. 203–225.
7. *Torras C.* Service Robots for Citizens of the Future // European Review. 2016. vol. 24. no. 1. pp. 17–30.
8. *Zachiotis G.A. et al.* A Survey on the Application Trends of Home Service Robotics // IEEE International Conference on Robotics and Biomimetics (ROBIO'2018). 2018. pp. 1999–2006
9. *Matamoros M. et al.* Robocup at home 2019: Rules and regulations. Available at: http://www.robocupathome.org/rules/2019rulebook.pdf (accessed: 20.09.2020).
10. *Baum L.E., Petrie T., Soules G., Weiss N.* A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains // The Annals of Mathematical Statistics. 1970. vol. 41. no.1. pp. 164–171.
11. *Rabiner L.R.* A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Readings in Speech Recognition // Morgan Kaufmann Publishers Inc. 1990. pp. 267–296.
12. *Mor B., Garhwal S., Kumar A.* A Systematic Review of Hidden Markov Models and Their Applications. Archives of Computational Methods in Engineering. Available at: https://doi.org/10.1007/s11831-020-09422-4. (accessed: 20.09.2020).
13. *Corcoran P., Iancu C.* Hidden Markov Models in Automatic Face Recognition – A Review. Reviews, Refinements and New Ideas in Face Recognition. Avail-able at: https://doi.org/10.5772/17664 (accessed: 20.09.2020).
14. *Rastghalam R., Pourghassem H.* Breast cancer detection using MRF-based probable texture feature and decision-level fusion-based classification using HMM on thermography images // Pattern Recognition. 2016. vol. 51. pp. 176–186.
15. *Hassan M. et al.* Robust Hidden Markov Model based intelligent blood vessel detection of fundus images // Computer Methods and Programs in Biomedicine. 2017. vol. 151. pp. 193–201.
16. *Sarmiento C. et al.* Feature detection using Hidden Markov Models for 3D-visual recognition // IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC'2019). 2019. pp. 1–6.
17. *Russakovsky O. et al.* ImageNet Large Scale Visual Recognition Challenge // International Journal of Computer Vision. 2015. vol. 115. no. 3. pp. 211–252.
18. *Krizhevsky A., Sutskever I., Hinton G.* Imagenet classification with deep convolutional neural networks // Neural Information Processing Systems (NIPS'2012). 2012. pp. 1097–1105.
19. *Simonyan K., Zisserman A.* Very Deep Convolutional Networks for Large-Scale Image Recognition // International Conference on Learning Representations (ILCR'2015). 2015.

20. *He K., Zhang X., Ren S., Sun J.* Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2016). 2016. pp. 770–778.
21. *Howard A.G. et al.* MobileNets:Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR. Available at: http://arxiv.org/abs/1704.04861 (accessed: 20.09.2020).
22. *Sculley D.* Web-scale k-means clustering. Proceedings of the 19th International Conference on World Wide Web (WWW '10). 2010. pp. 1177–1178.
23. *Lai K., Bo L., Ren X., Fox D.* A large-scale hierarchical multi-view RGB-D object dataset. IEEE International Conference on Robotics and Automation (ICRA'2011). 2011. pp. 1817–1824.
24. *Sharma G., Wu W., Dalal E.N.* The CIEDE2000 color-difference formula: im-plementation notes, supplementary test data, and mathematical observations. Color Research and Application. 2005. vol. 30. no. 1. pp. 21–30.
25. *Pedregosa F. et al.* Scikit-learn: Machine learning in python. Journal ´ of Ma-chine Learning Research. 2011. vol. 12. no. 85. pp. 2825–2830.
26. *Arthur D., Vassilvitskii S.* K-means++: The advantages of careful seeding. ACM-SIAM Symposium on Discrete Algorithms (SODA'07). 2007. pp. 1027–1035.
27. *Elkan C.* Using the triangle inequality to accelerate k-means. Proceedings of the 20th International Conference on Machine Learning (ICML'03). 2003. pp 147–153.
28. *Marina S.M., Lapalme G.* A systematic analysis of performance measures for classification tasks. Information Processing Management. 2009. vol. 45. no. 4 pp. 427–437
29. *Abadi M. et al.* TensorFlow: Large-scale machine learning on heterogeneous systems. Available at: www.tensorflow.org (accessed: 20.09.2020).