Convolutional Neural Networks and Pattern Recognition: Application to Image Classification

Christy Ntambwe Kabamba[1], Lucie Mpuekela .N[2], Simon Ntumba .B[3], Eugene Mbuyi .M[4]

[1] Mathématiques et Informatique, Université de Kinshasa, Kinshasa, DRC

[2]Informatique, Institut Supérieur Pédagogique de Mbujimayi, Mbujimayi, DRC.

[3]Mathématiques et Informatique, Université de Kinshasa, Kinshasa, DRC

[4]Mathématiques et Informatique, Université de Kinshasa, Kinshasa, DRC

Abstract

This research study focuses on pattern recognition using convolutional neural network. Deep neural network has been choosing as the best option for the training process because it produced a high percentage of accuracy. We designed different architectures of convolutional neural network in order to find the one with high accuracy of image classification and optimum bias. We used CIFAR-10 data set that contains 60000 Images to train our model on architectures. The best architecture was able to classify images with 95.55% of accuracy and an error of 0.32% using cross validation method. We note that, the numbers of epoch while running the model and the depth of the architecture are factors that contributed to get this performance.

Keywords: *Convolutional Neural Networks, Cross validation, Deep Learning, Overfitting, Deep neural network*

1. Introduction

A domain both ancient in its history and very young in its multiple evolutions over the last decades, pattern recognition has been perceived as a full-fledged branch of Artificial Intelligence, producing algorithms for environment perception. Quite quickly, it extended its field of action to machine learning frameworks. The aim of this study is to design a model that predicts a class to which belongs input image. Hence the need to set up an optimal model that minimizing error and maximizing accuracy.

Thus, it's a problem of supervised learning, that is to say when we observe certain input examples and the corresponding classes in order to approach or to correspond these inputs to expected outputs by adjusting parameters. The model that have been designed, learn from the observations of The DataSet and is then generalized on data besides sample on which the parameters will be trained.

To do this, we use Deep Learning, "a new research area of Machine learning whose aim is to bring Machine Learning closer to its main objective which is artificial intelligence"[7]. It is based on artificial neural networks idea and is tailored to handle large amounts of data by adding layers to the network.

A particular type of artificial neural networks is used, namely convolutional neural networks. Different architectures of this type of neural networks are tested in order to determine the one with best accuracy for our model. CIFAR-10 has been choosing to test our model on theses architectures.

Several works in this area focus more on empirical computer applications. However, from the perspective of this, we are interested in exploring the developments of the most recent computer vision used for neural networks in order to improve the results of previous work, such as in [D. Djabeur and Mohammed, 2017] and many others, having tackled the same problem with neural networks.

In the aforementioned paper, the MLP model classified images with an error estimated at 2.31% and 61% of accuracy for its experiments, while CNN's model achieved 78% of accuracy and an error estimated at 0.76%. We note that, for this study, the better architecture contained 4 layers of convolution, 2 layers of max pooling, the function being ReLU and two full connected layers. To train and validate their model, they used the MNST data set divided into two parts, one for the training of the model containing 50,000 images of handwritten figures and another 10,000 for model validation. This is why there was a negative impact on the performance of their model.

In order to improve the performance of our model, we define an architecture containing six convolution layers, three layers of Max pooling and three full connected layers. We train the model using the cross validation technique. It consists of divide the training set into k folds. Our training set has been divided into 5 folds each containing 10,000 images. Besides this

technique, the use of regularization methods (Dropout) is crucial to significantly reduce Overfitting.

2. Pattern recognition

2.1 Definition

Pattern recognition is a discipline straddling mathematics and artificial intelligence to which several definitions have been assigned. From all these definitions which so perfectly complement each other, it emerges that it is an activity which consists in designing automatic or semi-automatic systems which recognize the shapes or patterns presented to them [2]. The problem that it tries to solve is to associate a class to an unknown pattern. This is why it is often considered as a classification problem: *finding the function which affects its most relevant class to any unknown pattern* [6]. It takes place according to a protocol called "*Pattern recognition process*", (shown diagrammatically in Fig 1).

2.2. The pattern recognition process

This process allows to reduce the amount of data to handle, leaving the original information that is part of the observation or real world space (often an image or signal) to reach its symbolic description, in the space of interpretation or space of categories, passing through the space of representation or space of characteristics where the relevant primitives are extracted [5].
From this description, we deduce that in the general case, there are two main stages which are:

- The characteristic extraction stage: the passage from the observation space to the representation space.
- The classification stage: the transition from the representation space to the interpretation space.

Other steps are complementary and sometimes necessary such as acquisition, pre-processing and post-processing.

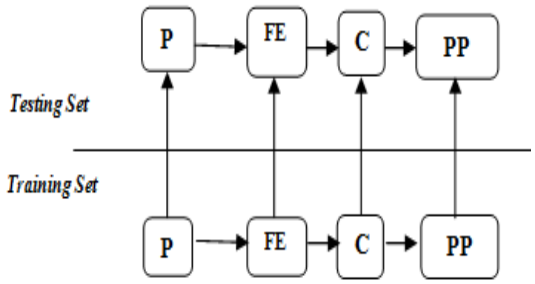

Fig 1 Pattern recognition process

Legends: *(**P**): Preprocessing, (**FE**): Features Extraction, (C): Classification, (**PP**): Post processing*

2.3 Pattern recognition methods

The pattern recognition is firstly a problem of automatic learning of the model, can be carried out by means of various algorithms of automatic learning such as: Statistical analysis, Hidden markov models, the decision tree, neural networks, support vector machines, Bayesian networks, etc. There are two families of methods of machine learning: Unsupervised Learning and Supervised learning [12].

⇨*Unsupervised learning*

In unsupervised learning context, the task consists to discover similarities between the observations in a collection of samples, in the aim of grouping them into subsets, called classes [6, 13]. These algorithms allow making it possible to bring together the most similar samples and remove those which have the least common characteristics. Hierarchical classification and K-Means (Search for nearest neighbors) are potential examples.

⇨*Supervised learning*

In this context, the observations are accompanied by additional information relating to whether or not to the concept.
The object of a supervised learning algorithm is to correctly classify the new samples in the classes defined in the learning phase. Supervised learning is generally carried out in two phases:
Learning phase: this phase is devoted to learning the basics rules of a so-called learning sample, defined at the start and whose classification is known.
Test phase: In this second phase, a second independent sample, called validation or test, is formed in order to study the reliability of the rules to compare them, apply them and assess the complexity of the model by verifying the sub cases learning or over-learning.
Thus, neural networks in general and in particular convolutional neural networks, which we address in the next point, are among the most used supervised learning algorithms today.

3. Convolutional Neural Networks (CNN)

3.1 Presentation

CNNs were specifically designed for *images classification* to address the problems of scaling up MLPs when they were to be used for classification purposes for a large data set, particularly images. Their basic architecture, where the entry of each neuron into a layer is connected to all the neural outputs of the previous layer, makes it impossible to learn parameters. CNNs are made up of two very distinct parts. The first, so called convolutional strictly speaking, functions as an extractor of image characteristics [11]. An image is passed through a succession of filters, or convolution kernels, creating new images called convolution maps. In the end, the

convolution maps are flattened and concatenated into a vector of characteristics, called the CNN code.

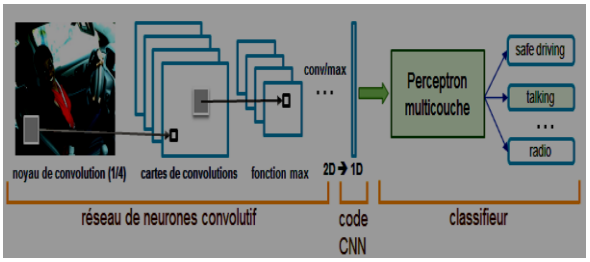

Fig 2: Convolutional neural network [5]

This CNN code at the end of the convolutional part is then connected to the input of a second part, consisting of layers of fully connected neurons. The role of this part is to combine the characteristics of the CNN code to classify the image. The output is a final layer with one neuron by category. CNNs are therefore based on MLP, and inspired by the behavior of the visual cortex of vertebrates. However, although effective for image processing, traditional MLP has great difficulty in handling large images, given the exponential growth in the number of connections with image size. From the point of view of pattern recognition, CNN induces local properties of translation invariance [15]. These properties are essential for recognition more generally images which can be seen from different angles. In most deep CNN architectures, the parameters are learned by the stochastic gradient descent algorithm (SGD) [4].

3.2 CNN Architecture

Architecture is formed by a stack of independent treatment layers:

- *The convolution layer* (CONV) which processes the data of a receiving field.
- *The pooling layer* (POOL) which makes it possible to compress information by reducing the size of the intermediate image and therefore reduces the quantity of parameters and calculation in the network.
- *The correction layer* (ReLU): Allows to improve the processing efficiency by inserting between the processing layers a layer which will operate a mathematical function on the output signals. The ReLU function $(F(x) = max(0, x))$therefore forces the neurons to return positive values.
- *The fully connected layer* (FC), which is a perception type layer.
- *The loss layer* (LOSS): it specifies how the training of the network penalizes the difference between the expected and actual signal. It is normally the last layer in the network [4]. Various loss functions suitable for different tasks can be used there. The "Softmax" function used in this search allows calculating the probability distribution on the output classes [10].

Usually, a convolutional layer is followed by an activation function and then a pooling layer; this sequence can be repeated several times before going to the fully connected layer to form a convolutional network which is called CONVNET. Here are some examples of CNN architecture [14]:

*INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC* Here, there is a single CONV layer before each POOL layer
*INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC* Here there are two CONV layers stacked before each POOL layer.

3.3 CNN Learning

To know the parameters of this network, that is to say the weights and the biases of each formal neuron which constitutes it, we used a training database providing input data and corresponding desired outputs. The calculation of these parameters is called the learning phase and is done using the gradient backpropagation algorithm introduced by [Rumelhart 1986, Lecun 1986]. Before learning, the network parameters are unknown and must be initialized to random values [1].
We consider a subset of our learning base composed of N examples: $\{(X^t, r^t)\}^N$
With $X^t = [x_1^t, \dots, x_n^t]^T$, an input data and $r^t = [r_1^t, \dots, r_k^t]^T$ the desired output corresponding to it.
Thus,(x^t, r^t) is the example of the batch (subset of the learning base).The parameters of a formal neuron$j$will be denoted respectivelyw_{ji} and b_j for its weight and bias. By passing the data x^t in the MLP, the error observed on the neuron j of the output layer is written [16]:

$$e_j^t = \mathcal{L}(r_j^t, y_j^t) \quad (1)$$

Wherey_j^t is the value returned by the output neuron$j$. $\mathcal{L}$ is called the loss function and represents what the network is trying to minimize on all of the data. This function is different depending on the problem to be solved. In the case of MLP which is also applicable to CNN, the loss function is defined as follows:

$$\mathcal{L}(r_j^t, y_j^t) = r_j^t - y_j^t \quad (2)$$

The quadratic error observed for the datax^t on the Kneurons of the output layer is written:

$$E^t = \frac{1}{2}\sum_{j=1}^{K}(e_j^t)^2 \quad (3)$$

And the mean square error over the entire *batch* is defined by:

$$E = \frac{1}{N}\sum_{t=1}^{N} E^t \quad (4)$$

The network weights are updated by stochastic gradient descent (SGD) from the mean square error.

In other words, the weight $w_{i,j}$ of the neuron j is updated by adding the $-\alpha\Delta w_{i,j}$ term to it.

The update of the bias b_j of j neuron is done by adding the $-\alpha\Delta b_j$ term to it. α is called learning rate [3, 9] . It is used to weight the update of network parameters. The two terms $\Delta w_{i,j}$ and Δb_j are defined as follows:

$$\Delta w_{j,i} = \frac{\partial E}{\partial w_{j,i}} = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial w_{j,i}} \quad and$$

$$\Delta b_j = \frac{\partial E}{\partial b_j} = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial E^t}{\partial b_j} \quad (5)$$

In what follows, we detail the back propagation of the gradient according to the type of layer considered (output layer or hidden layer).

3.3.1. Back propagation for the output layer

Using equation (5) and thanks to the chain decomposition of partial derivatives, we can write:

$$\frac{\partial E^t}{\partial w_{j,i}} = \frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial \alpha_j^t}\frac{\partial \alpha_j^t}{\partial w_{j,i}} \quad et \quad \frac{\partial E^t}{\partial b_j} = \frac{\partial E^t}{\partial e_j^t}\frac{\partial e_j^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial \alpha_j^t}\frac{\partial \alpha_j^t}{\partial b_j} \quad (6)$$

These partial derivatives can individually be expressed as follows:

$$\frac{\partial E}{\partial e_j^t} = \frac{\partial}{\partial e_j^t}\frac{1}{2}\sum_{j=1}^{K}(e_j^t)^2 = e_j^t \quad (7)$$

$$\frac{\partial e_j^t}{\partial y_j^t} = \frac{\partial}{\partial y_j^t}\left(r_j^t - y_j^t\right) = -1 \quad (8)$$

$$\frac{\partial y_j^t}{\partial \alpha_j^t} = \frac{\partial}{\partial \alpha_j^t}\frac{1}{1+e^{-\alpha_j^t}} = \frac{e^{-\alpha_j^t}}{(1+e^{-\alpha_j^t})^2} = y_j^t\,(1-y_j^t) \quad (9)$$

$$\frac{\partial \alpha_j^t}{\partial w_{j,i}} = \frac{\partial}{\partial w_{j,i}}\sum_{i=1}^{R} w_{j,i}y_i^t + b_j = y_i^t \quad (10)$$

$$\frac{\partial \alpha_j^t}{\partial b_j} = \frac{\partial}{\partial b_j}\sum_{i=1}^{R} w_{j,i}y_i^t + b_j = 1 \quad (11)$$

Which allows to obtain:

$$-\alpha\Delta w_{j,i} = \frac{\alpha}{N}\sum_{t=1}^{N}\delta_j^t y_i^t \quad and$$

$$-\alpha\Delta b_j = \frac{\alpha}{N}\sum_{t=1}^{N}\delta_j^t \quad (12)$$

Let's say:

$$\delta_j^t = e_j^t y_j^t\left(1-y_j^t\right) \quad (13)$$

3.3.2. Back propagation for hidden layers

In the case of the hidden layer preceding the output layer, the error e_j^t of the hidden neuron j is unknown. For this type of layer, the partial derivative of the quadratic error of equation (5) is written as follows:

$$\frac{\partial E^t}{\partial w_{j,i}} = \frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial \alpha_j^t}\frac{\partial \alpha_j^t}{\partial w_{j,i}} \quad and$$

$$\frac{\partial E^t}{\partial b_j} = \frac{\partial E^t}{\partial y_j^t}\frac{\partial y_j^t}{\partial \alpha_j^t}\frac{\partial \alpha_j^t}{\partial b_j} \quad (14)$$

The partial derivative $\frac{\partial E^t}{\partial y_j^t}$ can be expressed as:

$$\frac{\partial E^t}{\partial y_j^t} = \frac{\partial}{\partial y_j^t}\frac{1}{2}\sum_k (e_k^t)^2 = \sum_k e_k^t\frac{\partial e_k^t}{\partial y_j^t} = \sum_k e_k^t\frac{\partial e_k^t}{\partial \alpha_k^t}\frac{\partial \alpha_k^t}{\partial y_j^t}$$

$$= \sum_k e\frac{\partial(r_k^t - y_k^t)}{\partial \alpha_k^t}\frac{\partial(\sum_l w_{k,l}y_l^t + b_k)}{\partial y_j^t}$$

$$= \sum_k e_k^t\left(-y_k^t(1-y_k^t)\right)w_{k,j} \quad (15)$$

So we get:

$$-\alpha\Delta w_{j,i} = \frac{\alpha}{N}\sum_{t=1}^{N}\delta_j^t y_i^t \quad and$$

$$-\alpha\Delta b = \frac{\alpha}{N}\sum_{t=1}^{N}\delta_j^t \quad (16)$$

With:

$$\delta_j^t = y_j^t\left(1-y_j^t\right)\sum_k \delta_k w_{kj} \quad (17)$$

The term δ_k is the gradient defined in (13). Equation (17) generalizes to all hidden layers. The gradient δ_j^t of a neuron j in a hidden layer uses the gradient of the layer that follows it.

3.3.3. Generalized delta rule

The network parameters are updated using the generalized delta rule. So, we did it by stochastic gradient descent (SGD) with *momentum*.

For iteration τ , allowing to pass in the network a *batch* of data, the update is done as follows:
For the weights :

$$w_{j,i}(\tau) = w_{j,i}(\tau - 1) - \alpha \Delta w_{j,i}(\tau) + \beta \Delta w_{j,i}(\tau - 1) \quad (18)$$

For bias:

$$b_j(\tau) = b_j(\tau - 1) - \alpha \Delta b_j + \beta \Delta b_j(\tau - 1) \quad (19)$$

With β called *momentum* (value between 0 and 1) allowing to give inertia to the gradient descent by taking into account the corrections applied to the previous iteration $\tau - 1$.
Thus, there are other rules than SGD with *momentum* for the optimization of neural networks: ADADELTA [Zeiler 2012], ADAGRAD [Duchi 2011], ADAM [Kingma 2015], RMSProp [Tieleman 2012]. These optimization rules make it possible to adapt the learning rate to the parameters in order to better converge while being faster [9].

3.4. Complexity control

3.4.1. Regularization

In elementary networks, a simple option to avoid over-learning consists in introducing a term of penalization or regularization, as in regression ridge, in the criterion to be optimized. Higher the Parameter values is important, less the weights of the inputs of the neurons can take chaotic values contributing to limit the risks of over-learning [16].

3.4.2. Choice of parameters

The user must therefore determine:

⇨The input variables and the output variable; make them undergo, as for all statistical methods, possible transformations, normalizations.

⇨Network architecture: the number of hidden layers which corresponds to an ability to deal with non-linearity problems, the number of neurons per hidden layer. These two choices directly condition the number of parameters (of weight) to be estimated and therefore the complexity of the model. They participate in the search for a good compromise / bias / variance, that is to say the balance between quality of learning and quality of forecasting.

⇨Three other parameters are also involved in this compromise: the maximum number of iterations, the maximum tolerated error and a possible term of regularization (dropout).

⇨The learning rate and a possible evolution strategy for it.

⇨The size of the sets or batches of observations considered at each iteration.

In practice, all these parameters cannot be adjusted simultaneously by the user [8]. This one is confronted with choices mainly concerning the control of over-learning: limiting the number of neurons or the learning time or even increasing the penalization coefficient of the norm of the parameters.
This requires determining a method of estimating the error: sample validation or test, cross validation or bootstrap. A simple and undoubtedly effective strategy consists in introducing a rather large number of neurons then in optimizing the only regularization parameter (decay) by cross validation. CNNs use more parameters than a standard MLP. Even if the usual rules for learning rates and regularization constants still apply, the notions of number of filters, their form and the form of max pooling must be taken into consideration.

4. Application

4.1. Selected CNN architecture

The CNN architecture that we present is constituted with six convolution layers, three layers of Max pooling and three layers fully connected. The input image has size 32 * 32 * 3; the image passes firstly to the first convolution layer. This layer is composed by 32 filters of3 * 3 size, the activation function ReLU is used, it forces the neurons to return positive values, after this convolution, 32 feature maps of 32 * 32 * 3size will be created. Then, the 32 feature maps obtained are given as input to the second convolution layer which is also composed of 32 filters. The ReLU is applied to the convolution layers. Maxpooling is applied afterwards to reduce the image size and parameters. At the exit of this layer, we will have 32 features size of 16 * 16 maps. We repeat the same thing with convolution layers three and four which are composed of 64 filters, the activation function ReLU is always applied on each convolution. A layer of Maxpooling is applied after the four convolution layer. At the end of this layer, we will have 64 feature maps of8 * 8 size. The vector dimension of characteristics, resulting from convolutions is 4096.
After this convolutional part, we use a neural network composed of three fully connected layers. The first two layers each have 1500 neurons where the activation function used is the ReLU, and the third layer is a softmax which makes it possible to calculate the probability distribution of the 10 classes (number of classes in the CIFAR10 image base).

4.2. Experimental results

In order to evaluate the performance of our model, we present the curves showing the evolution of the error and the precision of the model on the batches used as well as the confusion matrix.
We point in the table, summaries results obtained with the MLP to compare to those of CNN.

4.2.1. Learning outputs

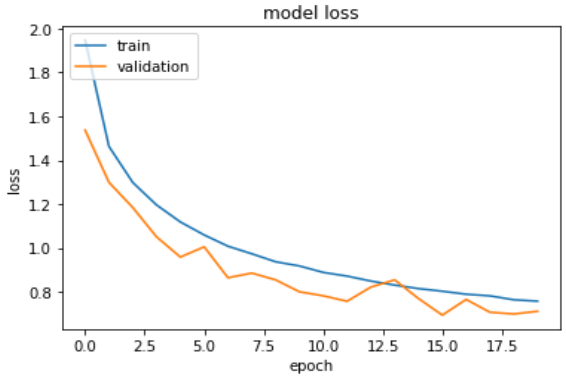

Fig 3 Curves of the model error

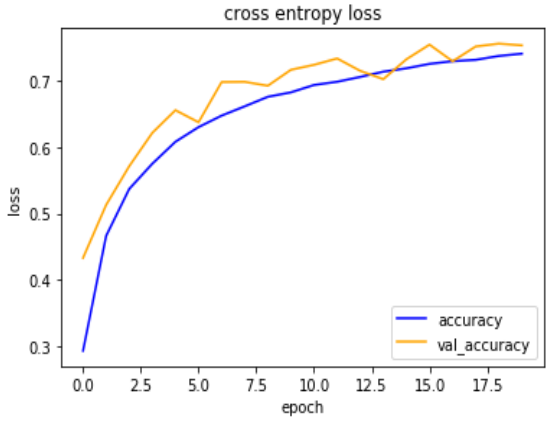

Fig 4 Precision curves of the model

The graph (Fig 3) shows the error evolution on the training data and those of validation, while the evolution of the accuracy of the model on the training data and those of validation (Fig 4).
It appears that the error on both sides gradually decreases with the number of epochs. It ranges from 1.8 to 0.91% for the learning sample and from 1.6 to 0.69 for the validation sample.
The accuracy of the model in turn, to both sides, increases with the number of epochs; this reflects that at each epoch, the model learns more information. It ranges from 39.90% to 73.13% for learning data and from 47.258% to 76.95% for validation data.
By applying this model to our test sample, we obtained an accuracy of 75.55% of well classified images and 24.45% of images were misclassified as you can see through the confusion matrix below.

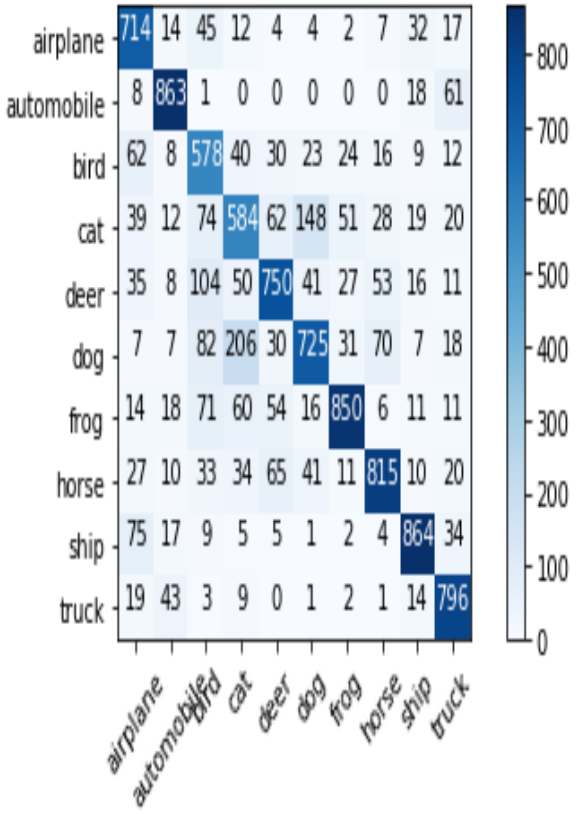

Fig. 5 Confusion matrix of the model

The table below gives the results of the first approach, allowing to compare performances of MLP to those of CNN.

Table 1: Summary of results.

Methods	Error	Precision
MLP	2.88 %	37 %
MLP + Dropout	1.79 %	48 %
MLP +Dropout +Batch	1.07 %	51 %
CNN	1.03 %	64 %
CNN + Dropout	0.69 %	75.55 %

With the first MLP network with ReLU activation function and with a less dense architecture, we obtained an error rate of 2.88% on the test set and an accuracy of 37%. The Dropout method and the upward revision of the number of units (1024) have tried to improve the result and decrease the rate of over-learning (over fitting) and reduced to 48% accuracy and 1.79% the classification error on the test set.
The Batch with the Dropout method for the third MLP, made possible to decrease slightly the error to 1.0 7% and increasing slightly the precision (51%).
Note that for the last two MLP networks with Dropout, we applied a probability of retention of the units ($p = 0.5$) on each of the layers

When we switched to a convolutional neural network, we saw the best results appear. As we can see in Table 1, the architecture for the convolutional neural network used on the CIFAR 10 is composed of 4 convolution series, 2 Max pooling followed by three fully connected layers.
With the first CNN network with ReLU activation function and a dense architecture, we obtained an error rate of 1.03% on the test set and an accuracy of 64%.
The Dropout method as well as the batch gave better results due to the decrease in the rate of over fitting for the second CNN.
The classification error was thus reduced to 0.69% and the accuracy to 75.55% on the test set. For all neural networks, a variant of the "Mini-Batch" stochastic gradient algorithm, the Adam adaptive moment estimation method was used and obtained better results.

4.3. Application of cross validation technique

To try to evaluate each of our models, we used the Cross validation technique which is in fact a method of estimating the reliability of the model based on a sampling technique.
The model will be evaluated using 5 fold cross validation. The value $k = 5$ was chosen to provide a baseline both for repeated evaluation and not to be large enough to require a long run time.
Each sample test data will represent 20% of the training data set, either 10,000 examples, which is the size of the actual test data set chosen for this problem.
The training data are intermixed before being divided, and the sample mixture is performed each time, so any model that we will evaluate, will have the same sets of training data and testing in each fold, providing a comparison of apples to apples between models.
Thus, we trained our model with 25 as the number of epochs and 32 examples as default batch size.
The data tests for each fold have been used to evaluate the model at each epochs training; this allowed us later to create learning curves to estimate the performance of the model.
Below, the different accuracy obtained. Each corresponding to a fold, as well as the curves illustrating the error variation for the learning and validation data (Fig 6), the precision variation for the same data (Fig 7). The hand-held mosquito box giving the median value of the precision of our model is evaluated at 95.5 %.

>72.270: Accuracy of the first fold
> 87.400: Accuracy of the second fold
> 95.520: Accuracy of the third fold
> 97.800: Accuracy of the fourth fold
> 98.180: Accuracy of the fifth fold

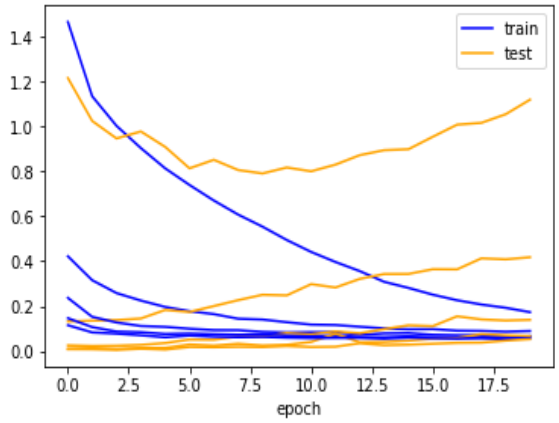

Fig 6 Error curves with cross validation

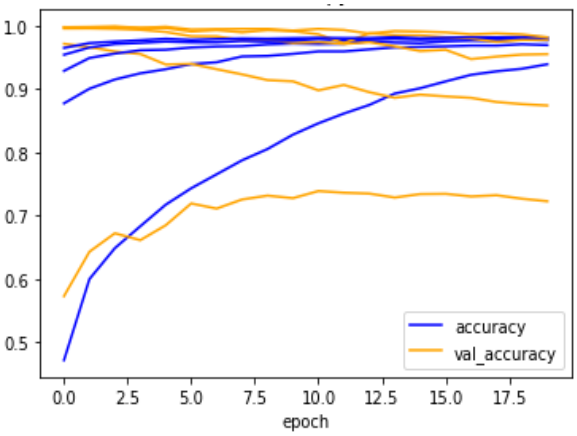

Fig 7 Precision curves with cross validation

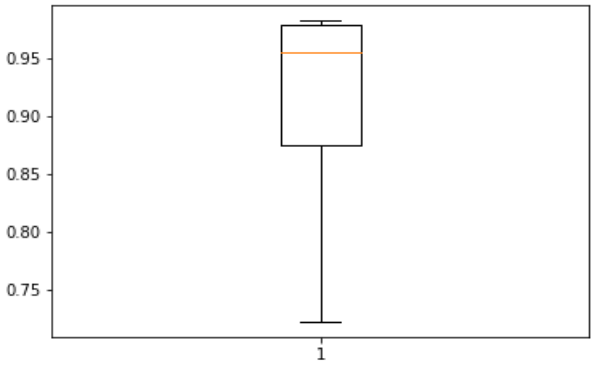

Fig. 8 The mustache box giving the median value of the model accuracy

Table 2 summarizes the results of this second approach, allowing to compare the performance of the used classifiers.

Table 2: Summary of results.

Methods	Error (%)	Accuracy (%)
MLP	0.97	58.8
CNN	0.32	95.5

The accent being placed on convolutional networks, we have found in general that an important and deep convolutional neuron network gives better results.

The performance of our network degrades if a convolutional layer is removed. For example, the elimination of one of two intermediate layers trains a loss approximately 5% for the performance of the network. Therefore, depth is essential to achieve good results. The results obtained improved as we deepened our network and increased the number of epochs.

The learning base is also a determining element in convolutional neural networks, so you must have a large learning base to achieve better results**.**

5. Conclusion

Our work was to develop a basic convolutional neural network solution for the automatic recognition of patterns. We have discussed the basics of pattern recognition and neural networks in general and those convolutional in particular. With regard to pattern recognition, an emphasis was placed on its process as well as a quick reminder of scientific methods of pattern recognition. Although all of its steps are necessary, only two are more than essential for the task of pattern recognition: Feature extraction and classification. We introduced the convolutional neural networks by presenting different types of layers used for classification: the convolution layer, rectification layer, the pooling layer and the fully connected layer. In terms of implementation, we designed a model and different architectures and we showed the different results obtained in terms of accuracy and error.

The comparison of founded results, whether with the CNN or MLP architectures that we had wanted to associate with this research, showed that the number of epochs, the size of the learning base and the depth of networks, are important factors that contribute to get the performance.

In addition, network parameters are difficult to define in advance. It is for this reason that we have defined different architectures in order to obtain better results in terms of accuracy and error.

We encountered some problems in the implementation phase, the use of a CPU made the execution time too expensive. In order to solve this problem we must use deeper convolutional neural networks deployed on a GPU instead of a CPU on larger bases [17].

The contribution of this research is at the same time, on the study, the design and the implementation of a specific type of neural networks with very interesting properties: the convolutional neural networks. These networks are specifically designed to process very large images so easily and able to take into account their variability.

Références

[1] C. Archaux, A. Martin, A. Khenchaf, “Détection par SVM – Application à la détection de churn en téléphonie mobile prépayée, Extraction et Gestion des Connaissances”, In Revue des nouvelles technologies de l’information, Vol 2, Clermont Ferrand, France, 2004, pp. 597-598.

[2] Nirmala and Sachchidanand., “Object Classification to Analyze Medical Imaging Data using Deep Learning”, International Conference on Innovations in information Embedded and Communication Systems (ICIIECS), ISBN 978-1-5090-3295-2, pp. 1-3, 2017

[3] C. M Bishop, “Neural Networks for Pattern Recognition”, Oxford University Press, 1995.

[4] L. BOTTOU, “Stochastic gradient descent tricks. Neural Networks: Tricks of the trade “, Paris 2012.

[5] D. Djabeur and Mohammed, «Mise au Point d’une Application de Reconnaissance de Formes », Master thesis in Computer Science, Abou Bakr Belkaid University, Tlemcen Algeria 2017.

[6] G. Dreyfus, J.M. Martinez, M. Samuelides, M.B. Gordon, F. Badran, S. Thiria, “Apprentissage statistique, Réseaux de neurones; Cartes topologiques; Machines à vecteurs supports”; Eyrolles Paris 2007

[7]RUI Wang, WEI Li, Runnan Qin and Jinzhong Wu “Blur Image Classification based on Deep Learning”, IEEE, ISBN 978-1-5386-1621-5 pp. 1-6, 2017

[8] M. YOUCEF, “Deep Learning pour la classification des images”, Master thesis in Computer Science, Abou bekr Belkaid Tlemcen University, Tlemcen, Algeria 2017

[9] P. Gillot, A. Temmari, B. Jenny and Yurii Nesterov, “Algorithme de Descente de Gradient Stochastique avec le filtre des paramètres pour entraînement des réseaux à convolution profonds”, University of Bordeaux INP, CNRS, LaBRI, UMR5800, F-33400 Talence France 2016.

[10] Krizhevsky, Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in Advances in neural information processing systems, pp. 1097–1105, 2012.

[11] R. Sustika, R.A. Yuliani, E. Zaenudin, F. Hilman, “On Comparison of Deep Learning Architectures for Distant Speech Recognition”, 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), ISBN 978-1-5386-0659-9, pp. 19-21, 2017

[12] M. Kunt, G. Coray, J.P. Granlund, R. Ingold, M. Kocher, “Reconnaissance des formes et analyse des scenes”, Technical and scientific collection of telecommunications of the CENT, ver. 3, 2000.

[13] H. Nemmour and R. Diboune, “Utilisation des machines à vecteurs de Support (SVM) pour la reconnaissance des chiffres manuscrits”, Final thesis of State Engineer in Electronics, University of Sciences and of Technology Houari Boumediene, Algeria 2006 - 2007.

[14] E. Poisson and C. Viard-Gaudin, “Réseaux de neurones à convolution. Reconnaissance de l’écriture manuscrite non contrainte”, Scientific article from the Polytechnic School of the University of Nantes, France IRCCyN UMR CNRS 6597, January 2014, pp. 5-8,

[15] A. Mohd Azlan, R. Muhammad Abdullah, Y. Mohamad Yusri, I. Muhammad Iqbal, “Design of Lighting Control System Using Back Propagation Learning Algorithm for Hydroponic Plant Via LABVIEW Environment”, Asia Pacific Conference on Engineering Technopreneurship 2012, pp. 50-56, 2012

[16] Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural

networks from Overfitting," The Journal of Machine Learning Research, vol. 15, pp. 3-6 2014.
[17] J.P. Buanga, S. Ntumba and R. Kabamba I, "Enhanced parallel skyline on multi-Core Architecture with low memory space cost", IJCSI, vol 13, Issue 5, September 2016

First Author received his B.Sc. in Computer Science degree in 2014 from Université Notre-Dame du Kasayi, Kananga, DR Congo. He is currently pursuing his M.Sc. studies at the Department of Computer Science and Mathematics of Université de Kinshasa, Kinshasa, DR Congo. His research interests include the Machine Learning, Software Engineering and Computer Networking. He works as visiting lecturer at the Université de Kananga (Kananga, DR Congo), Université Reverend KIM and Université Orthodoxe au Congo (Kinshasa, DR Congo) since October 2016.

Second Author received his B.Sc. in Computer Science degree from Université de Mbujimayi, Mbujimayi, DR Congo, in 2012. She is currently pursuing his M.Sc. studies at the Department of Computer Science and Mathematics of Université de Kinshasa, Kinshasa, DR Congo. Research area: Data analysis.

Third Author is Professor and Head of Mathematics and Computer Science department of the Université de Kinshasa. As publications, Author of many publications, such as: "Enhanced Parallel Skyline on multi-core architecture with lax Memory space Cost", IJCSI, volume 13, Issue 5, September 2016, Data mart approach for stock management model with a calendar under budgetary constraint, IJCSI, volume 15, Issue 5, September 2018, Poster et the 2nd International conference on Big Data Analysis and Data Mining, San Antonio, USA, 30 november-01 December 2015 "; Data Mart Approach for Stock Management Model with a calendar Uner Budgetary constraint, IJCSI, volume 15, Issue 5, September 2016.

Fourth author is professor at the Mathematics and Computer Science department of the Université de Kinshasa. Director of the Computer Science laboratory at the faculty of Sciences, at the Université de Kinshasa. He is author of many articles in many scientific journals like in IJCSI .Poster and the 2nd International conference on Big Data Analysis and Data Mining, San Antonio, USA, 30 November-01 December 2015.