ContourNet : a new deep convolutional neural network for 2D contour classification

Mhedhbi Makrem
CRISTAL LABORATORY
National School of Computer Sciences
Tunisia (2010), Manouba, Manouba
Makrem.mhedhbi@ensi-uma.tn

Mhiri Slim
CRISTAL LABORATORY
National School of Computer Sciences
Tunisia (2010), Manouba, Manouba
Slim.mhiri@ensi-uma.tn

Ghorbel Faouzi
CRISTAL LABORATORY
National School of Computer Sciences
Tunisia (2010), Manouba, Manouba
Faouzi.ghorbel@ensi-uma.tn

ABSTRACT

In this paper, we present a new deep convolutional neural network to classify 2d contours, described by a sequence of points coordinates representing the boundary of objects. Several works dealt with this subject, even those using learning, but few works use deep learning. This is due to the fact that contours data are very narrow and inappropriate for convolution. To enrich this representation, we use curve evolution and consider simultaneously a multi-scale representation of a contour. Associated with coordinates, curvature estimated at each point is the most used descriptor who can help distinguishing objects. Despite deficiency of large 2d contour datasets, required for a convergent learning, the use of several additional techniques, such as data augmentation, lead to results outperforming the state of the art. We train ContourNet on MPEG-7 database CE-1 part B, witch achieves 100% for Top-1 accuracy rate on MPEG-7 test set, and 91.78% on Kimia216 dataset.

Keywords

Contour description, Shape classification, CNN, Deep learning, Data augmentation, curve evolution.

1. INTRODUCTION

Classification and clustering of planar shapes described by their contour are of great importance for automation of several tasks such as the manufactory control (fault detection, automatic assembly), text recognition (sorting postal, RADAR control), security (digital fingerprints, facial, retinal)…. Given a set of planar shapes such as MPEG-7, KIMIA, ANIMAL bases, it is desirable to divide them into homogeneous groups sharing common characteristics, in order to facilitate the retrieval of images with similar content. When the number of groups is not known, it is a clustering, but when the number of groups is known, it is a classification. In both cases, it will be necessary to design discriminating descriptors making it possible to assign the form to a specific group in the first case, and to assign a label to a form in the second one. Several criteria should be verified by these descriptors, namely, invariance according to geometric transformations (translation, rotation and scale), robustness with respect to noise, occlusion and field of view. These descriptors must also be quantified and be part of a metric space, allowing measurement of similarity between shapes. Approaches cited in literature can be divided into main categories: category of approaches dealing with a single scale of a contour, and category of approaches using a multi-scale representation of contours.

In the first category, we can mention works like shape context [Bel02], curve edit distance [Seb03], the phase of Fourier descriptors [Bar05], the inner distance [Lin07], etc. With shape context [Bel02], authors introduce a new shape descriptor and a similarity distance measure to evaluate likeness between shapes. At each point of the shape (reference point), a local descriptor is created, based on the distribution of all remaining points, with regards to reference point. All obtained vectors are then embedded in a log-polar space, divided in many bins, according to length, and angle between a vector and a reference direction. This resulting histogram is called shape context descriptor, and used to evaluate correspondence between two points on two different shapes. Then, authors estimate the transformation that best align the two shapes. Distance between two

shapes is computed as the sum of matching errors between all matched points, added to the magnitude of aligning transformation. In [Lin07], Ling use also shape context but replace Euclidean distance by geodesic distance under some hypothesis to reduce computing complexity. This assumptions allow method to gain about 9% in accuracy rate, compared to original shape context. In [Seb03], Curve edit distance is used to estimate similarity of shapes. Based on curve length, arc-length parameterization and curvature, authors look for the function, from functions space, that minimize required energy to align curves. This function must satisfy the additivity property so that global alignment is the sum of many local alignments. Bartolini [Bar05] use phase of Fourier descriptors with dynamic time warping to compute similarity between shapes. This contribution focuses on importance of information contained in phase of Fourier coefficients, often neglected in favor of amplitude information, to ensure rotation invariance of descriptor. Using phase of Fourier descriptor makes inappropriate the use of Euclidean distance between shapes, thus dynamic time warping is well suited to measure similarity between shapes, as it allows matching of elastic deformations of a part of shapes. Best performances achieved for this category of approaches are those of shape context with inner distance [Lin07], with error rate of 14.60% for MPEG-7 dataset, and 2.63 for Kimia216 dataset.

Among approaches from multi-scale category, we can mention visual parts [Lat00], curvature scale space and its derivatives [6,7,8], beam angle statistics [Ari03], convexity-concavity multi scale representation [Ada04], triangle area representation [Ala07], Eigen and Fisher barycenter [Tho09], etc. In visual parts contribution [Lat00], Latecki aim to measure similarity between 2d shapes, by matching significant parts of shapes instead of matching the whole shapes. A shape is considered as a union of several concave or convex sub parts. The use of digital curve evolution is performed by substituting two consecutive line segments with a single line segment, joining the endpoints of initial segments. This contour simplification is done to eliminate noise digitization and segmentation errors. Then, each shape is represented by a tangent function, which is a step function. Similarity measure is deduced by computing area difference between tangent function of two shapes, after aligning their functions. Another contribution, part of current category of approaches, is Curvature Scale Space CSS [Mok03]. In this contribution, authors consider the normalized arc-length parameterization of the initial contour, with several variants extracted using curve evolution process. It consists of smoothing original contour shape by Gaussian functions with progressive σ parameter. These successive convolutions eliminate gradually points with high absolute value of curvature estimated, until reaching a convex shape (ellipse or circle in most cases). Shrinkage of evolved contours caused by curve evolution process is compensated with a motion vector to obtain evolved curves with same length as the original contour. CSS image descriptor is then obtained as solution of equation $\kappa(\mu,\sigma) = 0$, κ is curvature value, σ is the curve evolution parameter, and μ is the point coordinate. To compute similarity between two shapes, authors begin by shifting one shape until the two maximum of curvature coincides, and retain the sum of Euclidean distances between matched maxima. Another contribution in this category is the beam angle statistics BAS [Ari03]. In this work, a BAS descriptor is computed as follow: from each boundary point on the shape, a set of beams is considered, linking the reference point to all remaining points on the shape. Angles between each pair of beams help extracting the topological structure of contour. Use of multi-scale information in this context is realized by considering multiple levels of neighbors with a function called K-neighborhood. This leads to K-curvature function, regarded as a random variable. The BAS descriptor is a vector of third order statistical moments, which is invariant to translation, rotation, and scale and insensitive to distortions. In [Ada04], Adamek use curve evolution to generate a multi-scale representation of a contour. An estimation of convexity/concavity is done on each point of contour on original shape and all of its evolved versions, and result is a two-dimensional matrix descriptor (MCC). Columns of this matrix represents contour points while rows represents levels of evolution: σ. A dynamic time warping is then used to find the optimal global alignment between contours, and measure distance between corresponding shapes. In [Ala07], inspired by works on 3D shapes, Alajlan al introduce a new descriptor, called TAR, to characterize 2D shapes. Unlike previous works, where triangle area was normalized by global signature, alajlan and al locally normalize signature by dividing it by length of hypotenuse triangle. By using multiple neighborhood scales, TAR descriptor highlights both local and global details, making thus higher the discrimination power of shapes. Inspired by DTW, authors use a similar technique, called Dynamic Space Warping to match starting points of two shapes, before measuring similarity. To emphasize intuition behind matching, distance measurement is divided by a complexity term, which is the mean of absolute differences between highest and lowest signatures, through all points and all scales. Considering global features such as circularity, eccentricity and aspect ratio to increase discrimination shapes. In [Tho09], authors improve

works done in [Seb03], by using barycentric points coordinates, instead of original points coordinates of the boundary shape, to generate several shapes of the contour: making a multi-scale representation. To ensure invariance to reflection and starting point, a phase normalization of descriptor is realized on spectral domain, giving a matrix N*M, where N represent points number and M represent scales number. Then, a linear discriminant analysis is processed on Fisher matrix to reach final descriptor.

The remainder of this paper is organized as follow: in Section II, we introduce convolutional neural networks architecture, their components, and some of the most famous ones in literature, mainly works dealing with contours. In Section III, we describe datasets used in this work, and how we preprocessed their contents. Section IV contains proposed architecture, hyper parameters settings and details of learning. Results and discussion will be in Section V and VI, and we conclude with some future works in Section VII.

2. RELATED WORKS

Since 2012, and with the advent of deep learning, as well as the impressive results it achieves, we thought about applying these solutions to the classification of plane contours. Convolutional neural networks have proved their capability to solve complex tasks such as segmentation, detection and clustering and labeling. A CNN is an artificial neural network with several hidden layers. To overcome the huge number of parameters included, and who increases dramatically each time a layer is added, convolution has been used. Unlike fully connected layers, where each neuron in layer j is connected to all neurons in the previous layer j-1, convolutional layers take the form of a stack of small-scale filters. Therefore, the input of a neuron is no longer equal to the weighted sum of all neurons in the previous layer, but it is equal to the weighted sum of a narrow neighborhood surrounding the affected neuron. In addition, the sharing of parameters results in a modest number of them. As cited above, these models can fit linear functions with millions of parameters, but it remains inefficient when applied to complex data. Therefore, non-linearity was introduced to expand functions space covered by CNN approximations. Moreover, it makes sense to go through depth. From layer to layer, CNNs learn more and more complex discriminant features, leading to better results in classification. Activations functions used to introduce non linearity in CNNs are of two types, saturating functions and non-saturating functions. The former translate neuron's information inside a bounded interval and the latter inside unbounded one. To further reduce number of parameters, a pooling operation is convenient. It replaces each bloc of neurons with fixed size, by one neuron containing value extracted from the others. Applied inside a convolutional layer, it reduces output parameters number by a scale factor of four, at least. The first deep architecture proposed was AlexNet [Kri12]: an eight-layer neural network alternating convolution, pooling, and fully connected layers. AlexNet was used to label over than one million marked images and reach better performances of the time. Many improvements of accuracy were proposed within ZFNet [Zei14], VGG [Sim14], GoogLeNet [Sze15], etc.

Using CNN to classify 2D forms datasets, such as MPEG-7, Animal or Leaf, focusing on boundary information, and ignoring color and texture information, was first introduced in [Ata16]. This work uses binary images to learn BSCNN, composed of three convolutional layers and one fully connected layer, leading to a Softmax layer for final classification. Many data augmentation ways were used to improve its accuracy rate, which achieve on MPEG-7 dataset, 98.99 on TOP-1 metric, and 99.76 on TOP-5 metric.

In [Zha21], another architecture of CNN was introduced and called SCN, aiming to be more general by classifying forms from different datasets. SCN is composed of four convolutional layers, and a fully connected layer. After the third convolutional layer, one de-convolution layer was inserted. To overcome changes of data distribution within network, and speed up learning, a batch normalization layers were added to SCN. To increase data amount and improve learning process, data augmentation was used. Accuracy rate achieved by SCN, according to TOP-1 metric is 90.99 % on MPEG-7 dataset.

The first CNN used to classify boundary data shapes [Dro20] is called ContourCNN. Authors consider both Cartesian and polar representations for training their system. They combined circular convolution layers and priority pooling layers with two dimension space representation. Circular convolution help system to highlight circularity of contour points regardless of abstract representation of these points. Priority pooling layers do not remove points in a regular manner, but iterates on them until having a fixed size. ContourCNN is composed of three circular convolution layers, three priority pooling layers, and one global average pooling layers, connected to two fully connected layers used for classification. Because of poorness of datasets like MPEG-7 and Animal, authors have tested ContourCNN on EMNIST dataset, and achieve an accuracy rate less than 97%. However, considerations made by authors does not seem to be contribution in contour classification for these reasons: 1) Circular

convolution layer can simply be replaced by padding, 2) priority pooling do not make improvements with contour data since quantity of data is weak, 3) benchmark used to test robustness is EMNIST dataset, and we do not know performances on famous datasets, such as MPEG-7, Kimia, etc.

3. DATASET, COORDINATE SYSTEM AND CNN

In this paper, we aim to design a CNN to classify MPEG-7 CE-SHAPE-1 part B objects. This dataset is one of the most used benchmarks to compare performances of classification techniques. It contains images of seventy class of shapes, each class has twenty different objects, resulting in 1400 images. An overview of all these classes, and a sample of objects of some classes are illustrated in figures Fig.1 and Fig.2.

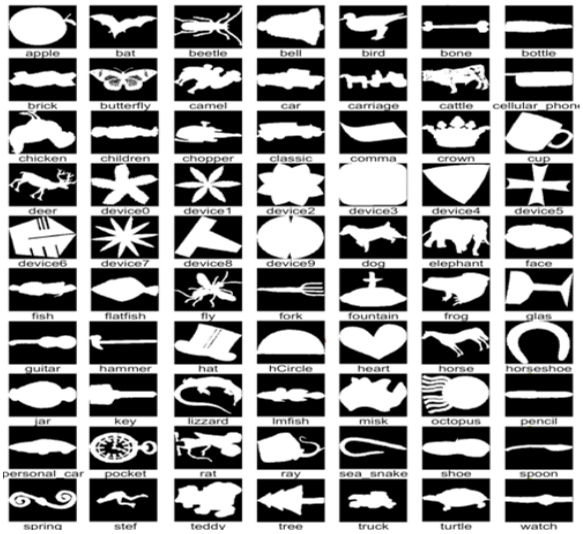

Fig. 1 : MPEG-7 class objects.

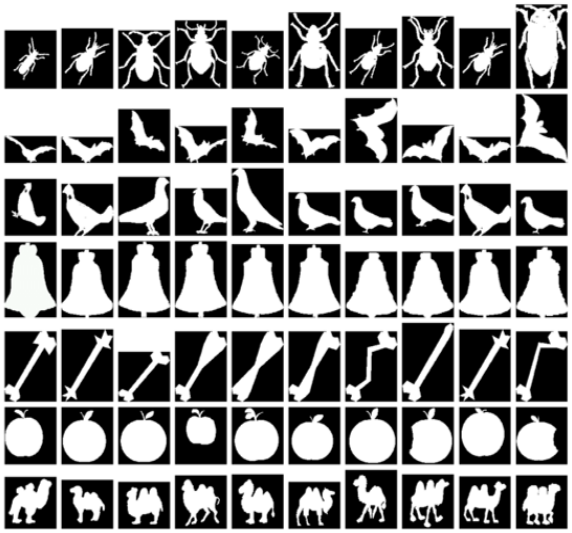

Fig. 2 : MPEG-7: some forms of some classes.

As we are concerned with system coordinates, we extract boundary from image object, and apply a natural normalized arc-length parameterization to get coordinates of N points forming contour. To preserve details of all shapes, we use a value of N=100. Then, each contour is represented by N*3 matrix: first column is x-coordinate, second column is y-coordinate, and third column is curvature estimated on this point. On one point of the curve, curvature describes how much curve direction varies over a small distance. It was used to construct invariant descriptors such as CSS [Mok03], CED [Seb03], BAS [Ari03], etc. So we add curvature information to improve discriminating power of our recognition system.

To see how CNN can perform a classification task on MPEG-7 dataset, we begin with the simplest way considerations. We design a CNN with only one convolutional layer, and three fully-connected layers. Experiments show that 256 filters in the first layer lead to better accuracy rates, compared to other filter numbers.

To overcome the small number of objects used for training, we use data augmentation. It is a label-preserving technique, allowing generation of new objects, obtained from initial ones, by applying rotations using angles of $\pi/4$, $\pi/2$, $3\pi/4$, π, $5\pi/4$, $3\pi/2$ and $7\pi/4$. This technique improve robustness of CNN and let it learn objects in different positions and orientation. Finally, our dataset is composed of 11200 contours of seventy class object, each class have 160 contours. We split then dataset to train set and test set using proportions of 70% and 30%.

Training CNN with various learning rates until 30 epochs show that optimal learning rate is 5 10^{-3}, according to Fig. 3.

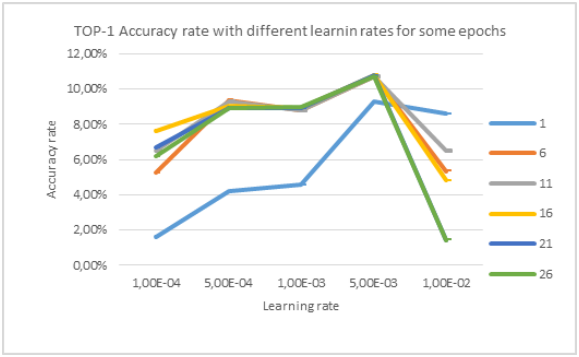

Fig. 3 : Choosing appropriate learning rate according to TOP-1 metric.

Recognition rate obtained with this CNN according to TOP-1 (resp. TOP-5) is very low: 10.71% (resp. 26.79%) after six epochs of training, and remain the same even with 30 epochs. To perform this rates, we added a batch normalization layer, after the

convolutional layer of our CNN. Theoretically, this layer attenuates effects of gradient instability, by standardizing and normalizing output of the previous convolutional layer. By adding batch normalization, Top-1 accuracy rate increases by 3.28%, Top-5 accuracy rate increases by 5.56%, both on test set.

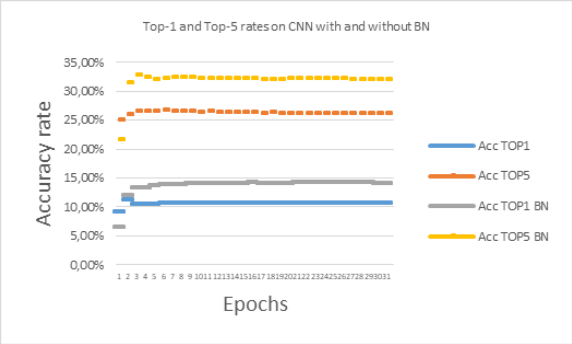

Fig. 4 : Impact of BN use on our CNN.

Figure Fig. 4 compares accuracy rates Top-1 and Top-5 for CNN with and without BN layer.

Despite slightly improving Top-1 and Top-5 accuracy rates, batch normalization negatively affects CNN stability. In fact, comparing accuracy rates of our CNN without and with BN, on noisy data generated from initial data by adding a Gaussian noise, show that CNN without BN learn better than CNN with BN.

With obtained results, it seems that CNN with one convolutional layer are very weak to classify contours of MPEG-7 images, and it is recommended to use more deep architectures. The problem arising with this idea is the shallowness of data representation. In fact, with convolution filters of size 3*3 in the first layer, its output is a one-dimensional vector, on which we cannot apply more convolutions. To defeat this drawback, we use curve evolution [6, 18].

4. CONTOURNET

Curve Evolution & Data Representation

Curve evolution is a technique allowing generation of several new curves, obtained from original one, by smoothing it with Gaussian function. Evolving a given curve with Gaussian function with σ parameter, try to smooth it by eliminating some salient point. Applying successive evolutions to a curve lead finally to a convex curve without salient points. Mathematically, it consists of convolving initial curve with a Gaussian function. Evolution process example is shown in Fig. 5.

The problem of choosing appropriate σ values is far from being resolved. At the beginning, Mokhtarian and Bober [Mok03] chose a regular range to evolve a curve and stop when curve become completely convex. Ben Khlifa and Ghorbel [7,8] introduce different discretization of σ value space.

By experiments, we find that using a regular range from 1 to σ_{max} = 60, leads to convex curves for all shapes we study. Using curve evolution process, a contour from MPEG-7 dataset will be represented by initial contour, concatenated to evolved versions of it. So, input to our CNN will be a 100*(3*61) matrix. Preprocessing process in mentioned in Fig.6. Such representation is very more dense and appropriate for applying convolution, and allowing us to go deeper with CNN.

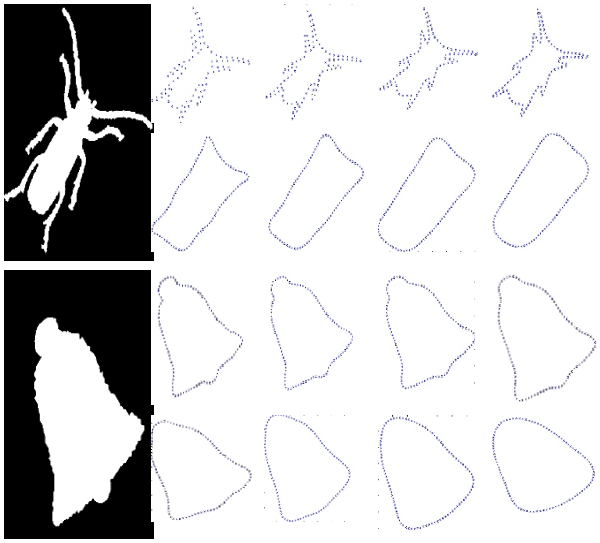

Fig. 5 : Evolution process with σ = 1, 2, 3, 4, 10, 20, 30, 40.

ContourNet

To design a CNN architecture, we need to decide about all of its hyper parameters: number of layers, number of filters in each layer, learning rate, etc. We used cross validation to define optimal number of layers of our CNN. Since CNN with one convolutional layer don't lead to good performances, we studied architectures with two convolutional layers. The table in Fig. 7 illustrates mean validation error with different number of filters. This table shows that architecture with 256 filters in the first layer and 96 filters in the second layer has the best qualification, regarding all studied architectures, to learn features from our dataset.

ContourNet Architecture

ContourNet is composed of two convolutional layers with respectively 256 and 96 filters, and three fully-connected layers. Besides convolution, the two first layers are using a ReLU function to introduce a non-linearity on data, and followed by a Max-Pooling operation, to keep important information while reducing data volume.

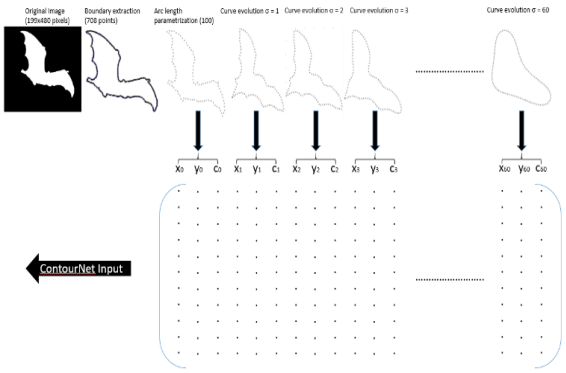

Fig. 6 : Preprocessing data for training.

	64	96	128	160	192	224	256
64	25.24	28.24	31.25	43.18	50.83	78.78	74.49
96	22.42	24.75	26.53	29.92	37.79	71.88	67.90
128	21.12	21.89	22.64	26.96	32.17	38.15	48.61
160	18.90	19.43	21.30	22.26	24.25	28.11	29.11
192	19.30	20.14	19.22	21.12	22.05	28.20	30.48
224	19.12	20.78	20.94	19.31	20.80	22.42	25.87
256	18.66	18.13	19.32	19.72	19.77	21.06	27.58

Fig. 7 : Mean validation error Top-1 on two convolutional layers architectures.

Our goal is to find discriminant features in initial contour or in one of its evolved version, that's why we set a filter size of 3*3. In the first layer of ContourNet, filters have size of 3*3, with a stride of 3. According to this configuration, each filter tries to learn discriminant features in the initial contour or in smoothed versions of it. On the convolution result, we apply a ReLU function and a max pooling using a bloc size of 2*2. Output of the layer is introduced to the second layer. In the second layer, filters have a size of 3*3, with unitary stride. This layer attempt to learn nonlinear combinations of features learned in the first layer. Output is then reshaped and delivered to the classification part of our network. Classification part of ContourNet contain three fully connected layers with respective size of 192, 128 and 70. Figure Fig. 8 shows ContourNet architecture. To guarantee circularity of contour data, we use padding by adding the last point to the beginning, and the first point to the end, that's why number of lines of an input is 102.

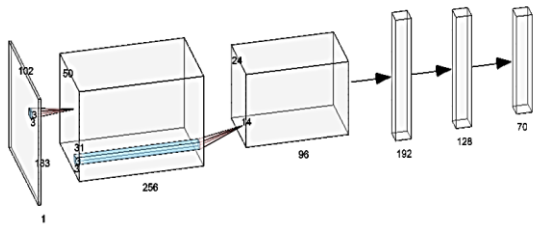

Fig. 8 : ContourNet architecture.

Learning Details

To train ContourNet, we use stochastic gradient descent with various learning rate within several epochs. Weights in all layers were initialized from a normal distribution. Training and tests were carried out on an i7-7700 processor machine, with an NVIDIA GeForce GT 730.

5. EXPERIMENTS

To evaluate ContourNet performances, we use a test set containing 3360 contours, extracted randomly from MPEG-7 dataset. To evaluate generalization of ContourNet, we use Kimia216 shapes, as all of them exist in MPEG-7 dataset. To evaluate stability of ContourNet, we also use a set of 1400 contours, obtained by adding a normal Gaussian noise to original dataset. We use Top-1 and Top-5 metrics to measure error rate.

Among learning rate values tested, a learning rate of 10^{-3} has best effect on learning process. It is shown in figure Fig. 9 that with this rate, ContourNet needs only two epochs to achieve 100% of recognition.

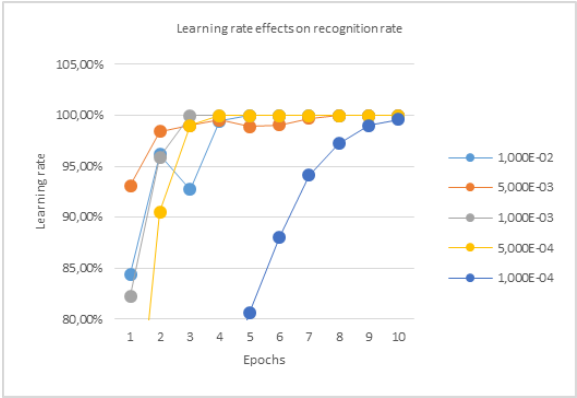

Fig. 9 : Top-1 recognition rate with different learning rates.

Using $5*10^{-4}$ as learning rate, performances achieved are not far from those obtained with 10^{-3}. It took four epochs before achieving 100% of recognition. Except for these two rates mentioned above, learning process is not monotone and show fluctuations when epochs increases. With learning rate of 10^{-2}, recognition rate of ContourNet exceed 96%, decrease to 92% and then reach 100% of recognition after four epochs. With learning rate of $5*10^{-3}$, recognition rate of ContourNet exceed 99.5%, decrease to 98% and then reach 100% of recognition after seven epochs. As we can see, performances gap between learning rates is not large enough, that's why we need to study generalization of ContourNet.

Kimia216 is a dataset containing eighteen object classes, each class has twelve images. All classes in Kimia216 are also part of MPEG-7 dataset. With data augmentation, we create a dataset containing common

classes in MPEG-7 and Kimia, composed of 1728 shapes, each class is represented by 96 shapes. Using labels of MPEG-7 classes, we tested these shapes on ContourNet. Results shows that training ContourNet with 10^{-2} learning rate, for six epochs, only through 7840 inputs, lead to an architecture with 100 % recognition rate on MPEG-7 test set, and 92.53 % recognition rate on Kimia classes object (129 misclassified shapes among 1728). Using 5 10^{-3} (resp. 10^{-3} and 5 10^{-4}) as learning rate, the best recognition rate reached is 83.56 % (resp 59.60 % and 55.28 %). Fig. 10 illustrates these results.

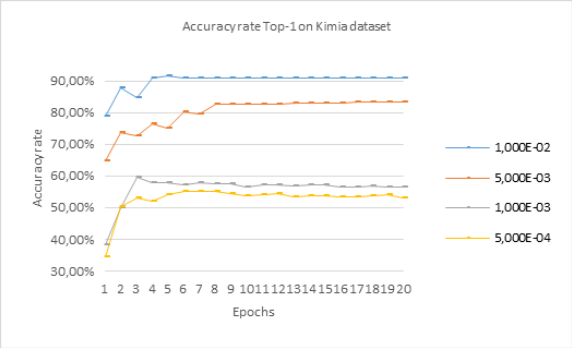

Fig. 10 : ContourNet Tests on Kimia dataset.

On noised dataset we use, we notice that using 10^{-2} as learning rate, give the most stable architecture, compared with others ones. Fig. 11 shows that this architecture achieves a recognition rate of 97.5% on noisy data, while a learning rate of $5*10^{-3}$ (resp. 10^{-3} and $5*10^{-4}$) cannot exceed 86% (resp. 65% and 60%).

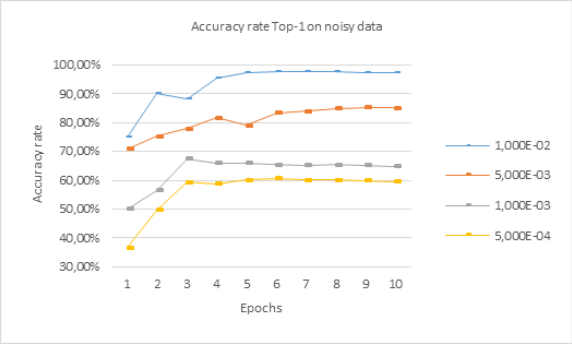

Fig. 11 : ContourNet stability.

Experiments above show that with a CNN with two convolutional layers, containing 256 and 96 filters, followed by three full-connected layers, and using a learning rate of 10^{-2} for five epochs, is an architecture who outperforms the stat-of-the-art on MPEG-7 CE-SHAPE-1 part B object classification. In Fig. 12 we dress a comparative results of ContourNet, with other works from state of the art on MPEG-7 dataset classification.

6. DISCUSSIONS

To understand more what ContourNet was learning, we study shapes from Kimia with which our architecture fails. We notice that all 96 shapes from bone, glass, heart, misk, camel, car, children, face, fountain and ray classes are wholly recognized. Bird class has 28 misclassified shapes, classic car class has 8 misclassified shapes, elephant class has 5 misclassified shapes, fork class has 40 misclassified shapes, hammer class has 8 misclassified shapes, key class has 32 misclassified shapes and turtle class has 8 misclassified shapes. The eight misclassified classic cars were all predicted as jar, corresponding objects have some similarity in their shapes. The five misclassified elephants were all predicted as turtle, corresponding objects have also some similarity in their shapes. The thirty-two misclassified keys were all predicted as guitar, and corresponding shapes are also similar. The eight misclassified turtles were predicted as beetles but corresponding shapes are not similar (see Fig. 16). For the remaining classes, prediction was various for each class. For the twenty-eight misclassified birds, eight were predicted as frog, eight were predicted as chicken, eight were predicted as fork, and four shapes were predicted as elephant (see Fig.13). For the eight misclassified hammers, three shapes were predicted as carriage, and five shapes were predicted as rat (see Fig 14). The worst class in prediction was the fork class, with forty misclassified shapes. One shape was seen as a bone, one other shape was seen as an lm fish, three shapes were predicted as misk, four shapes were seen as cup, five shapes were seen as hammers, seven shapes were predicted as shoppers, eight shapes were seen as spring and eleven shapes were considered as a lizard (see Fig.15) . We illustrate in Fig. 13, Fig. 14, Fig. 15 and Fig. 16 some misclassified shapes from Kimia216, and their corresponding prediction.

Approach	Retrieval Accuracy Rate (%)
WARP [Bar05]	58.50
VP [Lat00]	76.45
CED [Seb03]	78.17
CSS [Mok03]	81.12
BAS [Ari03]	82.37
MCC [Ada04]	84.93
IDSC [Lin07]	85.40
SCN [Zha21]	90.99
FBcC [Tho09]	95.50
ContourNet	100.00

Fig. 12 : ContourNet Performances comparison.

We observe also that in most cases, the number of misclassified shapes for one class, represent the same shape with multiple rotations applied. For the bird

class for example, the misclassified shapes were composed of three original shapes and their rotated objects for seven rotation angles, which is equal to 24 shapes. The remaining four shapes were predicted as belonging to elephant class.

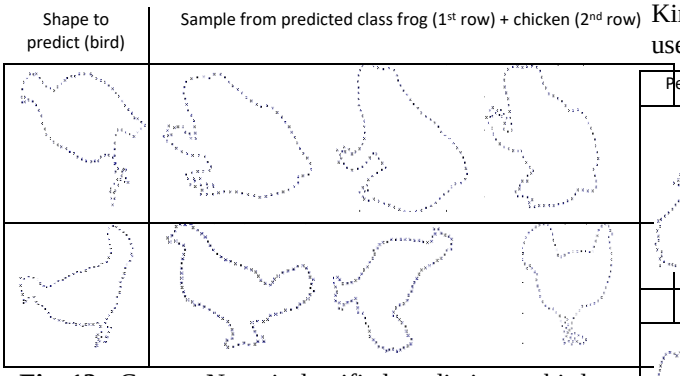

Fig. 13 : ContourNet misclassified prediction on bird objects.

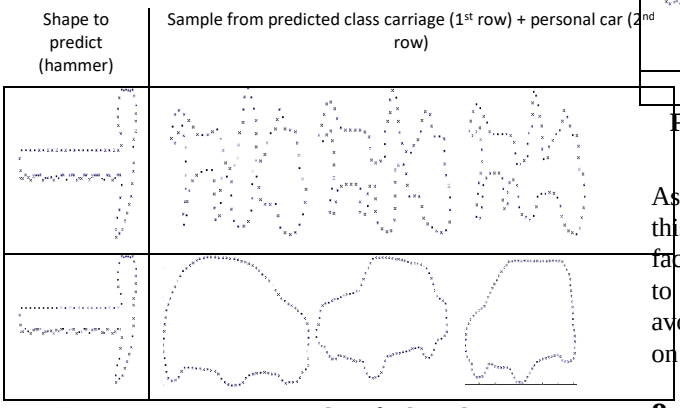

Fig. 14 : ContourNet misclassified prediction on hammer objects.

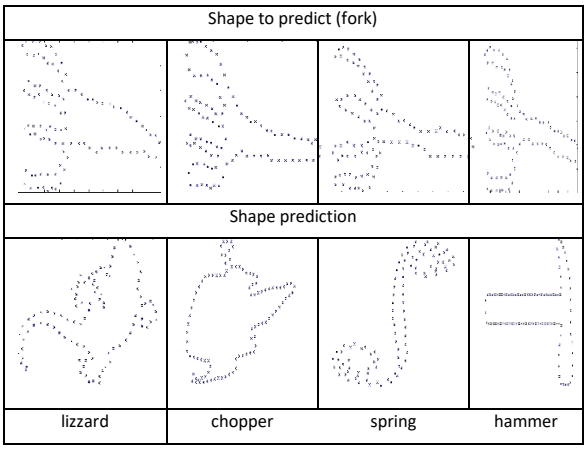

Fig 15 : ContourNet misclassified prediction on fork objects.

7. CONCLUSION & FUTURE WORKS

In this paper, we presented a new convolutional neural network, called ContourNet, to label contours of MPEG-7 dataset. To be convenient with convolution, we apply a curve evolution on initial contours to generate other versions of same objects. A data augmentation technique was used to enrich dataset, based on height angle rotations. A validation step was used to specify optimal hyper parameters to conceive ContourNet. We tested our architecture on both MPEG-7 test set and a noisy version dataset and Kimia dataset, and shows that it performs models used in the state of the art.

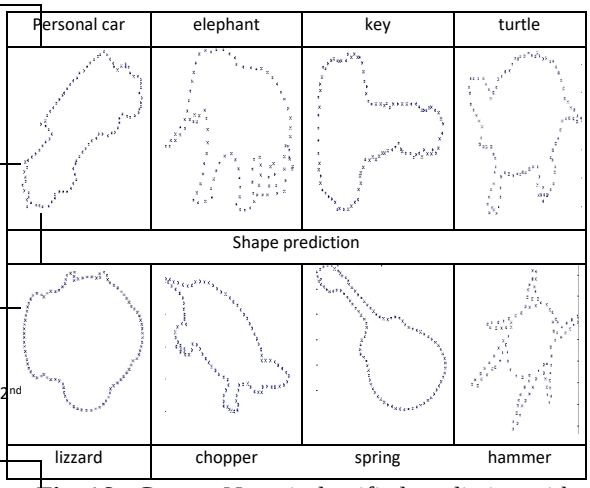

Fig. 16 : ContourNet misclassified prediction with only one class objects.

As a future work, we are thinking about extending this approach to 3D shapes, and how to overcome the fact that R3 is unordered space. Using auto-encoders to label contours is another track to explore, by avoiding supervisor need. More attention will be paid on how to choose scales for curve evolution.

8. REFERENCES

[Lat00] Latecki L.J. and Lakamper R., Shape Similarity Measure Based on Correspondence of Visual Parts. IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 22, No 10, October 2000, pp 1185-1190.

[Bel02] Belongie S, Malik J. and Puzicha J., Shape Matching and Object Recognition using Shape Context, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 24, No 4, April 2002, pp 509-522.

[Seb03] Sebastian T., Klein P. and Kimia B., On aligning Curves, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 25, No 1, January 2003, pp 116-125.

[Bar05] Bartolini I., Ciaccia P., and Patella M., WARP: Accurate Retrieval of Shapes using Phase of Fourier Descriptor and Time Warping Distance, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 27, No 1, January 2005, pp 142-147.

[Lin07] Ling H. and Jacobs D., Shape Classification using Inner-Distance, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol 29, No 2, February 2007, pp 286-299.

[Mok03] Mokhtarian F. and Bober M., Curvature Scale Space Representation : Theory, Applications, and MPEG-7 Standardization, Kluwer Academic Publishers, 2003.

[Ari03] Arica N. and Vural F., BAS: A perceptual Shape Descriptor based on the Beam Angle Statistics, Pattern Recognition Letters, 2003, pp. 1627-1639.

[Ada04] Adamek T. and O'Connor N. E. A Multiscale Representation Method for Nonrigid Shapes with a Single Closed Contour, IEEE Trans. On Circuits System and Video Technology, 2004, pp 742-753.

[Ala07] Alajlan N., Rube E. I., Kamel M. S. and Freeman G., Shape retrieval using triangle-area representation and dynamic space warping, Pattern Recognition 40, 2007, pp 1911-1920.

[Tho09] Thourn K., Kitjaidure Y. and Kondo S., Eigen and Fisher Barycenter contour for 2d shape classification, International Conference on Computing and Communication Technologies, 2009.

[Kri12] Krizhevsky A. Sutskever I. and Hinton G., Imagenet Classification with deep convolutional neural networks, NIPS, 2012.

[Zei14] Zeiler M. D. and Fergus R., Visualizing and understanding convolutional networks, ECCV, 2014.

[Sim14] Simonyan K. and Zisserman A, Very deep convolutional networks for large scale image recognition, arXiv preprint arXiv: 1409.1556.2014.

[Sze15] Szegedy C. and al, Going deeper with convolutions, CVPR, 2015.

[Dro20] Droby A. and El-sana J., ContourCNN: convolutional neural network for contour data classification, arXiv preprint arXiv: 2009.09412v2.

[Ata16] Atabay H. A., Binary shape classification using convolutional neural networks, IIOABJ, 7(5), 332-336, 2016.

[Zha21] Zhang C. and al, SCN: A Novel Shape Classification Algorithm based on Convolutional Neural Network, Symmetry 2021, 13, 499.