Solving Non-image Learning Problems by Mapping to Images

Boris Kovalerchuk
Dept. of Computer Science
Central Washington University
USA
BorisK@cwu.edu

Bedant Agarwal
Indian Institute of Technology
Kharagpur
India
BedantAgarwal9@gmail.com

Divya Chandrika Kalla
Dept. of Computer Science
Central Washington University
USA
DivyaChandrika.Kalla@cwu.edu

Abstract*—Transforming non-image Machine Learning (ML) problems into the image recognition problems, opens an opportunity to solve these problems, by powerful deep learning algorithms. This paper proposes a new CPC-R algorithm, which converts non-image data into images, by visualizing non-image data. Then several deep learning CNN algorithms are exploited to solve the learning problems. The CPC-R algorithm preserves high-dimensional information in 2-D. It splits the attributes of an n-D point into pairs of its values and visualizes pairs as 2-D points, in the same 2-D Cartesian coordinates. Next, it maps pairs to grey scale or color intensity values to encode the order of pairs producing a heatmap. This paper reports the results of computational experiments with CPC-R for different CNN architectures, and methods to optimize the CPC-R images. These results show that the combined CPC-R and deep learning CNN algorithms can solve non-image ML problems, at the state-of-the-art level of accuracy for the benchmark datasets.

Keywords: Compute Vision, Convolutional Neutral Networks, Deep Learning, Machine Learning, Raster images, Visualization, Non-image data, Data conversion.

1. INTRODUCTION

Integration of machine learning, and visual analytics is an important part of data science research. Deep Learning (DL) algorithms have shown remarkable success in solving image recognition problems. Several DL architectures developed for one type of images have been successful also in other types of images demonstrating efficiency of knowledge transfer to other types of images. Converting non-image data to images by using visualization expands this knowledge transfer opportunity to solve a wide variety of Machine Learning problems [4,10]. In this way, a non-image classification task is converted into the image recognition task to be solved by powerful DL algorithms.

A. Single value mapping

In the basic mapping method, a function F maps each value a_i of the n-D point $\mathbf{a}=(a_1,a_2,\ldots,a_n)$ to a cell/pixel p_i with coordinates (p_{i1},p_{i2}), $F(a_i)=(p_{i1},p_{i2})$. Here, pixel p_i gets intensity equal to the coordinate value a_i that can be normalized, $I(p_i)=a_i$. We call such mapping a *single value mapping*. The image size defines the *mapping density*.

Consider n-D point **a** and an image of size $\lceil n^{0.5} \rceil \times \lceil n^{0.5} \rceil$, where a_1 is mapped to pixel (1,1) of this image, a_2 is mapped to pixel (1,2), and so on. For n=3, the size of this image is 2×2 with mapping a_1 to pixel (1,1), a_2 to pixel (1,2) and a_3 to pixel (2,1). Such dense image representation produces very small images, e.g., 10×10 image for 100-D point **a** with all pixels densely map elements of **a**. In a *less dense mapping*, an image is larger, e.g., 3×3 with mapping a_1 to (1,1), a_2 to (2,2), and a_3 to (3,3) where only 3 pixels out of 9 encode point **a** with other pixels blank.

The number of choices to locate all n values a_i of **a** in the image of $k \times k$ pixels is large. These locations can be *optimized* for increasing the *classification accuracy*. In [10], it is done by discovering relations between attributes a_i and then locating pixels of related attributes next to each other instead of exhaustive search of best locations. The innovation of this approach is that PCA, K-PCA or t-SNE mapping is used not as a visualization itself, but only to identify the *location* of the pixels on the 2-D plot. While, in general, it is productive approach, a specific relation discovered by such unsupervised methods can be irrelevant to the supervised classification task. In addition, the distances that PCA, and t-SNE discover to locate pixels nearby can corrupt n-D distances [13].

The algorithm GLC-L in [4] represents an n-D point **a** as a 2-D *graph*, which is a sequence of connected straight lines under different angles where line L_i has the length of a_i of n-D point **a**. This disadvantage of this approach is the *size* of its raster image, because drawing graph edges requires many pixels. Both methods [4, 10] use images that they produce as inputs to CNN algorithms for *image recognition* and both reported success on different benchmark data sets. Their disadvantage is a single value mapping requiring a cell (pixel) for each a_i of n-D point **a**.

We propose a new **algorithm CPC-R**, that uses a *pair values mapping with preserving* n-D information. It requires *two times less* cells (pixels) than single value mappings. Its images are smaller than in [4] and with significantly less occlusion than in [4] by removing the graph edges.

B. Pair values mapping

An alternative to a single value mapping F is mapping a pair of values (a_i,a_j) to a single pixel p, $H(a_i,a_j)=(p_{1ij},p_{2ij})$. We call it a *pair values mapping*. In contrast with a single value mapping F, the values a_i and a_j are encoded by the *location* of the pixel (p_{i1},p_{i2}). In the simple mapping, the coordinates of the pixels are a_i and a_j, $(p_{1ij},p_{2ij}) = (a_i,a_j)$ and $(p_{1i,i+1},p_{2i,i+1}) = (a_i,a_{i+1})$. This can require *rounding* a_i and a_j to be integers. For the odd n, the last pair can be formed as (a_n,a_n). We use a *grey scale intensity* of the pixels to identify a pair, where the first pair is black; the second pair is dark grey and so on until light grey. Alternatively, a color sequence is used as in *heatmaps*. The values of intensities and colors can be optimized at the learning stage. We denote this method as **CPC-R**.

The *advantage* of pair values mapping is that $\lceil n/2 \rceil$ pixels can represent n-D point *without loss of information*, i.e., two times less pixels than a single pixel mapping, which *simplifies images*. CPC-R optimizes the location of the pixels by allowing any pairs (a_i,a_j) not only (a_i,a_{i+1}) to improve classification accuracy. A simplest version generates randomly

978-1-7281-9134-8/20/$31.00 ©2020 IEEE
DOI 10.1109/IV51561.2020.00050

a fixed number of alternatives and tests accuracy on training data for these alternatives.

C. Pair values mapping algorithm CPC-R

The main part of this paper describes the experiments with the CPC-R and CNN algorithms to classify the real world and simulated data. The efficiency of CPC-R+CNN algorithm is tested successfully by evaluating the accuracy of the classification of several CNN algorithms in solving learning problems, on images produced by CPC-R visualizations on benchmark data. For non-integer data, values are discretized to map to the CPC-R matrix.

The base CPC-R algorithm version 1.0 consists of the following steps:

S1. *Split attributes* of an n-D point $\mathbf{x}=(x_1,x_2,\ldots,x_n)$ to consecutive pairs (x_1,x_2), (x_3,x_4), $\ldots(x_{n-1},x_n)$. If n is odd repeat the last attribute of **x** to make n even.

S2. *Set up a cell size* to locate pairs in the image. Each cell can consist of a single pixel or dozens of pixels.

S3. *Locate pairs* (x_i,x_{i+1}) in the image at the cell coordinate pair (x_i,x_{i+1}) in the image.

S4. *Assign the grey scale intensity* to cells from black for (x_1,x_2) and very light grey for (x_{n-1},x_n). Alternatively, *assign heatmap colors*.

S5. *Optional: Combine* produced images with context information in the form of average images of classes.

S6. *Call/Run ML/DL* algorithm to discover a predictive model.

S7. *Optional: Optimize* intensities in S4 and pair coordinates beyond the sequential pairs (x_i,x_{i+1}) in S1 by the methods ranged from random generating a fixed number of alternatives to genetic algorithms with testing ML prediction accuracy on these alternatives.

The order of intensities allows full restoration of the order of the pairs and their values from the image if all pairs differ. If some pairs are equal, they will *collide* in the image location. In this case the intensity of the *last pair* is shown in the image in this basic version. If each cell uses only a single pixel in step 2, a small image with 10x10 pixels will represent a 10-D point, when each attribute has 10 different values, by locating five grey scale pixels in this image.

Below we describe and illustrate in Fig. 1 the generation of raster images in CPC-R for the Wisconsin Breast Cancer (WBC) data. It is a 10-D point **x**=(8, 10, 10, 8, 7,10, 9,7,1,1) built from the 9-D point by duplicating x_9=1 to make x_{10}=1 needed to have 5 pairs. The CPC-R generates an image for this **x** as follows:

- Forming consecutive pairs of values from **x**=(8, 10, 10, 8, 7,10, 9,7,1,1): (8.10), (10,8), (7,10), (9,7), (1,1).
- Filling cell (8,10) for the pair (x_1,x_2)=(8,10) according to the grey scale value for the first pair (black).
- Filling cell (10,8) for pair (x_3,x_4)=(10,8) according to the grey scale value for the second pair (very dark grey).
- Filling cell (7,10) for the pair (x_5,x_6)=(7,10) according to the grey scale value for the third pair (grey).
- Filling cell (9,7) for the pair (x_7,x_8)=(9,7) according to the grey scale value for the forth pair (light grey).
- Filling cell (1,1) for the pair (x_9,x_{10})=(1,1) according to the grey scale value for the fifth pair (very light grey).

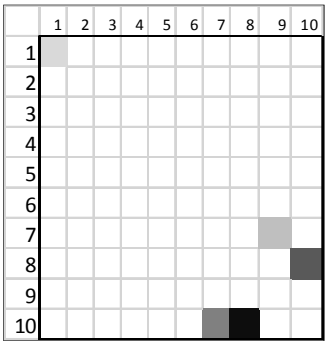


Fig. 1. 10- D point (8, 10, 10, 8, 7, 10, 9,7,1,1) in CPC-R.

This visualization is *lossless* if values of all pairs (x_i,x_{i+1}) do not repeat. Below we present a treatment of such colliding pairs and other steps in more elaborated versions of CPC-R algorithm. The first elaboration of steps 3 is below.

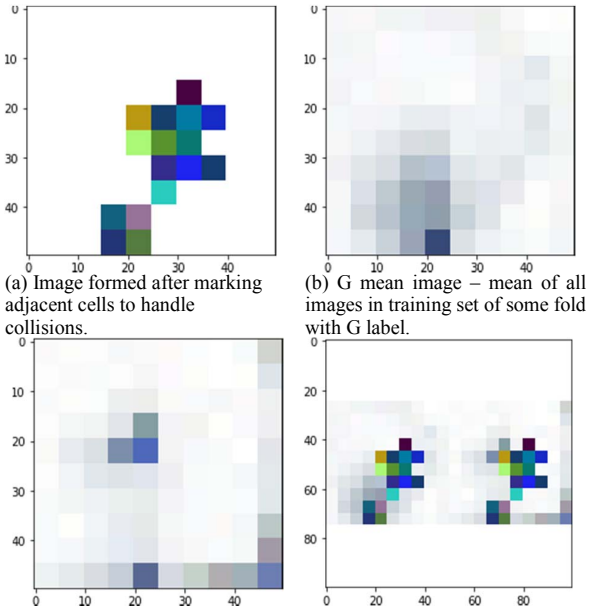
CPC-R version 2.1 with using **adjacent** cells in step 3:
3.1. *Set image coordinate system* (origin at upper left, or low left.
3.2. *Locate non-colliding pairs* (x_i,x_{i+1}) in the image at cell coordinate values (x_i,x_{i+1}).
3.3. *Set the starting adjacent cell* (right, left, top or bottom) for collided pairs.
3.4. *Set order of filling of adjacent cells* (clockwise or counterclockwise).
3.5. *Locate colliding pairs* (x_i,x_{i+1}) in the cell adjacent to cell (x_i,x_{i+1}) according to 3.3 and 3.4.

CPC-R version 2.2 with **splitting** cells in steps 3 and 4:
3.1. *Set image coordinate system* (origin upper left, or low left
3.2. *Locate non-colliding pairs* (x_i,x_{i+1}) in the image at cell coordinate values (x_i,x_{i+1}).
3.3. *Split* colliding cells to vertical strips.
4. 1. *Assign* the grey scale intensity from black for (x_1,x_2) and very light grey for (x_{n-1},x_n) for *non-colliding pairs.* Alternatively, *assign heatmap colors* for non-colliding pairs.
4.2. Assign the grey level intensity of the respective pair of values (x_i, x_{i+1}) to the strip for *colliding pairs*.

For instance, a cell 8×8 will consist of two strips 4x8 when two pairs are collided. For three colliding pairs, three strips 3x8, 3×8, and 2×8 are produces. For four colliding pairs, four strips 2x8, are produces and for five pairs, five strips 2x8, 2×8, 2×8, 1×8, and 1×8 are produces.

CPC-R version 2.3 with using the **darkest** intensity of colliding pairs step S4:
4. 1. *Assign* the grey scale intensity from black for (x_1,x_2) and very light grey for (x_{n-1},x_n) for *non-colliding pairs*. Alternatively, *assign heatmap colors* for non-colliding pairs.
4.2. *Assign* the darkest grey level intensity of all pair with equal values (x_i, x_{i+1}) for *colliding pairs*.

CPC-R version 3.0 with **context** incorporated in step 5:
S5: *Put* an image of each n-D point on the top of both G mean (mean image of the training cases of class 1), and B mean (mean image of the training cases of class 2). Both the images are put side by side, to form a double image.



(a) Image formed after marking adjacent cells to handle collisions.

(b) G mean image – mean of all images in training set of some fold with G label.

(c) B mean image – mean of all images in training set of a particular fold with B label.

(d) Final double image – image with image (a) superimposed with mean images (b) and (c)

Fig. 2. Building double color image with context of two means for 34-D ionosphere data.

The dimension of the image after forming a double image is (w)×(2*w). Then (2*w)×(2*w) image is formed by padding the double image. Fig. 2 illustrates this process of constructing a double image and superimposed with mean images for each class from training data with padding. This approach *enriches the images with context* for analysis by humans and deep learning algorithms. Fig. 2bc visualizes the difference between two means in its *34-D richness without degrading it to a lossy single distance number*.

CPC-R version 4.0 with incorporated **optimized intensities** and **pairing of coordinates** (x_i,x_j) in step 7:
S7. *Optimize* intensities and pairing coordinates beyond sequential pairs (x_i,x_{i+1}) by methods ranged from random generating a fixed number of alternatives and testing ML prediction accuracy on training data for these alternatives to genetic algorithms. In all experiments below with data from UCI ML repository. we use 10-fold Cross Validation (CV).

II. Experiments with WBC Data

Below we report the results of experiments with WBC Data to test the feasibility of the CPC-R approach for CNN architectures, and different number of pixels per cell, which represents each pair (x_i,x_{i+1}). In these experiments the grey scale intensities are first assigned to the pixels as follows: $I(x_1,x_2)$=0 (darkest), $I(x_3,x_4)$=51, $I(x_5,x_6)$=102, $I(x_7,x_8)$=153, $I(x_9,x_{10})$=204. In some experiments, intensities and pairing coordinates (x_i,x_j) are optimized for accuracy.

A. Initial experiments

In the first set of experiments **E1** and **E2** with CPC-R **1.0** and **grey scale** defined above, we worked with the CNN architecture used in [4] that we denote as CNN64 for short and summarize below to be able to compare results:

- Convolutional layer with 64 output channels, a kernel shape of 2x2, stride of 2×2 and RELU activation
- Convolutional layer with 64 output channels, a kernel shape of 2x2, stride of 2x2 and RELU activation
- Pooling layer with pooling size of 2×2
- Drop out layer with 0.4 fraction of input units to drop.
- Convolutional layer with 128 output channels, a kernel shape of 2×2, stride of 2×2 and RELU activation RELU Convolutional layer with 128 output channels, a kernel shape of 2x2, stride of 2×2 and relu activation
- Pooling layer with pooling size of 2×2
- Drop out layer with 0.4 fraction of input units to drop.
- Fully connected layer with 256 output nodes and RELU activation
- Drop out layer with 0.4 fraction of input units to drop
- Fully connected layer with number of output nodes equal to the number of classes, with softmax activation.

Other experiments **E2** use the InceptionResNetV2 CNN architecture [12], denoted as ResNetV2 below: Flatten → Dense(256, relu) → Dense(1,sigmoid). In E1 and E2 we explore the impact of image sizes and origins of coordinates: upper left corner (ULC) or low left corner (LLC)**.** The achieved accuracies are in Table 1.

TABLE 1. ACHIEVED ACCURACIES OF EXPERIMENTS E1 AND E2 IN 10-FOLD CROSS VALIDATION.

Experiments	Model	Image Size	Origin	Accuracy
E1	CNN64	100×100	ULC	95.61
E2	ResNetV2	80×80	LLC	94.45

B. Experiments dealing with colliding values

In experiments C1 and C2 we explored the impact of colliding values along with image sizes on accuracy using ResNetV2 and LeNet5 architecture [8]**.** See Table 2.

Experiments **C1** are with CPC-R **2.1** and grey scale images, different orders to **fill the adjacent cells,** and two CNN architectures. They show *slight increase in accuracy* over E1 and E2, demonstrating that such method to resolve collision is beneficial. It also shows that accuracy of 80x80 images is practically the same as of larger 160x160 images.

TABLE 2. ACHIEVED ACCURACIES IN EXPERIMENTS C1 AND C2 WITH PUTTING COLLIDING VALUES TO ADJACENT CELLS IN 10-FOLD CV

Experiments	Model	Image Size	Accuracy
C1	ResNetV2	80×80	95.61
C2	ResNetV2	320×320	96.20
C2	LeNet5	50×50	94.88

Experiments C2 with CPC-R **2.2** deal with colliding values in CPC-R visualization by **splitting colliding cells** to vertical strips. C2 results are slightly better than E1 and E2, which do not address collision. The LeNet5 architecture provided 94.88% accuracy for smaller images, 50x50.

C. Experiments with double images and padding

Experiments D1-D8 with CPC-R **3.0** explore the impact of adding **context** via double images on the accuracy for different architectures, image sizes, and other parameters listed below

for each D1-D8. It is testing a hypothesis that adding the context can increase accuracy. The goal of these experiments is exploring the impact of these combinations of properties of images on accuracy of classification.

We varied the parameters of experiments D1-D8. Table 3 contains only results with the parameters that produced the best accuracies. The settings of experiments are as follows:

D1 with superimposed colliding values (v1.0),
D2 with filling adjacent cells (v.2.1),
D3 with splits of cells (v 2.2),
D4 with darkest cells for colliding pairs (v2.3),
D5 with red levels and grey means,
D6 with red levels and colored means,
D7 with grey cases and colored means,
D8 with all colored images,

TABLE 3. ACHIEVED ACCURACIES OF 10-FOLD CROSS VALIDATION EXPERIMENTS D1 -D8 WITH DOUBLE IMAGES.

Experiments	Model	Double image size	Accuracy
D1	LeNet5	60×60	97.22
D2	LeNet5	200×200	96.77
D3	LeNet5	60×60	95.90
D3	CNN64	200×200	95.90
D4	CNN64	160x160	96.19
D5	LeNet5	60×60	97.66
D6	LeNet5	60×60	97.80
D6	CNN64	100×100	97.36
D7	LeNet5	60×60	97.51
D8	LeNet5	60×60	97.37

Table 3 shows improved accuracy in experiments D with double images over experiments E and C with singe images.

While the accuracy of experiments D2 is slightly less than D1, it is with much larger double images. The accuracy of D3 dropped relative to D1 and D2. We attribute it to the smaller image size. The split of cells requires larger images. The accuracy of D4 dropped slightly relative to D1 and D2, showing advantages of D1 and D2 that do not split cells.

The goal of the next experiments is exploring the impact of the color. Intensity of each color point is chosen randomly (3 intensities for 3 channels of a point). Therefore, 15 random values for 5 cells are generated. In experiments **D5** we used images with **red** levels, **grey means** and filling **adjacent** cells. Table 3 shows a *slight increase* of accuracy relative to the previous experiments D1-D4. In experiments **D6**, we generated images with **red** levels for n-D points (cases) and **colored means**. In experiments **D7**, we generated images with **grey** cases and **colored means.** In the experiments **D8**, we generated images, with **color** cases and **colored means** (see Fig. 2). In the experiments **D9**, we generated the images, with **red** levels, and **colored means** and **optimized intensities and pairing** of coordinates (x_i,x_j).

Table 3 shows that the *optimization of pairing was most beneficial* to improve the accuracy, while the accuracy can vary due to the DL random elements. One of the results of these experiments is slightly better than in the experiments presented above. The optimization method used was a simple one: random generating a fixed number of perturbed alternatives and testing ML prediction accuracy on training data for these alternatives. It indicates that more sophisticated methods can improve results further.

D. MLP experiments

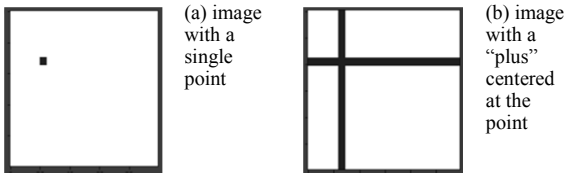
To check how important is the use of DL models vs. simpler MLP models, we conducted experiments with the following Multilayer Perceptron (MLP) architecture that is simplification of CNN64:

- Layer with 64 output channels and RELU activation. This hidden layer with 64 nodes and a rectified linear activation function.
- Layer with 64 output channels and RELU activation. This hidden layer with 64 nodes and a rectified linear activation function.
- Dropout layer with fraction of input units to drop is 0.4.
- Layer with 128 output channels and RELU activation.
- Drop out layer with fraction of input units to drop is 0.4.
- Fully connected layer with number of output nodes equal to the number of classes, with softmax activation.

The achieved accuracy in **8 experiments** with **MLP** using single and double images are from 70.31% to 76.51% in 10-fold cross validation using the same experiments setting as we used for DL models above. This clearly shows the advantages of using CPC-R with CNN models that gave from **94.45% to 97.8%** accuracy vs. much lower accuracy of MPL models.

III. EXPERIMENTS WITH SWISS ROLLS DATA

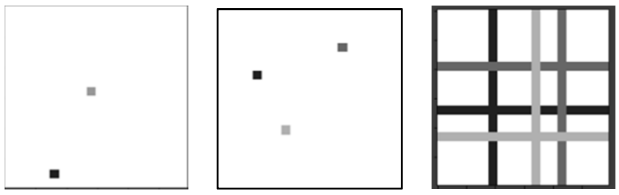
Below we present the results for two Swiss roll data sets (2-D and 3-D) [3] used as a benchmark for non-linear ML tasks. It is commonly used to test abilities of the algorithms to discovery patterns of the data located on a low-dimensional manifold being isometric to the Euclidean space. Since there are only two attributes in the 2-D Swiss roll, each CPC-R image contains a single point with coordinates (x_1, x_2) as shown in Fig. 3a. To enrich the image, we generated an image with a "plus" centered at the points. See Fig. 3b.



(a) image with a single point

(b) image with a "plus" centered at the point

Fig. 3. CPC-R images for 2-D Swiss roll point (x_1, x_2).



(a) 3-D point as two points (x_1,x_2), (x_3,x_3).

(b) 3-D point as three points: (x_1,x_2), (x_2,x_3), and (x_3, x_1)

(c) 3-D point with "pluses" at (x_1,x_2), (x_2,x_3), and (x_3, x_1)

Fig. 4. CPC-R images for 3-D Swiss roll point (x_1,x_2,x_3).

Each image for the 3-D point is visualized, in CPC-R, as two 2-D points: (x_1,x_2) and (x_3,x_3). See Fig. 4a. Alternatively, it is visualized as three 2-D points: (x_1,x_2), (x_2,x_3), and (x_3, x_1).

See Fig. 4b. We also generated CPC-R images, for 3-D Swiss roll, with "pluses" at three 2-D points: (x_1,x_2), (x_2,x_3), and (x_3, x_1) (Fig 4c). Table 4 shows the max accuracy 97.87% and 97.56% for 2D and 3D Swiss rolls, respectively, in our experiments.

TABLE 4. RESULTS FOR SWISS ROLL 2D AND 3D.

Model	Data	Image	Max accuracy by varying intensities
LeNet5	2-D roll	Point	96.56
LeNet5	2-D roll	"plus" at point	97.88
LeNet5	3-D roll	2 points	97.56
LeNet5	3-D roll	3 points	94.69
LeNet5	3-D roll	3 "pluses" at point	96.31

IV. EXPERIMENTS WITH IONOSPHERE AND GLASS DATA

These experiments have been conducted in the same setting as WBC data experiments above. Tables 5 and 6 summarize results of these experiments. All of them shows 10-fold cross validation for accuracy on the validation data.

TABLE 5. ACHIEVED ACCURACIES OF 10-FOLD CV EXPERIMENTS WITH SINGLE AND DOUBLE IMAGES FOR IONOSPHERE DATA.

Experiments	Model	Original image size	Accuracy
E1	ResNetV2	100×100	86.9
E1	MLP	50×50	74.58
C1	CNN64	100×100	89.04
C1	MLP	50×50	87.43
D5	ResNetV2	100×100	92.03
D6	CNN64	100×100	93.46
D7	CNN64	50×50	91.73
D7	MLP	100×100	90.91
D8	CNN64	100×100	92.46
D8	MLP	50×50	90.29

TABLE 6. ACHIEVED ACCURACIES OF 10-FOLD CV EXPERIMENTS WITH SINGLE AND DOUBLE IMAGES FOR GLASS DATA

Experiments	Model	Image size	Accuracy
E1	CNN64	100×100	96.45
E1	ResNetV2	100×100	95.58
E1	MLP	100×100	89.8
C1	ResNetV2	100×100	96.01
C1	MLP	50×50	94.6
D5	CNN64	50×50	94.13
D6	ResNetV2	100×100	95.06
D7	CNN64	100×100	95.9
D7	MLP	100×100	94.7
D8	ResNetV2	100×100	96.8
D8	MLP	200×200	92.78

V. COMPARISONS WITH OTHER STUDIES

The exact comparison of accuracies of different methods is difficult due to multiplicity of the ways to test the results. Some authors report percentage without telling a way how it was tested. Even when all accuracies are obtained in 10-fold cross validation of accuracies, the comparison is not exact, because different authors make different random splits of data into these 10 folds, put different time limitations on model optimization, and can use other unreported properties of the testing process. Some authors use 10-fold CV, but report it for the area under the ROC curve (AUC) or F measure not accuracy. Therefore, we refrain from direct comparison with results, which are not in 10-fold Cross Validation (CV) of accuracies that we used.

WBC data. In our experiments the best of accuracies for WBC data involve optimization of (1) pairing coordinates, (2) ordering these pairs, and (3) values of intensities of cells that encode them. The LeNet5 architecture is the winner that is simplest among the explored network architectures that allowed the smallest size of the images of 30x30 pixels. Our results of 95.9-97.8% in D1-D9 experiments **are in the range** of the current published results for WBC data for 10-fold CV. Other published results of WBC data for interpretable C4.5, J4 and fuzzy decision trees are 94.74%, 94.36% and 95.27%, respectively, on 10–fold CV summarized in [6]. Different less interpretable methods such as SVM and Neural Networks produced higher accuracy that range from 97.97% to 99.51% on the 10–fold CV also summarized in [6]

Swiss roll. Table 7 summaries the accuracies of 10-fold CV of models constructed by: (1) CNN architecture on the original numeric data 1-DC CNN without transforming them to images, (2) GLC-L CNN on images from [4] and (3) CPC-R proposed in this paper on images. For CPC-R this table shows the accuracies from Table 3 that are slightly higher.

TABLE 7. COMPARISON OF ACCURACIES WITH RESULTS FROM [4].

	Numeric data	Images	
	1-DC+CNN [4]	*GLC+CNN [4]*	*CPC-R*
Swiss roll 2-D	72.50	97.43	97.87
Swiss roll 3-D	96.18	97.55	97.56
WBC	96.92	97.22	97.66

The max accuracy reported in [14] for 2-D Swiss roll are 96.6% for Isomap, 71.24% for PCA, and 51% for Autoencoder. The DL algorithms applied to similar spirals in [15] are very accurate with multiple simpler models made by using Tensorflow playground. The accuracies from these papers on Swiss rolls are at the same level as in our experiments. Thus, our work shows abilities of DL methods to *model manifolds such Swiss roll represented in the visual form as images*, in addition to the original numeric form used in other studies. In this way, we expand the ways how manifolds can be modeled and discovered.

Ionosphere data. For these data, 98.78% on training data and 100% on validation data was reported in [6] for FSP algorithm that used another visual representation of n-D points as graphs and logical rules, not heatmaps and DL. It was done by single random splits of the data with 70% to training and 30% to validation data. Therefore, this result is not applicable for the direct comparison. The same paper summarized 10-fold CV from different sources: 93% for MLP, 94.87% for C4.5, 94.59% for Rule Induction, 97.33% for SVM without converting n-D data to images. Our achieved accuracy of 94.13% ae I this range. This means that CPC-R can be considered as competitive with these algorithms in accuracy. The advantage of CPC-R is that it can be used with any DL and MLP algorithm uniformly with lossless n-D data visualization.

Glass data. Multiple results are reported from 68.2 for C4.5 to 98.13 for random forest but without telling the accuracy evaluation method [1, 5, 9], or use 10-fold CV with area under the ROC curve (AUC) and F [11] not accuracy. Therefore, it is hard to compare these results directly with our achieved 96.01% accuracy in 10-fold cross validation. Thus,

our result is in the range of results reported in the literature and is competitive.

VI. Discussion and conclusion

How are generalizable the results with CPC-R to other data? Can we expect similar high accuracy for other data? This is a difficult question for many ML algorithms, including those used for decades. The *theoretical reason* is coming from two theorems. The first one is that Neural Networks are *universal approximators* of continuous functions, which includes CNN. The second one is Kolmogorov-Arnold superposition theorem, that every multivariate continuous function can be represented as a *superposition of continuous functions of one variable* [2]. It implies that instead of functions of one variable, **continuous functions of two variables** can be used. The CPC-R pairs (x_i,x_j) represent this situation.

How to realize this theoretical opportunity on real data? While it requires more experimenting with CPC-R and CNN architectures, extensive relevant experiments with positive results already have been conducted for another pairs-based method [7]. It is the extended Standard Generalized Additive Models called GA2M-models that consist of univariate terms and a **small number of pairwise interaction terms**. These authors announced: "the surprising result that GA2M-models have almost the same performance as the best full-complexity models on a number of real datasets".

How surprising is that CNN can find patterns in CPC-R images in datasets, where the points are spread in the image with possibly little presence of local features that would combine the nearby pixels? It would be surprising if the accuracy for the images with a single pixel per pair (x_i,x_j) will be high. In fact, CPC-R accuracy is low for very small images 10×10, where each pair occupied a single pixel. CNN was associated with discovering *local features* due to the layer design that generalizes nearby pixels. Locating colliding cells in the adjacent cells and using multiple pixels to represent each pair/cell (x_i,x_j) that ranged from 3×3 to 32×32 increased the neighborhoods and CNN efficiency. Moreover, neighborhoods are not required, because for many datasets, including bio-medical data, often individual pairs are enough [7]. To ensure that, important pairs are not missed we use perturbation (see experiments D9) in the optimization process. Also, CNN assign weights that can represent image pixels that are far away from each other, thus the locality is not mandatory.

Table 8. CPC-R use scenaria.

CPC-R version	Use scenario
1.0 Base version	Data without pair collision or unimportant collision
2.0 Collision treatment	2.1 adjacent cells are almost free - use adjacent cells 2.2 adjacent cells are occupied- split cells 2.3 first pair is most important- use darkest intensity
3.0 Adding context	Average images of different classes are distinct
4.0 Optimization	Data with non-optimal intensities and attribute pairs

The current versions of CPC-R algorithm are limited to the datasets, where the *pair relations* are enough (see Table 8) and by the abilities of CNN algorithms to *learn classification models* on these data. The Kolmogorov-Arnold and universal approximator theorems add theoretical basis to our study that such datasets and CNN models exist. The used datasets are very different showing that CPC-R can handle diverse datasets.

Can simpler models substitute CNN for CPC-R? So far, our experiments above had shown that MLP provided lower accuracy, while further studies can find such simpler models.

This paper had shown that CNN on CPC-R images produced from numeric n-D data is a feasible and beneficial ML approach. Among its advantages are **lossless visualization of n-D data,** and the abilities to add **context to the visualization** by overlaying the images of n-D points with the mean images of competing classes. The CPC-R opens the **opportunity for visual explanation** of the model it discovered, by tracing back to the **salient image pixels** and visualizing them using a heatmap similarly to what is done in many CNN image applications.

Data anonymization for ML models is another important opportunity to use CPC-R, where CPC-R converts numeric n-D data into *anonymized images* and CNN builds a model on them. Future studies are to explore the listed opportunities with more experiments, to enhance the approach and to solve more problems. The Python code at GitHub can be made available upon request.

References

[1] Aldayel MS. K-Nearest Neighbor classification for glass identification problem. In2012 International Conference on Computer Systems and Industrial Informatics 2012 Dec 18 (pp. 1-5). IEEE.

[2] Braun J., On Kolmogorov's Superposition Theorem and Its Applications, SVH Verlag, 2010, 192 pp.

[3] Balasubramanian M, Schwartz EL. The isomap algorithm and topological stability. Science. 2002, 295(5552):7-7.

[4] Dovhalets D, Kovalerchuk B, Vajda S, Andonie R. Deep Learning of 2-D Images Representing n-D Data in General Line Coordinates. In: Intern. Symp. on Affective Science and Engineering, 1-6,2018.

[5] Khan MM, Arif RB, Siddique MA, Oishe MR., Study and observation of the variation of accuracies of KNN, SVM, LMNN, ENN algorithms on eleven different datasets from UCI machine learning repository. In: 2018 4th Intern. Conf. iCEEiCT 2018, 124-129. IEEE.

[6] Kovalerchuk B, Gharawi A. Decreasing Occlusion and Increasing Explanation in Interactive Visual Knowledge Discovery. In: HCII 2018, 505-526. Springer.

[7] Lou Y, Caruana R, Gehrke J, Hooker G. Accurate intelligible models with pairwise interactions. In: 19th SIGKDD, 2013, 623-631. ACM.

[8] LeCun, Y.,Bottou, L., Bengio, Y. Gradient-based learning applied to document recognition. IEEE Proc. 1998, 86, 2278–2324.

[9] Mohit RR, Katoch S, Vanjare A, Omkar SN. Classification of Complex UCI Datasets Using Machine Learning Algorithms Using Hadoop. In: IJCSSE. 2015 Jul;4(7):190-8.

[10] Sharma A, Vans E, Shigemizu D, Boroevich KA, Tsunoda T. Deep Insight: A methodology to transform a non-image data to an image for convolution neural network architecture. Nature: Scientific reports. 2019 Aug 6;9(1):1-7

[11] Prachuabsupakij W, Soonthornphisaj N. Clustering and combined sampling approaches for multi-class imbalanced data classification. In: Advances in IT and industry applications 2012, 717-724. Springer.

[12] Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, inception-resnet and the impact of residual connections on learning. In: 31st AAAI Conference on Artificial Intelligence 2017.

[13] van der Maaten L. Dos and Don'ts of using t-SNE to Understand Vision Models, CVPR 2018, Tutorial.

[14] Van Der Maaten L, Postma E, Van den Herik J. Dimensionality reduction: a comparative. J Mach Learn Res. 2009, 10(66-71):13.

[15] Smilkov D, Carter S, Sculley D, Viégas FB, Wattenberg M. Direct-manipulation visualization of deep networks. arXiv:1708.03788. 2017