Towards Explainable Deep Neural Networks (xDNN)

Preprint submitted to Neural Networks Journal

Plamen Angelov, Eduardo Soares

***Abstract*—In this paper, we propose an elegant solution that is directly addressing the bottlenecks of the traditional deep learning approaches and offers a clearly explainable internal architecture that can outperform the existing methods, requires very little computational resources (no need for GPUs) and short training times (in the order of seconds). The proposed approach, xDNN is using prototypes. Prototypes are actual training data samples (images), which are local peaks of the empirical data distribution called *typicality* as well as of the data density. This generative model is identified in a closed form and equates to the pdf but is derived automatically and entirely from the training data with no user- or problem-specific thresholds, parameters or intervention. The proposed xDNN offers a new deep learning architecture that combines reasoning and learning in a synergy. It is non-iterative and non-parametric, which explains its efficiency in terms of time and computational resources. From the user perspective, the proposed approach is clearly understandable to human users. We tested it on some well-known benchmark data sets such as iRoads and Caltech-256. xDNN outperforms the other methods including deep learning in terms of accuracy, time to train and offers a clearly explainable classifier. In fact, the result on the very hard Caltech-256 problem (which has 257 classes) represents a world record [1].**

I. Introduction

Deep learning has demonstrated ability to achieve highly accurate results in different application domains such as speech recognition [2], image recognition [3], and language translation [4] and other complex problems [5]. It attracted the attention of media and the wider public [6]. It has also proven to be very valuable and efficient in automating the usually laborious and sometimes controversial pre-processing stage of feature extraction. The main criticism towards deep learning is usually related to its 'black-box' nature and requirements for huge amount of labeled data, computational resources (GPU accelerators as a standard), long times (hours) of training, high power and energy requirements [7]. Indeed, a traditional deep learning (e.g. convolutional neural network) algorithm involves hundreds of millions of weights/coefficients/parameters that require iterative optimization procedures. In addition, these hundreds of millions of parameters are abstract and detached from the physical nature of the problem being modelled. However, the automated way to extract them is very attractive in high throughput applications of complex problems like image processing where the human expertise may simply be not available or very expensive.

Feature extraction is an important pre-processing stage, which defines the data space and may influence the level of accuracy the end result provides. Therefore, we consider this very useful property of the traditional deep learning and step on it combined with another important recent result in the deep learning domain, namely, the transfer learning. This concept postulates that knowledge in the form of a model architecture learned in one context can be re-used and useful in another context [8]. Transfer learning helps to considerably reduce the amount of time used for training. Moreover, it also may help to improve the accuracy of the models [9].

Stepping on the two main achievements of the deep learning - top accuracy combined with an automatic approach for feature extraction for complex problems, such as image classification, we try to address its deficiencies such as the lack of explainability [7], computational burden, power and energy resources required, ability to self-adapt and evolve [10]. Interpretability and explainability are extremely important for high stake applications, such as autonomous cars, medical or court decisions, etc. For example, it is extremely important to know the reasons why a car took some action, especially if this car is involved in an accident [11].

The state-of-the-art classifiers offer a choice between higher explainability for the price of lower accuracy or vice versa (Figure 1). Before deep learning [12], machine-learning and pattern-recognition required substantial domain expertise to model a feature extractor that could transform the raw data into a feature vector which defines the data space within which the learning subsystem could detect or classify data patterns [4]. Deep learning offers new way to extract abstract features automatically. Moreover, pre-trained structures can be reused for different tasks through the transfer learning technique [8]. Transfer learning helps to considerably reduce the amount of time used for training, moreover, it also may helps to improve the accuracy of the models [9]. In this paper, we propose a new approach, xDNN that offers both, high level of explainability combined with the top accuracy.

The proposed approach, xDNN offers a new deep learning architecture that combines reasoning and learning in a synergy. It is based on prototypes and the data density [13] as well as *typicality* - an empirically derived pdf [14]. It is non-iterative and non-parametric, which explains its efficiency in terms of time and computational resources. From the user perspective, the proposed approach is clearly understandable

Plamen Angelov, and Eduardo Soares are with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK. E-mails: p.angelov@lancaster.ac.uk; e.almeidasoares@lancaster.ac.uk.

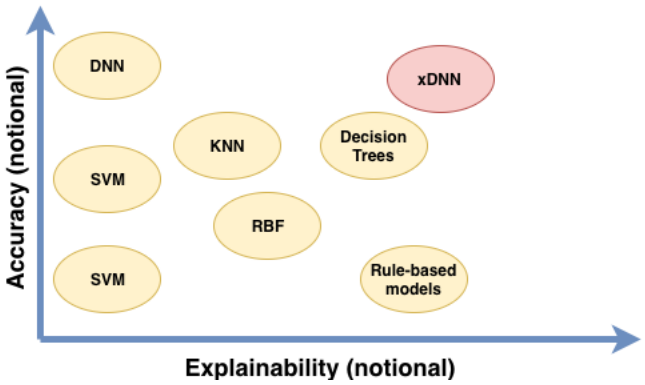

Fig. 1. Trade-off between accuracy and explainability.

to human users. We tested it on some well-known benchmark data sets such as iRoads [15] and Caltech-256 [16] and xDNN outperforms the other methods including deep learning in terms of accuracy, time to train, moreover, offers a clearly explainable classifier. In fact, the result on the very hard Caltech-256 problem (which has 257 classes) represents a world record [1].

The remainder of this paper is organized as follows: The next section introduces the proposed explainable deep learning approach. The experimental data employed in the analysis and results are presented in the results section. Discussion is presented in the last section of this paper.

II. Explainable Deep Neural Network

A. Architecture and Training of the proposed xDNN

The proposed explainable deep neural network (xDNN) classifier is formed of several layers with a very clear semantic and functional meaning. In addition to the internal clarity and transparency it also offers a very clear from the user point of view set of prototype-based $IF...THEN$ rules. Prototypes are selected data samples (images) that the user can easily view, understand and appreciate the similarity to other validation images. xDNN offers a synergy between the statistical learning and reasoning bringing both together. In most of the other approaches there is a dichotomy and preference of one over the other. We advocate and demonstrate that both, learning and reasoning can work together in a synergy and produce very impressive results. Indeed, the proposed xDNN method outperforms all published results [15], [1], [17] in terms of accuracy. Moreover, in terms of time for training, computational simplicity, low power and energy required it is also far ahead. The proposed approach can be described as a feedforward neural network which has an incremental learning algorithm that autonomously self-develops and evolves its structure adding new prototypes to reflect the possibly changing (dynamically evolving) data pattern [10]. As shown in Figure 3, xDNN is composed of the following layers–

1) Features descriptor layer;
2) Density layer;
3) Typicality layer;
4) Prototypes layer;
5) *MegaClouds* layer;

1) **Features descriptor layer**: (Defines the data space)
The Feature Descriptor Layer is the first phase of the proposed xDNN method. This layer is in charge of extracting global features vector from the images. This first layer can be formed by more traditional 'handcrafted' methods such as GIST [19] or HoG [20]. Alternatively, it can be formed by the fully connected layer (FCL) of the pre-trained convolutional neural network approaches such as AlexNet [21], VGG–VD–16 [18], and Inception [22], residual neural networks such as Resnet [3] or Inception-Resnet [23], etc. Using pre-trained deep neural network approach allows automatic extraction of more abstract and discriminative high-level features. In this paper, pre-trained VGG–VD–16 DCNN is employed for feature extraction. According to [24], VGG–VD–16 has a simple structure and it can achieve a better performance in comparison with other pre-trained deep neural networks. The first fully connected layer from VGG–VD–16 provides a 1×4096 dimensional vector.
a) The values are then standardized using the following equation (1):

$$\widehat{x}_{i,j} = \frac{x_{i,j} - \mu(x_{i,j})}{\sigma(x_{i,j})} \tag{1}$$

where $\widehat{x}$ denotes a standardized features vector x of the image I (x are the values provided by the FCL), $i = 1, 2, ..., N$ denotes the time stamp or the ID of the image, $j = 1, 2, ..., n$ refers to the number of features of the given x in our case $n = 4096$.
b) The standardized values are normalised to bring them to the range [0;1]:

$$\bar{x}_{i,j} = \frac{\widehat{x}_{i,j} - \min\limits_{i}(\widehat{x}_{i,j})}{\max\limits_{i}(\widehat{x}_{i,j}) - \min\limits_{i}(\widehat{x}_{i,j})} \tag{2}$$

where $\bar{x}$ denotes the normalized value of the features vector. For clarity in the rest of the paper we will use x instead of $\bar{x}$.
Initialization:
Meta-parameters for the xDNN are initialized with the first observed data sample (image). The proposed algorithm works per class; therefore, all the calculations are done for each class separately.

$$P \leftarrow 1; \quad \mu \leftarrow x_i; \tag{3}$$

where μ denotes the global mean of data samples of the given class. P is the total number of the identified prototypes from the observed data samples (images).
Each class C is initialized by the first data sample of that class:

$$\begin{aligned} \mathrm{C}_1 \leftarrow x_1; \quad p_1 \leftarrow x_1; \\ Support_1 \leftarrow 1; \quad r_1 \leftarrow r^*; \quad \hat{I}_1 \leftarrow I_1 \end{aligned} \tag{4}$$

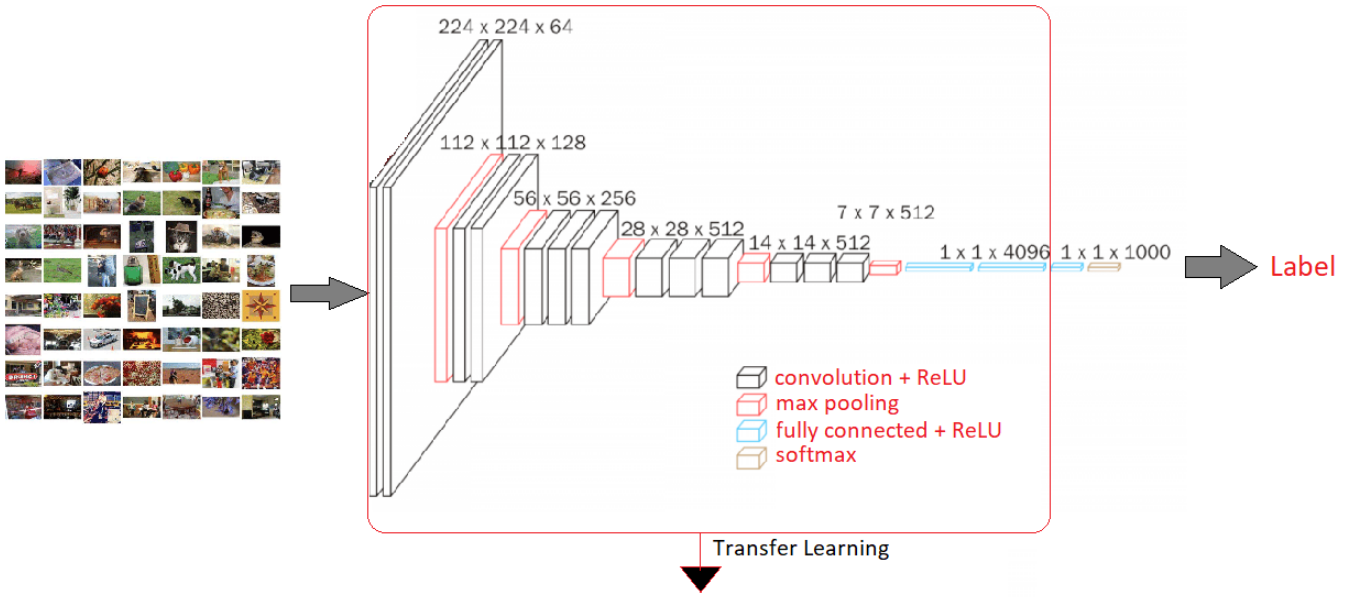

Fig. 2. Pre-training a traditional deep neural network (weights of the network are being optimized/trained). Using the transfer learning concept this architecture with the weights are used as feature extractor (the last fully connected layer is considered as a feature vector). Adapted from [18].

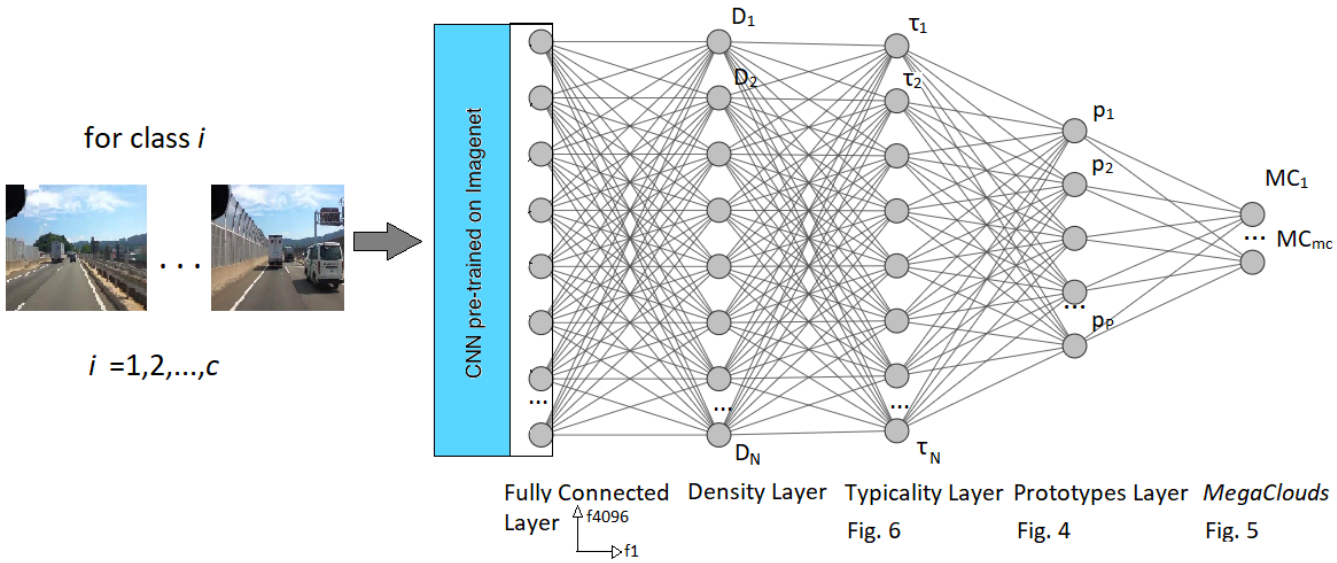

Fig. 3. xDNN training architecture (per class).

where, p_1 is the vector of features that describe the prototype $\hat{I}$ of the C_1; $\hat{I}$ is the identified prototype; $Support_1$ is the corresponding support (number of members) associated with this prototype; r_1 is the corresponding radius of the area of influence of C_1.
In this paper, we use $r^* = \sqrt{2 - 2cos(30^o)}$ same as [13]; the rationale is that two vectors for which the angle between them is less than $\pi/6$ or 30^o are pointing in close/similar directions d. That is, we consider that two feature vectors can be considered to be similar if the angle between them is smaller than 30 degrees. Note that r^* is data derived, not a problem- or user-specific parameter. In fact, it can be defined without *prior* knowledge of the specific problem or data through the following equation (5).

$$d(x_i, p_i) = \left\| \frac{x_i}{\|x_i\|} - \frac{p_i}{\|p_i\|} \right\|. \quad (5)$$

2) **Density layer**:
The density layer defines the mutual proximity of the images in the data space defined by the features from the previous layer. The data density, if use Euclidean form of distance, has a Cauchy form (6) [13]:

$$D(x_i) = \frac{1}{1 + \frac{||x_i - \mu_N||^2}{\sigma_N^2}}, \quad (6)$$

where D is the density, μ is the global mean, and σ is the variance. The reason it is Cauchy is not arbitrary [13]. It can be demonstrated theoretically that if Euclidean or Mahalanobis type of distances in the feature space are considered, the data density reduces to Cauchy type as referred in equation (6). Density can also be updated online [25]:

$$D(x_i) = \frac{1}{1 + ||x_i - \mu_i||^2 + \sum_i - ||\mu_i||^2}. \quad (7)$$

where μ_i and the scalar product, $\sum_i$ can be updated recursively as follows:

$$\mu_i = \frac{i-1}{i}\mu_{i-1} + \frac{1}{i}x_i, \quad (8)$$

$$\sum_i = \frac{i-1}{i}\sum_{i-1} + \frac{1}{i}||x_i||^2 \quad \sum_1 = ||x_1||^2. \quad (9)$$

Data samples (images) that are closer to the global mean have higher density values. Therefore, the value of the data density indicates how strongly a particular data sample is influenced by other data samples in the data space due to their mutual proximity.

3) **Typicality layer**:
Typicality is is an empirically derived form of probability distribution function (pdf). *Typicality* τ is given by the equation (10). The value of τ even at the point $x = p_i$ is much less than 1; the integral of $\int_{-\infty}^{\infty} \tau dx = 1$ [13].

$$\tau(x_i) = \frac{\sum_{i=1}^{c} Support_i D(x_i)}{\sum_{i=1}^{c} Support_i \int_{-\infty}^{\infty} D(x_i)dx} \quad (10)$$

4) **Prototypes layer**:
The prototypes identification layer is the core of the proposed xDNN classifier. This layer is responsible to provide the clearly explainable model. The xDNN classifier is free from *prior* assumptions about the data distribution type, as well as the random or deterministic nature of the data. In contrast, it extracts the actual distribution empirically form the data samples (images) bottom up [13]. The prototypes are independent from each other. Therefore, one can change the structure by adding a new prototype without influencing the other already existing prototypes. In other words, the proposed xDNN is highly parallelizable and suitable for evolving form of application where new prototypes may be added (if the data pattern requires this). The proposed xDNN method is trained per class forming a set of prototypes per class. Therefore, all the calculations are done for each class separately. Prototypes are the local peaks of the data density (and *typicality*) identified in the previous layers/ stages of the algorithm from the images of the corresponding class based on their feature vectors.

The prototypes can be used to form linguistic logical $IF...THEN$ rules of the following form:

R_c: IF $(I \sim \hat{I}_P)$ THEN $(class\ c)$
where $\sim$ stands for similarity, it also can be seen as a fuzzy degree of membership; p is the identified prototype; P is the number of identified prototypes; c is the class $c = 1, 2, ..., C$, I denotes an image.
One rule per prototype can be formed. All rules per class can be combined together using logical OR, also known as disjunction or S-norm:

R_c: IF $(I \sim \hat{I}_1)$ OR $(I \sim \hat{I}_2)$ OR ... OR $(I \sim \hat{I}_P)$ THEN $(class\ c)$

Figure 4 illustrates the area of influence of the identified prototypes. These areas around the identified prototypes are called *data clouds* [13]. Thus, each prototype defines a *data cloud*.

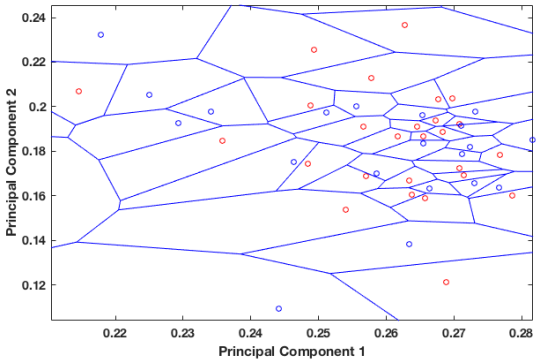

Fig. 4. Identified prototypes – Voronoi Tesselation.

We call all data points associated with a prototype *data clouds*, because their shape is not regular (e.g., hyper-spherical, hyper-ellipsoidal, etc.) and the prototype is not necessarily the statistical and geometric mean , but actual image [13]. The algorithm absorbs the new data samples one by one by assigning then to the nearest (in the feature space) prototype:

$$j^* = \underset{j=1,2,...,P}{\operatorname{argmin}} \left(||x_i - p_j||^2\right) \quad (11)$$

In case, the following condition [13] is met:

$$\begin{aligned} &IF\ (D(x_i) \geq \max_{j=1,2,...,P} D(p_j)) \\ &OR \quad (D(x_i) \leq \min_{j=1,2,...,P} D(p_j)) \quad (12) \\ &THEN\ (add\ a\ new\ data\ cloud\ (P \leftarrow P+1)) \end{aligned}$$

It means that x_i is out of the influence area of p_j. Therefore, the vector of features x_i becomes a new

prototype of a new *data cloud* with meta-parameters initialized by equation (13). Add a new *data cloud*:

$$P \leftarrow P+1; \quad \mathrm{C}_P \leftarrow x_i; p_P \leftarrow I_i; \quad Support_P \leftarrow 1; \\ r_P \leftarrow r_o; \hat{I}_P \leftarrow I_i; \tag{13}$$

Otherwise, *data cloud* parameters are updated online by equation (14). It has to be stressed that all calculations per *data cloud* are performed on the basis of data points associated with a certain *data cloud* only (i. e. locally, not globally, on the basis of all data points).

$$C_{j^*} \leftarrow C_{j^*}+1; \\ p_{j^*} \leftarrow \frac{Support_{j^*}}{Support_{j^*}+1} p_{j^*} + \frac{Support_{j^*}}{Support_{j^*}+1} x_i; \\ Support_{j^*} \leftarrow Support_{j^*}+1; \\ r^2_{j^*} \leftarrow \frac{r^2_{j^*} + (1-||p_{j^*}||^2)}{2}. \tag{14}$$

The xDNN learning procedure can be summarized by the following algorithm.

xDNN: Learning Procedure

1: Read the first feature vector sample x_i representing the image I_i of the class c;
2: Set $i \leftarrow 1; n \leftarrow 1; P_1 \leftarrow 1; p_1 \leftarrow x_i; \mu \leftarrow x_1; Support \leftarrow 1; r_1 \leftarrow r_0; \hat{I}_1 \leftarrow I_1$;
3: **FOR** i = 2, ...
4: Read x_i;
5: Calculate $D(x_i)$ and $D(p_j)$ $(j = 1, 2, ..., P)$ according to equation (9);
6: **IF** Equation (12) holds
7: Create rule according to Equation (13);
8: **ELSE**
9: Search for p_j according to Equation (11);
10: Update rule according to Equation (14);
11: **END**
12: **END**

5) ***MegaClouds* layer**:
In the *MegaClouds* layer the *clouds* formed by the prototypes in the previous layer are merged if the neighbouring prototypes have the same class label. In other words, they are merged if they belong to the same class. *MegaClouds* are used to facilitate the human interpretability. Figure 5 illustrates the formation of the *MegaClouds*.

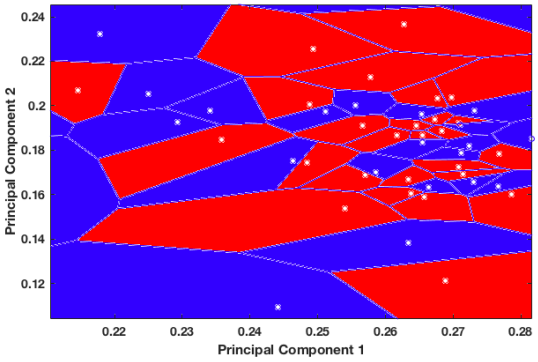

Fig. 5. *MegaClouds* – Voronoi Tesselation.

Rules in the *MegaClouds* layer have the following format:

R_c: IF $(x \sim MC_1)$ OR $(x \sim MC_2)$ OR ... OR $(x \sim MC_{mc})$ THEN $(class\ c)$

where MC are the *MegaClouds*, or the areas formed from the merging of the *clouds*, and mc is the number of identified *MegaClouds*. Multimodal *typicality*, τ, can also be used to illustrate the *MegaClouds* as illustrated by Figure 6.

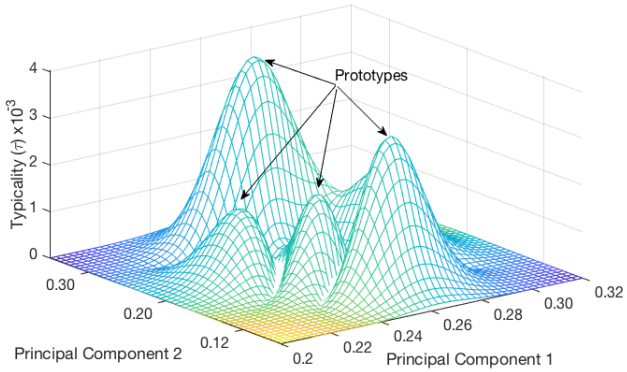

Fig. 6. *Typicality* for the iRoads dataset.

B. Architecture and Validation of the proposed xDNN

Architecture for the validation process of the proposed xDNN method is illustrated by Figure 7.

The validation process of xDNN is composed of the following layers:

1) Features descriptor layer;
2) Similarity layer (density);
3) Local decision-making.
4) Global decision-making.

Which is detailed described as following:

1) **Features descriptor layer**:
Similarly to the features descriptor layer described in the training process.
2) **Prototypes layer**:

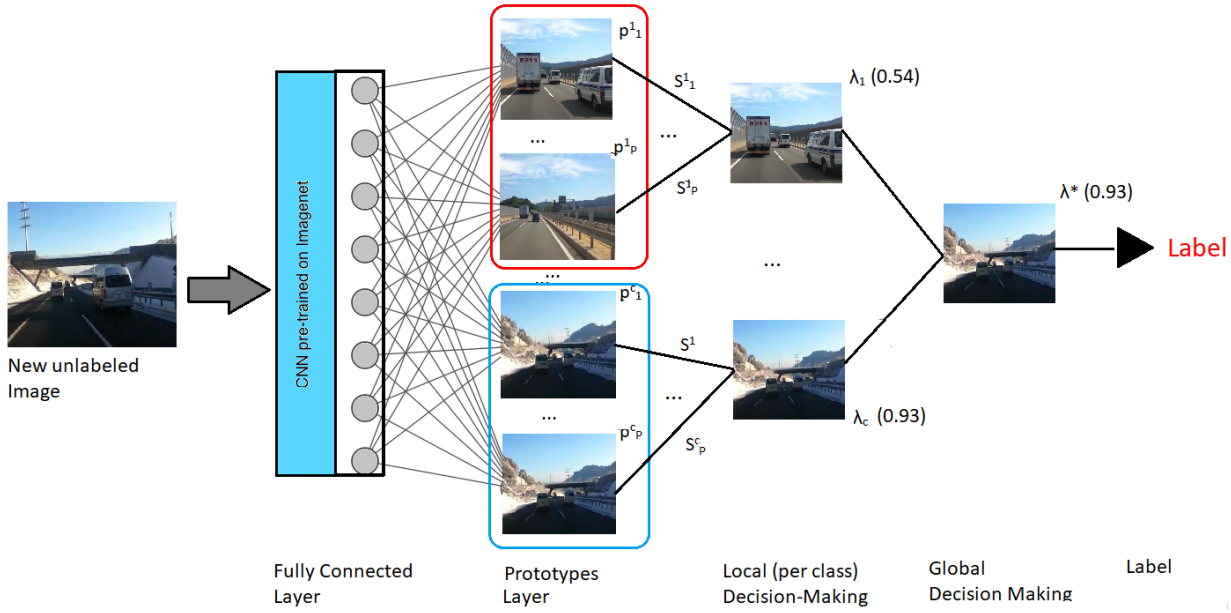

Fig. 7. Architecture for the validation process of the proposed xDNN.

In this layer the degrees of similarity to the nearest prototypes (per class) are extracted for each unlabeled (new/validation) data sample/image I_i defined as follows:

$$S(x, p_i) = \frac{1}{1 + \frac{(x-p_i)}{\sigma_i^2}}, \tag{15}$$

where S denotes the similarity degree.

3) **Local (per class) decision-making layer**:
Local (per class) decision-making is calculated based on the 'winner-takes-all' principle and can be obtained by:

$$\lambda_c = \underset{j=1,2,...,P}{max}(S_j), \tag{16}$$

4) **Global decision-making layer**: The global decision-making layer is in charge of forming the decision by assigning labels to the validation images based on the degree of similarity of the prototypes obtained by the prototype identification layer as illustrated by Figure 7 and determining the winning class.

$$\lambda_c^* = \underset{c=1,2,...,C}{max}(\lambda_c), \tag{17}$$

In order to determine the overall degree of satisfaction, the maximum of the local, per class winners is applied. The label is obtained by the following equation (18):

$$label = \underset{c=1,2,...,C}{argmax}\,(\lambda_c^*), \tag{18}$$

III. Experimental Data

We validated our proposed approach, xDNN using several complex, well-known image classification benchmark datasets (iRoads and Calltech-256).

A. iRoads dataset

The iROADS dataset [15] was considered in the analysis first. The dataset contains 4,656 image frames recorded from moving vehicles on a diverse set of road scenes, recorded in day, night, under various weather and lighting conditions, as described below:

- Daylight - 903 images
- Night - 1050 images
- Rainy day - 1049 images
- Rainy night - 431 images
- Snowy - 569 images
- Sun strokes - 307 images
- Tunnel - 347 images

B. Caltech-256

Caletch-256 has 30,607 images divided into 257 object categories (one of which is the background) [16].

C. Performance Evaluation

The performance of the classification methods is usually evaluated based on their accuracy index which is defined as follows:

$$ACC(\%) = \frac{TP + TN}{TP + FP + TN + FN}, \tag{19}$$

where TP, FP, TN, FN denote true and false, negative and positive, respectively.

All the experiments were conducted with MATLAB 2018a using a personal computer with a 1.8 GHz Intel Core i5 processor, 8-GB RAM, and MacOS operating system. The classification experiments were executed using 10-fold cross validation under the same ratio of training-to-testing (80% to 20%) sample sets.

IV. Results and Analysis

Computational simulations were performed to assess the accuracy of the proposed explainable deep learning method, xDNN against other state-of-the-art approaches.

A. *iRoads Dataset*

Table I shows that the proposed xDNN method provides the best result in terms of classification accuracy as well as time/complexity and simplicity of the model structure (number of parameters/prototypes). The number of model parameters for xDNN (and DRB) is, strictly speaking, zero, because the 2 parameters (mean, μ and standard deviation, σ) per prototype (*data cloud*) are derived from the data and are not algorithmic parameters or user-defined parameters. For kNN method one can argue that the number of parameters is the number of data samples, N. The proposed explainable DNN surpasses in terms of accuracy the state-of-the-art VGG–VD–16 algorithm which is a well-established convolutional deep neural network. Moreover, the proposed xDNN has at its top layer a set of a very small number of *MegaClouds* (27 or, on average, 4 *MegaClouds* per class) which makes it very easy to explain and visualize. For comparison, our earlier version of deep rule-based models, called DRB [17] also produced a high accuracy and was trained a bit faster, but ended up with 521 prototypes (on average 75 prototypes per class) [26]. With xDNN we do generate meaningful $IF...THEN$ rules as well as generate an analytical description of the *typicality* which is the empirically derived pdf in a closed form which lends itself for further analysis and processing.

TABLE I
Performance Comparasion: iRoads Dataset

Method	*Accuracy*	Time(s)	# Parameters
xDNN	**99.59%**	4.32	**27**
VGG–VD–16 [26]	99.51 %	836.28	Not reported
DRB [26]	99.02%	**2.95**	521
SVM [26]	94.17%	5.67	Not reported
KNN [26]	93.49%	4.43	4656
Naive Bayes [26]	88.35%	5.31	Not reported

MegaClouds generated by the proposed xDNN model can be visualized in terms of rules as illustrated by the Figure 8.

Voronoi tesselation can also be used to visualize the resulting *MegaClouds* as illustrated by Figure 9.

Typicality for classes 'night scene' and 'snow scene' are given by Figure 10.

Typicality can also be used for interpreatability and explainability as it is correspondent to the pdf. One can use the *typicality* to represent the likelihood that an image represents a specific type of driving conditions. For a given image a vector of features can be extracted, $x \in R^{4096}$ which can be standardized and normalized and used to demonstrate the likelihood of a certain type of driving condition as shown on Fig. 10.

B. *Caltech-256 Dataset*

Results for Caltech-256 are presented in Table II.

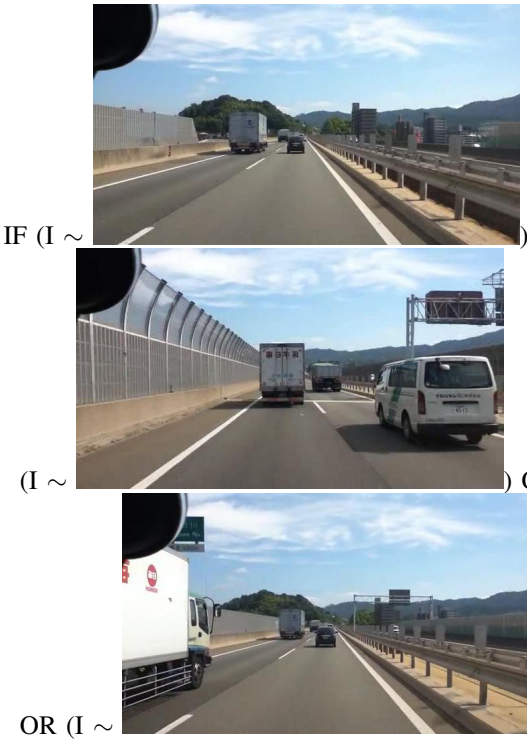

Fig. 8. xDNN rule generated for the 'Daylight scene'.

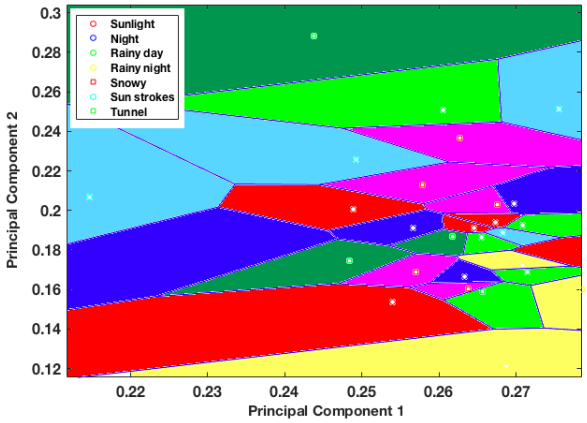

Fig. 9. *MegaClouds* for the iRoads dataset.

TABLE II
Performance Comparasion: Caltech-256 Dataset

Method	*Accuracy*
xDNN	**75.41**%
SVM(1) [27]	24.6 %
SVM(2) [27]	39.6%
SVM(3) [27]	46.0%
SVM(4) [27]	51.3%
SVM(5) [27]	65.6%
SVM(7) [27]	71.7%
Softmax(5)[27]	65.7%
Softmax(7) [27]	74.2%

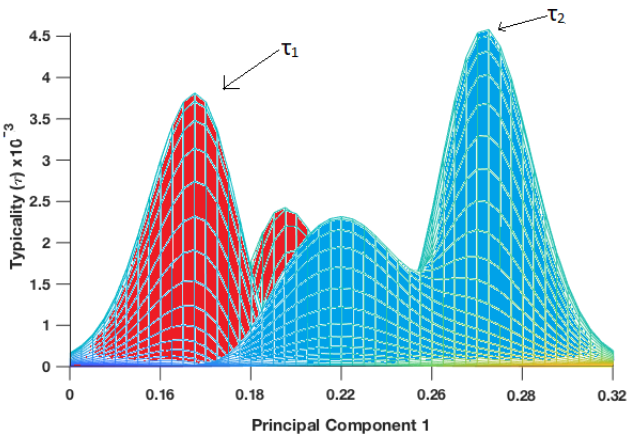

Fig. 10. *Typicality* for the iRoads dataset (2D), 2 classes, representing 'night scene' and 'snow scene'.

Results presented in Table II demonstrate that the proposed xDNN approach can obtain the best classification reported so far world wide for this complex problem, namely, 75.41%. The proposed approach did surpass all of the competitors, offering the highest accuracy, as well as, clearly explainable model. xDNN produced on average 3 *MegaClouds* per class (a total of 721) which are clearly explainable. Rules have the following format:

IF (x ∼ ) OR (x ∼ ) OR (x ∼ )
THEN 'CD'

Experiments have demonstrated that the proposed xDNN approach is able to produce highly accurate results surpassing state-of-the-art methods for different challenging datasets. Moreover, xDNN presents highly interpretable results that can be presented in the form of $IF...THEN$ logical rules, Voronoi tessellations, and/or *typicality* (empirically derived form of pdf) in a closed analytical form allowing further analysis. Because of its recursive, non-iterative and non-parametric form it allows computationally very efficient implementations to be realized.

V. Conclusion

In this paper we propose a new method, explainable deep neural network (xDNN), that is directly addressing the bottlenecks of the traditional deep learning approaches and offers a clearly explainable internal architecture that can outperform the existing methods. The proposed xDNN approach requires very little computational resources (no need for GPUs) and short training times (in the order of seconds). The proposed approach, xDNN is prototype-based. Prototypes are actual training data samples (images), which have local peaks of the empirical data distribution called *typicality* as well as of the data density. This generative model is identified in a closed form and equates to the pdf but is derived automatically and entirely from the training data with no user- or problem-specific thresholds, parameters or intervention. The proposed xDNN offers a new deep learning architecture that combines reasoning and learning in a synergy. It is non-iterative and non-parametric, which explains its efficiency in terms of time and computational resources. From the user perspective, the proposed approach is clearly understandable to human users. Results for some well-known benchmark data sets such as iRoads and Caltech-256 show that xDNN outperforms the other methods including state-of-the-art deep learning approaches (VGG–VD–16) in terms of accuracy, time to train and offers a clearly explainable classifier. In fact, the result on the very hard Caltech-256 problem (which has 257 classes) represents a world record [1][1]. Future research will concentrate on the development of a tree-based architecture, synthetic data generation, and local optimization in order to improve the proposed deep explainable approach.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[2] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[6] T. J. Sejnowski, *The deep learning revolution*. MIT Press, 2018.

[7] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[8] J. Hu, J. Lu, and Y.-P. Tan, "Deep transfer metric learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 325–333.

[9] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[10] E. Soares and P. Angelov, "Novelty detection and learning from extremely weak supervision," *arXiv preprint arXiv:1911.00616*, 2019.

[11] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[13] P. P. Angelov and X. Gu, *Empirical approach to machine learning*. Springer, 2019.

[14] P. P. Angelov, X. Gu, and J. C. Príncipe, "A generalized methodology for data analysis," *IEEE transactions on cybernetics*, vol. 48, no. 10, pp. 2981–2993, 2017.

[15] M. Rezaei and M. Terauchi, "Vehicle detection based on multi-feature clues and Dempster-Shafer fusion theory," in *Pacific-Rim Symposium on Image and Video Technology*. Springer, 2013, pp. 60–72.

[16] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[17] P. P. Angelov and X. Gu, "Deep rule-based classifier with human-level performance and characteristics," *Information Sciences*, vol. 463, pp. 196–213, 2018.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[1] https://martin-thoma.com/sota/

[19] B. Solmaz, S. M. Assari, and M. Shah, "Classifying web videos using a global video descriptor," *Machine vision and applications*, vol. 24, no. 7, pp. 1473–1485, 2013.

[20] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architectural study of hog feature extraction processor for real-time object detection," in *2012 IEEE Workshop on Signal Processing Systems*. IEEE, 2012, pp. 197–202.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[24] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1476–1481, 2016.

[25] P. Angelov, *Autonomous learning systems: from data streams to knowledge in real-time*. John Wiley & Sons, 2012.

[26] E. Soares, P. Angelov, B. Costa, and M. Castro, "Actively semi-supervised deep rule-based classifier applied to adverse driving scenarios," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[27] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.