Data Preprocessing Technique for Neural Networks Based on Image Represented by a Fuzzy Function

Petr Hurtik, Vojtech Molek, Jan Hula

***Abstract*—Although data preprocessing is a universal technique that can be widely used in neural networks, most research in this area is focused on designing new neural network architectures. In our work, we propose a preprocessing technique that enriches the original image data using local intensity information; this technique is motivated by human perception. To encode this information into an image, we introduce a new image structure named Image Represented by a Fuzzy Function. When using this structure, a crisp intensity value of each pixel is replaced by a fuzzy set given by a membership function constructed with the usage of extremal values from the particular neighborhood of that pixel. We describe this structure and its properties and propose a way in which it can be used as an input into existing neural networks without any modifications. Based on our benchmark consisting of three well-known datasets and five neural network architectures, we show that the proposed preprocessing can, in most cases, decrease classification error compared with a baseline and two other preprocessing methods. To support our claim, we have also selected several publicly available projects and tested the impact of the preprocessing with a positive result.**

***Index Terms*—Image Represented by Fuzzy Function, IRFF, image fuzzification, data preprocessing, neural network, image classification**

I. Introduction

THROUGH the years, fuzzy, in the sense of fuzzy modeling [1] has become a part of many applied computing fields such as image processing [2], time series analysis [3], and system controls [4] owing to its easy integration into existing systems [5]. Most of these cases involve a preprocessing phase by data fuzzification, application of fuzzy logic machinery and a postprocessing phase represented by data defuzzification and data interpretation [6]. In the field of Artificial Neural Networks (ANNs, or for simplicity only NNs), the justification for a fuzzification may not be so straightforward, especially if the input already lies in the interval [0, 1]. In that case, there is no reason to pretend that we are working with fuzzified data. To truly embrace the fuzzy approaches, the research has started to move in the direction of hybridization of NNs and fuzzy models called Fuzzy (Deep) Neural Network [7] or Neuro-Fuzzy Network [8], which adds layers for data fuzzification and for learning fuzzy rules [9]. Such models may help when working with data that are poorly defined because of their incompleteness, uncertainty or vagueness. The other approach, Fuzzy Neural Networks, as described in [10] or [11], is disregarded by the wider community owing to its complicated replication and a specific usage: it works well for predictions based on data dependencies (such as time series) but has limitations, for example, in image processing. For an exhaustive description of such networks, we refer to [12].

Instead of adding fuzzy layers, or modifying neural network architecture to work with fuzzy-based operations, we focus on data fuzzification as a general method of data preprocessing for NNs in this work. Let us recall that there are several ways in which a neural network classification error can be decreased: by collecting more data [13], by designing more suitable (usually more complex) architecture [14], [15] including an ensemble of several models [16], dealing with various activation functions [17] or optimization methods [18]. Furthermore, the classification error can also be decreased by suitable data preprocessing [19].

Currently, in computer vision, some progress has been made towards architecture modifications. This can be demonstrated on the MNIST dataset [20], which has been solved with an error rate of 1.6% by 2-layer NN [21], of 0.31% by 6-layer NN, of 0.23% by an ensemble of 35 5-layers NNs [22], and finally of 0.21% by 5 NNs with DropConnect [23]. Simultaneously, the architecture expansion increases the number of NN parameters, which (potentially) leads to a significant increase in training time. In contrast, preprocessing is a process that is executed once for the entire dataset (which can be then saved in the preprocessed form and used multiple times for training/testing phase) and therefore the training/testing time of NNs is not significantly affected. The common methods of preprocessing are mean normalization, standardization [24], data whitening [25], or principal component analysis for dimensionality reduction [26]. Even though data preprocessing techniques usually do not increase the neural network classification accuracy drastically, their benefits lie in their universal application for various neural network architectures.

Surprisingly, even though fuzzy methods are widespread within the field of pattern recognition [27], a preprocessing stage for NNs (without the necessity of its modification) based on fuzzy is very rare, if any. The only similar work we found [28] is fuzzy postprocessing implemented into a convolutional neural network. Unfortunately, this work is tightly restricted to work with MNIST dataset [20] and cannot be used for other datasets. Therefore, there is an open space for fuzzy-inspired data preprocessing. In our previous work [29], we have extensively studied the impact of standard vs alternative image representation method called Image Represented by a Fuzzy Function (IRFF, or fuzzy image) in terms of its sensitivity to various image distortions with the conclusion that the fuzzy image representation is more robust, which results

P. Hurtik, V. Molek and J. Hula were with Institute for Research and Applications of Fuzzy Modeling, Centre of Excellence IT4Innovations, University of Ostrava, 30. dubna 22, Ostrava, Czech Republic; e-mail: petr.hurtik@osu.cz

Manuscript received xyz, 2018; revised xyz, 2018.

in a better pattern recognition performance.

The goal of this study is to design a preprocessing method realized by a transformation of a standard image into Image Represented by a Fuzzy Function and examine its impact on the classification error of various neural networks.

The novelty of the study lies in the following aspects. First, we revised the IRFF description into a compact construction of four definitions and three conditions where we additionally extend the original approach into a general form that allows using various membership function shapes for image fuzzification; in the benchmark, we include three of them. Furthermore, we present a fast and effective IRFF Python-based implementation. To the best of our knowledge, supported by an intensive literature survey, we describe the very first fuzzy-based data preprocessing technique for neural networks, which does not require a modification of a neural network architecture. It is the first attempt to combine Image Represented by a Fuzzy Function with neural networks. The IRFF preprocessing enriches the original data instead of reducing them, which is quite rare in the field of NNs. The impact of the proposal is significant: the proposed preprocessing increased the baseline accuracy in 80% of the tested scenarios. The valuable knowledge for the fuzzy community is also the novel usage of two fuzzy approaches for NN preprocessing along with the finding that the transformation based on interval-valued fuzzy sets can also increase NN accuracy in several specific cases.

This paper is structured as follows. First, we will discuss the context of this study and its motivation in Section II. The formal basis of the IRFF is described in Section III and used in Section IV, where its integration into existing systems is examined. Section V contains the behavior analysis for various fuzzy set membership functions, setting of fuzzification procedure, and learning rates. Finally, the whole proposal is benchmarked in Section VI. Additionally, further comments, death-ends, and possible improvements are discussed in Section VII.

II. Paper context and motivation

When applying deep neural networks to computer vision tasks, it is usually recommended that an input to a network should be raw images (possibly with a certain normalization) without projecting it into a lower-dimensional feature space [30]. The rationale behind this advice is that such a projection may cause a loss of information that may be viable for the task at hand. It is believed that if these features are needed, a network would learn them, with enough examples, by itself. The same trend also occurs in speech recognition [31] where for a long time it seemed infeasible to achieve reasonable classification error without preprocessing the input signal by methods that reduce its dimensionality. However, recent research [32] showed that it is possible to train a neural network from the raw audio signal to recognize speech with a lower error rate than to train it on Mel-Frequency Cepstral Coefficients (MFCCs) which was the standard preprocessing step for speech processing [33]. In our work, we preprocess images in a way that will not remove any information but, on the contrary, will create enriched information that will be useful during training and testing phase.

In computer vision, the most popular [34] NN architecture is a Convolutional Neural Network (CNN). It consists of convolutional layers with learnable filters, nonlinear activation functions, and pooling layers that successively reduce the dimensionality in the spatial dimensions. The learnable filters of CNN can reconstruct many of the well-known preprocessing filters, e.g., Gaussian or Laplacian filter. Obviously, there are also types of preprocessing that cannot be represented by learnable filters, such as a median filter. Therefore, there are two possible reasons why one may want to apply certain preprocessing methods. First, preprocessing may save training time because it may convert images to a representation that would have to be found by optimization. Second, it could use operations that may not be representable as a convolution by a particular filter. In this study, we investigate a preprocessing method that could not be represented as a convolution by a particular filter in the standard setting of a convolutional network.

III. Image Represented by a Fuzzy Function

In this section, we will describe the formal background of the proposed fuzzification procedure. First, we consider a two-dimensional computer grayscale image f:

Definition 1: Let $f : D \to L$, be an image function where $D = \{1, \dots, W\} \times \{1, \dots, H\}$ is a discrete space where W and H denote width and height of the image and $L \subset \mathbb{R}$.

According to the definition, we are handling the image in a raster manner. However, human perception is based on a more complex mechanism, with the recognition of shapes and features taking into account visual saliency [35]. The visual saliency expresses what part of an image attracts human attention and how [36], [37] it can be modeled by using intensity, color or orientation contrast [38] with their possible joining into known shapes [39]. If we deal with various image processing tasks that reflect human perception, we hypothesize that algorithms that take into account the visual saliency of an image can solve the tasks with higher accuracy than algorithms based on a standard representation. Similar behavior can also be found in the case of convolutional NNs: first, basic features are extracted, which are then composed into shapes in higher layers [40]. If we deal with various image processing tasks which reflect human perception, we hypothesize that algorithms that consider visual saliency of an image can solve the tasks with higher accuracy than algorithms based on a standard representation.

One may ask whether it would be beneficial to represent an input to an NN by a structure that is closer to human perception and visual saliency. Our aim in this study is to answer this question. To enrich standard image information, we propose to use so-called Image Represented by a Fuzzy Function (IRFF, or fuzzy image for short). IRFF aims to bring an image representation closer to human perception. The IRFF was originally introduced in [41] and successfully applied in tasks of image enlargement [42], establishment of scale-spaces [43], enhancement of night movies [44] or road detection [41].

A. Fuzzification, defuzzification

First, let us recall the notation of universum of fuzzy sets and then start with the IRFF definition:

Definition 2: [41] *Let $\mathbb{F}(U)$ be a universum of fuzzy sets F over unit interval given as $\mathbb{F}(U) = \{F \mid F : U \to [0,1]\}$.*

As shown in the following definition, the difference between a standard and fuzzy image is in the range of values.

Definition 3: Image Represented by a Fuzzy Function f^F is a mapping given as $f^F : D \to \mathbb{F}(U)$, where $D = \{1,\dots,W\} \times \{1,\dots,H\}$ is a discrete space where W and H denote width and height of the image.

Note: the term "fuzzy function" can also be found in the literature as a special fuzzy relation [45] or as a mapping between fuzzy sets [46]. Nevertheless, because our notation fits better to the term "function evaluated by fuzzy sets", we will use the term fuzzy function in order to preserve the same terminology as is stated in previous papers.

Now, let us focus on a definition of particular fuzzy sets that will replace original crisp image intensities and thus encode local intensity contrast, which models visual saliency [38]. To capture the "locality", we need to establish δ-neighborhood around each pixel at first.

Definition 4: Let δ be a positive finite integer, and f be an image function with its domain D. Then, δ-neighborhood $\omega_{f,x,y}$ is a set containing spatial information around location (x,y) determined as $\omega_{f,x,y} = \{(x_i,y_i) \in D \mid |x - x_i| \leq \delta, |y - y_i| \leq \delta\}$.

Then, we need to extract intensity information from $\omega_{f,x,y}$ in order to capture the intensity contrast used for modeling the saliency.

Definition 5: Let $\omega_{f,x,y}$ be a spatial δ-neighborhood of a point (x,y) of an image function f with its domain D. Then, intensity δ-neighborhood $\Omega_{f,x,y}$ of such the point is set $\Omega_{f,x,y} = \{f(x_i,y_i) \mid (x_i,y_i) \in \omega_{f,x,y}\}$.

Now, let us encode $\Omega_{f,x,y}$ into a fuzzy set, which is identified by its membership function $f^F(x,y)$. Further, the membership degree $\gamma \in [0,1]$ expresses that the element $\beta \in [0,255]$ belongs into the fuzzy set in a degree γ, i.e., $f^F(x,y)(\beta) = \gamma$. For that purpose, let us define several conditions we do want to achieve:

C1: $f^F(x,y)(f(x,y)) = 1$,
C2: $\forall \beta \in \Omega_{f,x,y} : f^F(x,y)(\beta) > 0$,
C3: $f^F(x,y)(\beta) = 0$ iff $\beta \notin [\min(\Omega_{f,x,y}), \max(\Omega_{f,x,y})]$.

The aforementioned conditions ensure that pixel intensity $f(x,y)$ has maximal membership degree to its fuzzy set $f^F(x,y)$ (*C1*); it is natural as we define δ-neighborhood around pixel (x,y) and therefore $f(x,y)$ is known. Other pixel intensities around (x,y) in δ-neighborhood have a nonzero membership degree to $f^F(x,y)$ (*C2*). The pixel intensities that are not from $\Omega_{f,x,y}$ (not in interval $[\min(\Omega_{f,x,y}), \max(\Omega_{f,x,y})]$) have got zero membership degree to $f^F(x,y)$. Such behavior can partially mimic human visual perception and can even handle optical illusions - for example see [41].

Conditions *C1 – C3* allow us to use various definitions of membership functions - we can chose various shapes (sinusoidal, triangular, trapezoidal, ...) and define them using various intensity values from $\Omega_{f,x,y}$ such as minimum, maximum, quantiles, means, and probability of occurrence. However, it is obvious that all possible local intensity variance is in $\Omega_{f,x,y}$, so we can model absolute intensity contrast [47] $c \in [0,255]$ as

$$c(x,y) = \max(\Omega_{f,x,y}) - \min(\Omega_{f,x,y}). \tag{1}$$

Therefore, we will use values of $\max(\Omega_{f,x,y})$ and $\min(\Omega_{f,x,y})$ in order to define support of a fuzzy set and particular membership function. Further, considering differences between the two extremas ($\max(\Omega_{f,x,y})$ and $\min(\Omega_{f,x,y})$) and $f(x,y)$, we can also mimic human behavior called "centre surround mechanism" defined in [47] as "*centre-surround mechanism: These involve neurons that respond to image differences between a small central region and a broader concentric antagonistic surround region*".
For the following definition of a membership function, we will use the next substitutions:

$$\begin{aligned} f_L^F(x,y) &= \min(\Omega_{f,x,y})), \\ f_C^F(x,y) &= f(x,y), \\ f_R^F(x,y) &= \max(\Omega_{f,x,y})). \end{aligned} \tag{2}$$

Taking into account computational complexity, we will use triangular membership function given as

$$f^F(p)(\beta) = \begin{cases} \frac{\beta - f_L^F(p)}{f_C^F(p) - f_L^F(p) + \varepsilon} & \text{if } f_L^F(p) \leq \beta \leq f_C^F(p) \\ \frac{f_R^F(p) - \beta}{f_R^F(p) - f_C^F(p) + \varepsilon} & \text{if } f_C^F(p) < \beta \leq f_R^F(p) \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where $p = (x,y)$ is used in order to shorten the formula and ε is an extremely small positive integer. This proposed membership function is also a form of a fuzzy number. The full schema of the proposed fuzzification procedure is illustrated in Figure 1. The benefits of the triangular-based membership function lie in its easy manipulation. We can use known apparatus, namely, interval approach [1], i.e., we can identify our membership function only with the triplet $\big(f_L^F(x,y), f_C^F(x,y), f_R^F(x,y)\big)$. Computations with only these three points are extremely fast – see, e.g., [41].

To be complete, let us mention the inverse process to fuzzification called defuzzification. Concerning our application, it is given as follows (a more-general definition can be found in [1]):

Definition 6: Let F be non-empty fuzzy set and further $L \subset \mathbb{R}$. Then, defuzzification is a mapping $E : F \to L$.

Considering defuzzification methods [1] such as "First of Maxima", "Center of Maxima", or "Mean of Maxima", we can observe that for $f \mapsto f^F \mapsto f'$ holds $f \equiv f'$ if we do not realize a modification of f^F. The property is not preserved for, e.g., the "Center of Gravity" defuzzification method.

Note that the term *fuzzy image* or *image fuzzification* is not novel in the literature. The difference is that in the literature,

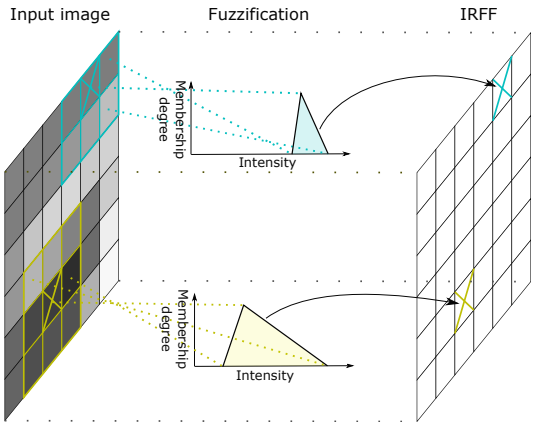

Fig. 1. Illustration of the proposed fuzzification schema.

a fuzzified image f^F is usually considered as $f^F : D \subset \mathbb{N}_0^2 \to L \subset \mathbb{R}$, where $f^F(x,y) \in [0,1]$ and the "fuzzification" procedure is realized as $f^F(x,y) = f(x,y)/\max(f)$ (or similar re-mappings), i.e., by replacing crisp values by fuzzy singletons [48]. It is obvious that such fuzzification allows using theoretical tools of fuzzy logic (for such applications see [49]). However, rescaling image intensities is not a proper method to fuzzify images, and it could be improved. Additionally, one may say that our approach is similar to Bustince's [50]. The difference is that Bustince models an interval and not a membership function – he is using only the two extreme values, not the central one. Moreover, his fuzzification procedure differs, and it is not motivated by human perception.

B. Additional manipulations

In addition to handling the uncertainty of a pixel intensity value, given by its neighborhood, the proposed structure includes one additional information: gradient magnitude. In computer graphics, a gradient magnitude can be computed with a first-derivative gradient operator, which results in a computation of intensity differences in a fixed-size window [51]. In the case of IRFF, all possible intensity differences in $\Omega_{f,x,y}$ clearly lie in between extremes, so the gradient magnitude ∇f^F in a point (x,y) can be approximated as

$$\nabla f^F(x,y) = f_R^F(x,y) - f_L^F(x,y). \tag{4}$$

Because $f_R^F(x,y) \geq f_L^F(x,y)$, the gradient magnitude is always positive. Such gradient magnitude approximation is also independent of a position of maxima and minima. That property may be useful for a convolutional neural network that learns directional gradient operators in its first layer by itself [40]. With IRFF as an input, an NN may (theoretically) learn gradient magnitude using the formula described above instead of establishing the directional gradient operators. For more details about gradient magnitude approximation and edge detection with the usage of IRFF, we refer to [52], where three methods for edge detection based on the IRFF are presented.

Note that a visualization of the IRFF structure is not so straightforward as in the case of a standard image, because instead of each pixel crisp intensity value, we have a membership function. Generally, there are two ways in which an IRFF can be visualized: by the usage of $\left(f_L^F(x,y), f_C^F(x,y), f_R^F(x,y)\right)$, or on the basis of membership degree – for an illustration see Figure 2. The former is visualized by three images (or single, three-channeled image) directly – for the illustration see Figure 7. The latter creates an image using the following definition:

Definition 7: *Let* $v_{i,f^F} : \mathbb{N}^2 \to [0,1]$ *be a function visualizing* f^F *for an intensity* $i = \{0,\ldots,255\}$ *as* $v_{i,f^F}(x,y) = f^F(x,y)(i)$.

Fig. 2. Visualization of IRFF on the princlipl of membership degree. From left: original image; $i = 10$; $i = 100$; $i = 200$; $i = 250$. Note that while the first image visualization includes intensities, the IRFF visualizations include membership degrees.

IV. Integration of Images Represented by a Fuzzy Function into Neural Networks

In this section, we will describe how the proposed structure (IRFF) is coupled with neural networks. Generally, this coupling can be realized in three different ways:

- *M1* Modifying all arithmetic operations in a neural network and working with fuzzy numbers in the sense of membership functions.
- *M2* Working with the fuzzy numbers as with intervals, i.e., processing the triplets with three separated networks and representing them as membership function in the last layer shared by all three networks.
- *M3* Using three/nine[1] channels of an image to store the triplets. This approach does not require the modification of the network in any way as the input data are still in the form of multichannel images.

Let us explore the three options in more detail. Conceptually, a modification of the entire network with the aim to work with a membership function as is described in [53], i.e., the realization of approach *M1*, is the cleanest way, and it opens a lot of space for future theoretical development. Additionally, in the literature, we can find attempts that fit the approach M1, such as multi-valued neurons [54] or neo-fuzzy neurons [55]. As these attempts have to use specialized neural network architectures, their usage is not general enough – they create a brand new solution and cannot extend existing ones. We omit this option, as the purpose of the study is to create an easily applicable extension for existing neural network architectures.

With the separate processing of a triplet representing membership function, which is the point of the option *M2*, we are getting further from a theoretical justification. Additionally, it significantly increases the complexity of the network (because all layers have to be tripled) and does not make applicability easier, because such integration needs a modification of whole

[1]Nine channels in case of an RGB image input data.

neural network architecture and extensive modification of an output layer. Taking these considerations into account, we omit *M2* option as well.

Finally, option *M3* resembles the formal background only remotely – using the triplets representing membership functions to create new image channels means that they will be processed by a neural network regularly. We cannot expect the preservation of formally correct computations with the triplets in an interval-based manner, so only original motivation remains. However, the integration of *M3* can be realized by just several new lines of source code (see Figure 4) without any further modification of existing neural network architectures. It means that the proposed approach is very easily applicable and may become a standard preprocessing technique helping researchers to improve the classification error of their designed NNs. The visualization of the proposed integration scheme is shown in Figure 3.

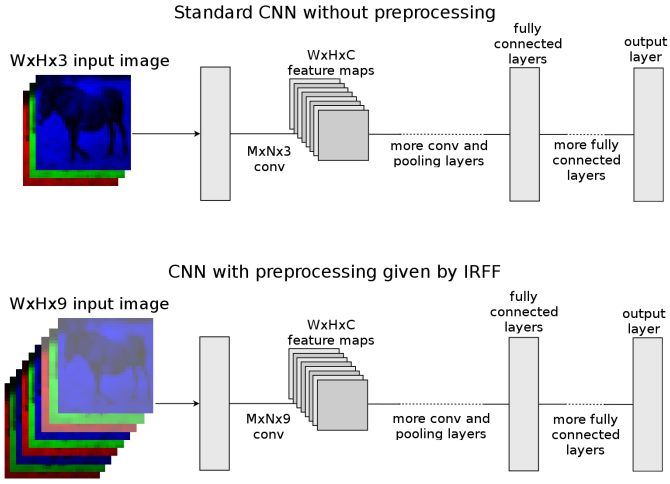

Fig. 3. A diagram that shows the difference between an RGB image and IRFF as an input to a neural network. Notice that the only difference is the depth of the first layer convolution filter, which is determined automatically using the shape of the input data.

In the proposed scheme illustrated in Figure 4, there is only one variable parameter: δ. Parameter δ affects the size of ω and is parallel to selecting a convolution filter window size. However, there is no exact definition of how to determine the window size, and therefore the process of choosing the right size is based on intuition, empirical evaluation, or experiments [56]. In general, δ depends on an image resolution and the size of (relevant) details. For example, considering a digit recognition task where the width of strokes is approximately 6px, the reasonable value of δ is ≤ 2; otherwise, the stokes will vanish. The impact of selecting various δ values for a particular task is examined in Section V.

Although the computation time of the preprocessing is not one of the possible bottlenecks, given that it is realized only once, it should not be unreasonably high. We supplement a simple Python implementation using *SciPy*[2] library; the snippet is shown in Figure 4. The preprocessing takes $\approx 43/46/47\mu s$ for $\delta = 2/5/8$ per a standard $28\times$ 28px image. The runtime was measured on the desktop processor - Intel Core i7-7700K, running on a single thread.

```
import numpy as np
from scipy import ndimage as nd
s = input.shape
temp  = np.zeros([s[0], s[1], s[2], 3])
for i in range(0, s[0]):
    temp[i,...,0] = nd.minimum_filter(input[i], size=delta)
    temp[i,...,1] = input[i]
    temp[i,...,2] = nd.maximum_filter(input[i], size=delta)
input = temp
```

Fig. 4. The reference implementation of the proposed preprocessing phase for a grayscale input image using Python language and SciPy library.

V. Analysis of the Behavior of the Proposal

This section focuses on an investigation of the proposed fuzzification scheme behavior in an image processing task concerning the following aspects:

- classification error improvement,
- change of size of a neural network,
- comparison of learning rates,
- impact of various δ values.

We have selected the Street View House Number (SVHN) recognition task [57], where we have taken a published solution available online[3] as our NN to be evaluated. The author of the solution preprocessed data by cropping images (to 32×32) and transforming them into grayscale. Each image displays one to five numbers. An example of images from the dataset is shown in Figure 5. The author's dataset includes 19000 images for training, 6500 images for validation and 6500 images for testing. The neural network architecture consists of Batch normalization [58], Convolution 2D (32,3), MaxPool (2), Convolution 2D (32, 3), MaxPool (2), Dropout (0.25), Convolution 2D (64, 3), Convolution 2D (64,3), Dropout (0.25), Convolution 2D (196, 3), Dropout (0.25), Dense (512), Flatten, and five Dense output layers.

Fig. 5. An example of the images from the dataset with the following labels (from left): 2; 203; 472; 2506; 97.

The original model has 307163 parameters in total. We ran the training and testing phases without any modifications to reference implementation, i.e., for 75 epochs with the average classification error of 8.90%. Afterward, we transformed the network input data to IRFF as is described in Subsection IV. With this input data, the network has 307739 parameters, i.e., the number of parameters has increased only by 0.18%. Simultaneously, the error of the network has decreased to 8.32%. The proposed preprocessing phase increased the number of the neural network parameters only by 0.18% and reduced the relative classification error by 6.97%.

[2]https://www.scipy.org/

[3]https://www.kaggle.com/olgabelitskaya/svhn-digit-recognition

To verify the obtained error rates, we joined the testing and training parts of the dataset into a single piece and ran k-fold cross-validation with $k = 10$. We received the classification error rate of 9.17 % (± 0.3) for the baseline and error rate of 8.72 % (± 0.3) for the baseline with IRFF preprocessing. Note that a neural network error rate varies for different training/testing phases as it depends on initial weights assignment, which is a random process. Therefore, in future experiments, we will use the average classification error rate calculated from multiple training/testing runs for the datasets as it is (delimited into train/test data) instead of the average error rate taken from the k-fold cross-validation.

During the experiment, we have also monitored the validation classification error – see Figure 6. According to the paired t-test value of $3.97 \cdot 10^{-5}$, we reject the hypothesis that the validation errors are statistically identical. By analyzing the differences, we have observed that for NN with IRFF, the validation error for the same epoch (Figure 6) is slightly lower than the validation error of the baseline NN (on average by 0.5%). In other words, the model with three times more information (with IRFF) in its input will converge as fast as or slightly faster as the baseline NN. However, such behavior is presented only in the case of the first ten epochs; then the classification errors tend to be statistically identical.

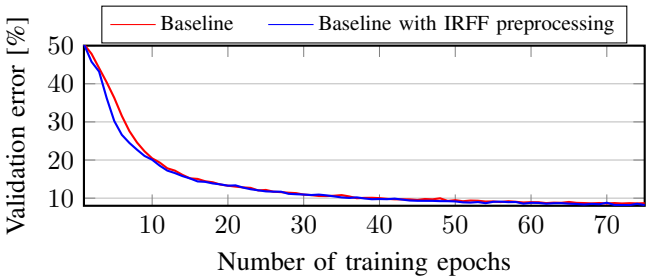

Fig. 6. Value of the validation classification error with respect to the number of training epochs.

The last aspect we want to examine is the impact of various δ values. Based on our implementation in Figure 4, we have identified δ by the parameter named "size". For our tests, we have used the IRFF preprocessing and trained the network for filter sizes of $1, \ldots, 6$. The visualization of preprocessed images for different filter sizes is shown in Figure 7. Note that the size equal to 1 is a special case when all the three input channels are equal to an original image. By testing this setting, we do want to show that merely by replicating an original image into three input channels we cannot decrease classification error. The size equal to six is quite an extreme setting for such small images so that the images will be strongly distorted. The measured errors are charted in Figure 9, from which it is clearly visible that IRFF created using a filter size equal to two has the lowest error and further that the error is much smaller than for the filter size equal to one. It means that replicating input image into three identical channels is not a good preprocessing technique.

As we stated in Section III-A, according to the conditions, there are several ways in which a fuzzy set membership function can be constructed. The first one is a triangular

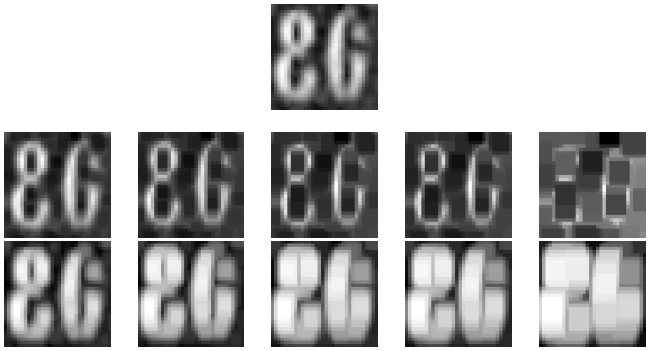

Fig. 7. Illustration of the impact of various δ values on an input image. Top row: original image. Middle row: f^F_{min} channel. Bottom row: f^F_{max} channel. The order of the images in the middle and the bottom rows goes from left to right for filter size $2, \ldots, 6$.

shape that bounds the support of a fuzzy set by minimum and maximum intensity values from a spatial neighborhood of a point. Here, we will name it *Shape 1*. Further, let us consider a shape which bounds fuzzy set support by percentiles of 0.25 and 0.75 instead of the extreme values – it will be named *Shape 2*. To be complete, we will also consider a partially linear shape bounded by the extremes and having a membership degree 0.5 for percentiles of 0.25 and 0.75 – it will be named *Shape 3*. These three cases are illustrated in Figure 8.

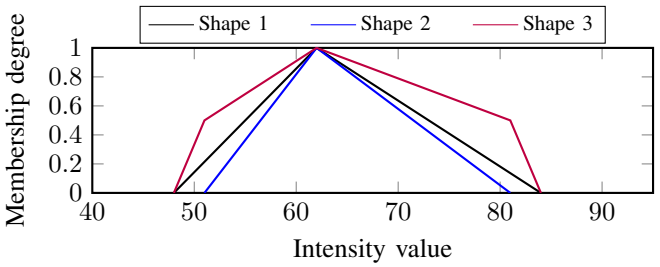

Fig. 8. The Illustration shows the three various fuzzy set membership functions for the same point in an image.

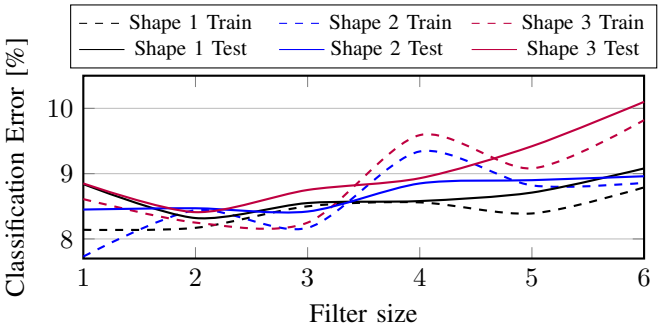

Fig. 9. The dependency of the chosen fuzzy set membership function and filter size used for constructing fuzzy sets on training and testing error.

The three types of a fuzzy set membership function are tested for the various settings of δ. As the best setting has been obtained for Shape 1 (triangular membership function that is given by two extremes) and $\delta = 2$, we are going to use this setting in the experiments realized in Section VI.

VI. Testing the proposal on various databases and NN architectures

To test the impact of the fuzzy image preprocessing on a neural network classification error (which is the inverse value of the classification accuracy), we have selected three well-known datasets: MNIST, Fashion MNIST, and CIFAR-10. MNIST is one of the most common datasets used for testing machine learning algorithms in computer vision. It consists of 60000 training and 10000 testing grayscale images with a resolution of 28×28px, capturing numbers 0-9. Fashion-MNIST is a dataset with the same number of images and resolution as MNIST, but it captures ten classes of fashion goods instead of the numbers. CIFAR-10 dataset proposes 50000 training and 10000 testing RGB images with 32×32px resolution, capturing 10 classes of various objects (horses, frogs, planes, cars…).

In the experiments, we use five different neural networks. Their detailed architectures are described in Table I. The architectures have been selected in order to cover simple (*S1*, *S2*, and *S5*), medium (*S3*) and complex (*S4*) architectures. Note, *S1* is represented by dense (a synonym of fully connected) input and output layers without any hidden layer(s). *S2* uses a single convolutional and single hidden dense layer. *S3* is a typical convolutional neural network [59] with seven layers. *S1*-*S3* include aslo dropout layers [60] to prevent overfitting and *S2*-*S3* include pooling layers [61] in order to increase the networks robustness to rigid transformations. *S4* is a special type of a neural network called capsule neural network [62], which is a representative of current state-of-the-art neural network architectures that should be able, besides recognizing objects, to recognize also their poses [63]. The last one, *S5* is a special form of NN called autoencoder that utilizes unsupervised learning.

Note that our approach is the first universal fuzzy-based preprocessing for neural networks. However, there are additional ways in which images can be represented so that a particular transformation method can be used as a preprocessing too. We selected two of them in order to compare them with our proposal. The first one, interval-valued fuzzy sets (IVFS) proposed by Bustince et al. [50] considers spatial neighborhoods of all points in an image. It uses proper t-norms and t-conorms to replace each intensity value in the image by an interval. Unlike our approach, the Bustince's one is not motivated by visual saliency and, moreover, does not preserve the original intensity. The second approach we use for the comparison is the well-known Fuzzy C-means (FCM) [64], which we use to replace pixel intensity using the information of a cluster it belongs to.

The benchmarking process is as follows: a particular network is trained on training data for 20 epochs and then tested on testing data (different from the training data). The absolute error difference expresses the difference between NN with and without IRFF preprocessing. A positive number means that the NN with IRFF is better, and vice-versa. The process of training and testing is repeated ten times, and the mean value of the ten testing errors is taken. The classification error is dependent on the training phase, which uses random initialization so that it can be trained more/less precisely depending on the initialization. Therefore, the results based on one random initialization may misinterpret the ranking of tested models.

The results for the three datasets and five architectures are listed in Tables II-IV. In the case of MNIST, the proposal reached the lowest error for all tested architectures among all algorithms, and it decreased the absolute error by 0.22%. Considering Fashion-MNIST, the proposal reached the smallest error for the three of five test cases – it has been beaten by the baseline for complex convolutional architecture and by IVFS for the autoencoder architecture. In summary, the error has been decreased by 0.3%. Finally, in the case of CIFAR-10, the proposal reached the lowest error for three of five test cases – the baseline has beaten it for the dense and the complex convolutional architectures, and it has been able to decrease the absolute error by 0.37%.

TABLE I
Used architectures

ID	Description
S1	Flatten → Dense (128) → Dropout (0.2) → Dense (output)
S2	Convolution 2D (32,2) → MaxPool (2) → Dropout (0.2) → Flatten → Dense (128) → Dense (output)
S3	Convolution 2D (128,3) → MaxPool (2) → Convolution 2D (128,3) → MaxPool (2) → Convolution 2D (128,3) → Convolution 2D (128,3) → MaxPool (2) → Convolution 2D (128,3) → Convolution 2D (128,1) → Convolution 2D (128,1) → MaxPool (2) → Flatten → Dense (output)
S4	Convolution 2D (256,9) → PrimaryCap (8,32,9) → Capsule (16) → Length (output)
S5	Encoder: Convolution 2D (8,3) → MaxPool (2) → Convolution 2D (4,3) → Convolution 2D (2,3) → MaxPool (7) Decoder: Upsampling 2D (7) → Convolution 2D (2,3) → Convolution 2D (4,3) → Upsampling 2D (2) → Convolution 2D (1,3) Classifier: Flatten → Dense (728) → Dropout (0.2) → Dense (10)

TABLE II
MNIST Classification errors

	Preprocessing [%]			
Network architecture	**None**	**IRFF**	**IVFS**	**FCM**
Dense	1.95	**1.93**	2.04	2.31
Simple convolutional	1.36	**1.26**	1.43	1.70
Complex convolutional	0.82	**0.67**	0.83	0.93
Capsule	0.50	**0.47**	0.49	0.51
Autoencoder	6.07	**5.26**	6.50	6.41

TABLE III
Fashion MNIST Classification errors

	Preprocessing [%]			
Network architecture	**None**	**IRFF**	**IVFS**	**FCM**
Dense	11.36	**10.69**	11.56	11.69
Simple convolutional	8.65	**8.41**	8.83	9.42
Complex convolutional	**8.28**	8.39	8.55	9.12
Capsule	9.00	**8.47**	9.49	12.51
Autoencoder	17.48	17.29	**17.05**	18.93

TABLE IV
CIFAR-10 CLASSIFICATION ERRORS

Network architecture	Preprocessing [%] None	IRFF	IVFS	FCM
Dense	**55.70**	56.64	56.67	61.64
Simple convolutional	37.99	**37.62**	38.85	45.30
Complex convolutional	**22.38**	24.24	28.95	35.51
Capsule	34.41	**30.52**	36.70	43.04
Autoencoder	56.24	**55.83**	55.88	61.78

VII. DISCUSSION

An important fact about our proposed preprocessing method is its compatibility with the existing methods. With the linear mappings, such as normalizing data into $[0, 1]$, the IRFF approach is fully functional. In fact, networks *S1*-*S3* and *S5* are used with the normalization; without it, the used optimizer (Adam [65]) does not work adequately and it is necessary to tune the learning rate. The additional preprocessings have been evaluated experimentally using Keras *ImageDataGenerator*[4] (in the form of image augmentation) for the combination of MNIST with *S1*. Usage of featurewise standardization [66] increased the classification error of both variations (with and without IRFF), and they may be considered as almost equal, 1.97% vs. 1.99%. With ZCA Whitening [67], both variations have started to overfit the data, while IRFF reached a slightly lower classification error, 4.89% vs. 4.67%. Finally, with rotations and shifts in x and y axis data, the error of both settings have again increased slightly, but IRFF reached a fractionally better result, 2.27% vs. 2.12%. The important fact is that the coupling of IRFF with other tested preprocessing methods does not significantly increase classification error. We can carefully conclude the experiment with the claim that IRFF preprocessing can be used together with the existing ones.

We have also tested our preprocessing on additional existing projects. From Github, we downloaded project[5] for CIFAR-10 classification, turned off data augmentation and used IRFF instead. With the IRFF preprocessing, we have achieved improvement in the error relative to the original model by 7.53%. Because of the lack of competitive results for CIFAR-10 in our benchmark, we used source code from Kaggle[6]. We trained the model for 125 epochs, with and without IRFF. The resulting model with IRFF achieved improvement in the error relative to the model without IRFF by 5.32%.

Our method also has certain drawbacks. The first one is that according to our benchmarked results presented in Tables II and IV, it cannot be guaranteed that the application of the preprocessing will decrease a network classification error – a user has to validate it on his particular task. The second drawback is that there does not exist an exact formula for setting-up filter size used for constructing IRFF, and therefore a user has to use his expert knowledge. The last possible issue is that our approach is motivated by visual perception and tested on image data. Consequently, we cannot make any conclusions about other domains such as, e.g., speech recognition. The answer to the question of how such data can be handled can be addressed in possible future work.

In our benchmark, the realization of all experiments took us approximately 15 days of computation and, unfortunately, handling more interesting datasets such as YFCC100M [68] is impossible owing to our current hardware capabilities. Based on that, this task remains open for other researchers as future work.

We also wish to stress that the proposed technique has certain limitations. The first one is that it is focused only on the spatial domain and not on the frequency domain. The second one is a data visualization – the images visualized by using multiple channels are not so descriptive as in the case of standard images. Finally, the preprocessing increases storage requirements – it needs three times more space for storing the preprocessed data, which can be a critical limitation for huge datasets.

VIII. CONCLUSION

Our primary focus in this work was on the problem of increasing neural network accuracy (decreasing classification error), and therefore we have investigated how it can be affected by a data preprocessing approach. Preprocessing is a powerful tool as models require little to no modifications of their architecture and may be applied to the most types of spatially dependent data. To design an image data preprocessing, we were motivated by literature dealing with a human perception, which mention that one of the factors that perception is based on is visual saliency, which can be modeled on the basis of intensity contrast. To mimic the visual saliency and to encode the intensity contrast, we proposed machinery that replaces crisp intensity values in an image with fuzzy sets identified by its membership functions. The structure is called "Image Represented by a Fuzzy Function". In contrast to our preliminary work, we proposed an extended definition of the structure allowing the use of various membership functions of fuzzy sets; in the experimental part of the study, we addressed three possible membership function shapes.

We presented two methods to visualize such structure using membership degrees or using a three-channeled image, where each channel represent one of the three crucial triangular membership function points – minimum, center and maximum. According to our knowledge, a fuzzy-inspired data preprocessing applicable without the necessity of a neural network architecture modification has not been published yet. We, therefore, consider our proposed preprocessing method as a novel contribution.

In the practical part of the paper, we discussed three ways in which the proposed data structure (Image Represented by a Fuzzy Function) can be integrated into neural networks. We have carefully investigated the option where IRFF is a multichannel image that serves as an input to a network. Such input is fully compatible with existing neural networks; therefore, no modification of their architecture is necessary, and our preprocessing can be used directly.

To show the impact of the preprocessing, we selected three different datasets – MNIST, Fashion-MNIST, and CIFAR-10.

[4]https://keras.io/preprocessing/image/
[5]https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py
[6]https://www.kaggle.com/c/cifar-10/discussion/40237

We tested them using five different neural network architectures with and without our preprocessing. We decided to check the behavior of the preprocessing on a complex problem – Street View House Number recognition task – solved by a neural network designed especially for the task. The proposed preprocessing was able to decrease its relative error by 6.97%, while it increased the number of trainable parameters only by 0.18%. We experimentally verified that the learning speed of a network with the preprocessing is not worse than the learning speed of a network without preprocessing. In the extensive benchmark, we compared the proposal with a baseline and two additional fuzzy-based methods that we modified to serve as preprocessing steps. We observed that the IRFF preprocessing could decrease a classification error of a neural network for most of the test scenarios. Further, in the discussion section, we briefly presented an application of the proposed preprocessing with various neural networks, where we also obtained positive results. Finally, we also tested a combination of the proposed preprocessing with the existing ones – we experimentally verified that they could be combined.

Because the study is focused on image processing, future work lies in integration and analysis of various data types, used by NN, such as time series prediction and speech recognition. Additionally, it would be valuable to code the proposed preprocessing into Keras framework so that it can be used as a separate neural network layer.

ACKNOWLEDGMENT

The work was supported from ERDF/ESF "Centre for the development of Artificial Intelligence Methods for the Automotive Industry of the region" (No. CZ.02.1.01/0.0/0.0/17_049/0008414)

For more supplementary materials and overview of our lab work see http://graphicwg.irafm.osu.cz/storage/pr/links.html.

REFERENCES

[1] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical principles of fuzzy logic*. Springer Science & Business Media, 2012, vol. 517.

[2] E. Kerre and M. Nachtegael, *Fuzzy techniques in image processing*. Physica, 2013, vol. 52.

[3] V. Novák, "Linguistic characterization of time series," *Fuzzy Sets and Systems*, vol. 285, pp. 52–72, 2016.

[4] C. W. De Silva, *Intelligent control: fuzzy logic applications*. CRC press, 2018.

[5] K. Metaxiotis, J. Psarras, and E. Samouilidis, "Integrating fuzzy logic into decision suppport systems: current research and future prospects," *Information management & computer security*, vol. 11, no. 2, pp. 53–59, 2003.

[6] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[7] S. Rajurkar and N. K. Verma, "Developing deep fuzzy network with takagi sugeno fuzzy inference system," in *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[8] L. Fortuna, G. Rizzotto, M. Lavorgna, G. Nunnari, M. G. Xibilia, and R. Caponetto, "Neuro-fuzzy networks," in *Soft Computing*. Springer, 2001, pp. 169–178.

[9] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 1006–1012, 2017.

[10] L.-Y. Shyu, Y.-H. Wu, and W. Hu, "Using wavelet transform and fuzzy neural network for vpc detection from the holter ecg," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1269–1273, 2004.

[11] P. Liu, "Representation of digital image by fuzzy neural network," *Fuzzy Sets and Systems*, vol. 130, no. 1, pp. 109–123, 2002.

[12] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, "Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence," 1997.

[13] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[14] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*. IEEE, 2015, pp. 730–734.

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

[16] C.-Y. Chiu, B. Verma, and M. Li, "Impact of variability in data on accuracy and diversity of neural network based ensemble classifiers," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–5.

[17] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[18] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 265–272.

[19] L. Yu, S. Wang, and K. K. Lai, "An integrated data preparation scheme for neural network data analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, pp. 217–230, 2006.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proceedings of the Seventh International Conference on Document Analysis and Recognition*. IEEE, 2003, p. 958.

[22] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3642–3649.

[23] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, 2013, pp. 1058–1066.

[24] K. K. Pal and K. Sudeep, "Preprocessing for image classification by convolutional neural networks," in *Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference on*. IEEE, 2016, pp. 1778–1781.

[25] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[26] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[27] Z. Chi, H. Yan, and T. Pham, *Fuzzy algorithms: with applications to image processing and pattern recognition*. World Scientific, 1996, vol. 10.

[28] E. Popko and I. Weinstein, "Fuzzy logic module of convolutional neural network for handwritten digits recognition," in *Journal of Physics: Conference Series*, vol. 738, no. 1. IOP Publishing, 2016, p. 012123.

[29] P. Hurtik, N. Madrid, and M. Dyba, "Sensitivity analysis for image represented by fuzzy function," *Soft Computing*, pp. 1–13, 2018.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[31] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[32] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 421–425.

[33] J. Benesty, M. M. Sondhi, and Y. Huang, *Springer handbook of speech processing*. springer, 2007.

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

[35] J. M. Henderson, J. R. Brockmole, M. S. Castelhano, and M. Mack, "Visual saliency does not account for eye movements during visual search in real-world scenes," in *Eye movements*. Elsevier, 2007, pp. 537–562.

[36] A. Borji, D. N. Sihite, and L. Itti, "Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 55–69, 2013.

[37] L. Itti and C. Koch, "Feature combination strategies for saliency-based visual attention systems," *Journal of Electronic Imaging*, vol. 10, no. 1, pp. 161–169, 2001.

[38] ——, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, no. 10, pp. 1489–1506, 2000.

[39] M. Cerf, E. P. Frady, and C. Koch, "Faces and text attract gaze independent of the task: Experimental data and computer model," *Journal of vision*, vol. 9, no. 12, pp. 10–10, 2009.

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

[41] N. Madrid and P. Hurtik, "Lane departure warning for mobile devices based on a fuzzy representation of images," *Fuzzy Sets and Systems*, vol. 291, pp. 144–159, 2016.

[42] P. Hurtik and N. Madrid, "Bilinear interpolation over fuzzified images: enlargement," in *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–8.

[43] C. Lopez-Molina and N. Madrid, "Non-linear scale-space based on fuzzy sharpening," in *Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS), 2017 Joint 17th World Congress of International*. IEEE, 2017, pp. 1–6.

[44] P. Hurtik, M. Vajgl, and N. Madrid, "Enhancement of night movies using fuzzy representation of images." in *FUZZ-IEEE*, 2016, pp. 1349–1354.

[45] F. Klawonn, "Fuzzy points, fuzzy relations and fuzzy functions," in *Discovering the World with Fuzzy Logic*. Physica-Verlag GmbH, 2000, pp. 431–453.

[46] I. Perfilieva, "Fuzzy function: Theoretical and practical point of view." in *EUSFLAT Conf.*, 2011, pp. 480–486.

[47] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience*, vol. 2, no. 3, p. 194, 2001.

[48] H. R. Tizhoosh, "Fuzzy image processing: potentials and state of the art," in *IIZUKA*, vol. 98, 1998, pp. 16–20.

[49] M. Nachtegael, D. Van der Weken, D. Van De Ville, and E. E. Kerre, *Fuzzy filters for image processing*. Springer, 2013, vol. 122.

[50] H. Bustince, E. Barrenechea, M. Pagola, and J. Fernández, "Interval-valued fuzzy sets constructed from matrices: Application to edge detection," *Fuzzy Sets and Systems*, vol. 160, no. 13, pp. 1819–1840, 2009.

[51] P.-E. Danielsson and O. Seger, "Generalized and separable sobel operators," in *Machine vision for three-dimensional scenes*. Elsevier, 1990, pp. 347–379.

[52] P. Hurtik and M. Vajgl, "Edge detection competition–algorithms based on image represented by a fuzzy function," in *Advances in Fuzzy Logic and Technology 2017*. Springer, 2017, pp. 260–265.

[53] V. Novák, I. Perfilieva, and A. Dvořák, *Insight into Fuzzy Modeling*. John Wiley & Sons, 2016.

[54] C. Hacker, I. Aizenberg, and J. Wilson, "Gpu simulator of multilayer neural network based on multi-valued neurons," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 4125–4132.

[55] Z. Hu, Y. V. Bodyanskiy, and O. K. Tyshchenko, "A deep cascade neural network based on extended neo-fuzzy neurons and its adaptive learning algorithm," in *Electrical and Computer Engineering (UKRCON), 2017 IEEE First Ukraine Conference on*. IEEE, 2017, pp. 801–805.

[56] C. P. Loizou, C. S. Pattichis, C. I. Christodoulou, R. S. Istepanian, M. Pantziaris, and A. Nicolaides, "Comparative evaluation of despeckle filtering in ultrasound imaging of the carotid artery," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 52, no. 10, pp. 1653–1669, 2005.

[57] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, no. 2, 2011, p. 5.

[58] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[59] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[61] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks–ICANN 2010*. Springer, 2010, pp. 92–101.

[62] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[63] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 44–51.

[64] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[66] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio, "Feature-wise transformations," *Distill*, 2018, https://distill.pub/2018/feature-wise-transformations.

[67] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters," *Vision research*, vol. 37, no. 23, pp. 3327–3338, 1997.

[68] M. F. Kragh, P. Christiansen, M. S. Laursen, M. Larsen, K. A. Steen, O. Green, H. Karstoft, and R. N. Jørgensen, "Fieldsafe: dataset for obstacle detection in agriculture," *Sensors*, vol. 17, no. 11, p. 2579, 2017.

Petr Hurtik has received M.Sc. (2011) in informatics and Ph.D. (2017) in applied mathematics from the University of Ostrava. From 2007 to 2012 he worked as a coder in several IT companies in the Czech Republic, and since 2012 he has been a member of IRAFM in Ostrava. His research interests are aimed at image processing and machine learning; they are projected into his publications where he published a lane detection algorithm for cars, real-time drone control by visual pattern matching, or collaborated on a system for sonouroflowmetry. He was awarded by the IEEE conference student Best paper in 2015, by the third place in IEEE CIS mobile application competition in 2015 and won EUSFLAT edge detection competition in 2017.

Vojtech Molek has received M.Sc. (2015) in informatics from the University of Ostrava. In 2014, he worked as an intern at IRAFM. Later, in 2016, he joined the graphics group of IRAFM. His field of study is Machine Learning with a focus on Deep learning and convolutional neural networks in particular. In his papers, he discusses the usage of Fuzzy approaches in Deep Learning or creation of synthetic datasets, among other topics.

Jan Hula has received M.A. (2011) in Industrial Design from the Tomas Bata University. Since 2015 he is a Ph.D. student of applied mathematics at the University of Ostrava. His research interests are aimed at various forms of Artificial Intelligence ranging from basic recognition mechanisms to higher forms of cognition connected to language and reasoning.