International Journal of Computational Intelligence Systems
Vol. **13(1)**, 2020, *pp.* 1404–1414
DOI: https://doi.org/10.2991/ijcis.d.200907.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

Deep Learning and Higher Degree F-Transforms: Interpretable Kernels Before and After Learning

Vojtech Molek, Irina Perfilieva*,

Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, Ostrava, 701 03, Czech Republic

ARTICLE INFO

Article History
Received 06 Jan 2020
Accepted 02 Sep 2020

Keywords
F-transform
Convolutional neural network
Deep learning
Interpretability

ABSTRACT

One of the current trends in the deep neural network technology consists in allowing a man–machine interaction and providing an explanation of network design and learning principles. In this direction, an experience with fuzzy systems is of great support. We propose our insight that is based on the particular theory of fuzzy (F)-transforms. Besides a theoretical explanation, we develop a new architecture of a deep neural network where the F-transform convolution kernels are used in the first two layers. Based on a series of experiments, we demonstrate the suitability of the F-transform-based deep neural network in the domain of image processing with the focus on recognition. Moreover, we support our insight by revealing the similarity between the F-transform and first-layer kernels in the most used deep neural networks.

1. INTRODUCTION

Deep neural networks (DNNs) significantly improve classification algorithms in various applications, giving a new impact to computer vision, speech recognition, etc. In the proposed contribution, we consider convolution neural networks (CNNs) and the deep learning (DL) methodology for improving parameters of their basic operations.

We are focused on a smart and conscious initialization of convolutional kernels in the first and second CNN layers where neurons have restricted receptive fields. Our motivation stems from the observation that although CNN is able to accurately generate the classification label, it does not report on features that cause this classification. Without an understanding of how DL comes to a solution, there is no guarantee that the trained networks will move from a laboratory to real systems [1]. The reason is that the inputs can significantly change during exploitation, and there is no guarantee that the machine learning tools will work effectively with these changes.

To fill this gap, we propose an insight into why and how CNN makes a decision, or why a specified object has been given a specific classification label. Our approach can be called "preprocessing of methodology" in the sense that we propose a CNN initialization, which ensures that features with known meaning are extracted. Then, we allow the network to learn the initialization parameters so that they can match the available data better.

Our approach differs from many similar ones, based on fuzzy rules, in which the explanation of the CNN decision is based on a posteriori analysis, i.e., after the features are extracted, see [1] and references therein.

We observe that smart preprocessing becomes more and more important in the DL methodology due to the increasing complexity of datasets and objects therein. It heavily depends on a network assignment and consists of traditional de-noising, regularization, reduction of dimensionality, labeling, etc. In most cases, preprocessing is realized in the first convolution layers, together with feature extraction. In subsequent fully connected layers, the extracted features are used for classification, recognition, etc. Therefore, the initial objects are modeled by the extracted features, so that the former ones can be approximately reconstructed from the latter.

Additionally, we observe that similarly to the above, we can characterize the technique of the higher degree fuzzy (F-) transforms [2–4]. This observation leads us to the idea that the higher degree F-transform kernels can be used in the first convolution layers of CNNs that perform recognition or classification.

We are based on long-term work on various approximation models in the theory of fuzzy systems, and in particular, on those, based on the theory of higher degree F-transforms [2–4]. We have collected rich experience regarding creating F-transform-based models in various applications, including image [5–7]/time series [8,9] processing and DL architecture of NNs [10–12].

The principal difference between the DL and the higher degree F-transform is in the criterion of optimality, which is a quality of approximation (F-transform), or a loss function (DL methodology). To reach optimality, it is recommended to refine fuzzy partition and increase the degree of the F-transform [4], or to increase the

* *Corresponding author. Email: irina.pefilieva@osu.cz*

number of kernels in convolutional layers and increase the number of layers. Both recommendations have the same nature.

The mentioned similarity between the higher degree F-transform and the DL, motivated us to confirm these theoretical considerations by experiments. The latter were conducted in two opposite directions: at first, we trained a neural network with F-transform kernels and estimated its success, and at second, we analyzed kernels of already trained known neural networks and compared them with the F-transform ones. We discussed the obtained results in several conference papers [10–12], where we fully confirmed the hypothesis we made. In the proposed manuscript, we summarize all the results and give extended explanations to the theoretical backgrounds and experimental tests.

The paper is organized as follows: in Section 2, we briefly explain the information-theoretical principles of DNN and the impact of our contribution; in Section 3, we recall essential facts about the higher degree F-transform with the focus on the F^2-transform; in Section 4, we give details of our new neural network (FTNet) design. In Section 4.2, the performance of the proposed FTNet is discussed from various angles and compared with the baseline network. In Section 5.1, we discussed the problem of the kernel interpretability, where we examined the first convolutional layers of several well-known networks.

2. DL—A GLIMPSE OF THE THEORY AND POSITION OF OUR CONTRIBUTION TO IT

In this section, we explain the essence of our proposal from a theoretical point of view. We will start with a brief and focused description of DNN, and then explain how we contribute to the current state.

We refer to [13], where the very general characterization of a DNN as a particular computing machine is given: a DNN is a parametric model that performs sequential operations on inputs. Each such operation consists of a linear transformation (e.g., convolution in CNN types), followed by a nonlinear "activation." An essential factor for DNN success is the availability of large data sets, such as ImageNet and hardware, with a graphics processor, solve to the problem of multidimensional optimization.

In our understanding and approach, we distinguish three key elements in the design of DNN and its corresponding functioning strategy: architecture, the ability to create a good representation of input data, and optimization algorithms. Leaving architecture and optimization aside, we will give a brief description of the second key.

Roughly speaking [13], representation of the input data is any function of it that is useful for the task. If we focus on the most useful (the "best") representation, then we think of some quantitation, e.g., in terms of complexity or invariance. The relevant line of research is known as representative learning. Despite great interest in this, a comprehensive theory that explains how deep networks with DL methodology contribute to this still does not exist.

However, one thing is clear—the crucial role of the dataset, which is used for training. There is a close connection between the DNN architecture (the number of levels, frames, activations, etc.) and the dataset, which is used to train network parameters. An interesting phenomenon has been reported in [14], where the almost linear relationship was revealed between the sizes of the DNN computational model and the required amount of training data. Obviously, large and multi-object databases require more levels and more complex learning and optimization process.

In a CNN, representation of the input data is realized in the form of a collection of features; the latter are results of convolutions. The collection of features should be complete in the sense of a possible reconstruction (backward representation) of any input object.

Mathematically, the backward "lossy" representation of an object is its approximation. A neural network's ability to produce approximate representations of initial data objects was reported in many papers. However, as shown by earlier work, even neural networks with one hidden layer and sigmoidal activations are universal approximators of functions, see, e.g., [15]. Therefore, the question of why DNNs are advantageous in this regard is still open [13].

One possible explanation is that deeper architectures are better than their shallower counterparts because they are capable of covering not only the requirement of a suitable approximation but also invariance with respect to some rigid transformations. As an example, scattering networks [16] are a class of deep networks whose convolution filter banks are defined by multiple-resolution wavelet families and whose stability and local invariance are confirmed.

This fact supports our initiative in a creation of a "convolutional filter bank" whose kernels are taken from the theory of higher degree F-transforms. Comparing with wavelet kernels, the F-transform ones have clear interpretability in a single and sequential layers in a DNN computation.

It has been proven in many papers [2–4,7] that the higher degree F-transforms are universal approximators of smooth and discrete functions. The approximation on a whole domain is a combination of locally best approximations called F-transform components. They are represented by higher degree polynomials and parametrized by coefficients that correspond to average values of local and nonlocal derivatives of various degrees. If the F-transform is applied to images, then its parameters are used in regularization, edge detection, characterization of patches [7,17], etc. Their computation can be performed by discrete convolutions with kernels that, up to the second degree, are similar to those widely used in image processing, namely Gaussian, Sobel, Laplacian. Thus, we can draw an analogy with the DNN method of computation and call the parameters of the higher degree F-transform features. Moreover, based on a clear understanding of these features' semantic meaning, we say that a DNN with the F-transform kernels extracts features with a clear interpretation. In addition, the sequential application of F-transform kernels with an up to the second degree gives average (nonlocal) derivatives of higher and higher degrees.

Last but not least, we note that after training DNN, initialized by the F-transform kernels, the shapes of the kernels were not significantly distorted. This fact has been empirically verified on the two datasets: MNIST and CIFAR-10, see Section 4.2 where we compare kernels of various known DNNs after being trained on the same datasets. We observe a similarity of kernel shapes of all considered DNNs. This confirms the stability of the proposed DNN and its sufficiency with respect to the selected datasets.

3. THE F-TRANSFORM OF A HIGHER DEGREE (F^m-TRANSFORM)

In this section, we recall the main facts (see [4,18] for more details) about the higher degree F-transform and specifically F^2-transform—the technique, which will be used in the proposed below CNN with the F-transform kernels (FTNet).

3.1. Fuzzy Partition

The F-transform components are the result of a convolution of an object function (image, signal, etc.) and a generating function of what is regarded as a *fuzzy partition* of a universe.

Definition 1. Let $n > 2$, $a = x_0 = x_1 < \dots < x_n = x_{n+1} = b$ be fixed nodes within $[a, b] \subseteq \mathbb{R}$. Fuzzy sets $A_1, \dots, A_n : [a, b] \to [0, 1]$, identified with their membership functions defined on $[a, b]$, establish a fuzzy partition of $[a, b]$, if they fulfill the following conditions for $k = 1, \dots, n$:

1. $A_k(x_k) = 1$;
2. $A_k(x) = 0$ if $x \in [a, b] \setminus (x_{k-1}, x_{k+1})$;
3. $A_k(x)$ is continuous on $[x_{k-1}, x_{k+1}]$;
4. $A_k(x)$ for $k = 2, \dots, n$ strictly increases on $[x_{k-1}, x_k]$ and for $k = 1, \dots, n-1$ strictly decreases on $[x_k, x_{k+1}]$;
5. for all $x \in [a, b]$ holds the Ruspini condition

$$\sum_{k=1}^{n} A_k(x) = 1. \tag{1}$$

The elements of fuzzy partition $\{A_1, \dots, A_n\}$ are called *basic functions*.

In particular, an h-uniform fuzzy partition of $[a, b]$ can be obtained using the so called *generating function*

$$A : [-1, 1] \to [0, 1], \tag{2}$$

which is defined as an even, continuous and positive function everywhere on $[-1, 1]$ except for on boundaries, where it vanishes. Basic functions $A_2, \dots, A_{n-1}$ of an h-uniform fuzzy partition are rescaled and shifted copies of A in the sense that for all $k = 2, \dots, n-1$;

$$A_k(x) = \begin{cases} A(\frac{x - x_k}{h}), & x \in [x_k - h, x_k + h], \\ 0, & \text{otherwise.} \end{cases}$$

Below, we will be working with one particular case of an h-uniform fuzzy partition that is generated by the triangular-shaped function A^{tr} and its h-rescaled version A^{tr}_h, where

$$\begin{aligned} A^{tr}(x) &= 1 - |x|, \ x \in [-1, 1] \\ A^{tr}_h(x) &= 1 - \frac{|x|}{h}, \ x \in [-h, h]. \end{aligned} \tag{3}$$

A fuzzy partition generated by the triangular-shaped function A^{tr} will be referred to as *triangular shaped*.

3.2. Space $L_2(A_k)$

Let us fix $[a, b]$ and its h-uniform fuzzy partition $A_1, \dots, A_n$, where $n \geqslant 2$ and $h = \frac{b-a}{n-1}$[1]. Let k be a fixed integer from $\{1, \dots, n\}$, and let $L_2(A_k)$ be a set of square-integrable functions $f : [x_{k-1}, x_{k+1}] \to \mathbb{R}$. Denote $L_2(A_1, \dots, A_n)$ a set of functions $f : [a, b] \to \mathbb{R}$ such that for all $k = 1, \dots, n$, $f|_{[x_{k-1}, x_{k+1}]} \in L_2(A_k)$. In $L_2(A_k)$, we define an *inner product* of f and g

$$\begin{aligned} \langle f, g \rangle_k &= \int_{x_{k-1}}^{x_{k+1}} f(x) g(x) d\mu_k \\ &= \frac{1}{s_k} \int_{x_{k-1}}^{x_{k+1}} f(x) g(x) A_k(x) dx, \end{aligned} \tag{4}$$

where

$$s_k = \int_{x_{k-1}}^{x_{k+1}} A_k(x) dx.$$

The space $(L_2(A_k, \langle f, g \rangle_k))$ is a *Hilbert space*. We apply the Gram–Schmidt process to the linearly independent system of polynomials $\{1, x, x^2, \dots x^m\}$ restricted to the interval $[x_{k-1}, x_{k+1}]$ and convert it to an orthogonal system in $L_2(A_k)$. The resulting orthogonal polynomials are denoted by $P^0_k, P^1_k, P^2_k, \dots, P^m_k$.

Example 1. Below, we write the first three orthogonal polynomials P^0, P^1, P^2 in $L_2(A)$, where A is the generating function of a uniform fuzzy partition, and $\langle \cdot, \cdot \rangle_0$ is the inner product:

$$\begin{aligned} P^0(x) &= 1, \\ P^1(x) &= x, \\ P^2(x) &= x^2 - I_2, \text{ where } I_2 = h^2 \textstyle\int_{-1}^{1} x^2 A(x) dx, \end{aligned}$$

If generating function A^{tr} is triangular shaped and h-rescaled, then the polynomial P^2 can be simplified to the form

$$P^2(x) = x^2 - \frac{h^2}{6}. \tag{5}$$

We denote $L^m_2(A_k)$ a linear subspace of $L_2(A_k)$ with the basis $P^0_k, P^1_k, P^2_k \dots, P^m_k$.

3.3. F^m-Transform

In this section, we define the F^m-transform, $m \geqslant 0$, of a function f with polynomial components of degree m. Let us fix $[a, b]$ and its fuzzy partition $A_1, \dots, A_n$, $n \geqslant 2$.

Definition 2. [4] Let $f : [a, b] \to \mathbb{R}$ be a function from $L_2(A_1, \dots, A_n)$, and let $m \geqslant 0$ be a fixed integer. Let F^m_k be the k-th orthogonal projection of $f|_{[x_{k-1}, x_{k+1}]}$ on $L^m_2(A_k)$, $k = 1, \dots, n$. We say that the n-tuple $(F^m_1, \dots, F^m_n)$ is an F^m-transform of f with respect to $A_1, \dots, A_n$, or formally,

$$F^m[f] = (F^m_1, \dots, F^m_n).$$

F^m_k is called the k^{th} F^m-transform component of f.

[1] The text of this and the following subsection is a free version of a certain part of [4] where the theory of a higher degree F-transform was introduced.

Explicitly, each k^{th} component is represented by the m^{th} degree polynomial

$$F_k^m = c_{k,0}P_k^0 + c_{k,1}P_k^1 + \cdots + c_{k,m}P_k^m, \tag{6}$$

where

$$c_{k,i} = \frac{\langle f, P_k^i\rangle_k}{\langle P_k^i, P_k^i\rangle_k} = \frac{\int_a^b f(x)P_k^i(x)A_k(x)dx}{\int_a^b P_k^i(x)P_k^i(x)A_k(x)dx},\ i = 0, \dots, m.$$

Remark 1. By the orthogonality of basis polynomials $P_k^0, P_k^1, P_k^2 \dots, P_k^m$, the k^{th} F^m-transform component F_k^m can be decomposed as follows:

$$F_k^m = F_k^{m-1} + c_{k,m}P_k^m,\ k = 1, \dots, n,\ m \geqslant 1.$$

This fact shows that all subsequent F^m-transform components, starting with $m \geqslant 1$, include the preceding ones. In addition, the higher m, the better the quality of local approximation of f on $[x_{k-1}, x_{k+1}]$ by k^{th} F^m-transform component F_k^m[4], i.e.,

$$\| f|_{[x_{k-1},x_{k+1}]} - F_k^m \|_k \leqslant \| f|_{[x_{k-1},x_{k+1}]} - F_k^{m-1} \|_k,$$

where $\| \cdot \|_k$ is the norm in $L_2(A_k)$.

Definition 3. Let $F^m[f] = (F_1^m, \dots, F_n^m)$ be the direct F^m-transform of f with respect to $A_1, \dots, A_n$. Then the function

$$\hat{f}_n^m(x) = \sum_{k=1}^n F_k^m A_k(x),\ \ x \in [a, b], \tag{7}$$

is called the *inverse F^m-transform* of f.

The following theorem proved in [4] estimates the quality of approximation by the inverse F^m-transform in a normed space L_1.

Theorem 1. *Let $A_1, \dots, A_n$ be an h-uniform fuzzy partition of $[a, b]$. Moreover, let functions f and A_k, $k = 1, \dots, n$ be four times continuously differentiable on $[a, b]$, and let $\hat{f}_n^m$ be the inverse F^m-transform of f, where $m \geqslant 1$. Then*

$$\| f(x) - \hat{f}_n^m(x) \|_{L_1} \leqslant O(h^2),$$

where L_1 is the Lebesgue space on $[a + h, b - h]$.

3.4. F^2-Transform in the Convolutional Form

Let us fix $[a, b]$ and its h-uniform fuzzy partition $A_1, \dots, A_n$, $n \geqslant 2$, generated from $A : [-1, 1] \to [0, 1]$ and its h-rescaled version A_h, so that $A_k(x) = A(\frac{x-x_k}{h}) = A_h(x - x_k)$, $x \in [x_k - h, x_k + h]$, and $x_k = a + kh$. The F^2-transform of a function f from $L_2(A_1, \dots, A_n)$ has the following representation:

$$F^2[f] = (c_{1,0}P_1^0 + c_{1,1}P_1^1 + c_{1,2}P_1^2, \dots, \\ c_{n,0}P_n^0 + c_{n,1}P_n^1 + c_{n,2}P_n^2), \tag{8}$$

where for all $k = 1, \dots, n$,

$$P_k^0(x) = 1,\ P_k^1(x) = x - x_k,\ P_k^2(x) = (x - x_k)^2 - I_2,$$

$I_2 = h^2 \int_{-1}^1 x^2 A(x)dx$, and coefficients are as follows:

$$c_{k,0} = \frac{\int_{-\infty}^{\infty} f(x)A_h(x - x_k)dx}{\int_{-\infty}^{\infty} A_h(x - x_k)dx}, \tag{9}$$

$$c_{k,1} = \frac{\int_{-\infty}^{\infty} f(x)(x - x_k)A_h(x - x_k)dx}{\int_{-\infty}^{\infty} (x - x_k)^2 A_h(x - x_k)dx}, \tag{10}$$

$$c_{k,2} = \frac{\int_{-\infty}^{\infty} f(x)((x - x_k)^2 - I_2)A_h(x - x_k)dx}{\int_{-\infty}^{\infty} ((x - x_k)^2 - I_2)^2 A_h(x - x_k)dx}. \tag{11}$$

In [4,18], it has been proved that

$$c_{k,0} \approx f(x_k),\ c_{k,1} \approx f'(x_k),\ c_{k,2} \approx f''(x_k), \tag{12}$$

where $\approx$ is meant up to $O(h^2)$.

Without going into technical details, we rewrite (9–11) into the following discrete representations

$$\begin{aligned} c_{k,0} &= \sum_{j=1}^{l} f(j)g_0(ks - j) \\ c_{k,1} &= \sum_{j=1}^{l} f(j)g_1(ks - j) \\ c_{k,2} &= \sum_{j=1}^{l} f(j)g_2(ks - j) \end{aligned} \tag{13}$$

where $k = 1, \dots, n$, $n = \lfloor \frac{l}{s} \rfloor$, s is the so called "stride" and g_0, g_1, g_2 are normalized functions that correspond to generating functions A_h, (xA_h) and $((x^2 - I_2)A_h)$. It is easy to see that if $s = 1$, then coefficients $c_{k,0}$, $c_{k,1}$, $c_{k,2}$ are the results (k-th coordinates) of the corresponding discrete convolutions $f \star g_0, f \star g_1, f \star g_2$, written in vector form. Thus, we can rewrite the representation of F^2 in (8), using the following vector form:

$$F^2[f]^T = ((f \star_s g_0)^T\mathbf{P}^0 + (f \star_s g_1)^T\mathbf{P}^1 + (f \star_s g_2)^T\mathbf{P}^2),$$

where $\mathbf{P}^0$, $\mathbf{P}^1$, $\mathbf{P}^2$ are vectors of the corresponding polynomials, and $\star_s$ denotes the convolution with the stride s, $s \geqslant 1$.

4. FTNET—CNN WITH F-TRANSFORM KERNELS

In this section, we discuss the details of our neural network design. We chose the LeNet-5 [19] as an architecture prototype and composed a new CNN—FTNet with the kernels initialization taken from the higher degree F-transforms theory [11]. We applied FTNet to several datasets and evaluated the results. For simplicity, we restricted the FTNet architecture to the fixed number of convolutional kernels in the first and second convolutional layers, making the one-to-one correspondence between set of convolutional kernels and set of F-transform kernels K. We use up to second-degree F-transform kernels and some of their modifications based on principal geometrical transformations (rotation and flipping).

In detail, we replace convolutional kernels in the first and second convolutional layers C_1 and C_3 with the F-transform kernels according to Eq. (13) adapted to functions of two variables. Together with negative versions of kernels, The FTNet C_1-layer has 8 distinct kernels. Each of C_1 feature maps is further processed in C_3 with the same 8 kernels, and as a result, 64 feature maps are obtained.

The details of the FTNet architecture are given below in Table 1. Note that layer FC_6 has *var*iable number of neurons, according to number of classes in dataset.

4.1. Datasets

All the discussed experiments were realized on the following databases: MNIST [20], CIFAR-10 [21], Caltech 101 [16], and Intel Image classification.[2] Datasets details are shown in Table 2

We convert all datasets to the grayscale because the F-transform kernels extract features with functional meaning and are insensitive to colors. Moreover, we downscale Caltech 1O1 and Intel to 64×64 resolution as FTNet is not suited for higher resolution. Additionally we normalize [22] both datasets. Since Caltech 101 does not have an official train/test split, we split it in 4:1 ratio.

Below, we give a short overview of the known neuro-fuzzy networks, trained on MNIST, and designed for the pattern recognition. We will use MNIST to compare some of the following approaches with FTNet.

Authors of [23] improved the MNIST recognition by optimizing features and architecture, and reached an accuracy of 99.52% on 10 000 testing images. This accuracy is similar to that claimed in [24,25].

In [26], the feature selection is based on the wavelet transform that uses 2D scaling moments and various classifiers (support vector machines/classifiers, artificial neural networks, neuro-fuzzy classifiers, and others). The SVM classifier demonstrates the best accuracy of 99.39%, while the neuro-fuzzy one achieves accuracy 98.72%.

Similar to MNIST, the database of handwritten characters Chars74k [27] has been studied in [28]. The authors used a three-fold cross-validation and achieved 97.22% accuracy.

In the recent publication [29], the Fuzzy Deep Belief Net (FDBN) was proposed to classify MNIST with different types and levels of noise. The FDBN architecture is described in [30,31], where the authors declared better results than using the standard Deep Belief Networks.

Table 1 FTNet architecture.

Hyper-parameter	Layers					
	C_1	S_2	C_3	S_4	FC_5	FC_6
Kernel size	5×5	-	5×5	-	-	-
# Kernels	8	-	64	-	-	-
Stride	1×1	2×2	1×1	2×2	-	-
Pooling size	-	2×2	-	2×2	-	
# FC units	-	-	-	-	500	*var*

[2]https://www.kaggle.com/puneet6060/intel-image-classification

4.2. Performance of FTNet and Comparison with the He Initialization

In this section, we compare FTNet initialized with F-transform kernels with Baseline network using the same architecture and He initialization [32]—one of the most common initializations.

We follow He initialization and scale F-transform kernels to $\left[-\sqrt{2/fan_{in}}, +\sqrt{2/fan_{in}}\right]$, where fan_{in} is number of incoming neurons to the layer. C_1 initialization is straight forward as the input is a single channel image. To initialize C_3 we set majority of the kernel values to zeros, effectively turning them into 64 $5\times5\times1$. This is the same for both FTNet and Baseline network.

We remark that in the FTNet, the C_3-layer uses the same kernels as in the C_1, i.e., C_1 computes feature maps $fm_1^{C_1},\dots,fm_8^{C_1}$ using F-transform kernels K and C_3 creates $fm_1^{C_3},\dots,fm_{64}^{C_3}$ feature maps applying all 8 F-transform kernels $fm_i^{C_3} = fm_{\lfloor i/|K|\rfloor}^{C_1} * K_{i\,mod\,|K|}$. This way C_1 and C_3 performs two successive convolutions with all possible kernel combinations.

Figure 1 shows normalized results of training Baseline network and two variants of FTNet. FTNet (red graph curve) corresponds to F-transform kernel initialization, where kernels are allowed to learn. During the learning the kernels are modified to a certain degree. From the graphs, we can see that **F-transform kernels initialization is advantageous over He initialization.** The Second variation does not allow F-transform kernels to learn and kept unchanged. We can observe that while it has lower loss values at the beginning of training, it falls behind later on. The advantage of static FTNet is higher training speed and lower number of trainable parameters; this can be particularly beneficial to overfitting problem and problem of small datasets. Lastly, static FTNet kernels and features they extract are clearly interpretable.

We trained each network 10 times in 2 epochs. For the training, we used Adam ($\alpha = 1e-3, \gamma = 1e-3$)[3] optimizer, cross entropy loss, FC_5 and FC_6 with $L_2(\lambda = 1e-3)^3$ regularization and batch size = 50. For initialization of any layer not initialized with F-transform kernels, we used He initialization.

Since disabling trainability of C_1 and C_3 decreases the number of free parameters in the network, the training time decreases accordingly. The average training time for each network setting and each database is shown in Table 3.

Due to MNIST being the most common dataset, we use it to compare FTNet with other approaches. In Table 4 you can see comparison of FTNet and results reported in selected publications (the latter are indicated by their reference numbers in the first row).

Table 2 Datasets used for experiments.

	MNIST	CIFAR-10	Caltech 101	Intel
Res.	28×28	32×32	*var*	150×150
Color	Gray	RGB	RGB	RGB
Train	60k	50k	7281	14034
Test	10 k	10k	1863	3000
Classes	10	10	101	6

[3]α - learning rate, γ - decay, λ - strength of regularization.

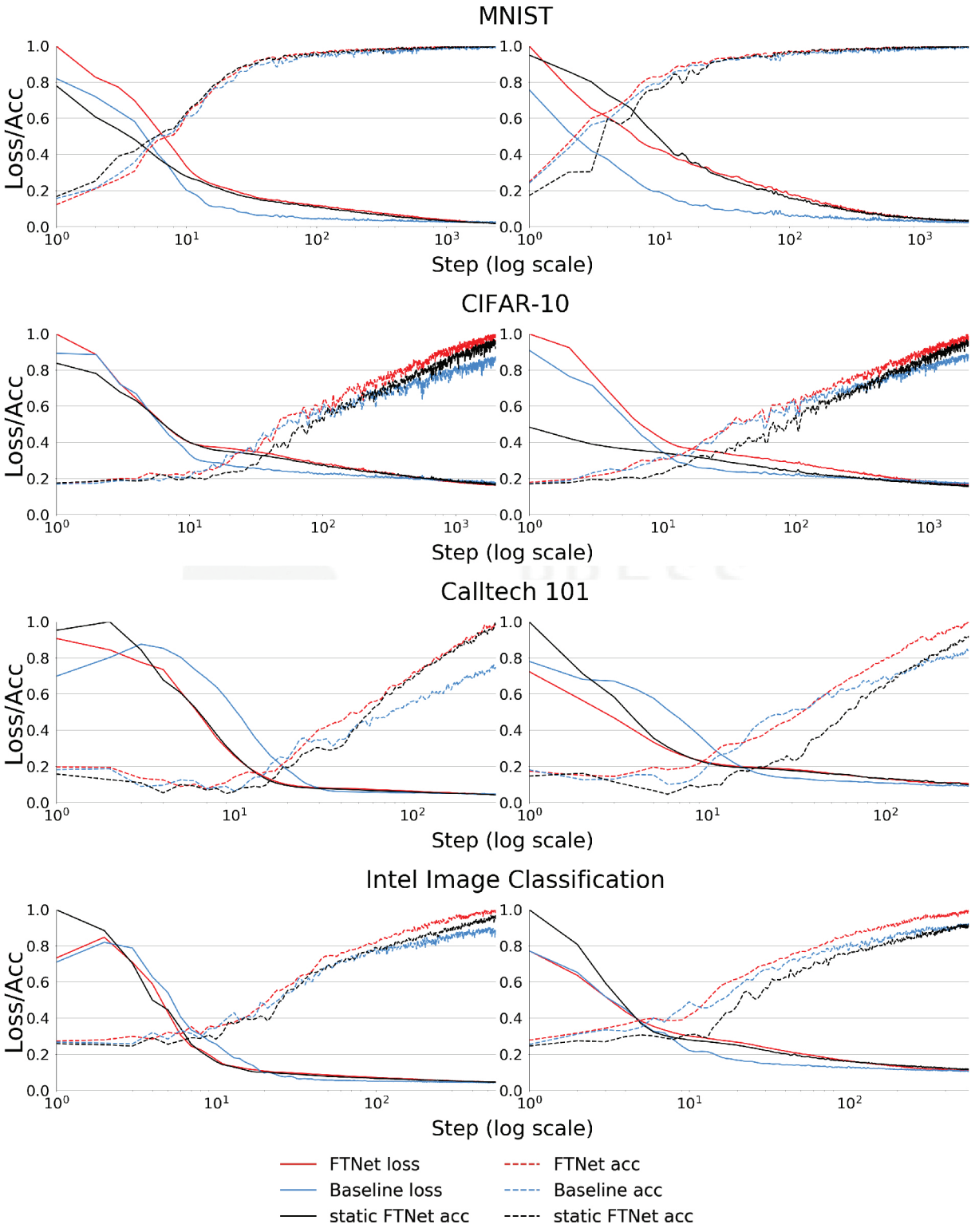

Figure 1 Results of 2 epoch training on datasets. In the left column are results of training with FTNet C_1 initialized with F-transform kernels. Second column contains results of training with both C_1 and C_3 initialized with F-transform kernels. Note that both accuracy and loss are scaled to [0, 1].

4.3. F-Transform Kernels as Preprocessing

One can use the F-transform to process data before feeding them into a network. We perform a comparative experiment between the F-transform and recently published fuzzy preprocessing technique—IRFF [35]. IRFF processes images in a convolutional manner, saving minimum, maximum, and central pixel values from a selected neighborhood. In general, the IRFF preprocessing increases the accuracy of a network.

We conducted experiment, comparing the IRFF and F-transform performances on CIFAR-10 using ResNet34 [36] with Shake-Shake regularization [37][4] that achieved accuracy up to 97.14%. From Figure 2, we see that the preprocessing of the F-transform is on the same level as IRFF.

5. FTNET HYPERPARAMETERS AND INTERPRETABILITY

To assure that we use a proper combination of hyperparameters, we searched through the hyperparameters space, determined by four hyperparameters: **Initialization** I, **kernels size** D, **Trainability** T, and **Subsampling** S. These hyperparameters applies to C_1, S_2, C_3 and S_4 layers.

Table 3 Average training times for FTNet and its variants and baseline network.

Dataset	FTNet		Baseline
	Trainable	Nontrainable	
MNIST	320s	299s	315s
CIFAR-10	290s	252s	289s
Caltech 101	20s	15s	20s
Intel	65s	53s	66s

Table 4 Comparison of FTNet with other approaches on MNIST.

[26]	[29]	[23]	[33]	[34]	FTNet
99.39%	99.01%	99.52%	99.23%	99.58%	99.39%

[4]https://github.com/jonnedtc/Shake-Shake-Keras

Let us describe the functionality of the hyperparameters mentioned above:

- I determines whether layer C_1 and C_3 are initialized with F-transform kernels or random ones with $\sim N(0,1)$.
- D determines the size of C_1 and C_3 convolution kernels.
- T determines C_1 and C_3 trainability.
- S determines whether C_1 and C_3 are followed by *maxpooling layer*[5] or are *strided convolutions* [40] or neither.

Scale-space [41] inspired **VA**riable **C**onvolutional **L**ayer (VACL) value of D specifies C_1 convolution sizes such that it has all three sizes (3×3, 5×5 and 7×7). With VACL, we force the network to process the input image through multiple scales (resolutions). VACL schema is in Figure 3.

Using VACL in C_3 leads to huge increase in trainable parameters and results in overfitting. For this reason we remove D = VACL option for C_3. The results of our searching are sorted with respect to the accuracy on the testing portion of datasets. Figures 4 and 5 contain relative frequencies of hyperparameters of the 500 best combinations. In the case of MNIST (Figure 4), the most prevalent kernel size for C_1 is VACL; this confirms our hypothesis regarding the scale-space methodology beneficence to the learning process. Among the optimal initialization of C_1 and C_3 we see significantly more combinations with **trainable** F-transform kernels. We conclude that the F-transform kernels initialization has serious, positive impact on the accuracy of FTNet. The results on CIFAR-10

Table 5 The four hyperparameters values: D, T, and I are associated with C_1 and C_3 layer; S is associated with C_1, S_2, C_3 and S_4 layer.

D	S	T	I
3×3	None	Trainable	F-transform kernels
5×5	Max-pool	Nontrainable	Kernels $\sim N(0,1)$
7×7	Stride	-	-
VACL	-	-	-

[5]Subsampling operation originates from Hubel and Wiesel [38]; comparison of pooling can be found in Ref. [39].

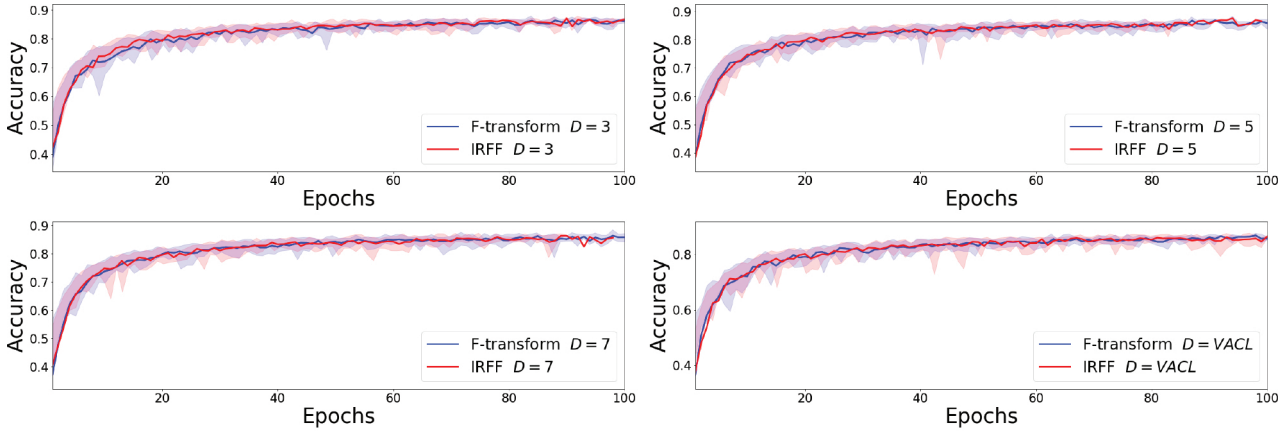

Figure 2 Average accuracy (over 10 runs) of ResNet34 with Shake-Shake regularization on CIFAR-10 over 100 epochs. Network uses Adam($\alpha = 1e-3$), cross entropy loss, early stopping, batch size 128, and light augmentation (width/height shift up to 0.1 and vertical flipping).

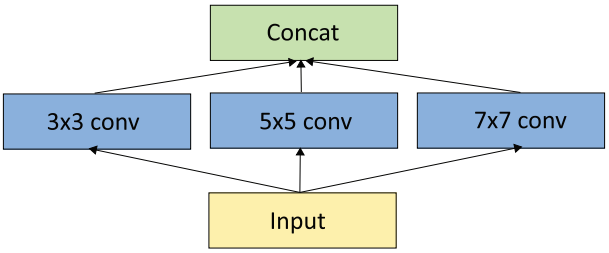

Figure 3 Scheme of the convolutional layer with variable kernels sizes, realized as multiple convolution layers with their outputs concatenated.

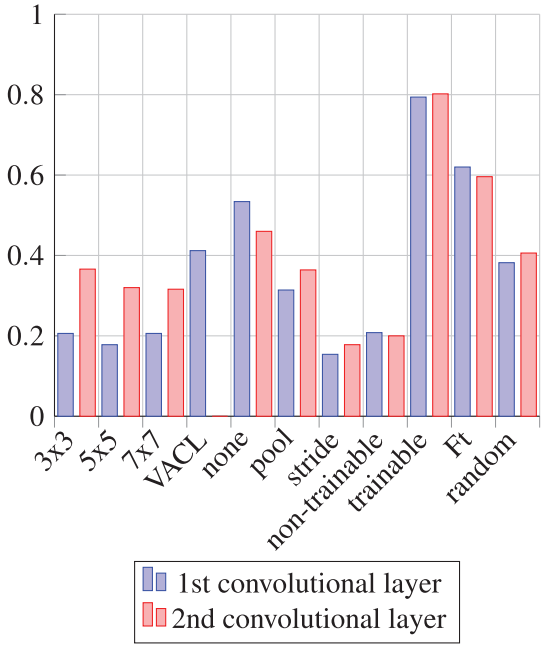

Figure 4 Relative frequencies of the hyperparameters values for C_1 and C_3 within the first 500 best combinations in terms of accuracy after 3 epochs of learning on MNIST.

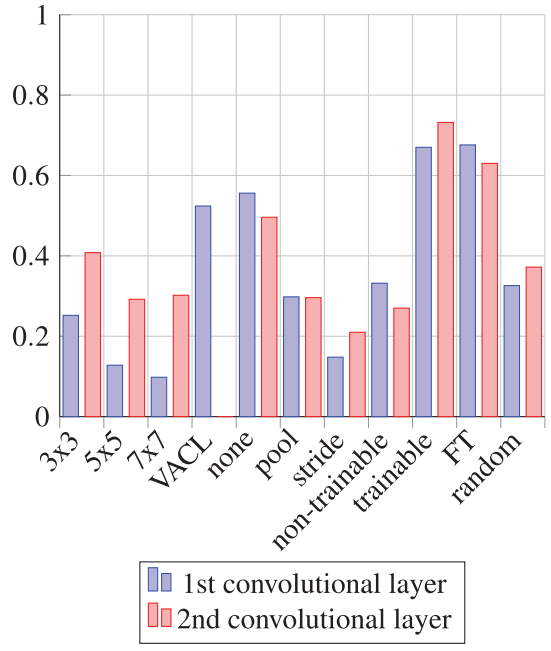

Figure 5 Relative frequencies of the hyperparameters values for C_1 and C_3 within the first 500 best combinations in terms of accuracy after 3 epochs of learning on CIFAR-10.

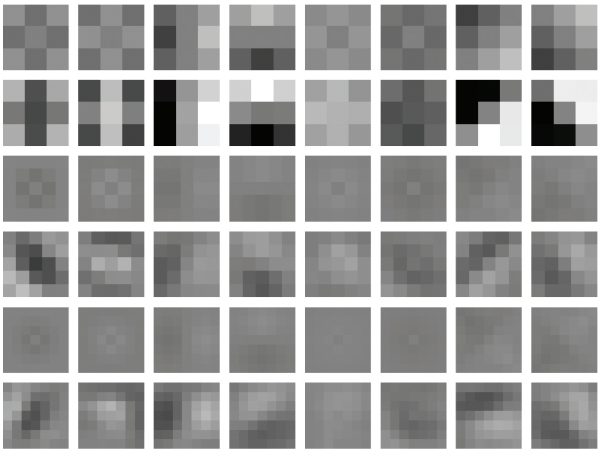

Figure 6 Visualization of the eight F-transform kernels before and after training (100 epochs) in C_1. The first two rows contain 3×3 kernels (training does not change the kernels significantly). The second two rows contain 5×5 kernels (the training added some variational details) and the third two rows contain 7×7 kernels (the training changed the F^0-transform kernels transforming them to the rotated F^1-fransform).

dataset have the same statistics and additionally prove the usefulness of VACL and F-transform initialization.

An unexpected result is a high relative frequency of the "no subsampling" in both cases while stride being worst out of the three.

As an additional argument in favor of our technique, we visualize the F-transform kernels in C_1 before and after 100 epochs of training with the following setting: I = *F-transform kernels*, $D = \{C_1 = \text{VACL}, C_3 = 3 \times 3\}$, T = *trainable*, S = *max pool* and dropout [42][6] between FC_5 and FC_6 combination. In Figure 6, we see that

1. Up to the small contrast changes, 3×3 kernels remain unchanged.
2. The shapes of 5×5 kernels are generally preserved; however, some variational details were added after training.
3. The shapes of 7×7 are preserved for F^1 and F^2 kernels; however the F^0 kernels became similar to the rotated F^1.

[6]We have employed dropout to reduce network overfitting.

Thus, we can summarize achieved results:

1. The scale-space inspired VACL is the most frequent size of the convolution kernels in the first convolutional layer of the considered CNN, trained on both datasets.
2. The initialization of C_1 and C_3 convolutional layers with the F-transform kernels leads to the higher network accuracy.

3. Excluding subsampling from CNN's architecture increases network accuracy; however, it contributes to an undesirable effect of overfitting.
4. Including subsampling into CNN's architecture leads to a quicker decrease of a loss function.
5. The F-transform kernels in the first layer C_1, do not significantly change their shapes during training. Therefore they are an ideal choice for feature extraction.

5.1. Semantic Meaning of Principal Kernels in Convolutional Layers

In this subsection, we tackle the problem of interpretability from the opposite angle. Instead of initializing a network with predefined kernels, we examine the first convolutional layers and the corresponding to them (already trained) kernels taken from several well-known networks. The purpose is to find general semantic meanings of kernels and through them compare with the F-transform kernels.

We tried to assign interpretation to kernels of already trained CNNs: We based on the known interpretation of the higher degree F-transform kernels and wished to reveal the similar meaning of kernels extracted from the first convolutional layer of several frequently used CNNs, trained on the ImageNet [43]. Let us review some of the existing contributions that connect both DL and fuzzy disciplines.

We selected 6 networks: VGG16 [44], VGG19 [44], InceptionV3 [45], MobileNet [46], ResNet [36], and AlexNet [47] as the representative examples of CNNs. All of the networks were trained on ImageNet [43], using the same training database, consisting of $\approx$ 1.2M RGB images[7] with various resolution (usually downsampled to 256×256).

We extracted kernels from the first convolutional layer of all considered networks and analyzed whether there are similarities among kernels across the networks. To reduce the space of kernels, we first apply the hierarchical clustering on every network kernel set separately, and then, look for the similarities among clusters. The *medoids* of the found clusters are shown in Figure 7.

We observed that the extracted clusters contain similar elements (kernels) across the different networks that share one of the following characteristics/functionality: **gaussian-like**; **edge detection (with various angle specifications)**; **texture detection**; **color blobs**.

If we compare the semantic meaning of the extracted clusters (in terms of the above-given characteristics/functionality) with that of the F-transform kernels in the FTNet, then we see the coincidence in the first two items from the above-given list. To be more precise, the F^0-transform kernels are the Gaussian-like, and the F^1-transform kernels are (horizontal or vertical) edge detectors. This again supports our conclusion regarding the suitability of the F-transform kernels in the first layers of CNNs.

[7]ImageNet database content depends on the year of ILSVRC competition.

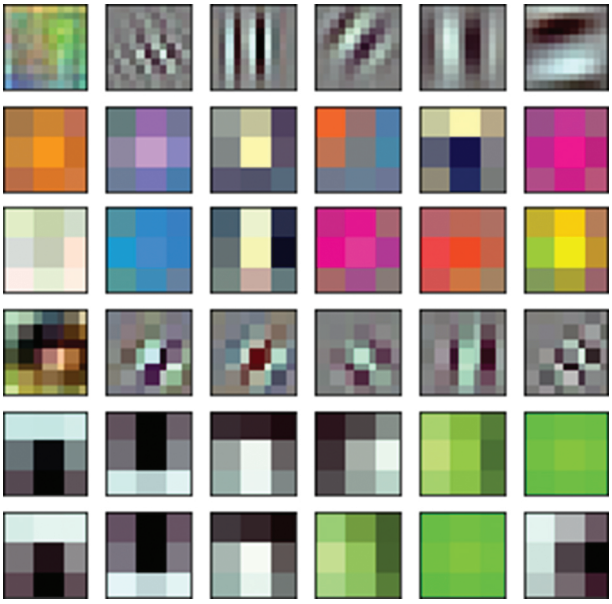

Figure 7 | From top to down: the medoids of the clusterized kernels from the first convolutional layer of AlexNet, InceptionV3, MobileNet, ResNet, VGG 16, and VGG 19.

The above-formulated general conclusion relates to the disclosed semantic meaning of convolutional kernels. This knowledge is helpful for the optimal and nonexhaustive design of CNNs.

6. CONCLUSION AND THE FUTURE WORK

We have proposed a new CNN learning methodology that is focused on a smart preprocessing with the meaningful initialization of CNN kernels in the first two layers. The methodology is based on the fuzzy modeling technique—F-transform. As a result, we have designed a new CNN-type network called FTNet.

The performance of FTNet was examined on several datasets and on them it converges faster in terms of accuracy/loss than the baseline network, subject to the same number of steps.

We compared the F-transform kernels in the first layer before and after training. We observed that the kernels remain unchanged. Moreover, their shapes are similar to the shapes of extracted kernel groups from the most known CNNs.

All these facts confirm our hypothesis that the smart initialization of the first layers kernels can be proposed based on their semantic meaning and the general network designation.

Our future work will be focused on neural nets with larger number of layers and other than recognition objectives.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

AUTHORS' CONTRIBUTIONS

Authors Irina Perfilieva and Vojtech Molek made equal contributions.

ACKNOWLEDGMENTS

The work is supported by ERDF/ESF "Center 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).

REFERENCES

[1] D. Bonanno, K. Nock, L. Smith, P. Elmore, F. Petry, An approach to explainable deep learning using fuzzy inference, in Next-Generation Analyst V, Proceeding of SPIE, SPIE, Anaheim, CA, USA, 2017, vol. 102070D, pp. 576–581.

[2] I. Perfilieva, M. Holčapek, V. Kreinovich, A new reconstruction from the F-transform components, Fuzzy Sets Syst. 288 (2016), 3–25.

[3] I. Perfilieva, Fuzzy transforms: theory and applications, Fuzzy Sets Syst. 157 (2006), 993–1023.

[4] I. Perfilieva, M. Danková, B. Bede, Towards F-transform of a higher degree, in IFSA/EUSFLAT Conference, Lisbon, Portugal, 2009, pp. 585–588.

[5] P. Hurtik, S. Tomasiello, A review on the application of fuzzy transform in data and image compression, Soft Comput. 23 (2019), 12641–12653.

[6] P. Vlasanek, I. Perfilieva, Interpolation techniques *versus* F-transform in application to image reconstruction, in 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, Beijing, China, 2014, pp. 533–539.

[7] I. Perfilieva, P. Hodáková, P. Hurtík, Differentiation by the F-transform and application to edge detection, Fuzzy Sets Syst. 288 (2016), 96–114.

[8] V. Novák, M. Štěpnička, A. Dvořák, I. Perfilieva, V. Pavliska, L. Vavříčková, Analysis of seasonal time series using fuzzy approach, Int. J. Gen. Syst. 39 (2010), 305–328.

[9] I. Perfilieva, V. Novak, V. Pavliska, A. Dvorak, M. Stepnicka, Analysis and prediction of time series using fuzzy transform, in 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, Hong Kong, China, 2008, pp. 3876–3880.

[10] V. Molek, I. Perfilieva, Relationship between convolutional neural networks and F-transform, in Uncertainty Modelling in Knowledge Engineering and Decision Making: Proceedings of the 12th International FLINS Conference, World Scientific, Roubaix, France, 2016, pp. 325–328.

[11] V. Molek, I. Perfilieva, Convolutional neural networks with the F-transform kernels, in International Work-Conference on Artificial Neural Networks, Springer, Cadiz, Spain, 2017, pp. 396–407.

[12] V. Molek, I. Perfilieva, Scale-space theory, F-transform kernels and cnn realization, in International Work-Conference on Artificial Neural Networks, Springer, Gran Canaria, Spain, 2019, pp. 38–48.

[13] R. Vidal, J. Bruna, R. Giryes, S. Soatto, Mathematics of deep learning, arXiv preprint arXiv:1712.04741, 2017.

[14] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, arXiv preprint arXiv:1808.01974 [cs.LG], 2018.

[15] K. Hornik, Deep learning in neural networks: an overview, Neural Netw. 4 (1991), 251–257.

[16] L. Fei-Fei, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006), 594–611.

[17] I. Perfilieva, P. Vlašánek, Total variation with nonlocal FT-laplacian for patch-based inpainting, Soft Comput. 23 (2019), 1833–1841.

[18] I. Perfilieva, V. Kreinovich, Fuzzy transforms of higher order approximate derivatives: atheorem, Fuzzy Sets Syst. 180 (2011), 55–68.

[19] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Pro. IEEE. 86 (1998), 2278–2324.

[20] L. Deng, The mnist database of handwritten digit images for machine learning research [best of the web], IEEE Signal Process. Mag. 29 (2012), 141–142.

[21] A. Krizhevsky, G. Hinton, *et al.*, Learning Multiple Layers of Features from Tiny Images, Technical Report, Citeseer, 2009.

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

[23] A.B. Bayat, Recognition of handwritten digits using optimized adaptive neuro-fuzzy inference systems and effective features, J. Pattern Recognit. Intell. Syst. 1 (2013), 25–37.

[24] D. Cireşan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, arXiv preprint arXiv:1202.2745, 2012.

[25] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber, Deep big simple neural nets excel on handwritten digit recognition, Neural Comput. 22 (2010), 3207–3220.

[26] B. Cetişli, R. Edizkan, Use of wavelet-based two-dimensional scaling moments and structural features in cascade neuro-fuzzy classifiers for handwritten digit recognition, Neural Comput. Appl. 26 (2015), 613–624.

[27] T.E. De Campos, B.R. Babu, M. Varma, *et al.*, Character recognition in natural images, in Proceedings of the Fourth International Conference on Computer Vision Theory and Applications (VISAPP), Lisboa, Portugal, 2009, vol. 2.

[28] S. Ahlawat, R. Rishi, Handwritten digit recognition using adaptive neuro-fuzzy system and ranked features, in 2018 International Conference on Computing, Power and Communication Technologies (GUCON), IEEE, Greater Noida, India, 2018, pp. 1128–1132.

[29] S. Feng, C.L. Philip Chen, C.-Y. Zhang, A fuzzy deep model based on fuzzy restricted boltzmann machines for high-dimensional data classification, IEEE Trans. Fuzzy Syst. 28 (2019), 1344–1355.

[30] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006), 1527–1554.

[31] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science. 313 (2006), 504–507.

[32] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, in Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 2015, pp. 1026–1034.

[33] E.A. Popko, I.A. Weinstein, Fuzzy logic module of convolutional neural network for handwritten digits recognition, in Journal of Physics: Conference Series, IOP Publishing, Athens, Greece, 2016, vol. 738, p. 012123.

[34] O. Yazdanbakhsh, S. Dick, A deep neuro-fuzzy network for image classification, arXiv preprint arXiv:2001.01686, 2019.

[35] P. Hurtik, V. Molek, J. Hula, Data preprocessing technique for neural networks based on image represented by a fuzzy function, IEEE Trans. Fuzzy Syst. 28 (2019), 1195–1204.
[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Nevada, 2016, pp. 770–778.
[37] X. Gastaldi, Shake-shake regularization, arXiv preprint arXiv:1705.07485, 2017.
[38] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, J. Physiol. 160 (1962), 106–154.
[39] D. Scherer, A. Müller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in Artificial Neural Networks-ICANN 2010, Springer, Thessaloniki, Greece, 2010, pp. 92–101.
[40] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net, arXiv preprint arXiv:1412.6806, 2014.
[41] J.M. Ogden, E.H. Adelson, J.R. Bergen, P.J. Burt, Pyramid-based computer graphics, RCA Eng. 30 (1985), 4–15.
[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014), 1929–1958.
[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, *et al.*, Imagenet large scale visual recognition challenge, Int. J. Comput. Vision. 115 (2015), 211–252.
[44] K. Simonyan, A. Zisserman., Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 2818–2826.
[46] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861, 2017.
[47] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in Neural Information Processing Systems, Nevada, USA, 2012, pp. 1097–1105.