

Convolutional Neural Networks with the F-transform Kernels

Vojtech Molek^(✉) and Irina Perfilieva

University of Ostrava, Ostrava 701 03, Czech Republic
{vojtech.molek, irina.pefilieva}@osu.cz
<http://www.osu.cz/>, <http://irafm.osu.cz/>

Abstract. We propose a new convolutional neural network – the FTNet and explain its theoretical background referring to the theory of a higher degree F-transform. The FTNet is parametrized by kernel sizes, on/off activation of weights learning, the choice of strides or pooling, etc. It is trained on the database MNIST and tested on handwritten inputs. The obtained results demonstrate that the FTNet has better recognition accuracy than the automatically trained LENET-5. We have also analyzed the FTNet and LENET-5 rotation invariance.

1 Introduction

Deep learning (DL) [1] neural networks have proven themselves as efficient tools for pattern recognition [2–4]. One of the main principles of the DL is based on automatic extraction of “good” features [5] using a general-purpose learning procedure [6, 7]. This is opposite to hand designed feature extractors that require a considerable amount of testing time and expert skills [8–10].

In this contribution, we argue with the absolutization of the above given main principle and propose the theoretical background of FTNets – convolutional neural networks (CNN) that use kernels related to a higher degree F-transform [11]. The FTNet is parametrized by kernel sizes, on/off activation of weights learning, the choice of strides or pooling, etc. It is trained on the database MNIST and tested on handwritten inputs.

The obtained results demonstrate that the FTNet has better recognition accuracy than the automatically trained LENET-5 [12]. The efficiency of the proposed FTNet (measured in training time) is higher. Last, but not least, we provide the theoretical justification of a suitability of the FTNet for the problem of recognition.

To confirm our conclusion, we compare the FTNet networks with the LENET-5 (both are trained on the dataset *MNIST*) on various recognition tests. The results are discussed in Sect. 4.2. We have chosen LENET-5, because it was specially designed for dataset *MNIST* whose objects are hand drawn integers from 0 to 9 together with their various transforms. LENET-5 has a reasonable size, good performance accuracy and serves as a prototype for many other convolutional networks. Moreover, LENET-5 and its modifications are included into many modern machine learning frameworks.

The structure of the paper is as follows: in Sect. 2 we give a short characterization of convolutional neural networks; Sect. 3 recalls the main facts about the higher degree F-transform and specifically F^2 -transform - the technique, which will be used in the proposed FTNet networks; Sect. 4 contains description of tests and discussion of their results.

2 Convolutional Neural Networks

Convolutional Neural Networks [13] are hierarchical models capable of learning. The hierarchy consists of layers of units. The layers are connected together in a cascade manner. They can be specified according to their types. One of the types is a convolutional type. Units in convolutional layer are partially connected to units of the previous layer, unlike units in fully connected layers. Each units of a convolutional layer performs operation known as *convolution*¹, thus the layer name. The purpose of a convolutional layer is to extract features. Multiple convolutional layers connected one after another extract features of higher abstractions. Multiple connected convolutional layers are interlarded with *pooling* (*sub-sampling*) layers which should ensure tolerance to translations and distortions (Fig. 1).

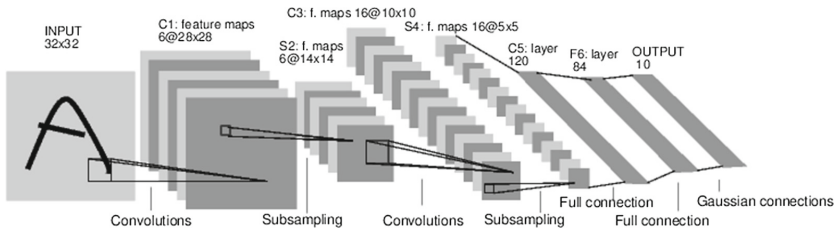


Fig. 1. LUNET-5 architecture reproduced from [12]

CNN architecture reproduced from paper [13] is considered as perhaps the first that deserves the label *deep* [1]. The difference between deep and shallow networks is not clearly distinguished (more on the topic can be found in the article [1]). Learning deep convolutional neural network (*DNN*) using a learning algorithm (the back-propagation with gradient descent [14]) proved to be computationally heavy. The problem of intense computations was simplified by the advent of programmable GPUs (frameworks *cuDNN*, *Caffe*, *Theano*, *Torch*, *Tensorflow*, etc.).

CNN is one of the best tools in the task of classification especially image classification. Dataset *MNIST* is an example of benchmark that confirms this claim. The following web page² contains error rates of different neural networks

¹ Weighted average in case of convolutional layers. Weights are being learned.

² <http://yann.lecun.com/exdb/mnist/>.

sorted into groups by their types. The best neural network displayed on the website has only 0.23% (23 miss-classifications in 10000) error rate. Another example is competition *ILSVRC* (Large Scale Visual Recognition Challenge). In some cases mean average precisions nearly doubled between 2014³ and 2015⁴.

3 The F-transform of a Higher Degree (F^m -transform)

In this section, we recall the main facts (see [11] 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 FT kernels (FTNet).

3.1 Fuzzy partition

The F-transform is 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 $A_1, \dots, A_n : [a, b] \rightarrow [0, 1]$, 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*

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

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 $k = 2, \dots, n - 1$;

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

³ <http://image-net.org/challenges/LSVRC/2014/results>.

⁴ <http://image-net.org/challenges/LSVRC/2015/results>.

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_h^{tr} , where

$$A^{tr}(x) = 1 - |x|, x \in [-1, 1], \text{ and } A_h^{tr}(x) = 1 - \frac{|x|}{h}, x \in [-h, h].$$

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 $n \geq 2$ and $h = \frac{b-a}{n-1}$ ⁵. Let k be a fixed integer from $\{1, \dots, n\}$, and let $L_2(A_k)$ be a set of square-integrable functions $f : [x_{k-1}, x_{k+1}] \rightarrow \mathbb{R}$. Denote $L_2(A_1, \dots, A_n)$ a set of functions $f : [a, b] \rightarrow \mathbb{R}$ 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

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

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 $P_k^0, P_k^1, P_k^2, \dots, P_k^m$.

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:

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

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

$$P^2(x) = x^2 - \frac{h^2}{6}. \tag{3}$$

We denote $L_2^m(A_k)$ a linear subspace of $L_2(A_k)$ with the basis $P_k^0, P_k^1, P_k^2, \dots, P_k^m$.

⁵ The text of this and the following subsection is a free version of a certain part of [11] where the theory of a higher degree F-transform was introduced.

3.3 F^m -transform

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

Definition 2 [11]. Let $f : [a, b] \rightarrow \mathbb{R}$ be a function from $L_2(A_1, \dots, A_n)$, and let $m \geq 0$ be a fixed integer. Let F_k^m be the k -th orthogonal projection of $f|_{[x_{k-1}, x_{k+1}]}$ on $L_2^m(A_k)$, $k = 1, \dots, n$. We say that the n -tuple (F_1^m, \dots, F_n^m) is an F^m -transform of f with respect to A_1, \dots, A_n , or formally,

$$F^m[f] = (F_1^m, \dots, F_n^m).$$

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

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

$$F_k^m = c_{k,0}P_k^0 + c_{k,1}P_k^1 + \dots + c_{k,m}P_k^m, \tag{4}$$

where

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

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

$$\hat{f}_n^m(x) = \sum_{k=1}^n F_k^m A_k(x), \quad x \in [a, b], \tag{5}$$

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

The following theorem proved in [11] 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 $m \geq 1$. Then

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

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 , $n \geq 2$, generated from $A : [-1, 1] \rightarrow [0, 1]$ 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

$$F^2[f] = (c_{1,0}P_1^0 + c_{1,1}P_1^1 + c_{1,2}P_1^2, \dots, c_{n,0}P_n^0 + c_{n,1}P_n^1 + c_{n,2}P_n^2), \tag{6}$$

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

$$P_k^0(x) = 1, P_k^1(x) = x - x_k, P_k^2(x) = (x - x_k)^2 - I_2, \tag{7}$$

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

$$c_{k,0} = \frac{\int_{-\infty}^{\infty} f(x) A_h(x - x_k) dx}{\int_{-\infty}^{\infty} A_h(x - x_k) dx}, \tag{8}$$

$$c_{k,1} = \frac{\int_{-\infty}^{\infty} f(x)(x - x_k) A_h(x - x_k) dx}{\int_{-\infty}^{\infty} (x - x_k)^2 A_h(x - x_k) dx}, \tag{9}$$

$$c_{k,2} = \frac{\int_{-\infty}^{\infty} f(x)((x - x_k)^2 - I_2) A_h(x - x_k) dx}{\int_{-\infty}^{\infty} ((x - x_k)^2 - I_2)^2 A_h(x - x_k) dx}. \tag{10}$$

In [11, 15], it has been proved that

$$c_{k,0} \approx f(x_k), c_{k,1} \approx f'(x_k), c_{k,2} \approx f''(x_k), \tag{11}$$

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

Without going into technical details, we rewrite (8)–(10) into the following discrete representations

$$c_{k,0} = \sum_{j=1}^l f(j) g_0(ks - j), c_{k,1} = \sum_{j=1}^l f(j) g_1(ks - j), c_{k,2} = \sum_{j=1}^l f(j) g_2(ks - j), \tag{12}$$

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 results of the corresponding discrete convolutions $f \star g_0, f \star g_1, f \star g_2$. Thus, we can rewrite the representation of F^2 in (6) in the following vector form:

$$F^2[f] = ((f \star_s g_0)^T \mathbf{P}^0 + (f \star_s g_1)^T \mathbf{P}^1 + (f \star_s g_2)^T \mathbf{P}^2), \tag{13}$$

where $\mathbf{P}^0, \mathbf{P}^1, \mathbf{P}^2$ are vectors of polynomials with components given in (7), and \star_s means that the convolution is performed with the stride $s, s \geq 1$.

Example 2. We choose the triangular shaped generating function $A^{tr} : [-1, 1] \rightarrow [0, 1]$ and consider it on the discrete domain $D = \{-1, -2/3, -1/3, 0, 1/3, 2/3, 1\}$. Below, we show four matrices $G_0, G_{1,1}, G_{1,2}, G_2$ of 5×5 kernels⁶ that are used for functions of two variables and correspond to the three above considered convolutions with g_0, g_1, g_2 .

$$G_0 = \begin{pmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.062 & 0.125 & 0.062 & 0.000 \\ 0.000 & 0.125 & 0.250 & 0.125 & 0.000 \\ 0.000 & 0.062 & 0.125 & 0.062 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}$$

⁶ The size is determined by only five non-zero values of A^{tr} on D .

$$\begin{aligned}
 G_{1,1} &= \begin{pmatrix} -0.074 & -0.074 & 0. & 0.074 & 0.074 \\ -0.148 & -0.148 & 0. & 0.148 & 0.148 \\ -0.222 & -0.222 & 0. & 0.222 & 0.222 \\ -0.148 & -0.148 & 0. & 0.148 & 0.148 \\ -0.074 & -0.074 & 0. & 0.074 & 0.074 \end{pmatrix} \\
 G_{1,2} &= \begin{pmatrix} -0.074 & -0.148 & -0.222 & -0.148 & -0.074 \\ -0.074 & -0.148 & -0.222 & -0.148 & -0.074 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.074 & 0.148 & 0.222 & 0.148 & 0.074 \\ 0.074 & 0.148 & 0.222 & 0.148 & 0.074 \end{pmatrix} \\
 G_2 &= \begin{pmatrix} 0.062 & 0.049 & -0.037 & 0.049 & 0.062 \\ 0.049 & -0.049 & -0.148 & -0.049 & 0.049 \\ 0.037 & -0.148 & -0.333 & -0.148 & 0.037 \\ 0.049 & -0.049 & -0.148 & -0.049 & 0.049 \\ 0.062 & 0.049 & -0.037 & 0.049 & 0.062 \end{pmatrix}
 \end{aligned}$$

Let us remark that in the context of convolutional neural networks, matrices G_0 , $G_{1,1}$, $G_{1,2}$ G_2 determine convolution filters. In the context of the F-transform, they depend on the chosen partition of underlying universe and do not depend on the functions they are applied to.

3.5 F^2 -transform in the FTNet Architecture

We propose to modify the LENET-5 [12] and replace convolution-type units in the first and third convolution layers C_1 and C_3 by the similar units which realize the computation of the F^2 -transform coefficients according to (12) and adapted to functions of two variables. We use the meaning (11) of the F^2 coefficients and specify features in the feature maps of the convolution layer C_1 as partial derivatives (positive and negative) of an input function (of two variables) with respect to each single variable up to the second degree. In more details, the six matrices G_0 , $-G_0$ $G_{1,1}$, $G_{1,2}$, G_2 , $-G_2$ are convolved with the input 2D image in order to produce the mentioned partial derivatives at uniformly distributed nodes over the image domain. Thus, we have six feature maps in C_1 . Each feature map of C_1 is connected (via subsampling layer S_2) with each feature map of C_3 - thus, we have thirty six feature maps in C_3 . The meaning of feature in C_3 corresponds to all possible mixed partial derivatives up to the third degree.

The features extracted in C_1 and C_3 are used to classify objects in MNIST. Our justification is based on the Theorem 1 which says that the inverse F^m (particularly, F^2) transform approximates any function with sufficient quality. The number 2 of convolutional layers was set up empirically, and this turned out to be sufficient for the recognition purpose from MNIST. Thus, in comparison with the LENET-5 we use less number of convolutional layers. Other layers in the FTNet are of the fully-connected types and serve the same purposes as in the LENET-5.

Let us discuss the learning of convolution filters represented by matrices G_0 , $-G_0$ $G_{1,1}$, $G_{1,2}$, G_2 , $-G_2$. These filters can be excluded from the learning

procedure on the basis of the mentioned above Theorem regarding the universal approximation. However, if they are learned (and this is confirmed by our tests), then the quality of approximation is adapted to a narrower class of objects (they constitute a certain dataset) and it is better than in the general case (valid for a generic class of objects).

4 Experiments and Results

We have used the FTNet (based of the LENET-5 [12]) architecture as the baseline for our experiments. The details of the FTNet architecture are described in Table 1 and Sect. 4.1 where the following notation is used: convolution layers C_1 and C_3 , subsampling layers S_2 and S_4 and fully connected layers FC_5 and FC_6 . We have examined the impact of the following hyper-parameters: convolution kernel size \mathcal{D} , presence and type of the subsampling \mathcal{S} , layer weights trainability \mathcal{T} , and a form of the layer weights initialization \mathcal{I} on the network performance. We have used all possible combinations of the hyper-parameters of C_1 , S_2 , C_3 and S_4 in the *grid search* with the purpose to select the optimal setting with respect to the quality of recognition (loss function).

Table 1. FT-Net architecture.

Hyper-paramater	C_1	S_2	C_3	S_4	FC_5	FC_6
Kernel size	5×5	-	5×5	-	-	-
# feature maps	6	-	36	-	-	-
Stride	1×1	<i>pooling</i>	1×1	<i>pooling</i>	-	-
Pooling size	-	2×2	-	2×2	-	
# FC units	-	-	-	-	500	10

4.1 FTNet Architecture

In this section, we describe details of one particular FTNet architecture where the hyper-parameters are: $\mathcal{D} = 5 \times 5$, $\mathcal{S} = \text{max pooling}$. Below, we characterize other details: layers, connection types, input, intermediate and output objects.

The first layer of the FTNet is convolutional C_1 , it has 6 feature maps with the size of 28×28 . To ensure the same size of the feature maps and image, padding is used. Each C_1 unit has 25 connections to input image. Unit connections are spatially close, forming 5×5 neighborhood called unit's *receptive field*; the latter overlaps with others unit's receptive fields. The C_1 has $25 \cdot 6 + 6$ (trainable) parameters and $784 \cdot 25 \cdot 6 + 6$ connections.

The C_1 feature maps are connected to the max pooling layer S_2 . The S_2 units have 2×2 non-overlapping receptive fields from which they select the maximum. S_2 effectively decreases the feature maps size to 14×14 . The S_2 has $784 \cdot 6$ connections.

The convolutional layer C_3 is connected to S_2 outputs. Each 14×14 output is convolved with all C_3 kernels (they are the same as in C_1) creating 6^2 new feature maps with the size of 14×14 . The C_3 has $25 \cdot 6^2 + 6^2$ (trainable) parameters and $784/4 \cdot 25 \cdot 6^2 + 6^2$ connections.

The C_3 feature maps are connected to the max pooling layer S_4 that further reduces their size to 7×7 .

The S_4 feature maps are inputs to the fully connected layer FC_5 . The FC_5 has 500 units, each connected to all outputs from S_4 , therefore the FC_5 has $7 \cdot 7 \cdot 6^2 \cdot 500$ (trainable) parameters/connections.

The FC_5 output vector is the input to the last fully connected layer FC_6 that has 10 units. The FC_6 has $500 \cdot 10$ (trainable) parameters/connections. The FC_6 output vector goes through the softmax layer. Softmax layer normalizes an input vector to that whose sum of components is equal to 1.

The C_1 , C_3 and FC_5 uses Rectified Linear activation function (RELU) [2].

4.2 Tests

The proposed network was tested on grayscale images from the database MNIST. The MNIST consists of 70000 28×28 images⁷. They were normalized to the size of 20×20 so that the centering and the color ratio of the original 28×28 images were preserved.

Two sets of kernels were selected for testing. They determine feature maps in layers C_1 and C_3 . The first set (referred to as “FT2”) is composed by the F^2 -transform kernels represented above by the six matrices G_0 , $-G_0$, $G_{1,1}$, $G_{1,2}$, G_2 , $-G_2$. The second set (referred to as “Conventional”) is composed by the widely used kernels with the same meaning as the F^2 -transform ones: they specify partial derivatives (positive and negative) of an image function with respect to each single variable up to the second degree. These kernels are: Gauss, Sobel, Laplace and their derivatives such as -Gauss (multiplied by -1), 90Sobel (rotated by 90°) and -Laplace.

Our first test was focused on the choice of an optimal combination with respect to the chosen loss function – the *cross entropy*. With this purpose, we have applied a grid search over all combinations of the hyper-parameters (\mathcal{D} , \mathcal{S} , \mathcal{T} and \mathcal{I}).

The results were clustered into 3 groups based on the following values of the loss function: $\{\approx 1.5, \approx 6, \approx 25\}$. We have observed that the clustering (and by this, the quality of recognition) essentially depends on the presence of subsampling. In more details:

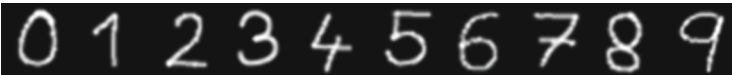












Fig. 2. Numbers used for testing rotation invariance.

⁷ 60000 training and 10000 testing.

Table 2. Selected optimal networks, their parameters and performance.

Kernel set	FT2	FT2	Conventional	Conventional	Conventional	LENET-5 ^a
C_1 kernel size	5×5	5×5	5×5	5×5	5×5	5×5
C_1 train	true	false	true	false	true	true
C_1 init.	fixed	fixed	fixed	fixed	fixed	random
S_2 subs.	stride	pooling	pooling	pooling	pooling	pooling
C_3 kernel size	5×5	5×5	5×5	3×3	5×5	5×5
C_3 train	true	false	true	false	true	true
C_3 init.	fixed	random	random	fixed	fixed	random
S_4 subs.	pooling	pooling	pooling	stride	pooling	pooling
Training time	34.7s	34.8s	39.9s	26.3s	41.2s	82.6s
Avg. loss	0.9037	0.9487	0.9039	0.9341	0.9008	1.56
	none	none	none	none	none	none
	[40 - 160] [215 - 345]	[30 - 160] [215 - 345]	[30 - 125] [215 - 345]	[35 - 130] [230 - 345]	[35 - 155] [220 - 345]	[35 - 125] [225 - 350]
	[50 - 80] [145 - 275]	[40 - 270]	[50 - 90] [130 - 285]	[45 - 170] [200 - 275]	[45 - 175] [210 - 265]	[50 - 100] [130 - 185] [210 - 275]
	[70 - 305]	[65 - 310]	[45 - 305]	[70 - 310]	[70 - 300]	[50 - 300]
	[25 - 340]	[35 - 50] [85 - 200] [240 - 340]	[15 - 185] [230 - 335]	[25 - 195] [235 - 340]	[35 - 205] [230 - 335]	[20 - 195] [230 - 340]
	[80 - 120] [180 - 305]	[50 - 135] [180 - 245] [285 - 300]	[50 - 115] [190 - 295]	[70 - 255] [275 - 300]	[75 - 120] [180 - 310]	[75 - 115] [180 - 315]
	[0 - 5] [85 - 355]	[0 - 30] [80 - 355]	[0 - 25] [95 - 355]	[0 - 15] [90 - 355]	[0 - 10] [85 - 355]	[85 - 350]
	[0 - 0] [40 - 355]	[0 - 0] [40 - 355]	[35 - 350]	[0 - 10] [35 - 95] [115 - 355]	[30 - 350]	[35 - 350]
	[45 - 140] [235 - 270]	[0 - 140] [245 - 285] [315 - 355]	[50 - 140] [215 - 285] [320 - 345]	[0 - 150] [220 - 285]	[10 - 25] [50 - 140] [240 - 325]	[45 - 145] [230 - 290]
	[55 - 320]	[0 - 355]	[30 - 30] [50 - 330]	[0 - 25] [60 - 355]	[45 - 355]	[60 - 325]

^a LENET-5 like network with 32 feature maps in C_1 and 64 feature maps in C_3 .

- in the group with the loss value ≈ 1.5 , the subsampling (in the form of the max pooling or stride) accompanies both convolutional layers;
- in the group with the loss value ≈ 6 , the subsampling was applied in combination with exactly one convolutional layer;
- in the group with the loss value ≈ 25 , the subsampling was not applied in combination convolutional layers.

Our second test was focused on the invariance of recognition by the FTNet with respect to rotation. On the basis of the first test, we selected several FTNet configurations for the analysis of the rotation invariance:

1. the one with the (absolute) lowest loss value,
2. the one with the lowest loss value among those with fixed kernels (FT2 or Conventional) in C_1 and C_3 layers,
3. the one with the lowest loss value among those with trainable kernels (FT2 or Conventional) in C_1 and C_3 layers.

It is worth noting that the configuration with the absolute lowest loss value (item 1) coincides with the one described in item 3 with the FT2 kernels.

For the purpose of testing, we have created 10 input images (manually) (Fig. 2) and rotated them from 0° up to 355° with the step 5° . All selected FTNets were trained for 10 epochs. In the bottom part of Table 2, we show those angle interval(s) where the network top prediction was not correct.

We have accomplished accuracy of 99.23% on MNIST which is competitive result (see LeCun MNIST web-page). This accuracy was achieved without any distortions on training set.

5 Conclusion

In this contribution, we have introduced a new convolutional neural network – the FTNet. In its two convolutional layers, the FTNet uses fixed kernels extracted from the discrete version of the F^2 -transform. Moreover, it has a certain number of trainable hyper-parameters. The MNIST database was used for the FTNet training. The results were compared with the LENET-5 like network. The tests have shown that the best FTNet configuration performs significantly better recognition (according to the training time and the loss function value) than the LENET-5 like network.

We have also analyzed the FTNet and LENET-5 rotation invariance. We came to the conclusion that all tested networks and their best configurations show similar results. In particular, the rotation invariance was demonstrated only for relatively small angles within the interval $[0^\circ, 30^\circ]$.

Last, but not least, we provided theoretical justification of a suitability of the FTNet for the problem of recognition.

References

1. Schmidhuber, J.: *Neural Netw.* **61**, 85 (2015)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks, In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc. (2012)
3. Szegedy, C., Ioffe, S., Vanhoucke, V.: CoRR abs/1602.07261 (2016)
4. He, K., Zhang, X., Ren, S., Sun, J.: CoRR abs/1512.03385 (2015)
5. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014
6. Werbos, P.J.: *Beyond regression: new tools for prediction and analysis in the behavioral sciences*, Ph.D. thesis, Harvard University. Department of Applied Mathematics (1974)
7. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: *Cogn. Model.* **5**, 1 (1988)
8. Lowe, D.G.: Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision* (1999)
9. McConnell, R.: *Method of and apparatus for pattern recognition*, 28 January 1986. US Patent 4,567,610
10. Maini, R., Aggarwal, H.: *Int. J. Image Process. (IJIP)* **3**, 1 (2009)
11. Perfilieva, I., Danková, M., Bede, B.: Towards f-transform of a higher degree. In: *IFSA/EUSFLAT Conference* (2009)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: *Proc. IEEE* **86**, 2278 (1998)
13. Fukushima, K.: *Biol. Cybern.* **36**, 193 (1980)
14. Williams, D.R.G.H.R., Hinton, G.: *Nature* **323**, 533 (1986)
15. Perfilieva, I., Kreinovich, V.: *Fuzzy Sets Syst.* **180**, 55 (2011)