Applied Soft Computing Journal 88 (2020) 105960

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Maintaining filter structure: A Gabor-based convolutional neural network for image analysis

Somayeh Molaei [a,b], Mohammad Ebrahim Shiri Ahmad Abadi [b,*]

[a] Department of Computer Science and Engineering, University of Michigan, Ann Arbor, MI 48105, United States of America
[b] Department of Computer Science, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 12 June 2018
Received in revised form 14 October 2019
Accepted 26 November 2019
Available online 10 December 2019

Keywords:
Convolutional neural networks
Image segmentation
Deep learning
Gabor filter

ABSTRACT

In image segmentation and classification tasks, utilizing filters based on the target object improves performance and requires less training data. We use the Gabor filter as initialization to gain more discriminative power. Considering the mechanism of the error backpropagation procedure to learn the data, after a few updates, filters will lose their initial structure. In this paper, we modify the updating rule in Gradient Descent to maintain the properties of Gabor filters. We use the Left Ventricle (LV) segmentation task and handwritten digit classification task to evaluate our proposed method. We compare Gabor initialization with random initialization and transfer learning initialization using convolutional autoencoders and convolutional networks. We experimented with noisy data and we reduced the amount of training data to compare how different methods of initialization can deal with these matters. The results show that the pixel predictions for the segmentation task are highly correlated with the ground truth. In the classification task, in addition to Gabor and random initialization, we initialized the network using pre-trained weights obtained from a convolutional Autoencoder using two different data sets and pre-trained weights obtained from a convolutional neural network. The experiments confirm the out-performance of Gabor filters comparing to the other initialization method even when using noisy inputs and a lesser amount of training data.

1. Introduction

Recently, deep convolutional neural networks have shown high performance in many different domains including speech analysis and machine vision [1–3]. The networks are able to learn very complex and large models without requiring many human design decisions [4,5]. However, achieving these valuable properties comes at a cost; the networks are very computationally demanding due to the high number of parameters and a large training set is required to extract enough information for parameter optimization [6]. This over-parameterization, however, seems to help in approximating the solution of highly non-convex optimization problems [7]. In recent years there have been several attempts to speed up CNNs by reducing the number of parameters [7–10]. These methods successfully reduce the number of parameters while attempting to maintain the performance and accuracy of the trained model, yet we believe there is still the need to make these complex and large models tolerant to smaller training sets. One practical way to do this is to guide the training process in such a way that useful properties of the parameters are maintained during training. Using this method, the number of computational steps needed to optimize the parameters is reduced. In this paper, we utilize the properties of Gabor filters to capture more information in the image that leads into extracting more discriminating features from the image. The generated filter bank covers many different modifications to the original Gabor filter by changing the Gabor parameters, effectively controlling the shape of the filters. Then, the structure of the Gabor filters are maintained during the training process. We achieve this by modifying the updating rule in CNN to determine the contribution of each Gabor parameter at each pixel we wish to classify. This work is a continuation of our previous work [11] where we showed that initializing a CNN with a Gabor filter bank outperforms the same network structure when initialized with a random filter bank on the segmentation task. In that work we trained the model using the regular Gradient Descent algorithm. Therefore, the properties of the Gabor filter bank disappeared after training the model with a few batches of data. In this paper, we propose a novel way of keeping the structure of Gabor filter bank along with comparing the performance of Gabor filters with other methods of initialization including random initialization and transfer learning initialization. To the best of our knowledge, there is no previous study that includes filter structure maintenance during

* Corresponding author.
E-mail addresses: smolaei@umich.edu (S. Molaei), shiri@aut.ac.ir (M.E. Shiri Ahmad Abadi).

https://doi.org/10.1016/j.asoc.2019.105960
1568-4946/

the training process. Also, this is the first time that all the parameters of Gabor filters controlling different aspects are being used to generate a filter bank. Additionally, in this study, we report the results of our method on the classification task as well. Our proposed approach can be used with all the compression methods mentioned above to make CNNs applicable to an extended range of tasks with smaller training sets or when the input data is noisy. The proposed modification is easy to implement in any of the current deep learning frameworks with small modifications in the updating rule. In this paper, we propose a DCNN to segment the LV wall in MRI images and simpler CNN structure to classify handwritten digits.

The motivation behind using Gabor filters is inspired by previous studies showing promising properties of Gabor filters in different vision tasks. Gabor filters are capable of providing an accurate description of most spatial characteristics of simple receptive fields [12,13]. They are shown to be both orientation and scale-invariant, as well as noise-resistant [14]. The 2D Gabor filter's inherent flexibility due to having different free parameters controlling the shape of the filter will grant advantages on the system that employs it [13]. Different parameters of the Gabor changing the appearance of the filter can easily be seen as the changes in the appearance of the LV wall in different images. Although preserving filter characteristics during model training will impose an extra constraint to the CNN model, the variance of the model will be dramatically reduced with this procedure. Thus, a lesser amount of training data is required to build the model. Additionally, compared to standard CNN, Gabor kernel CNN reduces the computations by having fewer parameters to be trained. Another motivation is the noise-resistant property of the Gabor filters [14]. The segmentation task is inherently noisy due to the difference in the intensity of the LV wall and surrounding structures (e.g. lung and liver), low tissue contrast between LV and blood, and low contrast between fat and LV wall in cases where the heart is surrounded by fat. The volume overlapping affect by the liver causes more noise, motion artifact and blurry edges. Gabor filters have been shown to be effective in detecting edges [15]. To show the noise-resistant property of Gabor filters in the image classification task we added the same artificial noise to different initialization methods to make a fair comparison.

1.1. Image segmentation

LV segmentation is crucial for cardiac function analysis. Most of the current LV segmentation methods are based on Active Contour and Region Growing approaches. These approaches involve manually setting the initial contour or selecting a proper slice, defining a threshold, transferring the image to polar coordinates, and separately segmenting the epicardium and endocardium contours. In such a complex procedure, the accuracy of each step directly affects the final accuracy. Thus, a fully automated LV segmentation method is desirable.

There are several challenges in cardiac MRI segmentation. In a short-axis MRI image, the blood pools of the right and left ventricle appear bright while the surrounding structures, such as the myocardium, lung and liver, appear darker. First, because of the surrounding tissues, there is low tissue contrast between myocardium and blood pool. In some cases physicians incorrectly include other muscles in the blood pool, resulting in a larger endocardial border. Second, due to noise, such as motion artifacts and volume overlap from the liver, the epicardium is more difficult to detect. Third, the myocardium splits around the left and the right ventricle. Finally, when the heart is surrounded by fat, the edges between fat and myocardium are much weaker than the edges separating fat and lungs.

Initially, we segment the LV wall using a deep convolutional neural network. Following this implementation, we incorporate adjustments to the network, including the region of interest extraction, filter design and updating procedure modification to obtain better results.

1.2. Image classification

To show the effectiveness of the Gabor filters, we have experimented with the classification task with various initialization methods, including pre-trained weights obtained from convolutional Autoencoders and pre-trained weights obtained from a Convolutional Neural Network.

1.3. Related work

In this section, we review the related research in two main categories: the works related to the segmentation task and the works related to the classification task. In the works related to the segmentation task, we first review the studies related to segment the LV, then we present the works that use CNN for the image segmentation task in general. For the classification related work, we mainly focused on the studies around pre-training weights to boost the performance of CNN for the classification tasks.

Segmentation task: Several approaches have been proposed for automatic segmentation of cardiac structures. Image processing-based methods include thresholding, region growing, connected component analysis, [16–18], pixel classification methods such as clustering, [19–21], deformable models such as active contour models, model-based methods such as Active Shape Model (ASM), Active Appearance Model (AAM), and statistical models [22–25], as well as the atlas guided method [26].

More recently, the DCNN has been applied to segment different objects in various image datasets. The authors of [27] apply a DCNN to segment biological images to map brain structure. In [28], a CNN which operates on large scale input windows is used to perform segmentation. The authors of [29] utilize CNN to perform the task of simultaneous detection and segmentation, which requires that the network both segment and determine bounding boxes for objects within an image. The same task, i.e. localization and segmentation, is performed in [30], along with supervised pretraining to compensate for scarce training data. A combination of DCNN and nearest neighbor has been used to segment natural edges or thin objects [31]. In 2016, the V-net was constructed as a CNN for volumetric medical image segmentation [32]. This work augmented sparse MRI training data sets via non-linear transformations and attempted to reduce bias due to class imbalance.

Ultimately, in 2015 it was shown that the CNN outperforms alternative semantic segmentation algorithms, adapting state of the art classification networks to the segmentation task on popular datasets [33] to perform the semantic segmentation and scene parsing tasks, exploring PASCAL VOC, NYUDv2, and SIFT Flow. In 2018, DCNNs have set the record for semantic segmentation tasks, with adjustments being made to improve aspects such as localization [34]. It is clear that CNN is well suited for the image segmentation task. In addition to image segmentation, the CNN has proven to be successful in many other image processing fields including generic feature extraction [35], object detection [1,2] and fast image scanning [36]. The study in [36] has been evaluated on the ISBI Electron Microscopy Segmentation Challenge. None of the aforementioned CNN projects has implemented the modified updating rule proposed in this work, for either the classification or segmentation tasks. Additionally, they evaluated their work on different datasets.

Classification task: Convolutional Autoencoders have been used in some studies as a pre-train step to capture some characteristics in the data to boost the performance. In [37] they

proposed a system that employs a deep 3D convolutional neural network (3D-CNN) pre-trained by 3D Convolutional Autoencoder (3D-CAE) to learn generic discriminative Alzheimer's disease.

In [38], to boost the classification performance, they utilized Sparse Autoencoder(SAE) to discover a latent feature-representation from the low-level features in MRI, PET, and CSF independently and then they trained the CNN on random patches of natural images. [39] applied SAE to extract features for the Synthetic aperture radar (SAR) image classification task. [40–43] are other examples of using autoencoder to extract more informative features as a pre-training step for the classification task.

Gabor Filters: The Gabor function has been recognized as a very useful tool in computer vision and image processing. Due to its optimal localization properties in both spatial and frequency domain, it is able to combine the advantages of both filters. Gabor filters are capable of providing an accurate description of most spatial characteristics of simple receptive fields [12,13]. Because these filters optimize simultaneous resolution in the spectral and spatial domain, they minimize the number of filters (cells) required to represent these two aspects of the information content of images [13]. The systems that employ 2D Gabor filters will benefit from them due to their inherent flexibility [13]. Other advantages of Gabor filters are as follows: Gabor features are well motivated and mathematically well-defined; they are easy to understand, fine-tune and implement; and they have also been found less sensitive to noises, a small range of translation, rotation, and scaling [14]. The set of free parameters in Gabor filters makes them a remarkably flexible functional form. Different researchers have utilized this flexibility to define and tune Gabor filter banks suitable for different tasks in vision, including but not limited to : edge detection [15,44], image coding [12], texture analysis [45–47], handwritten number recognition [14,48], face recognition [49], vehicle detection [50–52], face detection [49], object detection [53], text segmentation [54], speech analysis [55] and fingerprint enhancement [56]. None of the studies utilizing Gabor filters has attempted to preserve the Gabor filters while updating the filters.

The paper is organized as follows: Section 2 provides an overview of experimental details including extracting the region of interest, class balancing mechanism, Gabor filter bank generation and the procedure to maintain Gabor properties. Section 3 provides the results of our experiments for both segmentation and classification tasks with and without noise introduction in input data. Section 4 provides the conclusion.

2. Fully convolutional neural networks

In this paper, automatic segmentation of the LV wall (e.g. the epicardium and the endocardium contours) is performed simultaneously using a DCNN. We generate a binary label image corresponding to each input image, which assigns the LV wall pixels a label of 1 and all other pixels a label of 0.

For the base neural network architecture, we used MatConvNet, an open-source library in MATLAB, detailed in [57]. The basic DCNN architecture is reviewed in Section 2.1; convolutional layers consisting of smaller filters, ReLu layers and pooling layers are applied within the network. Finally, a fully connected layer is applied to calculate, per pixel, the probability that the pixel belongs to the LV wall as opposed to the background. Based on these probabilities we label each pixel. Note that there is no manual initialization required for this method.

Throughout the segmentation experiment, we iteratively apply several modifications tailored to the Left Ventricle segmentation task. We first describe the base architecture, then describe any modifications. Specifically, Section 2.2 details our technique to reduce classification bias based on pixel distribution, Section 2.3 details random and Gabor filter initialization, and Section 2.4 details our new filter updating procedure for backpropagation that is designed to maintain filter structure during network training.

2.1. Segmentation architecture

In this network, each layer is a 3-dimensional array of size $s_1 \times s_2 \times f$ (width, height and depth, respectively). In each layer, f denotes the channel dimension while s_1 and s_2 are the spatial dimensions of the data. The input layer takes an additional parameter to indicate whether the image is Red–Green–Blue or grayscale.

The network structure for ImageNet provided in MatConvNet contains 31 layers that alternate between convolution, ReLU and pooling. This includes 13 convolutional layers, each equipped with an increasing number of filters. After each of these layers is a ReLU layer. These layers are followed by an additional 8 computational layers, alternating between fully connected, ReLU, dropout and deconvolution. The 40th and final layer provides the segmentation prediction image. The full segmentation architecture can be seen in Fig. 1.

All pooling layers, on input matrix x provided output y, performed max pooling with a 2×2 window and stride 2:

$$y_{i,j,d} = \max_{1 \le i' \le x_2, 1 \le j \le s_1} x_{i+i'-1, j+j'-1, d} \tag{1}$$

The ReLU layers employed the rectifier activation function:

$$y_{i,j,d} = \max\{0, x_{i,j,d}\} \tag{2}$$

The convolutional, dropout, fully connected and deconvolutional layers remain unchanged from the default implementation of MatConvNet. The network is trained using the backpropagation algorithm, with the softmax log-loss function, L, which for prediction segmentation X, ground truth segmentation c, with q ranging over the images:

$$L(X, c) = -\log \frac{\exp(X(c))}{\sum_q \exp(X(q))} \tag{3}$$

MatConvNet implements three types of normalization techniques during image segmentation. These normalization methods are called batch normalization, special normalization and local response normalization, the details of which can be found in the MatConvNet literature.

Our first set of image segmentation results, detailed in Section 3, performs Left Ventricle segmentation using the DCNN outlined above.

2.2. Unbalanced pixel distribution

In subsequent experiments, we pre-process the images to combat classification bias prior to training the network. We also utilize cost-sensitive segmentation to reduce bias. The details of these methods and the network architecture are outlined below.

In the problem at hand, the ratio between pixels of interest and the pixel size of the image is very small. Therefore the learning algorithm is biased to predict that a given pixel is a member of the background. To reduce this bias towards the background and assist the algorithm in proper and timely labeling of pixels, we apply two steps:

1. Extracting the Region of Interest (ROI) to increase the pixel ratio and reduce computational complexity.
2. Class Balancing, incorporating global dataset information, to weight the image by the relative frequencies of object and background pixels.

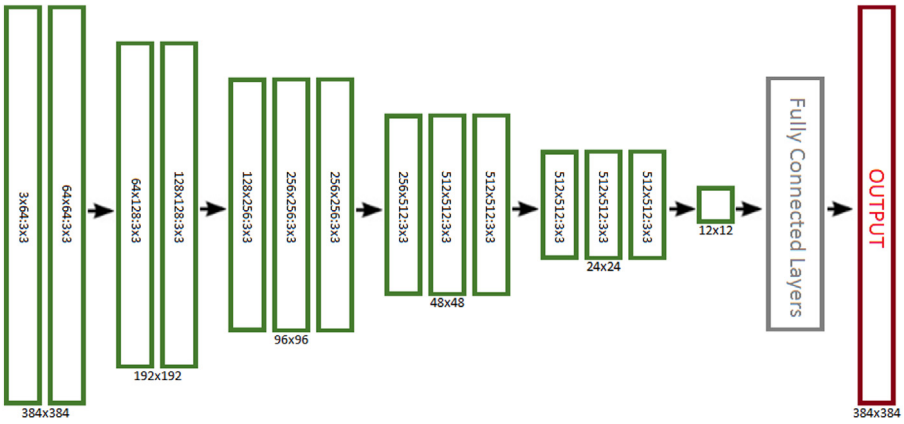

Fig. 1. MatConvNet ImageNet: each convolutional layer of the network is depicted. Inside the layer aaxbb:cxc indicates that aaxbb filters are used in the layer, each filter of size cxc. Below the layer is the input size to that layer. The arrows denote pooling layers, therefore the input size decreases accordingly.

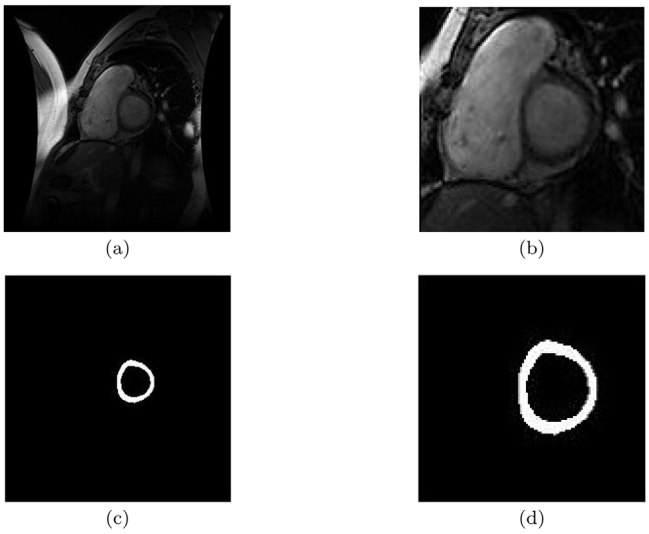

Fig. 2. (a) Original input image before ROI extraction. (b) Original image after extracting ROI. (c) Label image before ROI extraction. (d) Label image after extracting ROI.

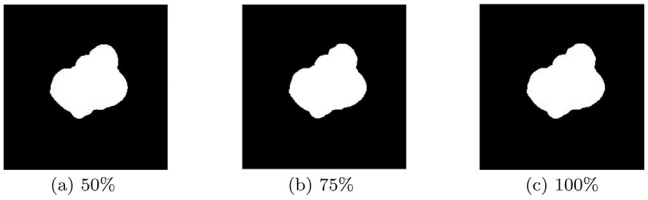

Fig. 3. Label-sum image on 50% and 75% randomly selected, followed by 100% of the images, to find the smallest region of interest.

2.2.1. Extracting the ROI

Reducing the amount of background in an image greatly improves classification accuracy and computational cost. In simulations conducted by Karianakis et al. [58], the authors propose an adaptive sampling method, a data-driven object proposal algorithm, which reduces the top_1 and top_5 errors on the ImageNet classification challenge [59]. Extracting the ROI reduces unnecessary computation by eliminating the need to segment sections of the image that a priori do not contain object pixels.

To extract the ROI, defined as the smallest window containing the object boundary in all images of the dataset, we provide a binary label mask based on 64 annotated points. The label mask has the same size as the image 256×256. We then sum all binary label mask images to calculate a single 256×256 image. In this final label-sum image value of 0 indicates that a pixel is not part of the object in any image. Next, we look at the label-sum image to determine the smallest rectangular window which contains all pixels with a non-zero value. We crop all of the original images to the collective ROI.

To validate the robustness of ROI-finding method we extracted the ROI on different subsets of images, randomly sampled with rates of 50% and 75%. The extracted window did not vary amongst these subsets, thus illustrating the correctness of the method. The summation images are shown in Fig. 3. Additionally, Fig. 2 contains examples of the original image alongside the corresponding label image both before and after ROI extraction.

2.2.2. Class balancing

In the ROI the number of pixels that belong to the LV wall remains significantly smaller than the number of pixels in the background, leading to continued classification bias. To increase the pixel balance we compute a matrix of weights, w, calculated using both global and local information from the labels and dataset.

For the entire dataset, the *summed label image* contains the global information that is incorporated into the weight matrix. The global information matrix, G, is a matrix with the same dimensions as the images and labels. Each element (i, j) of the global information matrix G is valued as the number of times pixel (i, j) has been an object image. For example, if the pixel 1,1 is always a background pixel across all labels then G will have the value 0 at $(1, 1)$, if the pixel $(50, 50)$ is an object pixel in 40 images, then the value of G at element $(50, 50)$ is 40. This is calculated for all 256x256 elements of the matrix G.

For each label image L_k, we define $f_{k,b}$ and $f_{k,o}$ to be the frequency of background and object pixels in the image, respectively. Then, we compute the weight matrix w for *each* input image, indexed by k, via the following algorithm:

In step (4) we can see that w', as calculated, only has non-zero elements at the indices (i, j) that are object pixels in L_k. Those indices take the value of G at that element, which is the number of times the pixel has been an object pixel throughout the entire dataset.

This final weight matrix w, for the training set, is applied during the calculation of the loss function and penalizes the network for misclassification of object pixels much more than misclassification of background pixels.

Algorithm 1: Class Balancing.

Input: Set of Images, L
Output: Weight matrix w

1: **for** Label Image L_k **do**
2: Calculate pixel frequencies $f_{k,o}$, $f_{k,b}$, as the raw count of object pixels and background pixels in the label image
3: Generate a frequency matrix F^k, of the same dimension as the label image, element by element as follows:

$$F^k_{(i,j)} = \begin{cases} f_{k,o}, & \text{if } (i,j) \text{ is an object pixel of } L_k \\ f_{k,b}, & \text{if } (i,j) \text{ is a background pixel of } L_k \end{cases} \quad (4)$$

4: Calculate weight matrix w', element by element, as a weighted element-wise product of L_k and G

$$w'_{i,j} = F^k_{(i,j)} \cdot (L_{k,(i,j)} \times G_{(i,j)}) \quad (5)$$

5: Normalize w' so that all non-zero elements are in the interval [3, 6] and all zero elements are valued at $f_{k,b}$, to create the weight matrix w

2.3. Gabor filter initialization

For Segmentation task, throughout the experiments, we utilized two distinct filter initialization methods: random initialization and Gabor initialization.

In random initialization, each entry of every filter in all layers is a number chosen uniformly at random from the interval [−0.3, 0.3]. This results in a set of completely random matrices as filters in the DCNN filter bank for all convolutional layers as well as fully connected layers.

In Gabor initialization, each filter in the filter bank for all convolutional layers is generated with a Gabor structure. We generate a Gabor filter bank based on the closed-form parameterization of Gabor filters detailed in [60] and outlined in Eq. (6).

The filter bank generator receives the filter dimensions in each layer and generates the filter bank with the same dimension as output. Let $[M, N, x, y]$ be the layer's input dimension, where $M \times N$ is the number of filters in each layer and x, y indicate the size of each filter. We generate the filter bank as follows:

The first step is to determine how many times to change the value of each parameter by calculating factors of the layer dimension $(M \times N)$ in an attempt to have a wide range of Gabor filters. Since M and N are factors of 2, we can compute how many times factor 2 appeared in $M \times N$ and decide how many times to change each parameter based on that. The Euler-phi totient function [61], utilized to determine factors of a large number, was employed to calculate the possible values for each of the five parameters. For example, the prime factors for a layer with dimension 64 × 128 are $64 \times 128 = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]$. This is what line 2 of the algorithm 2 returns. Since there are five parameters, we need to generate the five largest factors out of this list. The result of *ExtractFactor()* in line 4 is [8, 8, 4, 4, 8] for each parameter. Through lines 7 to 13 a list of parameter values for γ, σ and ϕ is generated randomly. The two remaining numbers are utilized to generate a list of values for f and θ. Finally, the algorithm loops through all 5 parameters and for each combination of parameters creates a matrix of size x, y using Gabor closed-form. The closed-form Gabor parameterization is as follows, where x and y are the indices of a Gabor matrix G:

$$G(x, y; f, \gamma, \sigma, \theta, \phi) = \frac{f^2}{\pi\gamma\eta} \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \exp(j2\pi f x' + \phi) \quad (6)$$

$$x' = x\cos\theta + y\sin\theta \quad (7)$$

Algorithm 2: Filter Bank Generation.

Input: Layer's input dimension, $[M, N, x, y]$
Output: Filter bank with the same dimension $[M, N, x, y]$

1: totalNumberOfFilters = $M \times N$
2: F = *list*(*primeFactors*(*totalNumberOfFilters*))
3: **if** (*Length*(*F*) > 5) **then**
4: $[u, v, p, q, r] = ExtractFactor(F)$
5: **else**
6: $[u, v, p, q, r] = F$
7: gamma, sigmma, phi = [ ], [ ], [ ]
8: **for** i in range (p) **do**
9: $gamma.append(rand + \sqrt{2})$
10: **for** i in range (q) **do**
11: $sigmma.append(rand + \sqrt{2})$
12: **for** i in range (r) **do**
13: $phi.append((rand \times \pi) + \pi)$
14: $f_{max} = 0.25$
15: $\eta = \sqrt{2}$
16: $filterBank = []$
17: **for** $i1$ in range (u) **do**
18: $f = \frac{f_{max}}{(\sqrt{2})^{i1}}$
19: **for** $i2$ in range (v) **do**
20: $\theta = \frac{i2}{v} \times \pi$
21: **for** $i3$ in range (p) **do**
22: $\gamma = gamma[i3]$
23: **for** $i4$ in range (q) **do**
24: $\sigma = sigmma[i4]$
25: **for** $i5$ in range (r) **do**
26: $\phi = phi[i5]$
27: $x' = x\cos\theta + y\sin\theta$
28: $y' = -x\sin\theta + y\cos\theta$
29: $filter = \frac{f^2}{\pi\gamma\eta} \exp(-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}) \exp(j2\pi f x' + \phi)$
30: $filterBank.append(filter)$
31: **Return** *filterBank*

$$y' = -x\sin\theta + y\cos\theta \quad (8)$$

As shown in algorithm 2, to generate the Gabor filter bank we incremented all five essential parameters $(f, \gamma, \sigma, \theta, \phi)$ in the 2-dimensional Gabor construction. Where θ is the amount of rotation, ϕ is the phase offset, f is the frequency of sinusoid, γ is spatial aspect ratio (i.e. the amount of ellipticity of the support of the Gabor function) and σ is the standard deviation of the Gaussian envelope. Then we normalized all the filters in the range [−0.3, 0.3] to avoid comparatively large numbers after applying the convolution operator.

Table 1 summarizes the number of times each parameter has been changed for different input dimension. An example of parameter values for one filter for each layer is also presented. Although carrying 5 for loops over 5 parameters might seem expensive, initialization is a one time task in the approach. It is worth mentioning that even for the largest filter dimension of 512 × 512, it takes only 4 s to produce 262, 144 filters. To the best of our knowledge, most of the networks used in real image applications usually do not go beyond 512 × 512. All other layers up to dimension 256 × 256 take at most 1 s to generate the filters.

Examples of Gabor filters can be seen in Fig. 4 which shows the filter structure and variations in the parameters. Variation in individual parameters while all other parameters are fixed can be seen in Figs. 5, 6, 7, and 8. An interested reader can easily produce and visualize all the filters for each layer using algorithm 2.

Table 1
Filter bank generation. For example, in order to generate 64 × 64 filters, we vary f 8 times, θ 8 times, γ 4 times, σ 4 times and ϕ 4 times. In each row, vector $[f, \theta, \gamma, \sigma, \phi]$ indicates the value of each parameter to generate one filter as an example. Time represents computation time in seconds.

Dimension	f	θ	γ	σ	ϕ	$[f, \theta, \gamma, \sigma, \phi]$	min	max	avg	std	Time
1 × 64	2	4	2	2	2	[0.176, 0.78, 2.30, 1.87, 3.52]	0.0	5.65	1.92	1.52	<1
64 × 64	8	8	4	4	4	[0.088, 0.39, 1.95, 1.71, 4.58]	0.0	5.06	1.56	1.38	<1
64 × 128	8	8	4	4	8	[0.124, 1.96, 2.04, 2.39, 3.62]	0.0	5.95	2.05	1.83	<1
128 × 128	8	8	4	8	8	[0.031, 0.78, 1.79, 2.12, 5.98]	0.0	5.98	2.05	1.72	<1
128 × 256	8	8	8	8	8	[0.022, 0.39, 1.50, 1.95, 4.24]	0.0	5.78	2.08	1.72	1
256 × 256	8	16	8	8	8	[0.250, 2.15, 2.41, 2.24, 3.72]	0.0	6.06	1.98	1.54	1
256 × 512	16	16	8	8	8	[0.002, 0.19, 1.49, 1.79, 3.30]	0.0	6.12	1.65	1.57	2
512 × 512	16	16	8	8	16	[0.015, 0.98, 2.24, 2.37, 6.17]	0.0	6.17	2.03	1.79	4

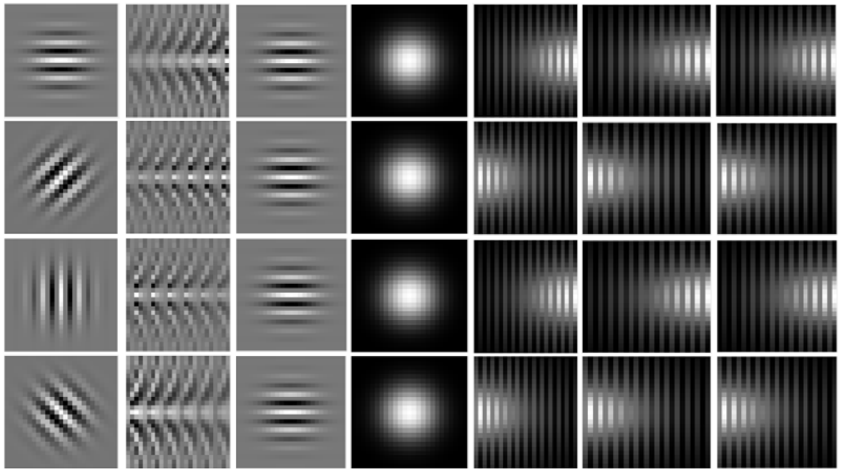

Fig. 4. The first three columns show the real part of example Gabor filters for different parameter values. The following four columns are the magnitude of example Gabor filters.

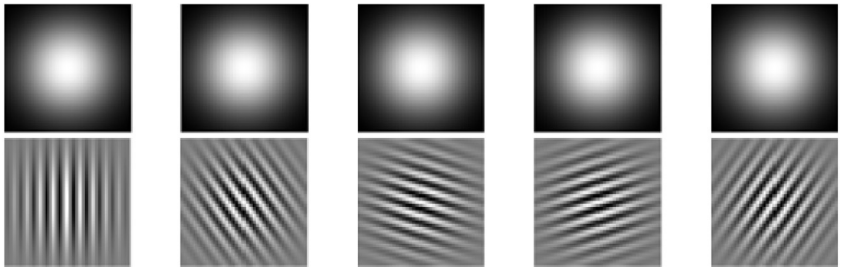

Fig. 5. $\theta \in \{0, 0.6283, 1.2566, 1.8850, 2.5133\}$ for fixed $f = 0.25, \sigma = 10, \gamma = 1, \phi = 0$.

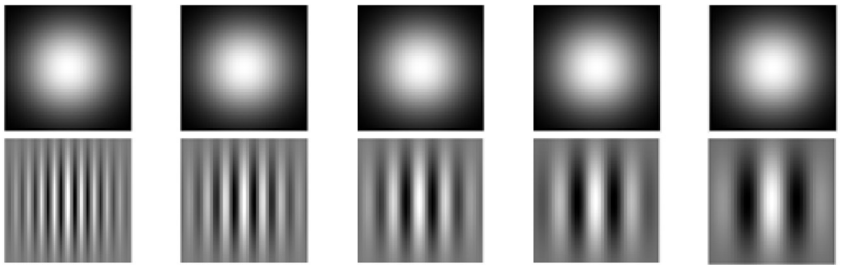

Fig. 6. $f \in \{0.25, 0.17, 0.125, 0.884, 0.625\}$ for fixed $\theta = \pi, \sigma = 10, \gamma = 1, \phi = 0$.

Gabor filters were chosen for LV segmentation due to their structure. In [62] it is stated that, for vision tasks, a good rule pertaining to the choice of kernels is that they should be able to be tuned to the task and have good localization. Gabor filters are rotation, scale and translation invariant. Ideally, the DCNN would inherit these properties from the filters, resulting in a classification that is invariant to the same transformations. Therefore, in the segmentation task, initialization with Gabor filters is capable of detecting the circular shape of the LV within the MRI image and results in more accurate segmentation. Additionally, the Gabor filters combat image noise caused by the MRI LV segmentation complications outlined above such as tissue contrast and blurred edges. Maintaining the Gabor filter properties during the learning process when all of the filters are constantly changing enables

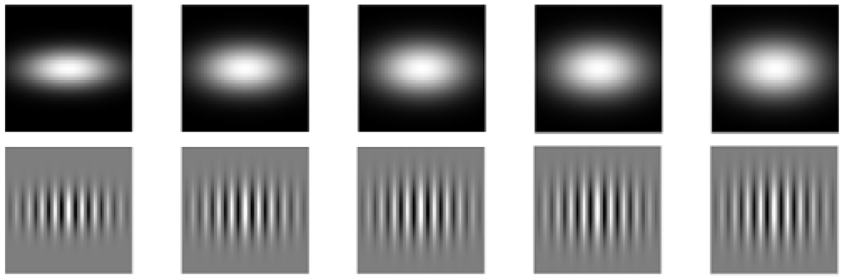

Fig. 7. $\gamma \in \{2.4142, 1.9142, 1.7475, 1.6642, 1.6142\}$ for fixed $f = 0.25, \theta = \pi, \sigma = 10, \phi = 0$.

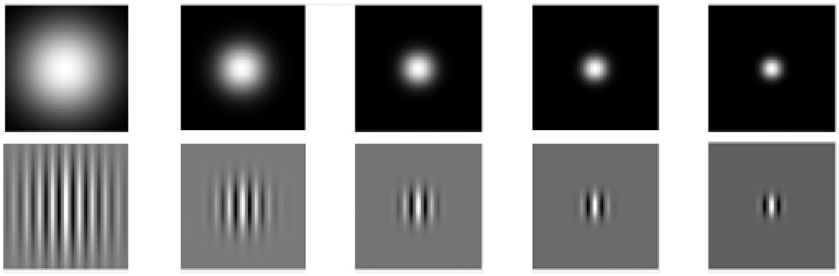

Fig. 8. $\sigma \in \{10, 5, 3.3333, 2.5, 2\}$ for fixed $f = 0.25, \theta = \pi, \gamma = 1, \phi = 0$.

the network to maintain the discriminative power coming from Gabor properties. In the results section, we presented the advantage of using Gabor properties in initialization and how the preservation of these properties enables the model to learn and extract almost the same amount of information using a smaller quantity of training data. We also experimented with the noise handling power of our proposed approach compared to other approaches and the results are promising. Another advantage of using Gabor filters and being able to preserve the properties is that the filter structure can be defined to fit different sources of data in real applications. Some examples of this include cell segmentation in medical imaging, sound frequencies, and vessel segmentation in X-ray coronary angiograms or retina images. The only possible disadvantage for utilizing Gabor filters we can think of is their need to be generated in different rotations, phase offset, frequency, spatial aspect ratio and the standard deviation to be able to capture more details from the image. However, as mentioned in Table 1 to generate 65,536 filters of size 3×3 it takes only 1 s and to double the number of filters 1 more second which is not a significant amount of time for learning with deep structures.

2.4. Maintaining Gabor structure

The updating procedure for our network is the standard gradient descent algorithm. When the network provides a segmentation, the distance of the prediction from the ground truth label is calculated and a loss function is applied. The gradient descent algorithm then calculates the amount of error that is attributable to each weight, i.e. filter. This calculation then provides an estimate of the amount and direction of weight change necessary for each filter in order to optimize network performance.

The gradient descent algorithm computes the partial derivative of the error with respect to each weight and the network iteratively updates the filters in each layer. More explicitly, for a weight w currently taking on the value of w_{old} and an error equation E, we calculate the following:

$$w_{new} = w_{old} + lr \times \frac{\partial E}{\partial w} \tag{9}$$

Where lr is the learning rate.

The gradient descent algorithm is a staple in neural networks. One problem with the standard implementation of the gradient descent algorithm is that even if the network filters are initialized to have a certain structure aimed at achieving a specific image processing task, the filters are randomly updated. Therefore, filter structure is lost after a few iterations through the network. We propose an adaptation to gradient descent that allows for a structured filter to be updated according to network requirements while maintaining the desired filter structure. This method is outlined for Gabor filters.

In our network it is desirable to maintain the Gabor structure of the filters because of their circle detection properties. As outlined above, the LV in MRI is circular and thus the circular behavior of the Gabor filters greatly assists in image segmentation and LV wall detection. Unfortunately the gradient descent algorithm causes the filters to deviate from the Gabor structure after a small amount of training. Once this happens, regardless of Gabor initialization, the circular behavior of the Gabor filters will no longer be helpful to the network.

In order to maintain the Gabor structure, the filter must be parameterized as seen above. Therefore, we generate the filters based on parameters. A simple addition to the gradient descent procedure is necessary. We calculate the error attributable to each Gabor parameter, $\{f, \gamma, \sigma, \theta, \phi\}$, by updating the parameter as opposed to the weight, as seen below for parameter p:

$$p_{new} = p_{old} + lr \times \frac{\partial E_{total}}{\partial p}, p \in \{f, \gamma, \sigma, \theta, \phi\} \tag{10}$$

$$\frac{\partial E_{total}}{\partial p} = \frac{\partial E}{\partial w} \times \frac{\partial w}{\partial p} = \sum_{i,j} \frac{\partial E}{\partial w_{i,j}} \times \frac{\partial w_{i,j}}{\partial p}, i, j \in \{s1, s2\} \tag{11}$$

Eq. (10). follows the chain rule in calculus and Eq. (11) follows the total derivatives rule to compute the derivative of parameter p at all location $i, j \in \{s1, s2\}$ where $s1, s2$ are filter's width and height respectively.

At the initialization step, we use a set of Gabor parameters to generate a two dimensional filter. Therefore, each entry of a two dimensional filter is affected by all parameters of the Gabor. To consider this, we use the total derivative rule from calculus

to compute the total contribution of all the Gabor parameters at each coordinate i, j in the filter.

$$\frac{\partial w_{i,j}}{\partial p} = \sum_{p} \frac{\partial G(x, y; f, \phi, \gamma, \theta, \sigma)}{\partial p}, p \in \{f, \phi, \gamma, \theta, \sigma\} \tag{12}$$

The partial derivative of the filter with respect to each parameter were calculated by deriving the Gabor parameterization with respect to each parameter, as shown below:

$$\frac{\partial G(x,y)}{\partial \theta} = y \cdot exp(-\frac{x'^2 + \gamma^2 \cdot y'^2}{2\sigma^2})[(\frac{\gamma^2 - 1}{\gamma^2}) \cdot x \cdot cos(\frac{2\pi x}{f} + \phi) - \frac{2\pi}{f} \cdot sin(\frac{2\pi x}{f} + \phi)] \tag{13}$$

$$\frac{\partial G(x,y)}{\partial f} = \frac{2\pi x}{f^2} sin(\frac{2\pi x}{f} + \phi) exp(-\frac{x'^2 + \gamma^2 \cdot y'^2}{2\sigma^2}) \tag{14}$$

$$\frac{\partial G(x,y)}{\partial \gamma} = -\frac{y'^2 \gamma}{\sigma^2} cos(\frac{2\pi x}{f} + \phi) exp(-\frac{x'^2 + \gamma^2 \cdot y'^2}{2\sigma^2}) \tag{15}$$

$$\frac{\partial G(x,y)}{\partial \sigma} = \frac{x'^2 + \gamma^2 y'^2}{2\sigma^3} cos(\frac{2\pi x}{f} + \phi) exp(-\frac{x'^2 + \gamma^2 \cdot y'^2}{2\sigma^2}) \tag{16}$$

$$\frac{\partial G(x,y)}{\partial \phi} = -sin(\frac{2\pi x}{f} + \phi) exp(-\frac{x'^2 + \gamma^2 \cdot y'^2}{2\sigma^2}) \tag{17}$$

After these calculations, the new weight update equation is as follows:

$$w_{new} = w_{old} + lr \cdot \frac{\partial E}{\partial w} = w_{old} + lr \cdot \sum_{k=1}^{5} \frac{\partial E}{\partial G} \times \frac{\partial G}{\partial p_k} \tag{18}$$

The extension of the gradient descent algorithm in this manner, the addition of one more partial derivatives, attributes all errors to the Gabor parameters as opposed to the filter elements themselves. In this way the filters can be adjusted through the parameters and can maintain their structure, as opposed to blindly updating the filters without considering the objective of the network.

Using Gabor filters to initialize both segmentation and classification tasks, the results of training the network with modifications in the gradient descent algorithm to maintain the Gabor properties can be seen in Section 3.

3. Results

In this section we explain different experiments in classification and segmentation tasks. We utilized an MRI cardiac dataset for the segmentation task and CIFAR10, MNIST, FashionMNIST and NotMNIST for classification task.

3.1. Segmentation task

We processed 7980 MRI images from the 33 patients, age 2–18 years, contained in the York Cardiac Segmentation database [63]. Each patient's data consisted of 8–15 slices for 20 time frames resulting in 7980 images. Therefore, the cardiac MRI images in the dataset are snapshots of MRI video taken from 33 subjects. In this dataset there may be some images appearing at both the beginning or end of these sequences in which the LV is not recognizable and therefore there is no annotation. We removed these slices because annotation is required for processing the images in the network, and we use the 5011 remaining images along with their annotations, i.e. labels, as the data for our DCNN. Subject MRI video details can be found in Table 2. The outlined variations in pixel spacing and spacing between slices speaks to the generalizability of our method.

During training, the network runs for 50 epochs. Each epoch processing and segmenting 10 batches of 20 images. The network is validated with 150 of the MRI images from the dataset. An example of image–label pairs for the dataset can be found in Fig. 9.

Table 2
Subject dataset image details.

Subject number	Pixel spacing (mm/pixel)	Spacing between slices (mm/slice)
1	1.3281	9
2	1.5625	9
3	1.4844	8
4	1.25	9
5	1.25	9
6	0.9375	7
7	1.1719	9
8	1.4063	10
9	1.1719	8
10	1.4062	9
11	1.1719	9
12	1.3672	9
13	1.3672	9
14	1.4844	13
15	1.3281	8
16	1.4844	8
17	1.5625	6
18	1.3281	9
19	1.4062	7
20	1.4844	8
21	1.4844	9
22	1.6406	10
23	1.5625	12
25	1.4844	8
26	1.25	8
27	1.4844	9
28	1.3281	10
29	1.6406	9
30	1.5625	8
31	1.4844	8
32	1.4844	9
33	1.3281	8

Discuss result comparison

We report four metrics to measure the performance of the model: pixel accuracy, mean accuracy, sensitivity and specificity. The performance metrics provided use True Positive (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), True Object Pixels (TO), True Background Pixels (TB).

$$PixelAccuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{19}$$

$$Sensitivity = \frac{TP}{TO} \tag{20}$$

$$Specificity = \frac{TN}{TB} \tag{21}$$

$$MeanAccuracy = \frac{1}{2}(Sensitivity + Specificity) \tag{22}$$

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{23}$$

It is important to note that given the relatively large background in the images, the sensitivity performance measure does not carry as much meaning as the specificity performance measure, which can be seen as the network's accuracy on the image pixels. The comparison of results for the network initialized randomly to the Gabor initialization can be seen in Table 3. An example of the image, prediction, label triples can be seen in Fig. 10.

In order to show the capability of our proposed method in capturing important information in the data using fewer number

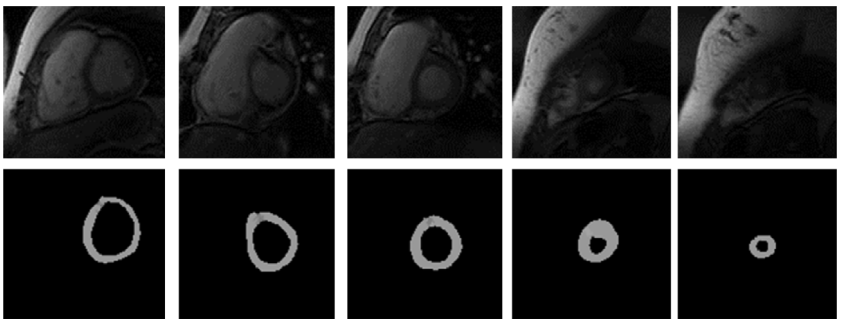

Fig. 9. Image–Label pairs for MRI slices in the York Cardiac Segmentation database.

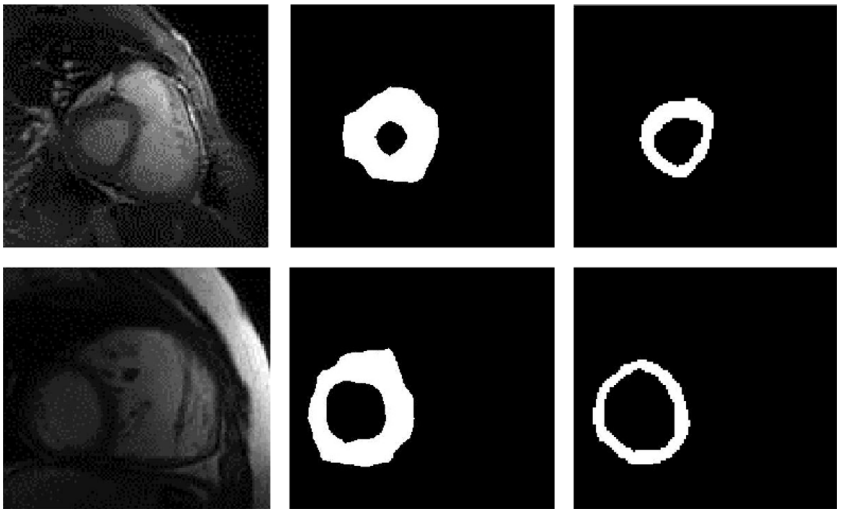

Fig. 10. From left to right, first row: Original input image 1, Our pixel-wise prediction for image 1, True label image 1, second row: Original input image 2. Our pixel-wise prediction for image 2. True label image 2.

Table 3
Segmentation results.

100% training set	Pixel accuracy	Specificity	Sensitivity	Mean accuracy	Dice
Gabor filters-maintained	**0.973**	**0.984**	**0.841**	**0.903**	**0.803**
Gabor filters-initialized	0.969	0.964	0.828	0.901	0.800
Random filters	0.961	0.961	0.821	0.901	0.798
80% training set					
Gabor filters-maintained	**0.961**	**0.967**	**0.822**	**0.887**	**0.789**
Gabor filters-initialized	0.952	0.946	0.809	0.781	0.778
Random filters	0.901	0.903	0.778	0.863	0.757
60% training set					
Gabor filters-maintained	**0.952**	**0.955**	**0.810**	**0.869**	**0.778**
Gabor filters-initialized	0.941	0.921	0.785	0.838	0.752
Random filters	0.822	0.887	0.723	0.809	0.709

of input images in training the model, we have done the same classification task using only 80% and 60% of the data. We have randomly selected 80% of the data for use as a training set. We then tested the learned model on unseen test data to compute the metrics. Next we repeated the same experiment using 60% randomly selected training set. The results are presented in Table 3. Since the original size of the training set for a complex architecture such as VGG-16 is not large, removing 20% and 40% of the data affects the performance of the network. However, as shown in Table 3 it affects the performance when using random filters more severely than the case when using Gabor filters. Sensitivity of the models with Gabor filter drops only 2% when dropping the number of training data by 20%, while the sensitivity of the model with random filters drops 5%. When reducing the number of training data by 40%, the sensitivity of the model initialized with Gabor filters drops by 4%, model with random filter drops by 10%, while the model with maintained Gabor filters only drops by 3%.

It is worth mentioning that the use of the Gabor filter does not directly reduce the training data. As we showed in experiments with different amounts of training data for segmentation tasks in Table 3 and later for classification task, Table 5, the model managed to reduce the loss when using Gabor filters. For random filters, the network obviously had difficulty finding a direction in gradient space, in which it can reduce the loss when using a lesser amount of training data.

In future research, we will investigate how performance will be affected when using a much smaller and simpler network (i.e. less complex architecture) for the same network with Gabor and random filters.

Table 4
Classification results with different initialization methods.

Initialization type	Test accuracy
Gabor initialization	**0.978**
Autoencoder-fashion initialization	0.932
Random initialization	0.901
Transfer learning initialization	0.874
Autoencoder-CIFAR10 initialization	0.832

The work proposed in [17] uses a completely different data set and they evaluate their method in a different way. To evaluate the performance of their proposed method, they have collected 19 patient datasets from 4 different clinical sites. Two experts manually outlined the endocardium and epicardium in the end diastole (ED) and end systole (ES) phases for the entire dataset. They worked together to produce a single contour set that was agreed upon. In [17], a total of 294 images in ED and ES phases (not all the images in all the phases) from 14 subjects were analyzed.

The work in [19] utilized images from 19 patients to evaluate their proposed method, not clear what are the properties of the dataset. For each 20 phases, they have extracted the middle slice and the slice before and after the middle slice where the heart has its largest size, ignoring more challenging slices to segment. They also manually annotated the label contours. Similar to other works in the literature, in [20,21], they utilized a subset of the images. [20] analyzed the images from 14 patients. They only selected images in ED and ES phases(total of 294 image) with manually annotated label contours. [21] also utilized only images in ED and ES phases with manually segmented contour outlined by one expert. It is not clear how many images they used in the training process and how many patients they had in their study.

Same as our data set, [22] uses the York data set to evaluate their proposed method. They split the dataset into three cross-validation sets: 11 subjects for training of the edge classifier, 11 subjects for CRF parameter estimation and 11 subjects for evaluation. Each set contains approximately 100 video sequences and each video contains 20 frames at different z-axis slice positions. The inner and outer contours are manually annotated for all frames and are used as the ground truth in their experiments. They only use video sequences of the sixth z-axis slice of each patient to estimate the CRF parameters.

The York data set is not annotated for all frames(64 pairs of coordinates are provided on inner and outer boundary of the Left Ventricle wall), therefore they have used their own experts to manually annotate the dataset. However, we automatically converted the provided coordination set into a boundary and used it as the ground truth. Therefore, our ground truth is different than theirs. Different ground truth along with different set of images will lead to the difference in the accuracy of the method in different studies.

All the studies reviewed here only segment a specific subset of images where the visibility of the left ventricle is maximum. We perform the segmentation in all images including those where the left ventricle is very small and the object/background ratio in the image is low. This makes the segmentation task much more challenging to perform accurately.

3.2. Classification task

To show our proposed addition to the CNN's performance on other tasks and data, we deployed our method to the MNIST data set to classify handwritten digits. We initialized the method using different schemes including: random initialization, pre-trained weights from a Convolutional Autoencoder trained on gray-scaled cropped CIFAR10 image data set, pre-trained weights from a Convolutional Autoencoder trained on fashionMNIST image data set, pre-trained weights from an CNN trained on gray-scaled cropped CIFAR10 and Gabor initialization.

3.2.1. Initialization with Gabor

For Gabor initialization, we initialized the filter wights in each layer using the real part of the Gabor filters, after changing all five parameters. Then we trained the model using regular gradient descent to see the results of Gabor initialization. We report the results for different initialization methods in Table 4.

In order to compare different initialization methods, we designed the experiments to have the same structure for input data and same number of epochs to train and test the model. We have used different data-sets to examine which initialization method can result in a better performance. The utilized datasets are: CIFAR10, MNIST, fashinMNIST and notMNIST.

3.2.2. Initialization with auto-encoder

Fashin MNIST [64] is a data-set of 60,000 images with 10 fashion item categories including types of clothing and shoes. Not MNIST [65] is a data set of 60,000 images of the first 10 letters of English alphabet (A to J). To report the performance on the test set, we have repeated the experiment 10 times, each time for 10 epochs, and have reported the average of 10 repeated experiments.

The purpose of this experiment is to compare the impact of initialization on the performance of the model to classify MNIST digits. We designed this experiment to initialize the CNN with pre-trained weights from Convolutional Autoencoder versus random initialization and Gabor initialization and train the network from scratch. We trained a Convolutional Autoencoder with the same encoder structure as our CNN using CIFAR10 dataset and fashion MNIST dataset.

Fashion MNIST, same as MNIST, contains gray images of size 24×24, while CIFAR10 contains color images of size 32×32. We applied some per-processing steps on CIFAR10 before using them to train the Transfer Learning CNN and Autoencoder including color-to-grayscale, cropping and flipping. We have examined different available color-to-grayscale approaches, but we have not observed any significant difference in performance. In order to control the computation load on the GPU, we divided the data sets into 5 batches and we trained the network for 10 epochs in each experiment. We obtained the test set results by running each experiment 10 times, each time for 10 epochs. We then calculated the average of the 10 experiments.

3.2.3. Initialization with transfer learning

For Transfer Learning, we have experimented with gray-scaled CIFAR10, fashionMNIST and notMNIST, but since the results of CIFAR10 was the best among the three experiments, we only included the results from CIFAR 10. We have trained a CNN with same structure using gray-scaled cropped CIFAR-10 dataset, then we initialized our CNN for the classification task using the final weights learned from CIFAR-10.

As reported in Table 4, the best result obtained when we used Gabor initialization. In the first epoch, the performance of initialization with fashionMNIST Autoencoder is better than Gabor initialization, but in all other epochs network continues increasing performance using Gabor initialization. However, the results of initialization using weights learned from Autoencoder trained with fashionMNIST is better than the random initialization, but the results from Autoencoder trained with CIFAR10 performs worse than random initialization. One simple informal explanation for this observation is that Autoencoders are data

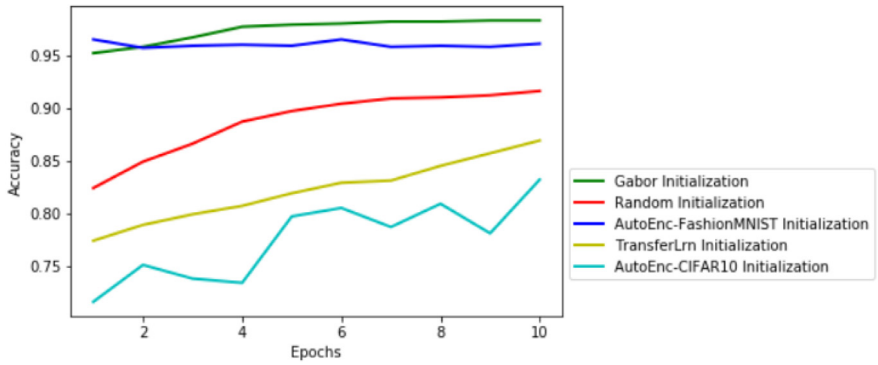

Fig. 11. Comparing model performance with Gabor initialization when we do not maintain the Gabor structure during training vs with Gabor initialization when we maintain the Gabor structure during training process for 5 batches, trained for 10 epochs.

Table 5
Performance of each initialization method on test set when training with different amounts of training data.

Training method	Acc (20%)	Acc (40%)	Acc (60%)	Acc (80%)
Gabor initialization	0.962	0.967	0.969	0.971
Random initialization	0.762	0.785	0.794	0.805
Autoencoder-fashion initialization	0.901	0.915	0.921	0.929
Transfer learning initialization	0.751	0.764	0.792	0.815
Autoencoder-CIFAR10 Initialization	0.706	0.728	0.732	0.802
Gabor maintained	**0.971**	**0.973**	**0.979**	**0.980**

Table 6
Classification results while maintaining filter structure.

Updating rule	Test accuracy
With maintaining filter structure	**0.987**
Without maintaining filter structure	0.978

specific [43]. This means that if the data set being used for training the pre-trained weights from the Autoencoder is similar to the data set being used to train the main model on an Autoencoder, the performance will likely increase. In this set of experiments the CNN initialized with pre-trained weights from a CNN trained with gray-scaled CIFAR10 data set showed the lowest performance. It is likely that the low quality of CIFAR10 images and blurriness contributes to the low performance.

3.2.4. Variation in number of images in training set

We Compared the performance of different approaches with respect to the size of training set. We utilized only 20%, 40%, 60% and 80% of the data and trained the CNN for 10 epochs. As shown in Table 5, the CNN with CIFAR-10 pre-trained weights drops the accuracy when we add more training data and continues fluctuating. One explanation is the more we train the network with a data set different than the original data used to train the Autoencoder, the more the weights deviate from what they have learned initially to perform the classification task. A summary of how different numbers of images in the training set affect the performance of each approach is presented in Table 5.

3.2.5. Maintaining the filter

We implement the new updating procedure to keep the initialization structure during training and to take advantage of filter properties during the learning process. The results of this step are presented in Table 6, compared to the training performance without maintaining filter structure. Considering the properties of the Gabor filter in capturing more information from the image, we were expecting two main behaviors to observe: (1) a significant difference in the amount of information captured from images in early stages of training and (2) observing a disappearance of the Gabor properties after a small number of iterations using the regular gradient descent algorithm. Fig. 11 demonstrates the first observation confirming our first expectation. Fig. 14 is a confirmation of our second expectation. Fig. 14 is an illustration for the response of the convolution layer on the input image of Fig. 13. As shown in part (c) of this figure, without implementing any modification to take the Gabor characteristic into account when training the model using SGD, the Gabor characteristic of the filters will disappear. The results of our new updating rule, integrating the Gabor formula into the gradient descent based updating rule, will update the filters based on the contribution of each Gabor parameter in the total error. This is illustrated in 14(b) and 12. It is important to note that MNIST is a very well behaved dataset and it is relatively easy to obtain a high accuracy, even with a small CNN. The images are small and variability in the data is well handled. Therefore, we designed a convolutional neural network with four convolution layers, each followed by a max pooling layer and a ReLu operation in Tensorflow. Each of the convolution layers uses a different number of pixel size 5×5 filters. We then use two fully connected layers to classify the images. We compared the results with and without Gabor initialization and the new updating rule as shown in Table 6 and Fig. 12.

3.3. Noise introduction

After training the models, we examined how they reacted to the noise introduced in the input of the network. The same amount of Gaussian noise ($mean = 0, std = 0.1$) was added to the input for each network and then the mean error and the standard deviation for the errors computed for 10 repeated experiments for each model. Table 7 summarizes the error rate comparison among different initialization methods. The difference between mean error rate with and without noise for all the models not using Gabor filters is significant. In the case of initialization with autoencoder trained with CIFAR-10, the network continued decreasing loss while training on more epochs after introducing noise in input data. For random initialization and transfer learning initialization, the performance of the model after ten epochs was worse than randomly assigning a class to each image in the test set. The networks with Gabor filters manged to recover from

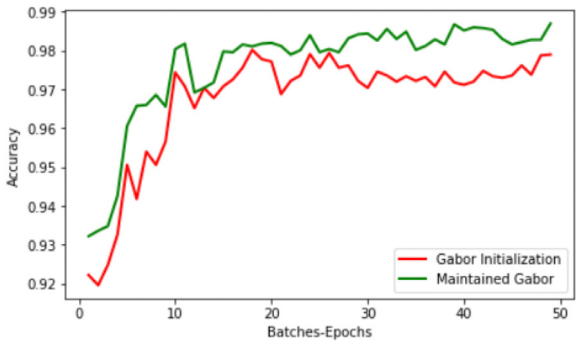

Fig. 12. Comparing model performance for different initialization methods. Autoencoder trained with fashionMNIST showed promising performance, while Autoencoder trained with cropped gray-scaled CIFAR10 had weaker performance than random initialization. Gabor initialization showed the strongest performance, while a pre-trained CNN, trained on cropped gray-scaled CIFAR10 showed the weakest performance to classify MNIST.

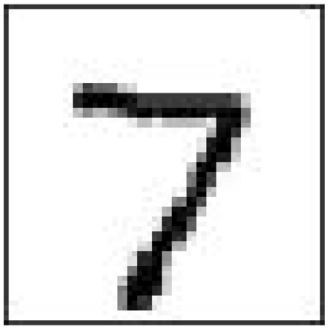

Fig. 13. Input image.

Table 7
Mean and standard deviation error for 10 repeated experiments with and without Gaussian noise with mean = 0 and std = 0.1.

Training method	Without noise		With noise	
	Mean	Std	Mean	Std
Gabor initialization	0.019	0.049	0.028	**0.018**
Random initialization	0.088	0.078	0.58	0.036
Autoencoder-fashion initialization	0.108	0.027	0.535	0.158
Transfer learning initialization	0.178	0.031	0.412	0.103
Autoencoder-CIFAR10 initialization	0.213	**0.027**	0.619	0.092
Gabor maintained	**0.015**	0.039	**0.021**	0.027

noise only after two epochs. The difference between the mean error rate with and without the noise for models with the Gabor filter is insignificant.

4. Conclusion

This paper presented a new updating rule maintaining the characteristics of the initial filters during the training process of convolutional neural network. We have utilized Gabor filters as filter initialization to take advantage of their properties to capture the variation of image content relying on their different parameters. We have presented the performance of our proposed method on both classification and segmentation tasks.

Our results illustrate that the Gabor filter initialization increases both specificity and sensitivity in segmentation task. The results show that this method correctly labels the pixels that belong to object of interest 84% of the time, while correctly labeling background pixels 98% of the time. Future work could continue to improve the methods for background bias elimination, increasing the sensitivity of pixel classification.

The Gabor filter bank performs marginally better than a randomly initialized filter bank, autoencoder initialization and CNN

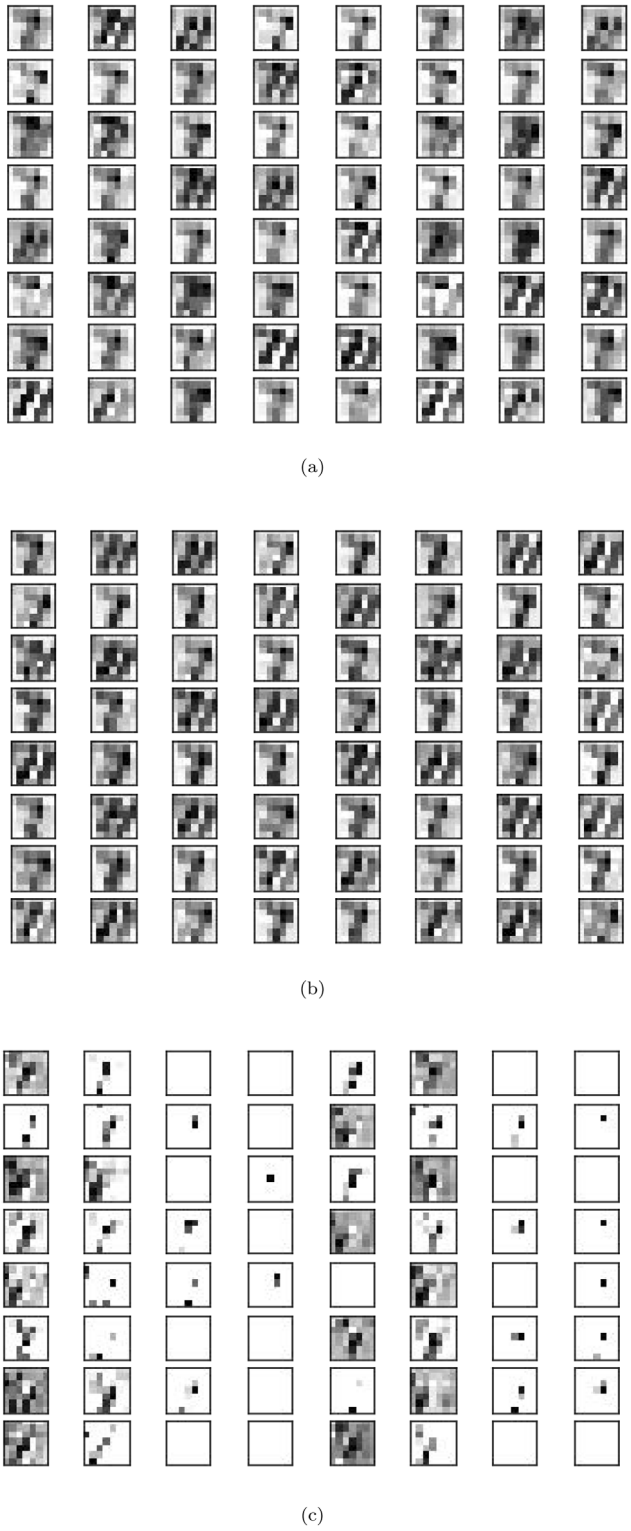

Fig. 14. (a) Filter initialization using Gabor filter bank. (b) Filters are initialized with Gabor and they are maintained as Gabor filter during the training process (c) Filters are initialized with Gabor but they are not maintained during the training process.

obtained initialization, indicating that the Gabor structure is more tailored to the task. Future work could continue to improve upon this accuracy by searching for even better filter initialization or by improving the updating procedure of the network to account for the new filter type. In the classification task, we have illustrated the capability of the proposed method to increase the performance.

We introduced the same amount of noise into the input data and we observed that the network initialized with the Gabor filter bank was able to recover from noise and learn the data more accurately than other methods in classification task.

After increasing the number of images in the training set to only 20%, 40%, 60% and 80% of the original number, the CNN with Gabor filters outperformed the other methods in the classification task in each experiment. After reducing the number of training images in the segmentation task to 60% and 80% of the original number, the model with the Gabor filter outperformed the same model with random filters.

In conclusion, future studies are required to process a higher volume of images, taking advantage of the network parameters, i.e. filters, for improved accuracy. We will further study the advantages of Gabor filter maintenance in order to reduce the complexity of the convolutional neural networks. In this study, we implemented the updating rule for gradient descent, but it is easily adaptable to any other optimizer algorithm like Adam optimizer, AdaGrad optimizer, etc. We utilized the new updating rule in a vgg-16 for the segmentation task and a smaller convolutional network for the classification task, but the proposed method can be used with any other network structure to take advantage of the specific characteristics of the filters.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105960.

References

[1] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, Yann LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks, 2013, arXiv preprint arXiv:1312.6229.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1904–1916.

[3] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, Dong Yu, Convolutional neural networks for speech recognition, IEEE/ACM Trans. Audio Speech Lang. Process. 22 (10) (2014) 1533–1545.

[4] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[5] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.

[6] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, Dongjun Shin, Compression of deep convolutional neural networks for fast and low power mobile applications, 2015, arXiv preprint arXiv:1511.06530.

[7] Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, Rob Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, in: Advances in Neural Information Processing Systems, 2014, pp. 1269–1277.

[8] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, Victor Lempitsky, Speeding-up convolutional neural networks using fine-tuned cp-decomposition, 2014, arXiv preprint arXiv:1412.6553.

[9] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, Pascal Fua, Learning separable filters, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2754–2761.

[10] Max Jaderberg, Andrea Vedaldi, Andrew Zisserman, Speeding up convolutional neural networks with low rank expansions, 2014, arXiv preprint arXiv:1405.3866.

[11] Somayeh Molaei, M.E. Shiri, Kelsey Horan, Delaram Kahrobaei, B. Nallamothu, Kayvan Najarian, Deep convolutional neural networks for left ventricle segmentation, in: 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC, IEEE, 2017, pp. 668–671.

[12] John G. Daugman, Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression, IEEE Trans. Acoust. Speech Signal Process. 36 (7) (1988) 1169–1179.

[13] Judson P. Jones, Larry A. Palmer, An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex, J. Neurophysiol. 58 (6) (1987) 1233–1258.

[14] Sukhpreet Singh, Renu Dhir, Recognition of handwritten gurmukhi numerals using Gabor filters, Int. J. Comput. Appl. 47 (1) (2012) 7–11.

[15] Neeraj Negi, Sanjay Mathur, An improved method of edge detection based on Gabor wavelet transform, WSEAS Recent Adv. Electr. Eng. Electron. Devices (2014) 184–191.

[16] YingLi Lu, Perry Radau, Kim Connelly, Alexander Dick, Graham A. Wright, Segmentation of left ventricle in cardiac cine MRI: An automatic image-driven method, in: Functional Imaging and Modeling of the Heart, Springer, 2009, pp. 339–347.

[17] Marie-Pierre Jolly, Hui Xue, Leo Grady, Jens Guehring, Combining registration and minimum surfaces for the segmentation of the left ventricle in cardiac cine MR images, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009, Springer, 2009, pp. 910–918.

[18] Huaifei Hu, Zhiyong Gao, Liman Liu, Haihua Liu, Junfeng Gao, Shengzhou Xu, Wei Li, Lu Huang, Automatic segmentation of the left ventricle in cardiac MRI using local binary fitting model and dynamic programming techniques, PLoS One 9 (12) (2014) e114760.

[19] Marwa M.A. Hadhoud, Mohamed I. Eladawy, Ahmed Farag, Franco M. Montevecchi, Umberto Morbiducci, Left ventricle segmentation in cardiac MRI images, Am. J. Biomed. Eng. 2 (3) (2012) 131–135.

[20] Amol Pednekar, Uday Kurkure, Raja Muthupillai, Scott Flamm, Ioannis A. Kakadiaris, Automated left ventricular segmentation in cardiac MRI, IEEE Trans. Biomed. Eng. 53 (7) (2006) 1425–1428.

[21] Marie-Pierre Jolly, Automatic segmentation of the left ventricle in cardiac MR and CT images, Int. J. Comput. Vis. 70 (2) (2006) 151–163.

[22] Janto F. Dreijer, Ben M. Herbst, Johan A. Du Preez, Left ventricular segmentation from MRI datasets with edge modelling conditional random fields, BMC Med. Imaging 13 (1) (2013) 24.

[23] Tuan Anh Ngo, Gustavo Carneiro, Left ventricle segmentation from cardiac MRI combining level set methods with deep belief networks, in: Image Processing (ICIP), 2013 20th IEEE International Conference on, IEEE, 2013, pp. 695–699.

[24] Ismail Ben Ayed, Kumaradevan Punithakumar, Shuo Li, Ali Islam, Jaron Chong, Left ventricle segmentation via graph cut distribution matching, in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009, Springer, 2009, pp. 901–909.

[25] Lijia Wang, Mengchao Pei, Noel C.F. Codella, Minisha Kochar, Jonathan W. Weinsaft, Jianqi Li, Martin R. Prince, Yi Wang, Left ventricle: Fully automated segmentation based on spatiotemporal continuity and myocardium information in cine cardiac magnetic resonance imaging (LV-FAST), BioMed Res. Int. 2015 (2015).

[26] Maria Lorenzo-Valdés, Gerardo I. Sanchez-Ortiz, Andrew G. Elkington, Raad H. Mohiaddin, Daniel Rueckert, Segmentation of 4d cardiac MR images using a probabilistic atlas and the EM algorithm, Med. Image Anal. 8 (3) (2004) 255–265.

[27] Dan Ciresan, Alessandro Giusti, Luca M. Gambardella, Jürgen Schmidhuber, Deep neural networks segment neuronal membranes in electron microscopy images, in: Advances in Neural Information Processing Systems, 2012, pp. 2843–2851.

[28] Clement Farabet, Camille Couprie, Laurent Najman, Yann LeCun, Learning hierarchical features for scene labeling, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1915–1929.

[29] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, Jitendra Malik, Simultaneous detection and segmentation, in: Computer Vision–ECCV 2014, Springer, 2014, pp. 297–312.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[31] Yaroslav Ganin, Victor Lempitsky, N^ 4-fields: Neural network nearest neighbor fields for image transforms, in: Computer Vision–ACCV 2014, Springer, 2014, pp. 536–551.

[32] Fausto Milletari, Nassir Navab, Seyed-Ahmad Ahmadi, V-net: Fully convolutional neural networks for volumetric medical image segmentation, in: 3D Vision (3DV), 2016 Fourth International Conference on, IEEE, 2016, pp. 565–571.

[33] Jonathan Long, Evan Shelhamer, Trevor Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[34] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 834–848.
[35] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, Trevor Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, 2013, arXiv preprint arXiv:1310.1531.
[36] Alessandro Giusti, Dan C. Cireşan, Jonathan Masci, Luca M. Gambardella, Jürgen Schmidhuber, Fast image scanning with deep max-pooling convolutional neural networks, 2013, arXiv preprint arXiv:1302.1700.
[37] Ehsan Hosseini-Asl, Georgy Gimel'farb, Ayman El-Baz, Alzheimer's disease diagnostics by a deeply supervised adaptable 3D convolutional network, 2016, arXiv preprint arXiv:1607.00556.
[38] Heung-Il Suk, Dinggang Shen, Deep learning-based feature representation for AD/MCI classification, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2013, pp. 583–590.
[39] Jie Geng, Jianchao Fan, Hongyu Wang, Xiaorui Ma, Baoming Li, Fuliang Chen, High-resolution SAR image classification via deep convolutional autoencoders, IEEE Geosci. Remote Sens. Lett. 12 (11) (2015) 2351–2355.
[40] Kai Chen, Mathias Seuret, Marcus Liwicki, Jean Hennebert, Rolf Ingold, Page segmentation of historical document images with convolutional autoencoders, in: 2015 13th International Conference on Document Analysis and Recognition, ICDAR, IEEE, 2015, pp. 1011–1015.
[41] Jonathan Masci, Ueli Meier, Dan Cireşan, Jürgen Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: International Conference on Artificial Neural Networks, Springer, 2011, pp. 52–59.
[42] Zhongling Huang, Zongxu Pan, Bin Lei, Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data, Remote Sens. 9 (9) (2017) 907.
[43] Esam Othman, Yakoub Bazi, Naif Alajlan, Haikel Alhichri, Farid Melgani, Using convolutional features and a sparse autoencoder for land-use scene classification, Int. J. Remote Sens. 37 (10) (2016) 2149–2167.
[44] Rajiv Mehrotra, Kameswara Rao Namuduri, Nagarajan Ranganathan, Gabor filter-based edge detection, Pattern Recognit. 25 (12) (1992) 1479–1494.
[45] Thomas P. Weldon, William E. Higgins, Dennis F. Dunn, Efficient Gabor filter design for texture segmentation, Pattern Recognit. 29 (12) (1996) 2005–2015.
[46] Anil K. Jain, Farshid Farrokhnia, Unsupervised texture segmentation using Gabor filters, Pattern Recognit. 24 (12) (1991) 1167–1186.
[47] Dennis Dunn, William E. Higgins, Optimal Gabor filters for texture segmentation, IEEE Trans. Image Process. 4 (7) (1995) 947–964.
[48] Yoshihiko Hamamoto, Shunji Uchimura, Masanori Watanabe, Tetsuya Yasuda, Yoshihiro Mitani, Shingo Tomita, A Gabor filter-based method for recognizing handwritten numerals, Pattern Recognit. 31 (4) (1998) 395–400.
[49] Ki-Chung Chung, S. Kee, S. Kim, Face recognition using independent component analysis of Gabor filter responses, in: IAPR Workshop on Machine Vision Applications, Citeseer, 2000, pp. 331–334.
[50] Zehang Sun, George Bebis, Ronald Miller, On-road vehicle detection using evolutionary Gabor filter optimization, IEEE Trans. Intell. Transp. Syst. 6 (2) (2005) 125–137.
[51] Zhonghua Zhang, Xuecai Yu, Feng You, George Siedel, Wenqiang He, Lifang Yang, A front vehicle detection algorithm for intelligent vehicle based on improved Gabor filter and SVM, Recent Patents Comput. Sci. 8 (1) (2015) 32–40.
[52] Zehang Sun, George Bebis, Ronald Miller, Improving the performance of on-road vehicle detection by combining Gabor and wavelet features, in: Proceedings. the IEEE 5th International Conference on Intelligent Transportation Systems, IEEE, 2002, pp. 130–135.
[53] Anil K. Jain, Nalini K. Ratha, Sridhar Lakshmanan, Object detection using Gabor filters, Pattern Recognit. 30 (2) (1997) 295–309.
[54] Anil K. Jain, Sushil Bhattacharjee, Text segmentation using Gabor filters for automatic document processing, Mach. Vis. Appl. 5 (3) (1992) 169–184.
[55] Tony Ezzat, Jake Bouvrie, Tomaso Poggio, Spectro-temporal analysis of speech using 2-D Gabor filters, in: Eighth Annual Conference of the International Speech Communication Association, 2007.
[56] Jianwei Yang, Lifeng Liu, Tianzi Jiang, Yong Fan, A modified Gabor filter design method for fingerprint image enhancement, Pattern Recognit. Lett. 24 (12) (2003) 1805–1817.
[57] Andrea Vedaldi, Karel Lenc, Matconvnet: Convolutional neural networks for matlab, in: Proceedings of the 23rd ACM International Conference on Multimedia, ACM, 2015, pp. 689–692.
[58] Nikolaos Karianakis, Jingming Dong, Stefano Soatto, An empirical evaluation of current convolutional architectures' ability to manage nuisance location and scale variability, 2015, arXiv preprint arXiv:1505.06795.
[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.
[60] Mohammad Haghighat, Saman Zonouz, Mohamed Abdel-Mottaleb, Cloudid: trustworthy cloud-based and cross-enterprise biometric identification, Expert Syst. Appl. 42 (21) (2015) 7905–7916.
[61] C.T. Long, Elementary Introduction to Number Theory, Prentice Hall PTR, 1987.
[62] Pietro Perona, Steerable-scalable kernels for edge detection and junction analysis, in: European Conference on Computer Vision, Springer, 1992, pp. 3–18.
[63] Alexander Andreopoulos, John K. Tsotsos, Efficient and generalizable statistical models of shape and appearance for analysis of cardiac mri, Med. Image Anal. 12 (3) (2008) 335–357.
[64] Han Xiao, Kashif Rasul, Roland Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.
[65] Yaroslav Bulatov, Notmnist dataset, Google (Books/OCR), Tech. Rep., 2011, [Online]. Available: http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html.