Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network*

Adhish Prasoon[1], Kersten Petersen[1], Christian Igel[1,**], François Lauze[1], Erik Dam[2], and Mads Nielsen[1,2]

[1] Department of Computer Science, University of Copenhagen, Denmark
[2] Biomediq, Denmark

Abstract. Segmentation of anatomical structures in medical images is often based on a voxel/pixel classification approach. Deep learning systems, such as convolutional neural networks (CNNs), can infer a hierarchical representation of images that fosters categorization. We propose a novel system for voxel classification integrating three 2D CNNs, which have a one-to-one association with the xy, yz and zx planes of 3D image, respectively. We applied our method to the segmentation of tibial cartilage in low field knee MRI scans and tested it on 114 unseen scans. Although our method uses only 2D features at a single scale, it performs better than a state-of-the-art method using 3D multi-scale features. In the latter approach, the features and the classifier have been carefully adapted to the problem at hand. That we were able to get better results by a deep learning architecture that autonomously learns the features from the images is the main insight of this study.

1 Introduction

Convolutional neural networks (CNNs, [1]) are deep learning architectures and have recently been employed successfully for 2D image segmentation tasks [2]. Here we use them for voxel classification in 3D images. There is a generic extension of CNNs from 2D to 3D images [3]. However, these truly 3D CNNs have large memory and training time requirements, retarding their application for time-constrained large-scale medical imaging tasks. In this paper we propose a classifier based on a system of *triplanar 2D CNNs* which classifies voxels from 3D images with high accuracy and is easy to train.

Deterioration of cartilage implies Osteoarthristis, which is one of the major reasons for work disability [4]. Cartilage segmentation in knee MRI scans is the method of choice for quantitative analysis of cartilage deterioration. We applied our method for segmenting tibial articular cartilage from knee MRI and

* This work has been funded by the Danish Strategic Research Council through the grant Learning Imaging Biomarkers (grant no. 09-065145).
** Christian Igel acknowledges support by the European Commission through project AKMI (PCIG10-GA-2011-303655).

K. Mori et al. (Eds.): MICCAI 2013, Part II, LNCS 8150, pp. 246–253, 2013.

compared the performance with a state-of-the-art method [5]. Cartilage segmentation in knee MRI scans is often performed by the radiologists in a slice-by-slice manner and is a time consuming and tedious job. Furthermore, the inter and intra observer variability is found to be rather high. Thus, for large studies, automated segmentation is desirable. Therefore, the automatic and semi-automatic segmentation of knee cartilage in MRI scans is an active field of research. These methods address the task either slice-by-slice or directly perform 3D segmentation [6–9].

State-of-the-art. Folkesson et al. [5] developed a highly efficient voxel classification based segmentation method which can be considered as the state-of-the-art in fully automatic cartilage segmentation from MRI scans. It served as the reference in our study. The approach is based on k nearest neighbor classification (kNN) with $k = 100$ and a predefined 178-dimensional feature vector. The training of their classifier involves reducing the number of features using a sequential floating forward feature selection algorithm and also selecting a posterior threshold that optimizes the segmentation results. The features calculated using 3D filters are: the intensity, the three-jet, the position, the eigenvectors and eigenvalues of the 3D Hessian matrix, as well as the eigenvector and eigenvalues of the 3D structure tensor and the third order tensor in the gradient direction. All features except position are calculated at 3 different scales.

2 Method

Convolutional neural networks were introduced by LeCun et al. [1], who applied it for handwritten digit recognition. Since then, CNNs have been used successfully for many recognition and segmentation tasks [2, 10–14]. The first part of this section will give a brief introduction to 2D CNNs. The second part will explain our technique for classifying voxels in 3D images.

Convolutional Neural Networks. Convolutional neural networks are deep learning architectures [1]. A CNN is a multilayer perceptron and mainly consists of *convolutional layers* and *subsampling layers*.

The network topology exploits the stationary nature of natural images by learning features using locally connected networks. Convolutional layers are used to learn small feature detectors based on patches randomly sampled from a large image. A feature in the image at some location can be calculated by convolving the feature detector and the image at that location. A subsampling layer is used to reduce the number of features in order to reduce the computational complexity, to introduce invariance properties, and to reduce the chances of overfitting. It summarizes the statistics of a feature over a region in the image.

Let $\mathbf{d}^{(l-1)}$ and $\mathbf{d}^{(l)}$ be the input and output for the lth layer, respectively. Let $\mathbf{d}^{(0)}$ be the 2D input image patch and $\mathbf{d}^{(L)}$ be the output of the last layer L. Let $S_I^{(l)} \times S_I^{(l)}$ and $S_O^{(l)} \times S_O^{(l)}$ be the size of the input and the output map, respectively, for layer l. Furthermore, let $N_I^{(l)}$ and $N_O^{(l)}$ be the number of input and output maps respectively for that layer. The input to lth layer is the output of $(l-1)$th

layer, $N_I^{(l)} = N_O^{(l-1)}$ and $S_I^{(l)} = S_O^{(l-1)}$. We denote the jth output-feature-map of layer l as $\mathbf{d}_j^{(l)}$. As the input-feature-maps of the lth layer are actually the output-feature-maps of the $(l-1)$th layer, $\mathbf{d}_i^{(l-1)}$ is the ith input-feature-map of layer l. The output of a convolutional layer is computed as

$$\mathbf{d}_j{}^{(l)} = f\Big(\sum_i \mathbf{d}_i^{(l-1)} * \mathbf{w}_{ij}^{(l)} + b_j^{(l)} \mathbf{1}_{S_O^{(l)}}\Big) \ , \tag{1}$$

where $*$ denotes the convolution and $0 \leq i < N_I^{(l-1)}$, $0 \leq j < N_O^{(l-1)}$. The mapping f is a *nonlinear activation function*, often a sigmoid, which is applied to each component of its argument individually. The 4D tensor $\mathbf{w}^{(l)}$ is of size $S_W \times S_W \times N_I^{(l)} \times N_O^{(l)}$, and $\mathbf{w}_{ij}^{(l)}$ is the kernel linking ith input map to jth output map. The size of $\mathbf{w}_{ij}^{(l)}$ is $S_W \times S_W$, and we refer to S_W as the sidelength of the kernel. Finally, the scalar $b_j^{(l)}$ is the bias element for jth output-feature-map of lth layer. $b_j^{(l)}$ is multiplied to every element of all-ones matrix $\mathbf{1}_{S_O^{(l)}}$ of size $S_O^{(l)} \times S_O^{(l)}$ before being added to $\sum_i \mathbf{d}_i^{(l-1)} * \mathbf{w}_{ij}^{(l)}$. We use *mean pooling* in the subsampling layer: the output of the element (x, y) of jth feature map of layer l is given as

$$d_j^{(l)}(x,y) = \frac{\sum_{m=0}^{s-1}\sum_{n=0}^{s-1} d_j^{(l-1)}(s \times x + m, s \times y + n)}{s^2} + b_j^{(l)} \ , \tag{2}$$

where s is the subsampling factor and $0 \leq x, y < S_I^{(l)}$. A subsampling layer has the same number of output and input maps. The parameters for a subsampling layer are only bias parameters. The final layer of a CNN is fully connected to the preceding layer. If L is the number of layers, we vectorize the output maps of layer $L-1$ and then concatenate them all together to obtain the output of the last layer. The number of output maps $N_O^{(L)}$ for this layer is $N_O^{(L-1)} \times (S_O^{(L-1)})^2$, each corresponding to a single scalar. The output of the last layer is

$$d_j^{(L)}(0,0) = d_i^{(L-1)}(x,y) \ \text{ with } \ j = i \times (S_O^{(L-1)})^2 + (y-1) \times S_O^{(L-1)} + x \ . \tag{3}$$

The output of the fully-connected layer L acts as input for the softmax classifier (i.e., logistic regression for a two class problem). Let $\boldsymbol{\theta}$ be the parameter matrix for softmax classifier. It has size $K \times N_O^{(L)}$, where K in the number of classes, each row being associated to one particular class. Let vectors ${\boldsymbol{\theta}_1}^T, {\boldsymbol{\theta}_2}^T \ldots \boldsymbol{\theta}_K^T$ be K rows of $\boldsymbol{\theta}$. Given the softmax matrix $\boldsymbol{\theta}$, the probability of mth training example belonging to class n is

$$p(n|\mathbf{d}^{(L)(m)}; \boldsymbol{\theta}) = \frac{e^{{\boldsymbol{\theta}_n}^T \mathbf{d}^{(\mathbf{L})(m)}}}{\sum_{c=1}^{K} e^{{\boldsymbol{\theta}_c}^T \mathbf{d}^{(L)(m)}}} \ , \tag{4}$$

where $n \in \{1, 2, \ldots . K\}$ and $\mathbf{d}^{(\mathbf{L})(m)}$ is the output of the last layer of the CNN when presented the mth training example. As $\mathbf{d}^{(L)(m)}$ is dependent on $\mathbf{d}^{(0)(m)}$,

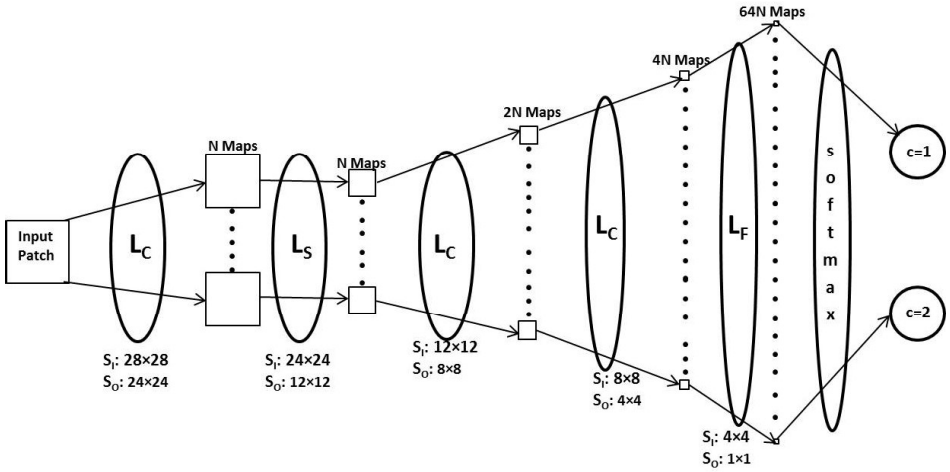
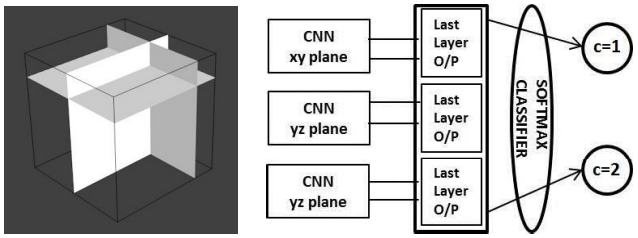


Fig. 1. Convolutional neural network

and the kernel and bias parameters of the CNN, $p(n|\mathbf{d}^{(L)^{(m)}}; \boldsymbol{\theta})$ can be seen as the probability of mth training example belonging to class n given the CNN parameters, input patch, and the softmax parameters. Let $\boldsymbol{\Omega}$ be a set of all the parameters comprising the kernel and the bias parameters from all the layers of the CNN as well as the softmax parameters. As $\mathbf{d}^{(L)}$ depends on $\mathbf{d}^{(0)}$ and the CNN parameters, $p(n|\mathbf{d}^{(L)^{(m)}}; \boldsymbol{\theta})$ can also be written as $p(n|\mathbf{d}^{(\mathbf{0})^{(m)}}; \boldsymbol{\Omega})$. Let $\boldsymbol{t}$ be a vector which stores the true labels of all the training data points and $t^{(m)}$ be the true label of the mth training example. The parameters are identified using a maximum likelihood approach. The cost function associated with the mth training example is $Q^{(m)} = -\ln(p(t^{(m)}|\mathbf{d}^{(0)^{(m)}}; \boldsymbol{\Omega}))$ and for all M training examples we have accordingly $Q = \frac{1}{M}\sum_{m=1}^{M} Q^{(m)}$. We use *weight decay*, which penalizes too large values of the softmax-parameters, to regularize the classification. The cost function is minimized by gradient-based optimization. The partial derivatives are computed using backpropagation, and LBFGS has proven to be well suited for the optimization [15].

Triplanar Convolutional Neural Network. For each voxel, we take the three planes $\mathbf{P}_{xy}, \mathbf{P}_{yz}, \mathbf{P}_{zx}$ that pass through the voxel and are parallel to the xy, yz and zx plane, respectively, of the 3D image (see Fig. 2a). A 2D patch centered around that voxel is extracted from each of the three planes. Let $\boldsymbol{d}_{xy}^{(0)}, \boldsymbol{d}_{yz}^{(0)}, \boldsymbol{d}_{zx}^{(0)}$ be those three patches. We have one CNN for each of the three patches. The 3 CNNs are not connected with each other except for the output of the last layer. Let $\boldsymbol{d}_{xy}^{(L)}$, $\boldsymbol{d}_{yz}^{(L)}$ and $\boldsymbol{d}_{zx}^{(L)}$ be the outputs of the last layers of the three CNNs and $N_{O_{xy}}^{(L)}$, $N_{O_{yz}}^{(L)}$ and $N_{O_{zx}}^{(L)}$ be the number of output-maps of the last layers of the three CNNs. The last layers' output-maps for all 3 CNNs are scalars. Thus $\boldsymbol{d}_{xy}^{(L)}, \boldsymbol{d}_{yz}^{(L)}$ and $\boldsymbol{d}_{zx}^{(L)}$ are vectors of length $N_{O_{xy}}^{(L)}$, $N_{O_{yz}}^{(L)}$, and $N_{O_{zx}}^{(L)}$, respectively. $\boldsymbol{d}_{xy}^{(L)}, \boldsymbol{d}_{yz}^{(L)}$ and $\boldsymbol{d}_{zx}^{(L)}$ are concatenated to obtain a joint output $\boldsymbol{d}_{T}^{(L)}$, which is fed into the softmax classifier. The softmax parameter matrix $\boldsymbol{\theta}_T$ in this case is of size $K \times N_{OT}$, where $N_{OT} =$



(a) (b)

Fig. 2. The three images planes giving rise to our triplanar convolutional neural network (CNN) architecture. One patch centered in the voxel is extracted from each of the planes. The three CNNs are fused in the final layer.

$N_{O_{xy}}^{(L)}+N_{O_{yz}}^{(L)}+N_{O_{zx}}^{(L)}$. Let $\boldsymbol{\Omega}_T$ be the set of all the parameters collecting the kernel and bias parameters from all the layers of the three CNNs as well as the softmax parameters. The cost associated with the mth training example in the triplanar CNN is ${Q_T}^{(m)} = -\ln(p(t^{(m)}|\boldsymbol{d}_{xy}^{(0)};\boldsymbol{d}_{yz}^{(0)};\boldsymbol{d}_{zx}^{(0)};\boldsymbol{\Omega}_T))$ and the overall cost function reads

$$Q_\lambda = -\frac{1}{M}\sum_{m=1}^{M}\ln(p(t^{(m)}|\boldsymbol{d}_{xy}^{(0)};\boldsymbol{d}_{yz}^{(0)};\boldsymbol{d}_{zx}^{(0)};\boldsymbol{\Omega}_T)) + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{N_{OT}}\theta_{T_{pq}}^2 \quad , \tag{5}$$

where $\theta_{T_{pq}}$ is the (p,q) element of the matrix $\boldsymbol{\theta}_T$ and λ controls the strength of the weight decay.

3 Application to Cartilage Segmentation in MRI Scans

In our cartilage segmentation experiments, we minimize $Q_{\lambda T}$ using LBFGS using mini-batches. For each mini-batch, we selected 2000 examples randomly from 120000 training examples and run 20 iterations of LBFGS. For each of the 3 CNNs we used exactly the same configuration: the sequence of layers as well as the number and size of feature maps in corresponding layers were the same. Let $\mathrm{L_C}$ denote a convolutional layer, $\mathrm{L_S}$ a subsampling layer, and $\mathrm{L_F}$ the final output layer. We used the sequence $\mathrm{L_C} \rightarrow \mathrm{L_S} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_F}$ for each CNN. The single subsampling layer performed mean pooling with subsampling factor 2 (see equation 2). Instead of a second subsampling layer after the fourth-layer (a convolution layer), we have another convolution layer as the fifth layer. Having more than one subsampling layers can lead to over-summarizing of the features, which did not seem to be a good approach for our application because of the thin structure of the cartilage. For each CNN, we fixed the size of input patches to 28×28 and kernel size to 5×5. Let N be the number output feature maps of the first convolutional layer. The numbers of output-feature-maps are $N \rightarrow N \rightarrow 2N \rightarrow 4N \rightarrow 64N$ for the whole sequence of layers. Fig. 1 depicts such a CNN. Combining three of these CNNs leads to the triplanar CNN for 3D images shown in Fig. 2b, where the output of the final CNN layers are concatenated before being fed into the classifier. We selected N using cross-validation (splitting

the training dataset) from $\{12, 16, 20, 24, 28, 32\}$. We also selected the weight decay parameter λ using cross-validation from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. As a result, $N = 28$ and $\lambda = 10^{-2}$ were selected. The heuristic design choices were inspired by Simard et al.'s article about best practices for CNNs for document analysis [16] and our previous experience in CNN design.

The patients knee is being scanned in standardized position in a specialized extremity MRI scanner. Still, there will be small variations from scan to scan between the image planes and corresponding body planes due to anatomical and placement differences. This variability is reflected by the training data. Thus, the triplanar CNN is trained to become invariant under these variations.

We use the same 25 training scans as in [5], where features from on average 84000 voxels from each of the scans are extracted. These 84000 voxels include all the cartilage voxels in a scan and sampled background voxels. As it is more difficult to classify the background voxels near the cartilage, they are sampled very densely close to the cartilage and rarely far from it, with sampling probability varying linearly. For generating our training data we extracted triplanar patches from 4800 randomly selected voxels from the pool of 84000 voxels used by [5] for each of the 25 scans. That is, while in [5], 2100000 voxels were used to generate the training data, we just considered 120000 voxels for limiting the computational cost and found this number to be sufficient. We applied ZCA whitening to preprocess our patches, which is very similar to PCA whitening, however, it also involves a rotation back to the input data basis.

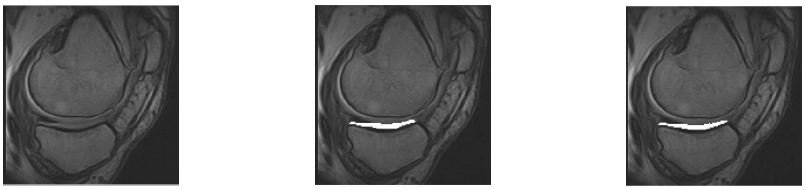
4 Results, Discussion, and Future Work

We evaluated our method on 114 unseen knee MRIs, each having approximately 2 million voxels to classify. We compared our method to the state-of-the-art method [5] which was also evaluated on the same test set of 114 scans. The evaluation was based on dice similarity coefficient(DSC) where $\mathrm{DSC}(A, B) = \frac{2(|A \cap B|)}{|A|+|B|}$ and A, B are manual and automatic segmentation. Table 1 shows results obtained by the triplanar-CNN and the kNN based method. Our method achieves better DSC, sensitivity, specificity and accuracy than [5]. The difference in DSC is statistically significant (Wilcoxon rank-sum test, $p < 0.05$). Fig. 3 shows a knee MRI slice, its segmentation by a radiologist, and the segmentation by our method.

Our method used just 120000 training voxels whereas in [5] 2.1 million training voxels are used. We also evaluated the performance of the method proposed in [5] using 120000 training voxels and observed a loss in performance. The average DSC, accuracy, sensitivity and specificity fell to 0.7952 ($\pm$9.10%), 99.91% ($\pm$3.18%) ,78.58% ($\pm$11.61%) and 99.95% ($\pm$2.80%)(standard deviations in brackets), respectively, although we adjusted k of kNN for the new training set size using cross-validation. The difference between the triplanar-CNN and the method from [5] with 120000 training voxels is statistically significant (Wilcoxon rank-sum test, $p = 8.6 \times 10^{-5}$). We also performed baseline experiments with 3D CNNs. With a $14 \times 14 \times 14$ 3D patch size and $3 \times 3 \times 3$ kernel size, the

Table 1. Comparison of methods applied for tibial cartilage segmentation

Method	Over 114 Scans	DSC	Accuracy	Sensitivity	Specificity
Triplanar CNN	Mean	0.8249	99.93%	81.92%	99.97%
	Standard Deviation	4.26%	1.86%	7.62%	1.74%
State-of-the-art [5]	Mean	0.8135	99.92%	80.52%	99.96%
	Standard Deviation	4.87%	2.31%	8.95%	2.37%



(a) Knee MRI slice (b) radiologist (c) triplanar CNN

Fig. 3. MRI slice with segmentations by a radiologist and the proposed triplanar CNN. Our method is based on voxel classification, a 2D slice is taken from our 3D segmentation just for visualization.

performance of a 3D CNN was worse (approximately 2% dip in DSC) than the triplanar CNN with patch size 28×28 and kernel size 5×5. Using these configurations, the training and testing times of the two approaches were comparable. Using a 3D CNN with 3D patch size $28 \times 28 \times 28$ (leading to 21952 elements and thus a 21952×21952 matrix to be inverted in the ZCA) and kernel size $5 \times 5 \times 5$ turned out to be computationally not feasible in our experimental setup as each iteration of back- and forward-propagation needed 10 to 15 times more time compared to our triplanar 2D CNN. We also experimented with a single 2D CNN, with patches extracted from just a single plane (the same which radiologists use for segmenting). However, the performance achieved by a single 2D CNN was inferior to that of the triplanar 2D CNN. Based on the above experiments we found that the triplanar CNN provides the best balance between computational costs and performance. The method by Folkesson et al. [5] – in particular the features used, but also the classifier – has been carefully tailored towards the task. That we can perform better using a method that *learns* the image features using a deep learning architecture is the main result of our study. This is all the more remarkable because of the following differences in comparison to [5] and other CNN architectures: **a)** The features used by our method were extracted using only 2D kernels learned from three 2D CNNs. However, the method from [5] uses 3D kernels to extract features. **b)** In [5], each feature is calculated at 3 different scales. Our method relies on the features extracted using single scale CNNs. **c)** The number of training data points used by our method is just 120000, while in [5] more than 2100000 training data points are used. **d)** We used a logistic regression classifier (a linear classifier) compared to kNN (a non-linear classifier). **e)** We did not employ at any stage layer-wise pre-training of the CNN, although pre-training is often claimed to improve the results. We train our system of three CNNs in a single training process.

We plan to extend our study on the segmentation of tibial cartilage to the multiclass problem of segmenting bones and all cartilage compartments.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
2. Schulz, H., Behnke, S.: Learning Object-Class Segmentation with Convolutional Neural Networks. In: ESANN (2012)
3. Turaga, S.C., Murray, J.F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K.L., Denk, W., Seung, H.S.: Convolutional Networks can Learn to Generate Affinity Graphs for Image Segmentation. Neural Computation 22(2), 511–538 (2010)
4. Jackson, D., Simon, T., Aberman, H.: Symptomatic Articular Cartilage Degeneration: The impact in the New Millenium. Clinical Orthopaedics and Related Research 133, 14–25 (2001)
5. Folkesson, J., Dam, E., Olsen, O., Pettersen, P., Christiansen, C.: Segmenting Articular Cartilage Automatically Using a Voxel Classification Approach. IEEE TMI 26(1), 106–115 (2007)
6. Solloway, S., Hutchinson, C., Vaterton, J., Taylor, C.: The Use of Active Shape Models for Making Thickness Measurements of Articular Cartilage from MR Images. Magnetic Resonance in Medicine 37, 943–952 (1997)
7. Pakin, S.K., Tamez-Pena, J.G., Totterman, S., Parker, K.J.: Segmentation, Surface Extraction and Thickness Computation of Articular Cartilage. In: SPIE Medical Imaging, vol. 4684, pp. 155–166 (2002)
8. Fripp, J., Crozier, S., Warfield, S., Ourselin, S.: Automatic Segmentation and Quantitative Analysis of the Articular Cartilages from Magnetic Resonance Images of the Knee. IEEE TMI 29(1), 55–64 (2010)
9. Yin, Y., Xiangmin, Z., Williams, R., Xiaodong, W., Anderson, D., Sonka, M.: LOGISMOS - Layered Optimal Graph Image Segmentation of Multiple Objects and Surfaces: Cartilage Segmentation in the Knee Joint. IEEE TMI 29(12), 2023–2037 (2010)
10. Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., Barbano, P.E.: Toward Automatic Phenotyping of Developing Embryos from Videos. IEEE TIP 14(9), 1360–1371 (2005)
11. Sermanet, P., LeCun, Y.: Traffic Sign Recognition with Multi-Scale Convolutional Networks. In: IJCNN, pp. 2809–2813 (2011)
12. Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: Flexible, High Performance Convolutional Neural Networks for Image Classification. In: IJCAI, 1237–1242 (2011)
13. Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-End Text Recognition with Convolutional Neural Networks. In: ICPR, pp. 3304–3308 (2012)
14. Ji, S., Xu, W., Yang, M., Yu, K.: 3D Convolutional Neural Networks for Human Action Recognition. IEEE TPAMI 35(1), 221–231 (2013)
15. Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A.Y.: On Optimization Methods for Deep Learning. In: ICML, 265–272 (2011)
16. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: ICDAR, pp. 958–962 (2003)