Classifying Volume Datasets Based on Intensities and Geometric Features

Dženan Zukić, Christof Rezk-Salama, and Andreas Kolb

Abstract Many state-of-the art visualization techniques must be tailored to the specific type of dataset, its modality (CT, MRI, etc.), the recorded object or anatomical region (head, spine, abdomen, etc.) and other parameters related to the data acquisition process. While parts of the information (imaging modality and acquisition sequence) may be obtained from the meta-data stored with the volume scan, there is important information which is not stored explicitly, e.g. anatomical region. Also, meta-data might be incomplete, inappropriate or simply missing.

This paper presents a novel and simple method of determining the type of dataset from previously defined categories. A 2D histogram of the dataset is used as input to the neural network, which classifies it into one of several categories it was trained with. Two types of 2D histograms have been experimented with, one based on intensity and gradient magnitude, the other one on intensity and distance from center.

A significant result is the ability of the system to classify datasets into a specific class after being trained with only one dataset of that class. Other advantages of the method are its easy implementation and its high computational performance.

Key words: volume visualization, 3D datasets, 2D histograms, neural networks, classification

Dženan Zukić `zukic@fb12.uni-siegen.de`
Christof Rezk-Salama `rezk@fb12.uni-siegen.de`
Andreas Kolb `kolb@fb12.uni-siegen.de`

Computer Graphics and Multimedia Systems Group, University of Siegen
Hölderlinstrasse 3, 57076 Siegen, Germany

1 Introduction

Volume visualization techniques have seen a tremendous evolution within the past years. Many efficient rendering techniques have been developed in recent years including 3D texture slicing [2, 26], 2D texture mapping [17], pre-integration [7], GPU ray-casting [12, 19, 22], and special purpose hardware [15].

Nevertheless, the users of volume visualization systems, which are mainly physicians or other domain scientists with only marginal knowledge about the technical aspects of volume rendering, still report problems with respect to usability. The overall aim of current research in the field of volume visualization is to build an interactive rendering system which can be used autonomously by non-experts.

Recent advances in the field of user interfaces for volume visualization, such as [16] and [18] have shown that semantic models may be tailored to the specific visualization process and the type of data in order to meet these requirements. The semantic information is built upon a priori knowledge about the important structures contained in the dataset to be visualized. A flexible visualization system must thus contain a high number of different semantic models for the huge variety of different examination procedures.

An important building block for an effective volume rendering framework is a classification technique which detects the type of dataset in use and automatically applies a specific semantic model or visualization technique. For example, some methods are created specifically for visualizing MRI scans of the spine or CT scans of the head, and those methods rely on the actual dataset being of that type (i.e. its modality and its anatomical region).

The prior knowledge required for selecting an appropriate visualization technique includes imaging modality, acquisition sequence, anatomical region, as well as other parameters such as chemical tracing compound. That is beyond the information stored in the file system or the meta-data, therefore we propose a technique which classifies the datasets using a neural network which operates on statistical information, i.e. on histograms of the 3D data itself.

We have tested our method and determined that it can delineate datasets depending on imaging modality and anatomical region. Although this method could possibly be used to separate datasets depending on which tracing compound has been used, if any, we did not have suitable datasets to test this.

The remainder of the paper is structured as follows: In the next section we review related work important to our paper. As we assume that not all the readers are familiar with neural networks, a very short introduction is included in Section 3. Section 4 describes our proposed method for automatic classification of 3D datasets. In Section 5 we describe the test environment our solution was integrated in. Section 6 presents and discusses the results of the standard histogram approach. In Section 7 we introduce a new type of histogram which incorporates geometric features for further delineation of intra-class datasets and Section 8 concludes the paper.

2 Related work

The 2D histogram based on intensity and gradient magnitude was introduced in a seminal paper by Kindlmann and Durkin [10], and extended to multi-dimensional transfer functions by Kniss et al. [11]. Lundström et al. [14] introduced local histograms, which utilize a priori knowledge about spatial relationships to automatically differentiate between different tissue types. Šereda et al. [25] introduced the so-called low/high (LH) histogram to classify material boundaries.

Rezk-Salama et al. [18] suggest a user-centered system which is capable of learning semantic models from examples. In order to generate such a semantic model, a visualization task is performed several times on a collection of datasets which are considered representative for a specific examination scenario. Throughout this training phase the system collects the parameter vectors and analyzes them using principal component analysis. Rautek et al.[16] present a semantic model for illustrative visualization. In their system the mapping between volumetric attributes and visual styles is specified by rules based on natural language and fuzzy logics.

Tzeng et al. [24] suggest an interactive visualization system which allows the user to mark regions of interest by roughly painting the boundaries on a few slice images. During painting, the marked regions are used to train a neural network for multi-dimensional classification. Del Rio et al. adapt this approach to specify transfer functions in an augmented reality environment for medical applications [6]. Zhang et al. [27] apply general regression neural networks to classify each point of a dataset into a certain class. This information is later used for assigning optical properties (e.g. color). Cerquides et al. [3] use different methods to classify each point of a dataset. They use this classification information later to assign optical properties to voxels. While these approaches utilize neural networks to assign optical properties, the method presented here aims at classifying datasets into categories. The category information is subsequently used as an *a priori* knowledge to visualize the dataset.

Liu et al. [13] classify CT scans of the brain into pathological classes (normal, blood, stroke) using a method firmly rooted in Bayes decision theory.

Serlie et al. [21] also describe a 3D classification method, but their work is focused on material fractions, not on the whole dataset. They fit the arch model to the LH histogram, parameterizing a single arch function by expected pure material intensities at opposite sides of the edge (L,H) and a scale parameter. As a peak in the LH-histogram represents one type of transition, the cluster membership is used to classify edge voxels as transition types.

Ankerst et al. [1] conduct classification by using a quadratic form distance functions on a special type of histogram (shell and sector model) of the physical shape of the objects.

3 Neural Network Basics

A neural network is a structure involving weighted interconnections among neurons (see Fig. 1). A neuron is structured to process multiple inputs, usually including a bias (which is weight for fixed input with value +1), producing a single output in a nonlinear way. Specifically, all inputs to a neuron are first augmented by multiplicative weights. These weighted inputs are summed and then transformed via a non-linear activation function, because non-linear activation functions are needed if a neural network is expected to solve a non-linear problem. The weights are sometimes referred to as synaptic strengths.

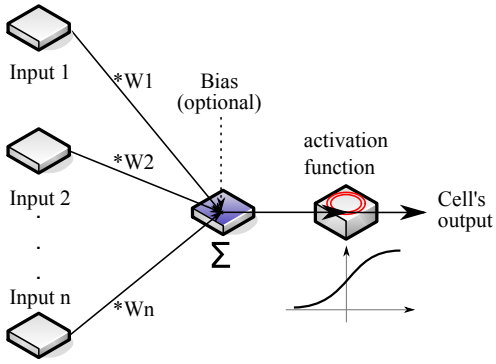


Fig. 1 A neuron

The output of each neuron (except those in the input layer) is computed like:

$$y_j = f(\theta + \sum_i w_{ij} * y_i)$$

where i is the previous layer index, j is the current layer index, w is the weight, y is the output, f is the activation function, θ is the bias (optional).

Feed-forward neural networks usually employ sigmoid activation functions. These functions are smooth and in the [-1,1] range they are approximately linear. Two most commonly used ones are logistic function (see Fig. 2), which has output domain [0,1] and hyperbolic tangent (output domain [-1,1]).

In order to train a the neural network, sets of known input-output data must be assembled. In other words, a neural network is trained by example. The most commonly used algorithm for training feed-forward networks is called back-propagation of errors [20]. The algorithm starts by comparing the actual output of the network for the presented input with the desired output. The difference is called output error, and the algorithm tries to minimize this error using a steepest descent method with the weights as variables.

The training process is repeated many times (epochs) until satisfactory results are obtained. The training can stop when the error obtained is less than a certain limit, or if some preset maximum number of training epochs is reached.

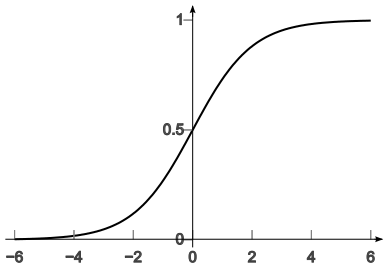


Fig. 2 Logistic activation function: $f(x) = \frac{e^x}{e^x+1}$

One of the most commonly used networks is the multilayer feed forward network (Fig. 3), also called multi-layer perceptron. Feed-forward networks are advantageous as they are the fastest models to execute. Furthermore, they are universal function approximators (see [9]).

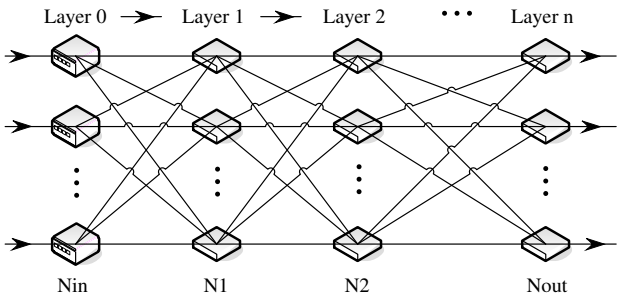


Fig. 3 General schematic of a feed-forward neural network

Feed-forward networks usually consist of three or four layers in which the neurons are logically arranged. The first and the last layer are the input and the output layers. All the others are called hidden layers.

From a general perspective, a neural network is an approximation to an arbitrary function.

A nice (and relatively short) introduction to feed-forward neural networks is presented by Svozil et al. [23].

4 Automatic Classification of Volume Datasets

The method described in this paper was mostly inspired by our previous work [28]. In [28], neural networks are used to position “primitives” on the 2D histogram in order to create transfer function aiming at an effective volume visualization. The method presented here is similar in the sense that it uses 2D histograms as inputs to neural networks.

One of the widely used visualization approaches of 3D data today is direct volume rendering (see [8]) by means of a 2D transfer function. 2D transfer functions are created in respect to the combined histogram of intensity and its first derivative. Although transfer functions rely on intensity/derivative histograms, other histogram types can also be constructed from a 3D dataset. This will be demonstrated later. 2D histograms in turn may be viewed as grayscale images.

All histograms of the same 3D dataset type, e.g. different CT scans of the thorax, look similar to human observers. Likewise, histograms of different datasets types usually look noticeably different, but the difference also depends on the type of the histogram (see Fig. 4). Our method stems from this fact.

Neural networks can easily be trained to approximate an unknown function for which we have observations in the form of input-output combinations. That makes neural networks suitable for classifying input histograms into categories.

The straight-forward approach is to use the histogram pixels (normalized to the $[0,1]$ range) as inputs to the neural network. On the output side, each output corresponds to one category. We take the outputs as representing the probability of the input to belong to the corresponding category. Thus we have a *k*-dimensional output for *k* categories. For example, assume that we have the following $[0,1]$ normalized [1] outputs for some input:

$$\begin{pmatrix} 0,893456 \\ 0,131899 \\ 0,044582 \end{pmatrix}$$

we interpret them as the probabilities of the input belonging to respective category (category one – 89%, category two – 13% and category three – 4%). Notice that the actual outputs in general do not add up to 100%.

In order to identify the most probable classification result, the output with maximum value is chosen. Therefore, this input would be classified as belonging to the category one. Fig. 6, 7 and 8 show actual outputs of a neural network (for easier discerning, descriptive names are given to the outputs).

A training sample consists of the histogram input and the desired output vector. In the desired output vector, only the correct output category has value 1, while all the others have value 0.

In our implementation we chose the multilayer perceptron (MLP), a type of neural network which is capable of performing the required task. It is trained by the back-propagation algorithm. One major benefit of MLP is that additional outputs can be added fairly easily, while retaining the function of all the other outputs. Using some other types of neural networks a new neural network would have to be created and trained from scratch, wasting time whenever a new category is added. Furthermore, this would cause differently randomized initial weights, thus leading to slightly different results. In our version, we only need to add weights between the newly inserted neuron in the output layer and all neurons in the last hidden layer (see Fig. 5).

[1] The activation function which is employed in the neural network we used produces outputs in the convenient range $[0,1]$, so no additional normalization is necessary

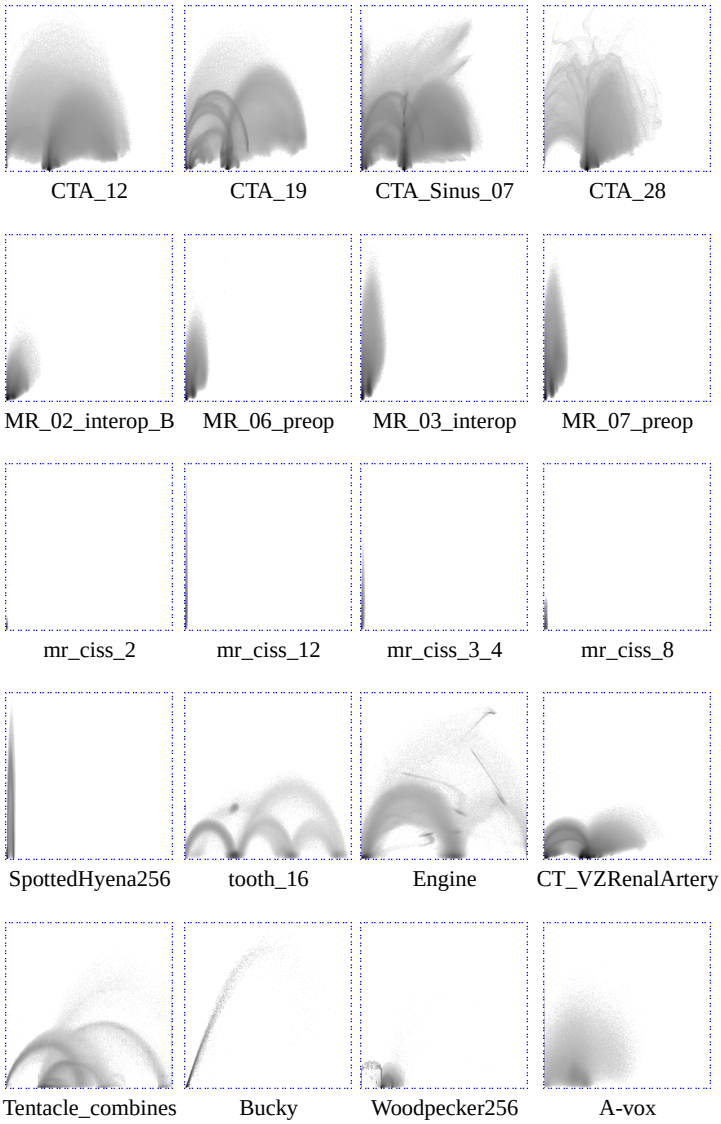


Fig. 4 Some of the histograms of intensity/derivative type. Each one of the first 3 rows represents one class. The histograms in the last two rows result from miscellaneous datasets.

As feed-forward networks can approximate any continuous real function with as little as 3 layers, we have only tested networks with 3 and 4 layers. Fewer number of layers can be compensated with a larger number of neurons in the hidden layer(s). Although some differences exist (see [4, 5]), they are not relevant for this method

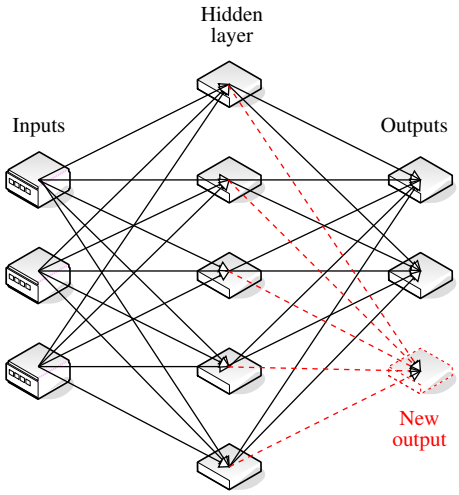


Fig. 5 Adding an output preserves existing weights. The neural network depicted here is very small compared to real examples.

(see Fig. 9). All the results (except Fig. 9) presented here are obtained using a 3 layer neural network.

4.1 Modeling the Rest Class

There are two ways to deal with datasets that do not fall into any of the well-defined classes, i.e. the miscellaneous datasets. The first approach is to have a “rest class”, to which all of these datasets are associated. The second approach assumes that elements from the rest class usually do not strongly activate any of the outputs, often having value of the maximum output around 0,5 (50%). So the second approach uses a threshold for successful classification: If the value of the maximum output is below that threshold, the dataset fails being classified into any of the well-defined classes and it is considered to be part of the rest class.

From a conceptual point of view, the threshold approach is independent from the rest-class approach, i.e. each of the concepts can be applied separately. From a practical point of view, both approaches are not completely independent: the better trained the rest class is, the less effect thresholding provides. Furthermore, providing a high amount of training samples to the rest class affects the reliability, i.e. the value of the maximum output in this context, of the classification of the normal (well-defined) classes. If this is coupled with a high threshold, a lot of “false negatives”, i.e. datasets misclassified as belonging to the rest class instead of a well-defined class, emerge . However, applying both approaches is beneficial for lower amounts of training samples for the rest class.

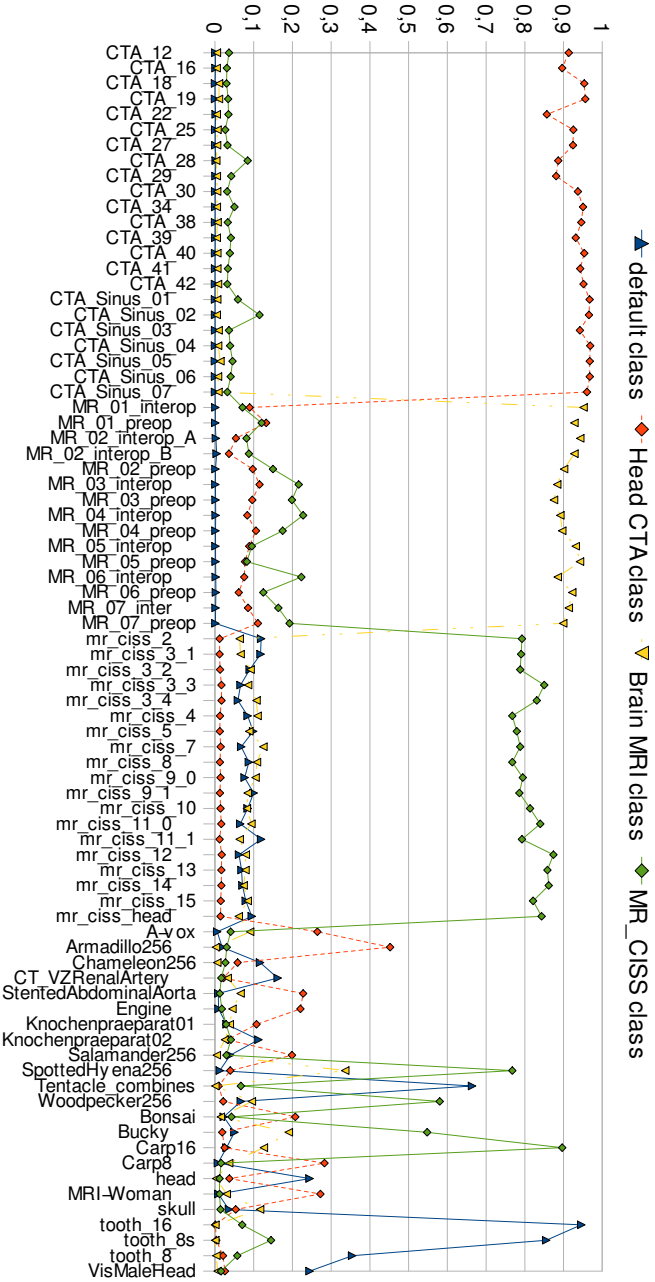


Fig. 6 Raw outputs of the network with the rest class approach (“default”). Trained with 1 sample per class.

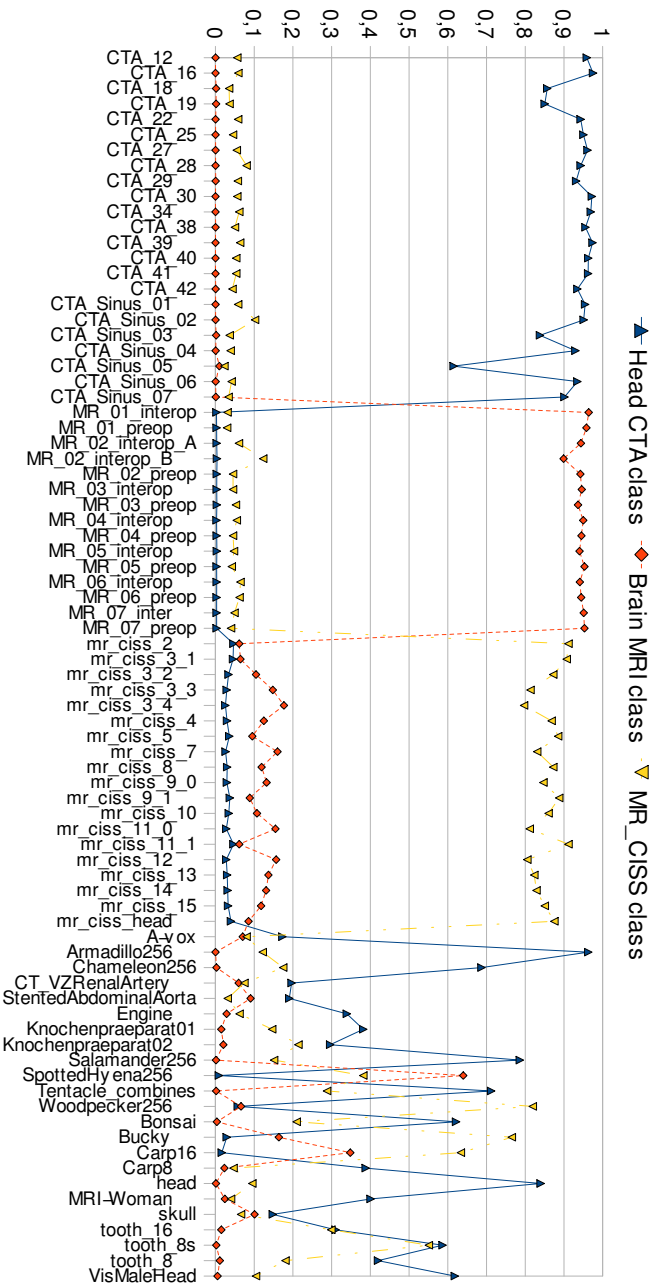


Fig. 7 Raw outputs of the network without the rest class. Trained with 1 sample per class.

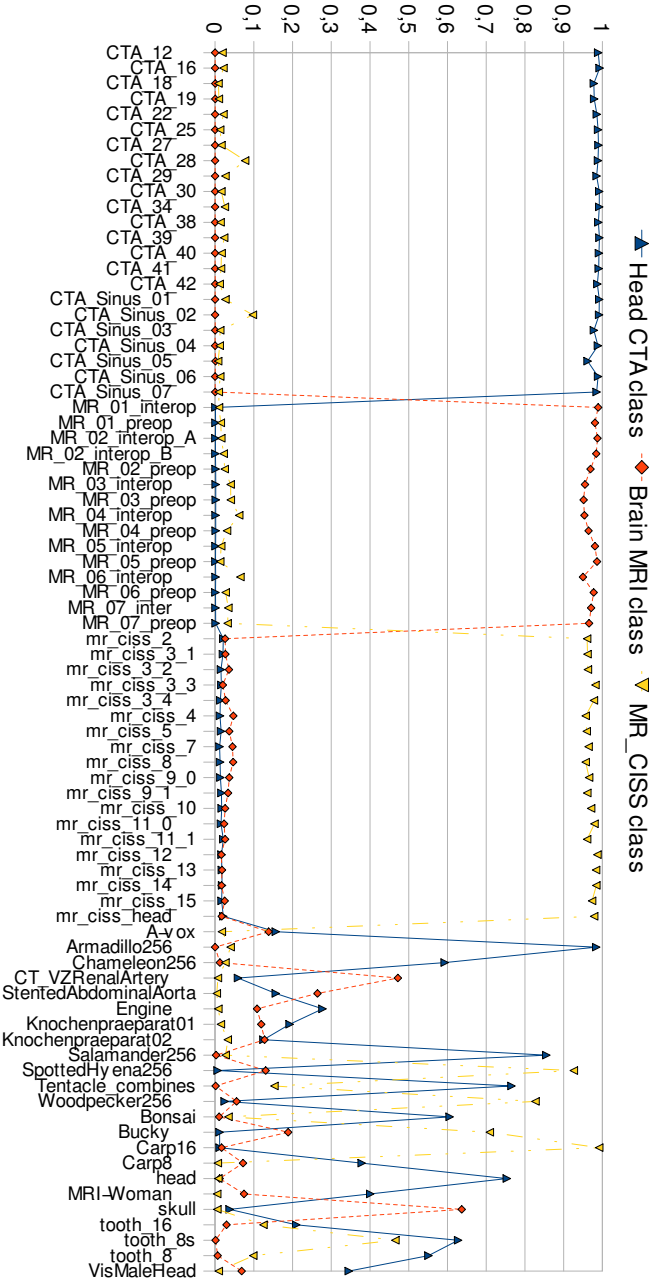


Fig. 8 Raw outputs of the network without the rest class. Trained with 3 samples per class.

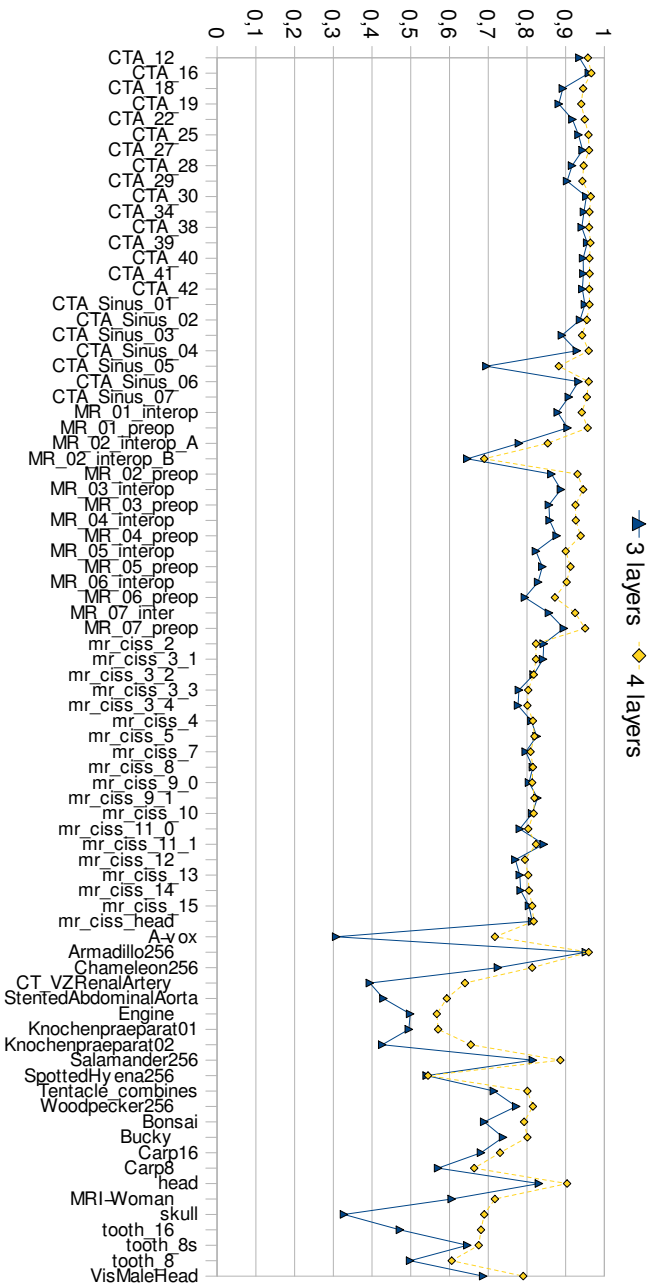


Fig. 9 Using 4-layer neural network does not significantly improve results. Only the value of the maximum output is shown for each dataset.

4.2 Performance Issues

If we directly use histogram pixels as the network's inputs, we have a large number of inputs, e.g. for a 256*256 histogram we get $64K^2$ inputs. If the second layer contains 64 neurons, the number of weights between 1st and 2nd layer is 4M. In our implementation, the weights are 32-bit floats, which leads to 16MB just for the storage of the weights between the 1^{st} and the 2^{nd} layer. The amount of weights between other layers is significantly smaller, due to the much lower number of neurons in these layers.

However, the overall memory consumption is relatively exhaustive. Furthermore, the training gets very slow, and an alternative persistent storage on a hard disk would not be convenient due to slow reading, writing and data transfer.

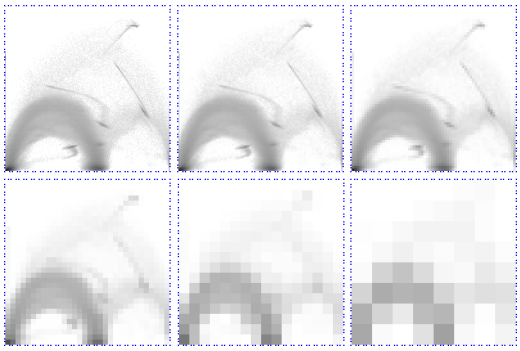


Fig. 10 Size reduction. Upper left is the original 256x256, lower right is 8x8

Therefore, we incorporated a downscaling scheme for the histograms by rebinning. This does not only greatly reduce the required data, but it also significantly eliminates small details present in the histograms. For every dataset, their exact positions are always different, so they are only an obstacle for comparison purposes.

For simplicity, our implementation only allows reduction by factors that are powers of 2. That is: 0 – no reduction, 1 – reduction to 128x128, 2 – reduction to 64x64, etc. Most of the tests have been conducted with reduction factor 3 (histogram size 32x32).

5 Testing Environment

The implementation of the described method is done in a visualization tool called OpenQVis. It is based on a collaborative research project of the *Computer Graphics Group of the University of Erlangen-Nuremberg*, the *VIS Group at the University of Stuttgart* and the *Computer Graphics and Multimedia Systems Group at the Univer-*

[2] prefixes K and M here mean 2^{10} and 2^{20}

sity of Siegen, Germany. OpenQVis focuses on real-time visualization, relying on the features of modern graphics cards (see [8]).

OpenQVis has different "models" of transfer functions, which are used to visualize different types of 3D datasets. Examples are: CT angiography of the head, MRI scans of the spinal cord, MRI scans of the head, and so on. These models were considered as classes for our method.

OpenQVis allows the user to navigate to a model list and to choose one for the currently opened dataset. If the chosen model is not in the list of the output classes, a new output class is added to the neural network and the network is re-trained with this new training sample. If the chosen class is already present in the outputs, the network is re-trained with this new training sample included. If the histogram of the currently opened dataset exists among the training samples, the sample is updated to reflect the new user preference.

Saving training samples with the neural network data is required because each re-training consists of many epochs, and if only the newest sample is used the network gradually "forgets" previous samples, which is, of course, undesired. So, all saved samples are used for each epoch in the re-training process.

For testing purposes, we had three series available:

1. Computed tomography - angiography of the head (CTA_*), 23 datasets
2. Magnetic resonance images of the head, both preoperative and inter-operative (MR_*), 15 datasets
3. Magnetic resonance - constructive interference in the steady state, mostly scans of the spine (mr_ciss_*), 19 datasets

Furthermore, we had 23 miscellaneous datasets, almost all freely available on the internet. 2 of those datasets were synthetic (bucky and tentacle), generated directly from computer 3D models and not acquired by means of a scanning device.

This method can differentiate between cases within the same scanning modality. We tested this with available but confidential CTA heart datasets, which were clearly discernible from CTA head datasets.

6 Results

The classification based on our neural network approach takes, depending on histogram reduction factor, mere microseconds. The training takes milliseconds for the reduction factor 4 and below. The training for the reduction factor 3 takes noticeable fractions of a second (0,2s to 0,6s) in our tests, and for the reduction factor 2 it takes seconds (3-10 seconds). The training time variations result from the termination condition. We use the Mean Squared Error (MSE) condition MSE<0,003 which was nearly almost met before the maximum number of epochs was reached.

The reliability of classification is directly associated with the reduction factor. As seen on Fig. 11, the reliability decreases as the histogram size decreases.

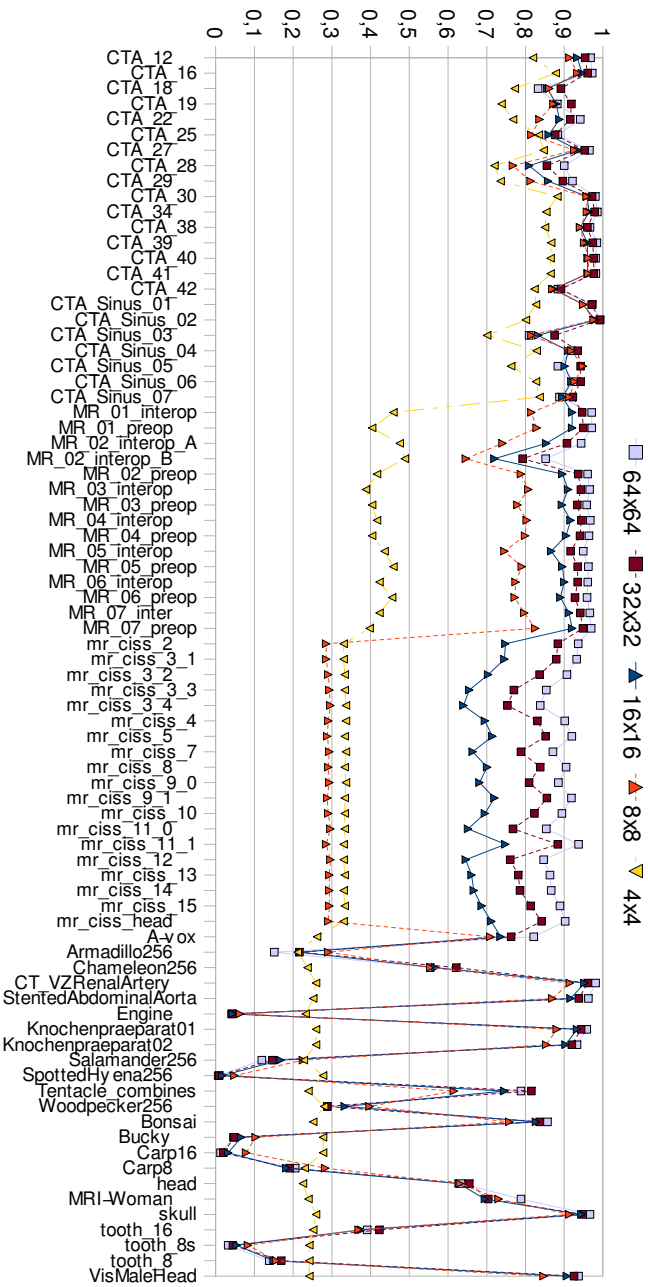


Fig. 11 Downscaling the histogram images on the input side of the neural network reduces it's reliability and, in extreme cases, disables the neural network from delineating datasets.

The choice of the dataset which is used to represent a class influences the results to some degree (see Fig. 12). This influence affects the classification outcome only in the miscellaneous group, i.e. the rest class. Choosing an average-looking histogram for the training, or average and extremes in a case of more training samples per class, results in a higher reliability of the classification and in more uniform output values across all datasets of that class.

A slight variation of the results with respect to the initial randomization of the neural network exists, but is negligible. After training the network with one sample of each type, the average difference in outputs (due to different initial weights) is around 1%. The maximum for any dataset is 5%. These differences get smaller with a greater number of training samples.

Training with multiple datasets of specific classes improves the reliability. Training with multiple datasets of the rest class lowers misclassification rate (see Fig. 13).

With the rest-class approach, all of the misclassifications occur in the miscellaneous group (see Tab. 1). This means, for example, that no CTA is classified as anything else other than CTA. Only datasets from the miscellaneous group are wrongly classified as something else (CTA, MR, or mr_ciss). This is true even if the neural network is trained with only one sample of each type.

The thresholding approach has a lower amount of misclassifications in the miscellaneous group, but it misclassifies some datasets of the other classes ("false negatives").

Approach/Setup	Misclassification rate	
	w.d.c. → rest	rest → w.d.c.
no rest class, no threshold	0	all(23)
with rest class, 1 ts/cl	0	15-20
with rest class, 2 ts/cl	0	10-15
with rc, 2 ts/wdc and 6 ts/rc	0	3-5
no rest class, threshold 50%	0	10-15
no rest class, threshold 70%	0-1	5-10
no rc, threshold 90%, 1 ts/cl	20-25	1-2
no rc, threshold 90%, 2 ts/cl	0	3-5
w. rc, threshold 50%, 1 ts/cl	0	5-15
w. rc, threshold 70%, 1 ts/cl	0-5	2-10
w. rc, threshold 90%, 1 ts/cl	25-30	0-2
w. rc, threshold 90%, 2 ts/cl	0-5	0-2
w. rc, th. 90%, 2 ts/wdc and 6 ts/rc	5-10	0

Table 1 Comparison of misclassification rates for different setups of the classifier. If not specified, the classification has been performed using varying parameters in terms of number of training samples per class (for some setups) or choice of datasets used for training, resulting in slightly different misclassification rates.
Abbreviations: w. – with, ts – training sample(s), cl – class, rc – rest class, wdc – well-defined class, th. – threshold.

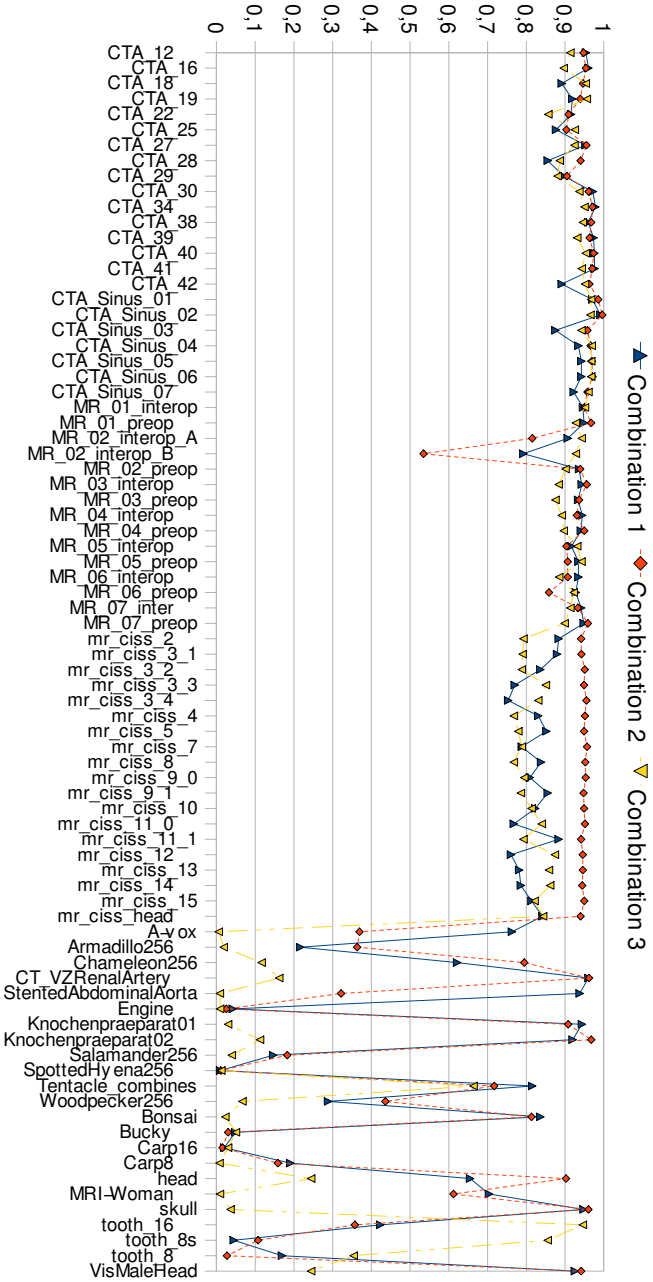


Fig. 12 Choosing different datasets for training the neural network influences the results. Trained with 1 sample per class. Only values of correct outputs are shown.

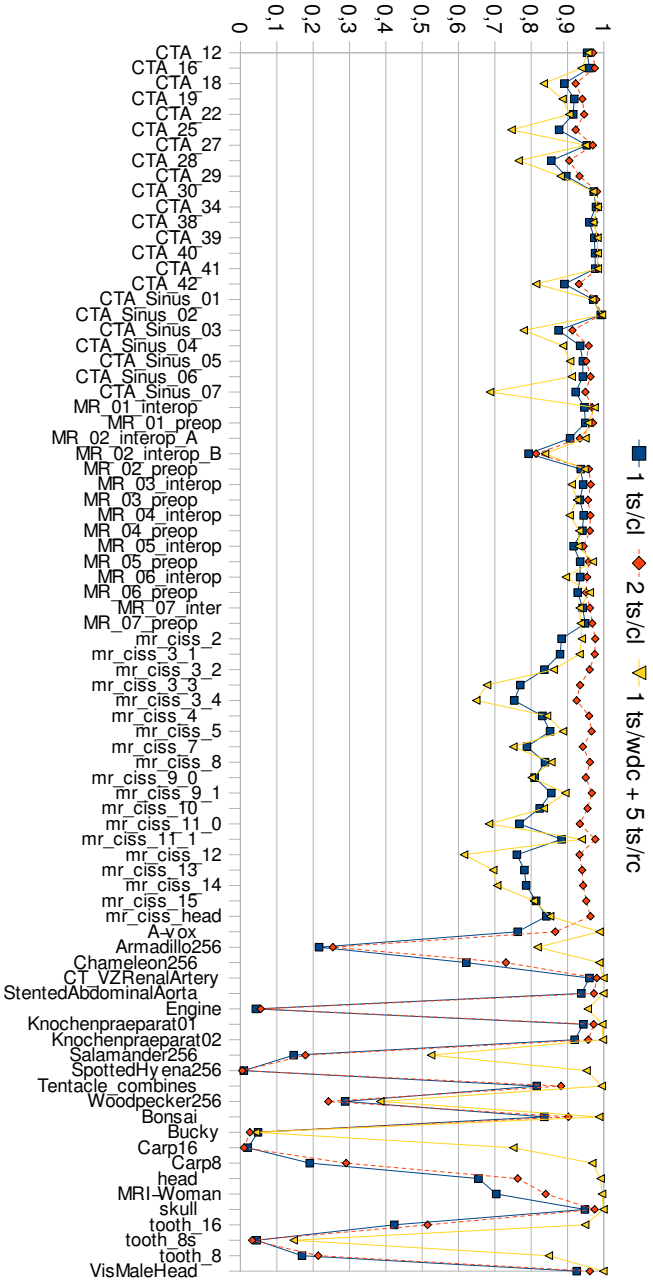


Fig. 13 Increasing the amount of training samples improves the results. Only values of correct outputs are shown.

From Fig. 6, 7, 8 and Table 1 it can be easily concluded that the threshold is a tweaking parameter. Therefore, it should be set high only in specific situations, and in most cases it should be set to a more conservative value (50%-70%).

7 Incorporating Geometric Features

The above described approach based on intensity/derivative histograms has some drawbacks, e.g. in the case of CTA datasets. Here, different subregions of the head can not be distinguished with the standard 2D histogram.

Thus, we propose to use a histogram based on intensity on x-axis, and distance from the "center" on the y-axis instead of intensity derivative. Thus, the y-axis can be thought of as an ordinal number of spherical shell from the center. To make this histogram completely rotationally invariant, the center we use is not the geometric center of the rectangular cuboid, but rather "center of mass", where the voxel intensities serve as "mass" weight (see Fig. 14). The shell thickness is not fixed, but it is proportional to the distance between the center of mass and the farthest corner. This makes this spherical histogram also scale invariant.

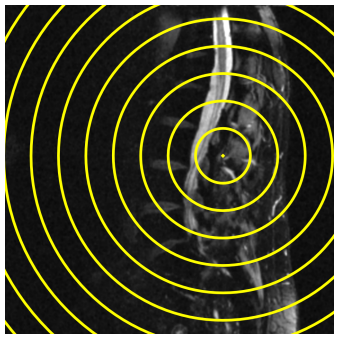
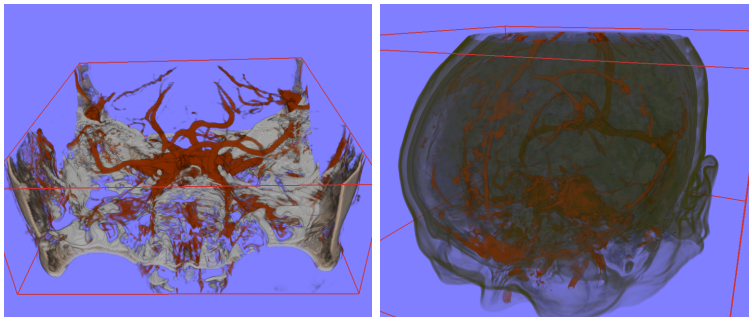


Fig. 14 To create a spherical histogram, a center of mass if calculated first. Then, the histogram is created based on the number of voxels having a certain intensity and distance from the center.

The spherical histogram alone has lower delineating power than the standard histogram. However, if it is applied *after* the standard one, it can further discriminate datasets within a class obtained using standard histogram. This can be accomplished relatively easily. As the only difference is in the type of the histogram, almost everything else remains the same. We first use the standard histogram of a dataset to get a class, then we use spherical histogram to get a subclass, if subclasses are defined for that class. Each subclass has it's own accompanying neural network which is applied to the spherical histogram.

Among the available datasets, it makes sense to further divide CTA class. The first subclass consists of datasets which represent only an inner portion of the head, thus containing only brain, blood vessels and bone. The second subclass consists of the dataset which encompasses the entire head, or a part of the head which also includes skin and surrounding air (see Fig. 15 and 16).



Inner part of a head | Outer part of a head (includes skin)

Fig. 15 Visualizations of two datasets.

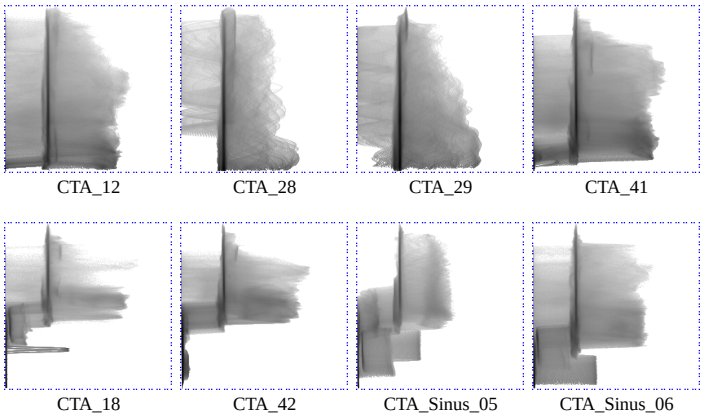


Fig. 16 Some of the spherical histograms from the head CTA series. The first row represents datasets containing an inner part of a head, while the second row is an outer part of, or a whole, head.

The conclusions for the standard histograms also hold true for spherical ones: more training improves results, which particular datasets are chosen for training influences results, higher resolution histogram provides better results etc.

In a classification problem within one class, there are no problematic "miscellaneous" datasets. So if there are significant geometric differences between subclasses, number of misclassifications should be minimal. An example is shown in Fig. 17.

An alternative method for classifying 3D datasets would be to use a downscaled version of the dataset itself instead of the 2D histogram as input to the neural network. This alternative, however, would strongly incorporate geometric aspects, like the individual orientation of the recorded specimen into the classification process. As a result, the training phase would become more difficult, more training samples

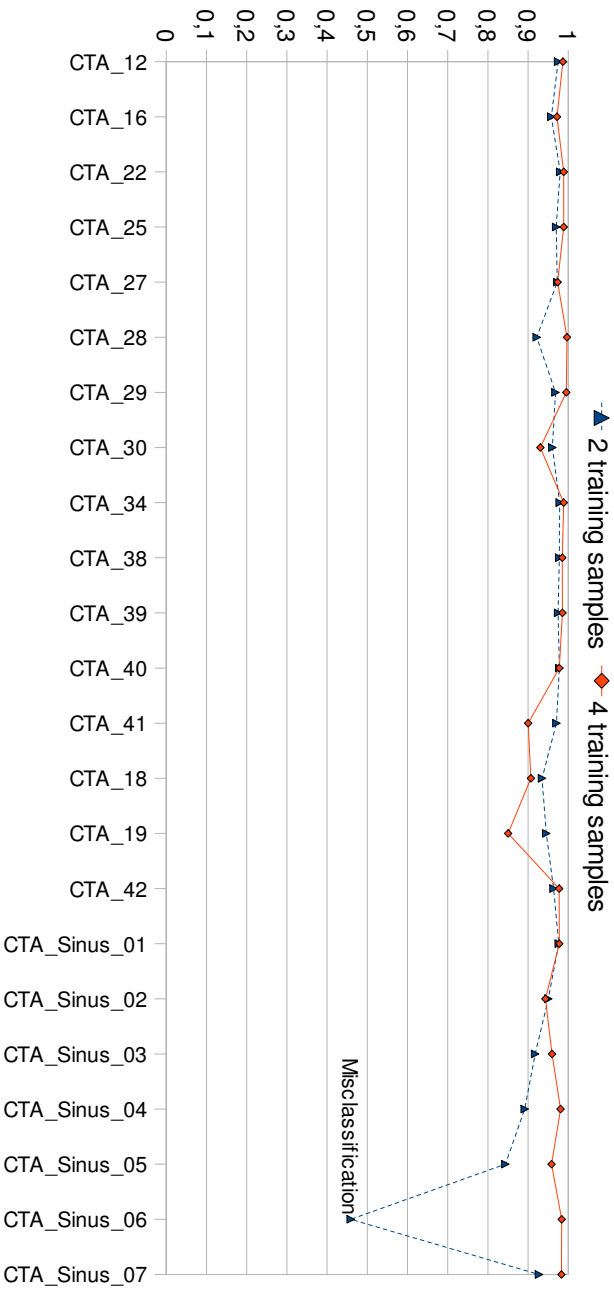


Fig. 17 Application of spherical histograms on top of classical 2D histograms for CTA head datasets: As the subclasses have significant geometric differences, the number of misclassifications is very low.

would be required, and the number of input nodes will increase considerably to achieve a robustness comparable to the described histogram method.

8 Conclusion

We have presented a robust technique to automatically classify 3D volume datasets according to the acquisition sequence, the recorded specimen and sequence-related parameters. The fact that only one training sample per class using standard histograms with the rest-class approach is sufficient to properly classify all the other datasets of the same type is remarkable. Depending in what type of visualization system this method is used, no prior experience might be required at all.

The adaptability of our technique is demonstrated by using a different histogram type (spherical histogram), which includes geometric features thus allowing it to be used as separator for different geometries of datasets. Although spherical histograms can be used on their own, it is best to combine them with standard histograms.

Depending on the amount of information about the data and the application scenario, the architecture of the neural network can be adapted to better suite typical use cases.

The majority of misclassifications are caused by datasets belonging to the miscellaneous group. As researchers, we had many different miscellaneous datasets readily available. However, in production systems the number of datasets in the rest class should be comparably smaller, thus making this method even more appropriate.

An additional advantage of this method is its easy implementation. Successful implementations may be based on one of the many free neural network implementations around. As a result, the benefits of including this method in a suitable production visualization system will easily outweigh the implementation costs.

9 Acknowledgments

Some of the datasets used are courtesy of Knut Eberhardt, Department of Neuroradiology, University of Erlangen, Germany.

References

1. Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *Proc. of the 6th Int. Symposium on Advances in Spatial Databases (SSD)*, pages 207–226, London, UK, 1999. Springer-Verlag.

2. Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proc. of the symposium on Volume visualization (VVS)*, pages 91–98, New York, NY, USA, 1994. ACM Press.
3. Jesús Cerquides, Maite López-Sánchez, Santi Ontañón, Eloi Puertas, Anna Puig, Oriol Pujol, and Dani Tost. Classification algorithms for biomedical volume datasets. In *Current Topics in Artificial Intelligence*, volume 4177/2006 of *Lecture Notes in Computer Science*, chapter 16, pages 143–152. Springer, 2006.
4. Daniel L Chester. Why two hidden layers are better than one. In *Int. Joint Conference on Neural Networks (Washington DC), Lawrence Erlbaum Associates*, pages 265–268, Jan 1990.
5. Jacques de Villiers and Etienne Barnard. Backpropagation neural nets with one and two hidden layers. *IEEE Trans. on Neural Networks*, 4(1):136–141, Jan 1992.
6. Angel del Río, Jan Fischer, Martin Köbele, Dirk Bartz, and Wolfgang Straßer. Augmented Reality Interaction for Semiautomatic Volume Classification. In Erik Kjems and Roland Blach, editors, *Eurographics Workshop on Virtual Environments (EGVE)*, pages 113–120, Aalborg, Denmark, 2005. Eurographics Association.
7. Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware (HWWS)*, pages 9–16, New York, NY, USA, 2001. ACM.
8. Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006.
9. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989.
10. Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proc. of the 1998 IEEE symposium on Volume visualization (VVS)*, pages 79–86, New York, NY, USA, 1998. ACM.
11. Joe Kniss, Gordon Kindlmann, and Chuck Hansen. Interactive Volume Rendering using Multi-dimensional Transfer Functions and Direct Manipulation Widgets. In *Proc. of IEEE Visualization (VIS)*, pages 255–262, 2001.
12. Jens Krüger and Rüdiger Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proc. of IEEE Visualization (VIS)*, pages 287–292, 2003.
13. Yanxi Liu and Frank Dellaert. A classification based similarity metric for 3D image retrieval. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 0:800–805, June 1998.
14. Claes Lundström, Patric Ljung, and Anders Ynnerman. Extending and simplifying Transfer Function design in medical Volume Rendering using local histograms. In *Eurographics / IEEE VGTC Symposium on Visualization (EuroVis)*, pages 263–270, June 2005.
15. Hanspeter Pfister, Jan Hardenbergh, Jim Knittel, Hugh Lauer, and Larry Seiler. The VolumePro real-time ray-casting system. In *Proc. of ACM SIGGRAPH*, pages 251–260, New York, NY, USA, August 1999. ACM Press/Addison-Wesley Publishing Co.
16. Peter Rautek, Stefan Bruckner, and Meister Eduard Gröller. Semantic layers for illustrative volume rendering. *IEEE Trans. on Visualization and Computer Graphics*, 13(6):1336–1343, 2007.
17. Christof Rezk-Salama, Klaus Engel, Michael Bauer, Günther Greiner, and Thomas Ertl. Interactive volume rendering on standard PC graphics hardware using multi-textures and multi-stage rasterization. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware (HWWS)*, pages 109–118, New York, NY, USA, 2000. ACM.
18. Christof Rezk-Salama, Maik Keller, and Peter Kohlmann. High-Level User Interfaces for Transfer Function Design with Semantics. *IEEE Trans. on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 11(5):1021–1028, 2006.
19. Stefan Roettger, Stefan Guthe, Daniel Weiskopf, Thomas Ertl, and Wolfgang Strasser. Smart Hardware-Accelerated Volume Rendering. In *Proc. of the symposium on Data visualisation 2003 (VISSYM)*, pages 231–238, Aire-la-Ville, Switzerland, 2003. Eurographics Association.
20. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, October 1986.

21. Iwo W.O. Serlie, Frans M. Vos, Roel Truyen, Frits H. Post, and Lucas J. van Vliet. Classifying CT image data into material fractions by a scale and rotation invariant edge model. *IEEE Trans. on Image Processing*, 16(12):2891–2904, Dec. 2007.
22. Simon Stegmaier, Magnus Strengert, Thomas Klein, and Thomas Ertl. A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting. In *Proc. of the Int. Workshop on Volume Graphics*, pages 187–195, 2005.
23. Daniel Svozil, Vladimír Kvasnička, and Jiři Pospíchal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39:43–62(20), November 1997.
24. Fan-Yin Tzeng, Eric B. Lum, and Kwan-Liu Ma. A Novel Interface for Higher-Dimensional Classification of Volume Data. In *Proc. of IEEE Visualization (VIS)*, pages 505–512, 2003.
25. Petr Šereda, Anna Vilanova Bartolí, Iwo W. O. Serlie, and Frans A. Gerritsen. Visualization of Boundaries in Volumetric Data Sets Using LH Histograms. *Trans. on Vis. and Comp. Graph.*, 12(2):208–218, 2006.
26. Orion Wilson, Allen Van Gelder, and Jane Wilhelms. Direct Volume Rendering via 3D-textures. Technical Report UCSC-CRL-94-19, Univ. of California, Santa Cruz, 1994.
27. Jiawan Zhang and Jizhou Sun. Automatic classification of MRI images for three-dimensional volume reconstruction by using general regression neural networks. In *IEEE Nuclear Science Symposium Conference Record*, volume 5, pages 3188–3189, Oct. 2003.
28. Dženan Zukić, Andreas Elsner, Zikrija Avdagić, and Gitta Domik. Neural networks in 3D medical scan visualization. In Dimitri Plemenos, editor, *In Proc. of the Int. Conf. on Computer Graphics and Artificial Intelligence (3IA)*, pages 183–190. TEI Athens, May 2008.