# Automatic Subclasses Estimation for a Better Classification with HNNP

Ruth Janning, Carlotta Schatten, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Marienburger Platz 22, 31141 Hildesheim, Germany
`{janning,schatten,schmidt-thieme}@ismll.uni-hildesheim.de`
`http://www.ismll.uni-hildesheim.de`

**Abstract.** Although nowadays many artificial intelligence and especially machine learning research concerns big data, there are still a lot of real world problems for which only small and noisy data sets exist. Applying learning models to those data may not lead to desirable results. Hence, in a former work we proposed a hybrid neural network plait (HNNP) for improving the classification performance on those data. To address the high intraclass variance in the investigated data we used manually estimated subclasses for the HNNP approach. In this paper we investigate on the one hand the impact of using those subclasses instead of the main classes for HNNP and on the other hand an approach for an automatic subclasses estimation for HNNP to overcome the expensive and time consuming manual labeling. The results of the experiments with two different real data sets show that using automatically estimated subclasses for HNNP delivers the best classification performance and outperforms also single state-of-the-art neural networks as well as ensemble methods.

**Keywords:** Image classification, subclasses, convolutional neural network, multilayer perceptron, hybrid neural network, small noisy data.

## 1 Introduction

Although nowadays most problems treated in artificial intelligence and especially in machine learning concern big data, there are still also many real world problems delivering less data than desired for machine learning approaches or delivering such noisy data that classification models deliver undesirable bad results. The first problem type addresses for instance problems in the field of radar like detecting reflections of buried objects in ground penetrating radar images ([5], [6], [7]) or recognizing craters on synthetic aperture radar images of the surface of planets ([10], [11]). The reason for small data sets in this field is the often very expensive and time consuming recording of the images as well as the need of manual labeling. Additionally, most of those data are not publicly available. Furthermore, in many cases it is unavoidable that the recording method or the ambiance induce noise into the data. The second problem type corresponds to challenges like phoneme recognition ([1]). The phonemes spoken by different humans or even by the same human may be very different and on the other hand different phonemes may be very similar. This leads to high intra class variances as well as to small inter class variances. Furthermore, the phoneme extraction process is not exact and induces noise. Hence, the phoneme classification on phoneme data sets like for instance the TIMIT data set ([15]) is challenging. In [8] we presented a plait of hybrid neural networks (HNNP) for improving the classification performance in data sets from those problem types. We observed in [8] large intra class variance and small inter class variance in the investigated radar data (see also fig. 4) and hence we labeled the data for the training and testing within the HNNP structure not only with the main classes but also with subclasses to reduce the intra class variance. This splitting into subclasses was done manually by assigning similar looking examples of one main class to the same subclass. However, the question arises, if this approach really leads to better results and if yes, if this splitting into subclasses can be done also automatically, as the manual labeling is expensive and time consuming or even not possible for data like phonemes. Both questions are answered in this paper. We investigated on the one hand the difference in classification performance of the HNNP approach with only main classes and with more subclasses. On the other hand we propose an approach for automatic subclasses estimation for HNNP and compare the classification performance of HNNP with different numbers of automatically estimated subclasses per main class. The results of our experiments with two different real data sets show that using several subclasses per main class within the HNNP structure leads to better classification results than using just the original main classes. Furthermore, they show that an automatic estimation of the subclasses is even more effective. The main contributions of this paper are: (1) experiments showing the advantage of using subclasses (instead of using only main classes) for HNNP, (2) presentation of an approach of HNNP combined with automatic estimation of subclasses, (3) experiments showing the possibility and effectiveness of using automatically estimated subclasses for HNNP.

## 2 Related Work

Convolutional neural networks are typically used for image classification tasks like handwritten digit classification but also for problems like phoneme recognition which can be also considered as an image classification problem as in [1]. The well known LeNet-5 convolutional neural network for digits recognition is presented in [9]. A more recent convolutional neural network for digit recognition with a simpler and shallower architecture, which we used for our experiments on radar data, is proposed in [13]. In [1] a convolutional neural network for phoneme recognition is described. We used a version of this network for the experiments with the phoneme data. The combination of neural networks to ensembles is investigated e.g. in [12] and an investigation of stacking, or stacked generalization, can be found e.g. in [14]. A combination of more than one convolutional neural network is proposed in [3], where a committee of 7 convolutional neural networks is used and the outputs are averaged. Different kinds of feature sets are used in [4], where 6 feature sets are trained by multilayer perceptrons and the outputs are merged by using another multilayer perceptron. Our HNNP approach ([8], fig. 1)

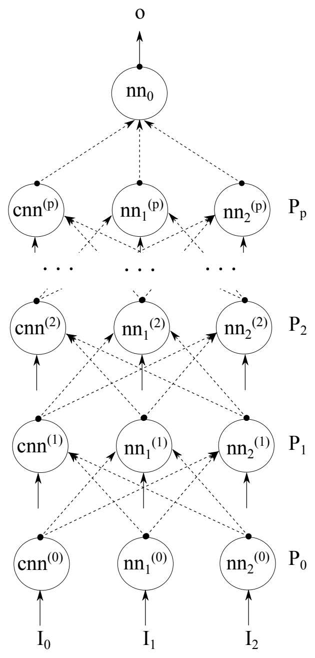

**Fig. 1.** Architecture of the hybrid neural network plait (HNNP). The plait is composed of $p+1$ layers $P_0, P_1, \ldots, P_p$. Each layer contains $k+1$ different neural networks (here $k=2$) according to the $k+1$ different information sources $I_0, I_1, \ldots, I_k$ for the input. $I_0$ corresponds to the input image and the appropriate learning model is a convolutional neural network (*cnn*). The learning models for the other information sources $I_1, \ldots, I_k$ are multilayer perceptrons ($nn_1, nn_2, \ldots, nn_k$). In every layer from $P_1$ on the neural networks are retrained with additional input from the former layer. After the last layer $P_p$ a further multilayer perceptron ($nn_0$) is attached to achieve one common output vector $\mathbf{o}$ delivering the final classification result.

combines the above mentioned approaches by using different feature sets and a committee of a convolutional neural network and multilayer perceptrons. However, for the HNNP approach different kinds of neural networks with adapted architecture are retrained interactively within a plait structure using additional side information gained before and during the retraining for a further improvement. Our approach for automatically estimating subclasses for HNNP is similar to the approach in [16] based on local clustering, however the focus of [16] lies on imbalanced class distributions, whereas we address the problem of high intra class variances and small inter class variances.

## 3 HNNP with Automatically Estimated Subclasses

In the following sections we will present the HNNP approach with automatically estimated subclasses. First we will introduce HNNP in section 3.1 and subsequently we will explain in section 3.2 how to estimate automatically the subclasses for HNNP and how to integrate them into the HNNP architecture.

### 3.1 HNNP

The hybrid neural network plait (HNNP) approach is based on two methods for incorporating additional side information: (1) integrating different information sources delivering different feature sets, (2) retraining the neural networks applied to this feature sets interactively within one common structure with additional improved side information. Point (1) implies that different neural networks are used for the different information sources. The different information sources are on the one hand the original information source – the pixel values of the image – and on the other hand sources of additional side information. The learning model for the original information source is, according to state-of-the-art approaches (see sec. 2), a convolutional neural network (*cnn*). For the training of the feature sets of the other information sources fully connected multilayer perceptrons are used. Point (2) gives the plait the complex structure and incorporates in every retraining step improved additional side information. The plait (fig. 1) is composed of several layers in each of which the networks are retrained by considering the classification decisions of the other networks from the former layer. A simplified version (with just 1 convolution layer instead of 2) of the *cnn*

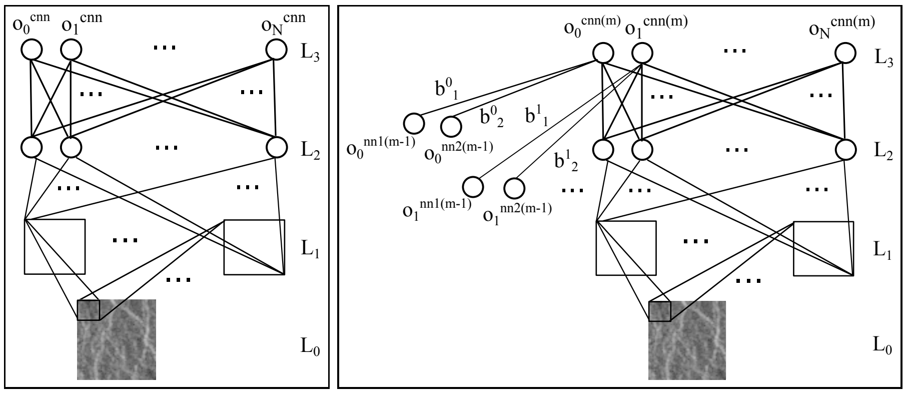

**Fig. 2.** Left: Architecture of a simplified *cnn*. It is composed of 4 layers $L_0, L_1, L_2, L_3$, where $L_0$ is the input layer fed with the image, $L_1$ is a convolution layer followed by a fully connected layer ($L_2$) and the output layer $L_3$. Right: Architecture of the *cnn* adapted to the plait structure with 3 information sources. In contrast to left the output neurons of this *cnn* get additional input from the other networks of the plait layer before.

used for HNNP is pictured in fig. 2. In these kind of networks the convolution layers and the subsampling serve as a local feature extractor and the fully connected layers as a trainable classifier. The output layer delivers an output vector $\mathbf{\hat{o}^{cnn}} := (\hat{o}_0^{cnn}, \ldots, \hat{o}_N^{cnn}), -1 \leq \hat{o}_i^{cnn} \leq 1, i = 0, \ldots, N$, where the actual output $\hat{o_i}^{cnn}$ of the $i$th neuron in output layer $n$ corresponds to

$$\hat{o_i}^{cnn} = \tanh(\sum_{l=0}^{N_{n-1}} w_n^{il} x_{n-1}^l) , \tag{1}$$

with $N_{n-1}$ as the number of neurons in layer $n-1$, $w_n^{il}$ as the weight of the $l$th connection between neuron $i$ and the neurons in layer $n-1$, $x_{n-1}^l$ as the output of the $l$th neuron in layer $n-1$ and tanh (hyperbolic tangent) as activation function with an output within the interval $[-1, 1]$. Each neuron of the output layer is assigned to one class in $\{C_0, \ldots, C_N\}$. The classification result $C_{\mathrm{argmax}_i \hat{o_i}^{cnn}}$ corresponds to the class $C_i$ assigned to the output neuron with maximum output value $\hat{o_i}^{cnn}$. As usual in such architectures, the error in the last layer $n$ for pattern $\mathcal{P}$ to be minimized is

$$E_n^{\mathcal{P}} := \frac{1}{2} \sum_{i=0}^{N} (\hat{o_i}^{cnn} - o_i^{cnn})^2 , \tag{2}$$

with $\hat{o_i}^{cnn}$ as actual output (eq. (1)) and $o_i^{cnn}$ as target output. In opposite to the *cnn*, fully connected multilayer perceptrons fulfill only a role as classifier (without local feature extraction) but use the same formulas for $\mathbf{\hat{o}^{nn_i}}, E_n^{\mathcal{P}}$. *cnn*s and multilayer perceptrons are interweaved in the HNNP structure like in a plait (fig. 1), which is enabled by adapting their architectures. In these new architectures (fig. 2) – using the example of the *cnn* – every neuron $o_i^{cnn^{(m)}}$ of the output layer $L_n$ of a *cnn* in plait layer $P_m$ is additionally connected to the outputs $\hat{o}_i^{nn_1^{(m-1)}}, \hat{o}_i^{nn_2^{(m-1)}}, \ldots, \hat{o}_i^{nn_k^{(m-1)}}$ of the $k$ other networks $nn_1, nn_2, ..., nn_k$ from the previous plait layer $P_{m-1}$. This leads to new output formulas for the neural networks within the plait. In the case of $k$ information sources for additional side information, the new output formulas for the $k+1$ adapted networks *cnn* and $nn_x, x = 1, \ldots, k$ are as follows:

$$\hat{o}_i^{cnn^{(m)}} = \tanh(\sum_{l=0}^{N_{n-1}} w_n^{il} x_{n-1}^l + \sum_{j=1}^{k} b_j^i \hat{o}_i^{nn_j^{(m-1)}}) , \tag{3}$$

$$\hat{o}_i^{nn_x^{(m)}} = \tanh(\sum_{l=0}^{N_{n-1}} w_n^{il} x_{n-1}^l + b_0^i \hat{o}_i^{cnn^{(m-1)}} + \sum_{j=1, j \neq x}^{k} b_j^i \hat{o}_i^{nn_j^{(m-1)}}) . \tag{4}$$

In this way each neural network learns to which degree it should consider the classification decisions of all other networks from the previous plait layer. The number $p+1$ of plait layers is a hyper parameter. The final component of the HNNP is an additional fully connected multilayer perceptron ($nn_0$ in fig. 1). $nn_0$ is fed with the outputs of every neural network in the last plait layer $P_p$ and delivers one common final classification decision.

## 3.2 Automatic Subclasses Estimation for HNNP

Originally, the number $N$ of output neurons in the last layer of the neural networks within the HNNP structure refers to the number $M$ of main classes. However, as mentioned before, for our experiments in [8] we split the main classes into subclasses, which were estimated manually in a time consuming process. Hence, we propose an approach for an automatic estimation of subclasses for HNNP. Our approach applies a $K$-means clustering on the histograms to the set of examples of each main class. Each of the $K$ clusters found is then assigned to one of $K$ subclasses. $K$ is, besides $p$, a further hyper parameter of our method. This approach leads for each neural network within the HNNP to a different number $N$ of output neurons, namely $N := M \cdot K$ – instead of $N := M$ if just main classes are considered – as there is one output neuron per subclass. By using more output neurons within the HNNP structure a more fine granulated improved side information is passed through the architecture. Only $nn_0$ maps

```
Input:
Data sets $\mathcal{D}^{train} := \{d_0^{train}, \dots, d_{|\mathcal{D}^{train}|}^{train}\}$
          $= \bigcup_{i=0}^{M} \mathcal{D}_{main_i} = \bigcup_{i=0}^{M} \{d_0^{main_i}, \dots, d_{|\mathcal{D}_{main_i}|}^{main_i}\}$
with main class labels $(main(d_0^{train}), \dots, main(d_{|\mathcal{D}^{train}|}^{train}))$ and
$\mathcal{D}^{test} := \{d_0^{test}, \dots, d_{|\mathcal{D}^{test}|}^{test}\}$,
number $K$ of subclasses per main class, number $p$ of plait layers.

Variables:
$(\mathcal{D}_{main_i}^0, \dots, \mathcal{D}_{main_i}^K)$,                               // list of clusters of subclasses of main class $i$
$(sub(d_0^{main_i}), \dots, sub(d_{|\mathcal{D}_{main_i}|}^{main_i}))$   // list of subclass labels of main class $i$
$\mathcal{D}^{train'}$, $\mathcal{L}^{train'}$                                  // train data and label set with subclasses
$(main(d_0^{test}), \dots, main(d_{|\mathcal{D}^{test}|}^{test}))$       // list of predicted main class labels

1. for $i = 0$ to $M$ do
       $((\mathcal{D}_{main_i}^0, \dots, \mathcal{D}_{main_i}^K), (sub(d_0^{main_i}), \dots, sub(d_{|\mathcal{D}_{main_i}|}^{main_i})))$
           $= K$-means(Histograms($\mathcal{D}_{main_i}$), $K$);
   end for

2. $\mathcal{D}^{train'} := \bigcup_{i=0}^{M} \bigcup_{j=0}^{K} \mathcal{D}_{main_i}^j$; $\mathcal{L}^{train'} := \bigcup_{i=0}^{M} (sub(d_0^{main_i}), \dots, sub(d_{|\mathcal{D}_{main_i}|}^{main_i}))$

3. trainHNNP($\mathcal{D}^{train'}, \mathcal{L}^{train'}, p, K$);

4. $(main(d_0^{test}), \dots, main(d_{|\mathcal{D}^{test}|}^{test}))$ = applyHNNP($\mathcal{D}^{test}, p, K$);

5. return $(main(d_0^{test}), \dots, main(d_{|\mathcal{D}^{test}|}^{test}))$;
```

**Fig. 3.** HNNP approach with automatically estimated subclasses

its input finally to the number $M$ of main classes, i.e. for $nn_0$ holds $N := M$. The whole approach is shown in fig. 3. The input to the approach is on the one hand the train set $\mathcal{D}^{train} := \{d_0^{train}, \ldots, d_{|\mathcal{D}^{train}|}^{train}\}$ with a main class label for every example and on the other hand a test set $\mathcal{D}^{test} := \{d_0^{test}, \ldots, d_{|\mathcal{D}^{test}|}^{test}\}$. The examples $d_l^{train}, l = 0 \ldots |\mathcal{D}^{train}|$ and $d_l^{test}, l = 0 \ldots |\mathcal{D}^{test}|$ consist of as many feature vectors as there are information sources $(k+1)$. A further input to the approach are the hyper parameters $K$ and $p$. In step 1 a $K$-means algorithm is applied to the histograms of train set examples of each main class. The $K$-means algorithm finds $K$ clusters in each main class set and assigns the appropriate subclass label to every example. The HNNP is trained in step 3 with these $M \cdot K$ gained subclasses. To predict the labels of test examples in step 4 the HNNP is applied to the test set. The output of the whole approach in step 5 is a predicted main class label for every test example. The described approach is proven by experiments presented in the next section 4.

## 4 Experiments

In the experiments we investigated three different questions: (1) Does HNNP outperform the single networks as well as the ensemble methods? (2) How does the classification performance behave if only main classes are used or if several subclasses are used? (3) How does the classification performance behave if automatically estimated subclasses are used? We focus in this paper on the two last questions, as the first one was already answered also for other data sets in [8]. We applied HNNP to 2 different real data sets (sec. 4.1). The first data set comes from a set of synthetic aperture radar (SAR) images of the surface of Venus, available at the UCI Machine Learning Repository [2]. The data were collected by the Magellan spacecraft ([10], [11]). The classification task in this data is the identification of volcanoes. To this SAR data set we applied HNNP on the one hand with 2 main classes (volcano, noise) and on the other hand with 10 manually estimated subclasses as well as with 10 automatically estimated subclasses. The second data set consists of examples of phonemes, or of the *Mel Frequency Cepstral Coefficients* (MFCC) vectors of speech signals interpreted as images by combining several subsequent vectors respectively (see [1]). The data are gained from the TIMIT data set ([15]) by choosing two different phonemes ('iy','n'). To this small TIMIT data set we applied HNNP with 2 main classes according to the 2 chosen phonemes and with 6, 10 and 20 automatically estimated subclasses. For both data sets we used the same experimental settings: the HNNP approach is compared to the five baselines *cnn*, $nn_1$, $nn_2$, *majority ensemble* and *stacking ensemble*. *cnn* is a single convolutional neural network (sec. 3.1) fed with the normalized pixel values of the images to classify. $nn_1$, $nn_2$ are single fully connected multilayer perceptrons (sec. 3.1) with input feature sets coming from the appropriate additional side information described in section 4.1. *Majority ensemble* classifies according to the majority vote of *cnn*, $nn_1$ and $nn_2$. *Stacking ensemble* learns to combine the classification decisions of *cnn*, $nn_1$ and $nn_2$ by using the subsequent multilayer perceptron $nn_0$. We used in the experiments $k + 1 = 3$ information sources, $p + 1 = 3$ plait layers, $N = 2$ or 10 for the SAR data set and $N = 2, 6, 10$ or 20 (sub)classes for the small TIMIT data set and we conducted a 5-fold cross validation for every data set.

## 4.1 Data Sets

For the SAR data set we have chosen 5 images of the Magellan data. We extracted 898 examples (451 volcano, 447 noise). The examples are $29 \times 29$ pixel images (fig. 4), which build the input for the *cnn*. The *cnn* used for the experiments with the SAR data set is a 5-layer *cnn* with 2 convolution layers (1 with 10 maps and 1 with 50 maps) with integrated subsampling, based on the architecture described in [13]. In the fully connected layer there are 50 neurons and in the output layer, according to the value of $N$, 2 or 10 neurons. The input feature sets for $nn_1$ and $nn_2$ come from statistical information. $nn_1$ is fed with 4 histograms of 16 gray values, each of which represents one quarter of the input image. Accordingly, $nn_1$ has $4 \cdot 16$ neurons in its input layer, 64 neurons in the hidden layer and 2 or 10 neurons in the output layer. $nn_2$ is fed with the number of pixels with a light gray value (a value within the upper quarter of all gray values of the image) per area, where an area is one of 100 areas of the image (partitioned by a $10 \times 10$ grid). $nn_2$ has 100 neurons in the input layer, 100 neurons in the hidden layer and 2 or 10 neurons in the output layer.

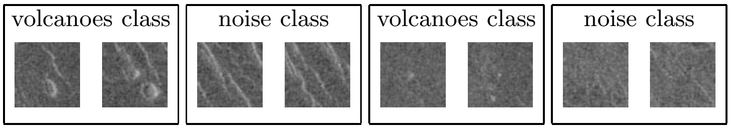

**Fig. 4.** Examples of 4 different automatically estimated subclasses of the SAR data

For the small TIMIT data set we have chosen 400 examples of the class of phonem 'iy' and 400 examples of the class of phonem 'n'. The examples are $43 \times 19$ pixel images consisting of MFCC vectors, which build the input for the *cnn*. The *cnn* used for the experiments with the small TIMIT data set is a 5-layer *cnn* with 1 convolution layer and 1 max pooling layer, based on the architecture described in [1]. In the convolution layer each $8 \times 8$ window of the input image is mapped to one value in each of the 6 maps. In the max pooling layer the maximum value of each 2nd $6 \times 6$ window of a convolution map is inserted in a pooling map. In the fully connected layer there are 100 neurons and in the output layer, according to the value of $N$, 2, 6, 10 or 20 neurons. The input feature sets for $nn_1$ and $nn_2$ are equivalent to the ones for the SAR data set.

## 4.2 Results

The results of the 5-fold cross validation experiments are shown in table 1. The first observation and answer to question (1) is that HNNP outperforms the

**Table 1.** Results of the 5-fold cross validation for the SAR and small TIMIT data set: Classification test errors (%) of the 5 baselines *cnn*, $nn_1$, $nn_2$, *majority ensemble* and *stacking ensemble* and the HNNP (standard deviations in brackets) with different numbers of classes

| data | # classes | *cnn* | $nn_1$ | $nn_2$ | majority | stacking | HNNP |
|---|---|---|---|---|---|---|---|
| SAR | 2 | 23.72 (3.77) | 22.38 (3.46) | 21.04 (2.81) | 18.48 (3.93) | 18.71 (4.12) | **17.61** (0.89) |
| SAR | 10 manual | 23.84 (4.07) | 23.50 (4.39) | 19.82 (3.13) | 17.60 (3.41) | 16.15 (1.74) | **14.03** (1.30) |
| SAR | 10 automatic | 26.50 (1.95) | 18.92 (2.27) | 20.05 (2.77) | 16.62 (3.21) | 14.94 (2.01) | **12.03** (2.56) |
| TIMIT | 2 | 26.75 (2.14) | 36.13 (3.55) | 30.00 (1.47) | 26.00 (2.71) | 24.63 (2.52) | **16.38** (2.39) |
| TIMIT | 6 automatic | 29.88 (2.18) | 35.88 (5.28) | 27.75 (1.22) | 25.38 (2,45) | 21.25 (1.71) | **13.75** (2.54) |
| TIMIT | 10 automatic | 27.88 (4.32) | 35.75 (4.04) | 30.88 (1.86) | 26.38 (1.90) | 22.13 (1.69) | **12.50** (0.99) |
| TIMIT | 20 automatic | 25.38 (3.24) | 34.88 (3.17) | 32.88 (3.18) | 26.00 (2.19) | 20.75 (1.90) | **12.50** (3.03) |

single networks as well as the ensemble methods. But in this work we focus on the answers of both other questions. Hence, for the SAR data set we investigated the difference in the classification performance of HNNP with only 2 main classes and HNNP with 10 manually or automatically estimated subclasses. Table 1 shows that using subclasses causes no big difference in the classification performance for the single neural networks but for *stacking* and HNNP the classification performance is improved by using subclasses. In the experiments with the small TIMIT data set we compared the classification performances of HNNP with different numbers of automatically estimated subclasses (as manually subclasses estimation is not possible). Already the use of 6 subclasses leads to a classification performance improvement of HNNP. Using 10 subclasses instead still improves the classification performance, whereas using 20 subclasses does not lead to a further improvement.

## 5 Conclusion

In this work we investigated the impact of using subclasses for HNNP and we proposed an approach of estimating subclasses for HNNP automatically. The experiments show that using (automatically estimated) subclasses within HNNP is able to improve the classification performance significantly. Next steps would be to apply our approach to multiclass problems like e.g. full phoneme recognition. We expect also for such problems a significant performance improvement by using HNNP with automatically estimated subclasses.

**Acknowledgments.** This work is co-funded by the EU project iTalk2Learn.

## References

1. Abdel-Hamid, O., Mohamed, A., Jiang, H., Penn, G.: Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4277–4280 (2012)
2. Bache, K., Lichman, M.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2013), `http://archive.ics.uci.edu/ml`
3. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Convolutional Neural Network Committees For Handwritten Character Classification. In: International Conference on Document Analysis and Recognition (2011)
4. Cruz, R.M.O., Cavalcanti, G.D.C., Ren, T.I.: Handwritten Digit Recognition Using Multiple Feature Extraction Techniques and Classifier Ensemble. In: 17th International Conference on Systems, Signals and Image Processing (2010)
5. Janning, R., Horváth, T., Busche, A., Schmidt-Thieme, L.: GamRec: A Clustering Method Using Geometrical Background Knowledge for GPR Data Preprocessing. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H. (eds.) AIAI 2012. IFIP AICT, vol. 381, pp. 347–356. Springer, Heidelberg (2012)
6. Janning, R., Horváth, T., Busche, A., Schmidt-Thieme, L.: Pipe Localization by Apex Detection. In: Proceedings of the IET International Conference on Radar Systems (Radar 2012), Glasgow, Scotland (2012)
7. Janning, R., Busche, A., Horváth, T., Schmidt-Thieme, L.: Buried Pipe Localization Using an Iterative Geometric Clustering on GPR Data. In: Artificial Intelligence Review. Springer (2013), doi:10.1007/s10462-013-9410-2
8. Janning, R., Schatten, C., Schmidt-Thieme, L.: HNNP – A Hybrid Neural Network Plait for Improving Image Classification with Additional Side Information. In: Proceedings of the IEEE International Conference on Tools With Artificial Intelligence (ICTAI) 2013, Washington DC, USA, pp. 24–29 (2013)
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
10. Pettengill, G.H., Ford, P.G., Johnson, W.T.K., Raney, R.K., Soderblom, L.A.: Magellan: Radar Performance and Data Products. Science 252, 260–265 (1991)
11. Saunders, R.S., Spear, A.J., Allin, P.C., Austin, R.S., Berman, A.L., Chandlee, R.C., Clark, J., Decharon, A.V., Dejong, E.M.: Magellan Mission Summary. Journal of Geophysical Research Planets 97(E8), 13067–13090 (1992)
12. Sharkey, A.J.C., Sharkey, N.E.: Combining diverse neural nets. The Knowledge Engineering Review 12(3), 231–247 (1997)
13. Simard, P.Y., Steinkraus, D., Platt, J.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In: External Link International Conference on Document Analysis and Recognition (ICDAR), pp. 958–962. IEEE Computer Society, Los Alamitos (2003)
14. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research 10, 271–289 (1999)
15. TIMIT Acoustic-Phonetic Continuous Speech Corpus, `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1`
16. Wu, J., Xiong, H., Chen, J.: COG: local decomposition for rare class analysis. Data Mining and Knowledge Discovery 20, 191–220 (2010)