EasyChair Preprint

№ 3206

A Committee of Convolutional Neural Networks for Image Classification in the Concurrent Presence of Feature and Label Noise

Stanisław Kaźmierczak and Jacek Mańdziuk

April 20, 2020

A Committee of Convolutional Neural Networks for Image Classification in the Concurrent Presence of Feature and Label Noise

Stanisław Kaźmierczak(✉) [0000−0002−8981−1592] and
Jacek Mańdziuk[0000−0003−0947−028X]

Faculty of Mathematics and Information Science, Warsaw University of Technology,
Warsaw, Poland
{s.kazmierczak,mandziuk}@mini.pw.edu.pl

Abstract. Image classification has become a ubiquitous task. Models trained on good quality data achieve accuracy which in some application domains is already above human-level performance. Unfortunately, real-world data are quite often degenerated by the noise existing in features and/or labels. There are quite many papers that handle the problem of either feature or label noise, separately. However, to the best of our knowledge, this piece of research is the first attempt to address the problem of concurrent occurrence of both types of noise. Basing on the MNIST, CIFAR-10 and CIFAR-100 datasets, we experimentally proved that the difference by which committees beat single models increases along with noise level, no matter it is an attribute or label disruption. Thus, it makes ensembles legitimate to be applied to noisy images with noisy labels. The aforementioned committees' advantage over single models is positively correlated with dataset difficulty level as well. We propose three committee selection algorithms that outperform a strong baseline algorithm which relies on an ensemble of individual (nonassociated) best models.

Keywords: Committee of classifiers, Ensemble learning, Label noise, Feature noise, Convolutional neural networks.

1 Introduction

Standard image classification task consists in assigning a correct label to an input sample picture. In the most widely-used supervised learning approach, one trains a model to recognize the correct class by providing input-output image-label pairs (training set). In many cases, the achieved accuracy is very high [6, 38], close to or above human-level performance [7, 15].

The quality of real-world images is not perfect. Data may contain some noise defined as anything that blurs the relationship between the attributes of an instance and its class [16]. There are mainly two types of noise considered in the literature: feature (attribute) noise and class (label) noise [11, 27, 36].

Despite the fact that machine algorithms (especially those based on deep architectures) perform on par with humans or even better on high-quality pictures, their performance on distorted images is noticeably worse [10]. Similarly, label noise may potentially result in many negative consequences, e.g. deterioration of prediction accuracy along with an increase of model's complexity, size of a training set, or length of a training process [11]. Hence, it is necessary to devise methods that reduce noise or are able to perform well in its presence. The problem is furthermore important considering the fact that the acquisition of accurately labeled data is usually time-consuming, expensive and often requires a substantial engagement of human experts [2].

There are many papers in the literature which tackle the problem of label noise. Likewise, a lot of works have been dedicated to studying attribute noise. However, to the best of our knowledge, there are no papers that consider the problem of feature and label noise occurring simultaneously. In this paper, we present the method that successfully deals with the concurrent presence of attribute noise and class noise.

1.1 The main contribution

Encouraged by the promising results of ensemble models applied to label noise and CNN-based architectures utilized to handle noisy images we examine how a committee of CNN classifiers (each trained on the whole dataset) deal with noisy images marked with noisy labels. With regard to the common taxonomy, there are four groups of ensemble methods [22]. The first one relates to *data selection mechanisms* aiming to provide different subset for every single classifier to be trained on. The second one refers to *the feature level*. Methods among this group select features that each model uses. The third one, *the classifier level group*, comprises algorithms that have to determine the base model, the number of classifiers, other types of classifiers, etc. The final one refers to *the combination of classifiers level* where an algorithm has to decide how to combine models' individual decisions to make a final prediction. In this study, we assume having a set of well-trained CNNs which make the ultimate decision by means of soft voting (averaging) scheme [26]. We concentrate on the task of finding an optimal or near-optimal model committee that deals with concurrent presence of attribute and label noise in the image classification problem. In summary, the main contribution of this work is threefold:

- addressing the problem of simultaneously occurring feature and label noise which, to the best of our knowledge, is a novel unexplored setting;
- designing three methods of building committees of classifiers which outperform a strong baseline algorithm that employs a set of individually best models;
- proving empirically that a margin of ensembles gain over the best single model rises along with an increase of both noise types, as well as dataset difficulty, which makes proposed approaches specifically well-suited to the case of noisy images with noisy labels.

The remainder of this paper is arranged as follows. Section 2 provides a literature review, with considerable emphasis on methods addressing label noise and distorted images. Section 3 introduces the proposed novel algorithms for classifiers' selection. Sections 4 and 5 describe the experimental setup and analysis of results, respectively. Finally, brief conclusions and directions for further research are presented in the last section.

2 Related literature

Many possible sources of label noise have been identified in the literature [11], e.g. insufficient information provided to the expert [11, 3], expert (human or machine) mistakes [25, 32], the subjectivity of the task [17] or communication problems [36, 3]. Generally speaking, there are three main approaches to dealing with label noise [11]. The first one is based on algorithms that are naturally robust to class noise. This includes ensemble methods like bagging and boosting. It has been shown in [9] that bagging performs generally better than boosting in this task. The second group of methods relies on data cleansing. In this approach, corrupted instances are identified before the training process starts off and some kind of filter (e.g. voting or partition filter [4, 37] which is deemed easy, cheap and relatively solid) is applied to them. Ultimately, the third group consists of methods that directly model label noise during the learning phase or were specifically designed to take label noise into consideration [11].

In terms of images, the key reasons behind feature noise are faults in sensor devices, analog-to-digital converter errors [30] or electromechanical interferences during the image capturing process [14]. State-of-the-art approaches to deal with feature noise are founded on deep architectures. In [24] CNNs (LeNet-5 for MNIST and an architecture similar to the base model C of [33] for CIFAR-10 and SVHN datasets) were used to handle noisy images. Application of a denoising procedure (Non-Local Means [5]) before the training phase improves classification accuracy for some types and levels of noise. In [28] several combinations of denoising autoencoder (DAE) and CNNs were proposed, e.g. DAE-CNN, DAE-DAE-CNN, etc. The paper states that properly combined DAE and CNN achieve better results than individual models and other popular methods like SMV [35], sparse rectifier neural network [13] or deep belief network [1].

3 Proposed algorithms

As mentioned earlier, classification results are obtained via a soft voting scheme. More specifically, probabilities of particular classes from single CNNs are summed up and a class with the highest cumulative value is ultimately selected (see Fig. 1).

Many state-of-the-art results in image classification tasks are achieved by an ensemble of well-trained networks that were not selected in any way [7, 18, 21]. In [31] the authors went further and noticed that limiting ensemble size just to two best-performing models increased accuracy in their case. We adopted that

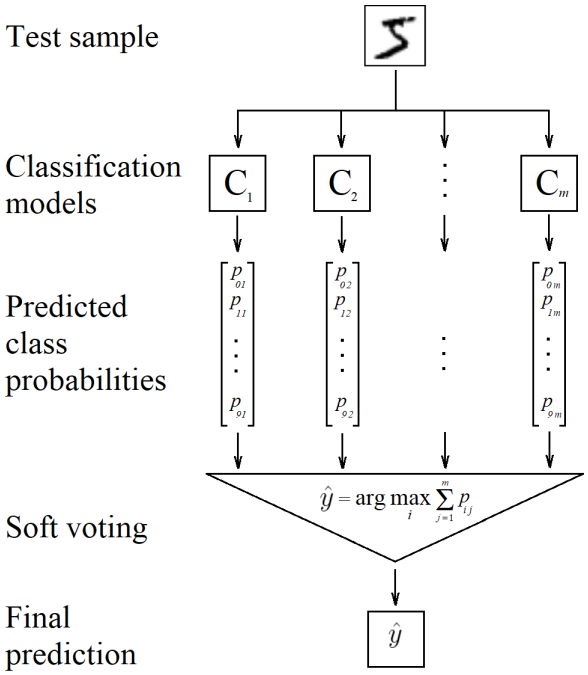

Fig. 1. The concept of the soft voting approach for the 10-class problem.

idea to the algorithm called (for the purpose of this study) *top-n*, which serves as a benchmark in our experiments. First, all models are sorted in descending order according to their accuracies. Then, ensembles constituted by k best networks where k ranges from 1 to the number of available models are created. Finally, the committee with the best score on the validation dataset is chosen. Algorithm 1 summarizes the procedure.

The first algorithm proposed in this paper, called *2-opt-c*, was inspired by the local search technique commonly applied to solving the Traveling Salesman Problem (TSP) [8]. The original *2-opt* formulation looks for any two nodes of the current salesman's route which, if swapped, would shorten the route length. It works until no improvement is made within a certain number of sampling trials. The *2-opt-c* algorithm receives an initial committee as an input and in each step modifies it by adding/subtracting/exchanging one or two elements in the way that maximizes accuracy on the validation set. Thus, there are eight possible atomic operations listed in Algorithm 2. The procedure operates until neither of the operations improves performance. The *1-opt-c* works in a very similar way but is limited to three operations that modify only one element in a committee (adding, removal and swap). The *top-n-2-opt-c* and *top-n-1-opt-c* operate likewise besides being initialized with the output of the *top-n* procedure, not an empty committee.

Algorithm 1: *top-n*

input : M_{all}–all available models
output: C_{best}–selected committee

```
1  C_best ← ∅;
2  C_curr ← ∅;
3  acc_best = 0;
4  M_sorted = sort(M_all);        // sort models descending by accuracy
5  for i ← 1 to size(M_sorted) do
6  |   C_curr ← C_curr ∪ M_sorted[i];
7  |   acc_curr ← accuracy(C_curr);
8  |   if acc_curr > acc_best then
9  |   |   acc_best ← acc_curr;
10 |   |   C_best ← C_curr;
11 |   end
12 end
```

Algorithm 2: *2-opt-c*

input : M_{all}–all available models, C_0–initial committee
output: C_{best}–selected committee

```
1  C_best ← C_0;
2  acc_best ← accuracy(C_best);
3  while acc_best rises do
4  |   acc_curr = 0;
5  |   acc_curr, C_curr ← add(C_best, M_all, acc_curr);
6  |   acc_curr, C_curr ← remove(C_best, acc_curr);
7  |   acc_curr, C_curr ← swap(C_best, M_all, acc_curr);
8  |   acc_curr, C_curr ← addTwo(C_best, M_all, acc_curr);
9  |   acc_curr, C_curr ← removeTwo(C_best, acc_curr);
10 |   acc_curr, C_curr ← addAndSwap(C_best, M_all, acc_curr);
11 |   acc_curr, C_curr ← removeAndSwap(C_best, M_all, acc_curr);
12 |   acc_curr, C_curr ← swapTwice(C_best, M_all, acc_curr);
13 |   if acc_curr > acc_best then
14 |   |   acc_best ← acc_curr;
15 |   |   C_best ← C_curr;
16 |   end
17 end
```

Another algorithm, called *stochastic*, relies on the fact that the performance of the entire ensemble depends on diversity among individual component classifiers, on the one hand, and the predictive performance of single models, on the other hand [29]. The pseudocode of the algorithm is presented in Algorithm 3. In the ith epoch, a committee size ranges from i to $i + range$ with the expected value equal to $i + \frac{range}{2}$. This property is assured by the formula in line 7 which increases the probability of adding a new model along with decreasing ensemble size and vice versa. Figure 2 illustrates this relationship. In each step, one model

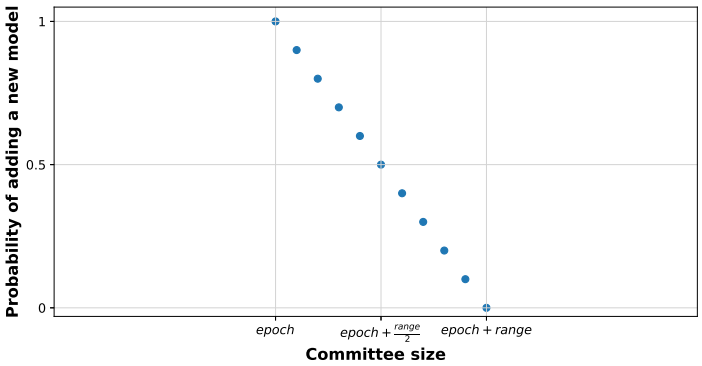

Fig. 2. Probability of adding a new model to the committee in the *stochastic* algorithm.

is either added or removed. In the first scenario a model with the best individual performance is appended to the committee with probability t_a (lines 11-14) or a model which minimizes the maximum correlation between any model from the committee and itself is added with probability $1 - t_a$ (lines 15-18). Analogously, if the algorithm decides to decrease a committee size it removes the weakest model with probability t_r (lines 22-25) or a model which minimizes the highest correlation between any two models in the committee with probability $1 - t_r$ (lines 26-29). In each epoch, the algorithm performs N_i iterations to explore the solution space. A correlation between two models is measured by the Pearson correlation coefficient calculated on probability vectors obtained from predictions on the validation set.

4 Experimental setup

4.1 MNIST, CIFAR-10 and CIFAR-100 datasets

As a benchmark, we selected three datasets with a diversified difficulty level. MNIST database contains a large set of 28x28 grayscale images of handwritten digits (10 classes) and is commonly used in machine learning experiments [23]. The training set and the test set are composed of 60 000 and 10 000 images, respectively.

CIFAR-10 [20] is another popular image dataset broadly used to assess machine learning/computer vision algorithms. It contains 60 000 32x32 color images in 10 different classes. The training set includes 50 000 pictures, while the test set – 10 000 ones.

The CIFAR-100 dataset is similar to CIFAR-10. It comprises 60 000 images with the same resolution and three color channels as well. The only difference is the number of classes–CIFAR-100 has 100 of them, thus yielding 600 pictures per class.

Algorithm 3: *Stochastic* algorithm

input : M_{all}–all available models, N–number of epochs, N_i–number of iterations within an epoch, t_a–probability threshold below which the strongest model is added, t_r–probability threshold below which the weakest model is removed, r–range of possible committee sizes in an epoch

output: C_{best}–selected committee

$C_{curr} \leftarrow \emptyset$;
$C_{best} \leftarrow \emptyset$;
$acc_{best} \leftarrow 0$;
$M_{left} \leftarrow M_{all}$;
for $i \leftarrow 0$ **to** $N - 1$ **do**
  for $j \leftarrow 1$ **to** N_i **do**
    $p_a \leftarrow 1 - (size(C_{curr}) - i)/r$;
    $u \leftarrow$ generate from the uniform distribution $\mathcal{U}(0, 1)$;
    if $u < p_a$ **then**
      $u_a \leftarrow$ generate from the uniform distribution $\mathcal{U}(0, 1)$;
      if $size(C_{curr}) == 0$ *or* $u_a < t_a$ **then**
        $m_a \leftarrow getStrongestModel(M_{left})$;
        $C_{curr} \leftarrow C_{curr} \cup m_a$;
        $M_{left} \leftarrow M_{left} \setminus m_a$;
      else
        // select model from M_{left} which minimizes maximum
        // correlation between any model from C_{curr} and itself
        $m_a \leftarrow getMarginallyCorrelatedModel(C_{curr}, M_{left})$;
        $C_{curr} \leftarrow C_{curr} \cup m_a$;
        $M_{left} \leftarrow M_{left} \setminus m_a$;
      end
    else
      $u_r \leftarrow$ generate from the uniform distribution $\mathcal{U}(0, 1)$;
      if $size(C_{curr}) == 1$ *or* $u_r < t_r$ **then**
        $m_r \leftarrow getWeakestModel(C_{curr})$;
        $C_{curr} \leftarrow C_{curr} \setminus m_r$;
        $M_{left} \leftarrow M_{left} \cup m_r$;
      else
        // select model from M_{curr} which minimizes maximum
        // correlation between any two models in C_{curr}
        $m_r \leftarrow getMaximallyCorrelatedModel(C_{curr})$;
        $C_{curr} \leftarrow C_{curr} \setminus m_r$;
        $M_{left} \leftarrow M_{left} \cup m_r$;
      end
    end
    $acc_{curr} \leftarrow accuracy(C_{curr})$;
    if $acc_{curr} > acc_{best}$ **then**
      $acc_{best} \leftarrow acc_{curr}$;
      $C_{best} \leftarrow C_{curr}$;
    end
  end
end

Table 1. CNN architectures used for MNIST, CIFAR-10 and CIFAR-100.

Layer	Type	#maps & neurons	kernel/pool size
1	convolutional	32 maps of 32x32 neurons (CIFAR) 32 maps of 28x28 neurons (MNIST)	3x3
2	batch normalization		
3	convolutional	32 maps of 32x32 neurons (CIFAR) 32 maps of 28x28 neurons (MNIST)	3x3
4	batch normalization		
5	max pooling		2x2
6	dropout (20%)		
7	convolutional	64 maps of 16x16 neurons (CIFAR) 64 maps of 14x14 neurons (MNIST)	3x3
8	batch normalization		
9	convolutional	64 maps of 16x16 neurons (CIFAR) 64 maps of 14x14 neurons (MNIST)	3x3
10	batch normalization		
11	max pooling		2x2
12	dropout (20%)		
13	convolutional	128 maps of 8x8 neurons (CIFAR) 128 maps of 7x7 neurons (MNIST)	3x3
14	batch normalization		
15	convolutional	128 maps of 8x8 neurons (CIFAR) 128 maps of 7x7 neurons (MNIST)	3x3
16	batch normalization		
17	max pooling		2x2
18	dropout (20%)		
19	dense	128 neurons	
20	batch normalization		
21	dropout (20%)		
22	dense	10 neurons (MNIST, CIFAR-10) 100 neurons (CIFAR-100)	

4.2 CNN architectures

Individual classifiers are in the form of a convolutional neural network composed of VGG blocks (i.e. a sequence of convolutional layers, followed by a max pooling layer) [31], additionally enhanced by adding dropout [34] and batch normalization [18]. All convolutional layers and hidden dense layer have ReLU as an activation function and their weights were initialized with *He normal initializer* [15]. Softmax was applied in the output layer while the initial weights were drawn from the *Glorot uniform distribution* [12]. Table 1 summarizes the CNNs architectures. Please note that slight differences in architectures for MNIST, CIFAR-10 and CIFAR-100 are caused by distinct image sizes and numbers of classes in the datasets. Without any attribute or label noise, a single CNN achieved approximately 99%, 83% and 53% accuracy on MNIST, CIFAR-10 and CIFAR-100, respectively.

4.3 Training protocol

The following procedure was applied to all three datasets. Partition of a dataset into training and testing subsets was predefined as described in Section 4.1. At the very beginning, all features (RGB values) were divided by 255 to fit $[0, 1]$ range. Images were neither preprocessed nor formatted in any other way. From the training part, we set aside 5 000 samples as a validation set for a single models training and another 5 000 samples for a committee performance comparison. From now on when referring to the training set we would mean all training samples excluding the above-mentioned 10 000 samples used for validation purposes.

To create noisy versions of the datasets we degraded features of the three copies of each dataset by adding the Gaussian noise with standard deviation $\sigma = 0.1, 0.2, 0.3$, respectively. The above distortion was applied to the training set, two validation sets and the test set. All affected values were then clipped to $[0, 1]$ range. Next, for the original datasets and each of the three copies influenced by the Gaussian noise, another three copies were created and their training set labels were altered with probability $p = 0.1, 0.2, 0.3$, respectively. If a label was selected to be modified, a new value was chosen from the discrete uniform distribution $\mathcal{U}\{0, 9\}$. If the new label value equaled the initial value, then a new label was drawn again, until the sampled label was different from the original one. Hence, we ended up with 16 different versions of each dataset in total (no feature noise plus three degrees of feature noise multiplied by analogous four options regarding the label noise).

The second step was to train CNNs on each of the above-mentioned dataset versions. We set the maximum number of epochs to 30 and batch size to 32. In the case of four consecutive epochs with no improvement on the validation set, training was stopped and weights from the best epoch were restored. Adam optimizer [19] was used to optimize the cross-entropy loss function: $-\sum_{c=1}^{K} y_{o,c} \log(p_{o,c})$ where K is the number of classes, $y_{o,c}$ – a binary indicator whether c is a correct class for observation o, and $p_{o,c}$ – a predicted probability that o is from class c. The learning rate was fixed to 0.001.

4.4 Algorithms parametrization

The *stochastic* algorithm was run with the following parameters: the number of available models – 25, the number of epochs – 16, the number of iterations within an epoch – 1000, probability threshold below which the strongest model is added – 0.5, probability threshold below which the weakest model is removed – 0.5, range of possible committee sizes in each epoch – 10. Please note that the above parametrization allows the algorithm to consider any possible committee size (from 1 to 25). Other analyzed algorithms are parameterless.

5 Experimental results

This section presents experimental results of testing various ensemble selection algorithms. For each pair $(\sigma, p) \in \{0, 0.1, 0.2, 0.3\} \times \{0, 0.1, 0.2, 0.3\}$ 50 CNNs

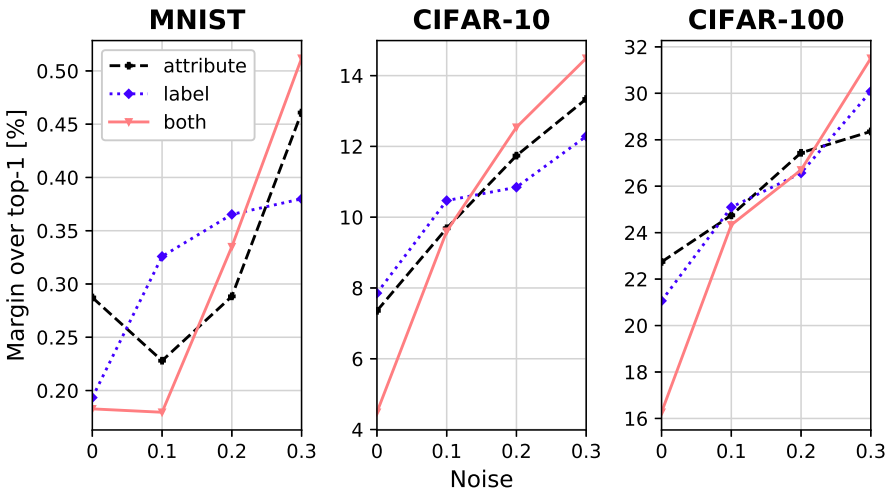

Fig. 3. Relative accuracy margin that committees gained over the *top-1* algorithm.

were independently trained from which 25 were drawn to create one instance of experiment. Each experiment was repeated 20 times to obtain reliable results. In the whole study, we assume not having any knowledge regarding either the type or the level of noise the datasets are affected by.

Figure 3 depicts the relative accuracy margin that committees gained over the *top-1* algorithm which selects the best individual model from the whole library of models. Scores are averaged over *top-n, 2-opt-c, 1-opt-c, top-n-2-opt-c, top-n-1-opt-c* and *stochastic* algorithms. For example, if the best individual model achieves 80% accuracy while the mean accuracy of ensembles found by analyzed algorithms equals 88% then the relative margin of ensembles over *top-1* is equal to 10%. *Attribute* curves refer to computations where all scores within particular attribute noise level are averaged over label noise (four values for every attribute noise level). *Label* curves are created analogously–for particular label noise all scores within specific label noise are averaged over attribute noise. *Both* curves concern increasing noise level concurrently on both attributes and labels by the same amount (i.e. with $\sigma = p$). For example, 0.2 value on the x-axis refers to $\sigma = 0.2$ attribute noise and $p = 0.2$ label noise.

Two main conclusions can be drawn from the plots. First, a committee margin rises along with an increase of both noise types (separately and jointly as well). Secondly, a difference increases further for more demanding datasets. In case of MNIST, the difference is less than 1% while when concerning CIFAR-100 the margin amounts to more than 20% for 0.1 and 0.2 noise level and around 30% for 0.3 noise level. Figure 4 illustrates in a concise, aggregated way how algorithms perform in comparison with *top-n* for various noise levels. Values on the y-axis indicate how much (percentage-wise) the margin achieved by *top-n* over *top-1* is better/worse than the margin attained by the rest of the algorithms. For

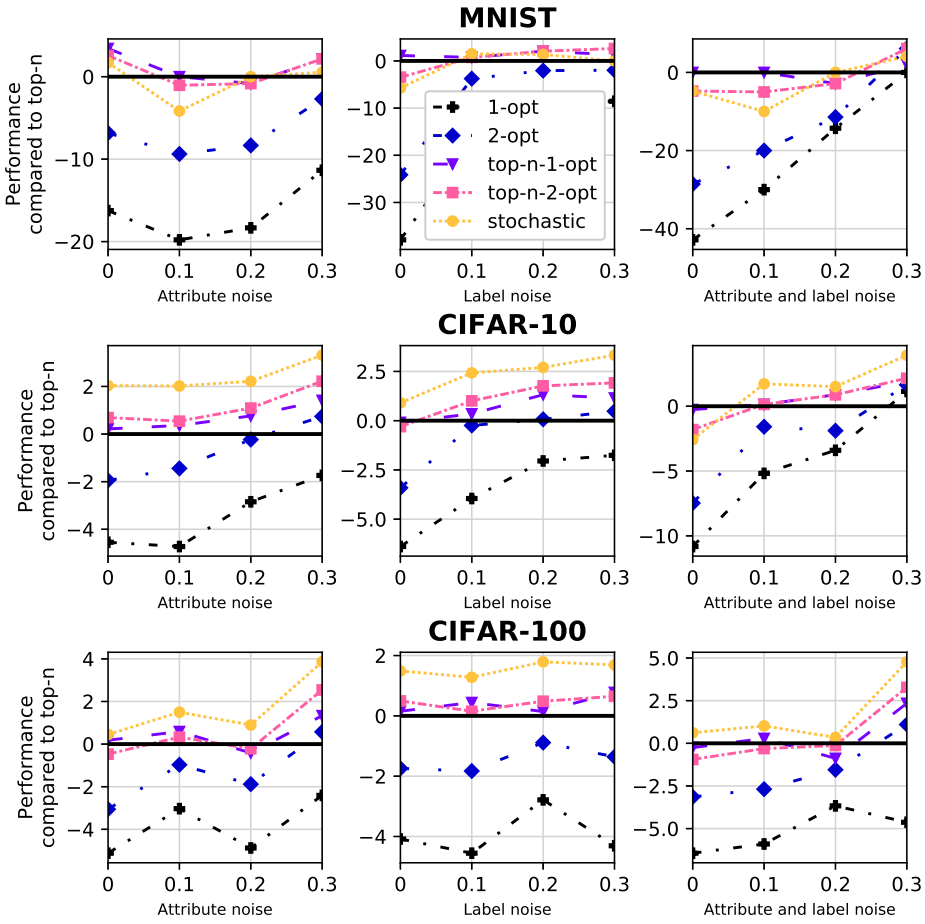

Fig. 4. Performance of designed algorithms contrasted with *top-n* results.

example, if accuracies of *top-1*, *top-n* and *stochastic* methods are 80%, 85% and 85.5%, respectively then the value for *stochastic* algorithm amounts to 10% in that case since 0.5% constitutes 10% of 5%. Line $y = 0$ refers to *top-n*. For each dataset the leftmost plot, for the given level of *attribute* noise, presents scores averaged over the *label* noise (four values for each level). Likewise, in the middle plot, for the given level of *label* noise, the scores averaged over the four values of *attribute* noise are depicted. In the third plot, the scores are not averaged since x-values refer to both *attribute* and *label* noise. From the first row of plots, which refers to the MNIST dataset, it stems that there are huge relative differences in results achieved by the algorithms which, furthermore, vary a lot between noise levels. This phenomenon is caused by the fact that all errors for all noise levels are below 1% in MNIST. Thus, even very little absolute difference between

scores may be reflected in high relative value (one instance constitutes 0.01% of test set size). Therefore, it is hard to draw any vital conclusions for this dataset other than a general observation that for a relatively easy dataset the results of all algorithms are close to each other.

From the plots related to CIFAR-10 and CIFAR-100, one can see that three of our algorithms noticeably surpassed the *top-n* one. The *stochastic* method achieved better results on all noise levels. The only yellow dot below zero refers to no noise case on either attributes and labels. Both *top-n-2-opt-c* and *top-n-1-opt-c* also beat *top-n* in most of the cases. Another observation is that our algorithms are positively correlated with a noise level in the sense that the attained margin rises along with increasing noise.

We have also analyzed 35-sized libraries of the models. The relationships between results achieved by the algorithms remain similar to those with 25 models, only the absolute accuracy values are slightly higher. It is not surprising since algorithms have a wider choice of models and may keep more of them in a committee. As the last remark, we noticed that *2-opt-c* and *1-opt-c* obtained very high accuracy on validation sets (greater than *top-n-2-opt-c* and *top-n-1-opt-c*, respectively) however it was not reflected on test sets. This observation suggests that one has to be careful when dealing with methods whose performance is measured solely on the validation set with neglecting models' diversity, as such committees tend to overfit.

6 Conclusions and future work

The main goal of this paper is to address the problem of concurrently occurring feature and label noise in the image classification task which, to the best of our knowledge, has not been considered in the existing literature. To this end, we propose five novel ensemble selection algorithms among which four are inspired by the local optimization algorithm derived from the TSP and one employs a stochastic search. Three out of five methods outperform the strong baseline reference algorithm that applies a set of individually selected best models (*top-n*). We have also empirically proven that a margin gained by the committees over the best single model rises along with an increase of both types of noise as well as with raising dataset difficulty, thus making proposed ensembles specifically well-suited to noisy images with noisy labels.

There is a couple of lines of inquiry worth pursuing in the future. Firstly, one may experiment with the parametrization of the *stochastic* algorithm (a range of possible committee sizes in an epoch, a tradeoff between individual performance and ensemble diversity, etc.). Analysis of other correlation measures could be insightful as well. Secondly, all our algorithms operate on probability vectors, which allows us to assume that they would achieve similar results in other domains and are not limited to noisy images only. Finally, this paper addresses only one aspect of ensembling - models selection. Other areas which could be considered when forming a committee model, briefly mentioned in Section 1.1, are also worth investigation.

References

1. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in NIPS. pp. 153–160 (2007)
2. Breve, F.A., Zhao, L., Quiles, M.G.: Semi-supervised learning from imperfect data through particle cooperation and competition. In: The 2010 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2010)
3. Brodley, C.E., Friedl, M.A.: Identifying mislabeled training data. Journal of Artificial Intelligence Research **11**, 131–167 (1999)
4. Brodley, C.E., Friedl, M.A., et al.: Identifying and eliminating mislabeled training instances. In: Proceedings of the National Conference on Artificial Intelligence. pp. 799–805 (1996)
5. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 2, pp. 60–65. IEEE (2005)
6. Chan, T.H., Jia, K., Gao, S., Lu, J., Zeng, Z., Ma, Y.: Pcanet: A simple deep learning baseline for image classification? IEEE Transactions on Image Processing **24**(12), 5017–5032 (2015)
7. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3642–3649. IEEE (2012)
8. Croes, G.A.: A method for solving traveling-salesman problems. Operations Research **6**(6), 791–812 (1958)
9. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40**(2), 139–157 (2000)
10. Dodge, S., Karam, L.: A study and comparison of human and deep learning recognition performance under visual distortions. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN). pp. 1–7. IEEE (2017)
11. Frénay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE Transactions on Neural Networks and Learning Systems **25**(5), 845–869 (2013)
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010)
13. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 315–323 (2011)
14. González, R., Woods, R.: Digital image processing. isbn: 9780131687288. Prentice Hall **60** (2008)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015)
16. Hickey, R.J.: Noise modelling and evaluating learning from examples. Artificial Intelligence **82**(1-2), 157–179 (1996)
17. Hughes, N.P., Roberts, S.J., Tarassenko, L.: Semi-supervised learning of probabilistic models for ecg segmentation. In: The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. vol. 1, pp. 434–437. IEEE (2004)

18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in NIPS. pp. 1097–1105 (2012)
22. Kuncheva, L.I.: Combining pattern classifiers: methods and algorithms. John Wiley & Sons (2014)
23. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist **2** (2010)
24. Nazaré, T.S., da Costa, G.B.P., Contato, W.A., Ponti, M.: Deep convolutional neural networks and noisy images. In: Iberoamerican Congress on Pattern Recognition. pp. 416–424. Springer (2017)
25. Pechenizkiy, M., Tsymbal, A., Puuronen, S., Pechenizkiy, O.: Class noise and supervised learning in medical domains: The effect of feature extraction. In: 19th IEEE Symposium on Computer Based Medical Systems (CBMS'06). pp. 708–713. IEEE (2006)
26. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization **30**(4), 838–855 (1992)
27. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1), 81–106 (1986)
28. Roy, S.S., Hossain, S.I., Akhand, M., Murase, K.: A robust system for noisy image classification combining denoising autoencoder and convolutional neural network. International Journal of Advanced Computer Science and Applications **9**(1), 224–235 (2018)
29. Sagi, O., Rokach, L.: Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(4), e1249 (2018)
30. Shapiro, L.G., Stockman, G.C.: Computer vision. Prentice Hall (2001)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
32. Snow, R., O'connor, B., Jurafsky, D., Ng, A.Y.: Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing. pp. 254–263 (2008)
33. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
34. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
35. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research **11**(Dec), 3371–3408 (2010)
36. Zhu, X., Wu, X.: Class noise vs. attribute noise: A quantitative study. Artificial Intelligence Review **22**(3), 177–210 (2004)
37. Zhu, X., Wu, X., Chen, Q.: Bridging local and global data cleansing: Identifying class noise in large, distributed data datasets. Data Mining and Knowledge Discovery **12**(2-3), 275–308 (2006)
38. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)