Boosted Convolutional Neural Networks

Mohammad Moghimi[1]
mmoghimi@cs.cornell.edu
Mohammad Saberian*[2]
esaberian@netflix.com
Jian Yang[3]
jianyang@yahoo-inc.com
Li-Jia Li*[5]
lijiali@cs.stanford.edu
Nuno Vasconcelos[4]
nvasconcelos@ucsd.edu
Serge Belongie[1]
sjb344@cornell.edu

[1] Cornell Tech, New York, NY
Cornell University, Ithaca, NY
[2] Netflix, Los Gatos, CA
[3] Yahoo Research, Sunnyvale, CA
[4] UC San Diego, La Jolla, CA
[5] Snapchat, Los Angeles, CA

Abstract

In this work, we propose a new algorithm for boosting Deep Convolutional Neural Networks (BoostCNN) to combine the merits of boosting and modern neural networks. To learn this new model, we propose a novel algorithm to incorporate boosting weights into the deep learning architecture based on least squares objective function. We also show that it is possible to use networks of different structures within the proposed boosting framework and BoostCNN is able to select the best network structure in each iteration. This not only results in superior performance but also reduces the required manual effort for finding the right network structure. Experiments show that the proposed method is able to achieve state-of-the-art performance on several fine-grained classification tasks such as bird, car, and aircraft classification.

1 Introduction

Deep convolutional neural networks (CNNs) have recently produced outstanding results in learning image representations for several vision tasks including image classification [15, 24, 41] and object detection [12, 16, 31]. These neural networks usually consist of several components including convolutional, fully connected and pooling layers. By stacking several of these layers, deep CNNs are capable of learning complex features that are highly invariant and discriminant [24, 37, 38, 39, 44]. Krizhevsky et al. [24] proposed an eight layer deep network architecture that produced state-of-the-art performance on the ImageNet Challenge [32]. Given the success of this network, it has been applied widely to many other problems such as video classification [20], face recognition [40] and action recognition [28]. However, the optimal image representation for each computer vision task is unique and finding the optimal deep CNN structure for extracting that representation is a challenging

* work done while at Yahoo Research.

problem. There are some general guidelines, inspired by biological vision systems, for designing these deep networks such as putting convolutional layers early in the network. These guidelines, however, do not address how to set structural parameters of the networks such as the number of layers, number of units in each layer or the size of the receptive fields in the pooling layers. As a result designing a new network can take weeks of trial-and-error.

To address this design challenge, we propose to combine boosting and deep CNNs. The idea is to leverage the ability of boosting to combine the strengths of multiple weaker learners to simplify the complicated design process of deep CNNs. This characteristic of boosting has been shown to be very significant for several tasks in Computer Vision. For example in the Viola and Jones face detector [42] boosting searches over a very large pool of simple classifiers (thresholds on Haar wavelet responses) and trains a classifier that is able to detect a complex object such as human face. Our approach is also motivated by the successful combination of decision trees [29] and boosting. As in the case of CNNs, the design of decision trees is not straightforward, e.g., what is the optimal depth of a trees? How many branches should there be per node? What is the optimal criteria for splitting tree nodes? These are all important aspects of learning a decision tree and much research was devoted to these questions in the 1990s. For example there are several proposals for node splitting criteria such as Gini impurity, information gain [26] and there are several tree induction algorithms such as CART [3] or C4.5 [30]. Today, boosted decision trees [9, 10, 11] have eliminated most of these problems via an optimal weighted vote over decision trees that are individually sub-optimal. Similar to the success story of boosted decision trees, we believe that combination of boosting and deep learning can significantly reduce the challenges in designing deep networks.

The idea of boosting neural networks or, more generally, working with ensembles of neural networks has been around for many years; see for example [1, 5, 6, 7, 8, 14, 19, 34, 35, 36, 46]. All these works demonstrated advantages of using an ensemble of networks over using a single large network. These works, however, either rely on simple averaging of several networks or rely on some heuristic weighting mechanism to impose boosting weights in the training process. In addition, some of these methods do not scale to the object recognition tasks in the modern computer vision literature.

In this work, we propose a new algorithm for boosting deep networks (BoostCNN) to combine the merits of boosting and deep CNNs. To learn this new model, we propose a novel algorithm to incorporate boosting weights into the deep learning architecture based on least squares objective functions. Experiments show that the proposed method is able to achieve state-of-the-art results on several fine-grained classification task as such bird, car, and aircraft classification. Finally we show that it is possible to use networks of different structures within the proposed boosting framework and BoostCNN is able to find the best network in each iteration. This not only results in superior performance, but also eliminates the required manual effort for finding the right network structure.

2 Multiclass boosting

We start with a brief overview of multiclass boosting. A multiclass classifier is a mapping $F : \mathcal{X} \rightarrow \{1 \dots M\}$ that maps an example x_i to its class label $z_i \in 1 \dots M$. Since this is not a continuous mapping, a classifier $F(x)$ is commonly trained through learning a *predictor*

$f : \mathcal{X} \to \mathbb{R}^d$ for some d. The classifier $F(x)$ is then implemented by

$$F(x) = \arg\max_{k=1\ldots M} \langle y_k, f(x) \rangle, \tag{1}$$

where y_k is a unit vector that represents label of the k^{th} class and $\langle \cdot, \cdot \rangle$ is the dot product.

For example in binary classification, labels are $y_1 = +1$ and $y_2 = -1$ and (1) is equivalent to the popular $F(x) = sign[f(x)]$ decision rule. Another example is one-vs-all multiclass classifiers. In this method, for each class k, a predictor $f_k(x) : \mathcal{X} \to \mathbb{R}$ is trained to discriminate between examples of that class versus others. In order to classify a new example $\hat{x}$, $f_k(\hat{x})$ is computed for all $k = 1 \ldots M$ and the class of largest predictor is declared as the label. This procedure is equivalent to (1) by defining $f(x) = [f_1(x) \ldots f_M(x)] \in \mathbb{R}^M$ and $y_k = \mathbf{1}_k \in \mathbb{R}^M$, i.e. k^{th} element is one and the rest are zero. In general, the choice of labels are not restricted to the canonical basis in $\mathbb{R}^M$ and it is possible to use any set of M distinct unit vectors $y_1 \ldots y_M \in \mathbb{R}^d$ [33]. For simplicity, in the rest of this paper, we assume that $d = M$ and $y_k = \mathbf{1}_k$.

Multiclass boosting is a method that combines several *multiclass predictors* $g_i : \mathcal{X} \to \mathbb{R}^d$ to form a strong committee $f(x)$ of classifiers, i.e., $f(x) = \sum_{t=1}^{N} \alpha_t g_t(x)$ where g_t and α_t are the weak learner and coefficient selected at t^{th} boosting iteration. There are several approaches for multiclass boosting such as [17, 27, 33] and we use GD-MCBoost method of [33] in this paper. GD-MCBoost trains a boosted predictor $f(x)$ by minimizing risk of classification

$$\mathcal{R}[f] = E_{X,Z}\{L(z, f(x))\} \approx \frac{1}{\|\mathcal{D}\|} \sum_{(x_i,z_i)\in\mathcal{D}} L(z_i, f(x_i)), \tag{2}$$

where $\mathcal{D}$ is a set of training examples and

$$L(z_i, f(x_i)) = \sum_{j=1, j\neq z_i}^{M} e^{-\frac{1}{2}[\langle y_{z_i}, f(x_i)\rangle - \langle y_j, f(x_i)\rangle]}. \tag{3}$$

The minimization is via gradient descent in function space. GD-MCBoost starts with $f(x) = \mathbf{0} \in \mathbb{R}^d \ \forall x$ and iteratively computes the directional derivative of the risk, (2), for updating $f(x)$ along the direction of $g(x)$

$$\delta\mathcal{R}[f; g] = \left.\frac{\partial \mathcal{R}[f + \varepsilon g]}{\partial \varepsilon}\right|_{\varepsilon=0} = -\frac{1}{2\|\mathcal{S}\|} \sum_{(x_i,z_i)\in\mathcal{D}} \sum_{j=1}^{M} g_j(x_i) w_j(x_i), \tag{4}$$

where

$$w_k(x_i) = \begin{cases} -e^{-\frac{1}{2}[f_{z_i}(x_i) - f_k(x_i)]} & k \neq z_i \\ \sum_{j=1|j\neq k}^{M} e^{-\frac{1}{2}[f_{z_i}(x_i) - f_j(x_i)]} & k = z_i \end{cases} \tag{5}$$

GD-MCBoost then selects/trains a weak learner g^* that minimizes (4),

$$g^* = \arg\min_{g\in\mathcal{G}} \delta\mathcal{R}[f; g], \tag{6}$$

and compute the optimal step size along g^*,

$$\alpha^* = \arg\min_{\alpha\in\mathbb{R}} \mathcal{R}[f + \alpha g^*], \tag{7}$$

using a line search. The boosted predictor $f(x)$ is finally updated as

$$f = f + \alpha^* g^*. \tag{8}$$

3 Boosting convolutional neural networks

Combining boosting and convolutional neural networks is possible by using convolutional neural networks (CNN) as weak learners in the GD-MCBoost algorithm. In this case, the weak learner $g(x) \in \mathbb{R}^M$ is a deep network, e.g., Alex-Net [24] without the last softmax loss layer. Using CNNs as weak learners (base-learners) requires training CNNs to minimize (4) in each iteration of boosting. However, the learning algorithms for CNNs are typically based on minimizing the error rate, e.g., log-likelihood or softmax and these objective functions are independent of the boosting weights and can be very different from (4). A possible solution is to replace the softmax loss layer with a layer to directly optimize (4) in the back-propagation algorithm. However, this is not practical and diverges quickly as (4) is unbounded, e.g., scaling $g(x)$ can make it infinite.

In order to address this issue, we first note that $\delta\mathcal{R}[f;g]$ of (4) is equivalent to

$$\delta\mathcal{R}[f;g] = -\frac{1}{2\|\mathcal{D}\|}\sum_{(x_i,z_i)\in\mathcal{D}}\langle g(x_i)w(x_i)\rangle, \tag{9}$$

where $\langle\cdot,\cdot\rangle$ is the Euclidean dot product. This shows (4) is a summation of dot products between the network output $g(x_i)$ and boosting weights $w(x_i)$. Therefore (4) measures the similarity between those vectors and thus the optimal network output, $g^*(x_i)$, has to be aligned with the boosting weights, i.e.,

$$g^*(x_i) = \beta w(x_i), \tag{10}$$

where $\beta > 0$. Note that the exact value of β is not crucial during the training of the network because $g^*(x)$ will be scaled appropriately by the optimal α in (7) afterwards. Therefore without loss of generality we can assume that $\beta = 1$ and the optimal network output has to replicate boosting weights. This is equivalent to train a network $g(x) = [g_1(x)\dots g_M(x)] \in \mathbb{R}^M$ to minimize the square error loss

$$\mathcal{L}_{se}(w,g) = \sum_{x_i\in\mathcal{D}}\sum_{j=1}^{M}(g_j(x_i) - w_j(x_i))^2. \tag{11}$$

Using $\mathcal{L}_{se}$ loss function for learning a CNN, the back-propagated derivatives are

$$-\frac{\partial\mathcal{L}_{se}}{\partial g_k(x_i)} = 2(w_j(x_i) - g_k(x_i)). \tag{12}$$

The proposed algorithm (BoostCNN) is summarized in Algorithm 1. It starts by initializing $f(x) = \mathbf{0} \in \mathbb{R}^M$. In each iteration, it first computes the boosting weights, $w(x) \in \mathbb{R}^M$ according to (5) and trains a network $g^*(x)$ to minimize the squared error between the network output and boosting weights using (11). Once the network is trained, BoostCNN finds the boosting coefficient by minimizing the boosting loss, (7), and adds the network to the ensemble according to $f(x) = f(x) + \nu\alpha^* g^*(x)$ where $\nu \in (0,1]$ is the shrinkage parameter which has been shown to act as a regularizer for boosting algorithms [11].

3.1 Implementation Details:

We implemented the proposed algorithm using the Caffe library [18]. For training networks to minimize the square loss, (11), we replaced the softmax loss layer of the popular networks

Algorithm 1 BoostCNN

Input: number of classes M, number of boosting, iterations N_b, shrinkage parameter ν, and dataset $\mathcal{D} = \{(x_1, z_1), ..., (x_n, z_n)\}$ where $z_i \in \{1 \dots M\}$ is label of example x_i.
Init: set $f(x) = \mathbf{0} \in \mathbb{R}^M$.
for $t = 1$ to N_b **do**
  compute $w(x_i)$ for all x_i, using (5).
  train a network $g^*(x)$ to optimize (11).
  find the optimal coefficient, α^*, using (7).
  update $f(x) = f(x) + \nu\alpha^* g^*(x)$.
end for
Output: predictor $f(x)$

such as Alex-Net [24] with the Euclidean loss layer. In each iteration of boosting we need to compute the weights, according to (5), however, computing $f(x)$ for each example requires running all of the networks in the current ensemble and can be very expensive. To avoid this problem, we note that if $w^t(x_i)$ are the weights at iteration t then using (5) and (8)

$$w_k^{t+1}(x_i) = \begin{cases} -e^{-\frac{1}{2}\nu\alpha^*[g^*_{z_i}(x_i) - g^*_k(x_i)]} w_k^t(x_i) & k \neq z_i \\ -\sum_{j=1|j\neq k}^{M} w_j^{t+1}(x_i) & k = z_i \end{cases} \tag{13}$$

where $g^*(x)$ and α^* are the network and the coefficient learned at iteration t. Similarly finding an accurate α^* by a line search in (7) can be computationally expensive and instead we used binary search to solve $\frac{\partial \mathcal{R}[f+\alpha g^*]}{\partial \alpha} = 0$. Finally, note that for training a deep network in each boosting iteration it is possible to initialize it with random parameters or via parameters of the network learned in the previous iteration. According to our experiments the latter is more effective and we will further discuss this issue in Sec. 5.

4 Analysis

The proposed BoostCNN algorithm has a couple of interesting properties. First note that there are two objective functions in the algorithm: first, the boosting loss function that is used to compute the boosting weights, (5), and coefficients (7), and second, the squared loss function (11) which is used to train deep networks. We can optionally change these objective functions, e.g., use logistic loss as boosting objective or use weighted error rate as network training objective as long as minimizing the network objective function is consistent with minimizing directional gradient of the boosting objective function (4).

Next we provide some intuition about the boosting weights and their effects on training CNN learners. According to (5), the boosting weights in BoostCNN are M-dimensional. These weights encode two types of information. First, the norm of vector $w(x) \in \mathbb{R}^M$ is proportional to how well example x is classified by the current ensemble of weak learners. If x is correctly classified then $f_{z_i}(x_i)$ will be larger than $f_k(x_i)$ $\forall k \neq z_i$, the terms in the exponents of (5) will be small and thus $w(x)$ will have a small norm. On the other hand, if x is mis-classified, some of the exponent terms in (5) will be positive and $w(x)$ will have larger norm. Second, the k^{th} components of vector $w(x_i) \in \mathbb{R}^M$, encodes the importance of k^{th} class in classification of example x_i. For an incorrect class label $k \neq z_i$, if $f_k(x_i) > f_z(x_i)$ then $w_k(x)$ will be large. In addition $w_k(x)$ will increase exponentially by increasing $f_k(x_i) >$

$f_z(x_i)$. Therefore weights of incorrect classes will get magnified exponentially. Similarly, if $f_k(x_i) < f_z(x_i)$, $w_k(x_i)$ will be a small value. These two weighting mechanisms will modulate the network output $g(x)$ in (4) and help CNN learning procedure to focus on more difficult examples and more confusing classes.

Another interesting property of BoostCNN method is that, for any network $\bar{g}(x)$ for which (4) is non-zero, adding $\bar{g}(x)$ (or $-\bar{g}(x)$) can help the gradient descent procedure to further minimize the classification risk of the boosted classifier, (2) and improve the performance. In fact if $g'(x)$ is a network with uniform random output, i.e.

$$Prob(k = \arg\max_{k=1\ldots M} g'(x)) = \frac{1}{M}, \tag{14}$$

then according to (4) and (5)

$$E\{\delta\mathcal{R}[f;g']\} = -\frac{1}{2|\mathcal{D}|}\sum_{x_i\in\mathcal{D}}\sum_{j=1}^{M} E\{g_j(x_i)\}w_j(x_i) = -\frac{1}{2M|\mathcal{D}|}\sum_{x_i\in\mathcal{D}}\sum_{j=1}^{M} w_j(x_i) = 0.$$

Therefore, BoostCNN can use any network whose output is slightly better than chance.

Finally, BoostCNN is not limited to a single type of base-leaner and its pool of weak learners can include networks with different structures. In this case, at each boosting iterations, we train these networks independently to approximate boosting weights using (11), and the network that leads to most reduction in the boosted classification risk (2) will be added to the ensemble. This is significant as it can reduce the trial-and-error that is required for finding the right CNN structure for a specific task; we discuss this further in Sec. 5.1.

4.1 Previous works

The proposed framework is similar to boosting methods proposed by Schwenk *et al*. [34, 35, 36] such as Diabolo classifier, multi-column deep network of [1, 5] and averaging several learning networks e.g. [24, 37].

Comparing with Diabolo classifier, the proposed BoostCNN has different boosting objective as well as different network learning objective, e.g., in Diabolo the network is trained to minimize the weighted error rate while in BoostCNN the network is trained to replicate the boosting weights. Comparing with multi-column CNN, first note that multi-column CNN trains a group of CNNs simultaneously to learn a linear combination of these network as the final predictor. This, however, will increase the complexity of the learning process as it will exponentially increase number of local minima in the optimization problem, e.g., any permutation of columns of a local optimum is also a local optimum. Comparing this method with the propose method, in BoostCNN networks are trained sequentially and each network is trained on the mistakes of the previous networks. This sequential network learning simplifies the optimization problem and avoids the local minimum problem of multi-column networks. Finally, BoostCNN is better than averaging several independently trained networks because BoostCNN optimizes coefficients of the linear combination and trains new networks on more difficult examples and classes.

5 Experiments

In this section we illustrate properties of the proposed BoostCNN algorithm and compare its performance with other methods on several image classification tasks. For implementation,

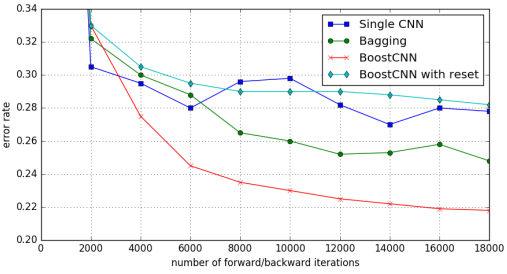

Figure 1: Comparison of error rate of BoostCNN with single CNN and Bagging.

we used the open source Caffe library [18][1].

We start by illustrating properties of BoostCNN using CIFAR-10 dataset [23]. This dataset consists of $60K$ 32×32 images in 10 classes, where $50K$ images are for training and the rest are for testing. For training, we used CIFAR10-quick network model provided in Caffe as the base-learner and replaced its its softmax loss layer with a Euclidean loss layer. This network consists of three convolutional layers, each followed by a pooling layer and a rectified linear unit (RELU). These layers are then augmented by two fully connected layers. Using BoostCNN, we trained an ensemble of 9 networks each with $2,000$ back-propagation iterations. For comparison, we trained 9 networks independently and averaged their results (bagging). We also trained a single network, using softmax loss layer, with more than $20,000$ back-propagation iterations. Figure 1 shows the error rates of these networks as a function of number of iterations. As shown in this figure, BoostCNN classifier was able to outperform both the bagged classifier and the single CNN classifier. We continued the training of the single CNN for $40K$ iterations, but error rate did not improve significantly. Manually changing the learning rate of the single CNN will decrease the error to around 25% which is still 3% higher than the error rate of BoostCNN. Figure 1 also shows the performance of BoostCNN where the base-learner is initialized randomly at the start of each boosting iteration (BoostCNN with reset). As shown in this figure, the random initialization leads to inferior performance and it is better to initialize the base-learner in each boosting iteration with the last learned base-learner.

Next we evaluate performance of the BoostCNN and compare it with state-of-the-art in tasks of bird species classification, fine grained car classification and fine grained aircraft classification tasks. In these experiments, we used the bilinear network of [21] (B-Net) as base-learner, we initialize the network with VGG16 weights and initialized the last layer (i.e., the linear layer connecting the bilinear features to the output classes) using an externally trained linear SVM to reduce the total training time. The Linear SVM classifier is very similar to learning softmax loss used in the network. However, we found that learning the last layer outside of the network to initialize the training leads to more stable classifiers and better results. Note that this is only possible when dealing with relatively small datasets, otherwise computing all of the features requires a significant amount of memory.

Bird Species Classification: For this task we used CUB-200-2011 dataset. [43]. This data consists of $11,788$ images of 200 bird species and comes with a pre-selected training and testing splits. We cropped the largest square window from the center of all images, resized them to 448×448, and mirrored the training images to double the training set. Table 1

[1]The code is available at http://github.com/mmoghimi/BoostCNN

Method	Accuracy
BoostCNN	85.6%
BoostCNN(heterogeneous)	**86.2**%
Bilinear CNN (B-Net) [41]	84.1%
Krause *et al*. [22]	82.0%
Pose Normalized CNN [2]	75.7%
Part-based RCNN[45]	73.9%

Table 1: Performance comparison for bird classification on CUB200 dataset

Method	Accuracy
BoostCNN	92.1%
Bilinear CNN (B-Net) [41]	91.3%
Krause *et al*. [22]	**92.6**%
Chai *et al*. [4]	78.0%
Fisher Vector [13]	82.7%

Method	Accuracy
BoostCNN	**88.5**%
Bilinear CNN (B-Net) [41]	84.1%
Fisher Vector [13]	80.7%
Chai *et al*. [4]	72.5%

Table 2: Performance comparison on Cars (left) and Aircraft (right) classification.

compares performance of BoostCNN with other state-of-the-art results. As shown in this table, BoostCNN was able to outperform the other methods.

Car make and model classification: For this task we used the car dataset of [21] which consists of 16,185 images of 196 different car make and models from Acura RL to Volvo XC90. This dataset comes with a pre-defined training and testing splits. The images include a variety of sizes and aspect ratios and we cropped the largest square window and resized it to 448×448 for pre-processing. Table 2-left compares performance of BoostCNN with other methods. In particular, BoostCNN was able to outperform its base-learner classifier B-Net and its error rate was only 0.5% higher than the state-of-the-art [21].

Aircraft Classification: For this task we used the FGVC-aircraft dataset [25] which consists of 10k images of 100 different aircraft models. It also contains different sub-models of the same aircraft design, e.g., different Boeing 737s. We resized the images to 448×448 while ignoring the original aspect ratio for this experiment and similar to other datasets we doubled the training set by mirroring. Table 2-right compares performance of different methods. As shown in this table BoostCNN was able to outperform other state-of-the-art-methods significantly, the next best method is the Bilinear CNN (B-Net) which is used as the base-learner in this BoostCNN classifier.

5.1 Boosting heterogeneous classifiers

In the previous section we showed that BoostCNN is able to achieve state-of-the-art performance on several image classification tasks. However, each of those tasks required trial-and-error for finding the best network structure, e.g., input size. BoostCNN can eliminate the need for this trial-and-error since it is not limited to a single type of base-leaner and its pool of weak learners can include networks with different structures. In this case, at each boosting iterations, we train these networks independently to approximate the boosting weights (11), and the network that leads to most reduction in the boosted classification risk (2) will be added to the ensemble.

We used this strategy in this experiment where the pool of base-learners consists of B-

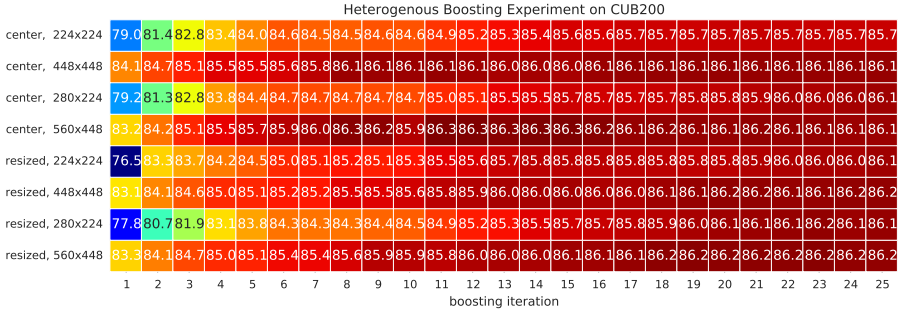

Figure 2: Testing accuracy on CUB for 25 iterations using 8 different network structures as base-learners.

Nets with 8 different input image variations, i.e. two different input sizes (448 and 224), two aspect ratios (1 and 1.25) and the choice of cropping the center patch or resizing the image to the desired size. Note that B-Nets work with any input size since the output of the bilinear layer only depends on the number of channels. Figure. 2 shows details about performance of this classifier and the candidate weak-learner in each boosting iteration. In this figure, each row corresponds to performance of one network candidate and each column corresponds to one boosting iterations. The entry at i^{th} row and j^{th} column shows testing accuracy at iteration i if j^{th} network was selected in that iteration. As shown in this figure, the network with (center, 448×448) input structure, recipe proposed by [41], was the best base-learner in the early iterations but in the later iterations, the network with (center, 560×448) is more effective. Finally, Table 1 compares performance of this approach with other methods. As shown in this figure, this approach is more effective than using a single type of network as weak-learner.

6 Conclusion and future work

In this paper, we proposed a novel model by combining the merits of boosting and deep CNNs. We are inspired by the powerful image representation learned by deep CNNs and the ability of boosting to combine the strengths of multiple learners to improve the classification. To learn this new model, we developed an algorithm to incorporate boosting weights into the deep learning architecture. We illustrated the properties of boosted convolutional networks and demonstrated the advantages of our model via state-of-the-art results on several fine-grained classification datasets: CUB [43], Cars [21] and Aircrafts [25]. Finally we showed that BoostCNN is able to boost networks with different structures. This not only resulted in better performance but also eliminated the required manual effort for finding the right network structure. In future work, we plan to apply BoostCNN to general large scale classification datasets, e.g., ImageNet, and extend our heterogeneous boosting experiments to include a more diverse set of networks with different depths and configurations.

References

[1] Forest Agostinelli, Michael R Anderson, and Honglak Lee. Adaptive multi-column deep neural networks with application to robust image denoising. In *Advances in Neural Information Processing Systems 26*, pages 1493–1501, 2013.

[2] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[4] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 321–328, 2013.

[5] Dan Ciresan, Ueli Meier, and Jurgen and Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1226-4.

[6] D.C. Ciresan, U. Meier, L.M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139, Sept 2011.

[7] Harris Drucker, Robert E. Schapire, and Patrice Simard. Improving performance in neural networks using a boosting algorithm. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 42–49, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-274-7.

[8] Harris Drucker, Corinna Cortes, L. D. Jackel, Yann LeCun, and Vladimir Vapnik. Boosting and other ensemble methods. *Neural Comput.*, 6(6):1289–1301, November 1994. ISSN 0899-7667. doi: 10.1162/neco.1994.6.6.1289.

[9] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Comp. and Sys. Science*, 1997.

[10] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999.

[11] Jerome H. Friedman and Of Known (y X)-values. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013.

[13] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014.

[14] P.M. Granitto, P.F. Verdes, and H.A. Ceccatto. Neural network ensembles: evaluation of aggregation algorithms. *Artificial Intelligence*, 163(2):139 – 162, 2005. ISSN 0004-3702. doi: 10.1016/j.artint.2004.09.006.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[17] Saharon Rosset Ji Zhu, Hui Zou and Trevor Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–3660, 2009.

[18] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[19] Nikolaos Karianakis, Thomas J. Fuchs, and Stefano Soatto. Boosting convolutional features for robust object proposals. *CoRR*, abs/1503.06350, 2015. URL http://arxiv.org/abs/1503.06350.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops*, ICCVW '13, pages 554–561, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-1-4799-3022-7. doi: 10.1109/ICCVW.2013.77.

[22] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[23] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Dept. of Computer Science, 2009.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013. URL http://arxiv.org/abs/1306.5151.

[26] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[27] Indraneel Mukherjee and Robert E. Schapire. A theory of multiclass boosting. In *NIPS*, 2010.

[28] Oliver Nina, Carlos Rubiano, and Mubarak Shah. Action recognition using ensemble of deep convolutional neural networks. *http://memkite.com/deep-learning-bibliography*, 2014.

[29] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[30] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*, 2014.

[33] M. Saberian and N. Vasconcelos. Multiclass boosting: Theory and algorithms. In *NIPS*, 2011.

[34] Holger Schwenk. The diabolo classifier. *Neural Computation*, 10(8):2175–2200, 1998. doi: 10.1162/089976698300017025.

[35] Holger Schwenk and Yoshua Bengio. Adaboosting neural networks: Application to online character recognition. In *Artificial Neural Networks - ICANN '97, 7th International Conference, Lausanne, Switzerland, October 8-10, 1997, Proceedings*, pages 967–972, 1997. doi: 10.1007/BFb0020278.

[36] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000. doi: 10.1162/089976600300015178.

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

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[40] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708. IEEE, 2014.

[41] Aruni RoyChowdhury Tsung-Yu Lin and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015.

[42] Paul Viola and Michael Jones. Robust real-time object detection. *Workshop on Statistical and Computational Theories of Vision*, 2001.

[43] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[44] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013.

[45] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *Computer Vision–ECCV 2014*, pages 834–849. Springer, 2014.

[46] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1âĂŞ2):239 – 263, 2002. ISSN 0004-3702. doi: 10.1016/S0004-3702(02)00190-X.