Towards Realistic Predictors

Pei Wang and Nuno Vasconcelos

Statistical and Visual Computing Lab, UC San Diego
{pew062,nvasconcelos}@ucsd.edu

Abstract. A new class of predictors, denoted realistic predictors, is defined. These are predictors that, like humans, assess the difficulty of examples, reject to work on those that are deemed too hard, but guarantee good performance on the ones they operate on. In this paper, we talk about a particular case of it, realistic classifiers. The central problem in realistic classification, the design of an inductive predictor of hardness scores, is considered. It is argued that this should be a predictor independent of the classifier itself, but tuned to it, and learned without explicit supervision, so as to learn from its mistakes. A new architecture is proposed to accomplish these goals by complementing the classifier with an auxiliary hardness prediction network (HP-Net). Sharing the same inputs as classifiers, the HP-Net outputs the hardness scores to be fed to the classifier as loss weights. Alternatively, the output of classifiers is also fed to HP-Net in a new defined loss, variant of cross entropy loss. The two networks are trained jointly in an adversarial way where, as the classifier learns to improve its predictions, the HP-Net refines its hardness scores. Given the learned hardness predictor, a simple implementation of realistic classifiers is proposed by rejecting examples with large scores. Experimental results not only provide evidence in support of the effectiveness of the proposed architecture and the learned hardness predictor, but also show that the realistic classifier always improves performance on the examples that it accepts to classify, performing better on these examples than an equivalent nonrealistic classifier. All of these make it possible for realistic classifiers to guarantee a good performance.

Keywords: hardness score prediction · realistic predictors

1 Introduction

Recent years have produced significant advances in computer vision, due to the introduction of deep convolutional neural networks. Like most other machine learning and computer vision models, they are trained to perform as well as possible on *every* example. In result, these models have no awareness of what they can and cannot do. This is unlike people, who have a sense of their limitations. Most humans can do certain things and do them well but, beyond these, will say 'sorry, I don't know how to do that'. Then they work on what they can do and gradually overcome their limitations. One could say that humans are *realistic predictors,* who would rather refuse tasks that are too hard than almost surely

fail. This is unlike most classifiers, who are *optimistic* and attempt to classify all examples, no matter how hard. This can be a problem for applications where incorrect decisions can have very negative consequences. For example, a significant problem for smart cars is that their vision systems offer no performance guarantees. For these applications, the vision system should guarantee that the error rate will not exceed some specifications, based on the scene, weather conditions, etc. Even more importantly, it should have a *reject* option, refusing to operate on instances that are too hard, preferring to bring the vehicle to a stop than risk accidents. Another beneficial example is that this new type of predictors could make use of computer use and human skilled labor effectively. In supervised learning, although many automatic annotation methods [30, 23] have been proposed, no performance guarantee of their results leads to the fact that humans still need to annotate all collected billions of data in practice, like by Amazon Turk. Instead of annotating all data manually, it is undoubtedly efficient to let realistic models handle on easy examples so as to guarantee accuracy comparable to humans, and just leave the hard ones aside for human experts.

A pre-requisite of realistic classifiers is the ability to *self-assess,* i.e. predict the likelihood of success or failure. This is, however, not easy in the current classification settings. One possibility is to design classifiers with a reject option. For example, classifier cascades are composed of stages that implement a series of reject decisions, efficiently zooming in on image region containing the object to detect [28]. Neural network routing [20], where samples are processed by different network paths, according to their difficulty, is a neural variant of this idea. While increasing computational efficiency, these methods frequently degrade classification performance. They produce a classifier that is faster but usually less accurate than one without rejection options. Many procedures have also been proposed to account for example hardness during training. For instance, curriculum learning [4] suggests using easy samples first and hard samples latter. On the other hand, hard example mining [24] techniques seek examples on which a classifier does poorly, to improve its performance. The goal of these methods is not to produce a *hardness predictor*, which can be applied to examples *unseen* during training, but to improve classifier performance or enable faster optimization convergence. Instead, realistic predictions require *inductive* hardness predictors, capable of operating beyond the training set.

This is, in general, a non-trivial pursuit. The main challenge is that there is no ground truth to train such a predictor. Even when human supervision is available, the ranking of samples is onerous and the identification of easy and hard samples is difficult. This is partly because what is intuitively hard for humans is not guaranteed to be hard for algorithms, and vice versa. Fig. 1 shows an example of easy and hard assessments produced by a classifier trained with different approaches. On a dataset of simple images, like MNIST, the hardness predictions are understandable to a human. One can say that easy samples are clearly written, close to prototypical digits, while hard samples are "in between" digits, e.g. "a 6 that looks like a 0," poorly executed digits, e.g. "a 6 that looks like an i" or "an open 0," etc. On the other hand, when the images are complex, as

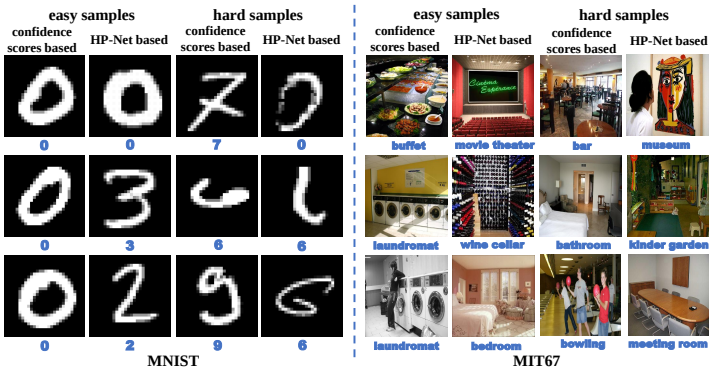


Fig. 1. Top 3 easiest and hardest examples on MNIST and MIT67 dataset according to different criteria. Ground truth labels are shown below each image. "Confidence score based" equates harder to smaller confidence scores. "HP-Net" refers to the ranking by the scores of the proposed hardness predictor.

in the MIT67 scene dataset, it is too difficult to understand why the classifiers finds the displayed examples easy or hard. In fact, the problem is not even well defined in general, since different classifiers can have different ground-truth for easy and difficult. This is certainly the case for humans, whose difficulty assessments tend to be personal and vary over time, e.g. with experience. Hence, it appears that hardness predictors should be learned in an unsupervised manner, and *personalized,* i.e., classifier specific. On the other hand, it does not appear that they can be *self-referential,* i.e. the hardness predictions cannot be produced by the classifier itself. If this were possible, the classifiers could simply implement a reject option. However, experience with hard example mining suggests that this is not very reliable. While useful for gathering difficult examples, it can produce a significant percentage of examples that are not difficult. Given all this, it appears that, for realistic prediction, the classifier should rely on an *independent* hardness predictor. However, this predictor should be trained *without* explicit supervision, *tuned* to the classifier, and *learn* from its mistakes.

Motivated by this, we propose to implement the hardness predictor as an auxiliary network, which we call the auxiliary *hardness prediction network* (HP-Net). Its input is the example to be processed by the classifier and its output a hardness score. To learn from the classifiers mistakes, the HP-Net is trained *jointly* with it. The two networks are trained in an *adversarial* setting, that resembles that of generative adversarial networks (GANs). While the proposed architecture is not a GAN, the two networks are trained *alternately.* During classifier training, the hardness scores produced by the HP-Net are used as loss weights, assigning more weight to harder examples. This encourages the classifier to classify all examples as best as possible. During HP-Net training, the classifier softmax probabilities are used to tune the HP-Net, using a variant of the cross entropy loss function that elicits adversarial behavior. In this way, as the classifier learns to *improve* its predictions, the HP-Net *refines* its hardness scores. At test

time, the HP-Net assigns a hardness score to each example. If this is above a threshold, the example is rejected. In this way, the classifier is never asked to produce class scores for examples that are deemed too hard. This is what we call *realism.* Overall, the proposed architecture has three interesting properties. First, while highly tuned to the classifier, the HP-Net is an inductive model that can be applied to unseen samples. Second, training requires no HP-Net supervision, priors, or hand-crafted rules regarding the nature of hard examples. Experiments show that the harness scores are accurate enough to enable realistic prediction, without compromise of classification accuracy. These properties are demonstrated by extensive evaluation on three datasets, which also provides interesting insights on the make-up of a hardness predictor. For example, they show that the HP-Net should be *tuned* to the classifier, with best results when the two networks have the *same* architecture. On the other hand, performance degrades substantially when the two networks have shared layers, showing that they are solving *fundamentally different* tasks. This is strong evidence against self-referential solutions. Finally, it is shown that classifier performance always increases upon restriction to easier examples. This enables the classifier to meet a specified error rate by simple control of the rejection threshold.

2 Related Work

Several criteria have been proposed to assess sample hardness. One possibility is to use task-specific criteria that leverage prior human knowledge [3, 25, 17, 27]. For example, Ionescu *et al.* [27] define image difficulty as human response time for solving a visual search task. Another popular approach is the use of loss values [13, 19, 24, 18], confidence scores [29], or the magnitude of the loss gradient [9, 12, 1, 33]. These criteria are mostly used to increase the speed of optimization procedures such as stochastic gradient descent. They are sensible for hardness prediction, since small losses tend to correspond to easy samples and vice versa. On the other hand, two samples of equal loss can be classified correctly and incorrectly. For example, adversarial examples of high confidence score are not necessarily easy to classify [26, 11]. To address this, Chang *et al.* [5] emphasize sample uncertainty when differentiating easy and hard examples. All these methods rely on handcrafted criteria for selecting and ranking examples. Similar networks with ours are proposed by [15, 31] to learn the significant samples for deep reinforcement learning and noisy labeled data, but the classifiers depending on unstable hyper-parameters are not realistic ones. The proposed approach is more closely related to the method of McGill *et al.* [20], who add a 2-way junction to neural network layers to dynamically route easy samples for direct classification and hard samples to the next layer. Nevertheless, all these methods are self-referencial, in the sense that a classifier is used to assess the hardness of the samples that it classifies. This is not easy, since hard samples are, by definition, those that the classifier makes mistakes on. We propose, instead, the use of an auxiliary predictor for this task, which learns from the classifier's mistakes.

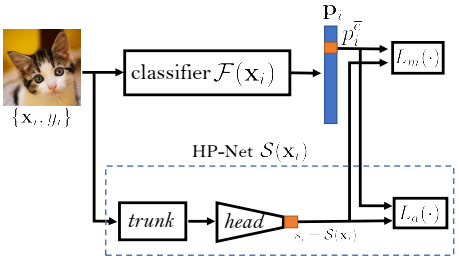


Fig. 2. Proposed architecture. $\mathcal{F}$ is the classifier, HP-Net the hardness predictor.

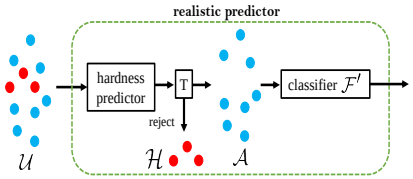


Fig. 3. Procedure of the realistic predictor.

Realistic prediction is closely related to the literature on failure prediction, where the goal is to build systems that can reliably predict the failures of a predictor. Jammalamadaka *et al.* [14] introduce evaluator algorithms to predict failures of human pose estimators, from features specific to this problem. Bansal *et al.* [2] characterize and group misclassified images using pre-selected attributes, with clustering algorithms that learn a semantic characterization of failure modes. Zhang *et al.* [32] reject probable failure samples with a binary SVM that predicts errors using 14 pre-defined kernels. Daftry *et al.* [6] define failure degree as the fraction of trajectories correctly predicted by an UAV and train a linear SVM to estimate it from the feature responses of a deep network trained for autonomous navigation. These methods rely on post-hoc analysis of the predictor performance, simply learning a regressor or classifier from its mistakes. Realistic prediction aims to go beyond this, by integrating the learning of hardness predictor and classifier, so as to guarantee optimal classifier performance on non-rejected examples. To the best of our knowledge, the proposed architecture is the first implementation of this idea. Our experiments also show that the features needed for classification are fundamentally different from those needed for difficulty prediction. This suggests that simply reading feature responses from the stages of a deep predictor [6] is sub-optimal even for failure prediction.

3 Realistic Predictor Architecture

In this section, we introduce the proposed realistic predictor architecture.

3.1 Architecture

While realistic prediction is of interest for many computer vision tasks, in this work we focus on image classification into one of C classes. The operation of a realistic predictor is illustrated in Figure 3. Consider a classifier $\mathcal{F}(\mathbf{x})$ faced with examples $\mathbf{x}_i$ from a universal example set $\mathcal{U}$. The classifier is denoted realistic if it rejects a subset of examples $\mathcal{H} \subset \mathcal{U}$ that it deems *too hard* so as to guarantee a certain performance on a subset of examples $\mathcal{A} = \mathcal{U} - \mathcal{H}$ that it agrees to

classify. Example rejection is determined by thresholding a *hardness score,* which is assigned to each example $\mathbf{x}$ by an auxiliary *hardness predictor* $\mathcal{S}(\mathbf{x})$, denoted the HP-Net. Note that, at inference time, $\mathcal{S}(\mathbf{x})$ predicts the hardness of *unseen test examples.* It must, therefore, be an inductive predictor, e.g. it does not suffice to assign weights to examples during training.

In the failure prediction literature, the classifier $\mathcal{F}$ is first learned from a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathcal{D} \subset \mathcal{U}$, y_i is the ground truth label of image $\mathbf{x}_i$, and N the number of training samples. Upon training, a failure predictor is then learned from its performance on the training set, i.e. from the set $\{\mathbf{x}_i, y_i, \hat{y}_i\}$, where $\hat{y}_i$ is the class prediction for sample $\mathbf{x}_i$. While this failure predictor could be used to implement the HP-Net of realistic prediction, this would fail to guarantee that $\mathcal{F}$ has optimal performance on the set $\mathcal{A}$ of accepted examples. One simple solution would be to use the failure predictor to reject training examples and then fine-tune $\mathcal{F}$ on those remaining. This, however, would make the failure predictor sub-optimal for the fine-tuned $\mathcal{F}$. To prevent these problems, we propose to learn $\mathcal{F}$ and $\mathcal{S}$ *jointly*, as illustrated in Figure 2.

The classifier $\mathcal{F}$ can be any convolutional neural network (CNN), usually containing a number of convolutional layers followed by fully connected layers. Its final layer implements a softmax function with C outputs, outputting a probability distribution $\mathbf{p}_i = \mathcal{F}(\mathbf{x}_i)$ in response to sample $\mathbf{x}_i$. The HP-Net has a similar structure. For notational convenience, we divide it into a set of convolution layers, the network *trunk*, and a set of fully connected layers, the network *head*. The network *trunk* is used for feature extraction while the *head* implements a multi-layer fully connected network with a single output node. This is implemented with a sigmoid unit and produces the predicted hardness score $s_i = \mathcal{S}(\mathbf{x}_i)$, $s_i \in [0, 1]$, for image $\mathbf{x}_i$. The overall operation of the realistic predictor is summarized as follows. At **training time,**

1. train classifier $\mathcal{F}$ and HP-Net $\mathcal{S}$ jointly on training set $\mathcal{D}$.
2. run $\mathcal{S}$ on $\mathcal{D}$ and eliminate hard examples, to create realistic training set $\mathcal{D}'$.
3. learn realistic classifier $\mathcal{F}'$ on $\mathcal{D}'$, with $\mathcal{S}$ fixed.
4. output pair $\mathcal{S}$, $\mathcal{F}'$.

At **test time,** run test example $\mathbf{x}$ by $\mathcal{S}$, reject hard examples, classify remaining with $\mathcal{F}'$. In all cases, $\mathbf{x}$ rejected if $\mathcal{S}(\mathbf{x}) > T$, for some threshold T.

3.2 Adversarial cross entropy loss function

The joint training of the classifier and HP-Net requires a loss function that induces the desired complimentary functions in the two networks. As is common in the literature, the classifier is trained by cross-entropy minimization. Denoting the one-hot code of ground truth label y_i by $\mathbf{y}_i$, the loss of sample $\{\mathbf{x}_i, y_i\}$ is $l(\mathbf{p}_i, \mathbf{y}_i) = -\mathbf{y}_i^T \log \mathbf{p}_i = -\sum_{c=1}^C y_i^c \log p_i^c = -\log p_i^{\bar{c}}$, where $p_i^{\bar{c}}$ is entry of $\mathbf{p}_i$ corresponding to the ground truth label location. The cross-entropy loss

$$L(\mathcal{D}) = -\sum_{i=1}^{N} \log p_i^{\bar{c}} \tag{1}$$

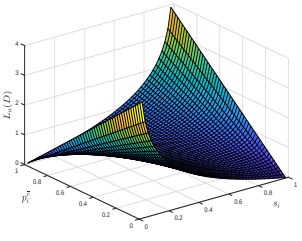


Fig. 4. HP-Net loss surface.

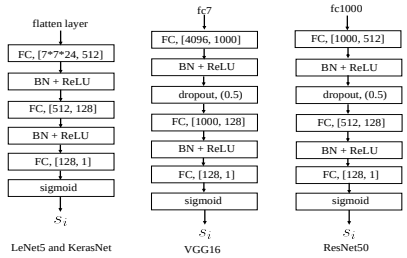


Fig. 5. Detailed structure of different *head* networks.

treats each sample equally. As is common in the cost-sensitive learning literature, we replace this with

$$L_m(\mathcal{D}) = -\sum_{i=1}^{N} s_i \log p_i^{\overline{c}}. \tag{2}$$

where $s_i \in [0, 1]$ is the hardness score of example $\mathbf{x}_i$, produced by the HP-Net. This makes harder examples (larger s_i) more important, while easier examples (lower s_i) are given less importance. In this way, the classifier is encouraged to learn from as many hard examples as possible and only reject examples that require an unreasonable amount of effort or expertise. This aims to reflect the behavior of a motivated human, who will attempt to learn as much as realistically possible about a problem and reject tasks that exceeds his or her expertise.

To encourage the HP-Net to produce scores s_i proportional to the difficulty of the corresponding samples, the HP-Net is trained with the loss function

$$L_a(\mathcal{D}) = -\sum_{i=1}^{N} \{p_i^{\overline{c}} \log(1 - s_i) + (1 - p_i^{\overline{c}}) \log s_i\}, \tag{3}$$

where $s_i, p_i^{\overline{c}} \in [0, 1]$. This is the binary cross entropy loss but with reversed semantics. It measures the cross-entropy between the distributions $(p_i^{\overline{c}}, 1 - p_i^{\overline{c}})$ and $(1 - s_i, s_i)$. Its minimization is equivalent to minimizing the Kullback-Leibler divergence between the two distributions and has a minimum when $s_i = 1 - p_i^{\overline{c}}$. This encourages large scores for poorly classified samples (low $p_i^{\overline{c}}$) and small scores for well classified samples. It can, thus, be seen as an adversarial loss that measures sample hardness, while sharing the appealing properties of the cross-entropy. These can be seen in Fig. 4, which shows a surface plot of the argument of the summation in (3). Note that this is always positive and has global minimum at the configurations $s_i = 1, p_i^{\overline{c}} = 0$ and $s_i = 0, p_i^{\overline{c}} = 1$. Hence, it encourages binary hardness scores. It is also smooth, penalizing heavily the configurations inconsistent with a hardness score ($s_i = p_i^{\overline{c}} = 0$ and $s_i = p_i^{\overline{c}} = 1$).

3.3 Training strategy

Our attempts to optimize the complete architecture of Figure 2, using the combined loss function $L(\mathcal{D}) = L_m(\mathcal{D}) + L_a(\mathcal{D})$, produced mixed results. We have

experienced difficulties to guarantee convergence of the learning procedure. It is not totally clear why at this point, we leave this for future research. Instead, we found it much easier to optimize the classifier and the HP-Net alternately. Specifically, the HP-Net is first frozen and the classifier updated. The classifier is then frozen and the HP-Net updated. The process is iterated until convergence. Note that the consistency of convergence of this process is quite intuitive. Given the classifier, the optimization of the HP-Net encourages predictions $s_i = 1 - p_i^{\overline{c}}$. Given these scores, the classifier then emphasizes the samples on which it did poorly, i.e. produced a low $p_i^{\overline{c}}$. This increases $p_i^{\overline{c}}$. In the next iteration, s_i decreases, becoming closer to $p_i^{\overline{c}}$. As $p_i^{\overline{c}}$ increases, the example acquires a smaller weight s_i and is ignored by the learning algorithm. Hence, the algorithm "puts away" the well classified examples and focuses on the poorly classified ones.

This is similar to boosting algorithms [8, 21], but has one fundamental difference. While, in boosting, the classifier reweights the examples by how well it performs on them, the proposed architecture uses an alternate predictor. Similarly, the procedure has some similarities to generative adversarial networks (GANs) [10] in the sense that there is an adversarial relationship between the classifier and hardness predictor. When the classifier produces bad predictions, the HP-Net generates an adversarial signal that encourages it to produce better predictions. Hence, the HP-Net can be seen as a signal generator that attempts to "confuse" the classifier into thinking that all samples are easy. This is similar to the GAN generator, which attempts to confuse the discriminator, rendering it unable to distinguish real from fake examples. Under this interpretation, the proposed architecture can be seen as an unsupervised generator of hardness scores. However, it is not a GAN. It is also not clear that formulating it as a GAN would add more clarity to its convergence, given the well known convergence issues surrounding GANs [10].

4 Experiments

4.1 Datasets and pre-processing

MNIST is a heavily benchmarked dataset. Although it is a relatively simple dataset, it is usefully to derive insights on network operation. We used 100 epochs, with batch size of 256, to train the network on this dataset.
MIT67 dataset [22] was proposed for indoor scene recognition. It contains 67 indoor categories and a total of 15,620 images. We follow the experimental setting of [22], where 5,360 images are used for training and 1,340 for testing. On this dataset, we fine-tune a pre-trained network, trained on ImageNet. The number of epochs is set to 50. Batch sizes 32 and 64 are used for VGG and ResNet.
ImageNet LSVRC 2012 [7] contains 1,000 classes with 1.2 million training images, 50,000 validation images, and 100,000 test images. Our evaluation is conducted on the validation set. On these two datasets, we adopt the same data augmentation and pre-processing of the previous studies [16]. Each RGB image pixel is scaled to $[0, 1]$ with mean value subtracted and standard variance divided. Then scale and aspect ratio augmentation are applied to the processed images.

The 224×224 crop is sampled from augmented images or random horizontal flips. Since, on this dataset, we used a pre-trained network, only 5 epochs are used. Again, batch sizes of 32, 64 batch are used for VGG, ResNet.

4.2 Setup

To study the impact of various network configurations, we consider several strategies for combining networks: simple classifier with complex HP-Net, complex classifier with simple HP-Net, simple classifier with simple HP-Net and complex classifier with complex HP-Net. On MNIST, LeNet5 is used as simple network and kerasNet, a network proposed by keras[1] as the complex one. On MIT67 and ImageNet, VGG16 and ResNet50 are used as simple and complex networks respectively. Additionally, we study the the setting where classifier and HP-Net have the same structure and shared convolutional layer weights.

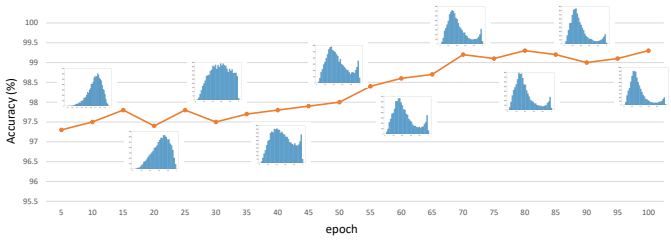
For notational convenience, we use 'A-B(-s)' to represent that A is used as classifier and B as HP-Net. If the '-s' added, A and B have shared weights. For the HP-Net, we also varied the structure of the network *head*. Three basic structures, shown in Fig. 5, are used: 'flatten layer', 'fc7' and 'fc1000' represent the flatten layer, fc7 layer and fc1000 layer of kerasNet, VGG16 and ResNet50 respectively. 'FC, [M1,M2]' represents a fully connected layer of M1 inputs and M2 outputs, 'BN' a batch normalization layer, and 'ReLU' a layer of rectified linear units.

The networks are trained using SGD with a momentum of 0.9. On MNIST, the initial learning rate is set to 0.1, 1e-3 for the classifier and HP-Net, respectively. On the other datasets, it is set to 1e-3, 1e-4 for two networks, for all of the models discussed herein, respectively. The learning rate is reduced by 5% after each epoch on MNIST, and is divided by 10 after every 10 epochs on MIT67, and 1 epoch on ImageNet.

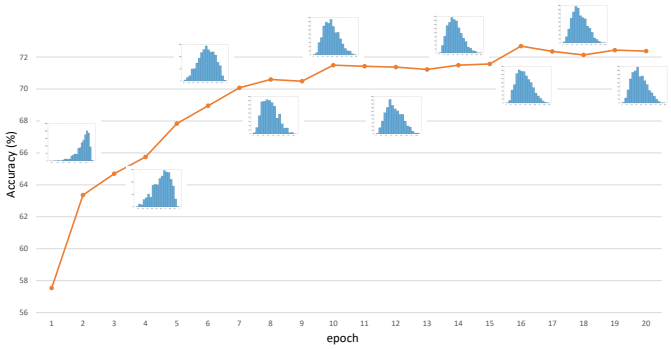
4.3 Learning to predict hardness scores

We start by presenting some results that provide some intuition on the joint learning of classifier and hardness predictor. Figure 6 shows 1) the evolution of distribution of scores produced by the HP-Net, and 2) the test set accuracy of the classifier as a function of training iteration, on MNIST and MIT67. These results were produced with the kerasNet-kerasNet network on MNIST and VGG16-VGG16 on MIT67. On MIT67, we only show the first 20 epochs because there is little change after that. Note that, as classification accuracy increases, the bulk of the mass of the hardness score distribution moves from right to left. This shows that the predicted scores decrease gradually as training progresses. As the classifier updates its predictions, the HP-Net refines its hardness scores to reflect this improvement. This, in turn, encourages the classifier to focus on the harder examples, as in hard example mining. Over training iterations, the hardness

[1] `https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py`.



a) MNIST



b) MIT67

Fig. 6. Evolution of classification accuracy and distribution of hardness scores during training.

predictor learns that samples initially considered hard are not hard after all. This enables it to make good predictions even for unseen examples. The process resembles human learning, which focuses on gradually harder examples that are eventually mastered and found easy.

4.4 Image recognition without rejection

We next consider image recognition results. LeNet5 and kerasNet are used as baseline on MNIST, VGG16 and ResNet50 on MIT67 and ImageNet. All baseline results are based on our experiments, and could differ slightly from results published by their authors. Table 1 summarizes the classification results, enabling a number of conclusions. First, the addition of the HP-Net can produce a slight performance decrease of the classifier on the entire dataset. In fact, this happened for all mixed models (different architectures for HP-Net and classifier) and when the networks are the same and share weights. Note that these numbers are for classification *on the entire dataset*. They do not imply that the classifier does not have improved performance on the examples that are accepted by the the HP-Net. This will be analyzed below. However, and somewhat surprisingly, when the HP-Net is based on the same model as the classifier, the performance of the latter on the entire dataset improves by some amount. This is likely due to

Table 1. Image recognition accuracy comparison among all model combinations

Classifier	HP-Net	shared weights	MNIST	MIT67	ImageNet top 1	ImageNet top 5
LeNet5			99.0%	—	—	—
kerasNet			99.0%	—	—	—
LeNet5	kerasNet		98.4%	—	—	—
kerasNet	LeNet5		98.2%	—	—	—
LeNet5	LeNet5		99.1%	—	—	—
kerasNet	kerasNet	s	97.9%	—	—	—
kerasNet	kerasNet		**99.2%**	—	—	—
AlexNet [34]			—	56.8%	—	—
CaffeNet [35]			—	56.8%	—	—
GoogleNet [34]			—	59.5%	—	—
VGG16	ResNet50		—	67.9%	65.6%	87.3%
ResNet50	VGG16		—	72.7%	70.4%	90.0%
VGG16			—	72.2%	71.6%	90.3%
VGG16	VGG16	s	—	67.6%	70.9%	89.9%
VGG16	VGG16		—	72.3%	73.3%	91.2%
ResNet50			—	75.6%	76.1%	92.8%
ResNet50	ResNet50	s	—	73.2%	75.9%	92.7%
ResNet50	ResNet50		—	**75.8%**	**76.4%**	**93.0%**

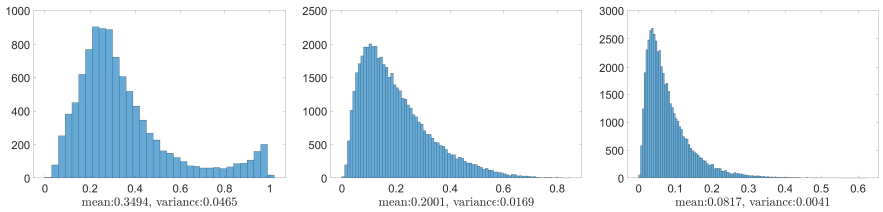
the hard example mining aspect of the procedure. The re-weighting of hard examples with large weights allows the classifier to improve on these. Although the goal of realistic prediction is not to improve image classification performance on all samples, it is interesting to see that the classifier outperforms the baselines.

Second, on all datasets, best performances occur when the classifier and the HP-Net have the same architecture. Combinations with simpler and more complex HP-Nets than the classifier have weaker performance. This is evidence that the hardness predictor has to be *tuned* to the classifier. Third, when this holds, different models can lead to variations of performance. On MNIST, there is no obvious difference between LeNet5 and the more complex kerasNet. This is probably because baseline performance is already saturated. On the other hand, on MIT67, ResNet50-ResNet50 outperforms VGG16-VGG16 by 3.5%. For the larger scale ImageNet, the increase in accuracy is 3.1% when the ResNet50 is adopted. Finally, when the convolutional layers are shared, all classifiers have slightly weaker performances on all datasets. This is interesting, given the best performance of identical models. While the two networks must be identical, sharing weights leads to a significant performance decrease. This shows that the networks are solving *fundamentally different tasks,* and argues against self-referential solutions based on a single network, such as boosting.

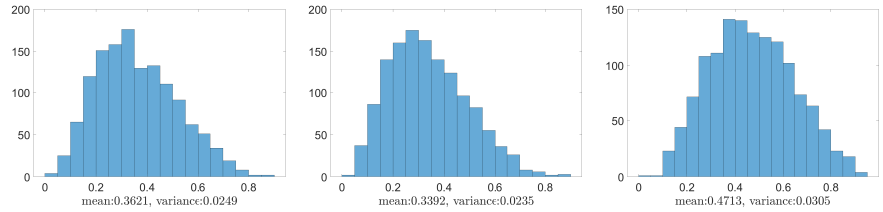
Overall, this section shows that realistic prediction does not have to sacrifice recognition performance even when no examples are rejected. This, however, requires careful selection of classifier and HP-Net architectures.

4.5 Hardness score predictions on test set

We next analyze the hardness scores produced by the various models. Fig. 7 presents the test set distribution of the scores learned by various network combinations. The mean and variance of each distribution are shown below it. The



(a) kerasNet-kerasNet (b) VGG16-VGG16 (c) Resnet50-ResNet50



(d) VGG16-VGG16 (e) ResNet50-ResNet50 (f) ResNet50-VGG16

Fig. 7. The distribution of predicted hardness scores on different settings. Sub-figure (a) is the results on MNIST; (b), (c) are on the ImageNet; The last three are on the MIT67.

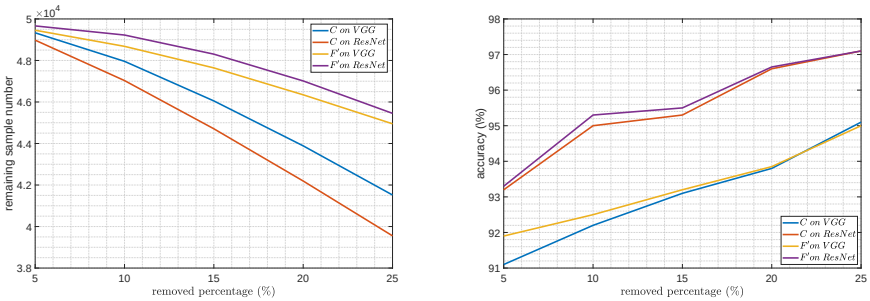
distributions produced by the different networks are consistent with the classification performances in Table 1. Plots a)-e), relative to configurations with equal models, assign small scores (less than 0.5) to most test examples. On the other hand, the ResNet50-VGG16 configuration produces a more uniform distribution, of larger mean value. Its lower classification performance has been learned by the hardness predictor, which assigns a larger score to many examples. Note also that, for the two network combinations tested on ImageNet (plots b) and c)), the ResNet50-ResNet50 configuration produces a distribution sharper than that of the VGG16-VGG16 and more concentrated on the neighborhood of 0. This shows that the ResNet50 hardness predictor is more *confident* on the outcome of the classification of the test samples. The hardness predictor, meanwhile, has learned that ResNet is a better model.

4.6 Realistic predictors

We finish with an evaluation of realistic predictions, based on three classifiers. The first, denoted $\mathcal{C}$, uses a standard (non-realistic) predictor. The second, denoted $\mathcal{F}$, is the realistic predictor produced by the training procedure, without fine-tuning to accepted examples. These two classifiers are trained on the entire training set. Finally, the third, denoted $\mathcal{F}'$, is obtained by fine tuning $\mathcal{F}$ on the training examples accepted by the hardness predictor $\mathcal{S}$. Two strategies are also compared for the rejection of examples. In both cases, a threshold T is found such that $p\%$ of the training examples are rejected. The first strategy is the self-referential strategy of rejecting examples based on the classifier confidence level.

Table 2. Performances of different methods when removing some hard samples

MIT67 (top 1 accuracy, mean(variance); VGG16-VGG16 architecture)							
classifier	rejection	0%	5%	10%	15%	20%	25%
$\mathcal{C}$	$max_c p_i^c < T$	72.2(1.5)	73.1(0.0)	75.6(0.0)	77.5(0.0)	81.0(0.0)	83.0(0.0)
$\mathcal{F}$	$max_c p_i^c < T$	72.3(1.6)	72.9(0.0)	75.4(0.0)	77.1(0.0)	80.8(0.0)	83.0(0.0)
$\mathcal{F}'$	$max_c p_i^c < T$	72.3(1.6)	73.4(0.0)	75.8(0.0)	77.2(0.0)	**81.3**(0.0)	**83.1**(0.0)
$\mathcal{F}$	$S(x) > T$	72.3(1.6)	75.0(0.0)	76.0(0.0)	77.5(0.0)	80.7(0.0)	82.6(0.0)
$\mathcal{F}'$	$S(x) > T$	**72.3**(1.6)	**75.4**(0.0)	**76.6**(0.0)	**77.9**(0.0)	81.1(0.0)	82.9(0.0)
ImageNet (top 5 accuracy, mean(variance); VGG16-VGG16 architecture)							
		0%	5%	10%	15%	20%	25%
$\mathcal{C}$	$max_c p_i^c < T$	90.3(0.0)	91.1(0.0)	92.2(0.0)	93.1(0.0)	93.8(0.0)	**95.1**(0.0)
$\mathcal{F}$	$max_c p_i^c < T$	91.2(0.5)	91.0(0.0)	92.2(0.0)	93.0(0.0)	93.8(0.0)	95.0(0.0)
$\mathcal{F}'$	$max_c p_i^c < T$	91.2(0.5)	91.2(0.0)	92.4(0.0)	93.1(0.0)	93.7(0.0)	**95.1**(0.0)
$\mathcal{F}$	$S(x) > T$	91.2(0.5)	91.8(0.0)	92.5(0.0)	93.1(0.0)	93.8(0.0)	94.8(0.0)
$\mathcal{F}'$	$S(x) > T$	**91.2**(0.5)	**91.9**(0.0)	**92.5**(0.0)	**93.2**(0.0)	**93.8**(0.0)	95.0(0.0)



a) remaining test sample number vs p b) accuracy vs p

Fig. 8. Comparison between realistic predictors and standard predictors on ImageNet.

Example $\mathbf{x}_i$ is rejected if $\max_c p_i^c < T$, where p_i^c is the softmax output for class c. The second strategy leverages the HP-Net: $\mathbf{x}_i$ is rejected if $\mathcal{S}(\mathbf{x}_i) > T$. Note that only the first strategy is possible for classifier $\mathcal{C}$, which is learned without an HP-Net.

Table 2 compares the performances of all classifiers and rejection strategies, as a function of the rejection percentage p. A few observations can be made. First, the performance of the realistic predictors is always superior to that of the standard classifier $\mathcal{C}$. As is the case for humans, by refusing to classify hard examples realistic predictors have better performance on those they classify. Second, the rejection by $S(x) > T$ outperforms standard rejection ($\max_c p_i^c < T$) almost in all settings. The gains can be quite significant, especially as p decreases, e.g. 2.3 points of top 1 performance for $p = 5\%$. This shows that it is important to learn the hardness predictor jointly with the classifier and that the commonly used self-referential confidence scores are not enough to guarantee good hardness predictions. Finally, when the hardness predictions are based on the HP-Net, there is little difference between the top 1 accuracy of the realistic predictor $\mathcal{F}$ trained on the whole training set and that ($\mathcal{F}'$) fine-tunned on accepted examples only. This shows that $\mathcal{F}$ is truly a realistic predictor, capable of close to optimal performance on the accepted examples without any finetuning.

Figure 8 illustrates more results and additionally compares the performance of the realistic predictor $\mathcal{F}'$ to the standard $\mathcal{C}$ over different network configurations (VGG16-VGG16 and ResNet50-ResNet50) on ImageNet. The realistic predictor $\mathcal{F}'$ implemented with the weaker VGG model approaches the performance of the original classifier implemented with the stronger ResNet model. This shows that, even though the VGG cannot learn everything that the ResNet can (i.e. it is not as *smart* as the ResNet), it can *guarantee* the same performance by rejecting some examples. In this case, the VGG passes the 2% rejection performance of the ResNet by rejecting around 10% test examples and the performance of the ResNet on the full test set by rejecting 5% test examples. On the other hand, in order to guarantee a target performance, the realistic predictor can accept and classify more examples than standard non-realistic predictor. For instance, to a target accuracy 93.2%, the ResNet $\mathcal{F}'$ only need to reject less than 2% samples, but for $\mathcal{C}$, it has to reject about 5% samples. In summary, while better models always have better performance, a realistic predictor can provide performance guarantees "above its pay-grade" by refusing to classify examples where it is likely to fail. This applies even to the best models. On ImageNet the superior ResNet is able to improve its performance from 93% to 97% by rejecting about 10% of the examples. While part of this is due to the fact that the examples are indeed easier, the gain is much larger than for the original predictor, which only increases its performance to 95%. The ability to predict which examples are hard, through the hardness predictor, and adapt to them enables this gain.

5 Conclusion

In this work, we have proposed a new class of classifiers, denoted realistic classifiers. These are classifiers that, like humans, assess the difficulty of examples, reject to classify those that are deemed too hard, but guarantee good performance on the ones they classify. The central problem in realistic classification, the design of an inductive predictor of hardness scores, has been then considered. It was argued that this should be a predictor independent of the classifier itself, but tuned to it, and jointly learned, so as to learn from its mistakes. A new architecture has been proposed to accomplished these goals by complementing the classifier with an auxiliary prediction network (HP-Net). The two networks are trained in an adversarial setting, that resembles that of generative adversarial networks (GANs). Experimental results have provided evidence in support of this architecture. While best results were achieved when the HP-Net has the identical architecture to the classifier, sharing weights between the two considerably degraded classification performance. This shows that, while the hardness predictor must be tuned to the classifier, the two solve fundamentally different tasks. Extensive classification experiments have also shown that the realistic classifier always improves performance on the examples that it accepts to classify, performing better on these examples than an equivalent nonrealistic classifier.

References

1. Alain, G., Lamb, A., Sankar, C., Courville, A., Bengio, Y.: Variance reduction in sgd by distributed importance sampling. International Conference on Learning Representations. (2016)
2. Bansal, A., Farhadi, A., Parikh, D.: Towards transparent systems: Semantic characterization of failure modes. In: ECCV. pp. 366–381 (2014)
3. Basu, S., Christensen, J.: Teaching classification boundaries to humans. In: AAAI Conference on Artificial Intelligence (2013)
4. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: International Conference on Machine Learning. pp. 41–48. ACM (2009)
5. Chang, H.S., Learned-Miller, E., McCallum, A.: Active bias: Training more accurate neural networks by emphasizing high variance samples. In: Advances in Neural Information Processing Systems. pp. 1003–1013 (2017)
6. Daftry, S., Zeng, S., Bagnell, J.A., Hebert, M.: Introspective perception: Learning to predict failures in vision systems. In: IEEE International Conference on Intelligent Robots and Systems. pp. 1743–1750. IEEE (2016)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255. IEEE (2009)
8. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning. vol. 96, pp. 148–156 (1996)
9. Gao, J., Jagadish, H., Ooi, B.C.: Active sampler: Light-weight accelerator for complex data analytics at scale. In: Advances in Neural Information Processing Systems. (2016)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. CoRR **abs/1412.6572** (2014)
12. Gopal, S.: Adaptive sampling for sgd by exploiting side information. In: International Conference on Machine Learning. pp. 364–372 (2016)
13. Hinton, G.E.: To recognize shapes, first learn to generate images. Progress in Brain Research **165**, 535–547 (2007)
14. Jammalamadaka, N., Zisserman, A., Eichner, M., Ferrari, V., Jawahar, C.: Has my algorithm succeeded? an evaluator for human pose estimators. In: European Conference on Computer Vision. pp. 114–128. Springer (2012)
15. Kim, T.H., Choi, J.: Screenernet: Learning curriculum for neural networks. arXiv preprint arXiv:1801.00904 (2018)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
17. Lapedriza, A., Pirsiavash, H., Bylinskii, Z., Torralba, A.: Are all training examples equally valuable? arXiv preprint arXiv:1311.6510 (2013)
18. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. IEEE International Conference on Computer Vision (2017)
19. Loshchilov, I., Hutter, F.: Online batch selection for faster training of neural networks. International Conference on Learning Representations Workshop. (2016)
20. McGill, M., Perona, P.: Deciding how to decide: Dynamic routing in artificial neural networks. In: International Conference on Machine Learning. pp. 2363–2372 (2017)

21. Moghimi, M., Belongie, S.J., Saberian, M.J., Yang, J., Vasconcelos, N., Li, L.J.: Boosted convolutional neural networks. In: British Machine Vision Conference (2016)
22. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 413–420. IEEE (2009)
23. Shin, H.C., Roberts, K., Lu, L., Demner-Fushman, D., Yao, J., Summers, R.M.: Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation. In: The IEEE Conference on Computer Vision and Pattern Recognition (June 2016)
24. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 761–769 (2016)
25. Spitkovsky, V.I., Alshawi, H., Jurafsky, D.: Baby steps: How less is more in unsupervised dependency parsing. NIPS: Grammar Induction, Representation of Language and Language Learning pp. 1–10 (2009)
26. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
27. Tudor Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D.P., Ferrari, V.: How hard can it be? estimating the difficulty of visual search in an image. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2157–2166 (2016)
28. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition. vol. 1, pp. I–I (2001)
29. Wang, X., Luo, Y., Crankshaw, D., Tumanov, A., Yu, F., Gonzalez, J.E.: Idk cascades: Fast deep learning by learning not to overthink. arXiv preprint arXiv:1706.00885 (2017)
30. Wu, B., Chen, W., Sun, P., Liu, W., Ghanem, B., Lyu, S.: Tagging like humans: Diverse and distinct image annotation. In: The IEEE Conference on Computer Vision and Pattern Recognition (2018)
31. Xiao, T., Xia, T., Yang, Y., Huang, C., Wang, X.: Learning from massive noisy labeled data for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2691–2699 (2015)
32. Zhang, P., Wang, J., Farhadi, A., Hebert, M., Parikh, D.: Predicting failures of vision systems. In: cvpr (2014)
33. Zhao, P., Zhang, T.: Stochastic optimization with importance sampling for regularized loss minimization. In: International Conference on Machine Learning. pp. 1–9 (2015)
34. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)
35. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems. pp. 487–495 (2014)