Towards Transparent Systems: Semantic Characterization of Failure Modes

Aayush Bansal[1], Ali Farhadi[2], and Devi Parikh[3]

[1] Carnegie Mellon University, Pittsburgh, USA
[2] University of Washington, Seattle, USA
[3] Virginia Tech, Blacksburg, USA

Abstract. Today's computer vision systems are not perfect. They fail frequently. Even worse, they fail abruptly and seemingly inexplicably. We argue that making our systems more transparent via an explicit human understandable characterization of their failure modes is desirable. We propose characterizing the failure modes of a vision system using semantic attributes. For example, a face recognition system may say "If the test image is blurry, or the face is not frontal, or the person to be recognized is a young white woman with heavy make up, I am likely to fail." This information can be used at training time by researchers to design better features, models or collect more focused training data. It can also be used by a downstream machine or human user at test time to know when to ignore the output of the system, in turn making it more reliable. To generate such a "specification sheet", we discriminatively cluster incorrectly classified images in the semantic attribute space using L1-regularized weighted logistic regression. We show that our specification sheets can predict oncoming failures for face and animal species recognition better than several strong baselines. We also show that lay people can easily follow our specification sheets.

1 Introduction

"*If you tell me precisely what it is a machine cannot do, then I can always make a machine which will do just that*" - John von Neumann

State-of-the-art computer vision systems are complex. In spite of their complexity, they fail frequently. And in part *due to* their complexity, they fail in *seemingly* inexplicable ways. As sophisticated image features and statistical machine learning techniques become core tools in our computer vision systems, there is an increasing desire and critical need to make our systems transparent.

Every student is different. A good teacher adapts his teaching style and the amount of time he spends on each topic to the student's strengths and weaknesses. But without knowledge of the student's misconceptions, it would be difficult for the teacher to help the student make progress. Similarly, as researchers, we can design vision solutions more effectively if we systematically understand the failure modes of our systems. Identification of recurring failure modes via manual inspection of instances where the system fails is not feasible given the scale of the data involved in realistic applications[1]. Automatic means of summarizing failure modes are required. These characterizations need to be semantic so humans (researchers, end users) can understand them. Semantic characterizations of failure modes of vision systems as seen in Fig. 1 would be useful at both training and testing time.

At training time, researchers can bring to bear their intuitions and domain knowledge to design better features and develop more effective models. Classifiers and features can be specialized for individual failure modes (e.g. for white young women with makeup and bangs). Researchers can also collect more training data geared towards a subset of categories prone to failures. For instance, if a celebrity recognition system consistently fails to recognize old Asian actresses, one could collect more data for these subset of categories to re-train the system and potentially improve it.

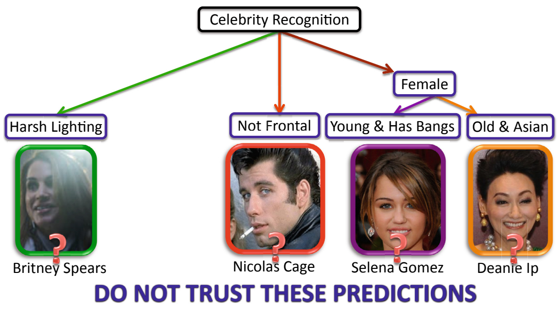

Fig. 1. We advocate transparent computer vision systems. We characterize failure modes of a vision system using semantic attributes.

At test time, our characterization of failure modes can be used to automatically detect oncoming failure. Downstream applications that use the output of computer vision systems as input can benefit from such warnings. For example, an autonomous vehicle performing semantic segmentation in a video feed can skip frames that are predicted to be unreliable, and can make slightly delayed but more accurate decisions instead. An automatic prediction of the type of failure mode can be used to raise a flag and resort to a specialized classifier for that failure mode. A semantic characterization of failure modes can also be used to empower a human user of a vision system. Consider a lay person using a vision system to recognize celebrities. It would be useful if the system came with a "specification sheet" of sorts describing the possible failure modes. The one shown in Fig. 1 can guide the user to take better pictures that are well lit and have a frontal view of the face, making the system more reliable. For some failure modes (*e.g.* regarding demographics of categories that are difficult to recognize), there may be nothing the user can do to make the system more accurate. But at least he would know to not trust the system when recognizing celebrities with a certain appearance. This results in the system being more reliable when it is used and provides precaution in scenarios where it would have likely failed anyway. The resultant fewer unpleasant surprises improves the overall user experience. A

[1] In practice, this is often how researchers debug their systems, but it can not be done very systematically and does not scale well.

semantic characterization of the failure modes of a system can thus allow us to make today's vision systems more usable even with their existing imperfections.

Finally, a semantic characterization of failure modes makes vision systems more interpretable. This helps gain operator trust in applications involving semi-autonomous systems. Numerous technologies go unused in practice simply because of insufficient operator trust [1]. Vision systems today are typically characterized by their accuracy and speed. A user (individual, startup, federal agency) decides which system to use based on a desired accuracy and speed trade off. Our spec sheets characterize the system's performance in more depth by describing the scenarios where it fails. Users can make an informed decision about which system best suits their needs. E.g. If a user expects to be using a celebrity recognition app frequently for Indian movies, he may not pick an app that has known failure modes for Indians.

Why should we expect that such a characterization exists? It is because vision systems often suffer from *systematic* failure modes. For instance, the quality of the input image – often describable by semantic attributes – affects the performance of a system drastically. Lack of enough training data of certain groups of categories (e.g. old Asian actresses, Fig. 1) may lead to the inability of the system to recognize them well. Low inter-class variance among another set of categories (many young white actresses with heavy make up and bangs may look similar) may lead to a different (characterizable) systematic failure mode. Of course, similar to other sophisticated systems, vision systems also suffer from arbitrary non-systematic mistakes. These are not the focus of this paper.

In this paper, we propose an approach that automatically identifies patterns in failures, and summarizes them with a semantic characterization that humans can understand. For instance, a face recognition system may say "If the image has harsh lighting or the face is not frontal, I may give you an incorrect answer" or "If the person you are trying to recognize is a young female with bangs, this system may give you an incorrect answer" (Fig. 1). Attribute-based representations are a natural choice to generate this semantic characterization. Given a trained classification system and a labeled set of training images, we identify images that are correctly classified ("not-mistake images"), and those that are misclassified ("mistake images"). Both sets of images are annotated with a vocabulary of binary semantic attributes. The mistake images are discriminatively clustered using weighted L1-regularized (sparse) logistic regression in the space of annotated attributes. The "discriminative" part ensures that the (mistake) clusters have only a few attributes in common with the not-mistake images, the "weighted" part encourages the mistake images within each cluster to have many attributes in common, and the "sparse" part ensures that each cluster can be characterized via just a few attributes, leading to a compact representation of the failure modes. We evaluate our approach in two domains: face (celebrity) and animal species recognition. Our experiments demonstrate that (1) Our semantic specification sheets can capture failure modes of the system well (2) They outperform strong baselines in automatic prediction of oncoming failure, and (3) non-experts can follow our specification sheets well.

2 Related Work

Our work relates to existing bodies of work on estimating classifier confidence, on predicting failures of systems, and on the use of attributes, particularly for better communication between humans and machines.

Classifier Confidence Estimation: The confidence of a classifier in its decision is often correlated to the likelihood of it being correct. Reliably estimating the confidence of classifiers has received a lot of attention in the pattern recognition community [2–4]. Applications such as spam-filtering [5], natural language processing [6, 7], speech [8] and even computer vision [9] have leveraged these ideas. However, unlike our proposed specification sheets, these confidence estimation methods are not semantically interpretable.

Predicting Failure: Methods that predict overall performance of a system on a collection of test images by analyzing statistics of the test data or post-recognition scores [10–15] are not applicable to our goal of identifying specific failure modes of the system, and semantically characterizing them. Detecting errors has received a lot of attention in speech recognition [16, 17]. In computer vision, Jammalamadaka *et al.* [18] recently introduced evaluator algorithms for human pose estimators (HPE) that can detect if the HPE has succeeded. These techniques all use non-semantic features specific to their applications for predicting failure. Most related to our work is the recent work of Hoiem *et al.* [19]. They *analyzed* the impact of different object characteristics such as size, aspect ratio, occlusion, etc. on object detection performance. Our work discovers combinations of image attributes that correlate with failure. Our generated compact semantic specification sheets can *predict* when a mistake will be made, making our vision systems more usable. The attributes we consider are generic attributes and are not explicitly tied to the workings of these underlying system.

Attributes: Attributes have been used extensively, especially in the past few years, for a variety of applications [20–34]. Attributes have been used to learn and evaluate models of deeper scene understanding [20] that reason about properties of objects as opposed to just the object categories. They have also been used for alleviating annotation efforts via zero-shot learning [23, 21, 22] where a supervisor can teach a machine a novel concept simply by describing its properties (*e.g.* "a zebra is striped and has four legs" or "a zebra has a shorter neck than a giraffe"). Attributes have also been explored to improve object categorization [23], face verification [35] and scene recognition [36]. Attributes being both machine detectable and human understandable provide a mode of communication between the two. This has been exploited for improved image search by using attributes as keywords [25] or as interactive feedback [24]. Attributes have also been leveraged for more effective active learning by allowing the supervisor to provide attributes-based feedback to a classifier [26, 34]. Knowledge of a classifier's failure modes can help the supervisor provide more focused feedback. Attributes have also been used for generating automatic textual description of images [22, 37] that can potentially point out anomalies in objects [23]. Our work exploits attributes for the novel purpose of characterizing failure modes of a machine. Attributes have been used at test time with a human-in-the-loop answering relevant questions about a test image to help the machine classify an image more reliably [31]. Our specification sheets can be used by a user at test time, but for predicting the failures of a machine rather than aiding it. A combination of these two scenarios may be interesting to explore.

3 Approach

While our approach can be applied to any vision system, we use image classification as a case study in this paper. We are given a set of N images along with their corresponding class labels $\{(\boldsymbol{x}_i, y'_i)\}, i \in \{1, \ldots, N\}, y' \in \{1, \ldots, C\}$, where C is the number of classes. We are also given a pre-trained classification system $H(\boldsymbol{x})$ whose failures we wish to characterize. Given an image $\boldsymbol{x}_i$, the system predicts a class label $\hat{y}'_i$ for the image i.e. $\hat{y}'_i = H(\boldsymbol{x}_i)$. We assign each image in our training set to a binary label $\{(\boldsymbol{x}_i, y_i)\}, y_i \in \{0, 1\}$, where $y_i = 0$ if $\hat{y}'_i = y'$ i.e. images $\boldsymbol{x}_i$ is correctly classified by H, otherwise $y_i = 1$. We annotate all images $\boldsymbol{x}_i$ using a vocabulary of M binary attributes $\{a_m\}, m \in \{1, \ldots, M\}$. Each image is thus represented with an M dimensional binary vector i.e. $\boldsymbol{x}_i \in \{-1, 1\}^M$ indicating whether attribute a_m is present in the image or not. We wish to discover a specification sheet, which we represent as a set of sparse lists of attributes – each list capturing a cluster of mistake images i.e. a failure mode.

3.1 Discriminative Clustering

We discriminatively cluster the mistake images in this ground truth attributes space. We initialize our clustering using k-means. This gives each of the mistake images a cluster index $c_i \in \{1, \ldots, K\}$. We denote all mistake images belonging to cluster k as $\{\boldsymbol{x}_i^k\}$. We train a discriminative function $h_k(\boldsymbol{x}_i)$ for each of the clusters that separates $\{\boldsymbol{x}_i^k\}$ from other "negative" images. Details of this function and the negative images follow in the next sub-section.

Let's say the score given by the discriminative function is $h_k(\boldsymbol{x}_i)$. We compute the score of all mistake images with respect to each of the K discriminative functions, and re-assign the image to the cluster whose function gives it the highest score. The updated cluster labels are

$$c_i^{(t+1)} = \underset{k}{\operatorname{argmax}}\, h_k(\boldsymbol{x}_i) \tag{1}$$

where $t+1$ denotes the next iteration. We re-train the discriminative functions using these updated cluster labels, and the process repeats. In our experiments, the process always converged, and took on average 3.6 iterations. We now describe the specifics of the discriminative function $h_k(\boldsymbol{x}_i)$.

3.2 L1-Regularized Logistic Regression

The discriminative function we train for each cluster is an L1-regularized logistic regression. It is trained to separate mistake images belonging to cluster k ($y_i^k = 1$) from all not-mistake images ($y_i^k = 0$). y_i^k is the label assigned to images for training the cluster-specific discriminative function. Notice that here y_i^k is not defined for images belonging to other mistake clusters $\boldsymbol{x}_i^l, l \in \{1, \dots, K\}, l \neq k$, as they do not participate in training the discriminative function for cluster k. All discriminative functions share the same negative set i.e. the not-mistake images $\{\boldsymbol{x}_i^0\}$. We also experimented with using all other images in the training set (including mistake images assigned to other clusters) and using only mistake images assigned to the other clusters as negative set. We select between these three strategies via cross validation (Section 4.3).

When using logistic regression, the conditional probability that the label of an image is 1 is given by

$$p(y_i^k = 1 | \boldsymbol{x}_i, \boldsymbol{w}_k) = \frac{1}{1 + \exp(-\boldsymbol{w}_k^T \boldsymbol{x}_i)} \tag{2}$$

where $\boldsymbol{w}_k$ are the parameters to be learnt. These are learnt by

$$\underset{\boldsymbol{w}_k}{\operatorname{argmax}} \sum_i \log \big(p(y_i^k = 1 | \boldsymbol{x}_i, \boldsymbol{w}_k)\big) - \alpha \sum_{m=1}^{M} |w_{k,m}| \tag{3}$$

where $w_{k,m}$ is the m^{th} entry in $\boldsymbol{w}_k$, $\sum\limits_{m=1}^{M} |w_{k,m}|$ is the L1 regularization term, α is the parameter that trades off maximizing the likelihood of the data with minimizing the regularization term leading to a sparse $\boldsymbol{w}_k$. We use interior based method for this optimization [38].

Since the feature vectors representing the image are binary vectors indicating the presence or absence of semantic attributes in the image, reading off the non-zero weights in the learnt parameters $\boldsymbol{w}_k$, allows us to describe each cluster in a semantically meaningful way. See Fig. 2.

3.3 Weighted Logistic Regression

In addition to identifying attributes that separate mistake from not-mistake images, we also wish to ensure that images belonging to the same cluster share many attributes in common and more importantly, the attributes selected to characterize the clusters are present in most of the images assigned to that cluster. This will help make the specification sheet accurate and precise. To encourage this, rather than using a standard L1-regularized logistic regression as described above, we use a weighted logistic regression. At each iteration, we replace each binary attribute in the image representation with the proportion of

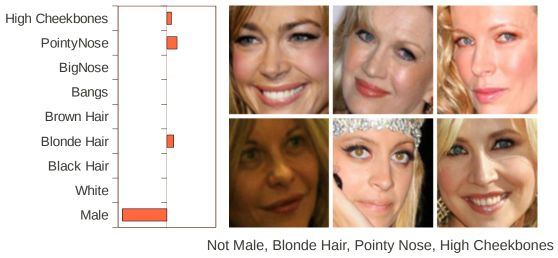

Fig. 2. The learnt sparse discriminative function for each cluster (Section 3.2) can be directly converted to a compact semantic description of the cluster. For clarity, not all attributes are shown in this illustration.

images in the cluster that share the same (binary) attribute value. That is at the $(t+1)^{th}$ iteration, the m^{th} feature value of $\boldsymbol{x}_i$ is

$$x_{i,m}^{(t+1)} = \begin{cases} \frac{1}{N^{k(t)}} \sum_{\{\boldsymbol{x}_i^k\}^{(t)}} \delta_{x_{i,m},1}, & w_{k,m} > 0 \\ \frac{-1}{N^{k(t)}} \sum_{\{\boldsymbol{x}_i^k\}^{(t)}} \delta_{x_{i,m},-1}, & w_{k,m} < 0 \\ x_{i,m}, & w_{k,m} = 0 \end{cases} \tag{4}$$

where δ_{ab}, the Kronecker delta, is 1 if $a = b$ and 0 otherwise, and $N^{k(t)}$ is the number of images assigned to the k^{th} cluster at iteration t. Recall that $\boldsymbol{x}_i \in \{-1, 1\}^M$. These are the ground truth attributes annotations of the image, and do not change with the clustering iterations. The summation counts the number of instances assigned to the k^{th} cluster at iteration t that have the m^{th} feature value agree with the sign of $\boldsymbol{w}$ for that feature. Hence, attributes that are present in most images in the cluster will have a higher weight, ensuring that it attracts even more images with that attribute to this cluster in the next cluster reassignment step. And same for the absent attributes. The weights will only impact those attributes for which $\boldsymbol{w}_k$ is non-zero.

As described above, correctly classified images form the negative set for our discriminative clustering approach. Hence, most images from reliable categories will be on the negative side, are unlikely to be captured in the characterization of failure modes. Our approach can be easily applied to individual or subsets of categories, which might also be insightful for researchers.

3.4 Hierarchical Clustering

The approach described above creates K scenarios, one for each cluster. Rather than having a list of scenarios to look through, a user may find a tree-structured specification sheet easier to navigate. To this end, we also experiment with performing the clustering described above in a hierarchical fashion. Specifically, given a branching factor B, we initialize the clustering using k-means with B clusters. We run the iterative discriminative clustering approach described above

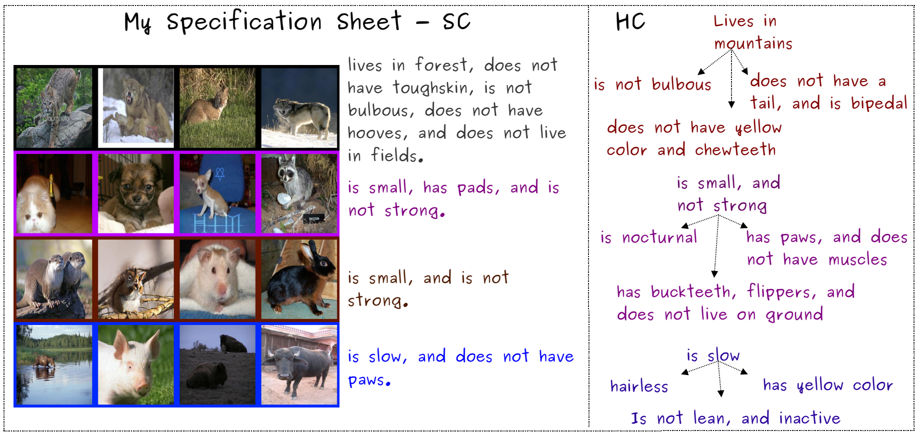
My Specification Sheet – SC
lives in forest, does not have toughskin, is not bulbous, does not have hooves, and does not live in fields.
is small, has pads, and is not strong.
is small, and is not strong.
is slow, and does not have paws.
HC
Lives in mountains
is not bulbous
does not have a tail, and is bipedal
does not have yellow color and chewteeth
is small, and not strong
is nocturnal
has paws, and does not have muscles
has buckteeth, flippers, and does not live on ground
is slow
hairless
has yellow color
Is not lean, and inactive

Fig. 3. Example specification sheets generated by our approach. Left: Simple clustering (SC): The failure modes are listed. For illustration, we show example images belonging to each cluster. Right: Hierarchical clustering (HC): Each path leading to a leaf is a failure mode *e.g.* "is slow and has yellow color" for the right most leaf of the bottom tree. Best viewed in color.

till convergence using weighted L1-regularized logistic regression. We then further cluster each of the B clusters into B clusters using the same iterative discriminative clustering, and so on, till the tree reaches a predetermined depth D. With this, we have now created a specification sheet. See Fig. 3 for an example.

4 Experiments

We now describe our experimental setup and the results we obtained.

4.1 Datasets

We experiment with two domains: face (celebrity) and animal species recognition. For faces, 2400 images from 60 categories (40 images per category) from the development set of the Public Figures Face Database (Pubfig) of Kumar *et al.* [35] are used. It contains 73 facial attributes such as race, gender, local features (e.g. pointy nose), hair color, etc. We annotated the categories with binary attribute annotations on Amazon Mechanical Turk. These will be made publicly available. For animals, 1887 images from 37 categories[2] (51 images per category) from the Animals with Attributes dataset (AwA) of Lampert *et al.* [21] containing 85 (annotated) attributes are used. 10 and 20 images per category from both

[2] We used the validation images from this dataset that were not used by the authors for training the attribute classifiers. Only 37 of the 50 categories had more than 50 such validation images.

datasets respectively were used to train their respective classifiers (SVM with RBF kernel) for recognizing the person or animal species in an image. Attribute predictors made available by the respective authors were used as image features to train these classifiers. This forms the pre-trained system provided as input to our approach, whose mistakes we wish to semantically characterize. For Pubfig / AwA, 10 / 12 images per category were used to generate our specification sheets, 10 / 8 images per category were used as a validation set and the remaining 10 / 11 images per category were used for testing. Results averaged across 10 splits are reported.

4.2 Metric

We evaluate the ability of our specification sheets to predict failure using precision and recall (PR), where we evaluate how often an image predicted by the specification sheet to be a failure truly is a failure (precision), and what percentage of the true failures are detected by the specification sheet (recall). Note that in the scenario where the user of a vision system uses our specification sheet to determine when to ignore the output of the system, another relevant dimension is the percentage of times the user would have to ignore the system. We define frequency-of-use for the user, FOU $= 1-$ proportion of test images classified to be failures. The lower the FOU, the worse the user experience. At low FOUs however, the vision system is likely to be highly accurate when it *is* used. Hence from a user perspective, the accuracy of system (ACC) vs. FOU trade-off might be more relevant than the precision-recall trade-off. The latter might be more relevant for researchers using these sheets to better understand their systems. A detailed discussion of the ACC vs. FOU metric and user-based evaluations of our specification sheets are contained in the supplementary material.

4.3 Selecting Specification Sheets

Our approach has the following parameters: (random) k-means initialization, the regularization weight α, number of clusters K for simple clustering or branching factor B and depth of tree D for hierarchical clustering and, the three choices of negative images to train the logistic regressors (Section 3.2). Different settings of these parameters can lead to specification sheets that tend to classify varying proportion of images as mistakes. We generate a pool of candidate specification sheets for 250 different k-means initializations, $\alpha \in \{5, 10, 20\}$ for hierarchical clustering and $\{10, 20, 30, 40, 50\}$ for simple clustering, $K \in [2, 20], B \in [2, 8], D \in [2, 4]$.[3] In total this leads to about 20k specification sheets generated for hierarchical clustering and 71k for simple clustering. We measured the precision and recall for each specification sheet on held out validation data. Similar to methods of computing AP from precision-recall curves, we sample S (=21)

[3] We did not use all possible combinations of these. We avoid bringing together extreme values of parameters because that leads to extremely large and cumbersome specification sheets.

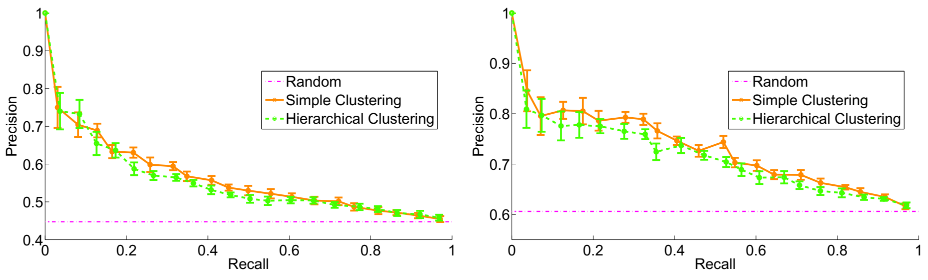

Fig. 4. Performance of our generated specification sheets capture failures. Left: Pubfig, Right: AwA.

Table 1. Area under the precision recall (PR) curve (left) and accuracy vs. frequency-of-use (ACC vs. FOU) curve (right) for different approaches. SC: simple clustering, HC: hierarchical clustering, all: using all attributes, sel: using a subset of attributes that are easy for lay people to understand.

	Random	SC - all	SC - sel	HC - all	HC - sel
Pubfig	0.4473	0.5473	0.5421	0.5370	0.5291
AwA	0.6061	0.7088	0.7079	0.6942	0.6963

	Random	SC - all	SC - sel	HC - all	HC - sel
Pubfig	0.5517	0.6181	0.6067	0.6157	0.5997
AwA	0.3929	0.4777	0.4734	0.4636	0.4606

recall points $\in [0, 1]$ in increments of 0.05. Among all specification sheets with recall closest to each sampled point, we selected the sheet with the maximum precision on a held out validation set. Given a desired operating point at test time, we use the corresponding specification sheet. Selecting specification sheets from a large pool is a proxy for the continuous threshold one can vary to select arbitrary operating points on a precision recall curve.

4.4 Automatic Failure Prediction

At the core of it our approach is separating mistakes from not-mistakes, and hence has the potential to be used as a classifier confidence measure of sorts, to automatically predict oncoming failures. To this end, we use the following approach. We run an image through each of our S specification sheets, using *predicted* attributes instead of ground truth attributes. Recall that each specification sheet is formed by multiple logistic regressors – one for each cluster – each of which produces a probability of the image being a mistake. We build a feature vector for an image by concatenating these output probabilities along with the entropy of the main classifier whose mistakes we are characterizing. We train an SVM on this new representation to classify mistake images from not-mistake images. We have S such classifiers, one for each specification sheet. We average their responses on a test image to estimate the likelihood of that image being a mistake. Varying the threshold on this likelihood will result in different PR operating points.

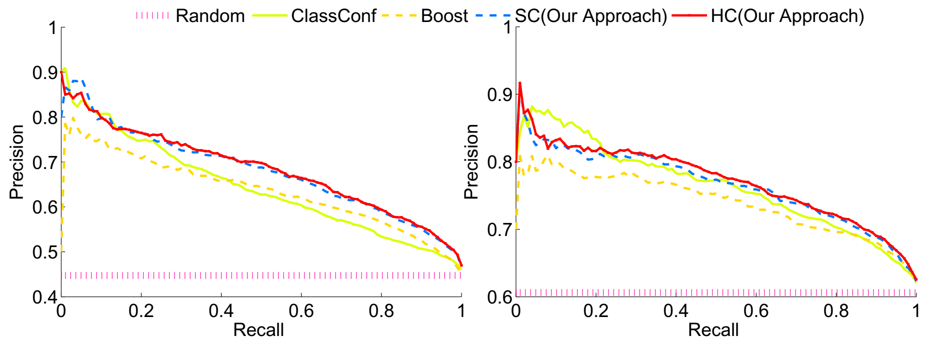

Fig. 5. Performance of our specification sheets *automatically* predicting oncoming failure. Left: Pubfig, Right: AwA.

Table 2. Area under the precision-recall (PR) curve. Comparison of various approaches to automatic failure prediction. CC: ClassConf, SC: simple (discriminative) clustering, HC: hierarchical (discriminative) clustering, GC: generative clustering.

	CC	Boost	SC	HC	CC+HC	Boost+CC	Boost+HC	HC+Boost+CC	GC	GC+CC	Rand
Pubfig	0.64	0.64	0.68	0.68	0.68	0.68	0.69	0.69	0.56	0.66	0.45
AwA	0.77	0.74	0.77	0.77	0.78	0.77	0.76	0.78	0.74	0.76	0.61

Table 3. Area under the ACC vs. FOU curve. Comparison of various approaches to automatic failure prediction. CC: ClassConf, SC: simple (discriminative) clustering, HC: hierarchical (discriminative) clustering, GC: generative clustering.

	CC	Boost	SC	HC	CC+HC	Boost+CC	Boost+HC	HC+Boost+CC	GC	GC+CC	Rand
Pubfig	0.7033	0.7130	0.7423	0.7316	0.7117	0.7390	0.7409	0.7387	0.6430	0.7293	0.5517
AwA	0.5594	0.5573	0.5752	0.5789	0.5640	0.5807	0.5821	0.5809	0.5297	0.5600	0.3929

4.5 Baselines

Our specification sheets are *fully* semantic, and thus should not be compared to non-semantic estimates of classifier confidence. We compare our automatic failure prediction approach to such non-semantic baselines. **ClassConf (CC):** The conventional approach to estimating the confidence of a classifier is computing the entropy of the probabilistic output of the classifier across the class labels (*e.g.* computed using Platts' method [39]) to a given test instance. This was one of the features used in our automatic failure prediction approach in Section 4.4. Placing a threshold on ClassConf to classify an image as being a likely mistake or not gives us a point on the PR curve. Varying this threshold gives us the entire curve. **Boost:** Our approach to automatic failure prediction employs multiple classifiers. This is related to boosting approaches [40]. We use Adaboost [41, 42] to learn the weights of 2000 decision trees[4], each with a maximum depth of 4 to differentiate between "mistake" and "not-mistake" images. We use the same image features as used by the classification system itself to train the weak learners.

[4] More trees did not further improve accuracy.

Perhaps using orthogonal features may lead to better failure prediction performance. **Rand:** We also compare to a baseline that assigns each image a random score between [0,1] as a likelihood of failure.

4.6 Results

Accuracies of the pre-trained classifiers on average were 55% and 40% for Pubfig and AwA respectively. Our goal is to semantically characterize the mistakes these classifiers tend to make. The results of oracle users[5] using our semantic specification sheets are shown in Fig. 4. Our specification sheets can predict oncoming failures with accuracy significantly better than chance.

Hierarchical vs. Simple Clustering: We compare the use of hierarchical clustering as opposed to simple clustering in Fig. 4. A hierarchical specification sheet is likely to be more convenient for a user to navigate through. But as we see for AwA (Fig. 4, right) it can perform slightly worse than simple clustering. See qualitative examples of specification sheets generated by our approach in Fig. 3. We also selected a subset of attributes that we thought were easier to understand by a lay person. We selected 45 attributes out of 73 for Pubfig and 58 out of 85 for AwA. Table 1 shows that performs stays fairly stable even with these fewer attributes.

Automatically Predicting Failures: The results of our specification sheet based *automatic* approach of predicting failures (Section 4.4) can be seen in Fig. 5 and Tables 2, 3. Our approach significantly outperforms the well accepted approach to estimating the confidence of a classifier. The boosting baseline is comparable to or worse than ClassConf. Adding our approach to ClassConf and Boost significantly improves performance. Combining all three generally leads to minor gains. Tables 2, 3 predict failure by combining predictions of multiple specification sheets (a total of 21 specification sheets; one for each sampled recall point) using an SVM. Hence, they shows improved performance over Table 1 which uses a single specification sheet.

Recall that the logistic regressors were trained on ground truth annotations of attributes. But for the automatic approach, at test time we use predicted attribute values for images. The performance may further improve if the logistic regressors were re-trained using the predicted attribute values for images at training time.

Note that Boost directly predicts failure from image features. We also learn a failure predictor, but on top of our specification sheet confidences. Our improved performance over Boost may be because attributes help transfer knowledge between categories and provide a semantic regularization of sorts. Other problems

[5] We assume that researchers can identify the presence/absence of attributes correctly, and hence will not make a mistake while following the specification sheet. Note that this does not result in a (even nearly) perfect failure prediction system. This is because the scenarios listed in the specification sheet are *learnt* summaries of the attributes incorrectly classified images tend to share in common.

(e.g. face verification [35]) have also shown that using attributes as an intermediate representation for classification outperforms direct classification from image features.

Additional Data: One might wonder: if the validation images used to train our specification sheets were instead used as additional training data to better train the underlying classification system, would its confidence measure be more accurate at failure prediction? To verify this, we retrained the base algorithm using train+val images. But performance of ClassConf did not improve (decreased little). This is not surprising. It is well known that strong classifiers can be overconfident.

Discriminative vs. Generative Clustering: We compare our discriminative clustering approach (Section 3.1) to generative clustering (GC). All mistake images are clustered using k-means clustering (which forms the initialization step for discriminative clustering) in the predicted attributes space.[6] Given a test image, its distance from the closest mistake cluster gives us an indication of its likelihood of being a mistake. Varying a threshold on this distance gives us a PR curve. We report the area under this curve in Table 2. We see that this generative approach performs significantly worse than our discriminative approach. To give it a further boost, we represent each image by its distance from all K clusters, and train a classifier on these K features and ClassConf to separate mistake images from not-mistake images. This (now partially discriminative approach: GC + ClassConf) results in better performance but still worse than our approach.

Human Studies: We conducted studies on Amazon Mechanical Turk to demonstrate that the semantic characterizations generated by our approach can be easily understood by non-computer vision experts also. Without any training about meaning of attributes, we showed subjects 24 failure modes each from celebrity face and animal species recognition by showing them the list of attributes that characterize the failure modes. The modes were selected by first randomy picking 50 failure modes (or clusters) from different specification sheets such that each was characterized by atleast 3 attributes. We then pruned out the ones that had attributes in common so as to ensure wide coverage of attributes. We had workers annotate 100 images as belonging to a failure mode or not (that is satisfying the attribute-based description or not). Each image was shown to 10 workers, and we took the majority vote. Workers were able to correctly identify whether an image belongs to a failure mode or not 85.37% and 73.96% of the time for Pubfig and AwA respectively (chance is 50%). Clearly, our specification sheets are truly human understandable. Note that our experimental evaluation covers the entire spectrum including 1. oracle users who can predict attributes reliably (Fig. 4) to evaluate the performance of our specification sheets in capturing failure modes; 2. real subjects on MTurk to see if they could easily understand these failure

[6] Performing the clustering in ground truth attributes space like our approach results in even worse performance because the test image is represented by predicted attributes and not ground truth for automatic prediction of failure. We use predicted attributes here to report a stronger baseline.

modes; and 3. without a user in the loop (Fig. 5) to demonstrate the effectiveness of our specification sheets for automatic (machine) failure prediction.

User Experience: For Pubfig, the simple clustering based specification sheets have 11 clusters on average. It involves the users having to check the values of about 7 attributes per cluster. Hierarchical clustering on the other hand has about 10 clusters but involves checking only about 4 attributes per cluster. For AwA, both simple and hierarchical clustering have 9 clusters on average, and involve checking on average about 7 and 4 attributes respectively per cluster.

5 Discussion

Like most machine learning systems, our approach can only predict what was seen during training. Existing vision systems suffer from plenty of systematic failure modes that are observed during validation. While capturing unseen failure modes is certainly desirable, capturing seen ones - even via predictive correlations (as opposed to causal relationships) - is a significant step towards making our systems transparent. The data, code, and specification sheets used in this work are available on the author's webpage.

Future Work: Discovering a vocabulary of application-specific attributes geared specifically towards predicting failures, and leveraging the sheets for the various applications discussed in the introduction is part of future work. Specification sheets can also help compare different vision systems designed to address similar tasks. This can explicitly reveal redundancies or complementary strengths among various approaches. This can be enlightening for the community, and can also be quite useful for a potential consumer of vision applications attempting to identify the system that is the best fit for the application at hand.

6 Conclusion

We proposed a discriminative clustering approach using L1-regularized weighted logistic regression to generate semantically understandable "specification sheets" that describe the failure modes of vision systems. We presented promising results for face and animal species recognition. We demonstrated that the specification sheets capture failure modes well, and can be leveraged to automatically predict oncoming failure better than a standard classifier confidence measure and a boosting baseline. By being better informed via our specification sheets, researchers can design better solutions to vision systems, and users can choose to not use the vision system in certain scenarios, increasing the performance of the system when it is used. Downstream applications can also benefit from our automatic failure prediction.

Acknowledgements. We thank Martial Hebert and anonymous reviewers for helpful insights and fruitful discussion. This work was supported in part by ARO YIP 65359NSYIP to D.P.

References

1. Stack, J.: Automation for underwater mine recognition: Current trends & future strategy. In: Proceedings of SPIE Defense & Security (2011)
2. Duin, R.P.W., Tax, D.M.J.: Classifier Conditional Posterior Probabilities. In: Amin, A., Pudil, P., Dori, D. (eds.) SPR 1998 and SSPR 1998. LNCS, vol. 1451, pp. 611–619. Springer, Heidelberg (1998)
3. Kukar, M.: Estimating confidence values of individual predictions by their typicalness and reliability. In: ECAI (2004)
4. Muhlbaier, M., Topalis, A., Polikar, R.: Ensemble confidence estimates posterior probability. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 326–335. Springer, Heidelberg (2005)
5. Delany, S.J., Cunningham, P., Doyle, D., Zamolotskikh, A.: Generating estimates of classification confidence for a case-based spam filter. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 177–190. Springer, Heidelberg (2005)
6. Dredze, M., Crammer, K.: Confidence-weighted linear classification. In: ICML (2008)
7. Bach, N., Huang, F., Al-Onaizan, Y.: Goodness: A method for measuring machine translation confidence. In: ACL (2011)
8. Jiang, H.: Confidence measures for speech recognition: A survey. Speech Communication (2005)
9. Zhang, W., Yu, S.X., Teng, S.H.: Power svm: Generalization with exemplar classification uncertainty. In: CVPR (2012)
10. Boshra, M., Bhanu, B.: Predicting performance of object recognition. PAMI (2000)
11. Wang, R., Bhanu, B.: Learning models for predicting recognition performance. In: ICCV (2005)
12. Scheirer, W.J., Rocha, A., Micheals, R.J., Boult, T.E.: Meta-recognition: The theory and practice of recognition score analysis. PAMI (2011)
13. Wang, P., Ji, Q., Wayman, J.L.: Modeling and predicting face recognition system performance based on analysis of similarity scores. PAMI (2007)
14. Scheirer, W., Kumar, N., Belhumeur, P., Boult, T.: Multi-attribute spaces: Calibration for attribute fusion and similarity search. In: CVPR (2012)
15. Scheirer, W., Rocha, A., Micheals, R., Boult, T.: Robust fusion: Extreme value theory for recognition score normalization. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 481–495. Springer, Heidelberg (2010)
16. Sarma, A., Palmer, D.D.: Context-based speech recognition error detection and correction. In: NAACL (Short papers) (2004)
17. Choularton, S.: Early stage detection of speech recognition errors (2009)
18. Jammalamadaka, N., Zisserman, A., Eichner, M., Ferrari, V., Jawahar, C.V.: Has my algorithm succeeded? An evaluator for human pose estimators. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 114–128. Springer, Heidelberg (2012)
19. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 340–353. Springer, Heidelberg (2012)
20. Farhadi, A., Endres, I., Hoiem, D.: Attribute-centric recognition for cross-category generalization. In: CVPR (2010)

21. Lampert, C., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: CVPR (2009)
22. Parikh, D., Grauman, K.: Relative attributes. In: ICCV (2011)
23. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: CVPR (2009)
24. Kovashka, A., Parikh, D., Grauman, K.: Whittlesearch: Image search with relative attribute feedback. In: CVPR (2012)
25. Kumar, N., Belhumeur, P., Nayar, S.: FaceTracer: A search engine for large collections of images with faces. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 340–353. Springer, Heidelberg (2008)
26. Parkash, A., Parikh, D.: Attributes for classifier feedback. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 354–368. Springer, Heidelberg (2012)
27. Berg, T.L., Berg, A.C., Shih, J.: Automatic attribute discovery and characterization from noisy web data. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 663–676. Springer, Heidelberg (2010)
28. Wang, J., Markert, K., Everingham, M.: Learning models for object recognition from natural language descriptions. In: BMVC (2009)
29. Wang, G., Forsyth, D.: Joint learning of visual attributes, object classes and visual saliency. In: ICCV (2009)
30. Ferrari, V., Zisserman, A.: Learning visual attributes. In: NIPS (2007)
31. Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., Belongie, S.: Visual recognition with humans in the loop. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 438–451. Springer, Heidelberg (2010)
32. Wang, G., Forsyth, D., Hoiem, D.: Comparative object similarity for improved recognition with few or no examples. In: CVPR (2010)
33. Parikh, D., Grauman, K.: Interactively building a discriminative vocabulary of nameable attributes. In: CVPR (2011)
34. Biswas, A., Parikh, D.: Simultaneous active learning of classifiers & attributes via relative feedback. In: CVPR (2013)
35. Kumar, N., Berg, A., Belhumeur, P., Nayar, S.: Attribute and simile classifiers for face verification. In: ICCV (2009)
36. Patterson, G., Hays, J.: Sun attribute database: Discovering, annotating, and recognizing scene attributes. In: CVPR (2012)
37. Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A.C., Berg, T.L.: Baby talk: Understanding and generating simple image descriptions. In: CVPR (2011)
38. Koh, K., Kim, S.J., Boyd, S.: An interior-point method for large-scale l1-regularized logistic regression. J. Mach. Learn. Res. (2007)
39. Platt, J.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Advances in Large Margin Classiers (2000)
40. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning International Workshop (1996)
41. Appel, R., Fuchs, T., Dollár, P., Perona, P.: Quickly boosting decision trees - pruning underachieving features early. In: ICML (2013)
42. Dollár, P.: Piotr's Image and Video Matlab Toolbox, `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`