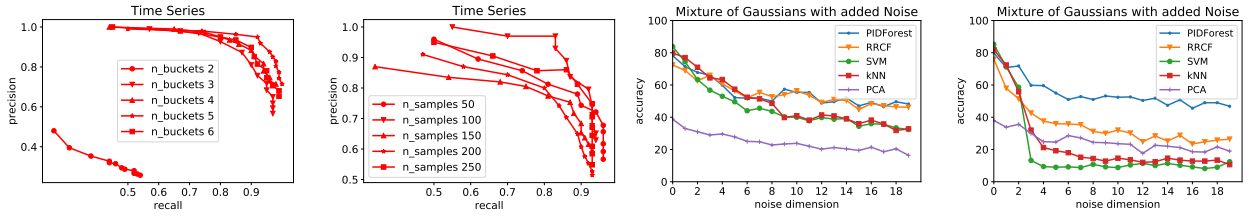

(a) Varying number of buckets (b) Varying number of samples (c) Noise uniform in [-2,2] (d) Noise uniform in [-10,10]

We thank the referees for their careful reading of the paper, their encouraging comments and thoughtful critiques. We will address all typographic and minor suggestions in the revised version. We will move the key related work and references, and more experimental results to the main body. R3 points out that we focus more on detecting anomalies rather than certifying them; we agree, and can drop the word "Certification" from the title.

(R1) Dynamic Program vs. approximate algorithm for histograms. On a 300 dimensional array, the approximate algorithm is 50x faster than the exact dynamic program in finding a best 5-bucket histogram (0.05 s vs 2.5s), with the approximation factor chosen to find a histogram having mean squared error at most 1.1 times the optimal histogram. This speedup is expected given that the complexity of the DP scales quadratically with array length, whereas the approximate algorithm is roughly linear. As histogram computation is in a deep inner loop and is invoked multiple times, we expect similar slowdowns when running the fit routine using the DP.

(R2) Why is maximizing variance a good choice? We will try to have more intuitive description of the main algorithm in the revised version. For intuition, consider the 1-dimensional case where points are generated according to a normal distribution. In this case, the outliers would be points in the tails. Indeed, if we partition the points into 3 intervals to maximize the variance in the sparsity, then we get a dense interval around the mean and two sparse intervals, one for each tail, which is exactly what we want. Moving to the general high dimensional setting, in any partition, if we pick a random point, then the expected sparsity is the same (see L202). A partition that produces large variance in the sparsity needs to partition space into some sparse regions and other dense regions, which will correspond to outliers and normal regions respectively. Alternately, one might choose partitions to optimize the maximum sparsity of any interval in the partition, or some higher moment of the sparsity. Since maximizing variance turns out to equivalent to a well-studied problem about histograms, it admits a very efficient streaming algorithm.

(R2) Provable guarantees for PIDForest. PIDForest is a heuristic, we do not have rigorous guarantees for it in the high-dimensional setting. This is addressed at the very start of Section 3 (L177), which states that we do not know polynomial algorithms in n and d that exactly compute or provably approximate PIDScore. Indeed, we suspect that exact computation might be hard as the dimension d grows (and we know an exact algorithm in 1-d).

(R2) Comparison with kNN and PCA. Isolation based algorithms work with the input basis, they do not compute linear combinations of attributes which are required to change basis. This is an advantage in dealing with heterogenous data. But for datasets that involve audio or visual inputs (like Vowels/MNIST), the input basis may not be the *right* basis. If the signal has sparsity in some special basis, then ℓ_2 distance (or some variant of it like Mahalanobis distance) might be a natural metric for such settings. In such situations kNN, PCA might do better. In general, anomaly detection problems are diverse and no single algorithm can be reasonably expected to be the best in every setting. The strength of PIDForest is that it makes minimal assumptions and hence works well in several settings that are common in practice and existing algorithms find challenging (e.g. heterogenous, noisy data).

(R3) Hyperparameter settings. One of the appealing properties of PIDForest is that the quality of the output is relatively insensitive to the exact value of the hyper-parameters. We tested multiple settings and generally found that each hyper-parameter has a moderate value above which the quality of the output doesn't change much. Figure 1a shows precision-recall in the synthetic time-series experiment, where we vary the parameter k (number of buckets). We see that $k = 2$ is too little, but there isn't much difference between $k = 3, 4, 5, 6$. Similar behavior was observed for the number of samples m (see Figure 1b where there is no clear trend), number of trees t and depth of each tree h, and again with the mixture of Gaussians experiment. Since these parameters do affect the running time directly, we set them to the smallest values for which we got good results. This is how we arrive at the guidelines in L220.

(R3) Noise Tolerance. We have included plots for the mixtures of Gaussians experiment with additional noisy dimensions for the other algorithms (kNN, SVM, RRCF, PCA). The conclusion is that PIDForest is more noise resilient than the other algorithms. In Figure 1c we choose the noise to be uniform in $[-2, 2]$ (as in Fig 1b in the paper) and see that RRCF and PIDForest are the best algorithms. In Figure 1d, we choose the noise coordinates uniformly in $[-10, 10]$ and find that PIDForest is markedly better than RRCF as well.