Appendix

A Asymmetry of $KL(D_i||C_i)$ for Positive and Negative Data

In this section, we use the gradient asymmetry for positive and negative of KL-divergence to show the need for the term marked III in Eq. 3 (also see Eq. 8 below). The term marked II in Eq. 3 will produce asymmetric gradients for positive and negative examples for both $D(\cdot)$ and $C(\cdot)$ due to asymmetry of $KL(D_i||C_i)$ for positive and negative data (explained below). If we don't have the term marked III, gradient of $D(\cdot)$ is:

$$\begin{aligned}\nabla_D V(D,C) &= \nabla_D[-\sum_{i=1}^{n} KL(P_i^{pu}||D_i^{pu}) + \lambda(\sum_{i=1}^{n_0} KL(D_i^u||C_i^u)] \\ &= \underbrace{\sum_{i=1}^{n_p} \frac{1}{D(x_i^p)} - \sum_{i=1}^{n_u} \frac{1}{1-D(x_i^u)}}_{(a)} + \underbrace{\sum_{i=1}^{n_u} \log \frac{D(x_i^u)(1-C(x_i^u))}{(1-D(x_i^u))C(x_i^u)}}_{(b)}\end{aligned} \tag{5}$$

where n_p and n_u are the size of positive and unlabeled set respectively. Term marked (a) is symmetric for positive and unlabeled data as they can obtain gradients with the same scale for the corresponding position, e.g., $D(x_i^p) + D(x_j^u) = 1$. But it is asymmetric for positive and negative data as positive data exist in the unlabeled set. That cause the positive being over optimized toward negative. Unfortunately, term marked (b) is also asymmetric for positive and negative data. We can see, the zero point of gradient term marked (b) is:

$$\begin{aligned}&\log \frac{D(x_i^u)(1-C(x_i^u))}{(1-D(x_i^u))C(x_i^u)} = 0 \\ &\Rightarrow D(x_i^u) = C(x_i^u)\end{aligned} \tag{6}$$

which means that the zero point is moved according to $C(x_i^u)$. In the worst case, if C overfitted to give small probability to instances in the unlabeled set, then $D(\cdot)$ is not easy to escape from overfitting. In summary, that will cause high precision and low recall.

Asymmetric phenomenon also exist in Eq. 8 below without the term marked III as the gradient for $C(\cdot)$ is:

$$\begin{aligned}\nabla_C V(D,C) &= \nabla_C[-\sum_{i=1}^{n} KL(P_i^{pu}||D_i^{pu}) + \lambda(\sum_{i=1}^{n_0} KL(D_i^u||C_i^u)] \\ &= \underbrace{\sum_{i=1}^{n_u} \log \frac{C(x_i^u) - D(x_i^u)}{(1-C(x_i^u))C(x_i^u)}}_{(c)}\end{aligned} \tag{7}$$

Clearly, it is asymmetric for positive and negative data, as positives have different gradient scale compared to negatives. And that can cause the unbalanced training problem. In this case, we propose to use the flipped distribution of C_i^u, denoted by $\widetilde{C}_i^u$, to address the problem, and please refer to the term marked III in Eq. 8. After adding the term marked III, the asymmetric gradient problem caused by the term marked II is eliminated. The gradient for $D(\cdot)$ now is $\sum_{i=1}^{n_u} \log \frac{(1-C(x_i^u))}{C(x_i^u)}$ which can be regarded as a constant when optimizing $D(\cdot)$. And the gradient for $C(\cdot)$ now is $\sum_{i=1}^{n_u} \log \frac{2D(x_i^u)-1}{(1-C(x_i^u))C(x_i^u)}$ which is symmetric between positive and negative.

B Simplification

Recall the loss function Eq. 3:

$$\begin{aligned}\min_C \max_D \ \mathrm{V}(D,C) = &\underbrace{-\sum_{i=1}^{n} KL(P_i^{pu}||D_i^{pu})}_{\text{I}} \\ &+\lambda \overbrace{(\underbrace{\sum_{i=1}^{n_0} KL(D_i^u||C_i^u)}_{\text{II}} - \underbrace{\sum_{i=1}^{n_0} KL(D_i^u||\widetilde{C}_i^u)}_{\text{III}})}^{①}\end{aligned} \tag{8}$$

KL-divergence is defined as:

$$KL(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log P(x) - P(x) \log Q(x) \tag{9}$$

$\mathcal{X}$ denotes the probability space, it is 1 or 0 ($\mathcal{X} = \{1, 0\}$) in our scenario. We first address term I in Eq. 8, if we use D to denote $D_i^{pu}(1)$ (the probability for ith instance being positive judged by discriminator) and P to denote $P_i^{pu}(1)$, then $D_i^{pu}(0) = 1 - D$ and $P_i^{pu}(0) = 1 - P$. Then we have:

$$\begin{aligned}&- KL(P_i^{pu}||D_i^{pu}) \\ &= -P \log P + P \log D - (1-P) \log(1-P) + (1-P) \log(1-D) \\ &= P \log D + (1-P) \log(1-D)\end{aligned} \tag{10}$$

Due to the fact that $P_i^{pu}(0) = 0$ and $P_i^{pu}(1) = 1$ if the ith instance is positive and $P_i^{pu}(1) = 0$ and $P_i^{pu}(0) = 1$ if the ith instance is unlabeled, we can rewrite the result as:

$$\mathbb{E}_{\mathbf{x}^p \sim P^p(\mathbf{x}^p)}[\log D(\mathbf{x}^p)] + \mathbb{E}_{\mathbf{x}^u \sim P^u(\mathbf{x}^u)}[\log(1 - D(\mathbf{x}^u))] \tag{11}$$

Similarly, for term ① in Eq. 8, if we use D to denote $D_i^u(1)$ and C to denote $C_i^u(1)$, then we get:

$$\begin{aligned}&KL(D_i^u||C_i^u) - KL(D_i^u||(1-C_i^u)) \\ &= D \log D - D \log C + (1-D) \log(1-D) \\ &\quad - (1-D) \log(1-C) - D \log D \\ &\quad + D \log(1-C) - (1-D) \log(1-D) + (1-D) \log C \\ &= -D \log C - (1-D) \log(1-C) + D \log(1-C) + (1-D) \log C \\ &= (\log(1-C) - \log C)(2D-1)\end{aligned} \tag{12}$$

Then we have:

$$\begin{aligned}&(\log(1-C) - \log C)(2D-1) \\ =&\mathbb{E}_{\mathbf{x}^u \sim P^u(\mathbf{x}^u)}[(\log(1 - C(\mathbf{x}^u)) - \log(C(\mathbf{x}^u)))(2D(\mathbf{x}^u) - 1)]\end{aligned} \tag{13}$$

Combining Eqs. 11 and 13, we get Eq. 14:

$$\begin{aligned}&\min_C \max_D \ \mathrm{V}(D,C) \\ &= \underbrace{\mathbb{E}_{\mathbf{x}^p \sim P^p(\mathbf{x}^p)}[\log D(\mathbf{x}^p)] + \mathbb{E}_{\mathbf{x}^u \sim P^u(\mathbf{x}^u)}[\log(1 - D(\mathbf{x}^u))]}_{\text{IV}:-H(P^L, D(\mathbf{x}^{pu}))} \\ &+ \lambda \cdot \underbrace{\mathbb{E}_{\mathbf{x}^u \sim P^u(\mathbf{x}^u)}[\underbrace{(\log(1 - C(\mathbf{x}^u)) - \log(C(\mathbf{x}^u)))}_{\text{V}}(2D(\mathbf{x}^u) - 1)]}_{\text{VI}}\end{aligned} \tag{14}$$

where P^p denotes the distribution of the positive data.

C Theoretical Analysis About the Learned Classifier

In this section, we analyze the properties of the learned classifier and show why Eq. 3 can perform PU learning. Intuitively, from Eq. 8 (same as Eq. 3 in the paper), we can see that $D(\cdot)$ is biased if we only consider term I because the unlabeled set contains both positive and negative examples. However, terms II and III help correct the bias. Eq. 14 (same as Eq. 4 in the paper), which is derived from Eq. 8 for training, shows the property more clearly than Eq. 8. Notice that the bias in term I in Eq. 8 will result in high precision and low recall for the positive class. Now back to Eq. 14 and let us imagine that most examples in the unlabeled set are regarded as negative by $C(\cdot)$ (meaning low recall). From Eq. 14, we can see that the value of term V will be below zero. But when optimizing $D(\cdot)$, term VI will push $D(\cdot)$ up for these data points, and thus the bias is reduced and the low recall problem is mitigated because in the next optimization iteration, $C(\cdot)$ will follow $D(\cdot)$ to go up for these data points.

We now theoretically discuss the optimal decision surface of the classifier $C(\cdot)$ learned by the proposed PAN.

Proposition 1. Let $T(\mathbf{x}) = \log[1 - C(\mathbf{x})] - \log[C(\mathbf{x})]$, $\varepsilon(\mathbf{x}) = f(T(\mathbf{x}))$, the learned optimal decision surface of $C(\cdot)$ is:

$$\varepsilon(\mathbf{x}) = \frac{1}{2} - \frac{P^p(\mathbf{x})}{P^p(\mathbf{x}) + P^u(\mathbf{x})} \tag{15}$$

$f(\cdot)$ is a type of function that satisfies $\varepsilon(\mathbf{x}) \cdot T(x) > 0$.

From Proposition 1, we can see that PAN finds the decision surface by combining $P^p(\mathbf{x})$ and $P^u(\mathbf{x})$. The combination is controlled by $\varepsilon(\mathbf{x})$. $\varepsilon(\mathbf{x})$ is a function of $C(\mathbf{x})$. As the precise expression of the optimal decision surface is highly complex, here only we show its properties, i.e., Eq. 19.

C.1 Proof of Proposition 1

Proof: The training criterion for discriminator D, given any classifier C, is to maximize the quantity $V(C, D)$,

$$\begin{aligned} V(C,D) = &\int_{\mathbf{x}} P^p(\mathbf{x}) \log D(\mathbf{x}) dx + \int_{\mathbf{x}} P^u(\mathbf{x}) \log(1 - D(\mathbf{x})) dx \\ &+\lambda \int_{\mathbf{x}} P^u(\mathbf{x})[\log(1 - C(\mathbf{x})) - \log C(\mathbf{x})](2D(\mathbf{x}) - 1) dx \end{aligned} \tag{16}$$

Clearly, the maximum point appears at the point with derivative 0. Then we calculate the partial derivative of $V(C, D)$ to D and get:

$$\underbrace{\frac{P^p(\mathbf{x})}{D(\mathbf{x})} - \frac{P^u(\mathbf{x})}{1 - D(\mathbf{x})}}_{ⓐ} + \underbrace{\lambda P^u(\mathbf{x})[\log(1 - C(\mathbf{x})) - \log C(\mathbf{x})]}_{ⓑ} = 0 \tag{17}$$

Directly computing the solution is complex. If we omit term ⓑ for the time being, the solution for Eq. 17 is $D(\mathbf{x}) = \frac{P^p(\mathbf{x})}{P^p(\mathbf{x}) + P^u(\mathbf{x})}$. After bringing back ⓑ, this solution should be revised as follows. With the definition of $T(\mathbf{x}) = \log[1 - C(\mathbf{x})] - \log[C(\mathbf{x})]$ and $\varepsilon(\mathbf{x}) = f(T(\mathbf{x}))$, we can re-write the solution after revision:

$$D^*(\mathbf{x}) = \frac{P^p(\mathbf{x})}{P^p(\mathbf{x}) + P^u(\mathbf{x})} + \varepsilon(\mathbf{x}) \tag{18}$$

where $\varepsilon(\mathbf{x})$ is a function of $T(\mathbf{x})$ since $P^p(\mathbf{x})$ and $P^u(\mathbf{x})$ are decided by the dataset and $\varepsilon(\mathbf{x})$ changes with the change of $T(\mathbf{x})$. The exact expression of $\varepsilon(\mathbf{x})$ is difficult but we can show $\varepsilon(\mathbf{x}) \propto T(\mathbf{x})$ in our case. In Eq. 17, term ⓐ decreases monotonously when $D(\mathbf{x}) \in (0, 1)$,[10] and both λ and $P^p(\mathbf{x})$ are greater than 0. In this case, if $T(\mathbf{x}) > 0$ ($T(\mathbf{x}) < 0$), to keep Eq. 17 equal to 0, $D(\mathbf{x})$ must move toward the positive (negative) direction, which indicates $\varepsilon(\mathbf{x}) > 0$ ($\varepsilon(\mathbf{x}) < 0$). Formally, we have:

$$\varepsilon(\mathbf{x}) \propto T(\mathbf{x}); \quad \varepsilon(\mathbf{x}) T(\mathbf{x}) > 0 \tag{19}$$

Note that the training objective of D can be interpreted as using the training data ($P^p(\mathbf{x})$ and $P^u(\mathbf{x})$) to find a discrimination bound and utilizing the learned knowledge in C to adapt it. The mini-max game in Eq. 4 can now be reformulated as:

$$\begin{aligned} L(C) = &\max_D \mathrm{V}(C, D) \\ =&\mathbb{E}_{\mathbf{x} \sim P^u(\mathbf{x})}[(\log(1 - C(\mathbf{x})) - \log(C(\mathbf{x})))(2D^*(\mathbf{x}) - 1)] \\ =&\mathbb{E}_{\mathbf{x} \sim P^u(\mathbf{x}^u)}[T(\mathbf{x})(\frac{2P^p(\mathbf{x})}{P^p(\mathbf{x}) + P^u(\mathbf{x})} + 2\varepsilon(\mathbf{x}) - 1)] \end{aligned} \tag{20}$$

Clearly, because the range of $T(\mathbf{x})$ is $(-\epsilon, \epsilon)$, $L(C)$ achieves its minimum when $T(\mathbf{x})$ and $(2D^*(\mathbf{x}) - 1)$ have opposite signs.[11] Then, the optimal $T^*(\mathbf{x})$ satisfies:

$$T^*(\mathbf{x})[\frac{2P^p(\mathbf{x})}{P^p(\mathbf{x}) + P^u(\mathbf{x})} + 2\varepsilon(\mathbf{x}) - 1] < 0 \tag{21}$$

As we introduced in Footnote 5, $C(\mathbf{x}) \in (0, 1)$. In this case, we use $C(\mathbf{x}) = 0.5$ as the decision surface to perform classification. Clearly, this decision surface equals to $T^*(\mathbf{x}) = \log(1 - 0.5) - \log(0.5) = 0$. In summary, we get the optimal decision surface:

$$\varepsilon(\mathbf{x}) = \frac{1}{2} - \frac{P^p(\mathbf{x})}{P^p(\mathbf{x}) + P^u(\mathbf{x})} \tag{22}$$

Table 4: Sensitivity of λ on MNIST.

Model	Varying λ - results are Acc				
	0.01	0.001	0.0001	0.00001	0.000001
PAN	82.70	88.90	90.30	88.23	86.44

D Additional Experiment Results

In the paper, we showed that PAN achieves significant improvement over the state-of-the-art baselines. Here we include additional results and detailed analysis of PAN in terms of robustness, varied positive ratio and the selection of hyper-parameter λ in Eq. 3 (also 4).

[10] We force $D(\mathbf{x})$ to satisfy the condition by adding a Sigmoid function to the end of D. We also force the output range of C into $(0, 1)$ using the same method.

[11] The original range of $T(\mathbf{x})$ should be $(-\infty, +\infty)$. However, such range can cause stability problems for training. We adopt a standard trick, i.e., adding a small value to the log function, e.g., $\log(C(\mathbf{x}) + 1e^{-8})$, to change the range of $T(\mathbf{x})$ to $(-8, 8)$.

Table 5: Varying the ratios of known positive data on MNIST.

Model	MNIST - results given as F / Acc				
	1%	5%	10%	20%	30%
NNPU	88.34/88.51	93.96/94.09	95.60/96.51	96.89/96.96	97.51/ 97.57
PAN	90.45/90.30	95.27 /95.36	96.51/96.42	97.38/97.43	97.90/97.95

Table 6: Varying the ratios of known positive data on CIFAR10.

Model	CIFAR10 - results given as F / Acc				
	1%	5%	10%	20%	30%
NNPU	81.41/84.22	86.09/88.84	87.84/90.14	89.05 / 91.04	90.01/91.66
PAN	82.70/86.10	87.22/89.70	88.37/90.77	89.74/91.85	90.65 / 92.49

Table 7: Accuracy (%) on different datasets for PAN with and without term III.

Model	***YELP***	***RT***	***IMDB***	***20News***	***MNIST***	***CIFAR10***
PAN without term III	80.67	60.32	78.45	72.63	96.30	89.38
PAN full model	83.56	64.10	78.84	81.00	96.42	89.70

Table 8: F-score comparison on different datasets with and without term III.

Model	***YELP***	***RT***	***IMDB***	***20News***	***MNIST***	***CIFAR10***
PAN without term III	81.86	66.28	78.52	73.59	96.27	86.68
PAN full model	83.45	66.58	77.10	81.06	96.51	87.22

D.1 Plots of Precision, Recall, and Accuracy

Figures 2-7 show the test accuracy, precision and recall in each epoch of each method. The usual first-order exponential weighted moving average smoothing with weight 0.7 is applied to the figures. To save space, we only show the curves of 100 epochs, but the final results in Table 1 in the paper are produced with more epochs as NNPU takes slightly longer to reach the peak (see below). Based on the Figure, we can make the following observation.

(1). PAN and NNPUSB are robust across all datasets with both high precision and high recall. And clearly PAN is also better than NNPUSB. NNPU is rather unbalanced for precision and recall for YELP and RT, either very high precision but very low recall, or vice versa. a-GAN and UPU have the same problem, which is highly undesirable. PAN also outperforms baselines consistently in accuracy for all six datasets.

(2). a-GAN is unstable. Precision, recall, and accuracy fluctuate greatly from one epoch to the next. Stability problem of GAN is well known. Our adaptation requiring reinforcement learning to train it is likely to have made the problem worse.

(3). NNPU converges slowly (Figure 3). It didn't converge even at 100 epoch. We report its best accuracy and F-score in Table 1 in the paper in 200 epochs (it converged earlier than 200).

D.2 Varying Known Positive Data Ratio

In this section, we show and analyse the performance of PAN with varied ratios of known positive examples. We vary the ratio of known positive examples from 5% to 30%, and show the accuracy and F-score of PAN and baseline NNPU on MNIST and CIFAR10 datasets. To stress-test the ability of PAN working in extreme conditions, we test PAN on the situation with only 1% positive examples as the known positive data. The results are reported in Tables 5 and 6.

From Tables 5 and 6, we can see that PAN can do well with different proportions of known positive data and with extremely few known positive examples. We note that the margin between PAN and NNPU goes large with the decrease of the ratio of known positive examples, which indicates that PAN is more effective than NNPU. The margin is smaller when the known positive ratio enlarges. That is because if we have enough positive data, the limitation of getting better results is no longer the PU learning method, but the performance of the classifier (?? what does this mean?)

D.3 Hyper-parameter Selection

λ is the hyper-parameter that balances the KL-divergences. Here, we show that λ should be a small value but it is not too sensitive when it is around 0.0001 (see Table 4). In our case, we set it to 0.0001.

D.4 Ablation Study for Term III in Eq. 3

Tables 7 and 8 show PAN's ablation results in accuracy and F-score with or without term III respectively. From the two tables, we can see that adding term III indeed improves the performance of PAN on 5 out of 6 datasets.

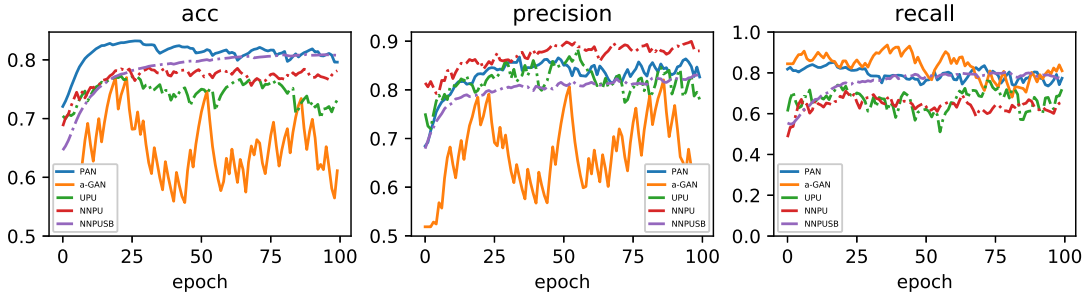

Figure 2: YELP - to save space, we only show 100 epochs.

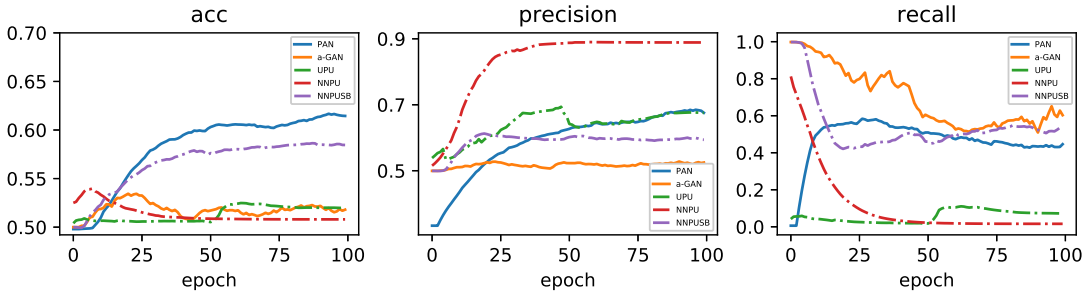

Figure 3: RT - to save space, we only show 100 epochs.

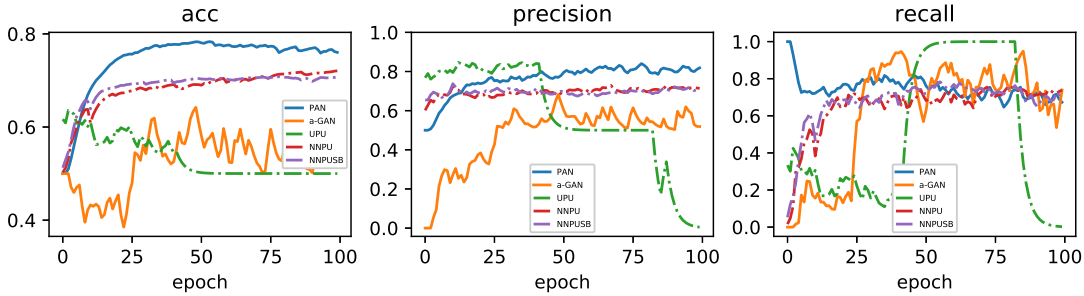

Figure 4: IMDB - to save space, we only show 100 epochs in this and the figures below.

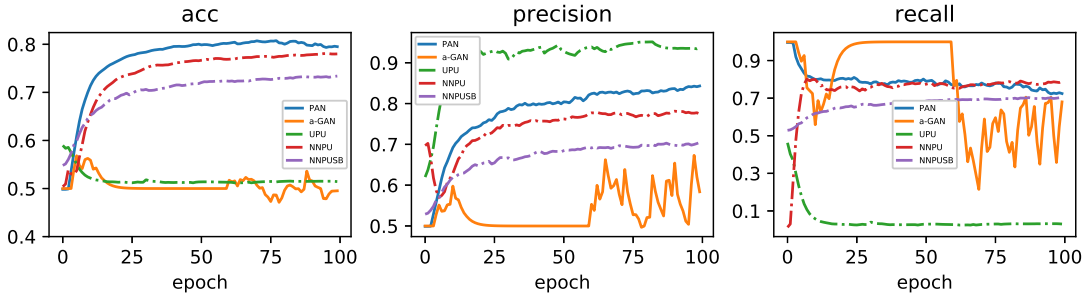

Figure 5: 20News

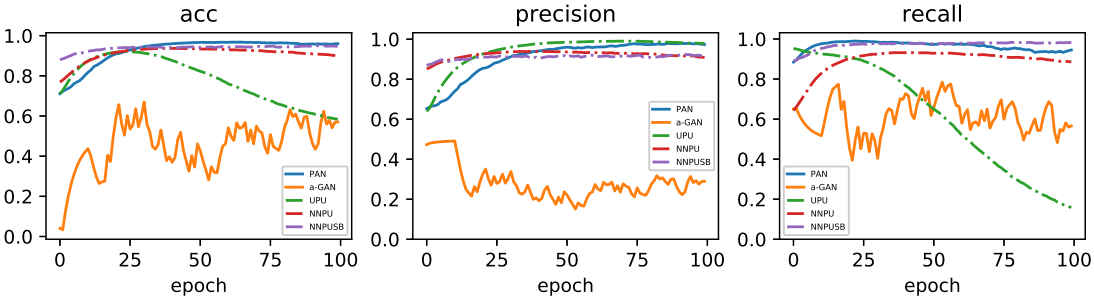

Figure 6: MNIST

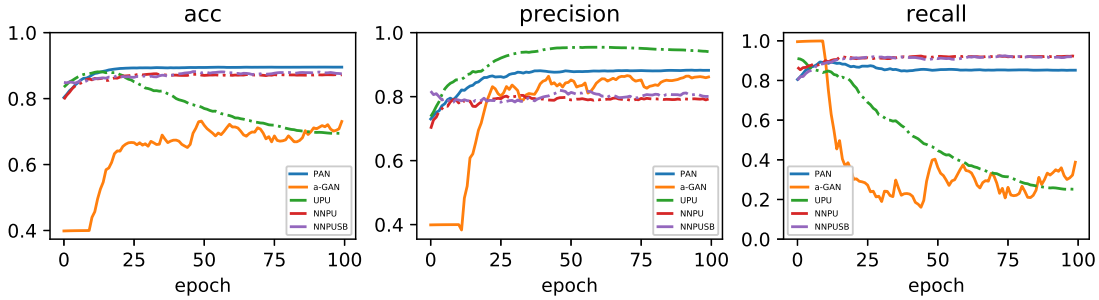

Figure 7: CIFAR10