# IS THE PERFORMANCE OF MY DEEP NETWORK TOO GOOD TO BE TRUE?

# A DIRECT APPROACH TO ESTIMATING THE BAYES ERROR IN BINARY CLASSIFICATION

**NOTABLE TOP 5%**

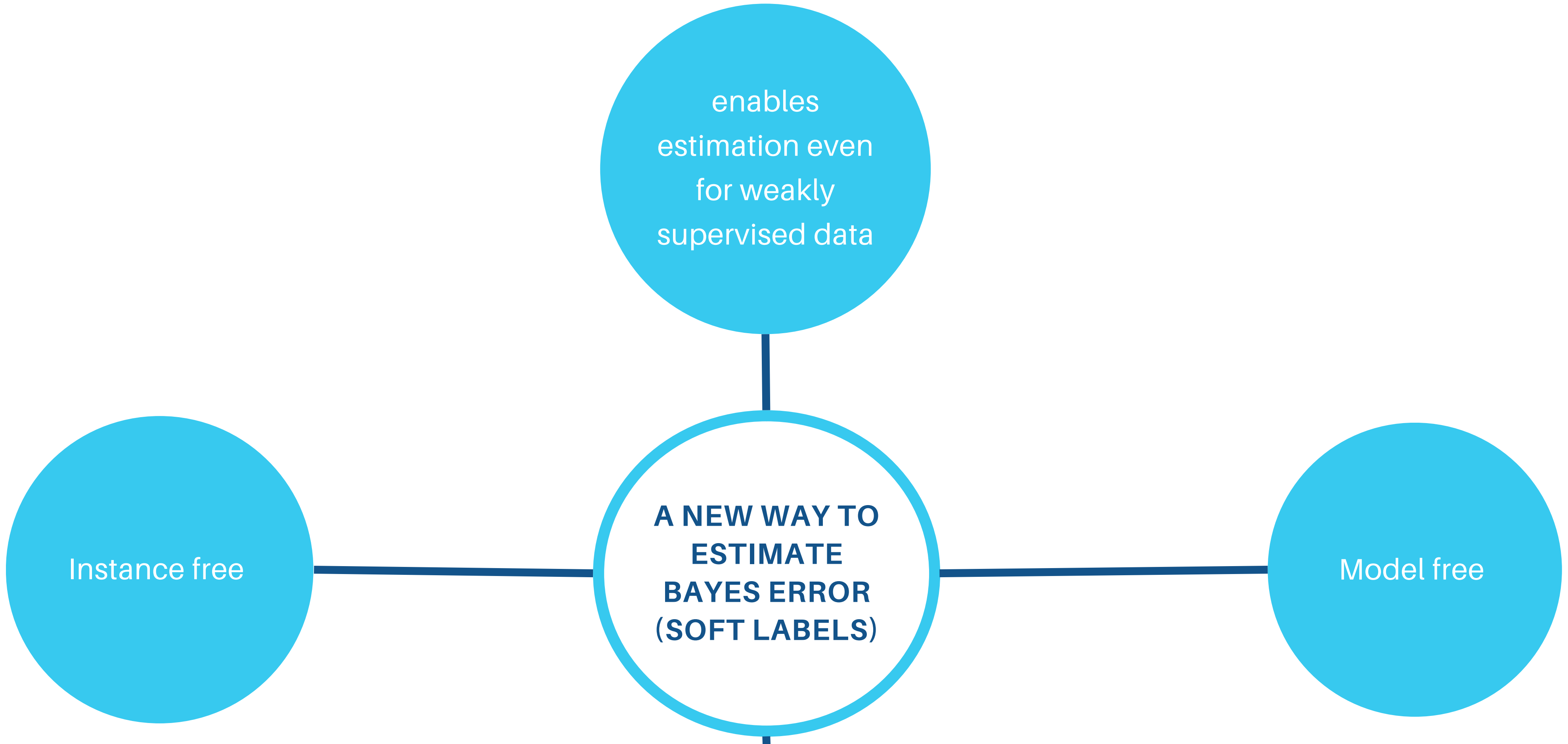
PUBLISHED: 02 FEB 2023, LAST MODIFIED: 02 MAR 2023 ICLR 2023

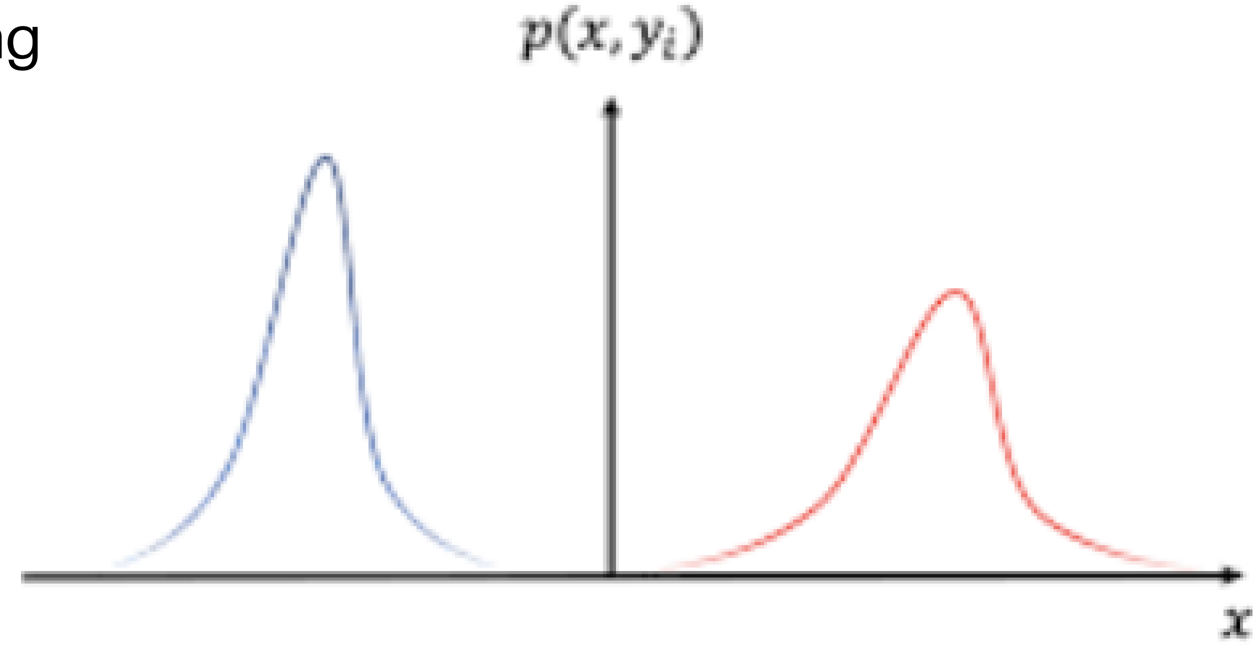
## Introduction

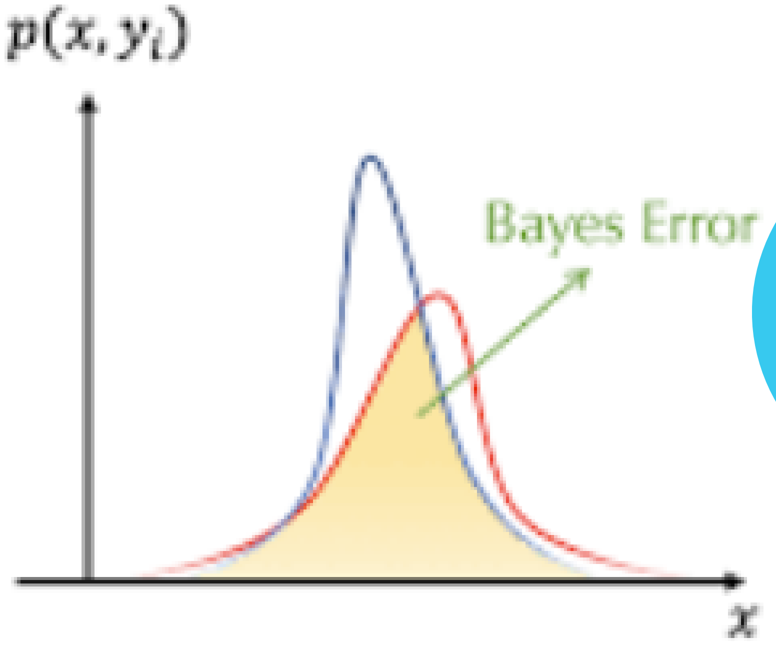
Bayes error rate is the lowest possible error rate for any classifier of a random outcome

one way to characterize the irreducible part in classification problems is the Bayes error

If the current model's performance has already achieved the Bayes error, it is meaningless to aim for further error improvement. We may even find out that the model's error has exceeded the Bayes error. This may imply test set overfitting



# SUMMARY

$$R(g) = \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(yg(\boldsymbol{x}))],$$

$$\beta = \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\min\{p(y=+1 \mid \boldsymbol{x}), p(y=-1 \mid \boldsymbol{x})\}\right]$$

$$\hat{R}(g) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i g(\boldsymbol{x}_i)),$$

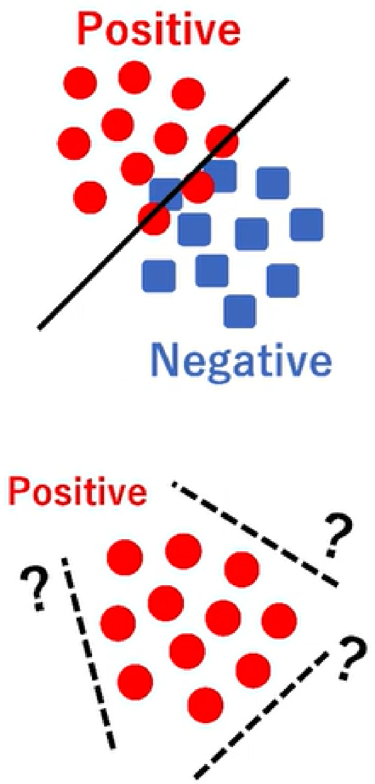
## CLEAN SOFT LABELS

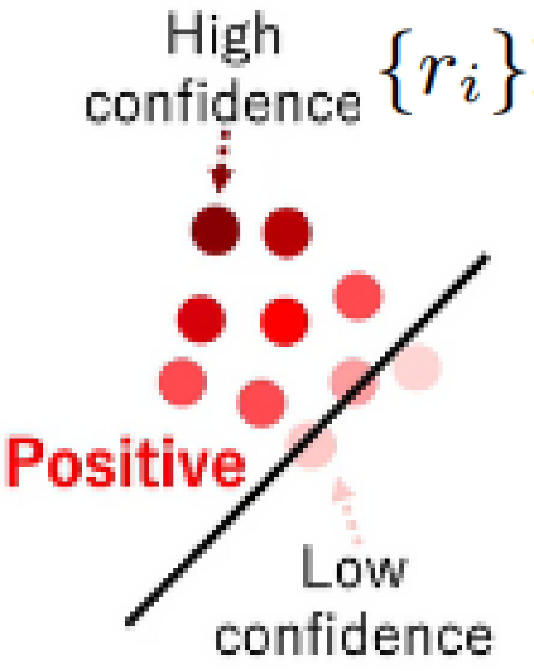
What we collect $\{c_i\}_{i=1}^{n}$

$$c_i = p(y = +1 \mid \boldsymbol{x}_i)$$

$$\hat{\beta} = \frac{1}{n}\sum_{i=1}^{n} \min\{c_i, 1 - c_i\}$$

## POSITIVE CONFIDENCE




$\{r_i\}_{i=1}^{n}$ and $\pi_+$

$$\beta_{\mathrm{Pconf}} := \pi_+\left(1 - \mathbb{E}_+\left[\max\left(0, 2 - \frac{1}{r(\boldsymbol{x})}\right)\right]\right)$$

$$\hat{\beta}_{\mathrm{Pconf}} = \pi_+\left(1 - \frac{1}{n_+}\sum_{i=1}^{n_+} \max\left(0, 2 - \frac{1}{r_i}\right)\right)$$

## NOISY SOFT LABELS

what we collect $\{(u_i, s_i)\}_{i=1}^{n}$

$$s_i = \mathrm{sign}\left[p(y=+1|\boldsymbol{x}_i) - 0.5\right]$$

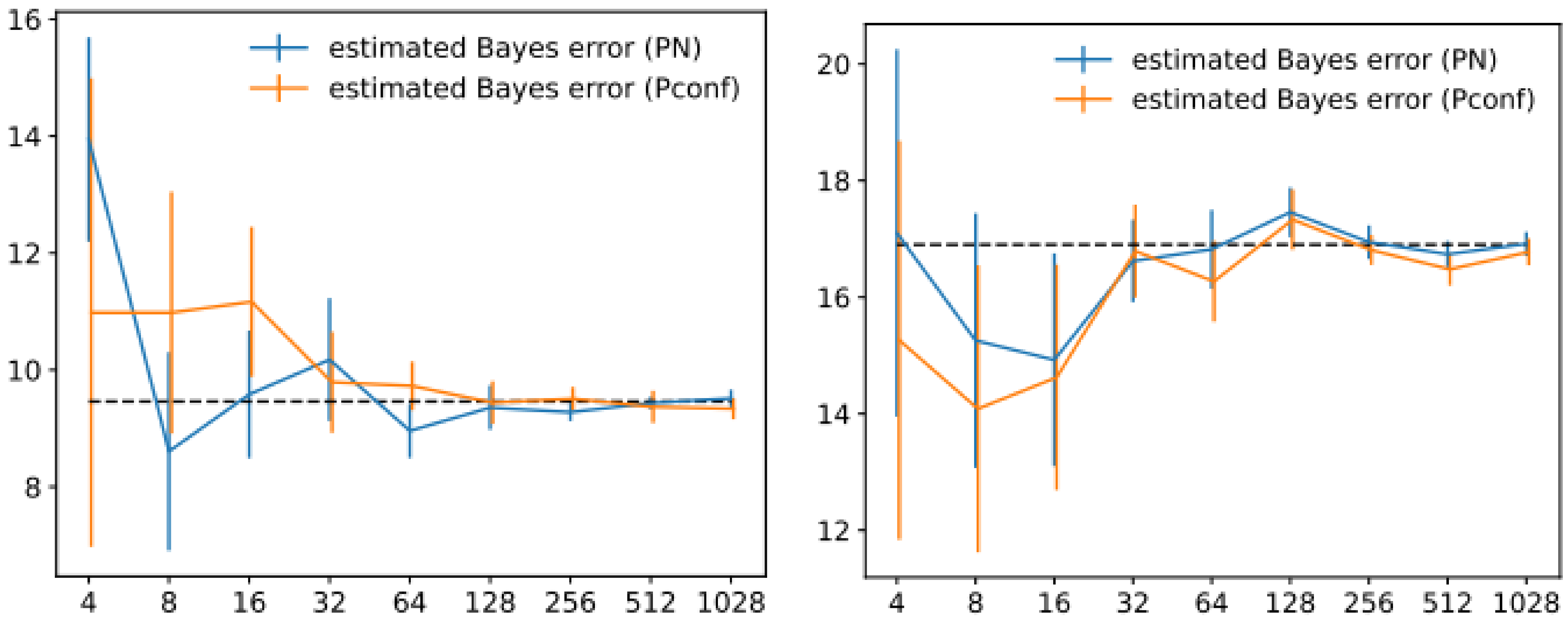
$\{u_i\}_{i=1}^{n}$ into $\{u_i^+\}_{i=1}^{n_s^+}$ and $\{u_i^-\}_{i=1}^{n_s^-}$

$$\tilde{\beta} = \frac{1}{n}\sum_{i=1}^{n}\left[\min\left(u_i, 1 - u_i\right)\right]$$

$$\hat{\beta}_{\mathrm{Noise}} = \frac{1}{n}\left(\sum_{i=1}^{n_s^+}\left(1 - u_i^+\right) + \sum_{i=1}^{n_s^-} u_i^-\right)$$

# RESULTS

Steup A and Setup B have different sets of Gaussian distributions




(a) Setup A (b) Setup B

```
C:\Users\eelab.CCLAB-IITJ\Downloads\irreducible-main>python be_synthetic.py
usage: Experiments for Fig 1, 2, and 3.
Synthetic experiments: error: the following arguments are required: -su/--setup

C:\Users\eelab.CCLAB-IITJ\Downloads\irreducible-main>python be_synthetic.py --setup=0
setup: 0 | noise: False | data size: 2 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.139 (0.0175) Pconf in Eq.8 mean (ste): 0.132 (0.0545)
setup: 0 | noise: False | data size: 4 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0861 (0.017) Pconf in Eq.8 mean (ste): 0.11 (0.0401)
setup: 0 | noise: False | data size: 8 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0958 (0.0109) Pconf in Eq.8 mean (ste): 0.11 (0.0207)
setup: 0 | noise: False | data size: 16 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.102 (0.0105) Pconf in Eq.8 mean (ste): 0.112 (0.0128)
setup: 0 | noise: False | data size: 32 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0896 (0.00463) Pconf in Eq.8 mean (ste): 0.0979 (0.00863)
setup: 0 | noise: False | data size: 64 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0935 (0.00378) Pconf in Eq.8 mean (ste): 0.0973 (0.00416)
setup: 0 | noise: False | data size: 128 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0928 (0.00154) Pconf in Eq.8 mean (ste): 0.0944 (0.00367)
setup: 0 | noise: False | data size: 256 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0943 (0.00109) Pconf in Eq.8 mean (ste): 0.0951 (0.00203)
setup: 0 | noise: False | data size: 512 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0951 (0.00151) Pconf in Eq.8 mean (ste): 0.0937 (0.00276)
setup: 0 | noise: False | data size: 1028 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.0952 (0.000911) Pconf in Eq.8 mean (ste): 0.0934 (0.0018)

C:\Users\eelab.CCLAB-IITJ\Downloads\irreducible-main>
```

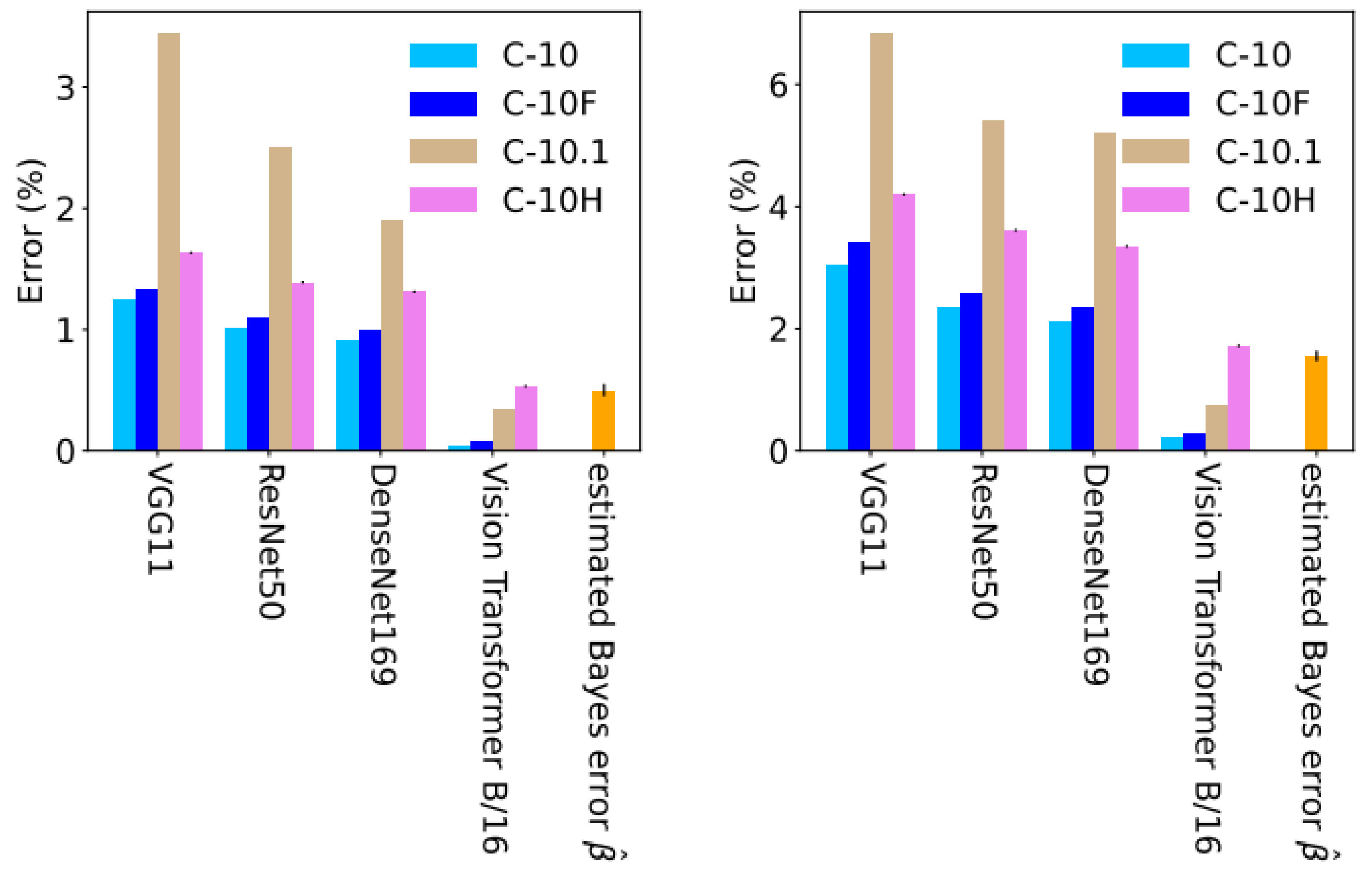
```
C:\Users\eelab.CCLAB-IITJ\Downloads\irreducible-main>python be_synthetic.py --setu
setup: 1 | noise: False | data size: 2 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.171 (0.0315) Pconf in Eq.8 mean (ste): 0.135 (0.04)
setup: 1 | noise: False | data size: 4 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.153 (0.0219) Pconf in Eq.8 mean (ste): 0.153 (0.0342)
setup: 1 | noise: False | data size: 8 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.149 (0.0183) Pconf in Eq.8 mean (ste): 0.141 (0.0246)
setup: 1 | noise: False | data size: 16 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.166 (0.00707) Pconf in Eq.8 mean (ste): 0.146 (0.0193)
setup: 1 | noise: False | data size: 32 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.168 (0.00675) Pconf in Eq.8 mean (ste): 0.168 (0.00796)
setup: 1 | noise: False | data size: 64 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.175 (0.00426) Pconf in Eq.8 mean (ste): 0.163 (0.00694)
setup: 1 | noise: False | data size: 128 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.169 (0.00288) Pconf in Eq.8 mean (ste): 0.173 (0.00516)
setup: 1 | noise: False | data size: 256 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.167 (0.00224) Pconf in Eq.8 mean (ste): 0.168 (0.00257)
setup: 1 | noise: False | data size: 512 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.169 (0.00207) Pconf in Eq.8 mean (ste): 0.165 (0.00281)
setup: 1 | noise: False | data size: 1028 | seed: 0 1 2 3 4 5 6 7 8 9
PN in Eq.4 mean (ste): 0.169 (0.00165) Pconf in Eq.8 mean (ste): 0.168 (0.00231)

C:\Users\eelab.CCLAB-IITJ\Downloads\irreducible-main>
```

# RESULTS

## CFIAR H

```
Bayes error of animals vs other: 0.502% (0.453%, 0.550%)
Bayes error of land vs other: 1.554% (1.464%, 1.645%)
Bayes error of odd vs other: 2.034% (1.926%, 2.143%)
Bayes error of firstfive vs other: 3.261% (3.123%, 3.399%)
```




(a) Animals vs. artifacts (b) Land vs. other

## ICLR

```
C:\Users\eelab.CCLA
2017: 7.0 \pm 1.0
2018: 8.7 \pm 0.9
2019: 8.0 \pm 0.7
2020: 8.7 \pm 0.5
2021: 9.2 \pm 0.5
2022: 9.6 \pm 0.5
2023: 8.0 \pm 0.4
```

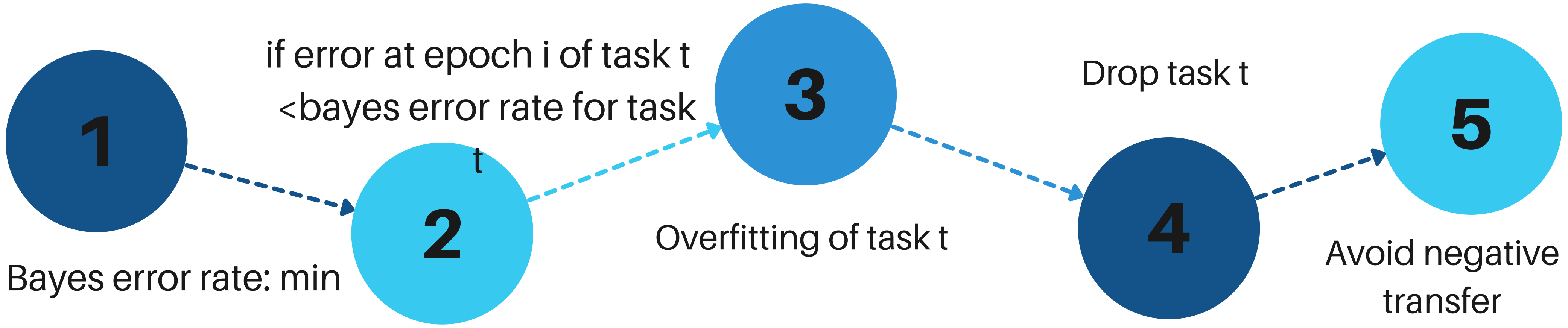
| ICLR's Bayes error | |
|---|---|
| 2017 | 6.8%(±1.0%) |
| 2018 | 8.7%(±0.9%) |
| 2019 | 7.9%(±0.7%) |
| 2020 | 8.8%(±0.5%) |
| 2021 | 9.3%(±0.5%) |
| 2022 | 9.6%(±0.5%) |
| 2023 | 8.0%(±0.4%) |

## FMNIST H

```
C:\Users\eelab.CCLAB-IITJ\Downloads\irreducible-main>
Bayes error of tops vs. other: 3.478% (3.398%, 3.557%
```

estimated Bayes error in paper was 3.478% (± 0.079%)

# 3. PREVENT OVERFITTING IN MTL (DST)




## 3. 1 ALGORITHIM

Pb_t=1

*calculate Bayes error for for all task*

B_t=Bayes error for take t

FOR n IN number_of_epochs:

IF (epoch==0):

*Calculate V1 for every task*

V1_t =loss at epoch 0 for a task t(Do this for all the tasks)

ELSE:

Vn_t= loss at epoch n for a task t(Do this for all the tasks)

*calculate the BayesGate for every task (returns 1 if current error rate <B_t else 0*

Bt= BayesGate for task t

*Calculate, the amount of "incompleteness"*

In_t = Vn_t/V1_t (Do this for all the tasks)

E(I(n)) = mean([In_t for t in range(num_tasks)])

Pn_t = MIN(1,In_t/E(I(n)))(Do this for all the tasks)

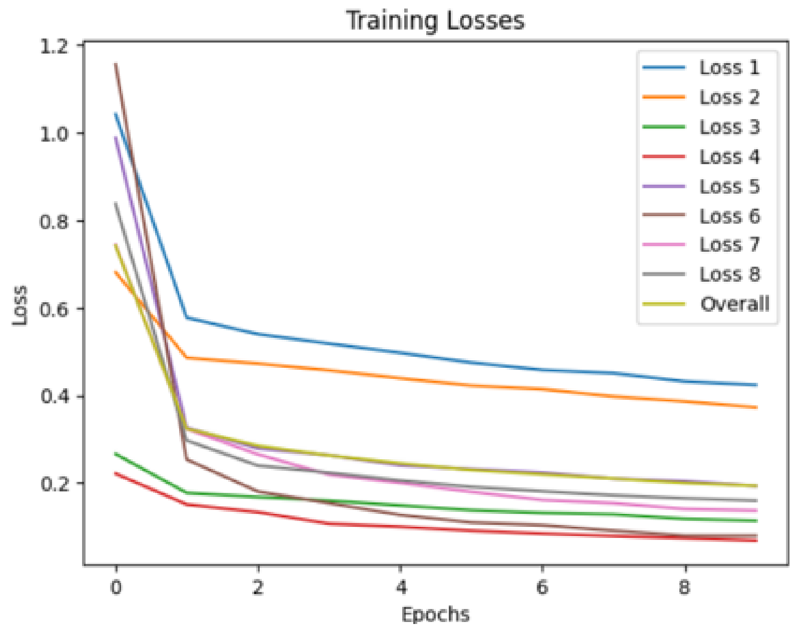
Pn_t = lambdap*Pn_t + (1-lambdap)*Pb_t

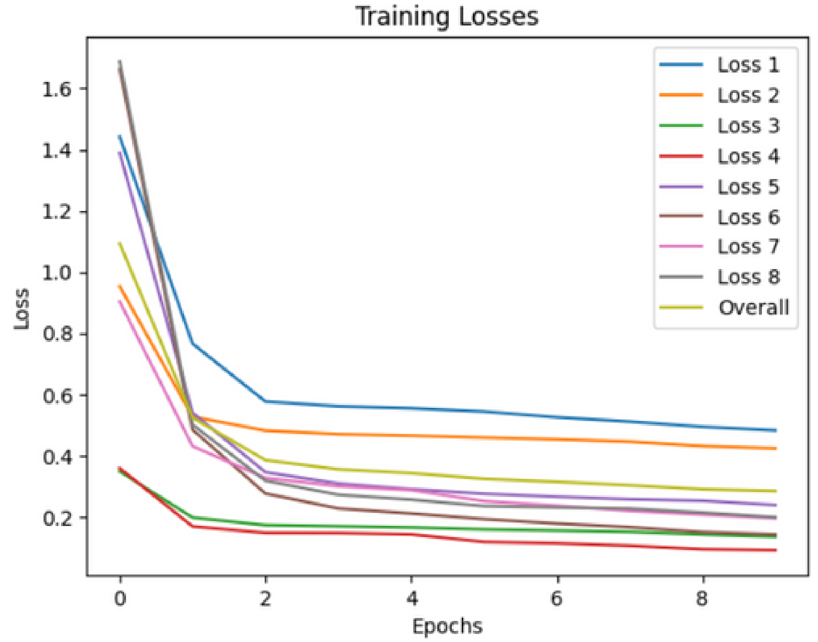
Gn_t=Bt*Bernoulli(Pn_t )

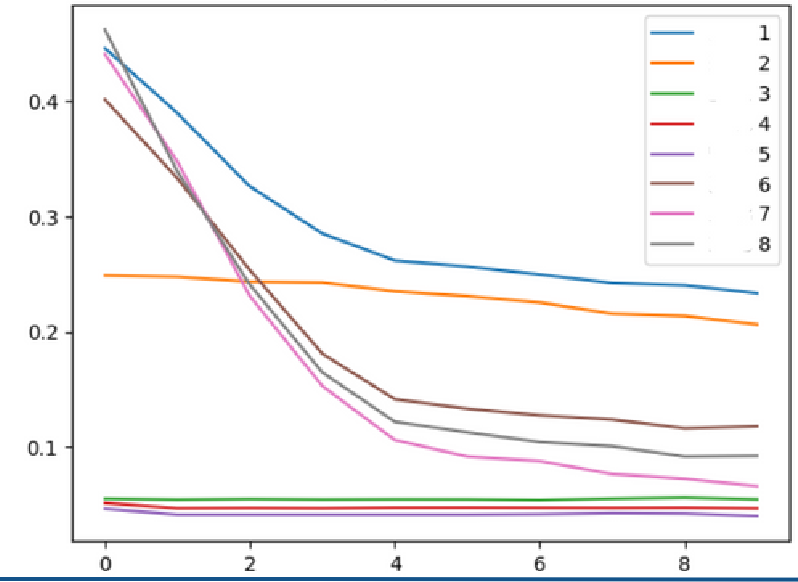
## 3. 3 DETAILS

Used celebA-H dataset for multitask learning, classes with top 8 MI were chosen

## 3.2 RESULT




| DST' | W/O DST |
|---|---|
| 88.33 | 87.27 |
| 75.58 | 70.78 |
| 79.06 | 78.55 |
| 93.98 | 92.81 |
| 95.06 | 96.86 |
| 94.86 | 86.16 |
| 87.25 | 92.01 |
| 90.88 | 90.86 |
| 90.01 | 90.16 |

# CHATGPT

The paper "Is the Performance of My Deep Network Too Good to Be True? A Direct Approach to Estimating the Bayes Error in Binary Classification" proposes a method for estimating the Bayes error, which is the lowest possible error rate that can be achieved by a classifier given the underlying distribution of the data. Here are some potential applications for this paper:

1. Model selection: The Bayes error can provide a benchmark for the performance of a binary classifier. If the estimated error rate is close to the Bayes error, it suggests that the model is performing well and is not overfitting the training data.
2. Detecting overfitting: If the estimated error rate is significantly lower than the Bayes error, it suggests that the model may be overfitting the training data. This can be a useful diagnostic tool to identify when a model is likely to perform poorly on new data.
3. Evaluating data quality: The Bayes error is a function of the underlying distribution of the data. If the estimated error rate is significantly higher than the Bayes error, it suggests that the data may be noisy or contain outliers.
4. Model debugging: In some cases, a model may achieve a high accuracy on the training set but perform poorly on the test set. Estimating the Bayes error can help to identify whether the model is learning a spurious pattern in the training data or whether there is a genuine problem with the model architecture or training procedure.
5. Benchmarking performance: Estimating the Bayes error can provide a useful benchmark for comparing the performance Regenerate response a given dataset. This can be

can this be used to detect overfitting in multi-task learning tasks?

Yes, the method proposed in the paper "Is the Performance of My Deep Network Too Good to Be True? A Direct Approach to Estimating the Bayes Error in Binary Classification" can be used to detect overfitting in multi-task learning tasks as well. In multi-task learning, a model is trained to perform multiple related tasks simultaneously, and overfitting can occur if the model learns to exploit spurious correlations between tasks.

The Bayes error is a measure of the lowest possible error rate that can be achieved by a classifier given the underlying distribution of the data. In the context of multi-task learning, the Bayes error can be estimated for each task separately, and discrepancies between the estimated error rates can indicate the presence of overfitting.

For example, if a model achieves a significantly lower error rate on one task compared to the estimated Bayes error, it suggests that the model may be overfitting to the training data for that task. On the other hand, if the Regenerate response estimated Bayes error for all tasks, it suggests that the model is not overfitting and is generalizing well to new data.

```python
def estimate_bayes_error(soft_labels):
    """
    Estimate the Bayes error from soft labels.

    Args:
        soft_labels: A numpy array of shape (n_samples, n_classes) r

    Returns:
        A float value representing the estimated Bayes error.
    """
    n_samples, n_classes = soft_labels.shape

    # Compute the empirical mean and covariance of the soft labels
    mean = np.mean(soft_labels, axis=0)
    cov = np.cov(soft_labels, rowvar=False)

    # Compute the inverse of the covariance matrix
    cov_inv = np.linalg.inv(cov)

    # Compute the Bayes error using the soft labels and the estimated
    bayes_error = 1 - np.max(np.sum(mean * cov_inv * mean, axis=1))

    return bayes_error
```

write a pseudocode to train a multitask model which uses estimated bayes error to drop tasks that are overfitting also use incompleteness along with Metric based on Task Incompleteness in Dropped Scheduled Task algorithm

shouldn't be inside the training loop

```python
    # Compute the estimated Bayes error for each task
    task_bayes_errors = []
    for task in tasks:
        task_bayes_error = model.estimate_bayes_error(task, validation_data[
        task_bayes_errors.append(task_bayes_error)
```

```python
for i, task in enumerate(tasks):
    if task_errors[task][-1] < task_bayes_errors[i] or task_incompletene
        task_dropped[task].append(epoch)
        model.drop_task(task)
```

```python
def return_bayes(soft_labels1,hard_labels1):
  hard_labels1= return_bool(hard_labels1)
  soft_labels1=soft_labels1.cpu().detach().numpy()
  min_soft = np.min(np.stack((soft_np, 1-soft_np)), axis=0)
  hatBER = np.mean(min_soft)

  conf_level = 0.95
  df = len(soft_np) - 1
  hatBERse = stats.sem(min_soft)
  conf_interval = scipy.stats.t.interval(conf_level, df, hatBER, hatBERse)
  return hatBERse
```