Data Shapley: Equitable Valuation of Data for ML

-Summary-

Introduction

Notation & Question

- Training set : $D = \{(x_i, y_i)\}_{i=1}^{n}$
- Learning algorithm : $\mathcal{A}$ (ex : NN, logistic regression …)
- Performance score : $V(\mathrm{S}, \mathcal{A})$ $(= V(\mathrm{S}))$ (ex : test accuracy) , where $S \subseteq \mathcal{D}$

  (performance score of the predictor trained on the data S using learning algorithm $\mathcal{A}$)
- Data value of i-th datum : $\phi_i(V)$ $(= \phi_i)$

Question 1 : What is equitable measure of the value of each $(\mathrm{x_i}, \mathrm{y_i})$ to the learning algorithm $\mathcal{A}$ with respect to the performance metric V?

➢ Common method = leave-one-out (LOO) : Compare $V(D, \mathcal{A})$ and $V(D - \{i\}, \mathcal{A})$ (use entire dataset)

Question 2 : How do we efficiently compute this data value (from question 1) in practical settings?

Equitable Data Valuation for ML

Equitable properties of data valuation

Question : What does equitable (=fair) means in ML?

1. (Valueless) :

For all $S \subseteq D - \{i\}$, $V(S) = V(S \cup \{i\})$, then $\phi_i = 0$

2. (Symmetric) :

For all $S \subseteq D - \{i, j\}$, $V(S \cup \{i\}) = V(S \cup \{j\})$, then $\phi_i = \phi_j$

3. (Additivity) :

$\phi_i(V + W) = \phi_i(V) + \phi_i(W)$ for performance scores V and W

(In most ML settings, $V = -\sum_{k \in test\ set} V_k$, where V_k = loss on kth test point)

Proposition 2.1

weighted sum of all possible marginal contributions of datum i

Any data valuation $\phi(D, \mathcal{A}, V)$ that satisfies properties 1-3 above must have the form

$$\phi_i = C \sum_{S \subseteq D - \{i\}} \frac{V(S \cup \{i\}) - V(S)}{\binom{n-1}{|S|}}$$

where C is an arbitrary constant. We call ϕ_i the 'data Shapley value' of point i.

Approximating Data Shapley

Monte Carlo method for approximation

Problem : Direct calculation of ϕ_i is extremely cost-inefficient => Intractable

1. Calculation of each $V(S)$ requires huge time to train the model, especially large NN
2. Computing all the possible marginal contributions is exponentially large in the train data size

Monte-Carlo approximation :

1. Observe $\phi_i = \mathbb{E}_{\pi\sim\Pi}[V\left(S_\pi^i \cup \{i\}\right) - V\left(S_\pi^i\right)]$: reformulation of ϕ_i (pick $C = \frac{1}{n}$)

   where S_π^i : the set of data points coming before datum i in permutation / Π = uniform distribution over all $n!$ permutations of data points / $S_\pi^i = \emptyset$ if i is the first element

2. Sample $\pi_k \sim \Pi$ and calculate $V\left(S_{\pi_k}^i \cup \{i\}\right) - V\left(S_{\pi_k}^i\right)$ for $k \in \{1, \dots, K\}$, K = approximation sample size

3. Calculate $\frac{1}{K}\sum_{k=1}^{K} V\left(S_{\pi_k}^i \cup \{i\}\right) - V\left(S_{\pi_k}^i\right)$, which is the MC approximation of ϕ_i.

Note : this process requires still calculation of $V(S)$ for many times, which is intractable part.

Approximating Data Shapley

Truncated Monte Carlo Shapley (TMC-Shapley)

Intuition :

1. As the size of S increases, the change in performance $(= V(S \cup \{i\}) - V(S))$ by adding only one more training point becomes smaller (Mahajan et al., 2018)

2. Since train set is finite in practice, $V(S)$ is also an approximation of true performance, So it is sufficient to estimate Shapley value up to the intrinsic noise in $V(D)$
   (i.e : $V(S) \in (V(D) - \delta, V(D) + \delta)$)
   (Note : the noise(δ) can be quantified by measuring variation in the performance of the same predictor across bootstrap samples of test set)

Approximating Data Shapley

Algorithm 1 Truncated Monte Carlo Shapley

Input: Train data $D = \{1, \ldots, n\}$, learning algorithm $\mathcal{A}$, performance score V
Output: Shapley value of training points: $\phi_1, \ldots, \phi_n$

Initialize $\phi_i = 0$ for $i = 1, \ldots, n$ and $t = 0$
while Convergence criteria not met **do**
  $t \leftarrow t + 1$
  π^t: Random permutation of train data points
  $v_0^t \leftarrow V(\emptyset, \mathcal{A})$
  for $j \in \{1, \ldots, n\}$ **do**
    if $|V(D) - v_{j-1}^t| <$ Performance Tolerance **then**
      $v_j^t = v_{j-1}^t$
    else
      $v_j^t \leftarrow V(\{\pi^t[1], \ldots, \pi^t[j]\}, \mathcal{A})$
    end if
    $\phi_{\pi^t[j]} \leftarrow \frac{t-1}{t}\phi_{\pi^{t-1}[j]} + \frac{1}{t}(v_j^t - v_{j-1}^t)$
  end for
end for

if v_{j-1}^t satisfies performance tolerance, then set $v_j^t, \ldots v_n^t = v_{j-1}^t$ (intuition 1) **[Truncation]**

MC approximation process

Truncated Monte Carlo Shapley - Algorithm

- Performance Tolerance is calculated based on the variation of V in bootstrap samples of train set (intuition 2)
- Convergence criteria for this paper : $\frac{1}{n}\sum_{i=1}^{n}\frac{|\phi_i^t - \phi_i^{t-100}|}{|\phi_i^t|} <$ 0.05

Approximating Data Shapley

Problem of TMC-Shapley / Gradient Monte Carlo Shapley (G-shapley)

For some algorithms $\mathcal{A}$ = logistic regression, LASSO (= linear regression + regularization of parameter L1-norm), It is tractable to calculate $V(S)$ (takes small time to calculate)

Problem : For a deep NN, calculating each $V(S)$ is intractable

Suggested solution :

- For a predictive model $\mathcal{A}$ using GD, train only one pass (= 1 epoch) for the training data S
  (simple approximation of model)

Note : Use SGD with batch size = 1 / Use bigger learning rates compared to ones used for multi-epoch training (result from author's experiments)

Approximating Data Shapley

Algorithm 2 Gradient Shapley

Input: Parametrized and differentiable loss function $\mathscr{L}(.;\theta)$, train data $D = \{1, \ldots, n\}$, performance score function $V(\theta)$

Output: Shapley value of training points: $\phi_1, \ldots, \phi_n$

Initialize $\phi_i = 0$ for $i = 1, \ldots, n$ and $t = 0$

while Convergence criteria not met **do**

  $t \leftarrow t + 1$

  π^t: Random permutation of train data points

  $\theta_0^t \leftarrow$ Random parameters

  $v_0^t \leftarrow V(\theta_0^t)$

  for $j \in \{1, \ldots, n\}$ **do**

    $\theta_j^t \leftarrow \theta_{j-1}^t - \alpha \nabla_\theta \mathscr{L}(\pi^t[j]; \theta_{j-1})$

    $v_j^t \leftarrow V(\theta_j^t)$

    $\phi_{\pi^t[j]} \leftarrow \frac{t-1}{t} \phi_{\pi^{t-1}[j]} + \frac{1}{t}(v_j^t - v_{j-1}^t)$

  end for

end for

Weight update using each single sample data for 1 epoch

MC approximation process

Gradient Shapley - Algorithm

- Convergence criteria for this paper : $\frac{1}{n}\sum_{i=1}^{n} \frac{|\phi_i^t - \phi_i^{t-100}|}{|\phi_i^t|} < 0.05$ (Same as TMC-Shapley)

Experiments & Applications - Disease prediction

Experiment – Cancer prediction

Experiment setting :

- Data set = UK biobank data set
- Task = Based on 285 features, predict whether the patient has a breast cancer or not / a skin cancer or not (binary classification)
- Size of train set = 1000 patients
- Learning algorithm = Logistic regression

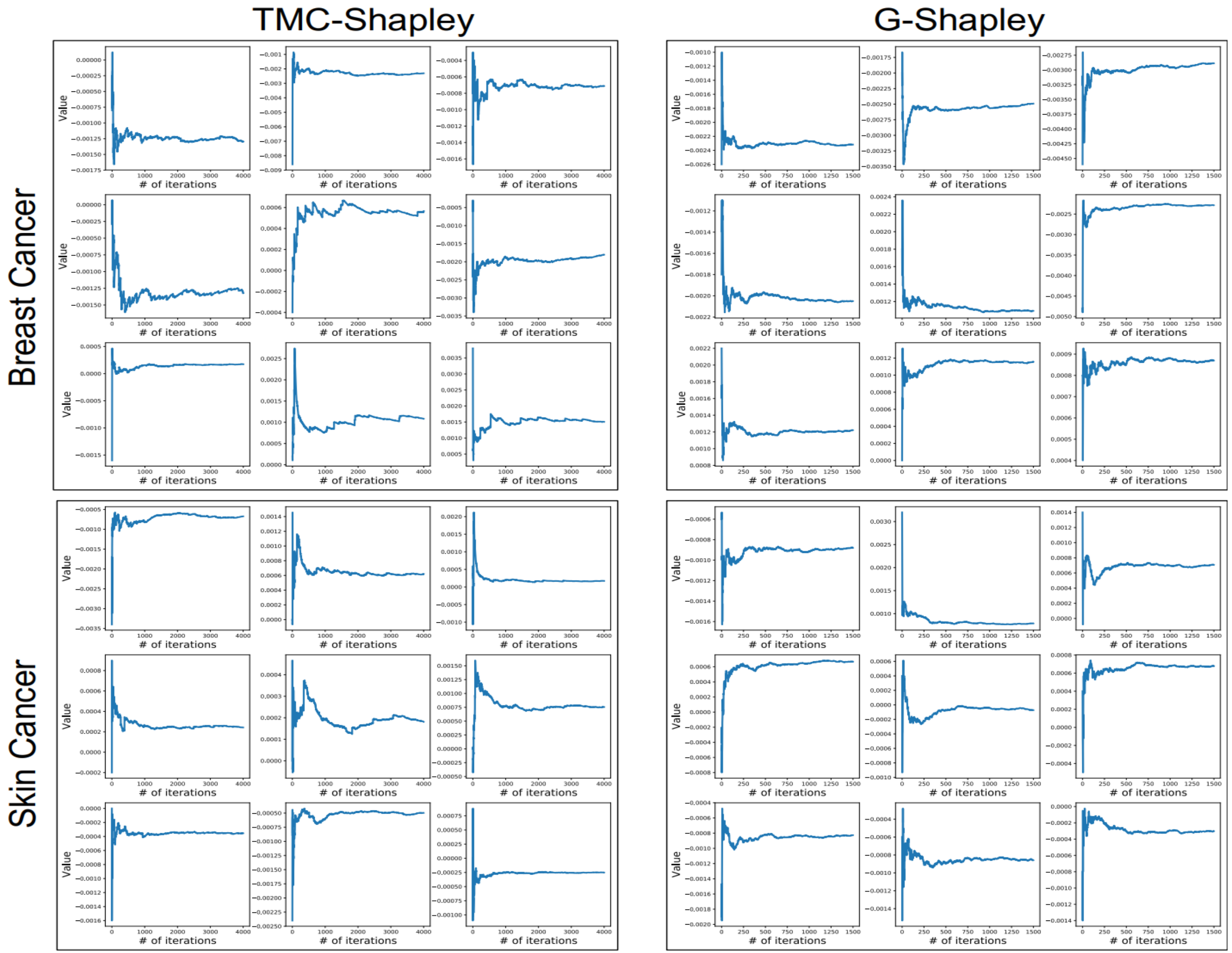
- Result (test accuracy) : 68.7% for breast cancer / 56.4% for skin cancer

- Convergence of TMC-Shapley / G-Shapley : 4000 iterations (TMC-Shapley) / 1500 iterations (G-Shapley)

- Experiment method :

  After calculating data values (approximated Shapley values), Remove and add data based on data values and track the change in model performance.

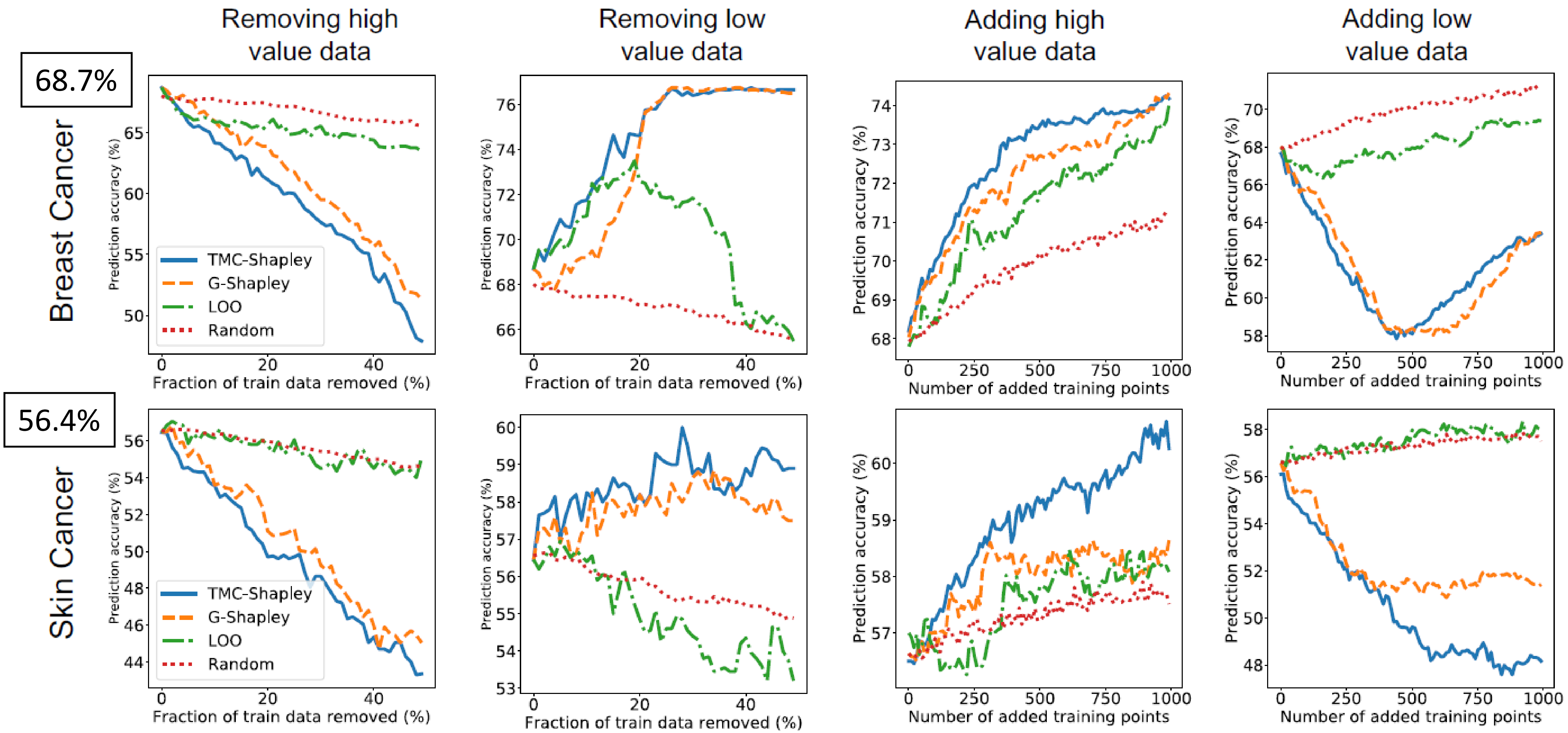
Experiments & Applications - Disease prediction



Convergence of Shapley algorithm:

- Randomly selected 9 training points for each task
  (breast cancer / skin cancer)

Experiments & Applications - Disease prediction



Experiments & Applications – Synthetic Data

Experiment – Synthetic Data

Experiment setting :

- Data set = Synthetic data
  1. $x_i \sim N(0, I_{50\times 50}) \in \mathbb{R}^{50}$ / $y_i \sim Bern(p_i)$, where $p_i = f(x_i) \in [0,1]$
  2. $f()$ is linear for 20 data sets / 3rd order polynomial for another 20 data sets

- Task = Based on 50 features, predict whether $y_i = 1$ or $y_i = 0$ (binary classification)
- Size of train set = 100 / 1000 (perform two experiments)
- Learning algorithm = Logistic regression (linear $f()$ set) / Logistic regression or NN(with one hidden layer) (3rd order polynomial $f()$ set)

- Experiment method :

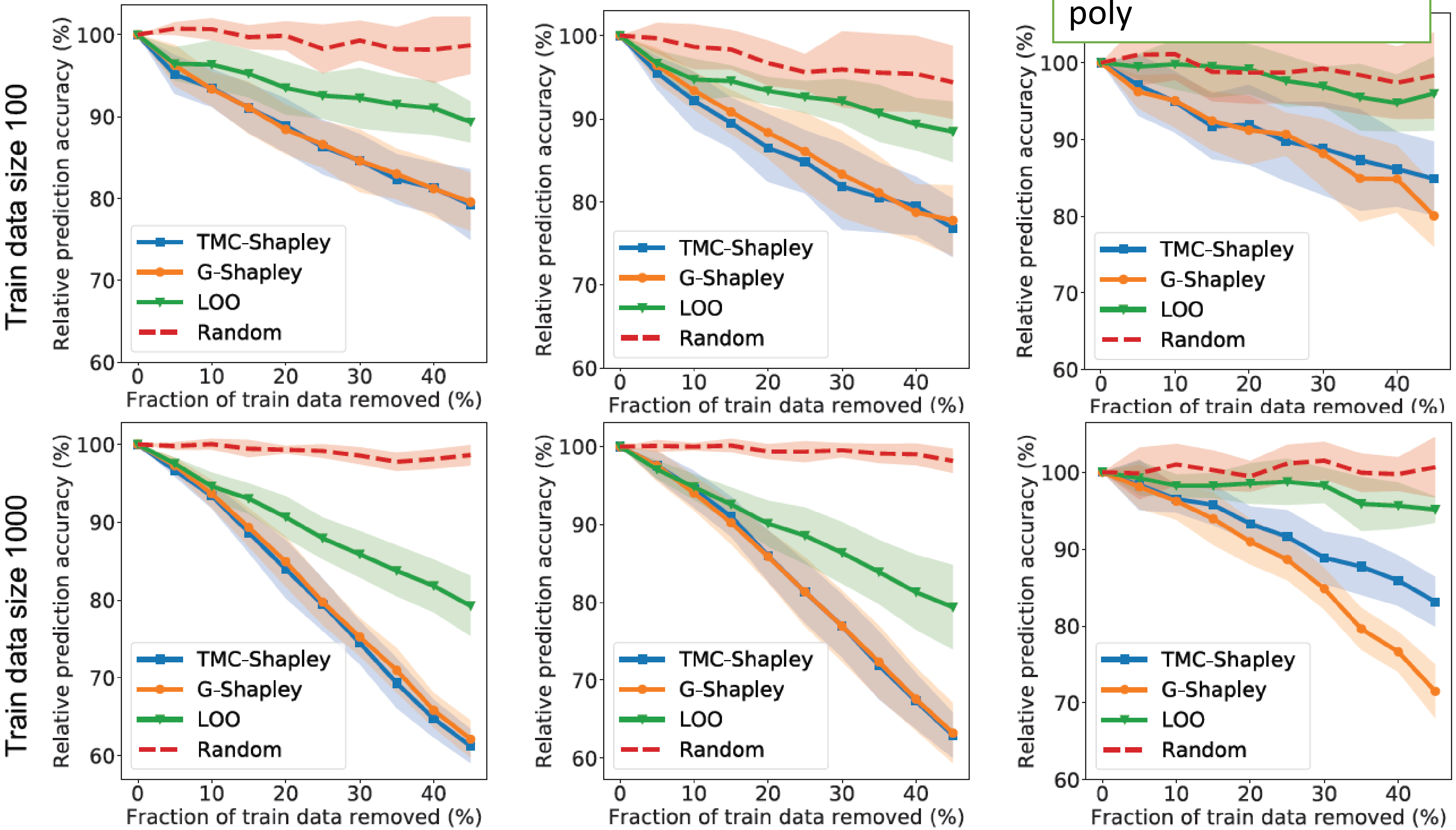
  Remove training points from the most valuable to the least valuable and track the change in model performance

Experiments & Applications – Synthetic Data

LR / f = linear

LR / f = 3rd order poly

NN / f = 3rd order poly

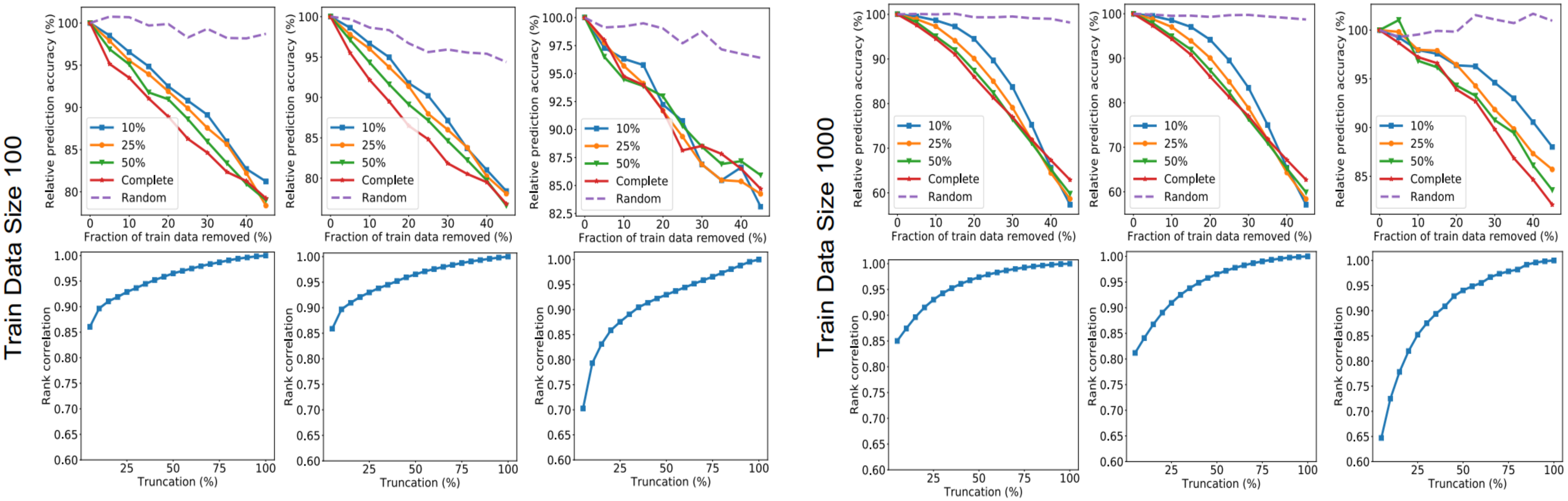


Color shaded area = std over results of 20 data sets

Relative prediction accuracy = accuracy with removal / accuracy without removal

Experiments & Applications – Synthetic Data - Appendix

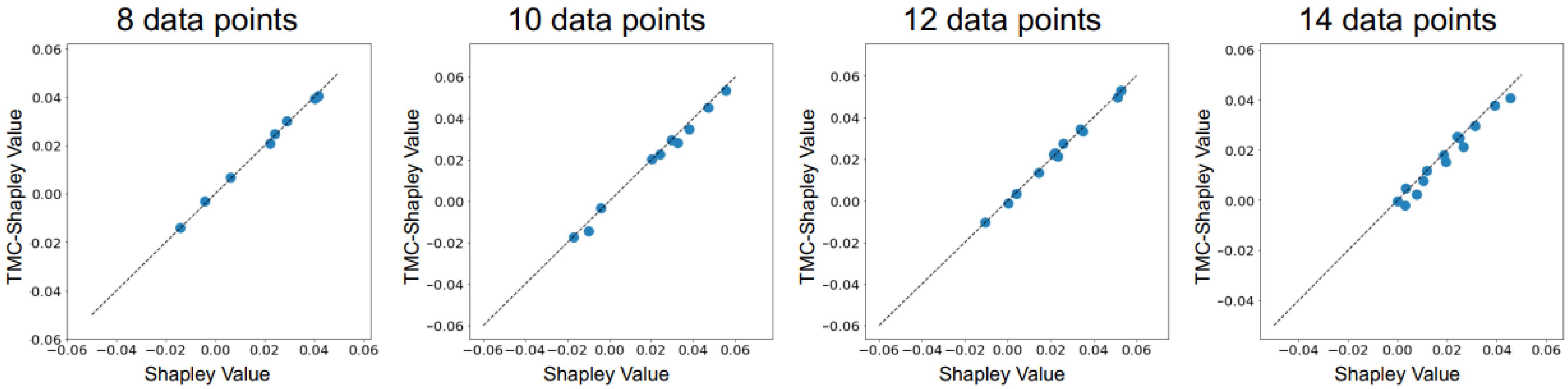
Does the approximation via TMC-Shapley is reasonable? => reasonable (by experiments)



Truncation $\alpha\%$: Calculate the marginal contributions of the first $\alpha\%$ elements of permutation and approximate the remaining by zero marginal contribution

Experiments & Applications – Synthetic Data - Appendix

Does the approximation via TMC-Shapley is reasonable? => reasonable (by experiments)



Experiments & Applications – Label Noise

Can we check and correct the mislabeled examples by inspecting the data values?

Experiment – Label Noise

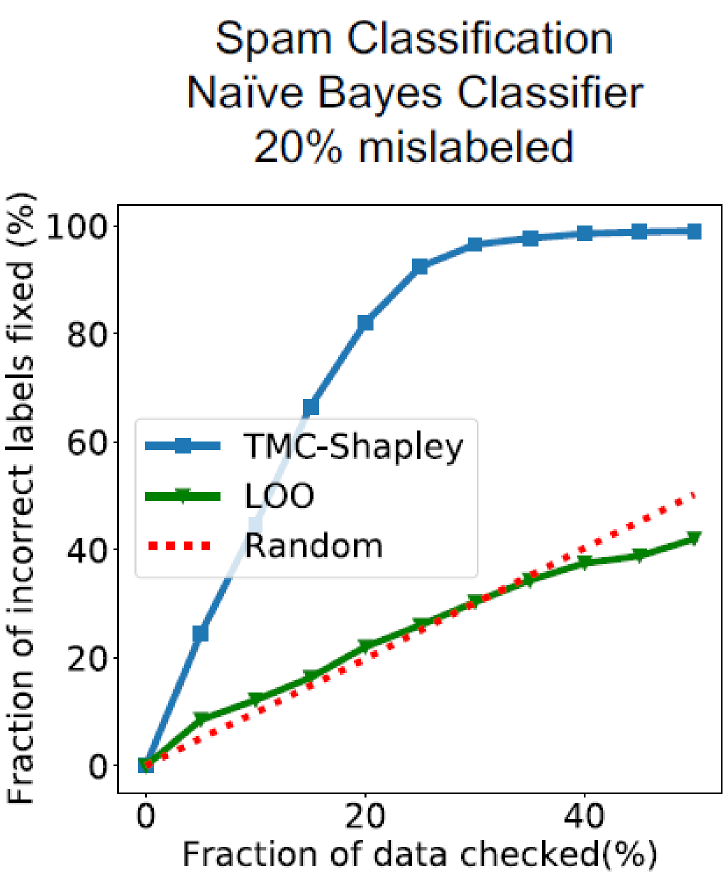
Experiment setting :

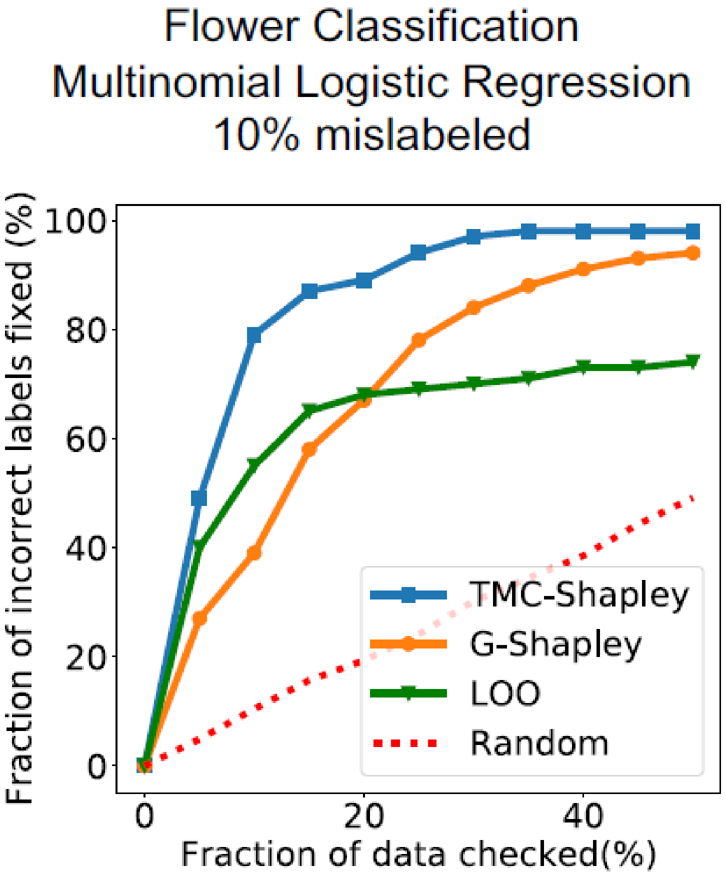
1. Spam classification
   - Data set = spam classification data / 2 classes / 3000 data points (train set)
   - Learning algorithm = Multinomial Naïve Bayes model (not use GD algorithm => G-Shapley X)
   - Convergence = 5000 iter (TMC-Shapley)
   - Preprocess : randomly flip the label of 20% training points

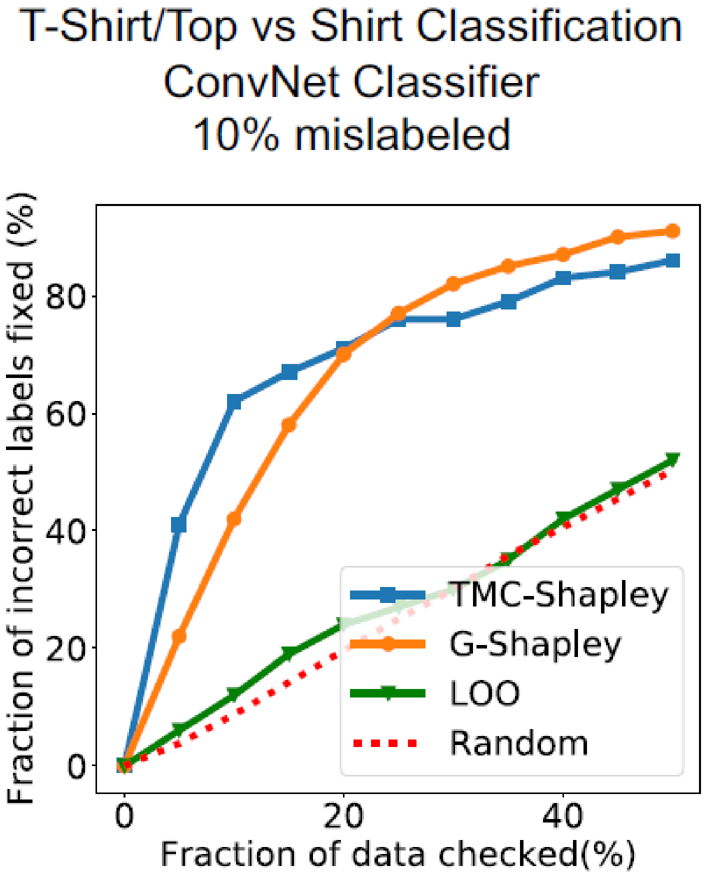
2. Flower image classification
   - Data set = flower image classification data / 5 classes / 1000 data points (train set)
   - Learning algorithm = multinomial logistic regression
   - Convergence = 2000 iter (TMC-Shapely / G-Shapley)
   - Preprocess : pass flower images through Inception-V3(CNN) and train the logistic model on the learned CNN representation of 1000 images / 10% of training points' labels are flipped

3. Clothes classification
   - Data set = Fashion MNIST / 2 classes / 1000 data points (train set)
   - Learning algorithm = CNN (one convolutional layer + two linear layers)
   - Convergence = 2000 iter (TMC-Shapley / G-Shapley)
   - Preprocess : 10% of data points' labels are flipped

Experiments & Applications – Label Noise



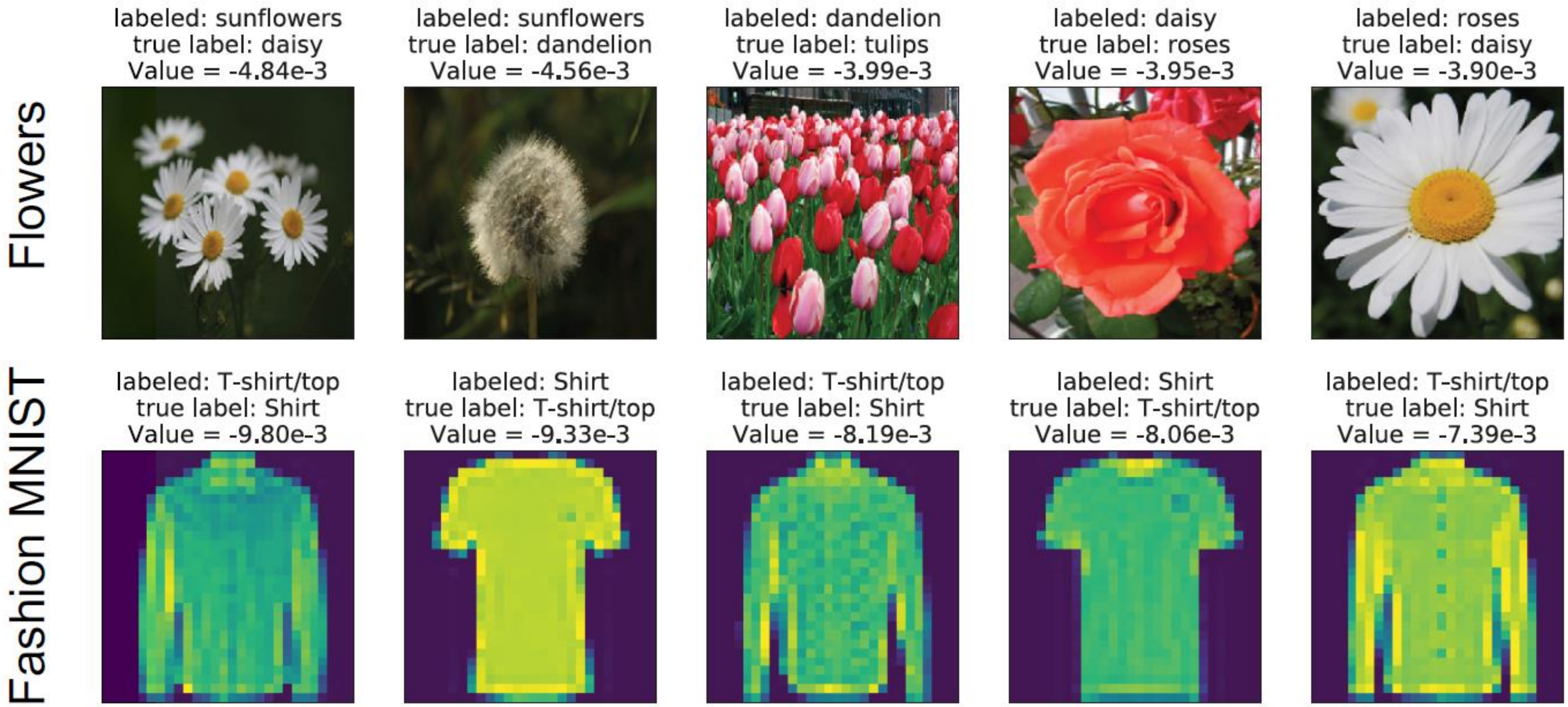




Scan data from lowest value to highest value

Data inspection is done in manual

Experiments & Applications – Label Noise



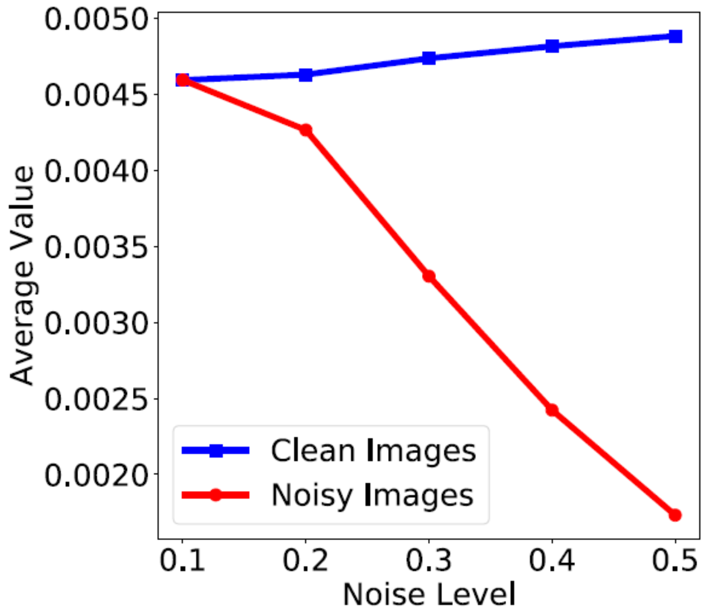
Images with the least TMC-Shapley value = > All of them are mislabeled

Experiments & Applications – Data Quality and Value

Experiment – Data Quality and Value

Experiment setting :

- Data set = Dog vs Fish data / 2 classes / 100 data points (train set) / 1000 data points (test set)
- Learning algorithm = Inception-V3 (CNN)
- Preprocess :
    1. Corrupt 10% of train data by adding white noise.
    2. Compute the average TMC-Shapley value of clean and noisy images.
    3. Repeat the same experiment with different levels of noise.



As the noise level increases, the average TMC-Shapley value of noisy images becomes decrease compared to clean images

Note :
Calculating Shapley values took < 24hrs (4 CPU x 4 by parallel) in 3 experiments.
But, experiment using CNN (above) took 120hrs (4 GPU x 4)