

A APPENDIX

APPENDIX

The appendix is organized as follows:

- Appendix [A](#) includes additional detailed algorithms in the Automatic-Dataset-Construction pipeline.
- Appendix [B](#) contains dataset statistics and more exploratory data analysis of Clothing ADC.
- Appendix [C](#) includes experiment details of our benchmark on label noise detection, label noise learning, and class-imbalanced learning.

BROADER IMPACTS

Our paper introduces significant advancements in dataset construction methodologies, particularly through the development of the Automatic Dataset Construction (ADC) pipeline:

- **Reduction in Human Workload:** ADC automates the process of dataset creation, significantly reducing the need for manual annotation and thereby decreasing both the time and costs associated with data curation.
- **Enhanced Data Quality for Research Communities:** ADC provides high-quality, tailored datasets with minimal human intervention. This provides researchers with datasets in the fields of label noise detection, label noise learning, and class-imbalanced learning, for exploration as well as fair comparisons.
- **Support for Customized LLM Training:** The ability to rapidly generate and refine datasets tailored for specific tasks enhances the training of customized Large Language Models (LLMs), increasing their effectiveness and applicability in specialized applications.

Furthermore, the complementary software developed alongside ADC enhances these impacts:

- **Data Curation and Quality Control:** The software aids in curating and cleaning the collected data, ensuring that the datasets are of high quality that could compromise model training.
- **Robust Learning Capabilities:** It incorporates methods for robust learning with collected data, addressing challenges such as label noise and class imbalances. This enhances the reliability and accuracy of models trained on ADC-constructed datasets.

Together, ADC and its accompanying software significantly advance the capabilities of machine learning researchers and developers by providing efficient tools for high-quality customized data collection, and robust training.

LIMITATIONS

While ensuring the legal and ethical use of datasets, including compliance with copyright laws and privacy concerns, is critical, our initial focus is on legally regulated and license-friendly data sources available through platforms like Google or Bing. Addressing these ethical considerations is beyond the current scope but remains an essential aspect of dataset usage.

Besides, similar to Traditional-Dataset-Construction (TDC), Automatic-Dataset-Construction (ADC) is also unable to guarantee fully accurate annotations.

A DETAILED ALGORITHMS IN THE GENERATION OF AUTOMATIC-DATASET-CONSTRUCTION

A.1 THE ALGORITHM OF IMAGE DATA COLLECTION IN ADC

Algorithm 2 Image Data Collection in ADC

```

1: procedure IMAGEDATACOLLECTION
2:   Part A: Get attributes from dataset design
3:   attributes  $\leftarrow$  Step 1 Dataset Design
4:   categories  $\leftarrow$  ["sweater", "shirt", "pants", ...]  $\triangleright$  List of categories
5:   target_category  $\leftarrow$  "sweater"  $\triangleright$  Target category (e.g. "sweater")
6:   attributes  $\leftarrow$  attributes[target_category]  $\triangleright$  Get attributes for target category
7:   colors, patterns, materials  $\leftarrow$  attributes["color"],
8:   attributes["pattern"],
9:   attributes["material"]
10:  Part B: Create search queries
11:  search_queries  $\leftarrow$  { c + p + m + target_category |
12:    c  $\in$  colors,
13:    p  $\in$  patterns,
14:    m  $\in$  materials }  $\triangleright$  (e.g. "beige fisherman cotton sweater")
15:  Part C: Launch distributed image search
16:  image_data  $\leftarrow$  distributed_search(search_queries,
17:    api = Google_Images | Bing_Images,
18:    n_process = 30)
19: end procedure

```

A.2 THE ALGORITHM OF LEARNING-CENTRIC CURATION METHOD IN ADC

Algorithm 3 Learning-centric curation (early-learning memorization behavior)

```

1: procedure EARLYSTOPCE(noisyDataset, percentage=25%)
2:   Part A: Train classifier over the dataset and apply early stopping
3:   D  $\leftarrow$  Load training data  $\triangleright$  (images and labels)
4:   model  $\leftarrow$  Initialize neural network model  $\triangleright$  (e.g. ResNet)
5:   loss_fn  $\leftarrow$  Define loss function  $\triangleright$  (e.g. cross-entropy)
6:   optimizer  $\leftarrow$  Choose optimizer  $\triangleright$  (e.g. SGD, Adam)
7:   for epoch = 1 to E  $\in$  {1, 2} do
8:     model  $\leftarrow$  Trainer(D, loss_fn, optimizer)
9:   end for
10:  Part B: Record predictions and confidence levels
11:  for batch in D do
12:    images  $\leftarrow$  Get batch of images
13:    outputs  $\leftarrow$  Forward pass: model(images)
14:    confidence  $\leftarrow$  Get confidence levels: softmax(outputs)
15:  end for
16:  Part C: Remove samples with lowest x% confidence level
17:  threshold  $\leftarrow$  Calculate threshold: percentile(confidence, 100 - x)
18:  D  $\leftarrow$  Filter out samples with confidence below threshold
19:  Return D
20: end procedure

```

B DATASET STATISTICS IN CLOTHING-ADC

B.1 COLLECTED CLOTHING ADC DATASET

Our collected Clothing-ADC dataset can be found here: [Google Drive](#).

B.2 ATTRIBUTES CANDIDATES IN CLOTHING-ADC

Our automated dataset creation pipeline is capable of generating numerous designs per attribute, as shown in Table 6. This table provides a detailed list of designs generated by our pipeline, from which we selected a subset to include in our dataset.

Color			Material			Pattern					
Animal print	Gold	Pastel	Acrylic	Lace	Tulle	Abstract	Camouflage	Fishnet	Leather	Printed	Thongs
Beige	Gray	Peach	Alpaca	Leather	Tweed	Abstract Floral	Chalk stripe	Floral	Logo	Quilted	Tie-Dye
Black	Green	Pink	Angora	Lightweight	Twill	Animal Print	Check	Floral print	Low rise	Reversible	Tie-dye
Blue	Grey	Plum	Bamboo	Linen	Velvet	Animal print	Checkered	Fringe	Mesh	Ribbed	Toile
Blush Pink	Heather	Purple	Breathable	Mesh	Viscose	Aran	Chevron	G-strings	Military	Ripples	Trench
Bright Red	Ivory	Red	Cashmere	Microfiber	Water-resistant	Argyle	Color block	Galaxy	Mock turtleneck	Satin	Tribal
Brown	Khaki	Rich Burgundy	Chambray	Modal	Windproof	Aztec	Colorblock	Garter Stitch	Mosaic	Scales	Tuck stitch
Burgundy	Lavender	Royal Blue	Chiffon	Mohair	Wool	Basket check	Cotton	Garter stitch	Moss stitch	Seamless	Tweed
Burnt Orange	Light Grey	Rust	Corduroy	Neoprene	acrylic	Basket rib	Cropped	Geometric	Mato	Seed stitch	Twill
Champagne	Maroon	Rustic Orange	Cotton	Nylon	bamboo	Basket weave	Damask	Gingham	Nailhead	Shadow stripe	Vintage-inspired
Charcoal	Metallic	Sage	Crochet	Organza	cotton	Basketweave	Denim	Glen check	Nehru	Sharkskin	Waterproof
Charcoal Grey	Mustard	Silver	Denim	PVC	hemp	Batik	Diagonal grid	Gradient	Nordic	Sherpa	Windowpane
Cream	Mustard Yellow	Soft Pink	Down	Polyester	linen	Bikini	Diamond	Graphic	Ombre	Silk	
Cream White	Navy	Striped	Embroidered	Rayon	lycra	Birdseye	Ditsy	Grid	Oversized	Slip Stitch	
Dark Plum	Navy Blue	Tan	Flannel	Reflective	modal	Blazer	Dogtooth	Herringbone	Oxford	Slip stitch	
Deep Blue	Neon	Teal	Fleece	Ripstop	nylon	Bomber	Embossed	High waisted	Paisley	Solid	
Deep Purple	Nude	Turquoise	Fringe	Satin	polyester	Boxer briefs	Embroidered	Honeycomb	Peacoat	Striped	
Earthy Beige	Olive	Vibrant Turquoise	Fur	Silk	rayon	Briefs	Emoji	Houndstooth	Pin Dot	Stripes	
Floral	Olive Green	Warm Brown	Gore Tex	Softshell	silk	Brioche	Entrelac	ikat	Pinstripe	Studded	
Forest Green	Orange	White	Gore-Tex	Spandex	spandex	Broken rib	Eyeclet	Intarsia	Plaid	Suede	
Fuchsia	Pale Yellow	Yellow	Hemp	Suede	tencel	Broken stripe	Fair Isle	Jacquard	Polka Dot	Tartan	
		liac	Insulated	Synthetic	viscose	Cable	Fibonacci	Knit and Purl	Polka dot	Teddy	
			Jersey	Synthetic Blend	wool	Cable knit	Fisherman	Lace	Prince of Wales	Textured	
			Knit	Tencel							

Table 6: The union of attributes across all clothing types in Clothing-ADC dataset.

B.3 HUMAN-IN-THE-LOOP CURATION FOR CLOTHINGADC TESTSET

Our automated dataset collection pipeline enabled us to create a large, noisy labeled dataset. We asked annotators to select the best-fitting options from a range of samples, as shown in Figure 5, with each task including at least 4 samples and workers completing 10 tasks per HIT at a cost of \$0.15 per task, totaling \$150 estimated wage of \$2.5-3 per hour, and after further cleaning the label noise, we ended up with 20,000 samples in our test set. To participate, workers had to meet specific requirements, including being Master workers, having a HIT Approval Rate above 85%, and having more than 500 approved HITs, with the distribution of worker behavior shown in Figure 6.

[Task 6 / 10] Please find 4 or more images of **T-shirt** with:

Color: **Navy**, Material: **silk**, Pattern: **Polka dot**

If you are not familiar with the any of these key words, feel free to search it on Google. Try your best to find the most relevant images. I am genuinely interested in your opinion and generous in accepting your answers.



Previous task

Next task

Figure 5: Collection of Clothing-ADC test set: A filtering task to the worker instead of annotation from scratch.

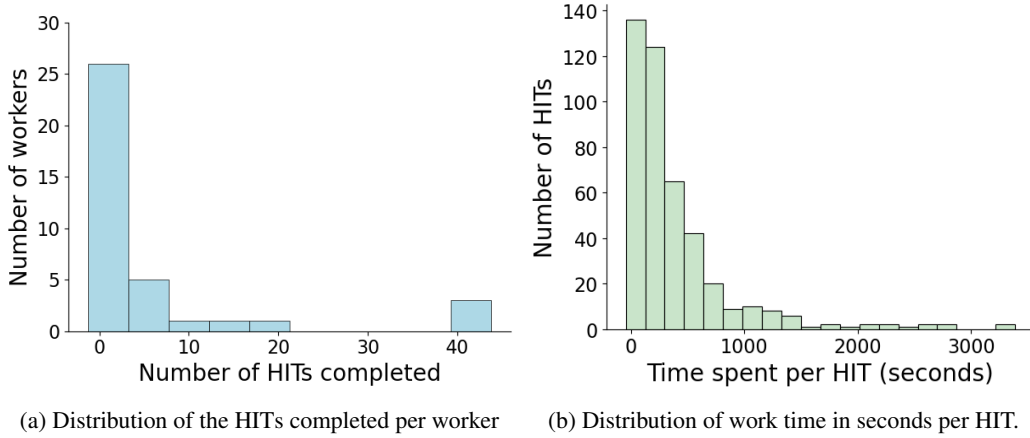


Figure 6: The behaviors of workers in the creation of test set.

B.4 COST ANALYSIS FOR CLOTHINGADC HUMAN-IN-THE-LOOP DATA CURATION

When clean data is required, we recommend combining human involvement with algorithmic approaches to ensure high accuracy. We collected 20,000 samples for both the test set and evaluation set, ensuring a robust and reliable dataset.

We evaluate human effort in Table 7. We used the number of mouse clicks required for each label, excluding overhead costs due to different layout designs across datasets. While other metrics like time spent or monetary cost could be used within the same dataset, they are not easily comparable across datasets with different setups and participants.

Dataset	Class Count	Noise Rate	Label per Sample	Cost per Label (Click)	Total Cost (\$)	Samples Collected
ClothingADC Testset	12k	Clean	4	0.25	\$150 / 150	20k / 20k
Cifar-10 N	10	~18%	1	3	\$450	50k
Cifar-100 N	100	~40%	1	1	\$700	50k
Cifar-10 H	10	5%	1	50	\$3,856.5	20k

Table 7: Human Effort Comparison with Existing Label Noise Datasets.

B.5 "CLEAN SET" FROM TRADITIONAL METHODS IS NOT ALWAYS CLEAN

The noise rate in the manually annotated dataset iNaturalist is close to 0, suggesting that traditional methods requiring experts are more robust than our proposed ADC pipeline. However, we would like to cite Northcutt et al. (2021b) that even well-curated and widely-adopted “clean” test datasets, which have invested significant effort in ensuring data quality, may still contain errors². This highlights that achieving a 0% noise rate is extremely challenging, even with expert annotation. The table below is the evidence of such observations (from Table 2 in Northcutt et al. (2021b)).

Moreover, a “fully-cleaned” set typically consumes much more time and money. When the budget is limited, the annotation accuracy is much lower. For example, the collection of CIFAR-10N Wei et al. (2022b), where each training image of CIFAR-10 (a relatively easy 10-class classification) is assigned to 3 independent annotators. To collect 3 annotations for each of the 50K images, it takes >2 days and >1000 dollars on Amazon Mturk. However, the overall annotation error is approximately 18%. As for CIFAR-100N Wei et al. (2022b), this is a much more challenging task where each annotator is requested to find out the most relevant label for each image among 100 classes (50K images in all). It takes >2 days and > 800 dollars on Amazon Mturk. However, the overall annotation error is approximately 40%.

²<https://labelerrors.com/>

Dataset (Test Set)	Size	% Error
MNIST	10000	0.15
CIFAR-10	10000	0.54
CIFAR-100	10000	5.85
Caltech-256	29780	1.84
ImageNet	50000	5.83
QuickDraw	50426266	10.12
20News	7532	1.09
IMDB	25000	2.90
Amazon Reviews	9996437	3.90
AudioSet	20371	1.35

Table 8: Error comparison across datasets (from Table 2 in Northcutt et al. (2021b))

C EXPERIMENT DETAILS

C.1 DISTRIBUTION OF HUMAN VOTES FOR LABEL NOISE EVALUATION

On the annotation page, we presented the image and its original label to the worker and asked if they believed the label was correct (Figure 7). They input their evaluation by clicking one of three buttons. Note that we encouraged workers to categorize acceptable samples as "unsure". The resulting distribution is shown in Table 9. Using a simple majority vote aggregation, we found that the noise rate in our dataset is 22.15%. However, if a higher level of certainty is required for clean labels, we can apply a more stringent aggregation method, considering more samples as mislabeled. In the extreme case where any doubts from any of the three annotators can disqualify a sample, our automatically collected dataset still retains 61.25% of its samples.

For the label noise evaluation task, we utilized a subset of 20,000 samples from the Clothing-ADC dataset, collecting three votes from unique workers for each sample. Each Human Intelligence Task (HIT) included 20 samples and cost \$0.05. To participate, workers had to meet the following requirements: (1) be Master workers, (2) have a HIT Approval Rate above 85%, and (3) have more than 500 approved HITs. The total cost for this task was \$150, estimated wage of \$2.5-3 per hour.

We show the distribution of worker behavior during the noise evaluation task in Figure 8. Figure 8(a) shows the distribution of the amount of HIT completed per worker while neglecting ids with 1-2 submissions. There is a total of 49 unique workers. Figure 8(b) shows the distribution of time spent per HIT.

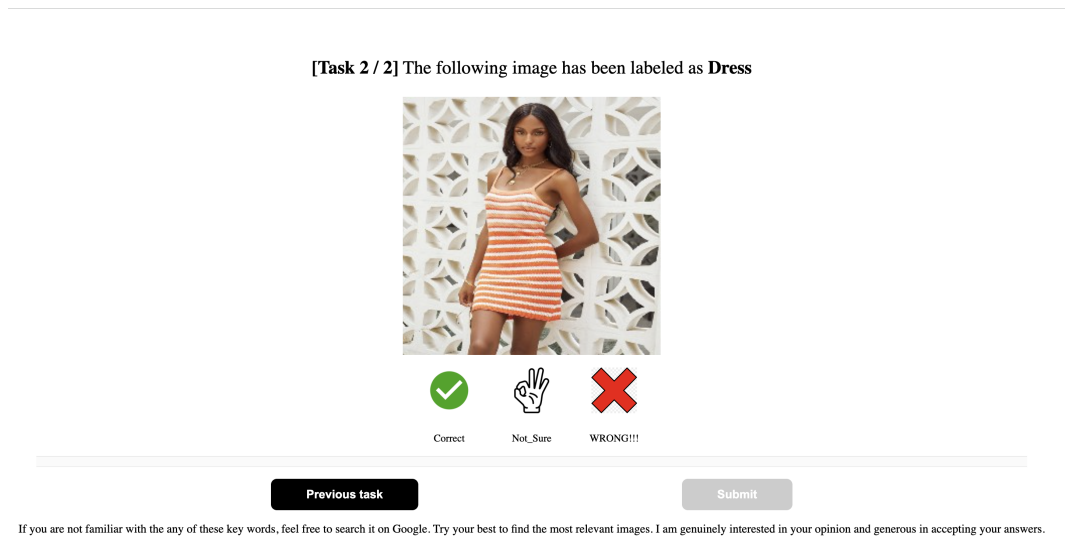
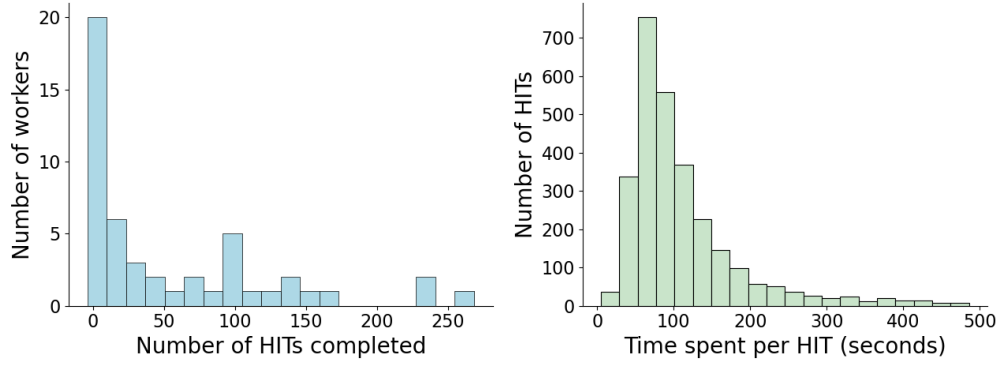


Figure 7: Label noise evaluation worker page



(a) Distribution of the HITs completed per worker (b) Distribution of work time in seconds per HIT.

Figure 8: The behaviors of workers in the collection of label noise evaluation.

Table 9: Distribution of Human Votes for Label Noise Evaluation: We employed human annotators to evaluate a subset of 20,000 samples from our collected dataset, with each sample receiving three votes from distinct annotators.

Human Votes	Percentage
Yes, Yes, Yes	61.25%
Yes, Yes, Unsure	6.10%
Yes, Yes, No	10.50%
Else	22.15%

C.2 NOISY LEARNING AND CLASS IMBALANCE LEARNING BENCHMARK IMPLEMENTATION DETAILS

Our code refers to zip file in supplementary material.

```

1138 1 train_set = Clothing1mPP(root, image_size, split="train")
1139 2 tiny_set_ids = train_set.get_tiny_ids(seed=0)
1140 3 tiny_train_set = Subset(train_set, tiny_set_ids) # Get the tiny version
1141   of the dataset
1142 4 val_set = Clothing1mPP(
1143   5     root, image_size, split="val", pre_load=train_set.data_package
1144   6 )
1144 7 test_set = Clothing1mPP(
1145   8     root, image_size, split="test", pre_load=train_set.data_package
1146   9 )
1147 10
1147 11 train_loader = DataLoader(
1148 12     train_set, batch_size=batch_size, shuffle=True, num_workers=
1149 13     num_workers
1150 14 )
1151 15 tiny_train_loader = DataLoader(
1152 16     tiny_train_set, batch_size=batch_size, shuffle=True, num_workers=
1153 17     num_workers
1154 18 )
1154 19 val_loader = DataLoader(
1155 20     val_set, batch_size=batch_size, shuffle=False, num_workers=
1156 21     num_workers
1157 22 )
1157 23 test_loader = DataLoader(
1158 24     test_set, batch_size=batch_size, shuffle=False, num_workers=
1159 25     num_workers
1160 26 )

```

Listing 1: How to load data. Line 1 loads the full set of our dataset. Line 2 and Line 3 load the tiny version of our dataset. Line 4 creates the validation set. Line 5 creates the testing set. Line 11 to Line 20 create the data loader.

```

1165 1 python examples/main.py --config configs/Clothing1MPP/default.yaml # Run
1166   Cross Entropy
1167 2 python examples/main_peer.py --config configs/Clothing1MPP/default.yaml #
1168   Run Peer Loss
1168 3 python examples/main_jocor.py --config configs/Clothing1MPP/default_jocor
1169   .yaml # Run Jocor
1170 4 python examples/main_coteaching.py --config configs/Clothing1MPP/
1171   default_coteaching.yaml # Run Co-teaching
1172 5 python examples/main_drops.py --config configs/Clothing1MPP/default_drops
1173   .yaml # Run drops

```

Listing 2: The example of the command we use to run the algorithm in one line

```

1175
1176 1 inherit_from: configs/default.yaml
1177 2 data: &data_default
1178   3   root: '/root/cloth1m_data_v3'
1179   4   image_size: 256
1180   5   dataset_name: "clothing1mpp"
1181   6   imbalance_factor: 1 # 1 means no imbalance
1182   7   tiny: False
1183
1183 9 train: &train
1184 10   num_workers: 8
1185 11   loss_type: 'ce'
1186 12   loop_type: 'default' # 'default', 'peer', 'drops'
1187 13   epochs: 20
1187 14   global_iteration: 999999999

```



```

1188 15  batch_size: 64
1189 16  # scheduler_T_max: 40
1190 17  scheduler_type: 'step'
1191 18  scheduler_gamma: 0.8
1192 19  scheduler_step_size: 2
1193 20  print_every: 100
1194 21  learning_rate: 0.01
1195 22
1196 23  general:
1197 24    save_root: './results/'
1198 25    whip_existing_files: True # Whip existing files
1199 26    logger:
1200 27      project_name: 'Clothing1MPP'
1201 28      frequency: 200
1202 29
1203 30  model: &model_default
1204 31    name: "resnet50"
1205 32    pretrained_model: 'IMAGENET1K_V1'
1206 33    cifar: False
1207 34
1208 35  test: &test_defaults
1209 36  <<: *train

```

Listing 3: The example of YAML config file

C.3 LABEL NOISE DETECTION BENCHMARK

We run four baselines for label noise detection, including CORES [Cheng et al. \(2020\)](#), confident learning [Northcutt et al. \(2021a\)](#), deep k -NN [Papernot & McDaniel \(2018\)](#) and Simi-Feat [Zhu et al. \(2022\)](#). All the experiment is run for one time following [Cheng et al. \(2020\)](#); [Zhu et al. \(2022\)](#).

The experiment platform we run is a 128-core AMD EPYC 7742 Processor CPU and the memory is 128GB. The GPU we use is a single NVIDIA A100 (80GB) GPU. For the dataset, we used human annotators to evaluate whether the sample has clean or noisy label as mentioned in Appendix [C.1](#). We aggressively eliminates human uncertainty factors and only consider the case with unanimous agreement as a clean sample, and everything else as noisy samples. The backbone model we use is ResNet-50 [He et al. \(2016\)](#). For all the baselines, the parameters we use are the same as the original paper except the data loader. We skip the label corruption and use the default value from the original repository. For CORES, the cores loss whose value is smaller than 0 is regarded as the noisy sample. For confidence learning, we use the repository³ from the clean lab and the default hyper-parameter. For deep k -NN, the k we set is 100. For SimiFeat, we set k as 10 and the feature extractor is CLIP.

C.4 LABEL NOISE LEARNING BENCHMARK

The platform we use is the same as label noise detection. The backbone model we use is ResNet-50 [He et al. \(2016\)](#). For the full dataset, we run the experiment for 1 time. For the tiny dataset, we run the experiments for 3 times. The tiny dataset is sampled from the full set whose size is 50. The base learning rate we use is 0.01. The base number of epochs is 20. The hyper-parameters for each baseline method are as follows. For **backward and forward correction**, we train the model using cross-entropy (CE) loss for the first 10 epochs. We estimate the transition matrix every epoch from the 10th to the 20th epoch. For the **positive and negative label smoothing**, the smoothed labels are used at the 10th epoch. The smooth rates of the positive and negative are 0.6 and -0.2. Similarly, for **peer loss**, we train the model using CE loss for the first 10 epochs. Then, we apply peer loss for the rest 10 epochs and the learning rate we use for these 10 epochs is $1e-6$. The hyper-parameters for **f -div** is the same as those of peer loss. For **divide-mix**, we use the default hyper-parameters in the original paper. For **Jocor**, the hyper-parameters we use is as follows. The learning rate is 0.0001. λ is 0.3. The epoch when the decay starts is 5. The hyper-parameters of **co-teaching** is similar to Jocor. For logitclip, τ is 1.5. For **taylorCE**, the hyper-parameter is the same as the original paper.

³<https://github.com/cleanlab/cleanlab>

C.5 CLASS-IMBALANCED LEARNING BENCHMARK

The platform we use is the same as label noise detection. The backbone model we use is ResNet-50 He et al. (2016). For different imbalance ratio ($\rho = 10, 50, 100$). The class distribution is shown in Table 10. For all the methods, the base learning rate is 0.0001 and the batch size is 448. The dataset we use is not full dataset because we want to disentangle the noisy label and class imbalance learning. We use Docta and a pre-trained model trained with cross-entropy to filter the data whose prediction confidence is low. Due to the memorization effect, we fine-tune the model for 2 epochs to filter the data. We remove 45.15% data in total where Docta removes 26.36% while CE removes 25.00% with a overlap of 6.20%. Thus, the dataset we use for class-imbalance learning is 54.85% of the full dataset.

imbalance ratio (ρ)	Class Distribution	Total Number
10	[39297, 31875, 25854, 20971, 17010, 13797, 11191, 9078, 7363, 5972, 4844, 3929]	191181
20	[39297, 27536, 19295, 13520, 9474, 6638, 4652, 3259, 2284, 1600, 1121, 785]	129461
100	[39297, 25854, 17010, 11191, 7363, 4844, 3187, 2097, 1379, 907, 597, 392]	114118

Table 10: The class distribution for different imbalance ratio

D DEMO APPLICATION OF ADC IN OTHER FIELDS

Our Automated Dataset Construction (ADC) pipeline is best suited for image classification tasks where the relevant knowledge can be easily searched and retrieved from the internet. Example applications include, but are not limited to:

- Food classification
- Hairstyle classification
- Vehicle classification
- Home decor classification
- Plant classification
- Sport equipment classification
- Jewelry classification

Food Classification To illustrate the effectiveness of our ADC pipeline, let’s consider a more detailed example of food classification. We used the prompt "Food Classification: Create a dataset with various types of cuisine, and sub-classes for specific dishes, ingredients, or cooking methods. Help me to find 10 different attributes to describe food." LLM generated a range of subcategories to describe different types of food, including, but are not limited to:

- Cuisine type (Italian, Chinese, Indian, etc.)
- Dish Type (Appetizer, main course, dessert, etc.)
- Protein source (Beef, Chicken, Tofu, etc.)
- Cooking method (Grilled, Baked, Fried, etc.)
- Spice level (Mild, Medium, Spicy, etc)
- Allergen warning (Gluten-free, Nut-free, Dairy-free, etc.)
- Texture (Crunchy, Chewy, Smooth, etc)

Please feel free to use the prompt on your favorite LLMs, or modify it slightly for other tasks that interest you more. We tried various LLM versions from OpenAI, Meta, Google, and Claude, and all of them are competent to solve this task, albeit with different preferences for suggesting labels and descriptions.

E COPYRIGHT ISSUE

One possible approach to mitigate the potential copyright issues is to rely on the advanced features in search engines provided by the leading industry companies. For example, we can use the "Advanced Image Search => usage rights" function in Google Image Search, which allows users to filter search results by usage rights.

However, We must clarify that our pipeline is provided "as-is" and that users are responsible for using the collected data at their own risk. We cannot guarantee that the data is free from copyright issues, and users must take their own steps to ensure compliance with applicable laws and regulations. This approach is similar to that taken by the LAION-5B dataset [Schuhmann et al. \(2022\)](#), which states that "The images are under their copyright."