Machine Learning (Fall 2025) Course Project

1. Description

Glass quality inspection is a common task in industrial production. In many factories, this process still relies heavily on manual inspection, which is both time-consuming and labor-intensive. With the rapid development of AI, training a model to automatically detect glass quality has become an effective way to reduce cost and improve efficiency. In this project, your task is to implement a model for **glass defect detection**.

2. Dataset

You can download the dataset from here. The dataset is organized as follows:

```
dataset/
    |- train/
    |    | - img/
    |    | - txt/
    |
    |- test/
    |    | - img/
    |    | - txt/
```

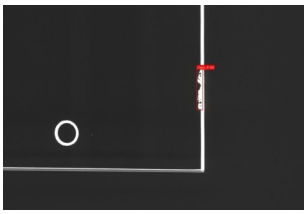
- The dataset has already been split into training and test sets with an **8:2 ratio**.
- The `img` folders contain PNG images of glass, each of size **340 × 340**.
- The `txt` folders contain label files with the **same filename** as the corresponding images.
- Each `.txt` file describes the **types and locations of defects** in that image.
- If there is **no corresponding `.txt` file** for an image, that image should be treated as **non-defective**.

The label files follow the standard **YOLO format**. Each line represents one defect and contains five values:

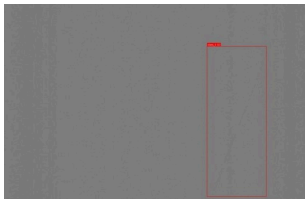
```
<class_id> <x_center> <y_center> <width> <height>
```

Where:

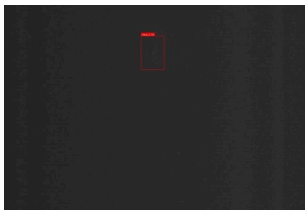
- `class_id` : defect category, there are 3 defect categories:



0: chipped edge



1: scratch



2: stain

- `x_center` : x-coordinate of the defect bounding box center, **normalized by image width**.
- `y_center` : y-coordinate of the defect bounding box center, **normalized by image height**.
- `width` : width of the bounding box, normalized by image width.
- `height` : height of the bounding box, normalized by image height.

All coordinates and sizes are defined with respect to the **top-left corner of the image** as the origin, using pixels before normalization.

Class imbalance:

- The dataset is **imbalanced**: there are fewer defective images than non-defective ones.
- However, the ratio of defective vs. non-defective images is roughly consistent between the training and test sets.

3. Tasks

3.1 Task 1 : Binary Defect Classification

Train a model on the train set to perform **binary classification**:

- **Input:** a glass image.
- **Output:** whether the image is **defective** or **non-defective**.

You should:

- Treat an image as **defective** if its corresponding `.txt` file exists.
- Treat an image as **non-defective** if there is no `.txt` file.

On the test set, you must compute and report at least:

- **Precision** for defective images
- **Recall** for defective images
- **F1-score** for defective images

(You may also report accuracy if you wish, but F1 for defective images is the main focus.)

3.2 Task 2 (Optional): Multi-Label Classification

Formulate the task as a **multi-label classification** task with **four labels**:

1. No defect
2. Chipped edge
3. Scratch
4. Stain

For each image, you should assign **one or more labels**:

- If an image has **no defects**, then it should be labeled as: `no defect = 1` , others = 0.
- If an image has defects, you should mark a label as 1 if **at least one defect of that type** appears in the corresponding `.txt` file.

You should report the **Micro F1-score** of your model on the test set.

4. Requirements

4.1 Implementation Constraints

- **Task 1 (from scratch):**
  - You are **not allowed** to use any **autograd** tools or any **built-in optimization algorithms** from machine learning libraries.
  - For example, if you choose a neural network, you must implement both forward and backward propagation yourself.
  - You can use the packages in the WhiteList. TAs will update the Whitelist if your requirements are reasonable.
- **Task 2 (optional, no restrictions):**
  - You may use **any machine learning libraries** and built-in tools you like (e.g. PyTorch, TensorFlow, scikit-learn, etc.).

4.2 Submission Format

You should submit a compressed package named:

```
[your_student_ID].zip
```

Inside it, there should be a directory with the same name:

```
[your_student_ID]/
    |- Task1/
    |   |- main.py
    |   |- For_TA_test.py
    |   |- ... (your other code, models, scripts, etc.)
    |
    |- Task2/ (optional)
    |   |- main.py
    |   |- For_TA_test.py
    |   |- ... (your other code, models, scripts, etc.)
    |
    |- [your_student_ID]-report.pdf   (your project report)
```

Send this `.zip` file to:

```
ml2025fall_ustc@163.com
```

4.3 Scripts for Training and Evaluation

- `Task1/main.py` and `Task2/main.py` is used by **you** to train your models.
- `Task1/For_TA_test.py` and `Task2/For_TA_test.py` is used by the **TAs** to evaluate your models.

TAs will run your code using the following commands:

```
cd [your_student_ID]
python Task*/For_TA_test.py --test_data_path /home/usrs/dataset/test
```

The script `For_TA_test.py` must:

- Load your trained model.
- Evaluate it on the given test dataset.
- The evaluation must complete execution within **1 hour**.
- Print out the **your_student_ID** and **F1-score** only. **Do not have any other outputs**. For example,

  ```
  PB23000000:0.85
  ```

For **Task 2**, if your program is complex (e.g., requires additional packages, special environment settings, etc.), please include a **`readme.md`** explaining:

- How to set up the environment.

- How to run your code and evaluation script.

4.4 Teamwork

- You may work in a team of up to **3** students. If you are confident in your ability, you may of course complete the assignment as a pair or individually..
- In your report, please list each member's **contribution ratio**, e.g.: `{San Zhang: 30%, Si Li: 35%, Wu Wang: 35%}`.

For team submissions:

- Use the **team leader's student ID** as the directory and package name, e.g. `leaderID.zip` containing `leaderID/`.
- The **team leader** should be the one to submit the package.
- At the **beginning of the report**, list the **names and student IDs of all team members**.

4.5 Trained Models and Dataset

- Remember to **save your trained model(s)**.
- You must submit your **trained model files** along with the code and report.
- The dataset is large; **do not include the dataset** itself in the submitted `.zip` file.

4.6 Report Requirements

Please submit a **detailed report** in PDF format, named:

```
[your_student_ID]-report.pdf
```

The report should include at least:

1. **Introduction**
2. **Methodology**
   - Model architecture
   - Loss functions
   - Optimization algorithm (for Task 1, your own implementation)
   - Data preprocessing, augmentation (if any)
3. **Experimental Settings**
   - Train/validation strategy
   - Hyperparameters
   - Hardware and software environment
4. **Results and Analysis**
   - Quantitative results (precision, recall, F1-score, etc.).

    - **Plots** showing how **precision, recall, and F1-score for defective images** evolve during training on the training set and the test set.
    - Discussion of overfitting/underfitting, class imbalance handling, etc.
5. **Conclusion**

5. Grading

The grading for Task 1 is based on two F1-scores for **defective glass images**.

1. **First score s_1**

   s_1 is the **F1-score for defective images** on the provided test set. You must write this value at the **very beginning of your report** and **highlight it in bold**.

2. **Second score s_2**

   s_2 is the **F1-score for defective images** obtained when the TAs run your `Task1/For_TA_test.py` script on **another hidden test set**.

3. **Overall score s**

   The final score combines s_1 and s_2 as:

   $$s = 0.3s_1 + 0.7s_2,$$

   the hidden test performance s_2 has higher weight to encourage good generalization.

4. **Bonus for Task 2**

   If you complete **Task 2**, the TAs will award **extra credit** based on:

    - Whether the task is completed correctly.
    - The performance (e.g., Micro F1-score).
    - The quality of your modeling and analysis.

6. System Environment

We will evaluate your model on a GeForce RTX 3090 (about 24G memory) under Ubuntu 18.04 system. Please limit the size of your model to avoid OOM.

7. Important Dates

- **Team formation deadline:**

  The **team leader** should inform the TAs of all team members (names and student IDs) **before 23:59, December 8, 2025**.

- **Final submission deadline:**

  Please submit your `[your_student_ID].zip` **before 23:59, January 23, 2026**.