1BM120 - Assignment 2

Convolutional Neural Networks, AutoML and Deep Reinforcement Learning

Dr. Zaharah A. Bukhsh, Dr. Laurens Bliek, Luca Begnardi,
Ya Song, Kjell van Straaten

May 2025

1 Convolutional Neural Networks and AutoML (20 points)

1.1 Problem description

Wefabricate, a Dutch manufacturing company, is building a fully autonomous factory to support the sustainability transition in the manufacturing industry by enabling affordable make-to-order for single products. To do so, they plan to create a visual inspection process to check whether a product is accepted or rejected (e.g., a defect). More specifically, they would like to classify images of products taken along the production line using **Convolutional Neural Networks**. This classification result is then passed to the robot, placing the product in the appropriate bin accordingly.

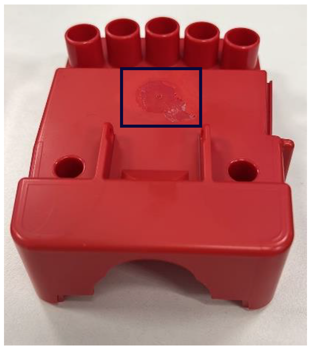

Figure 1: Example product that is scratched in the middle.

Over time, Wefabricate has collected a **balanced and labeled dataset** containing images of one product named the front plate of an industry plug. This product is created through injection molding; a mold is filled with liquid plastic, which turns solid when it cools down. Defects in this process are considered discolored or damaged products. Damages include (but are not limited to) holes, scratches, burns, or missing chunks. This Wefabricate dataset contains 170 labeled images of the product. We have already split it into a training set (136 images) and a test set (34 images). An example of defect product is illustrated in Figure 1.

In this assignment, your task will be to design a **Convolutional Neural Network** to **classify the different images of products**.

1.2 Work structure

To solve the proposed task, you will have to follow the steps below:

- Define a CNN to classify product images in the Wefabricate dataset, train the CNN model on the training set and test it on the test set;

- Choose five hyperparameters and use two hyperparameter optimization algorithms to tune the model. Do 5-fold cross-validation on the training set to calculate the validation accuracy for the hyperparameters. You are free to choose which five hyperparameters to tune, but the hyperparameter optimization algorithms have to be 1) random search and 2) a more sophisticated method than random search;

- Compare the results obtained before and after automatic hyperparameter optimization in terms of accuracy.

1.3 Submission details

The project must be implemented using Python 3.7+ and PyTorch 1.10.0+ (Pytorch Lightning is allowed if needed). For the hyperparameter optimization, you can use either HyperOpt[1] or Optuna[2]. The submission folder of the project should contain the following files:

- Python file, containing:
  - Code for defining, training and testing the CNN. (5 points);
  - Code for automatically tuning the CNN hyperparameters (5 points);
- **Trained model weights before hyperparameter tuning**;
- **Trained model weights after hyperparameter tuning**;

[1] `http://hyperopt.github.io/hyperopt/`

[2] `https://optuna.readthedocs.io/en/stable/index.html`

- Report file, containing:
  - Brief description of how CNNs are able to solve the given task (2 points);
  - Results of tests run with the CNN before hyperparameter optimization, including learning curves of training (2 points);
  - Brief description of how the hyperparameter optimization tool you used is able to improve the CNN (2 points);
  - Description of the chosen hyperparameters to be optimized, their search space, and motivation of the choice (2 points);
  - Results of tests run with the CNN after hyperparameter optimization, including learning curves of training and validation and comparison with previous ones (2 points).

  The files described above must be submitted in a zip file. And the maximum length of the report is **6 pages**, excluding a title page and optional references.

2 Tips for getting started

1. To install PyTorch and related packages use pip or conda:
   - pip CPU only:

     ```
     pip3 install torch torchvision
     ```

   - conda CPU only:

     ```
     conda install pytorch torchvision cpuonly -c pytorch
     ```

   - pip GPU support:

     ```
     pip3 install torch torchvision --extra-index-url
     https://download.pytorch.org/whl/cu113
     ```

   - conda GPU support:

     ```
     conda install pytorch torchvision cudatoolkit=11.3 -c pytorch
     ```

     Note that for the GPU support you have to make sure that the cuda version matches the one you have installed.

2. To load the training and test dataset of the Wefabricate dataset, use the *load_dataset()* in the **support.py** file on Canvas.

3. To save and load the model weights, see Pytorch documentation: `https://docs.pytorch.org/tutorials/beginner/basics/saveloadrun_tutorial.html`

A Assignment Rubric

A.1 Part 1 (20 points)

Training and testing code (5 points)	The code is functional and well-documented for training and testing the classifier. **Trained model weights are provided**.
Automatic hyperparameter tuning code (5 points)	The code is functional and well-documented for automatically tuning five different hyperparameters, using random search and a more sophisticated method as optimization algorithms. Trained model weights are provided for each of the two methods.
CNN description (2 points)	The report briefly describes how CNNs work and why they are suitable for solving the given task.
Results + visualization: non-optimized model (2 points)	Results of tests run with the CNN before hyperparameter optimization are provided, including training learning curve and a highlight of the highest accuracy value obtained.
Hyperparameter optimization tool (2 points)	The report describes the chosen hyperparameter optimization algorithms and the reasons why they can improve the CNN model.
Hyperparameters (2 points)	The report provides the details of the chosen hyperparameters to be optimized, their impact on the models' performances, their search spaces and the motivation of the choice.
Results + visualization: optimized model (2 points)	Results of tests run with the CNN after hyperparameter optimization are provided, including learning curves of training and validation and a highlight of the highest accuracy value obtained. The results obtained using the two different optimization algorithms are compared with each other and with those obtained with the non-optimized model.