Machine Learning

Lab 8:

Final Project (2): Industrial Anomaly Detection

Load: 6-10 hours in the lab

80 hours of personal work

Programming language: Python

1. Objective

This lab is the second part of your final project. In the final project, we wanted to analyze images from MILAN Stationary Company's automated imaging system, which are used to inspect the quality of manufactured erasers.

In this lab, you will apply transfer learning, train a multi-class classifier, compute key performance metrics, and finally export your model using ONNX.

2. Environment Set Up and Data Pre-processing

We are going to continue working in the notebook from previous lab. Create new code cells at the end of your notebook for the
Prepare your data for multi-class classification:

label 0: print error, label 1: physical error, label 2: no defect

3. Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is repurposed as the foundation for a second task. This approach is beneficial when the second task is related to the first or when data for the second task is limited. Using learned features from the initial task, the model can adapt more efficiently to the new task, accelerating learning and improving performance. Transfer learning also reduces the risk of overfitting, as the model already incorporates generalizable features useful for the second task [read more]. In this section we are going to load a pretrained model and fine tune its classifier layer for your industrial anomaly detection project.

Most transfer learning workflows follow these steps:

1. Load a pretrained model (with pretrained weights).
2. Freeze the base layers (optional, depending on dataset size).
3. Replace the final classification layer with one matching your number of classes.
4. Train the new model on your dataset.
5. Optionally unfreeze some layers and fine-tune.

Pytorch has some of the most popular deep models prepared in torchvision. However, you can import deep models from other libraries (e.g., Hugging Face) too. Here you can check the list of available models and how to import them (with or without pretrained weights). You should choose your model based on the its number of parameters, architecture and your available resources.
Remember that each model can have multiple different variations. For example, ResNet variations are ResNet-18, ResNet-34, ResNet-50, ResNet101, ResNet152 which are different in number of layers and parameters.

In the next part of this section you will finetune the model that you have selected. Also in this tutorial you will find a sample finetuning code for a ConvNet model.

After choosing the model, we will load it and replace its classifier layer with our own classifier:

- In your code, import models from torchvision
- Build your model and load the pretrained weights
- Print your model and check the layers. Scroll down to find the name of the classifier layer
- Store the input features of the classifier layer in a new variable named *in_features*
- Freeze all model layers to keep the pretrained weights
- redefine the classifier layer of the model using nn.linear with input features as *in_features* and output features as number of classes you want to classify (in your case it will be 3). **Note**: this layer is not frozen anymore because we redefined it and now it has random weights. Remember that we will only train this layer.

- Print your new model and check the classifier layer to make sure it is created correctly

Now we want to use another tool to check the details of your model. Torch summary will show layer by layer information of your model including output shape, number of parameters, size of total trainable layers and model size:

- Install torch summary using: !pip install torch-summary
- from torchsummary import summary
- pass your model and a random tensor with (batchsize, number of channels, image size, image size) as our input size to summary. Example:

```
summary(your_model, input_size= (1,3, 224, 224))
```

After preparing your new model, you need to define new hyperparameters. **Remember:** new models require new hyperparameter tuning.
Hint: start with the same hyperparamters that were used for pretraining the model that you chose. You can find the details of the hyperparamters used for training each model in their paper or their official github repository (or any repository that used that model!). During your tuning, you can unfreeze more layers of your model and retrain them if needed.

Then, train and validate your model and compare your deep model with the model you built in the previous lab and decide which one you want to tune as your final model.

4. Metrics

Since your dataset is highly imbalanced you should calculate your metrics per class (one-versus-all method) to get a better view of how your model works. You can compute the area under the precision and recall curve (AUC-PR) for each class which is the best metric for imbalanced data. Additionally, you can compute F1-score, sensitivity and specificity for each class.

In the final evaluation we will compute AUC-PR and F1-score for your submitted model

5. Exporting Your Model Using ONNX

You must save your final model in ONNX format for submission. This tutorial shows how to save your model properly using this format. Before submitting your model, try loading the saved onnx model and passing a random tensor to make sure it works properly.

6. Deliverables

➡ **Lab 8 assignment (deadline 27-11-2025):** submit your saved (multi-class) model as onnx file and we will make an initial evaluation of your model on the test set and give you the results. At this stage we will not give you any marks for your work, this is just for you to know how far you have come with your model.

➡ **For your final project (deadline 15-12-2025):** you are required to submit your best multi-class model (onnx file) and a written report of no more than six pages summarizing your project work. You do not need to submit your notebook in this final stage.

Final Project evaluation criteria:

- 5 points (out of 10 total mark for the lab) dedicated to final project deliverables
- 1 extra point for top 3 best models on the private test dataset
- 0.5 extra point for top 3 best reports