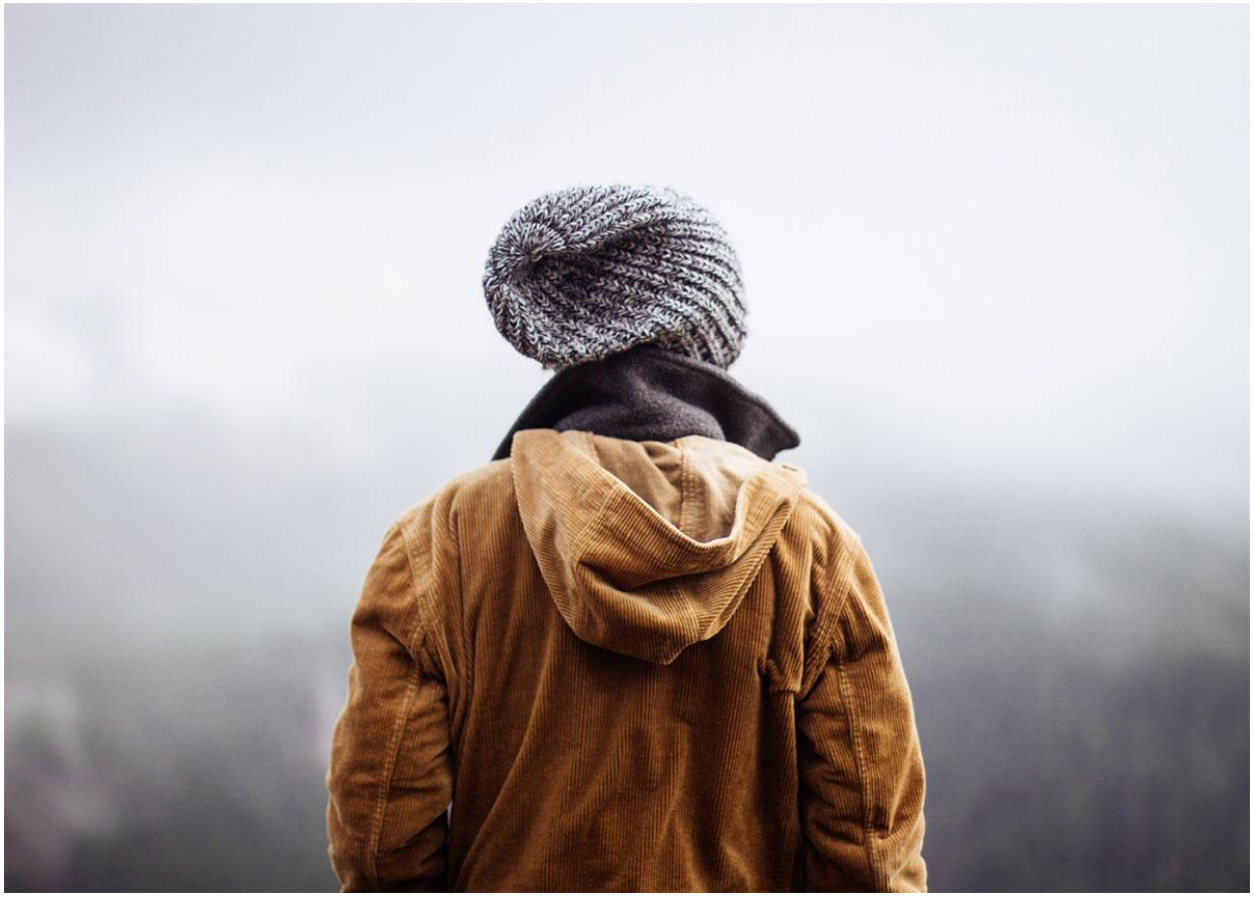
Machine learning

Injection molding machine monitoring problem
Classification



Introduction

The problem is related to the injection molding machine. The injection molding machine is commonly used to produce samples like plastic, toys, containers, water bottles, etc. Considering this specific case, an injection molding machine is used to produce a transparent mold where the goal of this assignment is to build a neural network classifier to monitor the quality of a transparent mold.

Dataset:

The dataset is prepared by extracting the data from the given .mat file. The Scipy python package is used to load the .mat file and a dataframe object is created with the loaded information.

The dataset contains 2952 rows and 49 columns and splitting is done using the torch-provided 'random_split' function following the ratio of 70:30 equivalent to training:testing. To prepare the dataset for training with pytorch after splitting phenomena, data loaders are created using train and test datasets. The train data loader is created following the batch size of 32 and the test data loader is created with the batch size of 1024. After dataset loading and splitting, training is done using the model explained in the next section.

Classifier

The Pytorch framework is used to propose a custom-made classifier in case of an injection molding machine. The classifier is used to predict the quality of the transparent mold: normal, weaving and short-forming. There are 48 input attributes which are used to produce a certain quality of the transparent mold while carrying out the monitoring process. So, there are 48 input attributes and these input columns are used to predict one of the three output classes: normal, weaving and short-forming. The classifier used to make quality predictions is explained briefly in the next section.

Model:

Deep learning technology is used to develop a neural network classifier. Several trials are done to propose the most feasible model for the given problem. The model is developed using 3 layers of architecture:

- The first input layer has 48 nodes as there are 48 input attributes.
- The second layer has 20 nodes to maintain the structure of the architecture
- The last output layer is introduced with 3 nodes as there are 3 classes to predict.

After deciding the model architecture, model training begins in the succeeding section.

Model training

After preparing the architecture, the model training is performed using the following parameters:

- Epochs: 20
- Criterion: cross entropy loss function
- Optimizer: stochastic gradient descent
- Learning rate: 0.01
- Momentum: 0.9

In search of best parameters, several hit and trails are made which are described below statistically:

Learning rate	Epochs	Accuracy	Loss
0.01	20	0.923	0.1187
0.03	20	0.997	0.0011
0.05	20	0.999	0.0006

The above table explains the effect of learning rate on performance of the model. Model learns faster when the learning rate is high. Given 20 epochs in each case, the learning rate is directly proportional to accuracy and inversely proportional to loss.

Mini batches	Epochs	Accuracy	Loss
1	20	0.347	1.46
8	20	0.588	1.16
16	20	0.998	0.0098
32	20	0.999	0.0004

The preceding table explains the effect of mini batches on the performance of the model. The increase in the mini batches increases the accuracy and decreases the loss but the relation is maintained to a certain limit.

Number of nodes	Number of layers	Accuracy	Loss
66	3	0.999	0.022
76	3	0.999	0.003
61	3	0.998	0.011
98	4	0.972	0.014
104	5	0.935	0.020
128	5	0.923	0.022

The above table describes the relation between architecture and model performance. Undoubtedly, performance is mainly dependent on the architecture of the model. After data preparation, model selection is the main issue in solving the deep learning problems. Most of the time, developers have to do several hits and trials to find the best-suited model. Considering the scenario, three-layers architecture is better enough. Complexity of the model is dependent on the format and length of the dataset. Smaller the dataset, the less complex the architecture is. The reader may analyze the relation between the model and the performance from the above table.

Model evaluation

After the completion of the training process, the trained model is tested using the test dataset which is 30 percent of the total dataset. Eventually, the model is evaluated to have the testing accuracy of 0.99 percent which is the best possible accuracy the trained model can have with loss of 0.022.

Tools and technologies

- Programming language: python
- Machine learning framework: pytorch
- Python packages: numpy, pandas, torch, scipy, sklearn