Title:

Improvements to a Machine Learning Machining Feature Recognition System

Authors:
Michael Lenover, michael.lenover@uwaterloo.ca, University of Waterloo
Sanjeev Bedi, sbedi@uwaterloo.ca, University of Waterloo
Stephen Mann, smann@uwaterloo.ca, University of Waterloo
William Melek, william.melek@uwaterloo.ca, University of Waterloo

Keywords:
CNC machining, machine learning, feature recognition

DOI: 10.14733/cadconfP.2024.18-22

Introduction:

For CNC machining, to select a toolpath, it is useful to identify the high-level machining features required to manufacture a part. For example, given knowledge that a set of faces require a pocketing operation to construct, there are empirical and mechanistic models [6] that can make decisions about tool type, tool shape and depth of cut automatically. If machining features can be identified automatically, curve generation algorithms can be implemented with fewer human decisions, reducing manufacturing costs for small- and medium-sized manufacturing companies.

Yeo et al. [9] developed a model that used machine learning to automatically identify machining features in a dataset of synthetically generated CAD files. Once trained, a CAD model could be encoded using the scheme developed by Yeo et al., and provided as an input to a machine learning model, which could identify which machining features, if any, could likely be found at a particular location of a given CAD file.

The approach proposed by Yeo et al. was successful at identifying machining features in synthetic CAD files with an accuracy of 93%. This is an impressive result, and indicated machining features can be accurately identified using a machine learning-based approach. However, the research conducted by Yeo et al. was limited to training and testing on a single synthetically generated dataset. This approach was successful in creating a large number of complex machining features for which to train on, but was limited based what set of features could be described using this algorithm.

To develop a system that is more useful for small- and medium-sized manufacturing businesses, an extension to the work developed by Yeo et al. is presented. To achieve such an improvement, a model was developed and validated using training data extracted from an existing dataset of generic human-created CAD files, to avoid the concerns about training on a synthetically generated training dataset. The machine learning model of Yeo et al. was augmented with a variety of data pre-processing and canonical machine learning techniques, to improve the classification of features.

Next, to estimate the classification performance of the system in a real-world environment, a dataset of real-world CAD files was collected and tagged using the same feature encoding approach. The model trained on the generic dataset was used to classify features from the real-world dataset. To improve the classification accuracy of real-world machining features, a re-training approach was developed that could be easily integrated with a hypothetical machinist workflow. In this way, a scalable machining feature recognition system that can identify machining features in real-world CAD files of machined parts was developed.

Machine Learning

The goal of machine learning is to identify patterns in a set of data to automatically make decisions. In some cases, the pattern recognition can take the form of segmenting the data based on their similarity into groups called *classes*. This approach is known as *unsupervised learning*. In other cases, a designer may wish to specify associations between classes and data points. This approach is known as *supervised learning*. One of the earliest approaches to supervised learning draws inspiration from the human brain. *Neural networks* use simplified models of neurons arranged in a network to learn about a system. One of the simplest approaches is known as a *feed-forward neural network*. During training, information encoded as a vector of values is passed into an input layer of neurons, which signal to subsequent neurons based on the magnitude of the inputs. These signals propagate through the network, though multiple layers of neurons, into one or more output neurons. Each time a signal is passed from one neuron to another the signal is scaled by a connection weight, which can be initialized with some scheme or randomized at the start of training. The output from the network is compared to the tag associated with that training sample, and an error value is computed. The errors are then passed through the network backwards, scaled by the existing network weights, until adjustments to every weight are calculated. This scheme is repeated for each sample in the training data set, referred to as an *epoch*.

When constructing a machine learning system, it is necessary to format the training data so that the data best fits the problem domain and learning technique. For a classification problem, such as selecting which of a set of common objects are in an image, a vector of binary values associated with each possible object can be tagged to each image, with each element in the vector indicating the presence or absence of a particular object. In the case where only one class can be present at a time (such as in the case of many implementations of machining feature recognition systems) a *1-of-n encoding scheme* is used. A 1-of-n encoding scheme takes the structure of a binary label vector, in much the same way as the previous example, but where only exactly one element in the vector is labelled as 1.

Once an encoding scheme has been selected, several approaches can be used to augment the data so that the data better represents the problem domain and can be used to train an effective classifier. If multiple samples are present in the dataset that are identical, the resulting model can place a greater emphasis on correctly identifying those samples, at the expense of distinguishing between subtle differences between similar classes. Instead, researchers often remove duplicate samples from their dataset. Similarly, techniques have been proposed that pull from the domain of genetics to recombine samples of the same classes into new generations of artificial data, based on the distribution of values in the parent training samples [7]. This approach is known as *crossover*.

Overtraining is a description of a machine learning model that, instead of learning the underlying properties of a given domain, learns some properties present in the dataset that are not generalizable. There are two common ways to combat overtraining. First, when calculating the output for a training sample, an algorithm can randomly select some internal neurons to output no signal, typically with a likelihood of around 5 percent. This approach is known as *dropout* [2]. Second, instead of evaluating the performance of a model based on its ability to classify the data it was trained on, researchers typically split data into three groupings: a training set, a validation set, and a test set. The training set is used to train the model. The validation set is used to make changes to the model to improve its performance. Once a model is trained, the test set is used to evaluate its classification performance.

When there exists a risk that small changes in the composition of the dataset may significantly impact the validation accuracy, a more robust testing approach may be selected. *K-fold cross validation* is an approach that involves training and validating the same dataset multiple times, and averaging multiple validation accuracy values to produce a better estimate of the model performance.

An alternative approach for implementing a supervised learning system is a *decision tree*. Unlike neural networks, decision trees are simpler to implement, do not require many training iterations, and once constructed can be easily interpreted by a human. However, as a consequence of their simplicity decision trees are often not as effective in learning subtle differences between classes. Despite this, decision trees can be useful in scenarios where the complexity of a neural network is not required.

Transfer learning describes a process by which a model is trained on a set of data collected under a specific set of conditions and evaluated based on its performance in a wider variety of new conditions [1]. This property has benefits for a machining feature recognition system, since computer aided design (CAD) files used for training are often proprietary and difficult to obtain in large quantity. When a transfer learning system is insufficient to effectively distinguish between classes in a specific domain, additional *incremental learning* can also be applied [5]. Incremental learning refers to training an existing model using domain-specific training data, to augment the classifier for that specific scenario.

Summary of Extensions

This paper extends the work by Yeo et al. [8] to develop a machine learning-based machining feature recognition system that can operate on B-Rep models of real-world machined parts encoded in a STEP file format. We used a supervised learning, feed forward neural network, with a 1-of-n encoding scheme, and tested a variety of machine learning techniques, including crossover, dropout, decision trees, transfer learning, and incremental learning, using 5-fold cross validation to score each test. Our extension presents a technique for classifying machining features in B-rep models of real-world machined parts, without training on a synthetic dataset with particular domain characteristics. Instead, our approach learns using real-world B-Rep model examples, and in so doing is scalable to a variety of manufacturing domains. To investigate the scalability of the system, an exploration of the ability of the classification system to transfer learning between datasets, as well as the rate at which the system can learn when adapting to a new sample domain will also be discussed.

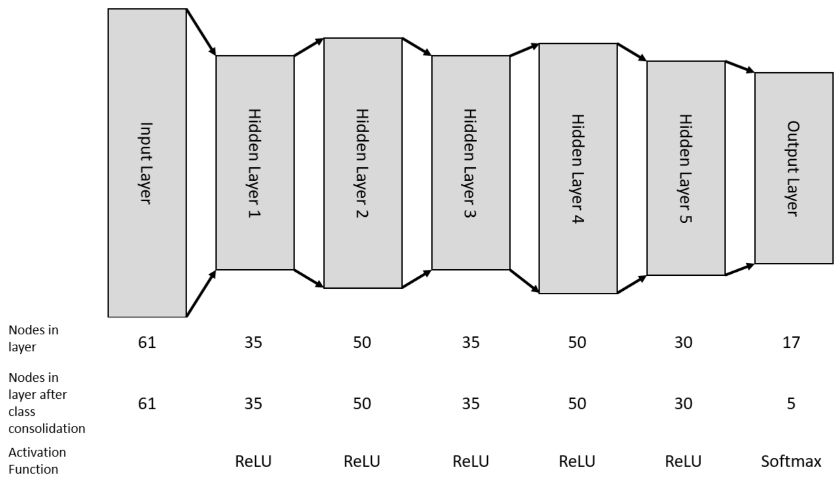
Dataset Creation

Yeo et al. [9] used an automated data generation technique to generate as much training data as possible, to maximize the likelihood that the dataset is fully representative of all possible CAD files and their constituent machining features. Generating the data automatically has the benefit of being able to generate a dramatically large number of parts (in Yeo et al.'s case, 170 000 unique CAD files), but poses a challenge when attempting to transfer the learned features to real-world CAD models.

To build a real-world CAD model dataset, we explored multiple existing CAD model research datasets, and eventually chose the ABC Dataset. The ABC Dataset (A Big CAD Model Dataset) is a collection of 3D models from the online CAD platform OnShape that were collected and published by Koch et al. [3]. The models contained in this dataset are published in a variety of formats including .step, which encodes the necessary parametric geometry information for extracting machining features.

77 CAD files that included CNC machining features were select from the ABC dataset. These files were tagged and encoded with the 17 machining features of Yeo et al. [8], giving 860 machining feature samples in the training dataset. In addition, another 77 files were collected from machine shops at the University of Waterloo, Hurco Inc. and Perfecto Manufacturing Inc. These files were tagged, and a dataset containing 998 real-world machining features was produced. This dataset of machining features extracted from real-world parts remained separate from the 860 features collected from the ABC dataset. In total, 1858 machining features were included in both datasets used in this research.

Once a dataset of CAD files tagged with their respective machining features was created, an encoding program was developed to extract relevant information from the parametric models based on the approach outlined by Yeo et al. In total, a 61-length feature vector is created.



Nodes in layer	61	35	50	35	50	30	17
Nodes in layer after class consolidation	61	35	50	35	50	30	5
Activation Function		ReLU	ReLU	ReLU	ReLU	ReLU	Softmax

Fig. 1: Deep learning network architecture.

Machine Learning Model

A fully connected feed-forward neural network (illustrated in Figure 1) was constructed. The network parameters are selected to match the conditions outlined by Yeo et al. In particular, the number of nodes in each layer are selected to match the number of nodes determined by Yeo et al. to yield the greatest classification accuracy. A 61-deep input layer is connected to 5 hidden layers, which are connected to an output layer.

Some changes were also made to the network hyperparameters outlined by Yeo et al. To minimize the likelihood of overtraining, the training batch size was reduced from 8 to 1. The system was trained with 0% dropout, as originally selected by Yeo et al., and compared with models trained with 10% and 20% dropout, to determine if dropout can be effective in reducing overtraining. Some machining features (such as tapered holes) had limited representation in the training dataset generated from CAD files in the ABC dataset, and as such would be inadequately classified by the learning system. To evaluate the relative performance of several potential model improvements, the 17 machining features proposed by Yeo et al. were consolidated into 5 generic machining features. To accommodate this change in the number of classes, the output layer depth of the machine learning network was reduced from 17 to 5.

Results and Conclusions

Three extensions to the system developed by Yeo et al. were evaluated in this work: the incorporation of dropout, the introduction of an ID3 tree pre-classification step, and the inclusion of additional training data using crossover data generation. Dropout was determined to improve the consistency of feature classification. Yeo et al. determined there was no benefit of incorporating dropout when training their model on synthetically generated machining feature data. In contrast, this work found evidence that the consistency of classification accuracy during training improved when 10% dropout was incorporated in models trained on real-world data. In addition, incorporating an ID3 tree pre-classification step before training a machine learning classifier was effective at reducing model training time, without reducing classification accuracy. Crossover data generation was deemed to not have any significant benefits for model classification accuracy or scalability.

The augmented feature recognition model was trained on generic CAD files, and used to classify machining features from real-world CAD files. Without any additional training, the augmented classifier was unable to identify machining features consistently. The classifier was re-trained using machining features collected from real-world CAD files. The re-trained classifier was significantly more effective at identifying machining features.

Re-training existing machining feature recognition models has significant future potential. A simple machining feature recognition model can be developed using a limited dataset of machining features, and re-trained based on the actions of a machinist selecting features in a CAM program. Collecting data in this way has several benefits. First, the concerns associated with training a model on CAD files with intellectual property protections can be mitigated by re-training the feature recognition system locally. Second, re-training can be done on-the-fly, without interrupting the work of a machinist. Finally, re-training can be used to tailor a feature recognition model to a specific workflow or industry, incorporating automatic feature detection into an existing workflow only when the historical classification accuracy for a particular machining feature reaches a threshold determined by the user.

Acknowledgement:
This research is a funded in part by NSERC.

Sanjeev Bedi, https://orcid.org/0000-0002-6993-8502
Stephen Mann, https://orcid.org/000-0001-8528-2921

References:

[1] Bozinovski, S.: Reminder of the First Paper on Transfer Learning in Neural Networks, 1976. Informatica, 44, 2020. http://doi.org/10.31449/inf.v44i3.2828.

[2] Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors, 2012. http://doi.org/10.48550/arXiv.1207.0580. ArXiv:1207.0580 [cs].

[3] Koch, S.; Matveev, A.; Jiang, Z.; Williams, F.; Artemov, A.; Burnaev, E.; Alexa, M.; Zorin, D.; Panozzo, D.: ABC: A Big CAD Model Dataset For Geometric Deep Learning, 2019. http://doi.org/10.48550/arXiv.1812.06216. ArXiv:1812.06216 [cs].

[4] Lenover, M.: Development of a Scalable Machining Feature Recognition System. Master's thesis, University of Waterloo, 2023.

[5] Luo, Y.; Yin, L.; Bai, W.; Mao, K.: An Appraisal of Incremental Learning Methods. Entropy, 22(11), 1190, 2020. ISSN 1099-4300. http://doi.org/10.3390/e22111190.

[6] Stephenson, D.A.; Agapiou, J.S.: Metal Cutting Theory and Practice. CRC Press, 2016. ISBN 978-1-4665-8754-0. Google-Books-ID: 77n1CwAAQBAJ.

[7] Vafaie, H.; De Jong, K.: Genetic algorithms as a tool for restructuring feature space representations. In Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence, 8–11, 1995. http://doi.org/10.1109/TAI.1995.479372. ISSN: 1082-3409.

[8] Yeo, C.; Cheon, S.; Mun, D.: Manufacturability evaluation of parts using descriptor-based machining feature recognition. International Journal of Computer Integrated Manufacturing, 34(11), 1196–1222, 2021. ISSN 0951-192X, 1362-3052. http://doi.org/10.1080/0951192X.2021.1963483.

[9] Yeo, C.; Kim, B.C.; Cheon, S.; Lee, J.; Mun, D.: Machining feature recognition based on deep neural networks to support tight integration with 3D CAD systems. Scientific Reports, 11(1), 22147, 2021. ISSN 2045-2322. http://doi.org/10.1038/s41598-021-01313-3.