www.cad-journal.net
Computer-Aided Design AND APPLICATIONS™

Improvements to a Machine Learning Machining Feature Recognition System

Michael Lenover[1] , Sanjeev Bedi[2] , Stephen Mann[3] , William Melek[4]

[1]University of Waterloo, mlenover@uwaterloo.ca
[2]University of Waterloo, sbedi@uwaterloo.ca

[3]University of Waterloo, smann@uwaterloo.ca
[4]University of Waterloo, william.melek@uwaterloo.ca
Corresponding author: Stephen Mann, smann@uwaterloo.ca

Abstract. Pre-processing and training techniques were applied to improve the performance of a model trained with an existing machining feature recognition approach by Yeo et al. [20] using a smaller dataset that more effectively mimics the complexity of CAD models used in industry.

Three improvements to the feature recognition algorithm developed by Yeo et al. were explored: the incorporation of dropout to improve model stability and accuracy, the incorporation of ID3 tree pre-classification to reduce training time by reducing the size of the deep learning dataset, and the incorporation of crossover data generation to improve classification accuracy by reducing over-fitting due to insufficient training data. Incorporating dropout improved stability and improved 5-fold cross validation accuracy. Further, incorporating a 2-deep ID3 decision tree pre-classification only marginally improved classification performance but was effective in reducing the size of deep learning training dataset. Crossover data generation did not improve model performance. Using the model trained on the generic CAD dataset, and incorporating 10% dropout and a 2-deep ID3 tree, models from the real-world dataset were classified. This classifier was effective in classifying some simple features, but had poor accuracy overall. To improve this accuracy, an incremental learning technique was applied. The generic model was re-trained using samples from the real-world dataset, which improved the classification accuracy of the system.

Keywords: CNC Machining, Machine Learning, Feature Recognition
DOI: https://doi.org/10.14733/cadaps.2025.119-135

1 INTRODUCTION

To increase manufacturing efficiency, it is often necessary to automate parts of the manufacturing process. CNC, or computer numerical control, has been in development since the 1950s as a method of automatically

executing movements of a robotic system. In the domain of manufacturing, CNC control has been used to automate subtractive machining processes such as lathing or milling. Often shortened to CNC machines, this equipment can automatically move a cutting tool into stock to remove material to produce a desired part, in much the same way as a human operator might do.

In a typical industry workflow, a technician or operator would begin by importing a computer aided design (CAD) model into a toolpath planning program. Next, they would make decisions based on their own experience, the machine tools available to them, the required manufacturing speed, the specified part tolerances, and a variety of other factors to determine the best approach for manufacturing a particular aspect of the part. Until recently, making these decisions about which approach to take (often dozens of times for a single part) was the responsibility of an experienced human operator.

Relying on a human operator has several limitations. CNC machines remain useful for producing thousands of identical parts, as the labour cost of generating the toolpath can be amortized over the many times that toolpath is executed. However, for one-off or low-volume production runs, the cost to generate the toolpath represents a significant portion of the final manufacturing cost. This cost has the effect of making CNC technology inaccessible to many smaller businesses.

Relying on a human technician to plan the toolpath also limits the volume of orders a single manufacturing company can accept. By reducing the work required from a human operator, more manufacturing orders can be processed, and the reach of a single manufacturing company can be increased.

There has been a large body of research in the domain of toolpath planning automation over the past four decades. One avenue of research is in machining feature recognition. 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 [14] 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.

Heuristic methods are a reasonable approach to identifying features [5, 17, 18]. However, if a region in a part should be resolved with a given machining feature, but an explicit rule was not included by a designer, a heuristic model will be unable to classify that region of the part. In contrast, feature recognition systems that leverage machine learning are able to learn new associations between model geometry and machining features, beyond those which can practically be associated by a designer.

Yeo et al. [20] 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 outlined above which arise from training on a synthetically generated training dataset. The machine learning model was based on the approach developed by Yeo et al. The learning algorithm 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

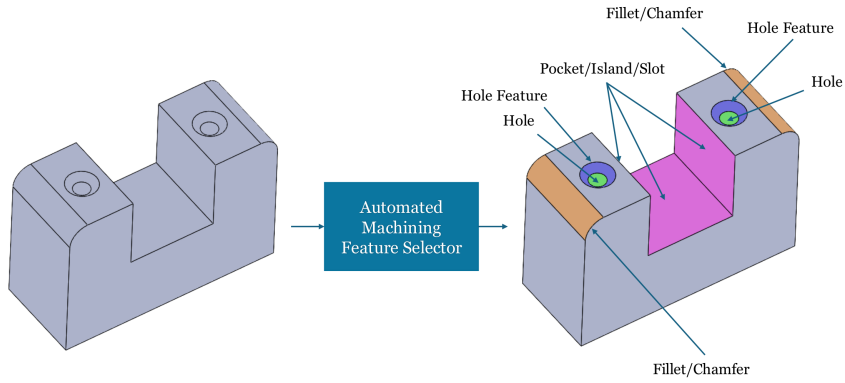


Figure 1: Example part with features labeled.

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.

Figure 1 illustrates our method. On the left is a part. The goal of our work is have software to recognize and located the features shown on the right.

2 BACKGROUND

The domain of machining feature recognition has been widely researched from several different perspectives for over four decades. Some of the earliest work into machining feature recognition was detailed by Kuprianou [7]. Since then, several approaches have been investigated that refine existing techniques to improve the quality of classification, including hint-based methods [2], graph-based methods [5], volume-based methods [15], and cell-based methods [12, 13]. In this paper, we will be looking at a machine learning method for feature recognition.

Machine learning is a popular classification technique in a variety of domains that has become increasingly popular in recent years. Early investigations into the application of machine learning to the machining feature recognition problem encoded information extracted from the B-Rep model as a matrix of feature vectors, with each face in a part encoded as a single vector [11]. This approach of encoding geometry information into a fixed-length vector allows these vectors to be treated as training samples in a feed-forward neural network.

Recently Yeo et al. presented a neural network-based machining feature recognition domain [20]. Their paper presents a proposal for an end-to-end pipeline that can extract relevant information from a CAD file and present an estimate of the machining features contained in the part. This work draws heavily from their previous publication [19], that outlines what information is best to encode in the data vector to distinguish between different machining features most effectively.

Yeo et al. identified attributes of a CAD file that could be used to uniquely identify machining features from a list of machining features they selected. These attributes were used to generate a feature vector: a list of integers encoding relevant information from the CAD file. These feature vectors were tagged with a corresponding machining features using an automated process. After several thousand features vectors were encoded and tagged with their respective machining feature, they were used to train a machine learning system to identify machining features in parts that that system had not seen before. In total, they used 2236 data

samples to train, test and validate the network, which was able to distinguish between 17 different machining features with 93% accuracy.

However, the data Yeo et al. used to train and test their model was synthetic data. While using synthetic data allows for the generation of large data sets, the performance of the resulting model on real data is typically worse than the synthetic test data. By training on real world data, a model more suitable for recognizing features in real world models can be achieved.

This paper proposes a machining feature recognition system designed to identify machining features in real-world CAD files. We created a CAD data set from real-world CAD files (Section 4), and we augmented Yeo et al.'s method using a variety of machine learning techniques (Section 5) to improve the model's classification accuracy when evaluated against real-world CAD files (Section 6). Our extensions use a variety of machine-learning techniques, which we review in the next section.

3 MACHINE LEARNING

The following is an overview of the machine learning techniques we applied in this paper. For additional details on these methods, the reader is referred to the textbook of Fieguth [3].

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 itself. *Neural networks* describe an approach that uses many 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*. To train a network to classify a system, it is often necessary to iterate over the same data set multiple times, for hundreds or thousands of epochs. In Section 5, several feed forward networks are developed to classify machining features.

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 (plants, animals, humans, cars, etc.) 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. The machine learning model constructed by Yeo et al., and our model implemented in Section 5 is trained using a 1-of-n class encoding scheme.

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 [16]. This approach is known as *crossover*. An extension to the work developed by Yeo et al. is proposed in Section 5 that incorporates crossover to generate new training data.

Overtraining is a description of a machine learning model which, instead of learning the underlying properties of a given domain (for example, the characteristic color or shape of a tumor in an image), learns some properties present in the dataset that are not generalizable (for example, a note from a doctor or a watermark present on all images that contain a tumor). 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. In this way, simple strategies that rely on the activation of only a handful of neurons (and as a result, represent a model with little complexity) are not as successful. This approach is known as *dropout* [4]. 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, in a manner as discussed before. The validation set is used to make changes to the model (such as changing the rate at which the neuron weights are updated each iteration, and the network structure) to improve its performance. Once a model is trained, the test set is used to evaluate its classification performance. The tests conducted in Section 6.1 and Section 6.2 incorporate training, test and validation datasets in this way to avoid overtraining.

Typically, the accepted approach for developing a deep learning classifiers involves training a system on a training data set multiple times, varying some aspect of the system that is to be evaluated. During training, the accuracy of the system is quantified by using the model to classify samples in the validation set, and comparing the model outputs to the ground truth labels associated with the validation training data. Then, once an optimal system has been developed, a final estimate of the system's accuracy is developed by classifying samples in the test data set, and calculating the proportion of samples which are classified correctly. This approach works well for large datasets which well represent the feature space of the classification problem. However, 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. To collect multiple validation accuracy values, the test and validation datasets are combined into a single training dataset, This single training dataset is split into (typically around 3-5) folds, groups of training samples with an equal or approximately equal number of samples. A model is trained using data from all but one of the folds, and the held-out fold is used to calculate the validation accuracy of the trial as before. This process is repeated, holding out a different fold each time. These validation accuracy estimates are then averaged to produce a cross validation accuracy of the model. The tests conducted in Section 6.1 and Section 6.2 incorporate K-fold cross validation, to more effectively estimate the classification performance of each model.

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. Another extension to the work by Yeo et al. is proposed in Section 5 that incorporates a decision tree to classify machining features.

Other machine learning techniques refined in the last few of years have focused on developing new approaches to improve the accuracy of existing models. Incremental learning and transfer learning are concepts that have been embraced by machine learning researchers as ways to integrate learning models into real-world classification problems without a robust existing dataset. *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 perfor-

mance 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. The ability of a machining feature recognition model to transfer learning between different machining feature datasets is evaluated in Section 6.3. When a transfer learning system is insufficient to effectively distinguish between classes in a specific domain, additional *incremental learning* can also be applied [10]. Incremental learning refers to training an existing model (which can be the model trained on data collected under a specific set of conditions, as described above) using domain-specific training data, to augment the classifier for that specific scenario. A proposed method of incrementally learning machining features is explored in Section 6.4.

3.1 Summary of Extensions

This paper extends the work by Yeo et al. [19] to develop a machine learning-based machining feature recognition system that can operate on B-Rep models of machined parts encoded in a STEP file format. Our extension improves the training time and/or accuracy of the system, and develops a technique that is scalable in 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 conducted.

4 MODEL CREATION

As compared to other steps in a typical CNC workflow, there exists a great divide between the typical level of automation for solutions to the machining operation selection problem and the most common solutions used to accomplish this task in industry. To bridge the gap between existing machining feature recognition research and industry needs, this paper investigates the effectiveness of an existing machining feature recognition when classifying data collected in the real world. This work will begin with collecting two datasets: a dataset of generic CAD files, which will be used to train a generic machining feature classifier, and a dataset of CAD files collected from real-world machine shops. These datasets will be tagged and encoded using the approach developed by Yeo et al. [19] to create dataset of feature vectors that can be used to train a deep learning system.

4.1 Dataset Creation

It is difficult to determine if a dataset size is sufficient to train and validate a learning approach. To address the issue of data sufficiency, Yeo et al. [20] 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. This approach involved generating part parameters (feature type, location, size, etc.) algorithmically, using those parameters to automatically generate 3D models in a parametric CAD program, and tagging those 3D models using their feature descriptor approach. 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 identify machining features in real-world CAD files, a system must be trained on files used in the real world.

We explored multiple CAD model research datasets, but 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 [6]. 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. Although the IP rights of the CAD files are not provided to researchers, the dataset can be used for machine learning applications, provided the files are not published. For both of these reasons, this dataset was selected. CAD files from the ABC

Dataset were downloaded, of which 600 were incorporated into an initial generic dataset. Figure 1, left, shows what a part in this dataset looks like.

4.2 Data Collection

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. [19], 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.

4.3 Feature Encoding

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. Information about the encoding scheme is given in Figure 2. Each machining feature is assigned a unique integer value. The machining feature class, along with the 61-element feature vector, is saved to a .csv file.

5 MACHINE LEARNING MODEL

It is expected that the performance of a system trained and tested on real-world CAD files will be less effective at distinguishing between different machining features than a similar system trained and tested on a heavily curated synthetic dataset. A real-world machining feature dataset will contain more examples of machining features that occur more frequently in machined parts. Real world datasets often includes fillets, chamfers, and holes more frequently than specific pocketing or slotting operations. When a deep learning system is trained on an unequal quantity of samples from different classes, there is a risk that instead of generalizing, the system will simply select the most common class (since that estimate is likely to be correct, by the nature of the dataset distribution).

Even if the distribution of machining features in the real-world dataset could be controlled, other concerns with class imbalance must still be addressed. Certain machining features such as holes, when encoded, are represented by a small set of unique feature vectors. In contrast, other features such as slots or pockets have much more variance, and so are represented by a greater number of unique feature vectors. When duplicate samples are removed prior to training, the machining features with more variance will have fewer duplicate samples, and so will end up over-represented in the final dataset. To address concerns about unequal distribution of machining features in the CAD file dataset, as well as the resulting imbalance resulting from removing duplicate entries, data pre-processing techniques can be used to improve the quality of the dataset. Two techniques were evaluated.

Decision trees are a classical pattern recognition technique used in addition to machine learning for distinguishing between classes. A large class imbalance in the unmodified dataset may contribute to overtraining, especially when one or two classes represent a large fraction of the total dataset. A deep learning system should perform better and be able to identify less common classes if a decision tree is used to identify the most common machining features first. By taking this approach, classes with lower variance features and significantly greater representation in the dataset can be identified with a simpler recognition system, leaving the more computationally expensive deep learning system to distinguish between the remaining classes.

Another technique used in machine learning that can be applied to this problem of class imbalance is a genetic algorithm. Genetic algorithms are not typically used to identify patterns directly, but instead are used

Feature vector element	Feature vector indices	Example value(s)	Example encodings
Base face type	1	Planar / cylindrical / toroidal	6 / 5 / 8
Curvature of base face	2	Flat / positive / negative	0 / 1 / 2
Width of base face for face-machining	3	Longer than 5 cm / shorter	1 / 2
Width of base face for edge-machining	4	Longer than 2 cm / shorter	1 / 2
Adjacent unknown / concave / convex faces in an outer loop	5-15 16-26 27-37	Proportion of outer loop adjacent faces, rounded up to the nearest 10%: • None with unknown convexity • 40% cylindrical concave • 50% planar concave • 20% planar convex	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 5 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0
Adjacent faces with C0 (non-tangent) continuity in an outer loop	38-48	Proportion of outer loop adjacent faces, rounded up to the nearest 10%: • 40% cylindrical • 50% planar • 20% faces with continuity other than C0	0 0 0 0 0 4 5 0 0 0 0
Perpendicular adjacent faces in an outer loop	49-59	Proportion of outer loop adjacent faces, rounded up to the nearest 10%: • 40% cylindrical • 50% planar • 20% faces with continuity other than perpendicular	0 0 0 0 0 4 5 0 0 0 0
Location and convexity of inner loops	60, 61	Is the inner loop not centered on the base face, and the angle between the base face and inner loop faces concave? Yes	1
		Is the inner loop centered, and the angle between the base face and inner loop faces convex? No	0

Figure 2: Feature vector element encodings, as defined by Yeo et al. [19]

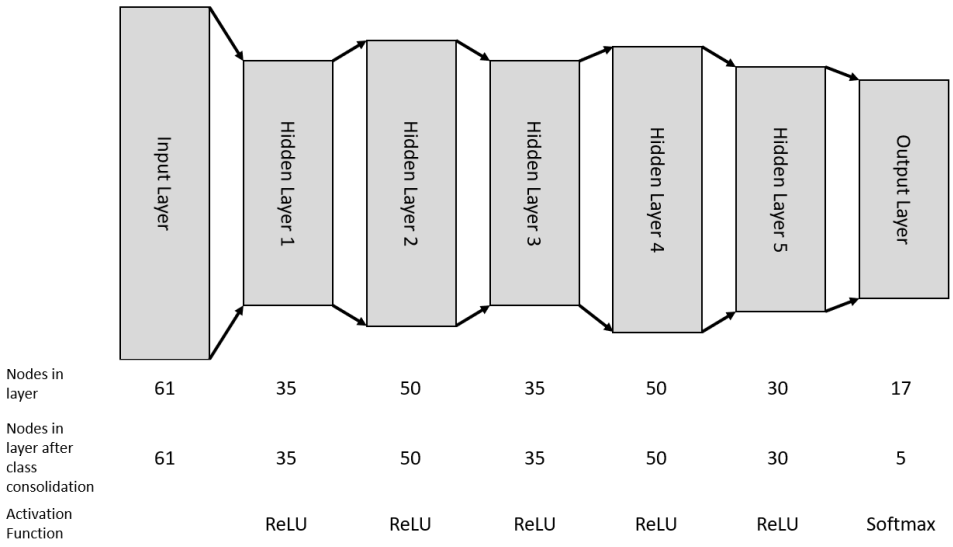


Figure 3: Deep learning network architecture

to optimize existing systems. However, one step in the genetic algorithm procedure, crossover, can be modified to augment an imbalanced dataset.

In genetic algorithms, crossover is used to randomly recombine parent individuals in a population to create children samples with properties of each parent. In the context of reducing class imbalance, crossover will be used to generate synthetic training data using the existing training data for each class, until all classes have an equal number of training samples. Since the complexity of the generated data will be proportional to the complexity of the existing ground-truth data for each class, this method will not be as effective in addressing issues with class imbalance once duplicate samples are removed. However, this approach might mitigate problems arising from the underlying feature imbalance in the CAD file dataset. As such, crossover data generation was also implemented, and its impact on classification accuracy evaluated.

The pre-processing data pipeline is constructed as follows. First, an ID3 decision tree is trained on the available training data. The decision tree classifies features up to a certain depth, which is specified in the test plan. Next, crossover data generation is applied to each feature class except for the class with the greatest number of samples. The data generation approach discussed previously is applied until the number of training elements in each class is equal. Next, duplicate samples in each class are removed. Finally, the remaining training samples are used to train a deep learning classifier.

A fully connected feed-forward neural network (illustrated in Figure 3) 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. The output layer encodes the class estimates with one-hot encoding. Each hidden layer has a ReLU activation function. The output layer weights are determined by a Softmax activation function, to estimate the probability of each potential classes given the encoded input. The standard Adam optimizer algorithm was used to update the model weights.

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,

None Feature	All other faces that are not a feature base face						
Hole	Simple Hole						
Pocket/ Island/Slot	Closed Pocket	Opened Pocket	Closed Slot	Opened Slot	Floorless Slot	Closed Island	Opened Island
Secondary Hole Feature	Counter-bore	Counter-drilled	Taper				
Fillet/ Chamfer	Inner Fillet	Outer Fillet	Inner Chamfer	Outer Chamfer			

Figure 4: Consolidated machining features and their corresponding machining features as defined by Yeo et al.

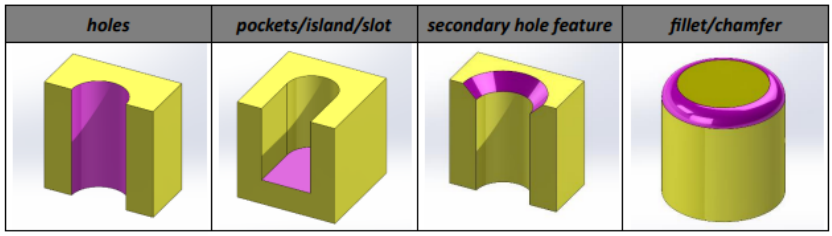


Figure 5: Samples of each of four features.

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. This consolidation was done primarily because there were few cases of 12 of Yeo et al.'s features in our data set, reducing the effectiveness of the machine learning algorithm is recognizing the underrepresented features. The consolidated machining features used in the remainder of this research are presented in Figure 4, with samples of the four features shown in Figure 5, and a sample of a part with these features labeled shown in Figure 1, right. To accommodate this change in the number of classes, the output layer depth of the machine learning network was reduced from 17 to 5.

In all models presented in the remainder of this paper, a learning rate of $1 \cdot 10^{-4}$ was selected.

6 EXPERIMENTS AND RESULTS

Several experiments were conducted to evaluate the effectiveness of proposed extensions to the machining feature recognition model developed by Yeo et al. Machining feature recognition models were trained on data collected from the ABC dataset. These models were augmented with canonical machine learning techniques

and pre-processing steps in an attempt to improve classification accuracy and model consistency. Next, once the most effective model improvements were identified, the ability to transfer knowledge of machining features collected from the ABC dataset to a dataset of machining features collected from real-world CAD files was evaluated. Finally, to improve cross-domain classification accuracy, an exploration of a proposed re-training approach was conducted.

Our interest is less in the details of the internal evaluations of the machine learning techniques and more on the success (or failure) of the variations in feature matching. As such, in this paper, we do not give the typical machine learning convergence graphs[1], but instead rely on *confusion matrices* to evaluate the results. Our confusion matrices have one row and one column for each feature we have tagged/wish to detect; the entry at row A and column B is the number of features of type A that were detected as a feature of type B. Thus, the closer the confusion matrix is to a diagonal matrix, the better. Further, entries off the diagonal indicate when feature A is being mis-detected as feature B, which at times can be helpful in analyzing a problem with a method.

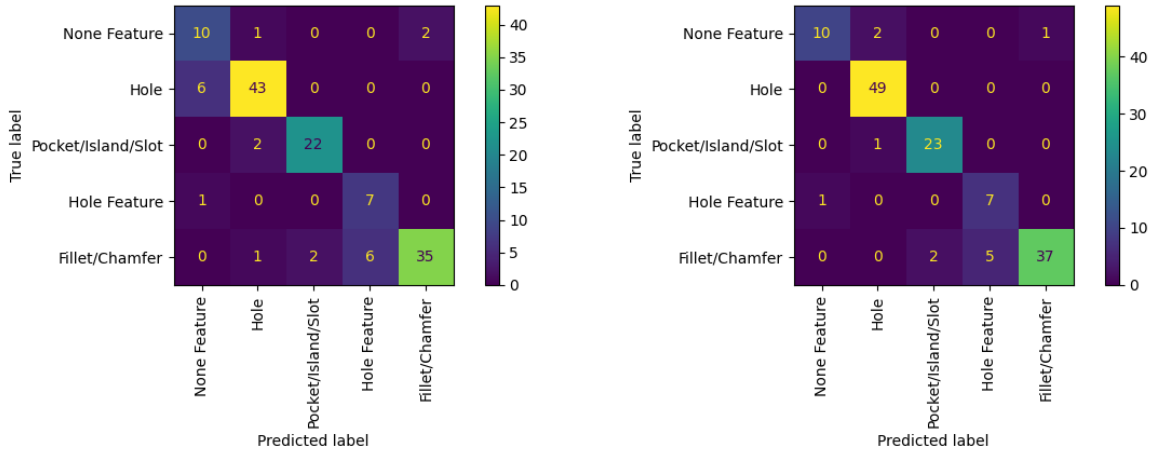
6.1 Evaluation of Dropout

A model was trained on ABC dataset samples with 0%, 10% and 20% dropout. The model trained with 0% dropout is essentially Yeo's model, but trained on the ABC dataset. For each likelihood of dropout, 5-fold cross validation was used to estimate the model performance. A confusion matrix was constructed using the corresponding validation data for each fold at the end of 1000 epochs to compare the estimated and corresponding ground truth class for each sample. The validation accuracy and total cross entropy loss was calculated at the end of each epoch, and recorded.

In our test, the validation accuracy across folds for a single test condition, and to a lesser extent across every test condition, exhibit the same general behaviour. This is also true of the cross entropy training loss, although the behaviour is opposite that of the validation accuracy. In every test scenario, the validation accuracy increased and the cross entropy loss decreased as the model was trained. For a typical result during training of single fold, in the first 200-300 epochs, model accuracy increased dramatically. For the same 200-300 epochs, cross entropy loss fell. For the remainder of training, model improvement slowed. After 1000 epochs, the model performance was visualized with a confusion matrix. Once again, the trends observed in the confusion matrix figures are similar across folds for a single test condition, and to a lesser extent across every test condition. Typical confusion matrices produced after training are shown in Figure 6. In about 80-90% of cases, the predicted label in the validation fold matched the ground-truth value for that sample. This is demonstrated by a prominent clustering of samples along the diagonal of the confusion matrix figures from the top left of the figure to the bottom right. Samples along this diagonal represent instances where a feature in the validation fold was identified correctly. Although there were many similarities across training instances, there were a few notable differences across different test conditions and folds.

Based on the data collected from the above tests, it was determined that incorporating 10% dropout in each hidden layer improved validation accuracy for a model trained on samples from the ABC dataset by 0.87% on average, as compared to a model trained without dropout. A model trained without dropout (i.e., Yeo et al.'s model) classified data from the held-out validation fold correctly 85.90% of the time, which improved to 86.77% when trained with 10% dropout. Increasing dropout to 20% resulted in a decrease in validation accuracy, with an average correct classification rate of 85.61%. The difference in average classification accuracy for different amounts of dropout, when considered alone, is small enough to be considered negligible. However, another qualitative observation also motivated the inclusion of dropout in future deep learning machining feature recognition systems. Incorporating dropout significantly increased the classification stability of the model. Although the models trained with dropout still contain some instability, based on qualitative observations the frequency and magnitude of variation decreased.

[1]The interested reader will find the standard machine learning convergence graphs for our method in [9].



(a) (b)

Figure 6: Confusion matrix for a typical fold after training. Fold 3, (a) No dropout; (b) 10% dropout.

By reducing the variation of the validation accuracy, one can have more confidence that an unknown feature will be identified correctly more consistently. A model with a large amount of validation accuracy instability indicates the model may have issues generalizing classification of certain classes. For this reason, as well as for the slightly improved classification accuracy, 10% dropout was incorporated into subsequent models presented in this paper.

6.2 Evaluation of ID3 Tree Pre-classification and Crossover Data Generation

ID3 tree pre-classification and crossover data generation were used to augment models trained on data tagged from the ABC dataset. The respective validation accuracy for each model was compared. Each model incorporated 10% dropout. Six models were trained, with every combination of the following techniques: no ID3 tree, 1-deep ID3 tree and 2-deep ID3 tree pre-classification; and with and without crossover data generation.

As before, 5-fold cross validation was used to create an unbiased estimation of the validation accuracy of the models trained with each combination of techniques. Incorporating a 2-deep ID3 tree pre-classification step improved average classification accuracy by a negligible amount, increasing from 86.77% validation accuracy without any data augmentation to 86.68% validation accuracy with a 2-deep ID3 tree. Incorporating crossover data generation did not improve validation accuracy.

Although ID3 tree pre-classification did not significantly improve the rate of feature classification, it also did not significantly reduce classification accuracy. Moreover, by incorporating ID3 tree pre-classification, the amount of data required to train a neural network with a given classification accuracy was significantly reduced as compared to an equivalent model without pre-classification. By incorporating a 2-deep ID3 tree pre-classification step, the average number of training samples for the deep neural network step was reduced from 206.6 after duplicate removal to only 95.2, without loss of classification accuracy. The number of training samples required to train a neural network is reduced by selecting for common features without significant complexity and identifying them using an ID3 tree. The training samples associated with those features are then omitted from the deep neural network training dataset. By reducing the number of training samples

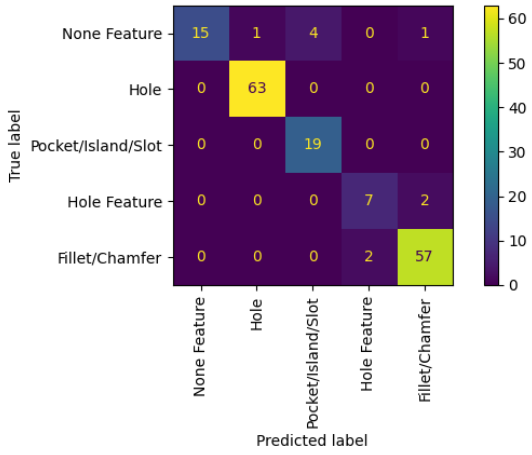


Figure 7: Confusion matrix of model trained on full ABC training dataset with 10% dropout and 2-deep ID3 tree

necessary to achieve a given classification accuracy, the time required to train the model is reduced, since fewer features must be analysed for each epoch.

Crossover data generation increased the number of samples used to train the deep neural network without improving the classification accuracy. By increasing the number of samples used to train the deep neural network, the time to train the model was increased without any benefit. The selected approach of recombining feature vector elements based on the distribution of those elements in the training data associated with a single feature was ineffective in improving the generalization of the system. Although crossover data generation may be effective for other classification problems, the increased noise associated with generating synthetic data outweighed any classification accuracy benefit associated with increasing the size of the training dataset.

Once the most effective combination of model improvements was selected, a single model was trained using all training samples collected from the ABC dataset (i.e., training data from all 5 folds). The confusion matrix resulting from this training is presented in Figure 7 (compare to Figure 6(a), which is Yeo's model trained on the same data). This model was evaluated against a dataset of held-out test data, which represented the remaining 20% of samples collected from the ABC dataset. This model, which was trained for 1000 epochs with 10% dropout and a 2-deep ID3 tree pre-classification step, was able to classify the held-out test data correctly 94% of the time, an improvement over the 85.9% achieved by Yeo's model noted in Section 6.1.

6.3 Evaluation of Transfer Learning

It was demonstrated by Yeo et al. that their model was effective at identifying features, even without the proposed improvements developed in the previous sections. However, Yeo et al. used the same algorithm to generate synthetic training data to train their model as they used to generate the test data that was used to evaluate their model performance. As a consequence, the ability of the system to adapt to new dataset domains, including different model encoding schemes or feature distributions, was left unexplored. This section evaluates using transfer learning between dataset domains. A model was trained on all features encoded from ABC dataset models. This model was trained using 10% dropout and a 2-deep ID3 tree pre-classification step. After training for 1000 epochs, the model was evaluated against 998 features encoded from real-world

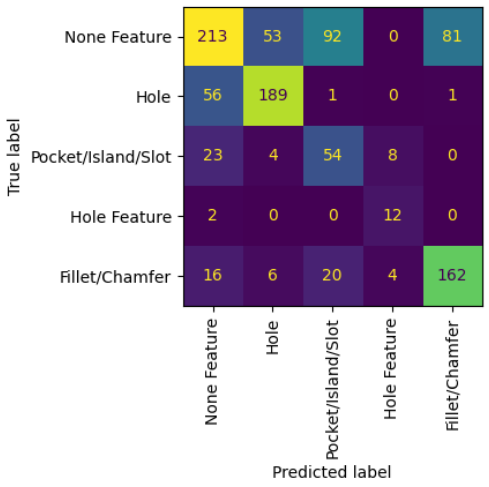


Figure 8: Confusion matrix of model trained on samples from ABC dataset, evaluated against real-world data

machined parts. A confusion matrix of the model classification performance after 1000 epochs is presented in Figure 8.

From Figure 8, it can be seen that no individual machining feature was identified correctly more than 70% of the time, indicating this model would not be effective as general-purpose machining feature classifier, even for simple features such as holes. This low classification performance is likely due in part to the limited training dataset that was available, resulting in a model with limited ability to generalize features. However, the test dataset may also simply contain a different distribution of features, with a slightly different set of valid encodings for each feature. Thus, the model may be able to be improved by retraining the general model trained on data from the ABC dataset using features from the real-world dataset.

6.4 Evaluation of Incremental Learning

To improve the performance of the Yeo's model, an incremental learning approach was developed. The real-world dataset was segmented into datasets of training and test data with a ratio of 80:20. The training dataset was further segmented into clusters of 20 machining features. This number was selected as it approximates the number of machining features that are contained in a single CAD file. The base model trained on the machining features extracted from the ABC dataset was retrained, 20 features at a time, for 20 epochs each. This retraining was completed in the same manner as the initial training method, with 10% dropout. At the end of each training cluster, the test data was used to calculate the classification accuracy of the model. After training against 39 clusters of 20 data points, the model performance was characterized and reported in a confusion matrix in Figure 9.

The confusion matrix generated by evaluating the test data after training was positive. Holes, which were previously regularly confused with the non-feature class, are consistently identified correctly. After retraining, 89% of hole predictions were correct, with only a single true hole predicted as another class. This level of accuracy is approaching a threshold where an automatic feature recognition system could provide value to a human operator.

It is promising to see such a notable increase in model performance by retraining on less than 800 machining features. Given access to more real-world training data, it is reasonable to expect real-world classification accuracy to approach that of a system trained and tested on entirely synthetic feature data.

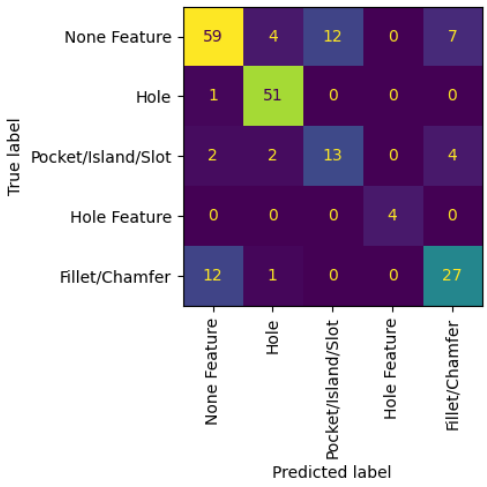


Figure 9: Confusion matrix of model evaluated against real-world data after re-training

7 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.

Several areas must still be explored before the feature recognition system developed in this paper can be incorporated into an industrial CNC machine workflow. We reduced the number of machining feature classes from 17 to 5 as our real world dataset lacked examples of 12 of the features used by Yeo et al. If a sufficiently diverse dataset of real world parts was available, our system could be trained to identify the same number of classes as the approach proposed by Yeo et al. The ID3 tree pre-classification step is presently

calculated once when training on generic CAD files, and is not updated during re-training. An exploration of possible methods for incrementally training the ID3 tree may be valuable to improve the classification accuracy of the re-trained system. More training data must also be collected in order to identify the point of diminishing returns for classification accuracy. Once classification accuracy can no longer improve by incorporating additional re-training data, other techniques (such as re-training the ID3 tree) may become necessary. To collect more training data, a system that integrates incremental training with CAM software should be developed. Integrating incremental training in a CAM software package, as proposed earlier in this section, will have the effect of significantly increasing the ease of collecting training data, and serve as a proof of concept demonstration of the proposed incremental learning technique.

Other enhancements to Yeo et al.'s method should be explored. Further, different approaches, such as BRepNet [8], could be used in place of Yeo et al.'s method. Our work should be considered one exploration within the evolving space of machine learning, with new enhancements and alternatives to Yeo etl al.'s basic approach being grounds for future work in feature recognition.

Source code for this work is available at
https://github.com/mlenover/machining-feature-recognition.

ACKNOWLEDGEMENTS

This research funded in part by the Natural Sciences and Engineering Reach Council of Canada (NSERC).

Michael Lenover, https://orcid.org/000-0000-1234-5678
Sanjeev Bedi, https://orcid.org/0000-0002-6993-8502
Stephen Mann, https://orcid.org/0000-0001-8528-2921
William Melek https://orcid.org/0009-0002-5616-0268

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] Donaldson, I.A.; Corney, J.R.: Rule-based feature recognition for 25D machined components. International Journal of Computer Integrated Manufacturing, 6(1-2), 51–64, 1993. ISSN 0951-192X. http://doi.org/10.1080/09511929308944555. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09511929308944555.

[3] Fieguth, P.: An Introduction to Pattern Recognition and Machine Learning. Springer International Publishing, Cham, 2022. ISBN 978-3-030-95993-7 978-3-030-95995-1. http://doi.org/10.1007/978-3-030-95995-1.

[4] 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].

[5] Joshi, S.; Chang, T.C.: Graph-based heuristics for recognition of machined features from a 3D solid model. Computer-Aided Design, 20(2), 58–66, 1988. ISSN 0010-4485. http://doi.org/10.1016/0010-4485(88)90050-4.

[6] 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].

[7] Kyprianou, L.K.: Shape classification in computer-aided design. Ph.D., University of Cambridge, 1980. https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.236370. Accepted: 1980.

[8] Lambourne, J.G.; Willis, K.D.D.; Jayaraman, P.K.; Sanghi, A.; Meltzer, P.; Shayani, H.: Brepnet: A topological message passing system for solid models. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 12768–12777, 2021. http://doi.org/10.1109/CVPR46437.2021.01258.

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

[10] 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.

[11] Prabhakar, S.; Henderson, M.R.: Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. Computer-Aided Design, 24(7), 381–393, 1992. ISSN 0010-4485. http://doi.org/10.1016/0010-4485(92)90064-H.

[12] Sakurai, H.; Chin, C.W.: CHAPTER 4 - Definition and Recognition of Volume Features for Process Planning. In J.J. Shah; M. Mäntylä; D.S. Nau, eds., Manufacturing Research and Technology, vol. 20 of Advances in Feature Based Manufacturing, 65–80. Elsevier, 1994. http://doi.org/10.1016/B978-0-444-81600-9.50009-2.

[13] Shah, J.J.; Shen, Y.; Shirur, A.: CHAPTER 7 - Determination Of Machining Volumes From Extensible Sets Of Design Features. In J.J. Shah; M. Mäntylä; D.S. Nau, eds., Manufacturing Research and Technology, vol. 20 of Advances in Feature Based Manufacturing, 129–157. Elsevier, 1994. http://doi.org/10.1016/B978-0-444-81600-9.50012-2.

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

[15] Tang, K.; Woo, T.: Algorithmic aspects of alternating sum of volumes. Part 1: Data structure and difference operation. Computer-Aided Design, 23(5), 357–366, 1991. ISSN 0010-4485. http://doi.org/10.1016/0010-4485(91)90029-V.

[16] 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.

[17] Verma, A.K.; Rajotia, S.: Feature vector: a graph-based feature recognition methodology. International Journal of Production Research, 42(16), 3219–3234, 2004. ISSN 0020-7543. http://doi.org/10.1080/00207540410001699408. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207540410001699408.

[18] Verma, A.K.; Rajotia, S.: A hint-based machining feature recognition system for 2.5d parts. International Journal of Production Research, 46(6), 1515–1537, 2008. http://doi.org/10.1080/00207540600919373.

[19] 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.

[20] 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.