Neural network architectures and feature extraction for lumber production prediction

Vincent Martineau, Michael Morin, Jonathan Gaudreault, Philippe Thomas, Hind Bril El-Haouzi

Neural Network Architectures and Feature Extraction for Lumber Production Prediction

Article · June 2021

DOI: 10.21428/594757db.89eadeff

5 authors, including:

Michael Morin
Laval University
26 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Jonathan Gaudreault
Laval University
84 PUBLICATIONS **545** CITATIONS
SEE PROFILE

Philippe Thomas
University of Lorraine
107 PUBLICATIONS **734** CITATIONS
SEE PROFILE

Hind BRIL EL HAOUZI
University of Lorraine
84 PUBLICATIONS **409** CITATIONS
SEE PROFILE

Neural Network Architectures and Feature Extraction for Lumber Production Prediction

Vincent Martineau[a,b,*], Michael Morin[a,c], Jonathan Gaudreault[a,b],
Philippe Thomas[d], Hind Bril El-Haouzi[d]
[a] FORAC Research Consortium, Université Laval, Québec, QC, Canada
[b] Department of Computer Science and Software Engineering,
Université Laval, Québec, QC, Canada
[c] Department of Operations and Decision Systems, Université Laval, Québec, QC, Canada
[d] Université de Lorraine, CNRS, CRAN, F-88000 Epinal, France

Abstract

We tackle the problem of predicting the lumber products resulting from the break down of the logs at a given sawmill. Although previous studies have shown that supervised learning is well suited for that prediction problem, to our knowledge, there exists only one approach using the 3D log scans as inputs and it is based on the iterative closest-point algorithm. In this paper, we evaluate the combination of neural network architectures (multilayer perceptron, residual network and PointNet) and log representation as input (industry know-how-based features, 2D projections, and 3D point clouds) in the context of lumber production prediction. Our study not only shows that it is possible to predict the output of a sawmill using neural networks, but also that there is value in combining industry know-how-based features and 3D point clouds in various network architectures.

Keywords: Forest-Product Industry Production Forecasting, Feature Engineering, Deep Learning Application, Machine Learning Application

1. Introduction

The task of forecasting the lumber products resulting from the break down of logs at a given sawmill is complex. As shown in Figure 1, a log break down generally results in multiple lumber products of different value at the same time. Although the equipment aims at maximizing the value of a log once transformed, i.e., the total value of the lumber products for that log, two sawmills would produce different lumbers from the same log due to the differences in their equipment configuration. The problem has a large output space. In fact, a sawmill of moderate size can produce more than eighty different products which could lead to more than a thousand feasible combinations of lumber products to choose from when predicting the outcome of the break down of a single log.

In practice, good production forecasts can lead tremendous gains to a forest-product company due to better planning opportunities [1]. One may, for instance, improve the assignment of cutting blocks to sawmills. When the output of a plant for a given set of logs

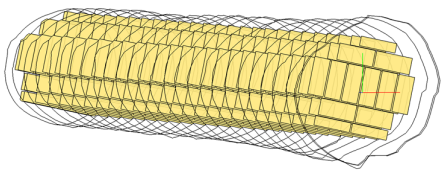

Figure 1. Representation of a log and of the lumber products resulting from its break down.

*vincent.martineau.1@ulaval.ca

is known (or estimated) prior to processing, the assignment can be optimized which in turn leads to a better response to the demand and a higher production value.

Nowadays, for forecasting purposes, forest-products industry companies have access to sawing simulators such as RAYSAW [2], Optitek [3], and SAWSIM [4]. These tools simulate the sawing of an individual log given a sawmill model and a log virtual representation. However, sawing simulations require too much time to be used on very large volumes of wood when a quick response is needed. Depending on the configuration of the simulator, the simulation can take anywhere from a few seconds to a few hours per log. Such a lengthy simulation time is impractical in some decision-making contexts involving an optimization process that needs to assess the quality of all possible log and sawmill pairs, e.g., for wood allocation planning [1] or for short-term decision-making such as operation scheduling [5].

As an answer to the issue of lengthy simulation time, previous studies have shown that it is possible to use machine learning to solve the problem of predicting the output of a sawmill with precision sufficient for decision-making in the context of wood allocation planning [1]. Our goal is therefore to find an approach that allows us to learn on a small batch of simulated logs to predict the optimal product basket for a large number of logs in a short time.

Currently, most developed machine learning models for that problem use features based on the know-how of forest-product industry specialists. Although useful, these models do not exploit all the available information found in virtual logs which are usually represented by 3D point clouds. To our knowledge, there exists, in the literature, a single approach to the lumber production prediction problem that uses the 3D point clouds of the logs in a machine learning setting [6]. It is based on the iterative-closest point algorithm [7, 8] and focuses on computing similarity between 3D point clouds for k-nearest neighbors processing.

We propose to use neural networks which are known to perform well in contexts where a large quantity of information is available (such as in the 3D point clouds of logs). To that end, we compare the performance of various neural network architectures and of various feature extraction approaches to the problem of forecasting a sawmill production.

Our study is conducted in two phases. First, we explore different representations of the logs for prediction purposes. Each representation can be viewed as a different level of abstraction of the logs 3D point cloud. Second, we assess the predictive performance of neural networks built using promising architectures from the literature when combined with the chosen representations. Empirical results on industrial data support our study.

The paper is organized as follows. We review the related literature and concepts in Section 2. In Section 3, we present the virtual log representations used to train our neural networks. We describe our experimental setting and our industrial data in Section 4. Finally, we discuss the results in Section 5 and conclude in Section 6.

2. Preliminary Notions

Regression and classification problems involving neural networks and 3D point clouds have generated a tremendous body of literature over the past few years. In this section, we first define the prediction problem we tackle. Then, we review neural network architectures for supervised learning tasks using 3D point cloud inputs, 2D projections of 3D shapes, and vectors of extracted features as inputs. Due to the high volume of literature on the subject, we focus on architectures known for their good performance in classification and regression on various problems. As a result, we discuss the multilayer perceptron, ResNet, and PointNet. An in-depth survey of the literature, e.g., [9], is out of the scope of this paper.

2.1. Prediction Problem Formulation

A solution to the lumber production prediction problem is a forecast of the lumber products resulting from the break down of a given log at a given sawmill. In practice, solving

this problem with a sufficient accuracy on a given log can lead better decisions for various planning problems which require forecasting the output of a sawmill for a batch of logs [1].

In the industry, a combination of lumber products resulting from a log processing is called the *product basket*. Previous works state the lumber production prediction problem in terms of a supervised learning problem [1, 10]. That is, pairs of feature vectors and product baskets are given to a supervised learning algorithm, such as Random Forest, and the algorithm builds a model from the feature space to the basket space.

In the Canadian forest-product industry, softwood lumber product types are normalized according to the NLGA (National Lumber Grades Authority) grading rules [11].[1] The result is that there is only a finite amount of lumber types. Moreover, not all sawmills can produce all types of products. The feasible product types depend on constraints such as the equipment configuration and the actual supply of logs, e.g., their size and species. Therefore, we represent the product basket by a vector of size p where p is the number of feasible lumber product types at the sawmill. Each entry is a count representing the number of lumbers of a particular type of feasible lumber product produced by the log break down.

Depending on the model, one can see the problem either as a classification problem or as a multi-output regression problem. In the case of *multi-output regression*, an attempt is made at directly predicting the number of each lumber product in the basket. Moreover, the outputs are real-valued and it becomes possible to predict fractions of products. In the *classification* setting, classes correspond to unique baskets seen during training. That is, each feasible combination of products encountered in the training set leads to a class in the problem definition. This approach has the advantage of having a finite number of outputs.

In the following subsections, we discuss the neural network architectures that we retained for our study.

2.2. **Multilayer Perceptron**

Multilayer perceptrons were shown to provide good results for both classification and regression problems [12, 13]. We will use multilayer perceptrons as our baseline for evaluating more evolved architecture.

2.3. **ResNet**

Residual networks (or ResNet) are a form of neural networks composed of several layers that use residual blocks and a skip connection mechanism [14]. A layer can be skipped using the skip connection mechanism. This design helps to reduce the problem of vanishing gradients. It is common for a ResNet to be able to skip more than one layer at a time. We use residual blocks of two layers separated by a non-linear ReLU activation.

2.4. **PointNet**

PointNet is an architecture that has been developed for the classification of objects in point cloud or scene segmentation based on point cloud [15]. In the case of our study, we are interested in the object classification functionality of PointNet. Furthermore, it has been developed to use point clouds directly and contains a mechanism that handles a variable number of points as input. This feature is of importance in the context of our prediction problem since the number of points in a 3D scan varies from one log to the next. In addition, this type of neural network is robust to the position of the points in the input and to the corruption of the point cloud.

[1] A lumber product type has four attributes, i.e., its length, width, thickness, and grade. The grade of a lumber product represents its quality. Defects, e.g., knots or bark remnants on the outer edge, reduce the quality of a lumber product.

3. Extracted Features Formats and Neural Networks

In this section, we describe the various representations for the input layer, namely, raw unprocessed point clouds and normalized point clouds, 2D projections of point clouds, features extracted from forest-product industry know-how, and a combination of these features. In the last subsection, we summarize the adaptation we made to the original architectures to better fit them to the problem at hand.

3.1. Raw Point Clouds

Logs 3D point clouds can be obtained in various fashions. For instance, sawmills usually have scanners at the beginning of their line which scans the entering logs. Moreover, some harvesters are equipped with a scanner that scans the whole tree at once while cutting branches (hence leading to a single scan containing multiple logs). In many cases, however, log scans come from log databases used for sawmill configuration or simulation and optimization-based decision-making.

Figure 2 shows a raw point cloud for a specific log. Such a point cloud has some peculiarities. The scan is composed of slices (or sections). Each section is the result of a reading at a fixed distance from the origin of the log. Considering this step produces a lot of points, an algorithm is used to reduce the number of points on the slice, but there is no guarantee on the number of points used to represent it. The average number of points in a log is 18,474 points, but some logs have fewer than 5,000 points and some have more than 25,000. Another peculiarity of this representation is that the resolution of the scan is not constant: Slices may be missing, and the points in a slice are not evenly distributed on the log surface.

Figure 2. A log raw 3D point cloud.

3.2. Normalized Point Clouds

Our second feature extraction approach involves *normalized 3D point clouds*. We normalize the raw point clouds by ensuring a constant number of points in each slice and aligned points from one slice to the next. Our normalization algorithm proceeds in two steps, first it constructs normalized slices, then it constructs the complete normalized scan from the normalized slices. *Slice normalization* is as follows.

(1) The points from a slice are connected to form a loop. Then, new points are organized evenly around the slice.
(2) The slice centroid is computed by averaging the position of the new points.
(3) The reference point is chosen so that it is intersected with the Y-axis and the periphery of the slice as defined by its basis spline (or B-spline).
(4) Then, assuming we need H points per slice, we position all points at the intersection of the line segment from the centroid to the slice periphery B-Spline in direction $\frac{2\pi i}{H}$ where i is the i^{th} clockwise point on the slice periphery of the slice.

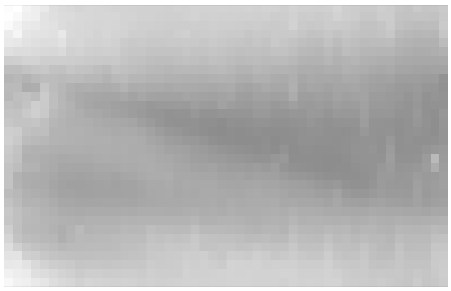

Figure 3. The pixel matrix of the 2D projection of a log.

We chose the B-spline since it is appropriate for surface fitting [16]. Once the number of points per slice is normalized, the algorithm proceeds with the construction of the normalized scan by creating new slices and *aligned points*. For that purpose, it computes W new slices containing H points each. In our experiments, $H = 128$ points per slice worked well. Furthermore, we determined that the maximum length of a log is under 560 centimeters and chose $W = 128$ slices. Therefore, we fixed the step between each slice to 4.73 cm. To position the new points for the desired slice we create a B-Spline from all points at index W_i. Since we want a constant number of points per log, dummy slices are added to lengthen shorter logs, i.e., logs with fewer than W slices are padded with slices containing H zero-valued points.

3.3. **2D Coordinates Projection**

2D projections of the log enable us to represent the log as a 2D pixel matrix. The idea is to encode the 3D log scans so that the information they contain is more readily available to the neural networks intended for 2D images processing. Intuitively, one can see the enclosing 3D shape of a log as a cylinder. The center of the cylinder is a straight line connecting the centroids at both ends of the log. The information we are interested in is the distance from the cylinder center to the log surface at various angles. To perform the projection, we use the normalized point cloud discussed in Section 3.2 to produce an H by W pixel matrix where each pixel represents the distance from the center of the cylinder for a given angle. Unless otherwise mentioned, all logs are aligned on the same end of the pixel matrix. Therefore, for logs shorter than the maximal log length in the data, a black pixel buffer is added to the right.

Figure 3 shows a representation of the projection of the log from Figure 2. Pixels are zoomed in to better show their shades of gray. For a given pixel, a lighter shade of gray indicates that the point is further from the center of the log and a darker shade of gray indicates it is closer to the center. To have a consistent level of white we used a maximum reference radius of 40 centimeters which corresponds to the maximum diameter for the logs in our industrial dataset.

One drawback of the approach is that logs with a highly pronounced curvature would lead to a reference axis that is outside some slices leading to an error in the measurement of the radius. Despite this restriction, this representation allows us to pass on the same information as normalized 3D point clouds with an input layer 66% smaller.

In the next few sections, we describe the variants of our 2D projection approaches.

3.3.1. **Circular and Alternate Two-Layer Padding**

To possibly alleviate the loss of information along the B-spline crossing reference points chosen to "unroll the log", we propose circular padding and alternate two-layer padding.

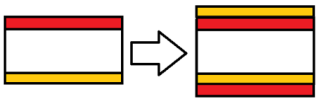

Figure 4. An example of circular padding.

Figure 5. An example of alternate two-layer padding.

In *circular padding*, we duplicate a portion of the upper (resp. lower) extremity of the projection and we concatenate it at the opposite end. This approach could be seen as if the points before 0 degrees and past 360 degrees are included twice in the projects. Figure 4 shows how data transfer is done for the top and bottom of the projection. In yellow (resp. red), we see the portion of the 2D projection we copy from the bottom (resp. top) of the pixel matrix of the 2D projection to the top (resp. bottom). In the case of our experiments, we copy 10% of the data.

In *alternate two-layer padding*, each point is represented exactly twice to retain all the information around the dividing line. To achieve this, we create two projections: the first as usual, the other following a 180-degree rotation on the length axis. The two projections are placed in two channels, e.g., a second color channel for an image. This way, all the information is present exactly twice and each point on the log surface is represented in its whole context. Figure 5 represents a simplification of the process. In this example, we use two layers that correspond to two projections. The second projection shows the projection of the same log rotated 180 degrees. This representation allows for an input layer that is 33% smaller than that of the normalized point cloud.

3.3.2. **Data Augmentation**

Considering that convolution neural networks are sensible to translation, we implemented data augmentation to provide more examples during training. *Data augmentation* consists in generating additional training examples by performing random modifications on the actual examples of the training set [17]. In our specific context, we perform a random translation of the nonzero pixels sub-matrix along the log length. The nonzero sub-matrix of all logs have as much or fewer columns as the number of slices we fixed when normalizing logs, i.e., W, the longest log cannot be translated. Since the total width of the pixel sub-matrix corresponds to the length of the longest logs, for all the missing slice we add an additional column of zeros to the projection sub-matrix prior to data augmentation. During data augmentation, a random rotation of the logs is also performed along their length axis. Logs can also be randomly flipped along the length axis. We perform data augmentation before doing the circular or alternate two-layer padding. These steps are performed when training and will be different each time a log is sampled for training.

3.4. **Automatic Feature Extraction Based on Forest Products Industry Know-How**

Forest products industry know-how-based features, also known as parametric data, are features engineered by forest engineers to characterize the logs. We use the wide-end and the small-end diameter, the length, the volume, a measure of the log curvature, and the taper, a measure of the decrease in diameter (from the wide to the small-end). Those features

can be computed from the raw 3D point clouds using specific algorithms without requiring sawing simulation.

3.5. **Combined Input**

As a last encoding on the input layer, we combined 2D projections and the know-how-based features representations, and the point clouds and know-how-based features representations. Our assumption is that know-how-based features would help the learning process to converge faster with fewer data.

3.6. **Adapting Artificial Neural Network Architectures**

To use neural networks for our experiments, we had to adapt the architectures mentioned in Section 2. One adaptation we made is to use multi-output regression, all models use the mean squared error as a loss function. Furthermore, for regression, we do not use activation after the last layer in the neural networks. For classification there was no need for adaptation, we use cross-entropy for the majority of models with the exception of experiments with PointNet. In this case, we used the PointNet's original loss function [15]. Finally, the last modification that we made to all neural networks is to adjust the size of the outputs to match the properties of our dataset. In classification the networks have 1188 outputs (1188 classes) representing all possible product baskets while in multi-output regression, the neural networks produce a vector of 85 elements (85 outputs) representing all possible lumber products for our sawmill.

In what follows, we detail the specifics of the networks we used.

Multilayer perceptron. In the case of multilayer perceptron, we use four fully connected layers separated by non-linear rectified linear unit (ReLU) activation units [18]. The model is compatible with all the data representations we have developed.

ResNet. For ResNet models, we use a modified version of the ResNet18 version provided in the Pytorch library. In the original implementation, the network input requires an image with three color channels. In our case we need one- or two-color channels. For the rest, we use the proposed implementation. We implemented ResNet models with the various 2D projection representations only.

PointNet. The PointNet model is used without any modifications. This model will use the raw point representation.

Models combining multiple representations. Models combining multiple data representations are either PointNet or ResNet18 networks with their aforementioned input layer and an additional exit. The exit of these networks are combined with the know-how-based features and then inputted to a multilayer perceptron. The training is done in a single pass to enable the multilayer perceptron and the PointNet network to train at the same time.

4. **Experiments**

The goal for our experiments is to determine a combination of the architecture and log representation that provides the best results. The combinations we explore are described in Table 1. We tested a total of 17 combinations of architecture and log representation in both the multi-output regression and classification settings leading to a total of 34 variants.

In what follows, we provide more details on the data that was used for the experiments, the process we used to train the models and how we evaluate the models performance.

Table 1. All combinations of log representations (on rows) and architectures (on columns); all models are tested for both classification and regression.

	Multilayer Perceptron	**ResNet18**	**PointNet**
Raw Points	–	–	✓
Normalized Points	✓	–	–
2D Projections			
Regular	✓	✓	–
Circular Padding	✓	✓	–
2-Layer Padding	✓	✓	–
2D Projections with Augmentation			
Regular	✓	✓	–
Circular Padding	✓	✓	–
2-Layer Padding	✓	✓	–
Know-how-based	✓	–	–
Raw Points + Know-how-based	–	–	✓
2D Projections + Know-how-based	–	✓	–

4.1. **Dataset**

The industrial dataset we use for the experiments consists of 2225 logs scanned and provided by FPInnovations. The break down of each log leads to a basket of products with at least one lumber product. The Optitek sawmill simulator and the data it provides were used as a ground-truth sawmill for our experiments, i.e., it provides the output baskets and therefore the labels used for training and testing. Furthermore, we used the Optitek simulator to compute the know-how-based features (although they could have been computed directly from the 3D scans). It is also important to mention that the logs chosen for the experiments are logs that have already been processed by a real sawmill. In fact, the model of an existing sawmill is used for the simulations. As in a real sawmill, the simulator allows for some randomization when breaking down a log. That is, log rotations have a small margin of error following a normal distribution with a mean of 0 degree and a standard deviation of 17 degrees. Although we used the Optitek simulator as our ground-truth sawmill, it could be replaced by the data from a real sawmill. At this time, however, real sawmills lack the traceability needed for such a direct application of our methods.

The sawmill chosen for the simulations can produce 85 different products. Therefore, in the case of multi-output regression, the model prediction must be an 85-dimensional vector. In the case of classification, our dataset contains 1188 unique product baskets.

Such a high number of feasible product baskets means that it is common for a class in the test set not to be present in the training set. This is the case for about 30% of the logs in the test sets of our experiments. Some product baskets are indeed more frequent than others which makes the problem unbalanced and particularly difficult.

4.2. **Model Training**

To ensure our observations generalize well, we randomly partitioned our industrial dataset so that 85% of the logs are used to train models and 15% of the dataset are used to evaluate our models. Data augmentation is done in a way that samples cannot leak from the training dataset to the test dataset. Furthermore, unless otherwise mentioned, the reported scores is the average of 10 runs on such partitions.

To train our models, we use the Adam optimizer, and we allow 500 training epochs. During the training we reduce the learning rate each time we reach 10 epochs without the network improving. Finally, we use early stopping to reduce overfitting. The training stops

when the learning rate is reduced twice without getting any gain in learning. The initial learning rate is set to 0.0001.

4.3. **Performance Evaluation**

To evaluate the performance of our networks we mainly use the F1-score [19] of which we adapted the interpretation to the context of our problem. Sawmill managers generally want to maximize precision and recall. S/he wants to know whether a model predicts too many or too few products. In this situation, it was natural to use the F1-score as a global metric since it combines precision and recall in a sound fashion.

In cases where the model is trained for classification, before applying the metrics the class is transformed into a product basket. Multi-output regression outputs are used as-is. Let b be the ground-truth product basket, i.e., the vector representing the lumbers resulting from the break down of a log. Let y be the predicted product basket vector of counts (real-valued for multi-output regression and integer-valued for classification). We define $TP(y, b)$ as the number of lumbers that are in both y and b, $FP(y, b)$ as the number of lumbers that are in y but not in b, and $FN(y, b)$ as the lumbers that are in b but not in y. Given those definitions, we use the standard precision and recall definitions:

$$precision(y, b) = \frac{TP(y, b)}{TP(y, b) + FP(y, b)}\,, \tag{4.1}$$

and

$$recall(y, b) = \frac{TP(y, b)}{TP(y, b) + FN(y, b)}\,. \tag{4.2}$$

The F1-score, which corresponds to the harmonic mean between precision and recall, can be defined as

$$F1(y, b) = \frac{2TP(y, b)}{2TP(y, b) + FP(y, b) + FN(y, b)}\,. \tag{4.3}$$

Finally, we use, as a baseline for comparison, a predictive model that predicts the average basket of products of the training set. It should be noted that, according to the literature [1], the average production is sometimes used in the forest product industry for forecasting purposes which make it a valuable baseline.

5. **Results and Discussion**

The experiments were conducted in two phases. In the first phase, we compare our multiple 2D projection variants to see if any variant outperforms the others. Then, in the second phase, we compare the architecture and representation combinations, including the 2D projections retained from the first phase.

5.1. **Phase 1: 2D Projections Performance Comparison**

Table 2 compares the performance, in terms of average F1-score, of the various entries for the multilayer perceptron and ResNet18 algorithm when using 2D projections on the input layer and a multi-output regression or classification output. Regular 2D projections, i.e., without padding, are compared to circular and alternate two-layer padding with and without data augmentation (on the right-hand side and left-hand side of the table respectively).

Our first observation is that on average the models perform much better in classification than in multi-output regression mode. Another important point is that data augmentation helps the learning algorithms obtain a higher average F1-score in the case of classification. We can see an increase of at least 8% in the average performance with data augmentation for classification. For multi-output regression, however, there do not appear to be substantial benefits in using the data augmentation. This observation holds true for every 2D

Table 2. Average F1-scores of the neural networks using our various 2D projections (with 95% confidence intervals); for each pair of architecture and output type, the best average F1-score value is highlighted in gray.

	Without Data Augmentation		**With Data Augmentation**	
	Classification F1-score	Regression F1-score	Classification F1-score	Regression F1-score
Sawmill Average	0.1481 ± 0.002	0.1481 ± 0.002	0.1481 ± 0.002	0.1481 ± 0.002
MLP				
Regular	0.4809 ± 0.008	0.4311 ± 0.003	0.5337 ± 0.007	0.4253 ± 0.006
Circular Padding	0.4914 ± 0.013	0.4288 ± 0.006	0.5338 ± 0.011	0.4250 ± 0.006
2-Layer Padding	0.4896 ± 0.007	0.4305 ± 0.009	0.5300 ± 0.010	0.4334 ± 0.004
ResNet18				
Regular	0.4412 ± 0.009	0.4514 ± 0.005	0.5144 ± 0.017	0.4417 ± 0.006
Circular Padding	0.4400 ± 0.007	0.4450 ± 0.005	0.5125 ± 0.022	0.4412 ± 0.005
2-Layer Padding	0.4382 ± 0.009	0.4438 ± 0.008	0.5256 ± 0.014	0.4381 ± 0.007

projection representations we developed. Given a fixed architecture (with or without data augmentation) and a fixed learning problem type (multi-output regression or classification), we cannot conclude there is a significant difference in terms of average F1-score between the representation approaches we developed. Nonetheless, the average performance is higher, most of the time, with regular padding when using the ResNet18 architecture whereas it is higher, most of the time, with circular padding when using the multilayer perceptron architecture.

Regarding the performance of ResNet18 against that of the multilayer perceptron, we observe that the multilayer perceptron performs better on average when both are in classification mode. There is a difference of up to 0.05 points in some cases between both models. However, ResNet18 outperforms the multilayer perceptron when both are in multi-output regression mode. In previous work, ResNet18 proved to work well on image classification [14]. In our case it is possible that it is hard to extract patterns from our 2D projections.

5.2. **Phase 2: Architectures and Data Representations Performance Comparison**

Table 3 shows a comparison between the various models, representations, and combined representations we used for our experiments. The average F1-scores, precision and recall performance are presented. For 2D projections, we retained, from Section 5.1, the circular padding with data augmentation variant for the multilayer perceptron, and the regular padding with data augmentation variant for ResNet18. All the representations we have used perform significantly better than the baseline model predicting the average of the training set. On our industrial data, we see strong benefits in using neural networks rather than using an average production for a given plant. We also observe that the combination of representations with know-how-based features gives a slightly better average performance in terms of F1-score for both 2D projections and point cloud representations in classification.

Results obtained by neural networks using the know-how-based features are similar to the other models which support the development of such features in further research. We also notice that the models using the know-how-based features as input layer do slightly better than normalized point clouds in a fully connected network. However, the best neural network, on average, combines both the know-how-based features and the normalized point cloud. This support our hypothesis that log scans contain information not yet exploited by the features extracted using industry know-how. It should be noted that it is prudent not to conclude on a single best combination of architecture, log representation, and output mode at this time as the differences between the best combinations remain small.

Table 3. Average F1-scores (with 95% confidence intervals), and average precision and recall for all network architectures and data representations including combined representations; for each pair of architecture and output type, the best average F1-score value is highlighted in gray.

	Classification			**Regression**		
	F1-score	Prec.	Rec.	F1-score	Prec.	Rec.
Sawmill Average	0.148 ± 0.002	0.158	0.152	0.148 ± 0.002	0.158	0.152
MLP						
Know-how-based	0.524 ± 0.015	0.521	0.550	0.441 ± 0.007	0.476	0.423
Circular 2D projection	0.533 ± 0.007	0.527	0.571	0.425 ± 0.006	0.455	0.413
Normalized Points	0.495 ± 0.015	0.502	0.517	0.420 ± 0.030	0.447	0.410
PointNet						
Raw points	0.533 ± 0.017	0.530	0.560	0.416 ± 0.012	0.430	0.410
Raw points + Know-how	0.543 ± 0.011	0.537	0.575	0.435 ± 0.008	0.471	0.419
ResNet18						
Regular 2D projection	0.514 ± 0.017	0.510	0.543	0.441 ± 0.006	0.477	0.423
Reg. 2D + Know-how	0.538 ± 0.011	0.532	0.568	0.422 ± 0.008	0.454	0.408

In general, classification models perform significantly better than multi-output regression models in our context. This behavior is consistent for all models. As can be seen in Table 3, the average precision and recall values are very different depending on whether we use the network in classification or multi-output regression mode. Interestingly, all classification models obtain a higher recall whereas for all multi-output regression models we observe the opposite. We conclude our multi-output regression models tend to underestimate the production whereas our classification models tend to overestimate it. This behavior could be explained by the fact that a class needs to be converted into a vector of counts to compute the precision and recall metrics. In a classification context, a class needs to be converted into a complete basket to compute the precision and recall metrics. During the conversion from a class to a vector of counts, a nonzero count might be forced for a given product type although there is a high probability of having a count of zero for that product type given the log shape. That is, a high probability of "belonging" to a class, does not necessarily translate into a high probability of a nonzero count for all products types of that class. For multi-output regression, there is no constraint to predict an integer for each count, a small probability of having a product can be outputted as an accordingly small count.

6. Conclusion

In this research, we developed and evaluated 34 combinations of neural network architectures, input representations, output modes and training processes for the lumber production prediction problem. Our research definitely shows that it is possible to predict the basket of products associated with a log using neural networks. Our results also show there is value in exploiting know-how-based features as well as the information from the 3D point clouds. In fact, both low-level and high-level representations enabled the networks to achieve a similar performance (in terms of average F1-score). Nonetheless, neural networks combining both know-how-based features and 3D point clouds tend to outperform our other architectures.

Also, as mentioned, we observed the results are better in classification than in multi-output regression mode. It also appears that regression and classification predictions are complementary in our specific case (see the discussion on their opposite performance in terms of precision and recall). In addition, considering that a sawmill has limited capacity for log processing, it would be false to assume there could be an infinite number of classes. It would therefore be interesting to do an analysis on the size of the product basket-space

with respect to a sawmill configuration and to evaluate the performance of classification against the number of feasible classes.

Acknowledgements

The authors would like to thank the FORAC Research Consortium and its partners. Our gratitude goes as well to the Natural Sciences and Engineering Research Council of Canada (NSERC) who provided funding for this research.

References

[1] M. Morin, J. Gaudreault, E. Brotherton, F. Paradis, A. Rolland, J. Wery, and F. Laviolette. "Machine learning-based models of sawmills for better wood allocation planning". In: *International Journal of Production Economics* 222 (2020), p. 107508.

[2] R. E. Thomas. "RAYSAW: A log sawing simulator for 3D laser-scanned hardwood logs". In: *18th Central Hardwood Forest Conference*. Vol. 117. 2012, pp. 325–334.

[3] FPInnovations. *Optitek 10. User's Manual*. 2014.

[4] HALCO. *HALCO Software Systems Ltd.* http://www.halcosoftware.com. [2021-02]. 2016.

[5] M. Rönnqvist. "Optimization in forestry". In: *Mathematical programming* 97.1 (2003), pp. 267–284.

[6] C. Selma, H. B. El Haouzi, P. Thomas, J. Gaudreault, and M. Morin. "An iterative closest point method for measuring the level of similarity of 3D log scans in wood industry". In: *Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, 2018, pp. 433–444.

[7] Y. Chen and G. Medioni. "Object modelling by registration of multiple range images". In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[8] P. J. Besl and N. D. McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–606.

[9] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. "Deep learning for 3d point clouds: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[10] M. Morin, F. Paradis, A. Rolland, J. Wery, J. Gaudreault, and F. Laviolette. "Machine Learning-Based Metamodels for Sawing Simulation". In: *Proceedings of the 2015 Winter Simulation Conference*. WSC '15. IEEE Press, 2015, 2160–2171. ISBN: 9781467397414.

[11] National Lumber Grades Authority. *Standard Grading Rules For Canadian Lumber*. Westminster, BC, Canada, 2017.

[12] D. Rumelhart, G. Hinton, and R. Williams. *Parallel distributed processing: Explorations in the microstructure of cognition*. 1986.

[13] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. "A comprehensive analysis of deep regression". In: *IEEE transactions on pattern analysis and machine intelligence* 42.9 (2019), pp. 2065–2081.

[14] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[16] P. Dierckx. *Curve and surface fitting with splines*. Oxford University Press, 1995.

[17] L. Perez and J. Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: (Dec. 13, 2017). arXiv: 1712.04621v1 [cs.CV].

[18] V. Nair and G. E. Hinton. In: *Linear Units Improve Restricted Boltzmann Machines*.

[19] D. Powers. "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation". In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.