Automated Rock Quality Designation Using Convolutional Neural Networks

Fatimah Alzubaidi¹ · Peyman Mostaghimi¹ · Guangyao Si¹ · Pawel Swietojanski² · Ryan T. Armstrong¹

Received: 2 October 2019 / Accepted: 29 January 2022 / Published online: 26 February 2022
© The Author(s) 2022

Abstract

Mineral and hydrocarbon exploration relies heavily on geological and geotechnical information extracted from drill cores. Traditional drill-core characterization is based purely on the subjective expertise of a geologist. New technologies can provide automatic mineral analysis and high-resolution drill core images in a non-destructive manner. However, automated rock mass characterization presents a significant challenge due to its lack of generalization and robustness. To date, the automated estimation of rock quality designation (RQD), a key parameter for rock mass classification, is based mostly on digital image processing techniques with significant user biases. Alternatively, we propose using computer vision and machine learning-based algorithms for drill core characterization using drill core images to determine the RQD. A convolutional neural network (CNN) is used to detect and classify intact and non-intact cores, and to filter out empty tray areas and non-rock objects present in the core trays. The model calculates the length of the detected intact cores and estimates the RQD. We train the CNN model with thousands of sandstone core images from different drill holes in South Australia. The proposed method is tested on 540 sandstone core rows and 90 limestone core rows (~ 1 m each), which produces average error rates of 2.58% and 3.17%, respectively.

Highlights

- Core tray images are automatically analyzed for estimating rock quality designation.
- A convolutional neural network for core image classification is developed.
- The proposed approach is tested on sandstone and limestone images and results in an error rate of 3% approximately.

Keywords Rock quality designation (RQD) · Convolutional neural networks (CNNs) · Automation · Core tray images

1 Introduction

Rock quality designation (RQD) is a key element in rock mass classification, and an important input parameter in widely used rating systems, such as the rock mass rating (RMR) and *Q* values (Bieniawski 1973, 1989; Barton et al.

1974; Lemy et al. 2001; Chen and Yin 2019). RQD was introduced by Deere et al. (1967) as a modified core recovery percentage, and is commonly used in mining and geotechnical engineering. It is defined as the percentage of intact core fragments equal to or greater than 10 cm in the total core run, not including fractured and weak rocks (Deere and Deere 1988). Traditionally, the RQD is measured by visual observation of the rock core. In many cases, this workflow routine is susceptible to human error as a geologist processes several core trays (ten or more) at a time (Hadjigeorgiou 2012). The limited time for core logging procedures (Olson et al. 2015) may result in inaccurate and unreliable measurements. In addition, the lack of personnel experience or

✉ Ryan T. Armstrong
ryan.armstrong@unsw.edu.au

¹ School of Minerals and Energy Resources Engineering, The University of New South Wales, Sydney, Australia

² School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia

motivation can lead to inconsistent and subjective core logging (Hadjigeorgiou 2012).

Limitations of core analysis can be addressed through automatic procedures, which have been the focus of several studies in the last two decades. Lemy et al. (2001) presented an innovative method to automatically calculate RQD from grayscale images of core trays. They used a Steger line-detection algorithm (Steger 1998) to identify fractures represented by dark regions. The method relied on the difference in light intensity between the fracture and core regions in the grayscale images. The method was validated on four core trays (15 core rows) with a total core length of 6 m. The RQD was calculated for each core row and resulted in up to 42% error. More recently, Saricam and Ozturk (2018) introduced a new approach known as the shadow-based method. Similar to the research of Lemy et al. (2001), this method was based on classic image-processing algorithms. However, Saricam and Ozturk (2018) suggested a specific image-capturing procedure, in which images were taken at three different lighting angles to generate shadows. The purpose of these shadows was to improve image segmentation, and thus the detection of fractures. Their approach resulted in several improvements to the method of Lemy et al. (2001): (1) it tested a larger number of images; (2) it resulted in more accurate RQD calculation; and (3) it included fracture characterization to distinguish between natural and mechanical cracks. However, with the special imaging requirements, this method cannot be used for images taken from a single lighting position. Another limitation is that the method involves a color-based image thresholding step that requires blue plastic trays. When using trays with less color contrast, such as wooden or metal trays, segmentation may be less accurate. Thus, more robust methods that do not rely on the color of core trays and can use ordinary tray images without additional imaging requirements remain to be developed.

Another approach for automatic geotechnical logging is 3-D laser core scanning (Olson et al. 2015; Harraden et al. 2016). Olson et al. (2015) used a 3-D laser profile along the core to identify discontinuities. They also characterized and distinguished natural and mechanical fractures based on the orientation and roughness obtained from the laser data. Laser-based methods can provide accurate RQD calculations as they generate 3-D representations of cores allowing one to accurately detect and characterize fractures. Although laser scanning is effective for new cores, it is not applicable to old cores, which are often damaged or lost after significant storage time. In addition, this technology is expensive and requires special handling. Thus, extracting geotechnical parameters using 2-D core tray images is advantageous for field reassessment using existing data.

In this study, we propose a convolutional neural network (CNN) (LeCun and Bengio 1995) for automatic drill core characterization, specifically to estimate the RQD from core

tray images. We created a CNN model that can detect intact cores and identify fractured rock, crushed rock, empty tray areas, and non-rock objects present in the core trays. Geometrical information extracted from our model is used to estimate the RQD, producing results comparable to those from a trained geologist.

2 Convolutional Neural Networks

2.1 Basic Architecture of CNNs

A CNN is a special type of deep neural network in which one or more layers are convolutional. A typical feed-forward CNN, such as the one used herein, uses a sequence of convolutional steps, pooling, and fully connected layers to build a hierarchy of nonlinear transformations. These layers will be explained in the following paragraphs with reference to a generic CNN architecture, as depicted in Fig. 1a.

Convolutional layers are the core feature of CNNs; image features are extracted by convolving small kernels, as depicted in Fig. 1b. In the convolution process, a kernel scans the input by moving at a defined stride to produce feature maps. The output at each location z_j is computed as

$$z_j = \sum_i w_i x_i + b, \quad (1)$$

where i is a location on the kernel, x_i and w_i are the input and kernel weights at i , respectively, and b is the kernel bias. For example, a kernel of 3×3 pixels having weights $w_1 - w_9$ and a bias b starts the convolution at the location $x_1 - x_9$ of the input and yields output z_1 of the feature map following Eq. (1) (Fig. 1b). The convolving kernel then moves a stride of one pixel to scan the next location of the input and produces the corresponding output of the feature map.

The output from Eq. (1) (z_j) passes through an activation function that helps the network to ignore irrelevant information in the input and only activate at important features. The activation function introduces non-linearity to the convolution process, which enables the network to learn more complicated features than that achieved by a simple linear transformation (Gu et al. 2018; Nwankpa et al. 2018). A common activation function is the rectified linear unit (ReLU) (Nair and Hinton 2010) that calculates $f(x) = \max(0, x)$, whereby negative outputs are suppressed, and only positive outputs are passed to the next layer. ReLU offers easier network optimization and faster learning than other functions, such as the sigmoid or tanh activation functions (Krizhevsky et al. 2012; Sonoda and Murata 2017; Ramachandran et al. 2017).

A key advantage of the convolution process is the preservation of the spatial relationship between the inputs and outputs, i.e., the feature maps. The learned features by the convolution are not position-dependent; the features can be

Fig. 1 Generic CNN structure and operation. **a** Simplified CNN structure with two convolutional layers, one pooling layer, and one fully connected layer. **b** Convolutional operation for input channel with a 3×3 pixels kernel sliding by one pixel as indicated by the red

arrow. **c** Example of max pooling operation to transform a 6×6 input into a 3×3 output, starting from the highlighted location. **d** Fully connected layer of 3 units; each unit is connected to all elements in the input vector

detected by CNNs when appearing at any location of the input. Overall, the convolution makes CNNs more efficient in terms of learning ability and memory requirements than traditional neural networks (Goodfellow et al. 2016).

The pooling layers depicted in Fig. 1a are inserted between convolutional layers to summarize features and potentially increase the robustness of the model to variabilities in the input space (Goodfellow et al. 2016; Julian 2018). Max pooling is a common pooling operation, whereby the spatial size of feature maps is reduced by reporting the maximum value within a small window, as shown in Fig. 1c. It is usually performed using a 2×2 pixels window moving with a stride of two pixels in the vertical and horizontal directions.

Convolutional layers are typically followed by fully connected layers that map the discovered features to the desired predictions, as shown in Fig. 1a. A fully connected layer is composed of several units (neurons) that are connected to all elements of the input vector (Fig. 1d). Each unit has a weight vector and a bias and it computes the output as provided by Eq. (1) (Goodfellow et al. 2016; Ketkar 2017); the only difference is that x is the entire input vector, while in convolutional layers, x is the location of the input under the kernel. In a classification CNN, fully connected layers are used as classifiers that receive the features from the convolutional layers and predict class probabilities. In our classification problem, the classifier outputs a probability distribution of two values

that each corresponds to one of the classes: intact core and non-intact core.

For further details on CNN structure, we refer the reader to Goodfellow et al. (2016) and Alom et al. (2018).

2.2 Training and Regularization of CNNs

The CNN training process is usually performed with a large quantity of labeled data. During training, all weights and biases in the network are found using optimizers, such as stochastic gradient descent (SGD) (Robbins and Monro 1951; Kiefer and Wolfowitz 1952) and adaptive moment estimation (Adam) (Kingma and Ba 2014). A CNN model is trained to minimize a loss function by comparing model predictions for the training data with the true labels, which are usually assigned in a supervised manner. Cross-entropy loss, CE, is a commonly used loss function for classification problems (Goodfellow et al. 2016).

The cross-entropy loss function is expressed as

$$CE(\theta) = - \sum_{i=1}^N \log P(y_i|x_i;\theta), \quad (2)$$

where $CE(\theta)$ is the cross-entropy loss at specific parameters θ (i.e., weights and biases in each layer of CNN); $P(y_i|x_i;\theta)$ is the network prediction for an input image x_i using parameters θ , i.e., the probability x_i is assigned the correct class label y_i ; N is the total number of images in the training set. During training, the loss function continuously assesses the quality of parameters θ by measuring the agreement between the predicted labels and true labels for the training images. The goal is to find the parameters θ for which CE is minimized. Based on the resulting loss in each training iteration the optimizer, e.g., SGD updates the network parameters by a small amount that reduces the loss and moves it towards the minimum. The optimizer uses a learning rate that controls the amount of the update applied to the parameters and determines the speed of the learning process. The learning rate needs to be defined before training the model. The learning rate is a critical hyperparameter, because it strongly influences model performance (Goodfellow et al. 2016). The definitions of all hyperparameters used to build the CNN model in this study are explained in Sect. 4.1.1.

Minimizing the training loss is not the only objective when building a CNN model. Another objective is ensuring an appropriate level of generalization, i.e., the ability of the model to perform accurately not only on the training data but more importantly on new data, such as the validation or test data. One way to improve the model generalization is using regularization techniques. Regularization can be defined as, “any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error” (Goodfellow et al. 2016).

Regularization reduces the model tendency to overfit the training data, which can be accomplished by several methods. A common regularization procedure is limiting the weights of the network during training by penalizing the weights of non-important features in the training examples; this method is called weight decay (Krogh and Hertz 1992). A weight decay factor is a hyperparameter that needs to be defined before training. Another effective regularizer is dropout (Srivastava et al. 2014), which prevents overfitting by randomly dropping units (neurons) from the network during training. The units are dropped independently with a predefined probability. Another option for regularization is adding batch normalization layers as a part of the network structure (Ioffe and Szegedy 2015), which normalize the input to each layer in the network. The method is essentially developed to accelerate the training of deep networks, but it also acts as a regularizer. In our work, we tested weight decay, dropout probability, and batch normalization as regularization procedures, as discussed in Sect. 4.1.1.

2.3 Application of CNNs for Rock Analysis

CNNs have been widely applied in various disciplines, such as computer vision, image processing (Wang et al. 2016; He et al. 2017; Zhang et al. 2017a, b), and medical studies (Liang et al. 2016; Iskandar et al. 2018; Bao and Chung 2018). The application of CNNs in the mineral and energy resources sector has only recently emerged as an active field of study. There have been several implementations of CNNs using X-ray micro-computed tomography (micro-CT) images of rocks, such as enhancing the quality of the images (Wang et al. 2019) and predicting physical properties of rocks from the images (Alqahtani et al. 2020). Moreover, CNNs have been used to perform rock classification from digital images. Cheng and Guo (2017) used a CNN for grain size analysis from thin-section images. The authors built a CNN with four convolutional layers and trained it with 4800 images. The model successfully classified three types of rocks with different grain sizes, with a classification accuracy greater than 98%. Zhang et al. (2017a, b) identified rock lithology from borehole images. The training set included images from three sedimentary rock types: sandstone, shale, and conglomerate. A CNN with two convolutional layers followed by one fully connected layer was used. The test results demonstrated that lithology classification using CNN had a 9% higher accuracy than an artificial neural network (ANN). Unlike previous methods that included building CNNs, Shu et al. (2018) used a pre-trained CNN (VGG-19) as a feature extractor to classify the sorting level of rock particles, ranging from well-sorted to poorly sorted. They compared the classification accuracy using features obtained from the CNN with features from two conventional methods

of k-means and manual textural statistics. The results demonstrated that feature classification using CNN was better than conventional methods. The study also demonstrated that using a CNN for automatic analysis increased the classification accuracy and eliminated the time-consuming process of particle segmentation, which was a required step in previous approaches.

Thus, it can be concluded that even relatively simple CNN models are promising for identifying important rock features, and can provide an accurate and reliable method for the classification of rock images.

3 Data Collection and Preparation

Core tray images were obtained from the Geological Survey of South Australia (GSSA) (2020). The images were produced by the HyLogger™ core scanner, a spectrographic core logging and imaging system developed by the Commonwealth Scientists and Industrial Research Organization (CSIRO) (Hancock and Huntington 2010). Although the system was developed mainly to automate core mineralogy

Table 1 Description of cores in each tray as whole cores, half cores, or both

	Number of core tray images containing:		
	Whole cores	Half cores	Both
Train data set	38	34	2
Test data set 1	7	3	0
Test data set 2	72	14	6
Test data set 3	15	0	0
Total number	132	51	8
Total percentage (%)	69	26	4

Many of the trays included more whole cores than half cores. Trays used for RQD evaluation (test data sets 2 and 3) contained mainly whole cores

logging, it also includes an imaging system that produces a significant number of core tray images. The HyLogger system produces two types of data: (1) spectral data that are used to identify dominant minerals in cores; (2) image data with a high resolution of approximately 0.1 mm. This study only used the core tray images produced by HyLogger and

Fig. 2 Typical core tray images having different numbers of core rows and different core and tray dimensions. The scale on the horizontal axis was enhanced for better visualization at the shown resolution. More HyLogger images can be found in the Geological Survey of South Australia (GSSA)

Fig. 3 The range of core diameter in the trays used in the study, analyzed per core row. Approximately 90% of the rows included cores with 4–6 cm diameter

Table 2 Data sets used for training and testing the CNN model and evaluating the proposed method

Data set	Tray images	SSIs
<i>CNN model training and testing</i>		
Training data set	74 sandstone trays	6400
Test data set 1	10 sandstone trays	655
<i>Evaluating RQD estimation by the proposed method</i>		
Test data set 2	92 sandstone trays	
Test data set 3	15 limestone trays	

We used 84 sandstone tray images to create small square images (SSIs, 225×225 pixels) for building the CNN model. We evaluated the proposed method for RQD calculation on 107 sandstone and limestone tray images

was not concerned with the spectral data. The images were 14,500–15,800 pixels wide and 4100–6300 pixels high. The images display the depths of the core rows on the vertical axis and a scale on the horizontal axis, as shown in Fig. 2 and detailed in Sect. 3.1.

We used a total of 191 sandstone and limestone tray images from 13 boreholes to develop and evaluate the method. Typical tray images with different core and tray dimensions are shown in Fig. 2. Approximately 70% of the trays include whole cores, and the remainder contain half cores or a combination of both whole cores and half cores, as detailed in Table 1. Core diameters for the 13 different boreholes also differed, as shown in Fig. 3. To build the CNN model, our procedure required creating small square images (SSIs) from the core trays as will be explained in Sect. 3.1. We selected 74 sandstone trays to create the SSIs for training the CNN model and ten sandstone trays to create the SSIs for testing the trained model (see Table 2). This test is for evaluating the accuracy of the CNN model in classifying

each test SSI as intact core or non-intact core. To evaluate the performance of the overall procedure in estimating the RQD, we used 92 sandstone trays and 15 limestone trays; denoted in Table 2 as test data sets 2 and 3, respectively.

3.1 Data Preparation

This section describes the process of creating the small square images (SSIs) to be used for training and testing the CNN model. The dimension of the extracted SSIs was equal to the core diameter, which changed for the different boreholes (Fig. 3). The first step was to separate the tray into rows. Each tray contains depth information (text) displayed on the left side of the image, as shown in Fig. 4a, b. These depths represent the underground locations from which the cores were extracted, measured from the surface. In each tray, the number of depths is equal to the number of rows. We detected and processed the depths using Python-tesseract (Smith 2007) to obtain the number of rows and use it to separate and extract each row from the tray, as in Alzubaidi et al. (2021). The Python-tesseract is a wrapper for the Google Tesseract optical character recognition (OCR) tool that is commonly used to detect text that is placed on images. Each row was further divided along its length into the SSIs using a sliding window moving horizontally along the row, as shown in the last row in Fig. 4b. This occasionally included overlapping to obtain images with different features, such as different fracture locations, as shown in Fig. 5.

To assign the true class labels, we visually inspected the SSIs and decided whether the image belonged to the intact core or non-intact core class. Example images for each class are shown in Fig. 4c. The core trays selected for training included more intact core regions than fractured core, crushed core, and empty tray regions, which produced more SSIs for the intact core class than the non-intact core class. Using an imbalanced training data set is not recommended as it can produce a biased prediction towards the class with more training images (Ali et al. 2015; Leevy et al. 2018). Thus, we applied data augmentation to approximately 15% of the non-intact core SSIs to equalize the number of images for each class, and thus avoid the class imbalance issue. Data augmentation included horizontal and vertical flip, whereby new images are created by mirroring a training image with respect to the horizontal axis or the vertical axis, respectively, as demonstrated in Fig. 6. We created a total of 6400 SSIs from the 74 tray images in the training data set, with each class having 3200 images. During optimization of the model hyperparameters (Sect. 4.1.1), 20% of the training images were used for validation to assess the performance of the model during the optimization process. The final model was trained with all 6400 training images.

Using the same procedure for defining the SSIs in Fig. 4 and assigning the true labels, we created a total of 655

Fig. 4 Procedure of creating small square images (SSIs) to train and test the CNN model: **a** input tray image with core depth of each row of the tray shown on the vertical axis; **b** illustration of row depth

detection, row separation, and defining the SSIs; **c** manually labeled SSIs from the two classes. In this example, seven depths were detected using OCR and thus the tray was separated into seven rows

images from the ten trays selected for testing the CNN model (test data set 1 in Table 2).

3.2 Data Pre-processing

The SSIs with sizes ranging from 700×700 pixels to 1200×1200 pixels (because of the different core diameters, refer to Fig. 3) were downsampled to a standard size of 225×225 pixels as input size for the CNN model. Although different input sizes can be explored, we focused on the optimization of the main architecture and training parameters of the CNN and used an input size similar to other classification

networks (Krizhevsky et al. 2012; He et al. 2016; Xie et al. 2017). We used color images, i.e., each image had three channels: red, green, and blue (RGB). The RGB channels were 2-D matrices (225×225) with each element having a 0–1 pixel value indicating the color intensity. A common pre-processing step for the input images to CNN models is normalizing the images to have a mean value close to zero and a standard deviation close to one, which helps the model to converge faster (LeCun et al. 2012). The normalization was applied to each channel of the input image as

$$x_{\text{normalized}} = \frac{x - \mu}{\sigma}, \quad (3)$$

Fig. 5 SSIs from a fractured core section with overlapping. The process overlapped in the fractured region and the resulting images show the fracture at different locations

Fig. 6 Illustration of the data augmentation method used to increase the number of images in the non-intact core class and equalize the number of images in both classes. Horizontal flip mirrors the original image with respect to the horizontal axis, while vertical flip mirrors the original image with respect to the vertical axis, generating two extra images

where x is the input channel (R, G, or B), $x_{\text{normalized}}$ is the output channel after normalization, and μ and σ are the mean and standard deviation of the channel, respectively. To accomplish this, we calculated the mean and standard deviation for each color channel of our training images. The means were 0.546, 0.479, and 0.460 and the standard deviations were 0.189, 0.189, and 0.185 for the R, G, and B channels, respectively.

4 Methodology

4.1 CNN Model Development

4.1.1 Optimization of Hyperparameters

To apply a CNN model to the task at hand, parameters related to the CNN architecture and the optimization procedure must be chosen; these are typically referred to as hyperparameters. Although expertise and intuition can be used to select hyperparameters, we determined the best hyperparameters using the Hyperopt library (Bergstra et al. 2015). The library includes optimization algorithms, such as random search (RS) and a tree of Parzen estimators (TPE). RS is a simple optimization algorithm that, during an optimization trial, randomly selects the next hyperparameter to be evaluated without consideration of results from previous trials. TPE algorithm, on the other hand, takes into account observations from previous trials and uses them to predict the next hyperparameter to be evaluated (Bergstra et al. 2011). We tested the following features to optimize the model: (1) the architecture of the model, (2) the learning process, and (3) regularization methods. These CNN characteristics were introduced in Sects. 2.1 and 2.2.

1. The tested architecture hyperparameters included: the number of convolutional layers, the number of kernels per convolutional layer, and the kernel size in each con-

Table 3 Hyperparameters included in CNN model optimization using Hyperopt (Bergstra et al. 2015)

	Hyperparameter	Range/choices	Best result
Architecture-related hyperparameters	Number of convolutional layers	Three, Four	Four
	Number of kernels F1–F2–F3–F4	16–32–64–128, 24–48–96–192, 32–64–128–256	16–32–64–128
	Kernel size in first layer	9×9, 7×7, 5×5	7×7
	Kernel size in second layer	5×5, 3×3	3×3
	Optimizer	SGD, Adam	SGD
Optimization technique	Initial learning rate	0.01–0.00001	0.0009
	Dropout probability	0–0.8	0.8
Regularization-related hyperparameters	Weight decay	0.1–0.000001	0.01136
	Batch normalization (BN)	With BN, Without BN	With BN

convolutional layer. The number of kernels determines the number of feature map outputs by each convolutional layer, while the size of the kernel defines the area of the scanning field, and thus controls the size and shape of the features to be extracted; for example, large kernels can extract edges and small kernels extract finer features, such as texture.

2. The learning process-related hyperparameters assessed the type of optimization method, including two common optimizers: the SGD and Adam, and the learning rate used by the optimizer.
3. Regularization hyperparameters tested included: the weight decay, the dropout probability in the fully connected layer, and whether to use batch normalization layers.

All the tested hyperparameters are summarized in Table 3. Hyperopt optimization included a total of 145 trials, starting with 15 trials using RS and the remainder using TPE. The optimization yielded a combination of hyperparameters that provided the best classification performance on the validation data set, i.e., yielding the minimum CE loss. The selected hyperparameters are shown in column 4 of Table 3.

4.1.2 Model Architecture and Training

The architecture of the CNN model was determined according to the best results from the hyperparameter optimization (see Table 3). We created a CNN model with four convolutional layers, one fully connected layer, and an output layer. A ReLU activation function was used after the convolutional layers and the fully connected layer. Each convolutional layer was followed by a max pooling layer. The size and number of kernels in each convolutional layer are detailed in the model architecture shown in Fig. 7.

Fig. 7 CNN architecture used in the study. The model consists of four convolutional layers (C1, C2, C3, and C4), one fully connected layer, and an output layer. Each convolution (Cn) shows the size of its kernels as width×height×depth. Max pooling layers were

Fig. 8 Training loss function. The loss obtained from Eq. (2) at each training epoch is shown; the loss was minimized, and the model converged

The final model was trained on the entire training set of 6400 images, using the best training-related hyperparameters: the optimizer, learning rate, and weight decay. During each iteration, a mini-batch of 32 images was fed into the network. The loss was calculated according to Eq. (2), and the errors were backpropagated through the network. We used the SGD (stochastic gradient descent) to update the model parameters. An initial learning rate of 0.0009 was used, with a weight decay of 0.01136. The model was trained for 280 epochs; an epoch was completed when all images from the training set were seen by the network, which for our training set included a total of 200 mini batches per epoch. The loss at each training epoch is shown in Fig. 8, indicating the convergence of the model toward the end of the training as the loss was minimized.

inserted between convolutional layers. Input images to the model are 225×225 pixels with red, green, and blue channels. Outputs are probability scores of the input image belonging to the intact core and non-intact core classes

Fig. 9 Schematic diagram of the developed workflow showing the steps of image preparation, CNN model development, and their integration to obtain RQD. The code for this workflow is provided in Sect. 4.1.3

4.1.3 Software and Hardware

Our study used the Python programming language (3.8) (van Rossum and de Boer 1991). Pytorch (1.5) (Paszke et al. 2019) and Skorch (0.7) (Tietz et al. 2017) were used for machine learning implementation. Other supporting packages/libraries were used for image processing and computation: Scikit-learn (0.23) (Pedregosa et al. 2011), NumPy (1.19) (Oliphant 2006), OpenCV (4.0) (Bradski 2000), OCR (0.3) (Smith 2007), and Hyperopt (0.2) (Bergstra et al. 2015). The training of the CNN used a graphics processing unit (GPU–NVIDIA GeForce GTX 1080 Ti) with 11 GB of memory; the testing was performed using a central processing unit (CPU).

The code for the entire workflow is summarized in Fig. 9, and all algorithms used are available at <https://github.com/fatimahgit/CNN-for-RQD-calculation/tree/main>.

4.2 RQD Calculation and Evaluation

RQD was calculated according to Deere and Deere (1988) as

$$RQD = \frac{\sum L_c \geq 10\text{cm}}{L_t} \times 100\%, \quad (4)$$

where L_c is the length of the core pieces and L_t is the total core length. L_t is preferable to be 1.5 m, as recommended by Deere and Deere (1988). Each input image was divided

into rows using the procedure described in Sect. 3.1. A small window slides along each row with a step of 100 pixels (~7 mm) to ensure enough overlap to detect an intact core using the CNN model. As the window scans the core, the model detects intact core pieces and calculates their length in pixels. The length in pixels was converted to centimeters by automatically detecting the scale presented at the bottom of each image (e.g., Figs. 2 and 4a). The total core length in each row was calculated as the difference between the start and end depths for each row. Many of the total core lengths in our data sets, and thus L_t in Eq. (4), were approximately 1 m. The results of the model were validated against ground truth RQD values. The ground truth values were obtained by manually measuring the lengths (in pixels) of the core pieces in the row, using the Straight Line measuring tool in ImageJ software (Schneider et al. 2012). To identify core pieces longer than 10 cm, we converted the lengths from pixels to centimeters using the scale on the core image.

To evaluate the accuracy of the proposed method, we calculated the error percentage between the manual and the model results of RQDs as,

$$\text{Error percentage} = 100 \times \frac{|RQD_{\text{manual}} - RQD_{\text{model}}|}{RQD_{\text{manual}}}. \quad (5)$$

Fig. 10 CNN visualization using the guided backpropagation method (Springenberg et al. 2014). Each column shows sample images from test data set 1 that share similar features (left) and their output feature

maps (right). The model trained to extract different representative features in the images from intact core images (column 1) and non-intact core images (columns 2–5)

5 Results and Discussion

5.1 Performance of Trained CNN Model

Before using the model for the RQD calculation, we assessed the model accuracy in classifying intact and non-intact cores for the 655 SSIs in test data set 1 (Table 2). The classification accuracy was calculated as the percentage of correctly classified images relative to the total number of test images. The test data produced an overall accuracy of 98%, with class accuracy of 95% and 99% for the intact core class and non-intact core class, respectively.

In addition to the classification accuracy, the performance of the model on test data set 1 was evaluated by visualizing the features learned by the model. We visualized the feature maps produced from the first convolutional layer of the model for several test images using the backpropagation visualization method (Springenberg et al. 2014). The images were selected to have different features that can be used by the model to identify an intact core or non-intact core image. The input images and the produced feature maps are shown in Fig. 10. As shown in the figure, the model was able to recognize rock texture from images in the first column, fractures with different angles and positions from images in the second column, crushed core edges from images in the third column, wooden blocks appearing in the fourth column, and empty tray regions in the last column. The results demonstrate that the model correctly learned representative features from the training images and was able to analyze new core images by differentiating between intact and non-intact cores.

Fig. 11 Error analysis for RQD calculations from test data set 2. The results were based on 540 sandstone rows (~1 m long). The model produced a low average error; most calculations had an error rate of less than 3%

5.2 RQD calculation results

To assess the performance of the CNN-based method in the RQD calculation, we used test data sets 2 and 3, which included 92 tray images of sandstone and 15 images of limestone, respectively. As detailed in Sect. 4.2, the RQD was calculated for each row of the core trays and validated through manual calculation. The next sections discuss the results from both data sets, the overall results, and the main reasons for misclassification.

Fig. 12 Visualization of model performance with examples from test data set 2: **a** dark homogeneous sandstone; **b** sandstone with veins and bedding. Each subfigure shows an input image (left) and the out-

put of the model (right). The model distinguished intact cores (highlighted in green for cores ≥ 10 cm and in purple for cores < 10 cm) from non-intact cores (non-highlighted area)

5.2.1 RQD Results from Sandstone Test Data Set

The model produced an average error rate of 2.58% for the 540 sandstone rows (92 trays) of test data set 2 (shown in Table 2). A histogram of the error rates for all rows is shown in Fig. 11. Approximately 76% of the rows had error rates less than 3%; 13% of the rows had error rates of 3–6%; only 11% of the rows had error rates greater than 6% (including less than 2% with error rates greater than 18%).

Visual inspection of the results indicated that the model was able to successfully detect intact core pieces and identify their lengths while excluding fractures, crushed rock, empty tray areas, and other non-rock objects. Tested sandstone trays are shown in Fig. 12. The intact core pieces were correctly recognized and highlighted according to their length, and other parts of the tray were eliminated. Overall, the model performed well on the test images of sandstone with different colors and textures, as shown in Fig. 12a, b. The model was able to recognize the intact core in Fig. 12b despite the presence of veins, lamination, and thin bedding. In addition, some trays in the data set

Fig. 13 Error analysis for RQD calculations from test data set 3. The results were based on 90 limestone rows (~1 m long). The model produced a low average error; most calculations had an error rate of less than 3%

Fig. 14 Backpropagation visualization of filters from the first convolutional layer for tested limestone images. Input images (first row) and their corresponding feature maps (second row) show that the

model captured representative features from intact cores (images 1–3), and fractured and crushed cores (images 4–7), which are essential for the intact core vs non-intact core classification

Fig. 15 Relationship between ground truth RQDs and estimated RQDs for 540 sandstone rows and 90 limestone rows (~1 m long); most points had an error rate less than $\pm 10\%$

contained half cores (see Table 1); these cores are susceptible to mechanical fracturing. The proposed method is data-driven meaning that all fractures are counted, and thus the network does not distinguish between natural and mechanical fractures.

5.2.2 RQD Results from Limestone Test Data Set

Although the trays in test data set 3 (shown in Table 2) included a different rock type (limestone) from that used for training (sandstone), the results agreed with the manually calculated RQDs. For a total of 90 rows (15 trays), the average error rate was 3.17%. As shown in Fig. 13, the error was less than 3% for 70% of the rows; the error was 3–12% for approximately 24% of the rows, and greater than 12% for less than 6% of the rows. The results

demonstrate the robustness of the proposed model and its ability to generalize new images. This was also confirmed by visualizing output filters of test images from the first convolutional layer of the model using the guided backpropagation method (as described in Sect. 5.1). An activated filter for each input image is shown in Fig. 14. Although the model was trained with sandstone images, it was able to capture universal features, such as intact cores, fractures, and crushed rock from the limestone images.

5.2.3 Overall Results

The results from test data sets 2 and 3 demonstrate that the CNN-based method recognizes intact core pieces of new unseen rock and can estimate the RQD with high accuracy. Many of the calculations agreed with the manually calculated RQDs, as shown in Fig. 15. For 94% of the tested sandstone and limestone rows (593 rows), the error rate was less than 10%; only 6% of the rows had an error greater than $\pm 10\%$. Because the majority of the points (90%) were located in the high-RQD area ($RQD > 50\%$), more outliers were located in this area, as shown in Fig. 15; however, the outliers only represented approximately 5% of the high-RQD rows.

We conclude that the CNN provides reliable core image analysis required for RQD calculation, and thus can accurately predict the RQD. The proposed method processed 10 m of core in less than 1 min using a CPU.

5.2.4 Model Misclassification and Limitations

Misclassification by the CNN model occurred in a few cases: (1) intact cores with natural dark lines or black handwritten numbers/lines were occasionally misinterpreted as non-intact cores; (2) fine cracks were not always recognizable by the model; and (3) some rusty or dirty tray areas covered with rock dust were misclassified as intact

Fig. 16 Examples of misclassification by the model: intact cores with a natural dark deposit (a) and a black line (b) were unrecognized by the model

cores. In general, the presence of joints did not introduce major misclassification issues; the model was trained to classify core pieces with joints as non-intact, as recommended by Deere and Deere (1988). Examples of misclassification are shown in Fig. 16. In the circled region (a), an intact core with a natural dark deposit was detected as non-intact. In the circled region (b), a core with a black line was misinterpreted as non-intact.

Intact cores with text/line marks or dark veins appeared less often than clean and homogeneous cores in the trays used for training; thus, they represented a small fraction of the intact cores used for training. Misclassifications could be minimized by including more images of cores with dark lines, veins, fine fractures, and dirty trays in the training images. More sophisticated data-augmentation algorithms could also be used to further expand the training set.

The CNN model was trained with core images of sandstone and validated with images of sandstone and limestone.

The performance of the model on other rock types, such as shale or coal, was not investigated in this study. The model could be further developed by expanding the rock types in the training images and using a deeper CNN model with more convolutional layers. Transfer learning could also be applied using our trained model to initialize the network parameters and retrain it using a new training set with more images or classes.

6 Conclusion

A CNN-based method was proposed for automatic drill-core characterization from 2-D core tray images to produce less subjective results, provide a means to assess historical databases, and reduce the time and cost associated with routine RQD calculations and rock mass classification. A CNN model was trained using a set of 6400 training images obtained from sandstone core trays from different boreholes. The method was applied to automatically calculate the RQD by identifying intact core fragments and eliminating fractures, crushed cores, empty tray regions, and other objects present in the core trays. The RQD was calculated for each row in the core tray, resulting in an average error rate of less than 3% for new unseen sandstone tray images. In addition, the proposed method yielded accurate results when applied to limestone tray images, with an average error rate of approximately 3%. The proposed approach promises an affordable, fast, and robust RQD calculation procedure that can be easily adopted and developed for further core logging automation. It offers remote and flexible core logging that does not require technical expertise. It relies on extracting geotechnical information from 2-D core tray images, which enables field assessment or reassessment using existing data sets without requiring access to old cores that may be damaged or lost. This method provides a new way to automatically analyze and quantify core data repositories, such as those available through Geoscience Australia and similar agencies.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source,

provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ali A, Shamsuddin SM, Ralescu AL (2015) Classification with class imbalance problem: a review. *Int J Adv Soft Comput Appl* 7:176–204
- Alom MZ, Taha TM, Yakopcic C et al (2018) The history began from AlexNet: a comprehensive survey on deep learning approaches. [arXiv:1803.01164](https://arxiv.org/abs/1803.01164)
- Alqahtani N, Alzubaidi F, Armstrong RT et al (2020) Machine learning for predicting properties of porous media from 2d X-ray images. *J Pet Sci Eng* 184:106514. <https://doi.org/10.1016/j.petrol.2019.106514>
- Alzubaidi F, Mostaghimi P, Swietojanski P et al (2021) Automated lithology classification from drill core images using convolutional neural networks. *J Pet Sci Eng* 197:107933. <https://doi.org/10.1016/j.petrol.2020.107933>
- Bao S, Chung ACS (2018) Multi-scale structured CNN with label consistency for brain MR image segmentation. *Comput Methods Biomech Biomed Eng Imaging vis* 6:113–117. <https://doi.org/10.1080/21681163.2016.1182072>
- Barton N, Lien R, Lunde J (1974) Engineering classification of rock masses for the design of tunnel support. *Rock Mech* 6:189–236. <https://doi.org/10.1007/BF01239496>
- Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyperparameter optimization. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger KQ (eds) *Advances in neural information processing systems*, vol 24. Curran Associates, Inc., Granada, Spain
- Bergstra J, Komer B, Eliasmith C et al (2015) Hyperopt: a Python library for model selection and hyperparameter optimization. *Comput Sci Discov* 8:0–24. <https://doi.org/10.1088/1749-4699/8/1/014008>
- Bieniawski ZT (1973) Engineering classification of jointed rock masses. *Trans S Afr Inst Civ Eng* 15:335–343. https://doi.org/10.1052/AJA10212019_17397
- Bieniawski ZT (1989) *Engineering rock mass classifications: a complete manual for engineers and geologists in mining, civil, and petroleum engineering*. Wiley, New York
- Bradski G (2000) The OpenCV library. *Dr Dobb's J Softw Tools* 25:120–125
- Chen Q, Yin T (2019) Should the use of rock quality designation be discontinued in the rock mass rating system? *Rock Mech Rock Eng* 52:1075–1094. <https://doi.org/10.1007/s00603-018-1607-x>
- Cheng G, Guo W (2017) Rock images classification by using deep convolution neural network. *IOP Conf Ser J Phys Conf Ser* 887:12089. <https://doi.org/10.1088/1742-6596/887/1/012089>
- Deere DU, Deere DW (1988) The rock quality designation (RQD) index in practice. *Rock Classif Syst Eng Purp ASTM STP* 984:91–101. <https://doi.org/10.1520/STP48465S>
- Deere DU, Hendron AJ, Patton FD, Cording EJ (1967) Design of surface and near surface constructions in rock. In: *Failure and breakage of rock: proceedings of the eighth symposium on rock mechanics*. American Institute of Mining, Metallurgical, and Petroleum Engineers, Minneapolis, Minnesota, pp 237–302
- Geological Survey of South Australia (GSSA) (2020) Hylogger data. http://www.energymining.sa.gov.au/minerals/geoscience/geoscientific_data/hylogger. Accessed 12 Apr 2020
- Goodfellow I, Bengio Y, Courville A (2016) *Deep learning*. MIT Press, Cambridge
- Gu J, Wang Z, Kuen J et al (2018) Recent advances in convolutional neural networks. *Pattern Recognit* 77:354–377. <https://doi.org/10.1016/j.patcog.2017.10.013>
- Hadjigeorgiou J (2012) Where do the data come from? *Minist Technol* 121:236–247. <https://doi.org/10.1179/1743286312y.0000000026>
- Hancock EA, Huntington JF (2010) The GSWA NVCL HyLogger: rapid mineralogical analysis for characterizing mineral and petroleum core
- Harraden C, Berry RF, Lett J (2016) Proposed methodology for using automated core logging technology to extract geotechnical index parameters. In: *Proceedings of 3rd international geometallurgy conference*
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: *2016 IEEE conference on computer vision and pattern recognition*. IEEE, Las Vegas, NV, USA, pp 770–778
- He K, Gkioxari G, Dollar P, Girshick R (2017) Mask R-CNN. In: *2017 IEEE international conference on computer vision*. IEEE, pp 2961–2969
- Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *32nd international conference on machine learning ICML 2015*, vol 1, pp 448–456
- Iskandar D, Alghazo J, Butt MM et al (2018) Deep CNN based MR image denoising for tumor segmentation using watershed transform. *Int J Eng Technol* 7:37–42. <https://doi.org/10.14419/ijet.v7i2.3.9964>
- Julian D (2018) *Deep learning with pytorch quick start guide: learn to train and deploy neural network models in Python*, 1st edn. Packt Publishing Ltd, Birmingham
- Ketkar N (2017) *Deep learning with python: a hands-on introduction*, 1st edn. Apress, Berkeley, CA
- Kiefer J, Wolfowitz J (1952) Stochastic estimation of the maximum of a regression function. *Ann Math Stat* 23:462–466. <https://doi.org/10.1214/aoms/1177729392>
- Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. In: *International conference learning representation*
- Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst* 25:1097–1105
- Krogh A, Hertz JA (1992) A simple weight decay can improve generalization. In: Lippmann JM and SH and RP (ed) *Advances in neural information processing systems*. Morgan-Kaufmann
- LeCun YA, Bottou L, Orr GB, Müller K-R (2012) *Efficient Back-Prop*. Springer, Berlin, pp 9–48
- LeCun Y, Bengio Y (1995) Convolutional networks for images, speech, and time-series. In: Arbib MA (ed) *The handbook of brain theory and neural networks*. MIT Press
- Leevy JL, Khoshgoftaar TM, Bauder RA, Seliya N (2018) A survey on addressing high-class imbalance in big data. *J Big Data* 5:1–30. <https://doi.org/10.1186/s40537-018-0151-6>
- Lemy F, Hadjigeorgiou J, Côté P, Maldague X (2001) Image analysis of drill core. *Min Technol* 110:172–177. <https://doi.org/10.1179/mnt.2001.110.3.172>
- Liang Z, Powell A, Ersoy I, et al (2016) CNN-based image analysis for malaria diagnosis. In: *2016 IEEE international conference on bioinformatics and biomedicine (BIBM)*. IEEE, pp 493–496

- Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: The 27th international conference on machine learning
- Nwankpa C, Ijomah W, Gachagan A, Marshall S (2018) Activation functions: comparison of trends in practice and research for deep learning. [arXiv:1811.03378](https://arxiv.org/abs/1811.03378)
- Oliphant TE (2006) Guide to NumPy. Trelgol Publishing
- Olson L, Samson C, McKinnon SD (2015) 3-D laser imaging of drill core for fracture detection and rock quality designation. *Int J Rock Mech Min Sci* 73:156–164. <https://doi.org/10.1016/j.ijrmms.2014.11.004>
- Paszke A, Gross S, Massa F et al (2019) PyTorch: an imperative style, high-performance deep learning library. *Adv Neural Inf Process Syst* 32:8026–8037
- Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. *J Mach Learn Res* 12:2825–2830
- Ramachandran P, Zoph B, Le QV (2017) Searching for activation functions. [arXiv:1710.05941](https://arxiv.org/abs/1710.05941)
- Robbins H, Monro S (1951) A stochastic approximation method. *Ann Math Stat* 22:400–407
- Saricam T, Ozturk H (2018) Estimation of RQD by digital image analysis using a shadow-based method. *Int J Rock Mech Min Sci* 112:253–265. <https://doi.org/10.1016/j.ijrmms.2018.10.032>
- Schneider CA, Rasband WS, Eliceiri KW (2012) NIH Image to ImageJ: 25 years of image analysis. *Nat Methods* 9:671–675. <https://doi.org/10.1038/nmeth.2089>
- Shu L, Osinski GR, McIsaac K, Wang D (2018) An automatic methodology for analyzing sorting level of rock particles. *Comput Geosci* 120:97–104. <https://doi.org/10.1016/j.cageo.2018.08.001>
- Smith R (2007) An overview of the tesseract OCR engine. In: Ninth international conference on document analysis and recognition (ICDAR 2007), vol 2. IEEE, pp 629–633
- Sonoda S, Murata N (2017) Neural network with unbounded activation functions is universal approximator. *Appl Comput Harmon Anal* 43:233–268. <https://doi.org/10.1016/j.acha.2015.12.005>
- Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2014) Striving for simplicity: the all convolutional net. [arXiv:1412.6806](https://arxiv.org/abs/1412.6806)
- Srivastava N, Hinton G, Krizhevsky A, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15:1929–1958
- Steger C (1998) An unbiased detector of curvilinear structures. *IEEE Trans Pattern Anal Mach Intell* 20:113–125. <https://doi.org/10.1109/34.659930>
- Tietz M, Nouri D, Bossan B (2017) Skorch documentation. <https://skorch.readthedocs.io/en/stable/index.html#>. Accessed 08 July 2019
- van Rossum G, de Boer J (1991) Interactively testing remote servers using the Python programming language. *CWI Q* 4:283–304
- Wang J, Yang Y, Mao J, et al (2016) CNN-RNN: a unified framework for multi-label image classification. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 2285–2294
- Wang YD, Armstrong RT, Mostaghimi P (2019) Enhancing resolution of digital rock images with super resolution convolutional neural networks. *J Pet Sci Eng*. <https://doi.org/10.1016/j.petrol.2019.106261>
- Xie S, Girshick R, Dollár P et al (2017) Aggregated residual transformations for deep neural networks. In: Proceedings—30th IEEE conference computer vision and pattern recognition, CVPR 2017 2017-January, pp 5987–5995. <https://doi.org/10.1109/CVPR.2017.634>
- Zhang K, Zuo W, Chen Y et al (2017a) Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising. *IEEE Trans Image Process* 26:3142–3155. <https://doi.org/10.1109/TIP.2017.2662206>
- Zhang PY, Sun JM, Jiang YJ, Gao JS (2017b) Deep learning method for lithology identification from borehole images. 79th EAGE conference exhibition, 2017b, pp 12–15. <https://doi.org/10.3997/2214-4609.2017b00945>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.