Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-14 (2019) 105–110

Deep Convolutional Neural Network for Mill Feed Size Characterization ⋆

Laurentz E. Olivier *,*** **Michael G. Maritz** ** **Ian K. Craig** ***

* *Moyo Africa, Centurion, South Africa (e-mail: laurentz.olivier@moyoafrica.com).*
** *University of Pretoria, Pretoria, 0002, South Africa (e-mail: u15016359@tuks.co.za).*
*** *University of Pretoria, Pretoria, 0002, South Africa (e-mail: ian.craig@up.ac.za).*

Abstract: Knowing the characteristics of the feed ore size is an important consideration for operations and control of a run-of-mine ore milling circuit. Large feed ore variations are important to detect as they require intervention, whether it be manual by the operator or by an automatic controller. A deep convolutional neural network is used in this work to classify the feed ore images into one of four classes. A VGG16 architecture is used and the classifier is trained making use of transfer learning.

Keywords: Deep learning, convolutional neural network, transfer learning, milling, run-of-mine ore.

1. INTRODUCTION

Variations in run-of-mine ore properties such as size, composition, and grindability strongly affect autogenous grinding (AG) and semi-autogenous grinding (SAG) mill performance (Tessier et al., 2007). When employing a control strategy that aims to suppress external disturbances, the size of the disturbance entering the system is valuable information (Coetzee et al., 2010; Olivier et al., 2012a). This work focuses on characterizing the feed ore size distribution.

The WipFrag system (Maerz et al., 1996) is a commercially available tool for characterizing the mill feed based on images of the feed ore. Palangio and Maerz (1999) shows some case studies of successful system implementations. The WipFrag system makes use of traditional computer vision algorithms and a calculation known as the Rosin-Rammler equation to calculate the cumulative percentage of particles passing a specific size (Ko and Shang, 2011).

Feed characterization using image analysis in conjunction with a neural network was proposed by Hamzeloo et al. (2014). In that work traditional computer vision techniques are used for feature extraction, but a neural network is used to estimate the particle size distribution. This is to overcome the potential inaccuracies that may arise with using a calculation like the Rosin-Rammler equation, as calibration of the equation parameters is important for its effective use.

A similar approach was proposed by Ko and Shang (2011). Both Hamzeloo et al. (2014) and Ko and Shang (2011) used traditional "shallow" networks with only fully connected layers.

To overcome the need for feature extraction deep convolutional neural networks (CNN) have recently been employed (Schmidhuber, 2015). These networks are very efficient at extracting features for classification and regression tasks, to the point where they can achieve superior performance to the combination of traditional feature extraction in combination with fully connected neural networks.

Deep CNNs have only recently been applied in the process industries. Fu and Aldrich (2019) shows how deep CNNs can be used to characterize flotation froth, and Wang and Liu (2018) shows how a stacked auto-encoder deep neural network is used for soft sensing of air pre-heater rotor deformation. In this work, a deep CNN is used to characterize the feed size distribution from feed ore images.

The importance and context of feed ore size characterization is described in Section 2. An overview of CNNs and the method of training used in this work is given in Section 3. Section 4 shows the data preparation, training, and results. Section 5 concludes the work.

2. MILL FEED SIZE CHARACTERIZATION

Run-of-mine (ROM) ore milling circuits deal with a wide possible range of feed sizes and distributions. By definition the ROM ore is not pre-processed before entering the milling circuit. This means that operations need to be robust enough to deal with large feed disturbances while producing an output of consistent quality.

When flotation is the next step in the metallurgical extraction process, there is usually a specific output particle size at which the flotation circuit performs optimally (Wei and Craig, 2009).

Having information about the feed size and distribution is therefore important for consistent production. It is in

⋆ This work is based on research supported in part by the National Research Foundation of South Africa (Grant number 111741).

10.1016/j.ifacol.2019.09.172

some cases possible to get a sense of the feed size from the operational data through parameter estimation (Olivier et al., 2012b), but a more direct approach is to monitor a video stream of the feed and inferring the size and distribution using computer vision techniques.

Having information available regarding the feed size, or when the feed size changes significantly, the appropriate adjustments can be made in the process. These adjustments are either made by the operator or an appropriate automatic controller.

Le Roux et al. (2013) developed a validated model that has been shown to be appropriate for model-based control. In that work it is remarked that for control purposes, the important feed size distribution parameters are the fraction of fines in the feed ore and the fraction of rocks in the feed ore.

In this context the term "rocks" has the specific meaning as set out by Le Roux et al. (2013), which describes ore that cannot exit the mill through the end discharge screen. "Fines" describes ore that is smaller than the product specification size, and are essentially inseparable from water in the milling circuit.

In this work the input image of the feed ore is classified into one of four categories, namely:

I Ore containing very little to no large rocks,
II Ore containing some rocks,
III Ore containing quite a few large rocks, and
IV Ore containing mostly large rocks.

These size classifications are somewhat subjective, but consistent labels lead to classes that the CNN must still be able to discern. These classes do not provide the fraction of rocks in the feed ore directly, but the classification goes some distance in proving that a deep CNN can be used to characterize the feed ore by size. This is an important stepping stone towards finally obtaining the fraction of rocks in the feed ore.

3. DEEP CONVOLUTIONAL NEURAL NETWORKS

The first real convolutional network seems to be that described by Fukushima (1980). The term CNN and the structures we know today were introduced by Yann LeCun in the late 1980s and early 1990s (see e.g. LeCun and Bengio (1995)). CNNs evolved into deep networks over time; Schmidhuber (2015) provides an overview of the development of deep learning in neural networks. A good overview of CNNs specifically, including how they work and their common layers, is provided by LeCun et al. (2015).

Fig. 1 shows a simple convolutional neural network layout to illustrate the common network elements.

As the name implies, convolutional layers apply a convolution operation to the input image. This convolution operation is the core concept that drives the efficacy of CNNs and overcomes the impracticality of fully connected feedforward neural networks for image analysis tasks. It reduces the number of free parameters that need to be trained, and also maintains some spacial information of pixels in a region of the image. The output of the convolutional layer is a feature map of numerical values, often with a different shape than the input image.

Another type of layer common in CNNs is a pooling layer. Pooling combines the outputs of a cluster on neurons in one layer into a single neuron in the next layer. Max pooling, for example, takes the maximum value from a cluster of neurons to the next layer.

A feature map can then be flattened, i.e. converting a higher dimensional array into a 1-D array, such that the feature values can be passed to a fully connected layer (or layers). These fully connected layers are the same in principle to the layers in a traditional multi-layer perceptron neural network.

Many other layer types exist in CNNs, such as recurrent, embedding, normalization, and locally connected layers. The ones shown in Fig. 1 are however arguably the most common in CNNs, and are the most relevant to the current work.

Most of the earlier layers in a CNN are devoted to feature extraction while the latter layers are devoted to decoding these features into the output space (whether it be for classification or regression).

3.1 Network layouts

Much research has gone into developing architectures for functional deep CNNs. Formulating the correct sequence of layers is not necessarily an easy task, which is why many implementations use standard CNN architectures (see e.g. Fu and Aldrich (2019)).

Common architectures include AlexNet, Inception, and ResNet. The solution presented in this work makes use of the VGG16 network architecture (Simonyan and Zisserman, 2014). The layout used here is similar to Simonyan and Zisserman (2014) but the input image dimensions are different and hence the feature maps have different dimensions. The VGG16 layout with the dimensions used in this work is shown in Fig. 2.

3.2 Transfer learning

In order to accomplish appropriate feature extraction before the fully connected network layers, CNNs can become very large in size. Fig. 2 shows the VGG16 network architecture.

More layers imply more network parameters, and as the network becomes very deep there can be millions of parameters that need to defined through training. VGG16, for example, has in the order of 138 million parameters. Such a large number of parameters consequently means that a very large number of training images is required to train the network from scratch.

Training a network with an architecture like that of VGG16 from scratch for each classification task would render the use of deep CNNs impractical. A network that was trained for a specific classification task can however be re-purposed for another task through a process called transfer learning. Pan and Yang (2010) notes that transfer learning often works across different domains of interest, feature spaces, and data distributions.

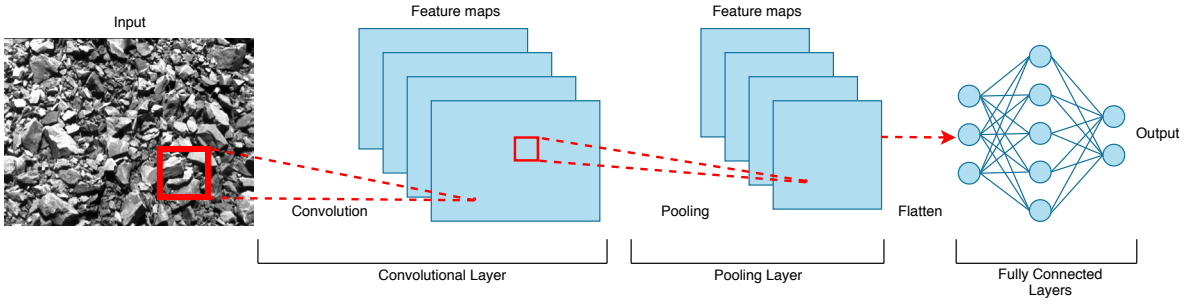

Fig. 1. Convolutional neural network layers.

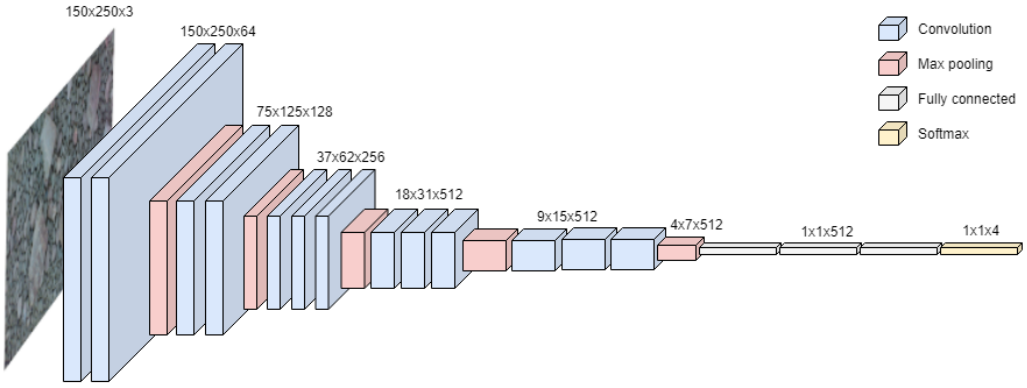

Fig. 2. VGG16 architecture.

In deep CNNs the process of transfer learning often involves loading a model pre-trained for one task, and then re-training the final layers of this network using the training data available for the task at hand.

In this work, a VGG16 model trained on ImageNet is used, and only the final fully connected layers are re-trained. ImageNet (Deng et al., 2009) is a large database of labelled images that is commonly used for training and testing of image classification algorithms.

The dimensions shown in Fig. 2 are for the input images and subsequent network layers used in this work.

4. DEEP CNN MILL FEED SIZE CHARACTERIZATION

The deep CNN is implemented using Keras (Chollet et al., 2015) with the Tensorflow (Abadi et al., 2016) back-end.

4.1 Input data

Input data was captured using a model conveyor belt with a vertically mounted camera capturing images from above, similar to the representation shown in Fig. 3. All images were captured under similar lighting conditions. Fig. 4 shows one example image from each class.

Images were captured for batches of ore conforming to each of the four defined classes, and labelled according to the class in question. In total 223 images were captured. Table 1 shows the number of images per category.

Images are normalized such that all pixel values lie in the range $[0, 1]$.

Fig. 3. Experimental setup.

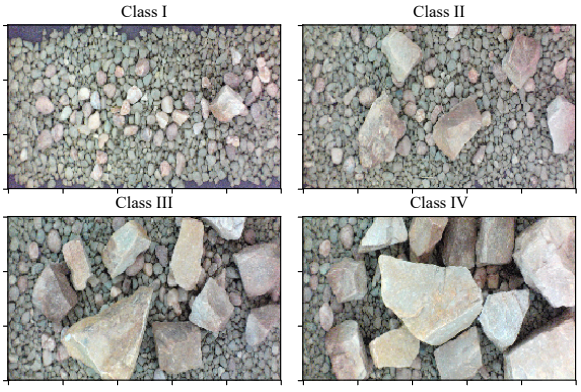

Fig. 4. One example image from each class.

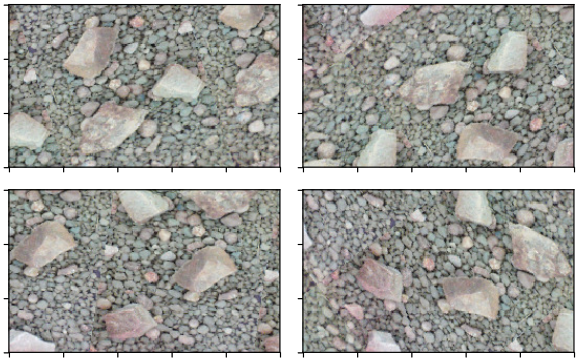

Fig. 5. Example of creating four augmented images from one input image.

4.2 Data augmentation

Data augmentation is a process by which training data can be up-sampled to improve network accuracy. Data augmentation helps with increasing the number of training samples, but more importantly it helps with the generalization of image features to help prevent over-fitting. In the present example, it is not desirable for the network to be sensitive to a large rock in a specific region of the image, as the network output should be affected by a large rock anywhere in the image.

Image augmentation could involve rotation, translation, flipping an image horizontally or vertically, brightness adjustments, and rescaling of input images. In the present example the first three options are applicable. Because the objective of this work is to characterize inputs based on size, rescaling images would defeat the purpose.

Fig. 5 shows how a single input image has been augmented to produce four images that may now be used for training. These variations are created from the same input image, which is even somewhat difficult to confirm with the naked eye. The image augmentation parameters are set up such that the input image may be flipped horizontally, rotated by as much as 90 degrees, translated horizontally or vertically by up to half of the width or height of the image, or any combination of these operations. When the image is translated, the empty space is filled by wrapping the image around the frame.

4.3 Separating training and validation data

It is important to split the input sample images into a training and validation set. The training set is used for training and the validation set is used for the subsequent testing. This ensures that the model is validated on images it has not seen during training, which provides a much better indication of the performance one may obtain when the model is in operation. The validation images are also used to calculate the validation loss during training. Even though this value is not used during training, it is an important metric which shows if the network is being over-trained.

Table 1. Number of images per category

Category	Number of images
I	70
II	51
III	51
IV	51

It is important to stress that the validation data is not used in any way to train the CNN, it is merely used to track the model accuracy on unseen data as training progresses. Tracking the accuracy on validation data during training is also important for verification that the network is not over-fitting on the training data. If the validation accuracy is significantly lower than the training accuracy, it is a clear sign of over-fitting. The validation data is not further split into a distinct test set as would be required for hyper-parameter tuning (which does not form part of this work).

When separating the data, 20 % is held out for validation. The selection is done in a stratified manner based on the class. This means that 20 % of the images in each class will be held out for validation, such that the fraction of images in the validation set per category will be the same.

4.4 Training

The loss function in use is the categorical cross-entropy. It is also sometimes called a softmax loss, as it uses a softmax activation and a cross-entropy loss.

The CNN provides a set of probabilities $S \in [0,1]^{1\times N} = [s_1, \cdots, s_N]$ for the input to belong to each class, where N classes exist. The softmax activation function is then given by:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^N e^{s_j}}, \tag{1}$$

and the cross-entropy loss is defined as:

$$CE = -\sum_i^N t_i \log\left(f(s)_i\right) \tag{2}$$

where t_i is the ground-truth value for class i. In the multi-label classification problem addressed here, the ground-truth vector contains only zeros expect for the entry corresponding to the correct class, which is 1. This means that

$$t_i = \begin{cases} 0, & \forall \quad i \neq p \\ 1, & \forall \quad i = p \end{cases} \tag{3}$$

where p is the positive class. The categorical cross-entropy can then be simplified to be:

$$CE = -\log\left(\frac{e^{s_p}}{\sum_j^N e^{s_j}}\right). \tag{4}$$

Training is run for 50 epochs, with 32 batches per epoch, and 32 samples per batch (using augmented images). This means that $32 \times 32 = 1024$ samples are used per epoch. The accuracy obtained at the end of each epoch is shown in Fig. 6 and the categorical cross-entropy loss at the end of each epoch is shown in Fig. 7.

It is visible from Fig. 6 and Fig. 7 that the classifier training progresses well, with the training accuracy increasing into the high nineties, and the validation accuracy remaining close to 100 % from about 20 epochs. The training loss decreases well and seems to start flattening out after the 50 epochs. The validation loss is also continually decreasing, which indicates that the network is not over-fitting.

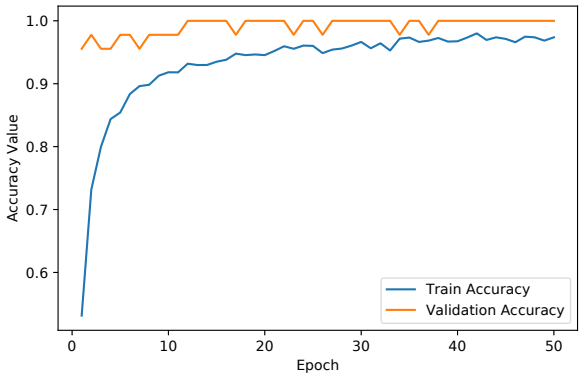

Fig. 6. Accuracy per epoch.

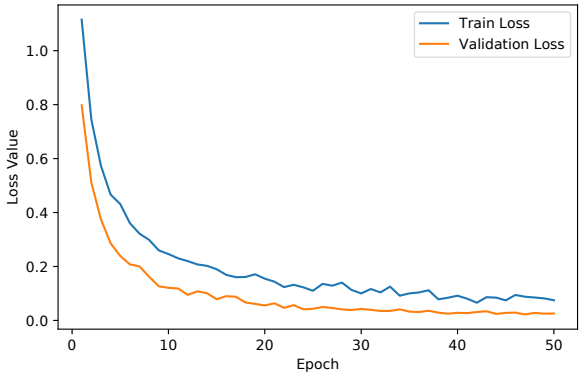

Fig. 7. Loss per epoch.

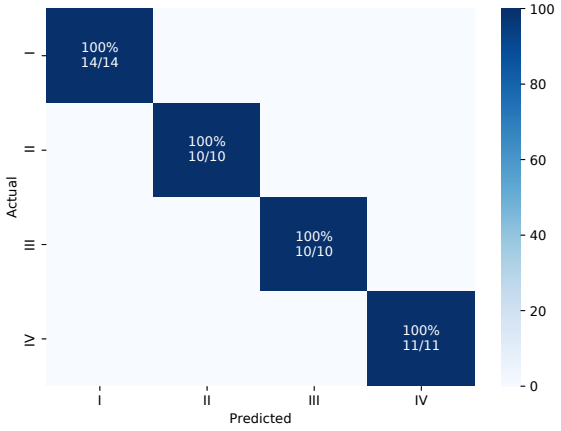

Fig. 8. Confusion matrix for validation images.

4.5 Classifier performance

To validate the classifier performance, the 45 validation images (which are images the classifier was not trained on) are classified and the predictions are compared to the ground-truth labels. The confusion matrix, which plots the actual label against the predicted label is shown in Fig. 8. The figure shows how many images from each class are contained in the validation set. It is visible from Fig. 8 that all 45 images are correctly classified.

This is a commendable result, but there are not that many samples and the test is not as stringent. It would be desirable to have the model work on a wide variety of images, and in this work the performance is subsequently tested using augmented validation images.

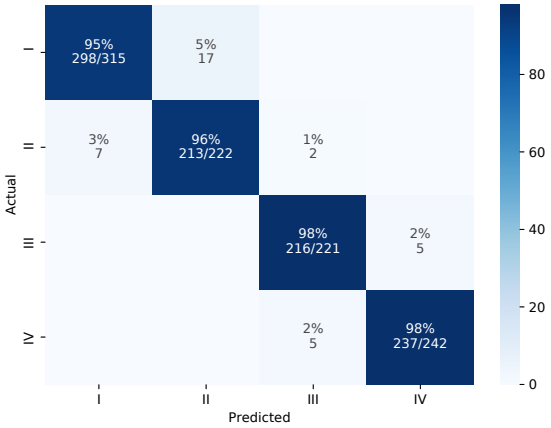

Fig. 9. Confusion matrix for augmented validation images.

The 45 validation images are augmented using the process described in Section 4.2 to create 1,000 samples from the original 45 on which the model was not trained. This is a much more stringent test and should go further in proving the classifier's worth.

Various metrics determine the classification performance of a multi-class classifier. The overall accuracy of the classifier, which is the total percentage of samples classified correctly, is 96.4 %. The precision, recall, F1-score, and number of images per class (support) is shown in Table 2.

Table 2. Classification metrics

Category	Precision	Recall	F1-score	Support
I	0.97	0.95	0.96	310
II	0.94	0.95	0.94	224
III	0.96	1.00	0.98	245
IV	1.00	0.96	0.98	221
Avg	0.97	0.97	0.97	1000

If TP, FP, and FN are respectively the number of true positives, false positives, and false negatives, then the classifier precision is given as:

$$Precision = \frac{TP}{TP + FP}. \tag{5}$$

The recall is given as:

$$Recall = \frac{TP}{TP + FN}, \tag{6}$$

and the F1-score is the harmonic mean of the precision and recall, given by:

$$F_1 = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}. \tag{7}$$

Fig. 9 shows the confusion matrix when testing with the augmented validation images and Fig. 10 shows the receiver operating characteristic (ROC) curve. This curve shows how the true positive rate increases with the false positive rate. For a binary classifier, when merely guessing, one expects the TP rate and FP rate to increase evenly. This is the reference line shown from the point (0,0) to (1,1). The closer the ROC curve is to the point (0,1) the better the classification.

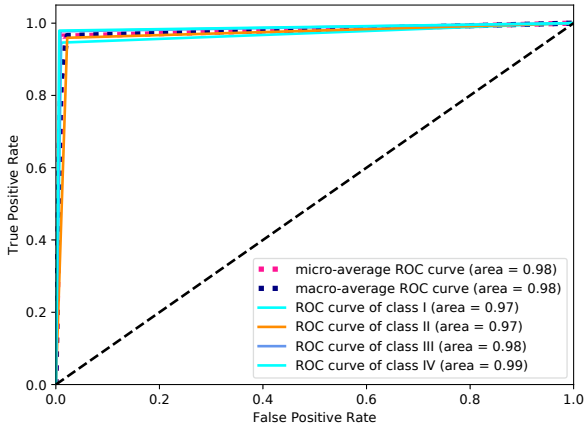

Fig. 10. Receiver operating characteristic curve.

5. CONCLUSION

Images of the feed ore captured from a model conveyor belt, as it is fed into a run-of-mine ore milling circuit, were used to characterize the feed ore size. A deep convolutional neural network based on the VGG16 architecture was used, and it was trained using transfer learning. Transfer learning reduces the number of input images required to successfully train the network. Image augmentation was also used, partly to increase the effective number of samples available, but also to make the network more robust.

The accuracy achieved shows how successfully a CNN can be employed to characterize the feed size, a result which may be difficult to obtain using only traditional computer vision techniques. The ore category information is an early indicator of changes that are required in the milling circuit to produce a consistent output within specification.

This deployment, and the success of the CNN, is an important step towards demonstrating that CNNs might be able to outperform traditional methods when it comes to fully extracting the feed size distribution from images obtained from an industrial feed ore conveyor.

ACKNOWLEDGEMENTS

The authors would like to thank Prof. Natasia Naude from the Department of Materials Science & Metallurgical Engineering at the University of Pretoria for providing ore samples.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., and Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. Retreived from http://arxiv.org/abs/1603.04467.

Chollet, F. et al. (2015). Keras. Retreived from https://github.com/keras-team/keras.

Coetzee, L.C., Craig, I.K., and Kerrigan, E.C. (2010). Robust model predictive control of a run-of-mine ore milling circuit. *IEEE Transactions on Control Systems Technology*, 18(1), 222–229.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Fu, Y. and Aldrich, C. (2019). Flotation froth image recognition with convolutional neural networks. *Minerals Engineering*, 132, 183–190.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36, 193–202.

Hamzeloo, E., Massinaei, M., and Mehrshad, N. (2014). Estimation of particle size distribution on an industrial conveyor belt using image analysis and neural networks. *Powder Technology*, 261, 185–190.

Ko, Y. and Shang, H. (2011). A neural network-based soft sensor for particle size distribution using image analysis. *Powder Technology*, 212, 359–366.

Le Roux, J.D., Craig, I.K., Hulbert, D.G., and Hinde, A.L. (2013). Analysis and validation of a run-of-mine ore grinding mill circuit model for process control. *Minerals Engineering*, 43, 121–134.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10).

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.

Maerz, N.H., Palangio, T.C., and Franklin, J.A. (1996). WipFrag image based granulometry system. In *Proceedings of the 5th Workshop on Measurement of Blast Fragmentation, FRAGBLAST 5*, 91–99. Canada.

Olivier, L.E., Craig, I.K., and Chen, Y.Q. (2012a). Fractional order and BICO disturbance observers for a run-of-mine ore milling circuit. *Journal of Process Control*, 22(1), 3–10.

Olivier, L.E., Huang, B., and Craig, I.K. (2012b). Dual particle filters for state and parameter estimation with application to a run-of-mine ore mill. *Journal of Process Control*.

Palangio, T.C. and Maerz, N.H. (1999). Case studies using the WipFrag image analysis system. In *Proceedings of the 6th International Symposium For Rock Fragmentation By Blasting, FRAGBLAST 6*.

Pan, S.J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22, 1345–1359.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, 1409(1556).

Tessier, J., Duchesne, C., and Bartolacci, G. (2007). A machine vision approach to on-line estimation of run-of-mine ore composition on conveyor belts. *Minerals Engineering*, 20, 1129–1144.

Wang, X. and Liu, H. (2018). Soft sensor based on stacked auto-encoder deep neural network for air preheater rotor deformation prediction. *Advanced Engineering Informatics*, 36, 112–119.

Wei, D. and Craig, I.K. (2009). Grinding mill circuits – A survey of control and economic concerns. *International Journal of Mineral Processing*, 90(1–4), 56–66.