Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 12038–12043

Estimating Ore Particle Size Distribution using a Deep Convolutional Neural Network ⋆

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: In this work the ore particle size distribution is estimated from an input image of the ore. The normalized weight of ore in each of 10 size classes is reported with good accuracy. A deep convolutional neural network, making use of the VGG16 architecture, is deployed for this task. The goal of using this method is to achieve accurate results without the need for rigorous parameter selection, as is needed with traditional computer vision approaches to this problem. The feed ore particle size distribution has an impact on the performance and control of minerals processing operations. When the ore size distribution undergoes significant changes, operational intervention is usually required (either by the operator or by an automatic controller).

Keywords: Deep learning, convolutional neural network, image analysis, minerals processing, neural regression.

1. INTRODUCTION

Silva and Casali (2015) argues that comminution (especially milling) is the most important minerals processing operation, partly because of its effect on the downstream processes, and further that the feed size distribution is one of the most important factors affecting semi-autogenous grinding mill performance. Tessier et al. (2007) also notes that feed size variations strongly affect mill performance, along with ore composition and grindability.

Particle size analysis is a crucial aspect of minerals processing. Product sizes are used to determine the optimal size of the feed for maximum efficiency, as well as to determine at which feed sizes losses occur in the plant, such that they may be reduced (Wills and Finch, 2015). Important changes in plant operation and control are made based on the results of the particle size analysis, either by an operator or automatic controller (e.g. Coetzee et al. (2010)).

The ore feed size may in some cases be inferred by applying parameter estimation to operational data (Olivier et al., 2012), but a more direct approach is to monitor images of the feed and inferring the size distribution using computer vision.

Methods exist that calculate the size distribution from input images, see e.g. Zhang et al. (2013) and Chen et al. (2018), or Maerz et al. (1996) which describes the commercially available WipFrag system. Wills and Finch (2015) notes that such systems generally segments the input images, carries out some corrections, and then calculates the size distribution. One common correction employed is the Rosin-Rammler equation to correct the cumulative percentage of particles passing a specific size, as these size distributions are generally non-uniform (Ko and Shang, 2011).

Such nonlinear corrections should be used with caution (Wills and Finch, 2015), which is in part why methods like those of Ko and Shang (2011) and Hamzeloo et al. (2014) have been proposed where image analysis is used in conjunction with a neural network. Traditional computer vision techniques are used for feature extraction, but a neural network is used to estimate the particle size distribution. Both methods used traditional "shallow" networks with only fully connected layers.

An often more accurate approach, which eliminates the need for explicit feature extraction, is deep convolutional neural networks (CNNs) (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 and fully connected neural networks.

Deep CNNs have only recently been applied in the process industries. Olivier et al. (2019) shows how the ore feed size category can be classified, 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. To the authors' knowledge,

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

10.1016/j.ifacol.2020.12.740

using a CNN for bulk solid size distribution estimation from a conveyor belt is novel.

In this work, a deep CNN is used to infer the feed size distribution from an ore image. The aim of using a CNN is to achieve superior accuracy without the need for explicit parameter selection for feature extraction, as is required when using traditional computer vision techniques.

2. CONVOLUTIONAL NEURAL NETWORKS

The first description of a true convolutional network seems to be that of Fukushima (1980). The structures and operations presently associated with CNNs are those introduced by LeCun in the late 1980s and early 1990s (see e.g. LeCun and Bengio (1995)). A recent overview of CNNs, including the common layers and how they work, is provided by LeCun et al. (2015). An overview of the development of deep learning in neural networks is given by Schmidhuber (2015).

Fig. 1 shows a simple CNN layout to illustrate common network elements. As the name implies, convolutional layers apply a convolution operation to the input image or feature map. This convolution is the core concept that drives the efficacy of CNNs and overcomes the impracticality of using only fully connected feedforward neural networks for image analysis. It reduces the number of free parameters that need to be trained, and also maintains some spacial information of neighbouring pixels. The output of the convolutional layer is a feature map of numerical values.

Pooling layers combine the outputs of a cluster of 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.

A feature map can then be flattened (converting it into a one dimensional array) such that the feature values can be passed to a fully connected layer. Fully connected layers are the same in principle to the layers in traditional multilayer perceptron neural networks.

Dropout layers may set the input from a fraction of neurons to zero, which in essence means that the layer is not fully connected. Other layer types may be used in CNNs, but those shown in Fig. 1 are the most pertinent to the present work.

The first number of layers in a CNN are devoted to feature extraction while the last layers are devoted to decoding these features into the output space (albeit for classification or regression).

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, like those in Fu and Aldrich (2019), use standard CNN architectures. Common architectures include AlexNet, Inception, and ResNet. The solution presented in this work makes use of the VGG16 network architecture (Simonyan and Zisserman, 2014), as shown in Fig. 2, specifically because it is more sensitive to the size of objects in the image than some of the other options mentioned.

2.1 Transfer learning

In order to accomplish appropriate feature extraction before the fully connected network layers, CNNs can become very large in size. More layers imply more network parameters, and as the network becomes very deep there can be millions of parameters that need to be defined through training. The VGG16 implementation used here has in the order of 28 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 task would render the use of deep CNNs impractical. A network that was trained for a specific 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, feature spaces, and data distributions.

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.

3. ESTIMATING ORE SIZE DISTRIBUTION

Minerals processing operations deal with a wide range of feed sizes and distributions as run-of-mine ore is generally the input, which is by definition not pre-processed. Operations therefore need to be robust enough to deal with large feed disturbances while producing an output of consistent quality. Having information about the feed size distribution is important for consistent production. Given this information, the appropriate operational adjustments can be made. These adjustments are either made by the operator or an appropriate automatic controller.

In this work, the VGG16 architecture as shown in Fig. 2 is used to infer the size fractions of ore in each of ten size classes. The dimensions shown in Fig. 2 are for the input images and subsequent network layers used in this work.

Note that the input is a color image (with three channels) with dimensions of 400x225. This is a 16:9 format that ensures the input image can be resized with a fixed aspect ratio. Following the fully connected layers is a layer with length ten which corresponds to the ten size classes. This layer employs a linear activation function, which is common for regression tasks.

3.1 Input data

Input images were taken from above using a vertically mounted camera to produce images similar to those from a typical on-belt system as shown in Wills and Finch (2015).

Images contain ore from one of the ten size classes, or from one of nine mixed batches containing various size

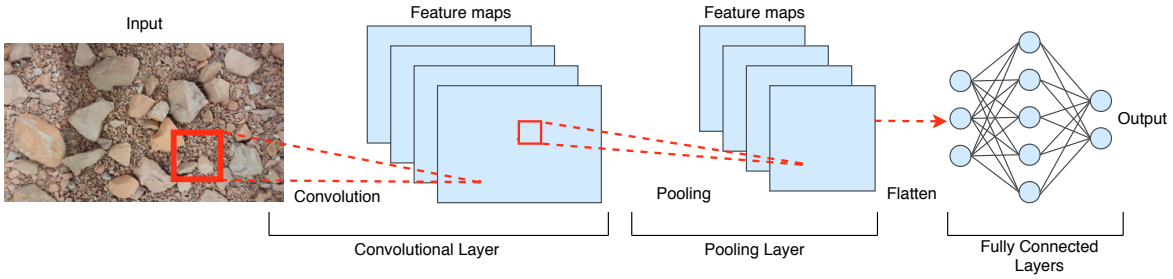

Fig. 1. Convolutional neural network layers.

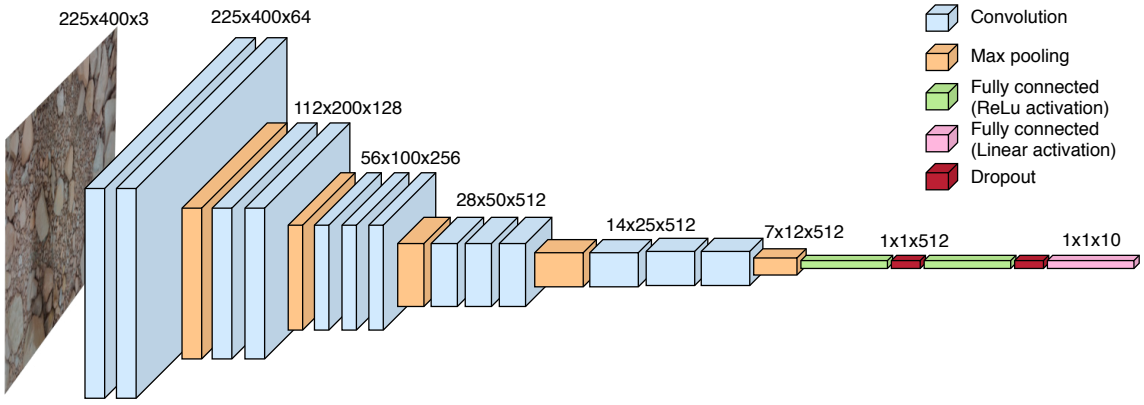

Fig. 2. VGG16 architecture (with specific dimensions used in this work).

classes; 781 images were captured in total. Table 1 shows the number of images taken from each source and also the normalized weight fractions per class. Fig. 3 shows one example image from each class.

The potential problems with nonlinear transformations like the Rosin-Rammler equation as highlighted by Wills and Finch (2015) is partly why correction factors are commonly applied to the equation in practice (see e.g. Ko and Shang (2011)). This is because the ore does not strictly originate from the distribution described by the equation. It is therefore necessary that the size distribution should be accurately inferred irrespective of the underlying distribution from which the ore was taken. As such, some of the mixed batches were purposefully prepared to have seemingly arbitrary distributions that cannot accurately be modelled by the Rosin-Rammler equation (or any of the other common distribution functions used to model ore sizes).

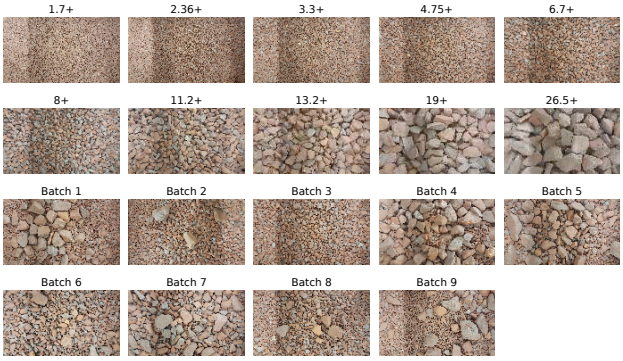

Fig. 3. One example image from each size class and mixed batch.

3.2 Data augmentation

Data augmentation is the process by which training data are up-sampled to improve network accuracy. Data augmentation increases the number of training samples, but more importantly it helps with the generalization of image features. In the present example, the network should not be sensitive to a large rock in a specific region of the image, it should indicate a large rock anywhere in the image.

Image augmentation may involve various image transformations. Here, augmentation is set up such that the input image may be flipped horizontally, rotated by as much as 10 degrees, translated horizontally or vertically by up to half of the width or height of the image, the color channels may be adjusted by up to 10 %, or any combination of these.

Fig. 4 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 somewhat difficult to confirm with the naked eye. When the image is rotated or translated, the empty space is filled by wrapping the image around the frame. This wrapping does not introduce significant image artefacts for translations, but might for rotations. This is why the rotation range is limited to 10 degrees. Rotations are however still important to ensure that the network does not over-fit on any vertical or horizontal shadows.

Table 1. Number of images and normalized weight per category by source.

Source	Num images	1.7+	2.36+	3.3+	4.75+	6.7+	8.0+	11.2+	13.2+	19.0+	26.5+
1.7+	25	1.000	-	-	-	-	-	-	-	-	-
2.36+	26	-	1.000	-	-	-	-	-	-	-	-
3.3+	25	-	-	1.000	-	-	-	-	-	-	-
4.75+	26	-	-	-	1.000	-	-	-	-	-	-
6.7+	26	-	-	-	-	1.000	-	-	-	-	-
8.0+	28	-	-	-	-	-	1.000	-	-	-	-
11.2+	26	-	-	-	-	-	-	1.000	-	-	-
13.2+	27	-	-	-	-	-	-	-	1.000	-	-
19.0+	25	-	-	-	-	-	-	-	-	1.000	-
26.5+	26	-	-	-	-	-	-	-	-	-	1.000
Batch 1	53	0.065	0.000	0.226	0.000	0.227	0.000	0.000	0.184	0.299	0.000
Batch 2	56	0.078	0.000	0.144	0.243	0.174	0.257	0.000	0.000	0.000	0.104
Batch 3	52	0.087	0.000	0.161	0.271	0.194	0.287	0.000	0.000	0.000	0.000
Batch 4	65	0.000	0.000	0.000	0.000	0.159	0.000	0.000	0.170	0.349	0.322
Batch 5	54	0.039	0.063	0.072	0.078	0.088	0.182	0.097	0.089	0.131	0.162
Batch 6	64	0.017	0.024	0.044	0.070	0.198	0.495	0.075	0.034	0.024	0.019
Batch 7	59	0.005	0.012	0.014	0.026	0.036	0.206	0.322	0.126	0.113	0.138
Batch 8	58	0.041	0.093	0.186	0.362	0.137	0.114	0.024	0.014	0.017	0.012
Batch 9	60	0.240	0.126	0.063	0.031	0.016	0.026	0.030	0.066	0.124	0.277

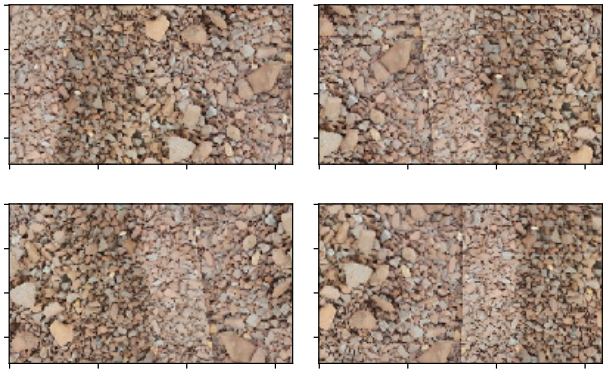

Fig. 4. Example of creating four augmented images from one input image.

3.3 Dividing the source data

The source images are divided into training, validation, and test sets.

The training set is used to train the CNN, and the validation set is used to track the training progress. This provides a good indication of the performance one may obtain when the model is in operation and these images are used to calculate the validation loss during training (which is an important metric that shows if the network is over-trained).

Hyper-parameter adjustments are partly made based on the reported validation loss. This means that the validation data is implicitly used during training. For this reason it is necessary to have a totally independent test set that was not used in any form during the training and re-training process.

When separating the data, 70 % is used for training (546 images), 15 % for validation (117 images), and 15 % for testing (118 images). This is done in a stratified manner, meaning that 70 % of the images from each source will be selected for training and the fraction of images in the training set per source will remain the same.

3.4 Training

The deep CNN is implemented using Keras with Tensorflow back-end. Training was conducted using Google Colab on an Nvidia Tesla K80 GPU.

The loss function in use is in the form:

$$J = \gamma_1 \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \gamma_2\left(1 - \sum_{i=1}^{N}\hat{y}_i\right)^2 + \gamma_3\sum_{i=1}^{N} s_i \quad (1)$$

where N is the number of size classes, y_i is the true weight fraction for size class i, $\hat{y}_i$ is the prediction of the weight fraction, γ_j with $j \in [1,3]$ and $\sum_{j=1}^{3}\gamma_j = 1$ is the relative importance weights of the loss function terms, and s_i is a slack variable in the form

$$s_i = \begin{cases} \hat{y}_i^2, & \text{if } \hat{y}_i < 0 \\ (\hat{y}_i - 1)^2, & \text{if } \hat{y}_i > 1 \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

The first term in (1) is the mean squared error (MSE) as is often used in training for regression problems. The second term is a soft constraint that tries to force the predictions to sum to 1. The third term tries to prevent predictions below 0 and above 1 by penalizing such values. This constrained loss function is not trivial to minimize. As such, an adaptive approach is followed whereby the relative importance weights γ_2 and γ_3 are set to zero for a number of epochs before being given a non-zero value. This means that the loss function only contains the MSE for a number of epochs, and once a decent solution space has been found the constraint terms are also included. With η representing the epoch number, the relative importance weights are set as:

$$\boldsymbol{\gamma} = \begin{cases} [\,1,\ 0,\ 0\,] & \forall\ \eta \leq 250 \\ [0.7, 0.1, 0.2] & \forall\ \eta > 250 \end{cases}. \quad (3)$$

Training is run for 500 epochs, with 32 batches per epoch, and 64 samples per batch. This means that $32 \times 64 = 2048$ samples are used per epoch. A stochastic gradient descent optimizer was used with an initial learning rate of 0.01 and Nesterov momentum. The training loss at the end of each epoch, as well as the components that constitute the

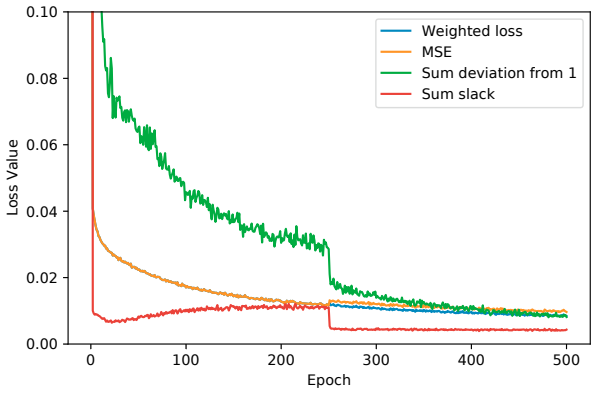

Fig. 5. Training loss values.

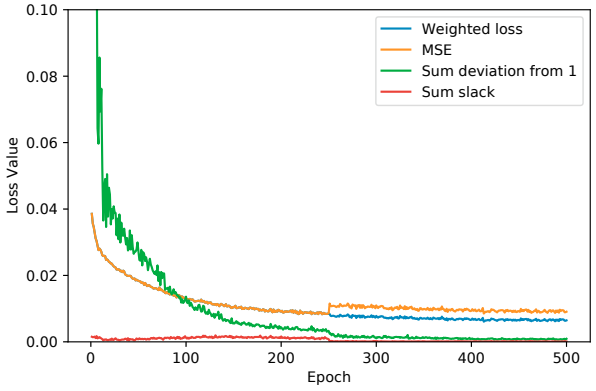

Fig. 6. Validation loss values.

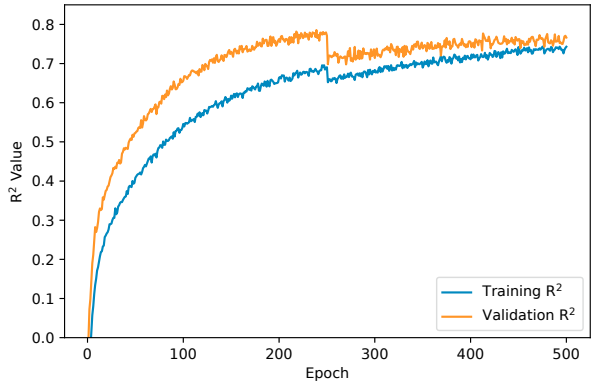

Fig. 7. Training and validation R^2.

loss, are shown in Fig. 5 and the validation loss values are shown in Fig. 6.

It is visible from Fig. 5 and Fig. 6 that before epoch 250 the weighted loss is simply the MSE, which has decreased significantly before this epoch. Once the relative weights are adjusted, the deviation of the sum of the predictions from 1 and the sum of slack variables, the second and third terms respectively in (1), decrease notably. After 500 epochs the rate of decrease of the loss has reduced. Fig. 7 shows the mean R^2 values achieved for the training and validation sets. The R^2 is not included in the loss function, but is shown as a proxy for the regression accuracy.

3.5 Size distribution estimation performance

To validate the network performance, the size estimates are produced for the 118 test images, which are the images the CNN has not yet encountered. The prediction constraints are then enforced by adjusting the produced prediction to be:

$$\tilde{y}_i = \frac{\text{median}\,(0, \hat{y}_i, 1)}{\sum_{i=1}^{N} \text{median}\,(0, \hat{y}_i, 1)}, \tag{4}$$

where $\tilde{y}_i$ is the final prediction for size i, N is the number of size classes, and median $(\cdot)$ in this case ensures $\hat{y}_i \in [0, 1]$. If $\hat{y}_i$ is accurate, this adjustment will not have a big impact, as the sum of the predictions will already be close to 1. The adjustment is however made to ensure that this is the case even with minor inaccuracies.

The cumulative size distribution is then calculated and presented along with the true distribution for each ore source in Fig. 8. The true normalized size distribution is shown in red, and the distribution of predictions per size class is shown as box plots in blue.

Next, the prediction error for each size class per image is calculated (10 prediction errors of each of the 118 test images produces 1180 prediction error values). The distribution of these values are shown in Fig. 9. The mean error is -0.012 and the standard deviation is 0.107.

4. CONCLUSION

Images of feed ore captured from above were used to estimate the ore size distribution. A deep CNN based on the VGG16 architecture was used, and was trained using transfer learning. 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 estimate the ore size distribution. Sizes are accurately estimated for arbitrary distributions as well as for those that may be accurately described using e.g. the Rosin-Rammler equation. An accurate indication of the ore feed size distribution allows early adjustments to be made to important operational parameters.

ACKNOWLEDGEMENTS

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

REFERENCES

Chen, H., Jin, Y., Li, G., and Chu, B. (2018). Automated cement fragment image segmentation and distribution estimation via a holistically-nested convolutional network and morphological analysis. *Powder Technology*, 339, 306–313.

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.

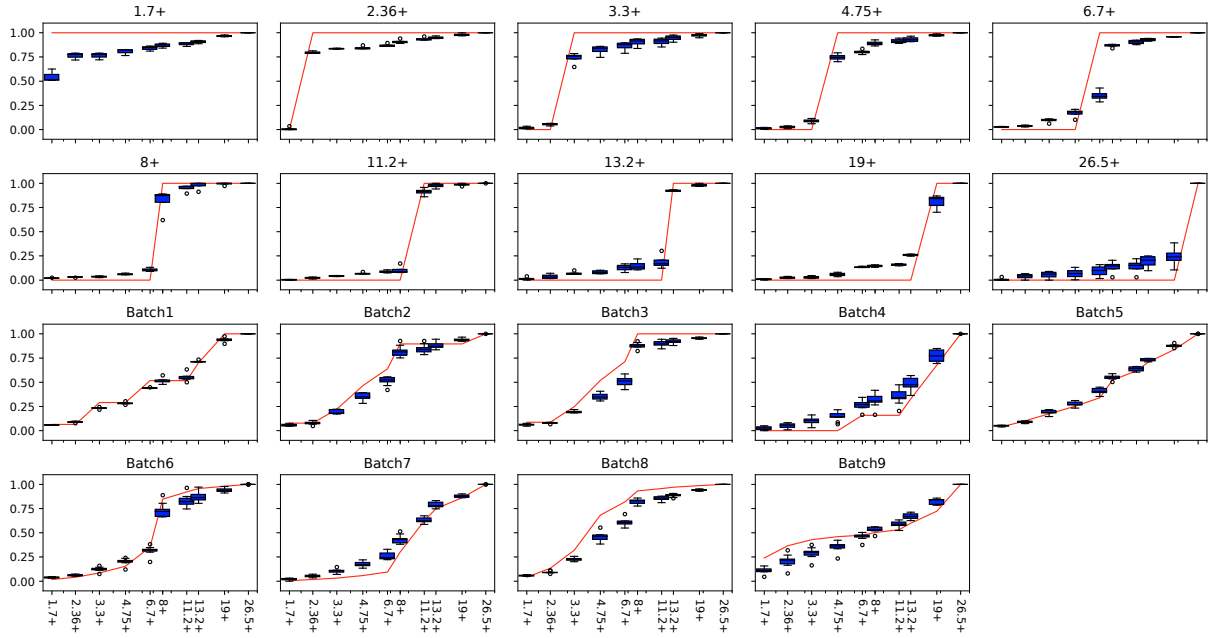

Fig. 8. Cumulative size distributions and predictions per ore source.

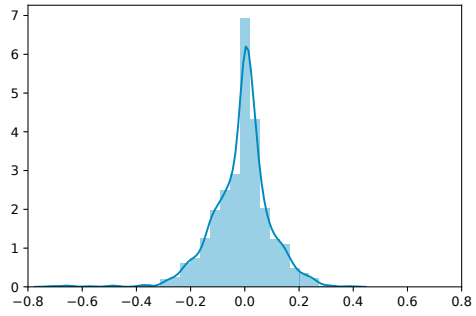

Fig. 9. Prediction error distribution.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., and Li, F. (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.

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., Huang, B., and Craig, I.K. (2012). Dual particle filters for state and parameter estimation with application to a run-of-mine ore mill. *Journal of Process Control*, 22, 710–717.

Olivier, L.E., Maritz, M.G., and Craig, I.K. (2019). Deep convolutional neural network for mill feed size characterization. *IFAC-PapersOnLine*, 52, 105–110.

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.

Silva, M. and Casali, A. (2015). Modelling SAG milling power and specific energy consumption including the feed percentage of intermediate size particles. *Minerals Engineering*, 70, 156–161.

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.

Wills, B.A. and Finch, J. (2015). *Wills' mineral processing technology: an introduction to the practical aspects of ore treatment and mineral recovery.* Butterworth-Heinemann.

Zhang, Z., Yang, J., Su, X., Ding, L., and Wang, Y. (2013). Multi-scale image segmentation of coal piles on a belt based on the Hessian matrix. *Particuology*, 11, 549–555.