Research article

QJEGH

Quarterly Journal of Engineering Geology and Hydrogeology

https://doi.org/10.1144/qjegh2020-183 | Vol. 55 | 2022 | qjegh2020-183

Machine learning applied to pore-space geometry in sandstones: a tool for evaluating grain-scale similarity?

Alexander Hall*, Martin Gillespie, Paul Everett, Vyron Christodoulou and Jo Walsh

British Geological Survey, The Lyell Centre, Research Avenue South, Edinburgh EH14 4BA, UK

AH, 0000-0003-4173-0555; MG, 0000-0002-1585-7344; PE, 0000-0002-9378-9615; JW, 0000-0003-0268-2263

* Correspondence: ahall@bgs.ac.uk

Abstract: The ability to identify similar sandstones to a given sample is important where the provenance of the sample is unknown or the quarry of origin is no longer in operation. In the case of building stones from heritage buildings in protected areas, it may be mandatory. Here, a proof of concept for an automated similarity measure is presented by means of a convolutional autoencoder that is able to extract features from a sample thin section and use these features to identify the most similar sample in an existing image library. The approach considers only the shape of the pore space between grains, as, if the pore space alone contains enough information to distinguish between samples, the required image pre-processing and training of a model is greatly simplified. The trained model is able to predict correctly the progenitor quarry of a thin section, from an eight-class dataset of Scottish sandstones, with an accuracy of 47.9%. This prototype, although insufficient for commercial purposes, forms a benchmark for future models against which improvements can be assessed and some of which are suggested.

Thematic collection: This article is part of the Digitization and Digitalization in engineering geology and hydrogeology collection available at: https://www.lyellcollection.org/cc/digitization-and-digitalization-in-engineering-geology-and-hydrogeology

Received 30 November 2020; **revised** 18 June 2021; **accepted** 18 June 2021

An objective and reliable means of quantitatively ranking multiple sandstone samples in terms of their grain-scale geological similarity would have wide application in, and beyond, the field of geology. An automated system that can accomplish this programmatically using digital image analysis software would find a variety of applications. If successful, it may be able to augment or even replace manual interpretation, offering potentially huge advantages, particularly when applied to large sample suites and image datasets. Some examples of such applications in the geosciences include; correlating strata across boreholes, defining regional lithological variations between sandstone units, and monitoring the consistency of a quarry's output. If such a system can be designed and proven to be successful for these tasks, it will have the potential to reduce errors, and drastically improve consistency and robustness, when compared to standard manual interpretation methods, particularly when they may be completed by various teams of geologists. However, this has remained an elusive goal to date.

Judging similarity is the central aim when identifying the most suitable currently available stones to use in repairing old sandstone buildings. As the original stone used for these buildings is often not known or actively quarried today, achieving a close match can be challenging. For sandstone buildings in particular, a key consideration is matching the pore-space characteristics of the original and replacement stone as closely as possible because the porous nature of sandstones exerts a strong control on their weathering characteristics. Microscope analysis should therefore be employed to evaluate grain-scale similarity when proposing stone replacement. The British Geological Survey (BGS) regularly provides 'stone-matching' advice to this end, and possesses a baseline of sandstone quarry samples and thin sections geared towards this purpose. The direct benefit that a computationally derived similarity measure could deliver for sandstone 'stone matching' and the availability of a well-established baseline dataset well suited for image analysis provided the impetus for the study we describe in this paper.

Human specialists using simple visual examination techniques such as petrographical analysis struggle to make objective assessments of similarity because of the sheer number of grain-scale variables such as constituents and texture, and the often subtle distinctions between sandstones. Instrument-based methods of analysing rocks – for example, X-ray diffraction or fluorescence (XRD, XRF) and low-count energy-dispersive X-ray spectra such as QEMSCAN^{TM} – provide far greater potential for objectivity; however, human expertise must still be applied to results to achieve a similarity ranking, thereby reducing overall objectivity. Most previous attempts to apply digital image analysis to sandstone have had the aim of quantifying specific rock properties such as mineral proportions, grain size, porosity and pore connectivity, rather than achieving a similarity ranking (Chen *et al.* 2001; Moreira *et al.* 2012; Aziz 2013; Bukharev *et al.* 2018). Success has been achieved in automatically calculating these pre-determined parameters by applying colour filters to images and extracting areas that are identified as pore space (Berrezueta *et al.* 2015; Buckman *et al.* 2017). For applications where the properties of a sandstone are to be derived from thin-section images, this approach is suitable. However, visual similarity between sandstones is governed by a wider selection of factors that cannot necessarily be simplified into human-perceptible metrics. An approach that did not require the pre-selection of parameters was therefore sought.

Machine learning (ML) software applied to image recognition and classification is a rapidly developing field that may present opportunities to create new tools for geomaterials analysis. Artificial neural networks (ANNs) are currently the most popular family of ML algorithms for image analysis. In this paper we describe the methodology and preliminary outcomes of a study that uses a ML approach to rank samples of sandstone according to the geometry of their pore spaces, using a type of ANN known as a convolutional autoencoder.

Some basic concepts of machine learning

ANNs are a family of ML algorithms that have found widespread use for analysing large or complex datasets (Schmidhuber 2015). Collectively, ANNs comprise a broad collection of individual algorithms with numerous applications. As such, it is difficult to group them into a single area on a map of the ML ecosystem. Figure 1 demonstrates a simplified taxonomy of the ML space and showcases some typical applications of selected algorithms for geostatistical problems. In the geosciences, ANNs can be applied to image analysis problems at any range of scales where a baseline of images exists; from the microscope scale in the case of geomaterial analysis, to landscape and continental scales in, for example, regional surveys or remote sensing studies. Furthermore, while 'image analysis' is a popular description of the purpose to which these algorithms are put, they can be applied in exactly the same way to any high-resolution raster dataset (e.g. geophysical data, hyperspectral scans, etc.) as in data terms a raster is analogous to a digital image, consisting of a grid of regular cells or pixels each with values that correspond to colour or any other quantified property.

ANNs consist of multiple interconnected hidden layers of nodes, where a node takes multiple inputs to produce a single output. Individual nodes are connected by weights, with the sum of products of the individual incoming inputs and weights being the overall input to each node. Typically, an input layer consists of the input data and is passed through sequential hidden layers until an output is computed on the final layer. If labelled data are available, and the class of each data point is known, it is possible to train a 'supervised' ANN to predict the class of new data. This is achieved by repeated feed-forward and backpropagation operations. A feed-forward operation involves passing through an element from a 'training set' of data and evaluating how well the network is able to predict the class of each input. For each data point passed forward through the network, the weights are adjusted to attempt to correct for prediction error. This intermittent and incremental correction of the network weights is referred to as backpropagation. A separate 'test set' of data is required to evaluate the performance of the network. This typically consists of some fraction of the total data available and must be distinct from the training set. By presenting the network with new data to which it was not exposed during training, a representative measure of the network's predictive accuracy is obtained. A model trained on a training set that is subsequently able to predict the class of new data is referred to as a classifier. While it is possible to train classifiers that can predict across thousands of classes, these are generally applied to distinct and discrete classes such as objects (He *et al.* 2015). However, natural geomaterials like sandstone are different: no two sandstones are identical, the features that distinguish them may be subtle and similar features may be shared by multiple sandstones. Consequently, a classifier used to assess a sandstone that was not included in the training set may not perform well as it can only choose from the classes it has been trained on. By contrast, a similarity-based model is able to produce results based on similarity alone and does not attempt to force predictions into predetermined classes. Thus, the aim of this study was to test the potential for ML to rank sandstone similarity rather than explicitly predict sandstone class.

For unlabelled data (i.e. data, such as images, that are not assigned a class label), ANNs may be used in an 'unsupervised' manner to cluster similar data points together, allowing a user to determine which points are the most similar. In the case presented here, we sought to use an unsupervised algorithm to evaluate the similarity of different sandstones from images of those sandstones but used the class labels to determine the performance of the network.

The following terminology is used below to discuss the operation of the neural network used in this study.

- Activation function: as connected layers only perform simple element-wise multiplication and addition, they can only compute linear functions. Applying a chosen non-linear activation function to a node's output allows the network to account for non-linearity.
- Loss function: this is a characterization of error produced by the network. The aim of a trained neural network is to minimize the loss function.
- Optimizer: the optimizer is a regime that defines how the network weights and biases are changed on each iteration of backpropagation.
- Hyperparameter: an ANN is also defined by a number of fixed parameters that can be adjusted to alter the model characteristics. The process of retraining a model with different hyperparameter values is referred to as 'hyperparameter tuning'.
- Learning rate: the learning rate is a hyperparameter that defines how quickly the network values are changed during backpropagation. An ideal learning rate is high enough to allow for fast training but low enough to allow the network to detect subtle features and maintain stability.
- Graphics processing unit (GPU): a class of computer processor typically used to train ANNs.

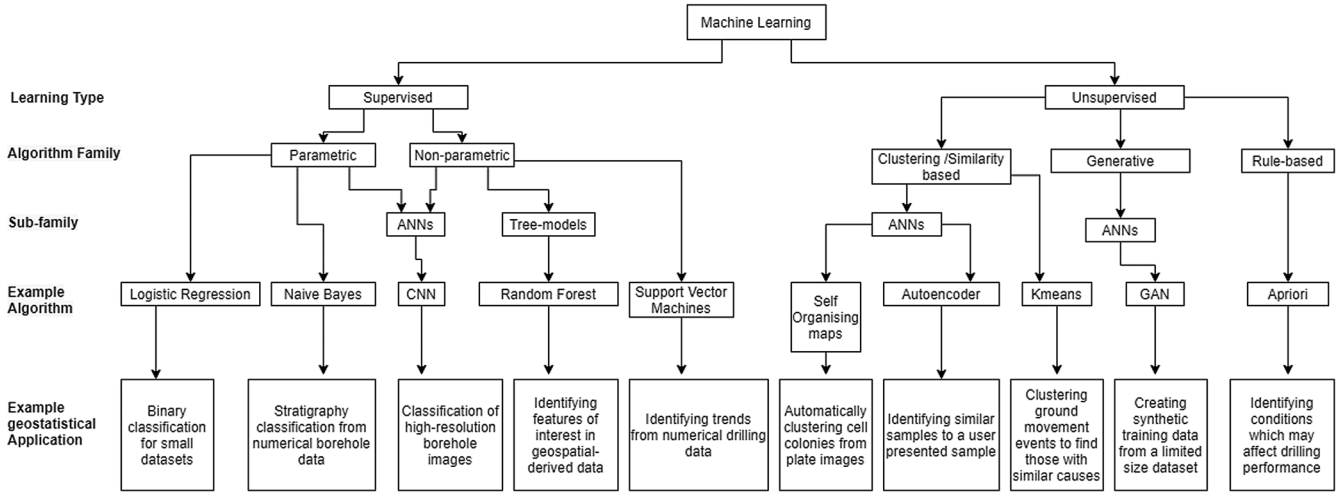

Fig. 1. Simplified map of some ML methods, along with typical applications in the field of geostatistics. Note that due to the broad and variable nature of ANN-based algorithms, they are generally found in most families of ML algorithm.

This study made use of a convolutional neural network (CNN). CNNs are a type of ANN that perform well in image analysis problems and have been demonstrated as solutions for geological thin-section classification (Cheng and Guo 2017). A CNN typically consists of the following layers:

- Input layer: this is the input data. For the data presented in this experiment, the input layer consisted of a 2D array, henceforth referred to as a map.
- Convolutional layers: these generate feature maps from the previous layer. This is achieved by passing a filter, or kernel, across the layer in incremental strides, with the stride size being predefined. The filter consists of a relatively small 2D array of weights that are multiplied element-wise by the values in the underlying layer and summed to produce a single output value for each step. The resulting array of output values forms a single 2D output array. A convolutional layer with n filters will result in an output array n layers deep.
- Max pooling: as a convolutional layer seeks to extract useful features, the presence of such features is then summarized by downscaling the image by downsampling to a degree defined in a max-pooling layer. This applies a small array stepwise across the feature map and takes the maximum pixel value from each step.
- Fully connected layer: repeated convolution and max pooling results in the feature maps being reshaped to a vector. A fully connected layer consists of another vector that is connected element-wise to the input vector by a series of weights and biases. Applying each weight, summing and adding the bias term results in a populated fully connected layer.
- Upsampling layer: this is the reverse of the max-pooling layers. By duplicating pixels stepwise, the prior layer can be increased in size.

In order to compute an output, a convolutional layer must perform the following number of operations:

$$N = k(m^2 f^2 n_\mathrm{i} n_\mathrm{o})$$

where N is the number of operations, k is a constant, m is the spatial size of the layer, f is the size of the filter, n_i is the number of input filters and n_o is the number of output filters. Hence, training time for a typical CNN grows at least proportionally to the square of the image size. However, since the number of filters is usually increased for larger images, so the relationship between training time and image size is often cubic (He and Sun 2015). Beyond a given threshold for image size, the required number of operations will exceed the available GPU memory and training will not be able to proceed.

Sandstone samples

Sandstone is a sedimentary rock composed mainly of sand grains between 0.064 and 2 mm in diameter. Most sandstones are porous, and up to around 30% of their total volume can consist of pore space. In any given sandstone, the character of the pore space, including the size, shape, distribution and connectivity of pores, and the proportion of rock volume they occupy is a product of two things: the physical attributes of the sand grains when they were deposited (including their size and roundness, and the extent to which they define a fabric such as layering); and the cumulative effect of subsequent processes, such as compaction, mineral growth and mineral dissolution, that occur in association with diagenesis and other geological processes. Sandstones from different bedrock formations are very unlikely to be identical, or even particularly similar, in these respects, and so the pore-space character of the sandstone from a single formation to some extent may be distinctive. Thus, a central hypothesis of this study is that the geometrical properties of the pore-space component in a sandstone contain a 'geological fingerprint' of that sandstone (and the rock formation from which it was sourced), which can be recognized by a ML algorithm. If that is so, it should be possible for the algorithm to compare multiple samples and produce a similarity ranking. The natural variability of geological processes means that pore-space character inevitably will vary to some extent within all sandstone formations, at all scales. Controlling parameters may include, for example, the variations in flow rate and turbulence that occur in space and time as sand grains are deposited from a fluid, and resulting changes in sand-grain properties both laterally and vertically in a succession. A key objective of a future phase of this study will be to determine whether sandstone samples from different parts of a formation retain a detectable 'fingerprint' in their pore-space component.

Analytical method

Eight samples of sandstone, available at the project Gitlab repository (Hall *et al.* 2020), were selected for the study, each representing a different sandstone-dominated bedrock formation in Scotland. The formations span four main chronostratigraphic divisions, and comprise sandstones of varying colour and from different depositional environments (Table 1). The samples also present a range of grain-scale characteristics that will have a material influence on the geometrical character of their pore spaces: for example, the extent to which quartz overgrowths, other mineral cements and intragranular (as well as intergranular) pore space are developed. The character and potential influence of these features have not been evaluated for this proof-of-concept study but their role in affecting the ML results will be important factors in future work. It should be noted that eight samples in isolation would be too few to build a reliable matching tool for all UK sandstones as the

Table 1. *Details of the sandstones included in the trial. The first column details four-letter abbreviations of the sandstone name that were used for image filenames, etc., and which also appear in subsequent figures*

Abbreviated label	Formation and source	Chronostratigraphic division	Depositional environment and colour
Gull	Gullane Formation: Craigleith Quarry, Edinburgh	Carboniferous	Fluvial, grey
Hopm	Hopeman Formation: Clashach Quarry, Hopeman, Moray	Jurassic–Cretaceous	Fluvial/aeolian, buff
Kinn	Kinnesswood Formation: Hawkhill Wood Quarry, Craigmillar, Edinburgh	Cretaceous	Fluvial, pinkish grey
Loch	Locharbriggs Formation: Knowehead Quarry, Locharbriggs, Dumfries	Permian–Triassic	Aeolian, orange
Pass	Passage Formation: Germiston Quarry, Glasgow	Carboniferous	Fluvial, grey to buff
Radd	Raddery Formation: Milton Of Redcastle Quarry, Muir of Ord	Devonian	Fluvial, brownish orange
Swan	Swanshaw Formation: Culzean Bay Quarry, Ayrshire	Devonian	Fluvial, purplish grey
ULFS	Upper Limestone Formation: Drumhead Quarry, Denny, Stirlingshire	Carboniferous	Seafloor, grey to buff

low sample number does not provide the required variability. However, with each sample consisting of eight high-resolution images, there was sufficient raw information available to train up a neural-network-based matching tool that is able to distinguish between only those eight samples. As discussed in the following 'Image processing' subsection, the images for each sample were subdivided to produce 8064 data points (1008 per sandstone sample).

Images for training and testing the ML algorithm were obtained from thin sections in which the rock slice has maximum dimensions of 20 × 40 mm. Prior to thin-section preparation, the rock samples were vacuum-impregnated with a blue resin; in the resulting thin sections, all of the pore space appears blue (Fig. 2). Thin-section images were captured using a Zeiss Axio Imager.A2m optical microscope with dedicated digital camera (Zeiss Axiocam 305 colour) and ZEN image capture software. Test images produced at ×1.25, ×2.5 and ×5 magnification were evaluated; those produced with the ×2.5 objective lens provided the best combination of fine detail, sharp focus and even lighting across the field of view, as well as a reasonably large field of view. The field of view of images produced with this lens are 3.4 × 2.8 mm (9.5 mm^2), which is roughly 1/70 of the area in a typical rock slice. To represent the natural variation within each thin section, a series of eight images was obtained from a single 'traverse' made across the middle of the thin section and parallel to its long axis (Fig. 3).

Image processing

The dataset used to train and test the ML algorithm consisted of 64 RGB images (eight images from each of eight thin sections), with a resolution of 2464 × 2056 pixels. As training time grows rapidly with image size, the raw images were much too large to use as inputs and so were sliced into 256 × 256 pixel subsections. This size was sufficiently small to train a network but empirically large enough to retain pore-space features that humans are able to distinguish. Choosing a size in the form 2^n also simplified the implementation of layers in the network. Using a large number of small images also avoided the problem that a small number of input raw images leads to few and infrequent backpropagation runs, making the trained network susceptible to overfitting (Le *et al.* 2020). The natural components of a sandstone (mineral grains, etc.) are never blue, so the presence of blue resin in the pore spaces made it simple to produce an accurate binary (i.e. black/white) map of the pore space in each subsection. The Python implementation of OpenCV (Bradski 2000) was used to apply colour filters defined by values for hue, saturation and value (HSV). The HSV values [70, 50, 50] and [140, 255, 255] were used for the lower-bound and upper-bound filters, respectively. This converted each image subsection to a 256 × 256 pixel binary map with a value of '1' representing pixels that passed the filter and '0' representing those that did not. The maps were saved as binary.png images for ease of analysis and to allow for visual inspection. For the purpose of the study, each sandstone (as represented by one thin section) is considered a 'class', and each map was labelled with its class using the abbreviated terms in 1.

Training and testing datasets

For most ML applications, the proportional split between train and test images is usually around 80/20 (i.e. 80% are used for training and 20% for testing). A rather different split of 25/75 was used in the current study because, with only eight full-size raw images available to derive subsections for each class, the feasibility of the underlying method is likely to be better demonstrated with a relatively large test set. To augment the unusually small training dataset, the maps comprising the dataset were each duplicated and rotated in three 90° increments. This 'rotational augmentation' quadrupled the amount of raw data in the training set but did not necessarily proportionally increase the amount of useful information available to the model, as the underlying data are unchanged with each rotation. After processing and augmenting the dataset, the final training set consisted of 4608 binary 256 × 256 pixel maps and the test set consisted of 3456 binary 256 × 256 pixel maps. The training set images were derived from two of the raw images for each class, while the test-set images were derived from the remaining six images from each class. The progression from raw images to the training and testing sets is summarized in Figure 4.

Machine learning methodology

The aim of this experiment was to demonstrate the concept that similarity between pairs of images can be determined and ranked. To do this, an algorithm capable of simplifying images into an easy-to-compare form by means of extracting spatial features had to be used (Wang and Rajan 2020). This was achieved by means of a convolutional autoencoder (Karimpouli and Tahmasebi 2019). This consisted of three parts, as shown in Figure 5:

(1) An encoder: this passed the input maps through repeated pairs of convolutional and pooling layers until the output tensor was reduced to a 2048 element vector. The repeated convolutions served to construct feature maps from the input images, while the max-pooling layers incrementally performed dimensionality reduction.
(2) A code layer: this was the vector produced by the encoder section of the autoencoder. If the autoencoder was able to identify distinguishing features between different classes, they would be encoded within the individual elements of this code layer.
(3) A decoder: this was the inverse of the encoder layer, and applied repeated pairs of convolutional and upsampling layers. By repeatedly performing these operations, a 256 × 256 pixel output map was the output of the autoencoder.

Training consisted of presenting batches of input maps to the autoencoder. Each map was reduced to a vector through the encoder section and then reconstructed through the decoder section. The reconstruction error between the input map and that reconstructed by the autoencoder was evaluated by calculating a loss function, in this case the binary crossentropy between the two. Binary crossentropy as a measure of reconstruction error has previously been demonstrated as a suitable loss function for maps with pixel values between 0 and 1, and is also the computationally cheapest function (Creswell *et al.* 2017). Then, through backpropagation, the network's weights between layers were adjusted in response to the value of the binary crossentropy. This process was repeated for each input map, with the network weights being repeatedly adjusted to minimize the loss between the input and output map. The entire training dataset was passed through the network multiple times, with each full pass referred to as an 'epoch'. To limit training time, training consisted of a fixed period of 300 epochs.

The aim was to train an autoencoder that could reconstruct the input map as accurately as possible. The lower the reconstruction error, the better the autoencoder would be able to encode maps to a vector using the encoder half of the network. A sufficiently low reconstruction error would only be possible if the input maps contained enough features to allow them to be meaningfully compressed in this manner.

The autoencoder was built in Tensorflow via the Keras API (Chollet *et al.* 2015) through a series of Python scripts (https://gitlab.com/bgsdatalab/building-stones). Training runs were carried out on a single machine equipped with an NVidia GTX1060 GPU.

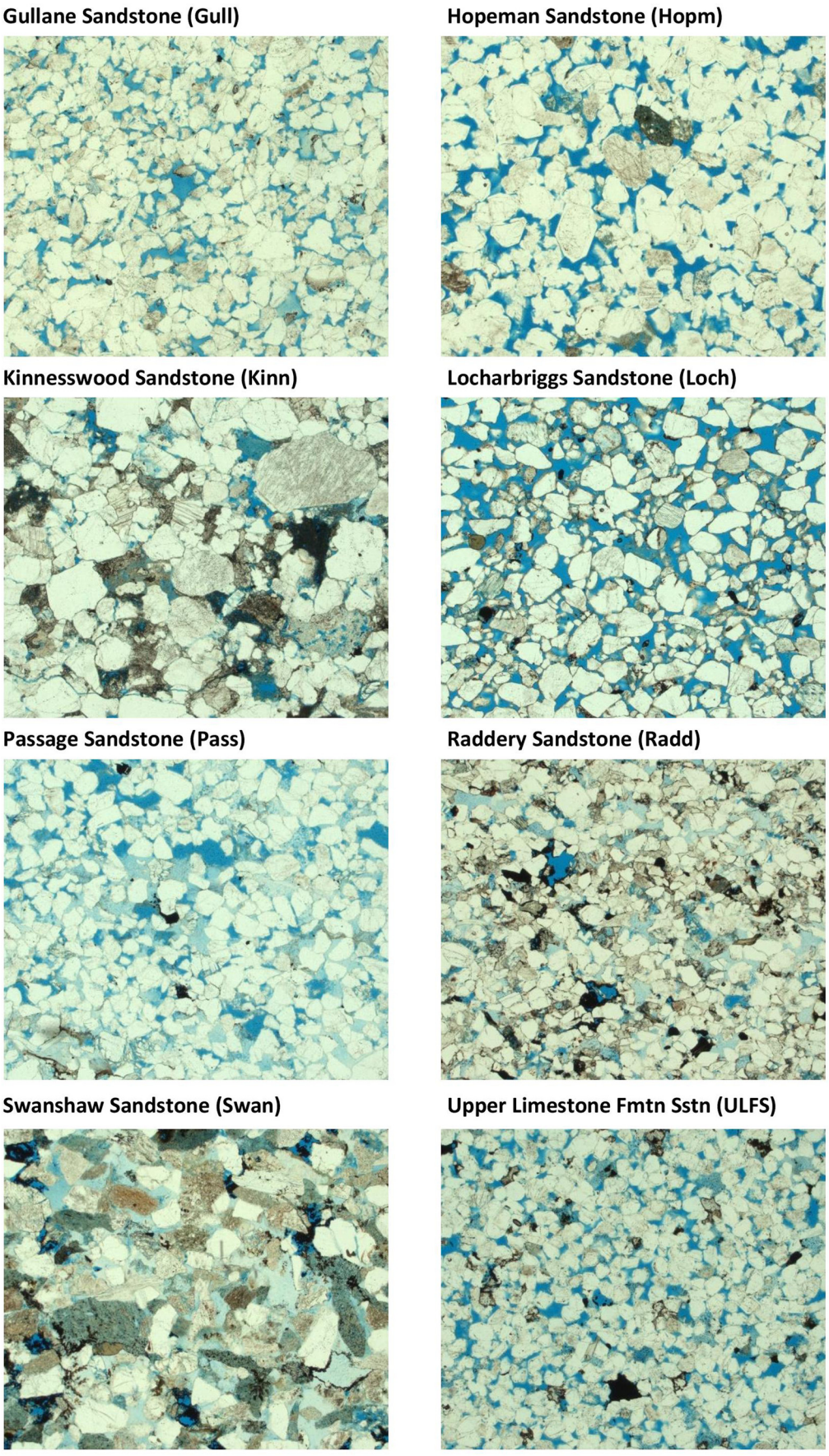

Fig. 2. Each of these full-size raw images has a size of 3.4 × 2.8 mm and each shows the typical appearance of each sandstone. Blue areas are resin-filled pore space.

Fig. 3. Black outlines show the position of each image in a 'traverse' across the thin section. Blue areas are resin-filled pore space.

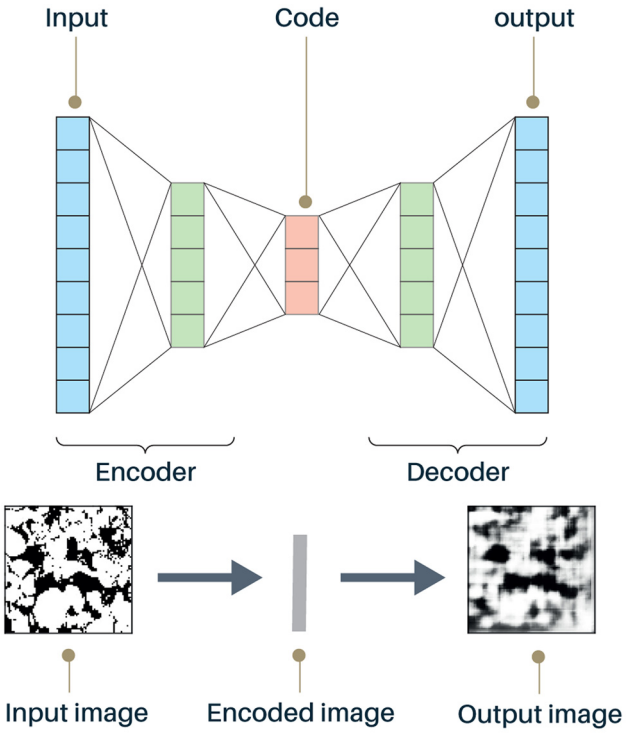

Fig. 5. Input compression and subsequent reconstruction via an intermediate code layer – collectively forming the autoencoder.

Model architecture

The encoder half of the network consisted of the layers detailed in Table 2 through which the input map was passed in order to reduce it to a 2048-length vector. Note, all convolutional layers had a stride of 1.

The resulting vector was then converted back to a 256 × 256 pixel map through the decoder, which consisted of the layers detailed in Table 3. Again, all convolutional layers had a stride of 1.

An Adam optimizer (Kingma and Ba 2014) was used to adjust the network weights after each training batch was fed forward through the above layers. Since the input data relied on differentiation of fine features, it was expected that relatively small adjustments would be necessary to the network's weights during each backpropagation run in order to preserve fine detail in the network. A learning rate value was chosen after several trial runs to determine a sensible value. The chosen value was 5E-4, which was lower than the default value of 0.01 assigned by Keras but not atypical for image recognition problems.

To arrive at the final chosen learning rate, training runs were carried out with learning rates between 1E-6 and 1E-2, with each run requiring 3 h of compute time using the available computing infrastructure. Model accuracy was highly sensitive to learning rate, with most values leading to an unstable model. In this context, instability refers to a network that adjusts its weights too quickly during the training process. This would lead to it being unable to detect fine detail in the input images as the large weight adjustments only allow it to detect coarse

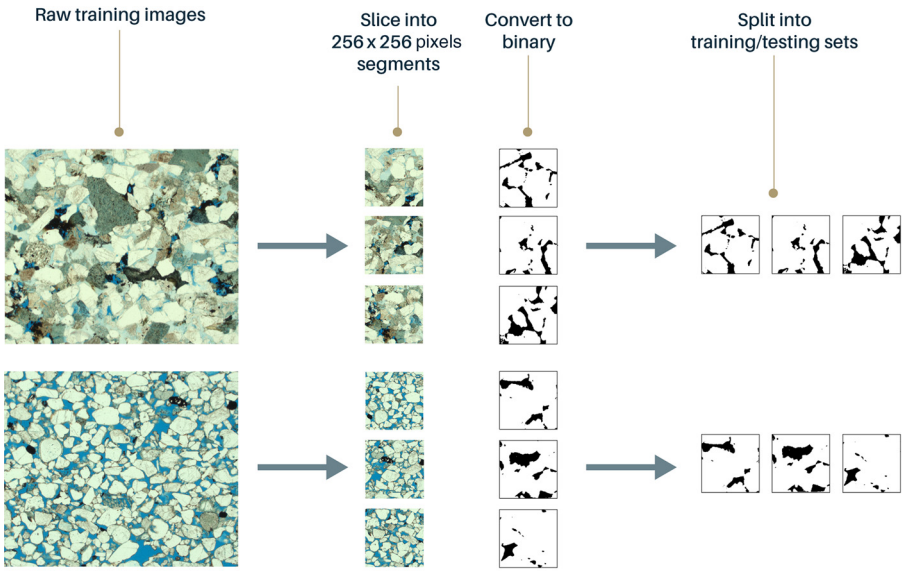

Fig. 4. Summary of the processing steps applied to the raw images to obtain training and testing datasets.

Table 2. *Structure of the encoder section*

Layer type	Filters	Kernel size
2D convolutional + LeakyReLU activation	8	5 × 5
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	16	5 × 5
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	32	3 × 3
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	64	3 × 3
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	128	3 × 3
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	256	3 × 3
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	512	1 × 1
Max pooling		2 × 2
2D convolutional + LeakyReLU activation	2048	1 × 1
Max pooling		2 × 2

features. An unstable network therefore tends to adjust its weight values rapidly in opposing directions on subsequent backpropagation runs and never converges on final values.

Model testing

Only the encoder half of the autoencoder was used during the testing stage. The trained encoder was used to compress all the images in the testing set into vectors, as illustrated in Figure 6. By computing the cosine of the angle between two vectors, the distance between the vectors in *n*-dimensional space is measured. This measure is referred to as the 'cosine similarity', and is a common and computationally cheap method to determine vector–vector similarity (Han *et al.* 2012).

The cosine similarity between every pair of vectors in the training set was computed. A 'most similar' candidate image was derived by taking each test-set image in turn and finding that with the lowest cosine similarity in the rest of the test set. These most similar candidates were then used to evaluate the model performance. As the original dataset was labelled (with its abbreviated sandstone name), it was possible to determine the extent to which each test-set image was predicted to be most similar to another image from the same class. Each 256 × 256 pixel extract from each individual raw test-set image was evaluated individually. For each raw test-set image, the most frequently predicted class across all of the extracts was recorded and taken to be the overall class prediction for that raw image. The process of evaluating individual extracts to predict an entire full-size test image is summarized in Figure 7. By summing the correct predictions across the entire test set, a measure of model accuracy was calculated.

Table 3. *Structure of the decoder section*

Layer type	Filters	Kernel size
Fully connected layer (length = 2048)		
2D convolutional + LeakyReLU activation	2048	1 × 1
Upsampling		2 × 2
2D convolutional + LeakyReLU activation	512	1 × 1
Upsampling		2 × 2
2D convolutional + LeakyReLU activation	128	3 × 3
Upsampling		2 × 2
2D convolutional + LeakyReLU activation	64	3 × 3
Upsampling		2 × 2
2D convolutional + LeakyReLU activation	32	3 × 3
Upsampling		2 × 2
2D convolutional + LeakyReLU activation	16	5 × 5
Upsampling		2 × 2
2D convolutional + LeakyReLU activation	8	5 × 5
Upsampling		2 × 2
2D convolutional + sigmoid activation	1	5 × 5

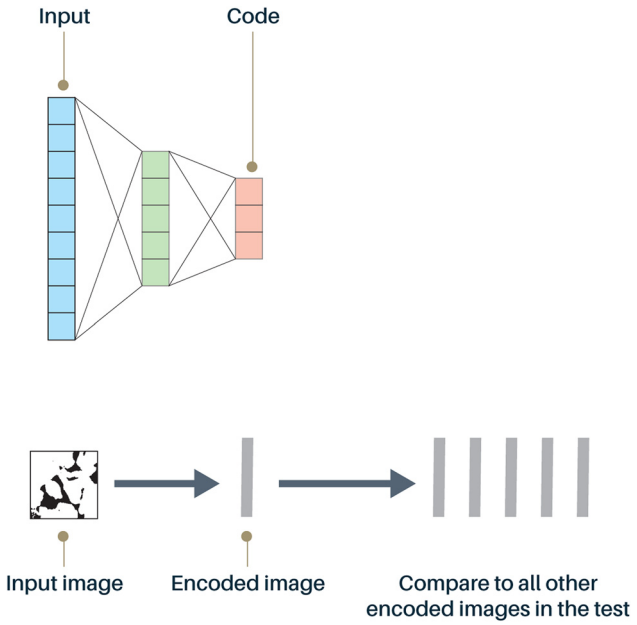

Fig. 6. Using the decoder, extracted from the trained autoencoder, to extract features from new images to be compared to feature representations from the other test-set images.

Results

The confusion matrix obtained by predicting on the test set is shown in Figure 8. This summarizes the class predictions for the entire test set in a tabulated format that displays the number of times the predicted class of each image coincided with the known class of that image. This was our primary means of evaluating the overall performance of the trained model.

The model accuracy was 47.9% across eight classes, where accuracy was defined as the ability of the model to predict correctly a sandstone from a given image. The accuracy was not uniform across classes, with some sandstones being correctly predicted more often than others. Of the six Swan images, all six were correctly predicted by the model, although it also incorrectly predicted an additional four images as Swan when they were actually other classes. Expressed formally, this gave a precision of 60% and a recall of 100% for the Swan class. The Hopm and Kinn classes also showed relatively good results with precisions of 50 and 45%, and recalls of 83 and 83%, respectively.

Plotting the training loss against the epoch number (Fig. 9) results in an initially rapidly falling loss value as the model learned coarser features, with the rate slowing over time as the network weights were progressively adjusted less and less. Ten per cent of the training set was reserved for model validation. Plotting the validation loss against the epoch number results in a less pronounced loss curve with relatively large fluctuations evident. This is likely to be due in part to the limited number of images available for validation and, indeed, the relatively small test set acting as a barrier to improved accuracy. The validation loss was generally higher than the training loss, which indicates some degree of overfitting. Again, this was not unexpected due to the limited training data.

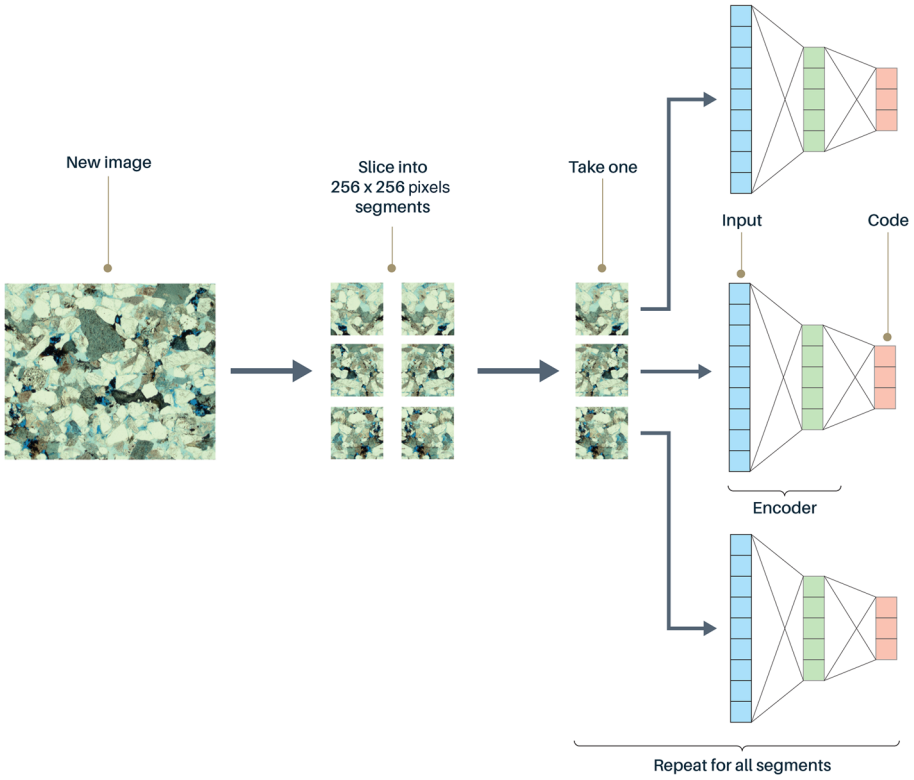

Fig. 7. Every individual 256 × 256 pixel segment is tested separately and the class with the most predictions is taken as the predicted class for the original raw image.

Discussion

The practical implementation of the network architecture, and subsequent training using the thin-section dataset, has demonstrated the potential for this class of algorithm to distinguish between sandstones. Using a limited dataset allowed for a relatively rapid training of the network in order to serve as a proof-of-concept for the methodology. However, the accuracy achieved (just under 50%) is insufficiently good for a practical similarity evaluation tool.

The relatively high recall for the Swan, Hopm and Kinn classes suggests that the model may be able to extract and identify features from these classes more effectively than others. However, the Kinn and Hopm samples in particular had a tendency to be over-represented in the predictions, leading to these classes being falsely attributed to other samples. This may be further evidence of an imbalance in the training set, where a large proportion of the images contained features seen in these over-represented classes. Such an imbalance may have led to a model biased towards certain features and so with the tendency to infer only a limited set of the classes available to it. Interestingly, these false positives seem to be attributed to certain classes. For example, of the 11 predictions for Kinn, five were correct but another five were for Radd images, which suggest an underlying similarity between these two classes. In the case of Radd, the inverse also appeared to be true, in that false negatives for this class were also clustered – of the five instances where the model predicted Radd, the actual image was Gull. Collectively, this appears to suggest an intransitive set of similarities between Radd, Kinn and Gull, where Gull is similar to Radd and Radd is similar to Kinn but Kinn is not similar to Gull.

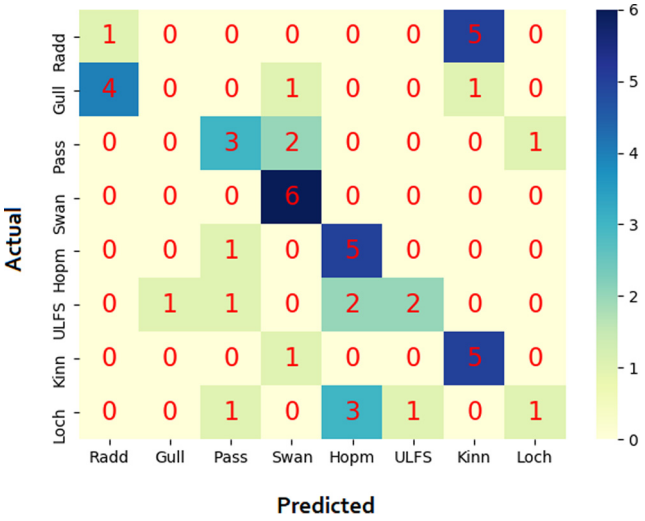

Fig. 8. Summary of model performance obtained by predicting on the test set. For example, the model predicted Radd sandstone correctly in one instance and misclassified it as Kinn in five other instances.

Conversely, classes such as Gull and Loch were barely correctly predicted. The model showed very little tendency to predict these classes at all (incorrectly or correctly). This suggests a model not sufficiently trained to identify the features exhibited by these sandstones either due to a lack of model training or to an absence of sufficiently similar samples in the training set.

It is important to stress that due to the low number of instances for each class in the test set (six images), statistical significance cannot be inferred from these results; therefore, individual classes outperforming the model may have occurred due to chance.

A number of refinements to the method and associated data, which should improve the model accuracy and prospects of creating a viable tool, are discussed below.

Improvements to input data

With just two full-size, high-resolution images used to derive training data for each class in this experiment, the dataset was too small for comprehensive training. A first step to improving performance would be to simply build a dataset with more images for training. Data for other sandstone components could be obtained by a similar method and used in combination with the pore-space

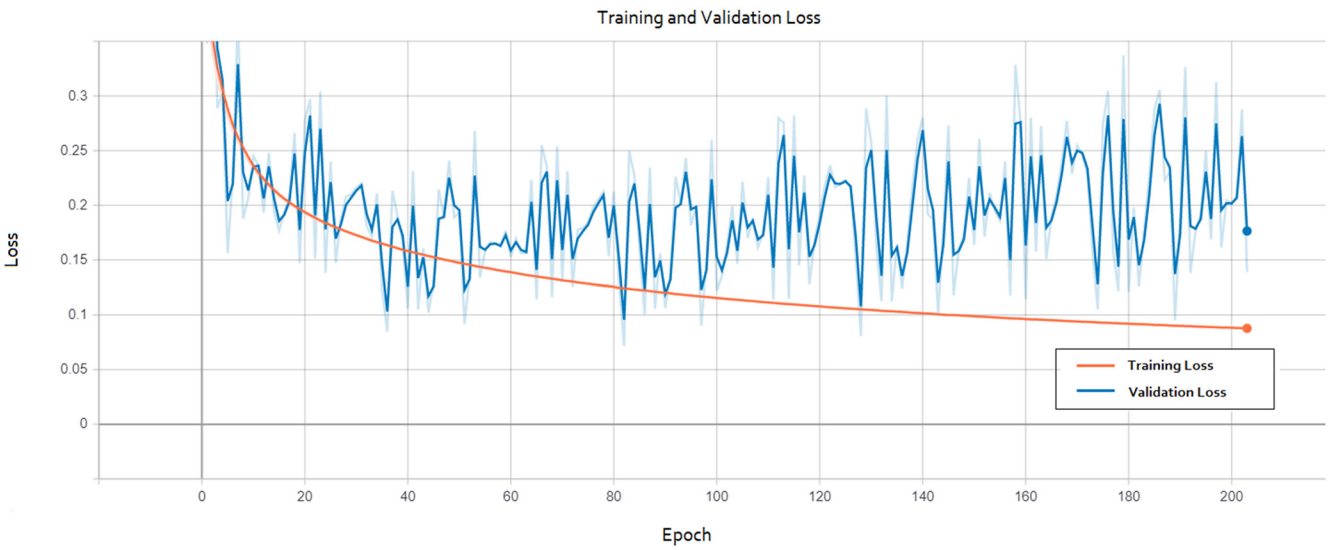

Fig. 9. Plot of the training and validation loss as the training epochs progressed.

data to provide a more powerful basis for discriminating and comparing different sandstones: for example, a map of the opaque mineral component in sandstone thin-section images could be extracted using colour filters optimized for black rather than blue. However, any measures that increase the training data volume come at the expense of additional training time. Table 4 details time estimations for a single training run for different dataset sizes and different hardware solutions. Note that it is common practice to perform training runs on multi-GPU clusters. For multi-GPU computation, the estimated times may be divided by the number of GPUs available in a cluster to give a final time estimate.

Improvements to model architecture

The addition of more filters or additional convolutional layers may be necessary to properly extract useful features from the input images. However, any such changes also involve a trade-off between training time and potential model performance. Although the primary solution to model overfitting will be an improved dataset, regularization (Poernomo and Kang 2018) may provide a secondary means of mitigating this problem. In this case, regularization would add random noise to the input, making the network more robust to anisotropy between input images. To improve training time and overall simplicity, no regularization was used during the training stage of this study; it is therefore unclear whether introducing regularization by means of dropout layers, for example, would yield an improvement. Other potential architecture improvements include alternative activation functions, the addition of batch normalization layers, varied filter and step sizes, and the addition of more fully connected layers. This experiment served to demonstrate a permutation of the network with relatively conventional choices and values for these parameters. A third time-performance trade-off lies in the model hyperparameters. With sufficient experimentation, a set of hyperparameters that result in a well-trained network, even with the addition of new training data, would be desirable. The hyperparameters that defined the training of the autoencoder were tuned to a limited degree for this experiment as different values of batch size and learning rate were considered but learning-rate decay and variations in the activation functions were not. An appropriate implementation of either Bayesian optimization (Wu *et al.* 2019) or a genetic algorithm (Itano *et al.* 2018) would allow for efficient tuning of hyperparameters. In addition, the loss graphs produced by lower learning rates did not level off after the training period of 300 epochs, which suggests longer training runs with lower learning rates would yield improved results. Reliable model testing presented difficulty in obtaining the final result. A successful 'sandstone similarity tool' will need to be able to accommodate many more sandstones than just the eight that were used in this study. An alternative approach to model testing that does not treat each sandstone as a separate class and, instead, directly evaluates how well the model predicts similarity between images would be preferable.

Improvements to model evaluation

The method used for model testing was designed to rapidly and simply assess the success by which the overall experimental approach was able to match images to the correct sandstone. Therefore, the degree to which this was achieved is not simply a reflection of the accuracy of the ANN's performance but also the

Table 4. *Estimated time for a single training run for different dataset sizes and GPU hardware options*

GPU	CNN speed benchmark index (**Dettmers 2020**)	Number of raw images in training dataset	Estimated training time (h)
NVidia GTX1060 (as used in this study)	1	16	3 (as recorded)
		160	30
		1600	300
NVidia RTX 3090 (fastest consumer model)	5.6	16	0.54
		160	5.36
		1600	53.57
NVidia Tesla A100 (fastest single GPU at time of writing)	8.8	16	0.34
		160	3.41
		1600	34.09

distinguishability of the input data. Therefore, our method of evaluating model accuracy is not a true representation of the actual performance of the ANN itself, as the two most similar images in a dataset will not always come from the same sandstone. Due to similar features appearing across multiple sandstones, it is inevitable that in some cases the most similar image will originate from a different sandstone. Inversely, there is considerable variation within natural materials like sandstones. So, relatively dissimilar images may also originate from the same sandstone. Introducing a prior stage of model testing that does not treat each sandstone as a separate class and, instead, directly evaluates how well the model predicts similarity between images in isolation may help to better tune the method. Regardless of distinguishability issues, in general, images from the same sandstone are expected to be more similar to each other than they are to those of another sandstone. Therefore, class prediction accuracy would still be expected to increase with better similarity prediction from the model. Since accuracy significantly higher than that which would be achieved through random chance is demonstrated, this supports the viability of this method for use with a larger dataset and a more finely tuned autoencoder.

An additional approach to model evaluation is possible by attempting to understand how the model operates internally. Although neural networks often appear as 'black boxes', it is possible to examine internal layers to understand how models interpret and classify features. One such method is by the creation of saliency maps that create visual representations of layers, allowing users to pick out manually features that the model identifies as important for classification (Dabkowski and Gal 2017).

Improvements to computing infrastructure

To train on a larger dataset, the model training time would increase proportionally as the number of images to feed into and back-propagate through the network increases. Using a more comprehensive dataset of sandstones would therefore increase the runtime for a single training run to the order of days rather than hours. However, as hyperparameter tuning and/or model architecture changes would require multiple training runs, training on a single consumer-grade machine would become infeasible.

One potential modification that was not explored in this study was altering the size of the input maps. This was due to the previously discussed relationship between input map size and training time, where large maps become prohibitively expensive to train on. Training image size is a compromise between maximum identifiable feature size, the speed at which the network is able to learn such features and the available memory, which is limited by the GPU choice. We were unable to explore this compromise due to the aforementioned hardware limitations. There are a number of options available to researchers wanting to access the required hardware for further investigation. The most readily available is through a number of commercial online cloud service providers, with Amazon EC2 being the most popular at the time of writing.

All of the discussed improvements would require multiple or longer training runs and therefore an increase in compute requirements. Due to cost limitations, only consumer-grade hardware was available for this experiment, so simply carrying out training runs on an up-to-date ML-optimized GPU would be likely to result in a training speed-up of around an order of magnitude (Wang *et al.* 2019). This would be a minimum requirement to fully explore the available training options.

Conclusion

With limited data and compute time, we have devised and trained an ANN architecture that is able to demonstrate a limited, but promising, capability to distinguish between sandstones on the basis of their pore-space geometrical character. Evaluating the ability of a model to assess similarity is difficult but using class prediction accuracy as a proxy the model demonstrated a classification accuracy of 47.9% across an eight-class dataset of Scottish sandstones. This is not yet sufficiently good for a viable tool but this was expected due to the limited size of the dataset. Some classes were predicted correctly more often than others, which suggests that certain sandstones exhibit features that are easier to distinguish between for the model. By using a larger dataset, it may be possible to statistically demonstrate that certain sandstones are consistently easier to predict. In addition, further analysis of the network structure through saliency mapping may allow us to understand why these sandstones are easier to predict, which could guide future optimizations to the model and allow predictive accuracy to be improved for less distinguishable sandstones.

Using the architecture presented here, a model that is able to consistently outperform a human at identifying sandstone samples would require an extensive training set to learn the sometimes subtle differences between sandstones. As this experiment was a broad assessment of the feasibility of ANNs to measure sandstone similarity, only limited variants of the model architecture and hyperparameters were considered. These showed, however, that the chosen architecture was highly sensitive to both the learning rate and selected activation functions. This suggests that future experiments based on varying the architecture and performing more comprehensive hyperparameter tuning will yield a more refined model.

Having shown promise for similarity detection, it is hoped that this or a comparable ANN architecture may find quick uptake for other geoscience problems, and its development within the BGS, as an organization with research interests that span the range of geoscience disciplines, can be expected to facilitate this. Another research angle would be to apply this style of ANN architecture to grouping large sets of image data (including mapping) to permit extrapolation of point observations to larger groups and areas. As the geosciences continue to advance into the realm of 'big data', geologists must equip themselves with tools to review large and complex datasets that human perception will struggle to digest without machine assistance.

Acknowledgements The authors wish to thank Andy Kingdon (BGS) for providing useful suggestions to the manuscript. This manuscript was published with the permission of the Executive Director of the British Geological Survey (UKRI).

Author contributions **AH**: conceptualization (equal), formal analysis (lead), investigation (supporting), methodology (lead), software (equal), writing – original draft (lead), writing – review & editing (equal); **MG**: investigation (equal), project administration (lead), supervision (equal), writing – original draft (equal), writing – review & editing (lead); **PE**: data curation (equal), investigation (supporting), validation (equal), writing – review & editing (equal); **VC**: formal analysis (equal), methodology (supporting), software (equal), validation (equal), writing – review & editing (supporting); **JW**: methodology (equal), software (equal), validation (equal).

Funding This work was funded by the British Geological Survey.

Data availability The datasets generated during and/or analysed during the current study are available in the building-stones repository, https://github.com/BritishGeologicalSurvey/building-stones

Scientific editing by Jonathan Smith; Martin Geach

References

Aziz, I.A. 2013. Comparing point counting & image analysis in sandstone, North Carnarvon Basin, Australia. *In*: *2nd International Conference on Geological and Environmental Sciences*. International Proceedings of Chemical, Biological and Environmental Engineering (IPCBEE), **52**, 20–24.

Berrezueta, E. González, L. *et al.* 2015. Optical image analysis applied to pore network quantification of sandstones under experimental CO_2 injection. Paper H41D-1351 presented at the AGU Fall Meeting Abstracts, 14–18 December 2015, San Francisco, California, USA.

Bradski, G. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, **120**, 122–125.

Buckman, J., Bankole, S., Zihms, S., Lewis, H., Couples, G. and Corbett, P. 2017. Quantifying porosity through automated image collection and batch image processing: case study of three carbonates and an aragonite cemented sandstone. *Geosciences*, **7**, 70, https://doi.org/10.3390/geosciences7030070

Bukharev, A.Y., Budennyy, S.A., Pachezhertsev, A.A., Belozerov, B.V. and Zhuk, E.A. 2018. Automatic analysis of petrographic thin section images of sandstone. *In*: Proceedings of ECMOR XVI – 16th European Conference on the Mathematics of Oil Recovery. European Association of Geoscientists and Engineers (EAGE), Houten, The Netherlands, https://doi.org/10.3997/2214-4609.201802177

Chen, Y., Nishiyama, T. and Ito, T. 2001. Application of image analysis to observe microstructure in sandstone and granite. *Resource Geology*, **51**, 249–258, https://doi.org/10.1111/j.1751-3928.2001.tb00096.x

Cheng, G. and Guo, W. 2017. Rock images classification by using deep convolution neural network. *Journal of Physics: Conference Series*, **887**, 012089, https://doi.org/10.1088/1742-6596/887/1/012089

Chollet, F. *et al.* 2015. *Keras*, Available at: https://github.com/fchollet/keras

Creswell, A., Arulkumaran, K. and Bharath, A.A. 2017. On denoising autoencoders trained to minimise binary cross-entropy, http://arxiv.org/abs/1708.08487

Dabkowski, P. and Gal, Y. 2017. Real time image saliency for black box classifiers. *In*: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17). Curran Associates Inc., Red Hook, NY, 6970–6979.

Dettmers, T. 2020. The Best GPUs for Deep Learning in 2020 – An In-Depth Analysis, https://timdettmers.com/2020/09/07/which-gpu-for-deep-learning/ [accessed 11 February 2021].

Hall, A., Walsh, J. and Christodoulou, V. 2020. Building-Stones. GitLab Repository, GitLab, https://github.com/BritishGeologicalSurvey/building-stones

Han, J., Kamber, M. and Pei, J. 2012. Getting to know your data. *In*: Han, J., Kamber, M. and Pei, J. (eds) *Data Mining*. 3rd edn. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, Boston, MA, 39–82, https://doi.org/10.1016/B978-0-12-381479-1.00002-2

He, K. and Sun, J. 2015. Convolutional neural networks at constrained time cost. *In*: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Institute of Electrical and Electronics Engineers (IEEE), New Piscataway, NJ, 5353–5360.

He, K., Zhang, X., Ren, S. and Sun, J. 2015. Deep residual learning for image recognition, http://arxiv.org/abs/1512.03385

Itano, F., de Abreu de Sousa, M.A. and Del-Moral-Hernandez, E. 2018. Extending MLP ANN hyper-parameters optimization by using genetic algorithm. *In*: *2018 International Joint Conference on Neural Networks (IJCNN)*. Institute of Electrical and Electronics Engineers (IEEE), New Piscataway, NJ, https://doi.org/10.1109/IJCNN.2018.8489520

Karimpouli, S. and Tahmasebi, P. 2019. Segmentation of digital rock images using deep convolutional autoencoder networks. *Computers & Geosciences*, **126**, 142–150, https://doi.org/10.1016/j.cageo.2019.02.003

Kingma, D. and Ba, J. 2014. Adam: A method for stochastic optimization, http://arxiv. org/abs/1412.6980

Le, X., Mei, J., Zhang, H., Zhou, B. and Xi, J. 2020. A learning-based approach for surface defect detection using small image datasets. *Neurocomputing*, **408**, 112–120, https://doi.org/10.1016/j.neucom.2019.09.107

Moreira, A.C., Appoloni, C.R., Mantovani, I.F., Fernandes, J.S., Marques, L.C., Nagata, R. and Fernandes, C.P. 2012. Effects of manual threshold setting on image analysis results of a sandstone sample structural characterization by X-ray microtomography. *Applied Radiation and Isotopes*, **70**, 937–941, https://doi.org/10.1016/j.apradiso.2012.03.001

Poernomo, A. and Kang, D.-K. 2018. Biased dropout and crossmap dropout: learning towards effective dropout regularization in convolutional neural network. *Neural Networks*, **104**, 60–67, https://doi.org/10.1016/j.neunet.2018.03.016

Schmidhuber, J. 2015. Deep learning in neural networks: an overview. *Neural Networks*, **61**, 85–117, https://doi.org/10.1016/j.neunet.2014.09.003

Wang, L. and Rajan, D. 2020. An image similarity descriptor for classification tasks. *Journal of Visual Communication and Image Representation*, **71**, 102847, https://doi.org/10.1016/j.jvcir.2020.102847

Wang, Y.E., Wei, G.-Y. and Brooks, D. 2019. Benchmarking TPU, GPU, and CPU platforms for deep learning, http://arxiv.org/abs/1907.10701

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H. and Deng, S.-H. 2019. Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, **17**, 26–40, https://doi.org/10.11989/JEST.1674-862X.80904120