Support information for “A deep learning approach for regional-scale 3D subsurface structure mapping”

Zhenjiao Jiang [1,2], Dirk Mallants [2], Lei Gao [2], Gregoire Mariethoz [3], Luk Peeters [4]

1 Key Laboratory of Groundwater Resources and Environment, Ministry of Education, College of Environment and Resources, Jilin University, Changchun, 130021, China
2 CSIRO Land & Water, Locked Bag 2, Glen Osmond, SA 5064, Australia
3 University of Lausanne, Faculty of Geosciences and Environment, Institute of Earth Surface Dynamics, Lausanne, Switzerland
4 CSIRO Mineral Resources, Locked Bag 2, Glen Osmond, SA 5064, Australia

1 Neural network structure optimization

The neural network involves multiple hyperparameters, e.g. network depth, width, and filter size, to define the network structure. It is almost impossible to obtain a best combination of these hyperparameters in a way similar to optimising physical-based process models. The hyperparameters are selected in this work by a large amount of try-and-error tests, where one hyperparameter is changed among three to five possible values, while the other hyperparameters are fixed. For each calculation, the performance of the neural network is expressed by (1) computation time cost, and (2) peak-signal-to-noise ratio (PSNR) between the generated images of 3D palaeovalley aquifer index (PAI) and the real image in both training and validation areas (80 km west to the training area), expressed as (Wang and Bovik, 2002):

$$PSNR = -10\log_{10}\left[\frac{1}{M}\sum_{i=1}^{M}(G(z)_i - y_i)^2\right], \qquad \text{(S1)}$$

where M is the number of voxels in the 3D image, $G(z)$ represents the PAI generated by the neural network, y is the PAI calculated from AEM-derived electrical conductivity, which is considered as the ground truth. PSNR is a traditional approach to image quality assessment. A high PSNR represents a high-quality PAI generation. A neural network structure that can result in high PSNR with low computation time cost is desirable.
Following the test results, it was found that four hyperparameters, including output layer size of the encoder, filter size, width, and the loss function of generator, affect the performance of the neural network most significantly. The details are given below.

1.1 Output layer size of encoder

As shown in Fig. S1a, when a large output layer size of 50×50×10 is employed to the encoder, a low PSNR is encountered in both training area and validation area, because the input information of the multiple resolution valley bottom flatness (MrVBF) index are not fully fused. In contrast, if a small layer size of 7×7×5 is employed, a decay in PSNR is still observed when compared to the size of 25×25×5. This suggests that a sufficient number of codes need to be contained in the output layer of the encoder to fully represent the information in the input image of MrVBF. The computation time does not vary significantly with the output layer size. The size of 25×25×5 is employed in this study.

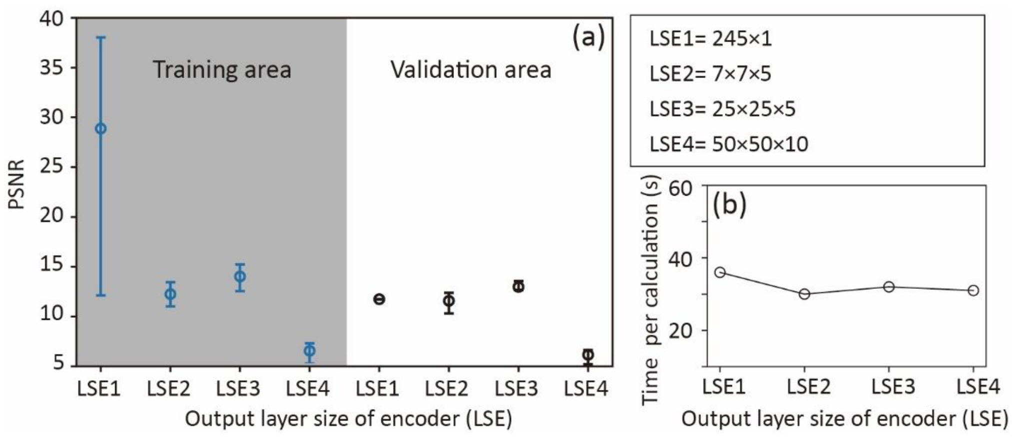


Figure S1. (a) PSNR of 3D PAI in training and validation areas (80 km west to the training area) generated by a neural network with varying output layer sizes in encoder, and (b) the run time for each calculation.

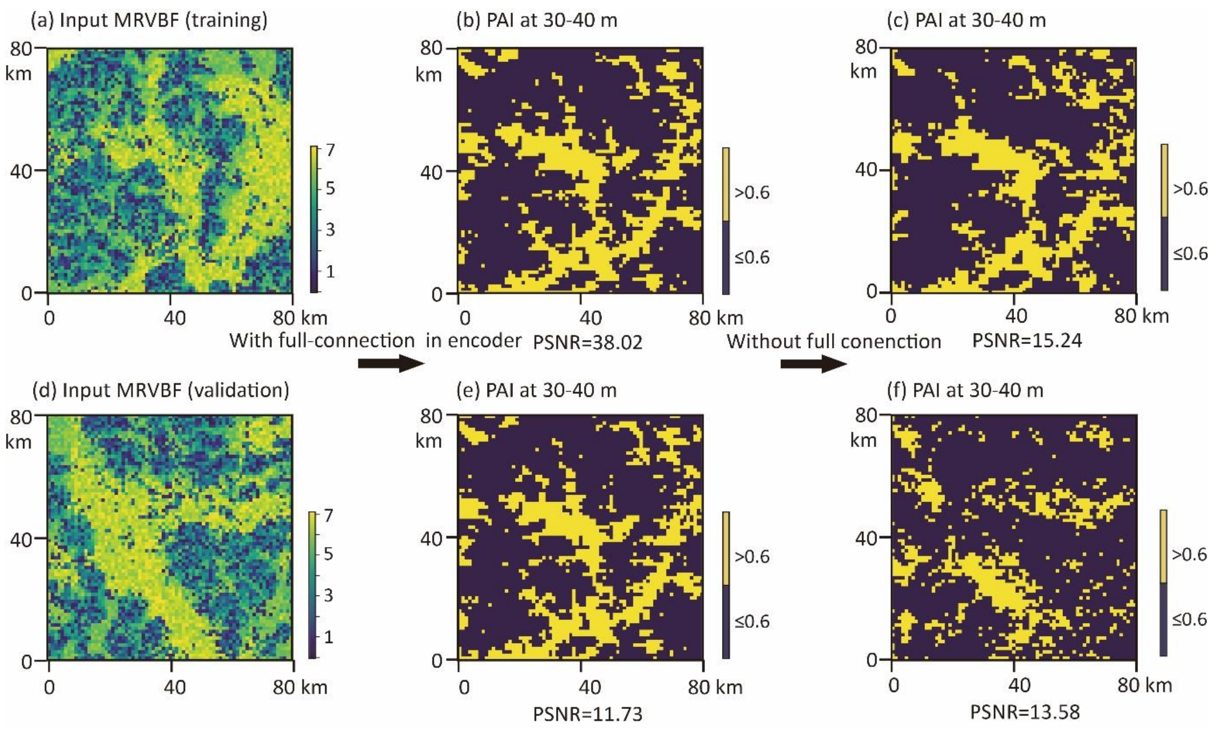

(a) Input MRVBF (training)
(b) PAI at 30-40 m
(c) PAI at 30-40 m
With full-connection in encoder PSNR=38.02
Without full conenction PSNR=15.24
(d) Input MRVBF (validation)
(e) PAI at 30-40 m
(f) PAI at 30-40 m
PSNR=11.73
PSNR=13.58
>0.6
≤0.6
80 km

Figure S2. Input MrVBF in (a) training area and (d) validation area (which is 80 km west to the training area), and (b) and (e) the generated PAI at depth of 30-40 m with full connection operation in the encoder, and (c) and (f) without full connection operation.

Furthermore, following the VAEGAN neural network proposed by (Wu et al., 2016), a full connection layer is used to convert the output layer of the encoder into a low-dimension vector. The performances of the network with full connection layer (LSE1) and without full connection layer (LSE2) are illustrated in Fig. S1. As shown, with the full connection layer, a much higher PSNR (reaching 38) with large fluctuation (13 to 38) can be obtained when training the neural network. However, when applying the trained network to the validation area, a low PSNR is obtained. Indeed, the generated palaeovalley pattern (e.g. at depth 30-40 m) in the validation area (Fig. S2e) is exactly the same as that in the training area (Fig. S2b), when applying the neural network with full connection layers. This means that the full connection operation in the end layer of the encoder makes the input information over-fused, leading to the overfitting problem. In contrast, without the full connection operation, the overfitting problem can be avoided (Fig. S2c and S2f).

1.2 Filter size

Filter size controls the spatial correlation scale able to be addressed by the neural network, which is one of the most important hyperparameters. As demonstrated in Fig. S3, the PSNR of generated PAI increases with the filter size in the decoder, and stabilizes at 12 to 14 when the filter size becomes larger than 4. As the number of weights to be optimized in the neural network increases with the filter size, the computation time increases significantly with the filter size. We here select a filter size of 5 (in the horizontal directions) in the decoder, which can result in a high PSNR in both training and validation areas within an acceptable computation time (~30s per calculation).

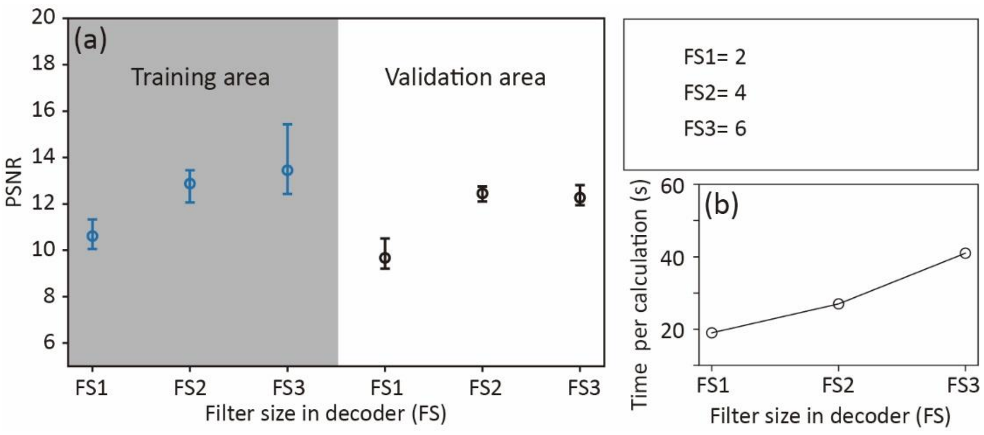


Figure S3. (a) PSNR of 3D PAI in training and validation area (80 km west to the training area) generated by a neural network with varying filter sizes in decoder, and (b) the run time for each calculation.

1.3 Width

Increases of width and depth of the neural network can increase the number of weights, and thus enhance the capability of the network in capturing complex nonlinear relationships between input and output images. However, too many weights to train in the neural network with a limited number of training datasets can lead to a high risk of overfitting. As demonstrated in Fig. S4, as the decoder width increases, the resulting PSNR in both training and validation areas increases. It is also noted that when the maximum width in the decoder is given as 256, PSNR in the training area varies between 12 to 14, because the stochastic weight updating by adaptive moment estimation algorithm (ADAM) (Kingma and Ba, 2014) can result in different weight sets, even under the same network structure. But the PSNR in the validation area concentrates at 12, without fluctuation. This indicates that the generated PAI is insensitive to the variation of the weights in the neural network, and a large number of waste weights are contained. Thus a maximum width of 128 is preferred in this study, which allows the neural network to generate the 3D PAI with relative high accuracy and low computation time.

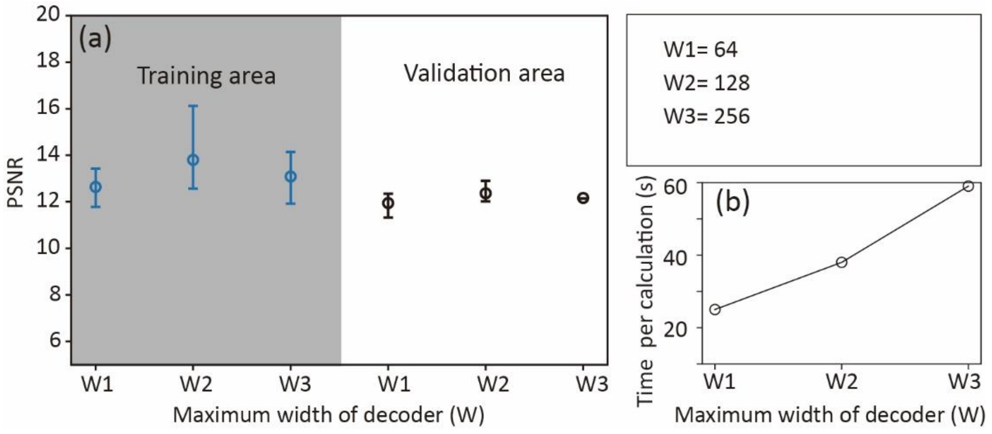


Figure S4. (a) PSNR of 3D PAI in training and validation areas (80 km west to the training area) generated by a neural network with varying network widths in decoder, and (b) the run time for each calculation.

1.4 Loss function

One advantage of the network employed in this study is the ability to update the weights in an adversarial way, based on both the voxel-wise independent criterion (mean square error) and the adversarial criterion. By adjusting the coefficients in the loss function of generator in Eq. 6, the PSNR in training and validation areas are calculated in Fig. S5. As shown, as the coefficient *b* of voxel-wise independent criterion increases from 0 to 0.1, the performance of the neural work enhances. The PSNR of the trained neural working to generate the PAI increases from 12 to 14 in the training area and from 12 to 12.5 in the validation area. Further increases in the coefficient of the voxel-wise independent criterion to 0.5 can increase the PSNR in the training area, but leads to a decrease in PSNR in the validation area. This indicates that the overfitting problem occurs. We here use the coefficient of voxel-wide independent criterion of 0.1 and that of adversarial criterion of 0.9.

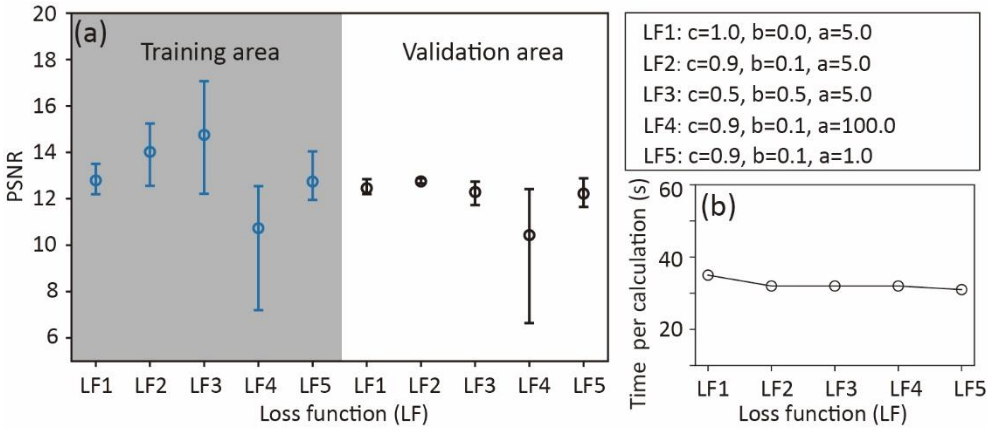


Figure S5. (a) PSNR of 3D PAI in training and validation areas (80 km west to the training area) generated by a neural network with different loss functions for updating the weights in generator, and (b) the run cost for each calculation.

Moreover, increasing the coefficient *a* of Kullback–Leibler divergence from 1 to 100 can give priority to weights updating in the encoder. The encoder is used to fuse the information of input images to a low-dimension code following the standard normal distribution, while the decoder reconstructs the 3D PAI from these low-dimension codes. When a large coefficient of *a* = 100 is applied, the network may only update the weights in the

encoder, rather than in the decoder. Thus, the capability of the trained network in PAI generation is weak, with a low PSNR in both training and validation areas.
Following the try-and-error tests above, we here select a set of hyperparameters (Table S1) that allow the neural network to produce an acceptable 3D PAI (with the PNSR in training area reaching 16, in validation area 14, and the computation time is less than 30s per calculation). However, these hyperparameters are not necessarily the theoretically best ones.

Table S1. Hyperparameters defining the convolution neural network structure in this study.

Component		**Depth**	**Layer**	**Operation**	**Filter size**	**Strike**	**Width**	**Output image size**	**Activation function**
Generator	Encoder	4	Layer 1	convolution	4×4×2	4×4×1	64	200×200×10	Leaky relu
			Layer 2	convolution	4×4×2	4×4×1	32	50×50×10	Leaky relu
			Layer 3	convolution	4×4×2	2×2×2	32	25×25×5	Leaky relu
			Layer 4	convolution	4×4×2	1×1×1	1	25×25×5	None
	Decoder	6	Layer 1	deconvolution	5×5×2	1×1×2	128	25×25×10	Leaky relu
			Layer 2	convolution	5×5×2	1×1×1	64	25×25×10	Leaky relu
			Layer 3	deconvolution	5×5×2	4×4×1	32	100×100×10	Leaky relu
			Layer 4	convolution	5×5×2	1×1×1	32	100×100×10	Leaky relu
			Layer 5	deconvolution	5×5×2	2×2×1	16	200×200×10	Leaky relu
			Layer 6	convolution	5×5×2	1×1×1	1	200×200×10	Sigmoid
Discriminator		4	Layer 1	convolution	4×4×2	2×2×2	128	100×100×5	Leaky relu
			Layer 2	convolution	4×4×2	2×2×1	64	50×50×5	Leaky relu
			Layer 3	convolution	4×4×2	2×2×1	32	25×25×5	Leaky relu
			Layer 4	convolution	4×4×2	4×4×1	1	7×7×5	Sigmoid

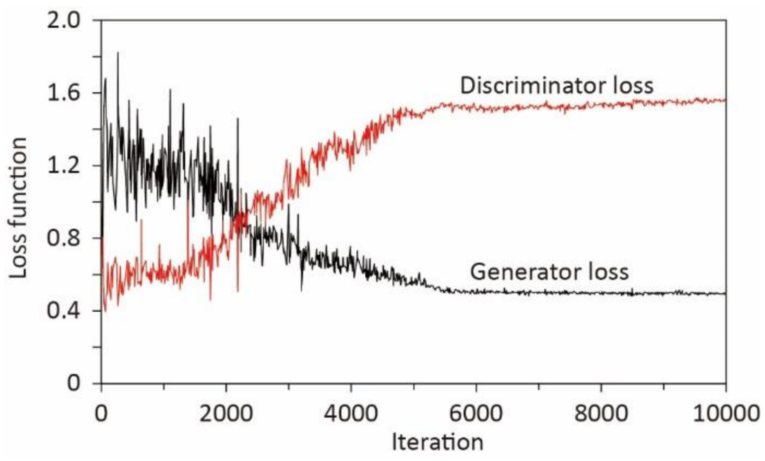
To covert 2D input image to 3D PAI images with 10 layers, 2D input image were first repeated at 10 layers before convolution, and then 3D filters were employed for convolution and deconvolution.

Leaky Relu: $f(x) = \max(x, 0.2x)$; Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$, and filter and strike expressed by e.g. 4×4×2 representing the value in eastward, northward and vertical direction, respectively.

Coefficients in generator loss function (Eq. 7) are 0.9, 0.1 and 5, respectively.

2 Neural network training

We monitor the loss functions when training the neural network to verify that the network is being adequately trained. For the generative adversarial training, there is no critical standard values required for the loss functions of the generator and discriminator. Nevertheless, it is reasonable that during the training processes, the discriminator and generator loss functions vary in an adversarial way, and both stabilize at constant values, indicating the network has been adequately trained. As shown in Fig. S6, the generator loss function decreases from about 1.5 to 0.5, while the discriminator loss increases from 0.5 to 1.5. Both stabilize after 5800 iterative calculations. The decrease of generator loss corresponds to the increase of discriminator loss, suggesting that the network was adequately trained.



110 **Figure S6. Adversarial variations of generator and discriminator loss functions during the training of the neural network.**

References

Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014. 2014.

115 Wang, Z. and Bovik, A. C.: A universal image quality index, IEEE signal processing letters, 9, 81-84, 2002.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, 2016, 82-90.