GEOPHYSICS, VOL. 85, NO. 4 (JULY-AUGUST 2020); P. WA159–WA171, 19 FIGS.
10.1190/GEO2019-0282.1

Missing well log prediction using convolutional long short-term memory network

Nam Pham[1], Xinming Wu[2], and Ehsan Zabihi Naeini[3]

ABSTRACT

Reservoir characterization involves integration of different types of data to understand the subsurface rock properties. To incorporate multiple well log types into reservoir studies, estimating missing logs is an essential step. We have developed a method to estimate missing well logs by using a bidirectional convolutional long short-term memory (bidirectional ConvLSTM) cascaded with fully connected neural networks. We train the model on 177 wells from mature areas of the UK continental shelf (UKCS). We test the trained model on one blind well from UKCS, three wells from the Volve field in the Norwegian continental shelf, one well from the Penobscot field in the Scotian shelf offshore Canada, and one well from the Teapot Dome data set in Wyoming. The method takes into account the depth trend and the local shape of logs by using ConvLSTM architecture. The method is examined on sonic log prediction and can produce an accurate prediction of sonic logs from gamma-ray, density, and neutron porosity logs. The advantages of our method are that it is not applied on an interval by interval basis like rock-physics-based methods and it also outputs the uncertainties facilitated by dropout layers and Monte Carlo sampling at inference time.

INTRODUCTION

The integration of multiple geophysical well logs with seismic data can greatly reduce the ambiguity of geologic interpretation and help to construct a better hydrocarbon reservoir model. However, some types of logs might be missing at some wells in an area of interest because of cost limitations or borehole problems. For example, seismic-well tie requires sonic and density logs (White and Simm, 2003; Herrera et al., 2014; Muñoz and Hale, 2015; Wu and Caumon, 2017), but either log sometimes can be missing, so they have to be estimated from other available log types (Bader et al., 2019).

Several statistical and empirical models have been used to estimate missing well logs. Gardner's equation provides a reasonable relationship between sonic and density for brine-saturated rock types (Gardner et al., 1974). The Faust (1953) and Smith (2007) methods can provide interval relationships between resistivity and sonic logs. Castagna et al. (1985) and Greenberg and Castagna (1992) propose empirical relationships to calculate shear wave (S-wave) velocity from compressional wave (P-wave) velocity. There are many other rock physics models depending on the reservoir for conventional and unconventional settings. These models can produce useful log predictions; however, they are interval-based, dependent on rock types, and their calibration requires human time and expertise.

An alternative data-driven method is to use nearby training wells with a complete suite of logs to predict missing logs at a specific well. This task is particularly suitable for artificial neural networks. Saggaf and Nebrija (2003) use regularized back-propagation neural networks to estimate missing sonic from gamma-ray, neutron porosity, and density logs. Rolon et al. (2009) and Salehi et al. (2017) use fully connected neural networks (FCNNs) for predicting nonrecorded logs from existing logs. However, FCNNs only produce a point-to-point mapping from input logs to output logs. Rock properties often demonstrate a trend with depths, which is important for geologic studies. Recurrent neural networks (RNNs) consider internal input from previous step (e.g., trend) and external inputs (other available log types). Zhang et al. (2018) use RNNs to generate synthetic well logs; however, the model does not take into account the local shaping information of logs and the uncertainty of the predicting model is not quantified.

We propose a method to estimate missing logs by using a bidirectional convolutional long short-term memory (ConvLSTM)

Manuscript received by the Editor 2 May 2019; revised manuscript received 8 February 2020; published ahead of production 17 March 2020; published online 14 May 2020.
[1]The University of Texas at Austin, Texas, USA. E-mail: nam-pham@utexas.edu.
[2]University of Science and Technology of China, School of Earth and Space Sciences, Hefei, China. E-mail: xinmwu@ustc.edu.cn (corresponding author).
[3]Ikon Science, Surbiton, London, UK. E-mail: ehsanzabihi@yahoo.com.

cascaded with FCNNs. The ConvLSTM improves the missing log predictions by incorporating the local shape of logs related to different geologic units. We further add dropout layers to quantify the uncertainty of the model. The output from our method is a Monte Carlo simulation of the predicted log, whose variance is the uncertainty of the model. We apply our method for predicting missing sonic logs from gamma-ray, density, and neutron porosity logs. We test our approach on six blind wells from different geologic areas and compare the results against the actual sonic logs and Gardner's estimations (Gardner et al., 1974). Gardner's estimations are obtained by dividing the wells into different lithologies based on changes in gamma-ray and density logs and then applying different parameters based on rock types.

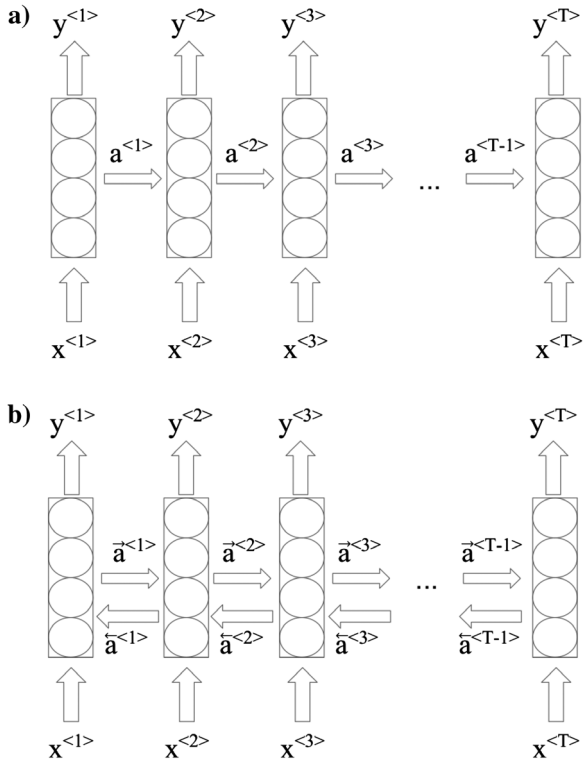


Figure 1. The architecture of (a) RNN and (b) BRNN.

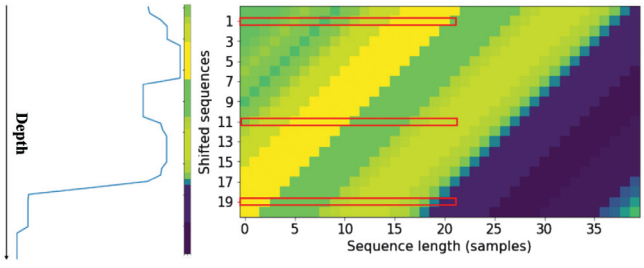


Figure 2. A 1D ConvLSTM. Each shifted log sequence is demonstrated by the colorbar on the left. Red boxes are 1D convolutional filters.

METHOD

Convolutional LSTM architecture

RNN is an internal self-looped deep-learning architecture used for natural language processing with sequential data. The output of an RNN at each time step is affected by the input of the current step and the input from previous steps (Figure 1a). Let $\mathbf{x}$ be the input sequence and $\mathbf{y}$ be the output sequence with length T. The terms $\mathbf{x}^{\langle t\rangle}$ and $\mathbf{y}^{\langle t\rangle}$ are the samples at time t of input and output sequence, respectively, $\mathbf{a}^{\langle t\rangle}$ is the activation output of the network. The forward propagation of an RNN is

$$\begin{aligned}\mathbf{a}^{\langle t\rangle} &= g(\mathbf{W}_{aa}\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{ax}\mathbf{x}^{\langle t\rangle} + \mathbf{b}_a),\\ \mathbf{y}^{\langle t\rangle} &= g(\mathbf{W}_{ya}\mathbf{a}^{\langle t\rangle} + \mathbf{b}_y),\end{aligned} \tag{1}$$

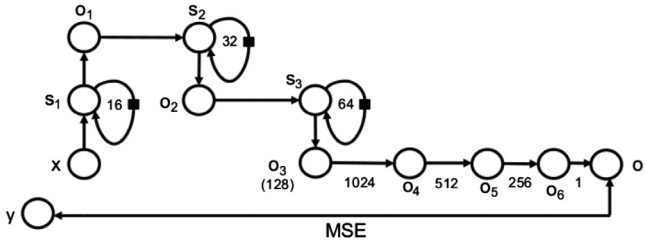


Figure 3. Our proposed model. The terms o_1, o_2, and o_3 are BRNN outputs. The terms s_1, s_2, and s_3 are ConvLSTM cells. The terms o_4, o_5, and o_6 are FCNN outputs. The term o is the final output. The term x is the input, and y is the true output.

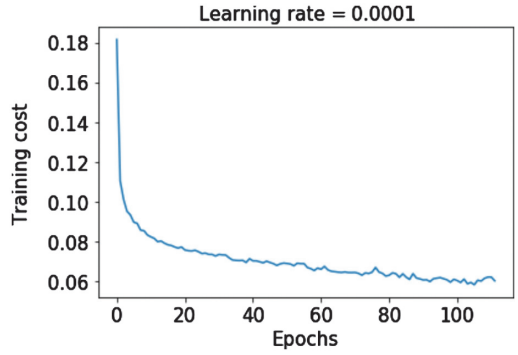


Figure 4. Training MSE.

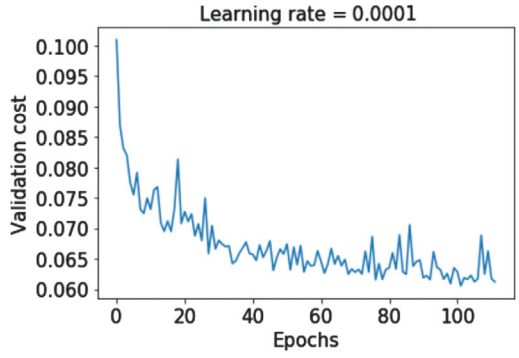


Figure 5. Validation MSE.

where $\mathbf{W}_{aa}$, $\mathbf{W}_{ax}$, $\mathbf{W}_{ya}$, $\mathbf{b}_a$, and $\mathbf{b}_y$ are trainable parameters, and g is an activation such as the hyperbolic tangent, rectified linear unit (ReLU), or leaky ReLU function. Bidirectional RNNs (BRNNs) are an acyclic graph with recurrent layers going forward and backward in time (Figure 1b), which use inputs from both earlier steps and later steps in the sequence (Schuster and Paliwal, 1997). The output of the network at each time step is calculated by activation outputs $\leftarrow\mathbf{a}^{\langle t\rangle}$ and $\vec{\mathbf{a}}^{\langle t\rangle}$

$$\begin{aligned}
\leftarrow\mathbf{a}^{\langle t\rangle} &= g(\leftarrow\mathbf{W}_{aa}\leftarrow\mathbf{a}^{\langle t-1\rangle} + \leftarrow\mathbf{W}_{ax}\mathbf{x}^{\langle t\rangle} + \leftarrow\mathbf{b}_a),\\
\rightarrow\mathbf{a}^{\langle t\rangle} &= g(\rightarrow\mathbf{W}_{aa}\rightarrow\mathbf{a}^{\langle t+1\rangle} + \rightarrow\mathbf{W}_{ax}\mathbf{x}^{\langle t\rangle} + \rightarrow\mathbf{b}_a),\\
\mathbf{y}^{\langle t\rangle} &= g(\leftarrow\mathbf{W}_{ya}\leftarrow\mathbf{a}^{\langle t\rangle} + \rightarrow\mathbf{W}_{ya}\rightarrow\mathbf{a}^{\langle t\rangle} + \mathbf{b}_y). \qquad (2)
\end{aligned}$$

The LSTM is a type of RNN that captures very long-term dependencies (Hochreiter and Schmidhuber, 1997) and is suitable for working with long and densely sampled well logs. The activation outputs are affected by different gates that decide whether to remove or add information to the cell states. A candidate value of memory cell state at each time step is computed from the activation output of the previous time step and external input at the current time step

$$\tilde{\mathbf{c}}^{\langle t\rangle} = g(\mathbf{W}_{ca}\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{cx}\mathbf{x}^{\langle t\rangle} + \mathbf{b}_c). \qquad (3)$$

The update and forget gate decide whether to update the cell state with the candidate

$$\begin{aligned}
\Gamma_u &= \sigma(\mathbf{W}_{ua}\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{ux}\mathbf{x}^{\langle t\rangle} + \mathbf{b}_u),\\
\Gamma_f &= \sigma(\mathbf{W}_{fa}\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{fx}\mathbf{x}^{\langle t\rangle} + \mathbf{b}_f),\\
\mathbf{c}^{\langle t\rangle} &= \Gamma_u\tilde{\mathbf{c}}^{\langle t\rangle} + \Gamma_f\mathbf{c}^{\langle t-1\rangle}, \qquad (4)
\end{aligned}$$

where σ is the sigmoid function. The activation output at a time step is calculated using the output gate Γ_o

$$\begin{aligned}
\Gamma_o &= \sigma(\mathbf{W}_{oa}\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{ox}\mathbf{x}^{\langle t\rangle} + \mathbf{b}_o),\\
\mathbf{a}^{\langle t\rangle} &= \Gamma_o g(\mathbf{c}^{\langle t\rangle}). \qquad (5)
\end{aligned}$$

We use leaky ReLU (Xu et al., 2018) as activation function g in our experiment.

To consider the local shape of logs, which is related to the depositional facies (Cant, 1994), we represent each temporal point by a shape descriptor to encode local shaping information around a point and apply ConvLSTM to capture the spatiotemporal correlations (Shi et al., 2015; Zhao and Itti, 2018). The ConvLSTM replaces the dot product in the state-to-state and input-to-state transitions with a convolution operator ($*$) and calculates the future state of a certain cell in a spatial grid by the inputs and past states of its local neighbors

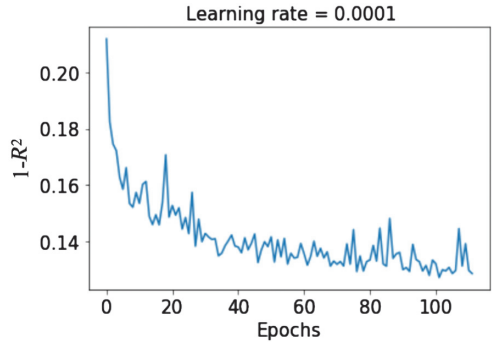


Figure 6. Validation $1 - R^2$.

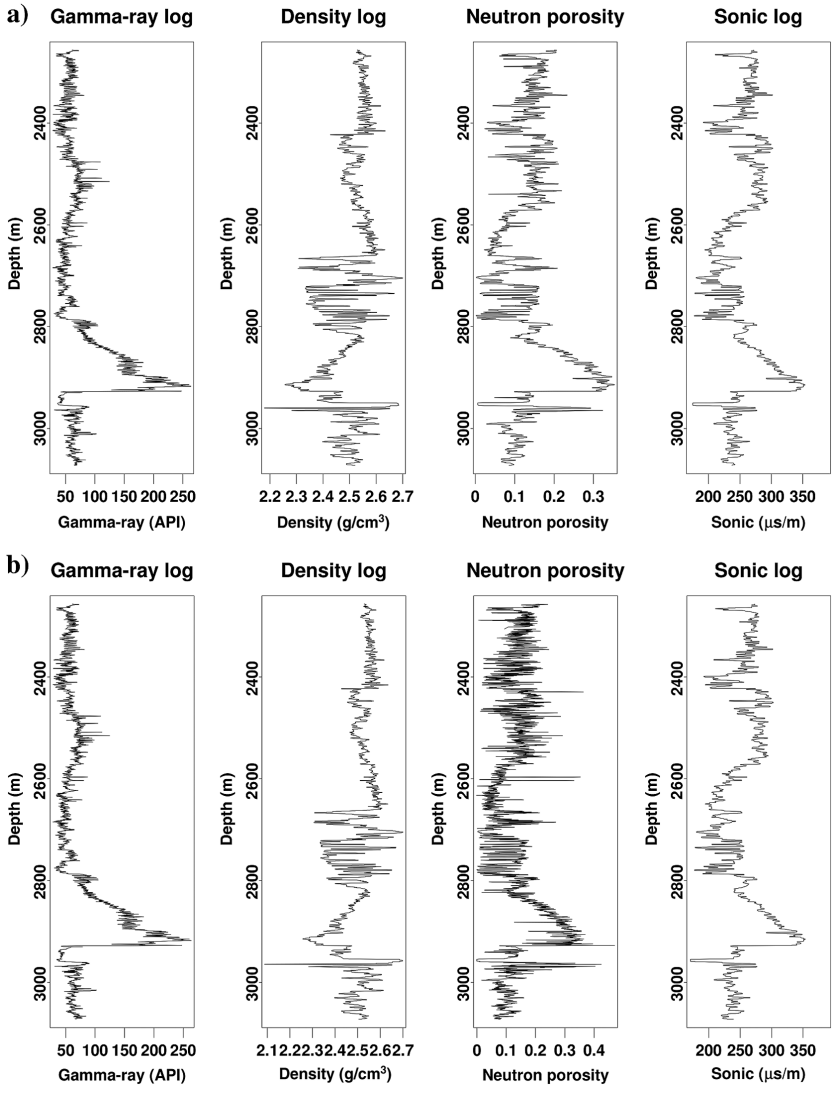


Figure 7. Validation well gamma-ray log, density log, neutron porosity log, and sonic log (a) after being despiked and (b) before being despiked.

$$\tilde{\mathbf{c}}^{\langle t\rangle} = g(\mathbf{W}_{ca} * \mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{cx} * \mathbf{x}^{\langle t\rangle} + \mathbf{b}_c),$$
$$\Gamma_u = \sigma(\mathbf{W}_{ua} * \mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{ux} * \mathbf{x}^{\langle t\rangle} + \mathbf{b}_u),$$
$$\Gamma_f = \sigma(\mathbf{W}_{fa} * \mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{fx} * \mathbf{x}^{\langle t\rangle} + \mathbf{b}_f),$$
$$\Gamma_o = \sigma(\mathbf{W}_{oa} * \mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{ox} * \mathbf{x}^{\langle t\rangle} + \mathbf{b}_o). \quad (6)$$

The shape descriptor is a 2D window centered at the log, which also consists of sequences shifted forward and backward. The backward-shifted sequences provide the previous "states" of each log sample in depth. However, the forward-shifted sequences provide the latter "states" of each log sample in depth. The original dimension of 1D input data is (a_1, a_2, a_3), where a_1 is the number of input sequences, a_2 is the length of input sequences, and a_3 is the number of log types in the input. For our method, the dimension of the shape descriptor is (a_1, a_4, a_2, a_3), where a_4 is the number of shifted sequences in the 2D window including the original sequence. In particular, the data of our experiment are 1D gamma-ray, density, and neutron porosity logs. Each log sequence has 61 samples, which is determined by the length of the shortest well in the training area. We decide to choose the length of the input sequence based on the shortest well length so that the samples from different wells are not mixed together. We have a total of 26,062 sequences extracted from 177 training wells. Three types of input logs are stacked along the last dimension to create a data set with size (26,062, 60, and 3). We start from sample 10 and continue until the tenth sample from the end and extract local windows of 21 samples, which are centered at each sample of the sequence. We do not need to worry about padding the boundaries. The size of each local window is now chosen by trial and error with a constraint of a Titan Xp GPU com-

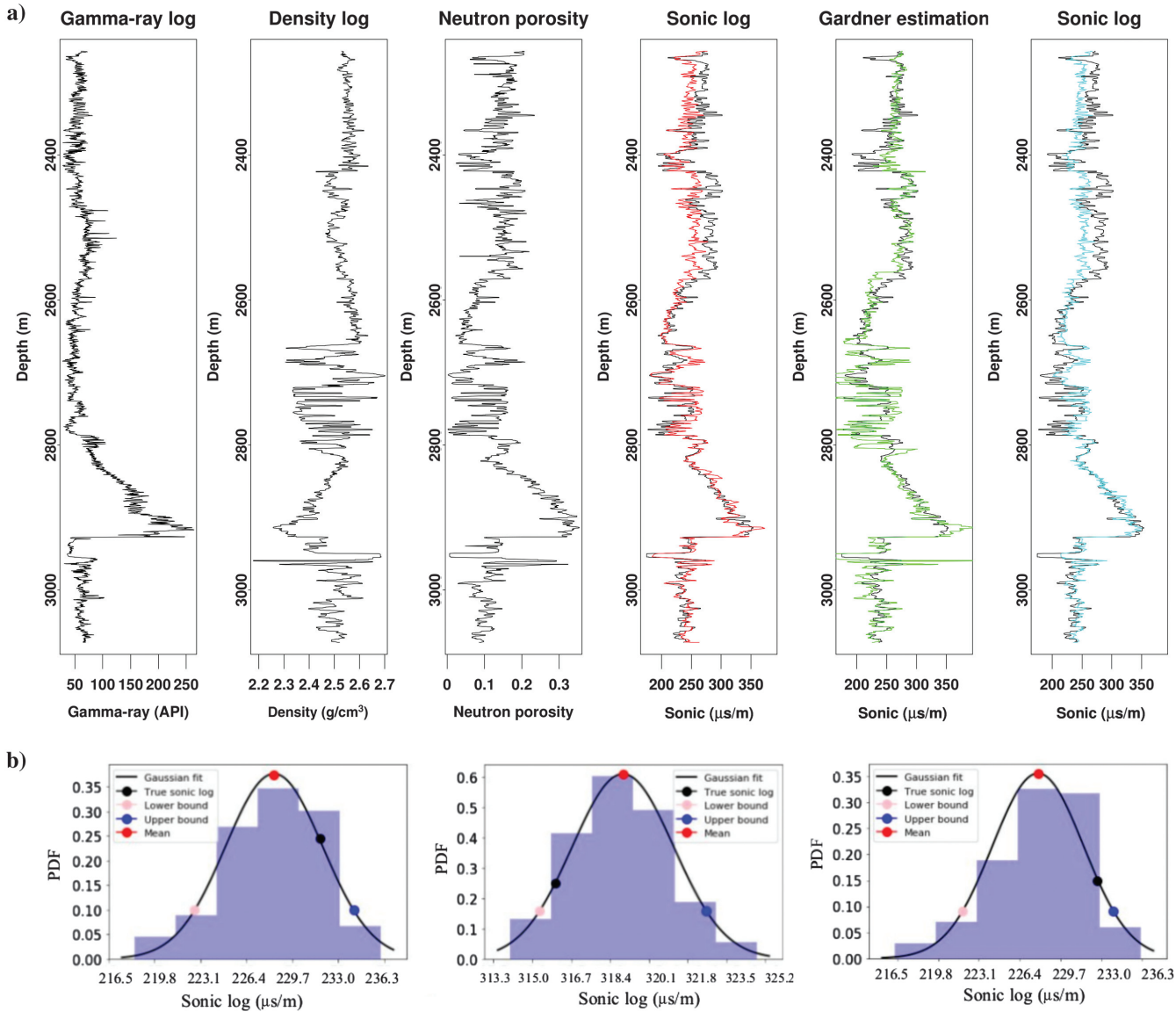


Figure 8. (a) Validation well: gamma-ray log, density log, neutron porosity log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in the training well: 2264.5 m (top row-left), 2888 m (top row-right), and 2956.4 m (bottom row-center). The vertical axes are the probability density functions.

putational resource. Because the temporal relationship between shifted sequences captured by the LSTM architecture is the geologic trend, the window size should be large enough but not too large to avoid vanishing and exploding gradients. The data set has a new dimension of (26,062, 21, 40, and 3). A convolutional filter is slid across the entire length of the original log to capture the local shape (Figure 2). The weight and bias of the convolutional filter are shared between different shifted sequences of a log type.

Our proposed deep-learning architecture has a BRNN, with three forward and three backward ConvLSTM blocks, which is cascaded with four layers of an FCNN (Figure 3). The forward block uses the previous local information at $\langle t-1\rangle$ of a log captured by a convolutional filter and the shifted log to predict the current information at $\langle t\rangle$. However, the backward block uses the latter local information at $\langle t+1\rangle$ of a log captured by a convolutional filter and the shifted log to predict the current information at $\langle t\rangle$. The blocks in the BRNN have 16, 32, and 64 neurons, respectively. The BRNN produces 64 feature maps in each forward and backward direction, which contain spatiotemporal information. We concatenate them together to obtain 128 feature maps; each feature map is a 1D sequence. They are fed into four layers of an FCNN with 1024, 512, 256, and 1 neuron, respectively.

We want to extract more features from the FCNN layers to help the network in learning, but it also can make the network easily overfitting. To control overfitting, we add a variational recurrent dropout (Gal and Ghahramani, 2015b) for output a in equation 5 after each ConvLSTM cell in the BRNN, which retains a random 50% of the neurons. The same dropout mask is repeated at each shifted version of the logs, which force the model not to rely on single shifted versions of the logs and capture the geologic trend better. This dropout technique has been applied successfully on RNN without drowning the signal (Gal and Ghahramani, 2015b). We further add two dropout layers (Srivastava et al., 2014; Gal and Ghahramani, 2016; Pham et al., 2019) keeping 50% of the neurons randomly, between the 1024-neuron, 512-neuron, and 256-neuron FCNN layer, to control overfitting and quantify the model uncertainty. The dropout layer imposes a Bernoulli distribution more than the filter weights, and it is used to model the weights probabilistically, which will be discussed below. We believe that the blocks in BRNN are important in capturing geologic trends and extracting the local shape of the logs, so they may not be appropriate to model probabilistically. Therefore, we turn off the variational recurrent dropout at the test time and only keep the dropout at deeper layers. We use Adam optimization (Kingma and Ba, 2015) with a learning rate of 0.0001 to minimize the mean squared error (MSE) between the predicted log and the target log.

To prevent the network from exploding and vanishing gradients because of stacking ConvLSTM cells, we normalize the update gate Γ_u, forget gate Γ_f, output gate Γ_o, and new cell value $\mathbf{c}^{\langle t\rangle}$ in equation 4 using layer normalization (Ba et al., 2016) so that the output when calculating gates is not trapped in the saturated area of the sigmoid function. We also use an orthogonal matrix to initialize the weights of BRNN so that the eigenvalues of the weight matrix neither exploded nor vanished, and the gradients can back-propagate more effectively (Vorontsov et al., 2017).

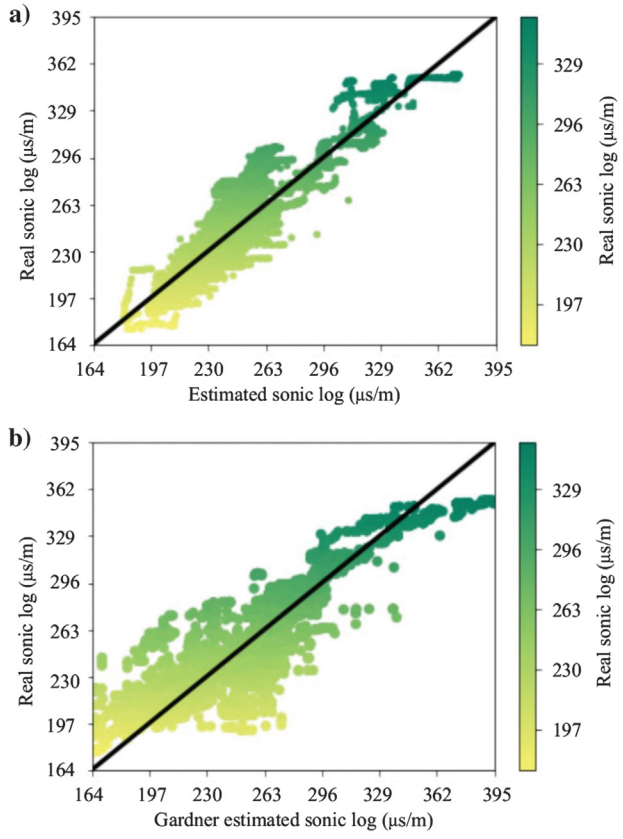


Figure 9. Validation well crossplots. (a) The real sonic log against the predicted sonic log. (b) The real sonic log against Gardner's estimation.

Model uncertainty

To understand the uncertainty of neural networks, we can express the training and prediction phase of deep learning using Bayes' rule (Ghahramani, 2015). Learning from data is the transformation of the prior probability distributions defined before observing the data into the posterior distribution defined after observing the data

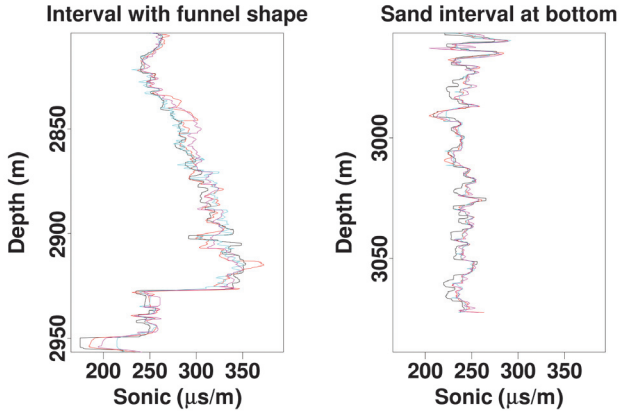


Figure 10. Different intervals of interest in the validation well: true sonic log (black), bidirectional ConvLSTM output (red), bidirectional LSTM output (light blue), and unidirectional ConvLSTM output (purple).

$$P(\theta|D,m) = \frac{P(D|\theta,m)P(\theta|m)}{P(D|m)}, \quad (7)$$

where θ is the model parameters, D is the training data, and m is the model. The prediction process also can be expressed by Bayes' rule:

$$P(x|D,m) = \int P(x|\theta,D,m)P(\theta|D,m)d\theta, \quad (8)$$

where x is a new input. Therefore, the uncertainty of neural networks can come from uncertainty of the model parameters, data, and model structure. We use a dropout layer at test time to calculate the uncertainty from the model parameters for our missing well-log prediction problem.

If we have training inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and corresponding outputs $\mathbf{y}_1, \ldots, \mathbf{y}_N$, the neural network is trying to estimate a function $\mathbf{y} = \mathbf{f}(\mathbf{x})$. Given new input $\mathbf{x}^*$, we have an expression for Bayes' rule (Gal and Ghahramani, 2015a):

$$P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) = \int P(\mathbf{y}^*|\mathbf{f}^*)P(\mathbf{f}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})d\mathbf{f}^*. \quad (9)$$

We can approximate the integral by conditioning the neural network on a finite set of random variables ω. Function f is defined through the weights of the neural network, so we can have $\omega = (\mathbf{W}_i)_{i=1}^{L}$, where L is the number of layers in the network

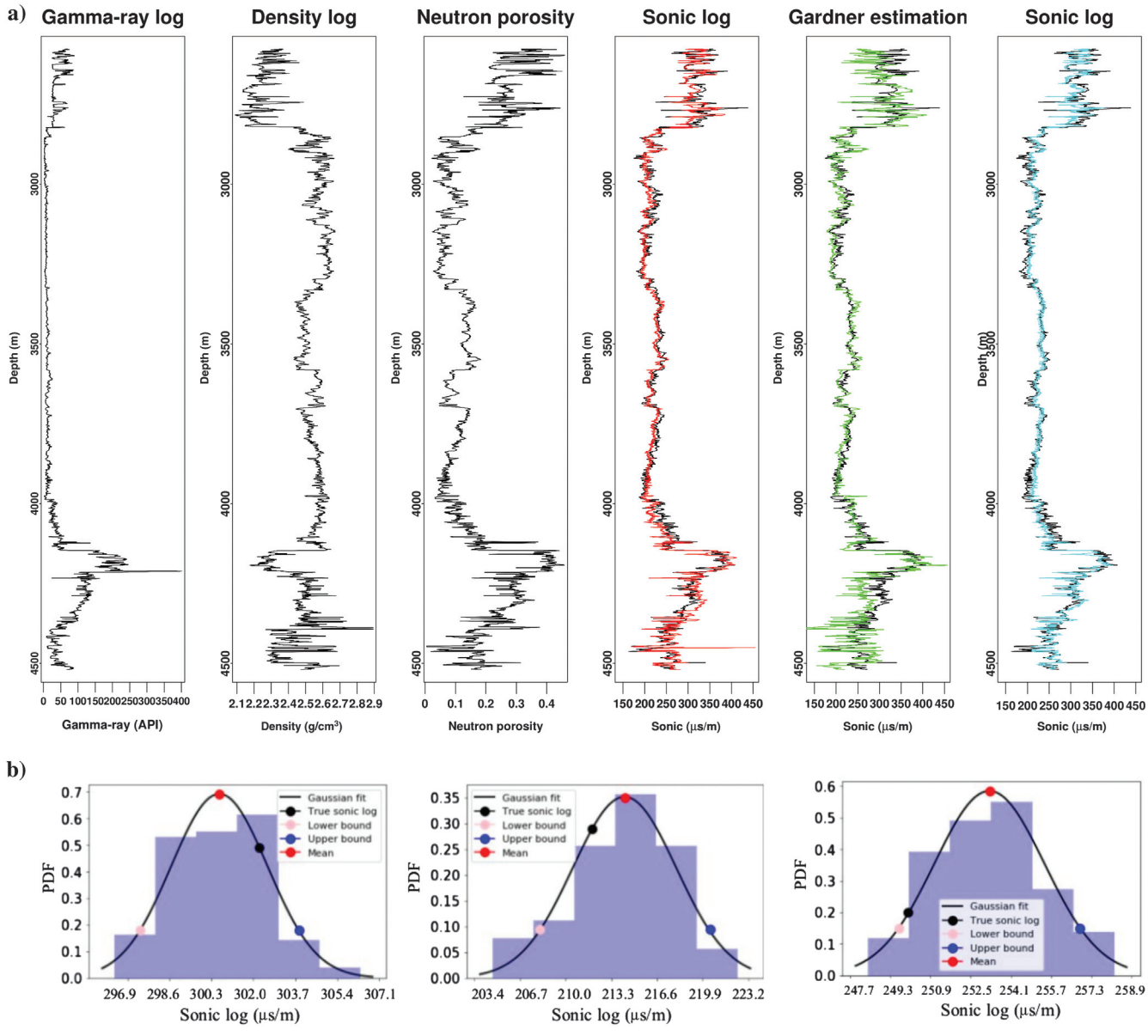


Figure 11. (a) Testing well 1: gamma-ray log, density log, neutron porosity log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in testing well 1: 2590 m (top row–left), 3877 m (top row–right), and 4387 m (bottom row–center). The vertical axes are the probability density functions.

$$P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) = \int P(\mathbf{y}^*|\mathbf{f}^*)P(\mathbf{f}^*|\mathbf{x}^*,\omega)P(\omega|\mathbf{X},\mathbf{Y})d\mathbf{f}^*d\omega. \tag{10}$$

We approximate $P(\omega|\mathbf{X},\mathbf{Y})$ by a variational distribution $Q(\omega)$ with a condition that the Kullback-Leibler divergence (Kullback, 1959) between two distributions is minimized

$$Q(\mathbf{y}^*|\mathbf{x}^*) = \int P(\mathbf{y}^*|\mathbf{f}^*)P(\mathbf{f}^*|\mathbf{x}^*,\omega)Q(\omega)d\mathbf{f}^*d\omega. \tag{11}$$

We can define $Q(\mathbf{W}_i)$ for every layer i as

$$\mathbf{W}_i = \mathbf{M}_i \cdot \mathrm{diag}([z_{i,j}]_{j=1}^{K_i})z_{i,j} \sim \mathrm{Bernoulli}(P_i)$$
$$\text{for } i = 1, \ldots, L, \quad j = 1, \ldots, K_{i-1}, \tag{12}$$

where K_i is the number of neurons in layer i, $z_{i,j}$ are Bernoulli distributed random variables with probabilities P_i, and $\mathbf{M}_i$ are variational parameters to be optimized.

Sampling from this $Q(\mathbf{W}_i)$ is identical to performing dropout on layer i in a network with $(\mathbf{M}_i)_{i=1}^{L}$ as weights. The dropout layer randomly removes units within the network with a predefined probability. Gal and Ghahramani (2015a) prove that minimizing the

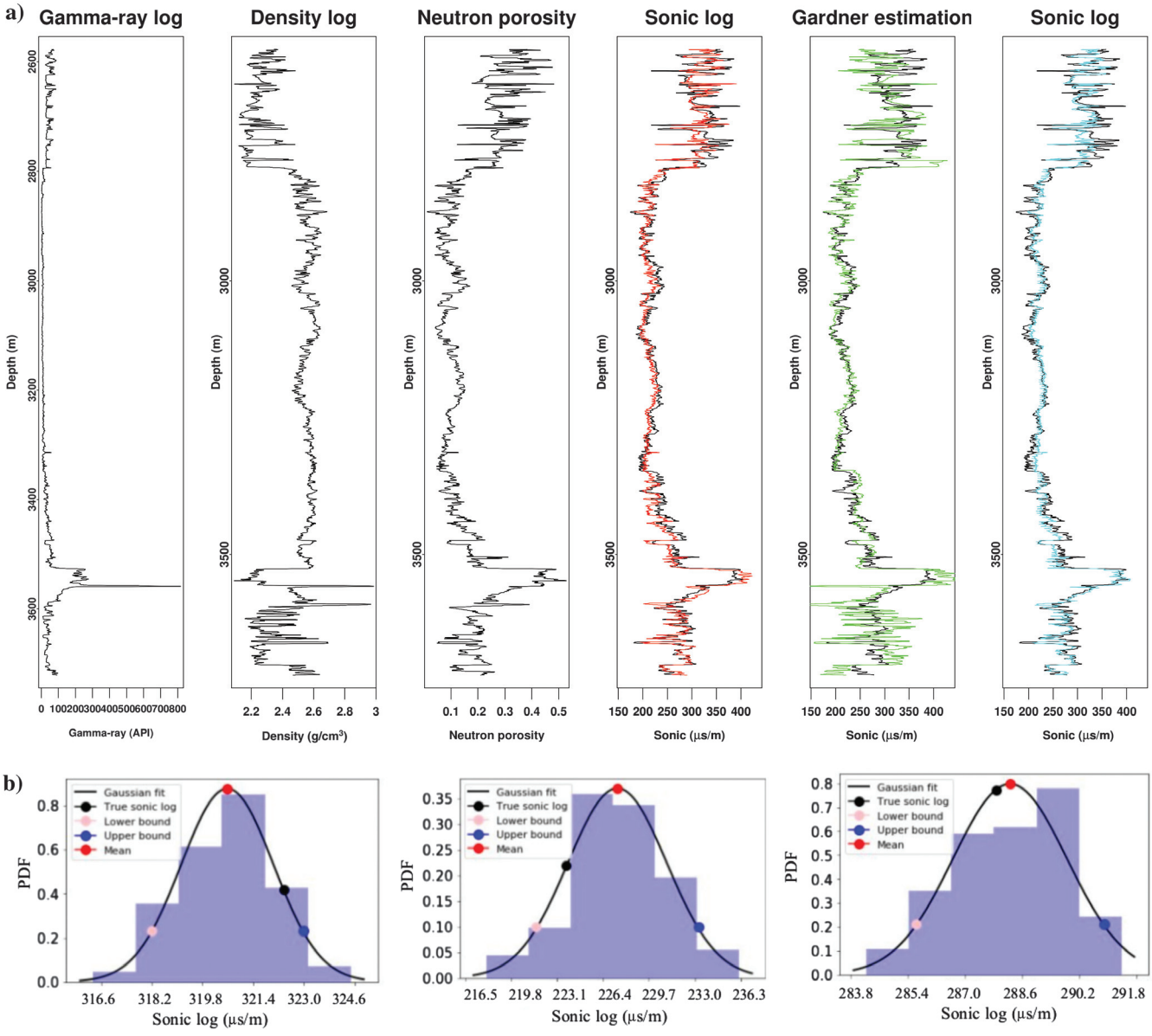


Figure 12. (a) Testing well 2: gamma-ray log, density log, neutron porosity log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in testing well 2: 2580 m (top row-left), 3481 m (top row-right), and 3697 m (bottom row-center). The vertical axes are probability density functions.

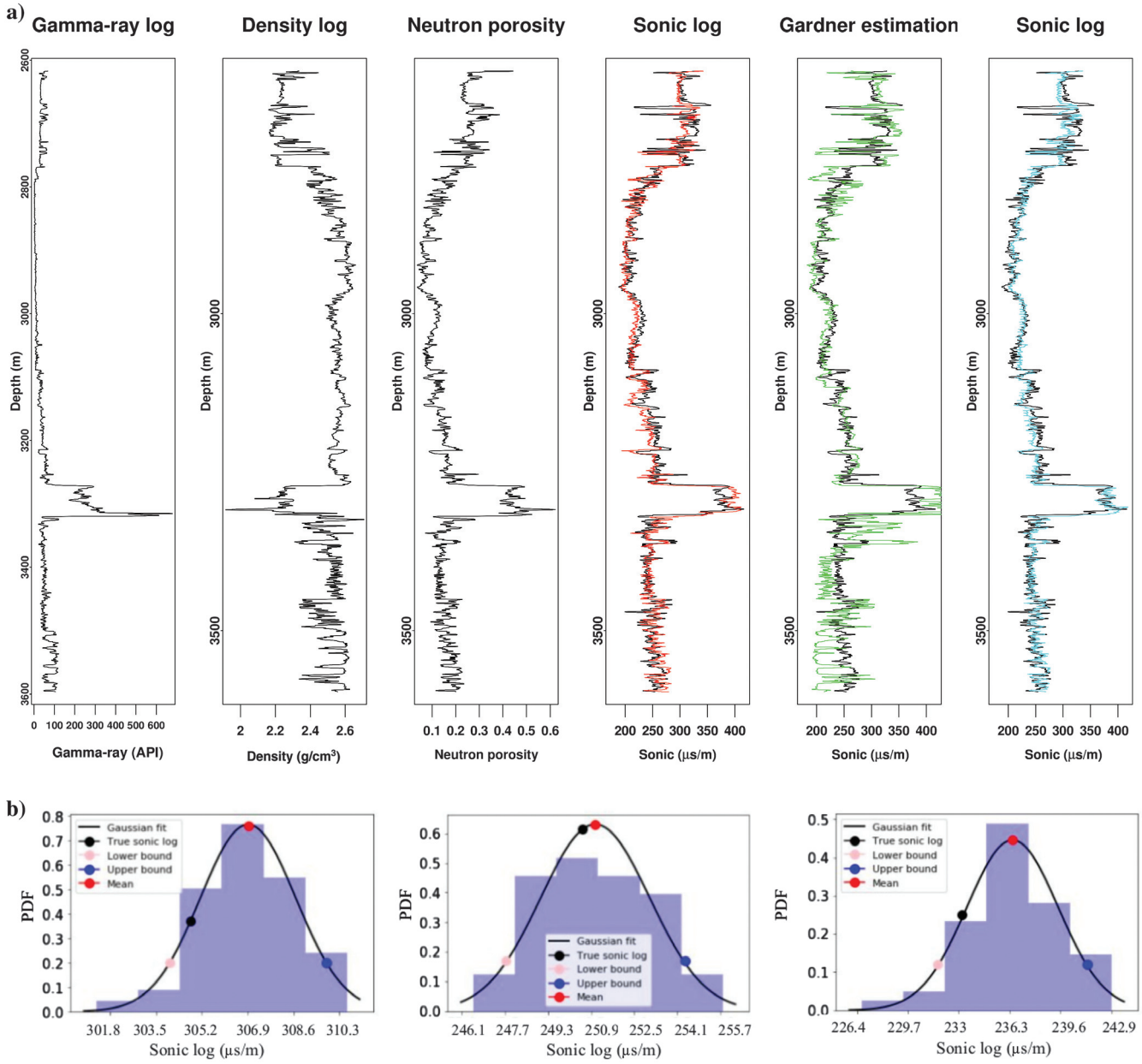
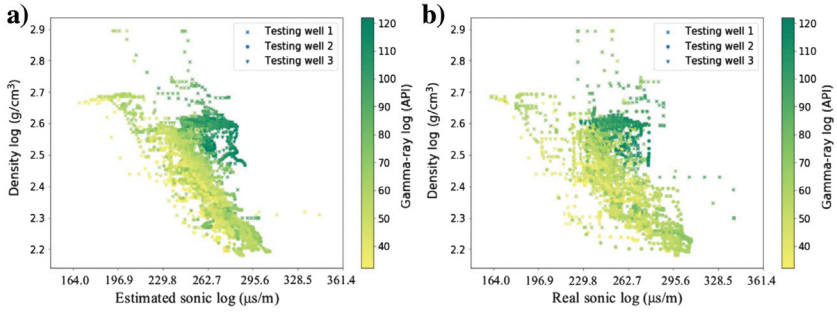


Figure 13. (a) Testing well 3: gamma-ray log, density log, neutron porosity log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in testing well 3: 2627.8 m (top row-left), 3266.8 m (top row-right), and 3500 m (bottom row-center). The vertical axes are the probability density functions.

Figure 14. Crossplots of three wells in the area of interest below the shale layer of the Volve field. (a) Between the density logs and the predicted sonic logs. (b) Between the density logs and the actual sonic logs.



objective function in a neural network such as the MSE used in our experiment is the same as minimizing the Kullback-Leibler divergence mentioned above. Therefore, we can approximate the integral in equation 9 with Monte Carlo integration

$$P(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) \approx P(\mathbf{y}^*|\mathbf{x}^*,\omega)Q(\omega)d\omega \approx \frac{1}{T}\sum_{t=1}^{T} P(\mathbf{y}^*|\mathbf{x}^*,\hat{\omega}_t), \tag{13}$$

where $\hat{\omega}_t \sim Q(\omega)$.

In short, we can perform the dropout at test time to generate a distribution of the output well log and take the mean for the prediction and the variance for model uncertainty. This is equivalent to taking the 95% quantile as an upper bound and 5% quantile as a lower bound. It is better to model the weights of deeper layers in a probabilistic way, so we only use the dropout layers at the FCNN layers at test time. The dropout rate could be optimized, but we fix it as 30% at test time. Also, instead of having a fixed dropout rate, we can generate the distribution of the prediction output by using a range of dropout rates from 10% to 50%.

TRAINING

Training data

The training data consist of 177 wells from mature areas of the UK continental shelf (UKCS) with a full suite of gamma-ray, den-

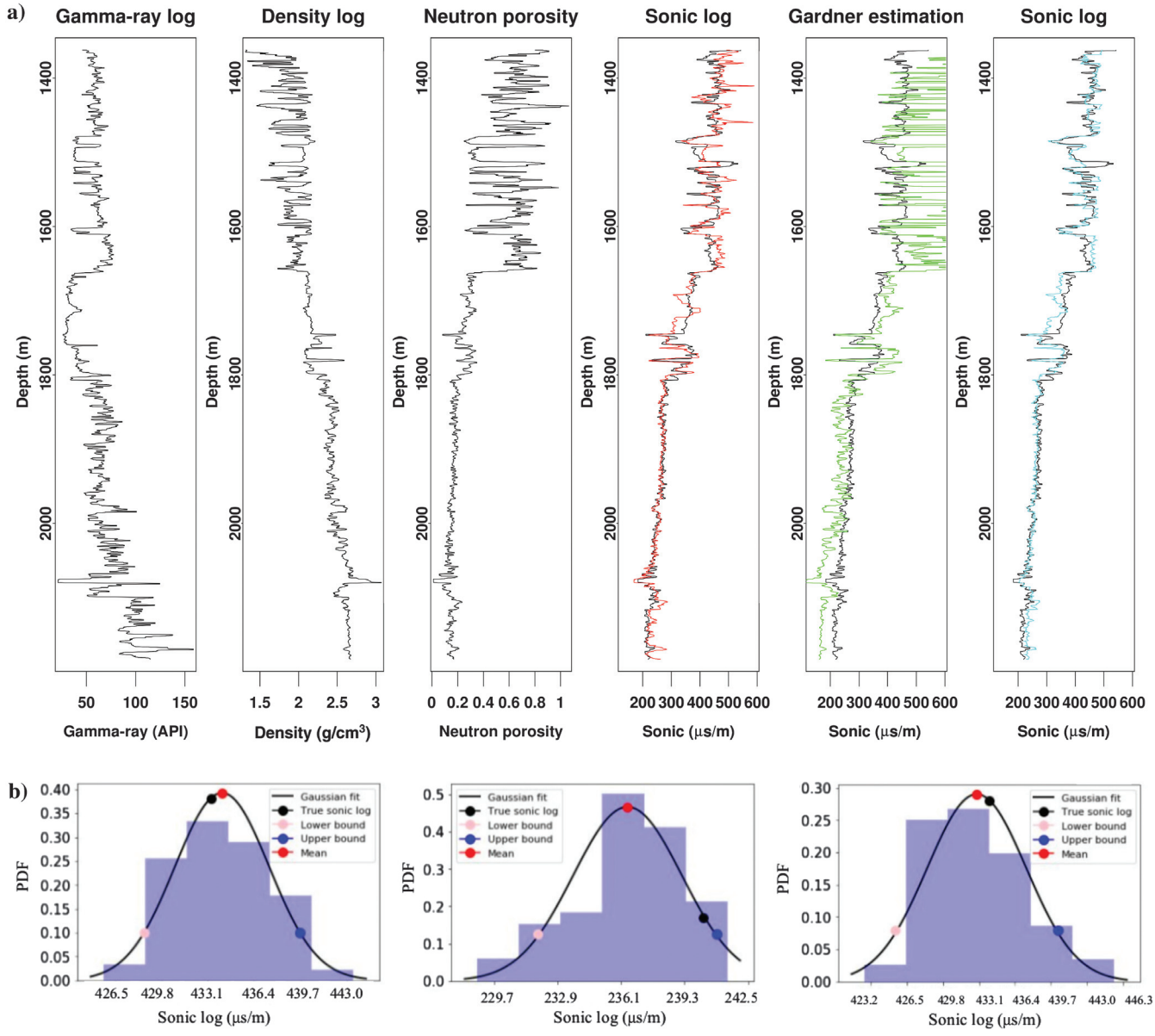


Figure 15. (a) Testing well 4: gamma-ray log, density log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in testing well 4: 1591.1 m (top row-left), 2048.3 m (top row-right), and 1530.1 m (bottom row-center). The vertical axes are the probability density functions.

sity, neutron porosity, and sonic logs. The sonic log is related to clay volume, density, and porosity, so we want to use these three log types to predict the missing sonic log. We conduct an experiment while having only gamma-ray and density as input logs, but the behavior of these logs is opposite (density decreases and P-wave velocity increases when the clay volume increases), making the network more difficult to identify, especially the local log shape such as in the validation well. Therefore, we believe that more input log types would increase the performance of the network. The adding of neutron porosity log guides the prediction of fluctuation trend better than having only two types of log with opposite trend.

We first find the shortest length of all of the training wells and take the samples from the beginning of each well until the largest multiple of "shortest length" to create sequences such that samples from different wells do not mix together in each sequence in a batch. The shortest training well length has 671 samples, so we choose a sequence length as 61 samples. We preprocess the data by removing the median and scaling according fo the quantile range, which is robust to the outliers and removes the effects of erroneous spikes in the training data. The samples from the gamma-ray logs are concatenated with those from the density logs and neutron porosity log. We create 2D maps centered at each log sequence in a batch, with 21 sequences. A 21-sample convolutional filter captures the local shape of a log.

We randomly split the data into 90% for training and 10% for validation. Using this validation data set, we also can tune the hyperparameters of the network such as number of layers, number of output feature maps. The metric we use on validation data set, to decide whether our network is performing well and when to have early stopping to control overfitting (Caruana et al., 2000), is MSE and $1 - R^2$ values

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \quad 1 - R^2 = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \quad (14)$$

where N is the number of samples, y_i is the predicted sonic log from the model, $\hat{y}_i$ is the true sonic log, and $\bar{y}$ is the mean of true sonic log. The term $1 - R^2$ is a scaled version of the MSE, so the metric is fair despite the data range difference. The training will stop if the MSE and $1 - R^2$ of the validation data set do not decrease in 20 consecutive epochs.

The batch size is 100 sequences. We train the model with a Titan Xp GPU, and the training stops at epoch 113 after 4 h. The training MSE decreases with increasing epochs, and the best training MSE is 0.059 (Figure 4). The validation MSE and validation $1 - R^2$ increase for 20 epochs after epoch 93 (Figures 5 and 6). The best validation MSE is 0.04, and the best validation R^2 is 0.75. During validation, we turn off the variational recurrent dropout so the validation cost is smaller than the training cost.

The training gamma-ray, density, neutron porosity, and sonic logs are despiked using median filtering of the rolling window, before feeding into the deep-learning model (Figure 7). The preprocessing steps make the training process fast. When comparing the results, we also compare with the despiked version of true sonic logs because the spikes in the logs sometimes are caused by the instruments and not by subsurface condition. Gardner's estimations of sonic logs also are calculated from the same preprocessed input logs.

Training results

After training the model with 113 epochs, we compare the predicted sonic log of one validation well with the actual sonic log and with the log estimated from Gardner's equation (Figure 8a). The dropout layer is used to simulate a distribution of 100 predicted sonic logs, whose mean is the final result. The 95% quantile is an upper bound, and the 5% quantile is a lower bound. The uncertainty range is very narrow, which cannot be observed easily in a plot of all depths, so we illustrate the uncertainty estimation process of three log samples at three different depths in Figure 8b. Figure 9a shows a close match between our result and the measured sonic log with a correlation coefficient of 88.3%, whereas Gardner's estimation achieves 87.6% correlation (Figure 9b).

We also train a bidirectional LSTM model and a unidirectional ConvLSTM model to predict the sonic log and compare with the bidirectional ConvLSTM output. The correlation and R^2 between bidirectional LSTM output and true sonic log are 86.1% and 0.72, respectively. The correlation and R^2 between unidirectional ConvLSTM output and true sonic log are 86.9% and 0.74, respectively. The bidirectional ConvLSTM performs a little better than bidirectional LSTM and unidirectional ConvLSTM in capturing the funnel shape in the log (Figure 10).

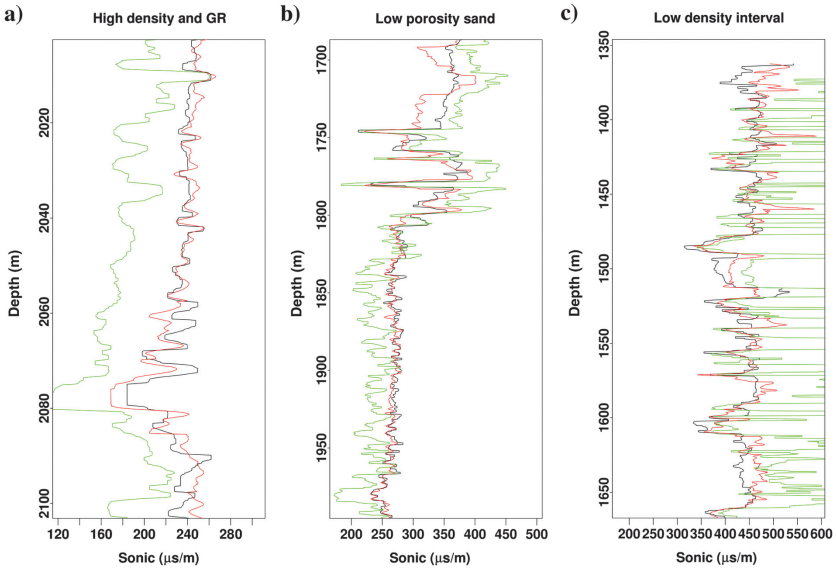


Figure 16. Different intervals of interest in testing well 4: true sonic log (black), predicted sonic log (red), Gardner's estimation (green).

TESTING

We apply the trained model to predict the sonic logs for three blind wells from the Norwegian continental shelf (NCS) and compare the predicted sonic logs with the actual sonic logs and with the logs estimated from Gardner's equation (Figures 11a, 12a, and 13a). The dropout layers are used to simulate a distribution of 100 predicted sonic logs to quantify the uncertainty (Figures 11b, 12b, and 13b). There is a

close match between our result and the measured sonic log of the first well with a correlation coefficient of 96.1%, whereas Gardner's estimation achieves 72.83% correlation. The term R^2 value between the predicted sonic log and the true sonic log is 0.915, whereas Gardner achieves 0.72. The predicted sonic log in the second well matches the actual sonic log with a correlation coefficient of 96%, whereas the correlation coefficient for Gardner's estimation is 83.9%. The term R^2 value is 0.91 for our prediction, whereas the R^2 value for Gardner's estimation is 0.61. Although the sand formation, after the shale layer, from 4226 m in the first well and from 3575 m in the second well seems to be missing in the third well, probably due to faulting, the trained model still produces a good match between the predicted and true sonic logs with a correlation coefficient of 92.8% and without knowing about the lithology, whereas Gardner's estimation achieves 83.4% correlation. The term R^2 values are 0.84 and 0.32 for our estimation and Gardner's estimation, respectively. Figure 14 shows the crossplot between the

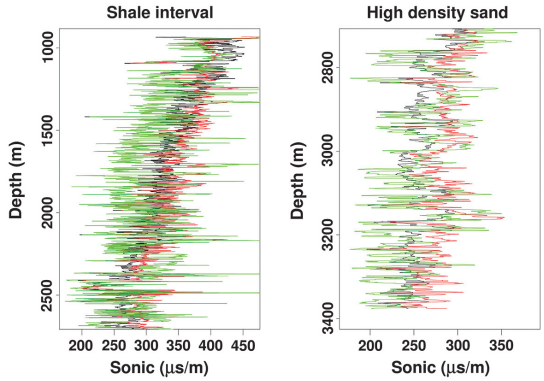


Figure 18. Different intervals of interest in testing well 5: true sonic log (black), predicted sonic log (red), and Gardner's estimation (green).

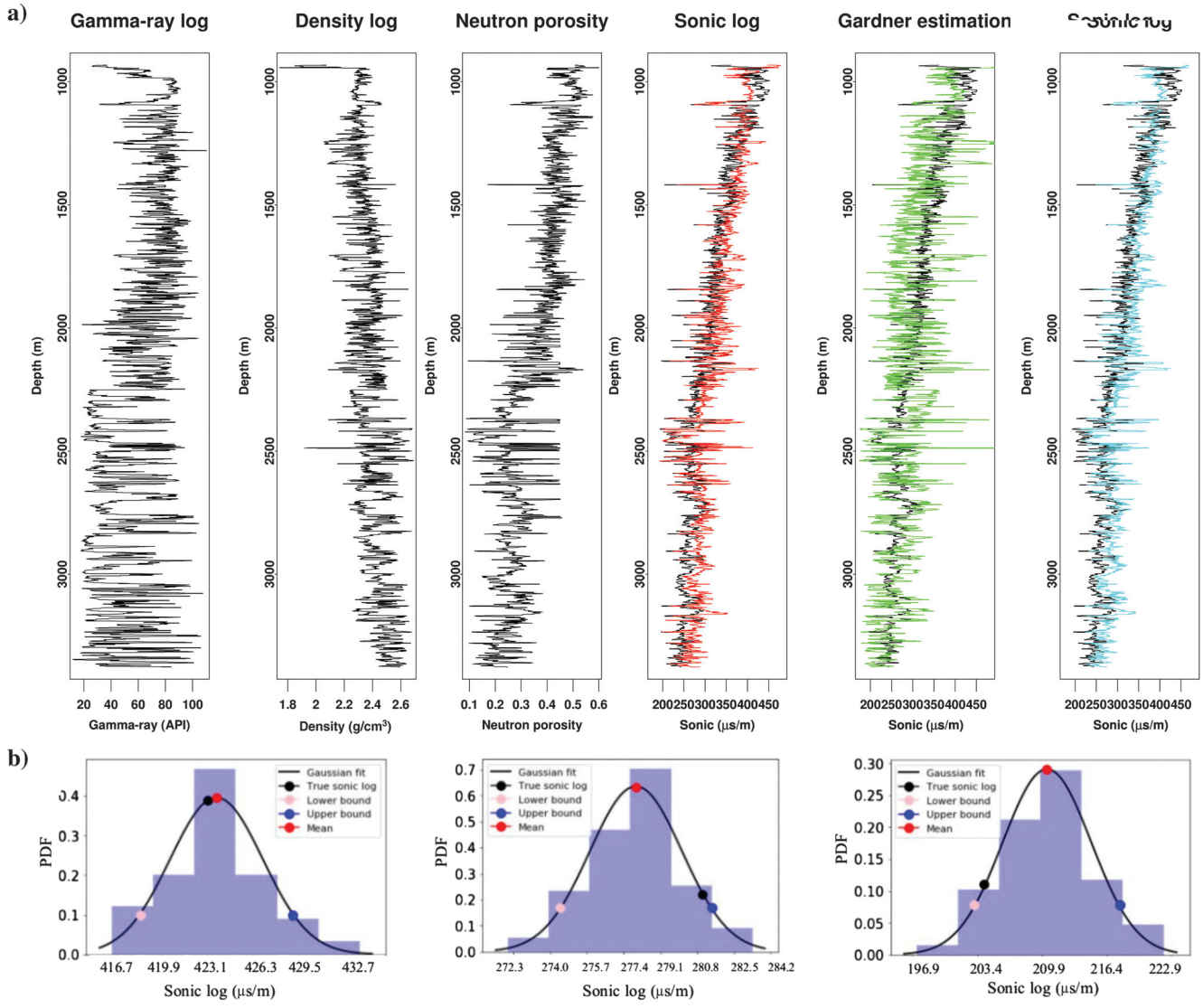


Figure 17. (a) Testing well 5: gamma-ray log, density log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in testing well 5: 1145.7 m (top row-left), 2151.6 m (top row-right), and 2456.4 m (bottom row-center). The vertical axes are the probability density functions.

density logs versus the predicted sonic logs and the crossplot between the density logs versus the actual sonic log at the oil-saturated sand layers below the shale layer. The correlation coefficients between the predicted sonic log and the actual sonic log are 99.37%, 99.88%, and 99.7% in the first, second, and third well, respectively.

We also apply the trained model to predict the missing sonic log of one blind well from UKCS with an addition of uncertainty estimation using dropout layers (Figure 15). The Monte Carlo simulation at three different depths is shown in Figure 15b. The predicted sonic log matches the measured log with a correlation coefficient of 96.7%, whereas Gardner's estimation achieves very low correlation. The problem with Gardner's estimation is that it is difficult to choose proper parameters for the layer with very low density greater than 1682.5 m (Figure 16).

The ConvLSTM outputs and LSTM outputs do not differ significantly, probably because there are no significant local shapes in the logs.

DISCUSSION

We apply our proposed trained model on the last testing well from offshore Canada, where the geologic conditions are very different from the training wells, to test for the generability of the model (Figure 17a). It produces a reasonable match between the predicted and actual sonic logs with a correlation coefficient of 92%, whereas Gardner's estimation produces a 64.8% correlation. At two intervals of interest, the predicted sonic log matches the actual sonic log with accuracy similar

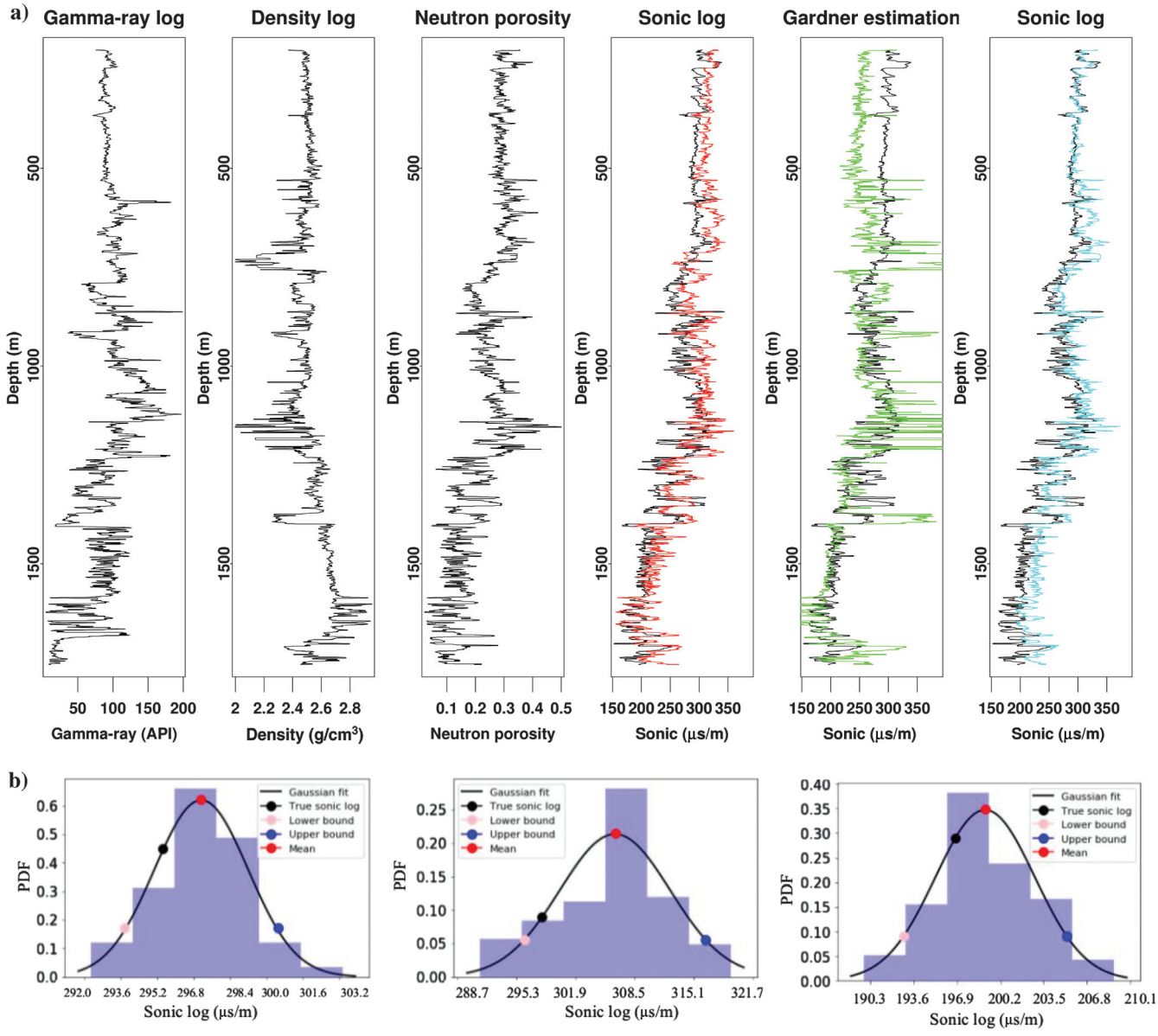


Figure 19. (a) Testing well 6: gamma-ray log, density log, predicted sonic log (red), true sonic log (black), Gardner's estimation (green), and bidirectional LSTM output (light blue). (b) Monte Carlo simulation at three different depths in testing well 6: 700 m (top row-left), 1200 m (top row-right), and 1600 m (bottom row-center). The vertical axes are the probability density functions.

to the conventional method (Figure 18), although our method does not need to be applied within different geologic intervals.

We also use the trained model to predict the sonic log at a well in the onshore Teapot Dome data set. The trained model can produce a good estimation of the sonic log for the last testing well, with a correlation of 95.2%, and without being trained on the wells in the testing area (Figure 19a). It can be explained that the network has the ability to learn a nonlinear relationship between gamma-ray, density, neutron porosity, and sonic logs when being calibrated to different rock types of the training data. However, the generalizability of our method can be improved by fine-tuning the trained model with new wells when it is applied to different areas. We can freeze the trained parameters except in the last FCNN layer and retrain the model with available wells in areas of interest.

CONCLUSION

We propose a method for estimating missing sonic logs from gamma-ray, neutron porosity, and density logs using a deep-learning model consisting of BRNN with ConvLSTM blocks cascaded with FCNNs. Our method, which takes into account the geologic trend and the local shape of logs, predicts sonic logs matching actual sonic logs with high accuracy. The proposed approach shows an improvement over a conventional method for estimating missing sonic logs without being applied internally and with an addition of uncertainty estimation by using dropout layers. The approach can be extended to predict other types of logs from different kinds of input logs. Future research will include other types of uncertainty such as data uncertainty and will make the local window size based on the geologic settings as an input of the network. The rock physics model can be combined with the deep-learning approach to create better predictions for missing logs.

ACKNOWLEDGMENTS

We thank the financial support by the National Science Foundation of China under grant no. 41974121 as well as the sponsors of the Texas Consortium for Computational Seismology (TCCS). We thank the NVIDIA GPU Grant Program for providing the Titan Xp for computations.

DATA AND MATERIALS AVAILABILITY

Data associated with this research are available and can be obtained by contacting the corresponding author.

REFERENCES

Ba, J., J. R. Kiros, and G. E. Hinton, 2016, Layer normalization: ArXiv, abs/1607.06450.
Bader, S., X. Wu, and S. Fomel, 2019, Missing log data interpolation and semiautomatic seismic well ties using data matching techniques: Interpretation, **7**, T347–T361, doi: 10.1190/INT-2018-0044.1.
Cant, D., 1994, Facies models: Response to sea level change: Geological Journal, **29**, 27–45..
Caruana, R., S. Lawrence, and L. Giles, 2000, Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping: Proceedings of the 13th International Conference on Neural Information Processing Systems, MIT Press, 381–387.
Castagna, J. P., M. L. Batzle, and R. L. Eastwood, 1985, Relationships between compressional-wave and shear-wave velocities in clastic silicate rocks: Geophysics, **50**, 571–581, doi: 10.1190/1.1441933.
Faust, L. Y., 1953, A velocity function including lithologic variation: Geophysics, **18**, 271–288, doi: 10.1190/1.1437869.
Gal, Y., and Z. Ghahramani, 2015a, Bayesian convolutional neural networks with Bernoulli approximate variational inference: ArXiv, abs/1506.02158.
Gal, Y., and Z. Ghahramani, 2015b, A theoretically grounded application of dropout in recurrent neural networks, *in* Advances in neural information processing systems: 1019–1027.
Gal, Y., and Z. Ghahramani, 2016, Dropout as a bayesian approximation: Representing model uncertainty in deep learning: Proceedings of the 33rd International Conference on International Conference on Machine Learning, JMLR.org.
Gardner, G., L. Gardner, and A. Gregory, 1974, Formation velocity and density — The diagnostic basics for stratigraphic traps: Geophysics, **39**, 770–780, doi: 10.1190/1.1440465.
Ghahramani, Z., 2015, Probabilistic machine learning and artificial intelligence: Nature, **521**, 452–459, doi: 10.1038/nature14541.
Greenberg, M. L., and J. P. Castagna, 1992, Shear-wave velocity estimation in porous rocks: Theoretical formulation, preliminary verification and applications: Geophysical Prospecting, **40**, 195–209, doi: 10.1111/j.1365-2478.1992.tb00371.x.
Herrera, R. H., S. Fomel, and M. van der Baan, 2014, Automatic approaches for seismic to well tying: Interpretation, **2**, SD9–SD17, doi: 10.1190/INT-2013-0130.1.
Hochreiter, S., and J. Schmidhuber, 1997, Long short-term memory: Neural Computation, **9**, 1735–1780, doi: 10.1162/neco.1997.9.8.1735.
Kingma, D. P., and J. Ba, 2015, Adam: A method for stochastic optimization: Presented at the 3rd International Conference on Learning Representations, ICLR.
Kullback, S., 1959, Information theory and statistics: Wiley.
Muñoz, A., and D. Hale, 2015, Automatic simultaneous multiple well ties: Geophysics, **80**, no. 5, IM45–IM51, doi: 10.1190/geo2014-0449.1.
Pham, N., S. Fomel, and D. Dunlap, 2019, Automatic channel detection using deep learning: Interpretation, **7**, SE43–SE50, doi: 10.1190/INT-2018-0202.1.
Rolon, L., S. D. Mohaghegh, S. Ameri, R. Gaskari, and B. McDaniel, 2009, Using artificial neural networks to generate synthetic well logs: Journal of Natural Gas Science and Engineering, **1**, 118–133, doi: 10.1016/j.jngse.2009.08.003.
Saggaf, M., and L. Nebrija, 2003, Estimation of missing logs by regularized neural networks: The American Association of Petroleum Geologists Bulletin, **87**, 1377–1389, doi: 10.1306/03110301030.
Salehi, M. M., M. Rahmati, M. Karimnezhad, and P. Omidvar, 2017, Estimation of the non records logs from existing logs using artificial neural networks: Egyptian Journal of Petroleum, **26**, 957–968, doi: 10.1016/j.ejpe.2016.11.002.
Schuster, M., and K. Paliwal, 1997, Bidirectional recurrent neural networks: IEEE Transactions on Signal Processing, **45**, 2673–2681, doi: 10.1109/78.650093.
Shi, X., Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, 2015, Convolutional LSTM network: A machine learning approach for precipitation nowcasting: Advances in Neural Information Processing Systems, Curran Associates, Inc., 802–810.
Smith, J. H., 2007, A method for calculating pseudo sonics from e-logs in a clastic geologic setting: Gulf Coast Association of Geological Societies Transactions, **57**, 675–678.
Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, Dropout: A simple way to prevent neural networks from overfitting: Journal of Machine Learning Research, **15**, 1929–1958.
Vorontsov, E., C. Trabelsi, S. Kadoury, and C. Pal, 2017, On orthogonality and learning recurrent networks with long term dependencies: Proceedings of the 34th International Conference on Machine Learning, PMLR, 3570–3578.
White, R., and R. Simm, 2003, Tutorial: Good practice in well ties: First Break, **21**, 75–83.
Wu, X., and G. Caumon, 2017, Simultaneous multiple well-seismic ties using flattened synthetic and real seismograms: Geophysics, **82**, no. 1, IM13–IM20, doi: 10.1190/geo2016-0295.1.
Xu, B., N. Wang, T. Chen, and M. Li, 2018, Empirical evaluation of rectified activations in convolutional network: Computing Research Repository, abs/1505.00853.
Zhang, D., Y. Chen, and J. Meng, 2018, Synthetic well logs generation via recurrent neural networks: Petroleum Exploration and Development, **45**, 629–639, doi: 10.1016/S1876-3804(18)30068-5.
Zhao, J., and L. Itti, 2018, shapeDTW: Shape dynamic time warping: Pattern Recognition, **74**, 171–184, doi: 10.1016/j.patcog.2017.09.020.