Artificial Intelligence Principles and Techniques

Improved Time Series Decline Curve Analysis For Oil Production Using Recurrent Neural Network

Cédric Fracès Gasmi*[1], Ouassim Khebzegga†[1], and Soheil Esmaeilzadeh‡[1]

[1]Stanford University, CA 94305, USA

Abstract

In this work, we provide a more accurate alternative to Decline Curve Analysis (DCA), one of the most prevalent forecasting techniques used in the oil & gas industry. DCA largely relies upon non-linear regression where a hyperbolic function is fitted through historical data to predict the future production. We propose a data-driven approach that utilizes both the most recent portion of a time series and a set of heuristic to improve upon the current benchmark. Given the time series of production history and a set of contextual meta data (e.g. well location, field's geological information, operations, etc.) we forecast the hydrocarbon production of a given well. We show how regression and clustering techniques can be combined to improve prediction accuracy. We use different sequential neural networks with domain informed feature engineering and enhance the accuracy of the forecast (in some cases by a large margin) compared to a DCA approach.

Introduction and Background

The most prevalent forecast method in the oil & gas industry is Decline Curve Analysis (DCA) (ref. Arps [1945]). This technique consists of fitting a curve to the history data in order to predict the future performance of a well. Based on DCA the pseudo-steady state production response of a well follows a differential equation of the form

$$D = -\frac{1}{q}\frac{dq}{dt}. \qquad (1)$$

The typical time dependent solution of this equation has a hyperbolic form of $q(t) = \frac{q_0}{(1+Dt)^{1/b}}$ where b is called the *b-factor*, D is *the initial nominal decline rate*, and q_0 is the *initial production rate*. These parameters are mainly chosen by an engineering intuition and are adjusted to account for heuristic considerations (e.g. the basin of production, the type of driving forces, and the completion techniques used). There are many commercial software packages (e.g., Aries, PHDWin, Palissade, OFM, and etc.) that help to find the best fit for any imposed constraints using DCA. Using these software packages the process of calibrating a *good* curve to the history data is typically done by the reservoir engineer who applies a combination of heuristics and analytical methods to produce a plausible forecast for the production of a well. This process is generally probabilistic and takes into account a range of uncertainties.

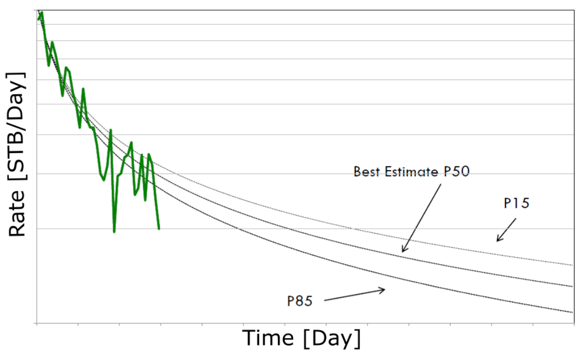


Figure 1: Typical decline curve forecast - Green: Production curve history - Black: production forecast (*Source: ERE 269, Engineering Valuation and Appraisal of Assets)*

As we see in Fig. 1, the prediction made using DCA (solid black line) is a smooth curve that cannot capture the rapid variations that usually happen in the production process (solid green line) and is therefore subject to a certain amount of error, nevertheless, engineers still favor this method due

*cfraces@stanford.edu
†ouassimk@stanford.edu
‡soes@stanford.edu

to its simplicity. In this work, we propose a new method of forecasting that increases the accuracy of the prediction compared to DCA. This method allows for the integration of various sources of information, starting with the history production of the well and adapting the accuracy of the prediction to the amount of available information. We use deep learning architectures along with clustering to adaptively integrate the various sources of information and features relevant to the forecast.

There is not a lot of related work on the use of time series prediction for the Oil and Gas industry. Application of data driven approaches to reservoir modeling is relatively new and coincides with the rise of unconventional production in the USA, combined with our limited understanding of the physics governing the behavior of shale formations. Due to the availability of data for this type of reservoirs, researchers and engineers started using data-driven approaches to get insights into the production patterns. Mohaghegh et al. [2011] developed a top-down approach for modeling and history matching of shale production based on statistical and pattern recognition models and applied their approach to three shale reservoirs. Sun et al. [2018] applied Long-Short Term Memory algorithm (LSTM) to predict a well's oil, water, and gas production. Their work led to an improvement of the production forecast in comparison with standard DCA models. However, the tested portion of their time series exhibits weak variations in comparison with the training part and it remains unclear whether the algorithm can generalize well to complex time series. Ristanto [2018] applied ANN algorithms to predict the production of a well based on pressure measurements and compared vanilla Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU), and LSTM algorithms and found that the GRU gives the best forecast results.

This document presents the procedure of applying three main models to improve the oil production forecast based on historical and contextual data. The analysis is divided into three categories as

- **Clustering & Unsupervised Learning**
  This covers the ensemble of methods that help with presenting, grouping, and filtering the data-set to better frame the problem. The main motivation behind clustering is that engineers tend to *tweak* certain parameters depending on a set of heuristic considerations (e.g. geology, basin, type of completion, and etc.) and clustering will allow for the integration of contextual static data into the predictions.
- **History Based Time Series Forecast**
  In this part, we use the history of oil production (the quantity of interest) in a single time series and forecast the future production. This is usually the most relevant source of information and is given a high importance in our work. The goal is to detect and capture a set of harmonics in the time series and use them for forecast.
- **Secondary Variables Based Sequence-to-Sequence Regression**
  In this class of methods, we establish a correlation between the quantity of interest (e.g. oil production) and other time series (e.g. gas production, water production, number of flowing days, and etc.) that may influence it. This will allow for integration of relevant contextual temporal data into the forecast process.

Data-set Statistics

Production records of over 35,000 wells, representing a subset of California public data, are downloaded from the Department of Conservation, Division of Oil, Gas, and Geothermal Resources (DOGGR). The data is cleaned and formatted and contains well names, operators, fields, completion dates, geographical locations along with time series of monthly fluids production and injection. As a first pass, the focus of this work is on a single field (North Belridge) with around 4000 wells where we use two sources of data i.e. the contextual (static) data and the historical production (time series).

- **Contextual Data:** Contains macro information for each well such as geographical location, name, type, operator, lease, geological production zone as well as the start and end dates of production. Contextual data is mainly used for the clustering exercise.
- **Historical Data:** Contains the production time series for each well such as the oil, water, and gas monthly productions as well as the number of active days per month for each well. Historical data is used for clustering, history based time series forecast, as well as the secondary variable based sequence-to-sequence regression.

Clustering

In this section, our goal is to cluster the wells using both static and time series data. The clustering process follows two steps. First we separate the wells into macro-clusters using static data. This requires the design of relevant features based on an engineer's expertise. This is achieved by taking a subset from a global list of 15 features including geolocation coordinates, cumulative oil and water productions, mean, peak, and standard deviation of the oil production, first month production, the cumulative of the first 12 months of production, and the production duration. The second step consists of sub-dividing each macro-cluster into sub-clusters using the production time series. We use dynamic time warping (DTW) to measure the similarity between the production time series of each two wells and use it to group similar wells into the same sub-cluster.

For each step, we benchmark different clustering algorithms. We test KMeans, Spectral Clustering, and Agglomerative Clustering, combined with data normalization and dimensionality reduction using Principal Component Analysis (PCA). For the second step we test Global Alignment Kernel KMeans and the Time Series KMeans algorithms. Table 1 summarizes the different combinations of these methods.

Table 1: Clustering Models

No.	Step 1 Algo.	Step 2 Algo.
1	KMeans	Time Series KMeans
2	KMeans	Global Alignment Kernel KMeans
3	Spectral Clustering	Time Series KMeans
4	Spectral Clustering	Global Alignment Kernel KMeans
5	Agglomerative Clustering	Time Series KMeans
6	Agglomerative Clustering	Global Alignment Kernel KMeans

Clustering Hyper-Parameters Tuning

Table 2 presents the set of hyper-parameters used for each one of the models in Table 1. The heuristic we define to compare the different models is determined by plotting the time series belonging to each sub-cluster and searching for a coherent structure characterizing the time series belonging to the same group, some of these coherency measures are

- Separation between High/Low amplitude of production time series
- Separation between Short/Long production time series
- Separation between time series that start with a high amplitude and then collapse sharply in contrast with a production that declines gradually, this reflects inherent differences in production conditions.

Table 2: Hyper-Parameters Tuning

Parameter	Values
# Clusters Step 1	[5, 7, 10]
# Clusters Step 2	[2, 3, 4]
Set of static features	tested different subsets
Weight of static features	[1, 5, 10]

Clustering Results

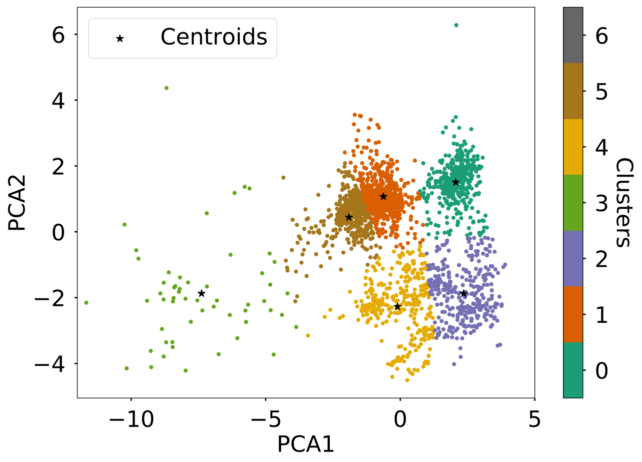
Both model 1 and model 5 in Table 1 lead to the best results. We present the results of model 1 composed of KMeans + Time Series KMeans algorithms with 7 macro-clusters, each one decomposed into 2 sub-clusters. For this case we use the following set of features: (geolocation coordinates, cumulative oil and water productions, mean, peak, and standard deviation of the oil production), and unity weights. The results of the first step are presented in Fig. 2, we see that wells belonging to a same cluster naturally regroup geographically, this reflects the underlying geological formation characteristics of each region. Fig. 3a and 3b show the decomposition of two macro-clusters, 2 and 3, into sub-clusters, the comparison of the top and bottom plots of Fig. 3a shows that the time series length is the characteristic that determines the well's sub-cluster, with long and short time series grouped into different clusters. Fig. 3b shows another configuration where the amplitude of the production is the determining characteristic. These clusters can then be used to apply different forecast models in the time series forecasts presented in the following.

Time Series Forecasting

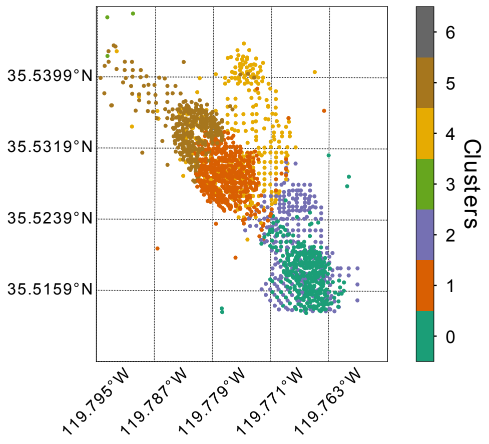
In this section, we illustrate forecasting of a time series *e.g. oil production* using different variants of Recurrent Neural Network (RNN) architectures.

History Based Time Series Forecast

We forecast time series of oil production using the history of production for a single well. We use different Recurrent Neural Network architectures, do a hyper-parameter study, and finally compare the best RNN model's outcome with the Decline



(a)



(b)

Figure 2: (a). wells clusters projected in PCA space, (b). same clusters in the physical geographical space

Curve Analysis results.

Data Pre-processing

The time series relevant to our work are usually non-stationary and have variable trends meaning that the statistical characteristics of the data e.g., frequency, variance, and mean vary in time. Stationary time series are easier to model and their trends are usually easier to capture. Prior to training, we convert the time series into stationary data using *differencing* (i.e. computing the differences between consecutive observations). We then normalize the data in the interval $[-1, +1]$ to avoid vanishing gradients when using RNN. Next, we split a time series into input set X and output y using a *lag time* approach (i.e. output will be lagged from the input by a few time steps). The sizes of lags are treated as a tuning hyper-parameter to get better accuracy. After training, we convert the data back for error calculations.

History Based Forecast Training Model

We use many-to-one (cref. Fig. 4) recurrent neural network. A many-to-one model produces one output value after receiving multiple input values (an input sequence). The internal state is accumulated with each input value before a final output value is produced. We use Root Mean Squared Error (RMSE) to measure the accuracy of the forecast during the training process. RMSE can be calculated as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i^{pred} - y_i^{real}}{y_i^{real}}\right]^2} \times 100, \qquad (2)$$

where n is the total number of time steps in the time series, and y_i^{pred}, y_i^{real} are the model predicted and actual values of the time series at each time step.

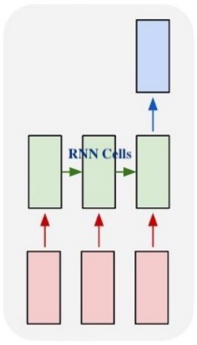
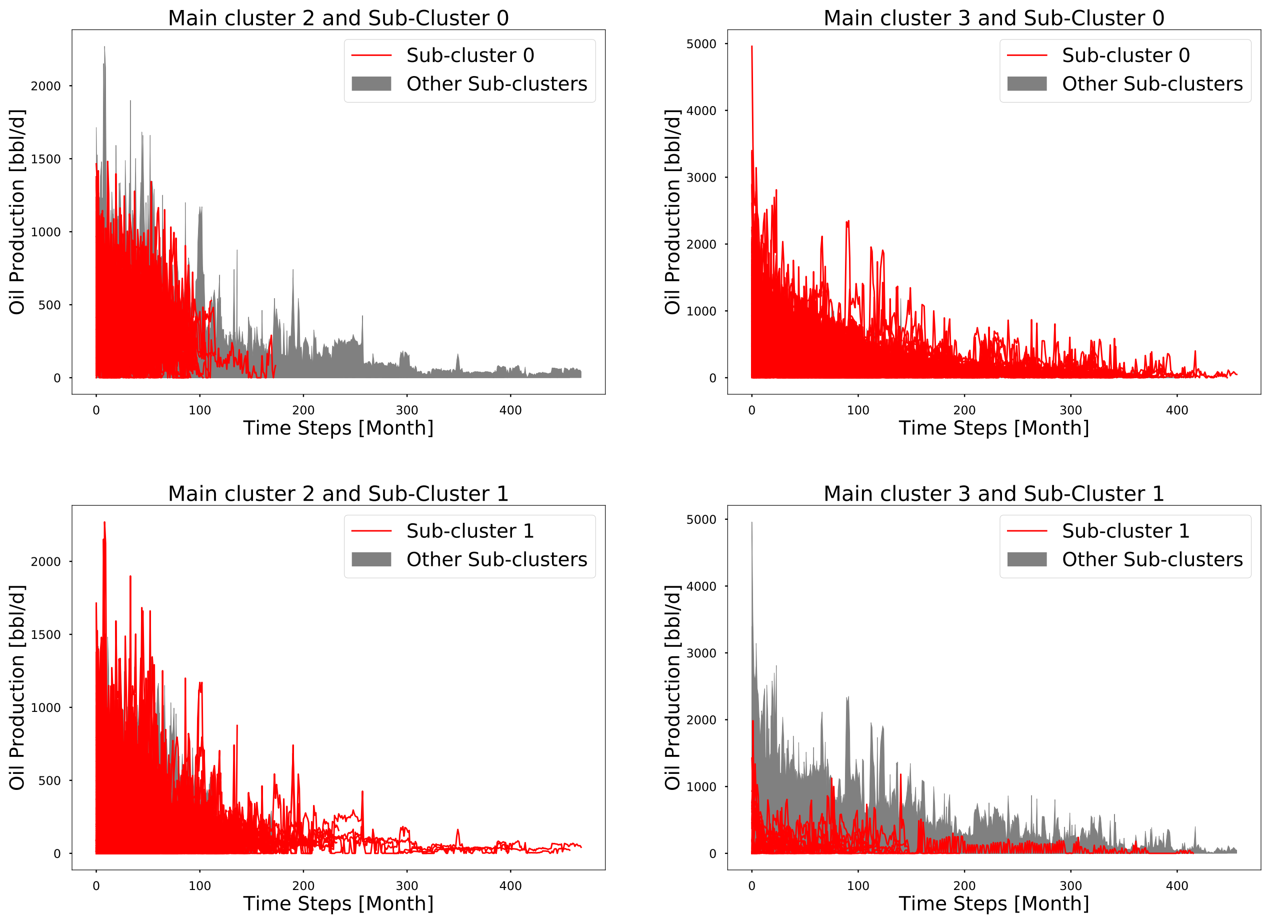


Figure 4: A many-to-one sequence model, where an input sequence is considered for an output through connections made by RNN states

We consider three common recurrent models as **(1).** Deep Vanilla Recurrent Neural Network (DVRNN), **(2).** Deep Gated Recurrent Units (DGRU) network, and **(3).** Deep Long Short Term Memory (DLSTM) network. RNNs have feedback loops in the recurrent layer which lets them maintain information in memory over time. But, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies due to vanishing gradient problems. On the other hand, LSTM uses special units in addition to the standard units that includes memory cells for maintaining information for long periods of time. A set of gates is used to control when information enters the memory, when



(a) (b)

Figure 3: (a). Macro-cluster 2 decomposition based on the length of time series; (b). Macro-cluster 3 decomposition based on the amplitude of time series

it gets outputted, and when it's forgotten. GRUs are similar to LSTMs, but use a simplified structure and have a set of gates to control the flow of information, but they don't use separate memory cells and have fewer gates. Before doing an actual time series forecast we do an experiment and compare a very simple Vanilla RNN's (with 5 neurons and 1 layer) forecast capability with Decline Curve Analysis (DCA). We synthetically create the time series by adding sine and cosine modes to a base exponentially decaying with time signal (i.e. $a_{t_i} \sin(\omega_{t_i} t) + b_{t_i} \cos(\omega_{t_i} t)$ with $a_{t_i}, b_{t_i}, \omega_{t_i}$ being random amplitude and frequencies at each time step t_i). Then by splitting the time series into 75%, 25% split as training and test set we forecast using both the Vanilla RNN and DCA. Fig. 5 shows the comparison of the forecast results using Vanilla RNN and DCA for a noisy exponentially decaying time series. Using Vanilla RNN improves the RMSE by **34%** which is a considerable improvement compared to a naive DCA approach.

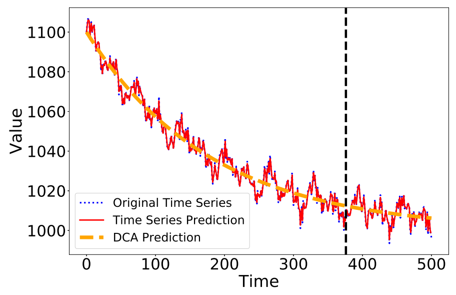


Figure 5: Comparison of Decline Curve Analysis v.s. Recurrent Neural Network for oil production time series forecast

History Based Time Series Forecast Results

Table 3 shows the experiment results of an actual time series forecast using the three recurrent architectures. We first use a one-layer Vanilla Recurrent Neural Network (VRNN) which leads to

Table 3: RMSE values of different experiments; GRU: Gated recurrent unit, LSTM: long short-term memory, D.O.: with Drop-out, Reg.: with regularization

No.	Experiments	RMSE (%)
1	Vanilla RNN (VRNN)	10.32
2	Deep Vanilla RNN (DVRNN)	8.71
3	DVRNN + D.O.	8.51
4	DVRNN + D.O. + Reg.	8.11
5	Deep GRU (DGRU)	7.72
6	DGRU + D.O.	7.61
7	DGRU + D.O. + Reg.	7.49
8	Deep LSTM (DLSTM)	7.11
9	DLSTM + D.O.	6.87
10	DLSTM + D.O. + Reg.	**5.92**

Table 4: Detailed parameter values of experiments in table 3; column *Layer sizes* gives the number of neurons in each layer(s) of a model, *D.O. Coef.* is the drop-out coefficient, and *Reg. Coef.* is the coefficient for L_2 regularization

No.	Layer sizes (# neurons)	D.O. Coef.	Reg. Coef.
1	(15)	0	0
2	(20,15,7)	0	0
3	(20,15,7)	0.4	0
4	(20,15,7)	0.2	$L_2(0.03)$
5	(25,10,5)	0	0
6	(25,10,5)	0.5	0
7	(25,10,5)	0.25	$L_2(0.01)$
8	(20,15,5)	0	0
9	(20,15,5)	0.45	0
10	(20,15,5)	0.15	$L_2(0.02)$

RMSE of 10.32%. We then use a Deep VRNN by stacking multiple layers of VRNN which improves the RMSE to 8.71%. The RMSE decreases by using drop-out and kernel normalization down to 8.11%. In the next step, we use a Deep Gated Recurrent Neural Network (DGRU) that helps to capture long term dependencies in the time series. After hyper-parameter study, a DGRU with drop-out and kernel regularization could achieve 7.49 % RMSE. In the next step, we use Deep Long Short Term Memory (DLSTM) network to better capture the long term temporal dependencies, and we could achieve the highest RMSE of 5.92 % after careful hyper-parameter study. Table 4 shows the details of the optimum network architecture that have come out of the hyper-parameter study for each type of recurrent network. As Table 4 shows for each type of network, in cases where both drop-out and regularization are used the best outcome in terms of RMSE is achieved, however only using large drop-out values under performs compared to using mixed drop-out and regularization. All the models have been trained using *Adam* optimizer and training has terminated once the improvement in the RMSE over 30 epochs is less than 1%-2%.

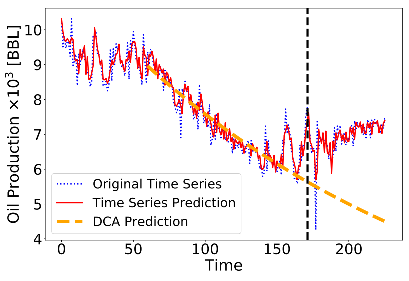
Fig. 6 shows the forecast results for the three types of architectures used i.e. Deep VRNN, Deep GRU, and Deep LSTM with RMSE of 8.11%, 7.49%, and 5.92% respectively. As Fig. 6a shows Deep VRNN can predict the trend however it undershoots the peaks and troughs with a delay. However, Fig. 6b shows that the Deep GRU sometimes undershoots/overshoots the peaks and troughs. On the other hand, Deep LSTM model in Fig. 6c captures both the trends and peaks and troughs reasonably well compared to the other two recurrent architectures. In Fig. 6c Deep LSTM improves RMSE by up to **92%** compared to DCA. The change in trend observed at the end of the time series amplifies the gap between RNN and DCA and is an example of a case where large improvements could be achieved using these deep models.

Secondary Variables Based Sequence-to-Sequence Regression

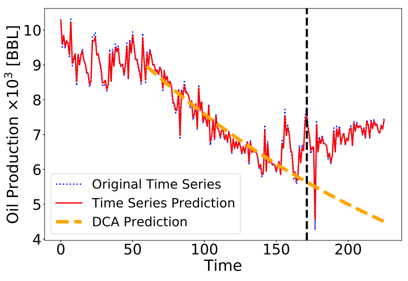
This third model aims at making prediction on oil production taking into account contextual time series data. We use sequence-to-sequence regression model as a way to correlate oil production at a well with a set of other data measured at the same well (gas production, water production, pressures) but also at the field level (offset production, injection) and even at a macro-economic level (oil/gas price, number of rigs available, financial of the operating company, and etc.). For this example, we only look at the contextual data for a given well.

Description of Sequence-to-Sequence Model

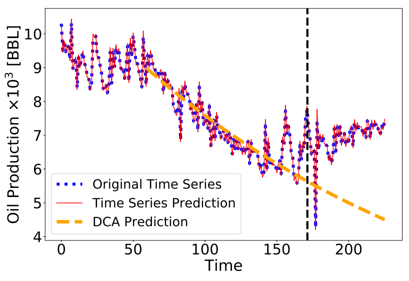
We use a sequence-to-sequence model similar to the one shown in Fig. 7. The inputs are gas production, water production, dates, number of flowing days in a month, API number (related to position). The output is Liquid cumulative production. The overall network architecture is as following: starting with an input sequence with a given dimension (10 as an example), followed by LSTM units that are connected to a fully connected layer and finally connecting to a one neuron layer with a RMSE response. The details about the size and architecture of each part of the network is given in Table 5 sub-



(a) Deep Vanilla RNN



(b) Deep Gated Recurrent Units



(c) Deep Long Short Term Memory

Figure 6: Oil production time series forecast using history of production - 75%-25% split over training and test sets - DVRNN under/overshoots the peaks and troughs with slight delay - DGRU leads to less delay, however, under/overshoots still present - DLSTM is reasonably robust in capturing the trend and spotting the peaks and troughs.

ject to a hyper-parameter study.

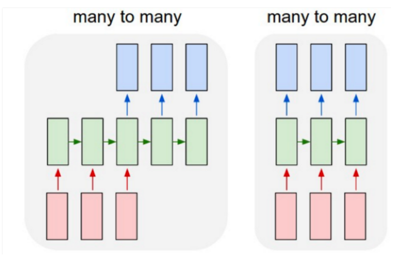


Figure 7: A many-to-many sequence model, multivariate input sequences are used to predict and output

We consider time series of various lengths and use a minibatch gradient descent with padding in order to train the model. LSTM networks typically input data with varying sequence lengths. When passing data through the network, we must pad or truncate sequences in each mini-batch to have the specified length. To reduce the amount of padded or discarded data when padding or truncating sequences, we sort our data. Figure 8 illustrates the benefits of sorting prior to padding.

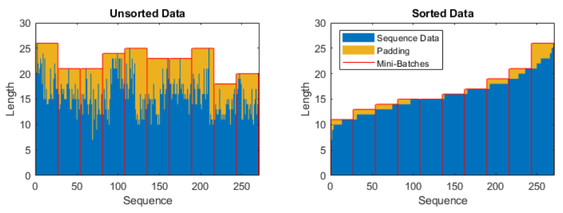


Figure 8: Padding sequences with and without sorting

We train the model on 3850 wells (testing on 350). Training loss (RMSE) is represented in Fig. 9. We train the model on a single GPU GeoForce X 1080 Ti. Each epoch takes about 10s for the hyper-parameter sets defined in Table 5.

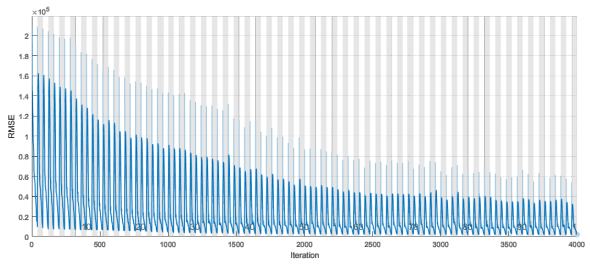
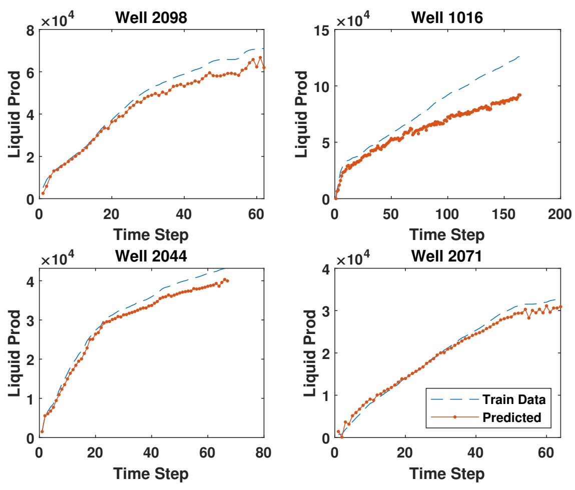


Figure 9: Training (blue) loss (RMSE) for sequence-to-sequence regression. Training occurs over 100 epochs.

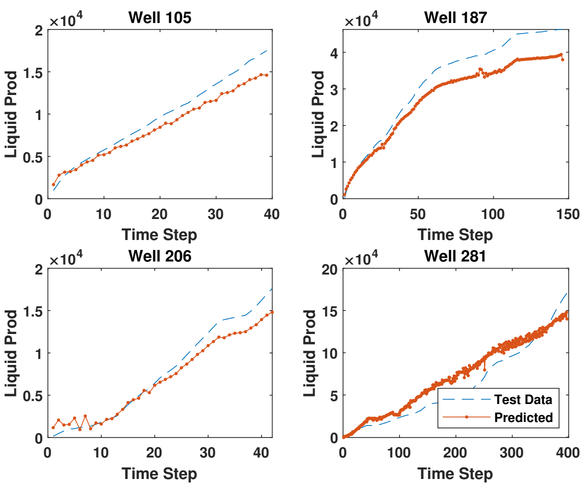
We observe a good convergence behavior for each batch and overall. We run experiments over around 1000 models. Our best model is obtained using an ensemble of models. The hyper-parameters used for a subset of these models are represented in table 5.

Sequence-to-Sequence Model Results

Fig. 10 shows the comparison of modelled and actual cumulative production for 4 wells randomly selected within the training (Fig. 10a) and test set (Fig. 10b). We see that they fit fairly well with the exception of well 1957. The mean RMSE error for the training set is 63,000 while it exceeds 120,000 for the test set . The model learns quite well and the match on some wells is nearly perfect while it misses on others. It is interesting to observe that the model fails to capture the monotone nature of cumulative production while the overall magnitude seems respected. As expected, the fit on the test set is slightly worse than for the training set. More importantly, we tested this model on the cumulative production. The signal is hence fairly smoothed compared to the original rate. We will migrate to the direct rate prediction in the future but in this work we aimed to produce accept-



(a) for training set



(b) for test set

Figure 10: Examples of predicted vs actual cumulative production

Table 5: RMSE values of different experiments on Sequence-to-Sequence Regression model

Hidden units	Dropout	FC units	Learning rate	L2-reg	RMSE
140	0.5	72	0.008	0.0005	21
100	0.5	72	0.008	0.0004	28
200	0.5	72	0.008	0.0005	31
200	0.4	76	0.008	0.0005	38

able results on the cumulative. The distribution of the RMSE over the whole test set is presented in Fig. 11.

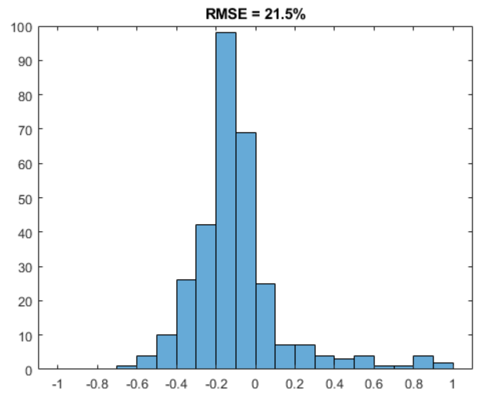


Figure 11: Distribution of RMSE over test set for Sequence-to-Sequence regression model

We present the results of sequence-to-sequence model's hyper-parameter tuning in Table 5. These results are slightly worse than the baseline set by the classical DCA approach (30% RMSE). However, none of the historical data is used to compute them. We use secondary metrics only to compute the liquid production. This exercise allows to capture long term effects of production (such as a slow down or an increase). This is what we see in the cumulative production trends. It is a necessary complement to the prediction based purely on historical production presented in an earlier part in this work as that approach can capture short term variations very well but cannot make predictions in the long term.

Future Works

An advantage of DCA is that it provides a long term estimate of production that is inaccurate but very consistent. It is a model with high bias and very low variance. LSTM models tend to have very high variance but they fail to identify long term trends in predictions unless the predictions constantly get adjusted. This approach, although valuable for short term predictions, fails to provide a holistic substitute for DCA which remains for simplicity and robustness reason the most widely used approach in the industry. In the future steps, we will improve the individual models we have hereby presented and merge them in order to produce a holistic forecast. A promising approach would be to use encoder-decoder Sequence-to-Sequence models where all the contextual data (static and dynamic) would be integrated into the encoder and the actual prediction would be made based on the encoding rather than the actual data. This would allow a denser representation of the data and help with comparing wells with each other.

References

J.J. Arps. Analysis of Decline Curves, 1945.

S.D. Mohaghegh, O. Grujic, S. Zargari, and M. Kalantari. SPE 143875 Modeling , History Matching , Forecasting and Analysis of Shale Reservoirs Performance Using Artificial Intelligence Top-Down , Intelligent Reservoir Modeling for Shale Formations. *SPE Digital Energy Conference and Exhibition*, page 14, 2011. doi: 143875.

Tita Ristanto. Deep Learning Application in Well Production. *Deep Learning Course Reports, CS230, Stanford University*, 2018.

J Sun, X Ma, M Kazi, and C S E Icon. SPE-190104-MS Comparison of Decline Curve Analysis DCA with Recursive Neural Networks RNN for Production Forecast of Multiple Wells. 2018.