IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version January 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Forecasting copper electrorefining cathode rejection by means of recurrent neural networks with attention mechanism

PEDRO PABLO CORREA[1], ALDO CIPRIANO[2] (Senior Member, IEEE), FELIPE NUÑEZ[2] (MEMBER, IEEE), JUAN CARLOS SALAS[3] and HANS LOBEL[1,4] (MEMBER, IEEE)

[1]Department of Computer Science, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860, Santiago 7820436, Chile
[2]Department of Electrical Engineering, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860, Santiago 7820436, Chile
[3]Department of Mining Engineering, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860, Santiago 7820436, Chile
[4]Department of Transport Engineering and Logistics, Pontificia Universidad Católica de Chile, Av. Vicuña Mackenna 4860, Santiago 7820436, Chile

Corresponding author: Pedro Pablo Correa (e-mail: pbcorrea@uc.cl).

This work was supported by CONICYT under Grant FONDEF/CONICYT 2017 IT17M10011, ANID - Millennium Science Initiative Program - Code ICN17_002, and ANID PIA ACT192013.

ABSTRACT Electrolytic refining is the last step of pyrometallurgical copper production. Here, smelted copper is converted into high-quality cathodes through electrolysis. Cathodes that do not meet the physical quality standards are rejected and further reprocessed or sold at a minimum profit. Prediction of cathodic rejection is therefore of utmost importance to accurately forecast the electrorefining cycle economic production. Several attempts have been made to estimate this process outcomes, mostly based on physical models of the underlying electrochemical reactions. However, they do not stand the complexity of real operations. Data-driven methods, such as deep learning, allow modeling complex non-linear processes by learning representations directly from the data. We study the use of several recurrent neural network models to estimate the cathodic rejection of a cathodic cycle, using a series of operational measurements throughout the process. We provide an ARMAX model as a benchmark. Basic recurrent neural network models are analyzed first: a vanilla RNN and an LSTM model provide an initial approach. These are further composed into an Encoder-Decoder model, that uses an attention mechanism to selectively weight the input steps that provide most information upon inference. This model obtains 5.45% relative error, improving by 81.4% the proposed benchmark. Finally, we study the attention mechanism's output to distinguish the most relevant electrorefining process steps. We identify the initial state as a critical state in predicting cathodic rejection. This information can be used as an input for decision support systems or control strategies to reduce cathodic rejection and improve electrolytic refining's profitability.

INDEX TERMS Deep learning, Electrorefining, Predictive models, Recurrent neural networks.

I. INTRODUCTION

The electrolytic refining process is one of the last steps in sulfide-extracted copper production. Through this process, cathodes with over 99.99% copper purity are obtained from raw anodes, using electrolysis [1]. First, copper is electrochemically dissolved into an electrolyte containing $CuSO_4$ and H_2SO_4. After this, the metal is selectively electroplated into high purity cathodes [2].

Due to commercial requirements, copper cathodes must meet several quality standards. These include a high metal purity and the absence of physical impurities, such as nodules or dendrites [3]. Non-compliant cathodes are rejected, and reprocessed or sold at a lower price. The fraction of cathodes rejected (over the complete production) is known as physical cathodic rejection.

Electrorefining process objective is to maximize the production of grade A copper cathodes. These offer a premium-price, and so a higher profitability of the whole process. Cathodic rejection represents the efficiency of the process in the production of grade A cathodes. An accurate estimation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074780, IEEE Access

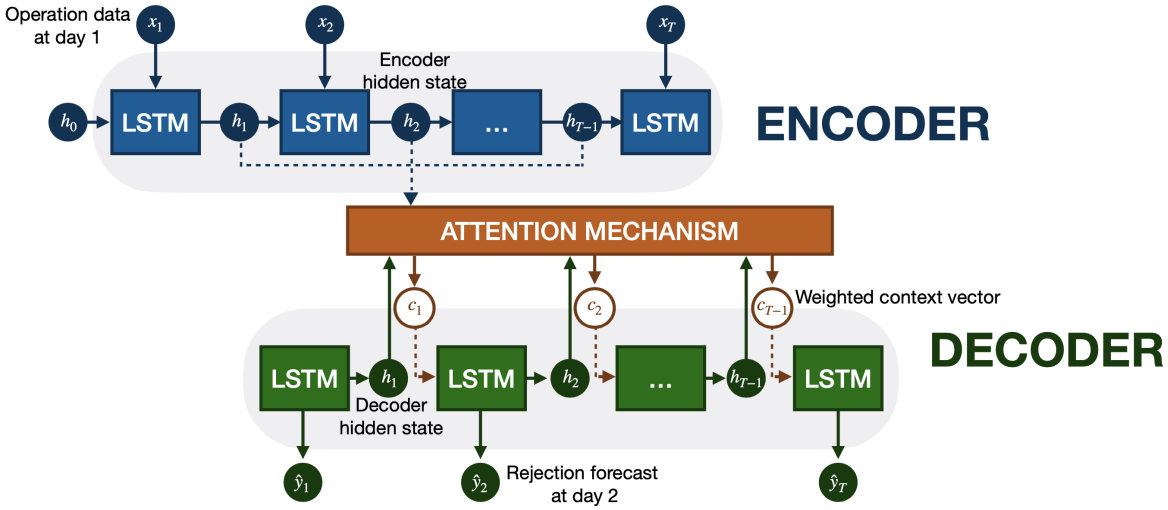

FIGURE 1. Proposed Encoder-Decoder model. Every step, the attention mechanism selects the most relevant input steps to make a prediction.

of this value can be used as input for operational decision support systems, as well as control strategies.

Several approaches have been proposed to characterize the electrorefining process outcomes. These are mainly based on physical representations of the electrolytic refining process. However, due to the underlying interactions' complexity, these methods have not been able to provide an accurate estimation.

Deep learning techniques have become a prominent tool in the time series forecasting setting [4], [5]. These are based on neural networks, a combination of non-linear operations. Feature engineering is avoided by learning the intermediate representations directly from the data. This allows obtaining a precise model of highly complex and non-linear processes, in a completely data-driven fashion. Neural networks have helped develop several breakthroughs in computer vision [6], natural language processing [7], and time series forecasting [8].

Recurrent neural networks (RNN) are a family of neural networks for processing sequential data [9]. Using a time-based backpropagation algorithm [10], they can learn complex time-dependent patterns. The Encoder-Decoder architecture is based on an ensemble of RNNs. This model can process variable-length sequences, by using a fixed representation in a latent space. Attention mechanisms [11] further improve this representation by creating a time-weighted encoding for the sequence at each step. Encoder-Decoder models with attention can handle multi-step time dependencies between multivariate inputs, which makes them particularly appealing for time series forecasting.

In this work, we propose a forecasting mechanism to estimate the cathodic rejection of the electrorefining process. We compare several RNN architectures using real operational data. We show that a Encoder-Decoder architecture with an attention mechanism, shown in Fig. 1, achieves the highest performance of the reviewed models. We also analyze the attention weights as sources of information to interpret the model's inference process.

Our contribution to the presented problem is two-fold:

- We present a robust forecasting methodology of electrorefining cathodic rejection. For this, we compare several recurrent neural network architectures, ranging from a vanilla RNN network to an Encoder-Decoder architecture with an attention mechanism.
- We explore the attention mechanism's weights, to analyze the most relevant information used by the model to predict this indicator. This helps us explain the inference process of the Encoder-Decoder model, which allows us to shed light on a traditionally black-box system, such as neural networks.

The remainder of the manuscript is organized as follows. Section II reviews previous approaches to the proposed problem. Section III details the deep learning methods used, as well as the proposed benchmark. Methodology is presented in Section IV. Section V analyzes and discusses the results. Finally, conclusions are included in Section VI.

II. RELATED WORK

The electrorefining process has been studied through different scientific perspectives. Prior research has been focused on the development of physically based representations to model the phenomena. This has been achieved by different means, such as the characterization of the physico-chemical processes involved [12] and simulation of the impurities behaviour [13] or variable effects on the process outcome [14].

Different authors have also studied the process results forecasting. They have been able to identify the correlation between process variables and cathodic rejection [3], predict current efficiency [15], [16] or select the most relevant variables that correlate with the quality of the outcomes [17].

However, the existing research is primarily based on the theoretical representation of the phenomena and has difficulties when dealing with the electrolytic refining operations' real conditions.

Data-driven methods have also been used to model operational indicators in copper refinery processes. Most of these are based on classical machine learning techniques for supervised learning, such as SVM, kNN for classification, and generalized linear models or single layer neural networks (NN) for regression tasks [18], [19]. However, these model don't fully exploit the time-series nature of the indicators, leaving space for testing more complex deep learning techniques, such as the use of recurrent neural network architectures.

Finally, the Encoder-Decoder architecture has proven to be successful in operational regression settings, such as forecasting operational indicators for thickening processes [20], [21]. However, the value of attention weights as information sources has not been established. To the best of our knowledge, this is the first proposed application of recurrent neural networks to predict cathodic rejection in an electrorefining operational setting.

III. PROPOSED MODELS

We present the theoretical background of the proposed models. A brief description for each one of these is provided. We also describe the benchmark used to compare the forecasting performance.

A. TIME SERIES APPROACH

To establish a baseline for Current Efficiency forecasting, we set an Auto Regressive Moving Average with Exogenous Inputs model (ARMAX) as a benchmark. This model allows forecasting a time series, by using multiple variables as external information. The proposed implementation follows [22].

Formally, given a sequence $Y = \{y_1, \dots y_{t-1}\}$, an ARMAX model predicts y_t through the following equation system:

$$y_t = \sum_{j=1}^{p} A_j y_{t-j} + \sum_{j=1}^{r} D_j x_{t-j} + \sum_{j=0}^{q} B_j \epsilon_{t-j} + \epsilon_t \quad (1)$$

Here, $A_j \in \mathbb{R}, B_j \in \mathbb{R}$ and $D_j \in \mathbb{R}^M$ are the autoregressive, moving-average and exogenous parameters, respectively. Also, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the error term for a given time step t.

B. RECURRENT NEURAL NETWORKS

Recurrent neural networks [23] are a family of neural networks used for processing sequential data [9]. The use of an internal recurrent representation of the sequence, allows these model to retain information from different steps, thus modelling temporal relationships within the sequence [24].

A recurrent neural network is governed by its hidden state at time step t, $h^{(t)}$. This value acts as an internal representation of relevant information from the existing series, up to the given step t. Given a sequence of values $\{x_1, \dots x_t\}$, the hidden state update process is computed recursively as a function of itself, the inputs, and other model parameters:

$$h_t = F(h_{t-1}, x_t; \theta) \quad (2)$$

In the case of a vanilla RNN, the weights update process is given by [9]:

$$h_t = \tanh(W \cdot [h_{t-1}, x_t] + b) \quad (3)$$

Where W is the model´s weight matrix, and b biases vector.

The training of these models involves the use of back-propagation-through-time algorithm, which implies the multiplication of the different step derivatives [10]. Therefore, small and large gradients effects is amplified. This affects the numerical stability of the training process, specially on larger sequences. This are known as the exploding and vanishing gradients problem, respectively [25].

C. LONG SHORT TERM MEMORY

Long Short Term Memory Cells [26] are introduced to handle longer sequences. Here, information is adaptively selected, using an *input*, *forget* and *output* gating operations. This reduces the effect of extreme values on downstream calculation, which in turn stabilizes the training process.

The update step of the cell state involves computing forget (W_f, b_f), input (W_i, b_i) and cell (W_c, b_c) states:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{c}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_c) \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \end{aligned} \quad (4)$$

After this, the new hidden state update process is computed by the output gate (W_o, b_o):

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \otimes \tanh(c_t) \end{aligned} \quad (5)$$

Here, $\otimes$ denotes the Hadamard product. This family of models perform better on longer sequences than vanilla RNNs [9]. However, the improved performance comes at a cost, as the increased number of parameters results in a more complex model. This adds training overhead, and makes the model overfitting-prone.

D. ENCODER-DECODER MODELS

Encoder-Decoder models [27] use a combination of RNNs to model sequences. Both Encoder and Decoder are trained to maximize the conditional probability of a target sequence, given an input sequence [28]. This is accomplished through the use of a fixed representation in a latent space, called context vector. A diagram of the forecasting process is shown in Figure 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074780, IEEE Access

Given the input sequence $\{x_1, \ldots, x_{T_x}\}$, an Encoder is used to compute the hidden states, using a LSTM model:

$$h_t^{\text{enc}} = \text{LSTM}_{\text{enc}}(h_{t-1}, x_t) \tag{6}$$

An attention mechanism [11], [29] uses the Encoder hidden states to adaptively weight the input values, creating a time-dependent context vector. The use of a softmax layer ensures this vector represents a distribution over input values. It's computation follows [11]:

$$c_t = \sum_{i=1}^{T_x} \alpha_i \cdot h_s^{\text{enc}} \tag{7}$$

$$\alpha_t(s) = \frac{\exp a(h_t^{\text{dec}}, h_s^{\text{enc}})}{\sum_{s'} \exp a(h_t^{\text{dec}}, h_{s'}^{\text{enc}})} \tag{8}$$

Here, $a(\cdot, \cdot)$ is known as the *scoring function*:

$$a(h_i^{\text{enc}}, h_s^{\text{dec}}) = (h_i^{\text{enc}})^{\text{t}} \cdot W_a \cdot h_s^{\text{dec}} \tag{9}$$

The time-variable context vector is then fed to the Decoder. This models the conditional distribution of the target sequence, using a second LSTM model:

$$p(y) = \prod_{t=1}^{T_x} p(y_t | \{y_1, \ldots, y_{t-1}\}, c_t)$$

$$p(y_t | \{y_1, \ldots, y_{t-1}\}, c) = \text{LSTM}_{\text{dec}}(y_{t-1}, h^{\text{dec}}, c_t) \tag{10}$$

Finally, the output layer generates the predicted sequence:

$$y_t = W_{\text{dec}} \cdot h_t^{\text{dec}} \tag{11}$$

IV. METHODOLOGY

A. DATA

The dataset used for the study consists of information from an electrolytic refining plant, containing 962 cathodic cycles, totaling 11,906 daily observations. Each of these contains several variables, measured with different granularity. Cathodic rejection is measured at the end of each cycle. To avoid time-scale effects, we consider the daily average of each measurement. Tables 1 and 2 show the data sources used. A detailed description of the input values is available in Appendix A.

TABLE 1. Input data description. We group the variables according to their source.

Group	#	Description	Examples
Electrolyte	10	Impurities concentration.	As(g/l); Pb(g/l).
Cathode	6	Impurities concentration.	Ag(g/t); Pb(g/t).
Electrical	4	Current measurements.	Short-circuit count.
Other	5	Operational measurements.	Acid level.

We scale the variables, to ensure the condition $x \in [0, 1]$. This allows to guarantee a correct training process and eliminate the effect of the input scales [30]. Missing values were filled using a kNN imputation strategy [31], to avoid reducing the dataset size.

TABLE 2. Output data description.

Variable name	Unit	Description
Cathodic rejection	%	Cathodes that don't meet the quality standard on a given cycle.

The resulting data is divided into a training split of 800 cathodic processes (83.2% of the total dataset), as well as a test split with the remaining 162 sequences (16.8%).

B. METRICS

To assess the quality of the forecast, we propose the use of two metrics. In the following section, we consider a series $Y = \{y_1 \ldots y_T\}$ and its corresponding predictions $\hat{Y} = \{\hat{y}_1 \ldots \hat{y}_T\}$. :

1) Root Mean Squared Error (RMSE):

This metric is defined through the following expression:

$$\text{RMSE}(Y, \hat{Y}) = \sqrt{\frac{1}{T} \sum_{i=1}^{T} (y_i - \hat{y}_i)^2} \tag{12}$$

This metric is used as the loss function to train the neural network models, as it provides a second-differentiable error term.

One of the main drawbacks of RMSE is its sensitivity to outliers. This is due to the quadratic term, which penalizes large differences between the forecast and actual values. Also, the resulting error term does not have a physical interpretation, as it depends on the original variable unit.

2) Mean Average Precision Error (MAPE):

We compute this metric as follows:

$$\text{MAPE}(Y, \hat{Y}) = \frac{2}{T} \sum_{i=1}^{T} \frac{|y_i - \hat{y}_i|}{|y_i + \hat{y}_i|} \tag{13}$$

This metric allows for an easily interpretable error measure, regardless of the original unit. We use the symmetric version of MAPE [32], to account for prediction of zero-values, as well as over and under-estimation asymmetry.

However, MAPE cannot be used as a loss function, as it is not everywhere differentiable. This may lead to unexpected behavior during the training of the neural networks.

C. EXPERIMENTS

A training epoch for a single sequence consists on the following steps:

1) A single step, $x_t \in \mathbb{R}^K$ is fed to the model. This generates a new hidden state (two in the case of the Encoder-Decoder model).
2) The hidden state is used to compute a forecast, $\hat{y}_t$, which is in turn used to update again the hidden state.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074780, IEEE Access

These steps are repeated until the full sequence has been predicted.

3) We compute the loss, using RMSE loss function on both the forecasted sequence and the actual values.
4) Once the error has been calculated, it is back-propagated through the steps, in order to update the model's parameters.

We repeat the above process for all the sequences, in a batched fashion. We considered a total 200 epochs, using a cross-validation approach [33]. This allows for better convergence of the training, given the size of the dataset. A leave-one-out method was used, dividing the data into five folds: one for validation and the rest for training.

Neural networks require the tuning of several hyperparameters, which in turn define the behavior of the model. To reduce the computational expense of the selection process, a Bayesian optimization approach was chosen [34], [35]. For each model, we carried an optimized hyperparameter search for 50 iterations, each consisting of a full training cycle. The following parameters were considered:

- Regularization parameters, including gradient clip [25], dropout value and teacher forcing ratio [9], defined as the probability of using a ground truth value instead of the model output when training.
- Optimizer parameters. These include the optimizer selection, where stochastic gradient descent and Adam optimizer [36] were compared. We also study different values for the optimizer's learning rate, as well as the number of steps considered for learning rate schedule, also known as patience.
- Network parameters, such as the number of layers considered for each architecture, as well as the internal hidden size. In the Encoder-Decoder architecture case, both encoder and decoder hidden size were set to the same value.

The best configurations obtained are detailed in Table 3.

TABLE 3. Hyperparameter selection: Best configurations obtained from Bayesian optimization

	Model		
Hyperparameter	**RNN**	**LSTM**	**Enc-Dec**
Dropout	0.17	0.19	0.22
Gradient clip	5	5	2
Hidden size	114	197	155
Learning rate	7.8×10^{-3}	9.0×10^{-3}	8.75×10^{-3}
Decoder layers	-	-	4
Encoder layers	-	-	7
Layers	6	7	-
Optimizer	adam	sgd	adam
Patience	10	11	9
Teacher forcing	0.11	0.13	0.21

V. RESULTS AND DISCUSSION

We present the cathodic rejection forecasting results.

First, we analyze the training process. For this, we compare training and validation metrics for all neural network models.

We then present inference results. Here, we use the test set to compare the proposed models with the benchmark. We discuss error rates, as well as its distribution over time.

We then focus on the Encoder-Decoder model. A random sample from the test set is selected to visualize the inference process. We include an evaluation of the error rate, as well as an analysis of the attention weights, obtained as a by-product of the attention mechanism.

A. TRAINING

Table 4 presents both training and validation metrics, for each model. To visualize the training process, Fig. 2 shows the evolution of the loss function on the validation dataset.

TABLE 4. Error metrics on training set

MAPE [%]			
Model	RNN	LSTM	Enc-Dec
Training	12.97	10.39	**3.59**
Validation	17.56	16.59	**5.26**
RMSE [%]			
Model	RNN	LSTM	Enc-Dec
Training	5.56	3.82	**1.41**
Validation	6.69	5.28	**1.53**

The low difference between training and validation error suggests a correct fit of each model on the proposed dataset.

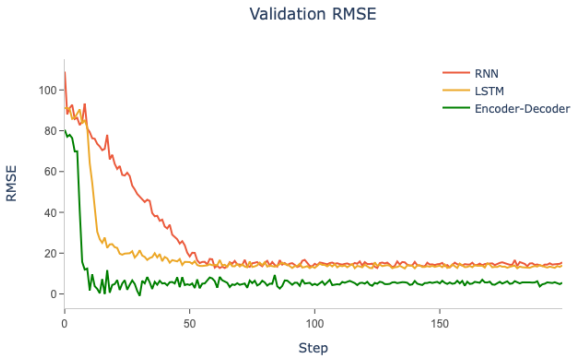

FIGURE 2. Loss function in validation set. The convergence rate increases with the complexity of the model.

Vanilla RNN obtains the highest error values for both metrics. This can be explained due to the lower complexity of the network and thus a lower capacity to model longer-range temporal dependencies, suggesting that capturing this kind of dependency is relevant for prediction.

The use of LSTM cells speeds up the convergence process. Even though the final error metrics of this model are similar to RNN, the convergence to this value is significantly faster. This suggests a higher capacity on modeling the sequences, but a cap on the dataset size.

The Encoder-Decoder model achieves the lowest error values, both on training and validation set. Also, this model is the fastest to achieve training convergence. This shows

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074780, IEEE Access

how temporal inter-dependencies are captured better by the attention mechanism, compared to the internal representation of RNN or LSTM.

B. INFERENCE

We now turn our attention to the inference results in the test set. The proposed time series model results are included as a benchmark. Table 5 compares the test set metrics, for all available models.

TABLE 5. Error metrics on test set

Model	ARMAX	RNN	LSTM	Enc-Dec
RMSE [%]	8.31	5.91	5.02	**1.96**
MAPE [%]	29.31	16.85	15.89	**5.45**

It is possible to see how all recurrent neural network models outperform the proposed benchmark on both metrics. Considering the relative error, vanilla RNN improves the baseline by 18.45%, while LSTM does it by 32.37%. Furthermore, the Encoder-Decoder shows an improvement of 84.14% on MAPE. This can be attributed to the attention mechanism's higher capacity to represent the relationship between the exogenous variables and the output.

To assess the forecasting horizon's impact on the inference quality, we analyze the temporal behavior of the relative error. Fig. 3 shows the forecast error for different amounts of information available for each model.

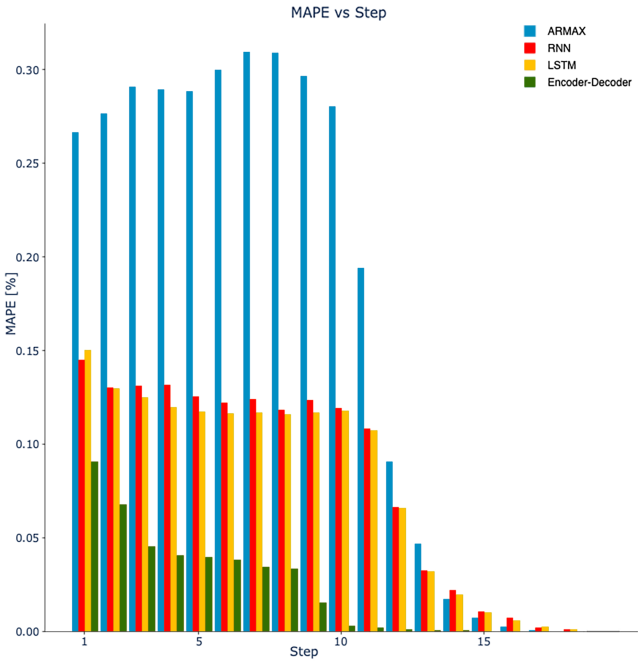

FIGURE 3. Distribution of MAPE. The Encoder-Decoder architecture shows a steady decrease, as the amount of available information increases.

It is possible to see how the ARMAX error terms do not improve when more information is available. Given the autoregressive components, the variance of the error increases, due to the lineal composition with previous predictions.

Both vanilla RNN and LSTM models show a steady decrease in their metrics when available information increases. In the latter case, the lower error levels are associated with a better handle of long term dependencies of the sequences.

Encoder-Decoder model requires the least amount of steps to achieve convergence in relative error. It also requires the least amount of epochs to achieve a stable error value. Because of this, it appears to be the most suitable tool to forecast the cathodic rejection of the electrorefining process.

Having an error-rate below 5% reduces the uncertainty associated to the electrorefining process outcomes. Furthermore, a precise estimation of cathodic rejection has direct relation with the amount of grade A (high purity) cathodes produced per cycle [1]. This is critical for short term profit estimation, as well as long term financial planning of the operation [37].

C. ATTENTION WEIGHTS

We present a sample sequence to analyze the forecasting process for the Encoder-Decoder model. The corresponding predictions, as well as the actual values, are included in Fig. 4.

FIGURE 4. Encoder-Decoder model prediction. The forecast rapidly converges to the actual value, keeping a low error rate on the following steps.

The Encoder-Decoder model starts with a MAPE of less than 1%, at the first prediction step. This value further decreases consistently up to step 7. While the relative error increases slightly after that, it stays under 1%. This behavior is consistent across all evaluated samples, as Fig 3 presents. This shows the predictive capability of the Encoder-Decoder model, which allows for an accurate forecast of cathodic rejection from the very beginning of the process.

In order to understand how the error rate convergence is achieved at the first few steps, we analyze the attention weights obtained during the inference process. These are depicted in Fig. 5.

The first row represents the only available attention value used to forecast. The second one shows the relevance of the two previous values, and so on. The last row shows the full weighted input, which is used to forecast the final value. From the results, it is clear that the attention mechanism assigns larger weights to the first steps of the sequence. This suggests that a higher relevance is placed on the electrore-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3074780, IEEE Access

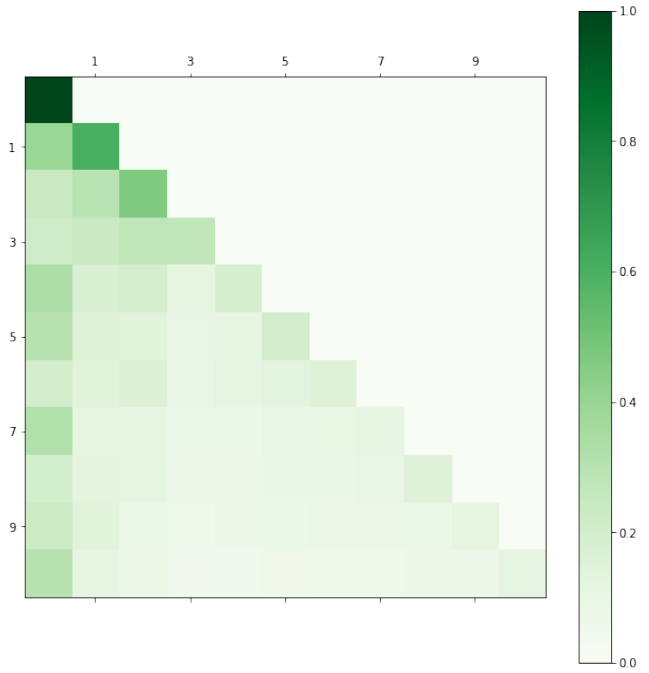

FIGURE 5. Attention values for a single sequence. Each row of the matrix represents the attention weights assigned to the sequence's previous values, up to the current step, represented by the column.

fining process's initial state, even when more information is available.

It is also important to notice the values assigned to the last known element, represented on the attention map's diagonal values. From these, it is clear that the last known value is also taken into consideration by the attention mechanism when performing inference. This makes sense if we take into consideration that this value represents the only new information available to the model between two consecutive steps.

VI. CONCLUSIONS

In this investigation, several recurrent neural network models were designed to predict the electrorefining cathodic rejection. Using real operational data from an electrorefining operation, different architectures were compared in terms of accuracy and convergence speed, using a traditional time series model as a benchmark.

Using a real-world example, we lay out a regression problem. Here, cathode rejection is forecasted throughout the cathodic cycle, while the actual value is obtained at the end of the process. We also present two metrics to train the models and compare the different approaches.

Three RNN networks are proposed to solve the problem mentioned above. A vanilla RNN, an LSTM based model, and a sequence-to-sequence architecture, equipped with an attention mechanism. The last one shows an improvement of 84.1%, in terms of relative error, against the benchmark. The error converges steadily from the second to the third step of the sequence. This allows obtaining an accurate forecast in the early stages of the process, which helps to reduce the uncertainty of the process outcomes.

We also show how the attention weights obtained from the model's inference can be used to obtain valuable information from the process. This allows for better process monitoring, as well as cycle optimization.

From the presented results, it is possible to conclude that the use of an attention mechanism allows obtaining an accurate forecast of the current efficiency from an electrorefining process in a timely manner.

Several research questions for future research arise from this investigation.

In the first place, a larger dataset is needed in order to test more complex approaches. Even though we present an attention mechanism that improves the existing approaches, there are other alternatives that, given a large enough dataset, can enrich the analysis. Some of these are:

1) **Double Attention:** This mechanism, proposed in [38], allows to obtain attention maps from both a temporal and a spatial level. This allows identifying critical steps in the process, as well as the most relevant variables that explain the forecasting result.
2) **Convolutional Attention:** The use of convolutional attention models, such as the Transformer [39] allows to speed up significantly the training process. This is due to the fact that convolutional neural networks don't require recursive computation, and thus may be trained in a parallel fashion.This, in turn, helps to achieve better results by using more complex models.

Finally, future research includes studying the interaction with model processing controllers. An accurate and robust modelling methodology for the electrorefining cathodic rejection, may be used to estimate the outcomes on process control actions. This may be used to further reduce rejection, enhancing the quality of the production.

.

APPENDIX A DATA DESCRIPTION

We provide a detailed description of the input values used for training the different models on Table 6. This includes the name of the variable, a brief description of it, and its main statistical indicators.

REFERENCES

[1] M. E. Schlesinger, K. C. Sole, and W. G. Davenport, Extractive metallurgy of copper. Elsevier, 2011.
[2] M. S. Moats, How to Evaluate Current Efficiency in Copper Electrowinning. Society for Mining, Metallurgy, and Exploration, 2012.
[3] G. Cifuentes, C. Vargas, and J. Simpson, "Análisis de las principales variables de proceso que influyen en el rechazo de los cátodos durante el electrorrefino del cobre," Revista de metalurgia, vol. 45, no. 3, pp. 228–236, 2009.
[4] J. C. B. Gamboa, "Deep learning for time-series analysis," arXiv preprint arXiv:1701.01887, 2017.
[5] M. Han and Y. Wang, "Analysis and modeling of multivariate chaotic time series based on neural network," Expert Systems with Applications, vol. 36, no. 2, pp. 1280–1290, 2009.

TABLE 6. Input data description

Group	Variable	Description	Mean	Std	NaN (%)
Cathodic	Ag(g/t)	Silver concentration.	4.38×10^{-5}	1.59×10^{-5}	0.0
Cathodic	Bi(g/t)	Bismuth concentration.	1.12×10^{-6}	4.0×10^{-7}	0.0
Cathodic	Ca(g/t)	Calcium concentration.	2.7×10^{-7}	1.49×10^{-7}	0.0
Cathodic	O2(g/t)	Oxigen concentration.	1.1×10^{-4}	3.86×10^{-5}	0.0
Cathodic	Pb(g/t)	Lead concentration.	5.39×10^{-6}	2.28×10^{-6}	0.0
Cathodic	Sb(g/t)	Antimony concentration.	1.63×10^{-5}	6.19×10^{-6}	0.0
Electrical	Current	Cumulative current intensity.	20.61	4.92	0.0
Electrical	Short Circuit A	Short-circuit count at circuit A.	26.07	31.64	0.0
Electrical	Short Circuit B	Short-circuit count at circuit B.	23.15	28.93	0.0
Electrical	Voltage	Electrode's potential difference.	11.23	2.01	0.0
Electrolyte	As(g/l)	Arsenic concentration.	7.31	0.93	0.0
Electrolyte	Bi(mg/l)	Bismuth concentration.	27.89	9.76	0.0
Electrolyte	Ca(g/l)	Calcium concentration.	0.36	0.05	0.0
Electrolyte	Cl(mg/l)	Chlorine concentration.	61.34	10.58	0.0
Electrolyte	Fe(g/l)	Iron concentration.	0.17	0.02	0.0
Electrolyte	Fe2(g/l)	Iron sulfate concentration.	0.1	0.02	0.0
Electrolyte	Ni(g/l)	Nickel concentration.	0.2	0.04	0.0
Electrolyte	Pb(mg/l)	Lead concentration.	9.92	1.74	0.0
Electrolyte	Sb(g/l)	Antimony concentration.	0.28	0.06	0.0
Electrolyte	Temperature(ºC)	Electrolyte temperature.	63.46	4.23	0.0
Other	Water flow (L/m)	Electrolyte water flow.	3.72	2.38	6.73
Other	Acid flow (L/m)	Acid flow.	1.62	1.57	6.19
Other	Anode weight	Anode weight.	9.29×10^{5}	3.96×10^{2}	0.0
Other	Tank A level (l).	Electrolyte stock at Tank A.	65.2	15.19	1.44
Other	Tank B level (l).	Electrolyte stock at Tank B.	64.99	12.45	5.02

[6] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural computation, vol. 1, no. 4, pp. 541–551, 1989.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," The journal of machine learning research, vol. 3, pp. 1137–1155, 2003.

[8] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in 2014 IEEE symposium on computational intelligence in ensemble learning (CIEL). IEEE, 2014, pp. 1–6.

[9] I. Goodfellow, Deep Learning (Adaptive Computation and Machine Learning series). The MIT Press, nov 2016. [Online]. Available: https://www.xarg.org/ref/a/0262035618/

[10] P. J. Werbos, "Backpropagation through time: what it does and how to do it," Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990.

[11] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," CoRR, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[12] D. C. Price and W. G. Davenport, "Physico-chemical properties of copper electrorefining and electrowinning electrolytes," Metallurgical Transactions B, vol. 12, no. 4, pp. 639–643, 1981.

[13] W. Zeng, M. L. Free, J. Werner, and S. Wang, "Simulation and validation studies of impurity particle behavior in copper electrorefining," Journal of The Electrochemical Society, vol. 162, no. 14, p. E338, 2015.

[14] M. S. Moats, J. B. Hiskey, and D. W. Collins, "The effect of copper, acid, and temperature on the diffusion coefficient of cupric ions in simulated electrorefining electrolytes," Hydrometallurgy, vol. 56, no. 3, pp. 255–268, 2000.

[15] Z. Zhang, J. Werner, and M. Free, "A current efficiency prediction model based on electrode kinetics for iron and copper during copper electrowinning," in The Minerals, Metals & Materials Series. Springer International Publishing, 2018, pp. 111–131.

[16] A. Alfantazi and D. Valic, "A study of copper electrowinning parameters using a statistically designed methodology," Journal of applied electrochemistry, vol. 33, no. 2, pp. 217–225, 2003.

[17] P. E. Aqueveque, E. P. Wiechmann, J. Herrera, and E. Pino, "Measurable variables in copper electrowinning and their relevance to predict process performance," in 2013 IEEE Industry Applications Society Annual Meeting. IEEE, Oct. 2013.

[18] J. McCoy and L. Auret, "Machine learning applications in minerals processing: A review," Minerals Engineering, vol. 132, pp. 95–109, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0892687518305430

[19] L. Perez, "Classification and regression models in copper refinery," Mineral Processing and Extractive Metallurgy, pp. 1–7, 2021.

[20] F. Núñez, S. Langarica, P. Díaz, M. Torres, and J. C. Salas, "Neural network-based model predictive control of a paste thickener over an industrial internet platform," IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2859–2867, 2019.

[21] Z. Yuan, J. Hu, D. Wu, and X. Ban, "A dual-attention recurrent neural network method for deep cone thickener underflow concentration prediction," Sensors, vol. 20, no. 5, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/5/1260

[22] M. S. Paolella, Linear Models and Time-Series Analysis: Regression, ANOVA, ARMA and GARCH. John Wiley & Sons, 2018.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, vol. 323, no. 6088, pp. 533–536, 1986.

[24] A. Graves, "Supervised sequence labelling," in Supervised sequence labelling with recurrent neural networks. Springer, 2012, pp. 5–13.

[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in International conference on machine learning, 2013, pp. 1310–1318.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[28] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 103–111. [Online]. Available: https://www.aclweb.org/anthology/W14-4012

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[30] S. García, Data Preprocessing in Data Mining (Intelligent Systems Reference Library). Springer, aug 2014.

[31] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays," Bioinformatics, vol. 17, no. 6, pp. 520–525, 2001.

[32] S. Makridakis, "Accuracy measures: theoretical and practical concerns," International Journal of Forecasting, vol. 9, no. 4, pp. 527–529, 1993.

[33] R. R. Picard and R. D. Cook, "Cross-validation of regression models," Journal of the American Statistical Association, vol. 79, no. 387, pp. 575–583, 1984.

[34] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained bayesian optimization with noisy experiments," Bayesian Anal., vol. 14, no. 2, pp. 495–519, 06 2019.
[35] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in Advances in neural information processing systems, 2012, pp. 2951–2959.
[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.
[37] S. A. Cerón Amigo, "Desarrollo de modelo de costos de operación de la línea de procesamiento fundición y refinería del cobre para evaluación económica," 2019.
[38] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," arXiv preprint arXiv:1704.02971, 2017.
[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, 2017, pp. 5998–6008.