**Battery Voltage Prediction Using LSTM, Regression, and ARIMA Models**

**Abstract**

Battery performance prediction is crucial in the development and maintenance of voltage laboratories with Josephson-Junction system. This study investigates the application of machine learning and statistical models to forecast battery voltage levels over time using a dataset consisting of 104 real-world measurements recorded under controlled laboratory conditions (T = 23°C, RH = 55%, [Ref. 1]). Three modeling approaches were explored: Long Short-Term Memory (LSTM)-Neural Networks, Polynomial Regression, and AutoRegressive Integrated Moving Average (ARIMA) models. LSTM, with its ability to learn temporal dependencies, was complemented by traditional time series and regression techniques for comparative analysis. Experimental results indicate that LSTM provides the most accurate predictions, followed by ARIMA and Polynomial Regression. The integration of these techniques presents a robust methodology for battery voltage forecasting.

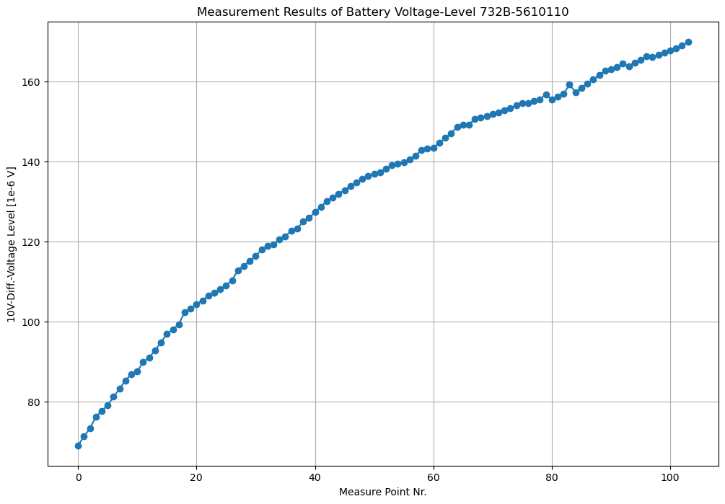
**1. Introduction**

Accurate prediction of battery voltage levels is vital for maintenance of calibration systems. It is important to forecast how long batteries can be used accurately for calibration in ppm voltage levels. Traditional regression and time series models, while useful, often fall short when modeling complex nonlinear temporal dependencies. This study leverages advanced deep learning — particularly LSTM and ARIMA models — and compares their performance with conventional models such as Polynomial and XGBoost Regression using a small, high-quality real-world dataset.

**2. Methodology**

**2.1 Data Description**

The dataset was measured in the Voltage Laboratory at TUBITAK UME [Ref. 1] and comprises 104 measurements of battery voltage levels taken at regular intervals under stable laboratory conditions (temperature 23°C, relative humidity 55%). Each voltage value corresponds to one year of use, capturing the degradation trend over time(*Fig. 2.1.1*).



*Fig. 2.1.1: Plot of the Dataset used in this Work*

**2.2 LSTM Model**

Four different models were implemented using TensorFlow/Keras Python packages. Three models were introduced in the Machine Learning Course [Ref. 2]. The 4th Model is a univariate LSTM-Model and was developed with one LSTM layer of 512 units and one Dense output layer. The input of this model was framed as a time window of past voltage values, and the model was trained over 300 epochs with **Mean-Squared-Error (MSE)** loss.

The main developed architecture part of the 4th Model in Python code is:

**window\_size = 4**

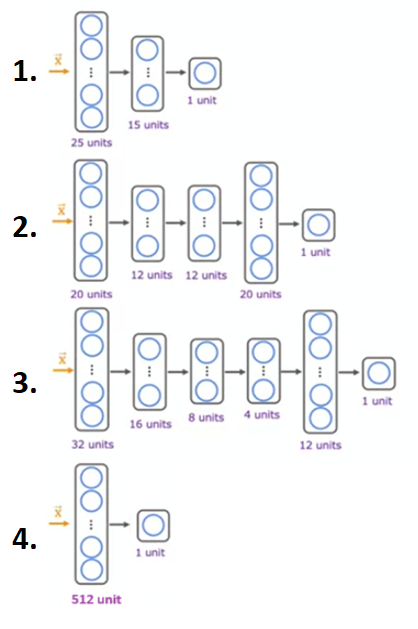
**model = tf.keras.Sequential([**

**tf.keras.layers.LSTM(512, input\_shape=(window\_size, 1)),**

**tf.keras.layers.Dense(1)**

**])**

All performed LSTM-Model architectures are illustrated in *Fig. 2.2.1*.



*Fig. 2.2.1: The four LSTM-Model Architectures used in this Work*

The first three models (1. – 3.) are Lab examples of Coursera’s “Advanced Learning Algorithms” [C2W3\_Diag\_Bias\_Variance, Ref. 2]. The 4th Model was developed by ourselves. It will be explained in detail in the next sections.

**LSTM cells**

The *Long Short-Term Memory* (LSTM) cell was proposed in 1997 by Sepp Hochreiter and Jurgen Schmidhuber [Ref. 3] and gradually improved over the years by several researchers, such as Alex Graves, Haşim Sak1, and Wojciech Zaremba2. If you consider the LSTM cell as a black box, it can be used very much like a basic cell, except it will perform much better; training will converge faster, and it will detect long-term dependencies in the data. LSTM cells are be used in Neural Network (NN) nets for sequence processing [Ref. 5].

A common method in deep learning algorithms is to calculate the **Cross-Validation (CV)** of the dataset to determine the minimum error and so to evaluate the performance of a model and to make a decision, which model fits the dataset optimal [Ref. 2].

1 Haşim Sak et al., “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition,” arXiv preprint arXiv:1402.1128 (2014).

2 Wojciech Zaremba et al., “Recurrent Neural Network Regularization,” arXiv preprint arXiv:1409.2329 (2014).

The equation for calculating the MSE of the dataset’s Cross-Validation set is:

***(2.1)*,**

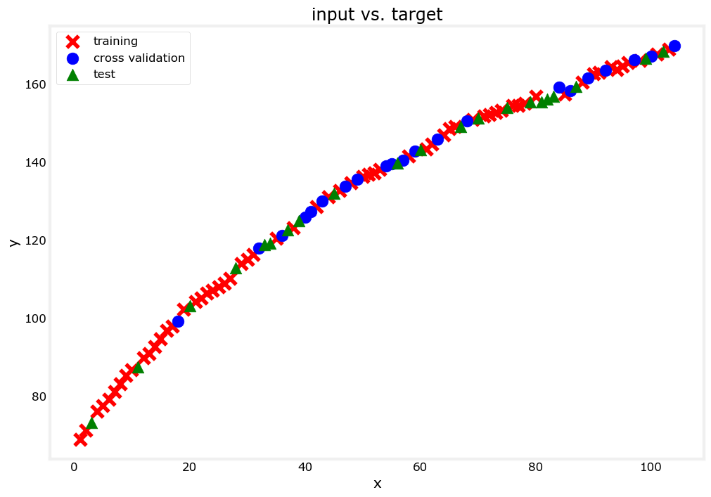
where = w

= input feature (i) of the cross-validation set ;

= cross-validation (i) value and

mcv = number of cross-validation examples.

The objective of *Cross-Validation Formula* *(2.1)* is to ***minimize JCV(w,b)*** and with Scikit-learn Python package, which provides a *train\_test\_split* function, our dataset was splatted into 60% *training set*, 20% *cross-validation set* and 20% *test set* of entire dataset, that can be viewed in *Fig. 2.2.2*:



*Fig. 2.2.2: Visualization of the Dataset used in this Work*

Before using *Cross-Validation Formula* *(2.1)*, it’s important to perform feature scaling to get better converging results. Because in this work the dataset has only positive features like “Measure Point Nr.” *(Fig. 2.1.1)*, the class initializer

***scaler\_linear = MinMaxScaler() ,***

was used in Python code with its output:

***Minimum value of the training set: 1.00***

***Maximum value of the training set: 103.00***

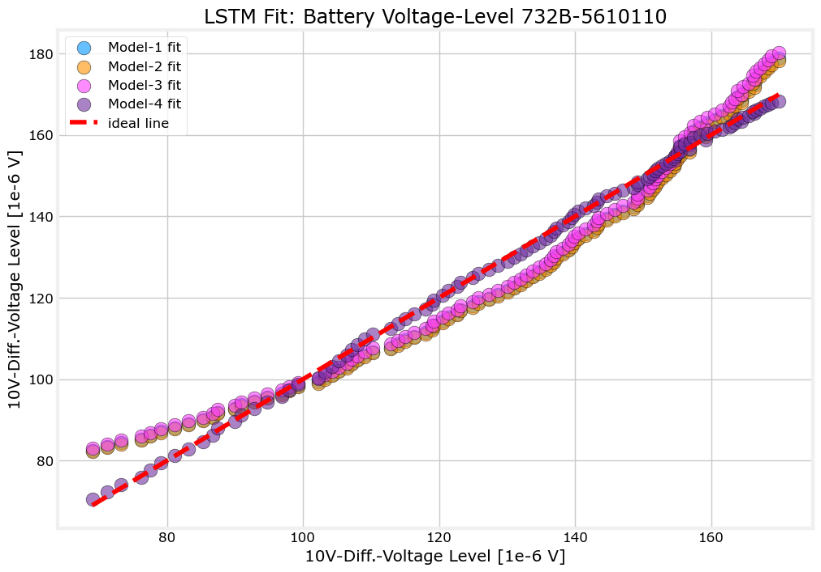
***Scaling factor of the training set: 0.01***

With these scaled values using the *Cross-Validation Formula* *(2.1)* we get these values and plot results of the four LSTM-Models for the Training and for the Cross-Validation set:

|  |
| --- |
| **Model 1: Training MSE: 16.37, CV MSE: 19.43** |
| **Model 2: Training MSE: 16.48, CV MSE: 19.98** |
| **Model 3: Training MSE: 16.84, CV MSE: 18.77** |
| **Model 4: Training MSE: 0.25, CV MSE: 0.26** |

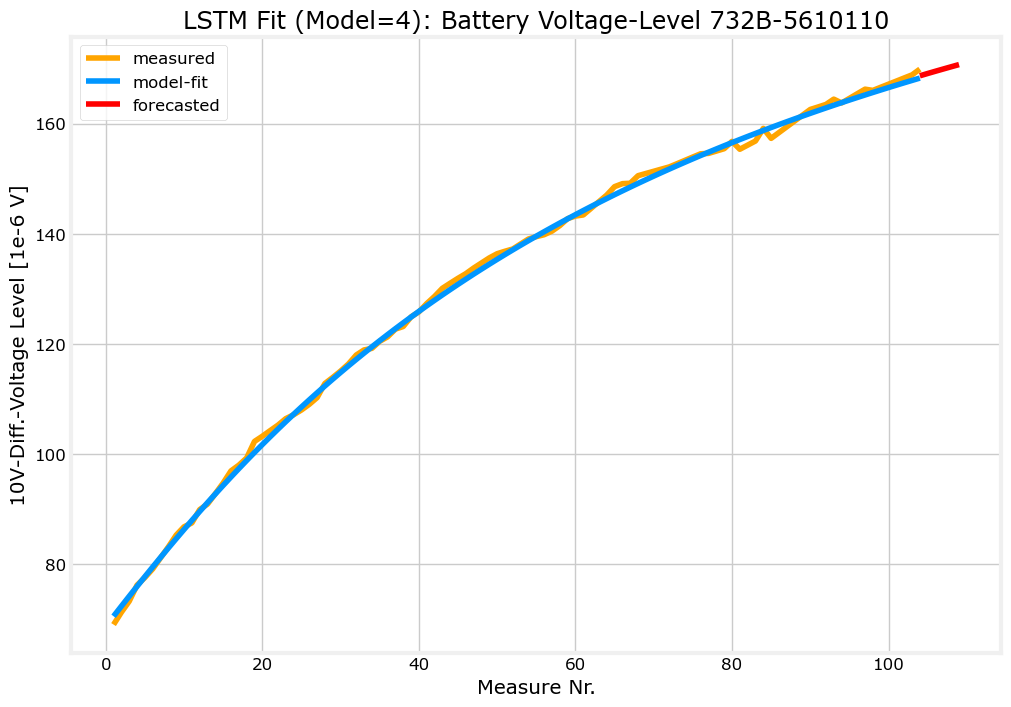
*Tab. 2.2.1: MSE Values of the four LSTM-Models*

In *Tab. 2.2.1* it’s clear that the 4th Model seems to be best fit to the orjinal dataset. In *Fig. 2.2.3* all four calculated model value sets was plotted against the orjinal dataset.



*Fig. 2.2.3: The four LSTM-Models Plot Result*

To better understand how far the 4th Model values differs from the measured values illustrates *Fig. 2.2.4.*



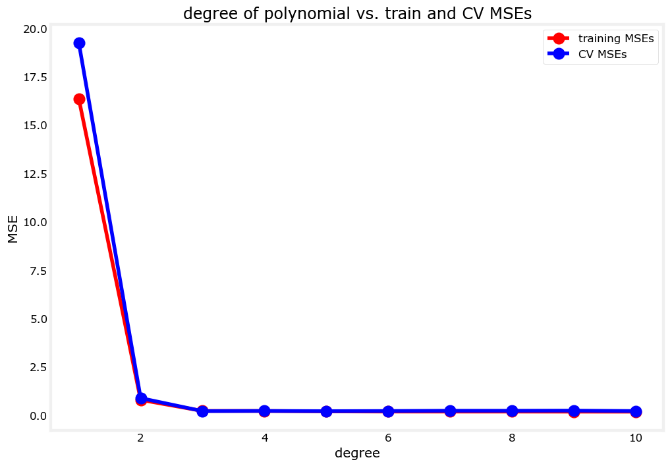
*Fig. 2.2.4: The 4th LSTM-Model Plot Result*

Furthermore, it’s obvious that this model fits very well to the original dataset *(Fig. 2.1.1)* and it can be used certainly for forecasting some steps.

In the forecasting process six future steps were created and transformed to the trained model with the same error calculation methodology. How the calculation of CV MSE value was be performed that will be discussed in detailed in the next section.

**2.3 Polynomial and XGBoost Regression**

Polynomial Regression of degree 1 to 10 was performed to find the optimum fitted voltage-time data. Like in the LSTM model creation, where a loop over different models calculates the lowest Cross-Validation MSE value, the different degrees of Polynomial Regressionwere performed to evaluate the lowest Cross-Validation error of the Cross-Validation set with the same CV MSE calculation method performed in Section 2.2 [Ref. 2]. The CV plot result of Polynomial Regression over until 10 degrees can be analyzed in *Fig. 2.3.1*:



*Fig. 2.3.1: The CV MSE Plot Result of the Polynomial Regression over 10 Degrees*

The lowest Cross-Validation MSE value was found in the 5th degree of the Polynomial Regression model and was calculated with the *Cross-Validation Formula* *(2.1)* performed like in Section 2.2 within a Python code. The code output is shown here:

*“Lowest CV MSE=0.216405 is found in the model with degree=5”*

The main part of this CV MSE calculation can be summarized like this [Ref. 2]:

1. ***Instantiate the class to make polynomial features***
2. ***Compute the number of features and transform the training set*** *(Fig. 2.2.2)*
3. ***Instantiate the class***
4. ***Compute the mean and standard deviation of the training set then transform it***
5. ***Initialize the model***
6. ***Train the model***
7. ***Compute the training MSE***
8. ***Add the polynomial features to the cross-validation set***
9. ***Scale the cross-validation set using the mean and standard deviation of the training set***
10. ***Compute the cross-validation MSE***

In Linear Regression, the “Gradient Descent” method is a commonly used as a base algorithm [Ref. 2]. In Decision Trees algorithms, several trees are trained with boosting methods, parameters such as the learning rate, which is the size of the step on the Gradient Descent method that the **XGBoost** uses internally to minimize the error on each train step. In the Results section the calculated values and the plot result of this model can be also analyzed.

**2.4 ARIMA Model**

An ARIMA model was fitted to the dataset *(Fig. 2.1.1)* using the statsmodels package in Python. Optimal parameters (p=0, d=2, q=2) were determined using AIC minimization and the **auto\_arima** tool. The model was trained and performed for ‘automatically difference and selecting best model’. The results of this model can be examined here:

**Original Series ADF Test:**

**ADF Statistic: -4.3808320332138955**

**p-value: 0.00032089638466132513**

**Result: Stationary**

**Running Auto-ARIMA...**

**Performing stepwise search to minimize AIC**

**ARIMA(2,2,2)(0,0,0)[0] intercept : AIC=197.202, Time=0.30 sec**

**ARIMA(0,2,0)(0,0,0)[0] intercept : AIC=275.106, Time=0.10 sec**

**ARIMA(1,2,0)(0,0,0)[0] intercept : AIC=234.884, Time=0.12 sec**

**ARIMA(0,2,1)(0,0,0)[0] intercept : AIC=196.012, Time=0.20 sec**

**ARIMA(0,2,0)(0,0,0)[0] : AIC=273.123, Time=0.01 sec**

**ARIMA(1,2,1)(0,0,0)[0] intercept : AIC=193.840, Time=0.16 sec**

**ARIMA(2,2,1)(0,0,0)[0] intercept : AIC=inf, Time=0.42 sec**

**ARIMA(1,2,2)(0,0,0)[0] intercept : AIC=195.711, Time=0.43 sec**

**ARIMA(0,2,2)(0,0,0)[0] intercept : AIC=193.266, Time=0.17 sec**

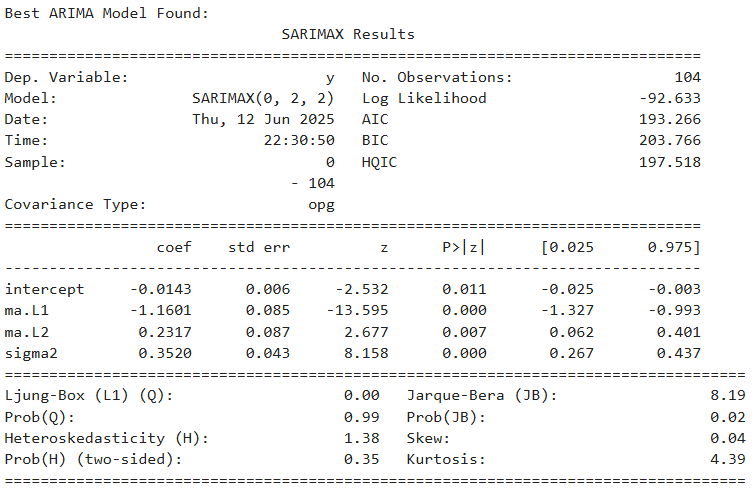
**ARIMA(0,2,3)(0,0,0)[0] intercept : AIC=195.203, Time=0.31 sec**

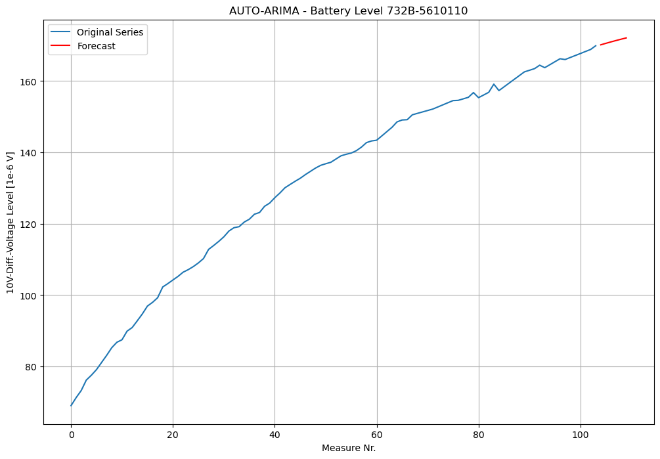
**ARIMA(1,2,3)(0,0,0)[0] intercept : AIC=196.692, Time=0.37 sec**

**ARIMA(0,2,2)(0,0,0)[0] : AIC=195.858, Time=0.04 sec**

**Best model: ARIMA(0,2,2)(0,0,0)[0] intercept**

**Total fit time: 2.649 seconds**

****

****

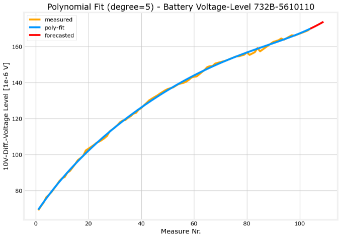
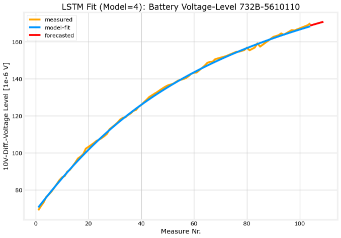
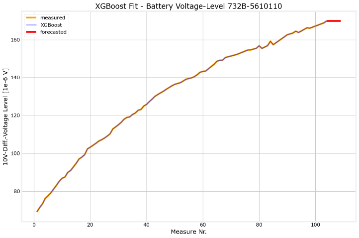
*Fig. 2.4.1: Calculation and Visual Results of the performed ARIMA-Model*

**3. Results**

The calculated MSE values of the analyzed models performed in this work show that **XGBoost** Model most accurately captures the temporal dynamics and nonlinear decay of the voltage trend:

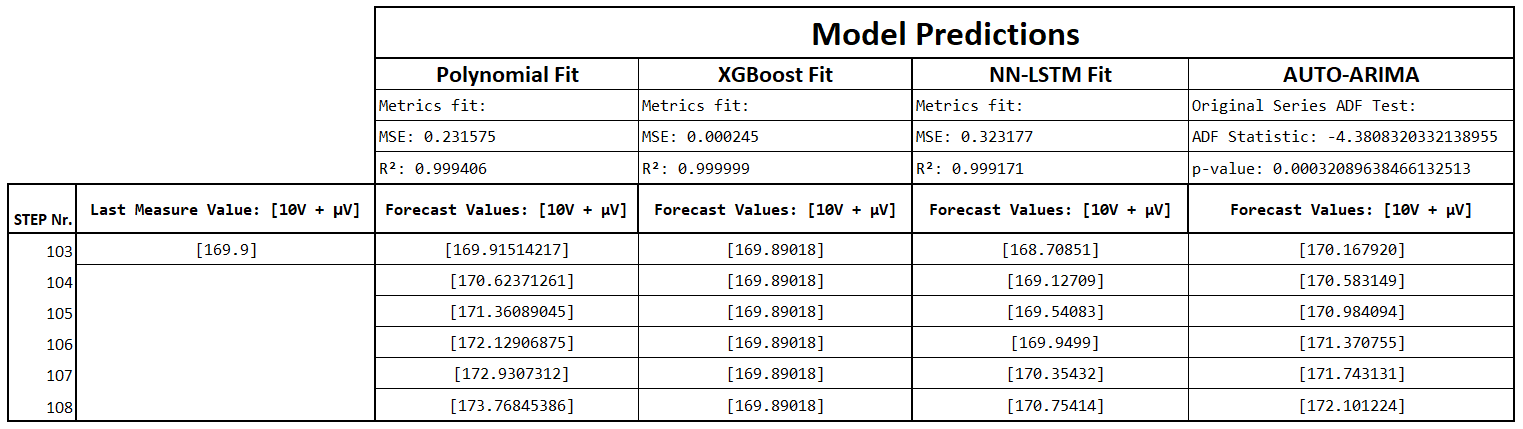
|  |  |  |
| --- | --- | --- |
| **Model** | **MSE** | **R2** |
| Polynomial (deg 5) | 0.231575 | 0.999406 |
| LSTM (4th model) | 0.323177 | 0.999171 |
| XGBoost | 0.000245 | 0.999999 |

But the very low MSE value result of **XGBoost** according to the other two models indicates that the model is overfitted to the original dataset, and the detailed value inspection shows its very poor forecasting ability *(Tab. 3.1)*. Visual plots of the model predictions performed in this work are be shown in *Fig. 3.1*:

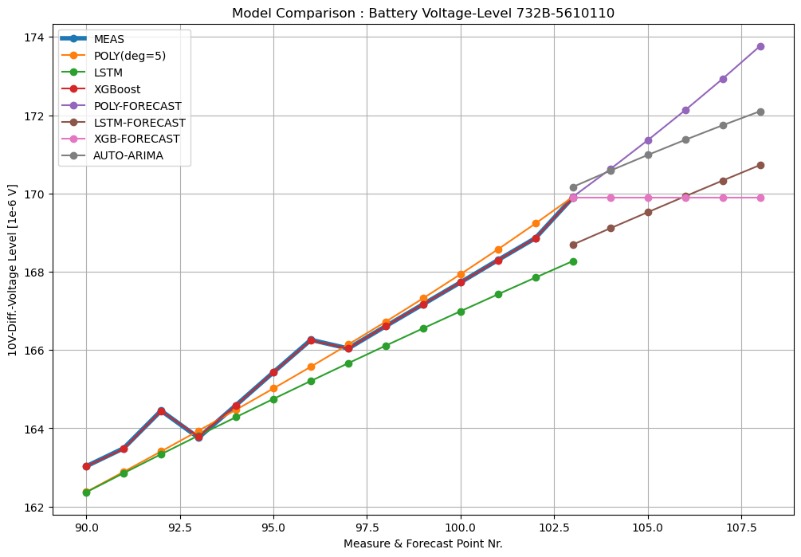
*Fig. 3.1: Comparison of Polynomial Regression – LSTM – XGBoost Model*

In *Fig. 3.1* visual plots make clear that the Polynomial Regression of degree 5 and the 4th LSTM model (*Fig. 2.2.1*) for the dataset in *Fig. 2.1.1* are very well model predictions without overfitting. This is the main issue of this work, to predict the measured values well and to forecast a plausible voltage value that would be measured in future at the next time point (1 year later). These values can be examined in *Tab. 3.1*.



*Tab. 3.1: Detailed Comparison of all Models performed in this Work*

To analyze the predicted values of the four models more in detail, the compacted detailed plot in *Fig. 3.2* can give a view of that, where only the last 13 measure points and the next 6 forecasted points are shown. In this plot example *(Fig. 3.2)* the forecasted values of the ARIMA Model from Section 2.4 were also be added.

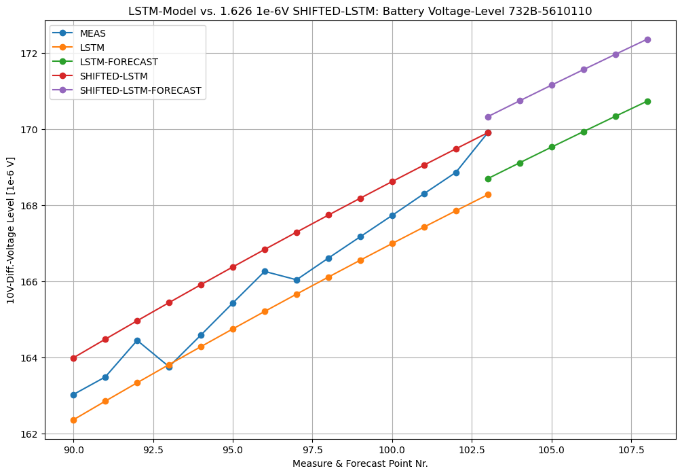


*Fig. 3.2: Detailed Plot of all Model Values performed in this Work*

In *Fig. 3.2* it’s obvious that the Polynomial Regression model is giving the most plausible predictions, but focusing on forecasting, the performed LSTM-Model gives an idea for improvement. This issue will be discussed in the next section.

**4. Discussion**

After visual examination of the plotted results, a proposal fit method for the 4th LSTM-Model was made to reach more plausible forecasting points. The main idea was to shift the predicted 4th LSTM-Model line to the last measured value and with the new shifted model line to compare its new forecasted values with the other models’ values. First, the 4th LSTM-Model plot line and its new shifted model line are shown in a detailed view *(Fig. 4.1)*.



shift\_lstm=1.626 1e-6 V

*Fig. 4.1: Detailed Plot of the 4th LSTM-Model and the proposal LSTM-Model Fit Line*

The calculation of the proposal LSTM model fit line was performed in Python code like this:

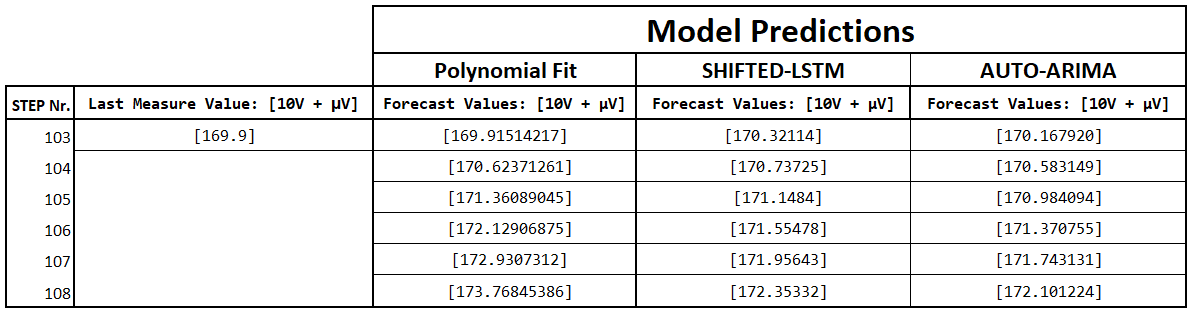
***# Calculate the shift gap between measured and LSTM value***

***shift\_lstm = y\_meas[-1] - y\_lstm[-1] # shift\_lstm = 1.626 1e-6 V***

***new\_lstm\_value = y\_lstm + shift\_lstm***

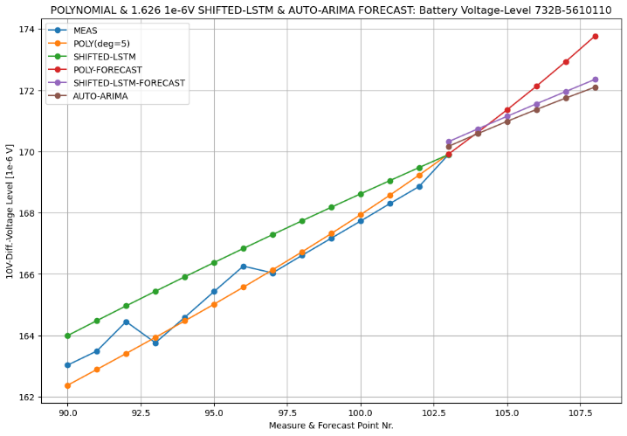
***new\_lstm\_f = y\_lstm\_f + shift\_lstm***

The new calculated values of the introduced SHIFTED-LSTM Model fit line can be analyzed in Tab. 4.1, where a detailed value forecasting comparison was made between three models.



*Tab. 4.1: Forecast Values Comparison between 3 Models performed in this Work*

It’s interesting to see how near are the values of the SHIFTED-LSTM and AUTO-ARIMA model values between each other. This can be obviously viewed in *Fig. 4.2*.



*Fig. 4.2: Detailed Plot of the 3 Model Values performed in this Work*

The plot result in *Fig. 4.2* highlights that the introduced SHIFTED-LSTM-Model behaves like the AUTO-ARIMA-Model for forecasting points. This issue seems interesting and is emphasizing the advantages of LSTM networks in handling time series data, like the dataset used in this work *(Fig. 2.1.1)*. Only the tough thing is, to develop a LSTM model. In this work the performed 4th LSTM-Model *(Fig. 2.2.1)* has reached a satisfied CV-MSE result after many trials of CV MSE calculations *[Tab. 3.1]*.

While the Polynomial Regression Model (degree = 5) performed in this work with the 2nd best MSE value and R2 score provided a well model fit *[Tab. 3.1]*, it seems less accurate in longer-term forecasting that can be estimate in *Fig. 4.2*.

Although with its best MSE value and R2 score the XGBoost Model is not able to forecast plausible values that can be viewed in *Fig. 3.2*. It’s very low MSE value indicates that the model overfits the original data *[Tab. 3.1]*.

ARIMA has a very fast calculation and predict plausible values in longer-term forecasting, that can be viewed in *Fig. 4.2*. But an unresolved issue in this work is, how the original dataset was modelled by this model in detail and therefore, the AUTO-ARIMA function was practically used for this model. That’s not the focus of this work. More detailed inspections for ARIMA models can be analyzed in [Ref. 4].

The small size of the dataset posed a challenge, but careful preprocessing and model tuning mitigated overfitting, accept the XGBoost-Model performed in this work. It was a particular interest to use a XGBoost model and to compare it with the other models performed in this work. Because of the very high overfitting score *(R2 ≈ 1.0, Tab. 3.1)* the performed XGBoost-Model seems not useful for time series data sets.

The focus of this work is to find a well-fitting deep learning model of measured battery voltage levels *(Fig. 2.1.1)* and to forecast legitimate voltage values in future time. Therefore, it’s a matter of great curiosity in this work what real voltage values will be measured in the future. Only then it will be clearer, which model will be fit optimal.

Finally, combining deep learning with traditional methods allows greater flexibility and provides useful baselines. Future work could incorporate additional battery parameters (e.g., temperature, load) to improve model accuracy.

**5. Conclusion**

This work demonstrates the advantages of LSTM models. With its flexible model parameters, it seems always to reach acceptable CV result values and plausible predictions. Because of this, it was possible to shift the performed LSTM-Model to reach a very close trend line to the ARIMA-Model performed in this work. Therefore, LSTM models significantly outperform traditional regression methods in predicting battery voltage over time. Despite the limited dataset, LSTM and also Polynomial Regression with a specific degree, here in this work the polynomial degree was determined as five, captured the temporal degradation patterns more effectively. In focusing of forecasting ARIMA and also LSTM models bring hopefully results. The findings support the application of deep learning methods in battery health management and predictive maintenance systems.

**References**

1. Arifoviç, M.: *Digital Twin Project*. TUBITAK UME, Voltage Laboratory (2024)
2. Ng, A.: *DeepLearning.AI. Machine Learning*. Coursera (2025)
3. Hochreiter, S., & Schmidhuber, J.: *Long Short-Term Memory*. *Neural Computation*, 9(8), 1735-1780. (1997)
4. Hyndman, R.J., & Athanasopoulos, G.: *Forecasting: Principles and Practice*. OTexts. (2018)
5. Géron, A.: *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly Media. (2019)