Computing and Software for Big Science (2022) 6:12
https://doi.org/10.1007/s41781-022-00084-4

REVIEW

Modelling Large-Scale Scientific Data Transfers

Joaquin Bogado[1,2] · Mario Lassnig[3] · Fernando Monticelli[2] · Javier Díaz[1]

Received: 5 April 2021 / Accepted: 5 May 2022 / Published online: 6 July 2022

Abstract
This work focuses on the study of a recently published dataset (Bogado et al. in ATLAS Rucio transfers dataset. Zenodo, 2020.) with data that allow us to reconstruct the lifetime of file transfers in the contexts of the Worldwide LHC Computing Grid (WLCG). Several models for Rule Time To Complete (TTC) prediction are presented and evaluated. The dataset source is Rucio, an open-source software framework that provides scientific collaborations with the functionality to organize, manage, and access their data at scale. The rich amount of data gathered about the transfers and rules, presents a unique opportunity to better understand the complex mechanisms involved in file transfers across the WLCG.

Keywords Data transfer analysis · Distributed computing modelling · Performance metrics

Introduction

There have been efforts to model data transfers since Rucio commissioning at the end of 2014 [1]. The work cited in [2] focuses on Transfers TTC predictions. The work cited in [3] focuses on the prediction of the network throughput. The work cited in [4] focuses on the prediction of the length of the queues of the system, with emphasis on the importance of network throughput. However, prior studies have failed to delve into Rucio's replication rules modeling and Rules TTC prediction, and to assess how accurate a model that can predict the Network Time of a transfer can also predict the Transfer or Rule TTC.

Rucio handles transfers between sites but also handles the deletion requests to comply with the data retention policy. Both transfers and deletion requests are stored in REQUESTS and REQUESTS_HISTORY tables: theREQUESTS table stores the current requests with no final state, while the REQUESTS_HISTORY table works as an archive of requests that will not be updated anymore, that is the requests with final state. Deletions requests do not affect RSE transfer performance, and so can be ignored. Only transfer requests were taken into account for this work.

There are several instances of the transfer tool on the grid. These instances work at WLCG level and serve transfers from several VOs and not only ATLAS specific transfers. The Rucio Database does not contain information about the FTS transfer tool other than the instance that is used for each transfer request. Information about FTS queues state, scheduling and retries, number of nodes, and configuration are hidden from Rucio. Transfers in an FTS server from other VOs are also hidden from Rucio.

The main hypothesis is that the load in FTS queues has a noticeable impact on the difference of the submission time and starting time of a transfer. The more transfers are queued at FTS the more time will elapse between the submission time and starting time of a transfer. Ergo, a model that can predict the Network Time of a transfer with 100% accuracy will not necessarily predict accurately the Transfer TTC, as the Network Time represents a small fraction of the total time.

The studied dataset has been made publicly available [1].

Metric Selection

Standard metrics for regression tests include root mean squared error (RMSE) and mean squared error (MSE) [5], mean absolute error (MAE) and median absolute error (MedAE), mean squared logarithmic error (MSLE) [5] and

✉ Joaquin Bogado
jbogado@linti.unlp.edu.ar

1 LINTI, Facultad de Informática, La Plata, Argentina

2 IFLP, UNLP, CONICET, La Plata, Argentina

3 European Organization for Nuclear Research (CERN), Geneva, Switzerland

root mean squared logarithmic error (RMSLE), explained variance score, R^2 score, mean Tweedie deviance, mean absolute percentage error (MAPE) [5], relative error (RE) and related metrics, and the Fraction of Good Predictions (FoGP) [6]

The mean squared error measures the mean of the squared difference between the vectors y and $\hat{y}$, according to Eq. 1:

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2. \tag{1}$$

As the differences are squared, this metric penalizes more the big differences and as it is a mean value, is sensitive to outliers. The RMSE version is the squared root of the MSE, making its units comparable with the units of y and $\hat{y}$, so if y and $\hat{y}$ are in seconds, the RMSE can be interpreted in seconds too. When several models are to be compared or when the values of y have great variance, MSE and RMSE are not particularly useful. Two models α and β will be considered with comparable performance if $RMSE(y, \hat{y}_\alpha)$ and $RMSE(y, \hat{y}_\beta)$ are in the same order of magnitude, but always the model with smaller RMSE will be preferred.

The Mean Absolute Error and the Median Absolute Error are the mean and median of the absolute value of the difference between y and $\hat{y}$, respectively. The MAE is calculated using Eq. 2, whereas MedAE is calculated using Eq. 3:

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i|, \tag{2}$$

$$MedAE(y, \hat{y}) = median(|y_i - \hat{y}_i|), i = 0, 1, 2, \ldots, n-1. \tag{3}$$

MAE and MedAE are easier to interpret than MSE and RMSE, but MedAE is preferred for its robustness to outliers in y and $\hat{y}$. However, the four metrics are sensitive to the scale of y. This means that when the same model is evaluated, using two different set of observations y and y', the metric will be different for the same model and will depend on the distribution of y and y'. If y present outliers and y' does not, the metric for the same model will be worse regardless of the performance of the model. This is not a problem when two models compared against the same y, but does not give an idea of the *goodness* of the models in general.

The mean squared logarithmic error is a metric robust against outliers and not sensitive to scale of y. It is defined by Eq. 4 as the mean of the squared differences between natural logarithms of $1 + y$ and the natural logarithm of $1 + \hat{y}$. The root mean squared logarithmic error is the squared root of MSLE:

$$MSLE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(ln(1 + (y_i) - ln(1 + (\hat{y}_i))^2. \tag{4}$$

Hidden in the definition lies the problem that this metric tends to penalize the negative errors more than the positive ones, and thus will favor a model that overestimates the predictions over one that underestimates them. But the metric is difficult to interpret and the results do not give a good idea of the goodness of a model.

The explained variance score and the R^2 score are two metrics related to each other. The differences are subtle but important. The explained variance is defined in Eq. 5. It can be interpreted as the proportion of the variance of y that is explained by the model though the predictions $\hat{y}$:

$$EV\_score(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}((y_i - \hat{y}_i) - \overline{(y_i - \hat{y}_i)})^2}{\sum_{i=0}^{n-1}(y_i - \overline{y_i})^2}. \tag{5}$$

The R^2 score, also known as coefficient of determination, is defined as in Eq. 6:

$$R^2\_score(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \overline{y})^2}. \tag{6}$$

In Eqs. 5 and 6 it is possible to spot the difference at a glance. Both results are equal if $\overline{y_i - \hat{y}_i}$ is zero, meaning the R^2 score does not account for biased models as explained variance does. This also makes the R^2 slightly more sensitive to the scale of y.

Interpretation of both metrics is not clear at a glance, but are implied directly from the equations. In both cases, if the prediction of the model is perfect, then $y - \hat{y} = 0$, and then both scores are equal to 1. This is the best score a model can achieve. By definition, a model that makes a prediction using the mean $\overline{y}$ has a score of 0, so any model with a score bigger than 0 will be better than the naive model. But then the predictions can be infinitely far away from the observed value. If that is the case both scores are negative.

Mean Absolute Percentage Error has the advantage of being easy to interpret. MAPE is based on the Relative Error, that is the absolute value of the difference between target and the prediction relative to the target, as shown in Eq. 7, and thus, is the error in the prediction relative to the observed value:

$$RE(y_i, \hat{y}_i) = \frac{|y_i - \hat{y}_i|}{y_i} \tag{7}$$

The MAPE is the mean of the RE expressed as a percentage, as in Eq. 8. But, as a mean value, is sensitive to outliers in the relative errors. It is possible to overcome this by taking the median instead of the mean in Eq. 8. Mean Absolute

Percentage Error and Median Absolute Percentage Error both diverge when y values are very close to zero:

$$MAPE(y, \hat{y}) = 100\frac{1}{n}\sum_{i=0}^{n-1}\frac{|y_i - \hat{y}_i|}{y_i} \tag{8}$$

MAPE penalizes more the positive forecast values than the negative ones [7]. sMAPE or Symetric Mean Absolute Percentage Error and sMedAPE try to address this issue but still can suffer from the divergence problem due to the sum $y_+\hat{y}_i$ being small.

In [5], the MASE or Mean Absolute Scaled Error is introduced to circumvent the mentioned problems. These derived metrics overshadow the straightforward interpretation of MAPE. Moreover, the models to evaluate in this work make predictions over positive integer targets, as both the Rule TTC and Transfer TTC are measured in seconds.

The metric selected to compare models in this work is described in [6] (p.16) as percentage of predictions with less than X percent RE. We call this metric Fraction of Good Predictions (FoGP), expressed as a number between 0 and 1, in which X is the threshold of relative error below which a prediction is considered good.

Formally, with the trivial function g defined as in Eq. 9, FoGP is defined in Eq. 10:

$$g(y_i, \hat{y}_i, \tau) = \begin{cases} 1 & \text{if } RE(y_i, \hat{y}_i) \leq \tau; \\ 0 & \text{else.} \end{cases} \tag{9}$$

$$FoGP(y, \hat{y}, \tau) = \frac{1}{n}\sum_{i=0}^{n-1} g(y_i, \hat{y}_i, \tau) \tag{10}$$

As an example, assume that a certain model made a prediction for y. We calculate the FoGP with threshold 0.05 and we obtain the 0.5 as results. Formally, this can be expressed as $FoGP(y, \hat{y}, 0.05) = 0.5$. This means that 50% of the predictions in $\hat{y}$ are less than 5% from their real values.

This metric is easy to interpret, robust to outliers both in y and $\hat{y}$, and can be easily and efficiently implemented. Thus, the models studied in this work are evaluated using this metric.

Models

Models α and $\alpha\prime$

The number of transfers per rule varies from rule to rule, but there are notable peaks in some numbers, most notably, rules with exactly 20 transfers. These rules are generated by an automated replication process by the experiment (Fig. 1).

The set of transfers going through the FTS BNL instance contains over 1.8M different rules. The mean

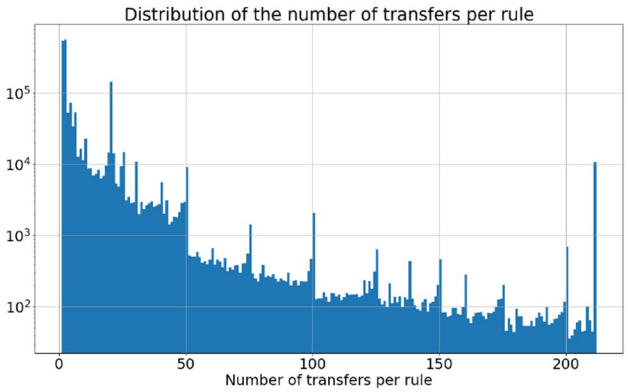

Fig. 1 Distribution of the number of transfers per rule. Rules with more than 210 transfers are counted in the last bin. The maximum number of transfers per rule is 205951. Notice the peaks in 20, 25, 30, 40, 50, 100, and 200. Other notable peaks are 1, 2, 4, and 6

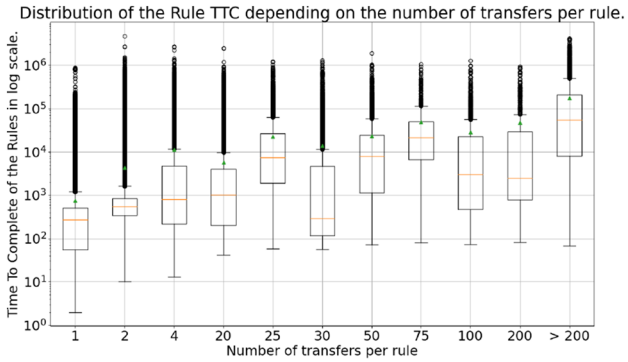

Fig. 2 Rule TTC given the number of transfers in the rule

rule TTC is 3.1 h but the mean varies within two orders of magnitude depending on the number of transfers per rule, as shown in Fig. 2.

A baseline model was developed using the dataset described in the previous section. The prediction for the Rule TTC is based on the `created` timestamp of the first created transfer, the `ended` timestamp of the first ended transfer and the total number of transfers to be created and transferred to fulfill the rule.

Formally, the method consists of a regression analysis using ordinary least squares to fit a 1-degree polynomial, where the independent variable is the completions percentage of the rule, and the dependent variable is the time at which the percentage of completion is reached. Figure 3 illustrates the application of the method to an observed rule. The rule has 20 transfers and the prediction is made after the first transfer ends, hence only using two points to fit the polynomial:

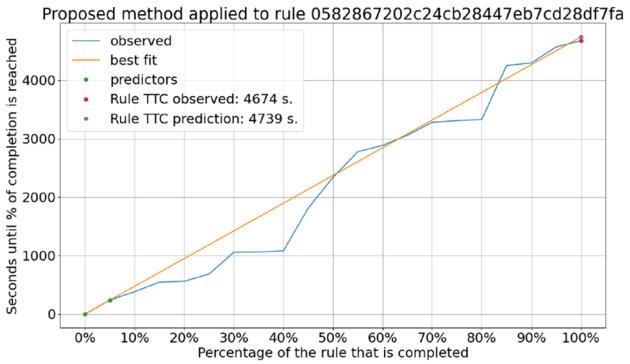

Fig. 3 Regression analysis applied to a rule to predict its TTC. Prediction is made after the first transfer ends

$$\%\ \text{Completed} = \frac{ended\_xfers \times 100}{total\_xfers}. \tag{11}$$

More than two points can be used in the regression analysis. But the more points that are used, the more transfers need to finish before the prediction can be made, ergo the more the rule needs to progress, and the less useful the prediction will be, rendering this method useless for scheduling purposes. However, these models can still be applied, for example, to give feedback to users about the time their transfers will be finished.

We define the family of models α_k as in Eq. 12, where $ax + b$ is the polynomial that results from the fit of the predictor vector X_k, using the Ordinary Least Squares method. We call the predictor X_k to the vector of points $((\chi_{1j}, \chi_{2j}))$, where χ_{1j} is the component that represents the percentage of the rule that is completed, and χ_{2j} is the time elapsed in seconds until that χ_{1j} is reached. The sub-index k is the number of points used to make the fit, so the range is $k = 2, 3, 4, \ldots, n$, where n is the number of transfers in the rule. Thus, j sub-index range from $1, \ldots, k$ (Fig. 4):

$$\alpha_k(X_k) = ax + b. \tag{12}$$

A more detailed analysis of the progress of the rules over time determined that only a fraction of the Rules TTC are co-linear with the first points of the progress of the rule, including the origin point.

From this observation, model $\alpha\prime_k$ was created and tested. Every member of the $\alpha\prime_k$ family is equal to its relative α_{k+1}, but the origin point, where the rule is created is removed from the regression. Then, $\alpha\prime_2$ is equal to α_3, and the ending times of the first two transfers are used to make the linear regression but not the origin point. An important consequence is that the models $\alpha\prime_k$ and α_{k+1} can make a prediction in the same stage of the progress of the rule, that is, when the transfer k is finished (Table 1).

Models β and γ

One important caveat with the models α and $\alpha\prime$ is that they will not be able to make any prediction for the Rule TTC before at least two transfers of the rule have finished. One approach is to use metrics such as the mean or the median of the Rule TTC of already finished rules as the predictor of Rule TTC of the newly created rules. Figure 5 shows an

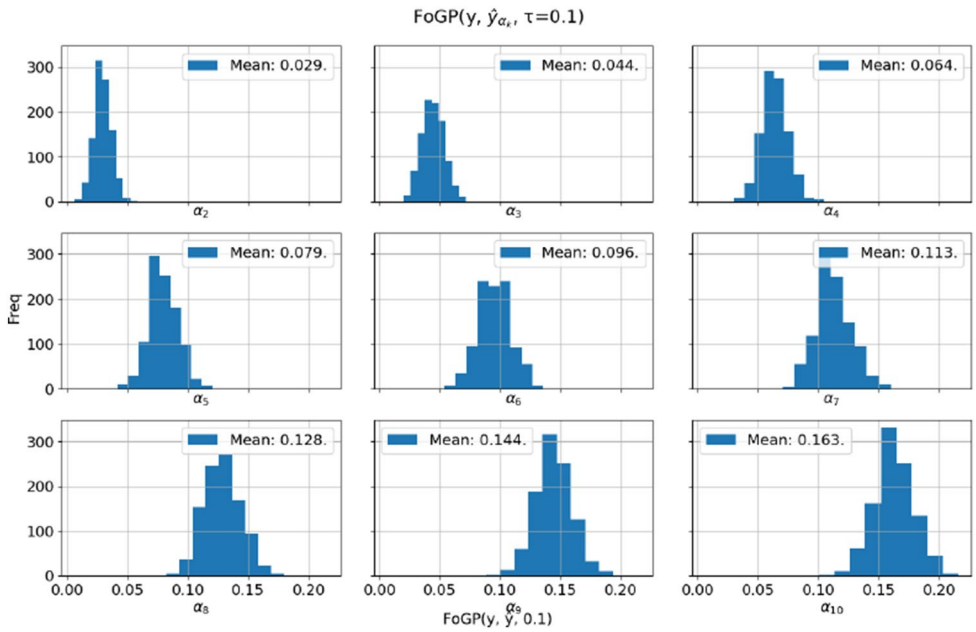

Fig. 4 Results for the experiment showing the distribution of the $FoGP(y, \hat{y}_{\alpha_k}, \tau = 0.1)$ for the first 9 members of the α_k family. The bigger the k, the more points are included in the regression, the more accurate the results, and the later the prediction. For α_2, at least one transfer needs to finish before a prediction can be made, while for α_{10}, at least 9 transfers need to finish to make a prediction

Table 1 $FoGP(y, \hat{y}_{\alpha\prime_k}, \tau)$ vs. $FoGP(y, \hat{y}_{\alpha_k}, \tau)$ comparison. Different thresholds τ are calculated for different $\alpha\prime_k$ and α_k models. Comparison criteria is based on number of finished transfers in the rule the model needs to make a prediction. $\alpha\prime_k$ and α_{k+1} are two models that need at least k transfers to finish before a prediction can be made. Top left corner shows that while for model α_3, only 4.4% of the predictions lay within 10% of its real value, for model $\alpha\prime_2$, 8.8% of the predictions lay in the same range. This represents an improvement of 95.4% of model $\alpha\prime_2$ with respect to α_3

	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
$\alpha\prime_2/\alpha_3$	0.086/0.044	0.425/0.224	0.801/0.398
$\alpha\prime_3/\alpha_4$	0.115/0.064	0.522/0.308	0.858/0.530
$\alpha\prime_4/\alpha_5$	0.134/0.079	0.580/0.384	0.892/0.639
$\alpha\prime_5/\alpha_6$	0.150/0.096	0.623/0.451	0.916/0.727
$\alpha\prime_6/\alpha_7$	0.166/0.113	0.662/0.513	0.932/0.799
$\alpha\prime_7/\alpha_8$	0.184/0.128	0.696/0.574	0.944/0.858
$\alpha\prime_8/\alpha_9$	0.202/0.144	0.730/0.631	0.953/0.901
$\alpha\prime_9/\alpha_1$	0.219/0.163	0.758/0.687	0.959/0.922

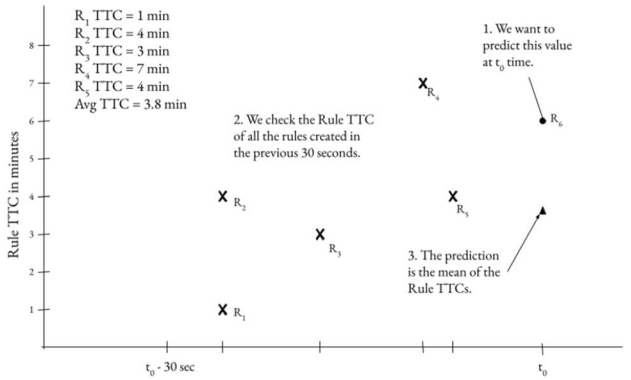

Fig. 5 Rule TTC of rule R_6 prediction, based on the mean of the Rule TTC of those rules created 30 s before rule R_6 creation

example of how this method could work. Assume we want to predict the Rule TTC of the rule R_6 created at time t_0. A look back window can be defined since t_0. Let's consider a window of 30 s and the mean of the Rule TTCs of those rules created between t_0 and $t_0 - 30$ s. Rules R_1 to R_5 were created over the last 30 s, and the mean TTC is 3.8 min, so that will be the prediction for the Rule TTC of R_6. We called this model β. The best window length was found through an optimization process described later.

However, some of the Rules R_1 to R_5 may not be completed at t_0, so their TTCs will be unknown and not available to make any prediction, and thus the β model cannot be implemented to be used in real time. The idea of this work is to use Time Series Analysis techniques to make a model that can predict the mean to use it as predictor of the rules created at t_0. We call this the model γ.

Formally, the $\beta_\mu(t_0, \rho)$ model family is defined as in Eq. 13. Here, y_{R_i} are the real Rule TTC of those rules created in the left-closed interval $[t_0 - \rho, t_0)$, that is, the rules in the dataset with $min\_created < t_0$ and $min\_created \geq (t_0 - \rho)$. The parameter ρ can be interpreted as the size of the rolling window, usually measured in seconds. The aggregation function μ will be a function that returns a number that represents a summary of the information contained in the $\{y_{R_i}\}$ set. The functions tested are *min*(), which returns the minimum element of the set, *max*(), which returns the maximum element of the set, *median*(), which calculates the arithmetic median, and *mean*, which returns the mean of the values of the set:

$$\beta_\mu(t_0, \rho) = \mu(\{y_{R_1}, y_{R_2}, ...\}) \tag{13}$$

Using this notation, the example in Fig. 5 can be annotated as $\beta_{\text{mean}}(\rho = 30)$.

Figure 6 summarizes the result of the experiment. The most accurate result was obtained by the β_{median} model with ρ values between 20 and 30 s, through an $FoGP(y, \hat{y}_{\beta_\mu(\rho)}, \tau = 0.1) = 0.22$, meaning around 22% of the predictions will present less than 10% of relative error. Other βs present lower FoGP, and thus, lower predictive power. A new scanning of ρ in the interval [20, 30] shows no significant improvement in the FoGP. This also shows that the values of the μ aggregation function had some correlation with time and that these values could hold important information about the Rule TTC of near future rules.

If the β_μ model is implemented using only the data at real time at t_0, that is, at the time to make a prediction for a rule created at t_0, then the only TTCs available will be the ones of those rules with created and finished in the semi-open interval $[t_0 - \rho, t_0)$, that is, the rules with $min\_created \geq (t_0 - \rho)$ and $max\_ended < t_0$. Figure 7 summarizes the results of the experiment of measuring the FoGP over 300 Rule TTCs predictions, repeated 100 times. The μ functions were calculated using real time data.

The Model γ_μ

Model β_μ is implemented using real time data. It has low FoGP, as the μ aggregation function depends on real observed Rule TTC, and real time aggregation is not representative of the future Rule TTCs. However, as was demonstrated in the previous section, there is a time dependency in the aggregated values of the Rule TTCs. Model γ_μ presented here uses a forecast of the time series of the aggregation function μ to predict the Rule TTC of the new rules. Special attention was put in the *median*() and *mean*() aggregation functions, both because of their hypothetical predictive power and their good statistical properties.

Formally, the γ_μ model is defined as in Eq. 14. This equation is very similar to the β_μ equation, but in the γ model, the μ function is estimated using an auto-regressive model with λ lags of size ρ seconds. The ψ parameter represents the

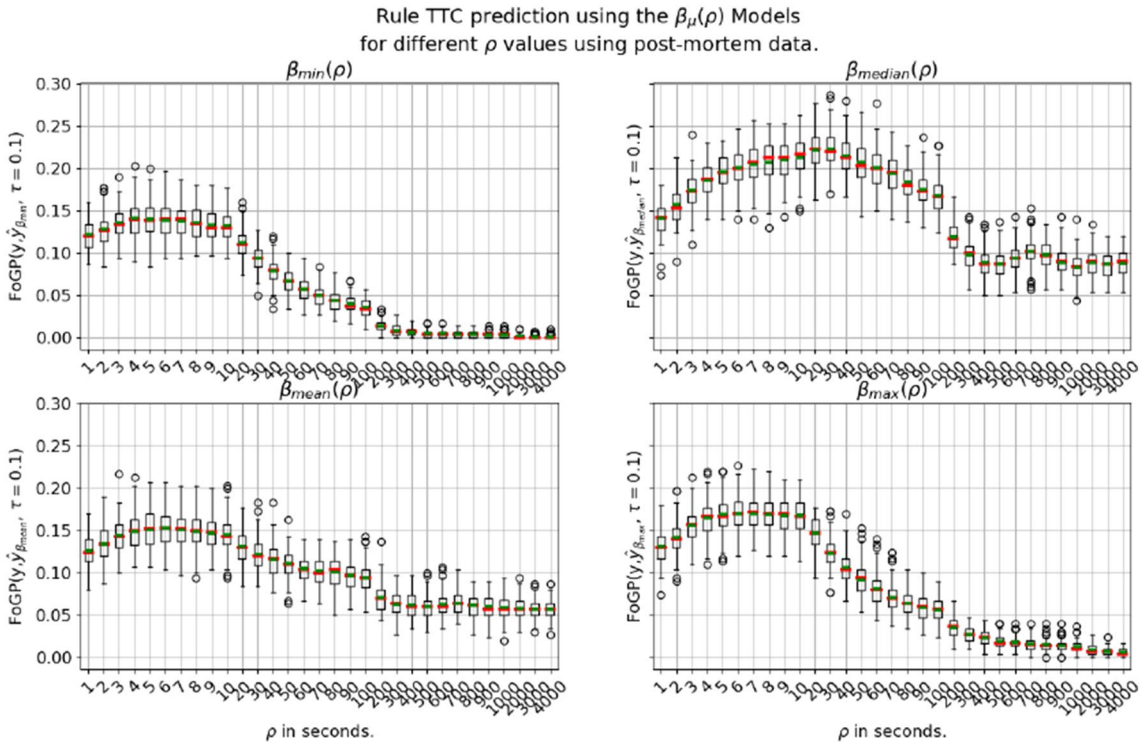

Fig. 6 Rule TTC prediction using the β_μ for μ one of the min(), median(), mean(), or max() functions. The ρ parameter was selected to explore space and test the predictive power of each model. The FoGP metric is calculated for the same 300 random rules for every window size ρ. The experiment is repeated 100 times. The red lines are the median FoGP of the experiment and the green lines are the mean FoGP, for every window size. Notice that the real min/median/mean/max Rule TTC was used to make a prediction and that this value usually is not available in real time, i.e., the rules created during the previous seconds usually take several minutes to complete

look back, or how many lags are used to fit the model. The ω parameter represents the look ahead, or how many lags in the future the model will predict:

$$\gamma_\mu(t_0, \rho, \lambda, \psi, \omega) = \hat{\mu} \tag{14}$$

Here, $\hat{\mu}$ is the estimation of the function $\mu(y_{R_i})$ through the use of an auto-regressive model. As in the β_μ model, the set y_{R_i} is the Rule TTC of those rules created in the left-closed interval $[t_0 - \rho, t_0)$ but in this case also the ones that have finished before t_0.

The algorithm proceeds as follows. First, all the rules that have been created between t_0 and $t_0 - \rho\psi$ seconds and that have finished before t_0 are selected. That is, all the transfers in the Rules Dataset which satisfy the conditions $t_0 - \rho\psi \leq min\_created < t_0$ and $max\_ended < t_0$. The Rules Dataset contains the fields *min_created* and *max_created* time stamps, that represent the time when the first transfer and last transfer of the rules were created. Thus *min_created* time stamp is equal to the time creation of the rule and the minimum *min_created* is the time of creation of the first rule in the dataset. Also, the *max_ended* is the time stamp of the last transfer to finish, and thus, the finishing time of the rule. The μ function is calculated over the bins of length ρ seconds, being the value of the first bin, the μ of the Rule TTC of the rules satisfying the condition $t_0 - \rho\psi \leq min\_created < t_0 - \rho\psi + \rho$, the value of the second bin the μ of the rules satisfying the condition $t_0 - \rho\psi + \rho \leq min\_created < t_0 - \rho\psi + 2\rho$, and so on. This generates a time series $\sigma_{\hat{\mu}}$ of frequency ρ with a total of ψ/ρ samples. Notice that this time series differs from the real time series σ_μ in that the μ Rules TTCs for the lags closer to t_0 differ due to the selection filter rules described before. Once the time series is obtained, a standard auto-regressive model $AR(p)$ [8] is fitted using the first $\psi/\rho - \omega$ samples. The parameter of the auto-regressive model is $p = \lambda$, meaning the model will need λ samples to make a prediction. Model train and prediction is implemented using the `AutoReg` function from the *Python statsmodels v0.11.1* package [9]. Once the model is fitted, a forecast is made using the last $\lambda - \omega$ samples to predict the following ω lags. The $\hat{\mu}$, i.e.,the prediction of the γ_μ, will be the last value of the returned forecast.

Figure 8 shows the case for $\gamma_{\text{median}}(\rho = 30, \lambda = 30, \psi = 90, \omega = 16)$ model. In this case, the t_0 is the date 2019-06-12 22:54:31. The transfers selected to create the time series were those satisfying the conditions $t_0 - 30 * 90\,\text{s} \leq min\_created < t_0$ and $max\_ended < t_0$. The

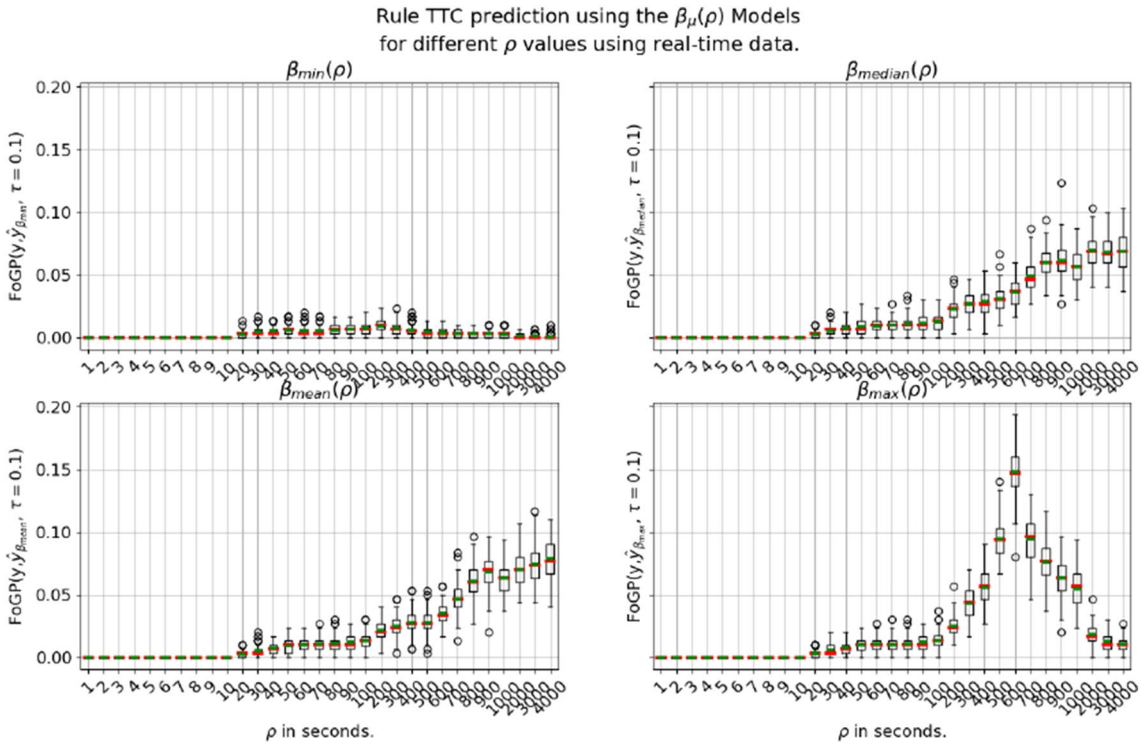

Fig. 7 Rule TTC prediction using the β_μ for μ one of the min(), median(), mean(), or max() of the real time data available at t_0. The β_μ function using the TTC of those rules that started between $t_0 - \rho$ and t_0 but also did finish before t_0. This simulates the real time data available in the system to make the prediction, as those times beyond t_0 are in the future and TTC data for rules finishing after t_0 is usually not available. The FoGP metric is calculated for the same 300 random rules for every window size ρ. The experiment is repeated 100 times. The red lines are the median FoGP of the experiment and the green lines are the mean FoGP, for every window size. Notice that if ρ is small, usually the prediction is zero. That is because the number of rules created and finished in the interval $[t_0 - \rho, t_0)$ is zero

median of the Rule TTC were calculated to get the time series using bins of 30 s. The $\sigma_{\hat{\mu}}$ time series is shown in orange while σ_μ time series values, that is the median of the Rule TTC including those of the rules that end after t_0, is plotted in blue. These time series are very similar except in the last minutes before t_0. The main reason for this discrepancy is that rules created some minutes before t_0 only finish after t_0 and are excluded, because they do not satisfy the condition $max\_ended < t_0$. This will happen in a hypothetical implementation of this method, where everything after t_0 is unknown as it is in the future.

The ω parameter for the experiments was set to 8 min and is estimated based on the median Rule TTC of the Rules Dataset. This means that most of the rules created until 8 min before t_0 will be finished at t_0 and thus, the difference between the time series of the real μ and the observed. All the models tested fix this parameter to represent 8 min but as it depends on ρ, it is different for every model. The parameter is calculated using the formula $\omega = 8 \times 60/\rho$. That is, for the model $\gamma_{\hat{\mu}}(\rho, \lambda, \psi, \omega)$ with $\rho = 30$ s, ω will be 16 lags and for the models with $\rho = 60$ s, ω will be 8 lags.

Figure 9 summarizes the results of the experiment for $FoGP(y, \hat{y}_{\gamma_{\hat{\mu}}(\rho,\lambda,\psi,\omega)}, \tau = 0.1)$ for a particular choice of the parameters, that is, how good the models $\gamma_{\hat{\mu}}(\rho, \lambda, \psi, \omega)$ are to predict the Rule TTC of the rules, for μ being the minimum, median, mean and maximum functions. This figure can be read as follows. In the lower left plot, for the model $\gamma_{\hat{mean}}(\rho = 30, \lambda = 45, \psi = 240, \omega = 16)$, on average, a bit over 10% of the TTC predictions made with the model will have less than 10% relative error. The $(\rho, \lambda, \psi, \omega)$ parameters were selected using a grid search to maximize the FoGP at $\tau = 0.1$ for the median function and can be sub-optimal for other μ functions.

The Model δ_n

In his book "Deep Learning with Python" [10], Francois Chollet introduces a DNN architecture able to predict the temperatures for the Jena Dataset [11] slightly better than a naive model. We tried a similar approach to predict the Rule TTC, based on the time series of a number of observables we suspect could determine the TTC of such a rule at its creation time. This study should not be considered final nor exhaustive, but a preliminary study about the use of Deep Neural Networks to predict the Rule TTC based on the available data at the time.

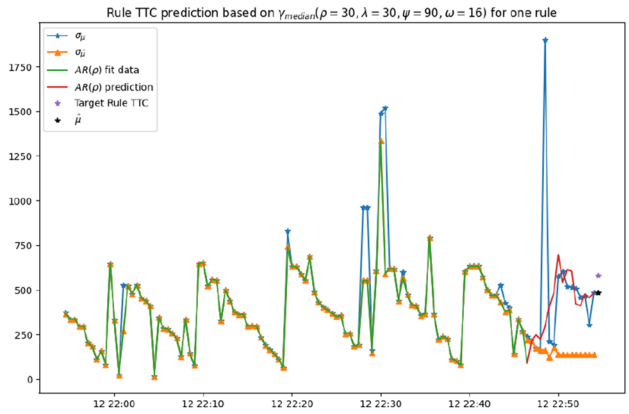

Fig. 8 Time series process to forecast the TTC of one Rule. The rule to predict was created on 2019-06-12 22:54:31. This defines the t_0 time, marked with a purple asterisk at the lower right part of the plot. Below there is a black asterisk that corresponds with the prediction done by the $\gamma_{\text{median}}(\rho = 30, \lambda = 30, \psi = 90, \omega = 16)$ model, that is a model which uses 30 lags of 30 s or the last 15 min to look ahead 8 min or 16 lags in the future and 45 min or 90 lags to fit the model. For this rule, the model underestimates the real Rule TTC. The real median Rule TTC Time Series is plotted in blue. This represents the real median Rule TTC of 30 s bins of those rules created before t_0. The orange line is the observed median Rule TTC, or the TTC of those rules that are finished before t_0. The agreement between the blue and orange line is good except in the minutes previous to t_0 itself. This results in ineffective β_μ models when using real time data. The green line in the plot is the data used to fit the γ_{median} model, that is the observed median Rule TTC till 8 min before t_0. The red line corresponds to the prediction made for the model for 16 lags ahead of $t_0 - 8$ min

From the Rules Dataset it is possible to create a set of time series from observables which can influence the Rule TTC. The minimum, median, mean, and maximum Rule TTC of previous transfers demonstrate at least some predictive power and have been used in previous models with limited success. Other variables that could influence the Rule TTC of future rules are the amount of transfers pending and also the amount of bytes pending. A way to calculate this is by extending the routines to calculate the time series for the minimum, median, mean, and maximum functions. The bins are filled with the sum of the bytes or the sum of the transfers of each rule for both time series, the observed and the real one. The difference between the two will be the time series of unfinished transfers and unfinished bytes. These values are known at rule creation time or can be approximated. As the majority of rules are over closed datasets, that is datasets to which new files can not be added, the number of files to be transferred and the size of each is mostly known at rule creation time.

We develop a model with a very similar structure to that proposed in Chapter 6 in [10]. We call it δ_n, where n is the number of convolutional filters or number of Long-Short Term Memory (LSTM) neurons. The main difference is the substitution of the Gated Recurrent Units (GRU) layer from the original Chollet model for a LSTM layer, as this was a proposed improvement suggested in the book. Input of the model consists of 10 channels, each of which represents the time series of some attribute calculated between t_0 or the rule creation time, and $t_0 - 120$ min, in bins of 30 s. The

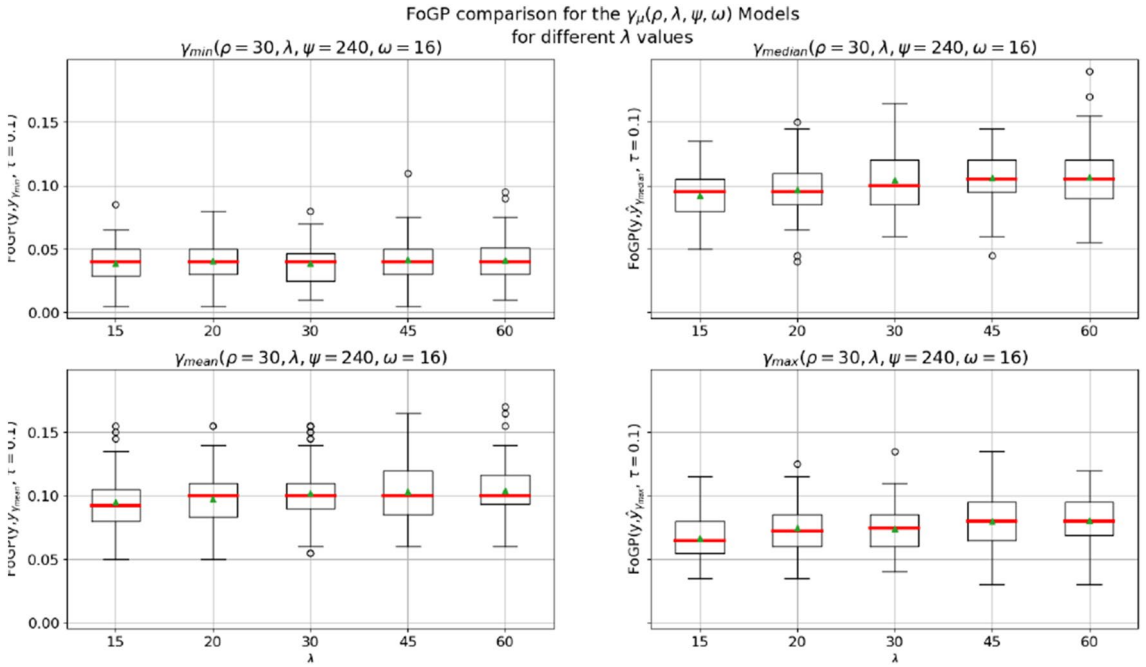

Fig. 9 FoGP comparison of different γ_μ models

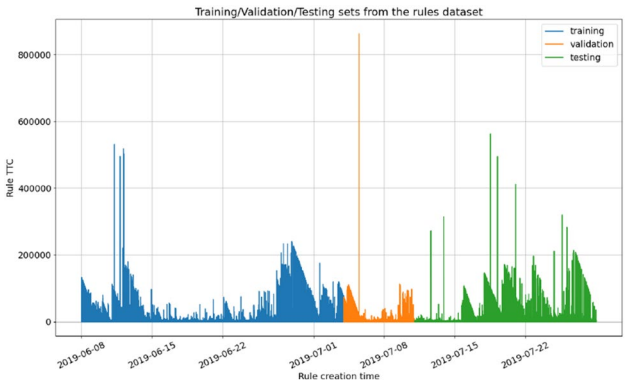

Fig. 10 Rule Dataset split for training, validation and testing

attributes used to build this time series were the minimum, median, mean, and maximum Rule TTC of each bin, plus the sum of transfers and bytes of finished, created, and pending rules. Each model was implemented using Keras/TensorFlow Python API and trained for 120 epochs using the RMSProp optimizer to minimize the Mean Absolute Error loss function.

Figure 10 shows the data splitting for training, validation, and testing. Training and validation data were selected based on the distribution of the the Rule TTCs at creation time. The training data comprises all the rules created between June 8th 2019 to July 3rd 2019. The validation set includes the rules created between July 4th 2019 and July 10th 2019. And finally, the testing set includes the rules created between July 11th 2019 and July 29th 2019.

The $\delta v v_n$ Model

The δ_n model family does not take into account information about the rule for which we want to predict the TTC, that is, the model does not include information about the target rule. In this section, we present a model that includes the number of transfers the target rule consists of, the sum of bytes of all the transfers, and the links this transfers will affect, that is, the list of sources and destinations for all the transfers. Unlike the time series information fed to the previous model, the data about the target rule is point wise, such that it is not data about the past state of the system, but of the present or t_0 time.

The model has 3 inputs, the several time series representing the past of the system, the sum of bytes and the number of transfers of the rule, and the list of links affected by those transfers. The only output of the system will be the Rule TTC. This kind of model cannot be implemented using the Keras Sequential Model. Instead, the Keras Functional API was used to conceive a family of models capable of handling the different types of inputs. We call this family the $\delta v v_n$ model family, where the n parameter is the number of convolutional filters or the number of LSTM neurons of the model.

Figure 11 shows the architecture of the $\delta v v_{32}$ model. Data flows from top to bottom. The left branch is the Chollet-Jena (δ_{32}) model in charge of digesting the time series data. The center branch input is the number of transfers and sum of bytes of the target rule. The right branch input is a list of integers, each of which represents a link that will be affected by one of the transfers. This list is truncated to 50, so only the first 50 links are going to be accounted. If less that 50 links are used, the sequence is padded with zeros. There is a special need to convert the (source, destination) pairs into a unique number to feed the emb_input layer. This process is done in a preprocessing stage using the Keras Tokenizer tool. The alphabet of links is 8762 words of the form `SRC-SITE__DSTSITE`. The LSTM layer after the embedding processes the links in order (Fig. 12). Even though link order should not matter, that is, the order in which the links appear in the embedding should not determine or affect the Rule TTC, the usage of this layer has proven important, because the prediction rate over the testing set is 1–2% better for the model family that use the LSTM layer, as shown in Fig. 13.

Normalization of all the numerical data was done using Eq. 15. This allows the model not to give more importance to some observables over others because of the scale. Typical normalization, where values are subtracted from the mean and divided by the variance is not enough in this case, due to the very long tail of the distribution of the values:

$$\ell = (\ln(x) - \ln(\text{mean}(x))/\ln(\text{std}(x)). \tag{15}$$

Figure 14 shows the histograms of the normalized vs. not normalized Rule TTC.

Both models were trained using the EarlyStopping callback that allows to monitor the progress of the validation loss. The callback stops training if there is no improvement after a fixed number of epochs and rolls back the weights to the ones of the last best model.

For δ_n models, the patience of the callback was set to 10 epochs. Figure 15 shows the δ_n model family stops after 10 or 11 epochs, meaning the best model is obtained after only 1 or 2 training epochs. The δ_n models are not able to generalize, and if trained for more epochs, model predictions converge to values around 480.

For the $\delta v v_n$ models, EarlyStopping patience was set to 5. Figure 12 shows that this model family learns from the training data until epoch 12 in the best case, that is for model $\delta v v_{32}$. After that, there is no improvement in validation loss. Naturally, the larger the model, that is the n, the faster the model overfits.

Figure 16 shows the comparison of the $FoGP(y, \hat{y}, \tau = 0.1)$ of the δ_{32} and $\delta v v_{32}$ models. For

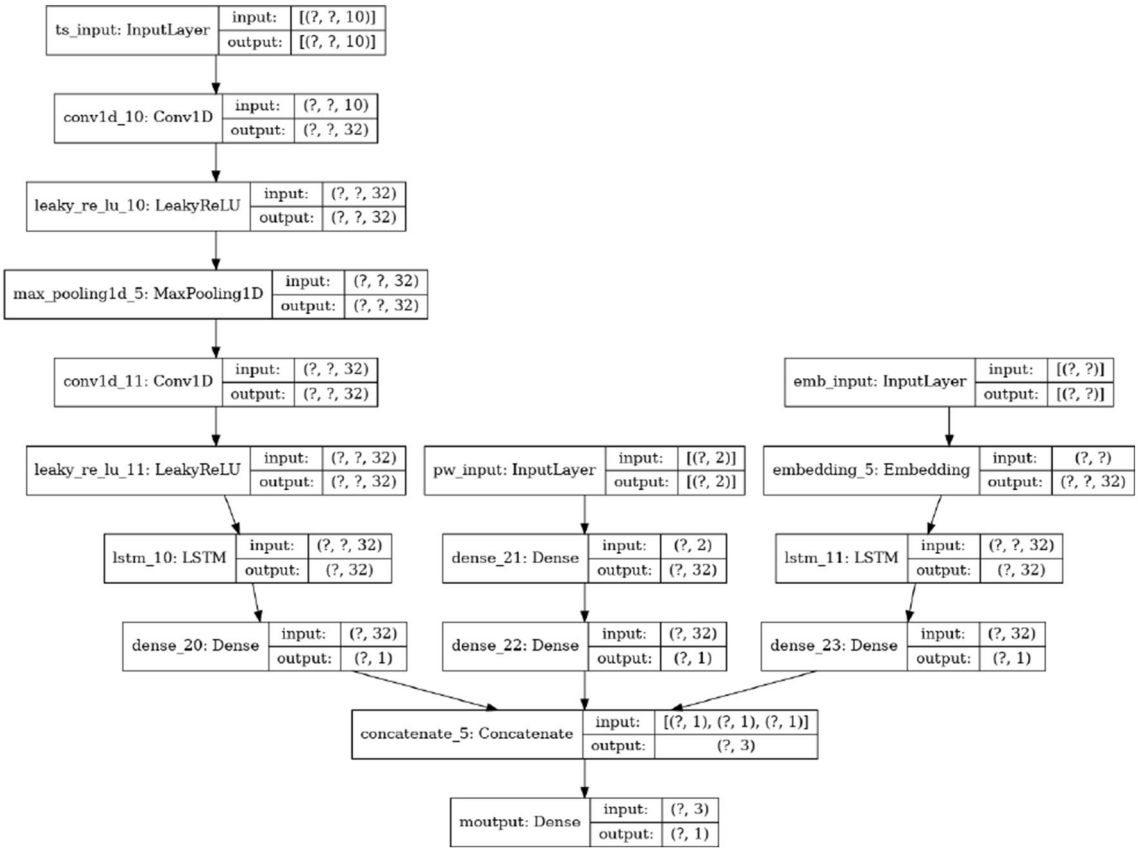

Fig. 11 FunnelNet architecture

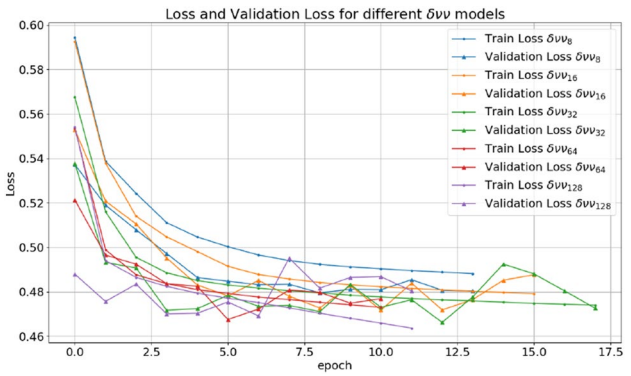

Fig. 12 Training loss vs. validation loss for several $\delta\nu\nu_n$ models. Training was done setting an EarlyStopping callback measuring the validation loss with patience of 5 epochs

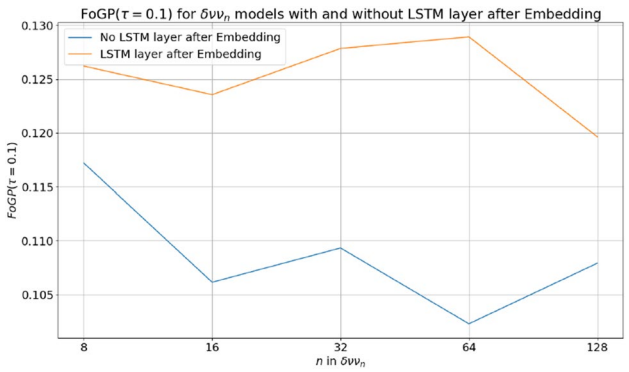

Fig. 13 $FoGP(\tau = 0.1)$ for two variants of the $\delta\nu\nu_n$ models. Models without a LSTM layer after the Embedding layer perform worse than the models with a LSTM layer after the Embedding layer

comparison, the $\kappa = 562$ model, the model that always predicts a constant value for the Rule TTC, was included. A subsample of 300 rules was selected. For each rule, a prediction is made using all the models. Then, the $FoGP(y, \hat{y}_\delta, \tau = 0.1)$, $FoGP(y, \hat{y}_{\delta\nu\nu}, \tau = 0.1)$, and $FoGP(y, \hat{y}_\kappa, \tau = 0.1)$ was calculated. The procedure was repeated 1000 times. The box plot shows the distribution of the different FoGP obtained. Model $\kappa = 562$ shows that on average, 12.2% of the predictions lies within a 10% relative error from the real value. The $\delta\nu\nu_{32}$ model outperforms the $\kappa = 562$ model by a modest 5.7% on average.

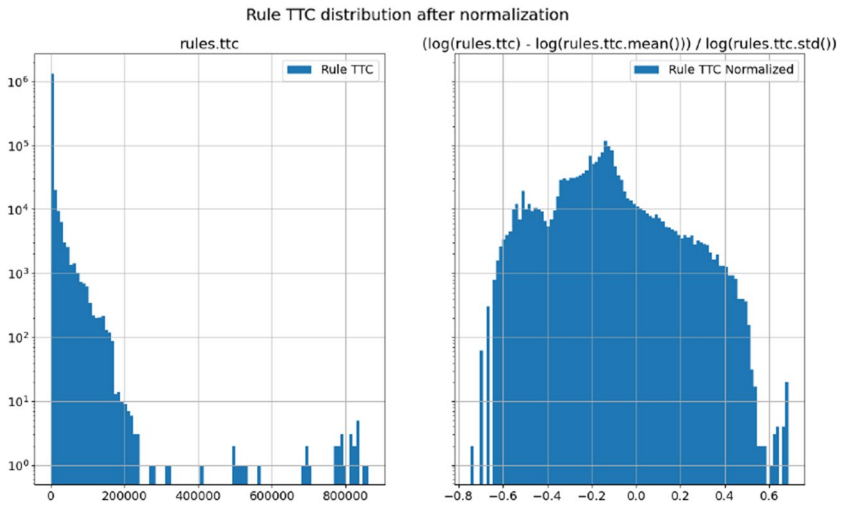

Fig. 14 Rule TTC distribution vs. Rule TTC distribution after normalization using Eq. 15

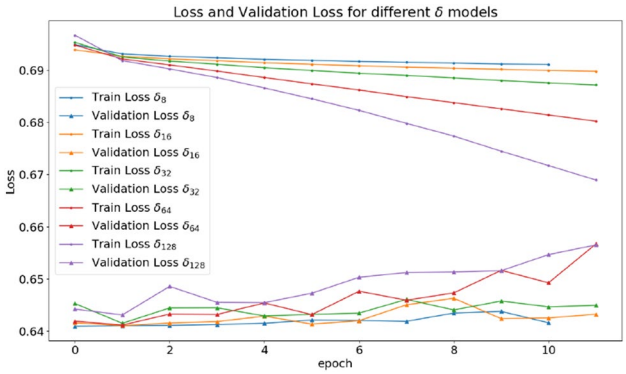

Fig. 15 Training loss vs. validation loss for several δ_n models. Training was done by setting an EarlyStopping callback measuring the validation loss with patience of 10 epochs. This plot suggests the model is not able to learn from the training data

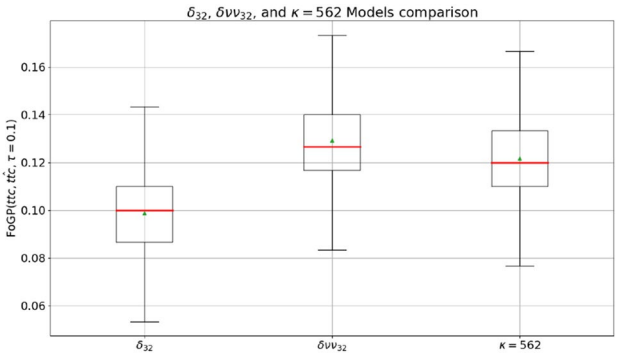

Fig. 16 This plot shows the distribution of the FoGP($y, \delta_{32}, \tau = 0.1$) and FoGP($y, \delta\nu\nu_{32}, \tau = 0.1$) over 1000 repetitions of the experiment to make a prediction for 300 samples. Numbers show that, on average, 9.9% of the predictions made with model δ also known as Chollet-Jena have less than 10% relative error. Meanwhile, the 13.0% of the predictions made with model $\delta\nu\nu$ also known as FunnelNet, have less than 10% relative error. For comparison, the results of model $\kappa = 562$ are shown. This is the model that makes a constant prediction for the Rule TTC of 562 s. For this model 12.2% have less than 10% relative error

Evaluation of Model Performance

It is instructive to compare the models with several values of τ, especially in the range (0.01, 0.25), to see how many of the predictions of each model have more than 1% and less than 25% relative error. For the comparison with the κ model to be fair, the best constant for each τ must be selected in order to maximize the $FoGP(y, \hat{y}_\kappa, \tau)$. Using the same training data used to fit models δ and $\delta\nu\nu$, the κ/τ space was scanned calculating the $FoGP(y, \hat{y}_\kappa, \tau)$ in the ranges $\kappa = (0, 2000)$ in steps of 1 and $\tau = (0.01, 2.0)$ in steps of 0.01. This procedure defines a surface defined in $\mathbb{R}^3$ with a local $FoGP(y, \hat{y}_\kappa, \tau)$ maximum for each κ and τ. We assume this is the optimal constant to predict the target with a given FoGP. Figure 17 shows this local maximum, that is, the constant that predicts the training set with the highest FoGP. Several things arise from this plot. First, there is a peak at $\kappa = 567$ which corresponds neither with the mean Rule TTC of the training set, that is 1962.1 s, nor with the median of 439 s. This means that both the prediction using the mean and the median are sub-optimal in terms of FoGP. Second, when $\kappa = 0$ the FoGP tops 1.0 all the predictions have less than 100% relative error. The explanation for this effect is straightforward. If the prediction for whatever value x is 0, then the relative error is calculated as $|x - 0|/x = 1$, meaning the error is 100%. As the FoGP measures how many predictions are less than τ, when $\tau > 1.0$, if the prediction is 0, all the predictions are accounted as having less than 100% relative error. Third, the FoGP values in the τ range (0.01, 0.25) fall between 0.027 and 0.251, meaning between 2.7% and 25.1% of the predictions presents less than between 1% and 25% relative error.

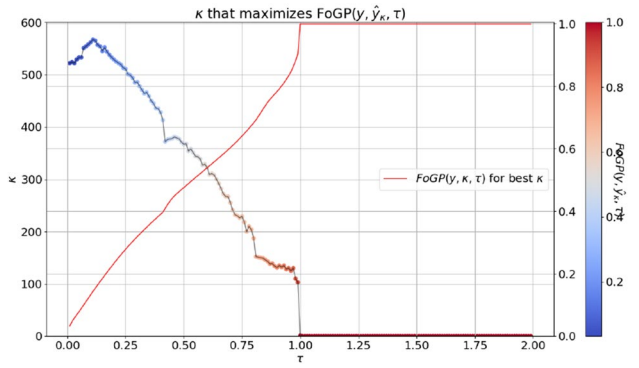

Fig. 17 Optimal constant prediction search based on the maximization of the $FoGP(y, \hat{y}_\kappa, \tau)$ function. Black line represents the κ with maximum $FoGP(y, \hat{y}_\kappa, \tau)$ for a given τ. The colored points represents the actual $FoGP(y, \hat{y}_\kappa, \tau)$ value, the bluish the worse, the redder the better. The red line also represents the achieved FoGP with the y-axis on the right

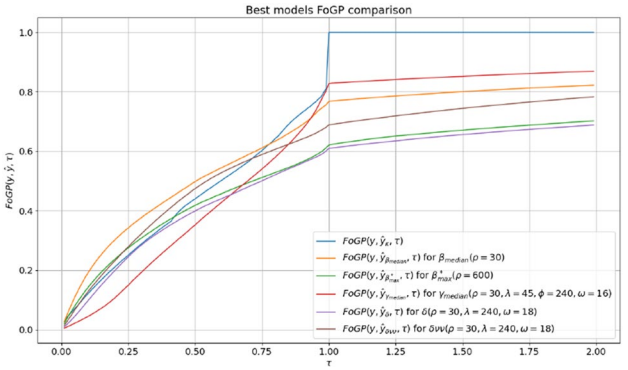

Fig. 18 FoGP comparison over a τ range from 0.01 to 2 for the best models known, which were presented in this work. Predictions for all the models were made for all the rules created between 2019-07-11 and 2019-07-29. Model $\beta_{\text{median}}(\rho = 30)$ outperforms all the models. However, the real median of previous Rule TTC needs to be known for the model to work and this information is not available at t_0. Model $\delta\nu\nu$ is the best model following the FoGP criteria with τ between 0.22 and 0.70 and is the model with greatest potential to be extended. The performance of all the models are comparable with the performance of Model κ, and for its simplicity, it should be the preferred model

The κ model outperforms when τ is in the range (0.01, 0.04). In the range (0.04, 0.65) the $\delta\nu\nu$ model is better.

Both δ and $\delta\nu\nu$ models return continuous values and hence do not make any sense to measure the FoGP when $\tau = 0$ as the probability of the model to predict the exact value of the Rule TTC is almost zero. It does make sense to measure it for the κ model as the constant value is integer. This explains the better performance of the κ model for low values of τ. However, there is no noticeable change in the FoGP when the predictions of the δ and $\delta\nu\nu$ models is rounded.

Among the possible uses of these models for real world applications, we count the benefits for rule and transfer requests scheduling and the ability to give feedback to the users of the system about the time to complete of their transfers. Before any model can be used to make predictions to improve the scheduling of transfers or rules, we need two conditions be satisfied. First, the model needs to be able to make a prediction at the time the rule or transfer is created or the t_0 time. Second, the accuracy of the model should be high enough to actually improve the schedule. From talks with the experts, we expect a useful number will be a $FoGP(y, \hat{y}, \tau = 0.1)$ of around 0.95. All presented models except model α can make predictions at the rule creation time, although the accuracy of the models presented here are below 0.2. The models that can make predictions at t_0 time can also be used to give feedback to the users about the TTC of their transfers. However, other models that include information of times post-t_0 have better accuracy in general and can be used too, depending on the need of the users to have the feedback early in the lifetime of the rule or transfer, or late, in which case the prediction will be more accurate (Fig. 18).

Model κ

The κ model, which always predicts a constant value, allows us to put a lower bound for the performance of the models over a range of interesting τ values. Optimizing the constant to maximize the FoGP results in a model that is surprisingly difficult to improve upon, both at high and low τ values. By its simplicity, and because its performance is comparable to other more sophisticated models, it should be the preferred to be implemented, for example to give feedback to users about the TTC of their transfers. If that is the case, the upper bound of a confidence interval could be interesting for users.

Model α_k

Model α is the only model of the studied ones that is not directly comparable with the other models due to inability to make predictions at the Rule creation time. Model α needs at least two transfers within the rule to finish to fit and forecast when the other transfers probably will finish. This makes the model suitable to give feedback to the users but will not be helpful to improve the scheduler, as the decision about where to send the transfers will need to be done at rule creation time and before any transfer is submitted or finished. The model shows the non-linearity of the progression of the transfers, giving insights of the nature of the rules and their behavior. The time between transfer submissions for the transfers of a rule is not constant. Rucio's Conveyor daemon may consider that FTS has a high enough number of transfers already and decide not to submit more transfers

until some of those active transfers finish, increasing the Rucio Queue Time for part of the transfers of the rule. This will impact directly in the Rule TTC and this model will not be able to forecast this future delays.

Models $\beta_\mu(t_0, \rho)$ and $\beta_\mu^*(t_0, \rho)$

Models $\beta_\mu(t_0, \rho)$ and $\beta_\mu^*(t_0, \rho)$ make a prediction calculating a function μ over the Rule TTC of those rules created in the last ρ s. The difference between β_μ and β_μ^* is that β_μ^* excludes those rules that ends after t_0. The β_μ model cannot be implemented with real time data as it calculates the μ function over the Rule TTC of all the rules that have started at some point in the past, including the ones that have not finished yet. This information from the future added to the model makes the two models radically different. One could assume that if the μ function could be predicted with 100% accuracy, then FoGP of the model β_μ represents the theoretical limit of FoGP of the model β_μ^*, as the first include more information than the second. Yet, this statement does not hold in general, for example, for the function that take the maximum, including more information in the model does not make it more accurate. The $\beta_{\max}$ model makes a prediction by calculating the maximum Rule TTC of all the transfers created between t_0 and ρ. The bigger the ρ is the bigger is the chance that there exists a very slow rule. But $\beta_{\max}^*$ filters out those transfers that have finished after t_0, and thus the Rule TTC is throttled to the value of ρ. For this reason, the $FoGP(y, \hat{y}_{\beta_{\max}^*}, \tau = 0.1)$ presents a peak when ρ is near 600. This value is close to the best value for the model κ at $\tau = 0.1$, which is 562. β_μ^* models with other parameters presents lower FoGP values than β_{max}^* at $\tau = 0.1$, and thus are considered inferior models.

Figure 19 shows that $\beta_{max}^*(\rho = 600)$ outperforms model κ in the τ range between 0.04 and 0.22. This is the best model known to date in that range. It is not possible to implement the β_{median} model without knowing the Rule TTC of rules that didn't finish yet. If a model for a perfect prediction of the median of the Rule TTC exists, then the $\beta_{\text{median}}(\rho = 30)$ shows the best performance across a wide range of τ.

Model $\gamma_\mu(t_0, \rho, \lambda, \psi, \omega)$

The γ_μ model family is the first approach to solve the problem using time series analysis. The γ_μ is an auto-regressive (AR) model, where the input is the time series of the Rule TTC. The function μ is calculated in bins of ρ s. The input for the AR model consists of λ lags. The model is fitted using ψ lags and the look ahead of the model is ω lags. The best model was obtained by scanning the parameter space and maximizing the FoGP, as detailed in Sect. 3.3. Model $\gamma_{\text{median}}(t_0, \rho = 30, \lambda = 45, \psi 240, \omega = 16)$ achieved the best

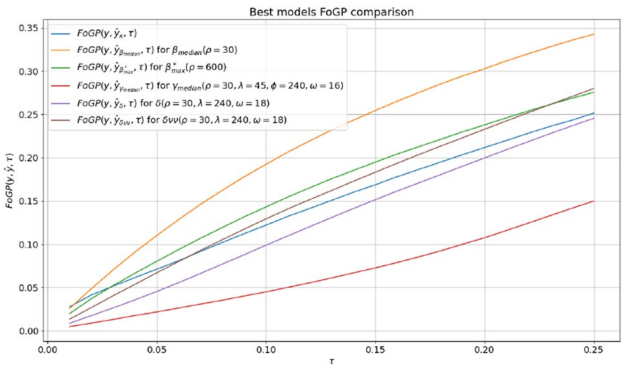

Fig. 19 FoGP comparison zoom over a τ range from 0.01 to 0.25 for the best models

FoGP at $\tau = 0.1$. If this model would predict the median with 100% accuracy then its results should be comparable with those obtained with the β_{median}. The results show that the γ_μ model is not as good, especially at low τ. The γ_{median} model is better than β_{median} only for $\tau > 0.91$. This model seems to be not accurate enough and other more complex models are worth to try. Integrated models were discarded after verifying that the time series show no trend, ergo there is no need for differentiation. Moving Average models are used after verifying that the time series are not stationary, which is not the case for long runs of Rule TTC time series. A straightforward check showed that the standard deviation from the mean changes over time, and thus a General Auto-Regressive Conditional Heterokedasticity (GARCH) model is more appropriate.

Models δ and δνν

The δ model is the first attempt to solve the forecast problem with neural networks using a modified model proposed by F. Chollet. This approach was shown to be ineffective but its accuracy is higher than the accuracy of model γ_{median}. Model δ includes the information of the past state of the system in the form of time series but it does not include information of the present. Information from the rule that is known at the creation time like the number of transfers or the sum of bytes the system must process to complete the rule are not included in model δ. This observation leads to the $\delta\nu\nu$ model, a deep neural network model with multiple inputs that includes the time series from δ model, but also the number of bytes, number of transfers, and the links affected by the rule. $\delta\nu\nu$ model is the best practical model in the τ range from 0.25 to 0.70, but more importantly, it is the easiest model to extend. We expect that this model would benefit enormously if information about failed transfers per link, history of transfers submitted to FTS, and history of the rate of the link were available and could be added as inputs.

Conclusions and Future Work

The distributed data management for the experiments using the Worldwide LHC Computing Grid form a complex ecosystem with dynamic interactions. Since its commissioning in 2014, Rucio has become the de-facto standard for scientific data management, even outside the CERN community.

The accuracy of the predictions of the models will be limited by the amount of data about the system available at a given moment, and by the stochastic processes involved in certain parts of this system. Rucio's importance and the rich amount of data gathered about the transfers and rules life cycles, contributes to the importance of this study.

Several models were presented and evaluated during this work, especially for Rule TTC prediction. All presented models except model α can make predictions at the rule creation time, although the accuracy of these does not allow the models to be used to improve the scheduling of the transfers or rules. The expected threshold that would make these predictions useful is a

$FoGP(y, \hat{y}, \tau = 0.1)$ of around 0.95. Even if the accuracy of the models presented here is not enough for scheduling purposes, excluding model α, these are the best models known to date for Rule TTC prediction at rule creation time.

This work lays the foundation for future models and establishes the metric by which they should be compared. The $\delta\nu\nu$ model is promising. The number of failed transfers per link, aggregated over a the last 10–30 min or some recent history in time series form could improve the performance of this model. It is known that the FTS Optimizer penalizes the links if there are failed transfers. Those links without failures in the recent history are preferred over the ones with failures. Thus, the transfers pending that use those links with failures will be delayed more than the transfers that will use a link without failures. Adding this information to the model could increase its performance.

The $\delta\nu\nu$ model could also benefit from knowing how many transfers have been submitted to the links that the target rule will affect. The way to calculate this from the transfers dataset is to get all the transfers that have been submitted, but did not finish yet, ignoring the started timestamp.

The authors also suggest a study of the prediction accuracy needed to improve the scheduling. A rationale of this value in terms of FoGP will shed more light on the usefulness of the models presented here and future models.

Data Availability Statement This manuscript has associated data in a data repository. The data is available in https://doi.org/10.5281/zenodo.4320937.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Bogado J, Lassnig M, Monticelli F, Díaz J, Beermann T (2020) ATLAS Rucio transfers dataset. Zenodo. https://doi.org/10.5281/zenodo.4320937
2. Lassnig M, Toler W, Vamosi R, Bogado J (2017) Machine learning of network metrics in ATLAS distributed data management. J Phys Conf Ser 898:062009. https://doi.org/10.1088/1742-6596/898/6/062009
3. Begy V, Barisits M, Lassnig M, Schikuta E (2020) Forecasting network throughput of remote data access in computing grids. J Comput Sci 44:101158. https://doi.org/10.1016/j.jocs.2020.101158
4. Bogado J, Monticelli F, Diaz J, Lassnig M, Vukotic I (2018) Modelling high-energy physics data transfers. In: 2018 IEEE 14th international conference on e-science (e-Science)
5. Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. Int J Forecast 22(4):679. https://doi.org/10.1016/j.ijforecast.2006.03.001
6. Zheng A (2015) Evaluating machine learning models. O'Reilly Media Inc., Newton
7. Makridakis S (1993) Accuracy measures: theoretical and practical concerns. Int J Forecast 9(4):527. https://doi.org/10.1016/0169-2070(93)90079-3
8. Shumway RH, Stoffer DS (2005) Time series analysis and its applications (Springer Texts in statistics). Springer, Berlin
9. Seabold S, Perktold J (2010) 9th Python in science conference
10. Chollet F (2017) Deep learning with Python, 1st edn. Manning Publications Co., Shelter Island
11. G.h.j. The dataset recorded at the Weather Station at the Max Planck Institute for Biogeochemistry in Jena. Weather archive Jena air temperature, atmospheric pressure, humidity, recorded over seven years. https://www.s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip. Accessed 4 Aug 2020