Hauck, Florian; Güth, Albrecht; Kliewer, Natalia; Rößler-von Saß, David

Working Paper

Applying generative adversarial networks to generate synthetic train trip data for train delay prediction

Discussion Paper, No. 2026/7

Provided in Cooperation with:
Free University Berlin, School of Business & Economics

Suggested Citation: Hauck, Florian; Güth, Albrecht; Kliewer, Natalia; Rößler-von Saß, David (2026) : Applying generative adversarial networks to generate synthetic train trip data for train delay prediction, Discussion Paper, No. 2026/7, Freie Universität Berlin, School of Business & Economics, Berlin,
https://doi.org/10.17169/refubium-51427

This Version is available at:
https://hdl.handle.net/10419/338080

FREIE UNIVERSITÄT BERLIN

Applying Generative Adversarial Networks to Generate Synthetic Train Trip Data for Train Delay Prediction

Florian Hauck
Albrecht Güth
Natalia Kliewer
David Rößler-von Saß

School of Business & Economics
Discussion Paper

Information Systems

2026/7

Applying Generative Adversarial Networks to Generate Synthetic Train Trip Data for Train Delay Prediction

Working Paper

Florian Hauck, Albrecht Güth, Natalia Kliewer, and David Rößler-von Saß*

Department of Business Information Systems, School of Business & Economics, Freie Universität Berlin

This paper examines the possibilities of creating synthetic train trip data with Generative Adversarial Networks (GANs). A real data set from Deutsche Bahn is enhanced with synthetic data created by using a Conditional Wasserstein Generative Adversarial Network (CWGAN). The synthetic data is analyzed and compared with the original data using statistical methods as well as machine learning models. The results show that the synthetic data is very similar to the original data in terms of data structure and dependencies, but at the same time contains enough noise to not just copy already existing instances. To analyze and measure the quality of the synthetic data, different supervised machine learning models are trained to predict the change of delay of trains at a specific station based on the arrival delays of other trains at that station. These models are then each trained once using the real data and once using the real data enhanced by synthetic data. All models are evaluated using a test set containing only real data that was not used to train the models. The results show that the R^2 value of delay predictions increases significantly when using the enhanced data set. In particular, neural network-based models can benefit from the larger amount of input data. The proposed approach of generating synthetic train trip data with a CWGAN can also be applied to various other railway data analysis projects that require a large amount of input data. In addition, the presented approach is particularly interesting because, unlike most GAN approaches discussed in current literature, the data basis contains numerical data and not image data.
Keywords— Generative Adversarial Networks; Train Delay Prediction; Railway Analysis

*Electronic address: david.roessler@fu-berlin.de; Corresponding author

1 Introduction

Data analysis projects generally require a large amount of input data to produce reliable results. Collecting and preparing the data can often be one of the most important and difficult parts of data analysis projects, as the input data determines the reliability and quality of the results. However, the required input data is not always available in the desired quality and quantity. In railway networks, for example, the available amount of train trip data is limited by the capacity of the network and the frequency at which the various trains run. In addition, the data often must be current, and historical train trip data from previous years may not be useful because those trains ran on a different schedule.

The literature discusses a variety of different railway analysis projects that rely on large data sets of train trip data. For an overview of current applications of big data analytics in railway systems, see Ghofrani et al. (2018). For example, some studies focus on analyzing the propagation of train delays (Büker & Seybold, 2012; Yuan, 2006) or on distinguishing between primary and secondary delays (Rößler et al., 2021). Many other works are concerned with the prediction of train delays. For example, approaches that use linear regression models to measure influences on train delays (Gorman, 2009; Goverde, 2005) or approaches that focus on stochastic or machine learning models to predict train delays (Corman & Kecman, 2018; Berger et al., 2011; Oneto et al., 2018). All of these approaches rely on large data sets of train trip data that they can analyze.

However, a large amount of historical train trip data is not always available. For example, a maximum of 365 train trips are available for an analysis of the delays of a specific train in the past timetable year and only if the train runs daily. For many less frequent trains, the number of trips is even lower. There are some simple ways to increase the number of trips, e.g., by combining different trains running the same route at different times of the day. But even then, the number of trips usually remains low, and grouping different trains can create new challenges, as it is not always possible to simply treat different trains the same way. For example, connecting trains, weather conditions, or other relevant factors may change throughout the day. Therefore, it is generally helpful to find other ways to increase the amount of available data.

Synthetic data can be created in a variety of ways. Some methods use mathematical models or distributions that fit the original data and can then be used to create new data, such as Monte Carlo simulations or discrete-event simulations (Goncalves et al., 2020). A major challenge with these methods is that it can be difficult to find a good distribution for some data sets. Other data-driven machine learning methods are trained on the original data and can then generate new synthetic data, e.g., deep learning approaches such as Variational Autoencoders (VAE) or Generative Adversarial Networks (GAN) (Kingma & Welling, 2014; Goodfellow et al., 2014). GANs have been successfully applied in several projects (Frid-Adar et al., 2018; Karras et al., 2017; Brock et al., 2018). They work particularly well with image data and are able to generate synthetic images that resemble the images of a given data set with original images. This is very useful, for example, in the field of healthcare analytics for diagnostic purposes. Although GANs work very well with image data, the concept has rarely been applied to other data formats, such as numerical data. We show that GANs can also be applied to such data formats by generating synthetic train trip data. We then analyze the newly created data and show how this data can be used to augment small data sets and thereby improve the performance of analysis tasks performed on this data. We demonstrate our approach using machine learning models to predict future train delays, but the results are also of interest for many other analysis tasks that rely on a large data set.

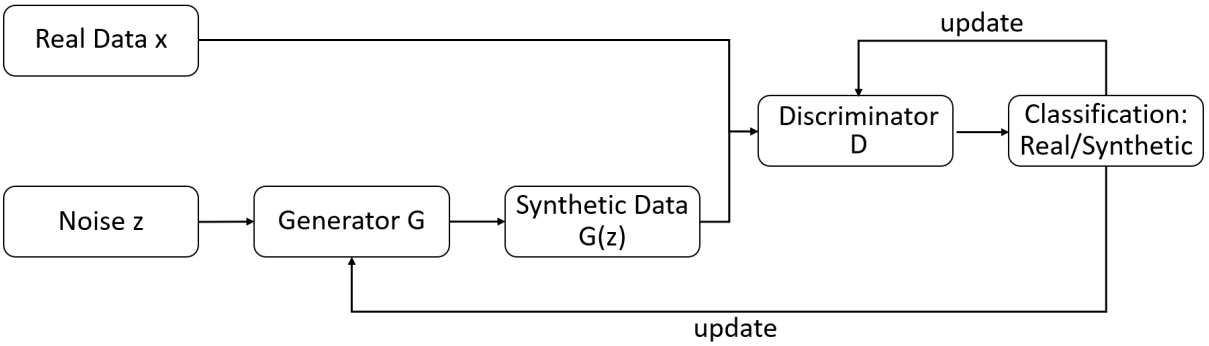

Figure 1: GAN architecture

2 Generative Adversarial Networks

The concept of GANs was first introduced by Goodfellow et al. (2014). A GAN consists of two neural networks, one called generator (G) and the other called discriminator (D). Both networks compete in a zero-sum game, which means that the gain of one network is equal to the loss of the other. The generator uses random noise data (z) to generate synthetic data (G(z)). The goal of the generator is to maximize the probability that the discriminator is unable to distinguish between real data (x) and synthetic data. Therefore, the synthetic data must have the same characteristics and statistical properties as the given set of real data. The discriminator aims to distinguish between synthetic and real data. In other words, the generator tries to fool the discriminator by producing synthetic data of high quality. Both networks are trained and updated in many iterations, iteratively improving the quality of the synthetic data, since both networks must improve simultaneously to improve their objective functions. The GAN architecture is shown in Figure 1.

The detailed description of the mathematical formulation for GANs is presented by Goodfellow et al. (2014). The loss function of a GAN is shown in equation 1. The generator tries to minimize the function and the discriminator tries to maximize the function. E(x) is the expected value over all real data instances. D(x) is the discriminator's estimate of the probability that real data instance x is real. G(z) is the generator's output when given noise z. D(G(z)) is the discriminator's estimate of the probability that a fake instance is real. E(z) is the expected value over all random inputs to the generator (Weng, 2019).

$$E_{\mathrm{x}}[log(D(x))] + E_{\mathrm{z}}[log(1 - D(G(z)))] \tag{1}$$

A common practical issue with GANs is mode collapse. This happens when the generator produces only one or a few modes of the original data set. For example, the generator creates only train trips within a certain delay range. Other practical issues with the original GAN architecture are that the GAN does not improve because the feedback between generator and discriminator does not work properly or the loss functions of the generator and the discriminator do not converge (Metz et al., 2016). There are several approaches in the literature to solve these problems by improving the GAN architecture, and a variety of adapted GANs have been discussed. Three improved GAN versions are described in the following.

2.1 Conditional GAN

A conditional GAN (CGAN) was developed by Mirza and Osindero (2014). They add an additional layer of information y, e.g., class labels, to the data and change the objective function

to the formula shown in equation 2. This conditional information allows, for example, to specify concrete modes for the generator, and can be used effectively for imbalanced datasets (Douzas & Bacao, 2018).

$$E_{\mathrm{x}}[log(D(x \mid y))] + E_{\mathrm{z}}[log(1 - D(G(z \mid y)))] \tag{2}$$

2.2 Wasserstein GAN

A Wasserstein GAN (WGAN) is an adaption introduced by Arjovsky et al. (2017) and Gulrajani, Ahmed, Arjovsky, Dumoulin, and Courville (2017). They try to solve many of the problems of the original GAN architecture by using Wasserstein distance as a metric for the loss function. The metric measures the difference of data distributions between a real data set and a synthetic data set (Shenvi et al., 2019).

2.3 Conditional Wasserstein GAN

An improvement of the WGAN is the conditional Wasserstein GAN (CWGAN) which was introduced by Zheng et al. (2020). It includes an additional layer of information, e.g., class labels, and also uses the Wasserstein distance as a metric for the loss function. The CWGAN therefore combines the advantages of CGAN and WGAN.

Ba (2019) or Zheng et al. (2020) compare and discuss the performances of different GAN architectures in various scenarios. Ba (2019) uses different GANs to create credit card fraudulent data. The results show that the CWGAN produces the most realistic synthetic data instances compared to the results of a GAN, CGAN and WGAN. Zheng et al. (2020) also compare the CWGAN with the other three GAN types. They use 17 data sets and create synthetic data for each of them. The results show that the CWGAN generally performs better than the other GAN models. Since the CWGAN combines the advantages of the other GAN types and also performs best in other papers discussed in the literature, it is also applied in this paper to generate the synthetic data instances.

3 Description of the Data Set

The data set used for this paper contains real train trip data from Deutsche Bahn from one timetable year. To analyze the effectiveness of the proposed CWGAN model, one station (S) and three target trains (A, B and C) stopping at station S are selected. The three trains are referred to as target trains because the prediction models later aim to predict their change of delay when departing from station S. All target trains operate on the same route, i.e., they pass the same stations, but at different times of the day. The entire route of the selected trains consists of 13 stations, with stops and transfer times for passengers at each station. Station S can be considered as a medium-sized connecting node in the network and is reached about halfway along the route. The selected trains run daily, providing 365 trips for each train.

A separate data set is created for each of the three target trains. Each data set contains a column for the absolute arrival delay of the target train in seconds (d_a) at station S. This delay is the difference between the scheduled arrival time at station S and the actual arrival time at station S. The delay value is positive if the train arrived too late and negative if the train arrived too early. Each data set also contains a column for the change of delay (Δy) of the target train in seconds that occurs during the dwell time at station S. Δy is the difference between the departure delay (d_d) and the arrival delay (d_a) at station S. A positive value means that the train built up a delay during the stop in S, and a negative value means that the train was able to

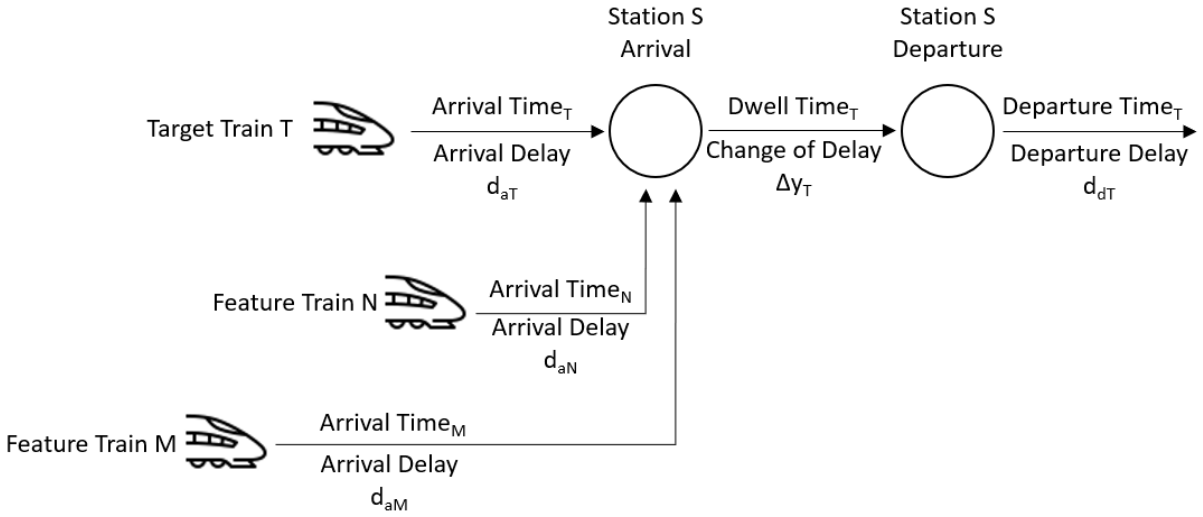

Figure 2: Dataset feature description

reduce the delay. In addition, the data sets contain the absolute arrival delays in seconds of other trains that arrive at station S within a time window of 30 minutes before or 30 minutes after the arrival of the target train at S. These other trains are referred to as feature trains because their delays are used as features to predict Δy. An illustrative representation of the data sets is shown in Figure 2.

The three resulting data sets for the target trains A, B, and C each contain up to 365 instances and a separate column for each feature train, as well as a column for d_a and Δy. See Table 1 for an illustration of the structure of the data set. Since the target trains reach station S at different times, the set of feature trains is different for each target train. As a result, the number of features in the three data sets also varies. Feature trains can only be included in the data set if they run almost as frequently as the target train. Otherwise, the data set would have missing values in all cases where the target train was running and the feature train was not. Missing values should be avoided because many prediction models cannot handle them. Therefore, the data set contains only feature trains that run almost daily. The remaining missing values are imputed using the R-package MICE (Multiple Imputation by Chained Equations). MICE imputes missing values based on other values in the data set using linear regression models (Van Buuren & Groothuis-Oudshoorn, 2011). Since these imputations cannot be perfectly correct and could negatively affect the predictions, an additional weighting feature is added to the data set. The weighting feature is used to give instances with few or no imputed values a higher weight than instances with many imputed values. The weighting factor ranges from 1 for no imputed values to 0.5 if half of the values are imputed. If more than half of all values are imputed, the instance is removed from the data set due to its low reliability. Therefore, the number of instances for the three data sets is slightly lower than 365. For a detailed description, see Table 2. The table shows that all three data sets contain around 360 instances, and the number of feature trains varies from 13 to 20. It also shows that the delay characteristics are similar, as the average arrival delay of the target trains at station S is always between 250 and 300 seconds. On average, all three target trains are able to reduce some of their delays during their dwell times in station S.

In addition to the columns for arrival delays, a column for the temperature at Station S is added to include weather information into the data sets. This is useful for the prediction models since weather information can affect train delays. Finally, standard transformation is performed

Table 1: Exemplary format of the data set

Target Train d_a	Target Train Δy	Feature Train 1 d_a	Feature Train 2 d_a	...
252	-28	102	57	...
-130	55	507	222	...
428	-72	242	44	...
...	...	...	...	...

Table 2: Descriptive data set analysis

Dataset	Target Train	Instances	Feature Trains	Mean d_a	Mean Δy
1	A	360	20	251	-148
2	B	360	19	301	-119
3	C	359	13	296	-238

for all features so that the mean value for each feature is 0 and the deviation is 1. The formula for the standard transformation is shown in equation 3 where Z stands for the transformed data, X for the original data, μ is the mean value and σ is the deviation.

$$Z = \frac{X - \mu}{\sigma} \tag{3}$$

This transformation is necessary for the machine learning models, especially for neural networks, since different features must be on the same scale to be used. The resulting data sets form the basis for the predictive models and for the GAN described in the following chapters.

4 Description of the Research Approach

The approach for this paper can be divided into four steps. First, the synthetic data is created using a CWGAN. Second, the synthetic data is analyzed using statistical methods. Third, different prediction models for train delay prediction are trained, once using the real data and once using the synthetic data. Fourth, the performance of the different prediction models is analyzed to determine if training on synthetic data leads to better predictions. Figure 3 shows an overview of the four steps. A detailed description of each step is given in the following subsection.

The three data sets with real data from trains A, B, and C are used to train and configure a CWGAN. After successfully training the CWGAN, the model can be applied to generate any number of synthetic data instances similar to the instances from the real data sets. In this way, three new and much larger data sets with synthetic data are created. The synthetic data sets have the same format and structure as the real data sets. This means that their instances also represent arrival delays at station S as well as delay changes of the target train at station S. However, these delays never actually occurred because the data is artificially generated.

The synthetic data is only useful if it has the same patterns as the real data. That is, even if the delays in the synthetic data never occurred, the instances must look like as if they could have occurred. To verify that the correlations and dependencies from the real data are transferred to the synthetic data, simple statistical measures are calculated. This includes a comparison of the mean values and standard deviations for each feature and a correlation analysis. If the metrics show no significant deviations between the real and the synthetic data sets, it can be assumed that the CWGAN produces high-quality synthetic data that cannot be easily distinguished from

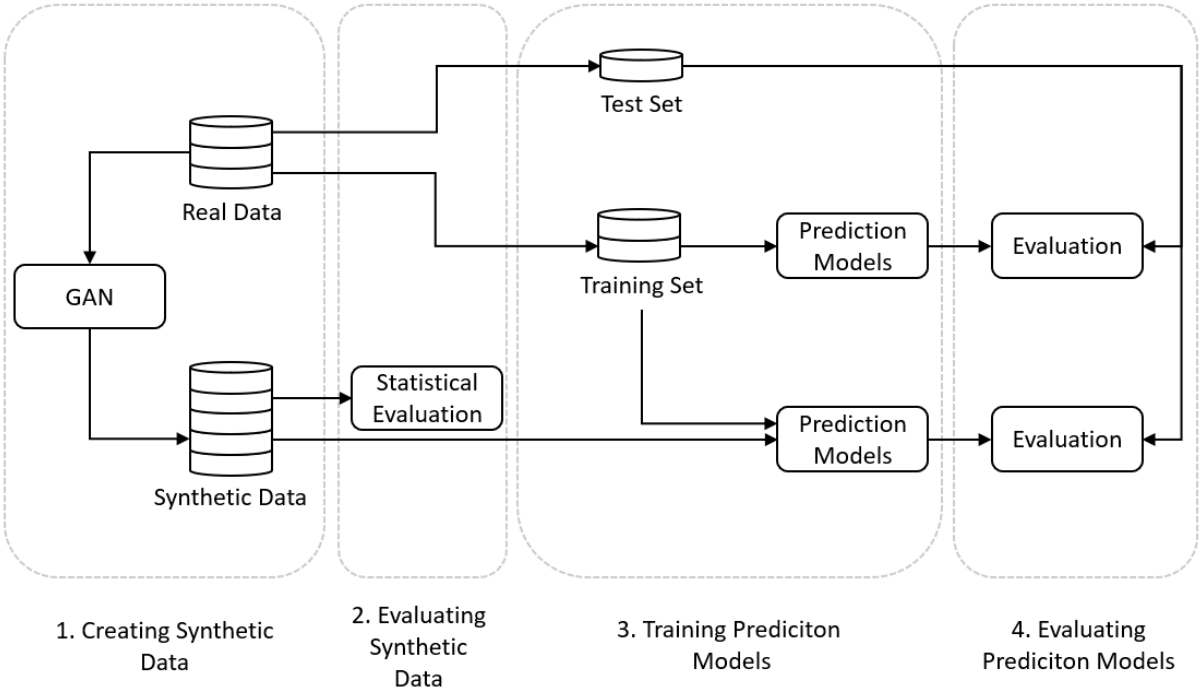

Figure 3: Schematic overview of the research approach

the real data.

To analyze whether the synthetic data can improve data analysis tasks, various supervised machine learning models are trained. The models aim to predict the change of delay (Δy) of the target train at station S by using the arrival delay of the target train (d_a) and the arrival delays of other feature trains in station S. Therefore, the real data set is divided into a training set (80%) and a test set (20%). The training set is used to train four different machine learning models (linear regression, AdaBoost, decision tree, MLP). During the training phase, the best performing parameters and input features are selected for each model using standard machine learning techniques. To evaluate the performance of the trained models, they are applied to the previously unseen data from the test set. The results are measured using the R^2 value to indicate how accurately the models can predict (Δy). Then, the previously used training set is combined with the synthetic data set. The result is a much larger training set with many more instances. The expanded training set is then used to train the same four machine learning models as before. The new models are then reapplied to the unseen data from the test set, which contains only real data and no synthetic data. The resulting R^2 values can be compared to the values from the previous step, when only the real data was used for model training. An overview of the described approach and the input data for both steps is shown in Figure 3.

The approach is applied to all three datasets for the target trains A, B, and C. For each dataset, four predictive models based on real data and four predictive models based on a combination of real and synthetic data are developed. The results of all 24 models are then compared and analyzed. If models trained on synthetic data produce better prediction results for the test set with real data than the models trained only on real data, the CWGAN could produce useful and realistic synthetic data that can also be used for various other analytic tasks. It is also possible to evaluate how different predictive models benefit from a larger training set, and which models can achieve the most performance improvements.

5 Developing the CWGAN Model

The CWGAN for creating the synthetic data sets is implemented in Python using the Tensorflow, Keras and Scikit-learn libraries. The basics of the implementation are oriented on the code example of Nash (2017). Like the standard GAN, the CWGAN contains a generator and a discriminator. Both of them are feedforward neural networks. The generator uses the original real data as input, and the number of neurons in the first layer for the generator is equal to the number of features in the real data set. The next 3 hidden layers of the network use 128, 256, and 512 neurons. The number of neurons in the last layer again is equal to the number of features in the input data. The discriminator network uses the original real data and the synthetic data created by the generator as input, and the number of features is equal to the number of neurons in the first layer. The next layers use 512, 256, and 128 neurons, and the last layer uses one neuron.

For the configuration of the CWGAN hyperparameters, we follow the suggestions of Gulrajani, Ahmed, Arjovsky, Dumoulin, and Courville (2017): The Adam optimizer is used with a learning rate of 0.0001, and the values for $\beta 1$ and $\beta 2$ are set to 0.5 and 0.9, respectively. The loss function is calculated using the Wasserstein distance, and the penalty parameter lambda is set to 10. The generator is updated once per iteration, and the discriminator is updated five times per iteration because it learns slower than the generator. The discriminator is pre-trained 100 times on the real data before the first iteration, so that it can keep up with the generator. The sample size is 128. This metric is used to define how many data instances are used in each iteration to train the generator and to evaluate the discriminator.

Then, the CWGAN is trained in many iterations. We use 150,000 iterations because we found that the CWGAN does not improve after this number of iterations for our data sets. In each iteration, the generator produces synthetic data, which is then passed to the discriminator. The discriminator attempts to distinguish between the real data and the synthetic data. Depending on the performance of the discriminator and the generator, both networks update the weights of the connections between the neurons before the next iteration begins. Every 100 iterations, the accuracy of the synthetic data is analyzed. For this purpose, an XGBoost classifier is used to distinguish between the real and the synthetic data. XGBoost stands for extreme gradient boosting and is a scalable tree boosting algorithm that can compute results for regression or classification problems very quickly (Chen & Guestrin, 2016). We use all the real data and the same amount of synthetic data for the XGB classifier. Half of this data set is then used to train the XGB classifier, and the other half is used as a test set to measure the performance. The ideal accuracy of this process would be 50%, as this means that the model cannot distinguish between real data and synthetic data. In this way, the different CWGAN models can be compared, and after all 150,000 iterations, the model with the best performance according to the XGB classifier is selected to create the synthetic data that is used for the following steps.

Figure4 shows the loss functions of the generator and the discriminator of the CWGAN for each data set. The x-axis depicts the number of iterations, and the y-axis depicts the value of the loss function. The loss function curves follow a similar path. At the start the curves indicate strong variation, and at around 2,000 iterations the curves stabilize and maintain a constant distance from each other. This means that neither the generator nor the discriminator can outperform the other. A stable state like this is a desirable result, as it indicates that both parts are improving equally.

Figure 5 depicts the moving average of the loss functions of the XGB classification models. The models attempt to distinguish between real and synthetic data and are applied after every 100th iteration of the CWGAN. The reason we do not apply the classification model after every CWGAN iteration is that it would significantly increase the computation time. An ideal result

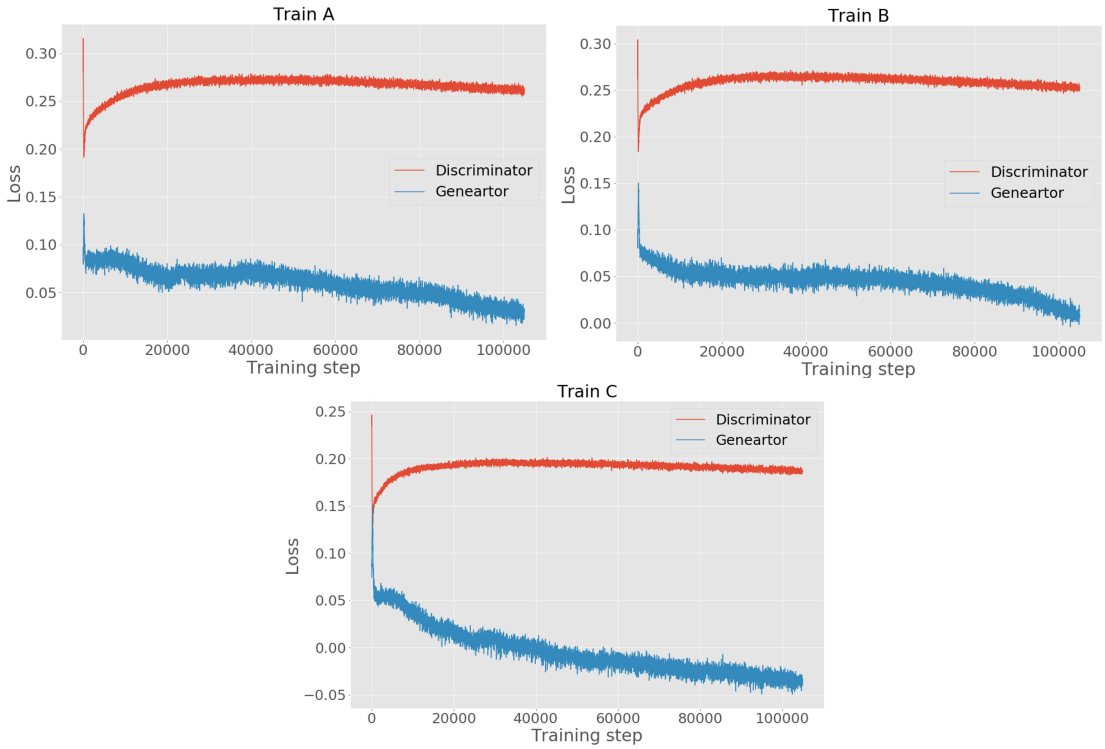

Figure 4: Loss functions of the generator and the discriminator

for the XGB loss function would be 50%, since the classifier then cannot distinguish between real and synthetic data and can basically only guess. This would mean that the CWGAN produces high-quality synthetic data. The figure shows that the performance of the classifier gets worse as the number of CWGAN iterations increases. This shows that the CWGAN is learning and improving during the iterations. It can also be seen that the performance does not improve after around 8,000 iterations.

The classification results derived from the XGB classifier are used as a benchmark to determine the best performing CWGAN model. The best model results in the lowest XGB accuracy, and this model can then be used to generate synthetic data with the highest possible data quality. The best classification results are shown in Table 3. The classification accuracy is 66% for the data set of target train A, 63.3% for the data set of target train B, and 62.1% for the data set of target train C. This shows that the quality of the synthetic data is at a similar level for all three use cases and that the classifier is only around 12% to 16% better than just guessing. This indicates an acceptable overall quality of the synthetic data which can be used for further analysis.

Table 3: XGB Classification results for real and synthetic data

Target Train	**XGB Accuracy**
A	66.6%
B	63.3%
C	62.1%

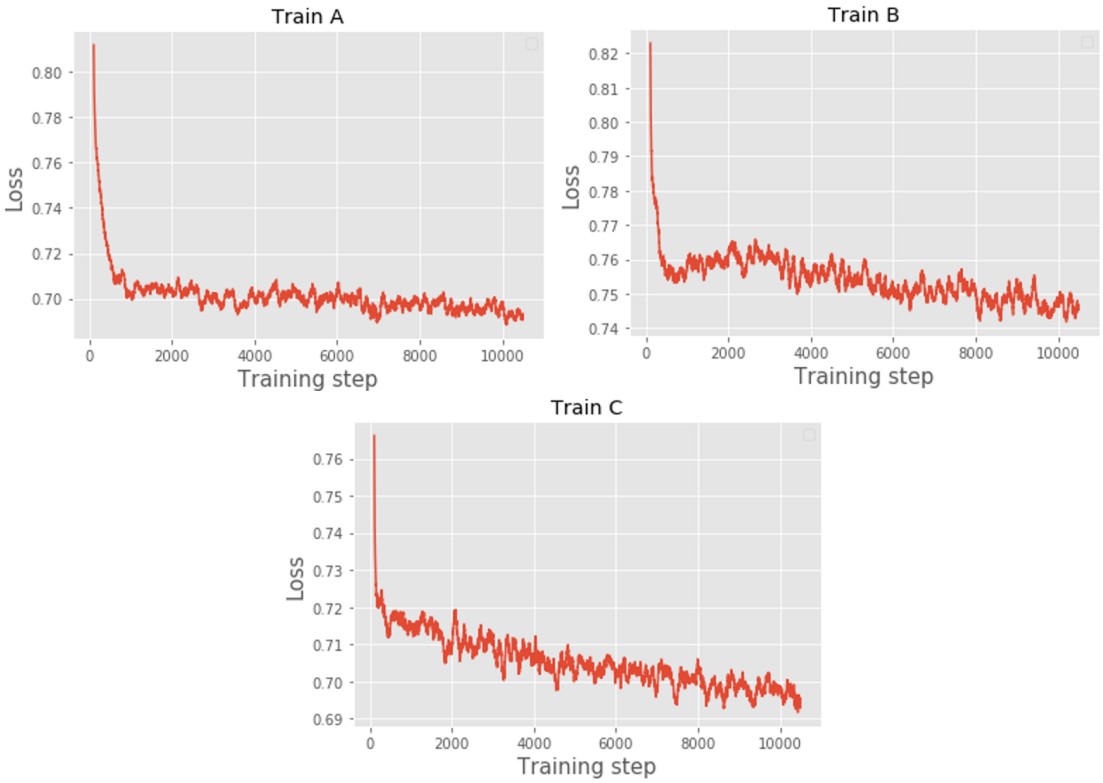

Figure 5: Moving average loss of the XGB classifier

6 Statistical Evaluation of the Synthetic Data Sets

The CWGAN model with the best performance for each data set, as described in chapter 5, is applied to generate three new synthetic data sets with 300,000 instances each. The model can theoretically create any number of instances, but the performance of the prediction models described in chapter 7 increases significantly with larger data sets. Therefore, the size of the synthetic data sets is limited for this paper. However, the synthetic data sets still have over 800 times as many instances as the original data sets.

The synthetic data sets are analyzed using statistical methods to evaluate their quality in more detail. The first analysis focuses on the distribution of the data. Therefore, the mean value and standard deviation are calculated for each column of the original data sets and for each column of the synthetic data sets. Ideally, the values for both metrics should be similar for the same columns in the real and synthetic data sets. Similar values indicate that both data sets have similar data distribution. Figure 6 shows a plot of the mean values and the standard deviation for all three data sets.

The upper plots in Figure 6 show the mean values, and the bottom plots show the standard deviation for all columns of the data sets. The values of the real data sets are shown on the x-axis, and the values of the synthetic data sets are shown on the y-axis. Every dot represents one column. If the dots lie exactly on the diagonal line, the values from the real and synthetic data sets are the same. The further away a dot lies from the line, the bigger the gap between real and synthetic data. If a dot is below the diagonal line, it means that the metric value of the corresponding column in the real data is higher than it is in the synthetic data. If a dot is above the diagonal line, it means that the metric value in the real data is lower than it is in the synthetic data.

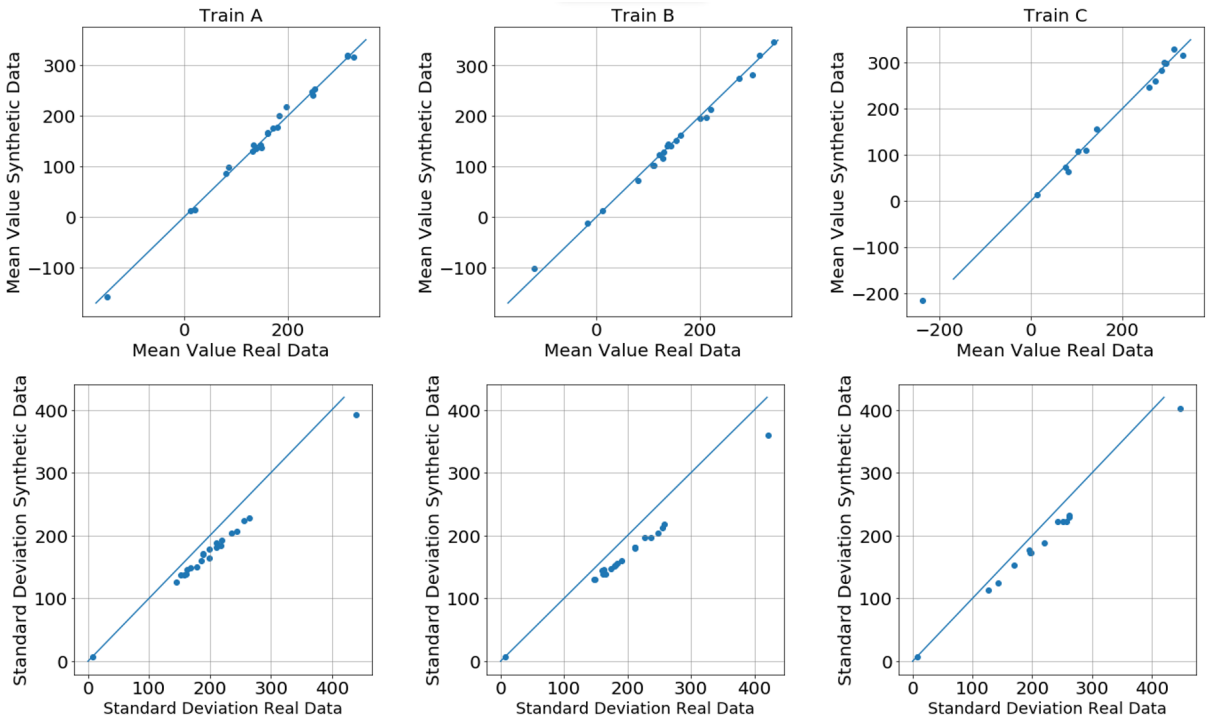

Figure 6: Comparison of the mean values and standard deviation from the real data with the synthetic data

The mean values are almost completely identical and differ only very little between the data sets. This shows that all columns in the synthetic data sets have similar mean values as the corresponding columns in the real data sets. The dot in the lower left of the upper plots represents the change in delay of the target train. It is the only negative value. This shows that on average, all three trains can reduce their delays during their dwell time at station S.

The values for the standard deviation differ somewhat more than the mean values. However, they are generally also very close to the diagonal line. It is noticeable that the standard deviation of the synthetic data is always somewhat lower than that of the real data. This is not a major problem, however, because it means that the synthetic data scatters less around the mean value. This is to be expected since the synthetic data basis is much larger. The plots show two outliers in the lower left and in the upper right corners. The left dot represents the temperature, which does not reach values as high as the arrival delays. The right dot represents the change in delay at station S of the target train. This value is very scattered in the data sets and therefore shown as an outlier in the graphs.

The correlations between all columns are calculated using Pearson Correlation Coefficient. The formula is shown in equation 4 where $\rho_{\mathrm{X,\,Y}}$ is the correlation coefficient for the variables X and Y, cov is the covariance and σ is the standard deviation.

$$\rho_{\mathrm{X,\,Y}} = \frac{cov(X,Y)}{\sigma_{\mathrm{X}}\sigma_{\mathrm{Y}}} \tag{4}$$

Figure 7 shows the correlation matrices for all columns of each data set. The top plots represent the real datasets, and the bottom plots represent the synthetic data sets. Each square represents the correlation between two columns. Darker shading indicates a strong negative correlation, and lighter shading indicates a strong positive correlation. The correlation values themselves

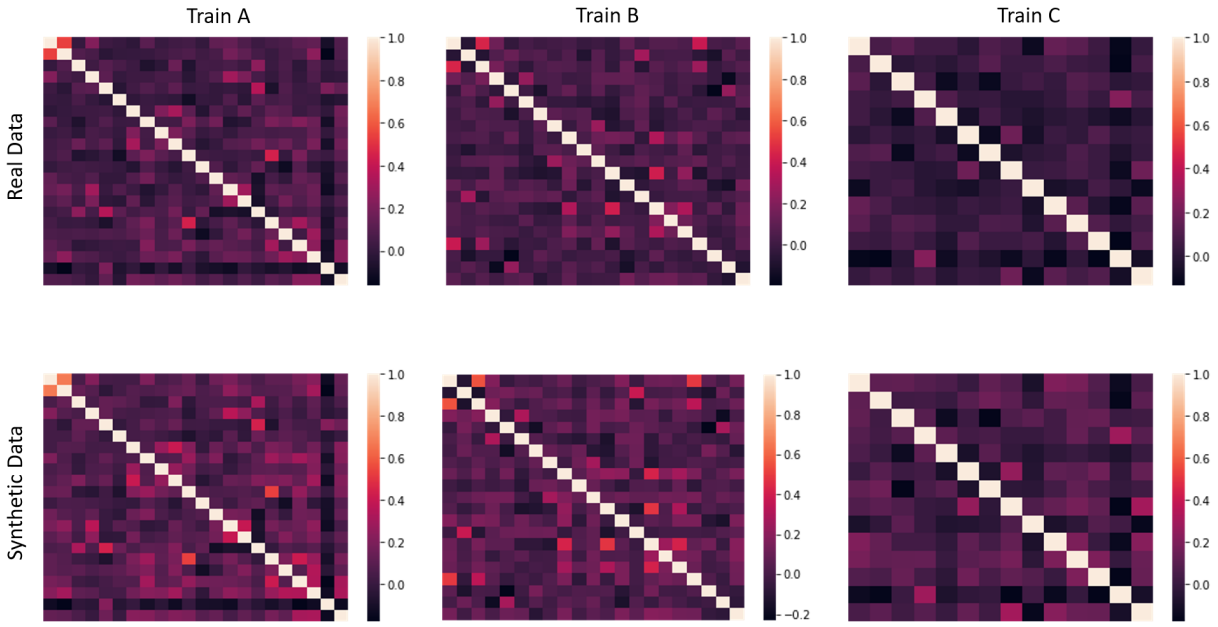

Figure 7: Correlation matrices for the real data sets and the synthetic data sets

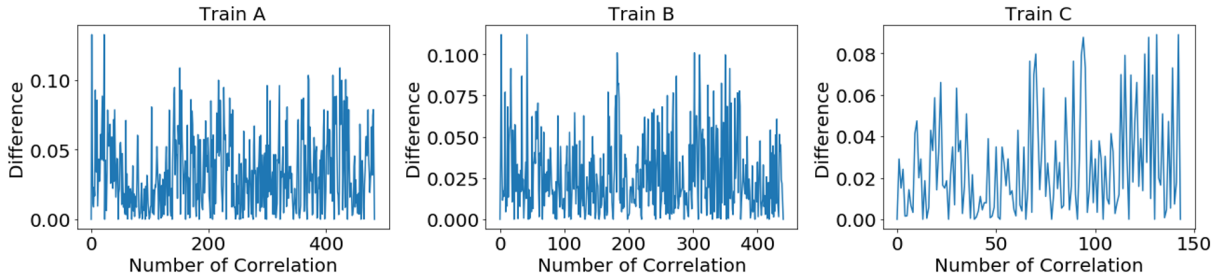

Figure 8: Difference in correlation values between real and synthetic data for every column combination

are not relevant to this analysis. What is important is whether the correlations in the real data sets are similar to the correlations in the synthetic data sets. This is because the synthetic data should inherit existing dependencies from the real data. It can be seen that the general structure of the correlations between the original and the synthetic data are very similar, and each upper plot looks almost like the corresponding lower plot. The most considerable difference is that the correlations are a little stronger in the synthetic data set. This is interesting because it might be helpful for the following machine learning process.

Figure8 shows the numerical differences between the correlation values for the real data set and the synthetic data set. The differences are calculated by determining the distance between each correlation value of the real data set and the corresponding correlation value of the synthetic data set. Since the correlation values can range from -1 to +1, the largest possible difference between two values is 2. It can be seen that the differences for all three data sets are small and fluctuate around 0. The biggest differences for each data set are 0.13 for A, 0.11 for B, and 0.08 for C. This confirms the results shown in the correlation matrices and shows the close similarity of the real and synthetic data sets.

7 Developing the prediction models

The prediction models are implemented in Python using the Scikit-learn library. The real data sets are each split into a training set containing 80% of the instances and a test set containing 20% of the instances. Then, different enhanced training sets are created by adding instances of the synthetic data to the real data training sets. All instances in training sets with synthetic data are randomly shuffled. This results in eleven different training sets for each train: one training set containing only real data and ten training sets containing real data with 1,000, 2,000, 5,000, 25,000, 50,000, 100,000, 150,000, 200,000, 250,000, and 300,000 synthetic data instances, respectively. The test set remains unchanged throughout the process and always contains the same 20% of the instances from the real data set. Four different supervised machine models (linear regression, AdaBoost, random forest, and MLP) are trained on the different training sets to predict the change of delay (Δy) at station S.

7.1 Linear Regression

A linear regression is used to model the influence of a set of independent variables on a dependent variable. The goal is to find a linear relationship between the independent variable and the target variable (Fahrmeir et al., 2009). We use the model as a baseline reference because it is easy to train and to implement since it requires no hyperparameter tuning.

7.2 AdaBoost

AdaBoost is an ensemble learning algorithm. The general idea is to combine the outputs of multiple weak learners into one strong learner that gives a final output based on the weighted outputs of the weak learners. The classifiers learn from each other, and the algorithm builds up one strong classifier over many iterations (Freund & Schapire, 1996).

7.3 Random Forest

Random forest is another ensemble learning algorithm. The model consists of several different decision trees. Each decision tree makes a prediction, and the evaluation of all individual outcomes is combined into the final prediction. Random forest models are usually more robust against outliers than AdaBoost models (Breiman, 2001).

7.4 Multilayer Perceptron

An MLP is a feedforward neural network. It consists of an input layer and an output layer and one or more hidden layers in between. Each layer consists of neurons with a nonlinear activation function. The neurons are connected through adaptive weights and feed information to each other. MLP models can learn non-linear dependencies, but they require a lot of parameter tuning and need a lot of training data (Arouri et al., 2014).

During the training process of the models, hyperparameters must be selected (except for the linear regression model). Hyperparameters are parameters that are not derived in the training process but must be specified beforehand. To find the best hyperparameters for each model, a grid search is performed. Grid search is an exhaustive search method that iteratively applies all parameter combinations from a given parameter set. For each parameter set, cross-validation is performed to find the best combination for each model.

In addition, feature selection is performed to further improve the performance of the models. The goal is to identify the most relevant features in the data set and include only those in the

prediction model. This is important because most likely not all arrival delays in the data set are relevant for predicting the delay change of the target train. Therefore, recursive feature elimination is used. In this method, the least useful features are eliminated in many iterations until only a subset of the best-performing features remains. To determine the features to be eliminated, a feature importance parameter is calculated, which indicates how important a feature is to the prediction.

After the training of each model for each training data set, the performance of the models is evaluated using the test data set. To measure the performance, the R^2 value is used. The value measures how well the feature variables can explain the variance of the prediction variable. The value is always between 0% and 100%, with higher values meaning that the model can better explain the variations of the dependent variable. The mathematical formula is shown in equation 5, where y_i represents the actual value, $\hat{y}_i$ represents the prediction value, and $\bar{y}_i$ represents the mean value.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \tag{5}$$

The four different machine learning models are trained on the training sets containing only real data and evaluated against the test sets. Then, the models are trained on the training sets that include different amounts of synthetic data, and their performances are measured using the unchanged test sets. The resulting R^2 values are shown in Figure 9. The x-axis shows the number of synthetic data rows added to the training set. The y-axis shows the R^2 values. Each line in the graphic represents one machine learning model.

When only the small sets of real training data are used, the prediction performances of the models are rather low. For train A, the AdaBoost model performs best with an R^2 value of 8.4%. For train B, the random forest model performs best with an R^2 value of 12.9%. For train C, the random forest model also leads to the best predictions, with an R^2 value of 7.3%.

It can be seen that the performance of the model increases steadily as more synthetic data is added to the training sets. For trains A and B, adding only a few hundred synthetic instances leads to significant performance improvements of all models. For train C, the MLP model and the random forest model provide better predictions after around 5,000 instances are added. For all three trains, it can be seen that the MLP model and the random forest model can benefit from larger training sets, as their performance continuously increases. The linear regression model and the AdaBoost model can only marginally benefit from the additional data and reach their peak performances very early. After about 300,000 added instances, none of the models can further increase the prediction accuracy, indicating that the best possible results were found.

For train A, the best R^2 value is 63.9% and is achieved by the MLP model with 200,000 synthetic data instances. For Train B, the best R^2 value is 57.1% and is achieved by the random forest model with 250,000 synthetic data instances. The MLP model performs only slightly worse, reaching a peak R^2 value of 55.5%. For Train C, the best R^2 value is 63.8% and is achieved by the MLP model with 200,000 synthetic data instances. A summary of the performance values is shown in Table 4.

8 Conclusions

In this paper, we have shown how a CWGAN can be used to improve railway analysis. Based on a real data set of Deutsche Bahn, the proposed CWGAN can generate new synthetic data instances. In this way, the data basis can be significantly increased without compromising the data quality. The generated synthetic data instances are very similar to the original instances in terms of data structure and distribution. Neither the applied statistical metrics nor the

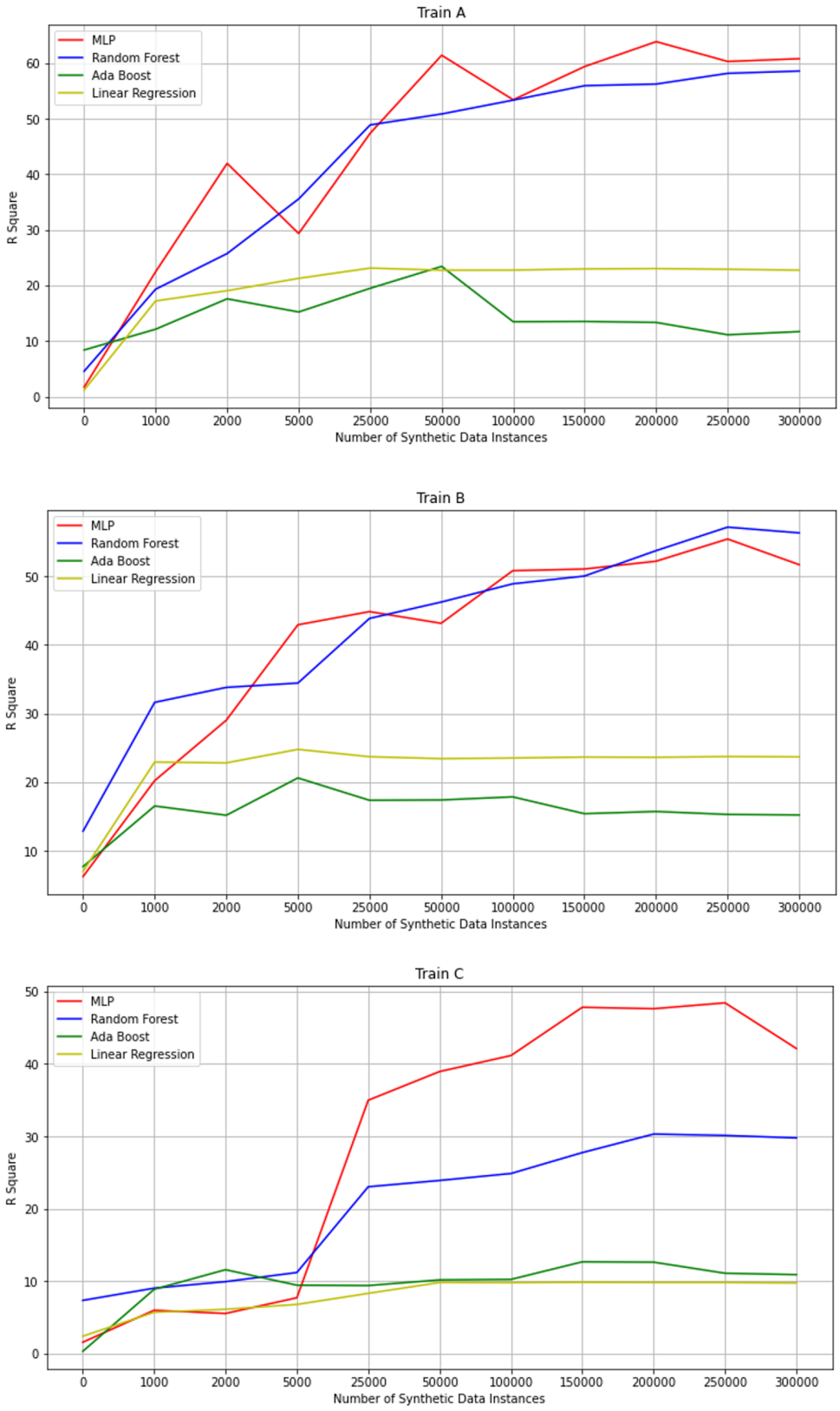

Figure 9: R^2 values for the predicitions

Table 4: R^2 values of the best performing models

Target Train	Best R^2 value without synthetic data	Best R^2 value with synthetic data	Number of synthetic data instances
A	8.4%	63.9%	200,000
B	12.9%	57.1%	250,000
C	7.3%	63.8%	200,000

machine learning classification model can easily distinguish between the real data set and the synthetic data set. The synthetic data show almost no deviations from the real data, neither in the mean values per column nor in the standard deviation per column. The correlations between all columns are also almost identical between the real data set and the synthetic data set. The applied XGB classification model had great difficulties distinguishing between real and synthetic instances, indicating the high quality of the generated data. The classification model achieved an accuracy of about 65%, which is only slightly better than just pure guessing.

To further demonstrate the high quality of the generated data, we have shown how train delay predictions can be improved by using synthetic data instances when training the models. The results show that the prediction models can achieve significantly better R^2 values when trained on a large data set of synthetic data compared to the R^2 values when trained only on a small set of real data. Since only real data is used to measure model performance, these results clearly demonstrate the effectiveness of the generated data instances. It can also be seen that a larger data set allows for more complex machine learning models. While the MLP model performed rather poorly with the small training set, it significantly outperformed the less complex linear regression and AdaBoost models when trained on a large synthetic data set. Interestingly, the random forest model also achieved high-performance values and was also able to benefit from the increased training set.

The presented CWGAN and the prediction models were applied to three use cases, obtaining similar positive results. This confirms that railway analysis can benefit from data-generative methods like GANs, and their application should be discussed for other scenarios as well. While similar approaches have been successfully applied to image data for some time in the literature, use cases for applying GANs to numerical data sets are rarely presented in the literature. In particular, the application of GANs for railway analysis projects is a new area of research whose potential has not yet been fully exploited. Another application for future research projects could be the use of synthetic data to address confidentiality issues. For example, if the data used for a research project is confidential, perhaps a GAN can generate synthetic data that is not confidential and can be more easily published. This also applies to railway analysis, since most railway companies do not want to publish their trip data. However, this still needs to be analyzed and discussed in more detail in future work.

References

Ghofrani, F., He, Q., Goverde, R. M. P., & Liu, X. (2018). Recent applications of big data analytics in railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, *90*, 226–246. https://doi.org/10.1016/j.trc.2018.03.010

Büker, T., & Seybold, B. (2012). Stochastic modeling of delay propagation in large networks. *Journal of Rail Transport Planning & Management*, *2*, 34–50. https://doi.org/10.1016/j.jrtpm.2012.10.001

Yuan, J. (2006). *Stochastic modeling of train delays and delay propagation in stations* [Dissertation, Eburon Academic Publishers, Delft].

Rößler, D., Reisch, J., Hauck, F., & Kliewer, N. (2021). Discerning primary and secondary delays in railway networks using explainable ai. *Transportation Research Procedia*, *52*, 171–178. https://doi.org/10.1016/j.trpro.2021.01.018

Gorman, F. M. (2009). Statistical estimation of railroad congestion delay. *Transportation Research Part E: Logistics and Transportation Review*, *45*, 446–456. https://doi.org/10.1016/j.tre.2008.08.004

Goverde, R. (2005). *Punctuality of railway operations and timetable stability analysis* [Doctoral dissertation, Delft University of Technology] [Dissertation, Delft University of Technology].

Corman, F., & Kecman, P. (2018). Stochastic prediction of train delays in real-time using bayesian networks. *Transportation Research Part C: Emerging Technologies*, *95*, 599–615. https://doi.org/10.1016/j.trc.2018.08.003

Berger, A., Gebhardt, A., Müller-Hannemann, M., & Ostrowski, M. (2011). Stochastic delay prediction in large train networks. *OASIcs-OpenAccess Series in Informatics*, *20*. https://doi.org/10.4230/OASIcs.ATMOS.2011.100

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., Mazzino, N., & Anguita, D. (2018). Train delay prediction systems: A big data analytics perspective. *Big Data Research*, *11*, 54–64. https://doi.org/10.1016/j.bdr.2017.05.002

Goncalves, A., Ray, P., Soper, B., Stevens, J., Coyle, L., & Sales, A. P. (2020). Generation and evaluation of synthetic patient data. *Bmc Medical Research Methodology*, *20*. https://doi.org/10.1186/s12874-020-00977-1

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *Arxiv Preprint*. %5Curlhttps://arxiv.org/abs/1312.6114

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, *27*. https://doi.org/10.1145/3422622

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, *321*, 321–331. https://doi.org/10.1016/j.neucom.2018.09.013

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1710.10196

Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1809.11096

Weng, L. (2019). From gan to wgan. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1904.08994

Metz, L., Poole, B., Pfau, D., & Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1611.02163

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1411.1784

Douzas, G., & Bacao, F. (2018). Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with Applications*, *91*, 464–471. https://doi.org/10.1016/j.eswa.2017.09.030

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1701.07875

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1704.00028

Shenvi, P., Samant, N., Kumar, S., & Kulkarni, V. (2019). Implementation of interpolation in credit card fraud detection. In V. S. Reddy, V. K. Prasad, J. Wang, & K. T. V. Reddy (Eds.), *Soft computing and signal processing* (pp. 125–136). Springer Singapore. https://doi.org/10.1007/978-981-15-2475-2_12

Zheng, M., Li, T., Zhu, R., Tang, Y., Tang, M., Lin, L., & Ma, Z. (2020). Conditional wasserstein generative adversarial network-gradient penalty-based approach to alleviating imbalanced data classification. *Information Sciences*, *512*, 1009–1023. https://doi.org/10.1016/j.ins.2019.10.014

Ba, H. (2019). Improving detection of credit card fraudulent transactions using generative adversarial networks. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1907.03355

Van Buuren, S., & Groothuis-Oudshoorn, K. (2011). Mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, *45*, 1–67. https://doi.org/10.18637/jss.v045.i03

Nash, C. (2017). Generative adversarial networks (GANs) for credit card fraud data [Last accessed: 2022-11-01]. %5Curl%7Bhttps://github.com/codyznash/GANs%5C%5Ffor%5C%5FCredit%5C%5FCard%5C%5FData%7D

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in Neural Information Processing Systems*, *30*. https://arxiv.org/pdf/1704.00028

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Fahrmeir, L., Kneib, T., & Lang, S. (2009). *Regression: Modelle, Methoden Und Anwendungen*. Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-01837-4

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, 148–156.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/A:1010933404324

Arouri, C., Nguifo, E. M., Aridhi, S., Roucelle, C., Bonnet-Loosli, G., & Tsopze, N. (2014). Towards a constructive multilayer perceptron for regression task using non-parametric clustering. a case study of photo-z redshift reconstruction. *Arxiv Preprint*. https://doi.org/10.48550/ARXIV.1412.5513

Diskussionsbeiträge - Fachbereich Wirtschaftswissenschaft - Freie Universität Berlin
Discussion Paper - School of Business & Economics - Freie Universität Berlin

2026 erschienen:

2026/1 Hundsdoerfer, Jochen; Löwe, Maren: How Do Value Added Taxes Affest Wages and Labor?
FACTS

2026/2 Gril, Lorena; Rendtel, Ulrich: Mapping High-Income Taxpayers in Berlin Using Kernel-Smoothed Proportions from Aggregated Georeferenced Data
Economics

2026/3 Gril, Lorena; Hossain, Md Jamal; Tzavidis, Nikos; Rendtel, Ulrich: Kernel density estimation under masking of geolocations with applications to DHS data
Economics

2026/4 Corneo, Giacomo: Narratives as Separating Equilibria: on the Origins of the Ukraine War
Economics

2026/5 Tauscher, Fabian; Kari, Arthur; Gersch, Martin: From Regulation to Realisation: Secure Processing Environment for the European Health Data Space (in Vorbereitung)
Information Systems

2026/6 Prummer, Anja; Nava, Francesco: Divisive By Design: Shaping Values in Optimal Mechanisms. – 2., überarb. Aufl. (in Vorbereitung)
Economics