A Supplementary Material

Supplementary material for *Simulating drug effects on blood glucose laboratory test time series with a conditional WGAN* by Alexandre Yahi and Nicholas P. Tatonetti.

A.1 Background: Generative Adversarial Networks (GANs)

GANs are an implicit density estimation framework proposed by Goodfellow et al. in 2014 [9]. Implicit generative models are defined as "a stochastic mechanism whereby the data are generated" [7]. They take as an input a latent variable z and map it using a deterministic function $\mathcal{G}_\theta$ defined on $\mathbb{R}^m \rightarrow \mathbb{R}^d$ using parameters $\boldsymbol{\theta}$. In their original formulation, GANs consist of a generator network G that maps random latent variables to synthetic samples in order to fool a discriminator network D that classifies real and synthetic samples. D and G play a two-player minmax game with the following value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

However this framework has known numerous improvements and variations over the past five years, to make up for the initial instabilities of the so-called "Vanilla GAN" known to have an arduous training, with several papers trying to address the issues by suggesting training techniques for GANs[18] or attempts at principled approaches [2]. The f-GAN by Nowozin et al. [16] demonstrated that Goodfellow's formulation with a Jensen-Shannon divergence was a special case of a broader and more general variational divergence estimation approach. Given two distributions P and Q that possess, respectively, an absolutely continuous density function p and q with respect to a base measure $\mathrm{d}x$ defined on the domain $\mathcal{X}$, an f-divergence is defined by:

$$D_f(P||Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x$$

where the generator function $f : \mathbb{R}_+ \rightarrow \mathbb{R}$ is a convex, lower-semicontinuous function satisfying $f(1) = 0$. f-divergences include Kullback-Leibler, Reverse KL, Pearson χ^2, Squared Hellinger and Jensen-Shannon.

Yet, it's another family of distance measures that has showed significant improvements in GAN training stability: integral probability metrics (IPMs)[15]. Given $\mathcal{F}$ a set of functions from $\mathcal{X}$ to $\mathbb{R}$, we can define:

$$d_{\mathcal{F}}(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

as an integral probability metric associated with the function class $\mathcal{F}$. Depending on this function class, the expression of that distance measure can vary widely. The intuition being IPMs is that they measure the distance between probability measures via the largest discrepancy in expectation over a class of "well behaved" witness functions.

The Wasserstein GAN [3] was the first GAN model proposed with such an IPM, using the Wasserstein-1 or Earth Mover (EM) distance:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g} \mathbb{E}_{(x,y) \sim \gamma}[||x - y||]$$

That can be translated using the Kantorovich-Rubinstein duality [19] into:

$$W_{\mathcal{F}}(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

where f is the family of 1-Lipschitz functions $f : \mathcal{X} \rightarrow \mathbb{R}$. Having it be the family of K-Lipschitz function does not matter because it would just multiply $W_{\mathcal{F}}(\mathbb{P}_r, \mathbb{P}_\theta)$ by K.

Enforcing this constraint in the GAN training is the hardest part, and the authors resolved to clamping the weights of the neural network within a fixed box (e.g., $\mathcal{W} = [-0.01, 0.01]^l$) after each gradient update. In their formulation, the discriminator becomes the *critic* that learns a witness function used to approximate the Wasserstein-1 distance between synthetic and real samples.

However, the way the WGAN enforced the K-Lipschitz requirements, using weight clipping, seemed to still cause issues in the training process and the quality of the outputs. In response to these issues, Gulrajani et al. [10] proposed a gradient penalty GAN, or WGAN-GP, showing the issues with weight clipping and how a penalization of the norm of the critic gradient with respect to its input addressed more efficiently the Lipschitz requirement.

Formally, the WGAN-GP defines a new objective function:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})]}_{\text{Original WGAN loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_{\tilde{\boldsymbol{x}}}}[(||\nabla_{\tilde{\boldsymbol{x}}} D(\tilde{\boldsymbol{x}})||_2 - 1)^2]}_{\text{WGAN-GP gradient penalty}}$$

Where $\mathbb{P}_{\tilde{\boldsymbol{x}}}$ is defined by sampling uniformly along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$, motivation by the fact that the authors demonstrated the optimal critic contains straight lines with gradient norm 1 connecting coupled points from $\mathbb{P}_r$ and $\mathbb{P}_g$.

More recently, variation of the WGAN-GP was proposed, using a Lipschitz penalty that consists of taking the maximum of the gradient penalty or zero, forcing it to be positive or null. [17]

In healthcare, there is only a handful of GAN applications to date, ranging from discrete categorical variables generation with medGAN [6], and its improved version using a WGAN [4], to natural language generation for EHR notes [14]. Two studies modeled time series, but in the context of ICU patients who are highly monitored, yielding high frequency regularly sampled time series about the patients states and their medication dosages. Esteban et al. [8] developed a recurrent conditional GAN (RCGAN) using an LSTM architecture and working with a type of time series very different from laboratory tests time series: oxygen saturation measured by pulse oximeter (SpO2), heart rate (HR), respiratory rate (RR) and mean arterial pressure (MAP) from the Philips eICU database [1]. Recently, Wang et al.[20] proposed a coupled GAN (SC-GAN) with two generators that can simulate a patient state time series (e.g., vitals, highly sampled blood measurements) along with the dosage of a specific medication, over windows of 4 hours and using k-nearest neighbors imputation when values were missing, and training with MIMIC-III ICU data [12]. None of these models focused on irregularly sampled laboratory tests with a focus on simulate drug effects conditionally. To this end, we believe we propose in this paper the first conditional GAN that models drug effects on irregular laboratory test time series.

A.2 Forecasting task

We developed forecasting models to evaluate different data representations and neural network architectures. These forecasting models are used in the main experiments for the "train on synthetic test on real" (TSTR) evaluation tasks of the generative adversarial networks (GANs), and in the applications with inference and data augmentation.

For each time series of 10 values, the first nine values where used as features, and the last value was used as the target of the regression. We randomly split each dataset between a training set (90%) and a testing set (10%). We kept for each dataset a consistent split between analogous experiments: the regular time series regression study has the same train/test split as the generative adversarial networks (GANs) experiment on regular time series, and the irregular time series dataset are the same between the regression and the conditional GAN models for consistency between experiments.

The data were pre-processed by truncating at the 1 and 99 percentiles for quality control to get rid of extreme or spurious values, and then scaled between 0 and 1. Time series with measurements that did not respect these criteria were removed entirely.

For every model, we tuned the hyperparameters with cross-validation on the training set: for machine learning models we used a 10-fold cross validation where a 10% random validation set is taken from the training set to choose the best performing hyperparameters. The deep learning models experimental pipeline is described in details in the next section. Once the hyperparameters of a model are set, the model is trained on the complete training set, and evaluated once on the testing set.

All the regression models rely on a mean square error loss function as preliminary studies showed that it was the best objective loss with regard to testing evaluation metrics.

The main evaluation metrics of this forecasting tasks were the MSE and the mean absolute loss (MAE). While the former emphasizes the larger errors, the latter provides a better idea of the error in real units of the measurement.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}|$$

where y is the true value and $\hat{y}$ is the predicted value.

A.2.1 Regular time series: Models

Baseline and machine learning models Often, deep learning models are evaluated relatively to one another without baseline. A number of misconceptions have resulted from these approaches that omit to compare advanced models with naive approaches and classic machine learning algorithms. For these forecasting experiments, we set the naive baseline to be a "repeat" prediction: this heuristic predicts the next value to be identical to the last value of the time series used for training.

We then selected two popular machine learning algorithms: linear regression, and random forest regression [5]. While the former is considered to be the simplest approach, generalized linear models (GLMs) have proven to be hard to beat in numerous prediction problems. Random forest regression illustrates in comparison the performances of an ensemble methods relying on bagging (i.e., bootstrap aggregation) built on the decision tree algorithm.

Deep learning models The first deep learning model to be evaluated was the multi-layer perceptron (MLP), a fully connected neural network that sets the baseline for deep models. The architecture was selected as follow:

- First layer:
  - Fully-connected layer (I, N)
  - Batchnorm
  - Leaky ReLU activation
- Middle layers (optional):
  - Fully-connected layer (N, N)
  - Batchnorm

  - Leaky ReLU activation

- Last layer
  - Fully-connected layer (N, N)
  - Batchnorm
  - Leaky ReLU activation
  - Dropout
  - Fully-connected layer $(N, 1)$

Where N is the width of the network, and I is the size of the input. For simplicity of tuning, the network width was constant once set for all layers except for the input and output layers that follow the structure given above (Figure 1).

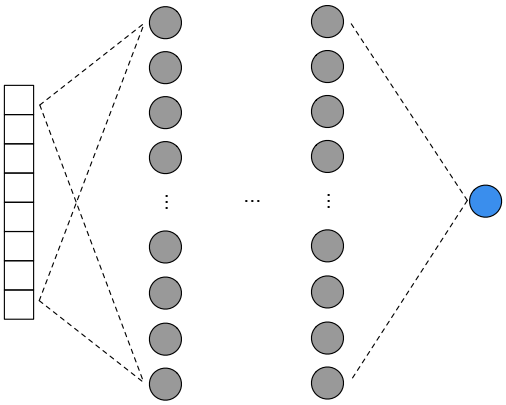


Figure 1: Multi-layer peceptron with time series input. Hidden layers are in grey, and the output of the forecasting model is in blue. Every hidden layer has a Batchnorm layer, and leaky ReLU activation function, and the last hidden layers contains a dropout mechanism.

Due to the sequential nature of the input, I compared MLP to long short-term memory (LSTM) cells.

The LSTM architecture [11] is a type of RNN that models both short and long term dependencies in data. LSTM tries to solve the vanishing gradient problem by not imposing any bias towards recent observations, but it keeps constant error flowing back through time.

The LSTM regressor I evaluated had the following general structure: an LSTM network, followed by a fully connected layers with BatchNorm, Leaky ReLU activation and Dropout, and a fully connected layer without activation that computes the prediction, similar to the last layer of the MLP model (Figure 2).

We evaluated different models of LSTMs where the hidden vector h_t was used either completely or at the last time step, where dropout between stacked layers was authorized or not, and where the size of the fully connected layer following the LSTM network varied 1.

The last deep learning model evaluated was the convolutional neural networks (CNN), pioneered by Yann LeCun in 1990 [13]. It consisted of two parallel CNNs with the following structure:

- CNN 1:
  - Conv1d: kernel size=2, padding=0
  - BatchNorm(scaleCoefficient)
  - ReLU

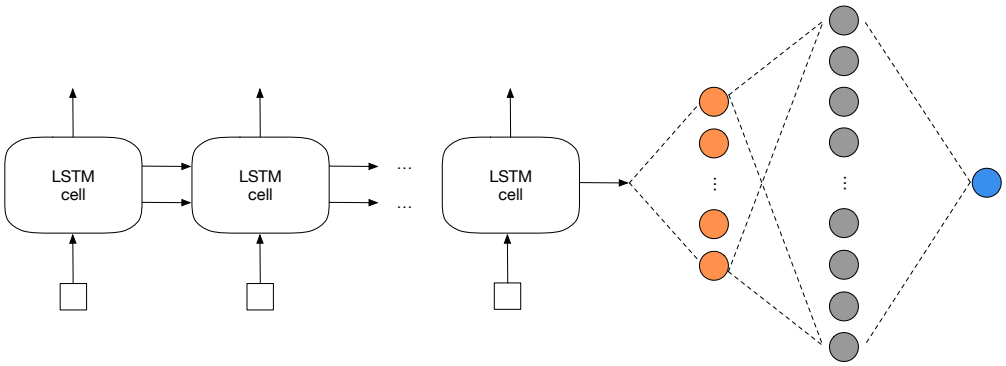


Figure 2: Long Short Term Memory (LSTM) architecture, where the LSTM cells are followed by a fully connected layer with Batchnorm, leaky ReLU and dropout.

	Dropout in LSTM*	Hidden vector h_t used	Size fully connected layer (vs. hidden size)
Model 1	Yes	Last	1x
Model 2	Yes	Last	2x
Model 3	Yes	All	1x
Model 4	Yes	All	2x
Model 5	No	Last	1x
Model 6	No	All	1x

Table 1: Different types of LSTM architectures evaluated. *No dropout in single layer LSTMs.

  - MaxPool
  - Conv1d: kernel size=2, padding=1
  - BatchNorm (2 x scaleCoefficient)

- CNN 2:
  - Conv1d: kernel size=3, padding=0
  - BatchNorm (scaleCoefficient)
  - ReLU
  - MaxPool
  - Conv1d: kernel size=3, padding=1
  - BatchNorm (2 x scaleCoefficient)

The outputs of these two parallel CNNs were then concatenated and passed into a fully connected layer stack similar to the last part of the MLP:

- Fully-connected layer(N, N)
- Batchnorm
- Leaky ReLU activation
- Dropout

- Fully-connected layer$(N, 1)$

Where N is the size of the concatenated vector from the two parallel CNNs (Figure 3).

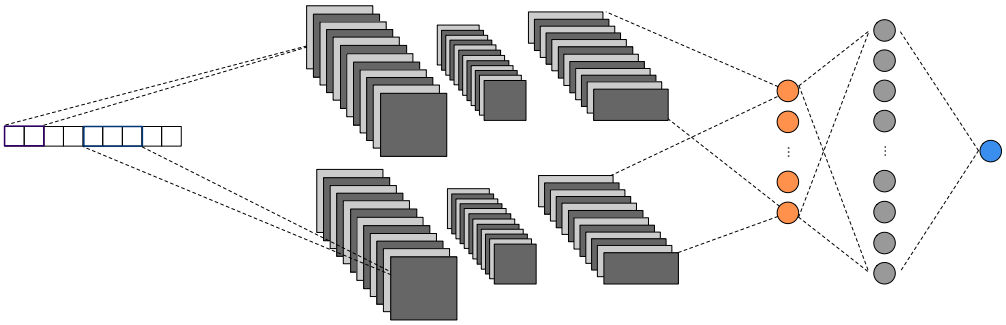


Figure 3: Convolutional Neural Networks (CNNs), where two CNNs with different filter sizes, two and three, have their outputs concatenated and fed into a fully connected layer with Batchnorm, leaky ReLU and dropout.

A.2.2 Irregular time series: Models

While regular time series were useful to get a baseline free of time interval variability by controlling sampling rate, laboratory test time series are usually not regular as patients do not get measurements every single day during their hospitalization. The frequencies vary between patients, and depend on the different stages of care. Therefore it seems important to evaluate models that can deal with these irregular time series. Another important aspect of these time series is their context, in the form of all the other types of clinical events that occur during their time span, such as diagnoses, procedures, or drug exposures. To this end, the previously introduced models were adapted to add two types of auxiliary data: time intervals and binary drug exposures.

Time interval data were obtained by indexing measurements in days elapsed since the beginning of the visit during which they were performed. I then computed the intervals in days between two measurements as $\Delta(n) = t(n) - t(n-1)$ where $t(n)$ is the day index of the measurement n and $t(0) = -1$. The heatmap of average time intervals per time series for irregualar glucose lab measurements is available in figure 4.

Drug exposures were represented by a binary vector that indicates if the patient was exposed to the given drug concept during any of the days where the first nine measurements (i.e., the measurements used to train models) were performed.

Classic machine learning models and MLP integrated these auxiliary data by concatenating them to the measurement features, directly at the input of the models. LSTM and CNN followed a different approach. Time intervals were added as a second channel of features for LSTM and CNN, as they are associated to measurements in a pair-wise fashion. Therefore the input in these cases went from $1x9$ to $2x9$. Binary drug exposure information are not specifically associated to a given measurement, therefore these data were concatenated at the output of the CNN module, and LSTM module respectively, at the input of the final stack of fully connected layers.

A.2.3 Regular time series: forecasting results

We compared the forecasting performances of the machine learning and deep learning models to the repeated measure baseline that assumes that the 10th value is identical to the 9th. While this baseline resulted in a 1502.62 MSE (25.15 MAE), machine learning models outperformed it with performances about

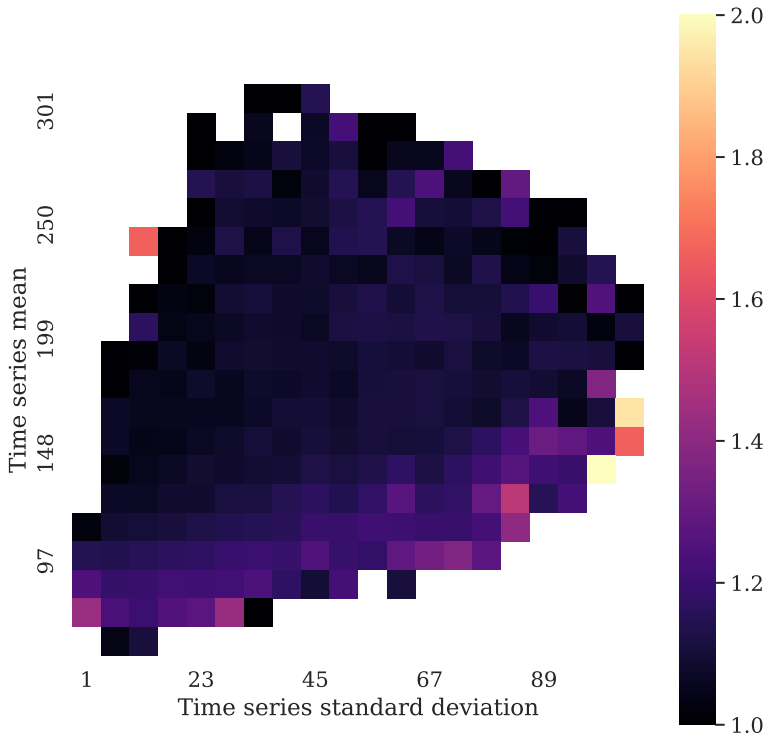


Figure 4: Time intervals heatmap of regular glucose lab measurements time series represented by their standard deviations and mean.

30% better. Linear regression was the best machine learning methods with an MSE of 1058.43. Among deep learning models, LSTM was on average worse than linear regression, and at best performed the same. The MLP model had the best average MSE, lower than the linear regression error by 3 points, and performing as well as 7 points lower than linear regression. Finally, CNNs displayed MSEs that on average were very similar to MLP, with a best performance about 1 point better than MSE due to a larger variance (Table 2). The heatmap of the regression MSE are illustrated in figure 5.

The best MLP model for regular glucose time series had the following hyperparameters (Figure 6):

- Batch size: 64
- Dropout: 0.5
- Dimensions: 2 layers of width 64

Model	best MSE	best MAE	avg. MSE (±SD)	avg. MAE (±SD)
Repeated (baseline)	1502.62	25.15	–	–
Linear regression	1058.43	21.99	–	–
Random Forest Regression	1083.69	22.45	–	–
MLP	1051.11	21.85	**1055.10 (± 3.328)**	**21.97 (± 0.069)**
LSTM	1058.51	**21.84**	1065.92 (± 5.928)	22.17 (± 0.258)
CNN	**1049.93**	21.93	1055.63 (± 5.625)	22.04 (± 0.106)

Table 2: Evaluation of the forecasting model with regular time series for blood glucose (2345-7)

Model	best MSE	best MAE	avg. MSE (±SD)	avg. MAE (±SD)
Repeated (baseline)	1461.90	25.14	–	–
Linear regression	1064.70	22.22	–	–
Random Forest Regression	1084.54	22.59	–	–
MLP-reg	1053.20	22.06	1056.34 (± 2.795)	22.15 (± 0.077)
MLP	1058.06	22.06	1066.46 (± 7.683)	22.30 (± 0.134)
LSTM	1067.45	22.10	1069.53 (± 2.214)	22.33 (± 0.149)
CNN	**1049.49**	**21.91**	**1054.58 (± 2.838)**	**22.09 (± 0.106)**

Table 3: Evaluation of the forecasting model with irregular time series for blood glucose (2345-7) with time intervals.

- Leaky ReLU coefficient: 1e-3
- Learning rate: 5e-5
- Weight decay: 1e-4
- Adam coefficients: 0.0; 0.999
- Epochs: 90

A.2.4 Irregular time series with time interval: Forecasting results

Integrating time intervals to the data representation yielded unexpected results. Machine learning and deep learning models still outperformed the baseline. With blood glucose, all deep learning models had lower best MSE except for LSTM, and MLP and LSTM had an average MSE of 1066.46 (± 7.683) and 1069.53 (± 2.214) respectively, higher than linear regression at 1064.70 of MSE (Table 3). Interestingly, when running the best MLP selected for regular time series on the irregular time series, without time interval features, called *MLP-reg* in the tables below, I obtained lower MSE than the MLP using time intervals for blood glucose (MSE: 1053.20 vs. 1058.06) and urea nitrogen (MSE: 32.14 vs. 33.66). For irregular hematocrit time series, selected for the low dispersion, the best performing model overall was MLP with time intervals (MSE: 5.71) with a large variance between runs (avg. MSE: 5.81 (± 0.084)).

When put in perspective with the regular time series without time intervals, there is a general increase in forecasting error by adding the time interval information. Finally, the best model for irregular blood glucose time series was the CNN model with a best performance at an MSE of 1049.49.

As a consequence, we decided not to include time intervals in the generative adversarial models developed to generate laboratory test time series.

A.2.5 Irregular time series with drug exposure information: forecasting results

Multi-layer perceptron (MLP) In this study, we re-used the 10 best models from the regular time series experiments:

- Batch size: 128
- Dropout: 0.25
- Dimensions: 2 layers of width 64

- Leaky ReLU coefficient: 1e-2
- Learning rate: 5e-5
- Weight decay: 1e-3
- Adam coefficients: 0.9; 0.999
- Epochs: 100

Overall, adding drug exposure representations resulted in lower MSE than using only irregularly sampled measurements. While time intervals were harming the performances, binary drug exposure improves them for all the models tested. On average, MLP were the best models for every drug representation. Adding drug exposures to the binary vector, from 5 to 10, did not improve the models except for ATC-5 (1039.97 vs. 1040.90 for best MSE) but each model was well within each other's standard deviation. The best drug representation was ATC-3 with 5 drug concepts, illustrated with its MSE heatmap for 5 and 10 drugs in figure 8. The worse drug representation on average was surprisingly RxNorm with 10 drugs (Figure 11), while ATC-4 with 10 drugs (Figure 9)and ATC-5 with 5 drugs ((Figure 10)) were the worse models in terms of best possible performance (1045.55 vs. 1045.45 respectively). It is important to note that these models were however better than the MLP with time interval features (1058.06 MSE).

A.2.6 Conclusions

This annex study was designed to understand how deep learning can be applied to the data we wanted to model, what were the data representation that carry the most information as evaluated by the forecasting task, and what are the architectures that perform the best with these data representation. We demonstrated that MLP in the specific context of laboratory test measurements collected at the visit scale was the most robust model in terms of capacity versus generalizibility trade-off. Time intervals do not add information and therefore will not be used in the generative models as auxiliary information. The validation laboratory tests presented similar results as blood glucose which strengthen the validity of these conclusion.

On a more general note, when it comes to forecasting laboratory test time series with machine learning, classic machine learning provides performances close to deep learning models results, for a fraction of the time and complexity, but deep learning does make a better use of multi-modal features.

References

[1] eicu collaborative research database. june 2018. URL `https://eicu-crd.mit.edu/`.

[2] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv.org*, January 2017. URL `http://arxiv.org/abs/1701.04862v1`.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv.org*, January 2017. URL `http://arxiv.org/abs/1701.07875v3`.

[4] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 2018.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. *arXiv.org*, March 2017. URL http://arxiv.org/abs/1703.06490v2.

[7] P J Diggle and Richard J. Gratton. Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society*, 46:193–227, 1984. URL http://www.jstor.org/stable/2345504.

[8] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv.org*, June 2017. URL http://arxiv.org/abs/1706.02633v1.

[9] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv.org*, June 2014. URL http://arxiv.org/abs/1406.2661v1.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *arXiv.org*, March 2017. URL http://arxiv.org/abs/1704.00028v3.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997. URL https://dblp.org/rec/journals/neco/HochreiterS97.

[12] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[13] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[14] Scott H Lee. Natural language generation for electronic health records. *NPJ digital medicine*, 1(1):63, 2018.

[15] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279. Microsoft Research, Redmond, United States, January 2016. URL https://www.scopus.com/inward/record.uri?partnerID=HzOxMe3b&scp=85018914753&origin=inward.

[17] Henning Petzka, Asja Fischer, and Denis Lukovnicov. On the regularization of wasserstein gans. *arXiv preprint arXiv:1709.08894*, 2017.

[18] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, January 2016. URL https://www.scopus.com/inward/record.uri?partnerID=HzOxMe3b&scp=85018875486&origin=inward.

[19] Cédric Villani. Optimal Transport, 2009. doi: 10.1007/978-3-540-71050-9.

[20] Lu Wang, Wei Zhang, and Xiaofeng He. Continuous patient-centric sequence generation via sequentially coupled adversarial learning. In *International Conference on Database Systems for Advanced Applications*, pages 36–52. Springer, 2019.

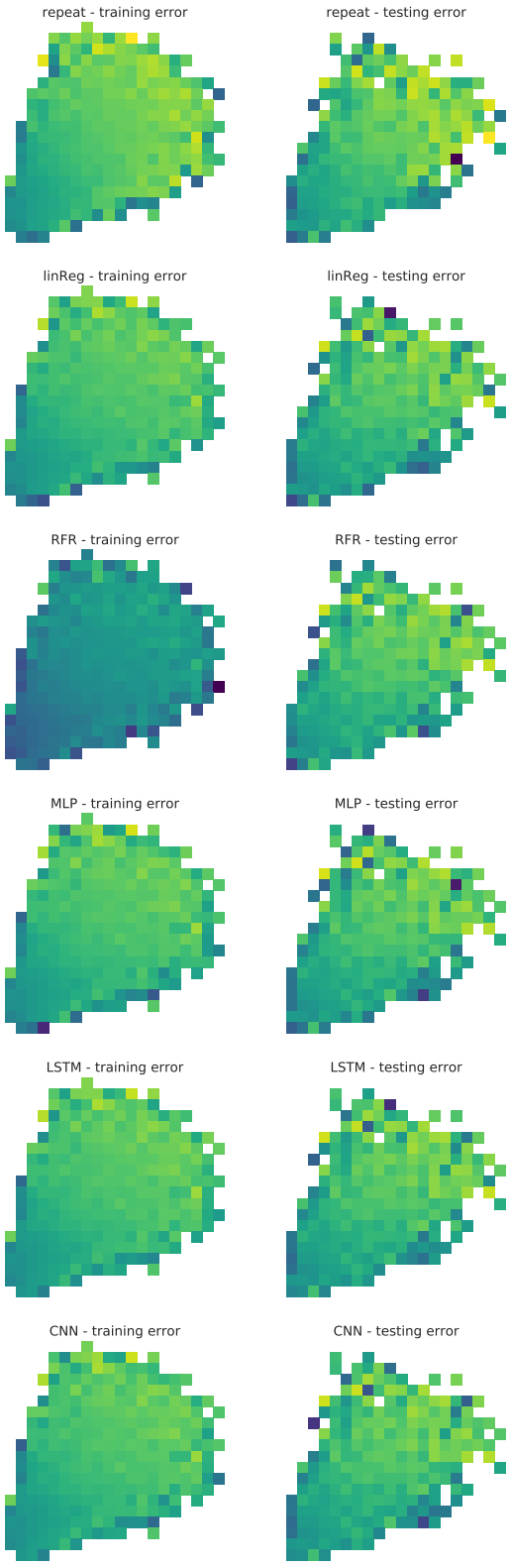


Figure 5: Heatmap of the average training and testing MSE between forecasting models on regular glucose lab time series.

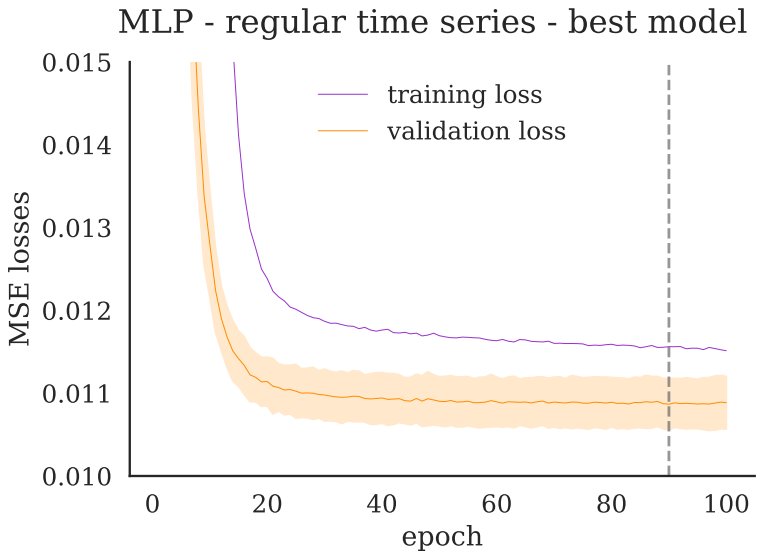


Figure 6: Best MLP model for regular glucose lab time series during tuning.

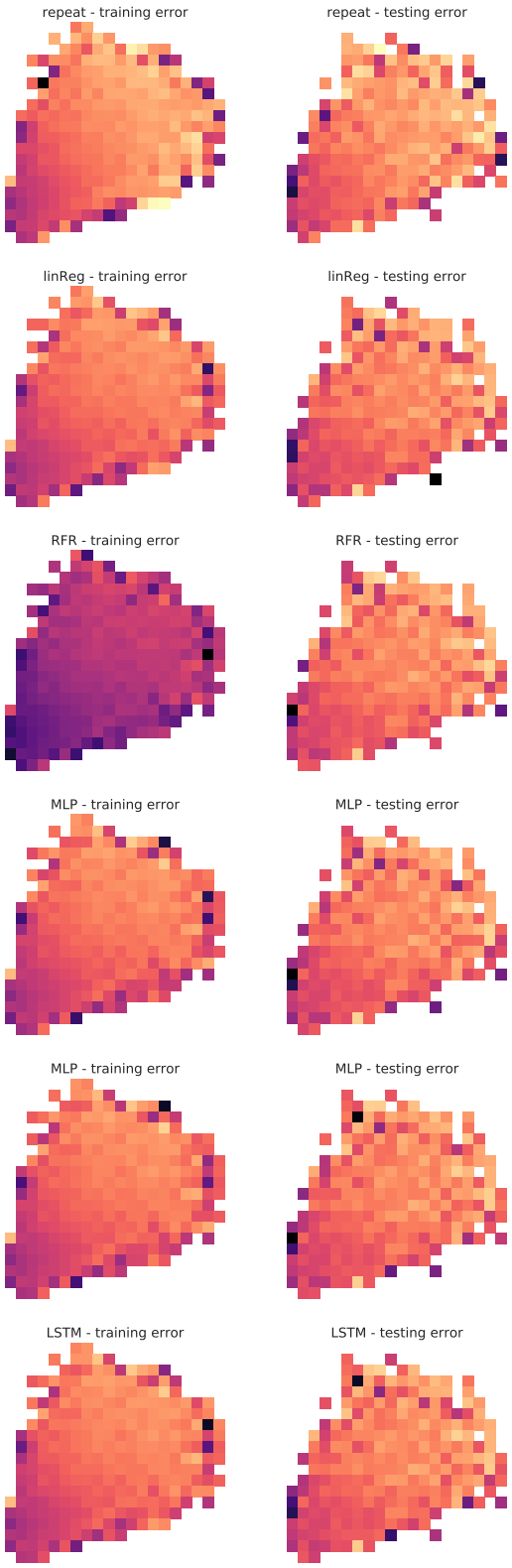
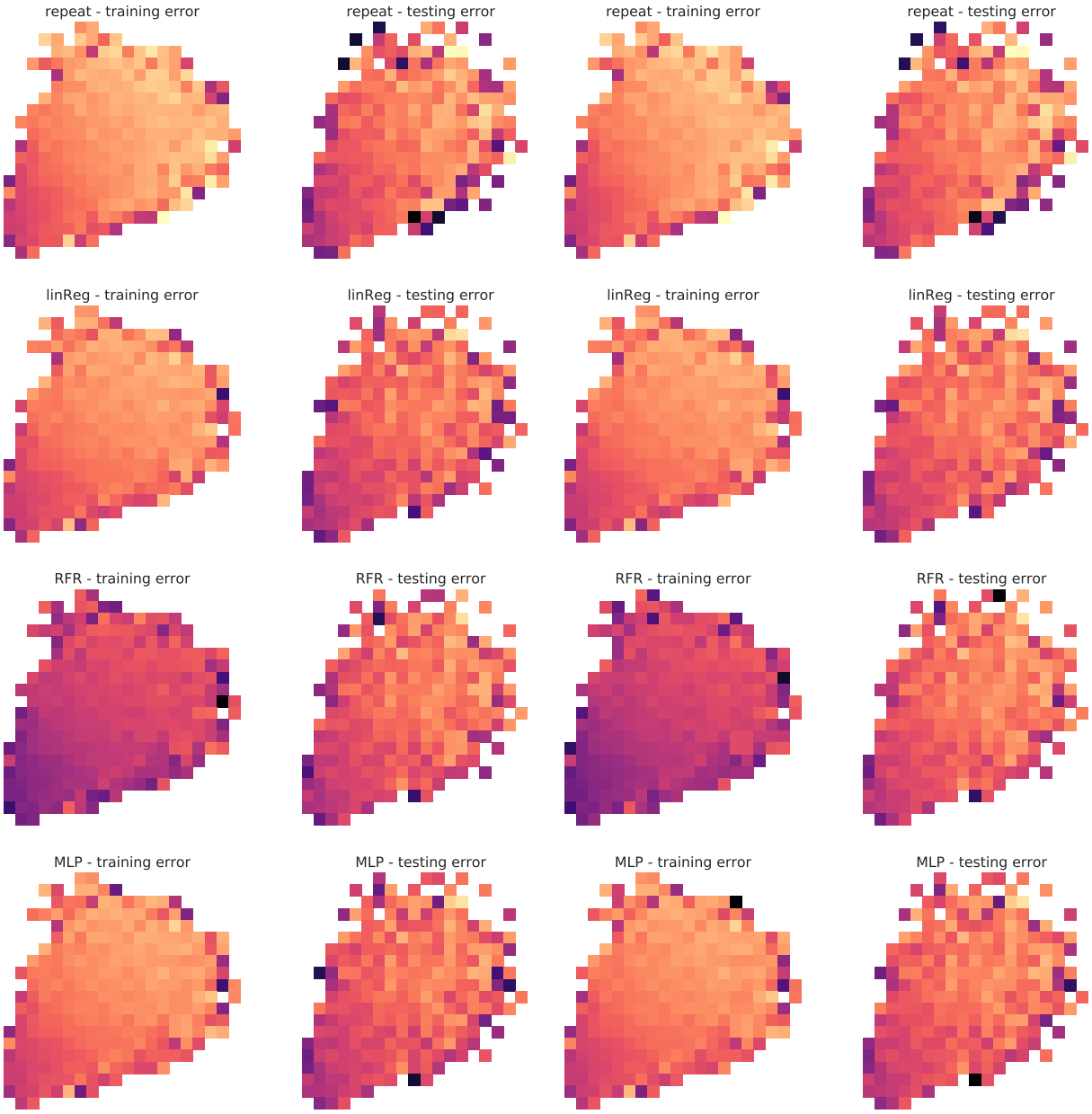


Figure 7: Heatmap of the average training and testing MSE between forecasting models on irregular glucose lab time series with time intervals.

drug representation	# of drug features	Model	best MSE	avg. MSE (±SD)
ATC-3	5	Repeated (baseline)	1463.61	–
		Linear regression	1051.82	–
		Random Forest Regression	1080.76	–
		MLP	**1039.97**	**1044.48 (± 2.963)**
	10	Repeated (baseline)	1463.61	–
		Linear regression	1051.79	–
		Random Forest Regression	1075.55	–
		MLP	1040.90	1046.98 (± 4.125)
ATC-4	5	Repeated (baseline)	1463.61	–
		Linear regression	1055.27	–
		Random Forest Regression	1080.15	–
		MLP	1044.10	1046.63(± 1.949)
	10	Repeated (baseline)	1463.61	–
		Linear regression	1052.64	–
		Random Forest Regression	1072.78	–
		MLP	1045.55	1047.71 (± 2.363)
ATC-5	5	Repeated (baseline)	1463.61	–
		Linear regression	1057.28	–
		Random Forest Regression	1080.88	–
		MLP	1045.45	1048.78 (± 2.884)
	10	Repeated (baseline)	1463.61	–
		Linear regression	1054.62	–
		Random Forest Regression	1073.09	–
		MLP	1042.86	1047.87 (± 3.141)
RxNorm	5	Repeated (baseline)	1463.61	–
		Linear regression	1056.04	–
		Random Forest Regression	1080.90	–
		MLP	1042.71	1046.74 (± 2.065)
	10	Repeated (baseline)	1463.61	–
		Linear regression	1054.40	–
		Random Forest Regression	1073.87	–
		MLP	1044.30	(1050.93 ± 4.956)

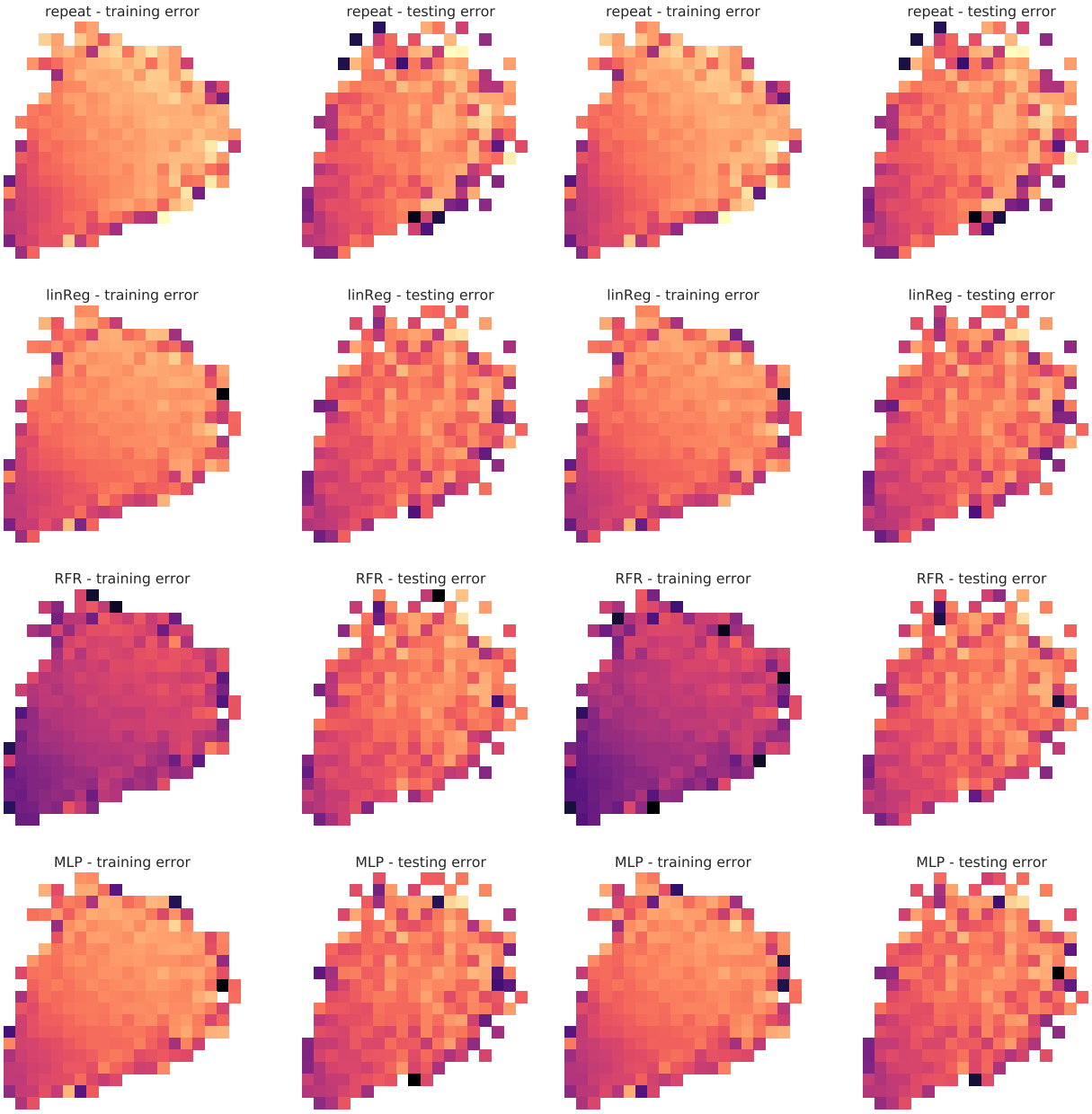
Table 4: MSE on test set of irregularly sampled blood glucose (2345-7) for different drug representations and numbers of features



(a) ATC-3 with 5 drugs

(b) ATC-3 with 10 drugs

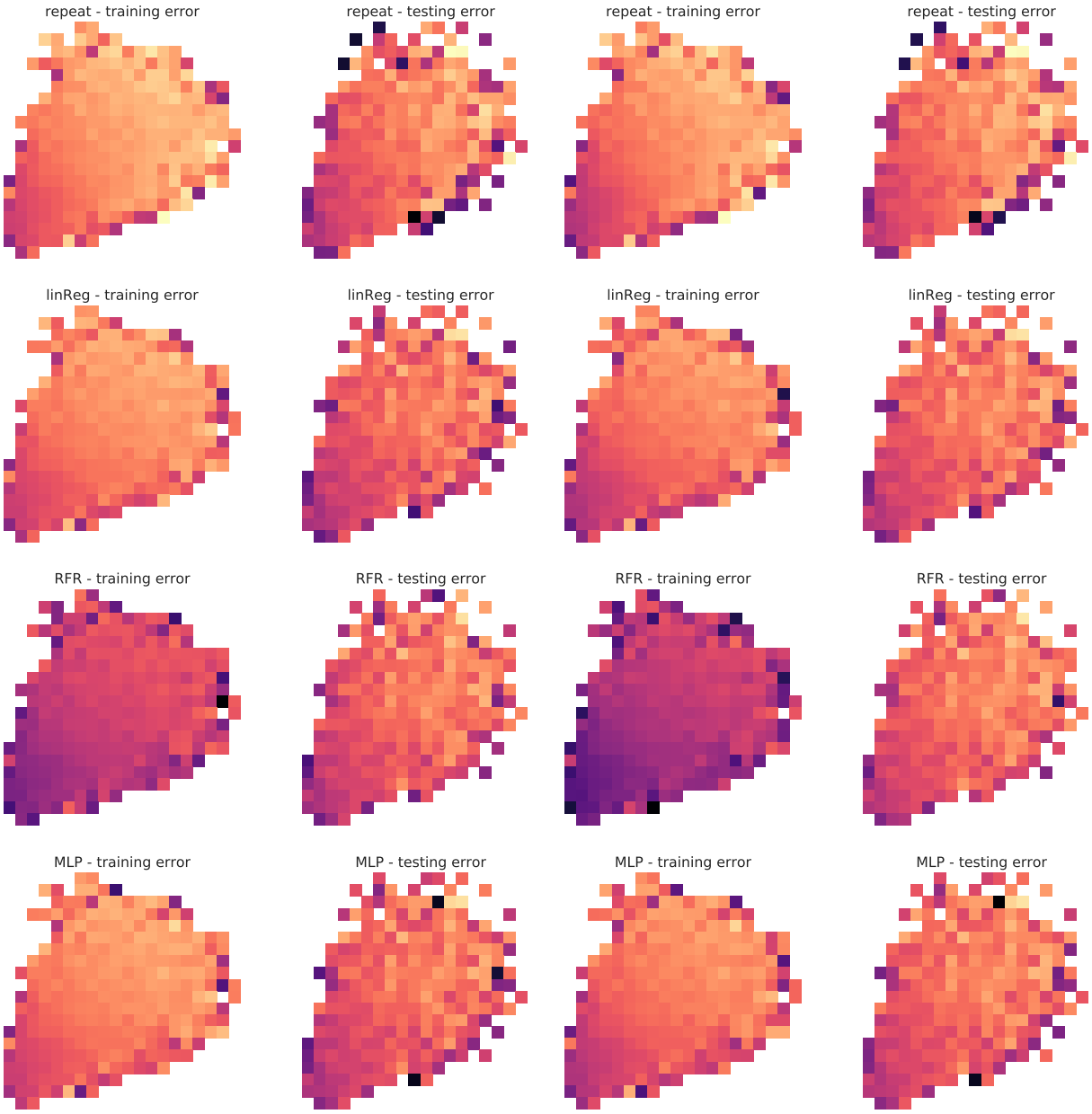
Figure 8: Heatmap of MSE errors by model for glucose lab irregular time series with ATC-3 exposure representation with 5 and 10 drugs.



(a) ATC-4 with 5 drugs.

(b) ATC-4 with 10 drugs

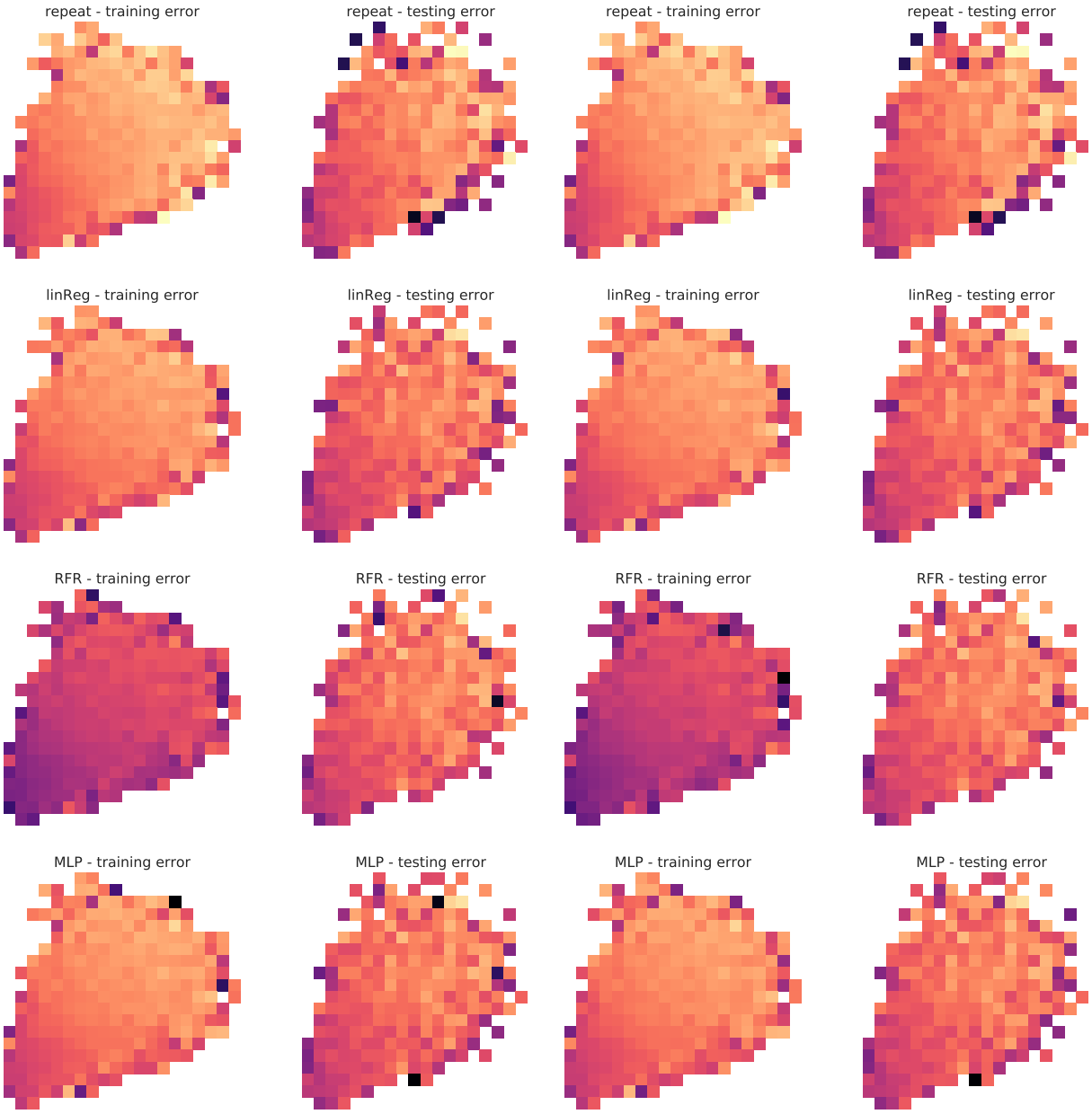
Figure 9: Heatmap of MSE errors by model for glucose lab irregular time series with ATC-4 exposure representation with 5 and 10 drugs.



(a) ATC-5 with 5 drugs

(b) ATC-5 with 10 drugs

Figure 10: Heatmap of MSE errors by model for glucose lab irregular time series with ATC-5 exposure representation with 5 and 10 drugs.



(a) RxNorm with 5 drugs

(b) RxNorm with 10 drugs

Figure 11: Heatmap of MSE errors by model for glucose lab irregular time series with RxNorm exposure representation with 5 and 10 drugs.