Reservoir Computing with Bacteria

Jean-Loup Faulon*, Paul Ahavi, Thi-Ngoc-An Hoang, MICALIS, INRAE,

University of Paris Saclay, Jouy-en-Josas, France

* Jean-Loup.Faulon@inrae.fr

Abstract

This study explores the use of bacterial strains in reservoir computing (RC) to solve regression and classification tasks. We employ an *Escherichia coli* K-12 MG1655 strain as the physical reservoir, training it on M9 minimal media supplemented with 28 metabolites, and measuring growth rates across various media compositions. Our physical RC system, using an *Escherichia coli* strain, demonstrates superior performance compared to multi-linear regression or support-vector machine and comparable performance to multi-layer perceptron in various regression and classification tasks. Additionally, the performances of RC based on genome-scale metabolic models for several bacterial species correlate with the diversity and complexity of phenotypes they produce. These findings highlight the potential of bacterial RC systems for complex computational tasks typically reserved for digital systems and suggest future research directions, including optimizing feature-to-nutrient mappings and integrating with emerging technologies for enhanced computing capabilities.

Keywords: Machine Learning, Reservoir Computing, Genome-Scale Metabolic Models, *Escherichia coli*

Abbreviation: **AMN**: Artificial Metabolic Network, **ANN**: Artificial Neural Network, ***E. coli*** : *Escherichia coli* **FBA**: Flux Balance Analysis, **GEM**: Genome Scale Metabolic Model, **MLP**: Multi-Layer Perceptron, **MLR**: Multi-Linear Regression, **RC**: reservoir computing, **R^2**: Regression coefficient. **SVM**: Support Vector Machine. **XGB**: XGBoost (parallel tree boosting).

INTRODUCTION

One of the main endeavors of synthetic biology has been to engineer bottom-up devices that perform computation within organisms[1–4]. Starting with digital computation, simple logic gates were constructed and layered into larger circuits[5,6], including switches[7], amplifiers[8], memory units[9] or oscillators[3]. These devices have been designed for a variety of applications, including biosensing for medical[10,11] and environmental monitoring[12]. However, engineering and tuning computational devices is challenging and their computational complexity is limited by several factors. First, incorporating large circuits into a cell imposes a substantial metabolic burden due to the need of expressing several transcription factors[13,14]. This metabolic burden can lead to variations in circuit behavior and may impair cell growth[15]. Second, the number of inducible promoters exhibiting a digital-like behavior (low background expression and step-like transfer function) is limited[16], restricting the number of usable inputs and of modules in a circuit. Orthogonality issues (with the host[14,17] or cross-

talk between different circuits[18–20]) restrict even more the number of usable promoters/ transcription factor couples. Third, digital circuits are prone to noise[18,21,22].

Several solutions have been developed to address the above shortcomings. Directed evolution was used on several promoters to improve their dynamic range, reduce their leakiness and their cross-talks with other modules[18]. Multicellular computing was implemented to distribute the burden among different cells and be able to reuse genetic parts within a same circuit[23,24]. New energy-efficient circuits have been engineered which have been less noisy and have required fewer genetic parts to perform complex computations. These are analog circuits[25,26], collective analog circuits[26,27], and perceptron architectures[26,28–30]. Despite these advancements, solutions are still limited in scope, and the inherent drawbacks continue to restrict the applicability of orthogonal bottom-up devices in real-world scenarios[31,32].

Synthetic biology devices have largely been inspired by behaviors observed in natural systems. Indeed, information processing systems analogous to logic gates[33,34], switches[35–37], and perceptrons[38,39] have all been discovered in various organisms. This raises the question: instead of engineering bottom-up devices, could we develop a top-down approach where natural organisms are repurposed to perform complex computational tasks?

The study of physical systems' ability to solve computational problems falls within the field of Artificial Intelligence known as reservoir computing (RC). Initially developed as an alternative to classical artificial neural networks, particularly recurrent neural networks (RNNs)[40], RC offers a more efficient training process. There are two kinds of RC systems: conventional RC and physical RC.

In a typical conventional RC framework, input data is provided to a reservoir, the states of the reservoir nodes are read, and this information is fed into a post-readout feedforward network—usually a simple linear classifier or regressor—that interprets the reservoir states to produce the final output (cf. Fig. 1). Unlike RNNs with trainable weights, conventional RC uses a fixed-weight RNN reservoir, making it faster to train and suitable for applications such as time series prediction and forecasting[41], dynamic system control, and Internet of Things (IoT) applications where quick, efficient processing is crucial[41].

In physical RC, the reservoir is replaced by a physical object. The system operates by first feeding input data into the physical reservoir, which processes the inputs and transforms them into a high-dimensional, dynamic state. These transformed states are then passed to a post-readout layer, similar to conventional RC. Physical RC has many applications, including the seminal liquid state machine for pattern recognition in a bucket of water[42], chemical RC developments for classification tasks and forecasting solutions of ODE systems[43], and biological RC systems using rat cortical neurons cultured on micropatterned substrates to solve classification tasks[44].

The aim of the current paper is to use bacteria as reservoirs and to probe to which extent bacterial models in conventional RC systems and an *Escherichia coli* bacterium in a physical RC system can solve regression and classification tasks of increasing difficulties.

RESULTS

Bacterial reservoir computing framework

Our aim is to demonstrate that bacterial strains have the capacity to be used in reservoir computing (RC) to solve regression and classification problems. In typical machine learning regression and classification tasks, training sets are composed of features and labels, and the goal is to predict the labels from the features. Similar to the RC systems mentioned in the introduction, the bacteria act as our reservoir. We transform the problem features into nutrients provided to the strain, observe the strain's behavior by measuring its phenotype, and feed these phenotype measurements to a feedforward neural network, which outputs the problem's solution.

One challenge in this workflow is finding the most appropriate mapping between features and nutrients. There is no generic procedure in the RC literature ; instead, there are various methods depending on the reservoir (e.g., using water waves to mimic sound waves in liquid state machines, or using reactant concentrations to create classification patterns in chemical reservoirs). In our case, since we are working with several features, one strategy could be an arbitrary one-to-one mapping of features to nutrients. However, this strategy might not yield the best performance. Another approach could be to find the optimal nutrients by training a prior input layer that generates nutrient sets. While this is not customary in RC, there is no reason why the prior layer cannot be trained. To train the prior layer, one needs to backpropagate a gradient from the post-readout layer to the prior input layer. This requires a gradient-compatible model that mimics the behavior of the strain, which in turn necessitates replacing the physical strain reservoir with a conventional reservoir that acts as a surrogate for the strain.

Many solutions have been developed over several decades to model strain behavior. Among these, we are particularly interested in models dealing with metabolism, as we provide the strain with various metabolites (nutrients) and measure phenotypes such as growth rates. While kinetic models have been proposed to model metabolism, they are still somewhat limited in scope regarding species and sizes[45]. Instead, we propose to use genome-scale metabolic models (GEMs) that have been developed for many strains[46]. To capture the dynamics of these models, we choose flux balance analysis (FBA)[47], a well-established technique in systems biology and bioengineering.

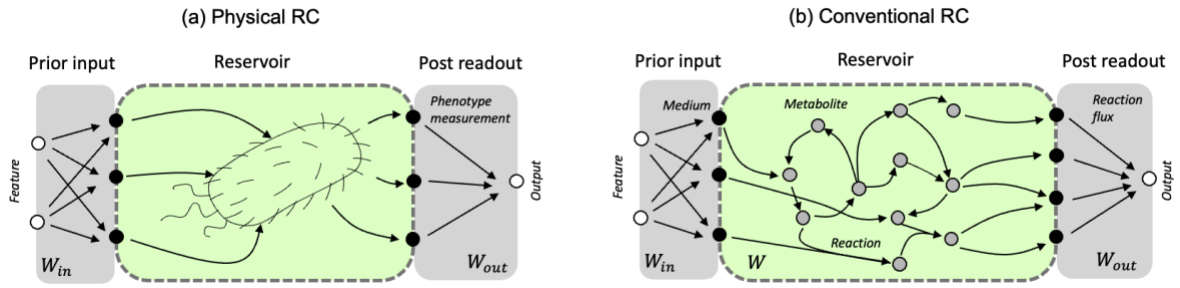

Fig. 1. Bacterial physical and conventional RC. *The Figure, inspired by Tanaka et al.*[40]*, was adapted for bacterial strain and models. (a) In physical RC the reservoir is realized by a physical (here biological) system or device (here a strain). (b) A conventional RC is a recurrent neural network (here a GEM model). In both cases the post readout neural network weights (W_{out}) are learned during training and the prior input weights (W_{in}) and reservoir weights (W) are fixed. Yet, with conventional RC, the weights W_{in} can be learned through training as long as the reservoir allows gradient backpropagation.*

Throughout this paper we will be using an *E. coli* physical RC and a conventional twin based on an *E. coli* GEM model. We will also make use of conventional RC for other bacteria for which GEMs are available.

RC to improve the performances of GEM models

To evaluate the ability of a conventional reservoir to accurately surrogate a physical reservoir, we need to assess how well the conventional reservoir reproduces phenotype measurements when the strain is grown in different media. To this end, we grew *E. coli* MG1655 on M9 minimal media supplemented with 28 metabolites (including 20 amino acids, 5 nucleobases, and 3 sugars or acids; see Fig. 2a and 'Bacterial strain and culture media' in the Methods section). All metabolites were added at fixed concentrations, allowing each composition to be represented by a binary vector. We then measured the growth rate of *E. coli* on 280 such media compositions (see 'Experimental training set' in the Methods section).

For modeling *E. coli*, we used iML1515[48], the latest *E. coli* genome-scale metabolic model (GEM). When performing flux balance analysis (FBA) using the 280 growth media, we found a poor correlation coefficient with the experimental measurements (see Fig. 2b). It is well known that FBA struggles with reproducing quantitative measurements accurately[49]. This difficulty arises because FBA does not take input media compositions directly but instead relies on media uptake fluxes. While uptake fluxes can be measured for individual metabolites, the interactions among multiple metabolites in a medium make such measurements prohibitive.

To enhance the performance of FBA, we developed an RC system where the prior-input layer performs the mapping between media composition and media uptake fluxes, the reservoir carries out FBA calculations, and the post-readout layer, which does not need to be trained, just prints out the FBA calculated growth rates (Fig. 2c). FBA is a linear program making use of the Simplex algorithm[50], which does not allow gradient backpropagation to the prior layer. To overcome this issue, we replaced the FBA reservoir by an Artificial Metabolic Network (AMN), a surrogate of FBA which we previously developed[51], and is gradient backpropagation compatible (see 'Building GEM reservoirs' in the

Methods section and Fig. S1). Results obtained using 5-fold cross-validation indicate a substantial improvement over FBA when predicting growth rate (Fig. 2d).

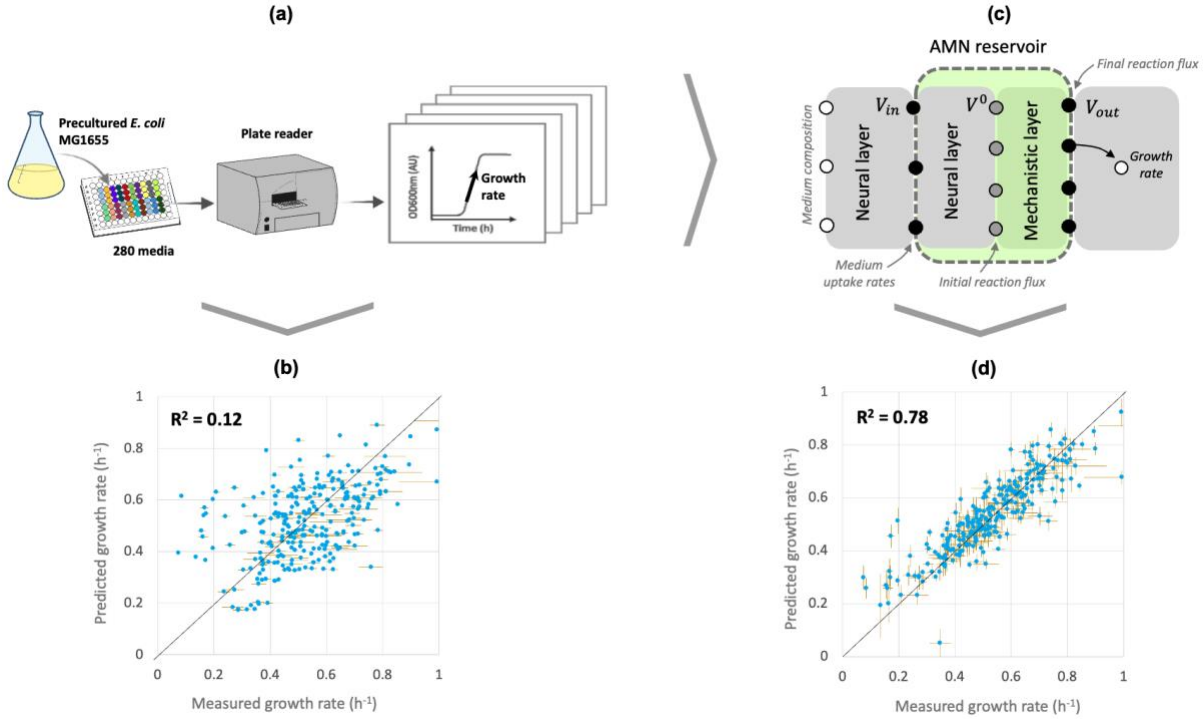

Fig. 2. GEM-FBA vs. RC growth rate predictions for *E. coli* grown on 280 different media compositions. *(a) Experimental setup where E. coli MG1655 was pre-cultured and grown on M9 minimal media with 28 added metabolites. Growth rates were measured using a plate reader. (b) Correlation between predicted and measured growth rates using the GEM-FBA approach with the CobraPy library[52], showing a poor correlation coefficient (R^2 = 0.12). Prior to running FBA, the binary media compositions were scaled by a constant to produce media uptake rates. The constant was searched greedily in order to maximize the regression coefficient between measurements and predictions, the optimal constant value was found to be 6.5. (c) Schematic of the developed RC system with an input layer mapping media composition to uptake fluxes, a reservoir performing FBA calculations, and a post-readout layer printing growth rates. (d) Correlation between predicted and measured growth rates using the RC system with AMN, showing a substantial improvement (R^2 = 0.78). As in panel(b), predicted growth rates were calculated using CobraPy but in this case fed with the medium uptake rates (V_{in}) and computed during training. All points were acquired on aggregated cross-validation sets (5-fold); error bars (both measured and predicted) represent standard deviations over 3 repeats.*

Conventional GEM RC to solve regression problems

Having shown that RC can substantially improve phenotype predictions when compared to measurements, we now utilize RC to solve regression problems extracted from the OpenML platform (see Table 2 in the Methods section). Our RC system architecture (Fig. 3a) consists of a prior network mapping problem features to medium composition, a reservoir composed of the trained neural layer and AMN shown in Fig. 2c, and a post-readout layer transforming the reaction fluxes computed by the AMN into the problem readout via a feedforward neural network. We developed two different methods for the prior layer. The first method (fixed prior) does not require training; instead, features are one-to-one mapped to medium metabolites using permutation importance (see 'Regression with

RC / fixed prior' in the Methods section). The second method (generative prior) outputs medium composition from features, with a generative model trained to produce the best medium composition for each element of the training set (see 'Regression with RC / generative prior' in the Methods section).

We tested these architectures on 10 different regression problems of increasing difficulty and compared their performance with classical regressors : multi-linear regression (MLR), multilayer perceptron (MLP), and XGBoost (XGB) (Figs. 3b, 3c).

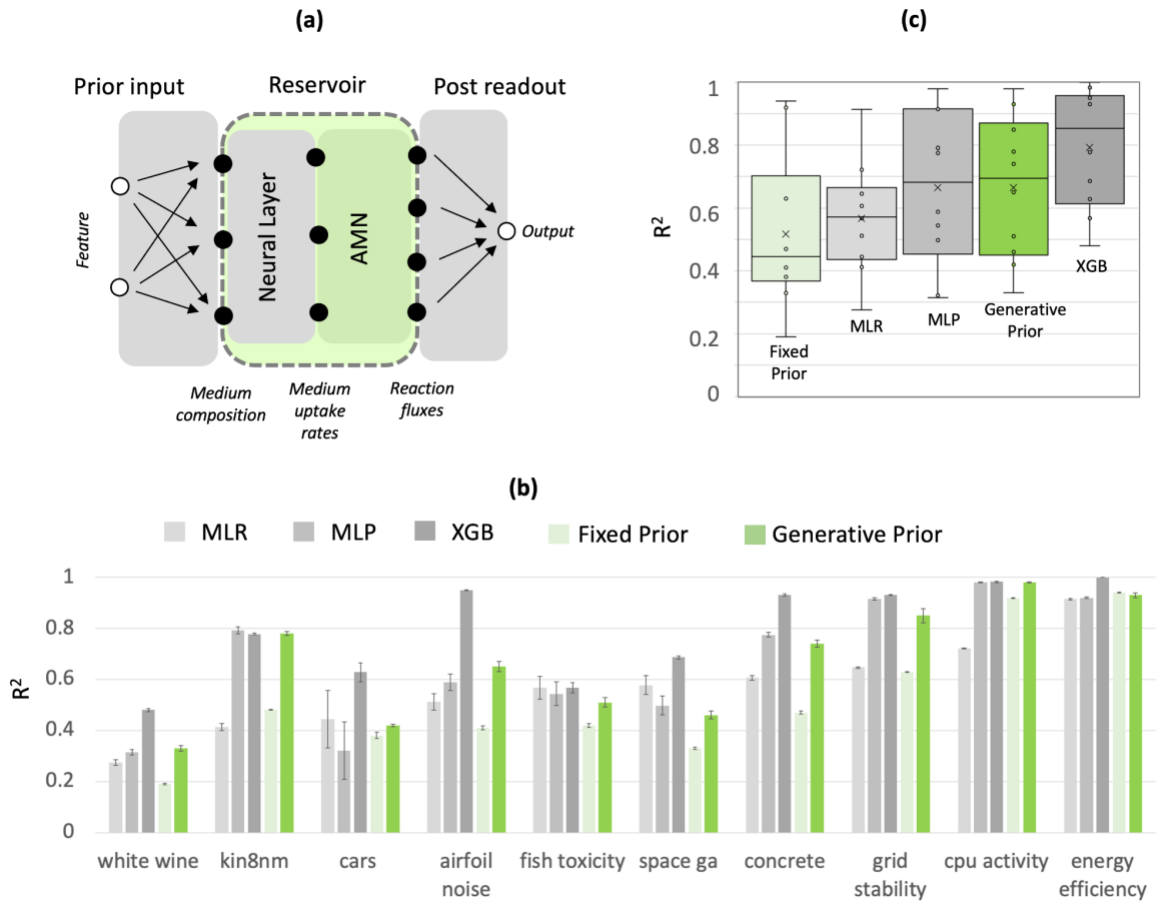

Fig. 3. Regression problem performances with conventional *E. coli* RC. *(a) Schematic of the RC system architecture used for regression problems. The prior input layer maps problem features to medium composition, the reservoir, consisting of a trained neural layer coupled to an AMN, computes all reaction fluxes, and the post-readout layer converts reaction fluxes into problem readouts via a feedforward neural network. (b) Performance comparison of different regression models (MLR, MLP, XGB, and conventional RC with fixed and generative priors) across 10 problems from the OpenML platform. (c) Box plot of R^2 values for the 5 different models and 10 regression problems of panel (b). In all panels R^2 values were acquired on aggregated cross-validation sets (5-fold), standard deviation was calculated on 3 replicates.*

We observe from Fig. 3 that the architecture with fixed prior is less performant than all other methods, and that the generative prior outperforms MLR and has performances similar to MLP. This indicates that a one-to-one mapping between features and medium metabolites is not necessarily the best strategy to find the most appropriate medium composition. On average, we find an improvement of

0.15 in the regression coefficient between the fixed and generative priors, indicating that some learning is still performed in the fixed prior architecture and consequently by the GEM reservoir (since nothing is learned through training in the fixed prior). Additionally, we observe that for some problems, the fixed prior architecture outperforms MLR.

To further assess the ability of bacterial models to learn, we tested 10 different GEMs with the same 10 regression problems. As experimental phenotype measurements are not available for all these strains, our GEM reservoirs were trained on FBA simulated data (see 'Building GEM reservoirs' in the Methods section) with the 28 media metabolites we used previously. In agreement with our previous observations, results shown in Fig. 4 indicate that the fixed prior architecture is less performant overall, and the generative prior architecture outperforms MLR and even MLP for many strains. Interestingly, we also observe that different strains yield different results, and these differential performances are independent of the chosen prior layer.

To probe what makes a given strain GEM better (or worse) suited to learning than another, we computed the number of different phenotypes the GEMs could produce with media composed of the 28 supplemented metabolites we already used (see 'Building GEM reservoirs' in the Methods section). Figs. 4b and 4c indicate a trend: the higher the number of phenotypes, the higher the regression coefficient. The ranking of the strain thus obtained is conserved between the fixed and generative prior layers. This behavior can naively be understood when considering that a strain producing *n* different phenotypes (e.g., 2 like growth, no-growth) should yield poor performance when subjected to regression problems requiring far more than *n* outputs. The trends observed in Figs. 4b and 4c are even more pronounced when computing the Shannon entropy for the phenotypes (see 'Phenotype number and entropy calculation' in the Methods section). Indeed, Fig. S2 indicates Pearson correlation coefficients between the log of the entropy and the mean R^2 values of 0.74 and 0.87 for the fixed and generative priors, respectively.

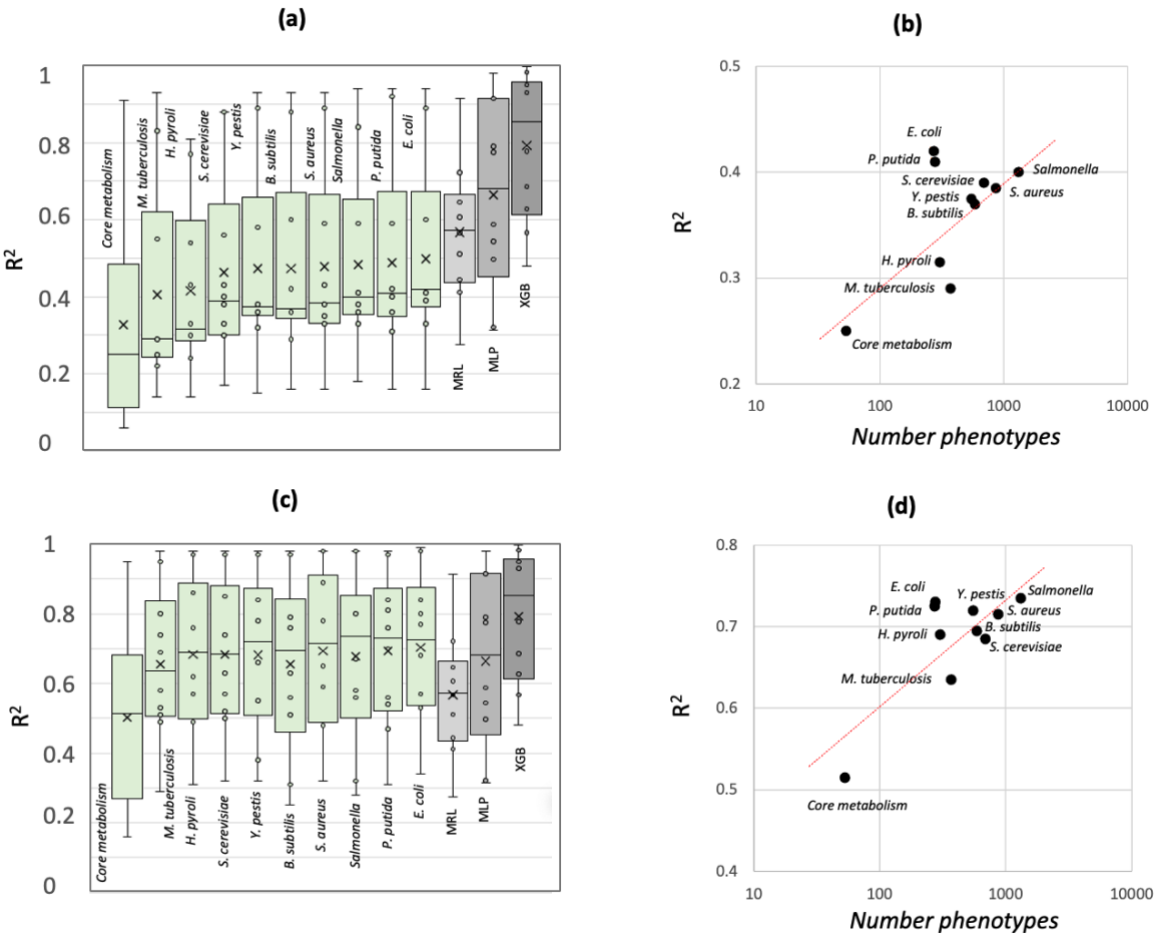

Fig. 4. Regression problem performances for several bacterial conventional RC systems. *RC was used for 10 GEMs listed in Table 1 and for the 10 regression problems listed in Table 2. (a) Box plots for regression coefficients (R^2) obtained for each bacterium using a fixed prior layer. (b) Median regression coefficient (R^2) taken from panel (a) vs. number of phenotypes when using a fixed prior. The Pearson correlation coefficient between R^2 values and the linear fit (y = 0.03 x + 0.11) is 0.65. (c) Box plots for regression coefficients obtained for each bacterium using a generative prior layer. (d) Median regression coefficient (R^2) taken from panel (c) vs. number of phenotypes when using generative prior. The Pearson correlation coefficient between R^2 values and the linear fit (y = 0.14 x + 0.33) is 0.78. In all panels R^2 values were acquired on aggregated cross-validation sets (5-fold) for 3 repeats.*

A physical *E. coli* RC to solve regression problems

As shown in Fig. 5a, with a physical *E. coli* RC the reservoir is an actual strain. There are two main differences between such a physical RC and a conventional one: (1) only growth rates are considered among the phenotypes, and (2) the growth rates used to predict problem readouts are measured rather than being predicted by a model. Following these constraints, our physical *E. coli* RC is built following the workflow given below.

We first train a conventional *E. coli* RC having a reservoir that mimics the behavior of the strain. The reservoir of this conventional twin is similar to the one presented in Fig. 2 with phenotypes reduced to only growth rates. This reservoir is trained on the set of 280 media compositions and growth rates presented in Fig. 2. We next collect the media compositions predicted by the conventional twin to best solve our 10 regression problems, and then grow *E. coli* MG1565 with the provided media and

measure growth rates. Lastly, we feed black the growth rates to the post layer of the twin to predict the problem readout.

Our conventional twin makes use of two different prior architectures : generative and selective. The generative prior, seen previously, is producing media compositions some of which might already be in our 280 media compositions set while others are novel. We note that growth rates need to be measured for these novel media compositions when feeding the generative prior output to the physical RC. The selective prior is forced to choose media compositions that are in the 280 media compositions set. That selection is performed using a Gumbel-softmax activation function as described in 'Regression with RC / Selective prior' in the Methods section. With this prior all growth rates have already been measured and it is not necessary to carry out additional growth measurements.

We note that since our physical RC and conventional twin are considering only growth rate as phenotype is it not necessary to train a post-layer, in fact after normalizing the growth rate and the problem readout the problem readout *is* the growth rate.

Results on the 10 regression problems obtained with the twin conventional RC and with generative or selective priors are presented in Fig. 5b and Fig. S3. As seen before, the twin conventional RC outperforms MLR and yields performances similar to MLP.

To assess the capacity of our physical *E. coli* RC to solve regression problems we selected 3 problems for which the conventional twin predicted low and high R^2 values. Additional growth rate measurements were carried out when using the generative prior but not when using the selective prior. Results presented for three regression problems in Fig. 5 panels c, d, e show that the physical reservoir leads to a performance drop compared to results obtained with the conventional twin when using the generative prior but not when using the selective prior. The drop with the generative prior can easily be explained by the fact the reservoir of the conventional twin is not a perfect twin as it predicts growth rate with a regression coefficient of 0.78 (cf. Fig. 2 and Fig. S3). As we have seen before with conventional RC, our physical RC generally outperforms MLR and when using the selective prior has similar or better performances than MLP for some regression problems (cf. Fig. S4).

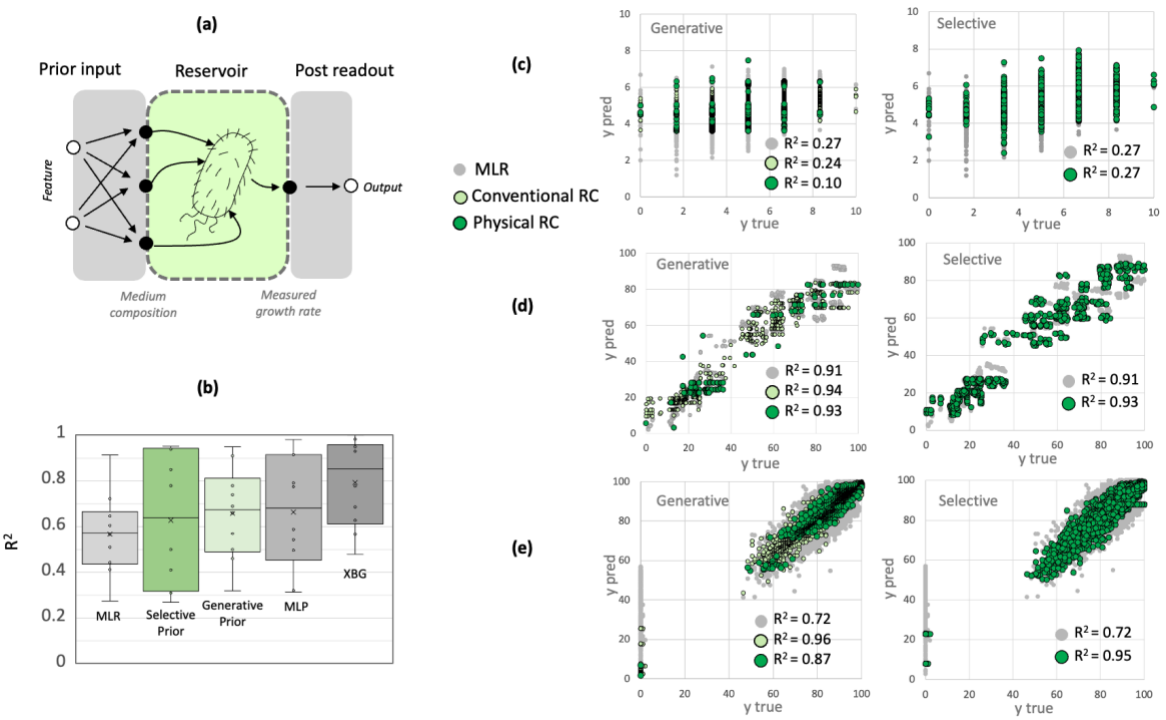

Fig. 5. **Regression problem performances with physical *E. coli* RC**. *(a) The architecture of the physical RC system. (b) Box plot of R^2 values obtained for 10 regression problems of Table 2 with MLR, MLP , XGB and conventional twin models with the selective and generative priors. (c) Results obtained for the 'white wine' regression problem with MLR and conventional and physical RC making use of generative and selective priors. All physical RC results are derived from measured growth rates. With the generative prior 4898 y values were predicted with conventional RC, 1639 of which corresponding to an experimentally measured growth rate. With the selective prior all y values were derived from experimentally measured growth rates. (d) Results obtained for the 'energy efficiency' regression problem. With the generative prior 768 y values were predicted with conventional RC, 162 of which corresponding to an experimentally measured growth rate. With the selective prior all y values were derived from experimentally measured growth rates. (e) Results obtained for the 'cpu activity' regression problem. With the generative prior 8192 y values were predicted with conventional RC, 3057 of which corresponding to an experimentally measured growth rate. With the selective prior all y values were derived from experimentally measured growth rates. Results presented in all panels were acquired on aggregated cross-validation sets (5-fold).*

Conventional and physical *E. coli* RC to solve classification problems

Turning our attention to classification problems, we chose to benchmark our *E. coli* reservoirs against both linear and non-linear classification tasks, similar to those used in a chemical reservoir computing system[43]. We focused on eight classification tasks, generating a two-dimensional input space from 500 randomly sampled x_1, x_2 values in the [0,1] interval. Labels (y = 0 or 1) were assigned based on patterns shown in Fig. 6a (see also 'Classification Problems' in the Methods section for further details). As with our regression tasks, the x_1, x_2 values were transformed into medium metabolite compositions using a prior layer (see 'Regression and classification with RC' in the Methods section for further information).

We utilized the conventional reservoir from Fig. 3a with a fixed prior, alongside the *E. coli* physical reservoir from Fig. 5a with a selective prior. In all cases, the post-readout layer consisted of a simple

sigmoid function that took as input all reaction fluxes for the fixed prior and growth rates for the selective prior. The results shown in Fig. 6 are consistent with those obtained for regression tasks in Fig. 3, with XGB being the best performer.

We observe that the conventional RC with fixed priors performs better on classification tasks compared to regression tasks. With this RC system, the prior layer remains untrained, and learning occurs exclusively in the post-readout layer. Nevertheless, for non-linear tasks such as CIRCLE, CONCENTRIC, SINE, and DOTS, the fixed-prior RC system outperformed classical SVM and MLP classifiers. We recall that RC performance correlates with the diversity of strain phenotypes (cf. Fig. 4). Our classification tasks involve only two classes, requiring the distinction between two phenotypes, this may explain the observed improvement in the performance of the fixed-prior RC on classification tasks compared to regression tasks.

Additionally, our physical RC system with a selective prior, where all reservoir outputs are experimentally validated, outperformed SVM in all tasks and exhibited similar or superior performance to MLP for the AND, OR, LINEAR, SINES, and DOTS tasks. These findings indicate that our physical *E. coli* reservoir with a selective prior is capable of handling non-linear classification tasks. Similar trends have already been observed in chemical reservoirs[43], although in our case, we rely solely on growth rate measurements instead of monitoring the concentrations of multiple species, as was done in chemical reservoir systems.

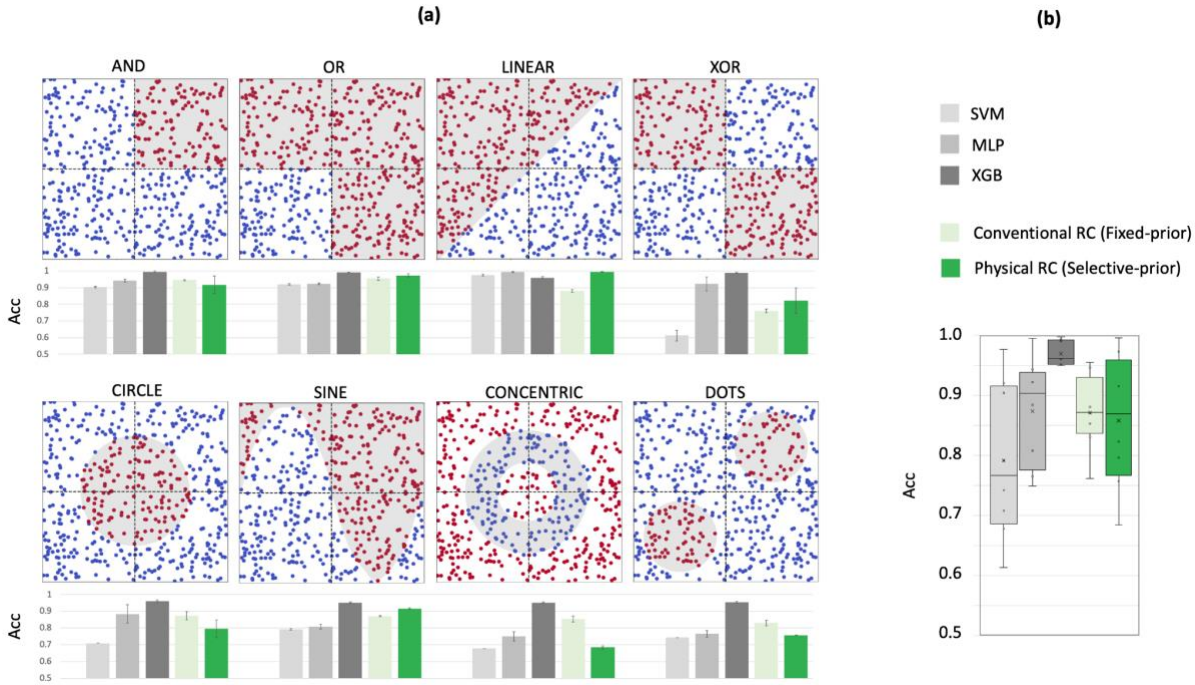

Fig. 6. **Classification problem performances with conventional and physical *E. coli* RC.** ***(a)*** *Performances for 3 linear (AND, OR, LINEAR) and 5 nonlinear (XOR, CIRCLE, SINE, CONCENTRIC, DOTS) classification tasks with SVM, MLP and XGB classical classifiers along with E. coli convention RC with fixed prior and E. coli physical RC with selective priors.* ***(b)*** *Color codes and box plots of accuracies obtained for the 8 classification tasks of panel (a).*

DISCUSSION

This study demonstrates the potential of utilizing *E. coli* strains in reservoir computing (RC) to solve regression and classification tasks. Our approach, combining a conventional RC model and a physical bacterial RC system, shows promise in computational tasks typically performed by digital systems, with the added benefit of using biological systems for computation, such as low energy consumption, inherent parallelism, self-repair mechanisms, and the ability to adapt to dynamic environments.

Our first key finding is the significant improvement in prediction accuracy when transitioning from traditional flux balance analysis (FBA) to an artificial metabolic network (AMN)-based RC system. Initially, the FBA model exhibited low correlation when predicting growth rates from media compositions. By integrating an AMN within the RC framework, we were able to achieve a much higher prediction accuracy ($R^2 = 0.78$). This improvement highlights the effectiveness of the RC approach in capturing complex metabolic dynamics that FBA alone struggles to model.

In terms of problem-solving capabilities, our conventional bacterial RC systems outperform traditional regression methods, such as multi-linear regression (MLR) and multi-layer perceptron (MLP), especially when using a generative prior. This generative prior layer is crucial in optimizing media compositions and leads to better regression outcomes compared to the fixed prior. Interestingly, we observed a correlation between the number of phenotypes produced by different genome-scale metabolic models (GEMs) and the performance of the RC system. A higher number of distinct phenotypes was associated with better regression performance, suggesting that phenotypic diversity is an important factor for effective computational performance in biological systems.

For classification tasks, we found that conventional RC systems with fixed priors performed better than anticipated, particularly for non-linear tasks such as CIRCLE, CONCENTRIC, and DOTS. The fixed-prior RC system even outperformed classical SVM and MLP classifiers for some of these tasks. This improvement in performance compared to regression tasks could be attributed to the reduced complexity of classification tasks, which only require the system to produce two distinct phenotypes to differentiate between two classes.

Our physical RC systems, utilizing *E. coli* strain with selective priors, further demonstrated the robustness of bacterial RC systems in handling complex, regression and classification tasks. The selective-prior RC system consistently outperform MLR and SVM across all tasks and achieved comparable or better performance than MLP in 6/10 regression tasks (kin8nm, cars, airfoil noise, concrete, grid stability, energy efficiency) and in 4/8 classification tasks (AND, OR, LINEAR, and SINES). These results underline the capacity of physical bacterial reservoirs to tackle computational problems traditionally solved by digital systems.

Overall, our results highlight the promise of using bacterial strains in physical RC systems. We have shown with our fixed prior layer that bacteria have inherent computational capabilities solving both regression and classification tasks, and that bacterial RC can emulate computational tasks typically reserved for digital systems, providing a biomimetic approach to information processing. Our findings echo the concept of cellular computing supremacy proposed by Goñi-Moreno[53]. Indeed, our physical RC system showed superior performance in certain regression and classification tasks, suggesting that

bacterial strains could be optimized to achieve similar or even better results than traditional computing methods in specific applications. Future work should focus on further optimizing the mapping of features to nutrients and exploring additional bacterial strains and metabolic models to expand the applicability of this approach. Moreover, integrating this approach with other emerging technologies, such as neuromorphic computing, edge computing, advanced material science, and engineered living materials (ELMs), could lead to even more robust and versatile computing systems.

METHODS

Genome Scale Metabolic Models (GEMs)

A set of 10 GEMs were downloaded from the BiGG database[46]. Following a method outlined in Faure et al.[51] metabolic reactions having no specific directions were duplicated one forward one backward. Further information on these models are provided in the Table below.

Table 1. GEMs model characteristics.

BiGG ID	Short name	Organisms	Metabolites	Reactions	Mono directional reaction	Genes	Reference	Year published
e_coli_core	*Core E. coli*	Core *Escherichia coli* Metabolic Model	72	95	154	137	Orth *et al.*[54]	2010
iEK1008	*M. tuberculosis*	*Mycobacterium tuberculosis* H37Rv	998	1226	1597	1008	Kavvas *et al.*[55]	2018
iIT341	*H. pylori*	*Helicobacter pylori* 26695	485	554	793	339	Thiele *et al.*[56]	2005
iJN1463	*P. putida*	*Pseudomonas putida* KT2440	2153	2927	4034	1462	Nogales *et al.*[57]	2018
iML1515	*E.coli*	*Escherichia coli* str. K-12 substr. MG1655	1877	2712	3682	1516	Monk *et al.*[48]	2017
iMM904	*S. cerevisiae*	*Saccharomyces cerevisiae* S288C	1226	1577	2226	905	Mo *et al.*[58]	2009
iPC815	*Y.pestis*	*Yersinia pestis* CO92	1552	1961	2767	815	Charusanti *et al.*[59]	2011
iYO844	*B. subtilis*	*Bacillus subtilis subsp. subtilis* str. 168	990	1250	1806	844	Oh *et al.*[60]	2007
iYS1720	*Salmonella*	*Salmonella* pan-reactome	2436	3357	4451	1707	Seif *et al.*[61]	2018
iYS854	*S. aureus*	*Staphylococcus aureus subsp. aureus* USA300_TCH1516	1335	1455	2086	866	Seif *et al.*[62]	2019

Bacterial strain and culture media

Escherichia coli K-12 MG1655 (accession number: NC_000913.3) was the strain used in all experiments described in this study. All the media were based on a M9 minimal medium composed of (i) M9 salts at pH = 7.2 : 47.76 mM Na_2HPO_4 (Merck/Sigma-Aldrich, CAS number : 7558-79-4), 22.04 mM KH_2PO_4

(Merck/Sigma-Aldrich, CAS number : 7778-77-0), 8.56 mM NaCl (Merck/Sigma-Aldrich, CAS number : 7647-14-5), 18.7 mM NH_4Cl (Merck/Sigma-Aldrich, CAS number : 12125-02-09), (ii) magnesium sulfate : 2 mM $MgSO_4$ (Merck/Sigma-Aldrich, CAS number : 7488-88-9), (iii) calcium chloride dihydrate : 0.1 mM $CaCl_2{\cdot}2H_20$ (Merck/Sigma-Aldrich, CAS number : 10035-04-08), and (iv) trace elements : 0.4 mM $Na_2EDTA{\cdot}2H_2O$ (Merck/Sigma-Aldrich, CAS number : 6381-92-6), 0.03 mM $FeCl_3{\cdot}6H_2O$ (Merck/Sigma-Aldrich, CAS number : 7705-08-0), 6.16 µM $ZnCl_2$ (Merck/Sigma-Aldrich, CAS number : 7646-85-7), 0.76 µM $CuCl_2{\cdot}2H_2O$ (Merck/Sigma-Aldrich, CAS number : 7447-39-4), 0.42 µM $CoCl_2{\cdot}6H_2O$ (Merck/Sigma-Aldrich, CAS number : 7791-13-1), 1.62 µM H_3BO_3 (Merck/Sigma-Aldrich, CAS number : 10043-35-3), 0.08 µM $MnCl_2{\cdot}4H_2O$ (Merck/Sigma-Aldrich, CAS number : 13446-34-9), 1.65 µM $Na_2MoO_4{\cdot}2H_2O$ (Merck/Sigma-Aldrich, CAS number : 7631-95-0).

This minimal medium was supplemented with different combinations of nutrients selected from 3 sugars or acids, 20 proteinogenic amino acids and 5 nucleobases. The 3 sugars or acids used and their final concentrations are : 44.4 mM D-(+)-glucose, 53.29 mM D-(+)-xylose (Merck/Sigma-Aldrich, CAS number : 58-86-6), 49.37 mM sodium succinate dibasic (Merck/Sigma-Aldrich, CAS number : 150-90-3). Final amino acids and nucleic bases concentrations used are based on the EZ rich defined medium composition[63]: 0.8 mM L-alanine (Merck/Sigma-Aldrich, CAS number : 56-41-7), 5.2 mM L-arginine (Merck/Sigma-Aldrich, CAS number : 74-79-3), 0.4 mM L-asparagine (Merck/Sigma-Aldrich, CAS number : 70-47-3), 0.4 mM L-aspartic acid (Merck/Sigma-Aldrich, CAS number : 56-84-8), 0.1 mM L-cysteine (Merck/Sigma-Aldrich, CAS number : 52-90-4), 0.55 mM L-glutamic acid (Merck/Sigma-Aldrich, CAS number : 56-86-0), 0.615 mM L-glutamine (Merck/Sigma-Aldrich, CAS number : 56-85-9), 0.8 mM glycine (Merck/Sigma-Aldrich, CAS number : 56-40-6), 0.2 mM L-histidine (Merck/Sigma-Aldrich, CAS number : 71-00-1), 0.4 mM L-isoleucine (Merck/Sigma-Aldrich, CAS number : 73-32-5), 0.8 mM L-leucine (Merck/Sigma-Aldrich, CAS number : 61-90-5), 0.5 mM L-lysine (Merck/Sigma-Aldrich, CAS number : 56-87-1), 0.2 mM L-methionine (Merck/Sigma-Aldrich, CAS number : 63-68-3), 0.4 mM L-phenylalanine (Merck/Sigma-Aldrich, CAS number : 63-91-2),0.4 mM L-proline (Merck/Sigma-Aldrich, CAS number : 147-85-3),10 mM L-serine (Merck/Millipore, CAS number : 56-45-1), 0.4 mM L-threonine (Merck/Sigma-Aldrich, CAS number : 72-19-5), 0.1 mM L-tryptophan (Merck/Sigma-Aldrich, CAS number : 73-22-3), 0.2 mM L-tyrosine (Merck/Sigma-Aldrich, CAS number : 60-18-4), 0.6 mM L-valine (Merck/Sigma-Aldrich, CAS number : 72-18-4), 0.04 mM adenine (Merck/Sigma-Aldrich, CAS number : 73-24-5), 0.04 mM cytosine (Merck/Sigma-Aldrich, CAS number : 71-30-7), 0.04 mM uracil (Merck/Sigma-Aldrich, CAS number : 66-22-8), 0.04 mM guanine (Merck/Sigma-Aldrich, CAS number : 73-40-5), 0.04 mM thymidine (Merck/Sigma-Aldrich, CAS number : 50-89-5).

Experimental training set

The objective was to develop a set of media capable of supporting a broad spectrum of maximal growth rates for *E. coli* K-12 MG1655. To this effect, different combinations of amino acids, nucleobases, and other carbon sources were designed to supplement a M9 minimal medium. Amino acids involved in the same biosynthesis pathway were grouped together as well as purines and pyrimidines (regarding nucleobases). 9 groups of amino acids were formed : Gly/Ser/Thr, Asn/Asp/Ala, Glu/Gln, Arg/Pro, Val/Leu/Ile, Lys, Cys/Met, His, Phe/Tyr/Trp (His and Lys were often associated together for the sake of simplicity, Glu and Gln were separated from Asn/Asp/Ala to ensure balanced groups). Each group was ranked according to their metabolic cost for *E. coli*, as defined in Zampieri *et*

al.[64]. It has been demonstrated that amino acids with a low metabolic cost are consumed at a higher rate than the other ones[64]. The same study suggested that these low-cost amino acids could be used as carbon and energy sources[64]. Therefore, the hypothesis underlying the ranking was that amino acids with lower metabolic costs would have a greater contribution to the growth rate than the other ones. By varying the number of groups of amino acids supplemented in the medium as well as their rank, it was possible to generate a first set of media capable of supporting a wide range of maximal growth rates for *E. coli* K-12 MG1655. This range was further widened by designing new media supplemented with amino acids but also nucleobases and different energy-rich or energy-poor carbon sources (D-(+)-glucose, D-(+)-xylose, succinate).

Bacterial growth monitoring. An *E. coli* K-12 MG1655 glycerol stock was streaked onto a M9 agar plate supplemented with 44.4 mM (0.8%) glucose (the composition is described in « Bacterial strain and culture media ») which was then incubated overnight at 37°C. A colony was used to inoculate a preculture of 10 mL of M9 liquid medium supplemented with 44.4 mM glucose in a 100 mL erlenmeyer flask. The flask was incubated at 37°C under agitation at 180 rpm (Orbital floor shaker ZWY-B3222, Labwit) until the optical density at 600 nm of the culture reached 0.5 (measured on a UVisco V-1100D spectrophotometer, LC-Instru). 198 µL of each medium to test was prepared in each well of a clear flat-bottom 96-well plate (except the edge wells to avoid the edge effect) and inoculated with 2µL of preculture, so that the starting optical density at 600nm was estimated at 0,005. Each medium was tested in triplicate. The plate was then incubated in a plate reader (BioTek HTX Synergy, Agilent Technologies) at 37°C with continuous orbital shaking at 807cpm (1mm). The growth of *E. coli* K-12 MG1655 was assessed by monitoring the optical density at 600nm every 20 minutes for a period ranging from 16 to 24 hours.

Growth rate determination. Optical density measurements at 600 nm were smoothed to reduce the noise. A natural logarithm function is applied to these smoothed measurements. First, the two timepoints between which the growth rate is maximal were determined using linear regression. The growth rate determination window was then extended in order to englobe all timepoints of the exponential phase. To do so, a fitting score was calculated between the linear model and the experimental data. The growth rate determination window was expanded in both directions (before and after the two initial timepoints) as long as the fitting score remained below a chosen threshold. The maximal growth rate was finally calculated using the timepoints within the final window. Each medium was tested in triplicate. The mean and standard deviation of each triplicate was computed. The training set file contained the media (in a binary form capturing the presence or absence of a specific medium metabolite) the average growth rate and the corresponding standard deviation.

FBA simulated training set

A set of 10,000 media were generated for each of the 10 GEM models listed in Table 1. Each medium comprised two parts: a minimal medium and a variable medium. The minimal media were specific to each model, and defined as the set of metabolites necessary for the model to produce a growth. The variable media were generated at random among the same 28 metabolites listed in the previous section. As with the experimental training set each variable medium was represented by a 28 bits vector. FBA was run for each medium and each model, all reaction rates (including the growth rate) were collected and stored in the training set files.

Building GEM reservoirs

Two kinds of GEM reservoirs were constructed. The first made use of *E. coli* iML1515 GEM. Following the procedure outlined in Faure *et al.*[51] an AMN was first trained on an FBA simulated training set comprising 10,000 instances (cf. training performances in Fig. S1). Among the solutions proposed in Faure *et al*. we chose the AMN-QP architecture which was inspired by Physics Informed Neural Networks (PINNs). As in PINN, the loss function was composed of 3 terms capturing the fit to the true values (FBA calculated growth rates), the boundary condition constraints (lower and upper bounds for media uptake fluxes) and the stoichiometric constraint (SV = 0, where S is the GEM stoichiometric matrix and V the vector of fluxes from all GEM reactions). The trained AMN was then used as the reservoir in the RC architecture depicted in Fig. 2 and trained on the experimental training set composed of 280 different medium compositions and measured growth rates. In this RC architecture, only the prior layer was trained to find the mapping between media compositions and media uptake rates. The iML1515 GEM RC model was saved and later used as a reservoir to solve regression problems.

The second kind of GEM reservoir was constructed for all strains listed in Table 1. These GEM reservoirs comprise an input layer representing a medium composition which is directly fed to an AMN. One note that the conversion from medium compositions to medium uptake rates is carried out by the neural layer of the AMN computing initial flux values (V^0 in Fig. 2, see also Faure *et al*.[51]). As with iML1515 GEM we used the AMN-QP architecture with the loss function described above. All GEM RC models were saved and later used as reservoirs to solve regression problems.

Regression problems

The 10 regression problems downloaded from OpenML are listed in the Table below.

Table 2. OpenML regression problems

Problem	Instances	Features	Dataset reference
white wine	4898	11	https://www.openml.org/search?type=data&sort=runs&id=40498&status=active
kin8nm	8192	8	https://www.openml.org/search?type=data&sort=runs&id=189&status=active
cars	392	7	https://www.openml.org/search?type=data&sort=runs&id=21&status=active.
airfoil noise	1503	5	https://www.openml.org/search?type=data&sort=runs&id=44957&status=active
fish toxicity	908	6	https://www.openml.org/search?type=data&sort=runs&id=44970&status=active
space ga	3107	6	https://www.openml.org/search?type=data&sort=runs&id=507&status=active
concrete	1030	8	https://www.openml.org/search?type=data&sort=runs&id=44959&status=active
grid stability	10000	12	https://www.openml.org/search?type=data&sort=version&status=any&order=asc&id=43007
cpu activity	8192	21	https://www.openml.org/search?type=data&sort=runs&id=197&status=active
energy efficiency	768	8	https://www.openml.org/search?type=data&sort=runs&id=1472&status=active

To produce Figs. 3, 4 and 5, we used the MLR, MLP and XGB regressors from SciKitLearn[65] to acquire regression coefficients (R^2) on 5-fold cross validation aggregated sets. We iterated 5 times to compute standard deviation. The hyperparameters were the default ones for MLR and XGB, for MLP we used one hidden layer of size twice the number of features, an Adam solver[66] with early stopping and adaptive learning rate, and a maximum number of iteration of 1000.

Classification problems

To generate data for the classification tasks shown in Fig. 6, we draw pairs from a uniform [0,1] distribution to get 500 samples with independence x_1 and x_2 features. Similar to Baltussen *et al.*, 8 different conditions (AND, OR, LINEAR, XOR, CIRCLE, SINE, CONCENTRIC, DOTS) are used to calculate classes labeled 0 or 1. To produce Fig. 6, we used the SVM, MLP and XGB classifiers from SciKitLearn[65] to acquire accuracies on 5-fold cross validation aggregated sets.

Regression and classification with RC

Below are various architectures for prior-input and post-readout layers for all conventional RC systems present in the study. In all cases the reservoir has fixed weights (W) and biases (b) and is transforming the output of the prior layer (X_{prior}) into an input for the post layer (X_{post}): $X_{post} = W X_{prior} + b$. The parameters W and b and those calculated after training the reservoir (cf. 'Building GEM reservoir' subsection)

Fixed prior. In this instance the prior-input layer is not trained, instead, features are one-to-one mapped to medium metabolite compositions. We recall we have 28 metabolites in our media each being either turned on or off (cf. 'Experimental training' set subsection). In the case of classification tasks (cf. Fig. 6), the two x_1 and x_2 input features are transformed into binary vectors of length 14 each, the binary vectors are initialized with zero values and the bit turned on is the one corresponding to the bin of the feature value (modulo 14). In the case of regressions tasks (cf. list in Table 2) the mapping between features and medium metabolites is performed following permutation importance[67]. Precisely, individual features and medium metabolites are first ranked independently depending on their importance toward respectively the problem readout and the growth rate. This is achieved using the ScikitLearn 'permutation_importance' function with a Random Forest Regressor model (with 100 estimators). Ranking is performed by sorting by decreasing importance. Next, the ith ranked feature is mapped to the ith ranked medium metabolites. To obtain an equal number of features and medium metabolites, when the number of features is lower than the number of medium metabolites, features are randomly selected and duplicated, conversely, when the number of features is greater than the number of medium metabolites, features with less importance are ignored. Since the medium composition fed to the reservoir is a binary vector, the feature values need to be binarized; this is achieved by normalizing the feature values and applying a threshold to transform features into 0 (< threshold) and 1 (≥ threshold) values.

Generative prior. The purpose of the generative prior is to produce media compositions fed to the GEM reservoir. Since media compositions are binary vectors it is necessary to transform the features into 0 or 1 values. To that end the features are first processed by a hidden layer with linear activation functions, values are then fed to an output layer with sharp sigmoid activation functions. The sharp sigmoid is a standard sigmoid function where the operand is multiplied by a large constant value.

Selective prior. Instead of generating media compositions, the selective prior selects compositions from a predefined set. This set comprises media compositions previously used to grow *E. coli* and measure growth rates (see subsection 'Experimental Dataset'). To ensure each feature vector in the training set is associated with one of the experimental dataset media, we first process the features through several hidden layers to generate a vector sized to match the number of media. Applying a softmax activation function to this vector yields the probability of selecting each medium from the experimental dataset. However, we need to transform this vector so that the highest probability is set to 1 and all others to 0. To achieve this one-hot encoding, we use Gumbel-softmax[68], a differentiable approximation to argmax when the temperature approaches 0. During training, we start with a high temperature and gradually reduce it at each epoch. We found that using a logarithmic temperature decay provides better results than a linear decay.

Post-readout. A post-readout is trained when the GEM reservoir outputs all reaction rates, these rates are fed to a feedforward neural network which outputs the problem readout (y). When the GEM reservoir outputs only growth rates, we normalize both the growth rate and the problem readout and the problem readout equals the growth rate. In call cases the activation function of the post-readout is ReLU for regression tasks and sigmoid for classification tasks.

The hyperparameters corresponding to the above architectures are summarized in Table 3. Additionally, for all runs carried out in this study and making use of the AMN reservoir , the number of epochs was 1000 for the selective prior architecture and 500 for all others, dropout was set to 0.25 and a learning rate of 10^{-4} was used using the Adam optimizer. Cross-validation was systematically used for all runs and all results reported in the current study (such as R^2 and Accuracy) are computed on aggregated 5-fold cross validation sets. The batch sizes were set to 10 when the training set comprised less than 1000 elements, and 100 otherwise. All runs were carried out in triplicates to compute standard deviations.

Table 3. RC architectures summary. The table provides for each layer the input size, the number of hidden layers, the hidden layer size, the output layer size and the final activation function of the layer. The activation function of all hidden layers is ReLU except for the generative prior where a linear activation function is used.

Reservoir type	Prior input (X)			Reservoir	Post readout (y)
	Fixed*	Generative	Selective		
Physical reservoir	n/a	Variable input size** Hidden number = 1 Hidden size = 28 Output size = 28 Activation = Sharp-sigmoid	Variable input size** hidden number = 3 Hidden size = 280 Output size = 280 Activation = Gumbel-softmax	input size = 28 Output size = 1 (μ=growth-rate)	ReLU(μ) for regression $\sigma(\mu)$ for classification
Conventional reservoir	Variable input size** Output size = 28 Activation = Permutation importance rank	Variable input size** Hidden number = 1 Hidden size = 28 Output size = 28 Activation = Sharp-sigmoid	n/a	input size = 28 output size = All reaction rates ***	Variable input size*** Hidden number = 1 Hidden size = 500 Output size = 1 Activation = ReLU for regression, sigmoid (σ) for classification

* No training performed. ** Depends on the problem input (2 for classification tasks and Table 2 for regression tasks), *** Depends on the number of monodirectional reactions of the GEM model (Table 1)

Phenotype number and entropy calculations

To compute the number of phenotypes for the 10 strains considered in this study we used the 10,000 instances training sets described in the 'Training set' subsection and used to produce Fig. 4. The number of different phenotypes is the number of different flux distributions of all reactions of the models. Several media can lead to the same flux distribution especially if one does not distinguish two distributions when they are separated by a distance lower than a predefined threshold. We chose an arbitrary threshold of 0.01 mol h^{-1} $cell^{-1}$ which is the limit of what can be measured experimentally using a spectrophotometer. We note that the other thresholds we tested (0.1, 0.05, 0.005 and 0.001) produced similar trends in Fig. 4. To obtain the number of phenotypes we used the *fcluster* hierarchical clustering method from SciKitLearn with a distance matrix between flux distributions and a ward linkage.

We also computed the phenotype Shannon entropy (H) for each strain using the formula:

$H = \sum_i p_i log_2(p_i)$ where p_i is the probability of appearance of cluster i , in other words the number of flux distributions in cluster i divided by training set set size (10,000)

REFERENCES

1. Purnick, P. E. M. & Weiss, R. The second wave of synthetic biology: from modules to systems. *Nat. Rev. Mol. Cell Biol.* **10**, 410–422 (2009).
2. Gardner, T. S., Cantor, C. R. & Collins, J. J. Construction of a genetic toggle switch in Escherichia coli. *Nature* **403**, 339–342 (2000).
3. Elowitz, M. B. & Leibler, S. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**, 335–338 (2000).
4. Xiang, Y., Dalchau, N. & Wang, B. Scaling up genetic circuit design for cellular computing: advances and prospects. *Nat. Comput.* **17**, 833–853 (2018).
5. Nielsen, A. A. K. *et al.* Genetic circuit design automation. *Science* **352**, aac7341 (2016).
6. Andrews, L. B., Nielsen, A. A. K. & Voigt, C. A. Cellular checkpoint control using programmable sequential logic. *Science* **361**, eaap8987 (2018).
7. Green, A. A., Silver, P. A., Collins, J. J. & Yin, P. Toehold Switches: De-Novo-Designed Regulators of Gene Expression. *Cell* **159**, 925–939 (2014).
8. Bonnet, J., Yin, P., Ortiz, M. E., Subsoontorn, P. & Endy, D. Amplifying Genetic Logic Gates. *Science* **340**, 599–603 (2013).
9. Farzadfard, F. & Lu, T. K. Genomically encoded analog memory with precise in vivo DNA writing in living cell populations. *Science* **346**, 1256272 (2014).
10. Slomovic, S., Pardee, K. & Collins, J. J. Synthetic biology devices for in vitro and in vivo diagnostics. *Proc. Natl. Acad. Sci.* **112**, 14429–14435 (2015).
11. Kotula, J. W. *et al.* Programmable bacteria detect and record an environmental signal in the mammalian gut. *Proc. Natl. Acad. Sci.* **111**, 4838–4843 (2014).
12. Shin, H. J. Genetically engineered microbial biosensors for in situ monitoring of environmental pollution. *Appl. Microbiol. Biotechnol.* **89**, 867–877 (2011).
13. Borkowski, O., Ceroni, F., Stan, G.-B. & Ellis, T. Overloaded and stressed: whole-cell considerations for bacterial synthetic biology. *Curr. Opin. Microbiol.* **33**, 123–130 (2016).
14. Liu, Q., Schumacher, J., Wan, X., Lou, C. & Wang, B. Orthogonality and Burdens of Heterologous AND Gate Gene Circuits in *E. coli*. *ACS Synth. Biol.* **7**, 553–564 (2018).
15. Snoeck, S., Guidi, C. & De Mey, M. “Metabolic burden” explained: stress symptoms and its related responses induced by (over)expression of (heterologous) proteins in Escherichia coli. *Microb. Cell Factories* **23**, 96 (2024).
16. Lorberbaum, D. S. & Barolo, S. Gene Regulation: When Analog Beats Digital. *Curr. Biol.* **23**, R1054–R1056 (2013).
17. Wang, B., Kitney, R. I., Joly, N. & Buck, M. Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nat. Commun.* **2**, 508 (2011).
18. Meyer, A. J., Segall-Shapiro, T. H., Glassey, E., Zhang, J. & Voigt, C. A. Escherichia coli “Marionette” strains with 12 highly optimized small-molecule sensors. *Nat. Chem. Biol.* **15**, 196–204 (2019).
19. Lee, S. K. *et al.* Directed Evolution of AraC for Improved Compatibility of Arabinose- and Lactose-Inducible Promoters. *Appl. Environ. Microbiol.* **73**, 5711–5715 (2007).
20. Kylilis, N., Tuza, Z. A., Stan, G.-B. & Polizzi, K. M. Tools for engineering coordinated system behaviour in synthetic microbial consortia. *Nat. Commun.* **9**, 2677 (2018).
21. Swain, P. S., Elowitz, M. B. & Siggia, E. D. Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proc. Natl. Acad. Sci.* **99**, 12795–12800 (2002).
22. Ho, J. M. L., Miller, C. A., Parks, S. E., Mattia, J. R. & Bennett, M. R. A suppressor tRNA-mediated feedforward loop eliminates leaky gene expression in bacteria. *Nucleic Acids Res.* **49**, e25–e25 (2021).
23. Tamsir, A., Tabor, J. J. & Voigt, C. A. Robust multicellular computing using genetically encoded NOR gates and chemical ‘wires’. *Nature* **469**, 212–215 (2011).
24. Regot, S. *et al.* Distributed biological computation with multicellular engineered networks. *Nature* **469**, 207–211 (2011).
25. Daniel, R., Rubens, J. R., Sarpeshkar, R. & Lu, T. K. Synthetic analog computation in living cells. *Nature* **497**,

619–623 (2013).
26. Sarpeshkar, R. Analog synthetic biology. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **372**, 20130110 (2014).
27. Rubens, J. R., Selvaggio, G. & Lu, T. K. Synthetic mixed-signal computation in living cells. *Nat. Commun.* **7**, 11658 (2016).
28. Rizik, L., Danial, L., Habib, M., Weiss, R. & Daniel, R. Synthetic neuromorphic computing in living cells. *Nat. Commun.* **13**, 5602 (2022).
29. Li, X. *et al.* Synthetic neural-like computing in microbial consortia for pattern recognition. *Nat. Commun.* **12**, 3139 (2021).
30. Sarkar, K., Bonnerjee, D., Srivastava, R. & Bagh, S. A single layer artificial neural network type architecture with molecular engineered bacteria for reversible and irreversible computing. *Chem. Sci.* **12**, 15821–15832 (2021).
31. Courbet, A., Endy, D., Renard, E., Molina, F. & Bonnet, J. Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates. *Sci. Transl. Med.* **7**, (2015).
32. Moraskie, M. *et al.* Microbial whole-cell biosensors: Current applications, challenges, and future perspectives. *Biosens. Bioelectron.* **191**, 113359 (2021).
33. Sherlock, M. E., Sudarsan, N., Stav, S. & Breaker, R. R. Tandem riboswitches form a natural Boolean logic gate to control purine metabolism in bacteria. *eLife* **7**, e33908 (2018).
34. Silva-Rocha, R. & De Lorenzo, V. Mining logic gates in prokaryotic transcriptional regulation networks. *FEBS Lett.* **582**, 1237–1244 (2008).
35. Oppenheim, A. B., Kobiler, O., Stavans, J., Court, D. L. & Adhya, S. Switches in Bacteriophage Lambda Development. *Annu. Rev. Genet.* **39**, 409–429 (2005).
36. Wilson, R. A., Gibson, R. P., Quispe, C. F., Littlechild, J. A. & Talbot, N. J. An NADPH-dependent genetic switch regulates plant infection by the rice blast fungus. *Proc. Natl. Acad. Sci.* **107**, 21902–21907 (2010).
37. Ficarrotta, V. *et al.* A genetic switch for male UV iridescence in an incipient species pair of sulphur butterflies. *Proc. Natl. Acad. Sci.* **119**, e2109255118 (2022).
38. Hellingwerf, K. J., Postma, P. W., Tommassen, J. & Westerhoff, H. V. Signal transduction in bacteria: phospho-neural network(s) in *Escherichia coli* ? *FEMS Microbiol. Rev.* **16**, 309–321 (1995).
39. Scheres, B. & Van Der Putten, W. H. The plant perceptron connects environment to development. *Nature* **543**, 337–345 (2017).
40. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100–123 (2019).
41. Chen, H. *et al.* Emerging memristors and applications in reservoir computing. *Front. Phys.* **19**, 13401 (2024).
42. Fernando, C. & Sojakka, S. Pattern Recognition in a Bucket. in *Advances in Artificial Life* (eds. Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P. & Kim, J. T.) vol. 2801 588–597 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003).
43. Baltussen, M. G., De Jong, T. J., Duez, Q., Robinson, W. E. & Huck, W. T. S. Chemical reservoir computation in a self-organizing reaction network. *Nature* **631**, 549–555 (2024).
44. Sumi, T. *et al.* Biological neurons act as generalization filters in reservoir computing. *Proc. Natl. Acad. Sci.* **120**, e2217008120 (2023).
45. Srinivasan, S., Cluett, W. R. & Mahadevan, R. Constructing kinetic models of metabolism at genome-scales: A review. *Biotechnol. J.* **10**, 1345–1359 (2015).
46. King, Z. A. *et al.* BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Res.* **44**, D515–D522 (2016).
47. Orth, J. D., Thiele, I. & Palsson, B. Ø. What is flux balance analysis? *Nat. Biotechnol.* **28**, 245–248 (2010).
48. Monk, J. M. *et al.* iML1515, a knowledgebase that computes Escherichia coli traits. *Nat. Biotechnol.* **35**, 904–908 (2017).
49. O'Brien, E. J., Monk, J. M. & Palsson, B. O. Using Genome-scale Models to Predict Biological Capabilities. *Cell* **161**, 971–987 (2015).
50. Nabli, H. An overview on the simplex algorithm. *Appl. Math. Comput.* **210**, 479–489 (2009).
51. Faure, L., Mollet, B., Liebermeister, W. & Faulon, J.-L. A neural-mechanistic hybrid approach improving the predictive power of genome-scale metabolic models. *Nat. Commun.* **14**, 4669 (2023).
52. Ebrahim, A., Lerman, J. A., Palsson, B. O. & Hyduke, D. R. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Syst. Biol.* **7**, 74 (2013).
53. Goñi-Moreno, Á. Biocomputation: Moving Beyond Turing with Living Cellular Computers. *Commun. ACM* **67**, 70–77 (2024).
54. Orth, J. D., Fleming, R. M. T. & Palsson, B. Ø. Reconstruction and Use of Microbial Metabolic Networks:

the Core *Escherichia coli* Metabolic Model as an Educational Guide. *EcoSal Plus* **4**, 10.1128/ecosalplus.10.2.1 (2010).
55. Kavvas, E. S. *et al.* Updated and standardized genome-scale reconstruction of Mycobacterium tuberculosis H37Rv, iEK1011, simulates flux states indicative of physiological conditions. *BMC Syst. Biol.* **12**, 25 (2018).
56. Thiele, I., Vo, T. D., Price, N. D. & Palsson, B. Ø. Expanded Metabolic Reconstruction of *Helicobacter pylori* (*i* IT341 GSM/GPR): an In Silico Genome-Scale Characterization of Single- and Double-Deletion Mutants. *J. Bacteriol.* **187**, 5818–5830 (2005).
57. Nogales, J. *et al.* High-quality genome-scale metabolic modelling of *Pseudomonas putida* highlights its broad metabolic capabilities. *Environ. Microbiol.* **22**, 255–269 (2020).
58. Mo, M. L., Palsson, B. Ø. & Herrgård, M. J. Connecting extracellular metabolomic measurements to intracellular flux states in yeast. *BMC Syst. Biol.* **3**, 37 (2009).
59. Charusanti, P. *et al.* An experimentally-supported genome-scale metabolic network reconstruction for Yersinia pestis CO92. *BMC Syst. Biol.* **5**, 163 (2011).
60. Oh, Y.-K., Palsson, B. O., Park, S. M., Schilling, C. H. & Mahadevan, R. Genome-scale Reconstruction of Metabolic Network in Bacillus subtilis Based on High-throughput Phenotyping and Gene Essentiality Data. *J. Biol. Chem.* **282**, 28791–28799 (2007).
61. Seif, Y. *et al.* Genome-scale metabolic reconstructions of multiple Salmonella strains reveal serovar-specific metabolic traits. *Nat. Commun.* **9**, 3771 (2018).
62. Seif, Y. *et al.* A computational knowledge-base elucidates the response of Staphylococcus aureus to different media types. *PLOS Comput. Biol.* **15**, e1006644 (2019).
63. Neidhardt, F. C., Bloch, P. L. & Smith, D. F. Culture Medium for Enterobacteria. *J. Bacteriol.* **119**, 736–747 (1974).
64. Zampieri, M., Hörl, M., Hotz, F., Müller, N. F. & Sauer, U. Regulatory mechanisms underlying coordination of amino acid and glucose catabolism in Escherichia coli. *Nat. Commun.* **10**, 3354 (2019).
65. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
66. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. Preprint at https://doi.org/10.48550/ARXIV.1412.6980 (2014).
67. Altmann, A., Toloşi, L., Sander, O. & Lengauer, T. Permutation importance: a corrected feature importance measure. *Bioinformatics* **26**, 1340–1347 (2010).
68. Jang, E., Gu, S. & Poole, B. Categorical Reparameterization with Gumbel-Softmax. Preprint at https://doi.org/10.48550/ARXIV.1611.01144 (2016).

ACKNOWLEDGEMENTS

Authors would like to acknowledge funding provided by the ANR funding agency grant numbers ANR-22-PEBB-0008 (PEPR B-BEST France 2030 program) and ANR-21-CE45-0021-01 (AMN project) and the UE HORIZON BIOS program (grant number 101070281). P.S. is supported by the AMN project and A.H. by the BIOS project. We thank Alexandra Martin (INRAE) for her help in acquiring *E. coli* growth rates for different media and Bastien Mollet (INRAE) for providing OpenML regression problems.

DATA AVAILABILITY

Data used to generate all figures are available upon request. All source codes will be released following journal publication.

SUPPLEMENTARY FIGURES

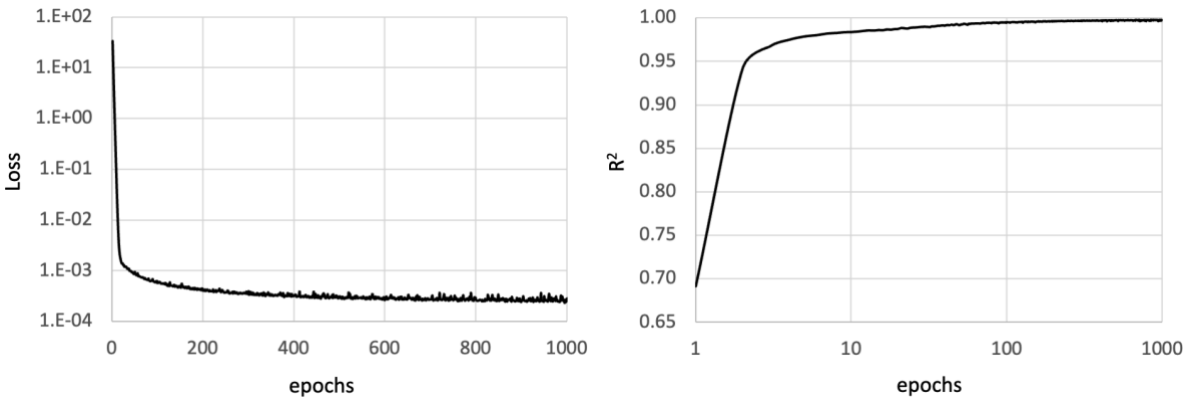

Fig. S1. AMN reservoir training results. *The left plot shows the loss vs. the epoch number. That loss is described in 'Building GEM reservoirs' in the Methods section). The right plot shows the regression coefficient between measured and predicted growth rate vs. the epoch number*.

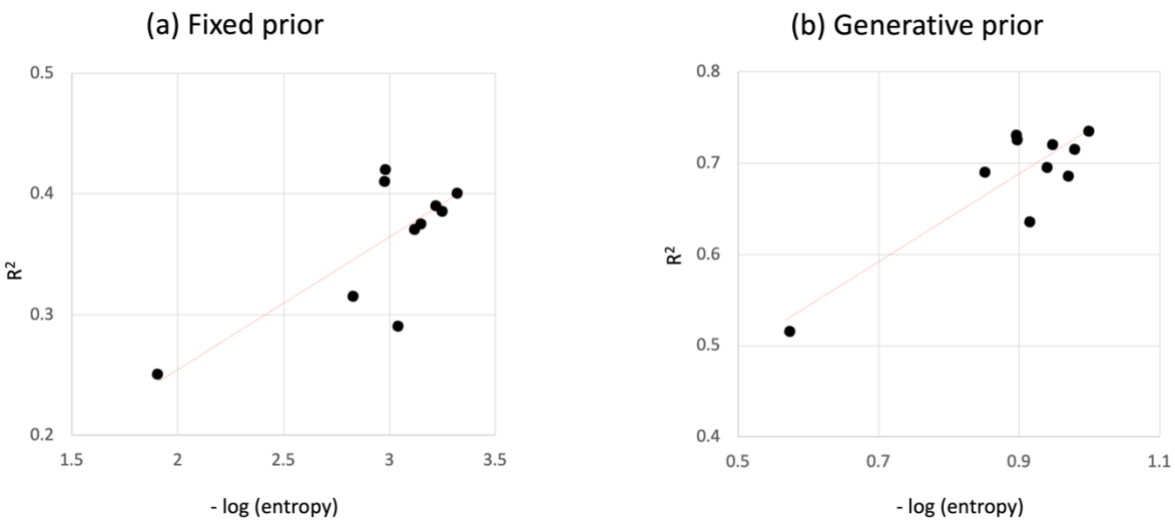

Fig. S2. Regression coefficient vs. log phenotype entropy. *Entropy was computed from the number of phenotypes as described in the Methods section. (a) Median regression coefficient (R^2) over the 10 regression problems of Table 2 vs. number of phenotypes when using a fixed prior. The Pearson correlation coefficient between R^2 values and the linear fit ($y = 0.05\ x + 0.10$) is 0.74. (b) Median regression coefficient (R^2) over the 10 regression problems of Table 2 vs. number of phenotypes when using a generative prior. The Pearson correlation coefficient between R^2 values and the linear fit ($y = 0.48\ x + 0.26$) is 0.87*.

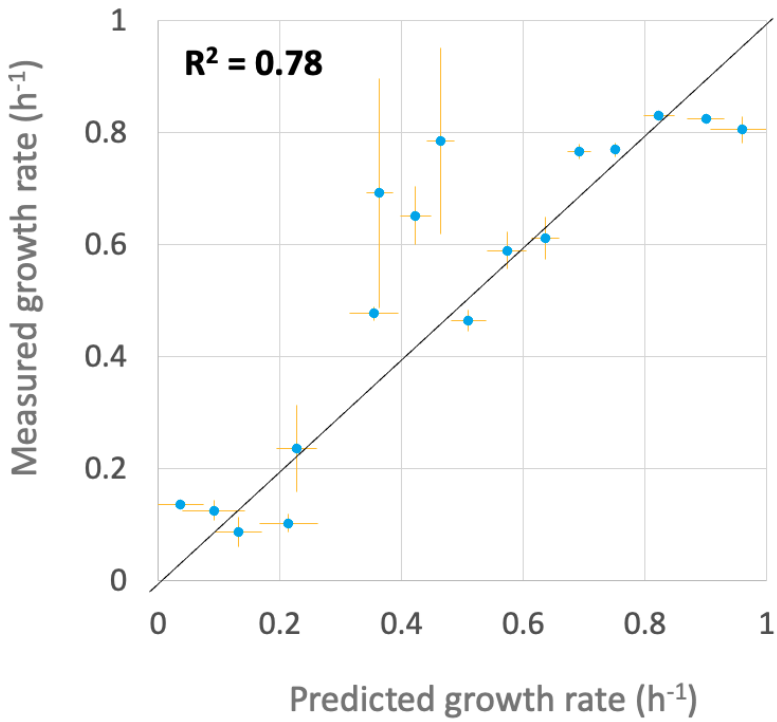

Fig. S3. Measured growth rate vs. predicted growth rate for media output by the generative prior. *Experimentally tested predicted growth rates for a subset of 17 media output by the generative prior. The regression coefficient is consistent with the one calculated on the initial experimental data set (cf. Fig. 2).*

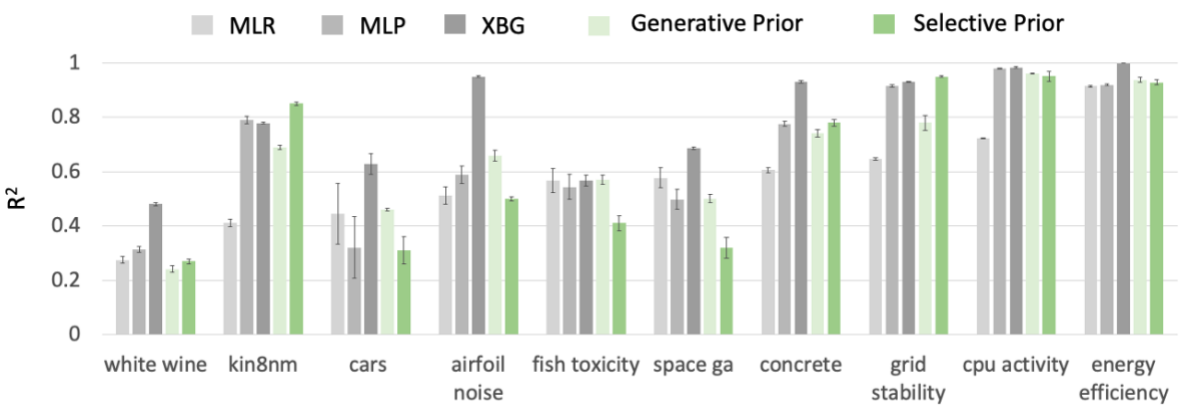

Fig. S4. Regression problem performances with a conventional twin of the physical *E. coli* RC. *The regression problems are those listed in Table 2 and the prior methods are described in 'Regression with RC' in the Method section. R^2 values were acquired on aggregated cross-validation sets (5-fold) standard deviations were computed from three repeats*.