TITLE

Towards AI-designed genomes using a variational autoencoder

AUTHORS

Dudek, N.K.[1,2], Precup, D.[1,2]

AUTHOR AFFILIATIONS

1. School of Computer Science; McGill University; Montreal, QC, H3A 0G4; Canada
2. Mila - Québec Artificial Intelligence Institute; Montreal, QC, H2S 3H1; Canada

KEYWORDS

Microbial genomics, machine learning, VAE, generative AI, bioinformatics, synthetic biology

ABSTRACT

Synthetic biology holds great promise for bioengineering applications such as environmental bioremediation, probiotic formulation, and production of renewable biofuels. Humans' capacity to design biological systems from scratch is limited by their sheer size and complexity. We introduce a framework for training a machine learning model to learn the basic genetic principles underlying the gene composition of bacterial genomes. Our variational autoencoder model, DeepGenomeVector, was trained to take as input corrupted bacterial genetic blueprints (i.e. complete gene sets, henceforth 'genome vectors') in which most genes had been "removed", and re-create the original. The resulting model effectively captures the complex dependencies in genomic networks, as evaluated by both qualitative and quantitative metrics. An in-depth functional analysis of a generated gene vector shows that its encoded pathways are interconnected and nearly complete. On the test set, where the model's ability to re-generate the original, uncorrupted genome vector was evaluated, an AUC score of 0.98 and an F1 score of 0.82 provide support for the model's ability to generate diverse, high-quality genome vectors. This work showcases the power of machine learning approaches for synthetic biology and highlights the possibility that just as humans can design an AI that animates a robot, AIs may one day be able to design a genomic blueprint that animates a carbon-based cell.

SIGNIFICANCE STATEMENT

Genomes serve as the blueprints for life, encoding complex networks of genes whose products must seamlessly interact to result in living organisms. In this work, we develop a framework for training a machine learning algorithm to learn the basic genetic principles that underlie genome composition. This innovation may eventually lead to improvements in the genome design process, increasing the speed and reliability of designs while decreasing cost. It further suggests that AI agents may one day have the potential to design blueprints for carbon-based life.

INTRODUCTION

A requisite goal of synthetic biology is to understand the basic principles of how biological systems function. This involves mapping and understanding the function of the molecular processes in a cell, from which genes encode that process to how those genes interact with one another and with their environment. For perspective, it is estimated that 10^{10}–10^{13} genes across 10^{7}–10^{8} bacterial species have evolved since the dawn of life on Earth over four billion years ago (1, 2).

Mycoplasmas have been proposed as an optimal model for understanding the basic principles of life, as they are the most minimal, and therefore the most “simple”, cells capable of self-sufficient growth (3). Working with this model system, (4) used transposon mutagenesis to identify non-essential genes in the *Mycoplasma mycoides* genome (901 genes), and successfully reduced the genome to encode only the most basic gene set required for life in a laboratory setting (473 genes), including 149 of unknown function. This reduction of life to its bare essentials was a major breakthrough in our understanding of what constitutes life and was hailed as setting the stage for the design of life from scratch. Efforts such as this illustrate that there is structure (i.e. sets of rules) to genome composition.

Genome design has been singled out as a major obstacle hindering advances in synthetic genomics, and also one of the four critical focal areas for technological innovations in the field (5). Machine learning excels at learning complex patterns in data. In particular, the ability to use generative machine learning models such as VAEs to design genomes opens the door to vast possibilities in synthetic biology. Like autoencoders, VAEs are trained to compress and decompress data. While autoencoders encode the input as discrete representations in latent space, with VAEs those representations are probabilistic distributions characterized by a mean and variance. The code is then generated by sampling the probability distribution, and that sample is then decoded. Sampling of the distributions in the latent space is what confers the VAE its generative, rather than discriminative, properties (6). VAEs have previously been used in bioinformatics for tasks including generating novel functional variants of proteins (7), predicting individualized drug response from transcriptomic perturbations (8), and predicting the effects of genetic variation (9). In brief, VAEs learn to map their input to latent space, sample from the latent space, and use that sample to generate a novel instance of the data.

Here, we propose a novel approach to bacterial genome design, which involves training a generative variational autoencoder (VAE) to model the basic tenets of bacterial genome composition (i.e., gene content). First, we source 2,584 high-quality, complete bacterial genome assemblies from the KEGG database, and transform each into a genetic blueprint or “genome vector” that is a binary representation of the genes encoded in each genome. We create 100 heavily corrupted copies of each genome vector by turning off bits representing genes. The VAE is then trained to generate the original genome vector from the corrupted input, such that the output of the model is sets of genes that theoretically encode for viable bacterial cells. An in-depth analysis of a generated genome vector provides evidence that the model has learned to generate largely complete, interconnected pathways, while an AUC score of 0.98 and an F1 score of 0.82 suggest that our model has high discriminative ability, precision, and recall. This work highlights the potential of machine learning algorithms to learn the basic genetic principles that encode for life.

RESULTS

Overview of DeepGenomeVector

In this work we propose and implement DeepGenomeVector, a machine learning approach to model the gene composition of bacterial genomes. This model generates sets of genes that hypothetically encode for viable bacterial cells. Gene sets are represented in the form of “genome vectors”, which we define as a binary representation of all genes encoded by a complete genome.

Annotated bacterial genomes were sourced from a set of 2,584 complete, high-quality, annotated genomes from 46 major bacterial lineages (approximately phyla) from the KEGG genomes database (10–12) (Supplemental Figure 1A, B). Annotations in the form KO identifiers were used to construct genome vectors consisting of 9,863 bits. While KO identifier annotations do not cover the scope of all genes that exist (a feat which is impossible as many genes do not even have known functions), or even likely that are essential for a cell, they do provide annotations for core metabolic and functional processes and are well curated, consistent, high-quality annotations. Genome vectors included in the full dataset for this work (train + test sets combined) encode a median of 1,885 genes, with a range of 528 - 4,536 genes per genome (Supplemental Figure 1C) Genes in the KEGG database are further organized into modules, which are functional units of gene sets. For example, KEGG modules encode reactions such as glycolysis, the pentose phosphate pathway, methanogenesis, denitrification, etc.

One of the biggest obstacles to using data-driven machine learning methods to generate genome vectors is that the total number of complete, high-quality genome assemblies is limited. After splitting genomes into training (~90%) and testing (~10%) datasets, we created 100 corrupted versions of every genome vector. Each corruption was achieved by retaining genes encoding 10 randomly selected modules encoded by the real genome. These corrupted genome vectors served as input to an ML algorithm.

From the generative algorithms family, a VAE was selected for this task due to the strong performance of VAEs on generative modeling tasks and the ability of an end-user to evaluate a generated output against the input (or the uncorrupted input) in a high-throughput manner. Indeed, one of the greatest challenges in this project was how one could evaluate the realism and potential viability of a genome vector. Using a VAE permits one to evaluate the similarity of the output to the original, uncorrupted input. Note that methods such as the Fréchet Inception Distance (13) require prohibitively large sample sizes for the limited dataset available here. The encoder and decoder of our trained model, DeepGenomeVector, consist of three layers each, with a latent representation size of 100 neurons (Figure 1A), as this architecture was determined to yield superior model performance during the training phase (see Methods). DeepGenomeVector was trained to optimize (i.e. minimize) the sum of binary cross-entropy (BCE) loss and Kullback-Leibler divergence (KLD). Quantitative performance during development was evaluated using the F1 score, such that the output of the VAE was compared against the original, uncorrupted genome vector from which the corrupted input to the model was derived.

Learning curves showing loss and F1 score as evaluation metrics show that the model was able to learn principles underlying genome composition (Figure 1B, C). To gain insight into the model’s tradeoff between sensitivity and specificity, we plotted an ROC curve (Figure 1D**)**. A

micro-average ROC-AUC score of 0.98 suggests the DeepGenomeVector model has high discriminative ability.

Quantitative evaluation of DeepGenomeVector's performance

To obtain a measure of performance, we started by quantitatively evaluating how similar genomes generated from corrupted inputs were to the original, uncorrupted genome from which they derived. Model performance was evaluated using a test set of 27,900 corrupted genomes derived from 279 real genomes (~10% of all genomes). DeepGenomeVector's F1 score of 0.82 on the test set substantially outperformed those of five baseline models (Table 1). The DeepGenomeVector model obtained F1 scores ranging from 0.99-0.39, with a median of 0.83 (Supplemental Figure 2). Adjusting the degree of corruption (i.e. the number of modules input to the model) substantially affected model performance (Supplemental Table 1). We further evaluated the fidelity of the model with respect to the presence of user-input genes in the generated genome (Supplemental Figure 3). In 85% of cases, 90% or more of the original input genes were also present in the output generated genome.

To gain insight into where the model over- vs under- performed, we examined biological and data structure correlates of F1 score. Both taxonomic group (approximately phylum) and metabolic process effected model performance (Kruskal-Wallis test, $p = 0$ and $p = 1.01e^{-154}$, respectively), with Chlorobi test genomes and photosynthetic genes having the highest median F1 scores (Supplemental Figure 4). Notably, genes pertaining to processes that are often acquired via horizontal gene transfer, such as drug resistance, tended to score poorly, as one might expect from modules not inherently tied to any given genome composition or lifestyle.

With regards to data structure, DeepGenomeVector's performance may be mechanistically correlated with the number of training examples in the dataset (both at the level of genome vector genus and gene occurrence counts), how strongly correlated certain genes are correlated with other genes, the complexity of each genome (as a rough proxy, the number of genes it encodes), as well as the number of training set genome vectors that encode all ten modules used as input to the VAE to generate a test set genome vector (Supplemental Figure 5). Interestingly, post-hoc analysis of the model outputs suggests that DeepGenomeVector has learned a strategy whereby it uses fewer genes but more complete genetic modules per genome vector than a typical bacterial genome encodes (Supplemental Figure 6), despite the fact that it was provided with no *a priori* information about which genes function together in any given module.

To gain insight into where the model performs quantitatively well or not, we investigated genome vectors with the best, median, and worst F1 scores. The 10 input modules used for the best scoring generated genome vector were derived from *Brucella abortus* (F1 = 0.99). The generated genome vector had high similarity to other *Brucella* genome vectors (Figure 2A), containing only one gene that is not in the two closest training set genomes (as measured by the Hamming distance). Meanwhile, the 10 input modules used for the median scoring generated genome vector were derived from *Desulfotomaculum ruminis* (F1 = 0.83). When looking at the overlap between the generated genome vector and the two closest in the training set, the generated genome vector appears to be a blend of the other two (and likely others), with 28 / 1255 not appearing in either (Figure 2B). Finally, in the worst scoring genome vector in the test set (F1 = 0.39), the model was given ten input modules from a representation of a

member of the *Sodalis* genus, which is an endosymbiont of the ground bug *Henestaris halophilus* (Figure 2C). Of note, endosymbionts have typically undergone extreme genome reduction (this one encodes only 768 genes). In line with this, the poor F1 score is driven by a large number of false positives (17.56% false positives, compared to 2.34% false positives across the full test). The ten input modules to the VAE were all present in 94 genomes in the training set, implying that numerous other viable pathways exist by which a genome with those modules could be reconstructed. Rather than reconstruct the outlier *Sodalis spp.* genome vector, DeepGenomeVector generated a genome vector characteristic of members in the Family *Erwiniaceae*.

Qualitative evaluation of DeepGenomeVector's performance

As quantitative evaluation metrics are limited in their ability to evaluate the realism of a genome vector, we next sought to gain a deeper qualitative understanding of how the real vs. generated genomes compare. First, we aimed to gain a sense of the distance between real and generated genomes. To do so, we calculated and plotted the Jaccard similarity between test genomes and a set of 100 generated genomes, using their Hamming distance, which is the number of bits that differ between two vectors (Figure 3A). While the distribution of generated genomes does appear to be slightly shifted from that of the real genomes, there is nonetheless a significant overlap in the distribution of real and generated genomes. Also noteworthy is that the distance between any given generated genome to its nearest real genome neighbour is smaller than the distance between the two most distantly related real genomes.

Another way of estimating the similarity of the generated genome vectors to real ones is to consider how similar the lifestyle of organisms they could encode for would be. This is arguably a richer comparison metric than one based on pure distance, as distance does not capture which genes are different and what the significance of such differences may be. Therefore to give a qualitative perspective on whether the composition of generated genome vectors is realistic, we placed the real and generated genomes in a parsimony-based dendrogram, built from a character matrix of gene presence/absence. We hypothesized that all the generated genomes clustered in a single branch of the dendrogram, while the real genomes clustered within another, it would suggest that the composition of genes was seriously flawed and unrealistic. Conversely, if the generated genome vectors cluster within the real ones, it supports that they are qualitatively similar.

The dendrogram shows that real and generated genome vectors are both distributed across the dendrogram, with generated genomes often bounded within well-defined clades of real genomes from a taxonomic group (e.g.: Alphaproteobacteria) (Figure 3B). This suggests that the composition of generated genome vectors is not deeply aberrant from that of real genomes and that DeepGenomeVector is able to generate genome vectors representative of organisms from across the tree of life (albeit skewed towards certain groups, likely reflective of the data imbalance in the training set and in nature itself).

Functional profile of an AI-designed genome vector

Metrics such as F1 score give an estimate of whether the general gene content of a generated genome vector is correct. Missing, however, is information on critical parameters such as whether metabolic pathways are complete, whether they are properly linked together,

whether they encode plausible networks / lifestyles, and whether core cellular machinery is present. To gain insight into these questions, we performed an in-depth functional analysis of an example AI-generated genome vector, dubbed “g13” (generated #13). This genome vector was selected quasi-randomly, with the sole criterion being that its position in the PCA ordination in Figure 3A (see arrow) overlap with the distribution of real genomes.

To qualitatively address whether metabolic networks are complete and properly linked together, we created a metabolic reconstruction diagram depicting genome vector g13’s central carbohydrate metabolism, energy metabolism, and the 10 original input modules to the VAE and how they connect to the former (Figure 4; Supplemental Data 1). This is a small, meaningful subset of the full metabolic network that can be assessed by a human and correlates with one of the study goals which is to design genomes encoding user-defined functional properties. Of 30 modules, only two are partial: guanine ribonucleotide biosynthesis (missing one gene) and anaerobic photosynthesis (missing light harvesting complexes 1 and 2, and cytochrome C2). This is striking when considering, for example, that all enzymes required for a complex process such as oxidative phosphorylation (n=41) are present. The genome does lack the ability to synthesize ubiquinol, however, which is a component of the electron transport chain. Remarkably, modules were appropriately connected together. For example, the original input module C1-unit interconversion was connected to the genomes’s central carbohydrate and energy metabolism via the PRPP => IMP and guanine ribonucleotide biosynthesis that the model encoded.

We next sought to gain insight into the ecological plausibility and theoretical viability of an organism hypothetically encoded for by genome vector g13. Generated genome g13 encodes 2,015 genes, including 50/51 universal bacterial single-copy genes often used to assess bacterial genome completeness in metagenomics studies (14) (missing aspartyl tRNA synthetase). Core cellular machinery was near complete. For example, all genes required for an RNA degradosome were all present, while 9/10 required for the RecFOR homologous recombination pathway were encoded. The genome vector encodes genes that would allow an organism to utilize compounds such as acetate and trans-cinnamate as carbon and energy sources. The Calvin cycle is also encoded, suggestive of a capacity for carbon fixation. The genome vector encodes glycolysis, the Entner-Doudoroff pathway, oxidative decarboxylation of pyruvate, the TCA cycle (forward direction), and oxidative phosphorylation, and would thus theoretically support aerobic respiration. The genetic potential to generate energy from inorganic compounds is present via both the assimilatory sulfate reduction and nitrogen fixation pathways. Anaerobic photosynthesis is partially encoded in g13. The lifestyle encoded by genome vector g13 is reminiscent of a bacterium capable of both heterotrophic and lithoautotroph growth, and would theoretically display a high metabolic diversity.

DISCUSSION

In this work, we presented a proof-of-concept that machine learning algorithms can learn the basic genetic principles underlying genome composition. In-depth functional analysis of a generated genome vector suggests that it is near-complete and encodes largely intact pathways that are interconnected. When evaluating whether the model was able to generate genome vectors similar in nature to those used as input seeds, quantitative evaluation metrics support strong model performance, with DeepGenomeVector achieving an AUC score of 0.98 and an F1 score of 0.82 on the test set.

While the model presented here serves as a proof of concept that gene repertoires can be generated by machine learning algorithms, there are limitations to this study. From the biological perspective, a major limitation is that our features only included genes annotated with a KO number from the KEGG dataset. This means our study excludes many genes that have evolved in nature but for which humans have not yet studied and characterized their function. Genes of unknown function can play essential roles in cellular biology;for example, when (4) engineered a minimal bacterial genome, they found that the minimal viable gene set included 149 genes of unknown function that were absolutely essential for producing a viable organism. Another important contextual consideration for this study is that genomes are more than just “bags of genes”. They include intergenic spacing, promoters, transcription factor binding sites, and more. Furthermore, a major piece of the problem that would need to be addressed for the true design of genomes, rather than genome vectors, is figuring out how to convert binary vectors of 0’s and 1’s to gene sequences. The field of synthetic biology is ripe with challenges that machine learning may help to address, with the potential to provide both valuable scientific and engineering insights.

From a machine learning perspective, one area for future improvement to the DeepGenomeVector approach could be to impose sparsity constraints that reduce the metabolic diversity of output genomes. Quantitative analysis showed that generated genomes encoded more modules than would be expected from real bacterial genomes. In line with this, the metabolic analysis of generated genome g13 was suggestive of a highly metabolically diverse bacterium, for example as demonstrated by the capacity for chemoautolithotrophy using both nitrogen and sulfate as energy sources. From an evolutionary perspective, a sparsity constraint would be akin to imposing the microevolutionary forces in nature that shape bacterial genome architecture (15). For example, it has been theorized that resource specialization is driven by competition through natural selection (16, 17). The realities of living in real environments, such as the intense competition that exists, are not reflected in our model.

Another tantalizing possibility would be to employ GANs or even language-like models that generate sequences for the purposes of generating artificial genome blueprints. A major barrier to doing so, however, is the limited amount of data available. Here, we circumvented this issue by creating corrupted versions of genomes for input to the model, which allowed us to train and evaluate the VAE. Critically, using a VAE allowed us to compare the original, uncorrupted genome vector against the genome vector generated from the corrupted input. This provided a rough evaluation of the quality of the generated product, albeit an imperfect one as many combinations of genes can be viable but may not be represented in the genome set available. Evaluation methods that are used to evaluate the realism of outputs of generative

models in other fields, such as using the Frechet Inception Distance (13) for evaluating photo realism, require prohibitive amounts of data for this application. Methods for evaluating genome realism that are minimally data-hungry would broaden the scope of future approaches (e.g.: GANs) to genome vector design. Similarly, ever increasing dataset sizes will provide greater flexibility and breadth to machine learning applications in microbial genomics.

DeepGenomeVector serves as proof-of-concept that a machine learning algorithm can learn and model the basic tenets of how genomes, and thus life forms, are encoded. The development of such technologies may eventually lead to their use in synthetic biology, including incorporation into synthetic genomic CAD software (18). From a more philosophical perspective, this work highlights the possibility that AI agents will one day be able to design viable carbon-based lifeforms.

CODE AVAILABILITY STATEMENT

Code for this project is available on Github at https://github.com/natasha-dudek/DeepGenomeVector.

METHODS

Genome selection, annotation, and representation

Genome annotations were acquired from the KEGG knowledge base (10–12) which can also be used to link sequence information to high-level cellular functions (e.g.: pathways). The KEGG genomes database includes a set of 6542 complete reference and representative genomes from NCBI that have been annotated using KEGG's internal pipeline (accessed May 14, 2020, includes Eukaryotic, Archaeal, and Bacterial genomes). The KEGG bacterial genomes are strongly phylogenetically skewed. As such, we performed thinning to reduce the representation of over-represented taxa. For each genome, we used the taxid to recover the NCBI taxonomic identification. First, any genome not identified to the phylum level was discarded. Next we randomly selected only one genome to represent each species, and no more than five species to represent each genus. Per phylum we allowed for up to 50 genomes unclassified at the species level. Finally, we removed genomes with incomplete KEGG annotations, fewer than 500 annotated genes / genome (presumed symbionts with highly atypical lifestyles compared to the "average" bacterium), and genomes encoding fewer than 10 KEGG modules (same rational re: symbionts). The final dataset consisted of 2,584 bacterial genomes spanning 46 major lineages (approximately phyla) and 9,863 features. For greater detail into the rationale behind these decisions, see the Supplemental Methods.

Genomes can be thought of as blueprints for life that either encode a gene or don't. Thus we represented each genome as a vector of 1's and 0's representing whether, for each gene in the dataset's 9,863 genes, a given gene is encoded.

Train-test split and corruption process

Genomes were split into a training set (~90%) and testing set (~10%) prior to undergoing corruption. For major lineages (approximately phyla) represented in our dataset by ≥10 genomes, 88% were assigned to the training set and 12% to the test set. For those represented by 2 - 9 genomes, 50% were assigned to each of the training and test sets. Finally, for lineages represented by a single genome, that genome was assigned to the training set.

Corrupted genomes were used as input to the VAE. Corruptions were achieved by turning on the genes from 10 randomly selected KEGG modules that naturally co-occur within real genomes, and turning all other bits off. This was done 100X per genome in the training set and 100X per genome in the test set. The extent to which a genome was corrupted had a significant impact on the quality of generated genomes, as measured by the F1 score (Table 2). We decided to use the genes from 10 modules as input in an effort to minimize the number of genes a user would theoretically need to specify while simultaneously avoiding the rapid drop in performance that comes with using genes from fewer than ~10 modules.

Machine learning model

The DeepGenomeVector model was built in PyTorch v1.5.0 (19)**.** Hyperparameter tuning was implemented in Tune v0.8.2 (20) and performed using 10-fold cross-validation with AdamW as the optimizer (21, 22). Leaky ReLU activation functions (23) were applied to all layers except

the output, for which a sigmoid activation was applied. Correspondingly, He initialization (24) was used for all layers except the last, for which Xavier initialization (25) was used. Sweeps over the following hyperparameters were conducted: number of layers (1, 2, 3, 4), batch size (32, 64, 128, 256), learning rate (log uniform selection between $1e^{-4}$ - $1e^{-1}$), and weight decay (log uniform selection between $1e^{-5}$ - $1e^{-2}$). While the number of layers had a strong impact on model performance, the effect of other hyperparameters was negligible. The final model used 3 layers for the encoder and decoder each, batch size 128, learning rate 0.001, and weight decay 0.1. At every iteration, loss was calculated by summing together binary cross-entropy (BCE) loss and Kullback-Leibler divergence (KLD). Training was performed over 10 epochs. When the model is used to generate a genome, the probabilities for each bit being on are converted to 0 or 1 using a 0.5 replacement threshold.

Parsimony-based dendrogram

The 279 test genomes, 100 generated genomes, and an archaeal genome were used to create a parsimony-based dendrogram with the Wagner method (26, 27). This was achieved using the mix program in Phylip v3.695 (28). The input order of species was randomized 10 times using seed=99 and using the archaeal genome as an outgroup root. All other settings were set to default. The dendrogram was visualized using iTOL v5.7 (29).

Metabolic reconstruction of a generated genome

Identification of metabolic pathways was achieved using the KEGG Mapper “reconstruct” tool, available at https://www.genome.jp/kegg/mapper/reconstruct.html (30, 31).

WORKS CITED

1. E. Koonin, M. Y. Galperin, *Sequence — Evolution — Function: Computational Approaches in Comparative Genomics* (Springer Science & Business Media, 2002).

2. H. C. Betts, *et al.*, Integrated genomic and fossil evidence illuminates life's early evolution and eukaryote origin. *Nat. Ecol. Evol.*

3. H. J. Morowitz, The completeness of molecular biology. *Isr. J. Med. Sci.* **20**, 750–753 (1984).

4. C. A. Hutchison 3rd, *et al.*, Design and synthesis of a minimal bacterial genome. *Science* **351**, aad6253 (2016).

5. N. Ostrov, *et al.*, Technological challenges and milestones for writing genomes. *Science* **366**, 310–312 (2019).

6. D. P. Kingma, M. Welling, Auto-encoding variational bayes. *arXiv preprint arXiv:1312. 6114*.

7. A. Hawkins-Hooker, *et al.*, Generating functional protein variants with variational autoencoders. *PLoS Comput. Biol.* **17**, e1008736 (2021).

8. L. Rampášek, D. Hidru, P. Smirnov, B. Haibe-Kains, A. Goldenberg, Dr.VAE: improving drug response prediction via modeling of drug perturbation effects. *Bioinformatics* **35**, 3743–3751 (2019).

9. A. J. Riesselman, J. B. Ingraham, D. S. Marks, Deep generative models of genetic variation capture the effects of mutations. *Nat. Methods* **15**, 816–822 (2018).

10. M. Kanehisa, S. Goto, KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28**, 27–30 (2000).

11. M. Kanehisa, Y. Sato, M. Kawashima, M. Furumichi, M. Tanabe, KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Res.* **44**, D457–62 (2016).

12. M. Kanehisa, M. Furumichi, M. Tanabe, Y. Sato, K. Morishima, KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Res.* **45**, D353–D361 (2017).

13. M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Adv. Neural Inf. Process. Syst.*

14. J. Raes, J. O. Korbel, M. J. Lercher, C. von Mering, P. Bork, Prediction of effective genome size in metagenomic samples. *Genome Biol.* **8**, R10 (2007).

15. L.-M. Bobay, H. Ochman, The Evolution of Bacterial Genome Architecture. *Front. Genet.* **8**, 72 (2017).

16. F. Fiegna, T. Scheuerl, A. Moreno-Letelier, T. Bell, T. G. Barraclough, Saturating effects of species diversity on life-history evolution in bacteria. *Proc. Biol. Sci.* **282** (2015).

17. D. Lawrence, *et al.*, Species interactions alter evolutionary responses to a novel environment. *PLoS Biol.* **10**, e1001330 (2012).

18. S. M. Richardson, *et al.*, Design of a synthetic yeast genome. *Science* **355**, 1040–1044 (2017).

19. A. Paszke, *et al.*, Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.*

20. R. Liaw, *et al.*, Tune: A Research Platform for Distributed Model Selection and Training. *arXiv [cs.LG]* (2018).

21. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412. 6980*.

22. I. Loshchilov, F. Hutter, Decoupled weight decay regularization. *arXiv preprint arXiv:1711. 05101*.

23. A. L. Maas, A. Y. Hannun, A. Y. Ng, Others, Rectifier nonlinearities improve neural network acoustic models in *Proc. Icml*, (Atlanta, GA, 2013), p. 3.

24. K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification in *2015 IEEE International Conference on Computer Vision (ICCV)*, (IEEE, 2015) https:/doi.org/10.1109/iccv.2015.123.

25. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research., Y. W. Teh, M. Titterington, Eds. (PMLR, 13--15 May 2010), pp. 249–256.

26. M. A. Chang, M. O. Dayhoff, R. V. Eck, M. R. Sochard, “Atlas of protein sequence and structure” (1965).

27. A. G. Kluge, J. S. Farris, Quantitative Phyletics and the Evolution of Anurans. *Syst. Biol.* **18**, 1–32 (1969).

28. J. Felsenstein, *PHYLIP (phylogeny inference package), version 3.5 c* (Joseph Felsenstein., 1993).

29. I. Letunic, P. Bork, Interactive Tree Of Life (iTOL) v4: recent updates and new developments. *Nucleic Acids Res.* **47**, W256–W259 (2019).

30. M. Kanehisa, Y. Sato, KEGG Mapper for inferring cellular functions from protein sequences. *Protein Sci.* **29**, 28–35 (2020).

31. M. Kanehisa, Y. Sato, M. Kawashima, KEGG mapping tools for uncovering hidden features in biological data. *Protein Sci.* **31**, 47–53 (2022).

FIGURES AND TABLES

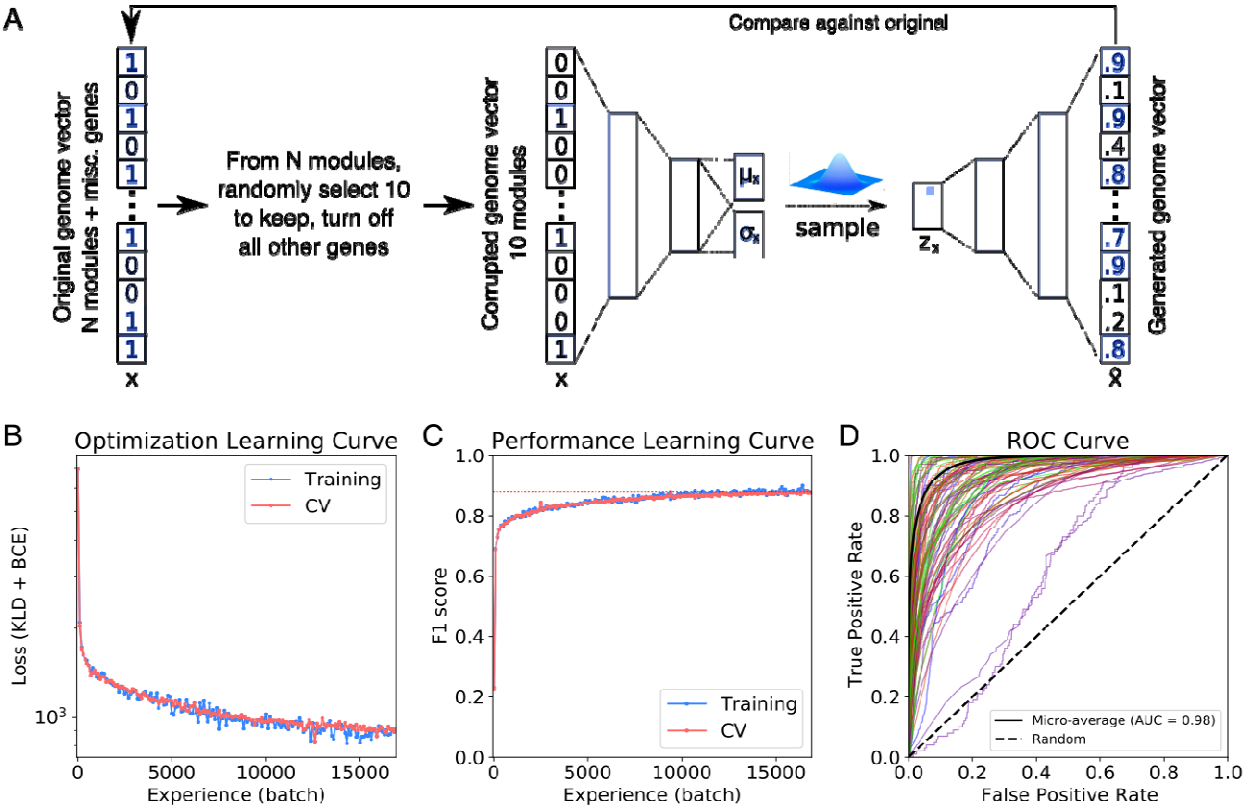

Figure 1: Overview of DeepGenome. (A) Data corruption process and the DeepGenome architecture. Genomes are represented as binary vectors (genome vectors) that denote which genes are encoded. To create input for the model, 100 corrupted versions of each genome vector are created by randomly selecting 10 functional modules it encodes, and creating a new corrupted genome vector that only encodes the genes from those 10 modules. The input to the model is a corrupted genome vector (x), where the presence (1) / absence (0) of genes is encoded. The model consists of three encoder layers (9,863, 500, 250, and 2 x 100 neurons) and three decoder layers (100, 250, 500, 9,863 neurons). The latent representation (z_x) is sampled from the multivariate Gaussian distribution parameterized by μ_x and σ_x. The output from the model is a probabilistic reconstruction of the corrupted input genome vector ( ). Per gene probabilities output by the model are binarized using a threshold of 0.5. The performance of the VAE is measured by comparing the generated genome vector against the original, uncorrupted one from which it was derived. **(B)** Optimization learning curve, with batch loss (KLD + BCE) on the y-axis. Training was performed over 10 epochs. **(C)** Performance learning curve, with F1 score on the y-axis. The maximum cross-validation fold F1 score of 0.88 is indicated with a red dotted line. **(D)** Receiver operating characteristic (ROC) curve. Each coloured line represents one of 100 randomly selected genes (for clarity only a subset of the full 9,863 genes is visualized). The micro-average is plotted in solid black and corresponds to a micro-average AUC score of 0.98.

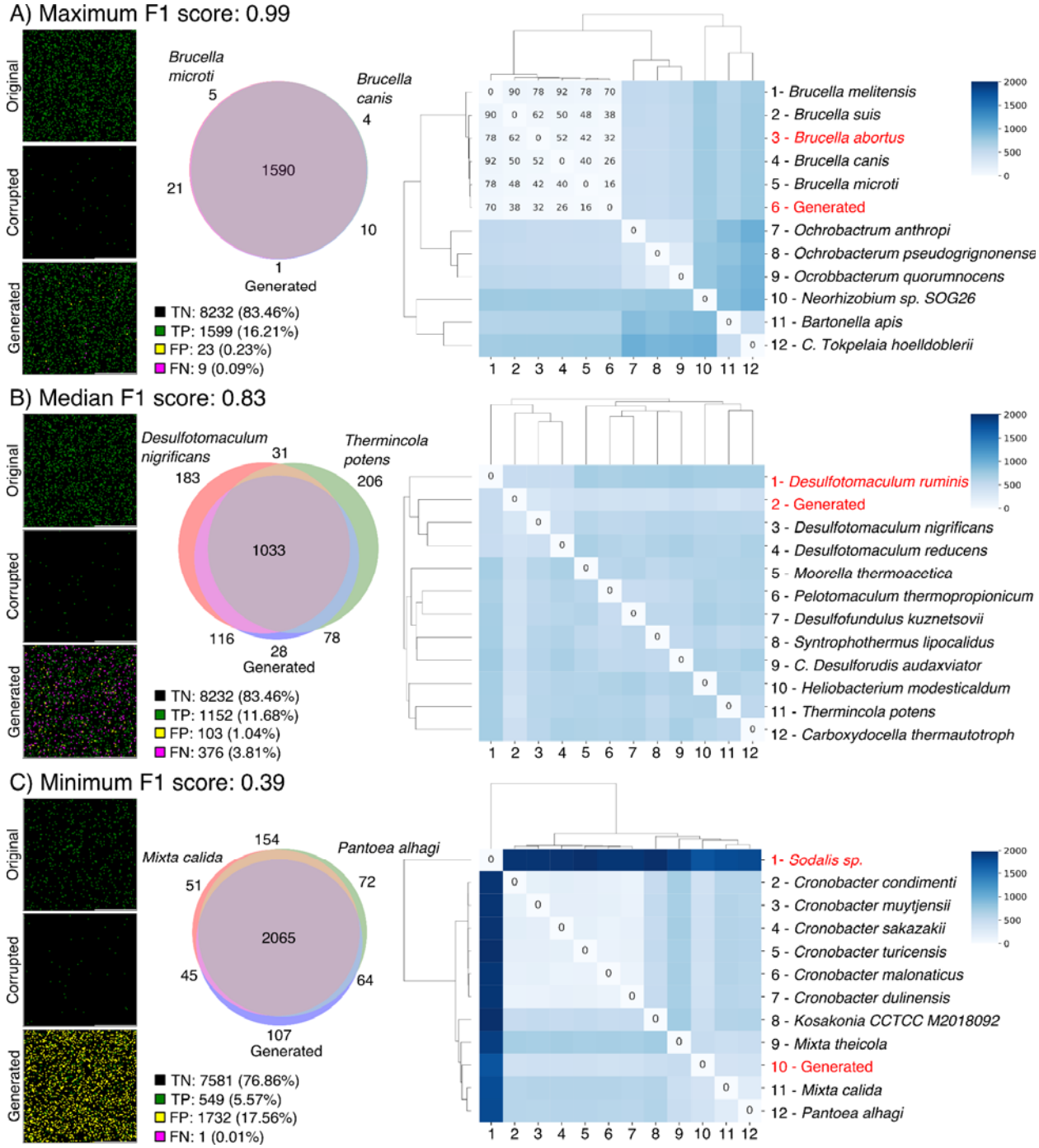

Figure 2: Investigation of the best, median, and worst scoring generated genome vectors in the test set. (A,B,C) Panels represent data from the best, median, and worst scoring genomes in the test set. For each original genome vector in the dataset, corrupted genome vectors were synthesized and input to the model. For each generated genome that resulted, we compared how similar it was to the original genome vector from which the input used to

generate it was derived. **(Left)** On the right, the three vertically stacked boxes represent original (uncorrupted), corrupted, and generated / reconstructed genome vectors. Black pixels represent bits (genes) that are off, while green pixels represent bits that are turned on. In the final box, yellow pixels represent false positives, while pink pixels represent false negatives. **(Center)** For each generated genome vector, the top two most similar genome vectors from the training set were identified via the Hamming distance. The Venn diagram shows the number of genes encoded by between the three genome vectors. **(Right)** The ten training set genome vectors most similar to the generated genome vector were identified, as well as the original genome the VAE was trained to reconstruct (highlighted in red). The heatmap depicts the Hamming distance between the genome vectors. In cases where the Hamming distance between a pair of genome vectors is <100, the number of substitutions is shown.

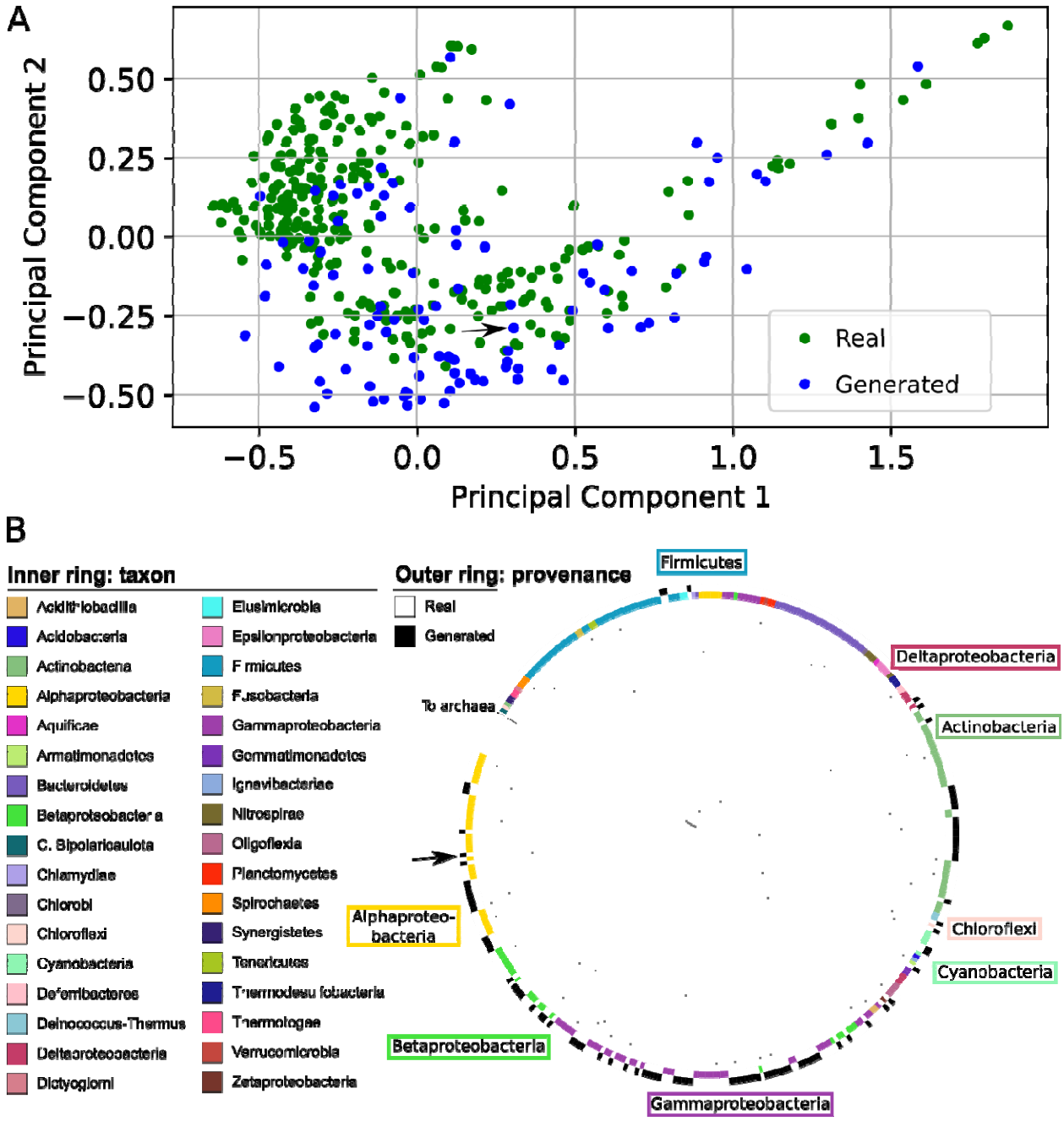

Figure 3: Qualitative comparison of real vs generated genome vectors. The 279 real genome vectors from the test set were compared against 100 generated genome vectors with randomly selected input modules. The generated genome vector selected for functional analysis (**Figure 6**) is denoted by an arrow in both panels. **(A)** Principal component analysis (PCA) Jaccard similarity ordination of real (green) vs generated (blue) genome vectors. Jaccard similarity was calculated using the Hamming distance metric. Axes are scaled in proportion to the variance explained by each principal component. **(B)** Parsimony-based dendrogram of real vs generated genome vectors. The dendrogram was generated from a character matrix where each gene's presence/absence in a genome vector is a character. The inner ring around the dendrogram is a colour-coded representation of genome vector phylum (real genome vectors

only), while the outer ring indicates whether a genome vector was real (white) vs generated (black). Names of major lineages (approximately phyla) whose clades in the dendrogram neighbour or include generated genome vectors are indicated for reference.

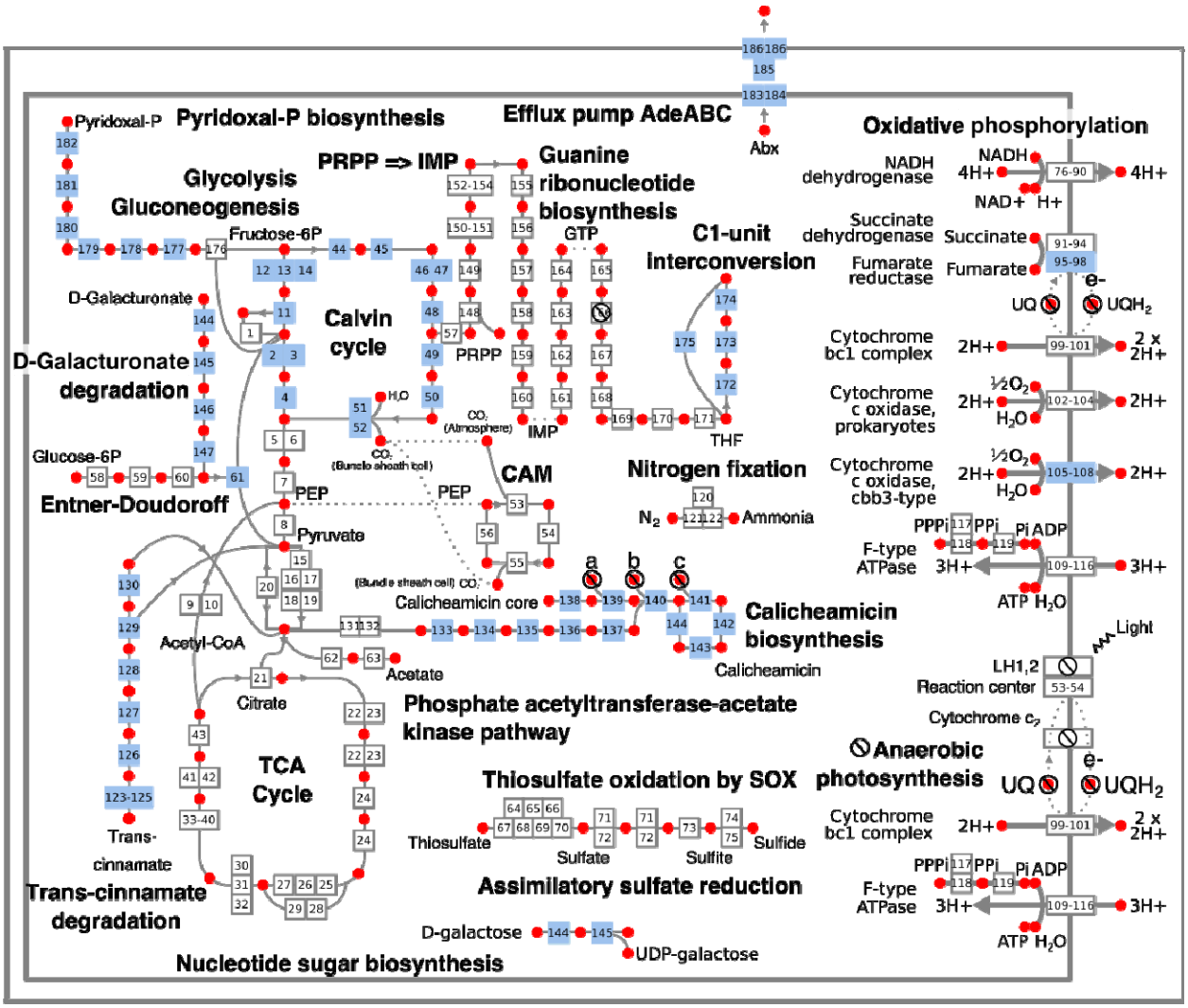

Figure 4: Cell diagram for a generated genome, depicting central carbohydrate metabolism, energy metabolism, and the 10 modules used as input. Boxes represent reactions catalyzed by genes encoded in the generated genome vector g13, while red dots represent key compounds involved in reactions. The 10 modules used as input to the model are colour-coded blue. Cross icons indicate genes that are not encoded or compounds whose synthesis is not encoded for. Metabolic pathways that are encoded by generated genome vector g13 but not directly related to central carbohydrate metabolism, energy metabolism, or an input module are not shown. Numbers in boxes allow for the identification of the reaction/gene via **Supplemental Data 1**.

TABLES

	AUROC	F1	TN	FP	FN	TP
Baseline 1	0.52	0.19	68.49%	16.77%	11.39%	3.36%
Baseline 2	0.49	0.16	67.95%	17.3%	11.91%	2.83%
Baseline 3	0.50	0.23	42.54%	42.72%	7.32%	7.42%
Baseline 4	0.62	0.38	**83.89%**	**1.37%**	10.98%	3.76%
Baseline 5	0.76	0.52	70.15%	15.11%	4.34%	10.40%
DeepGenome Vector	**0.98**	**0.82**	82.92%	2.34%	**2.91%**	**11.83%**

Table 1: DeepGenomeVector's performance on the test set compared to baseline models. AUROC and F1 scores are micro-averages. True negatives (TN), false positives (FP), false negatives (FN), and true positives (TP) are the percentage of each, respectively, across all genomes in the test dataset. For baselines 1 & 2, let n = a randomly selected number of genes encoded by a real genome from the training set. Baseline 1 consisted of randomly turning on n bits in a corrupted genome vector from the test set. Baseline 2 consisted of randomly turning on the n bits with the highest probability of being on across the entire training set. Baseline 3 consisted of making predictions using an untrained VAE model. Baseline 4 consisted of always predicting the smallest real bacterial genome vector in the training set (*Hoaglandella endobia*, Gammaproteobacteria). Baseline 5 consisted of always predicting the largest bacterial genome vector in the training set (*Paraburkholderia caribensis*, Betaproteobacteria).