Building a cell-free metabolic model using variational autoencoders

Nicholas L. Larus-Stone
Queens' College

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Department of Computer Science and Technology
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: nl363@cl.cam.ac.uk

June 4, 2018

Declaration

I Nicholas L. Larus-Stone of Queens' College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 0

Signed:

Date:

Abstract

This is the abstract. Write a summary of the whole thing. Make sure it fits in one page.

Programmable matter (the synthetic biology aspect)

Contents

List of Figures

List of Tables

Acronyms

μL Microliter. 19, 20

E. coli Escherichia Acid. 5, 13–16

AA Amino Acid. 18, 20, 44

AE Autoencoder. 16

cAMP Cyclic AMP. 18, 44

CFPS Cell-free Protein Synthesis. 3, 4, 7–9, 15, 19–23, 29, 30, 44

CoA Coenzyme-A. 44

COBRA Constraint-Based Reconstruction and Analysis. 3, 22

CSV Comma separated Values. 21, 22, 24

DNA Deoxyribonucleic Acid. 3, 18–20, 39, 44

FBA Flux Balance Analysis. 2–5, 8, 9, 13, 15, 16, 22–24, 26, 27, 37, 39

GEM Genome scale Models. 3, 4, 14, 16, 27, 29, 30, 38

HMP Hexose Monophosphate. 20, 44

K Potassium. 20, 44

KL Kullback-Leiber. 12

LB Lysogeny Broth. 18

LP Linear Programming. 13

ME-Model Metabolic and gene Expression Models. 14

Mg Magnesium. 20, 44

MILP Mixed Integer Linear Programming. 14

mM Micromolar. 44

NAD Nicotinamide Adenine Dinucleotide. 18, 44

nL Nanoliter. 20

NTP Nucleoside Triphosphate. 18

OD Optical Density. 18

PCA Principal Component Analysis. 4, 10, 15

PEG Polyethylene Glycol. 20

PEMA Principal Element Mode Analysis. 15

ReLU Rectified Linear Unit. 26

RFP Red Fluorescent Protein. 19

RNA Ribonucleic Acid. 3

TL Translation. 20

tRNA Transfer RNA. 18, 44

TX Transcription. 20

VAE Variational Autoencoder. 2, 4, 5, 11, 12, 16, 24–27, 29–32, 35, 39, 40

Chapter 1

Introduction

Despite more than a century of active biology research, creating good biological models is still difficult. What makes a computational model good? Most importantly, a good model needs to accurately describe how the biological system acts in the real world. A model is only useful if it accurately represents the biological system and can be used to help answer research questions. This allows experiments and hypotheses to be tested using a model instead of in vivo, speeding up the pace of research. A good model should also be consistent–running it multiple times should produce the same results. Though biological experiments can have wide variance due to external conditions, we want our model to vary only due to intrinsic biological sources of variation. Exact reproducibility is very difficult in laboratory environments, but a computer model can and should eliminate the extrinsic sources of variance that one encounters in a laboratory setting. Finally, the best models not only describe the system, but also facilitate deeper insights into the underlying biology. A good model should help researchers improve their understanding of the system that is modeled and provide a starting point for novel insights.

Biological organisms are complex systems that can be examined at many different levels. For example, one model could use a physical description of the system to predict the shape of an cell as it grows. A different model could

use -omic data—genomic, transcriptomic, and metabolomic measurements—to model the production of compounds that allow for cellular growth. As all models are inherently limited in some way, so the choice of the type of model that is used to describe a system is extremely important.

This dissertation has built an end-to-end system that produces metabolic models for cell-free systems. Our system takes in experimental data and a metabolic model of an organism and produces a reduced metabolic model for the relevant cell-free system. In particular, we use a Variational Autoencoder (VAE) with a custom loss function on top of a common metabolic modeling technique called Flux Balance Analysis (FBA). Our pipeline learns which reactions are most important for a given cell-free system and then removes the irrelevant reactions. The goal is to provide a model for biologists to better understand what is occurring in cell-free systems. This will allow them to anticipate how different starting conditions might affect their experiments before actually running the experiment.

Metabolic models Metabolism is defined as all of the chemical reactions that occur within an entity–usually an organism or a cell–in order to sustain its life and growth. Biologists are interested in how the metabolism of a system works as it is the end result of the other biological process of interest (e.g. transcription and translation). Ongoing work continues to develop a more robust understanding of metabolism, while recent advances have begun to engineer the metabolism. Metabolic engineering is particularly interesting for its potential to create novel advances in medicine and energy production [1]. One of the most important tools in aiding metabolic engineers are metabolic models. These models are used to predict how the phenotype of the organism will respond to various environmental inputs or changes performed by the metabolic engineer.

Most metabolic models represent the chemical reactions in a cell by a mathematical representation of the reaction coefficients. Individual reactions are represented as equations in these models. These reactions can then be constrained based on our knowledge of biological pathways. Models of this type are collected under the heading of constraint-based reconstruction and

analysis (Constraint-Based Reconstruction and Analysis (COBRA)) models [2] and are very common among the metabolic engineering community [3]. In particular, Flux Balance Analysis (FBA), is one of the most popular metabolic modeling techniques that has been used in hundreds of different works on metabolism [4]. These applications range from optimizing metabolic pathways [5] to investigating gene networks [6]. FBA is based on a genome-scale metabolic models (GEMs) reconstructed from the genome of an organism of interest. However, these GEMs describe living organisms, not cell-free systems, thus most FBA models cannot be applied out of the box to a cell-free system.

Cell-free systems Cell-free protein synthesis (CFPS) systems are a reduced version of cells that allows these systems to produce proteins without being alive. The Central Dogma of biology, as enunciated by Francis Crick, is that Deoxyribonucleic Acid (DNA) makes Ribonucleic Acid (RNA) makes protein [7]. While this may be a slight oversimplification, the point remains that the production of protein is the end goal for biological systems. CFPS systems take that to an extreme by removing all endogenous DNA and RNA and membranes while retaining the machinery for transcription and translation. A CFPS system is therefore a type of programmable matter, a veritable biological computer. The CFPS system acts like a computer because it can "execute" any DNA "program" that is added to the mixture. The biologist determines the output of the CFPS system by customizing the DNA that he or she adds to a cell-free system. The combination of this simplicity with the inexpensive nature of cell-free systems have made them popular for an array of applications [8, 9, 10].

CFPS systems have two key aspects that make it an ideal platform to model for this dissertation. First of all, running experiments in a CFPS system is much faster than typical *in vivo* methods because executing an experiment does not require growing cells. This meant that we were able to start from scratch and generate relevant data within a few months instead of the many months or years that a cell-based system could take. CFPS systems are also simpler than a full cellular system, and should therefore be easier to

model accurately. By definition, CFPS systems contain a strict subset of the reactions that occur in a cell-based system. However, one issue is that we do not know which reactions are included in that subset.

Despite their reduced nature, there is a scarcity of good models for CFPS systems. Our CFPS systems are composed of blended cells and therefore we do not know exactly what is in the system. This makes modeling CFPS systems difficult. However, few prior attempts at modeling CFPS systems have used metabolic models to examine CFPS systems. We believe that metabolic models are a natural fit to model CFPS systems. Since CFPS are not living organisms, their effectiveness is entirely based on reactions constraints and energy regeneration. Thus, it makes sense to examine these systems from the perspective of metabolism. The few existing metabolic models for CFPS systems are hand-constructed by a specific lab. We use the standard GEMs that already exist for the original organism, providing a more scalable and generalizable way to generate these models of CFPS systems.

Contributions This dissertation makes a number of contributions in the field of metabolic modeling for CFPS systems.

- Firstly, we introduce a new loss function for VAEs that incorporates a correlation loss term with experimental data. Although we focused specifically on using it for cell-free systems, the loss function could be used in any application with an external data source.

- Secondly, we provide one of the first applications of deep learning or autoencoders to FBA models. Principal Component Analysis (Principal Component Analysis (PCA)) is the main dimensionality reduction technique in biology, so this work provides a proof of concept of the utility of using autoencoders instead.

- We also produced a number of new, reduced models for CFPS systems that can be used by biologists working in this field.

We note that our approach was designed to be as general as possible. Our

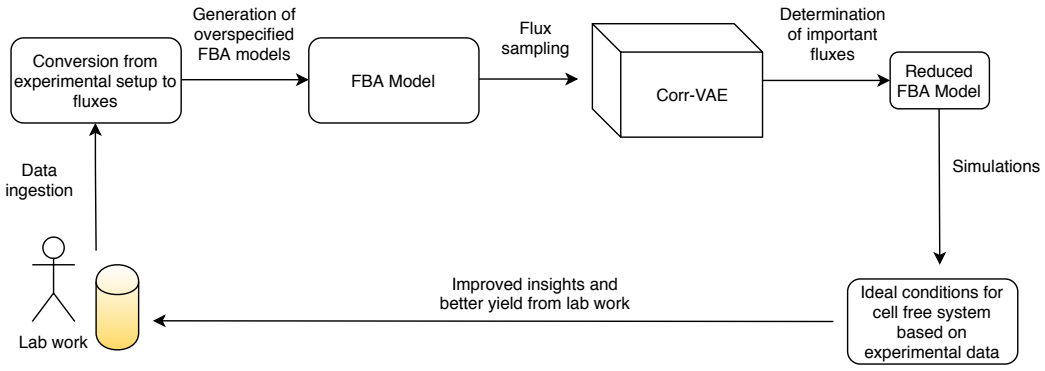

Figure 1.1: A figure showing the overall pipeline for the system to generate cell-free metabolic models.

system can out-of-the-box apply to other types of cell-free systems; i.e. systems that use organisms other than *Escherichia coli* (Escherichia Acid (*E. coli*)) as their base. Additionally, the techniques developed within can be applied to any biological system that uses experimental data and metabolic models.

Outline

The structure of this dissertation is as follows. Chapter 2 provides background information on cell-free systems, FBA, and VAEs. Chapter 3 provides related work in the fields of modeling cell-free systems, FBA, reduction of metabolic models, and VAEs. Chapter 4 describes the system we have designed and built to model cell-free systems. The entire pipeline is shown in fig 1. Chapter 5 describes the results of the experiments. This includes results showing the improvement of our reduced model over naive models. We conclude with a chapter of discussion and directions of future work in this area.

Chapter 2

Background

2.1 Cell-free systems

CFPS systems have existed since the early 1960's as a way to express proteins that are otherwise difficult to express in a living cell [11]. These CFPS systems were quite crude and had low protein yields, limiting their utility. More recently, work involving the removal of certain genes allowed for stabilized amino acid synthesis leading to improved yield of CFPS systems [12]. Another crucial development was the ability to ensure cofactor regeneration pathways were maintained in the CFPS system [13].

When discussing CFPS systems, it is important to distinguish between the multiple types of cell-free systems. One type of cell-free system is created by combining individual purified proteins involved in transcription and translation. The most widely utilized of these reconstituted cell-free systems is the PURE system [14]. The benefit of these systems is that the system is completely known and controlled. Since the only things in the system are the proteins that have been deliberately aded, users of the systems can be assured that there are no undesirable elements such as nucleases or proteases present. However, these systems have relatively high costs due to the time and effort required to isolate and purify each of the requisite proteins.

The other type of CFPS system is an extract-based system created using a crude cell extract. This is the type of CFPS system we will be using throughout this dissertation. Creating an extract-based CFPS system involves growing cells and then blending them into cell extract. This extract is then used as the basis of the CFPS system with the assumption that all the important transcription/translation machinery will be in the cell extract. Growing up large quantities of cells is relatively cheap, so these types of systems have the potential to scale more effectively than reconstituted cell-free systems. The downside to this way of creating CFPS systems is that the exact composition of these cell extracts is unknown and could potentially vary between batches of CFPS systems. This motivates our work to provide a system that can accurately model these crude CFPS systems.

2.2 Flux Balance Analysis

Flux Balance Analysis (FBA) is one of the most common metabolic modeling tools. FBA relies on a mathematical representation of all of the metabolic reactions occurring in an organism. These reactions can be represented using a matrix S, where each row represents a separate reaction and each column is a metabolite. Entries in this matrix consist of the stoichiometric coefficient for each metabolite in the reaction, where metabolites that are consumed are represented using negative numbers. Flux through each of these reactions can be represented by a vector v, which is a 1-dimension vector with a length equal to the number of the reactions. FBA then makes the assumption that the system is at steady state, so the overall flux through all of the reactions is not changing. We can represent this by writing the following:

$$S * v = 0 \tag{2.1}$$

where S is the known stoichiometric matrix and v is the unknown flux vector.

This system is underspecified–we typically have more reactions than metabolites–and therefore there is no unique solution to this equation. We deal with

this issue by specifying reaction constraints and choosing an objective function. We can specify constraints on different reactions in order to limit the amount of flux proceeding through a given reaction. For instance, if we want to represent a certain reaction as irreversible, we could set the lower bound on that reaction to be 0, specifying that it can only proceed in one direction. When choosing an objective function, we can either choose a specific reaction to optimize the flux through or create an artificial reaction that contains metabolites we care about and optimize for that. A common choice to optimize for is the Biomass Objective Function, which incorporates many essential metabolites necessary for cell growth [15]. Our choice of objective functions will be discussed further in Section 4.2.2 For now, we can represent our objective function as Z, which we can specify using c, a vector that selects which reactions are part of the objective function.

With the constraints and objective function in mind, we can reformulate our problem to be linear programming problem of the form:

$$maxZ = c^T vs.t.S * v = 0and - 1000 < v_0 < 1000and0 < v_1 < 1000... \quad (2.2)$$

where v_i is the flux through a reaction. We can then use a linear programming solver to find a feasible solution to this problem. It may be useful to imagine the potential space of all FBA solutions that can satisfy the constraints as a hypercube. Then, the linear programming solver finds an optimum at one corner of the hypercube based on the objective function.

The use of FBA involves this key steady state assumption–i.e. the system is stable. CFPS systems don't have the same type of steady state as a typical cell where intake and output are equal. However, we can consider the steady state of a CFPS system to be the period of maximal production. Thus, we can use FBA to analyze this pseudo-steady state and optimize that time of production, which will lead to greater protein production overall.

A typical FBA model includes thousands of reactions–this is extremely high dimensional data. In order to extract insights from the data, we need to use dimensionality reduction techniques.

2.3 Autoencoders

Autoencoders are a dimensionality reduction technique with a very simple idea: given a dataset, try to reconstruct that dataset by passing it through a lower dimensional subspace. This is performed by training an encoder network that learns how to map from the original dataset to the lower dimensional data and a decoder network that can map from the low dimensional subspace back to the subspace of the original data. The original idea behind autoencoders was that this lower dimensional representation of the data functioned as a compressed version of the data. Instead of hand designing compression/decompression algorithms, these functions can be learned to be application specific. These autoencoders are often worse than hand-designed compression algorithms and have difficulty competing with classic algorithms on problems such as image compression [16]. However, the lower dimensional compressed representation can be used as a dimensionality reduction technique to uncover non-obvious relationships within the original data. This is similar to the idea behind Principal Component Analysis (PCA), though PCA projects the data into a subspace in a linear manner, while autoencoders are able to learn non-linear transformation.

The implementation of autoencoders usually uses neural network layers to learn the encoding and decoding functions. Figure 2.3 demonstrates an example structure of a very simple autoencoder. There are a number of different types of autoencoders that are extensions of this basic idea. One example is a sparse autoencoder. In order to force the autoencoder to learn a more general dimensionality reduction, we can add a sparsity constraint to limit the number of activations among the network layers. This forces the autoencoder to learn a representation with sparse network weights. Another type of autoencoder is a denoising autoencoder. This type of autoencoder involves taking a noisy representation and mapping it back to a clean representation of the data. This can either be trained on noisy data or it can be given the original representation, corrupt it, and then try to learn the original representation from the corrupted data.

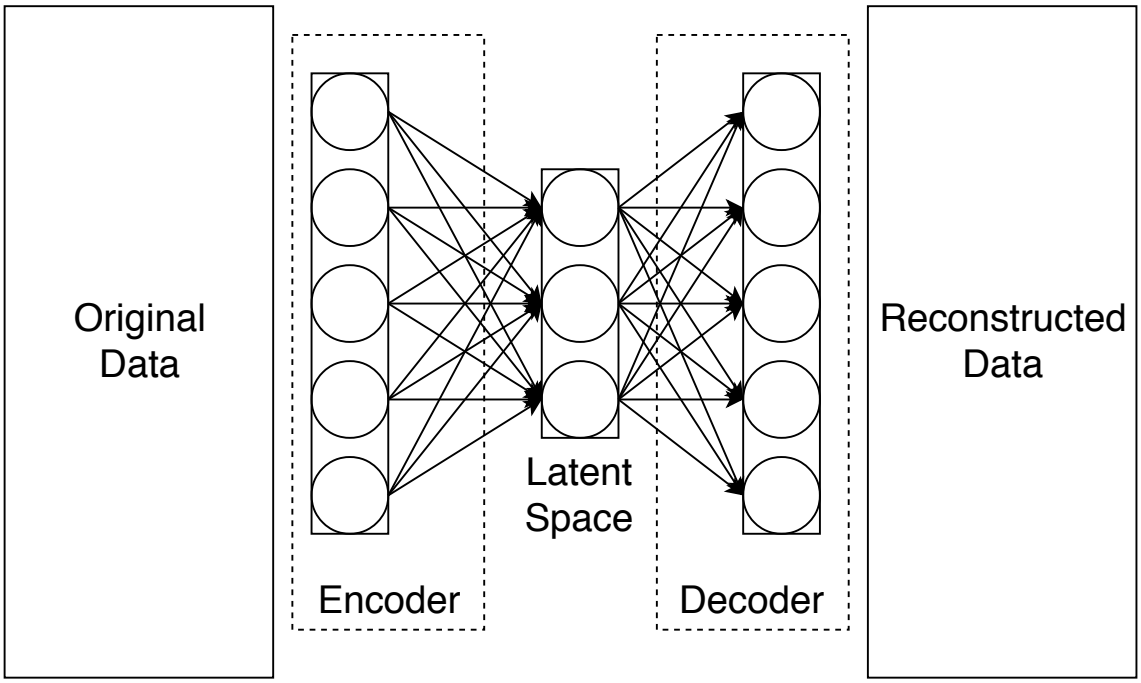

Figure 2.1: Example of a single layer autoencoder. In this case, both the encoder and the decoder are a single layer neural network. These layers can be stacked to create even more complex dimensionality reductions.

Sometimes we want to do more than just learn arbitrary encoding and decoding functions. For instance, we might want to impose constraints on the encoded representation, also known as the latent space. Variational autoencoders (VAEs) are a class of autoencoder that does exactly this. VAEs constrain the encoded representation to be a probability distribution. So, instead of learning functions that encode/decode the data, the VAE actually learns the parameters of the encoded probability distribution. Specifically, the encoder learns to map samples to a mean and standard deviation, which is our latent space. Then, when decoding the sample, we can sample randomly using the encoded mean and standard deviation as parameters for our probability distribution.

We measure loss in this scheme through a combination of two loss terms. First, we still care about how well we are able to reconstruct our original data. So, we can have a reconstruction loss term which measures how well

the reconstructed data maps to the original data. Our second loss term constrains the latent space to ensure that it is well formed and provides a regularization term against overfitting. We use Kullback-Leiber (Kullback-Leiber (KL)) divergence, a common way of measuring difference between probability distributions. We can calculate the the KL divergence between our latent space and our prior distribution, which in most cases is a normal distribution. With the combination of those two loss terms, we can then train the VAE as we would any other neural network–by using gradient descent to minimize the loss function.

We denote $x_i \in X$ as our data and z as our latent representation. Our encoder network can be represented by the function p which has parameters θ. Similarly, our decoder can be represented as a posterior distribution q that is parameterized by ϕ. Then the loss function of a VAE can therefore be written out as follows:

$$\mathcal{L}(\theta, \phi) = -E[\log p_\theta(x_i|z)] + KL(q_\phi(z|x_i)||p(z)) \tag{2.3}$$

Chapter 3

Related Work

This chapter introduces related work in the relevant fields of computer science and biological modeling. In particular, we examine work in the field of building metabolic models using flux balance analysis. Next, we present a series of automatic reduction systems for metabolic models. None of these systems have been applied to modeling cell-free systems, though we present other versions of models. Finally, we investigate recent work in the field of variational autoencoders.

3.1 Flux Balance Analysis

Early metabolic models in the 1980s pioneered the use of stoichiometric equations to describe a biological system and predict the yield of a specific product [17]. This work was soon improved through the use of linear programming (Linear Programming (LP)) to solve for the optimal result [18]. After the transition to using LP solvers, this field was referred to as constraint based analysis, while the primary technique used to solve these equations is called FBA. These techniques were soon applied to describe the main metabolic systems in *E. coli* [19]. Importantly, FBA has repeatedly been shown to predict phenotypes that corresponded to real-world data [20, 21, 22, 23].

Improvements on these early models have primarily included the addition of new genes, metabolites, and reactions [24]. In particular, after the first *E. coli* genome was sequenced and annotated, the size of these models grew rapidly. The earliest genome scale model (GEM) for *E. coli* was iJE660a, a model that accounted for 627 unique reactions in a typical *E. coli* cell [25]. Over the years, more and more reactions were progressively added to the model to better describe the biology. This work will use the most recent *E. coli* GEM, iJO1366 from 2011, which contains 2583 reactions [26]. Since 2011, work in this field has focused on extending metabolic models to incorporate more biological information. For instance, recent work has involved the addition of gene expression data to create metabolism and gene expression models (Metabolic and gene Expression Modelss (ME-Models)) [27]. The most recent ME-Model, iJL1678-ME has 79871 reactions.

3.2 Reduction of metabolic models

As these GEMs begin to incorporate even more information, the issue of high dimensionality arises. To deal with this issue, a number of papers have tried to reduced these genome scale models to only the most essential reactions. One group has developed multiple tools to reduce the dimensionality of these models. redGEM performs a graph search with the objective to minimize information loss to find core metabolic models [28]. Similarly, lumpGem identifies and collapses reaction networks into single balanced reactions in order to reduce the overall dimensionality of the model [29]. The other major tools in this area are NetworkReducer and minNW. NetworkReducer iteratively prunes and compresses reaction networks while maintaining a set of protected reactions until a core model is found [30]. minNW uses mixed-integer linear programming (Mixed Integer Linear Programming (MILP)) techniques to compute minimal subnetworks with certain constraints [31].

All of these methods are searching for minimal core models using only the stoichiometric description of the system. However, there has also been work

that uses general dimensionality reduction techniques such as Principal Component Analysis (PCA). This idea was incorporated into a search technique called Principal Element Mode Analysis (Principal Element Mode Analysis (PEMA)) [32]. PEMA searches for subnetworks that maximize the observed variance similar to how the principal components of PCA work. More recent methodology called "Principal metabolic flux mode analysis" explicitly incorporates PCA and FBA [33]. They reduce the dimensionality of the system by running PCA on the stoichiometric matrix while using the concepts from FBA as regularization.

We differ from earlier reduction techniques in a few ways. Firstly, none of these techniques have incorporated deep learning methods. In addition, these techniques all deal with the stoichiometric matrix in the abstract without having actual experimental data to learn from. Finally, none of the techniques above have specifically targeted CFPS systems.

3.3 Modeling cell-free systems

Models of cell-free systems have typically been based on kinetic models of transcription and translation. These models have a narrow focus because they only examine reactions involved in transcription and translation. Those reactions can then be described using differential equations with known rate constants to describe the rate of transcription and translation. This type of model has been applied to the commercial *E. coli* cell-free system TXTL, and was able to accurately describe the dynamics of their gene circuit of interest [34]. Other work has proposed a general framework to model metabolic networks in cell-free systems using these kinetic models [35]. Even more recent work in this field includes using Bayesian parameter inference to infer the kinetic parameters for these reactions in less well studied organisms [36].

There have only been a few attempts to use metabolic models and FBA to model cell-free systems. One model attempted to adapt a full-scale *E. coli* model to a cell-free system by hand [37]. The authors decided which parts

of the full model were relevant for cell-free systems and then removed any irrelevant reactions from the model. A more recent attempt took a bottom-up approach to building a FBA model for cell-free systems [38]. Instead of removing reactions from the full *E. coli* GEM until they had a cell-free model, the authors instead selected a few pathways they felt were crucial for a cell-free system. They then built a model using only the reactions they had selected as important for protein synthesis. These models show that it is possible to use FBA to describe cell-free systems,. However, neither approach is able to construct cell-free systems in a systematic way that could scale to other labs or other organisms.

3.4 Variational autoencoders

Autoencoders have been around for many years, but VAEs were first introduced only in 2014 [39, 40]. Since their introduction, VAEs have been used for everything from transferring image features [41] to molecule generation [42]. Due to their ability to function as generative networks, much of the work in the field has focused on applying VAEs to images. VAEs have only just begun making their way into analyzing biological data. Recent work includes using a VAE on cancer transcriptomics data to extract meaningful features of cancer gene expression [43].

Work has also been done to incorporate correlation terms with Autoencoders (AEs). Correlation Neural Networks were first introduced as a way to make AEs more effective on multi-modal data such as labeled images [44]. By maximizing the correlation between the latent representations for each view of the data, the authors were able to improve performance. This idea has been expanded in Relational Neural Networks, which also looks at correlation within the data to affect the AE loss term [45].

Chapter 4

Design and Implementation

The primary deliverable of this dissertation is a system (shown in Figure 1) that is able to generate metabolic models of cell-free systems. Our system has four major parts. First, we ingest experimental data and incorporate it into a standardized format. Next, we convert the data into a group of cellular metabolic models and sample fluxes from those models to create a dataset. Then, we perform dimensionality reduction to elucidate underlying trends in the experimental data. Finally, we use those insights to reduce the original, overspecified models to standalone cell-free metabolic models.

4.1 Data gathering

In order to build a robust model that reflected biological reality, we first had to generate the data we wanted to use for training. Gathering high quality biological data is difficult and was crucial key to the success of the system as a whole. We describe the high level process of creating a cell-free system and running the experiments below. The full experimental protocol can be found at dx.doi.org/10.17504/protocols.io.kz2cx8e.

4.1.1 Cell-free systems

As shown in Fig 4.1.1, creating a cell-free protein expression system involves three main steps: growing and lysing cells, supplementing with energy substrates, and adding custom DNA constructs. Our protocol is based off of the open protocol out of the Federici lab [46]. We begin by growing up 1L of BL21 *E. coli*cells in an overnight culture of Lysogeny Broth (LB) until they have reached optical density (Optical Density (OD)) of 1.6. Then, we spin down the cells and remove the supernatant, storing the pellet in the refrigerator overnight if necessary. Next, we perform 3 more sets of spins and washes with S30 buffers to remove any extracellular debris. Finally, we use a bead beater to lyse the cells and spin one last time to remove the cellular debris and beads. At the end of this process we have cell extract which can be stored in a -80°freezer.

Once we have this cell extract, we have the core cellular machinery that is necessary for transcription and translation. However, we need to add the cofactors and reactants that are important for the transcription and translation reactions. So, we add energy substrates to the cell extract to create a cell-free protein expression system. These substrates include energy substrates such as Cyclic AMP (cAMP), maltodextrin, and Nicotinamide Adenine Dinucleotide (NAD). They also include vital parts of transcription and translation such as Nucleoside Triphosphates (NTPs), AAs, and Transfer RNAs (tRNAs). See Table 4.1 for a complete listing of the reactants and their purpose in the system. Table A.1 shows the final concentration for each substrate.

Given this cell-free expression system, we have essentially assembled a biological computer. Whatever DNA "program" is added, the system will work to produce the appropriate protein result. This platform is incredibly powerful because it allows us to easily insert these DNA programs and see results within a few hours. A typical biological timeline would take far longer because living cells would have to be coerced to uptake the DNA and incorporate it into their production process.

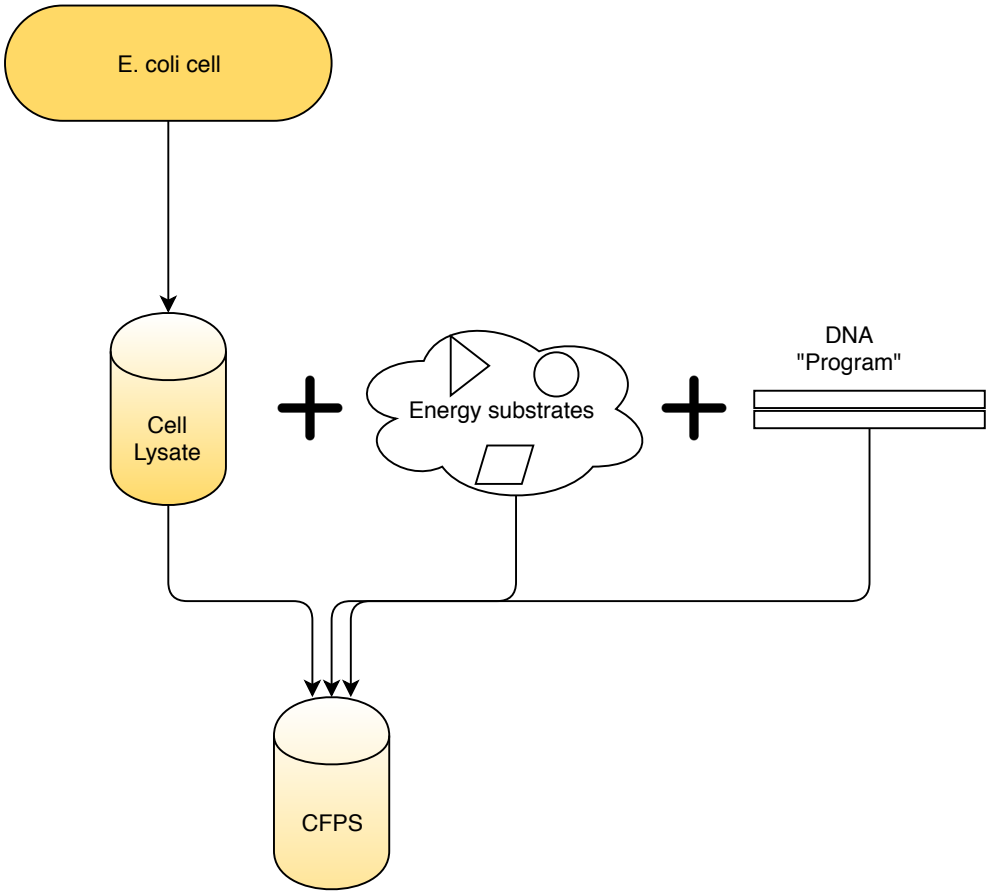

Figure 4.1: Overall process for creating a CFPS

4.1.2 Datasets

We used the protocol above to create two datasets for training. The first dataset was created by hand and followed standard lab procedures. We pipetted each of the reactants in the ratios specified by Table 4.1 to create a 200 μL mastermix. From that mastermix, we then used 5 μL aliquots combined with each of our different reaction conditions. In order to test different reaction conditions, we added an additional 1 μL of various reactants. For this dataset, we experimented by adding 16 differing ratios of sugar, phosphate, potassium, and nucleotides. The DNA circuit in the CFPS system simply produces Red Fluorescent Protein (RFP), so we were able to judge the amount of protein production by measuring fluorescence readout with a

Table 4.1: List of reactants for a 50 μL CFPS reaction. Note that MDX is a mixture of substrates (see the complete list in A.1. AAs includes all 20 of the amino acids.

Reactant	Amount (μL)	Purpose
MDX	5	Energy substrates
AAs	10	Building blocks for translation
Polyethylene Glycol (PEG)	2.5	Molecular crowding
Hexose Monophosphate (HMP)	1	Phosphate source
Magnesium (Mg)	0.74	Important cofactor
Potassium (K)	0.8	Charge homeostasis
DNA	0.8	Template for protein production
Cell Extract	25	Transcription (TX)/Translation (TL) ma
CFPS	50	Protein production

plate reader.

However, there are multiple downsides to doing this by hand, so our second dataset was created using a Labcyte Echo acoustic liquid handler. Creating large datasets by hand takes a long time and limits the amount of data one can generate. Additionally, pipetting by hand has an intrinsic error, so we hoped that automating the creation of the reactions would reduce human error. The Echo is an acoustic liquid handler that can combine different amounts of liquid to create each of the CFPS reactions. We were able to perform the experiments at $2\mu L$ of CFPS system and 500 Nanoliter (nL) of additional reactants. The use of automation allowed us to test 48 different reaction conditions and shows that this could be grown to an even larger scale.

Finally, we received a third dataset from the recent Karim and Jewett paper that uses CFPS systems to perform metabolic engineering [47]. Our protocol is very similar to the one used in the Jewett lab, though some of the reactant concentrations differ. This dataset is primarily useful because it was created in a different lab and they investigated a metabolic pathway whose output is not fluorescence. Since it was created in different lab conditions, we can check that our model is learning something about cell-free systems in general, not just overfitting our data. The pathway that they investigated is an inherently

metabolic pathway, so we can see if our metabolic model is better able to describe these types of gene circuits.

4.2 Data ingestion and incorporation

4.2.1 Data ingestion

Since this system is intended to aid biologists in their experiments, we wanted to make it end-to-end and as easy to use as possible. With that in mind, we created tools to automatically incorporate the experimental setup into the models. Figure 5.2 shows the two main ways that the experimental setup could show up. The first is in the form of end concentrations of the different reactants in the cell-free system. This is the easiest because in that case we can simply convert concentrations into fluxes. Fluxes have the unit mol/min/gDW, so we can convert metabolite concentrations into flues using the following equation from MetaboTools [48]:

$$f = \frac{m}{c * t * 1000.0} \tag{4.1}$$

where f is the flux value, m is the concentration of our metabolite, and c is the overall concentration of the cell. We set t to be 24, the time scale of our reactions. Using this equation our tool is able to ingest a CSV mapping the name of the reactant to the final concentration and turn them into flux constraints.

However, we also support users who might only know the relative amounts of each reactant that they are adding. This is important because many biologists may have a protocol that they follow that tells them how to make a CFPS system, but they may not know the concentrations in their mixture. To support these users, we allow them to upload a CSV (or combination of CSVs) that contains information about each reactant used to make the system. For each reactant, we require name of each reactant, the molecular weight, the volume used to create the stock, the weight used to create the

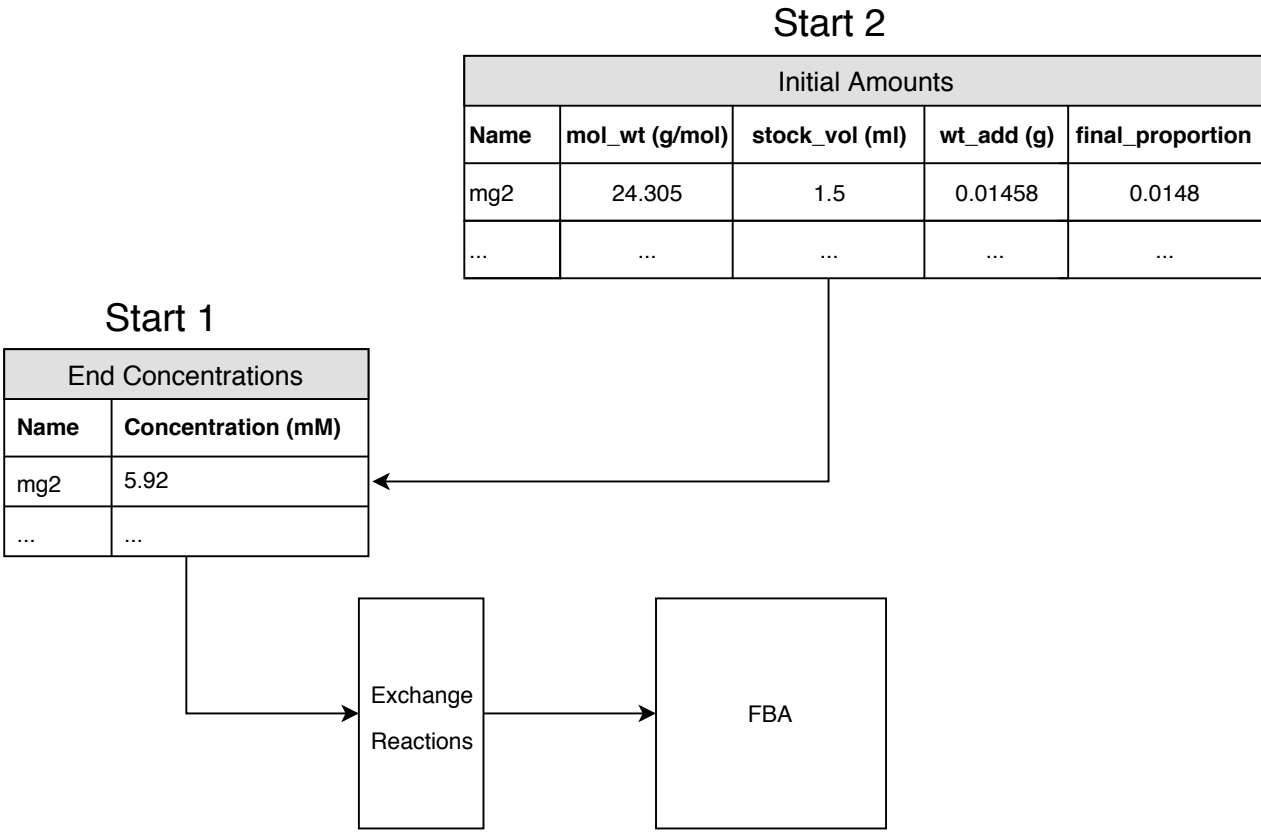

Figure 4.2: Two possible ways to incorporate the experimental setup in our pipeline. In Start 1, the user already knows the final concentrations of each reactant and can upload that as a CSV. Otherwise, the user can upload information about the reactants that are added and our tools will automatically calculate the final concnetrations of each reactant.

stock, and the final volume added to the cell-free reaction. We parse this information to calculate the final concentration of each reactant in the CFPS system. Once we have manipulated the data into the same form as the first type of upload, we can proceed with the rest of our computational pipeline.

4.2.2 Data Incorporation

After we ingest the experimental setup, we need to incorporate it into our FBA model. We use the COBRApy python package [49] to handle our FBA

models. The base model that we start with is the iJO1366 model for *E. coli*which consists of 1805 metabolites and 2583 reactions [26]. This model needs to be adapted to cell-free systems because they no longer have the same physical structure as *E. coli*cells. First, we need to make the cell-free reaction reflect the fact that there are no longer any membranes or cellular compartments. We do this by iterating over every reaction that contains a periplasmic or extracellular component. For each reaction, we then replace every metabolite with either the corresponding cytosolic metabolite or, if no corresponding metabolite exists, we just change the metabolite to be in the cytosol. At this point, we have all of the reactions occurring in the same location, which reflects how a CFPS system works.

Our next task is to handle how exchange reactions are dealt with in the model. Exchange reactions are pseudo-reactions that allow us to constraint the amount of a reactant that is allowed to be in the model. For each of the reactants in our CFPS system, we replace the bounds on the exchange reactions based on the fluxes we calculated from the setup data earlier. At the end of this process, we have converted the full-scale *E. coli*model to better approximate a CFPS system. While still overspecified, we have now created a new base cell-free model.

We also extend this model to explicitly incorporate the transcription and translation reactions that are so crucial to CFPS systems. Most FBA models do not explicitly consider the transcription and translation reactions. However, earlier work from the Varner lab incorporated these reactions into a FBA model specifically to try to model a cell-free system [38]. This earlier work created the FBA model by hand and is written in Julia. We built helper tools such that, given the sequence of the gene of interest, we can automatically generate a version of the Varner model. Next, we convert that model to Python and extract the relevant transcription and translation reactions. We incorporate those reactions into our model and then can set the production of the protein of interest as our FBA objective. This model is called our TXTL cell-free model.

4.3 Dataset generation

4.3.1 Model generation

Once we have created these base models, we can use them in conjunction with the different experimental conditions to create a different model for each condition. To do this, we take in a CSV consisting of the experimental conditions and a quantitative output. Each row represents a different experiment. The columns contain either the different concentrations for starting reactants or the different amounts added. In the latter case, we can re-calculate the final concentrations of each of the reactants for each experimental starting condition. In both cases, once we have the new final concentrations for each reactant, we create a new FBA model.

Were we simply to solve the different models we have created, many would have the same flux results. For these naive models, Section 5.1 shows that they do not describe our biological results very well. Clearly, FBA alone cannot describe the full richness of a real system. We also have the problem that the model is overspecified. These FBA models contain every reaction that is occurring in a steady state bacterium While we do harvest the bacteria at steady state, some enzymes will degrade and others will not be as important in cell-free systems. These types of changes cannot be encapsulated solely through a change of objective function and changing to a single-compartment reaction.

One can imagine FBA as a coarse bijective function, so points that vary only slightly in the input space will get projected into the same output point. We want to do better by first passing the fluxes through a dimensionality reduction technique such as a VAE. However, in order to do this, we first need a large dataset of fluxes to train on. Since we only have a small number (dozens, not hundreds) of models, we cannot simply use the fluxes from the optimal solution to the model. We supplement our dataset by flux sampling from each model.

4.3.2 Flux sampling

Flux sampling is the process of sampling possible fluxes through each reaction. This gives us a distribution of fluxes for each reaction, for each model. We use optGpSampler to sample 2000 fluxes from each of our models [50]. Figure **??** shows what happens as we increase the number of samples we draw from each model. Clearly, we can see that increasing the number of samples leads to a more normal distribution.

Each model has different experimental conditions and therefore has slightly different flux distributions for each reaction. So, after we generate flux samples for each model, we can combine the sampled fluxes to generate two new datasets. One dataset is flat–meaning we simply concatenate all of the sampled fluxes into a dataset of size $2000ExR$ where E is the number of experiments and R is the number of reactions. This dataset is useful to investigate the fluxes and see if we can perform dimensionality reduction to gain insights on the fluxes in our models in general. The other dataset we can make involves sampling with replacement from the fluxes of each model and then stacking the fluxes to create a new dataset. This dataset has size $NxExR$ where N is the number of samples with replacement we drew. Using this dataset allows us to incorporate the correlation term in our VAE.

4.4 VAE

The key part of our system was the usage of a Variational Autoencoder in order to reduce the dimensionality of our model. Our VAE was implemented using the Keras framework [51] and the implementation was designed to be as modular as possible. We wanted to make the VAE as modular as possible so that end-users would be able to easily change the parameters of the VAE without having to have any prior training in deep learning. Thus, our implementation is able to take in either a flat or stacked dataset, a list of layer sizes, the number of latent dimensions, how to scale the inputs, and whether or not to use the custom correlation loss function.

The structure of the VAEs we used are as follows. We experimented with two different layer structures, a 2-layer VAE with layer sizes 1024 and 256 and a 3-layer VAE with layer sizes 1024, 512, and 256. We also tried using latent dimensions of size 2 and 10. For activations between layers of the encoder and the decoder, we used Rectified Linear Unit (ReLU) activations [52]. The activation of our final layer of the decoder is one of sigmoidal, tanh, or linear, depending on how the dataset was scaled. When using the custom correlation loss function, we required a stacked dataset, otherwise a flat dataset could be used.

4.4.1 Correlated VAE

The key insight for the correlation loss is that we did not just want to reduce our dimensionality in an unbiased way–we care about how well it describes real-world data. Thus, we can use our additional information to perturb the latent space to better reflect biological reality. We know what the relative rankings of the different models should be, so we can add a correlation loss term between the experimental data and the latent representation. Recall Equation 2.3 that described the VAE loss function. We now train with a modified loss function:

$$\mathcal{L}(\theta, \phi) = -E[\log p_\theta(x_i|z)] + KL(q_\phi(z|x_i)||p(z)) + corr(x_i', d) \tag{4.2}$$

where x_i' is the reconstructed data and d is our scaled biological data.

4.5 FBA model reduction

Now that we have run our models through our correlational VAE we have a lower dimensional representation of the model. We can use this lower dimensional representation to create a better cell-free model. One way we could use our system to do this is by running the whole system on each set of

new data. We first generate the FBA model using the tool chain described above. Then, using the trained autoencoder, we can transform the optimal fluxes and get the reconstructed fluxes that better describe a cell-free system. This could be very useful for batch variation or comparisons between different labs.

However, one of our goals is to build a new cell-free model. So we can use the dimensionality reduction from the VAE to create a more accurate reduced model. We do this by thresholding the latent space and removing reactions that have fluxes close to 0. This simple thresholding allows us to identify reactions that are not important in cell-free systems because they do not vary between different runs of our cell-free reactions. Section 5.1 shows that these reduced models with the differential conditions give us better explanations of experimental data than full GEMs.

Chapter 5

Results

Our system is able to produce biologically relevant dimensionality reductions that allow us to make better reduce models for CFPS systems. We show the following key points:

- The reduced models produced by our system are more representative of biological data than naive models.
- The latent space of the VAE and show that it learns a separation between the different models.
- As noise is added to our model, the VAE performance falls off.
- We can train a model on one dataset and predict on another dataset.

5.1 Comparison of models

A key goal for our system was to produce reduced models that could accurately explain our biological data. We tested this by solving for the optimal flux distribution for each model of our reaction conditions. We then found the correlation between the predicted output and the real biological output. Table 5.1 shows the results for each of our starting models that are at the GEM scale as well as some reduced models. As expected, we found that

Table 5.1: Comparison of models. Models that have been produced by our pipeline end in 'reduced'. We compare the full GEMs to our reduced models and show that the using our system improves the correlation with real data

Dataset	TXTL	Correlation	
echo	Yes	No	0.22
echo	No	No	0.14
hand	Yes	No	0.23
hand	No	No	0.24
karim	No	No	0.17
coli-pruned	No	No	0.34
hand-reduced	Yes	Yes	0.50
karim-reduced	Yes	Yes	0.43

the full-scale GEMs have very low correlations, between 0.14 and 0.24. This models are intended to describe living *E. coli*cells and thus do a poor job of modeling CFPS systems.

We also compare our model to pre-existing pruned models that other algorithms have created. For instance, the ColiPruned model from NetworkReducer has a correlation of 0.34 with our biological data. This is better than the full-scale genome models, though it is not as good as our customized cell-free models. This make sense because the goal of NetworkReducer is to reduce to a minimal core model, not to specify it for cell-free systems. Our system produces models that with more than 2x better correlations than a full-scale GEM. While these correlations show that we still cannot perfectly describe the biological system, the reduction is far better than a full-scale model.

5.2 Unsupervised Learning

In addition to helping us reduce our models, we can use our trained VAE to perform unsupervised learning. By examining the latent space, we can uncover patterns that we otherwise might not be able to discover by eye. Figure 5.2 shows one way we can visualize the latent space of the VAE. We

trained our 3-layer Corr-VAE with a latent dimension of 10 on the training split of our stacked hand dataset. We then projected the test split into then 10-dimensional latent space and passed that through the tSNE [53] to project it to 2 dimensions. Each of the points in Figure 5.2 are colored according to the different starting experimental conditions.

We can see that the VAE is able to cluster the models reasonably well based on the experimental conditions. This is a good sign because it means the VAE is able to distinguish between the fluxes of each model when we vary the experimental conditions. Additionally, similar experimental conditions appear to be closer to each other. For example, we can see the yellow dots at the bottom are near to two clusters of purple dots. All three of the experimental conditions have no additional sugar and raised phosphate levels. This implies that the latent space of the VAE can not only distinguish between experimental conditions but the distance in the latent space has biological meaning.

5.3 Validation

We also validated that our VAE was performing as intended. We can examine a few aspects of our reconstructed fluxes to ensure that the results we are seeing are not simply due to chance. Figure **??** plots the standard deviations of each of the reactions for our test dataset as well as our reconstructed dataset. We can see that the reconstructed dataset maintains a very similar shape to our original dataset, while also reducing the standard deviation of fluxes with each reaction. Although we did not investigate this further, this implies that the VAE could also be used to de-noise our dataset.

Another key point we investigated was whether or not the correlations we produce are spurious. We wanted to ensure that our VAE is not creating correlations out of thin air just to reduce its loss function. Figure 5.3 shows what happens when we add noise to our experimental data. First, we generate a dataset with fluxes that are highly correlated to our experimental

data. This is represented by the green line. We can then run those fluxes through our VAE and check the correlation. The blue line shows that this improves the correlation of our fluxes because of our loss function. Next, we can add noise to the data by randomly drawing from a normal distribution and adding it to the flux for each reaction. The dots show that as we add increasing amounts of noise to the data, the correlation goes to about 0. This means that the VAE is not simply artificially creating a correlation but is learning something about the underlying data.

5.4 Transfer learning

TODO

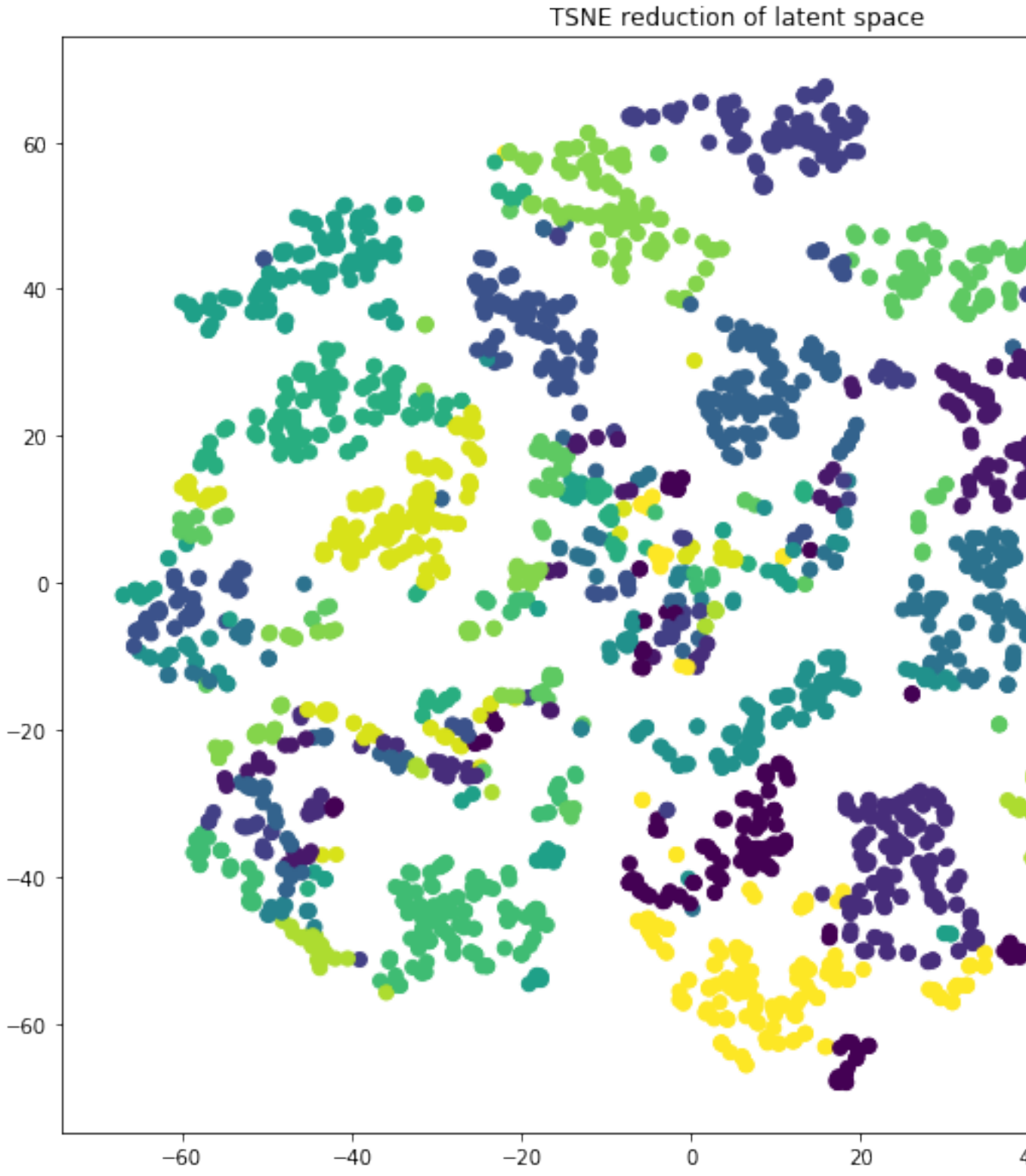

Figure 5.1: We show that we can reconstruct each individual model from the latent space.

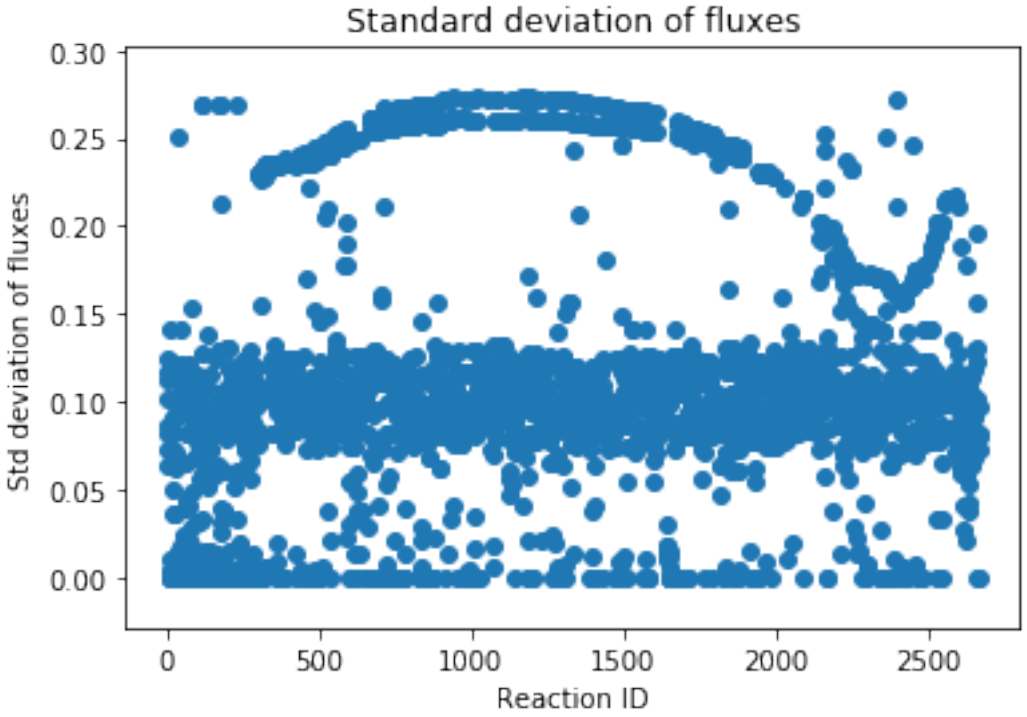

Figure 5.2: Reconstructed fluxes reduce the amount of noise while also maintaining the shape

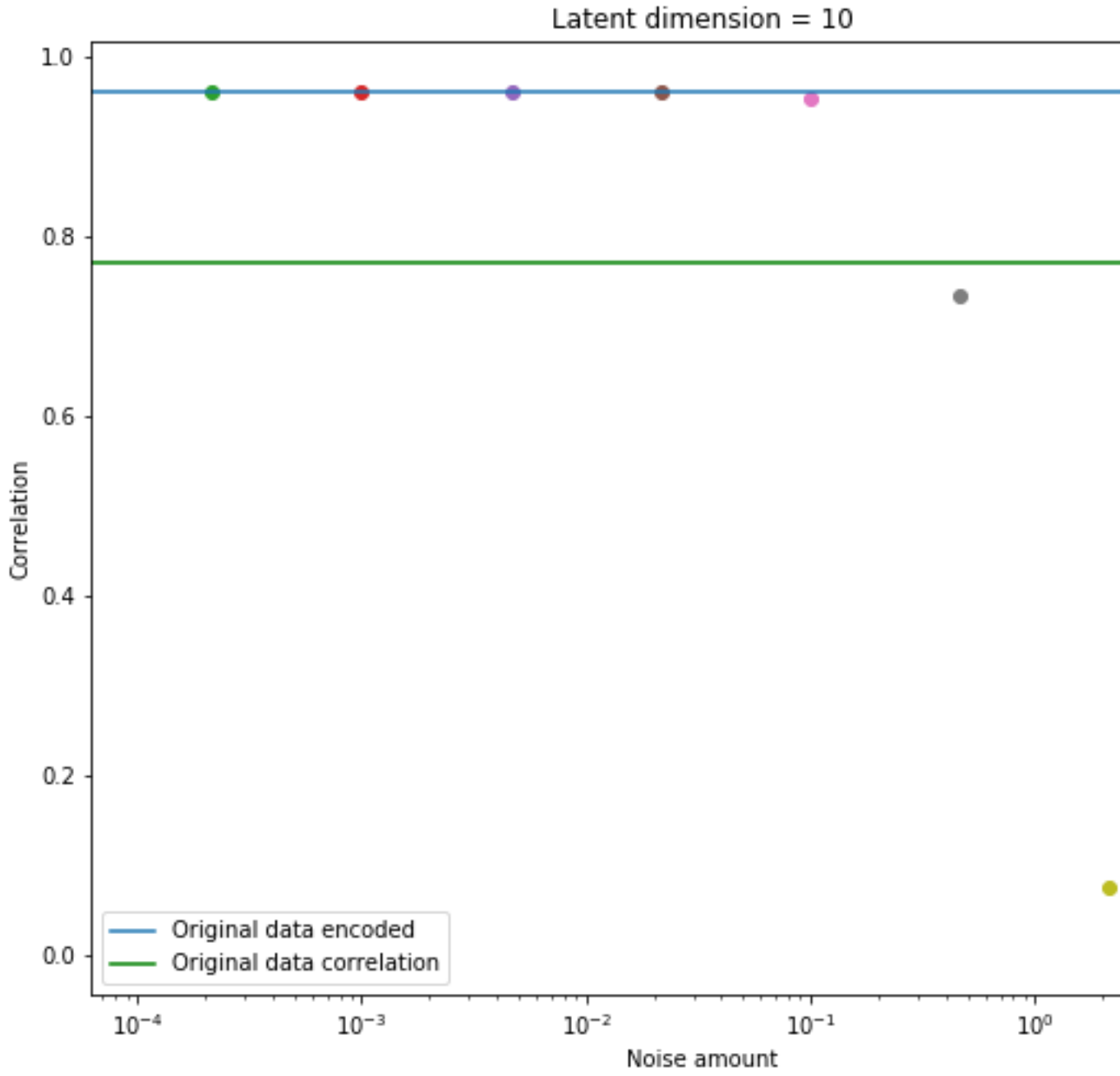

Figure 5.3: Adding noise to the data reduces the correlation from our VAE

Chapter 6

Discussion and Future Work

6.1 Improved data gathering

As we have shown above, increasing the amount of data is useful for training a more generalizable model. Generating large biological datasets is difficult, but there is a movement towards greater automation of lab tasks. In particular, we showed that a lab robot such as the Labcyte Echo can be used to generate larger scale datasets. We encourage future work to take this even further and generate datasets with hundreds or even thousands of starting conditions.

6.2 Different cell types

We have only explored the usage of this technique on a *E. coli*cell-free model. However, there is a lot of work currently ongoing to use non-*E. coli*cell-free models. This system was written with generalizability in mind and should transfer to other organisms as well. In particular, this was the intent with creating a reduction system instead of hand-building a model from scratch. There are dozens of FBA models of other organisms that this could be applied to. Figure 6.1 demonstrates our vision as a general purpose system for

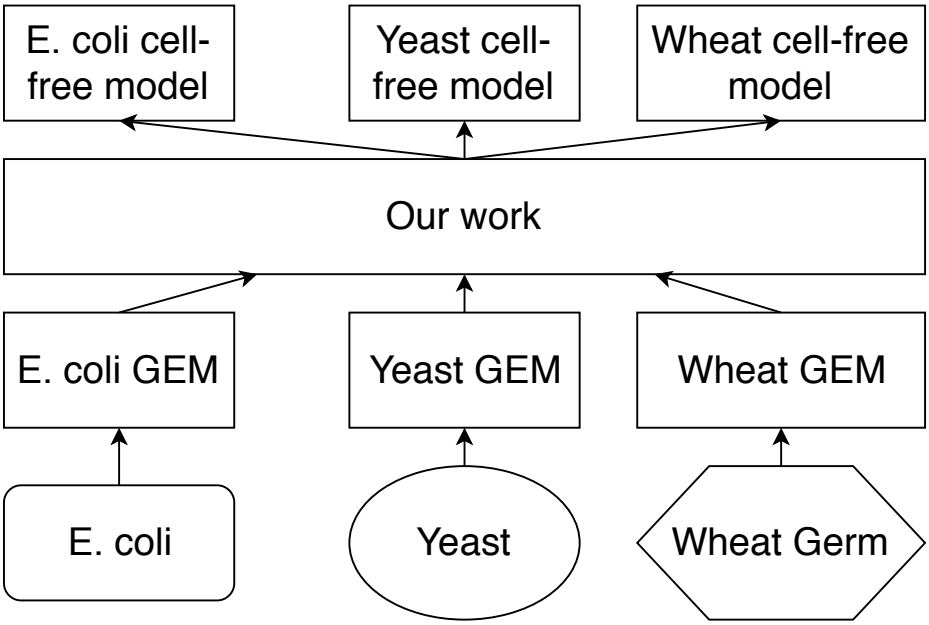

Figure 6.1: Potential future usage for our system: converting different organism's GEMs into appropriate cell-free models.

creating cell-free models.

In particular, the primary author of this work is a part of an Open Plant grant which involves the exploration of using wheat-germ cell-free systems. The data is currently being generated through a collaboration with the Earlham Institute. After the data generation is done, we want to apply this to that data to see its effectiveness. Additionally, this system will get a web interface to encourage other biologists to use it for their own data.

6.3 RL system for reduction

One thing that we noticed while thinking about the best way to reduce the model is that we could frame the problem in terms of a reinforcement learning problem. We originally tried to determine which reactions to remove by hand. However, the way we went about this was to remove a reaction and re-run the model to see the output. Then, we compared how well we match

the experimental data. So, we built a reinforcement learning framework to do this. Rewards are based on how well our collection of models reflects the experimental data.

Simply doing a naive search with this RL framework would take far too long. Starting with 2600 reactions, performing a random search over this space could take up to TODO. Thus, we could use the VAE that we wrote in conjunction with this framework to perform a better directed search. This could allow us to reduce the model even further and get even more specificity in the reduction.

6.4 Improved starting model

One potential issue that is limiting the usefulness of our pipeline is how well described the initial models are. They're only as good as they are allowed to be by the reactions that exist. FBA models undergo improvements periodically and we're using the most prominent model iJO1366. However, as this model is updated, our model can adapt and use more up-to-date reactions.

6.5 Batch variation

Batch variation is an important problem facing the cell-free community. Some studies have shown up to 20% variation between batches for the same protocols and DNA constructs. This is an incredibly important issue because the typical workflow for a researcher may involve the characterization of a circuit through a series of different conditions or slightly different DNA programs. However, most people need to compare the performance of these variations. In order to deal with variation, researchers will perform all of their experiments with a single batch. When that batch runs out, though, they will be required to re-characterize and redo all of the experiments they

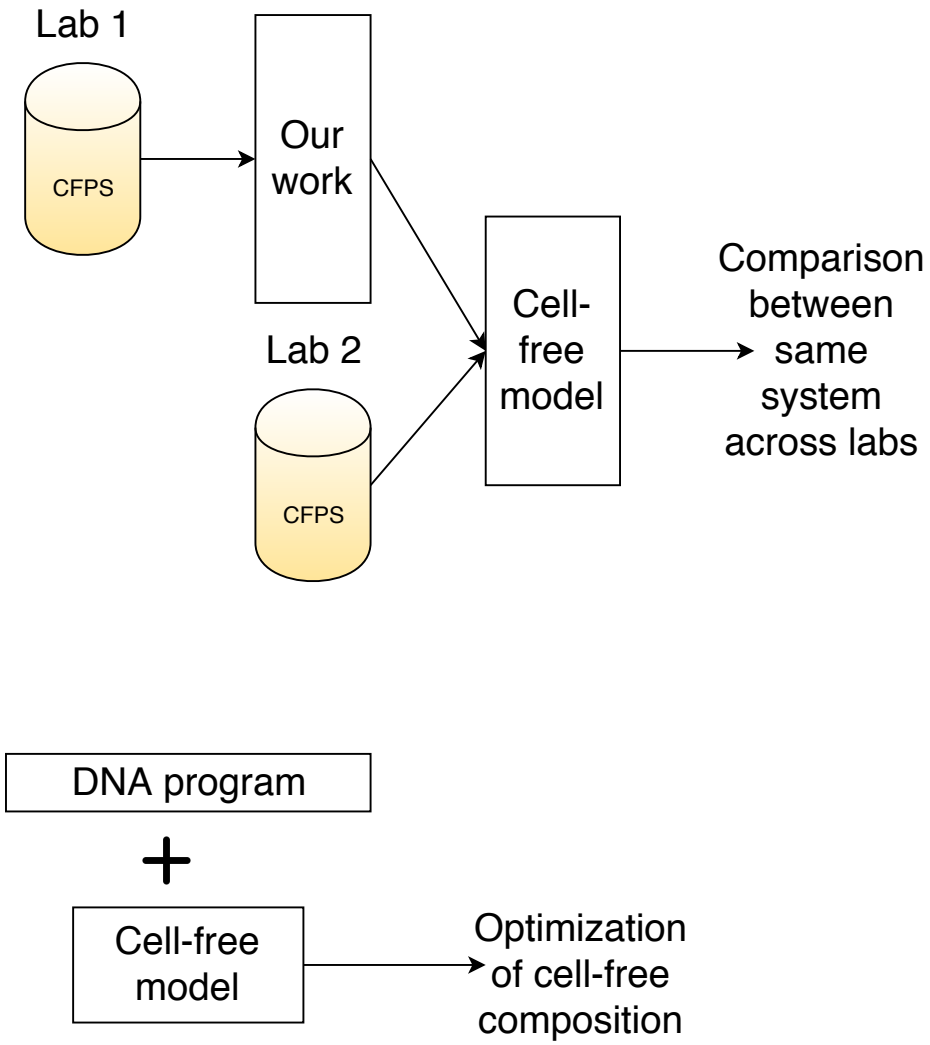

Figure 6.2: Two future applications of our system. Above, multiple experiments can be projected in the same subspace to better compare experimental results. Below, the system can help optimize the energetic composition of a cell-free system

have already done.

Figure 6.4 shows how our system provides a potential solution to this issue of batch variation. A researcher runs a calibration set of experiments and then use those experiments to generate a VAE model based on that data. Then, any future experiments that are performed (either in the same batch or a different one) could be projected into the same latent space in order to allow

comparison between the experiments. This would allow the translation of all of the data generated into the same subspace and allow better comparisons across batches.

Appendix A

Experimental Setup

Table A.1: Final concentrations of reactants in our CFPS reaction

Compound	Concentration (Micromolar (mM))
HMP	0.981
Mg	5.92
K	48.0
NAD	0.454
Coenzyme-A (CoA)	0.353
cAMP	1.01
Folinic acid	0.092
Spermidine	0.181
Glucose	5.42
ATP/GTP	2.09
CTP/UTP	1.26
AAs	35.6
tRNA	0.010
DNA	0.071

Bibliography

[1] Jay D Keasling. Synthetic biology and the development of tools for metabolic engineering. *Metabolic engineering*, 14(3):189–195, 2012.

[2] Jan Schellenberger, Richard Que, Ronan MT Fleming, Ines Thiele, Jeffrey D Orth, Adam M Feist, Daniel C Zielinski, Aarash Bordbar, Nathan E Lewis, Sorena Rahmanian, et al. Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox v2. 0. *Nature protocols*, 6(9):1290, 2011.

[3] Jeffrey D Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245, 2010.

[4] Adam M Feist and Bernhard Ø Palsson. The growing scope of applications of genome-scale metabolic reconstructions using escherichia coli. *Nature biotechnology*, 26(6):659, 2008.

[5] E_ Almaas, B Kovacs, T Vicsek, ZN Oltvai, and A-L Barabási. Global organization of metabolic fluxes in the bacterium escherichia coli. *Nature*, 427(6977):839, 2004.

[6] Tomer Shlomi, Yariv Eisenberg, Roded Sharan, and Eytan Ruppin. A genome-scale computational study of the interplay between transcriptional regulation and metabolism. *Molecular systems biology*, 3(1):101, 2007.

[7] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561, 1970.

[8] C Eric Hodgman and Michael C Jewett. Cell-free synthetic biology: thinking outside the cell. *Metabolic engineering*, 14(3):261–269, 2012.

[9] Joseph A Rollin, Tsz Kin Tam, and Y-H Percival Zhang. New biotechnology paradigm: cell-free biosystems for biomanufacturing. *Green chemistry*, 15(7):1708–1719, 2013.

[10] Erik D Carlson, Rui Gan, C Eric Hodgman, and Michael C Jewett. Cell-free protein synthesis: applications come of age. *Biotechnology advances*, 30(5):1185–1194, 2012.

[11] Marshall W Nirenberg and J Heinrich Matthaei. The dependence of cell-free protein synthesis in e. coli upon naturally occurring or synthetic polyribonucleotides. *Proceedings of the National Academy of Sciences*, 47(10):1588–1602, 1961.

[12] Kara A Calhoun and James R Swartz. Total amino acid stabilization during cell-free protein synthesis reactions. *Journal of biotechnology*, 123(2):193–203, 2006.

[13] Michael C Jewett, Kara A Calhoun, Alexei Voloshin, Jessica J Wuu, and James R Swartz. An integrated cell-free metabolic platform for protein production and synthetic biology. *Molecular systems biology*, 4(1):220, 2008.

[14] Yoshihiro Shimizu, Akio Inoue, Yukihide Tomari, Tsutomu Suzuki, Takashi Yokogawa, Kazuya Nishikawa, and Takuya Ueda. Cell-free translation reconstituted with purified components. *Nature biotechnology*, 19(8):751, 2001.

[15] Adam M Feist and Bernhard O Palsson. The biomass objective function. *Current opinion in microbiology*, 13(3):344–349, 2010.

[16] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[17] Eleftherios Terry Papoutsakis. Equations and calculations for fermentations of butyric acid bacteria. *Biotechnology and bioengineering*, 26(2):174–187, 1984.

[18] David A Fell and J Rankin Small. Fat synthesis in adipose tissue. an examination of stoichiometric constraints. *Biochemical Journal*, 238(3):781–786, 1986.

[19] RA Majewski and MM Domach. Simple constrained-optimization view of acetate overflow in e. coli. *Biotechnology and bioengineering*, 35(7):732–738, 1990.

[20] Amit Varma and Bernhard O Palsson. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type escherichia coli w3110. *Applied and environmental microbiology*, 60(10):3724–3731, 1994.

[21] Jeremy S Edwards, Rafael U Ibarra, and Bernhard O Palsson. In silico predictions of escherichia coli metabolic capabilities are consistent with experimental data. *Nature biotechnology*, 19(2):125, 2001.

[22] Daniel Segre, Dennis Vitkup, and George M Church. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences*, 99(23):15112–15117, 2002.

[23] Aarash Bordbar, Jonathan M Monk, Zachary A King, and Bernhard O Palsson. Constraint-based models predict metabolic and associated cellular functions. *Nature Reviews Genetics*, 15(2):107, 2014.

[24] Amit Varma and Bernhard O Palsson. Metabolic capabilities of escherichia coli: I. synthesis of biosynthetic precursors and cofactors. *Journal of theoretical biology*, 165(4):477–502, 1993.

[25] JS Edwards and BO Palsson. The escherichia coli mg1655 in silico metabolic genotype: its definition, characteristics, and capabilities. *Proceedings of the National Academy of Sciences*, 97(10):5528–5533, 2000.

[26] Jeffrey D Orth, Tom M Conrad, Jessica Na, Joshua A Lerman, Hojung Nam, Adam M Feist, and Bernhard Ø Palsson. A comprehensive genome-scale reconstruction of escherichia coli metabolism2011. *Molecular systems biology*, 7(1):535, 2011.

[27] Colton J Lloyd, Ali Ebrahim, Laurence Yang, Zachary Andrew King, Edward Catoiu, Edward J O'Brien, Joanne K Liu, and Bernhard O Palsson. Cobrame: A computational framework for models of metabolism and gene expression. *bioRxiv*, page 106559, 2017.

[28] Meric Ataman, Daniel F Hernandez Gardiol, Georgios Fengos, and Vassily Hatzimanikatis. redgem: Systematic reduction and analysis of genome-scale metabolic reconstructions for development of consistent core metabolic models. *PLoS computational biology*, 13(7):e1005444, 2017.

[29] Meric Ataman and Vassily Hatzimanikatis. lumpgem: Systematic generation of subnetworks and elementally balanced lumped reactions for the biosynthesis of target metabolites. *PLoS computational biology*, 13(7):e1005513, 2017.

[30] Philipp Erdrich, Ralf Steuer, and Steffen Klamt. An algorithm for the reduction of genome-scale metabolic network models to meaningful core models. *BMC systems biology*, 9(1):48, 2015.

[31] Annika Röhl and Alexander Bockmayr. A mixed-integer linear programming approach to the reduction of genome-scale metabolic networks. *BMC bioinformatics*, 18(1):2, 2017.

[32] Moritz von Stosch, Cristiana Rodrigues de Azevedo, Mauro Luis, Sebastiao Feyo de Azevedo, and Rui Oliveira. A principal components method constrained by elementary flux modes: analysis of flux data sets. *BMC bioinformatics*, 17(1):200, 2016.

[33] Sahely Bhadra, Peter Blomberg, Sandra Castillo, and Juho Rousu. Principal metabolic flux mode analysis. *bioRxiv*, page 163055, 2017.

[34] Zoltan A Tuza, Vipul Singhal, Jongmin Kim, and Richard M Murray. An in silico modeling toolbox for rapid prototyping of circuits in a biomolecular breadboard system. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1404–1410. IEEE, 2013.

[35] Joseph A Wayman, Adithya Sagar, and Jeffrey D Varner. Dynamic modeling of cell-free biochemical networks using effective kinetic models. *Processes*, 3(1):138–160, 2015.

[36] Simon J Moore, James T MacDonald, Sarah Wienecke, Alka Ishwarbhai, Argyro Tsipa, Rochelle Aw, Nicolas Kylilis, David J Bell, David W McClymont, Kirsten Jensen, et al. Rapid acquisition and model-based analysis of cell-free transcription–translation reactions from nonmodel bacteria. *Proceedings of the National Academy of Sciences*, 115(19):E4340–E4349, 2018.

[37] Matthias Bujara and Sven Panke. In silico assessment of cell-free systems. *Biotechnology and bioengineering*, 109(10):2620–2629, 2012.

[38] Michael Vilkhovoy, Nicholas Horvath, Che-hsiao Shih, Joseph Wayman, Kara Calhoun, James Swartz, and Jeffrey Varner. Sequence specific modeling of e. coli cell-free protein synthesis. *bioRxiv*, page 139774, 2017.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[40] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[41] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[42] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*.

[43] Gregory P Way and Casey S Greene. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *bioRxiv*, page 174474, 2017.

[44] Sarath Chandar, Mitesh M Khapra, Hugo Larochelle, and Balaraman Ravindran. Correlational neural networks. *Neural computation*, 28(2):257–285, 2016.

[45] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J Kennedy. Relational autoencoder for feature extraction. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 364–371. IEEE, 2017.

[46] Anibal A Medina, Contreras Juan P, and Fernan Federici. Preparation of cell-free rnapt7 reactions. `dx.doi.org/10.17504/protocols.io.kz2cx8e`, 2017.

[47] Ashty S Karim, Jacob T Heggestad, Samantha A Crowe, and Michael C Jewett. Controlling cell-free metabolism through physiochemical perturbations. *Metabolic engineering*, 45:86–94, 2018.

[48] Maike K Aurich, Ronan MT Fleming, and Ines Thiele. Metabotools: A comprehensive toolbox for analysis of genome-scale metabolic models. *Frontiers in physiology*, 7:327, 2016.

[49] Ali Ebrahim, Joshua A Lerman, Bernhard O Palsson, and Daniel R Hyduke. Cobrapy: constraints-based reconstruction and analysis for python. *BMC systems biology*, 7(1):74, 2013.

[50] Wout Megchelenbrink, Martijn Huynen, and Elena Marchiori. optgpsampler: an improved tool for uniformly sampling the solution-space of genome-scale metabolic networks. *PloS one*, 9(2):e86587, 2014.

[51] François Chollet et al. Keras. `https://keras.io`, 2015.

[52] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[53] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.