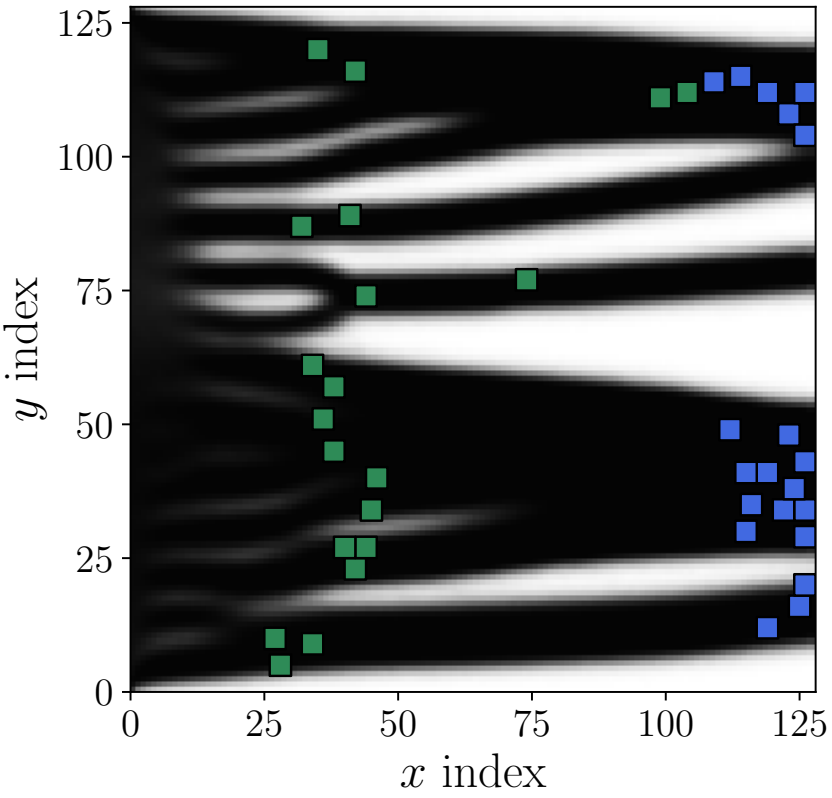


Optimizing the Design of Nanofluidic Chips with Graph Reinforcement Learning

A General Graph Attentional Framework Applied on Nanoscale Catalytic Reactor Systems

Master's thesis in Complex Adaptive Systems

MATTIAS ULMESTRAND

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Optimizing the Design of Nanofluidic Chips with Graph Reinforcement Learning

A General Graph Attentional Framework Applied on Nanoscale Catalytic Reactor Systems

MATTIAS ULMESTRAND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chemical Physics
Langhammer Lab
Chalmers University of Technology
Gothenburg, Sweden 2022

Optimizing the Design of Nanofluidic Chips with Graph Reinforcement Learning
A General Graph Attentional Framework Applied on Nanoscale Catalytic Reactor Systems
MATTIAS ULMESTRAND

Supervisors: Henrik Klein Moberg, Physics
Henrik Ström, Mechanics and Maritime Sciences
David Albinsson, Physics
Examiner: Christoph Langhammer, Physics

Master's Thesis 2022
Department of Physics
Chemical Physics
Langhammer Lab
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Experimental nanofluidic chip design with two types of reactivity sites, designed with deep geometric reinforcement learning and ant colony optimization.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Optimizing the Design of Nanofluidic Chips with Graph Reinforcement Learning
A General Graph Attentional Framework Applied on Nanoscale Catalytic Reactor Systems
MATTIAS ULMESTRAND
Department of Physics
Chalmers University of Technology

Abstract

Nanofluidic chips are devices where fluids are controlled at nanoscale. Langhammer Lab at Chalmers University of Technology researches nanofluidic chips for catalysis reactions. On these nanoscale reactors, catalysts are placed to maximize some property such as the reactant conversion rate to product. However, to this point, no framework exists for optimizing the design of the group's nanofludic chips. Chips are currently designed through laborious trial and error. In this master's thesis, a framework based on reinforcement learning with graph attentional neural networks is presented and applied for optimizing the design of nanofludic chips. A reinforcement learning agent is trained on a reward system based on computational fluid dynamics (CFD) and consistently outperforms simulated manual designs. Additionally, a considerably lighter reward system based on ant colony optimization (ACO) is developed for placing catalysts and forming channels. The ACO reward system is shown to be highly correlated with the CFD reward system, but requires some further developement in order to achieve the same performance as the CFD reward system.

Keywords: Reinforcement learning, graph convolutional networks, attention, computational fluid dynamics, nanofluidics, stochastic optimization algorithms.

Acknowledgements

I would like to thank my main supervisor Henrik Klein Moberg for assisting me during the entire project, discussing ideas and providing feedback for the project progress. It impacts the work noticeably when you have a supervisor who is dedicated and excited about the project, but also a good friend.

I also thank Henrik Ström for assisting me with all things related to fluid dynamics, and for joining me in likely the longest running mail conversation known to man. The thesis would have been near infeasible as a six month project without the Ansys case files and reaction handling scripts provided by Ström.

Thanks to David Albinsson for providing feedback on the model progress and its usability for designing chips.

Lastly, I thank Christoph Langhammer, the department of Chemical Physics and Langhammer Lab for taking me in as a master's thesis worker, giving me insight to academic work and accepting me as a part of the crew. Mattias Ulmestrand,

Gothenburg, July 2022

List of Acronyms

Below is a list of acronyms that have been used throughout this thesis listed in alphabetical order:

ACO	Ant Colony Optimization
CFD	Computational Fluid Dynamics
DQN	Deep Q-Network
GATConv	Graph Attention Convolutional layer
GCN	Graph Convolutional Network
GDL	Geometric Deep Learning
GraphConv	Graph Convolutional layer
GraphNorm	Graph Normalization layer
LOC	Lab-on-a-chip
PG	Policy Gradient
PPO	Proximal Policy Optimization
TRPO	Trust Region Policy Optimization

Contents

List of Figures

List of Tables

1
Introduction

A lab-on-a-chip (LOC) is a device capable of performing experiments and analysis on miniature scale. Over the last years, such chips have quickly gathered attention from researchers in for example biology, chemistry and medical sciences, [1], [2], [3]. One instance of a LOC is the nanofluidic chip, where fluids are controlled on nanoscale. Langhammer Lab at Chalmers University of Technology conducts research on nanofluidic chips. The chips are used for investigating catalysis of particles. Specifically, it is of high interest to investigate single particle catalysis, since this avoids averaging effects otherwise seen when investigating an ensemble of particles, see Levin *et al.* [4]. Single particle catalysis allows researchers to better understand the properties of specific particles.

The nanofluidic chips used for catalysis research require specialized design in order to meet research goals. The chip components along with their effect on reactant and product flow form a complex system which is difficult to manually optimize. For example, maximizing the reactant conversion rate to product depends on several aspects such as pressure, reactant concentration and flow speed. The group's nanofluidic chips are currently designed through laborious trial and error. A framework capable of automatizing the design process would therefore be a powerful tool in aiding research. Additionally, since the prevalence of LOCs is growing quickly in many fields, the framework should desirably be easily applicable to other chip systems. This thesis aims to optimize the design of nanofluidic chips with easy transferability to similar systems, using machine learning (ML) techniques.

A promising indicator for use of ML in nanofluidic chip design is Mirhoseini *et al.*, where deep reinforcement learning and graph neural networks were used to optimize the design of Google's TPU chips [5]. The approach was able to match or outperform previous techniques, in a much smaller time scale. The goal of this thesis is to employ a similar policy gradient (PG) technique to optimize the reactant conversion rate of nanofluidic chips. A major benefit of using reinforcement learning over supervised learning is that no previously gathered database of training data is required. Instead, training data is sampled in a simulation environment during training. Another similar work is the model PrefixRL by Roy *et al.*, where prefix circuits were optimized with deep Q-networks (DQN) [6]. Optimization of LOCs with ML, however, appears to be an unexplored field prior to this thesis.

2
Theory

In the following sections, the theory behind policy gradient (PG) algorithms and ant colony optimization (ACO) is briefly explained, and nanofluidic chips are introduced.

2.1 Nanofluidic chips

A nanofluidic chip is a surface where a fluid is pumped through nanoscale channels and undergoes a reaction. The fluid is driven by applying a pressure gradient across the chip outlet and inlet. One example of a phenomenon that can be studied on a nanofluidic chip is catalysis. A nanofluidic catalytic reactor chip consists of channels to direct the current, and sites of reactivity which act as catalysts. The sites and channels investigated at Langhammer lab are typically approximately of order 100 nm in cross section, while the channel lengths can be of order 10-100 μm. At the 100 nm scale, the fluid flow may be analyzed with continuum dynamics [7]. The discrete nature of the reactant fluid is evident in nanofluidic chips, however, using continuum dynamics has still been found to accurately describe the fluid dynamics [8]. In order to simulate the flow through nanofluidic chips, computational fluid dynamics (CFD) can then be used. Ansys® Fluent is an example of commercial hardware where CFD is used and can be extended with chemical reaction simulations.

2.1.1 Catalysis

Catalysis is the process of increasing the rate of a reaction through addition of a catalyst. A perfect catalyst only accelerate reactions without being altered or depleted itself. In practice, however, there are several complications that may arise with catalysts.

One example of a reaction which can be accelerated with catalysts is the synthesis of ammonia from hydrogen and nitrogen, the so called Haber-Bosch process [9]. Ammonia is a widely produced chemical used for fertilizers, various nitrogen-containing compounds, and can also be used as a renewable energy transportation media [10], [11]. The Haber-Bosch process made it feasible to produce ammonia on an industrial scale [12]. The reaction is as follows,

$$N_2 + 3H_2 \rightarrow 2NH_3. \tag{2.1}$$

Iron-based catalysts are often used in ammonia synthesis, but have been challenged by the higher activity of ruthenium-based catalysts [9]. A problem with ruthenium-based catalysts, however, is that they can suffer catalyst poisoning, meaning reduced

activity in the presence of a chemical. Ruthenium-based catalysts can specifically be affected by hydrogen poisoning as well as ammonia poisoning [13]. Hydrogen poisoning is difficult to combat by nanofluidic chip planning, but ammonia poisoning could be reduced by placing active sites such that the product flow is redirected from other downstream active ruthenium sites. Hybrid chips could also be developed, where the high activity of ruthenium catalysts are complemented by the insensitivity to poisoning of iron catalysts.

2.1.2 Reaction kinetics

A first-order reaction depends linearly on the concentration of only one reactant. The reaction rate for a reactant A is then given by

$$-\frac{\mathrm{d}[A]}{\mathrm{d}t} = k[A], \tag{2.2}$$

where $[A]$ is the concentration of the reactant and k is the reaction rate coefficient. This is the differential equation of an exponential function for reactant concentration. The reactant concentration is then given by

$$[A] = [A]_0 e^{-kt}, \tag{2.3}$$

where $[A]_0$ is the initial reactant concentration. If a reactant undergoes a first-order reaction, the concentration will decrease exponentially. The maximum conversion to product thus happens in the beginning of the reaction. At this stage, some intuition can be found about the problem of placing catalysts optimally. It would likely not be beneficial to place a catalyst right after another one, since reactant concentration is likely to be low after a catalyst site. A better solution may be to place another catalyst further downstream, where reactant concentration may be higher due to fluid mixing.

2.2 Policy Gradient

A policy is a principle of action. Given the state of some system, the policy describes what action to take next in order to maximize some objective. In the context of nanofluidic chips, the chip canvas along with placed chip components can be seen as a state. The next component to be placed can also be added to make out the state, see Figure 2.1. This is beneficial if not all chip components are identical. The state can then be interpreted by some function, which provides a policy. The policy in turn provides which position to place the next component. Taking an action according to the policy creates a new state. This chain of states is repeated until reaching a termination condition, for example a maximum number of placed components.

The policy is typically given by an artificial neural network. With a neural network, the policy takes the form of a mapping of Q-values assigned to each action. In order to optimize the mapping, PG utilizes the policy gradient theorem [14], [15]. One formulation of the theorem is as follows:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]. \tag{2.4}$$

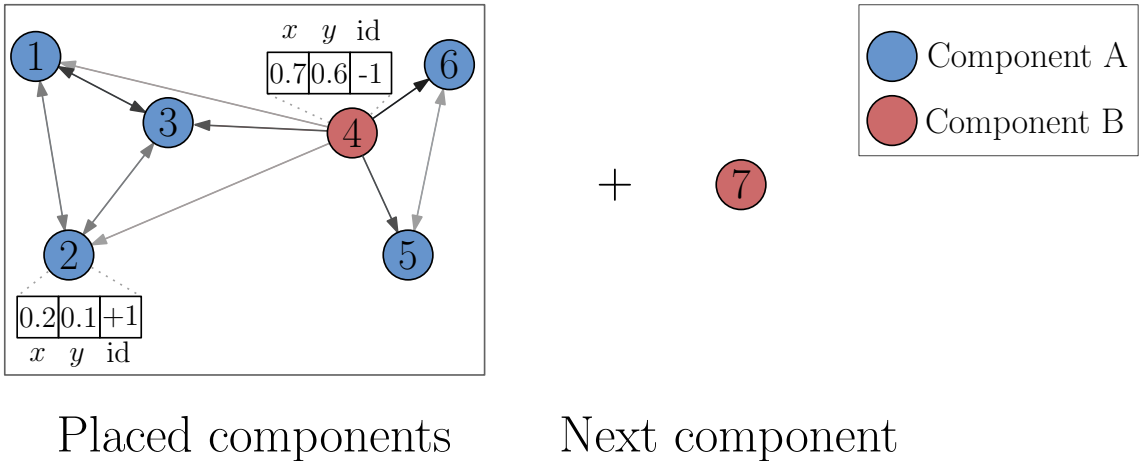


Figure 2.1: The current state of a chip can be made up of placed components along with the next component to be placed. The current node is needed if more than one node species exist, since different components may interact differently with one another. In this case, components A interact mutually, whereas components B interact with component A, but are not influenced by A. More on this in section 2.7.

The left-hand side is the performance gradient $\nabla_\theta J(\pi_\theta)$ of the policy π_θ over neural network parameters θ. According to the theorem, this gradient is equal to the expectation value $\mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}$ of the gradient of the logarithmized policy multiplied by the Q-values $Q^\pi(s,a)$. The Q-values are typically estimated by a given reward function which rewards the agent based on its performance. The subscripts in the expectation value represents that the state s is sampled from the state distribution ρ^π, and the action is sampled from the policy π_θ.

Contrary to many other machine learning standards, PG utilizes gradient ascent instead of gradient descent. The objective is to maximize expected returns, instead of minimizing a loss function. As such, the optimization problem can be written as

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) Q^\pi(s_t, a_t) \right], \tag{2.5}$$

where subscripts t have been added to emphasize that states and actions are dependent on the time step. With nanofluidic chips, the time step corresponds to how many components have been placed. Note that the notation for the expectation value has been simplified for convenience. The simplified notation will henceforth be used.

2.3 Advantage functions and actor-critic networks

Continuous action problems are often hard to optimize. To ease optimization, the Q-values $Q^\pi(s_t, a_t)$ in equation (2.5) may be substituted for the advantage function, first presented by Gu *et al.* [16]:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t, a_t). \tag{2.6}$$

$V^{\pi}(s_t, a_t)$ is the value function, which is typically estimated by another neural network [16], [17]. This forms the actor-critic scheme, where the actor network learns the policy $\pi_\theta(a_t|s_t)$, and the critic network learns the estimate $\hat{V}^{\pi}(s_t, a_t)$. The critic network is sometimes also referred to as baseline network. This designation speaks more of the critic's purpose – it acts as a point of reference, suggesting which direction to update the policy [18]. The optimization objective with the advantage function is written as

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right], \tag{2.7}$$

where $\hat{A}_t$ is the estimated advantage at time step t.

2.4 Trust Region Policy Optimization

As an alternative to the standard actor-critic formulation in equation (2.7), Schulman *et al.* [19] proposed trust region policy optimization (TRPO). The optimization algorithm has guaranteed monotonic improvement. The surrogate objective in TRPO takes the appearance

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right], \\ \text{subject to} \quad & \mathbb{E}_t \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right) \right] \leq \delta, \end{aligned} \tag{2.8}$$

where $\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the policy in the current iteration before having taken any steps of gradient ascent. The constraint enforces that the average Kullback-Leibler divergence D_{KL} over all actions is less than a hyperparameter δ. The objective can be efficiently solved through approximations and the use of conjugate gradients [20]. TRPO often outperforms the standard objective [19], [20]. However, according to Schulman *et al.* [20], TRPO is undesirably complicated and cannot be used with noise-inducing layers such as dropout. Additionally, parameters cannot be shared between the policy and value function. The shortcomings of TRPO thus lead to the next proposed learning algorithm, proximal policy optimization (PPO).

2.5 Proximal Policy Optimization

PPO takes inspiration from TRPO by altering the objective in equation (2.8). A similar constraint is used, but instead incorporated in the objective. Setting

$$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} = r_t(\theta), \tag{2.9}$$

the main term in PPO takes the form

$$L_t^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t) \right], \tag{2.10}$$

where ϵ is a hyperparameter, usually set to 0.1 or 0.2 according to Schulman *et al.* [20]. The penalization is added to ensure that the policy update is not excessively

large. In addition to $L_t^{\text{CLIP}}(\theta)$, two more terms are added. The first is $L_t^{\text{VF}}(\theta)$, which is the value function loss, given by mean squared deviation from returns. The second term is the entropy of the policy, $S[\pi_\theta](s_t)$, which is added to promote exploration. The full optimization problem is then given by

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t) \right], \tag{2.11}$$

where c_1 and c_2 are hyperparameters controlling the influence of entropy and value function loss. PPO is meant to emulate the monotonic improvement of TRPO, under arguably simpler conditions. In addition, PPO has been found to have better empirical sample complexity than the alternatives [20].

2.6 Sampling methods

In order to improve the policy, it first needs to be evaluated in some manner. Evaluating the policy is suitably done by sampling from it. Sampling avoids the need of evaluating all actions, while still giving a fair representation of the policy. The sampling algorithms that were tested during the thesis are elaborated below.

2.6.1 Standard sampling

The policy given as output from a neural network cannot directly be interpreted as a probability distribution, as the Q-values are not non-negative and do not sum to 1. Normalization can be done by applying the softmax function on the entire policy. To get a representative image of the policy, its corresponding probability distribution can simply be sampled from a certain number of times. In the context of nanofluidic chips, one complete sample from the policy is interpreted as placing all available components on the chip surface according to the state-dependent probability distribution.

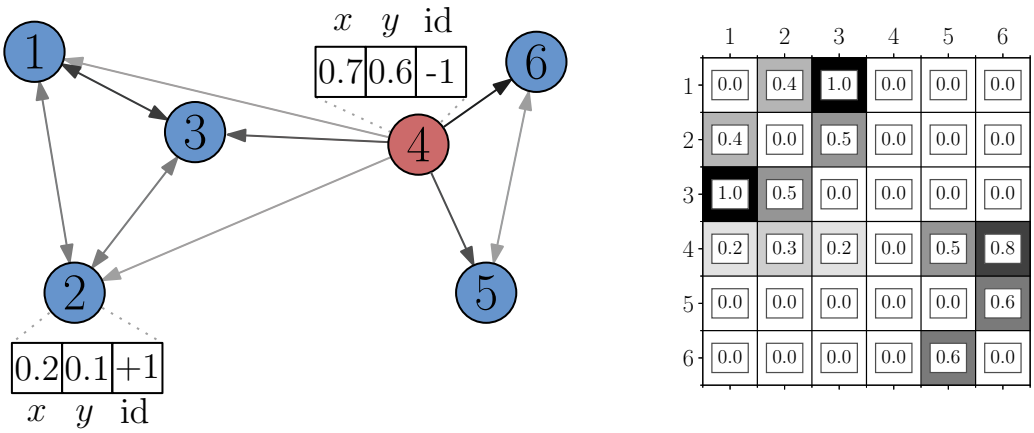
It is desirable to be able to use the policy neural network for giving definite predictions by placing components at the locations with maximal policy values. Standard sampling could lead to a good average performance, whereas individual runs may fail, and argmax placements may not succeed. It could therefore be beneficial to use a sampling scheme which gives better performance for deterministic placements, leading to the next proposals.

2.6.2 ϵ-greedy

The ϵ-greedy scheme mixes exploration with exploitation by choosing a_t as

$$a_t = \begin{cases} \text{any action} & \text{with probability } \epsilon, \\ \operatorname{argmax} \pi_\theta(\cdot|s_t) & \text{with probability } 1-\epsilon. \end{cases} \tag{2.12}$$

To promote exploration, ϵ can be chosen near 1 at first. Letting ϵ decay each generation lets the agent exploit the learned policy more. By choosing the action with the best predicted outcome, the agent is trained in a more deterministic manner than with standard sampling.



(a) **(b)**

Figure 2.2: A graph **(a)** with its associated adjacency matrix **(b)**. Each node has a set of features with identical schema. In this case, two node species exist, with identifiers ± 1. The nodes also have positions as features. The adjacency matrix is weighted, meaning that the connection strength is included. In the displayed case, the connection strength decreases with distance.

2.6.3 Policy-informed ϵ-greedy

A variant of the ϵ-greedy scheme is to combine both above approaches by choosing a_t as

$$a_t = \begin{cases} \text{sampled from policy} & \text{with probability } \epsilon, \\ \operatorname{argmax} \pi_\theta(\cdot|s_t) & \text{with probability } 1-\epsilon. \end{cases} \tag{2.13}$$

By decaying ϵ each generation, the agent is first primarily trained to optimize its average performance, while converging to a maximized deterministic performance. Sampling instead of choosing actions at random restricts the agent to making more informed decisions, while still exploring the search space.

2.7 Geometric Deep Learning

Geometric deep learning (GDL) (see for example Bronstein *et al.* [21]) leverages geometries and relations between objects. A subfield of GDL is deep learning on graphs. In graph learning, graph convolutional networks (GCN:s) have lately become a popular approach [21]. A chip canvas with placed components can be modeled as a graph, where chip components are nodes. The graph representation allows a manner of modeling particle-to-particle interactions by propagating messages through edges. Edges may be weighted, for example depending on the distance between nodes, see Figure 2.2.

2.7.1 Graph Convolutional Networks

GCN:s have been applied successfully in a wide variety of scenarios, see the overview by Bronstein *et al.* [21] and Zhou *et al.* [22]. Some examples include molecular

property identification [23], text classification [24], combinatorial optimization [25] and much more. A promising indicator for usage of GCN:s on nanofluidic chips is Mirhoseini *et al.* [5], where Google's TPU chips were optimized with GCN:s and reinforcement learning. An advantage over many other neural network architectures is that GCN:s are not specific to a specific number of nodes. The number of nodes can be changed without changing the network architecture.

Several graph convolutional operations have emerged over the years. A simple version is presented in Morris et al. [26], named GraphConv by PyTorch Geometric standard (PyG) [27]:

$$\mathbf{x}'_i = \mathbf{\Theta}_1 \mathbf{x}_i + \mathbf{\Theta}_2 \sum_{j \in \mathcal{N}(i)} e_{ji} \cdot \mathbf{x}_j, \tag{2.14}$$

where $\mathbf{x}$ is a batch of node features and $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ are weight matrices. e_{ji} denotes the edge weight from node j to i. $\mathcal{N}(i)$ is the neighborhood of node i, meaning the node indices which connect from node i. The operation (2.14) is an extension of a fully-connected layer, where additionally the neighboring nodes are weighted, summed and propagated through a fully-connected layer.

Edge features may also need to be multi-dimensional, for example if several different particle-to-particle interactions exist. With multi-dimensional edge features, one possibility is to use the graph attentional layer defined in Veličković *et al.* [28], named GATConv in PyG:

$$\mathbf{x}'_i = \alpha_{ii} \mathbf{\Theta} \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{\Theta} \mathbf{x}_j, \tag{2.15}$$

where the attention coefficients α_{ij} are calculated as

$$\begin{aligned} \alpha_{ij} &= \frac{c_{ij}}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} c_{ik}}, \\ c_{ij} &= \exp\Big(\text{LeakyReLU}(\mathbf{a}^\top \left[\mathbf{\Theta}\mathbf{x}_i \,||\, \mathbf{\Theta}\mathbf{x}_j \,||\, \mathbf{\Theta}_e \mathbf{e}_{ij}\right])\Big), \end{aligned} \tag{2.16}$$

where $\mathbf{a}$ is a weight vector, $\mathbf{\Theta}$ and $\mathbf{\Theta}_e$ are weight matrices, and $||$ is the concatenation operation. With both above layers, the order of nodes does not matter. This property can make the search space much smaller than for example encoding node features as inputs to a fully-connected neural network.

Additionally, normalizing layer outputs can noticably speed up training. The most effective type of normalization varies on application. For graphs, Cai *et al.* [29] found the following normalization to be beneficial, named GraphNorm:

$$\mathbf{x}' = \frac{\mathbf{x} - \boldsymbol{\alpha} \odot \mathbb{E}[\mathbf{x}]}{\sqrt{\text{Var}[\mathbf{x} - \boldsymbol{\alpha} \odot \mathbb{E}[\mathbf{x}]] + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}. \tag{2.17}$$

The parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ all represent learnable parameters, ϵ is a small parameter added for numerical stability, and $\odot$ is the Hadamard product.

2.8 Ant Colony Optimization

Ant colony optimization (ACO) is a stochastic optimization algorithm inspired by biological ants. When biological ants forage, they leave a trail of pheromones. The pheromone trail has been noted in some ant species to be volatile [30]. The volatile trail of pheromones tends to attract ants, following the path and further intensifying it.

The volatile trails and the tendency to follow pheromones is mimicked in ant colony optimization. A colony of usually hundreds or thousands of ants is modeled, wherein each ant can take one action at each time step. The problem statement naturally forms a graph formulation. Each action leads to a location and is weighted according to the amount of pheromones deposited on the edge connecting the two locations. Each edge also has a desirability associated with it. The desirability can for example be the inverse distance required to travel between locations. The probability of the k^{th} ant to traverse the edge e_{ij} given pheromone concentration τ_{ij} and desirability η_{ij} is then given by

$$p^k(e_{ij}) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in A_i} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \tag{2.18}$$

where A_i is the set of allowed actions to take at location i and α and β are parameters controlling the influence of pheromones and desirability. Trails are usually updated when all ants have completed their solution. The update typically takes the shape

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k}^{N} \Delta\tau_{ij}^{[k]}, \tag{2.19}$$

where ρ is the pheromone evaporation coefficient, N is the number of ants and $\Delta\tau_{ij}^{[k]}$ is the amount of pheromones deposited by the k^{th} ant.

ACO has achieved world-class performance in several tasks such as sequential ordering, scheduling, assembly-line balancing, the traveling salesperson problem, DNA-sequencing and much more, see for example the overview by Dorigo and Stützle [31]. For an introduction to ant colony optimization, see Wahde [32, chapter 4], which has been used as basis for this section.

2.9 Fluid Dynamics

The Reynolds number is the ratio of inertial forces to viscous forces and is given by the expression

$$\text{Re} = \frac{\rho u L}{\mu}, \tag{2.20}$$

where ρ is the fluid density [kg/m^3], u is the flow speed [m/s], L is a characteristic length scale [m] which depends on the geometry, and μ is the dynamic viscosity [kg/(ms)]. The Reynolds number is helpful for predicting flow patterns. The number is dimensionless, meaning that none of the scales matter by themselves, as the

fluid will behave the same for a constant Reynolds number. For example, decreasing the flow speed can be countered by increasing characteristic length scale by the same factor, giving the same Reynolds number and the same flow behavior. Fluids with Reynolds numbers above about 2300 are typically regarded as turbulent, although exceptions can occur where turbulence is seen in microfluidics for lower Reynolds numbers [33]. Micro- and nanofluidics are typically characterized by low Reynolds numbers $\mathrm{Re} < 1$ [34, chapter 2], [35], [36].

Another fluid dynamical quantity of importance for chemical transport phenomena is the Schmidt number,

$$\mathrm{Sc} = \frac{\mu}{\rho D}, \tag{2.21}$$

where D is the mass diffusivity [m^2/s]. The Schmidt number is also dimensionless and describes the ratio between momentum diffusivity and mass diffusivity [37]. The Schmidt number is usually approximately of order 1000-10000 for nanofluidics [34, chapter 7].

2.10 Discrete blockages or continuous channels

Ant colony optimization can be used to form channels to the active sites of reactivity placed by the deep learning agent. The agent can alternatively place pieces of obstacles one by one to direct the fluid. Obstacles may then be represented as component B in Figure 2.1, while active sites are of type A. The advantage of using ACO instead could be that channels are more easily formed, and the agent not needing to place as many particles. ACO would be particularly beneficial on very fine resolutions where a large amount of obstacle parts need to be placed to have a significant effect on the fluid.

On the other hand, it is not certain whether channels are the most beneficial to maximize reaction rate. One possible disadvantage of forming channels is that the speed of the current gets higher due to the narrower paths. There is more reactant flowing through the reactive sites when forming tight channels to the particles, but the trade off between reactant amount and speed is not known in detail. Placing obstacle pieces at discrete locations may bring some advantages over more well defined channels.

3
Methodology

This chapter describes the methodology employed during the thesis. The chapter firstly includes the model architecture. Secondly, testing of the GCN part in a supervised setting is described, as well as testing the entire model on simpler test cases. Some results for the preliminary test cases are presented here, as they are not strictly part of the aim of the thesis.

3.1 Model architecture

The architecture is inspired by Mirhoseini *et al.* [5]. Instead of the Edge-GNN presented in the paper, the graph encoder consists of cascaded graph convolutional layers, either GraphConv or GATConv, each followed by GraphNorm layers. The node embeddings obtained from the GCN can then be processed in three different ways, depending on the problem statement. If only one node species is present, applying global pooling on the node embeddings is sufficient to obtain an expressive enough representation for the upsampling cascade.

If more than one node species exist, the node embedding can additionally be used as key and query in a series of multi-head attention layers, while an embedding of the current node species is used as query. Each node embedding is then viewed as a point in a series – a database which is queried by the current node ID. Lastly, the index of the current node can be used to retrieve the embedding of the current node from the node embeddings.

The three constituents – graph mean, attended graph and current node embedding – relay similar semantics, but in different orders of specificity. The graph mean is an average over all node embeddings, describing the graph on a general level. The attended graph is a weighted mean, with more attention focused on the parts of the graph that are most important for the query node. Lastly, the current node embedding is a node-specific bias, further accentuating information about the specific node to be placed.

An important difference from Mirhoseini *et al.* is that every instance of the 2D deconvolutional layers was changed to bilinear upsampling followed by reflection padding and one 2D convolutional layer. Upon visual inspection of heatmaps of network policies, this variant produced much smoother policies. Deconvolutional layers tended to produce a “checkerboard” effect, where the policy values varied

greatly while still displaying the same general distribution properties. These artifacts tend to occur when the kernel size is not divisible by the strides, see Odena *et al.* [38]. Some works within super-resolution avoid the checkerboard problem by instead using upsampling followed by convolutional layers, see for example Dong *et al.* [39].

The number of upsampling layers is variable, depending on the desired output shape. The output shape can be chosen in powers of 2. During testing, the output shape 32×32 was used to speed up training. To get a higher resolution and more realistic simulations after testing the model, the output shape 128×128 was used. See Figure 3.1 an overview of the network architecture.

3.2 Supervised tuning of graph encoder

To specifically tune the architecture of the graph encoder part of the neural network, two distinct test cases were investigated, each with two node species. In both cases, the graph was formed by connecting all nodes of the same species with unit connection strength. The first case is only dependent on placed nodes, not the properties of the current node. The target position of the next node in the first case is given by

$$\mathbf{x}_{N_{t+1}} = \frac{1}{N_t} \left(\sum_{i=1}^{N_t^{(1)}} \mathbf{x}_i^{(1)} - \sum_{i=1}^{N_t^{(2)}} \frac{1}{\mathbf{x}_i^{(2)}} \right), \tag{3.1}$$

where $N_t = N_t^{(1)} + N_t^{(2)}$ is the current number of placed components and $\mathbf{x}_i^{(1)}$ is the i:th component of the first kind. The vector division is element-wise. The GCN was found to best solve this problem with two graph convolutional layers, where the number of inputs was set to 64 in both layers. The readout layer was set as a global mean pool followed by a linear output layer. GraphConv and GATConv performed similarly, with a mean-squared error of about 10^{-5}. The initial positions in the graph were chosen at random in the interval $[0, 1)$, with between 5 to 20 placed nodes with node species chosen with equal probability.

The target position in the second test case is given by the average position of the placed components belonging to its own species. In this case, information about the current node has to be added to the graph encoder for the predictions to be accurate. The model architecture is the same as in Figure 3.1, but with the policy and value network instead replaced by a fully connected feedout layer with two neurons, representing x and y position components.

The GCN architecture was tested with only GraphConv layers, with added GraphNorm and with an added single layer multi-head attention. A comparison of performances for said cases is seen in Figure 3.2. Using both GraphNorm and attention as in Figure 3.2 **(d)** was the most efficient, and was found to lower the loss with about two orders of magnitude compared with the base case in Figure 3.2 **(a)**.

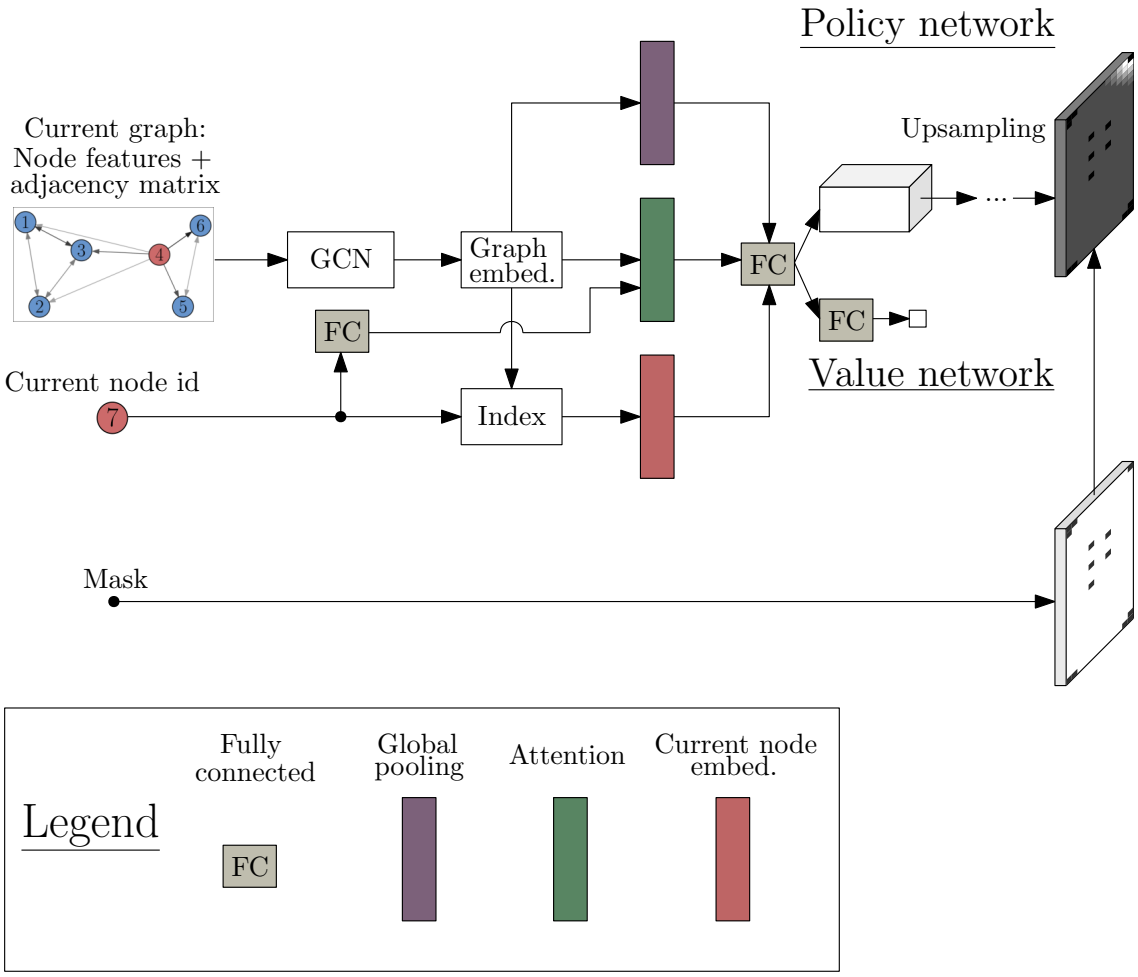
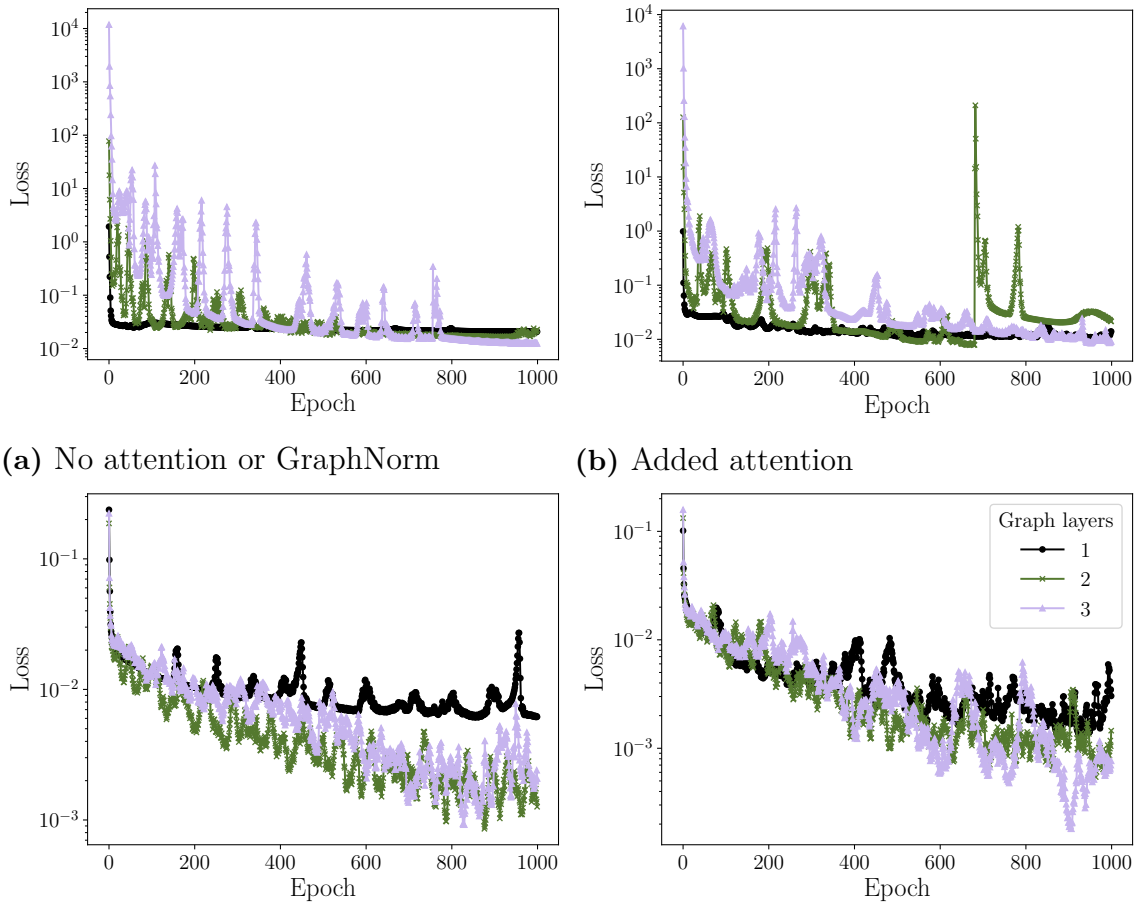


Figure 3.1: Network architecture of policy and value networks. Placed and unplaced nodes are all included. Unplaced nodes are disjoint from the graph and kept at the origin (0, 0). The current node to be placed has its ID included as feature, whereas other unplaced nodes have no ID. The GCN interprets the current graph and outputs an encoded vector embedding of all nodes. Global mean pooling is then applied on the graph to obtain a single vector representation for the entire graph. For more than one species, the current node to be placed is used to index the node embedding to get the embedded properties of the current node. The current node ID is also passed through a fully-connected layer to be used as query in multi-head attention. The node embeddings are used as key and value. The mean-pooled graph representation, attended graph and current node embedding are then concatenated and passed to policy and value networks. The policy network gives square-shaped output of placement logits. Illegal placements are masked. The value network gives scalar output.



(a) No attention or GraphNorm

(b) Added attention

(c) Added GraphNorm

(d) Added GraphNorm and attention

Figure 3.2: Comparison of different architectures for the second supervised case in section 3.2. **(a)** Three stacked GraphConv layers, each with output shape 64, achieves the best final loss, but is also the most unstable during training. **(b)** Only adding attention has no major impact. **(c)** GraphNorm has significant impact on training. Two and three stacked GraphConv layers followed by GraphNorm achieves one order of magnitude lower loss than one layer. Two and three layers perform similarly, but two layers is more stable. **(d)** Using both attention and GraphNorm can further lower loss with one order of magnitude, compared with **(c)**.

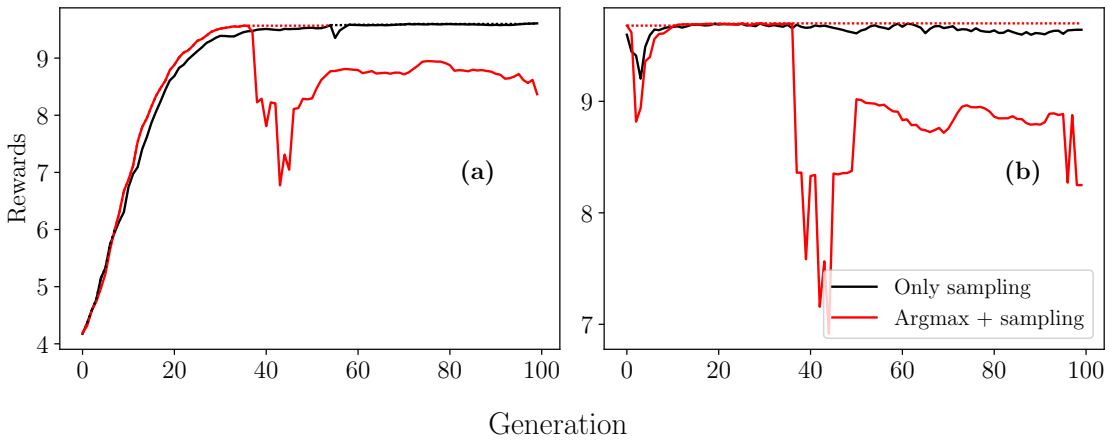


Figure 3.3: Comparison of rewards between standard sampling (black line) and policy-informed ϵ-greedy (red line). **(a)** shows the average rewards during training, and **(b)** shows the post-generational rewards with argmax placements. The dotted lines show the maximum found rewards and are color coded according to sampling algorithm. ϵ was decreased linearly from 1.0 to 0.5 from generation 1 to 100. Policy-informed ϵ-greedy attained the highest rewards with deterministic placements, but was found to be prone to collapses when ϵ was decreased too much. In this example, such a collapse happens shortly after generation 30.

3.3 Model verification on toy systems

A major part of the methodology was to verify and tweak the entire model on simple test cases. Some tests were invented to loosely emulate the behavior of interacting components on a nanofluidic chip. With this approach, lengthy CFD simulations were avoided and instead replaced with simpler and more easily verified reward functions. In this section, some of these test cases are described. Note that these cases do not describe the actual physical cases of interest in the thesis, and are rather used for verifying the model.

3.3.1 Comparison of sampling algorithms

The three sampling methods mentioned in section 2.6 were tested in a simple setting were an agent was given higher rewards the lesser the mean distance is between placed components. The basic ϵ-greedy scheme was found the least useful in all aspects. The standard sampling algorithm gave the best long-term average rewards. The policy-informed ϵ-greedy scheme more quickly gained higher average rewards and achieved the highest rewards with argmax placements. However, the scheme was found to be prone to collapses in rewards after having decreased ϵ to a certain point. Figure 3.3 compares the rewards between standard sampling and policy-informed ϵ-greedy. Both schemes have advantages and disadvantages, and the best choice largely depends on if components are sought to be placed by sampling or on argmax policy values.

3.3.2 Active components

To emulate interacting components and to attempt to solve a light problem where a graph formulation is beneficial, a test case was invented where some function is to be dampened by placing active components. The investigated function is given by

$$\begin{aligned} f(x,y) &= 2\sin(\pi x)^2 \\ x,y &\in [-1,1], \end{aligned} \tag{3.2}$$

meaning that it initially has two distinct peaks in x and is uniform in y, see Figure 3.4. One type of component exists, which dampens the flow with decreasing effect depending on the distance to the particle. The altered intensity of any point in the flow in the presence of an active site is given by

$$f_{\text{new}}(x,y) = (1 - \exp\left(-c \cdot d^2\right)) \cdot f(x,y), \tag{3.3}$$

where c is a constant determining the locality of the active site, and d is the distance to the active site. The function $f(x,y)$ is updated whenever an active site is placed. The reward R is given at the last stage of the graph depending on how high the mean squared intensity is in the end,

$$R = 10 \cdot \exp\left(-\frac{1}{N}\sum_{i,j} f_{\text{final}}(x_i,y_j)^2\right), \tag{3.4}$$

where N is the number of grid points and f_{final} is the final intensity resulting from all placed active sites. The factor 10 was heuristically added and appears to make the learning objective easier. To form the input for the graph encoder, node features were set as the component positions. All placed components are connected with edge weights exponentially decreasing with distance between components. In this way, nodes can effectively exchange information of which positions are occupied. Since the active components locally decrease the function intensity, the information exchange can be thought to decide where to place the next node at a more beneficial position.

To test the model solely on placing active components, an agent was trained to place 10 active components 128×128 approximation of the flow function in Figure 3.4. The components form a fully-connected graph with edge weights decaying exponentially with distance. The results in Figure 3.5 show that the agent has learned to place components evenly along the two maxima to dampen the mean intensity. The solution is in line with intuition and gets a reward of 9.54, where 10.00 is the maximum with particles placed on all positions.

3.3.3 Inhibitory components

To model interactions between different types of components, the test case explained in section 3.3.2 was extended with another component type. The new component type locally inhibits the effect of active sites. The final intensity at any point $\mathbf{x} =$

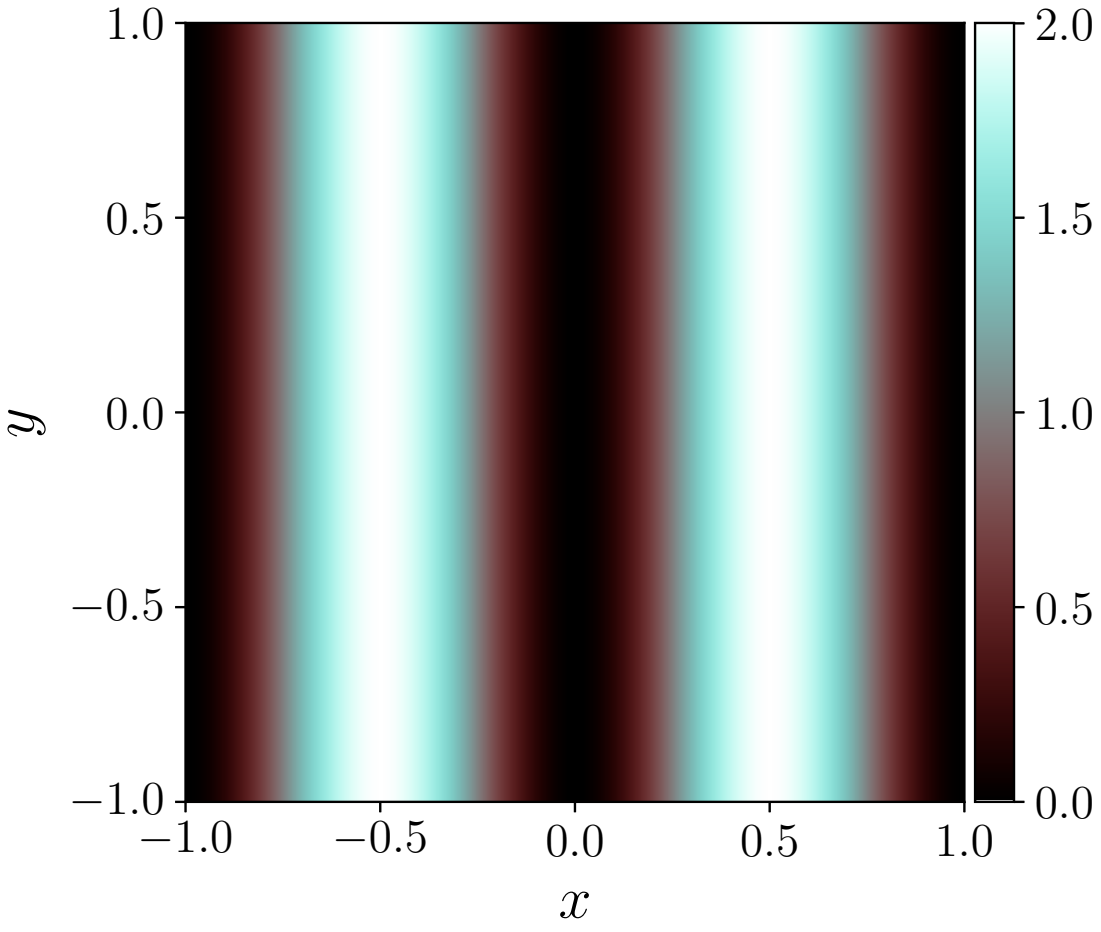
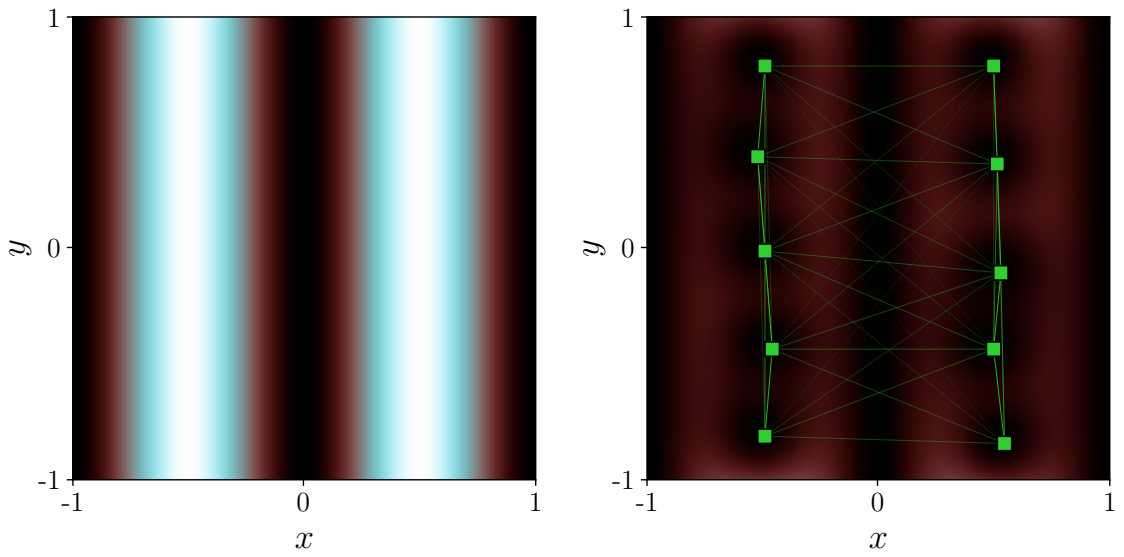


Figure 3.4: The base function $f(x, y) = 2\sin(\pi x)^2$ used in sections 3.3.2 and 3.3.3. The colorbar shows the values of the function. The custom colormap is explained in greater detail in Appendix A.



(a) **(b)**

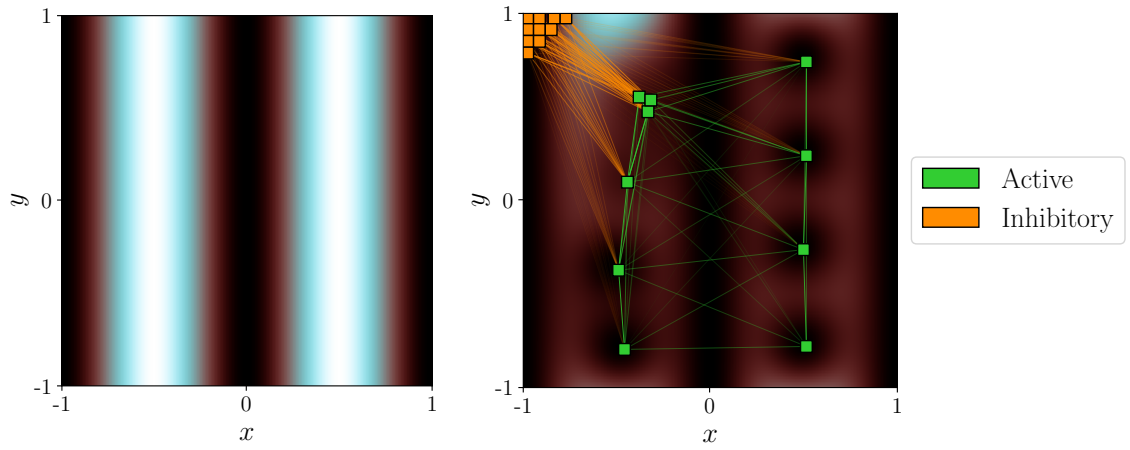
Figure 3.5: The function seen in Figure 3.4 at 128×128 resolution. **(a)** shows the unaltered function $f(x, y)$, and **(b)** shows the values after a trained agent has placed 10 active components. Active components are placed evenly spaced along the two maxima, effectively dampening the mean intensity. Edges are displayed with greater transparency the further away the connected nodes are.

(x, y) is then given by

$$
\begin{aligned}
f_{\text{final}}(\mathbf{x}) &= \sum_{j}^{N_{\text{act}}} \left[1 - \alpha_j^{-1} \exp\Big(-c_{\text{act}} \cdot d^2(\mathbf{x}, \mathbf{x}_{\text{act},j})\Big)\right] \cdot f(\mathbf{x}) \\
\alpha_j &= 1 + \sum_{k}^{N_{\text{inh}}} \exp\Big(-c_{\text{inh}} d^2(\mathbf{x}_{\text{act},j}, \mathbf{x}_{\text{inh},k})\Big),
\end{aligned}
\tag{3.5}
$$

where $d^2(\cdot, \cdot)$ is the squared Euclidean distance and subscripts "act" and "inh" implies active and inhibitory components, respectively. The same function $f(\mathbf{x})$ as before was used, c_{act} was set to 8 and c_{inh} was set to 4. Rewards are given as in equation (3.4)

An agent was trained to place 10 active and 10 inhibitory components on a 128×128 approximation of the function in Figure 3.4 such that the mean squared intensity in equation (3.5) is minimized. The state is represented effectively as in Figure 2.1, where active components are type A and inhibitory components are type B. Similarly to the Figure 2.1, type B component send messages to type A, but do not receive messages. The results in Figure 3.6 show that the agent is capable of placing active components in a near-optimal way along the function peaks, while still taking into account the presence of inhibitory components. The reward in Figure 3.6 was 8.79. As expected, the reward is somewhat lower than before since the inhibitory components affect the active components.



(a) **(b)**

Figure 3.6: The function seen in Figure 3.4 at 128 × 128 resolution. **(a)** shows the unaltered function $f(x, y)$, and **(b)** shows the values after a trained agent has placed 10 active and 10 inhibitory components. Inhibitory components are clustered in a corner, where they affect the active components minimally. Active components are placed evenly spaced along the two maxima, effectively dampening the mean intensity. Near the inhibitory components, active components are displaced from the right maximum, to avoid being too affected by the inhibitory components. A cluster of three active components is seen at this location, intuitively because the flow intensity is greater there, and requires more attention. Edges are displayed with greater transparency the further away the connected nodes are.

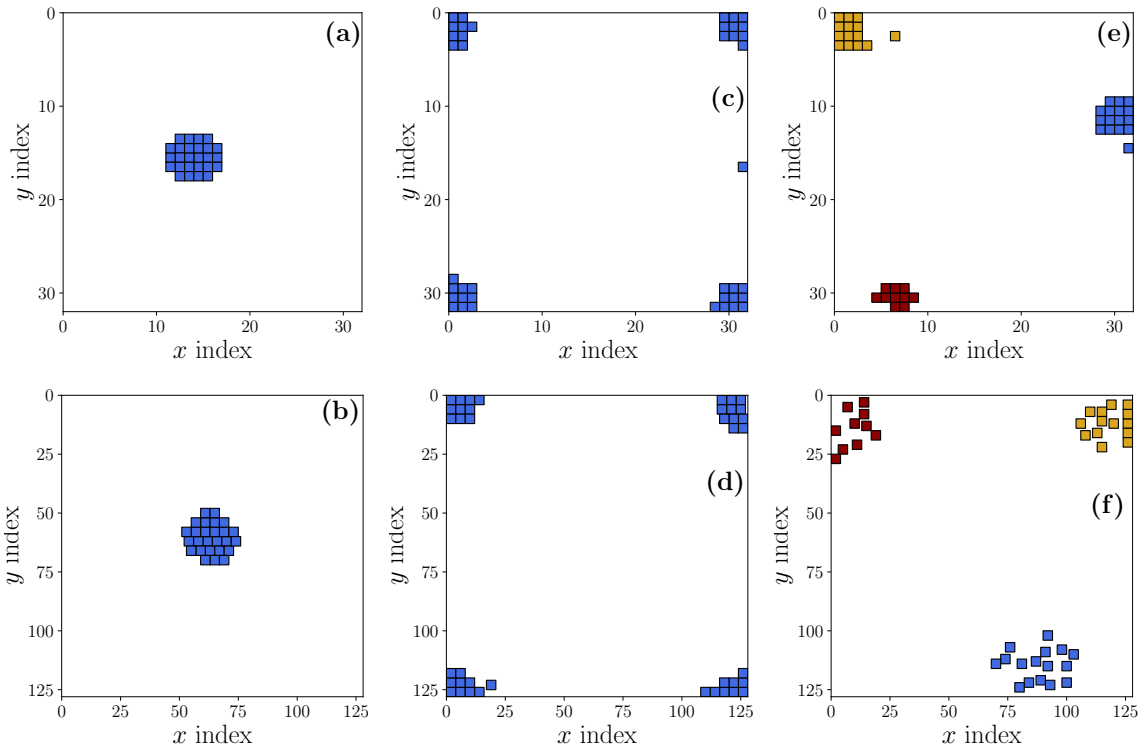


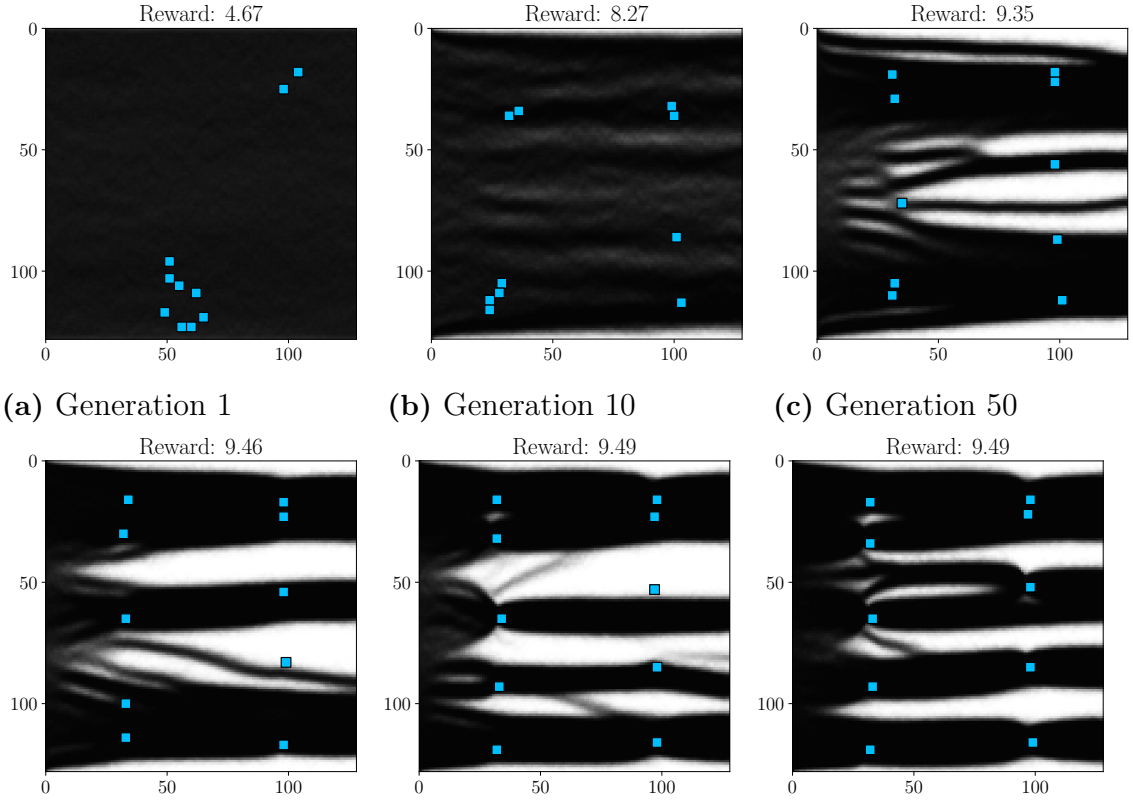
Figure 3.7: Best found final solutions for a few toy reward systems. **(a)**, **(b)** minimizing average distance between all components in on a 32×32 grid and 128×128 grid, respectively. The solution in **(a)** appears to be optimal because of the close approximation of a circle. The solution in **(b)** also appears near-optimal. **(c)**, **(d)** maximizing distance between components. Both solutions appear near-optimal. **(e)**, **(f)** minimizing distance between similarly colored components but maximizing distance between differently colored components. The best found solution in **(f)** was found by sampling, thereby the slightly higher distance between similarly colored components.

3.3.4 Other toy systems

A few other reward systems were tested to see how the RL agent performs in different scenarios. For example minimizing distance and maximizing distance between components were tested. Some results are shown in Figure 3.7, all of which the RL agent was able to find near-optimal solutions for. Especially **(a)** appears to be optimal.

3.3.5 Adding Ant Colony Optimization

As mentioned in section 2.8, ant colony optimization can be used to form channels on a chip canvas. To form channels, an extra step can be added after having sampled graphs in the policy gradient algorithm as in Algorithm 1. Ants are rewarded if they encounter an active site of reactivity, allowing them to deposit more pheromones. After all ants have deposited pheromones, heights are taken as the exponentiated negative pheromone levels. As such, fluid may flow freely through particles, but tra-



(a) Generation 1 **(b)** Generation 10 **(c)** Generation 50

(d) Generation 100 **(e)** Generation 190 **(f)** Generation 240

Figure 3.8: Placed active components in the toy model with added ant colony optimization. Brighter tone indicates greater obstacle height. Pathways are clearly formed to the particles from the inlet at the western face to the outlet at the eastern face. At this stage, however, heights do not affect the reward system, and the toy model still lacks ties to reality.

jectories where no active site is encountered receive high obstacle heights because of pheromone dissipation. In a setting where CFD simulations are run, ant colony optimization will thus alter the rewards by redirecting the current.

To test the scheme in a simple setting, Algorithm 1 was added to the reward system in section 3.3.2. In this setting, heights do not change the rewards, but nonetheless form clear paths from the inlet at the western face to the outlet at the eastern face, passing through the active sites. Since the policy exhibits high entropy in the beginning of training, active sites of reactivity will at first be sampled spuriously. As the policy becomes more certain, the ACO channels become more well-defined. See Figure 3.8 for a test case where ACO is added to the toy model.

3.4 Reward systems for designing nanofluidic chips

To simulate the functionality of nanofluidic chips, two reward systems were designed. The first one is based on CFD simulations to calculate the percentage of reactant

Algorithm 1 Retrieving heights from ACO. Ants are sampled according to the inlet fluid mass distribution (usually uniform) and travel from inlet to outlet. At each time step, ants travel one step in the principal direction of the flow (x), but a variable amount of steps perpendicular to it (y). Ants can for example, at each time step, travel straight forward, diagonally left and diagonally right, or some other set of actions. Heights are taken as the exponentiated negative pheromone levels. The number δ can for example be $1/(N \cdot A_{\text{particle}})$, where A_{particle} is the area of the encountered particle as the product of the amount of indices in the x and y directions. $\rho \in [0, 1]$ is the pheromone evaporation rate.

Inputs: Set of graphs $\mathbb{G}$ and a $d \times d$ pheromone matrix τ_{ij} describing the pheromones at positional indices $(x, y) = (i, j)$.
Outputs: Heights h_{ij}.

1. Initialize $\Delta\tau_{ij}$ as a matrix of zeros.
2. For each graph $\mathcal{G} \in \mathbb{G}$, initialize a colony of N ants with positions sampled according to the inlet fluid mass distribution.
   2.1. Let ant n travel between $(x, y) = (i - 1, j)$ to (i, k) with probability
   $$p(i, k) = \frac{\tau_{ik}}{\sum_{l \in A} \tau_{il}}.$$
   2.2. If ant n encounters an active site in $\mathcal{G}$, increase its pheromones by δ.
   2.3. If ant n has reached the end of the outlet, deposit pheromones on $\Delta\tau_{ij}$ for all visited indices (i, j). Otherwise, repeat steps 2.1-2.2.
3. After having applied step 2 for all graphs, update τ_{ij} as
   $$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{n=1}^{N} \Delta\tau_{ij}^{[n]}.$$
4. Set $h_{ij} = \exp(-\tau_{ij})$

undergoing reaction. The second one is a more computationally efficient and novel approach based on ACO, named ant fluid. The rewards in ant fluid are based on the amount of fluid units passing through active sites of reactivity.

3.4.1 Computational Fluid Dynamics

To simulate the fluid dynamics of a nanofluidic chip, the Ansys software Fluent was used [40]. Fluent can output several different quantities, but the most interesting quantities is the amount of fluid through the outlet and inlet. These quantities makes it possible to define the reactant conversion rate,

$$R_{\text{CFD}} = C \cdot \frac{x_{\text{in}} - x_{\text{out}}}{x_{\text{in}}}, \tag{3.6}$$

where x_{in} is the amount of reactant through the inlet and x_{out} is the amount of reactant flowing through the outlet. The factor C was added to scale the rewards to approximately the range 0 to 10, and was typically chosen as 100.

The CFD reward system should approximately simulate the actual fluid dynamics of a nanofluidic chip, as it operates with continuum dynamics which was mentioned section 2.1. The reward system takes as input a list of indices corresponding to which discretized faces are active of the bottom chip wall. If a fluid element passes over an active face, it experiences a first-order reaction. In addition to the list of active face indices, the reward system also takes as input a list of inactive volume indices, which can range from fully blocked to completely free from blockage.

The main disadvantage of using the CFD reward system is that it introduces a significant computational overhead, far greater than sampling graphs and training the reinforcement learning agent. To combat the difficulties introduced by the CFD solver, the lighter reward system ant fluid, based on ACO, was also developed.

3.4.2 Ant Fluid

The ant fluid reward system envisions ants as individual units of reactant. Rewards are based on how many ants pass through an active site of reactivity. Ant fluid attempts to optimize the chip such that all possible fluid paths reach as many active sites as possible, under some movement constraints. Additionally, in a system such as an ammonia synthesis reactor with ruthenium and iron catalysts, ants individually keep count of how many catalysts they have reached. If an ant reaches a ruthenium-type catalyst, it receives a higher score than if it reaches an iron-type catalyst, but with diminishing returns depending on how many catalysts it has previously reached. This is to emulate the behavior of ammonia lowering the activity of a ruthenium catalyst. The pipeline is shown in greater detail in Algorithm 2.

As before, obstacle heights are retrieved as $h_{ij} = \exp(-\tau_{ij})$ after having applied the algorithm to all sampled graphs. With a C-compiled implementation, the reward system introduces no major computational overhead, even for thousands of ants.

Algorithm 2 The ant fluid algorithm for rewarding a single sampled graph. Ants are sampled uniformly over the inlet and travel from inlet to outlet. Ants can for example travel straight forward, diagonally left and diagonally right, or some other set of actions. Ants always travel one step in the principal direction of the flow (x), but a variable amount of steps perpendicular to it (y). After having run the algorithm for all sampled graphs, obstacle heights are taken as the exponentiated negative pheromone levels. The number δ can for example be $1/(N \cdot A_{\text{particle}})$, where A_{particle} is the area of the encountered particle as the product of the amount of indices in the x and y directions. $\rho \in [0, 1]$ is the pheromone evaporation rate, and $a > 1$ controls how much higher the activity of a ruthenium-type particle is compared to an iron-type particle.

Inputs: A sampled graph $\mathcal{G}$ and a $d \times d$ pheromone matrix τ_{ij} describing the pheromones at positional indices $(x, y) = (i, j)$.
Outputs: Reward R_{ant}.

1. Initialize $\Delta\tau_{ij}$ as a matrix of zeros.
2. Initialize a colony of N ants with positions sampled according to the inlet fluid mass distribution.
   2.1. Initialize the product count $c = -1$.
   2.2. Let ant n travel between $(x, y) = (i - 1, j)$ to (i, k) with probability
   $$p(i, k) = \frac{\tau_{ik}}{\sum_{l \in A} \tau_{il}}.$$
   2.3. If ant n encounters an active site in $\mathcal{G}$, increase c by 1 and the ant's pheromones by
   $$\begin{cases} a\delta \cdot e^{-c}, & \text{if the particle is ruthenium-like} \\ \delta, & \text{if the particle is iron-like.} \end{cases}$$
   2.4. If ant n has reached the end of the outlet, deposit pheromones on $\Delta\tau_{ij}$ for all visited indices (i, j). Otherwise, repeat steps 2.2-2.3.
3. Set the reward R_{ant} as the average of all ants' pheromone levels.

4
Results

In this chapter, resulting chip designs are presented. Designs are compared with basic reference chips to obtain a baseline of performances and expected reactant conversion rates. Firstly, agents are trained to place active catalyst sites on existing geometries without any obstacles. Later, the RL agent is also trained to slightly alter the chip geometry by placing obstacles. The agent is then also trained to place both ruthenium-like and iron-like active sites along with blockages. Lastly, the agent is trained with the ant fluid reward system to form channels and decide the chip geometry completely, while placing active sites of both sorts.

For all tests, the edge weights between placed active sites of the same species are set to $\exp(-d_{ij})$, where d_{ij} is the distance between sites i and j. If obstacles are present, edge weights are defined from obstacles to active sites, again with weights exponentially decreasing with distance. See also Appendix B for a detailed table of used neural network and PPO parameters.

4.1 Manual placements

Three designs were manually created and are shown as Figures 4.1 **(a)**-**(c)**. The reactant conversion rates of these designs are not necessarily meant to be surpassed since the conversion rate depends heavily on the chip geometry. The designs are rather there to give a reasonable baseline for comparison and to attempt to find a pattern in what constitutes a high quality chip. All designs feature 16 active catalyst sites, to compare how geometry and placements affects conversion rate. Black background signals non-obstructed volume, whereas white background means fully blocked.

The reactant inlet is at the left face, and the outlet is at the right face. The actual physical dimensions of the chips are not nanoscale, but the Reynolds and Schmidt numbers were chosen to be representative of a nanofluidic system. The Reynolds number was chosen to be $\mathrm{Re} \approx 0.1$ and the Schmidt number as $\mathrm{Sc} \approx 1000$. The grid resolution is $3.125 \cdot 10^{-5}$ m and the average speed of solution on a non-obstructed canvas was chosen as 0.001 m/s, meaning that a fluid element passes from inlet to outlet in the 32×32 grid in one second. Chip designs were evaluated with Ansys Fluent with a mesh resolution of $128 \times 128 \times 10$ in $x-, y-$ and $z-$coordinates. The $z-$coordinate is perpendicular to the chip canvas and represents the depth of the chip.

The first design, **(a)**, is similar to Fritzsche *et al.* [41], where gold antennas are used as catalysts. As can be expected from decreasing amount of reactant in the channels, the reaction rate is the highest by the earliest active sites. This behavior is seen in Figure 4.1 **(d)**.

The second design **(b)** was made with inspiration from **(a)** to investigate what happens when all sites are connected in a long, meandering path. The same tendency of decreasing reaction rate along the channel is seen in **(e)**, but the reaction rate also decreases for the middle sites, which are not situated in curvatures.

The third design **(c)** was created to supply a roughly even amount of reactant to all active sites. Placing the column of active sites by the inlet was found to achieve the highest conversion rate as opposed to placing it further downstream. The top and bottom active sites appear to be slightly more reactive than the others as seen in **(f)**, which is likely due to a slight increase in pressure by the edges. The flow is at first uniform over all indices, meaning that the channel ranging from y indices 8 to 24 immediately tightens the flow, which seemingly increases the pressure by the edges.

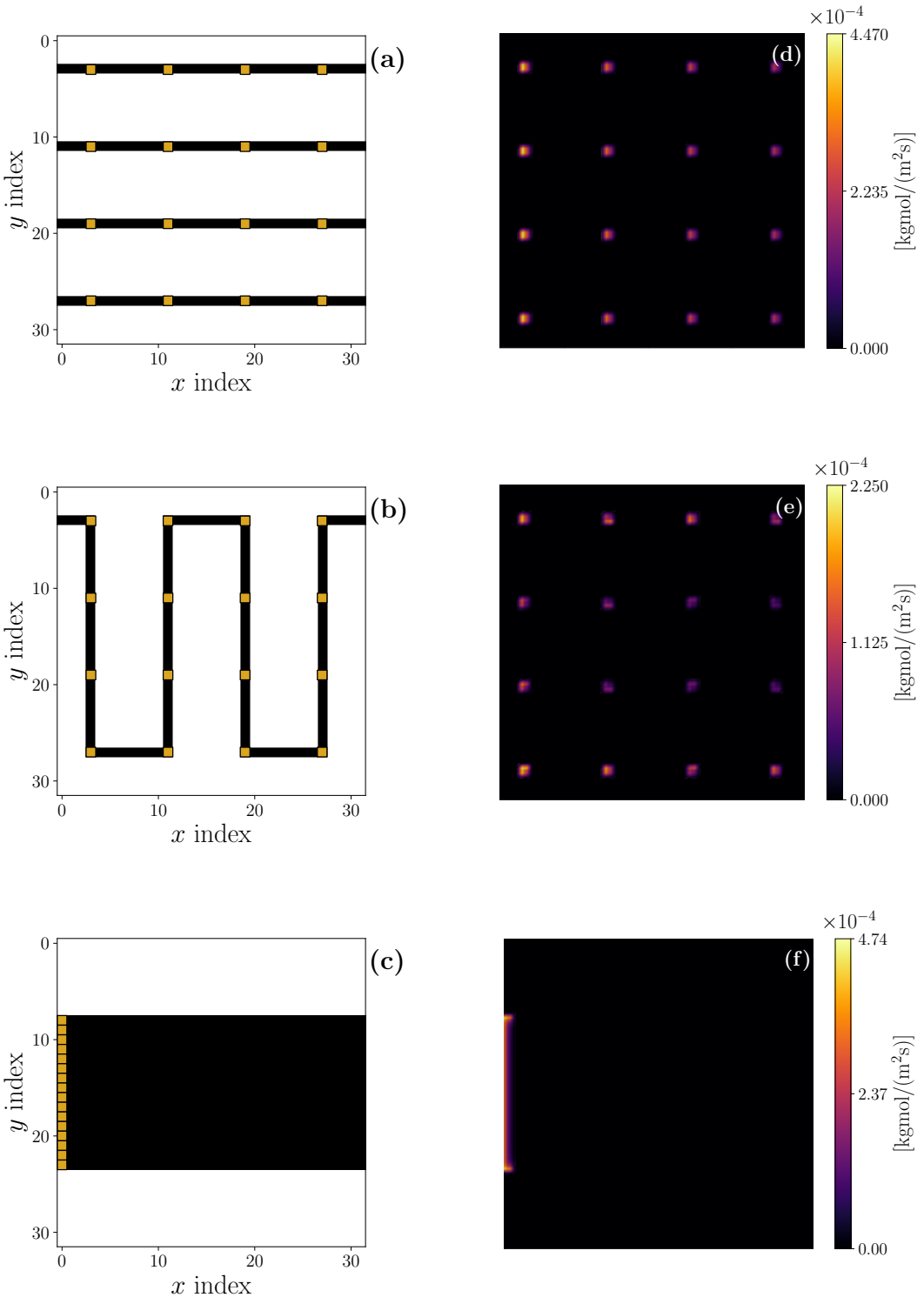


Figure 4.1: Reference designs **(a)**-**(c)** with corresponding kinetic rates of reaction in kgmol/(ms^2) **(d)**-**(f)**. **(a)** is similar to the design by Fritzsche *et al.* [41]. **(b)** was constructed with inspiration from **(a)**. **(c)** was constructed to give roughly the same flow through each of the active sites. Placing particles in a column by the inlet was found to give higher conversion rate compared with placing them towards the outlet. In all cases, there is a pressure gradient between inlet and outlet.

The increased activity for active sites situated by the turning points in Figure 4.1 **(e)** may be because of increased pressure by the turning points, see Figure 4.2. If increased pressure leads to higher activity, placing obstacles at strategic locations may also have a positive effect on reaction rate by forming points of increased pressure near active sites.

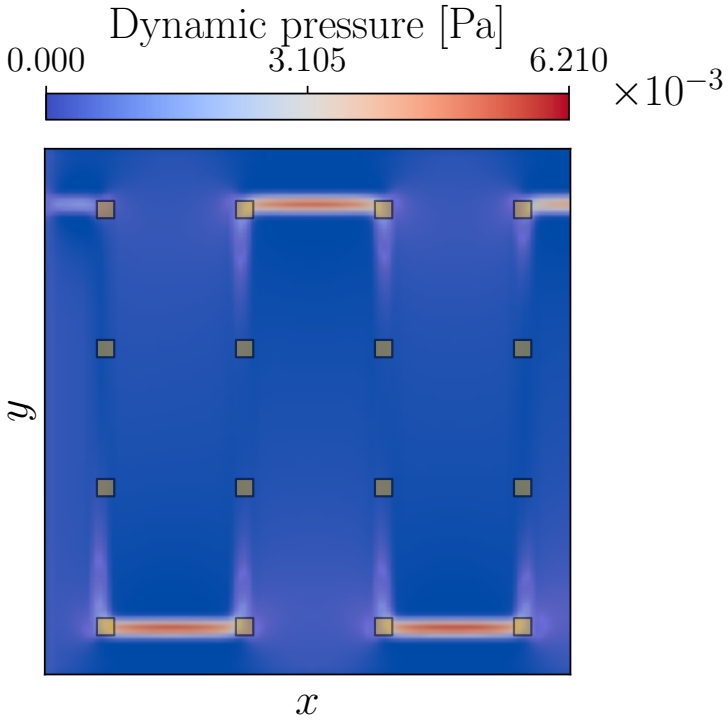
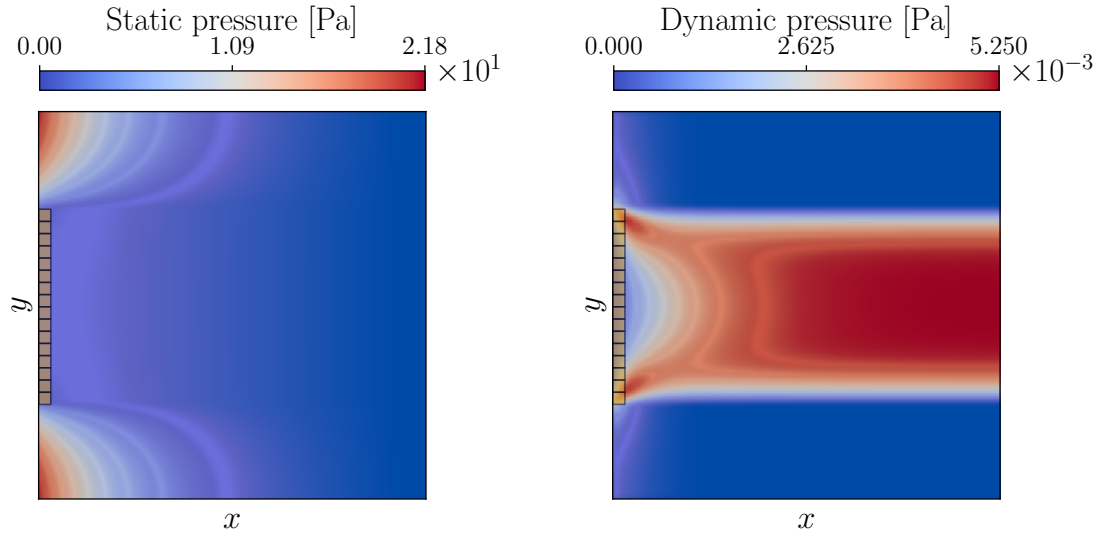


Figure 4.2: The dynamic pressure (kinetic energy per unit volume) is higher by the turning points in the channel. The higher pressure may help explain why the activity is higher by the active sites situated at the turning points in Figure 4.1 **(e)**.

Increased pressure is also typically seen by the inlet, since the driving force of a liquid is pressure difference. Figure 4.3 shows that the pressure is the greatest by the inlet, where active sites were empirically found to have the highest activity as shown by the placement in Figure 4.1 **(c)**.



(a) **(b)**

Figure 4.3: **(a)** the static pressure is the highest by the inlet. Higher pressure can lead to higher reaction rate, which explains the empirical observation that active sites have higher activity closer to the inlet. **(b)** the dynamic pressure is the highest by the edges of the inlet, which further could explain the somewhat higher reactivity by the edges.

4.2 Wide tunnel optimization

An agent was trained to place 16 active sites on the same geometry as in Figure 4.1 **(c)**. Statistics from training are shown in Figure 4.8. The best chip placement from the training session was exactly the same as in Figure 4.1 **(c)**. In the simple case where all active sites fit as a single column in a tunnel, the best solution thus appears to be to place all of them by the inlet, as suggested by both empirical evidence and findings of the RL agent.

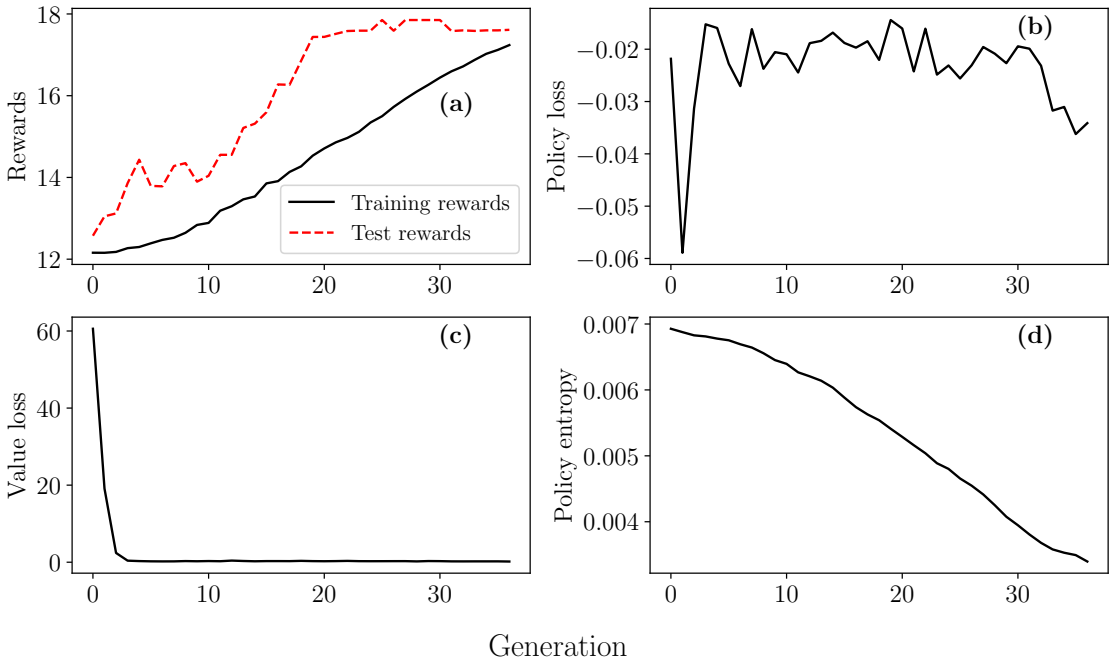


Figure 4.4: Statistics from training an agent to place 16 active sites on the same geometry as in Figure 4.1 **(c)**. **(a)** the average rewards from sampling during training and the test rewards from placing active sites on the highest policy values each step. **(b)** the policy "loss" L_t^{CLIP} from equation (2.10) per generation. Note that L_t^{CLIP} can be negative. **(c)** the value network loss per generation. The loss quickly decreases from the first few generations, after which it stays rather constant. **(d)** the entropy of the network policy. The entropy decreases overall, signaling that the policy becomes more and more deterministic. The best chip placement from the training session was exactly the same as in Figure 4.1 **(c)**.

If not all active sites fit by the inlet, however, more intricate designs are needed. If, for instance, 40 active sites need to be placed, the majority will need to be placed away from the inlet. To avoid local droughts of reactant concentration, it may additionally be beneficial to place sites with some distance apart from each other. One solution would be to place one column by the inlet, one halfway to the outlet, and half a column by the outlet, see Figure 4.5. This is slightly reminiscent of the situation in Figure 4.1 **(a)**, but with one wide tunnel and wider catalyst sites.

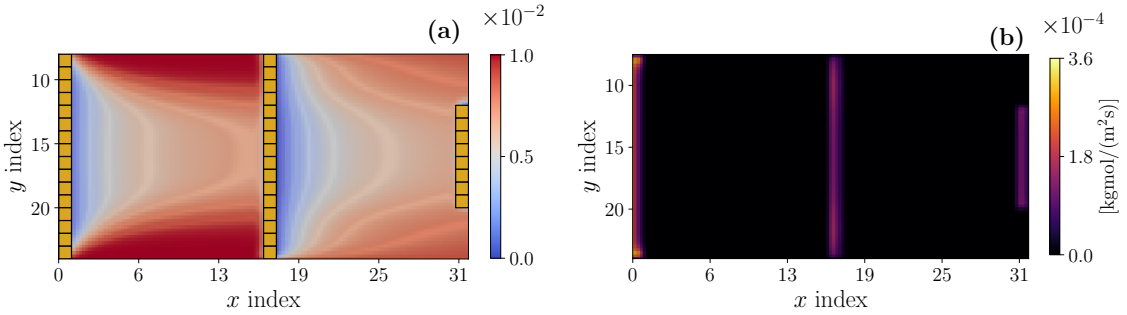


Figure 4.5: **(a)** the best found placement through trial-and-error of 40 active catalyst sites with molar reactant concentration and **(b)** the corresponding kinetic rate of reaction. The reaction rate appears even over the individual columns.

The solution found by the RL agent was slightly different. A column was placed perfectly by the inlet. Two additional active sites were placed immediately after the column by the edges, where the pressure is high. In the middle of the chip, sites are placed porously but seemingly without clustering tendencies, similar to a low-discrepancy sequence. The agent has also placed catalysts arranged almost as a smaller column by the outlet. The agent was able to achieve 98.6 % of the conversion rate that the manually constructed design did. The reaction rate over each individual column seen in Figure 4.5 **(b)** is even, which is intuitively desirable in order to evenly utilize all sites. The sites placed by the RL agent are not as evenly active, but some sites in the middle are particularly active.

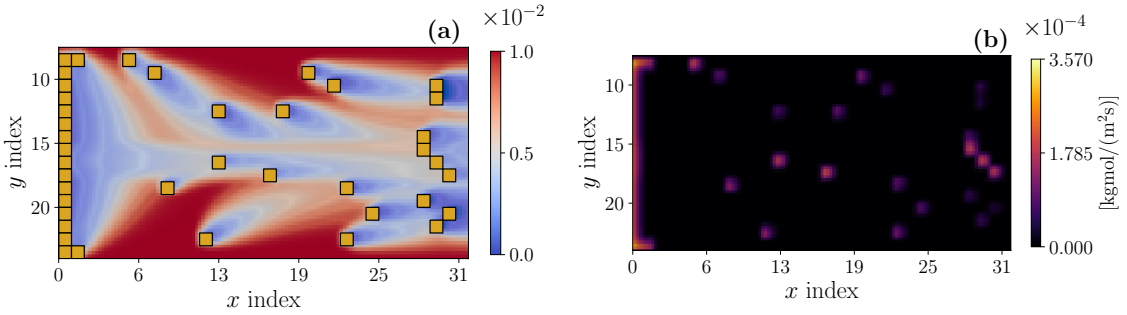


Figure 4.6: **(a)** the best RL agent placement of 40 active catalyst sites with molar reactant concentration and **(b)** the corresponding kinetic rate of reaction. The reaction rate is not as evenly distributed over all active sites as the manual placement in Figure 4.5, and some middle sites are particularly active.

Placing sites in columns thus seems like a good strategy in the case of wide channels, and the RL agent is almost able to reach the same performance. Additionally, the RL agent can be used to fine-tune the currently best found design under the constraints of active sites being placed in columns. Sites can then be modeled as three elongated

single sites, which the RL agent is tasked to place. An agent was trained to place two columns of 16 active sites each and one column of 8 active sites. The agent was able to achieve 100.07 % of the reaction rate as the manual placement in Figure 4.5. The optimization is quite minuscule, and the manual placement seems close to optimal.

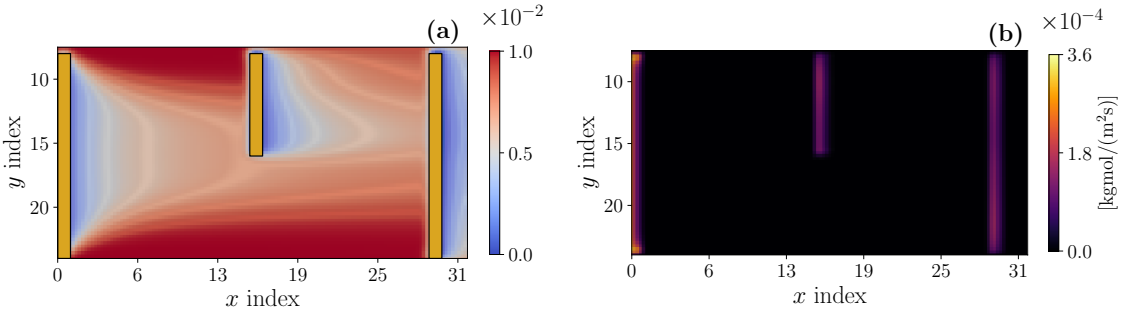


Figure 4.7: **(a)** the best RL agent placement of two columns of 16 active sites each and one column of 8 active sites with molar reactant concentration. **(b)** the corresponding kinetic rate of reaction. The reaction rate surpasses the manual placement in Figure 4.5 by 0.7 %.

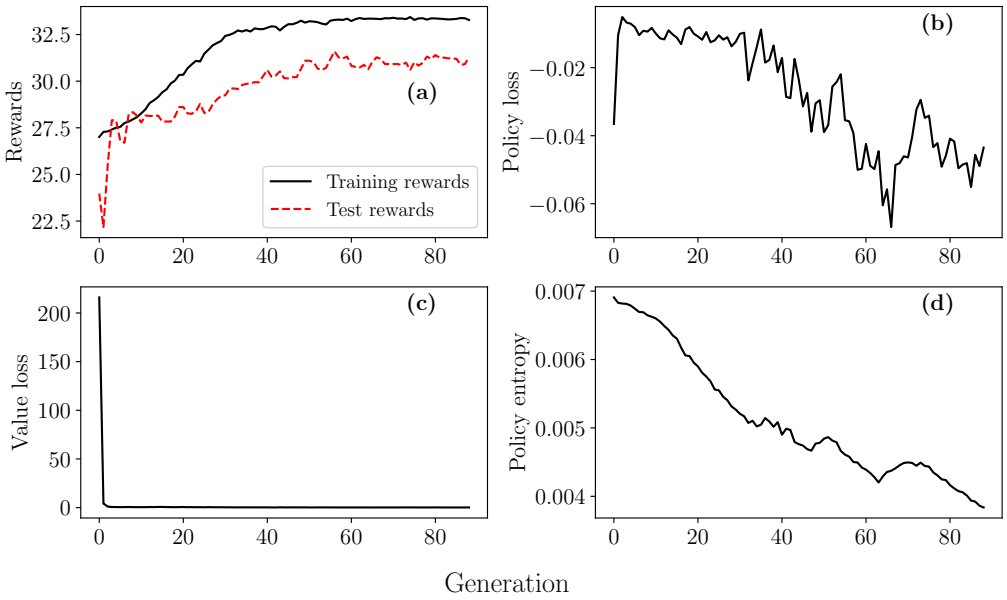


Figure 4.8: Statistics from training an agent to place 40 active sites on the same geometry as in Figure 4.1 **(c)**. **(a)** the average rewards from sampling during training and the test rewards from placing active sites on the highest policy values each step. Average training rewards are mostly higher than test in this case, suggesting that sampling is more successful than argmax placements. **(b)** L_t^{CLIP} from equation (2.10) per generation. **(c)** the value network loss per generation. **(d)** the entropy of the network policy.

4.3 Narrow tunnel optimization

Agents were trained to place 16 active sites on a narrow geometry where only y-indices 15-18 are free. A few manual reference designs are shown in Figure 4.9, for comparison with reactant conversion rates. **(a)** features all active sites by the inlet, where the pressure is the highest. **(b)** is a low-discrepancy Sobol sequence meant to counteract low concentrations of reactant reaching active sites. The third design, **(c)**, features evenly spaced columns of active sites, to attempt to gain high reactant concentrations due to even spacing, but also an even flow to all sites.

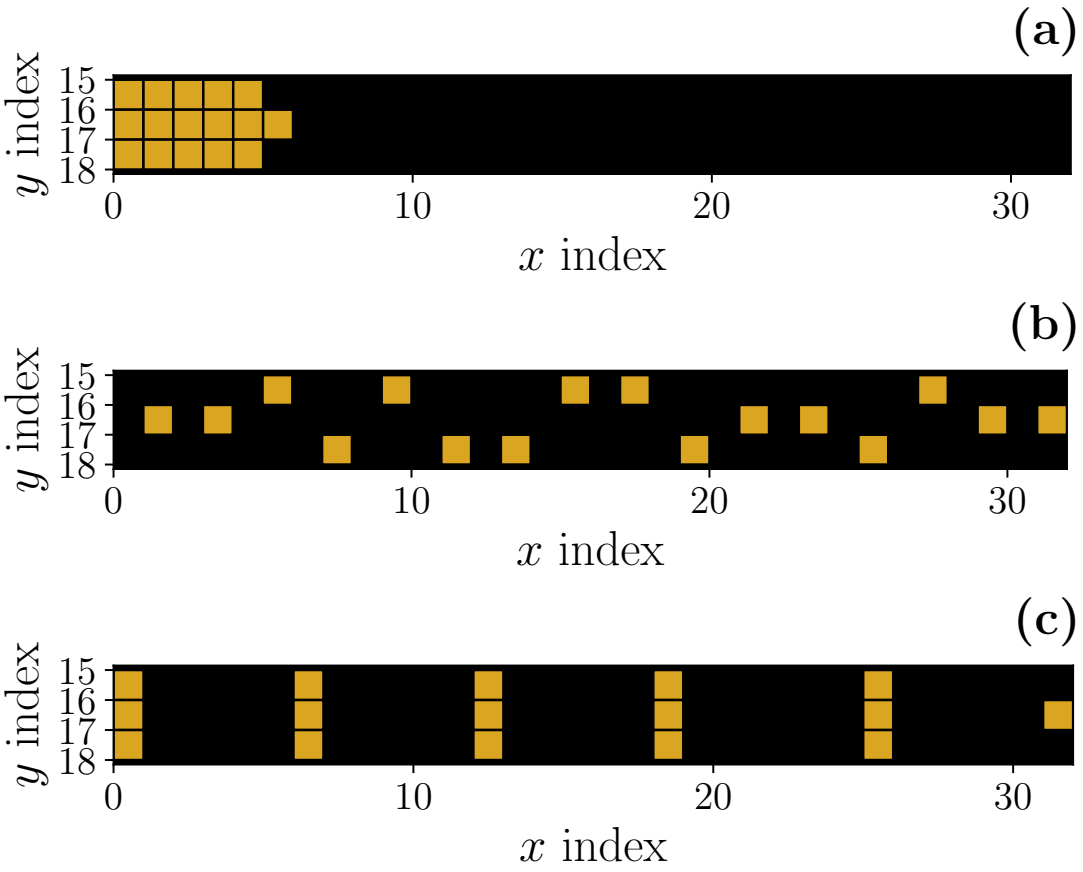


Figure 4.9: Manual reference designs for narrow tunnel optimization. **(a)** a cluster of active sites by the inlet. **(a)** a low-discrepancy Sobol sequence. **(c)** evenly spaced columns of active sites.

Kinetic rates of reaction along with molar fraction of reactant for the designs in Figure 4.9 are shown in Figure 4.10. Almost only the outer catalysts are active for the first design, which is shown in **(a)**. As can be seen in **(d)**, the molar fraction of reaction is low at the inner catalysts, which likely causes the low reactivity. The catalysts in the Sobol sequence are more evenly active as seen in **(b)**, and the fraction of reactant just before the catalysts is typically higher, as shown in **(e)**. The design with catalysts in columns also has a more even reactivity which is shown in **(c)**, but the top and bottom catalysts appear more active. The higher reactivity also coincides well with the reactant fraction, which is higher before the top and bottom

catalysts, see **(f)**. High reactant fraction should naturally be expected to give higher reactivity, since more reactant particles can reach the catalysts. From the results in Figure 4.10, some conclusions can be drawn. Firstly, it is not preferable to place catalysts in clusters, since this immediately introduces low reactant concentration for the innermost catalysts. Secondly, catalysts placed in columns are again found to be favorable designs. Sparsely distributed catalysts such as in Figure 4.10 **(b)** are favored over clusters, but do not outperform the column design. The differences in conversion rate are seen in Table 4.1.

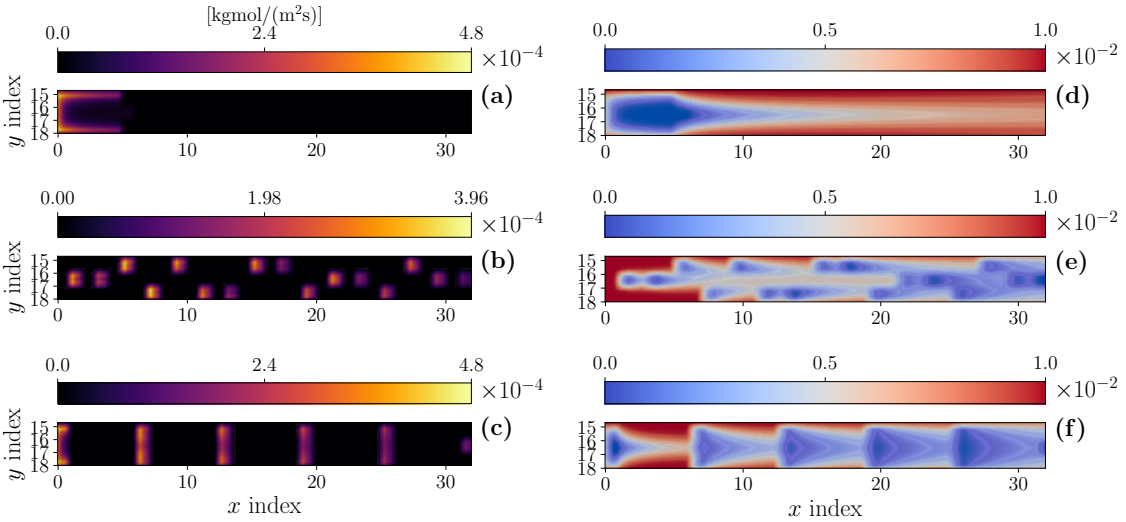


Figure 4.10: Kinetic rates of reaction **(a)**-**(c)** for designs shown in Figure 4.9 along with molar fraction of reactant **(d)**-**(f)**.

Some designs made by the RL agent are shown in Figure 4.11. **(a)** is an early generational design. **(b)** is a later design and accomplishes a significant 31 % higher conversion rate than **(a)**. The higher rate could be due to lesser degree of clustering, as **(b)** appears more porous, allowing fewer droughts of reactant concentration. **(c)** features elongated columns of sites, effectively consisting of 16 single active sites. The setup was made to fine-tune the design in Figure 4.9 **(c)**. **(d)** features volume blockages shown as white squares. Designs **(b)**-**(d)** outperform all manual designs in Figure 4.9, see Table 4.1.

Catalyst reaction rates and molar fractions of reactant for the RL agent placements in Figure 4.11 are shown in Figure 4.12. All reaction rates **(b)**-**(d)** appear to be quite even, but the reaction rates in the early generational placements in **(a)** are not as even and generally lower. The reaction rates are again reflected by the molar fractions of reactant in **(e)**-**(h)**.

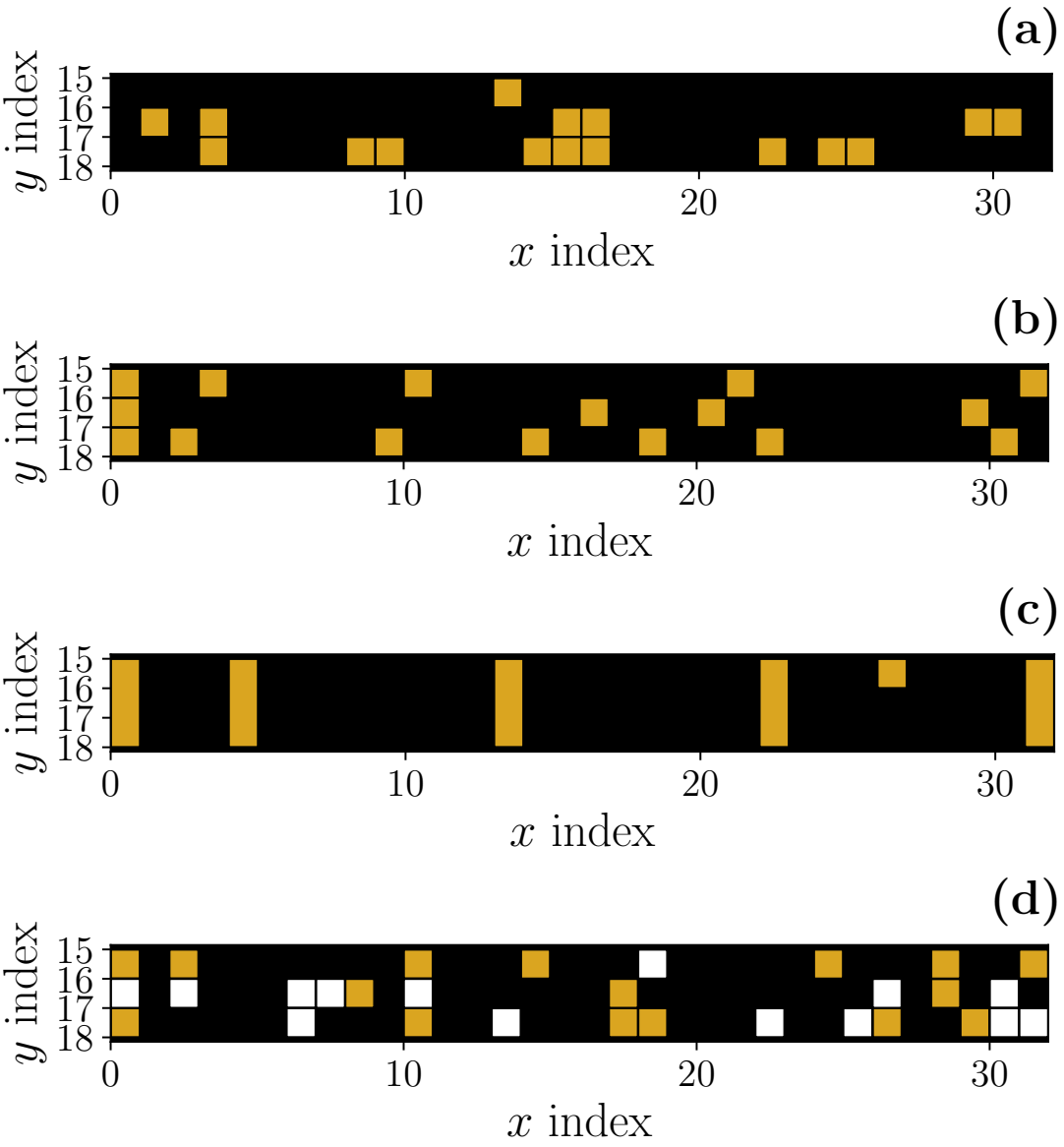


Figure 4.11: Catalyst site placements in a narrow channel by the RL agent with 16 active sites. **(a)** an example of site placement with tendencies of clustering. **(b)** a better found placement with 31 % higher reactant conversion rate than **(a)**. **(c)** active site placements under the constraint that sites are placed in columns. **(d)** active site placements with added obstacles (white squares). Designs **(b)**-**(d)** surpass all designs in Figure 4.9 in conversion rate.

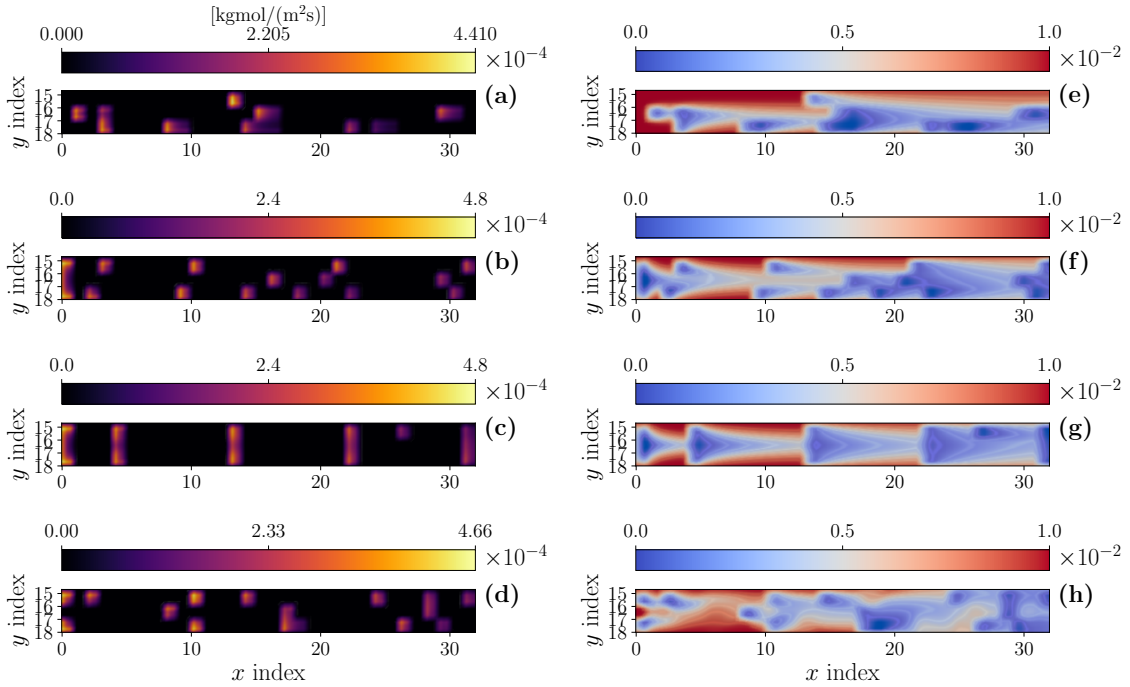


Figure 4.12: Kinetic rates of reaction **(a)**-**(d)** for designs shown in Figure 4.11 along with molar fraction of reactant **(e)**-**(h)**.

Table 4.1: Comparisons of conversion rates relative to Figure 4.9 **(a)**. All three best-found RL agent placements in Figure 4.11 are highlighted in bold text and outperform the manual reference placements in Figure 4.9. The designs in Figure 4.1 are also included to compare with a few different geometries. Out of the designs in Figure 4.1, **(a)** performs the best.

Chip design (Figure number)	Relative conversion rate
4.1 **(a)**	1.707
4.1 **(b)**	1.437
4.1 **(c)**	1.378
4.9 **(a)**	1.000
4.9 **(b)**	1.457
4.9 **(c)**	1.604
4.11 **(a)**	1.258
4.11 **(b)**	**1.646**
4.11 **(c)**	**1.663**
4.11 **(d)**	**1.651**

Upon investigation of the site placements with included volume blockages, active sites appear to be placed mostly at areas of increased dynamic pressure. The ones at the inlet are an exception, as the static pressure is the highest at the inlet, and seems to contribute more than dynamic pressure. Figure 4.13 shows the active site placements in Figure 4.11 **(d)** with dynamic pressure in the background.

Summarizing the results in section 4.3, the RL agent is able to outperform all narrow tunnel reference designs. Additionally, it outperforms the designs in Figure 4.1 **(b)** and **(c)**, but not **(a)**. 4.1 **(a)** and **(b)** have the same catalyst placements, but not the same geometries. Evidently, the geometry alters the relative conversion rate significantly from 1.437 to 1.707. The four-tunnel design in Figure 4.1 **(a)** may be especially suited from supplying a high amount of reactant to all catalysts, while not building up too high droughts of reactant such as design 4.1 **(b)** due to placing all catalysts after each other. Narrower tunnels appear especially apt at achieving high conversion rate, but the geometry of the tunnel plays an important role as well.

However, the Reynolds number also changes the conversion rate. It was found that lowering the Reynolds sufficiently made the conversion rate of design 4.1 **(b)** higher than **(a)**. This could for example be due to better mixing in design 4.1 **(b)**, or it could be an artifact of the CFD simulation. Figure 4.2 shows an increased dynamic pressure between for example the top-left catalyst and the top-middle-left catalyst. This could indicate leakage from the tunnel into the fully blocked area, which physically cannot happen but would nevertheless likely lower conversion rate.

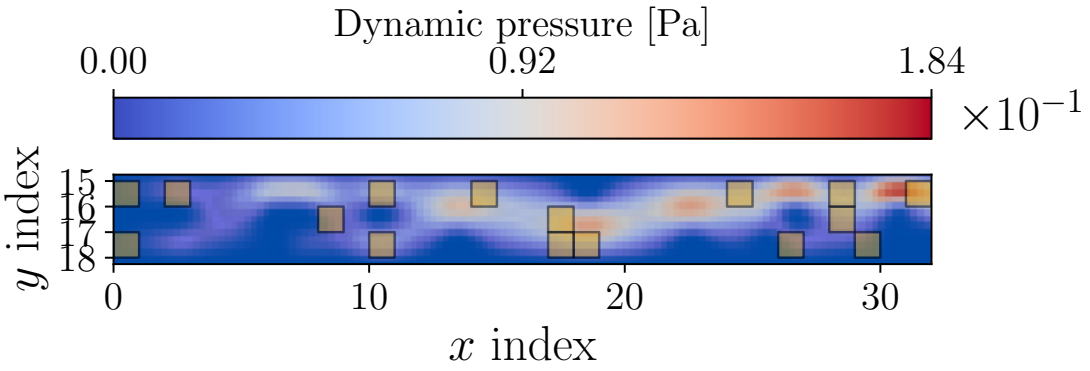


Figure 4.13: Active site placements from Figure 4.11 **(d)** with dynamic pressure in the background. Active sites are mostly placed at areas of increased dynamic pressure.

4.4 Ruthenium-iron reactor optimization

To test the RL agent in a more complex scenario, it was trained to place 20 ruthenium-like and 20 iron-like active sites. The best found placement is seen in Figure 4.14 **(b)**. Additionally, it was trained to also place 10 obstacles. The best found placement with obstacles is seen in **(c)** A manual placement was also made for comparison, which is seen in **(a)**. The RL agent outperformed the manual placement in both cases, see Table 4.2. Interestingly, the agent had quite the different approach from the manual placement. The manual placement was made to not accumulate product for the ruthenium-like active sites by placing them early. The RL agent instead tends to place ruthenium-like sites after iron-like sites.

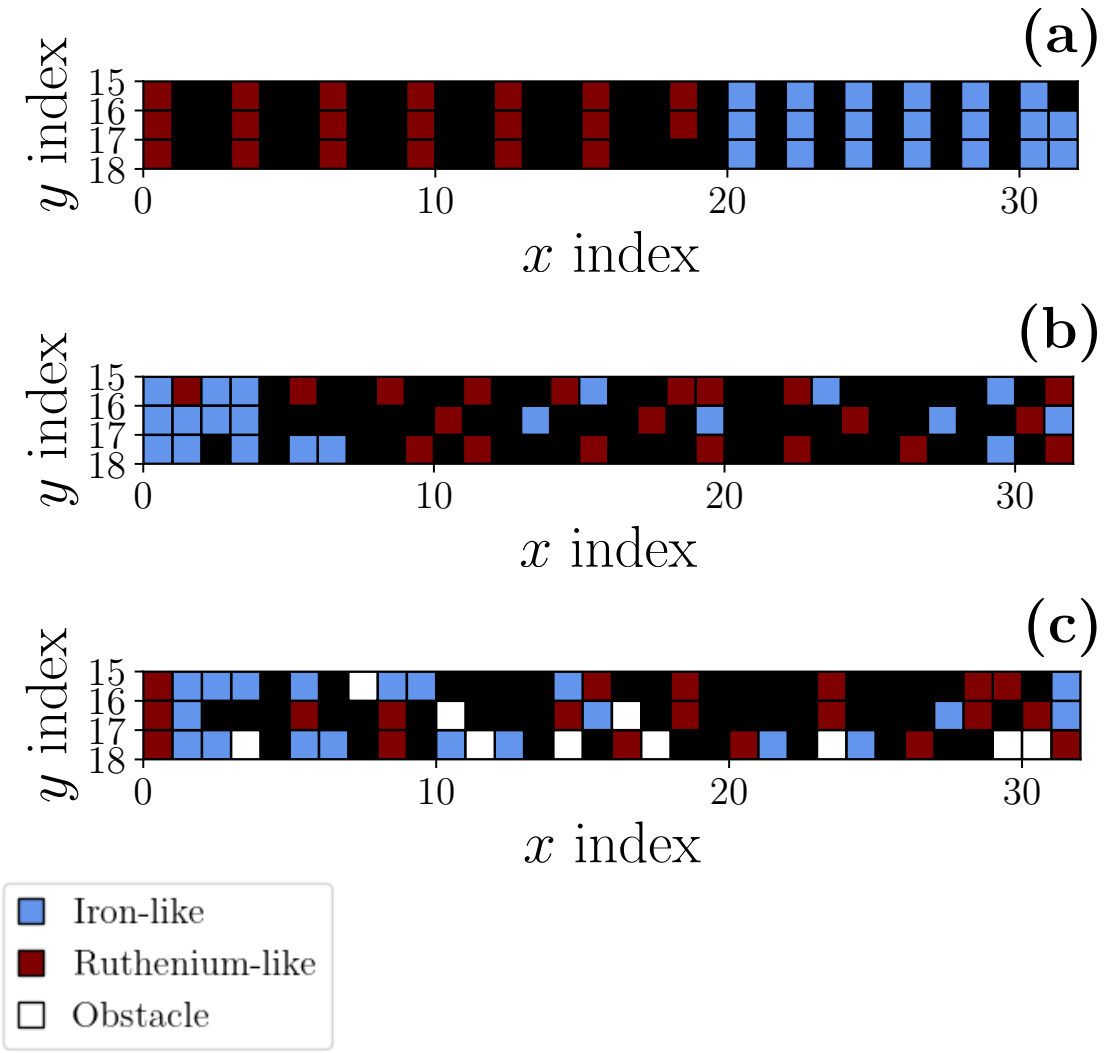


Figure 4.14: Placements of ruthenium- and iron-like active sites in narrow tunnels. **(a)** manual placement made as a baseline. **(b)**, **(c)** RL agent placements without and with added obstacles, respectively. Both **(b)** and **(c)** outperform **(a)**.

Table 4.2: Reactant conversion rates for placements in Figure 4.14, relative to 4.14 **(a)**. **(b)** and **(c)** are 6.5 % and 6.7 % better than **(a)**, respectively.

Chip design (Figure number)	Relative conversion rate
4.14 **(a)**	1.000
4.14 **(b)**	1.065
4.14 **(c)**	1.067

The differences in conversion rate in Table 4.2 are not significant, but the agent was only tested against a design similar to previously high-performant designs, such as Figure 4.9 **(c)**. Additionally, the chosen reactivities for ruthenium- and iron-based catalysts were chosen heuristically. As argued in more detailed in section 5, increas-

ing the overall reactivity could also increase differences in conversion rate. Still, even small increases in conversion rate could potentially make a significant economical difference in an industry such as ammonia synthesis.

Kinetic rates of reaction for the designs are shown in Figure 4.15. The column design is again less active for the middle catalyst sites. The RL agent designs are moderately even in site reactivities and feature an overall increase in reactivity.

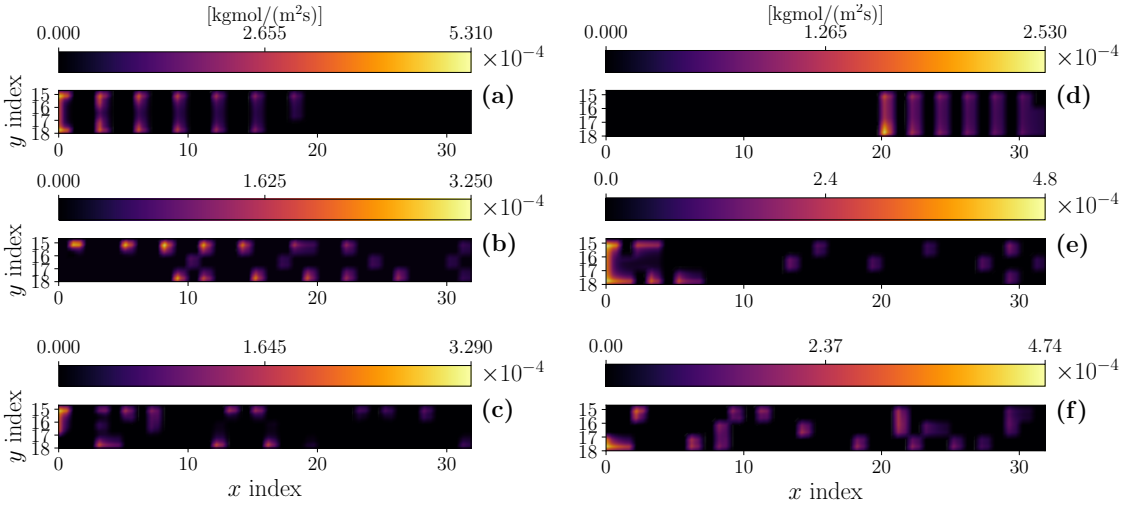


Figure 4.15: Kinetic rates of reaction **(a)**-**(c)** for ruthenium-like sites and **(d)**-**(f)** iron-like sites for the designs shown in Figure 4.14.

Additionally, molar fractions of reactant are shown in Figure 4.16.

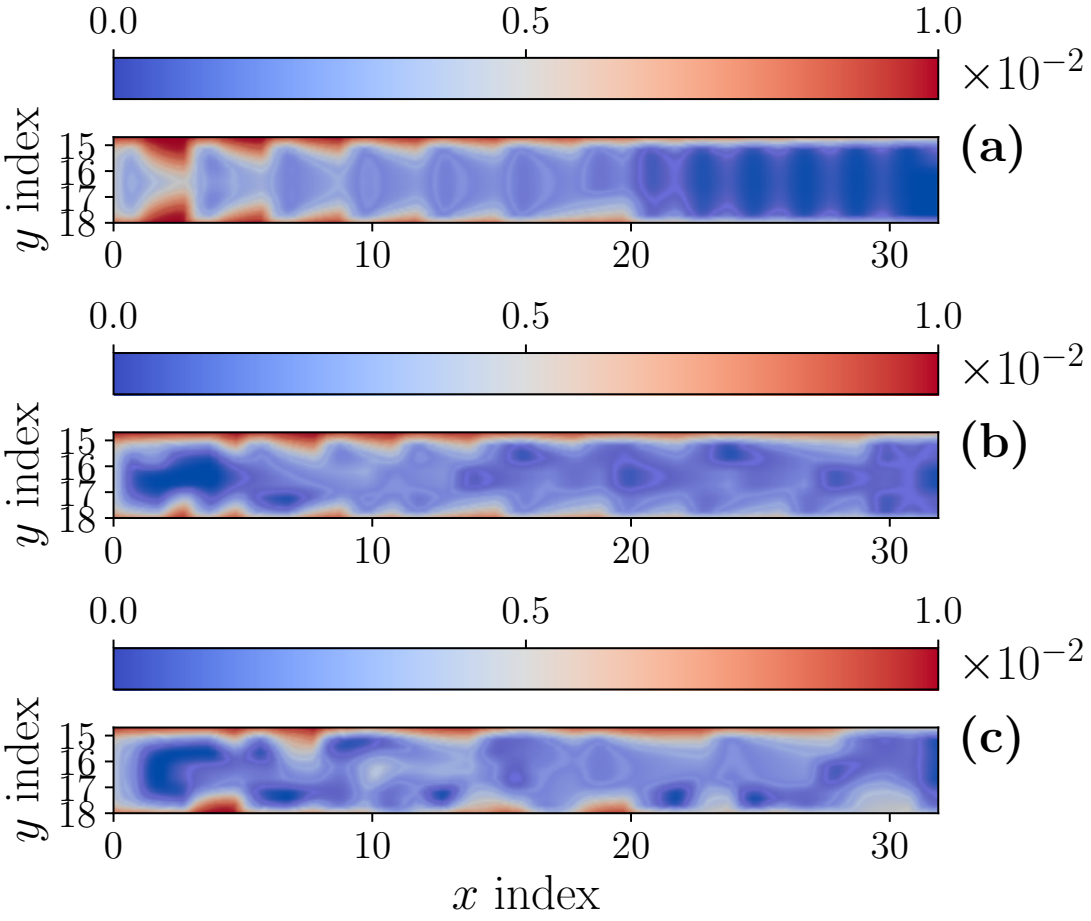


Figure 4.16: Molar fractions of reactant for the designs shown in Figure 4.14, respectively.

The molar fractions of reactant for the RL agent placements are overall more even than the manual placement. The manual placement suffers from a drought from index 20 and onward. The more even molar fraction of reactant may explain the better conversion rate in the RL-designed chips.

4.5 Designing geometries with Ant Fluid

The final test was to design channels and place active sites simultaneously using ant fluid. One test case was investigated, where an agent was given the task to place 20 ruthenium-like and 20 iron-like active sites.

4.5.1 Correlation between CFD and ACO rewards

To motivate the use of ant fluid over CFD simulations, it has to be sufficiently correlated with the CFD simulations. Otherwise, the optimization tasks may be too far apart and optimizing ant fluid rewards may not lead to an increase in CFD rewards. Figure 4.17 shows a comparison of ant fluid rewards with CFD rewards. During training, ant fluid was used. After training, CFD was used on checkpoints

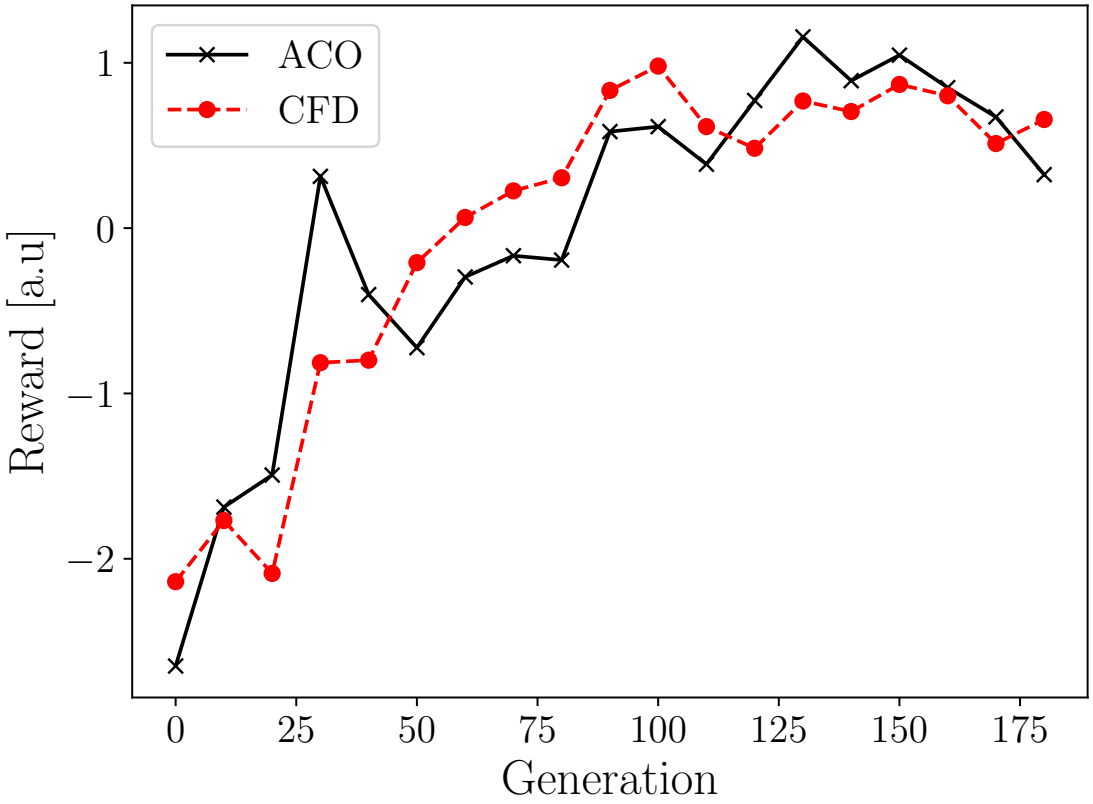


Figure 4.17: Comparison of rewards from the ant fluid reward system (ACO) and the CFD-based reward system (CFD). The rewards were standardized by subtracting their means and dividing by standard deviation. The Pearson correlation coefficient for the two reward series is 0.907, suggesting a strong positive correlation. Since the rewards from ant fluid correlate well with the rewards from the CFD-based reward system, the optimization tasks are similar and motivates the use of the much more computationally efficient ant fluid.

of particle placements to evaluate the quality of the placements. The Pearson correlation coefficient between the two series is 0.907, suggesting a quite strong positive correlation.

The heights extracted from ant fluid affect the CFD reward system by restricting the flow to pass via the channels. Roughly the same proportion of fluid should pass through the active sites as the proportion of ants, which may explain the correlation between the reward systems.

4.5.2 Resulting chip design

The agent was able to improve conversion rate by 39 % compared to after the first generation of training the agent. Here, one generation corresponds to sampling a batch of chips and running PPO on them. However, when comparing with a manual design where 20 ruthenium-like sites were placed by the inlet and 20 iron-like sites

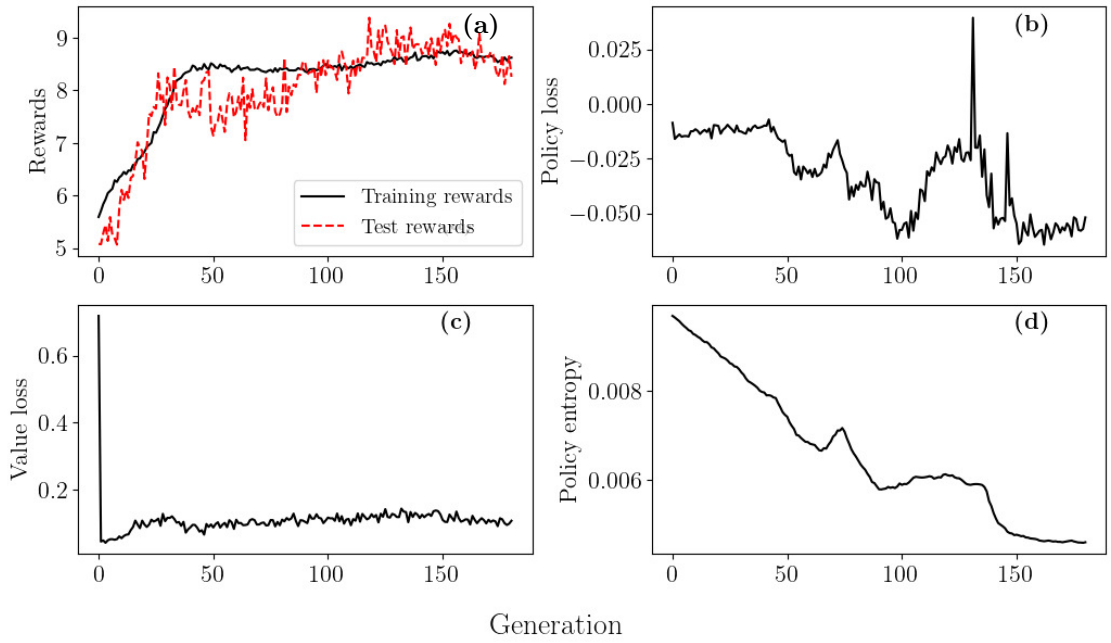


Figure 4.18: Statistics from training an agent with the ant fluid reward system with 20 ruthenium-like active sites and 20 iron-like active sites. **(a)** the average rewards from sampling during training and the test rewards from placing active sites on the highest policy values each step. **(b)** L_t^{CLIP} from equation (2.10) per generation. **(c)** the value network loss per generation. **(d)** the entropy of the network policy. The entropy decreases overall, signaling that the policy becomes more and more deterministic. The best chip placement from the training session increased reactant conversion rate by 39 % compared with the first generation of training.

were placed halfway to the outlet, the agent was found to give a conversion rate of 87.7 % of that of the manual design. Some similarities are prevalent in both designs, such as the tendency to put active sites in columns, but also to place ruthenium sites first, see Figure 4.19.

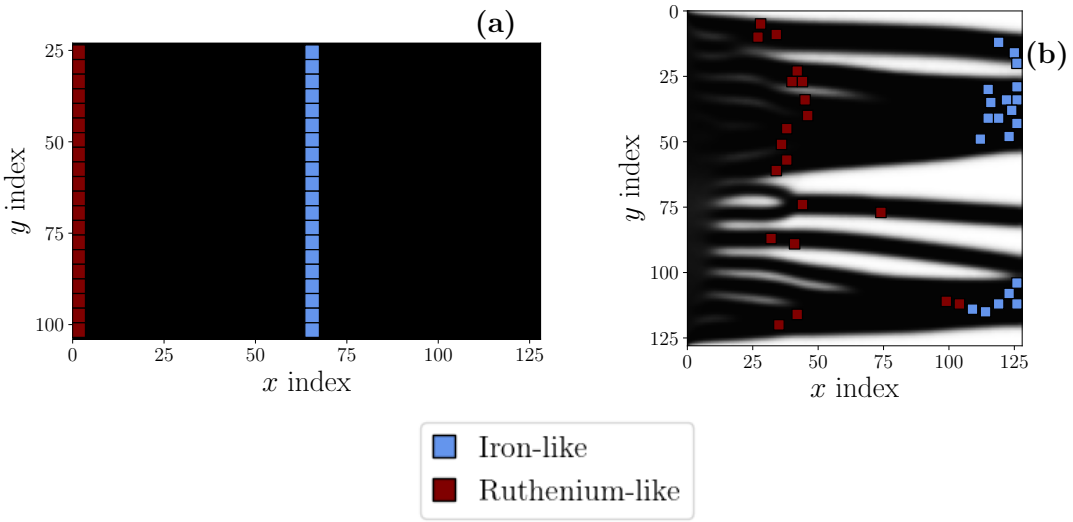


Figure 4.19: **(a)** manual design of a ruthenium-iron reactor and **(b)** a design where the geometry and active sites have been decided by the RL agent.

5
Discussion

In this chapter, results are interpreted and alternative approaches and further developments are discussed. Additionally, the practical applicability of the framework is discussed to put into perspective the challenges and limitations of using the framework for design of real nanofluidic chips.

5.1 Practical applicability

The RL agent has been shown to be capable of optimizing simulated nanofluidic chips in a variety of different situations. However, the model is not at its current state fully mature for creating real nanofluidic chips optimized for high conversion rate. Currently, it serves as a proof-of-concept to demonstrate that such a framework is feasible for complex fluid dynamical optimization. The practical applicability is mostly limited by lack of knowledge of physical quantities, as most quantities were merely heuristically estimated in the simulations.

When applying the framework on real nanofluidic chips, it is important to approximately know certain physical quantities of the chip, since the reward system must accurately be able to simulate the environment to be useful. Some important quantities are the Reynolds and Schmidt numbers, which affect the fluid dynamics. Additionally, catalyst reactivities in ANSYS Fluent should be motivated by experiments or theory.

5.2 CFD reward system

The reward system is crucial for the success of the reinforcement learning agent. If the CFD rewards do not accurately emulate the behavior of an actual nanofluidic chip, the agent will likely not be useful in practice. Future work includes building and testing the AI-designed nanofluidic chips to see if the conversion rates from the CFD solver are correct. It would be especially relevant to try variations of narrow chip designs and comparing with the RL agent chips, since the conversion rate appears to vary more dramatically in these settings. Additionally, wide chip canvases with many catalysts could also be interesting to compare with RL agent-designed chips, since these settings introduce more possible chip configurations and create greater droughts of reactant fluid.

While CFD likely is the currently most accurate way of simulating the nanofluidic chip functionality, it enforces a major bottleneck when training a reinforcement learning agent. For scalable training of RL agents, an efficient GPU implementation may be beneficial for parallel computation on the fine computational mesh. All CFD simulations were run on a single 6-core CPU, which is sub-optimal.

Reaction rates were kept quite low in the simulation tool. Higher reaction rates caused instabilities, long simulation times and poor estimates of conversion rate. As a result, about 4-10 % normally converted to product, which is moderately low. Optimizations may be greater if the reaction rate can be brought up, since for example reactant droughts would be even larger and the placements by the RL agents could have a more dramatic effect. For more realistic simulations of a nanofluidic chip environment, catalysts should also be given volumes, so that the catalysts themselves may alter the flow to a greater extent. Currently, a segment of the bottom plate of a simulated nanofluidic chip is activated if a catalyst is placed there, neglecting the volume of the catalyst and viewing it as a surface element.

5.3 Ant Fluid

The ant fluid reward system conveniently optimizes the number of ants reaching an active site, but it does not directly capture aspects such as pressure drop or decreasing reactant concentration in the wake of an active site. When only iron-type catalysts exist, there is a tendency of forming a single tunnel and placing particles in a row. Such a solution is expected by use of the reward system, but is typically not near-optimal. More useful solutions are typically acquired when both ruthenium-like and iron-like sites exist, which is suggested from the strong correlation with CFD rewards in Figure 4.17. A future development could be to make ant fluid more realistic and to take into account pressure drop or similar effects.

5.4 Other approaches for building obstacles

The framework can readily assign heights to obstacles due to an additional neural network with the same structure as the upsampling decoder in the policy network. However, since the computational mesh is discretized in height, the heights given as output from the network cannot vary continuously, posing problems during training. As such, only total volume blockages were tested with CFD rewards.

Other stochastic optimization algorithms such as genetic algorithms can also be used for producing obstacles with varying heights [32, chapter 3]. The canvas grid could then be viewed as a two-dimensional chromosome in a genetic algorithm, where each free cell is assigned a height after placing active sites. The genetic algorithm could then be applied as a post-processing step, where the best found chip is chosen and fine-tuned with obstacles. However, the described pipeline requires evaluating the expensive CFD objective function again several times.

5.5 Discretized fluid simulations

Using discretized particle simulations over CFD can in many situations be detrimental. With implementations such as k-d trees, the time complexity can be made $\mathcal{O}(n \log n)$, where n is the number of particles [42], [43]. In macroscopic situations, CFD may be a better choice since the amount of particles needed to accurately simulate fluid dynamics can make it infeasible to use discretized particle simulations. However, since the discrete nature is more evident on the nanoscale, and because the amount of particles involved in the flow is smaller, discretizing the simulations may bring benefits.

One benefit with discretized particle simulations could be that catalyst poisoning is more accurately replicated. The poisoning could then be reproduced by giving ammonia particles a probability of sticking to ruthenium catalysts and blocking other particles from sticking to them by taking up a part of the catalyst area. Because of the probabilistic nature, these simulation would need to be averaged over time in order to get accurate results. Depending on the number of particles used and the number of averaging steps, the simulation could potentially be more computationally efficient than CFD.

5.6 Alternative reinforcement learning schemes

Other reinforcement learning schemes than policy gradient have also been considered. One such approach would be a deep Q-network (DQN) in the setting of deep Q-learning. The problem with DQN in this context, however, is the explosively large action space.

One setting with DQN would be to form one new subgraph for each possible particle placement given a partially placed graph. The Q-network would then rate all possible graphs. The graph corresponding to the highest Q-value is then chosen as the next state. The task that the DQN is responsible for is then simply to recognize how a well-performing graph is structured. With an example of 20 particles of a single species and a 128×128 grid, the number of possible actions is approximately $(128 \cdot 128)^{20} \approx 2 \cdot 10^{84}$.

While deep Q-learning allows the use of a replay buffer, the buffer cannot nearly be in the same order of magnitude as the number of states due to memory limitations. The learning task itself may potentially be easier. A policy network outputs 128×128 individual numbers, whereas DQN would only need to output one number per graph candidate. Policy gradient also does not allow a universal replay buffer, since the policy continuously gets updated. The sampling efficiency may thus be considerably lower for policy gradient, but it does not suffer from the same action space difficulties that deep Q-learning does. Additionally, the implemented PPO method appears capable of learning near-optimal solutions for a variety of different graph problems.

6
Conclusion

An environment based on deep reinforcement learning has been developed for optimizing the design of fluidic reactor systems. Additionally, a minor investigation of what makes a high-quality chip design was conducted, in order to interpret acquired results. Findings point towards active catalyst sites placed at areas of high static and dynamic pressure with well-mixed reactant solution having the highest reactivity. Agents have been shown to successfully optimize fluid dynamical situations corresponding to nanofluidic chips, often being able to balance high pressure with high reactant concentration.

Agents were tested in a few feasible scenarios to showcase some use cases of the framework, such as optimized catalyst placements in wide or narrow chip canvases. Both scenarios are relevant since nanofluidic chips may take different shapes, causing different optimization tasks. The agent was the most successful in the narrow chip scenario, where catalysts have to be more closely packed. Nanofluidic chips optimized for high conversion rate could for example be used for increased synthesis of ammonia or other desirable chemicals. While optimizing the conversion rate, the framework could also remove part of the manual labor required to design nanofluidic chips. Since chips are designed through trial-and-error, the framework may also reduce material loss by instead simulating the chip environment. The next step is to build actual nanofluidic chips suggested by the RL environment. If successful, the environment can be used by Langhammer Lab for future design of nanofluidic chips for better conversion rate or other desirable properties.

The framework itself is not specific to nanofluidic chip design and can be adapted to other problems simply by changing the objective function. A few examples were tested in order to showcase the generality of the framework, such as minimizing or maximizing distance, or forming clusters of components. Clearly, these optimization tasks are toy systems, but are widely different from the CFD-based reward system. As the framework was designed to be generally applicable, it could hypothetically be used for optimizing several other LOC systems, or even completely different scenarios where different components need to placed in relation to each other to optimize some quantity. As LOCs are increasingly used, the framework may find increasingly more use cases.

Additionally, the ant fluid reward system was developed, where policy gradient and ant colony optimization work in tandem. The reward system is considerably faster during training compared with CFD simulations, but requires some development in

order to be useful for chip design. While this reward system likely would not find uses in its current state, it is a first step for a computationally lighter reward system and a novel application of ACO for CFD-based optimization tasks.

Bibliography

[1] Y. C. Lim, A. Z. Kouzani, and W. Duan, "Lab-on-a-chip: A component view", *Microsystem Technologies*, vol. 16, no. 12, pp. 1995–2015, 2010.

[2] H. Craighead, "Future lab-on-a-chip technologies for interrogating individual molecules", *Nanoscience and Technology: A Collection of Reviews from Nature Journals*, pp. 330–336, 2010.

[3] D. Figeys and D. Pinto, *Lab-on-a-chip: A revolution in biological and medical sciences*. 2000.

[4] S. Levin, J. Fritzsche, S. Nilsson, A. Runemark, B. Dhokale, H. Ström, H. Sundén, C. Langhammer, and F. Westerlund, "A nanofluidic device for parallel single nanoparticle catalysis in solution", *Nature communications*, vol. 10, no. 1, pp. 1–8, 2019.

[5] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, *Chip placement with deep reinforcement learning*, 2020. DOI: 10.48550/ARXIV.2004.10746. [Online]. Available: https://arxiv.org/abs/2004.10746.

[6] R. Roy, J. Raiman, N. Kant, I. Elkin, R. Kirby, M. Siu, S. Oberman, S. Godil, and B. Catanzaro, "Prefixrl: Optimization of parallel prefix circuits using deep reinforcement learning", in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 853–858. DOI: 10.1109/DAC18074.2021.9586094.

[7] H. Daiguji, "5.08 - nanofluidics", in *Comprehensive Nanoscience and Nanotechnology (Second Edition)*, D. L. Andrews, R. H. Lipson, and T. Nann, Eds., Second Edition, Oxford: Academic Press, 2011, pp. 207–228, ISBN: 978-0-12-812296-9. DOI: https://doi.org/10.1016/B978-0-12-812295-2.00132-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012812295200132X.

[8] W. Sparreboom, A. van den Berg, and J. C. Eijkel, "Principles and applications of nanofluidic transport", *Nature nanotechnology*, vol. 4, no. 11, pp. 713–720, 2009.

[9] J. M. Modak, "Haber process for ammonia synthesis", *Resonance*, vol. 7, no. 9, pp. 69–77, 2002.

[10] J. Kim, H. J. Kim, and S. Chang, "Synthetic uses of ammonia in transition-metal catalysis", *European Journal of Organic Chemistry*, vol. 2013, no. 16, pp. 3201–3213, 2013.

[11] S. Giddey, S. Badwal, C. Munnings, and M. Dolan, "Ammonia as a renewable energy transportation media", *ACS Sustainable Chemistry & Engineering*, vol. 5, no. 11, pp. 10 231–10 239, 2017.

[12] V. Smil, *Enriching the earth: Fritz Haber, Carl Bosch, and the transformation of world food production*. MIT press, 2004.

[13] Z. You, K. Inazu, K.-i. Aika, and T. Baba, "Electronic and structural promotion of barium hexaaluminate as a ruthenium catalyst support for ammonia synthesis", *Journal of Catalysis*, vol. 251, no. 2, pp. 321–331, 2007, ISSN: 0021-9517. DOI: https://doi.org/10.1016/j.jcat.2007.08.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021951707003168.

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation", in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12, MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[15] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms", in *Proceedings of the 31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, 22–24 Jun 2014, pp. 387–395. [Online]. Available: https://proceedings.mlr.press/v32/silver14.html.

[16] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration", in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2829–2838. [Online]. Available: https://proceedings.mlr.press/v48/gu16.html.

[17] J. Peters and S. Schaal, "Natural actor-critic", *Neurocomputing*, vol. 71, no. 7, pp. 1180–1190, 2008, Progress in Modeling, Theory, and Application of Computational Intelligenc, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2007.11.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231208000532.

[18] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients", *Logic Journal of the IGPL*, vol. 18, no. 5, pp. 620–634, 2010.

[19] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, *Trust region policy optimization*, 2015. DOI: 10.48550/ARXIV.1502.05477. [Online]. Available: https://arxiv.org/abs/1502.05477.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. DOI: 10.48550/ARXIV.1707.06347. [Online]. Available: https://arxiv.org/abs/1707.06347.

[21] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data", *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017. DOI: 10.1109/MSP.2017.2693418.

[22] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *Graph neural networks: A review of methods and applications*, 2018. DOI: 10.48550/ARXIV.1812.08434. [Online]. Available: https://arxiv.org/abs/1812.08434.

[23] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments", *Journal of Molecular Biology*, vol. 330, no. 4, pp. 771–783, 2003, ISSN: 0022-2836. DOI: https://doi.org/10.1016/S0022-2836(03)00628-4. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022283603006284.

[24] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, "Text level graph neural network for text classification", *arXiv preprint arXiv:1910.02356*, 2019.

[25] Q. Cappart, D. Chételat, E. Khalil, A. Lodi, C. Morris, and P. Veličković, *Combinatorial optimization and reasoning with graph neural networks*, 2021. DOI: 10.48550/ARXIV.2102.09544. [Online]. Available: https://arxiv.org/abs/2102.09544.

[26] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks", *CoRR*, vol. abs/1810.02244, 2018. arXiv: 1810.02244. [Online]. Available: http://arxiv.org/abs/1810.02244.

[27] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric", in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, 2017. DOI: 10.48550/ARXIV.1710.10903. [Online]. Available: https://arxiv.org/abs/1710.10903.

[29] T. Cai, S. Luo, K. Xu, D. He, T.-Y. Liu, and L. Wang, *Graphnorm: A principled approach to accelerating graph neural network training*, 2020. DOI: 10.48550/ARXIV.2009.03294. [Online]. Available: https://arxiv.org/abs/2009.03294.

[30] J. Tumlinson, J. Moser, R. Silverstein, R. Brownlee, and J. Ruth, "A volatile trail pheromone of the leaf-cutting ant, atta texana", *Journal of Insect Physiology*, vol. 18, no. 5, pp. 809–814, 1972, ISSN: 0022-1910. DOI: https://doi.org/10.1016/0022-1910(72)90018-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022191072900182.

[31] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances", *Handbook of metaheuristics*, pp. 311–351, 2019.

[32] M. Wahde, *Biologically inspired optimization methods: an introduction*. WIT press, 2008.

[33] G. Wang, F. Yang, and W. Zhao, "There can be turbulence in microfluidics at low reynolds number", *Lab on a Chip*, vol. 14, no. 8, pp. 1452–1458, 2014.

[34] E. Michaelides, "Nanofluidics", *Cham: Springer International Publishing*, 2014.

[35] R. D. Sochol, K. Iwai, J. Lei, D. Lingam, L. P. Lee, and L. Lin, "A single-microbead-based microfluidic diode for ultra-low reynolds number applications", in *2012 IEEE 25th International conference on micro electro mechanical systems (MEMS)*, IEEE, 2012, pp. 160–163.

[36] R. Fishler, M. K. Mulligan, and J. Sznitman, "Mapping low-reynolds-number microcavity flows using microfluidic screening devices", *Microfluidics and nanofluidics*, vol. 15, no. 4, pp. 491–500, 2013.

[37] D. Camuffo, "Chapter 8 - dry deposition of airborne particulate matter: Mechanisms and effects", in *Microclimate for Cultural Heritage (Second Edition)*, D. Camuffo, Ed., Second Edition, Boston: Elsevier, 2014, pp. 283–346, ISBN: 978-0-444-63296-8. DOI: https://doi.org/10.1016/B978-0-444-63296-8.00009-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444632968000093.

[38] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts", *Distill*, 2016. DOI: 10.23915/distill.00003. [Online]. Available: http://distill.pub/2016/deconv-checkerboard.

[39] C. Dong, C. C. Loy, K. He, and X. Tang, *Image super-resolution using deep convolutional networks*, 2015. DOI: 10.48550/ARXIV.1501.00092. [Online]. Available: https://arxiv.org/abs/1501.00092.

[40] ANSYS, *Fluent - CFD Software*, version R1, 2022. [Online]. Available: http://www.ansys.com/products/fluids/ansys-fluent.

[41] J. Fritzsche, D. Albinsson, M. Fritzsche, T. J. Antosiewicz, F. Westerlund, and C. Langhammer, "Single particle nanoplasmonic sensing in individual nanofluidic channels", *Nano letters*, vol. 16, no. 12, pp. 7857–7864, 2016.

[42] D.-J. Kim, L. J. Guibas, and S.-Y. Shin, "Fast collision detection among multiple moving spheres", *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 3, pp. 230–242, 1998.

[43] J. Schauer and A. Nüchter, "Collision detection between point clouds using an efficient kd tree implementation", *Advanced Engineering Informatics*, vol. 29, no. 3, pp. 440–458, 2015.

[44] M. Ulmestrand, *Algorithmically perfect colormaps*, Jul. 2022. [Online]. Available: https://github.com/m-ulmestrand/perfect-cmaps.

A

Appendix 1

A.1 Custom colormap

A custom colormap has been used in Figures 3.4, 3.5 and 3.5. The colormap was designed to be diverging, yet sequential, while being algorithmically perfectly linear in intensity and nearly perfectly linear in luminance. Luminance means the perceived intensity, where the red, green and blue channels are given different weights due to the human eye having different sensitivity to each channel. The intensities for each channel, total intensity and luminance are shown in Figure A.1. To demonstrate the sequential structure of the colormap, Figure A.2 shows the colors given by the colormap for inputs in the range 0 to 1 along with its grayscale-converted counterpart. The colormap is nicknamed *cold blooded* because of the transition from blood red to icy blue, see Ulmestrand [44].

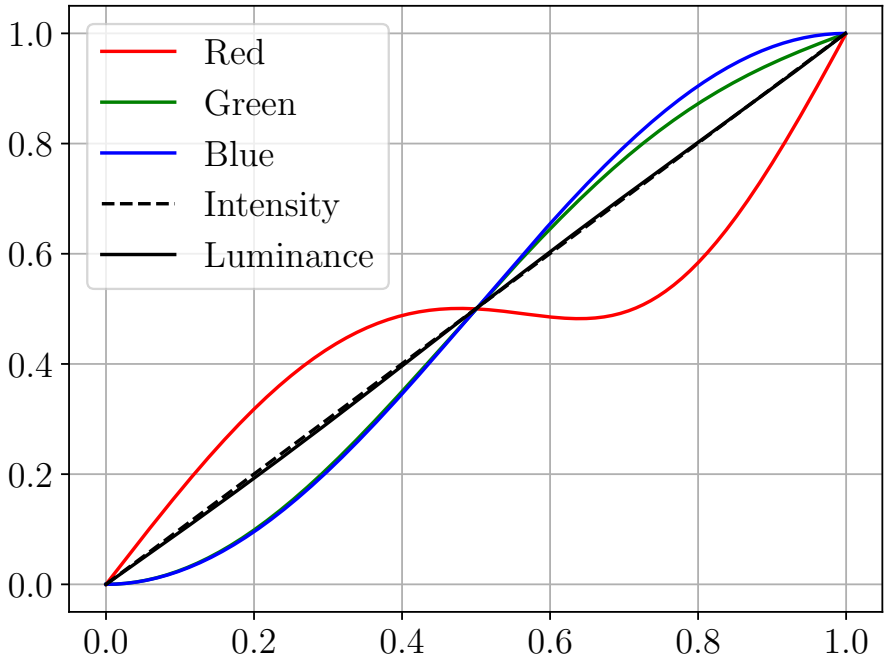


Figure A.1: Intensity profiles for red, green and blue channels along with total intensity and luminance for the "cold blooded" colormap.

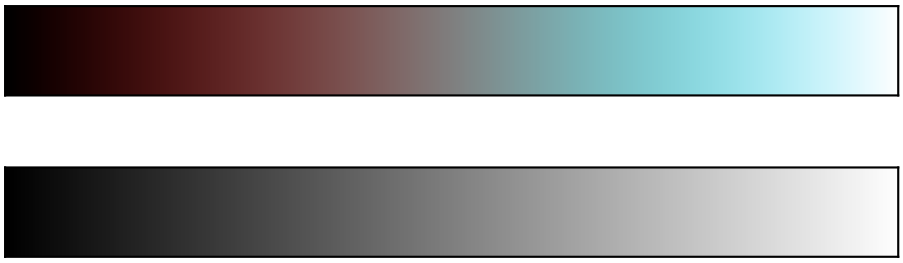


Figure A.2: The profile from values 0 (left corner) to 1 (right corner) for the "cold blooded" colormap along with its grayscale counterpart. Both versions are perceptually sequential and no visible artifacts are present.

B

Appendix 2

The most frequently used neural network and PPO parameters are given in Table B.1.

Table B.1: Most frequently used neural network and PPO parameters. Some deviations from below setting occur in the methodology chapter, but all of the main results were acquired with these settings.

Neural network parameters		
Parameter	**32 × 32**	**128 × 128**
Graph encoder layers	1	1
Graph encoder hidden neurons	512	512
Graph encoder output neurons	128	512
Attention layers	3	3
Attention layers heads	8	8
Upsampling layers	3	5

Proximal policy optimization parameters	
Parameter	**Value**
Policy network learning rate	$2.00 \cdot 10^{-5}$
Value network learning rate	$1.00 \cdot 10^{-4}$
Value loss factor c_1	$0.75 \cdot 10^{0}$
Entropy factor c_2	$1.00 \cdot 10^{-3}$
Clip value ϵ	$2.00 \cdot 10^{-1}$

Additionally, the used hardware specification is given in Table B.2.

Table B.2: Used hardware specification for training the RL agent and running various simulations.

Hardware specification	
Hardware	**Specification**
GPU	NVidia GeForce GTX 1080
CPU	Intel(R) Core(TM) i7-8700K CPU @ 3.70 GHz
RAM	16.0 GB

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY