Model-Based Reinforcement Learning for Cavity Filter Tuning

Doumitrou Daniil Nimara DDNIMARA@GMAIL.COM
Ericsson GAIA, Sweden

Mohammadreza Malek-Mohammadi MOHAMMA@KTH.SE
Qualcomm, Sweden *

Jieqiang Wei JIEQIANG.WEI@ERICSSON.COM
Ericsson GAIA, Sweden

Vincent Huang VINCENT.A.HUANG@ERICSSON.COM
Ericsson GAIA, Sweden

Petter Ogren PETTER@KTH.SE
Division of Robotics, Perception and Learning, KTH

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

The ongoing development of telecommunication systems like 5G has led to an increase in demand of well calibrated base transceiver station (BTS) components. A pivotal component of every BTS is cavity filters, which provide a sharp frequency characteristic to select a particular band of interest and reject the rest. Unfortunately, their characteristics in combination with manufacturing tolerances make them difficult for mass production and often lead to costly manual post-production fine tuning. To address this, numerous approaches have been proposed to automate the tuning process. One particularly promising one, that has emerged in the past few years, is to use model free reinforcement learning (MFRL); however, the agents are not sample efficient. This poses a serious bottleneck, as utilising complex simulators or training with real filters is prohibitively time demanding. This work advocates for the usage of model based reinforcement learning (MBRL) and showcases how its utilisation can significantly decrease sample complexity, while maintaining similar levels of success rate. More specifically, we propose an improvement over a state-of-the-art (SoTA) MBRL algorithm, namely the Dreamer algorithm. This improvement can serve as a template for applications in other similar, high-dimensional non-image data problems. We carry experiments on two complex filter types, and show that our novel modification on the Dreamer architecture reduces sample complexity by a factor of 4 and 10, respectively. Our findings pioneer the usage of MBRL which paves the way for utilising more precise and accurate simulators which was previously prohibitively time demanding.

Keywords: Reinforcement Learning, Model Based Reinforcement Learning, Telecommunication

1. Introduction

Automated microwave filter tuning via artificial intelligence (AI) is an established scientific field, which was pioneered in the middle of the past century [Dishal (1951)]. At the core of telecommunication is the manipulation of information which inhibit certain frequencies. However, to process such frequencies, telecommunication devices employ filters which allow desirable frequency bands

* The work of second author was done while he was with Ericsson.

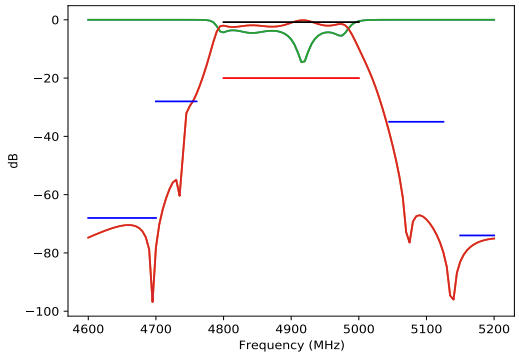


Figure 1: Frequency response of a detuned filter. The x and y axis correspond to the frequency (in MHz) and S-Parameter (in dB) respectively. The plotted lines correspond to $20 \log |S_{ij}|$; see Section 2 for definition of S_{ij}. Red: S_{21}, green: S_{11}. A filter is tuned when S_{11} lies below the red bar in the passband (center) and S_{21} lies below the blue bars in the stopband (non-center) and above the black bar in the pass band (center). Our goal is to devise an agent which can reach such a configuration.

.

while simultaneously rejecting undesired ones (noise and interference) which are present in the spectrum. In the precipice of 5G, (microwave) radio frequency (RF) filters are of particular importance as they can operate on medium to high frequency ranges. From the vast array of different RF filters, cavity filters (CF) are widely employed for their high Q-factor, steep skirt selectivity and ability to handle high power signals

CF are vital and numerous instances are used in the base stations. Their high frequency selectiveness, however, comes at a cost, as they require an extremely accurate manufacturing process which is unattainable at a reasonable cost. Thus, a post-production tuning step is required, in which one alters the filter's tunable characteristics by interacting with screws that lie on top of the filter. However, the tuning process is both costly and time consuming, as highly-trained personnel must be employed to manually tune CFs in a lengthy process (which can take up to 30 minutes per filter). Therefore, there is an ever increasing interest in automating this process, as illustrated in Figure 1.

We narrow our attention to a particularly promising solution for automating tuning, which leverages Reinforcement Learning (RL). The training of the RL agent is done once, by interacting with i.i.d. detuned instances of filters of a particular type. Following this, no further training is required and the agent can be deployed for tuning any filter of that type. We examine and propose relevant modifications to a particularly efficient state-of-the-art (SoTA) model based RL (MBRL) algorithm named Dreamer [Hafner et al. (2019, 2020)]. We present our novel modification on the Dreamer and compare it with a well established model free RL (MFRL) algorithm, SAC [Haarnoja et al. (2018)]. Broadly speaking, MBRL is expected to exhibit good sample efficiency at the cost of asymptotic performance [Mordatch and Hamrick (2020)]. This, along many other MBRL characteristics was studied by Wang et al. (2019). However, such comparisons are made in typical gym environments [openAI (2021)] exhibiting low-dimensional observations or images. Our work explores this with high-dimensional, non-image observations.

The main contribution of this paper can be summarised as follows; our work:

- provides a novel extension of the Dreamer architecture which outperforms its predecessor for CF tuning. Our modifications allow the Dreamer's observation model to become progressively more robust, in a modular way which can be easily extended for similar tasks,
- compares and showcases how MBRL reduces sample complexity by a factor of 4 and 10 on two different type of filters, and
- pioneers the usage of MBRL for the well established field of automated CF tuning in a way that allows the tackling of more complex filters

1.1. Related Work

As mentioned earlier, automated CF tuning is a well-established field, spanning numerous diverse approaches. Broadly speaking, we can categorise them in optimisation based or RL based.

Perhaps the most intuitive optimisation approach is proposed in Harscher et al. (2001) which decomposes the problem into two steps. In the first stage, they try to infer the underlying filter parameters based solely on observed S-parameters. They do so by finding the model parameters θ^{model} which showcases a similar frequency response. Next, one tries to map screw adjustements to changes in θ^{model} via sensitivity analysis. Finally, they compare θ^{model} with θ^{ideal} and exert the necessary screw adjustments (based on their prior sensitivity analysis), such that $\theta^{model} \rightarrow \theta^{ideal}$.

A different approach was employed by Wang (2015), which advocated for the usage of MFRL. They examined a simplified scenario, in which only two out of the seven screws were interactable by the agent. Furthermore, only one screw could turn at a time. A binary $r \in \{0, 1\}$ reward was used, rewarding the agent with 1 if the action resulted in a more tuned configuration. Actions were discretized and DQN [Mnih et al. (2015)] was employed. Their attempts proved fruitful in tackling the so called 4p0z filter[1].

In a later work, Wang et al. (2018) reexamined the problem, this time, however, in the continuous domain, by leveraging DDPG [Lillicrap et al. (2016)] and allowing the tuning of more than one screw at a time. This resulted in a decrease in sample complexity. Larsson (2018) adopted a similar approach to tackle 4p0z and 4p1z.

A more intricate scenario was examined by Lindståh and Lan (2020), which tackled a 13 screw filter (6p2z) by developing the DIRT algorithm. Their approach first leverages imitation learning to initialize the agent with guidance from expert trajectories $\mathcal{D}_{expert} = \{(s_i, \pi_{expert}(s_i)\}$. Next, DDPG is employed, followed by DQN for screw selection. More precisely, DQN receives as an input the proposed screw rotations from DDPG and outputs m Q values (one for each screw). The agent may then turn the screw showcasing the largest Q value. This approach not only tackled more complicated filters, but also presented a screw selection method.

This study differs from the one presented above in that MBRL is applied to the filter tuning problem. To the best of our knowledge, our work pioneers the usage of MBRL in this field.

2. Problem Formulation

In this section, we will first describe key components of our CF environment which are present in the simulations. We will next formulate this environment as a Markov Decision Process so that we can treat it within the RL setting.

1. PpZz denotes a filter with P poles and Z zeroes. Generally, filter complexity increases with P and Z.

CFs consist of several resonators, each of which encapsulated within a conducting box (cavity). The location of the passband along the frequency spectrum dictates its gating capabilities and depends on the overall resonance frequency f_r. In turn, f_r depends on the geometry of the cavities, which, if shifted, will result in a change in f_r and hence in the passband. For production, filters are often designed using 3D simulators; however, sometimes, it is easier to approximate their functionalities by adopting a simpler (less computationally intensive) circuit model. This model is comprised of capacitors and inductors (each subscribing to a tunable resonance frequency).

The number of cavities and the topology of the filter characterise the filter's complexity [Richard J. Cameron (2007)]. We describe the different filters by typical industry convention, writing PpZz when referring to a filter with P number of poles and Z number of zeros (e.g., 6p2z refers to a 6-pole, 2-zero filter).

The frequency response of the filter is characterised by its Scattering Parameters (S-parameters). S-parameters are complex values and a function of frequency f, given by:

$$S_{ij} = \frac{I_{i,out}(f)}{I_{j,in}(f)}, \; |S_{ij}| = \frac{P_{i,out}(f)}{P_{j,in}(f)} \tag{1}$$

where $I_{i,out}$, $I_{j,in}$ describe the Laplace transform of the outward and inward current at port i and j respectively, while P describes the electrical power. The circuit model employed herein is a 2-port circuit, which implies $i, j \in \{1, 2\}$. A particularly useful tool employed by human experts when tuning CFs is Vector Network analysers (VNA) which plot $20 \log_{10} |S_{ij}|$ (dB scale) over a range of frequency values (see Figure 1). Overall, four curves may be observed (S_{11}, S_{12}, S_{21} and S_{22}). However, for cavity filters, these relations hold:

$$S_{11} = S_{22}, \; S_{21} = S_{12}^{*} \tag{2}$$

We formulate this environment as an MDP:

- **Environment**: We utilise a circuit simulator developed at Ericsson, which we extended to provide feedback and interactability to the agent. Two filters are examined, namely 6p2z and 8p4z, which vary in topology, number of screws and cavity resonators and overall complexity.
- **Action Space** (continuous): An agent's actions define the way in which it can interact with the filter. The filter's tunable parameters can be affected via tuning screws located on top of the filter. As such, the action space is $[-1, 1]^m$, where m denotes the number of screws. For example, executing $\Delta x_i = 1$ results in turning the i-th screw by a max angle, which we set to 1080 degrees.
- **State Space** (continuous): The state or observation space needs to describe what is visible to the agent. It is only natural, then, to define the state space as that which contains all possible S-Curve configurations. The agent does not have access to the screw's absolute position. Instead, its observation consists of N 4-tuples $\{S_{i,j} | i, j \in \{1, 2\}\}_{n=1}^{N}$. We generate those 4-tuples by discretizing $[f_{min}, f_{max}]$ into N frequency points and examining the filter's behaviour in each one of these points. As the S-parameters are complex, we further separate the imaginary and real part, leading to a state space $S \subset \mathbf{R}^{8N}$.
- **Reward**: The reward function must be such that it favours configurations of tuned filters. As such, we examine the VNA frequency curves (recall Figure 1) on the aforementioned N

frequency points and check whether they satisfy the requirements or not. More specifically, we sum up the $-\ell_2$ point-to-threshold distance (points that do not violate the threshold do not negatively impact the reward) as reward. If all the requirements are met, we reward the agent with $r_{extra} = 100$.

- **Transition probability** $\mathcal{T}$: Measurement noise (noisy S parameters) in conjunction with non-tunable parameters (parameters which are unaffected by screw rotations, e.g. Q-factor) means that actions a_t performed on a particular state s_t can lead to a different next state $s_{t+1} \sim \mathcal{T}(.|s_t, a_t)$.
- **Goal**: Our goal is to devise a policy π_θ which optimises $J(\theta) = \mathbb{E}_{s_0}[\sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)|a_t \sim \pi_\theta(s_t)]$, where γ denotes the discount factor, $\mathbb{E}_{s_0}$ is the expectation over transitions and initial states s_0.

3. Model

In this section, we will present the Dreamer Algorithm as well as the modifications we suggest to address the problem defined above. Since the original implementation was not tested in a similar environment (high dimensional, non image observations), we also propose modifications which adapt it to this particular setting.

The Dreamer algorithm builds upon the MFRL baseline algorithm named Soft Actor Critic (SAC). SAC augments the traditional RL reward, by adding an entropy term which acts as a exploration regulariser, like so:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{s_{t+1} \sim \mathcal{T}(.|s_t, a_t)} \Big[\sum_t (r(s_t, a_t) + \beta \mathcal{H}(\pi(s_t)))|a_t \sim \pi(s_t) \Big] \tag{3}$$

where $\mathcal{H}(\pi) = \mathbb{E}_a[-\log \pi]$ is the entropy of the distribution of our policy. Recall that the entropy measures the randomness of a distribution. For instance, the uniform distribution exhibits the highest entropy. In essence, the policy is such that it not only maximises the accumulated immediate reward $r(s_t, a_t)$, but is also exploratory enough (when needed) to exhibit high entropy. The trade off between exploratory and exploitative policies is being handled by β.

The Dreamer owes its name to the fact that the unfolding of the world model happens in a lower dimensional latent space. This is particularly helpful when dealing with large dimensional environments (as is the case in our setting), where compounding errors can be devastating when utilising the world model for long horizons. Like in the original paper, training consists of three phases which loop until convergence: (i) learning world dynamics, (ii) training Actor Critic on latent space and (iii) generating new observations. During stage (i), an encoder decoder structure is utilised to embed our observations o_t into latent states s_t. During this phase, the encoder $q(s_t|o_t)$, decoder $q(o_t|s_t)$ (also called observation model), reward network $q(r_t|s_t)$, a transition model $q(s_t|s_{t-1}, a_{t-1})$ and representation model $q(s_t|s_{t-1}, a_{t-1}, o_t)$ are trained. During stage (ii), the policy and critic networks are trained. Stage (iii) populates the replay buffer with more experiences for training. The function optimised by the world model is the one presented in the original paper by Hafner et al.

(2019):

$$J_{REC} = \mathbb{E}_p\Big[\sum_t (\log q(o_t|s_t) + \log q(r_t|s_t) - \beta \mathrm{KL}\big(q(s_t|s_{t-1}, a_{t-1}, o_t)||q(s_t|a_{t-1}, s_{t-1})\big)\Big]. \tag{4}$$

Notice how maximising J_{REC}, means better reconstructions ($\log q(o_t|s_t)$) and more accurate reward estimates ($\log q(r_t|s_t)$), while simultaneously decreasing the distance (KL divergence) between the prior $q(s_t|a_{t-1}, s_{t-1})$ and posterior $q(s_t|s_{t-1}, a_{t-1}, o_t)$.

Changes had to be made on hyperparameters to address the high dimensional aspect of our setting. Furthermore, unlike the original work, the terminal network, which predicted early episode termination was not utilised. In their work, terminal states occur early and are the result of a failed agent. In our environment, terminal states are "goals" and occur only with good agents. Adopting such a network is not beneficial and destabilises training, as it only starts getting terminal examples deep within the training cycle (once the agent becomes better).

We also applied some more substantial changes on the Dreamer. First, we swapped convolutional layers with Fully Connected ones. Next, our experiments found that certain changes had to be made to the decoder $q(o_t|s_t)$ network. The decoder is used to generate S-curve reconstruction based on the latent space representation s_t. In the original work, $q(o_t|s_t)$ modelled a generative $N(\mu(s_t), I)$. Hence, during training, only the mean was learnable by the network. We propose that it instead also learns the diagonal covariance matrix $\Sigma = \Sigma_\theta(s_t)$, to allow for more precise and more confident reconstructions.

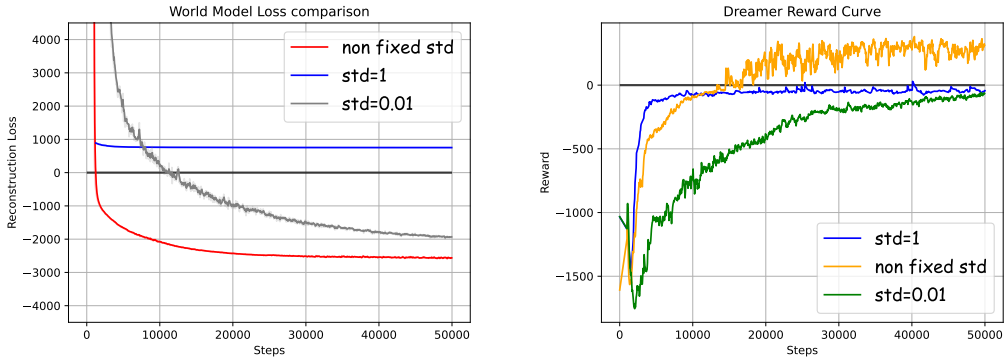
3.1. Dreamer Modification Significance

What proved pivotal in making MBRL a viable and efficient alternative in CF Tuning was proposing the modification in the decoder section of the Dreamer.

As stated previously, the authors of the Dreamer utilised the decoder $q(o_t|s_t)$ subnetwork to model $N(\mu(s_t), I)$. This fixed level of confidence proved too large for our reconstructed observations where each component lies in $[-1, 1]$. Naturally, we tried modelling it as $N(\mu(s_t), \sigma^2 I)$, for different values of $\sigma^2 < 1$, but found that asking for high precision from an initially suboptimal agent destabilised training, due to high gradient magnitudes.

Instead, we propose a simple, modular modification to help address this issue. We alter the final layer of the decoder to also allow the learning of the diagonal elements of the Covariance matrix. In other words, the decoder models $N(\mu(s_t), \Sigma(s_t))$. Intuitively, this approach allows the agent to be less precise initially, before increasing it as it becomes more proficient. In practice, this led to improved training metrics, as seen in Figure 2a. This approach is modular, as the notion of learning all the parameters of a distribution for your decoder can be extended for other distributions described by similar statistics.

The training metrics lead to actual improved performance, as is evident in Figure 2b. First, we find that the instability induced in training from using $\sigma = 0.01$ leads to slower training. The simpler task of $\sigma = 1$ suffers from the bottleneck shown in Figure 2a and leads to plateauing at a suboptimal performance. Our proposed modification is significantly more efficient, reaching tuned configurations (positive reward) after only 16k steps. Figure 3 showcases a typical example of the tuning process.



(*a*) (*b*)

Figure 2: (a) Reconstruction loss (negative log likelihood) of observations. Using a fixed $\sigma = 1$ leads to fast convergence (since the task is simpler), bottlenecking performance. Using a smaller fixed $\sigma = 0.01$ is a more difficult task and training commences slower. Notice, also, how the curve is not as smooth. Allowing for learnable, non fixed standard deviation combines the best of both approaches. (b) Reward curve comparison before and after modification.

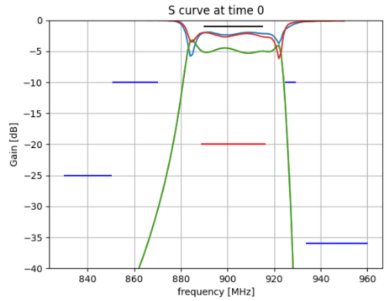
4. Experiments

All experiments leveraged a circuit model simulator developed at Ericsson. Two different type of filters were simulated: 6p2z and 8p4z. The first one serves as a proof of concept, as it is a non trivial, yet substantially simpler filter than 8p4z. The latter is a real filter, currently deployed in many BTS. As training is faster on 6p2z, all hyperparameter and MBRL comparison experiments were conducted using it. More specifically:

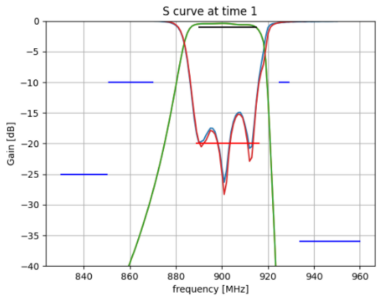
- 6p2z: The examined frequency range is [850, 950] MHz which is sampled with a step of 1, leading to a 808 dimensional observation space with 13 tunable screws.
- 8p4z: The examined frequency range is [4600, 5200] MHz which is sampled with a step of 5, leading to a 968 dimensional observation space with 19 tunable screws.

The resonance frequencies f_i's and mutual coupling between resonators, BW_{ij}, describe the filter's frequency response. These parameters are tunable, as the agent is able to alter them by interacting with the filter screws. A tuned filter exhibits an ideal configuration for these parameters. We model different instances of detuned filters, by applying a uniform pertubation around each such nominal value (e.g. ± 50 MHz). The filter is also described by other sets of parameters which are not affected by the screws and are thus called non-tunable (e.g. Q factor). These model different filter variants within the filter type. All parameters take a new value at the beginning of each episode. On top of this, some non-tunable parameters are pertubed slightly at every step within the episode to model measurement noise.

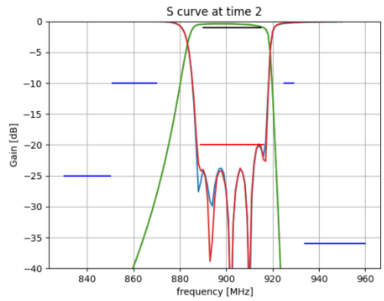
All experiments were conducted on a 16-core CPU (Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz) with 32 GB of RAM. Although training time varied, training of MFRL took around 2 hours on 6p2z and 1 day on 8p4z, while MBRL techniques averaged around one and three days respectively. When comparing MFRL with MBRL, all shared hyperparameters were kept the same,



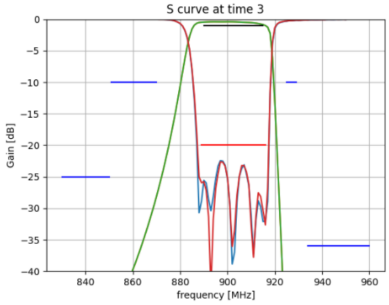
(a) Initial, detuned S-parameter configuration



(b) S-parameter configuration after 1 step



(c) S-parameter configuration after 2 steps.



(d) S-parameter configuration after 3 steps.

Figure 3: Typical example of tuning process on 6p2z.

to ensure that any performance boost can be directly attributed to the addition of a world model. We used the openly available implementations of SAC[2] and Dreamer[3], on top of which we performed the modifications described in the previous section.

4.1. Comparison with MFRL

Two properties play an integral role in model comparison: **success rate** and **sample complexity**. To measure success rate, we generate 10000 i.i.d. initially detuned filters and measure the proportion that gets tuned within the episode horizon. For the latter, we measure how many interactions with the environment are required before we reach a desirable success rate. Our findings are summarised in the following tables.

Table 1 summarises the results of the Dreamer on 6p2z. We see that the modified Dreamer performs similarly with that of the MFRL agent, while decreasing sample complexity by a factor of 4. It is also worth noting that the unmodified Dreamer was also less robust to its hyperparameters, often exhibiting too unstable training to even tune a filter (0 % success rate). Finally, table 2 showcases the performance of the modified Dreamer on the 8p4z environment. The agent manages to reduce sample complexity by a factor of 10. Figures 4a and 4b illustrate the reward training curves.

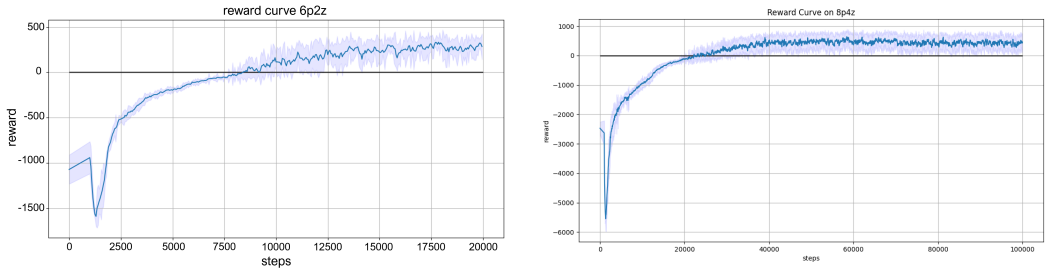
2. https://stable-baselines.readthedocs.io/en/master/modules/sac.html
3. https://github.com/danijar/dreamerv2

Dreamer Results on 6p2z			
	MFRL (SAC)	Dreamer (Original)	Dreamer (Improved)
Success rate	99.93 %	69.81 %	98.87 % / 99.72 %
Steps	100k	100k	16k / 32k

Table 1: Dreamer results on 6p2z. Success rate was measured on 10000 i.i.d. randomly initialised filter instances. MFRL reaches 99.93 % performance after 100k iterations, however it reaches similar performance with the improved Dreamer (99 %) after roughly 70k steps.

Dreamer results on 8p4z		
	MFRL (SAC)	Dreamer (Improved)
Success rate	93.00 % / 98.95 %	93.69 %
Steps	700k / 1 M	70k

Table 2: Dreamer results on 8p4z. Success rate was measured on 10000 i.i.d. randomly initialised filter instances.



(*a*) (*b*)

Figure 4: (a) Modified Dreamer on 6p2z. Mean and 95 % confidence interval plotted from 10 trials. (b) Modified Dreamer on 8p4z. Mean and 95 % confidence interval plotted from 10 trials.

5. Discussion and Future Work

One of the characteristics of MBRL is that they are very environment dependent [Wang et al. (2019)]. That is, certain models can outperform others in one environment and get outperformed in another. This can be attributed to the different assumptions and structures their world models employ. For instance, the modified dreamer architecture is able to effectively traverse the high dimensional observations of our environment, by leveraging its learned latent state representation. In fact, we also tried two different MBRL methods, namely SVG(H)-SAC [Amos et al. (2020)] and MBPO [Janner et al. (2019)] and found that their performance were not on par with our modified Dreamer. We argue that the Dreamer consistently outperforms them in this particular settings for two key reasons: (i) Its low dimensional latent space (the other two techniques were never tested on such high dimensional data) and (ii) it employs the usage of a model with memory (unlike MBPO). In essence, memory allows the agent to retain prior information gained throughout the

episode which can then be use to infer about the untunable background parameters which affect the environment dynamics.

Regarding hyperparameters, we found that the entropy coefficient and the discount γ are by far the most influential in model performance. Overall, the modified dreamer was fairly robust to its hyperparameters, exhibiting great, consistent performance for many different values. This is crucial, as low hyperparameter sensitivity shortens the duration of the hyperparameter tuning stage, which can be a prohibitively lengthy process in more complex environments or simulators.

Overall, we can say that MBRL does indeed showcase the potential to outperform MFRL on the CF Environment. The improvement, however, generally requires more extensive hyperparameter tuning and careful model selection. Unlike the MFRL SAC method, which was able to perform well (albeit not optimally) even with the hyperparameter settings provided by the original paper, the MBRL techniques required more meticulous hyperparameter search and modifications to ensure that we did not exhibit exploding gradients and bad performance (plateauing reward curve).

For more detailed information regarding hyperparameter significance, SVG(H)-SAC and MBPO performance comparison, please refer to the Thesis Work [Nimara (2021)] this paper is based on (in particular, sections 6.1, 6.2 and appendix).

There are many potential avenues for future work. First, a more thorough hyperparameter tuning process for 8p4z can be performed in order to further increase the performance of the agent. Alternatively, one could try to test our agent on more complex filters or realistic simulators (e.g. 3D simulator instead of a circuit model).

6. Conclusions

In this paper, we pioneer the usage of MBRL for the well established field of automated cavity filter tuning. Traditional techniques have proven efficient in tackling simpler filters, while MFRL, which can indeed tackle more complicated tasks, suffers from high sample complexity. This is particularly prohibitive in our setting, where more accurate simulators may require several minutes to generate a single environment observation. Our modifications on the Dreamer allowed the robust application of MBRL, decreasing sample complexity by orders of magnitude, while maintaining comparable performance with MFRL techniques. Finaly, we believe our findings may also be applicable in other environments with high dimensional non-image observations.

References

Brandon Amos, Samuel Stanton, Denis Yarats, and Andrew Gordon Wilson. On the model-based stochastic value gradient for continuous reinforcement learning, 2020.

M. Dishal. Alignment and adjustment of synchronously tuned multiple-resonant-circuit filters. *Proceedings of the IRE*, 39:1448–1455, 1951.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018. URL http://arxiv.org/abs/1812.05905.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

P. Harscher, R. Vahldieck, and S. Amari. Automated filter tuning using generalized low-pass prototype networks and gradient-based parameter extraction. *IEEE Transactions on Microwave Theory and Techniques*, 49(12):2532–2538, 2001. doi: 10.1109/22.971646.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

H. Larsson. Deep reinforcement learning for cavity filter tuning. 2018.

T. Lillicrap, Jonathan J. Hunt, A. Pritzel, N. Heess, T. Erez, Yuval Tassa, D. Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

S. Lindståh and X. Lan. Reinforcement learning with imitation for cavity filter tuning. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1335–1340, 2020. doi: 10.1109/AIM43001.2020.9158839.

V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, S. Petersen, C. Beattie, A. Sadik, Ioannis Antonoglou, H. King, D. Kumaran, Daan Wierstra, S. Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Igor Mordatch and Jessica Hamrick. Model based reinforcement learning tutorial. https://sites.google.com/view/mbrl-tutorial/, 2020.

Doumitrou Daniil Nimara. *Model Based Reinforcement Learning for the tuning of Cavity Filters*. M.Sc., Dept. of Electrical Engineering and Computer Science, KTH, 2021.

openAI. https://gym.openai.com/, 2021.

Raafat R. Mansour Richard J. Cameron, Chandra M. Kudsia. *Microwave Filters for Communication Systems: Fundamentals, Design, and Applications*. 2007.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019. URL http://arxiv.org/abs/1907.02057.

Zhiyang Wang. Reinforcement learning approach to learning human experience in tuning cavity filters. 12 2015. doi: 10.1109/ROBIO.2015.7419091.

Zhiyang Wang, Yongsheng Ou, Xinyu Wu, and Wei Feng. Continuous reinforcement learning with knowledge-inspired reward shaping for autonomous cavity filter tuning. 10 2018. doi: 10.1109/CBS.2018.8612197.