ETH zürich

medical data science

Personalised Sepsis Treatment with Reinforcement Learning

Research in Data Science

Marvin Koch

June 23, 2024

Supervisor: Prof. Dr. Julia E. Vogt
Advisors: Dr. Kacper Sokol, Ričards Marcinkevičs

CONTENTS

1 INTRODUCTION

In this project, we aim to train a Reinforcement Learning agent for optimizing the treatment of ICU patients suffering from sepsis. The use of Reinforcement Learning in clinical decision support systems holds a lot potential for improving patient care and outcomes. However, many existing solutions lack the ability to effectively handle temporal data and only make decisions based on the patient's current state. They also lack the ability to perform continuous actions and, as a consequence, can only administer predefined discrete dosages, which isn't an accurate representation of the real world. Recurrent actor-critic models offer a potential solution by leveraging past patient observations to provide more accurate and personalized recommendations, in addition to being able to perform continuous actions, which allows the agent to administer continuous drug dosages. Implementing these features lets the agent reason and act in a way that better reflects real-life clinical decision making.

2 RELATED WORK

The most notable work in this area stems from [1], who were the first to develop a Reinforcement Learning agent for sepsis treatment with results claimed to be better than a clinician. This work formed a basis for following works which tried to improve the model, for example [2] which focused on representation learning of the patient states.

However, these solutions still had not addressed key underlying issues such as the lack of consideration of patient history, the usage of discretized drug dosages rather than continuous values and the lack of reliable evidence that these solutions were indeed better, as the results only relied on estimations of the performance.

Recent research has tried to solve these issues by developing solutions which take patient history into account using the Deep Attention Q-Networks [3] and solutions which can administer continuous drug dosages using the Deep Deterministic Policy Gradient (DDPG) [4]. However, each of these solutions only focuses on trying to fix one problem and still suffers from the remaining issues.

In this project, we leverage current research and solutions in recurrent model-free Reinforcement Learning models using LSTMs [5] and Transformers [6]. We also build on top of the recent paper introducing a continuous solution for sepsis treatment using DDPG [4]. For evaluation, we use current research in the evaluation and interpretation of clinical Reinforcement Learning algorithms by [7].

3 BACKGROUND

3.1 Sepsis and the SOFA Score

Sepsis is a severe medical condition characterized by a systemic inflammatory response to infection, often leading to organ dysfunction and, in severe cases, septic shock. Treating sepsis often involves administering intravenous (IV) fluids to maintain adequate blood volume and ensure sufficient tissue perfusion. In addition, vasopressors may also be

administered to constrict blood vessels and raise blood pressure to achieve target blood pressure levels. The SOFA (Sequential Organ Failure Assessment) score is a scoring system used to assess the extent of a sepsis patient's organ function or rate of failure during critical illness. It evaluates six organ systems, assigning a score to each based on the degree of dysfunction, with higher scores being worse. The final score is an addition of the six subscores and ranges from 0 to 24, with each individual subscore ranging from 0 to 4. The following subscores are:

1. PaO2/FiO2 ratio – measures the ratio of arterial oxygen partial pressure (PaO2) to the fraction of inspired oxygen (FiO2).
2. Platelet count – measures the amount of platelets in blood, which are responsible for coagulation.
3. Bilirubin level – measures the amount of bilirubin in the liver.
4. Mean arterial pressure or administration of vasoactive agents – assesses cardiovascular health.
5. Glasgow coma scale score – assesses a patient's level of consciousness.
6. Creatinine level – measures serum creatinine levels or urine output.

3.2 Reinforcement Learning

Reinforcement Learning (RL) is a subdomain of machine learning focused on training agents to make sequences of decisions by interacting with an environment. Unlike supervised learning, where models learn from labeled data, RL agents learn from the outcomes of their actions, trying to maximize cumulative rewards. In RL, the agent represents the model making decisions, while the environment is the external system with which the agent interacts. The state refers to the current situation of the environment, actions are the decisions the agent can take, and rewards are the feedback from the environment following an action, quantifying the immediate benefit of that action. The policy, which can be deterministic or stochastic, is the strategy the agent uses to decide on actions based on the states. The value function estimates the expected cumulative reward from a state, while the Q-value estimates the expected cumulative reward from taking a specific action in a specific state.

3.2.1 Value-based, Policy-based and Actor-critic Methods

RL algorithms can be broadly categorized into value-based methods, policy-based methods, and actor-critic methods. Value-based methods estimate the value functions, with Q-Learning being the most popular algorithm. On the other hand, Policy-based methods directly optimize the policy without requiring a value function. Finally, Actor-critic methods combine value-based and policy-based approaches, where the actor updates the policy and the critic evaluates the action-value function.

3.2.2 Online vs Offline RL

Reinforcement Learning algorithms can also be classified into online and offline categories based on how they sample data. Online RL algorithms require generating new

data from the updated policy each time the policy is improved. In contrast, offline RL algorithms, such as Q-learning, can evaluate and improve a target policy using data generated from a different policy. In medical applications, the policy is the clinician's policy. Online RL is almost impossible to use in healthcare due to safety and ethical constraints, making offline RL algorithms the only viable option.

3.2.3 MDP

The underlying theoretical framework for RL is often a Markov Decision Process (MDP). An MDP is defined as a tuple $(S, A, T, T_0, R, H, \gamma)$, where S is the set of states, A is the set of actions, $T : S \times A \times S \to [0, 1]$ is the transition function, $T_0 : S \to [0, 1]$ is the initial state distribution, $R : S \times A \times S \to \mathbb{R}$ is the reward function, $H \in \mathbb{N}$ is the time horizon, and $\gamma \in [0, 1)$ is the discount factor. Solving an MDP involves learning a policy $\pi : S \times A \to [0, 1]$ that maximizes the expected discounted return.

3.2.4 POMDP

An expansion of the MDP is the Partially Observable Markov Decision Process (POMDP). A POMDP is defined as a tuple $(S, A, O, T, T_0, Z, O_0, R, H, \gamma)$, where S is the set of states, A is the set of actions available to the agent, and O is the set of observations that the agent can receive. The transition probabilities are given by $T : S \times A \times S \to [0, 1]$, with the initial state distribution defined by $T_0 : S \to [0, 1]$. The observation function $Z : S \times A \times O \to [0, 1]$ specifies the probability of observing an outcome given a state and action, while $O_0 : S \to [0, 1]$ represents the initial observation distribution. The reward function is determined by $R : S \times A \times S \to \mathbb{R}$, $H \in \mathbb{N}$ is the time horizon and $\gamma \in [0, 1)$ the discount factor. In a POMDP, the agent's decision-making process is based on its belief state, which represents its current knowledge or belief about the true state of the environment. The belief state is updated recursively. A policy in a POMDP maps belief states to actions, specifying how the agent should act based on its current belief state.

3.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data. RNNs process input sequences one element at a time, maintaining a hidden state that captures information about previous elements. The hidden state is updated at each time step based on the current input and the previous hidden state, enabling the network to memorize information. The hidden state h_t at time step t is computed as:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

where x_t is the input at time step t, W_{hx} and W_{hh} are weight matrices, b_h is the bias vector, and σ is an activation function, typically a non-linear function like $\tanh$ or ReLU. However, during training, gradients can become very small or very large, making it difficult to learn long-range dependencies. This is known as the vanishing/exploding gradient problem. Another issue with the vanilla RNN is that they struggle to remember information from many time steps earlier in the sequence.

3.3.1 Long Short-Term Memory Networks (LSTMs)

An LSTM [8] is a type of RNN designed to address the vanishing gradient problem and capture long-term dependencies. It utilizes a cell state C_t as memory, which is updated through various gates: the forget gate f_t decides what information to discard from the previous cell state C_{t-1}, the input gate i_t determines what new information to add via the candidate cell state $\tilde{C}_t$, leading to the updated cell state C_t. Finally, the output gate o_t controls the output, producing the hidden state h_t. These mechanisms enable LSTMs to retain essential information over long sequences. We formalize the desired behavior of each component with the following equations:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) & \qquad o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) & \qquad h_t &= o_t \cdot \tanh(C_t) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t
\end{aligned}
$$

We note that f_t, i_t, o_t and $\tilde{C}_t$ have their own weight matrix, with which we perform a linear transformation of the new input h_{t-1}, x_t, which we then pass through an activation function.

3.3.2 Transformers

Transformers [9] are another type of RNNs. Transformers use a self-attention mechanism, which allows the model to weigh the importance of different elements in the input sequence, regardless of their position. Unlike RNNs, transformers process the entire sequence simultaneously, leading to faster training and inference. In the self-attention mechanism, given a sequence, self-attention computes a weighted sum of the values based on the similarity (attention scores) between elements:

$$\text{Attention}(Q, K, V) = soft\ max\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q (Query), K (Key), and V (Value) are projections of the input sequence, and d_k is the dimensionality of the key vectors.

The original transformer model consists of an encoder and a decoder, each composed of multiple layers of self-attention and feedforward neural networks. The encoder processes the input sequence and produces a contextual representation, while the decoder uses the encoder's output to generate the target sequence, employing self-attention and encoder-decoder attention mechanisms. Since transformers do not process sequences sequentially, positional encoding is used to give the model information about the position of each element in the sequence. This is typically achieved using $sine$ and $cosine$ functions of different frequencies.

3.3.3 GPTs

Generative Pre-trained Transformers (GPT) [10] are a series of transformer-based models developed by OpenAI. GPTs specifically utilize the decoder part of the original transformer model. The transformer decoder in GPT consists of multiple layers of self-attention and feedforward neural networks. Each layer includes masked self-attention, which ensures that the prediction for a particular position depends only on the known outputs before it, layer normalization for stabilizing and accelerating training, and a feedforward neural network to add non-linearity and complexity to the model.

4 METHODS

4.1 Data Preprocessing

We use the electronic health records in the MIMIC-III dataset [11] and use the preprocessing steps from [2] which were also used in [4]. We extract a list of observations necessary for patient selection and to construct our state representations. The result is 38 features, which consist of demographics, vital signs, and lab tests. The patient's trajectory are then discretized into 4-hour windows. Missing values are interpolated using k-nearest neighbors. If there were multiple observations for a variable within a 4-hour window, the values are averaged. This is then used to select the patients that meet the sepsis-3 criteria [12].

4.2 Environment

In our work, we propose to model the environment as a POMDP. For clinical decision making, this provides a more accurate representation of the constraints and uncertainties that we have to face. In a clinical setting, the true state of the patient, such as the underlying health condition, is not truly fully observable. Instead, we have access to observations like test results and symptoms that provide partial information about the patient's state. This setup mirrors real-world clinical scenarios where doctors must make decisions based on limited information. In our setup we have:

States As mentioned before, the state representations are built in the preprocessing step using an autoencoder as proposed in [2].

Actions Our action space consists of a continuous range of IV fluids and vasopressor dosages, which the agent can administer to treat sepsis patients, as mentioned in Section 3.1.

4.3 Model

4.3.1 DDPG and TD3

DDPG Deep Deterministic Policy Gradient (DDPG) [13] is an algorithm designed for environments with continuous action spaces. It is an off-policy method that combines the strengths of Deep Q-Networks and Actor-Critic methods. DDPG uses a deterministic

policy to select actions (the actor) and relies on a Q-function to evaluate the expected return of these actions (the critic). In short we would like to train two neural networks, the actor network, which outputs the action given a state, and the critic network which learns a Q-function to be able to evaluate our actions. In practice, we train two additional networks called the actor target and critic target. The goal of these target networks is to slowly track the main networks to provide stable values for the updates. All that being said, DDPG often overestimates Q-values, leading to suboptimal policies.

TD3 Twin Delayed DDPG (TD3) [14] was introduced to address these issues with the following three changes:

1. TD3 learns two Q-functions, Q_{ϕ_1} and Q_{ϕ_2}, and uses the minimum of the two Q-values to form the target, reducing overestimation bias.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s'))$$

2. The policy and target networks are updated less frequently than the Q-networks. Typically, the policy is updated once for every two updates of the Q-networks, which reduces the variance of the policy updates.
3. TD3 adds noise to the target action to make the Q-function less prone to exploitation by the policy, effectively smoothing the Q-values along action dimensions.

$$a'(s') = clip\left(\mu_{\theta_{\text{targ}}}(s') + clip(\epsilon, -c, c), a_{Low}, a_{High}\right) \quad \text{for} \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

4.3.2 Recurrent TD3

Project's Goal The core idea we explore in this project is to use a recurrent version of the TD3 algorithm to perform sequential clinical decision making. We propose replacing the original actor and critic networks with recurrent neural networks (RNNs) to handle environments with partial observations or dependencies over time – in our case both apply. More specifically, the recurrent actor will be able to determine the next action given the history of observations and actions, in addition to the current state. To do this, we first feed the mentioned inputs to the RNN which encodes the dependencies and maintains a hidden state that summarizes the information from the past inputs. We then pass the output of the RNN to another neural network, usually a multi-layer perceptron, which is responsible for predicting the next action. The recurrent critic works the same way, first encoding the inputs using an RNN, and then forwarding the encoding to a neural network which estimates the Q-function.

Implementation For the implementation, we use the two most popular RNNs, which are LSTMs and Transformers. We use the recurrent TD3 implementation using LSTMs which was proposed by [5]. In addition, we use the variant using transformers proposed by [6] which uses the GPT-2 model [15] implemented by Hugging Face Transformer library [16].

Shared Encoder vs Separate Encoder One of the principal design choices mentioned in [5] and [6] is whether the actor and the critic should share the RNN encoder or have

separate ones. When using an LSTM as the encoder, the shared model can lead to higher gradient norms, which can hinder the learning process. This approach is less effective as it may cause interference between the learning signals of the actor and the critic. Using separate RNN encoders has been shown to improve performance. This separation helps in reducing the gradient interference and allows each network to learn its task-specific features more effectively [5]. In the case of using Transformers as the encoder, a solution was proposed to improve the shared architecture. It was observed that the best results were obtained with a shared architecture, where the critic parameters are frozen in the policy loss to prevent high gradient norms [6]. In consequence, in our experiments we explore a separate architecture using LSTMs and a shared transformer architecture using GPT-2.

4.3.3 Training the Critic

In TD3 [14], we train our critic using Q-learning. Recall the Bellman equation:

$$Q^{\star}(s,a) = \mathop{\mathrm{E}}_{s' \sim P}\left[r(s,a) + \gamma \max_{a'} Q^{*}(s',a') \right] . \tag{1}$$

This equation predicts the best action-value function $Q^{*}(s,a)$, where s is the current state and a is the action taken. Here, $s' \sim P$ denotes that the next state s' is sampled from the environment's transition probability distribution $P(\cdot|s,a)$. Like in the original TD3 implementation, we train $Q_{\phi}(s,a)$, parameterized by ϕ, to approximate this function:

$$L(\phi, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d) \sim \mathcal{D}}\left[\left(Q_{\phi}(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_{\phi}(s',a') \right) \right)^{2} \right] . \tag{2}$$

Here γ is the discount factor representing the importance of future rewards, and $(1-d)$ accounts for the terminal state. Our goal is to minimize this loss function.

4.3.4 Training the Actor/Policy

Original Loss Function The original loss function for the policy of the TD3 algorithm is:

$$\max_{\theta} \mathbb{E}_{s \sim D}\left[Q_{\varphi}(s, \mu_{\theta}(s)) \right] . \tag{3}$$

The equation is simple, we want to learn a policy μ_{θ} parameterized by θ which maximizes the Q function, ensuring that our action is the best possible.

Previously Proposed Loss Function The altered policy loss function to train TD3 for sepsis treatment proposed by [4] is the following:

$$\max_{\theta} \mathbb{E}_{s \sim D}\left[Q_{\varphi}(s, \mu_{\theta}(s)) - \lambda \cdot (a - \mu_{\theta}(s))^{2} \right] . \tag{4}$$

In this implementation a penalty was added to penalize divergence between the model's actions and those of the clinicians. Note that there is a time delay between their action and the clinician's action to avoid over-fitting by trivial action cloning. In addition, the time delay was added to nudge the model to consider the previous action taken.

Our Proposed Loss Function In turn, we propose the new policy loss function:

$$\max_{\theta} \mathbb{E}_{s\sim D}\left[Q_{\varphi}(s,\mu_{\theta}(s)) - \lambda_1 \cdot (a_{iv} - \mu_{\theta}(s)_{iv})^2 - \lambda_2 \cdot (a_{vp} - \mu_{\theta}(s)_{vp})^2\right] . \tag{5}$$

We have split the penalty into two components, one penalizing the difference in IV fluid dosage and the other the difference in vasopressor dosage. This new loss function gives us more flexibility to tune our model according to our needs. We also have removed the time delay, as the core advantage of our model is that is takes past decisions into account. In consequence, we don't need to incorporate the time delay as before to drive our model to consider previous actions.

4.4 Reward Function

4.4.1 Previously Proposed Reward Function

The reward function previously used in [4] for this problem is the following:

$$r(s_t, s_{t+1}) = \gamma_1 \tanh(s_t^{\text{SOFA}} - 6) + \gamma_2(s_{t+1}^{\text{SOFA}} - s_t^{\text{SOFA}}) , \tag{6}$$

where the values set were $\gamma_1 = -0.125$ and $\gamma_2 = -0.2$. These hyperparemeters can of course also be changed. The function aims to positively reinforce improvements in patient condition while penalizing deterioration. It has two components, one based on the current SOFA score, reflecting the patient's immediate state, and the other representing changes in the patient's state over time. We aim to penalize high sofa score and reward low ones, we are thus obliged to set our hyperparameters to negative numbers.

4.4.2 Proposed Modified Reward Function

We can enhance our training process by modifying the reward function to align with specific desired behaviors. Specifically, we aim to adjust the SOFA score to emphasize particular subscores more heavily. By decomposing the SOFA score within the reward function, we can fine-tune the weight given to each subscore. For instance, if we want to prioritize the Mean Arterial Pressure, which corresponds to the fourth subscore, we can decompose the SOFA score and introduce a bias towards this specific subscore. There are two ways we can alter the reward function. The conservative way would be to only change the first term of the function, which evaluates the current state of the patient:

$$r(s_t, s_{t+1}) = \gamma_1 \tanh(s_t^{\text{biasedSOFA}} - 6) + \gamma_2(s_{t+1}^{\text{SOFA}} - s_t^{\text{SOFA}}) . \tag{7}$$

The more aggressive way would be to change both terms in the function, so in addition to the evaluation of the current state, we also bias the way we evaluate the trend of the patient's health:

$$r(s_t, s_{t+1}) = \gamma_1 \tanh(s_t^{\text{biasedSOFA}} - 6) + \gamma_2(s_{t+1}^{\text{biasedSOFA}} - s_t^{\text{biasedSOFA}}) . \tag{8}$$

4.5 Evaluation

To evaluate our models we use evaluation techniques from [7], which we describe below. In addition, we propose some new methods.

4.5.1 Quantitative Evaluation

Quantitative evaluation is very difficult for this problem because we don't have access to a simulated environment where we can get real-time feedback and evaluate our models using standard on-policy evaluation. Instead, we have to resort to off-policy evaluation techniques. Unlike in the discrete setting where methods such as *weighted importance sampling* and *doubly robust methods* can be used, we use the direct method which was also the method used for continuous off-policy evaluation in [4]. The idea behind the direct method is to replace the rewards received for every action that we take with the values given by our learnt Q-function for that action. In short, the Q-function replaces the role of a simulated environment to determine if performing a certain action was adequate or not. More formally we have:

$$\hat{\rho}_{DM} = \mathbb{E}_N \left[\mathbb{E}_\pi \left[\hat{Q}(s_0, a_0) \mid s_0 \right] \right] \quad \text{with} \quad \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0, a_0 \right] = Q(s_0, a_0) \ . \tag{9}$$

This method is of course far from optimal and unfortunately too unreliable to be able to make any strong conclusions, as the estimate of the Q-function could easily be wrong.

4.5.2 Qualitative Evaluation

Existing Methods To further evaluate our models, we can use qualitative methods to gain a better understanding of their behavior. We will plot the following graphs for both IV fluids and vasopressors:

- the dosage distribution for both the clinician and our model
- the mean dosage per SOFA score for both the clinician and our model
- the U-curve

The U-curve plots the mean observed mortality for different dosage deviations between the clinician and our model. The idea is that the higher the deviation, the higher the observed mortality, thus creating a U-shape. These plots were proposed in [7] and used in previous works [4].

Our Proposed Methods We also introduce some new methods. We can plot the mean Q-function value per SOFA score to better understand how our Q-function evaluates our actions. Furthermore, we would like to analyze and gain a further understanding of how the mean dosages evolve in relation to each criterion of the SOFA score. We recall that the SOFA score is composed of six subscores. Using this, we can decompose our SOFA score and plot the mean dosage per subscore.

4.5.3 Further Remarks

Both the LSTM and Transformer models require hyperparameter tuning to achieve desired results. The most important hyperparameters in our case concern the complexity of the model. For the LSTM, the key hyperparameter we focus on is the number of hidden nodes per layer in our Actor and Critic networks, which we decide to keep equal. For the

Transformer, the hyperparameters we tune are the number of heads and the number of hidden layers each head has, which are consistent across all heads. Additionally, we also tune the hyperparameters newly defined in the loss function and reward for our models.

A *very important* aspect to consider is that defining what constitutes a good or bad result can ultimately only be decided by a clinician or medical expert. Although we have defined quantitative and qualitative methods to evaluate and reason about a model, these methods are neither reliable nor informative enough to conclusively determine the performance of our models. In general, it's important to exercise caution when utilizing off-policy evaluation techniques, as the results are not as dependable as we may think. Taking this into account, we do not seek to find the model with the best quantitative score, but instead aim to present the most interesting results and demonstrate what is possible. Furthermore, we cannot make any claims about our model being better or worse than previous works as long as we only have these evaluation methods at our disposal.

5 Results

5.1 TD3 with LSTM Encoder

5.1.1 Varying the Number of Hidden Nodes per Layer

We first concentrate on changing the amount of hidden nodes per layer in our Actor and Critic, observing how this influences our policy. In this case, we neither bias our loss nor reward function, keeping them in the original state. We let λ_1 and λ_2 in our loss function (Equation 5) equal to 1 and keep the original values of γ_1 and γ_2 in the original reward function with an unbiased SOFA score (Equation 6).

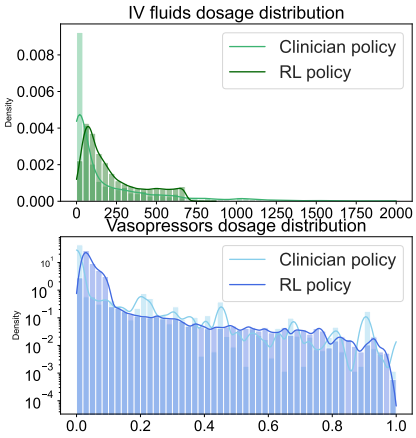

Figure 1: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the LSTM with 16 nodes per layer and the clinician.

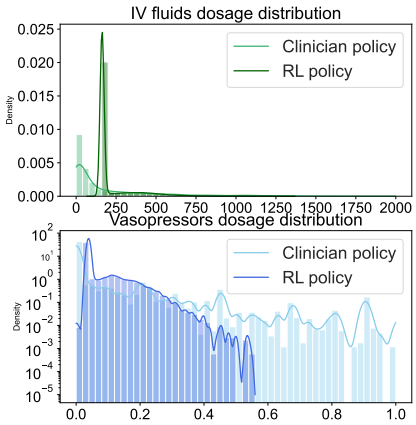

Figure 2: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the LSTM with 128 nodes per layer and the clinician.

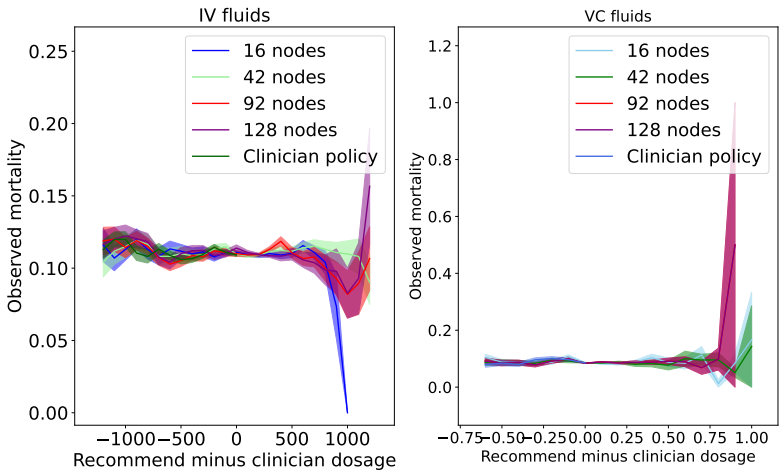

Figure 3: Comparison of IV fluid and vasopressor U-curve for different LSTM nodes per layer, where the U-curve shows how the mean patient mortality (visualized with the standard error) evolves the more our model deviates from the clinician.

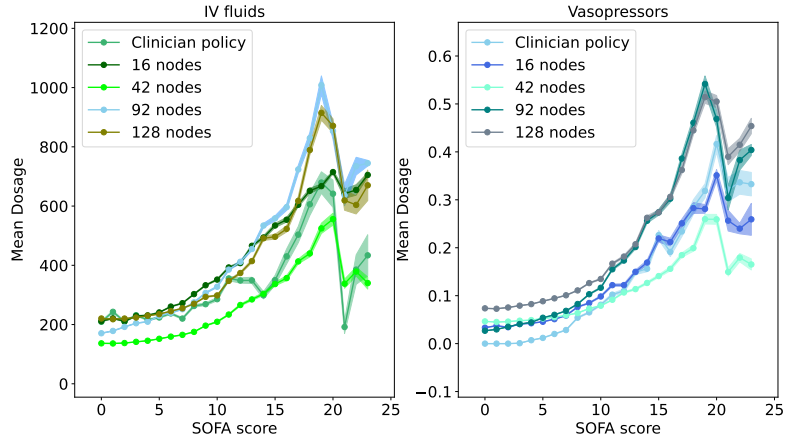

Figure 4: Comparison of the mean dosage per SOFA score for IV fluid and vasopressors (visualized with the standard error) between different LSTM nodes per layer and the clinician.

Looking at the distribution of the dosages (Figures 1–2), we observe that the model is more aggressive for IV fluids, assigning higher dosages than the clinician on average. For vasopressors, a clear trend is harder to point out.

Looking at Figure 3 for the U-curve, we get some surprising results. First of all, looking at the general trend, we see that our models do not generate the desired curve, with the results being particularly flat for both IV fluids and vasopressors, especially when our policy doesn't assign enough dosages. This can be explained by looking at the dosage distribution per SOFA score (Figure 4). Our policies generally are only more conservative for lower SOFA scores, meaning that the mortality rate for these cases is also very low. On the other hand, the results when our policy is more aggressive vary a lot. Policies such as the LSTM with 16 nodes per layer tend to assign higher dosages than the clinician for lower SOFA score, resulting in the curve trending downwards. On the other hand, we have policies such as the LSTM with 128 nodes per layer which are very aggressive

for high SOFA scores, which naturally suggests high mortality. Finally, it is interesting to remark that matching the clinician's policy doesn't guarantee low mortality rate. The policies tend to match the clinician for patient states which naturally have higher mortality rate, resulting in a flawed result. Further plots of the dosage distribution and the mean dosage per each SOFA subscore can be found in the appendix (Figure 37).

	Model	**Clinician**
16 nodes per layer	0.7430	0.7486
42 nodes per layer	1.0940	1.1097
92 nodes per layer	1.3676	1.3744
128 nodes per layer	1.2947	1.3074

Table 1: Evaluation of results of the different LSTM models compared to the clinician using the direct method.

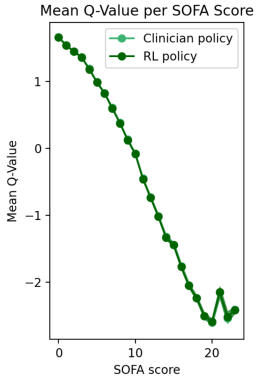

Figure 5: Comparison of the mean Q-value per SOFA score between the LSTM with 16 nodes per layer and the clinician (they perfectly overlap).

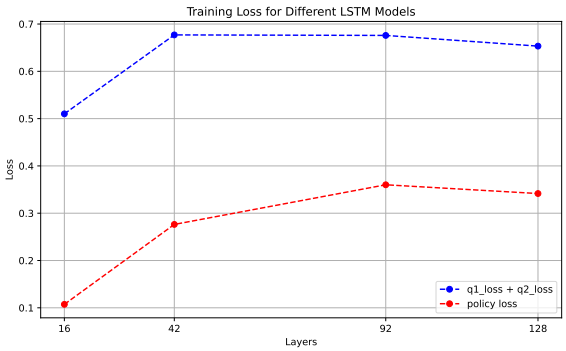

Figure 6: Comparison of the critic loss (q1 + q2 loss) and the policy loss for the different LSTM models (total loss is the sum of both).

From a quantitative (Table 1) and qualitative (Figures 1–5) view, we can see that our LSTM-based model gives results which we consider more desirable for a lower amount of hidden nodes per layer. Looking at Figure 6 this stems from the model under fitting for a high amount of hidden nodes per layer. Analyzing the model further, we find out that the model is suffering from the vanishing gradient problem, with the norm of our gradients becoming infinitesimally small as we increase the complexity of the model.

5.1.2 Varying the Hyperparameters of the Loss Function

Next, let's observe how our model changes when we change parameters λ_1 and λ_2 in our loss function given by Equation 5. We recall that these are the hyperparameters which determine how strongly we penalize the deviation between our and the clinician's actions, with λ_1 regulating the deviation between IV fluids and λ_2 between vasopressors. We use an LSTM with 16 nodes per layer for demonstration, as it was the best performing model according to our evaluations. We are interested to see if the model can be further improved or if changing the parameters worsens the results.

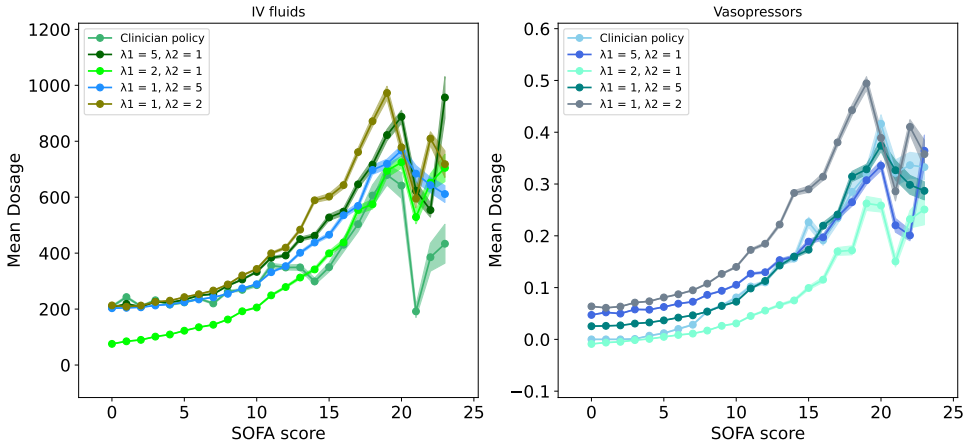

Figure 7: Comparison of mean dosage per SOFA score for IV fluids and vasopressors between different values of λ_1 and λ_2 in the loss function. We use an LSTM implementation with 16 nodes per layer.

We observe in Figure 7 that for high values of λ_1 or λ_2, we can indeed bias the behavior of our model towards the IV fluid or vasopressor dosages administered by the clinician. In cases where we only slightly adjust λ_1 or λ_2, our modification isn't strong enough to achieve the desired influence. The issue is that biasing our behavior towards one dosage can worsen the performance related to the other dosage. Optimal behavior would need to strike the perfect balance between the two. A candidate in our example would be $\lambda_1 = 1$, $\lambda_2 = 5$. Again in Figure 7, we see that this policy not only fits the vasopressors well, as it should, but also is arguably the best for IV fluids, as it closely matches the clinicians' peaks. As mentioned previously, only a medical expert can determine which behavior would be the most desirable for real-world applications. The mean dosage per each SOFA subscore can be found in the appendix (Figure 45).

5.1.3 Varying the Hyperparameters of the Reward Function

Simple Modification We now move on to the reward function. We first start by observing the behavior of the model when we simply modify the first term of our reward function, as given by Equation 7. The model we use for illustration is again an LSTM with 16 nodes per layer.

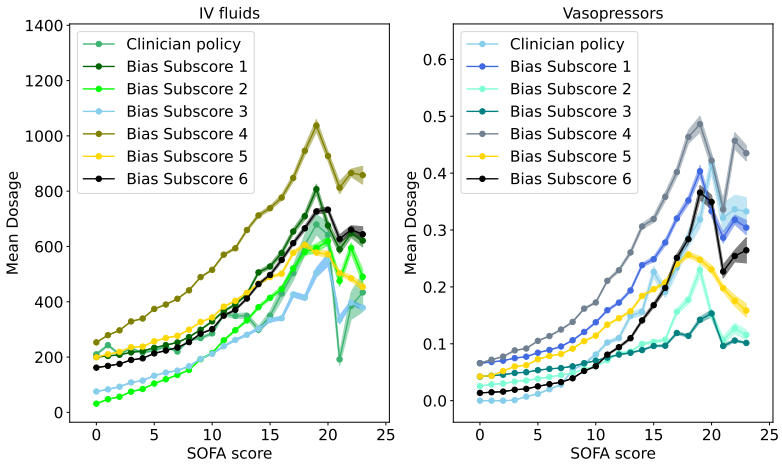

Figure 8: Comparison of mean dosage per SOFA score for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

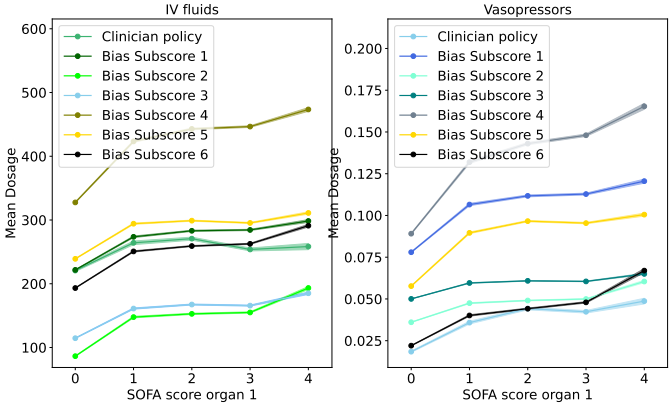

Figure 9: Comparison of mean dosage per SOFA subscore 1 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

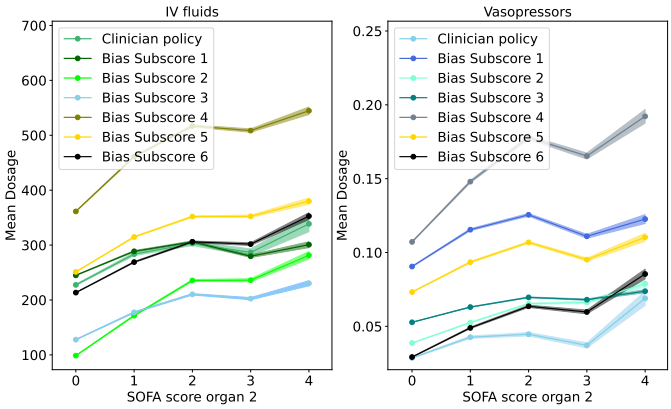

Figure 10: Comparison of mean dosage per SOFA subscore 2 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

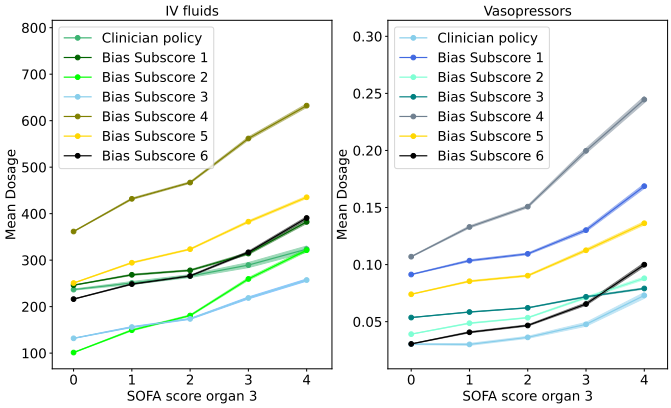

Figure 11: Comparison of mean dosage per SOFA subscore 3 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

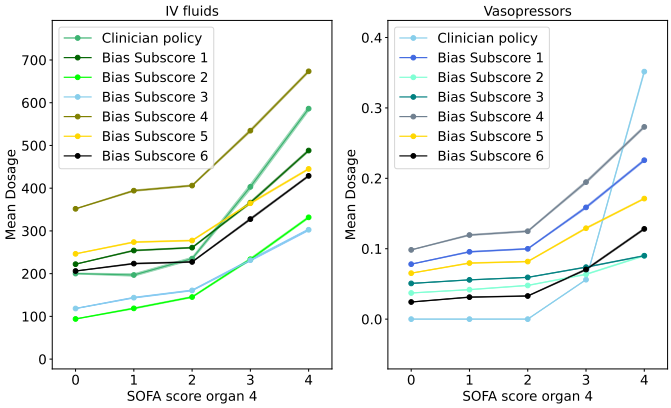

Figure 12: Comparison of mean dosage per SOFA subscore 4 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

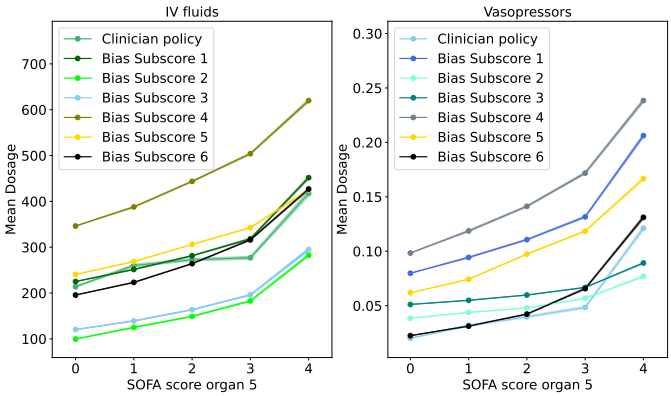

Figure 13: Comparison of mean dosage per SOFA subscore 5 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

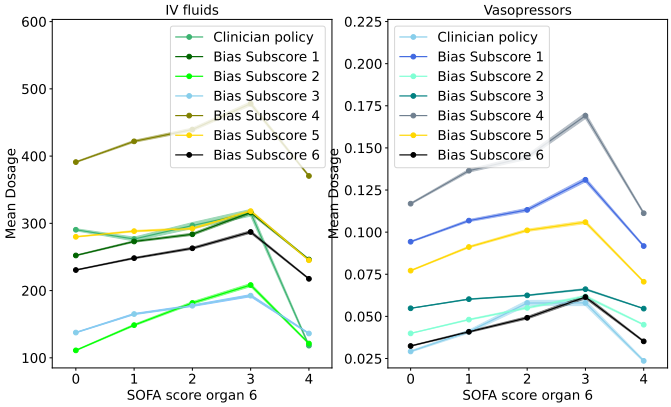

Figure 14: Comparison of mean dosage per SOFA subscore 6 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing only 1 term).

To evaluate the effectiveness of our modification, we examine each specific subscore graph (Figures 9–14). Our desired outcome is that each biased model will fit best to the clinician policy of the corresponding subscore. This would show that our biased model is indeed fitting itself to the specific subcriteria. Looking at our results (Figures 9–14), we observe that the set of policies nearest to the clinician's actions indeed always contain the corresponding biased model. That being said, very often, it isn't the one which fits the clinician the closest. This isn't surprising, as a few different subscores have the same trend, which means that biases toward certain criteria can also improve performance on others. This is most the noticeable for subscore 2 (Figure 10) and subscore 3 (Figure 11), where the model biased toward the corresponding subscore isn't as close to the clinician as we would like. Instead for both of these examples, the model which fits the clinician best is the model biased to subscore 6. Looking at Figure 14, we see that the policy bias towards subscore 6 is encouraged to assign dosages in a similar range as subscores 2 and 3, which results in it performing very well for other subscores too. Nonetheless, we achieve what we wanted, which is to demonstrate that our model shows a tendency to bias itself towards a specific sub-criteria, which is backed up by subscore 4 (Figure

12), subscore 5 (Figure 13) and subscore 6 (Figure 14), where the corresponding biased model fits the clinician very well.

Modifying Both Terms Next, we modify both elements in the reward function, biasing both the current and next SOFA score, as shown in Equation 8.

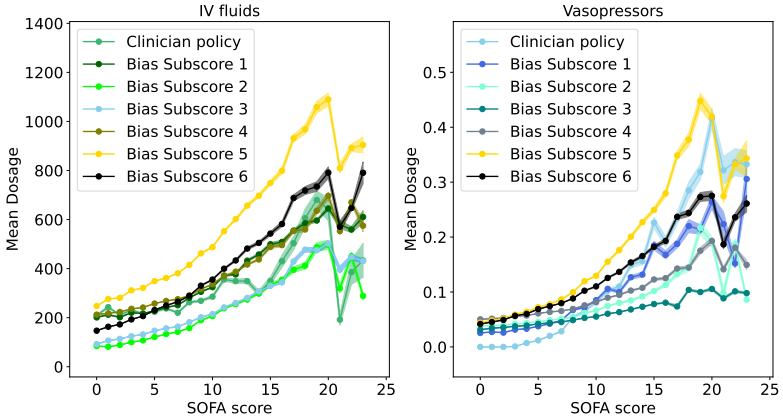

Figure 15: Comparison of mean dosage per SOFA score for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing the whole reward function).

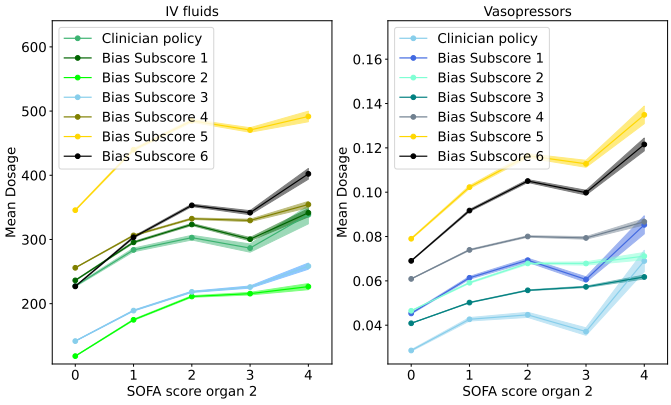

Figure 16: Comparison of mean dosage per SOFA subscore 2 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing the whole reward function).

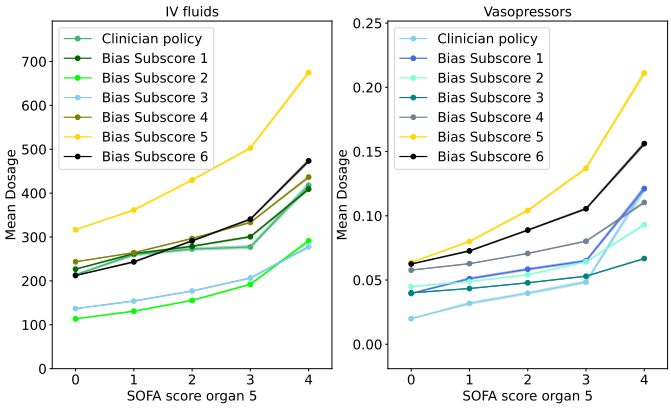

Figure 17: Comparison of mean dosage per SOFA subscore 5 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing the whole reward function).

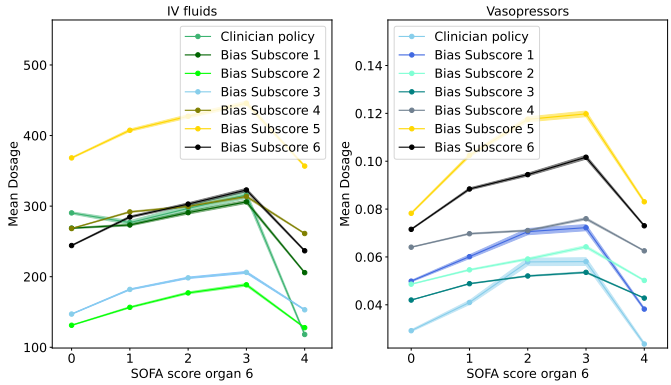

Figure 18: Comparison of mean dosage per SOFA subscore 6 for IV fluids and vasopressors between models (LSTM with 16 nodes per layer) with different subscore biases (biasing the whole reward function).

We recall our hypothesis which is that this additional modification will push our biased model to fit its corresponding clinician policy more closely. Upon examining our results, we assert that they do not align with our hypothesis. The biased models actually fit themselves slightly less to their corresponding clinician policies than before. We can see this for example in Figure 16, where the model biased to subscore 2 is further away from the clinician than in Figure 10. In addition, we observe policies which assign very high dosages, especially for IV fluids. The policies with by far the highest dosages are those biased towards subscores 5 (Figure 17) and subscore 6 (Figure 18). We hypothesise in the discussion (6) what the reasons may be. The remaining subplots can be found in the appendix (Figure 57).

Varying Hyperparameters γ To further analyze how the two components in our reward function influence the total reward, we remove the biases, reverting to the original reward function (Equation 6), and now tune γ_1 and γ_2.

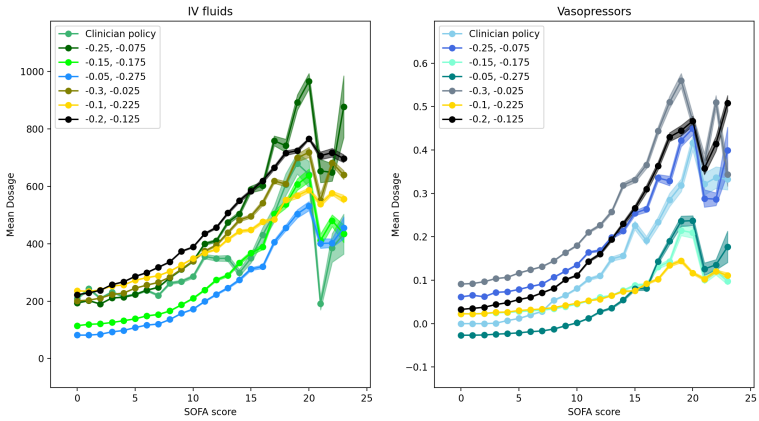

Figure 19: Comparison of mean dosage per SOFA score for IV fluids and vasopressors for different parameters γ_1 and γ_2 in the reward function using LSTM implementation with 16 nodes per layer.

Adjusting the hyperparameters yields interesting results. In Figure 19, we recognize a clear trend where higher values for γ_1 increase the dosage, and higher values for γ_2 decrease the dosage, for both IV fluids and vasopressors. We assert that fitting our model too strongly to the difference in scores (increasing γ_2) gives results which we consider to be worse according to this graph, as the model deviates very strongly from the clinician. Taking into account the previous observations from the total biased modification of our reward (Figures 15–18), we assert that biasing our function too much towards the difference in scores in the reward function can make the results less desirable. The subplots of the mean dosage per SOFA subscore can be found in the appendix (Figure 51).

5.2 TD3 with Transformer Encoder

5.2.1 Varying the Number of Hidden Layers and Heads

We now move our attention (no pun intended) to the transformer implementation. We first change the amount of heads and hidden layers of our GPT. Again, we keep λ_1 and λ_2 in our loss function (Equation 5) equal to 1 and keep the original values of γ_1 and γ_2 in the reward, with an unbiased SOFA score (Equation 6).

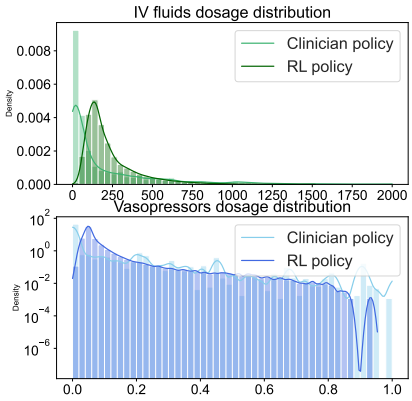

Figure 20: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the Transformer with 1 layer, 1 head and the clinician.

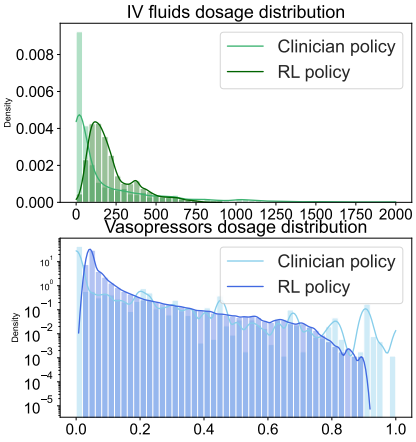

Figure 21: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the Transformer with 4 layers, 4 heads and the clinician.

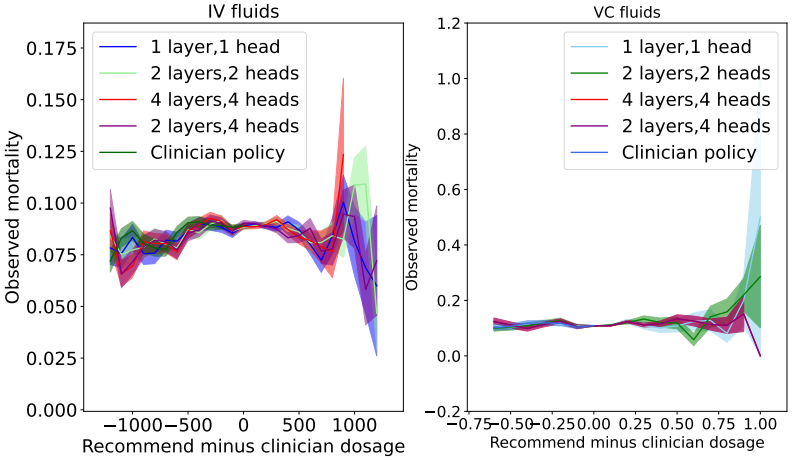

Figure 22: Comparison of IV fluid and vasopressor U-curve for different Transformer models, where the U-curve shows how the mean patient mortality (visualized with the standard error) evolves the more our model deviates from the clinician.

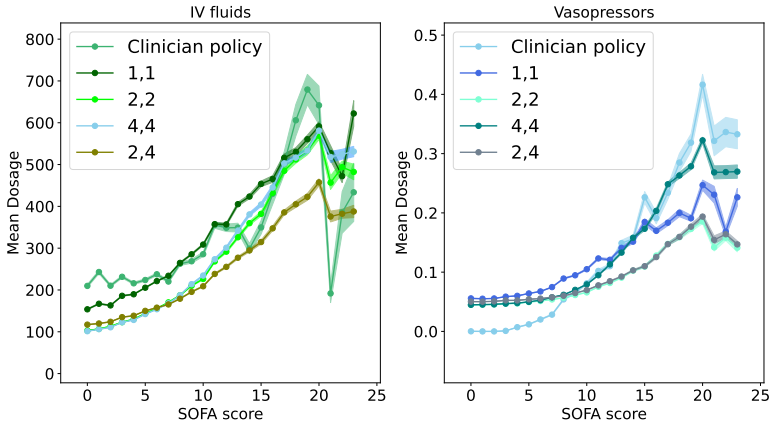

Figure 23: Comparison of the mean dosage per SOFA score for IV fluid and vasopressors (visualized with the standard error) between different Transformer models and the clinician.

We observe a relatively similar IV fluid dosage distribution (Figures 20 -61) and U-curve (Figure 22) for IV fluids to the LSTM implementation (Section 5.1.1), with the model being a bit more aggressive than the clinician on average, judging from the distribution of the dosages (Figures 20 -61). That being said, looking at Figure 23, the model this time tends to be more conservative on IV fluids for patients with low SOFA scores. For the vasopressors, we observe a very different behavior compared to the clinician and the LSTM implementation (Section 5.1.1), with the model being very conservative. If we purely judge our model based on Figure 23, it is clearly underfitting, assigning dosages which are too high for low SOFA scores, and not assigning enough vasopressors for patients with high scores. Looking at the U-curve for vasopressors (Figure 22), we see that for most policies our curve stays flat for a negative difference. When the difference becomes positive, it shoots up for the model with 1 layer, 1 head and for the one with 4 layers, 4 heads, while staying flat for the others. We point this behavior out because looking at Figure 23, positive difference for the models only occur for low SOFA scores. This seems very incoherent, as high mortality should occur for high SOFA scores, but could also suggest that the model may not be totally wrong after all, assigning higher vasopressor dosages for patients with low SOFA scores but who have died unexpectedly. Further plots of the dosage distribution and the mean dosage per each SOFA subscore can be found in the appendix (Figure 60).

	Model	**Clinician**
1 head, 1 layer	0.7897	0.8018
2 heads, 2 layers	0.9844	0.9952
4 heads, 4 layers	1.0662	1.0780
4 heads, 2 layers	1.1212	1.1336

Table 2: Average Q-value estimations from the critic for Transformer models.

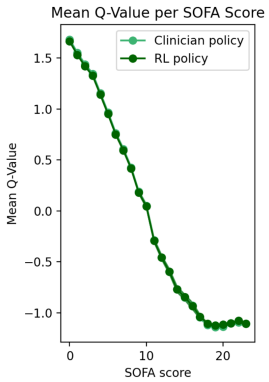

Figure 24: Comparison of the mean Q-value per SOFA score between the Transformer and the clinician (they perfectly overlap).

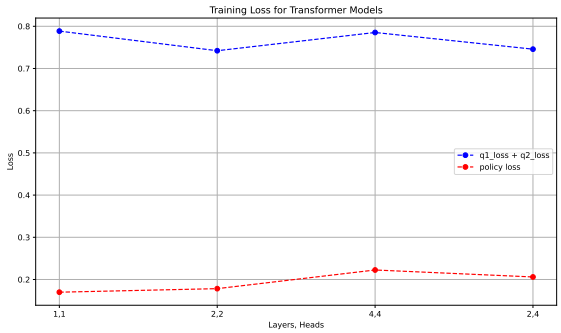

Figure 25: Comparison of the critic loss (q1 + q2 loss) and the policy loss for the different Transformer models (total loss is the sum of both).

From a quantitative (Table 2) and qualitative (Figures 20–67) view, there isn't a trend suggesting how many layers and heads could be the optimal amount, as each flavor has its pros and cons. We can support this claim by observing that the training loss stays equal for every variant of the model (Figure 25).

5.2.2 Varying the Hyperparameters of the Loss Function

Here again, we tune parameters λ_1 and λ_2 in the loss function given by Equation 5. We use a transformer with 1 head and 1 layer for demonstration. We decided to opt for this version because, along with the transformer with 4 heads and 4 layers, it had the closest results to the clinician if we evaluate our model using Figure 23. We are interested to see if the model can be further improved and corrected with our modifications.

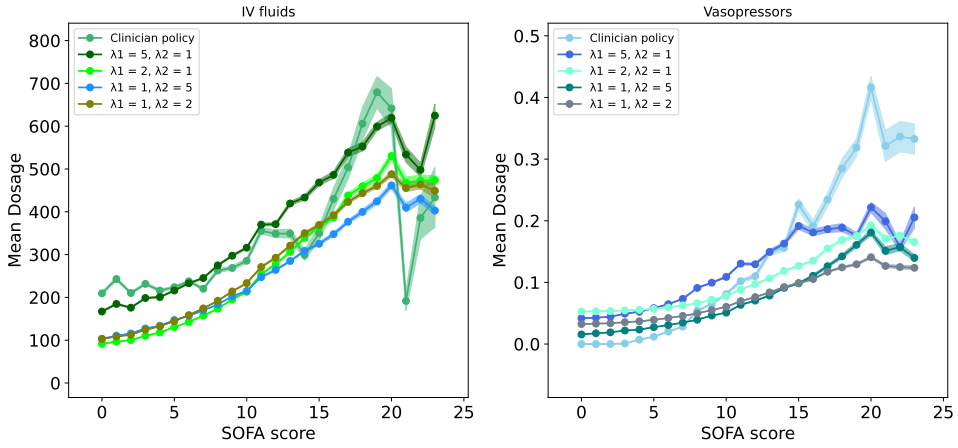

Figure 26: Comparison of mean dosage per SOFA score for IV fluids and vasopressors between different values of λ_1 and λ_2 in the loss function. We use a Transformer implementation with 1 layer and 1 head.

Similar to the LSTM implementation (Section 5.1.2), we observe in Figure 26 that for high values of λ_1 or λ_2, we can indeed bias the behavior of our model towards the IV fluid or

vasopressor dosages administered by the clinician. Looking at plots for $\lambda_1 = 1, \lambda_2 = 2$ and $\lambda_1 = 2, \lambda_2 = 1$, we assert that small adjustments for λ_1 or λ_2 aren't strong enough to achieve the desired influence, as was already the case for the LSTM implementation (Figure 7). It is interesting to remark that Figure 26 also illustrates the limits of our method. Looking at the vasopressors, we see that $\lambda_1 = 1$, $\lambda_2 = 5$ is the policy which fits the clinician the best, but it still is very far away from the desired result. We would have to increase λ_2 even more, but that would worsen our policy too much for IV fluids. The mean dosage per each SOFA subscore can be found in the appendix (Figure 68).

5.2.3 Varying the Hyperparameters of the Reward Function

Simple Modification We now again focus our attention on the reward function. As for the LSTM implementation, we try out both alterations of the reward function. We start with the simpler alteration, given by Equation 7, again by using a transformer with 1 head and 1 layer.

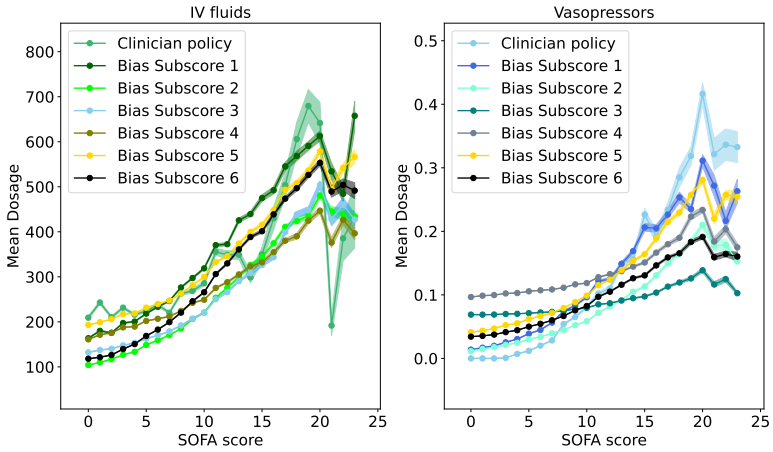

Figure 27: Comparison of mean dosage per SOFA score for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term).

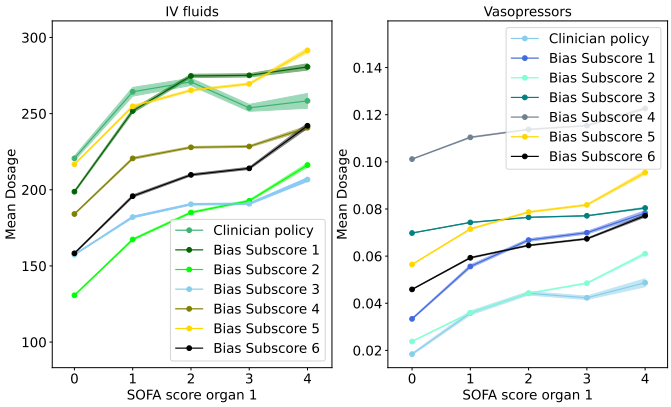

Figure 28: Comparison of mean dosage per SOFA subscore 1 for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term).

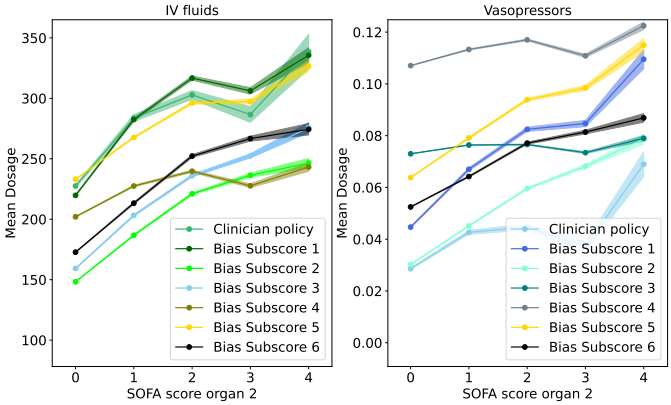

Figure 29: Comparison of mean dosage per SOFA subscore 2 for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term).

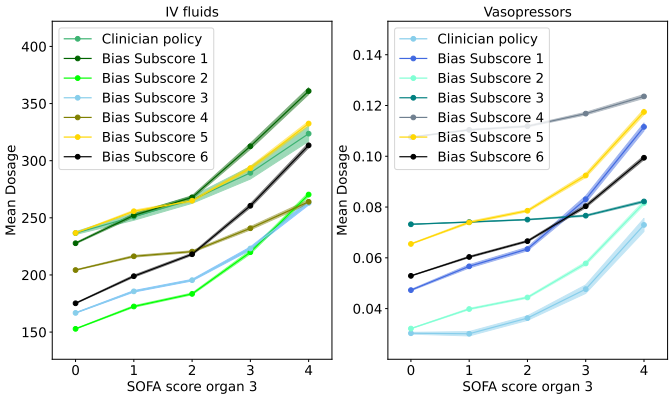

Figure 30: Comparison of mean dosage per SOFA subscore 3 for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term).

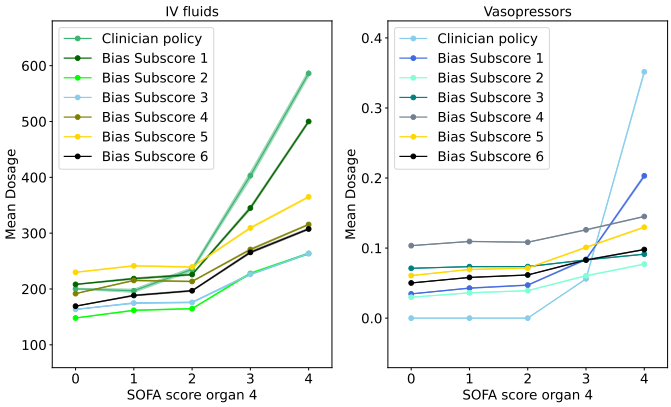

Figure 31: Comparison of mean dosage per SOFA subscore 4 for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term).

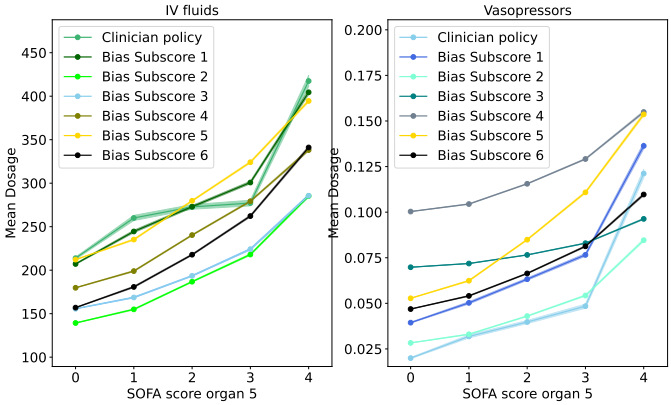

Figure 32: Comparison of mean dosage per SOFA subscore 5 for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term)

.

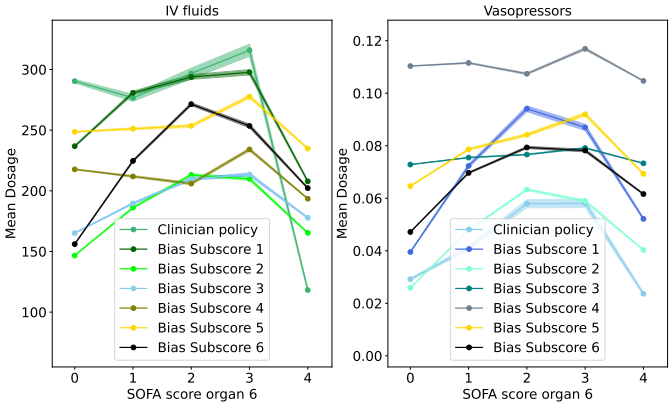

Figure 33: Comparison of mean dosage per SOFA subscore 6 for IV fluids and vasopressors between models (Transformer with 1 layer, 1 head) with different subscore biases (biasing only 1 term).

We notice the same trends in Figures 27–33 as in the LSTM implementation (Figures 8–14), albeit they are less noticeable. We still do see that our biased policy begins to shape itself to the corresponding clinician policy, but it is much less pronounced. This can clearly be seen for example for subscore 6 (Figure 33), where the biased model is shaping itself to the clinician but isn't totally in the correct dosage range. We also have cases where a biased model itself doesn't behave as desired. For example in Figure 30, the policy biased towards subscore 3 doesn't assign enough IV fluids and way too many vasopressors compared to the desired policy of the clinician. The deviation from this model to the clinician wasn't as large for the LSTM case (Figure 11).

Modifying Both Terms Moving on, we observe what happens if we completely change the reward function, as given by Equation 8.

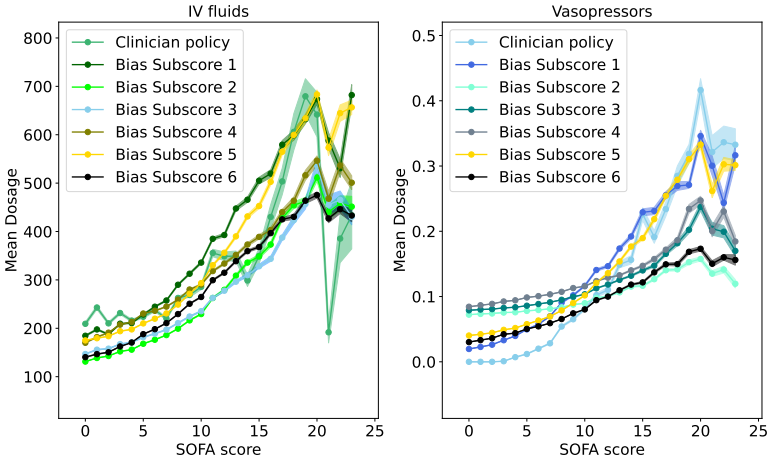

Figure 34: Comparison of mean dosage per SOFA score for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

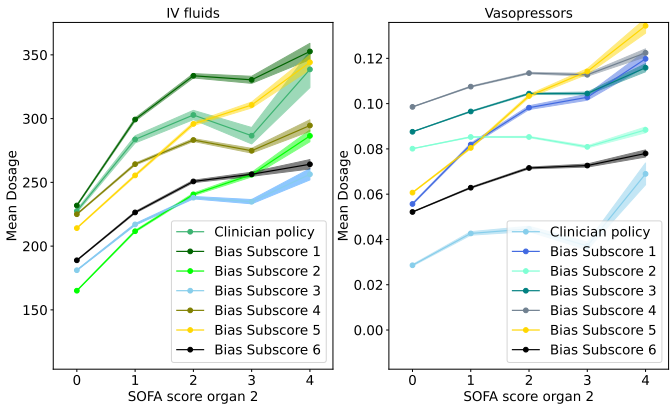

Figure 35: Comparison of mean dosage per SOFA subscore 2 for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

Comparing the LSTM implementation (Figure 15) with the Transformer implementation

(Figure 34), we notice a similar trend, but which is again much less pronounced, with the model not biasing itself as strongly. Taking subscore 5 as an example, in Figure 15 it clearly was the policy which assigned the highest dosages, whereas for transformers (Figure 34) it still assigns high dosages but doesn't distinguish itself as much from the other policies anymore. Comparing subscore 2 from the LSTM implementation (Figure 16) and the transformer implementation (Figure 35) also shows us the similarity between the two models. We again assert that this contradicts our initial expectation that further biasing our reward function would further encourage our model to further fit itself to the clinician. The remaining subplots can be found in the appendix (Figure 80).

Varying Hyperparameters γ Finally, we once more follow the same steps as the LSTM section (Section 5.1.3) and look at the results for different values of γ_1 and γ_2 in our original reward function (Equation 7).

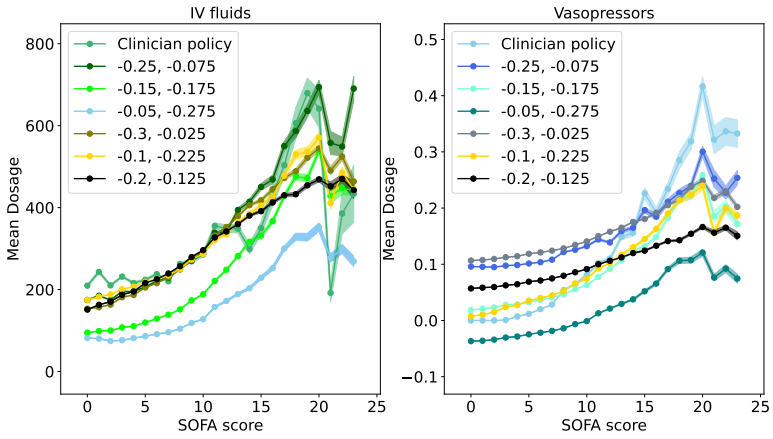

Figure 36: Comparison of mean dosage per SOFA score for IV fluids and vasopressors for different parameters γ_1 and γ_2 in the reward function using Transformer implementation with 1 head, 1 layer.

Our results are consistent with all our previous observations. Looking at Figure 36, higher values for γ_2 have lower the dosages for IV fluids and vasopressors, while higher values for γ_1 tend to have higher dosages. This mirrors the observation we made for the LSTM implementation (Figure 19), but this behavior is once again more ambiguous and less pronounced here. The subplots of the mean dosage per SOFA subscore can be found in the appendix (Figure 74).

6 Discussion

6.1 TD3 with LSTM Encoder

When adjusting the number of hidden nodes per layer in LSTM models, we find that simpler architectures tend to yield results which are more desirable by our metrics, which suggest that the LSTM model with 16 nodes per layers performs on par with the clinician. As mentioned before, this is certainly due to the more complex models suffering from the vanishing gradient problem.

Adjusting the loss function parameters (λ_1 and λ_2) reveals that biases towards specific dosages can be introduced, but finding the optimal balance is challenging, as biasing too strongly towards one dosage can negatively impact the performance on the other.

In the case of the reward function, modifying biases towards specific SOFA subscores demonstrates mixed results. While implementing a simple bias in the current SOFA score can sway our model to align itself with the clinician policy of the specific sub-criteria, biasing the whole reward function doesn't yield the desired results. We hypothesises that in this case, the model assigns high dosages for subscores with high variance and that in consequence subscore 5 (Figure 17) and subscore 6 (Figure 18) have the most variance. The only difference to our first modification is that we take the difference between the current score and the next score into account. Therefore, we can suppose that these high dosages are likely triggered by sudden changes in the patient's subscore, indicating frequent increases in the subscore. It remains unclear whether this behavior is desirable, as we must rely on the expertise of medical professionals to make that determination. However, based on our evaluation, this approach does not appear to be ideal. We also have observed that adjusting the hyperparameters γ_1 and γ_2 in the reward function can significantly influence the outcomes, with high values for γ_1 encouraging high dosages, and high values for γ_2 encouraging a more conservative policy.

6.2 TD3 with Transformer Encoder

Concerning Transformer models, we observe similar trends but also distinct differences in dosage distribution compared to LSTM models, with Transformer models exhibiting a conservative behavior towards vasopressor dosages. Additionally, the lack of a clear trend in the optimal number of layers and heads indicates that the model may not be able to capture and learn the patterns underlying the problem. We hypothesize that this may be due to our transformer model being a shared model, where our policy loss is combined with the Q-learning loss and is backpropagated to our actor and critic simultaneously, contrary to our LSTM implementation, where the policy loss is only backpropagated to the actor. This restricts our freedom to train the actor and the critic independently, which may diminish our model's capacity to fully learn this specific problem.

Regarding the loss function, adjusting parameters λ_1 and λ_2 in Transformer models exhibited outcomes comparable to those in LSTM models. This adjustment can be used to incrementally fine-tune the model, but isn't a magical fix. Modifying the reward function also produces a similar behavior to the LSTM implementation, however the model has a notable tendency to underfit and be less sensitive to modifications. We hypothesize that this is again due to the model having a shared-architecture, which doesn't allow it to fully capture the underlying patterns of the modifications.

7 Conclusion

In this project, we have explored the use of recurrent TD3 with both LSTM and Transformer encoders, to optimize the sequential treatment of sepsis patients in the ICU. By varying the parameters of our model, we have demonstrated the capability to achieve results comparable to those of clinicians. Moreover, we have shown that it is feasible to tune the model to meet more specific requirements. However, our experiments have

highlighted the significant impact that key design decisions have on the results, such as the choice of architecture, loss function, reward function and hyperparameter adjustments. In addition, our results also show that separate-architecture LSTM models tend to outperform shared-architecture Transformer models for this problem, according to our metrics.

Future Work Our work opens up many paths for future research. First of all, our work hasn't explored all the possible design choices. More specifically, it would be very valuable to explore shared-architecture LSTM models and separate-architecture Transformer models. In addition, there are also several hyperparameters which were kept constant and haven't been further tuned, such as the number of layers of our LSTM or nodes per layer in the Transformer. Solving the vanishing gradient problem for the LSTM implementation with a high number of layers, for example through clipping, is also a path worth exploring.

A key limitation of our project is the lack of reliable continuous off-policy evaluation techniques, which prevents us from performing a true direct comparison with the clinician and previous works. Further advancements in off-policy evaluation are crucial to improve and reason about Reinforcement Learning methods in clinical settings.

Further down the line, incorporating expert clinical knowledge to tune, validate and interpret the model should be considered.

REFERENCES

[1] M. Komorowski, L. A. Celi, O. Badawi, A. C. Gordon, and A. A. Faisal, "The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care," 2018.

[2] T. W. Killian, H. Zhang, J. Subramanian, M. Fatemi, and M. Ghassemi, "An empirical study of representation learning for reinforcement learning in healthcare," in *Machine Learning for Health*, pp. 139–160, PMLR, 2020.

[3] S. Ma, J. Lee, N. Serban, and S. Yang, "Deep attention q-network for personalized treatment recommendation," 2023.

[4] Y. Huang, R. Cao, and A. Rahmani, "Reinforcement learning for sepsis treatment: A continuous action space solution," 2022.

[5] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free rl can be a strong baseline for many pomdps," 2022.

[6] T. Ni, B. Eysenbach, and R. Salakhutdinov, "When do transformers shine in rl? decoupling memory from credit assignment," 2023.

[7] O. Gottesman, F. Johansson, J. Meier, J. Dent, D. Lee, S. Srinivasan, L. Zhang, Y. Ding, D. Wihl, X. Peng, J. Yao, I. Lage, C. Mosch, L. wei H. Lehman, M. Komorowski, A. Faisal, L. A. Celi, D. Sontag, and F. Doshi-Vele, "Evaluating reinforcement learning algorithms in observational health settings," 2018.

[8] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023.

[10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *OpenAI*, 2018. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[11] A. E. W. Johnson, T. J. Pollard, L. Shen, *et al.*, “Mimic-iii, a freely accessible critical care database,” *Scientific Data*, vol. 3, p. 160035, 2016.

[12] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche, C. M. Coopersmith, *et al.*, “The third international consensus definitions for sepsis and septic shock (sepsis-3),” *JAMA*, vol. 315, no. 8, pp. 801–810, 2016.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2019.

[14] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” 2018.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI Blog*, 2019.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, “Huggingface’s transformers: State-of-the-art natural language processing,” 2020.

A Appendix

A.1 Implementation

The code can be found in the following link: https://github.com/marvin-koch/RL-for-Sepsis

A.2 LSTM Encoder

A.2.1 Hyperparameters

- **minibatch_size**: 16
- **num_actions**: 2
- **context_dim**: 5
- **Actor Network Number of Layers**: 3
- **Critic Network Number of Layers**: 3
- **Replay Buffer Size**: 350000

- **Reward Discount Factor**: 0.99
- **Weight Update Parameter**: 0.01
- **Training Iterations**: 20000
- **Evaluation Iterations**: 5000
- **actor_learning_rate**: 3e-3
- **critic_learning_rate**: 3e-5

A.2.2 Further Plots

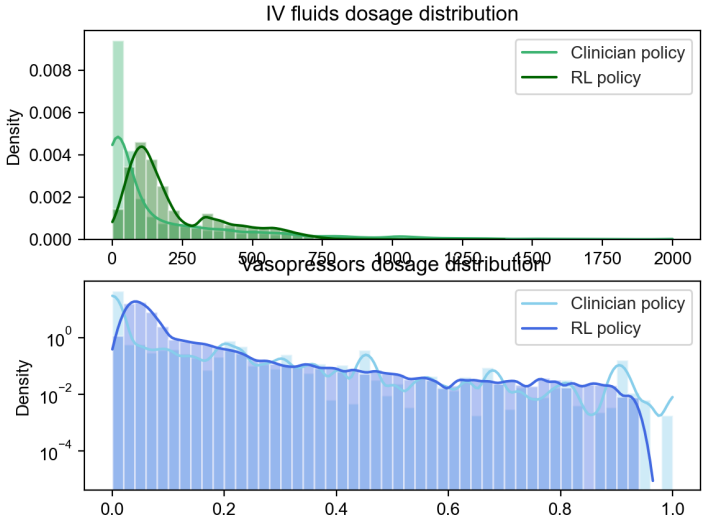

Figure 37: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the LSTM with 42 layers and the clinician.

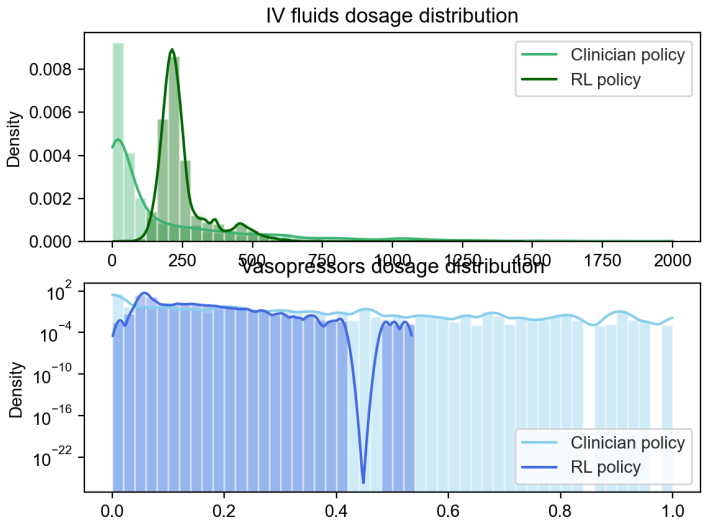

Figure 38: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the LSTM with 92 layers and the clinician.

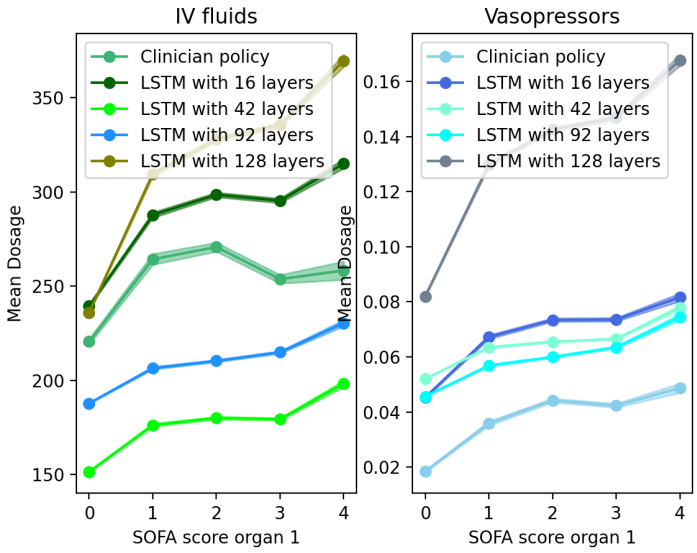

Figure 39: Mean Dosage per SOFA subscore 1 for different LSTM layers.

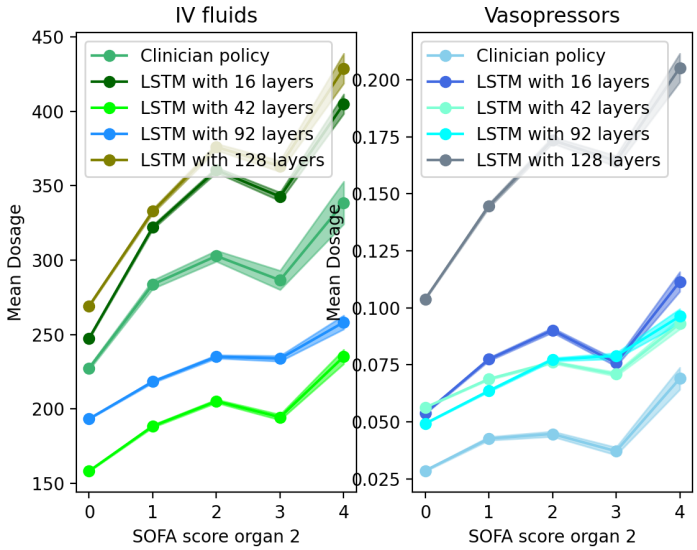

Figure 40: Mean Dosage per SOFA subscore 2 for different LSTM layers.

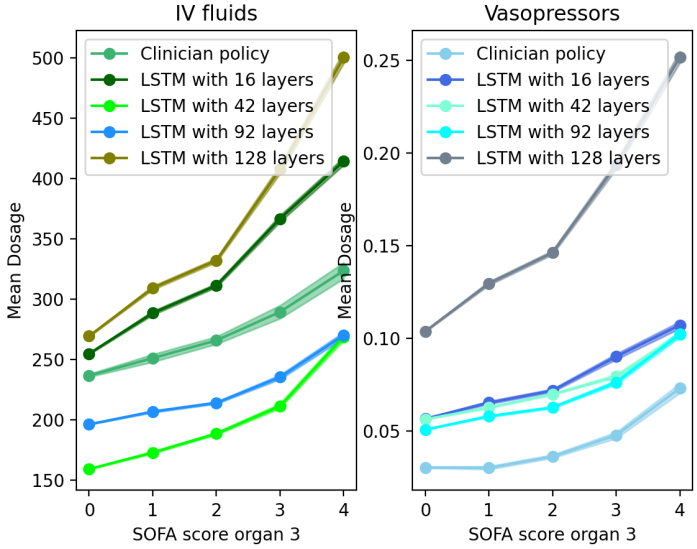

Figure 41: Mean Dosage per SOFA subscore 3 for different LSTM layers.

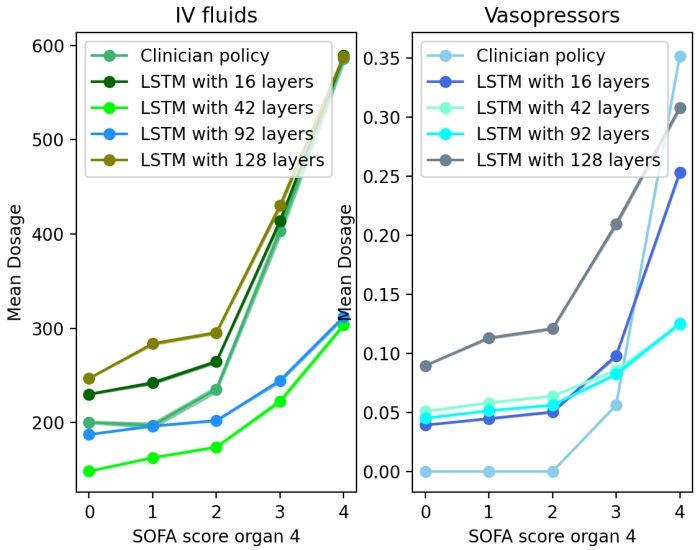

Figure 42: Mean Dosage per SOFA subscore 4 for different LSTM layers.

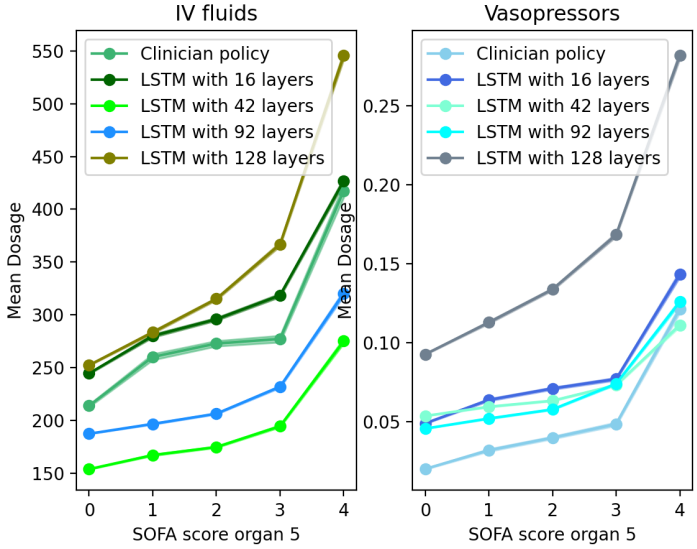

Figure 43: Mean Dosage per SOFA subscore 5 for different LSTM layers.

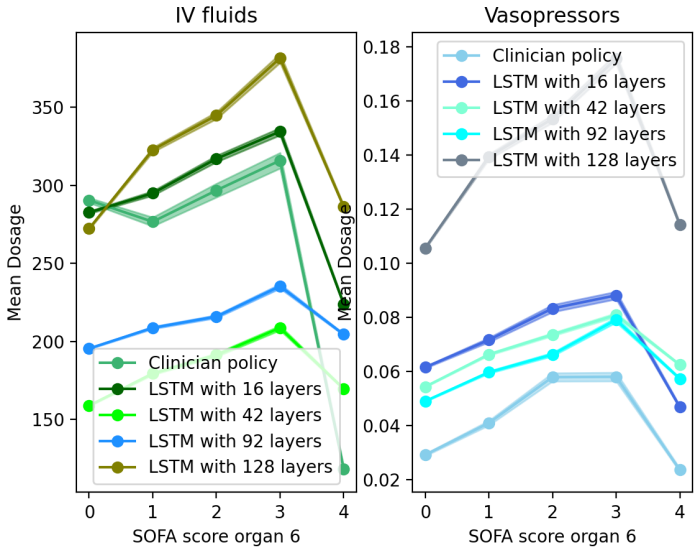

Figure 44: Mean Dosage per SOFA subscore 6 for different LSTM layers.

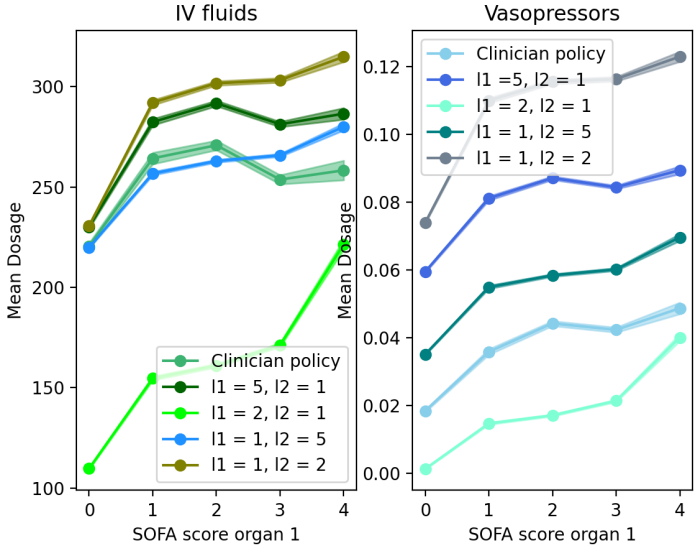

Figure 45: Mean Dosage per SOFA subscore 1 for different lambdas.

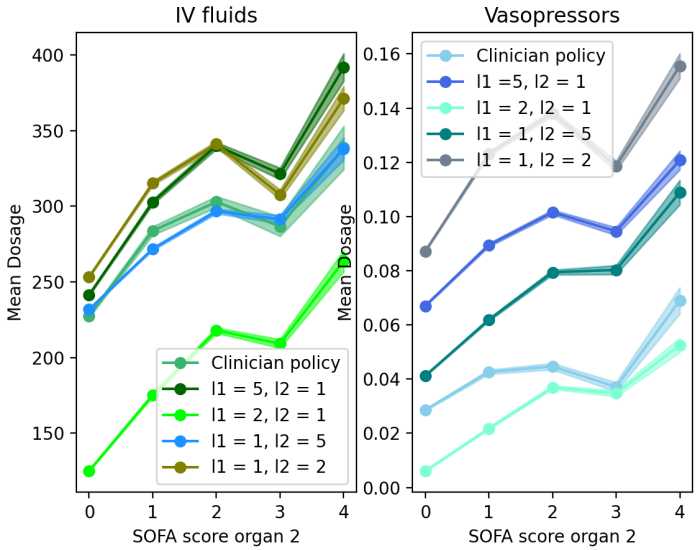

Figure 46: Mean Dosage per SOFA subscore 2 for different lambdas.

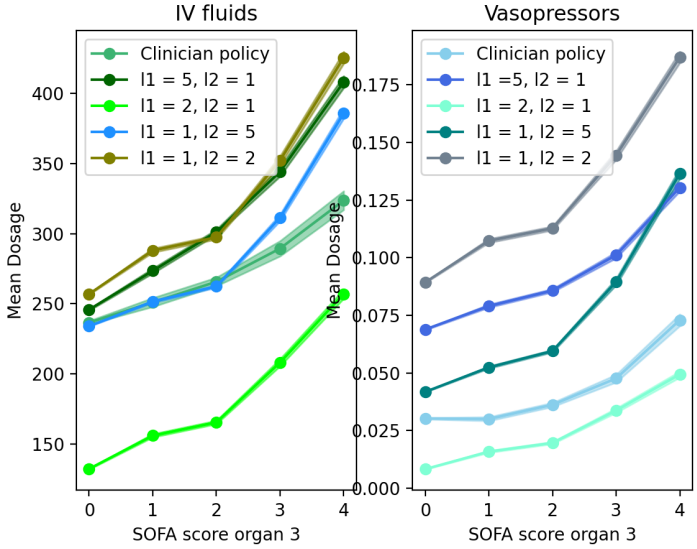

Figure 47: Mean Dosage per SOFA subscore 3 for different lambdas.

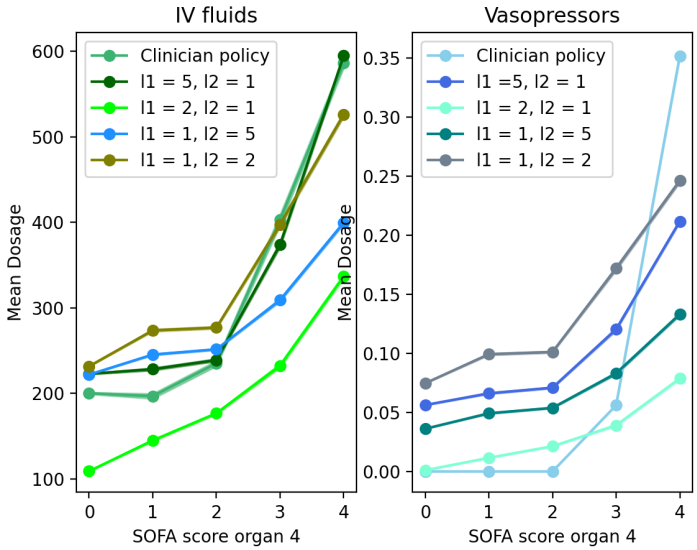

Figure 48: Mean Dosage per SOFA subscore 4 for different lambdas.

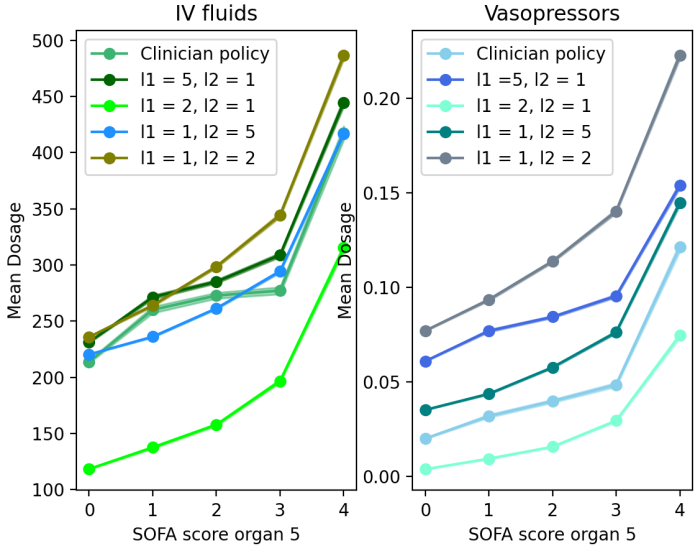

Figure 49: Mean Dosage per SOFA subscore 5 for different lambdas.

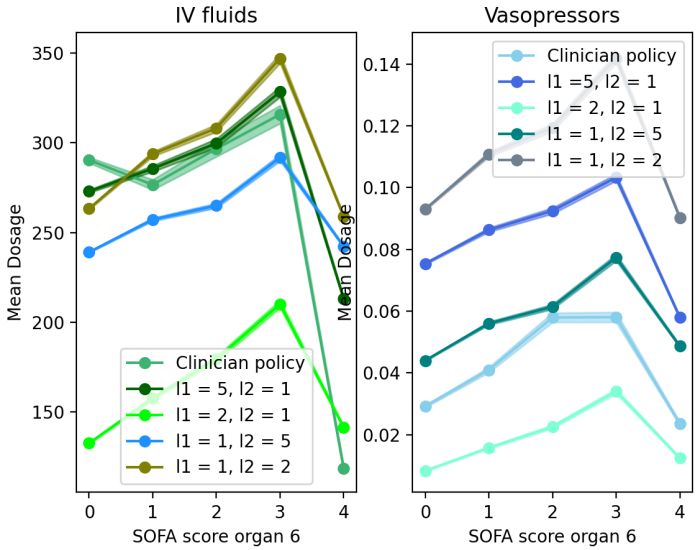

Figure 50: Mean Dosage per SOFA subscore 6 for different lambdas.

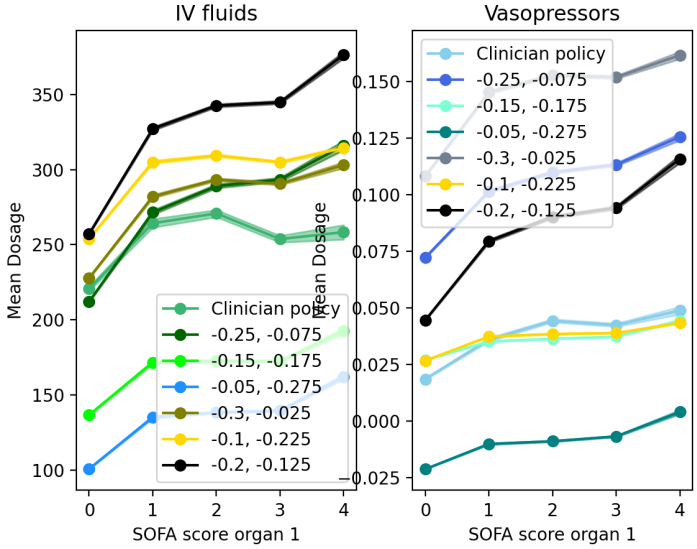

Figure 51: Mean Dosage per SOFA subscore 1 for different gammas.

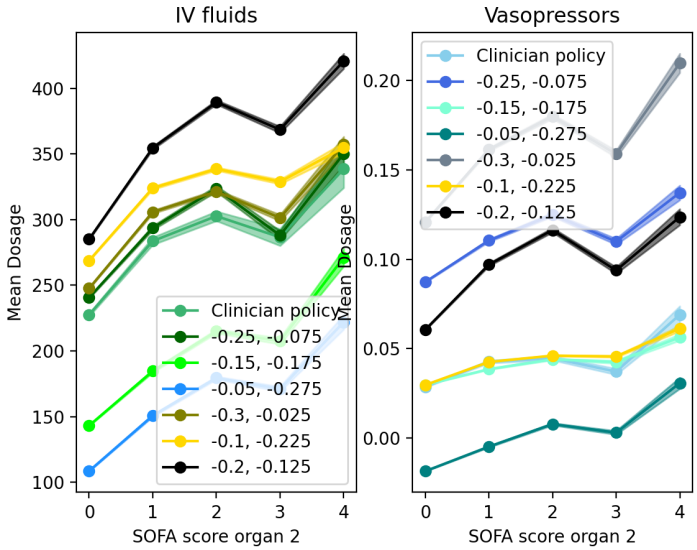

Figure 52: Mean Dosage per SOFA subscore 2 for different gammas.

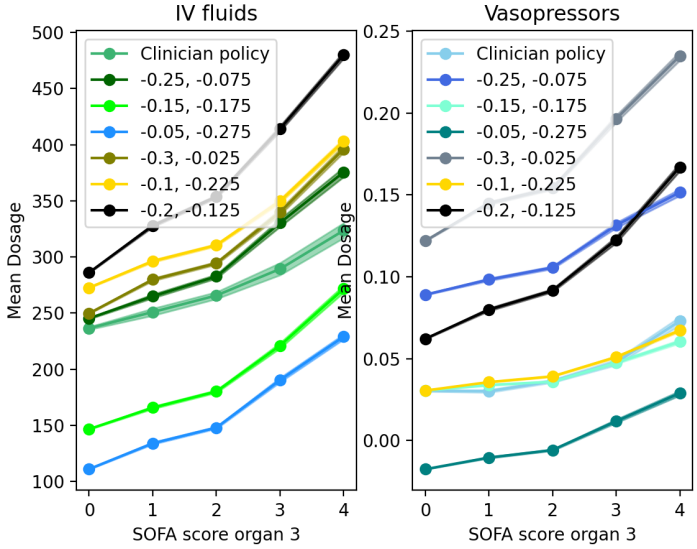

Figure 53: Mean Dosage per SOFA subscore 3 for different gammas.

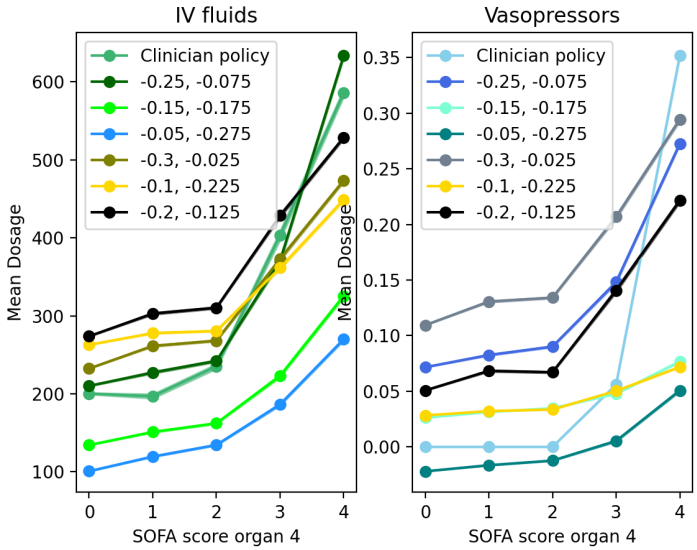

Figure 54: Mean Dosage per SOFA subscore 4 for different gammas.

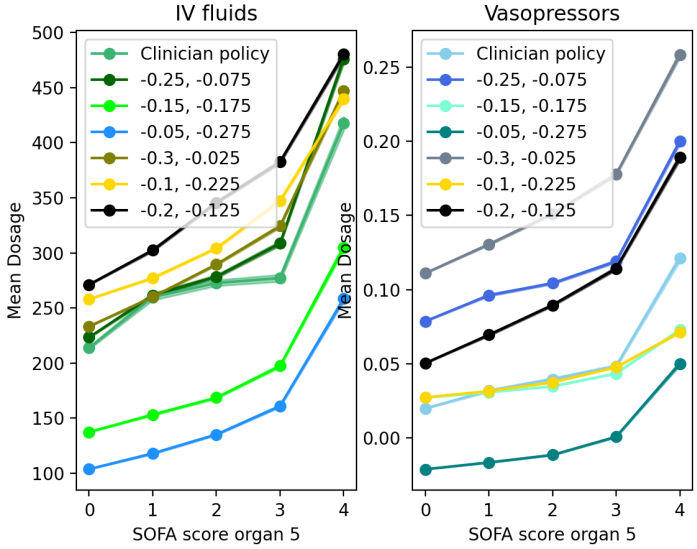

Figure 55: Mean Dosage per SOFA subscore 5 for different gammas.

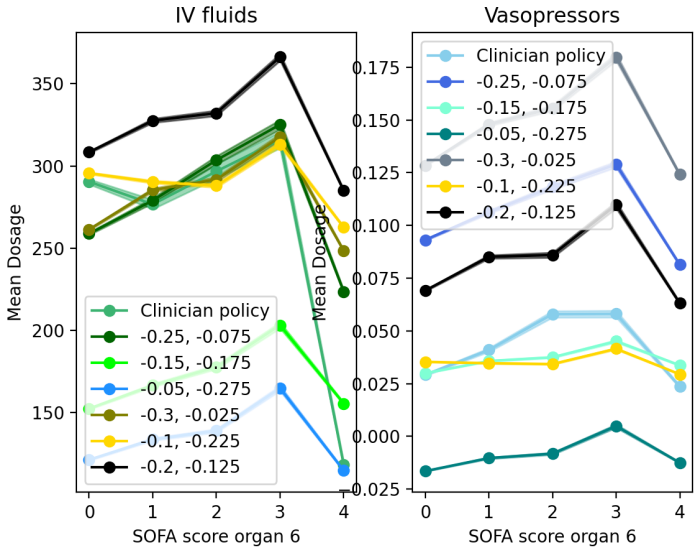

Figure 56: Mean Dosage per SOFA subscore 6 for different gammas.

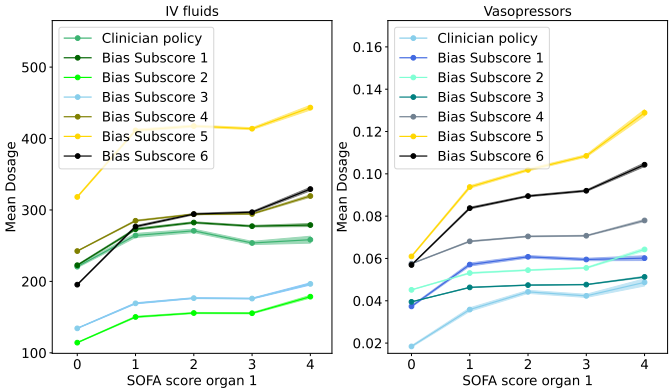

Figure 57: Comparison of mean dosage per SOFA subscore 1 for IV fluids and vasopressors between models (LSTM with 16 layers) with different subscore biases (biasing the whole reward function).

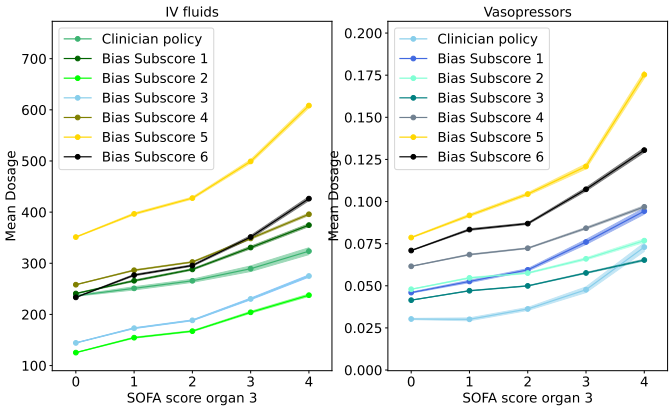

Figure 58: Comparison of mean dosage per SOFA subscore 3 for IV fluids and vasopressors between models (LSTM with 16 layers) with different subscore biases (biasing the whole reward function).

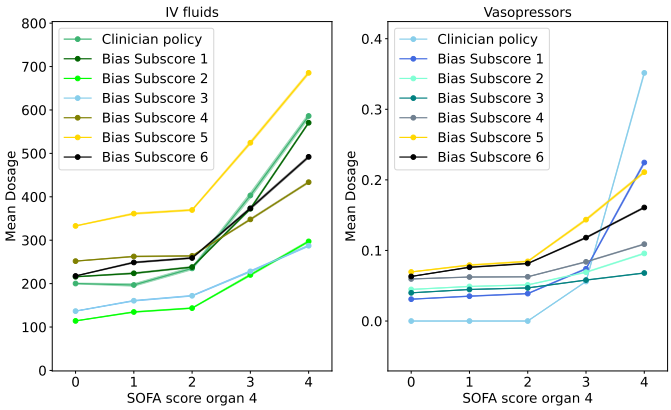

Figure 59: Comparison of mean dosage per SOFA subscore 4 for IV fluids and vasopressors between models (LSTM with 16 layers) with different subscore biases (biasing the whole reward function).

A.3 Transformer Encoder

A.3.1 Hyperparameters

- **Use Dropout**: True
- **minibatch size**: 16
- **context_dim**: 5
- **Replay Buffer Size**: 350000
- **Reward Discount Factor**: 0.99
- **Weight Update Parameter**: 0.01
- **Training Iterations**: 20000
- **Evaluation Iterations**: 5000
- **Sampled Sequence Length**: -1
- **Clip**: False
- **Max Norm**: 1.0
- **Use L2 Norm**: False

- **Model Sequence Hidden Size**: 128
- **Model Sequence Dropout Probability**: 0.1
- **Model Sequence Position Encoding**: sine
- **Observation Embedder Hidden Size**: 64
- **Action Embedder Hidden Size**: 64
- **Reward Embedder Hidden Size**: 0
- **Learning Rate Actor and Critic**: 3e-4

A.3.2 Further Plots

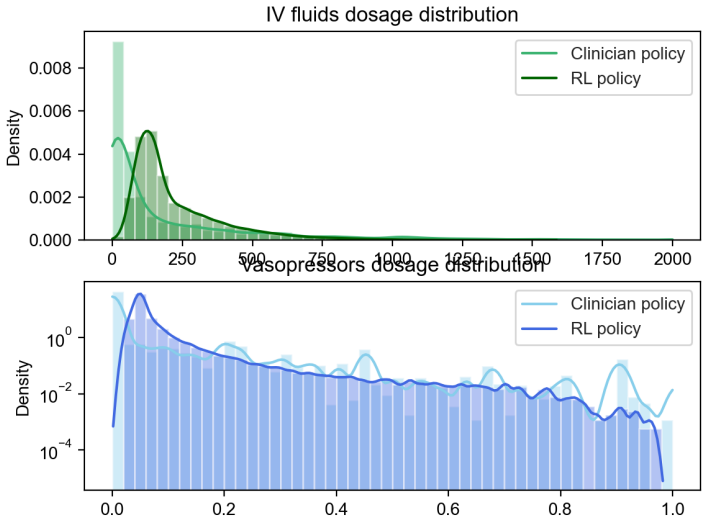

Figure 60: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the Transformer with 2 layers, 2 heads and the clinician.

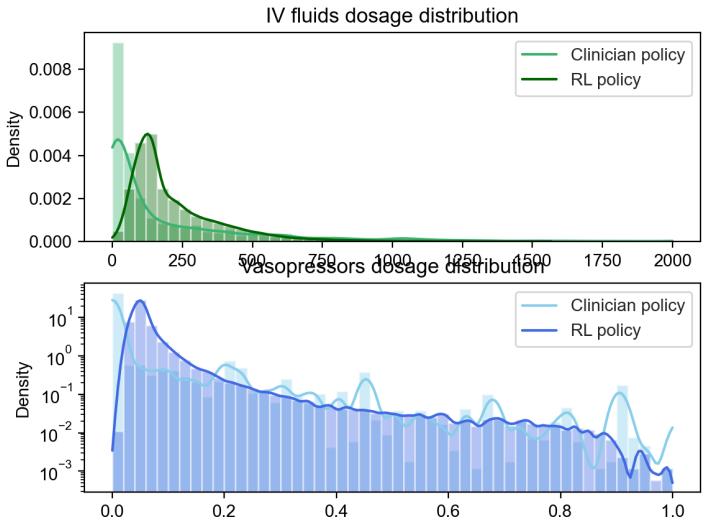

Figure 61: Comparison of IV fluid and vasopressor dosage distribution (log scaled) between the Transformer with 2 layers, 4 heads and the clinician.

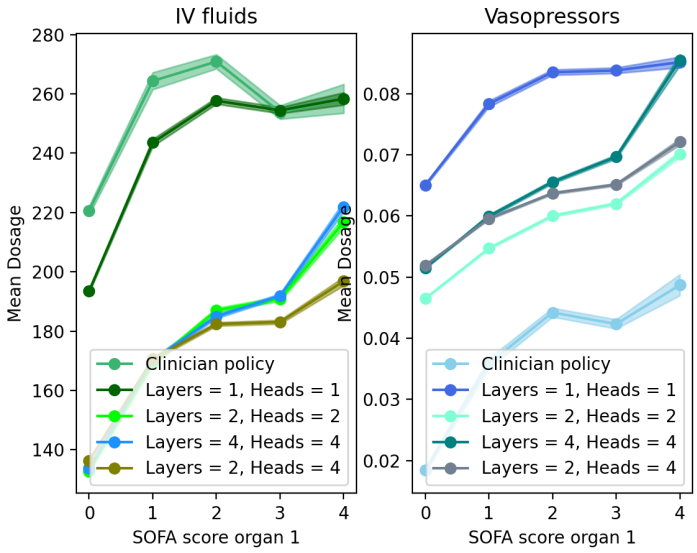

Figure 62: Mean Dosage per SOFA subscore 1 for different transformer heads and layers.

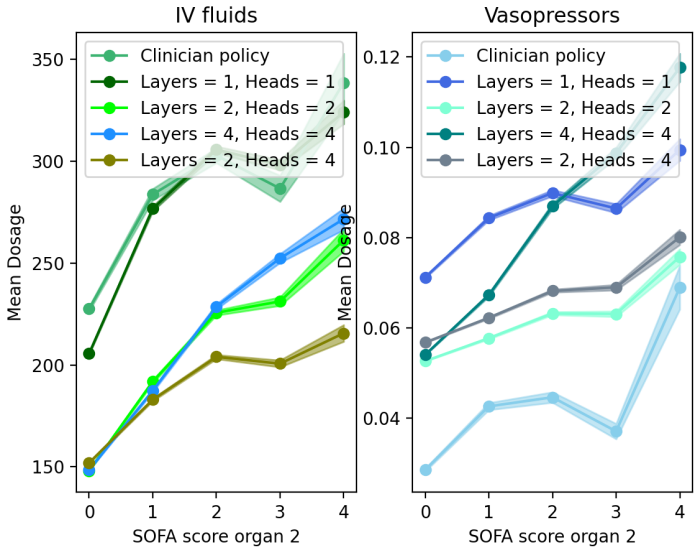

Figure 63: Mean Dosage per SOFA subscore 2 for different transformer heads and layers.

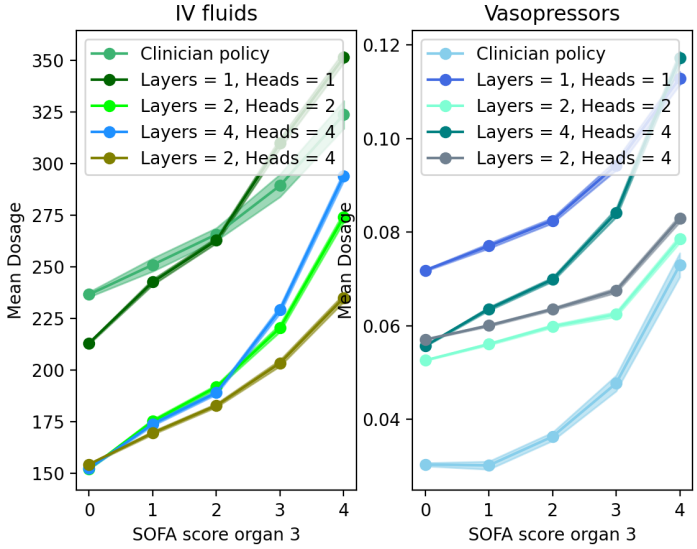

Figure 64: Mean Dosage per SOFA subscore 3 for different transformer heads and layers.

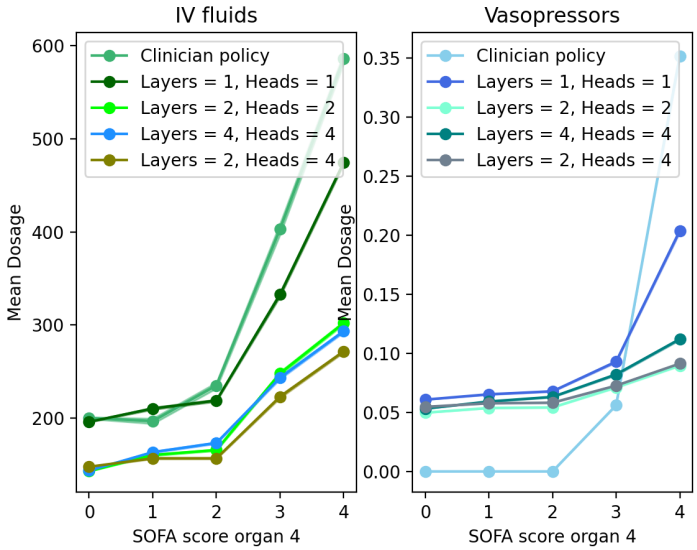

Figure 65: Mean Dosage per SOFA subscore 4 for different transformer heads and layers.

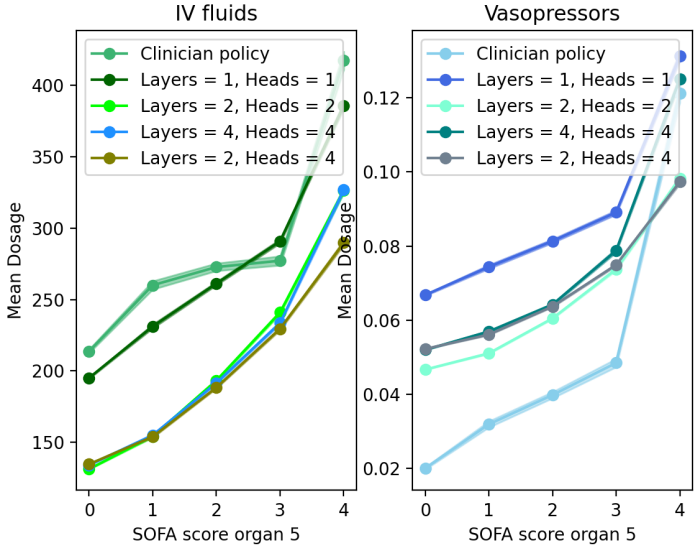

Figure 66: Mean Dosage per SOFA subscore 5 for different transformer heads and layers.

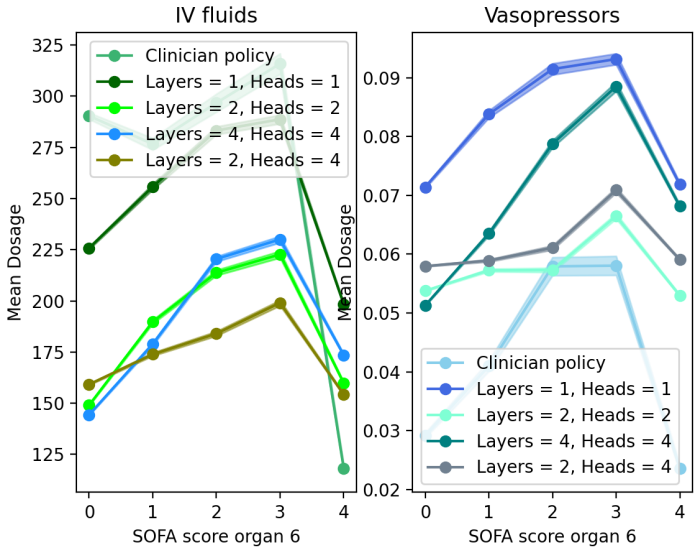

Figure 67: Mean Dosage per SOFA subscore 6 for different transformer heads and layers.

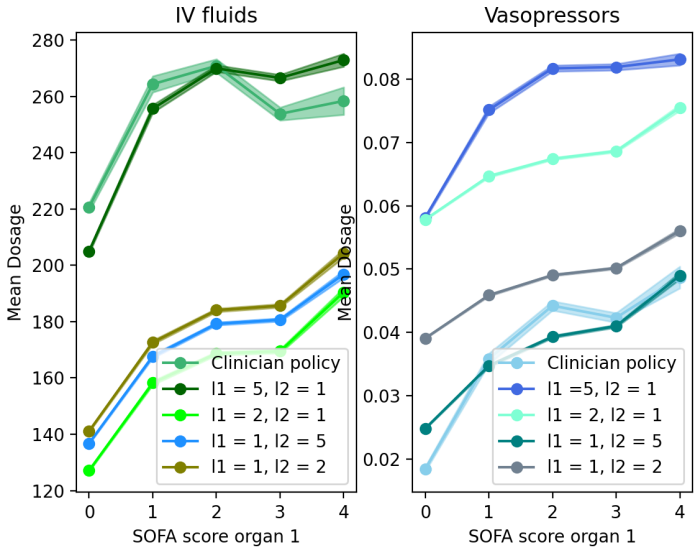

Figure 68: Mean Dosage per SOFA subscore 1 for different lambdas.

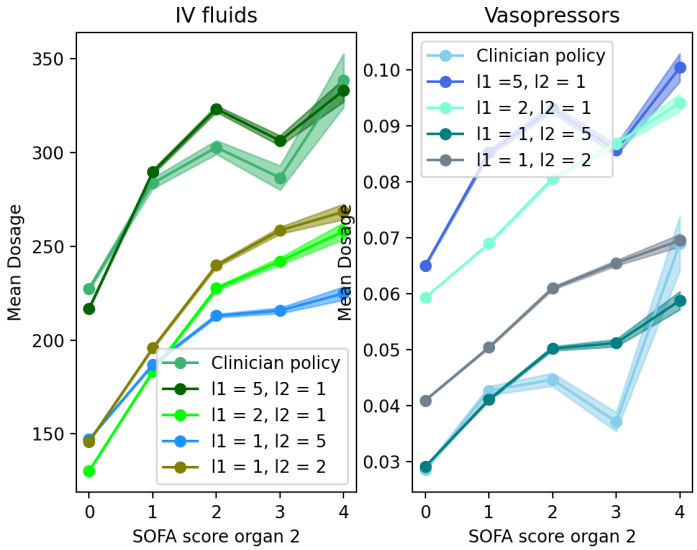

Figure 69: Mean Dosage per SOFA subscore 2 for different lambdas.

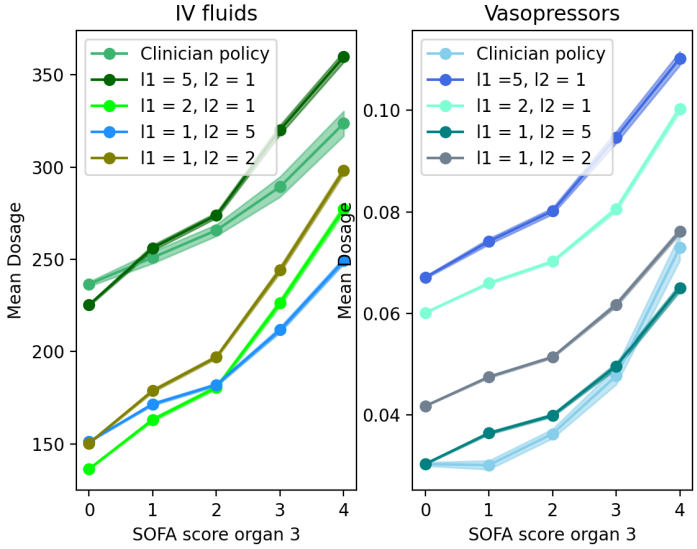

Figure 70: Mean Dosage per SOFA subscore 3 for different lambdas.

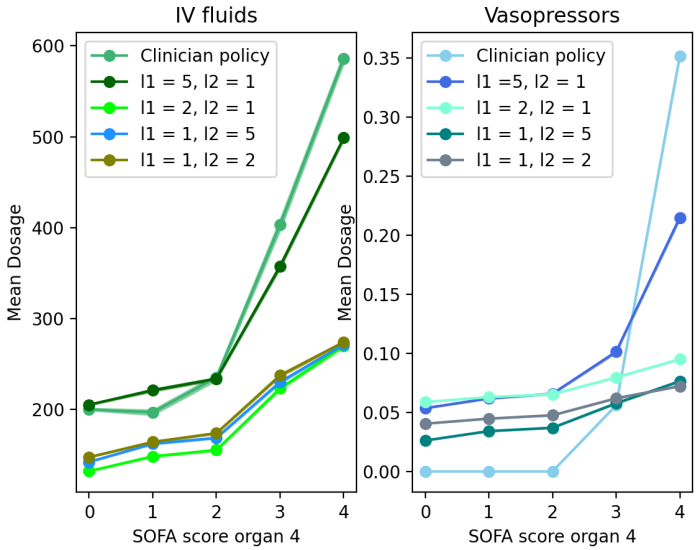

Figure 71: Mean Dosage per SOFA subscore 4 for different lambdas.

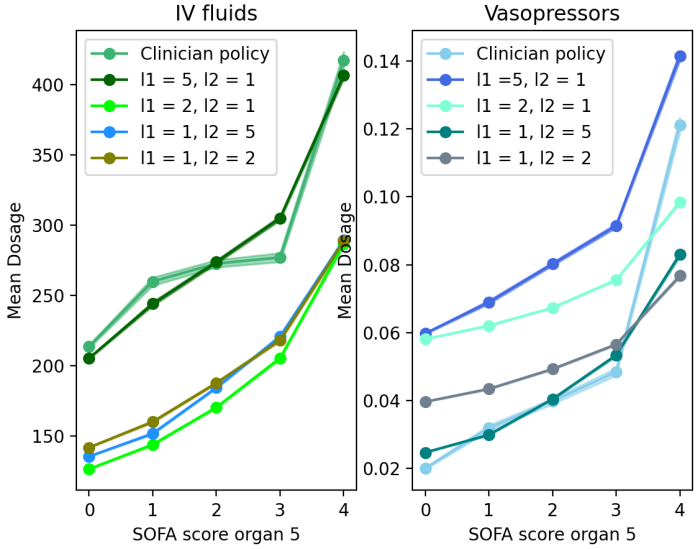

Figure 72: Mean Dosage per SOFA subscore 5 for different lambdas.

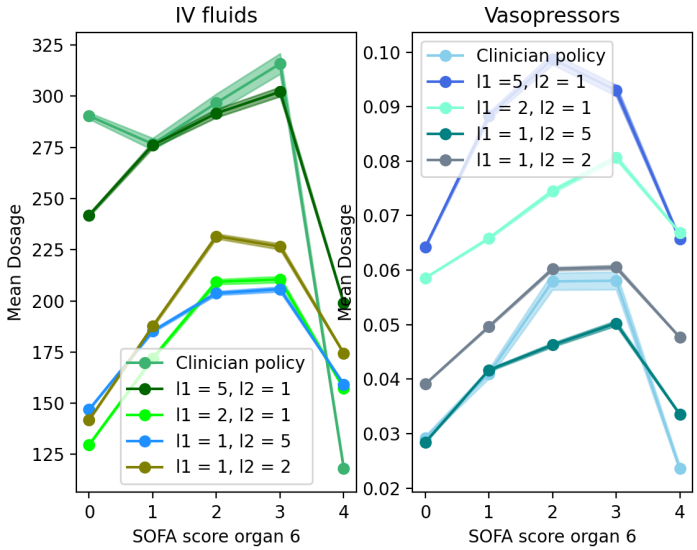

Figure 73: Mean Dosage per SOFA subscore 6 for different lambdas.

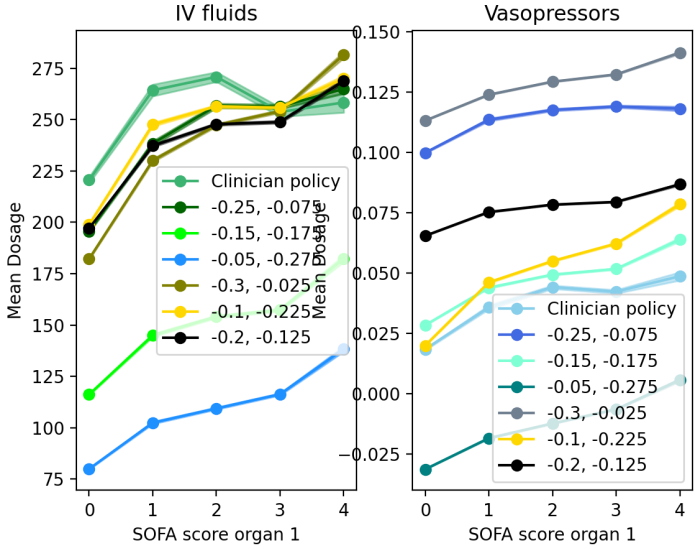

Figure 74: Mean Dosage per SOFA subscore 1 for different gammas.

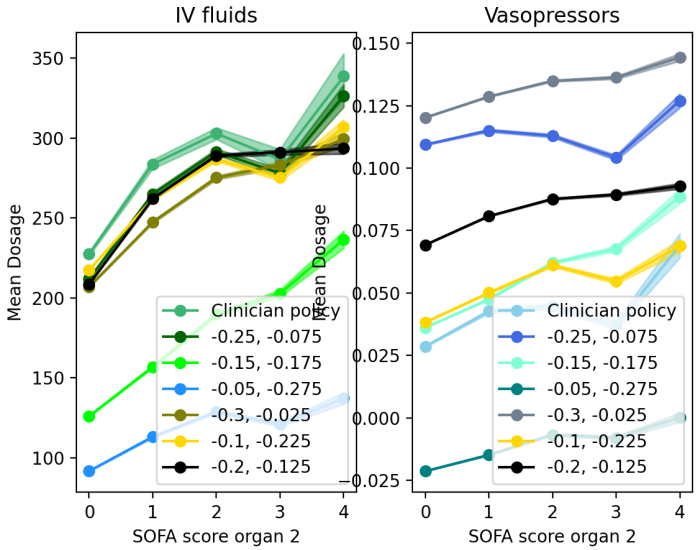

Figure 75: Mean Dosage per SOFA subscore 2 for different gammas.

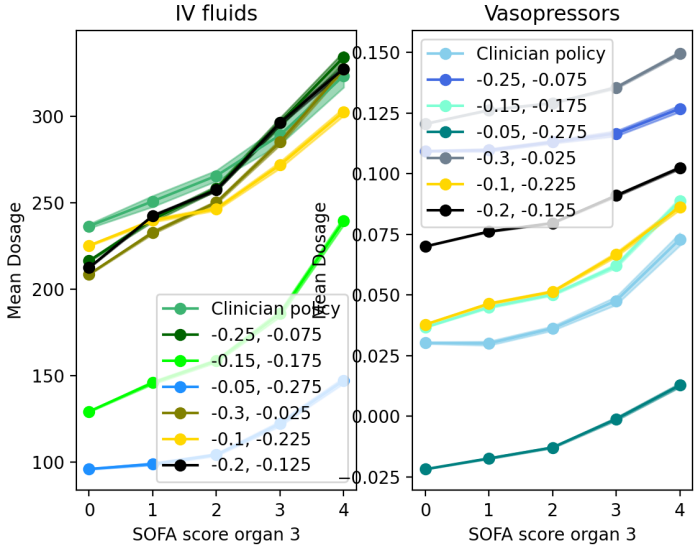

Figure 76: Mean Dosage per SOFA subscore 3 for different gammas.

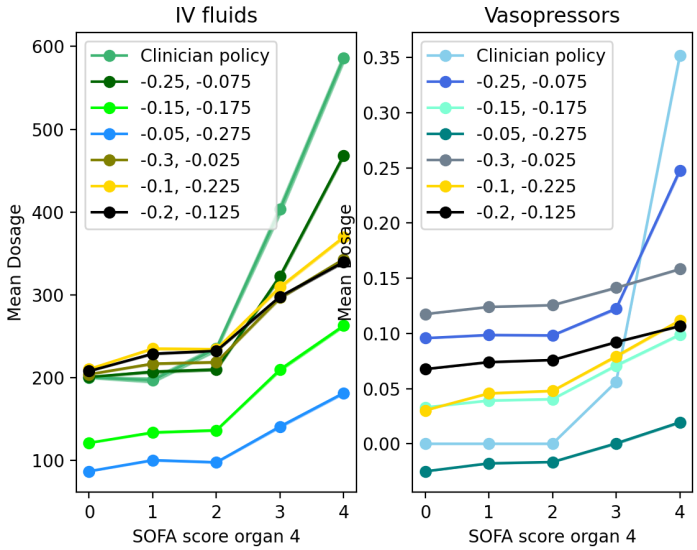

Figure 77: Mean Dosage per SOFA subscore 4 for different gammas.

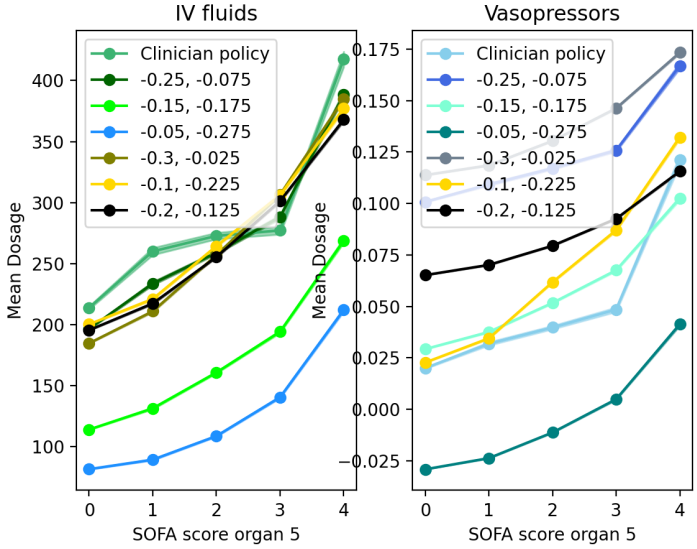

Figure 78: Mean Dosage per SOFA subscore 5 for different gammas.

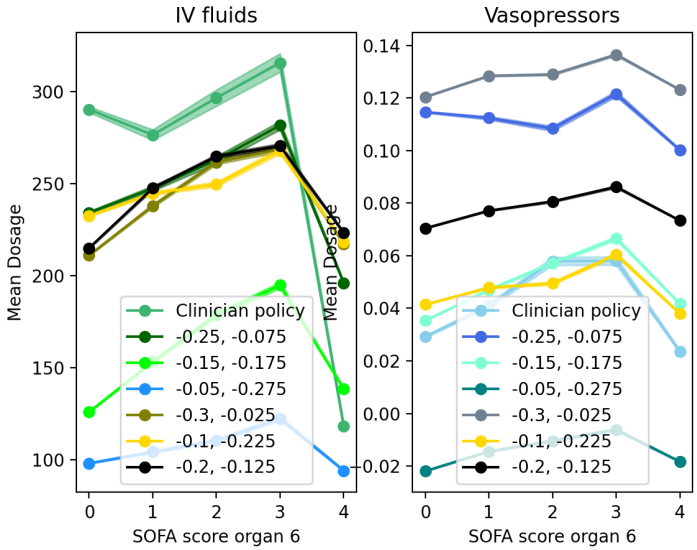

Figure 79: Mean Dosage per SOFA subscore 6 for different gammas.

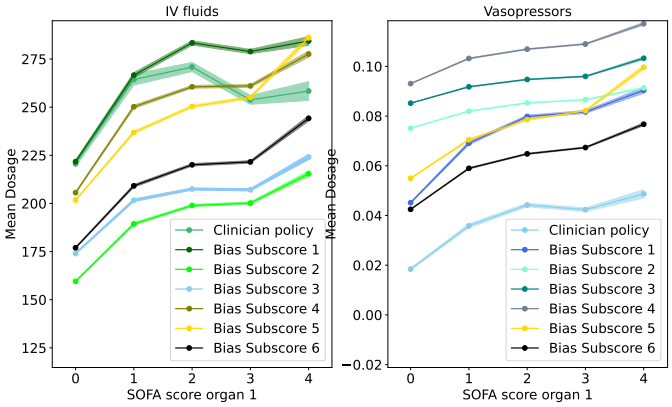

Figure 80: Comparison of mean dosage per SOFA subscore 1 for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

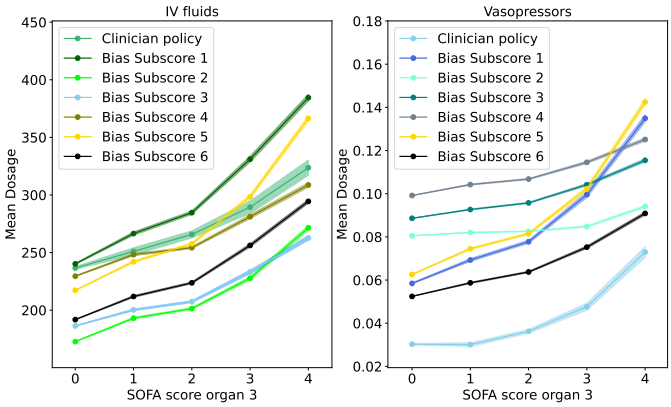

Figure 81: Comparison of mean dosage per SOFA subscore 3 for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

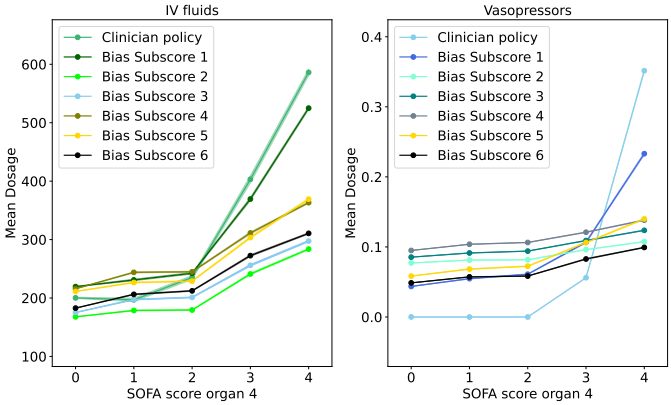

Figure 82: Comparison of mean dosage per SOFA subscore 4 for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

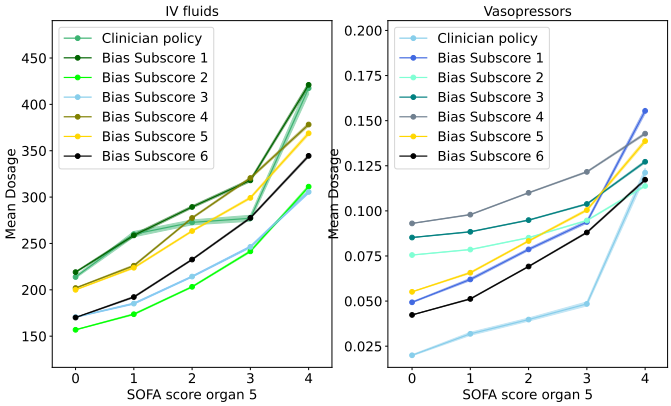

Figure 83: Comparison of mean dosage per SOFA subscore 5 for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

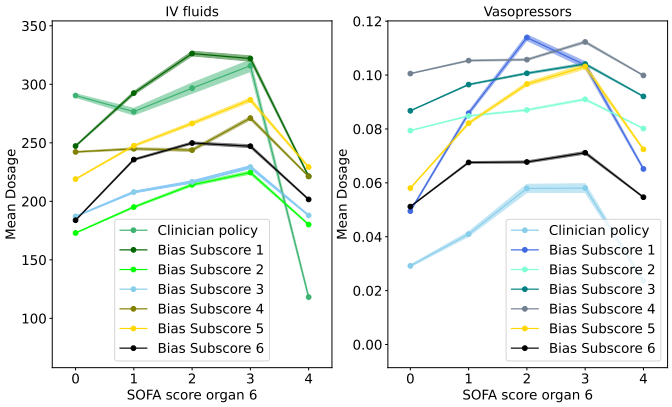

Figure 84: Comparison of mean dosage per SOFA subscore 6 for IV fluids and vasopressors between models (Transformer with 1 head, 1 layer) with different subscore biases (biasing the whole reward function).

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

◉ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Personalised Sepsis Treatment with Reinforcement Learning

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Koch	Marvin

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zürich, 25.06.2024	

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard