A Appendix

A.1 Experimental Settings

We provide the detailed model configurations and hyperparameter settings for DEEPCTRL in the following three experiments: double pendulum dynamics, sales forecasting, and cardiovascular classification.

A.1.1 Model Configurations

Double Pendulum The input and output states are 4 dimensional (two angular displacements and two angular velocities). The input state is fed into a shared layer whose configuration is [FC64,ReLU,FC16] where FC(n) denotes a fully-connected layer with n units. Then, output from the shared layer is fed into two encoders: Rule encoder [FC64,ReLU,FC64,ReLU,FC64] and Data encoder [FC64,ReLU,FC64,ReLU,FC64]. The combined representation from the two encoders is fed into a decision block [FC64,ReLU,FC4]. We use Adam optimization algorithm for training with learning rate 0.001.

Sales Forecasting The input dimension is 13, consisting of an item price as well as derivative features, and the output dimension is 1 (total weekly sales). Both encoders have same configuration [FC64,ReLU,FC64,ReLU,FC16] followed by a decision block [FC64,ReLU,FC1]. We use Adam optimization algorithm for training with learning rate 0.001.

Cardiovascular Classification The number of original input features is 11: AGE, HEIGHT, WEIGHT, GENDER, SYSTOLIC BLOOD PRESSURE, DIASTOLIC BLOOD PRESSURE, CHOLESTEROL, GLUCOSE, SMOKING, ALCOHOL INTAKE, PHYSICAL ACTIVITY. We expand categorical features to one-hot encoding (the input dimension is increased to 19) and the output dimension is 1 (Presence or absence of cardiovascular disease). Both encoders have same configuration [FC100,ReLU,FC16] followed by a decision block [FC1,Sigmoid]. We use Adam optimization algorithm for training with learning rate 0.001.

A.1.2 Data Splits

Double Pendulum The total length of simulated dynamics is 30,000 and it is split into training (18,000), validation (3,000), and testing (9,000).

Sales Forecasting Per the Kaggle task definition, first 273 weeks are used in a training set, the following 4 weeks are for a validation set, and the last 4 weeks are for a testing set. In each set, there are 22,543, 700, and 700 records from the preprocessed dataset.

Cardiovascular Classification The data partitioning is described in Section 4.3. For SOURCE partition, we have 70% training samples, 10% validation samples, and 20% testing samples to train and evaluate a model. Then, the trained model is applied to TARGET 1, TARGET 2, and TARGET 3, respectively.

Table 1: Dataset splits and USUAL vs. UNUSUAL partitioning.

DATA	SOURCE	TARGET 1	TARGET 2	TARGET 3
USUAL	6,007	20,000	6,000	4,000
UNUSUAL	14,018	6,009	6,009	6,009
RATIO	0.30	0.77	0.50	0.40

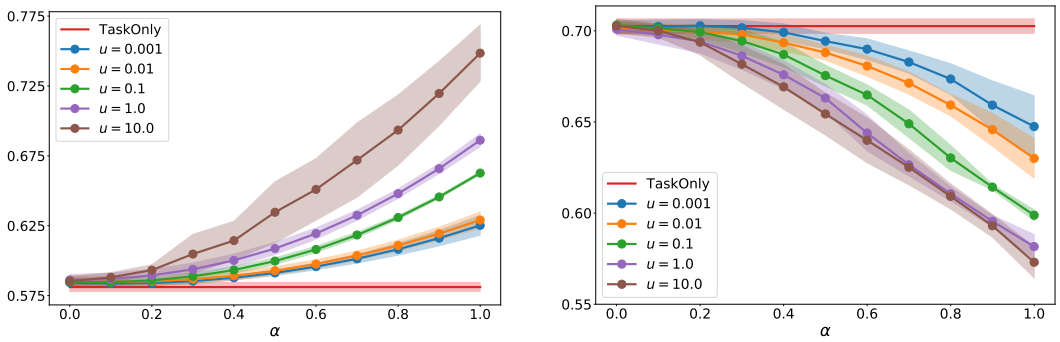
A.1.3 Training Settings

We commonly train DEEPCTRL for all tasks with a batch size of 32 on a single GPU (Nvidia T4 GPU) for 1000 epochs with early stopping where a validation error is not improved for 10 epochs. All results in the paper are mean values from 10 different random seeds.

A.2 Perturbations

A.2.1 How to set $\delta \boldsymbol{x}$?

In Section 3, we provide a method to integrate non-differentiable $\mathcal{L}_{rule}$ via input perturbations. In this paper, we use the rules obtained via perturbation-based method for two tasks: sales forecasting (Section 4.2) and cardiovascular classification (Section 4.3). We summarize how the perturbations are generated and used to define



(a) Cross Entropy (b) Accuracy

Figure 8: (Left) Cross entropy and (Right) Accuracy vs. rule strength for various upper bounds of perturbation scale.

the corresponding rule-based constraint for each task. Note that both tasks have a similar non-differentiable form of rule-constraint that the input $\boldsymbol{x}$ and output $\boldsymbol{y}$ have a negative or positive correlation.

Perturbations in Sales Forecasting There are a number of weekly-sales-records per an item at a particular store. For each record (week t), we have input features including the price of an item $\boldsymbol{x}_t$ and the target sales $\boldsymbol{y}_t$. The correlation coefficient between $\boldsymbol{x}$ and $\boldsymbol{y}$ is:

$$R = \frac{\sum_{t=1}^{T}(\boldsymbol{x}_t - \overline{\boldsymbol{x}})(\boldsymbol{y}_t - \overline{\boldsymbol{y}})}{\sqrt{\sum_{t=1}^{T}(\boldsymbol{x}_t - \overline{\boldsymbol{x}})^2}\sqrt{\sum_{t=1}^{T}(\boldsymbol{y}_t - \overline{\boldsymbol{y}})^2}}, \tag{4}$$

where $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$ are the sample mean of $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$, respectively.

The rule-based constraint we want to impose is *price and sales should have a negative correlation coefficient*, i.e. $R < 0$. While the constraint is based on the exact definition of the correlation coefficient over T samples (Eq. 4), we use a constraint based on individual sample instead:

$$\frac{\Delta \boldsymbol{y}}{\Delta \boldsymbol{x}} = \frac{\boldsymbol{y}_p - \boldsymbol{y}}{\boldsymbol{x}_p - \boldsymbol{x}} < 0, \tag{5}$$

where $\Delta \boldsymbol{x}$ is a price-difference and $\Delta \boldsymbol{y}$ is a sales-difference, respectively, and $\boldsymbol{x}_p$ is a perturbed price where $\boldsymbol{x}_p = \boldsymbol{x} + \delta \boldsymbol{x}$ and $\boldsymbol{y}_p$ is an output from the perturbed price. Note that Eq. 5 is identical to $\boldsymbol{y}_p < \boldsymbol{y}$ once $\delta \boldsymbol{x} > 0$. There are two reasons not to use Eq. 4 directly. First, it is costly to compute the coefficient at every iteration. Second, it is possible to control $\delta \boldsymbol{x}$ in Eq. 5. We set $\delta \boldsymbol{x} = \gamma |\boldsymbol{x}|$, where $\gamma \sim \mathrm{U}[0, u]$ such that γ is a perturbation scale parameter and u is an upper bound of γ. Thus, the magnitude of $\delta \boldsymbol{x}$ is bounded by $[0, u|\boldsymbol{x}|)$.

Perturbations in Cardiovascular Classification Similarly, we impose a positive correlation between blood pressure $\boldsymbol{x}$ and a risk of the cardiovascular disease $\boldsymbol{y}$:

$$\frac{\Delta \boldsymbol{y}}{\Delta \boldsymbol{x}} = \frac{\boldsymbol{y}_p - \boldsymbol{y}}{\boldsymbol{x}_p - \boldsymbol{x}} > 0. \tag{6}$$

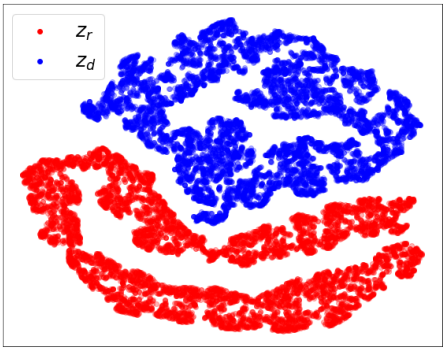
Eq. 6 is analogous to $\boldsymbol{y}_p > \boldsymbol{y}$ as long as the perturbation $\delta \boldsymbol{x} > 0$.

Upper Bound u: We set the upper bound u as 0.1 for both tasks via the analysis described in Section A.2.2.

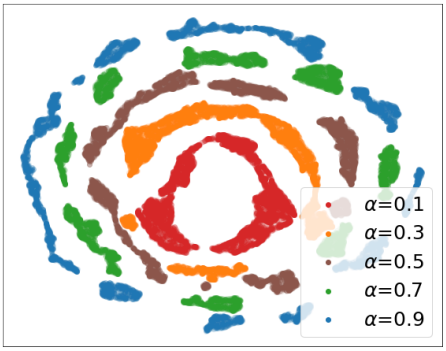
A.2.2 Impact of Perturbation Scale

In previous section, we define $\delta \boldsymbol{x}$ as a random scalar that is upper-bounded by $u|\boldsymbol{x}|$. We experiment on the cardiovascular classification task with different upper bounds, u to see how the scale of perturbation affects the model's behavior.

Fig 8 shows how the cross entropy and accuracy are changed as the rule strength is changed. When the upper bound u is increased, the perturbation scale γ can be also increased, and thus, it leads to generate larger perturbations $\delta \boldsymbol{x}$. As u increases, the performance of a classifier is degraded when rule strength is non zero ($\alpha > 0$). The curves imply that larger u leads more degraded performance when the rule strength is higher. When $\delta \boldsymbol{x}$ increases, the perturbed output $\boldsymbol{y}_p$ is more different to $\boldsymbol{y}$ as $\boldsymbol{y}_p$ is a function of $\boldsymbol{x} + \delta \boldsymbol{x}$. According to Eq. 5 and 6, the rule-based objective $\mathcal{L}_{rule}$ is a function of $\boldsymbol{y}_p - \boldsymbol{y}$, and thus, the larger $\delta \boldsymbol{x}$ eventually causes larger $\mathcal{L}_{rule}$. In other words, as $\mathcal{L}_{rule}$ increases, DEEPCTRL is more driven by the rule-based objective when α is non zero and thus, the task-based performance is degraded. As discussed, it is desired to have



(a) t-SNE mapping of z_r and z_d.



(b) t-SNE mapping of $z = \alpha z_r + (1 - \alpha z_d)$ over different α.

Figure 9: (Left) t-SNE visualization of z_r and z_d from rule encoder ϕ_r and data encoder ϕ_d, respectively, from the double pendulum task. z_r and z_d from 9,000 test samples do not have overlapped representations and it implies that two representations are distinct. (Right) t-SNE visualization of $z = \alpha z_r + (1 - \alpha) z_d$ over different α. It shows that task-/rule-specific representations are placed in inner/outer space, respectively. Note that the proposed representations gradually interpolate two extremes rather than being abruptly crossed.

Table 2: Training time (in seconds) for Sales forecasting (Retail) and Cardiovascular classification (Healthcare).

DATA	TASKONLY (TOTAL)	DEEPCTRL (TOTAL)	TASKONLY (PER EPOCH)	DEEPCTRL (PER EPOCH)
RETAIL	362.7±62.3	328.2±91.0	2.31±0.04	2.32±0.01
HEALTHCARE	154.7±11.0	168.1±23.1	3.33±0.24	3.45±0.17

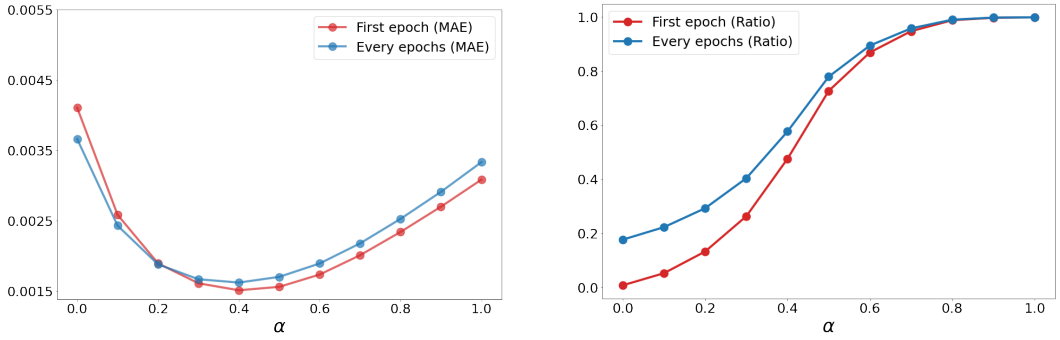
distinct model's behavior when $\alpha = 0$ and $\alpha = 1$ and thus, too small perturbation less incorporate rule-based representations. However, if too large perturbations are considered, the model is dominated by the rule mostly and the performance can be worse when α is close to 0 (the brown curve is slightly higher than others when $\alpha \to 0$ in Fig. 8a).

A.3 Visualization of z_r, z_d and z

In this section, we analyze the learned representations to demonstrate the rule vs. data disentanglement capability of DEEPCTRL. We first visualize what the rule encoder ϕ_r and data encoder ϕ_d learn to support that the two encoders actually handle distinct representations rather than similar or overlapped representations. Then, we show how $\boldsymbol{z} = \alpha \boldsymbol{z}_r + (1 - \alpha)\boldsymbol{z}_d$ is changed over different α to show meaningful combined representations with varying rule strength. Fig. 9 demonstrates t-SNE mapping of $\boldsymbol{z}_r, \boldsymbol{z}_d$, and $\boldsymbol{z}$, respectively. The learned representations of $\boldsymbol{z}_r$ and $\boldsymbol{z}_d$ are observed to be separated, while their linear combinations are clustered together with an orientation of the clusters highly dependent on the rule strength.

A.4 Computational Complexity

Compared to TASKONLY training, the proposed method (DEEPCTRL) does not cause any additional computations that are proportional to the sample size. However, if perturbations are included, teaching rules via perturbation-based method cause an additional computation. For each batch, it is required to compute both the main forward pass ($\boldsymbol{x}$ to $\boldsymbol{y}$) and the perturbation-based forward pass ($\boldsymbol{x}_p$ to $\boldsymbol{y}_p$). Thus, the time complexity of DEEPCTRL is $2C_{forward} + C_{backward}$ and that of TASKONLY is $C_{forward} + C_{backward}$ where $C_{forward}$ is the time complexity of forward propagation and $C_{backward}$ of backward propagation. As $C_{backward}$ takes the majority of training time, the extra computations from the perturbation-based method is not significant. Furthermore, since the computational complexity is independent on the size of samples, DEEPCTRL is still scalable. Table 2 shows the training time from TASKONLY and DEEPCTRL over 10 repeats. Note that the running time per an epoch from DEEPCTRL is similar to that of TASKONLY and it proves that the computational complexity of DEEPCTRL is not significantly increased.



(a) MAE

(b) Verification ratio

Figure 10: Results on a double pendulum task (different ρ).

A.5 Adaptive loss combination:

To balance the contributions from $\mathcal{L}_{task}$ and $\mathcal{L}_{rule}$, we propose to use the scale parameter $\rho = \mathcal{L}_{rule,0}/\mathcal{L}_{task,0}$. One potential issue with this proposal is that at the beginning of training, the model is far from convergence, and the initial estimates of loss values may not be representative. One straightforward idea to address could be adapting ρ as the ratio after every epoch. In Fig. 10, we compare this approach to the proposed $\rho = \mathcal{L}_{rule,0}/\mathcal{L}_{task,0}$ and we indeed show that the results are quite similar, and indeed $\rho = \mathcal{L}_{rule,0}/\mathcal{L}_{task,0}$ yields better decoupling of $\alpha = 0$ cases (as evident from 0 verification ratio).

We attribute this to the fact that the scale mismatching mostly happens at early phase of training, and $\mathcal{L}_{rule}$ and $\mathcal{L}_{task}$ become rapidly very small and the parameters are not significantly changed. This property is dependent on the type of rule, type of task, and dataset so that it is necessary to choose a proper method beforehand. Having a fixed coefficient is particularly beneficial for stable training because the model has a fixed target, rather than a varying one.