A. Appendix

A.1. Ablation Study

Role of Number of Observations (J) The number of observations J corresponds to the number of partial observations that we have of a functions f^k. Ideally, we only need two such observations to learn the representations via contrastive objective. However, it has been shown that having more positive pairs result in learning better representations (Chen et al., 2020; Tian et al., 2019).

It should be noted that in our setting, the number of observed context sets N do not necessarily correspond to the number of observations J. This is because for some datasets, we aggregate the context points to get one partial observation (see Figure 2). This is important for the simple 1D and 2D regression functions where a single context point does not provide much information, hence the individual partial observations need more than one context point. In our setting, we treat J as a hyperparameter whose optimal value varies depending on the function. For instance, the MPI3D scene dataset has only 3 views per scene, therefore J can not be greater than 3 and we keep it fixed. For 1D and 2D regression functions, we observe that for a fixed number of context points N, the optimal number of observations J varies. For understanding the role of J better in these experiments, we perform an ablation study on MNIST2D dataset with FSCC (few-shot content classification) as the downstream task Figure 10. For each hyper-parameter configuration we train three models, initialized with different random seeds. The maximum number of context points is fixed to 200 while the J varies between 2 and 40.

It can be seen that the smaller values of J do not result in better FSCC score, however, the accuracy seems to plateau after $J = 10$. Therefore, we fix it to 10 for MNIST2D. For RLScene dataset, we fixed the maximum number of context points to be 20 and found the optimal number of partial observations to be $J = 4$. Note that the FSCC accuracy seems to be more influenced by the hyperparameters of critic and temperature, shown concurrently in the Figure 10. We discuss their roles in the next section.

Role of Critics and Temperature τ. We regard the discriminative scoring functions, including the projection heads, as critics. The simplest critic function does not contain any projection layer, regarded as *dot product* critic, where the contrastive objective is defined directly on the representations returned by the base encoder h_Φ. However, recently the role of critics in learning better representations has been explored in depth (Oord et al., 2018; Chen et al., 2020). Building on these findings, we evaluate the role played by different critics in learning the functions representations. Figure 10 shows the ablation for three different critics on MNIST2D validation dataset. It can be seen that the performance of critics is also highly linked with the temperature parameter τ. For an optimal temperature value τ, the non-linear critic performs consistently better.

Such hyperparameter grid search (done for MNIST2D) is very expensive for the ablation studies on the bigger datasets, such as, the MPI3D and RLScenes datasets. Therefore, we define the range for the t and perform a random sweep of 80 models with randomly selected hyper-parameter values for critic and temperature on MPI3D dataset. We did not find any pattern for the effect of temperature τ on the downstream tasks, however the pattern emerged for the class of critics. Figure 12 shows the ablation for critics on MPI3D dataset. It can be seen that the non-linear critic performs better in extracting features which are useful for the downstream classification tasks. Because of this trend across two different datasets, we therefore performed all our experiments with non-linear critic. The project head in nonlinear critics is defined as an MLP with one hidden layer and batch normalization in between.

B. Robustness to Noise

Functions with additive noise have been well-studied, however, the contemporary literature in meta-learning mostly considers them to be noise free. In this work, we explore whether the learned representations of the functions are prone to the additive Gaussian noise. We consider the form of the function as given in Equation (1) and vary the standard deviation of the added noise. It can be seen that with the increased level of noise the features in the image start to diminish, shown in the Figure 11. GQN approach the learning problem by reconstructing these noisy images where the signal to noise ratio is very low. On the other hand, FCRL learns to contrast the scene representations with other scenes without requiring any reconstruction in the pixel space. This helps it in extracting invariant features in the views of a scene, getting rid of any non-correlated noise in the input. In addition to the analysis on MNIST 2D regression task in Figure 4, we test the performance of these representations learning algorithm on MPI3D factors identification task in Figure 13. It can be seen that FCRL representations can recover the information about the data factors, even in the extreme case where the noise level is very high (standard deviation of 0.2). Whereas, GQN performs very poorly such that the downstream probes achieve the random accuracy.

C. Estimating Density Ratios corresponding to the Functions

The contrastive objective in Equation (4), in essence, tries to solve a classification problem i.e. to identify whether the given observation O^i comes from the function f^i or not. The supervision signal is provided by taking another

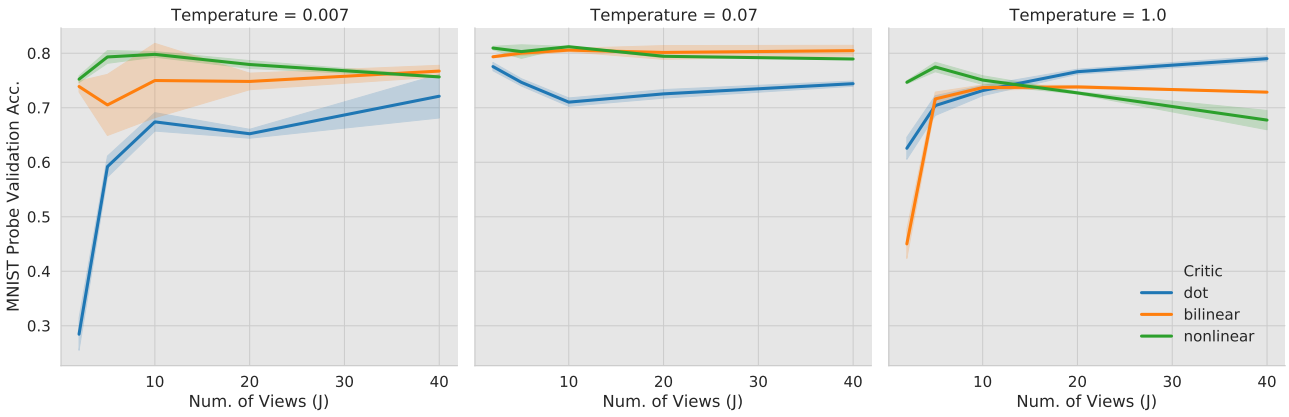


Figure 10. An ablation study for understanding the role of the number of partial observations (J), the critics and the temperature parameter (τ) for learning FCRL based representation. The graph shows the accuracy achieved by the linear classifier (Few-shot content classification task) on MNIST digits validation dataset. Note that each image corresponds to a 2D functions, and the accuracy bands show the standard deviation of three independent runs.

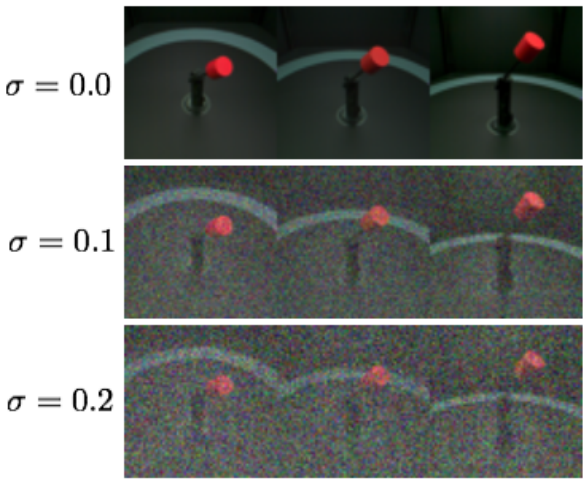


Figure 11. MPI3D dataset with the varying level of additive Gaussian noise.

observation $\hat{O}$ from the same function f^i as an anchor (a target label), thus making it a self-supervised method. This self-supervised, view-classification task, for a function f^i, leads to the estimation of density ratios between the joint distribution of observations $p(O^1, O^2|i)$ and their product of marginals $p(O^1|i)p(O^2|i)$. This joint distribution in turn corresponds to the joint distribution of the input-output pairs of the function $p(x, y|i)$. This way of learning a function's distribution is different from the typical regression objectives, which learn about a given function f^i by trying to approximate the predictive distribution $p(y|x)$.

By assuming the universal function approximation capability of $g_{(\phi,\Phi)}$, and the availability of infinitely many functions $f^k \sim p(f)$ with fixed number of context points N each, the model posterior learned by the optimal classifier corresponding to Equation (4) would be equal to the true posterior given by Bayes rule.

$$f^k \sim P(f) \quad \forall \ k \in \{1, .., K\} \tag{9}$$

$$O^k \sim P(O|f^k) \quad \forall k \in \{1, .., K\} \tag{10}$$

$$i \sim \mathcal{U}(K) \tag{11}$$

$$\hat{f} = f^i \tag{12}$$

$$\hat{O} \sim P(O|\hat{f}) \tag{13}$$

$$p(i|O^{1:K}, \hat{O}) = \frac{p(O^{1:K}, \hat{O}|i)p(i)}{\sum_i p(O^{1:K}, \hat{O}|i)p(i)} \tag{14}$$

$$= \frac{p(O^i, \hat{O}|i)p(i)\prod_{k\neq i} p(O^k|i)p(\hat{O}|i)}{\sum_j p(O^j, \hat{O}|j)p(j)\prod_{k\neq j} p(O^k|j)p(\hat{O}|j)} \tag{15}$$

$$= \frac{\frac{p(O^i,\hat{O}|i)}{p(O_i)p(\hat{O})}p(i)}{\sum_j \frac{p(O^j,\hat{O}|j)}{p(O_j)p(\hat{O})}p(j)} \tag{16}$$

The posterior probability for a function f^i is proportional to the class-conditional probability density function $p(O^i, \hat{O}|i)$, which shows the probability of observing the pair $(O^i, \hat{O})$ from function f^i. The optimal classifier would then be proportional to the density ratio given below

$$\exp(sim_{(\phi,\Phi)}(\hat{O}, O^i)) \propto \frac{p(O^i, \hat{O})}{p(O_i)p(\hat{O})} \tag{17}$$

Similar analysis has been shown by the (Oord et al., 2018) for showing the mutual information perspective associated with self-supervised contrastive objective (infoNCE). The joint distribution over the pair of observations correspond to

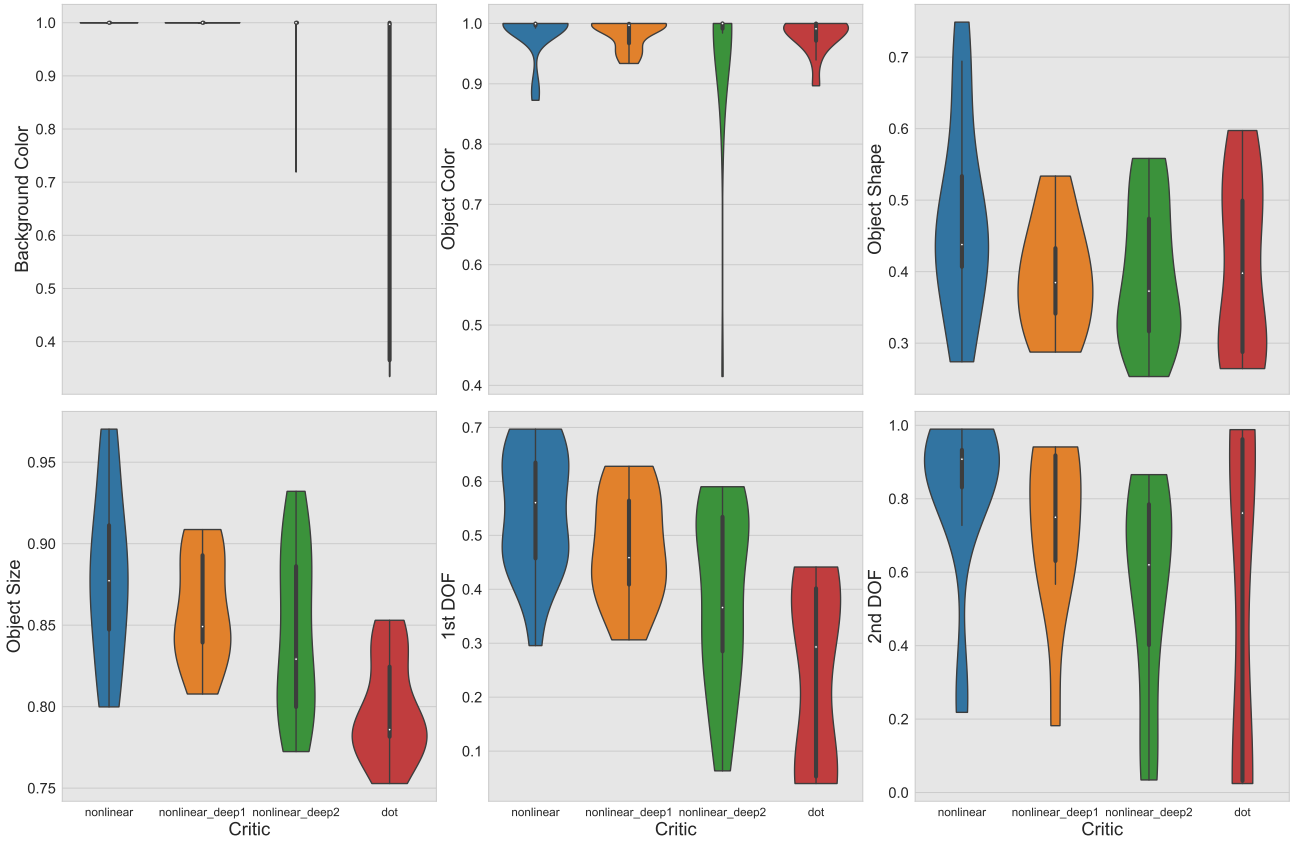


Figure 12. An ablation study to understand the role of different critics in learning meta-representations of MPI3D scenes via FCRL. Shown here is the accuracy achieved by six different linear classifiers, trained on the representations, for identifying the six factors of variation. Non-linear critic consistently performs better than the other critics.

the distribution of the underlying function f^i. Thus, given some observation of a function, an optimal classifier would attempt at estimating the true density of the underlying function.

D. Scenes' Datasets

MPI3D Dataset. The MPI3D dataset (Gondal et al., 2019) is introduced to study transfer in unsupervised representations learning algorithms. The dataset comes in three different formats, varying in the levels of realism i.e. real-world, simulated-realistic and simulated-toy. Each dataset contains $1,036,800$ images of a robotic manipulator each, encompassing seven different factors of variations i.e., object colors (6 values), object shapes (6 values), object sizes (2 values), camera heights (3 values), background colors (3 values), rotation along first degree of freedom ((40 values)) and second degree of freedom ((40 values)). Thus, each image represents a unique combination of all the factors. See Figure 14(a) to see a sample of the dataset.

In this work, we consider the real-world version of the dataset. The multi-view setting is formulated by considering the images of a scene captured by three different cameras, placed at different heights. This effectively gives us $345,600$ scenes with three views each. We split the dataset into training and validation chunks, where the training dataset contains $310,000$ scenes and the validation dataset contains the rest $35,600$ scenes, approximately 10% of the dataset.

RLScenes. The RLScenes dataset is generated in simulation using (Joshi et al., 2020) for a single 3-DOF robotic manipulator in a 3D environment. The dataset consists of $40,288$ scenes, each scene parametrized by: object colors (one of 4), robot tip colours (one of 3), robot positions (uniformly sampled from the range of feasible joint values), and object positions (uniformly sampled within an arena bounded by a high boundary as seen in Figure 14). Each scene consists of 36 views, corresponding to the uniformly distributed camera viewpoints along a ring of fixed radius and fixed height, defined above the environment. As can be seen in Figure 14, the robot finger might not be visible completely in all the views, or the object might be occluded in

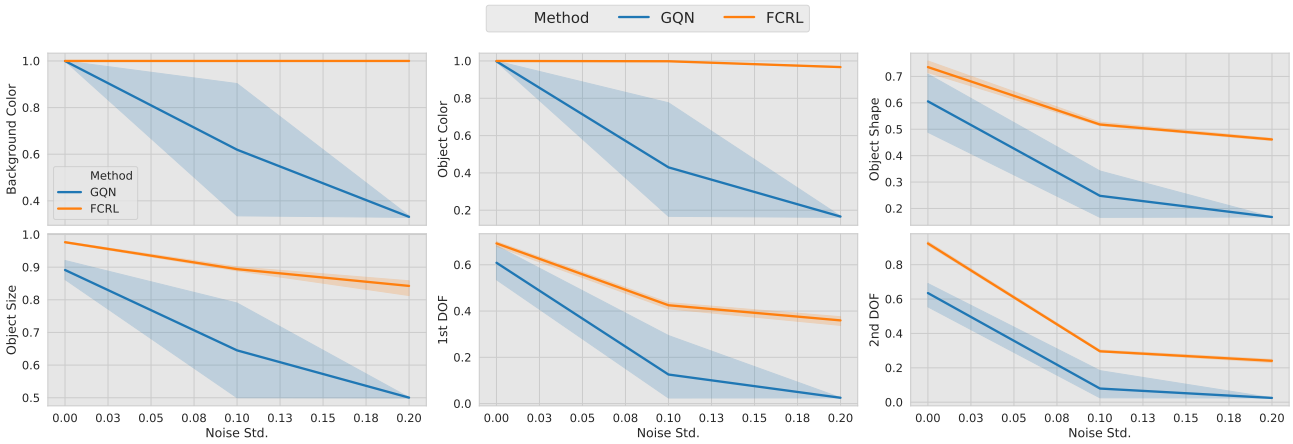
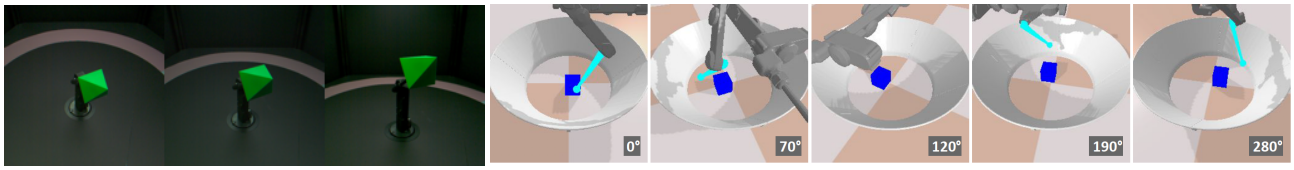


Figure 13. Quantitative comparison of FCRL and GQN for noise robustness on MPI3D downstream tasks. X-axis in each plot corresponds to the varying level of Gaussian noise (as depicted in Figure 11). The representations are extracted from one view only, and the accuracy bands show the standard deviation of five independent runs. It can be seen that the linear classifiers trained on FCRL representations are able to classify the digits even in extremely noisy case, significantly outperforming the classifiers trained on GQN representation.



(a) (b)

Figure 14. Datasets for Scenes Representation Learning (a) MPI3D (Gondal et al., 2019) has three camera viewpoints, with images of a robotics arm manipulating an object. (b) RLScenes has 36 possible camera viewpoints for capturing an arena consisting of a robot finger and an object.

some view. The 36 views help by capturing a 360 deg holistic perspective of the environment. First a configuration of the above scene parameters is selected and displayed in the scene, then the camera is revolved along the ring to capture its multiple views. For learning the scene representations via both FCRL and GQN, we split the dataset into 35000 training and 5288 validation points.

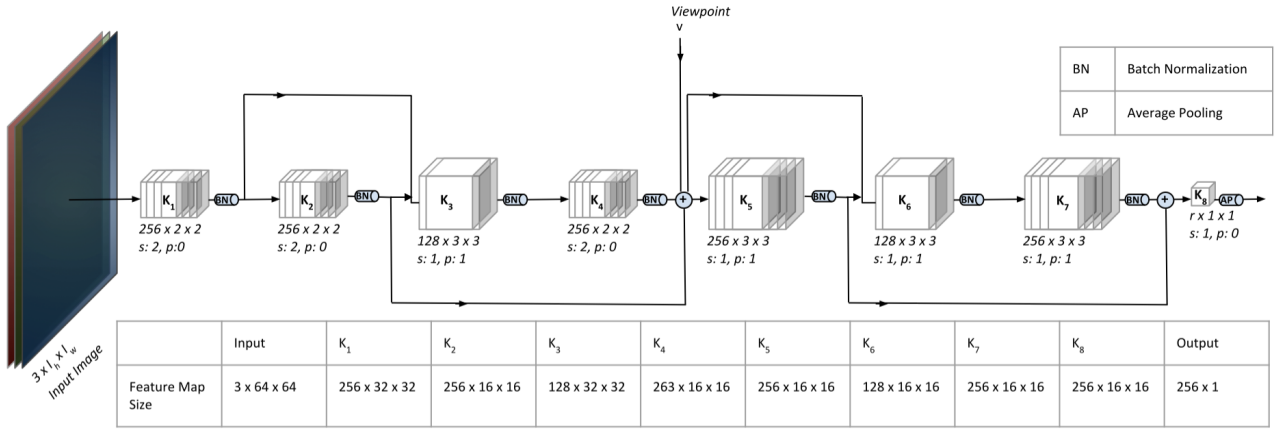
E. Details of Experiments on Scene Representation Learning

For learning the scene representations for both MPI3D dataset and RL Scenes, we used similar base encoder architecture. More specifically, we adapted the "pool" architecture provided in GQN (Eslami et al., 2018), as it has been regarded to exhibit better view-invariance, factorization and compositional characteristics as per the comprehensive study done in (Eslami et al., 2018). We further augmented this architecture with batch-normalization. The architecture we use is shown in Figure 15:

E.1. Learning Scenes for MPI3D Dataset.

Since we have three view per each scene in MPI3D dataset, therefore we are restricted in defining the number of context points and the number of views. In all our experiments the number of context points N is fixed to three while the number of views J is set to two. In contrast to the experiments for regression datasets where more datapoints per each view resulted in better representations, the restriction to a single image view in MPI3D dataset did not hurt the representation quality, measured in terms of downstream performance. In the downstream experiments on MPI3D, we use only one image to train and validate the probes. Due to the limitation of available views, we could not measure the effect of varying the number of views on the downstream performance.

Even though no explicit structural constraint was imposed for learning the representation. The FCRL algorithm implicitly figures out the commonality between the factors in different scenes. We visualize these latent clusterings in the Figure 5. We plot the 2D TSNE embeddings of the 128D representations inferred by the model. Thereafter, to visual-



	Input	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	Output
Feature Map Size	3 x 64 x 64	256 x 32 x 32	256 x 16 x 16	128 x 32 x 32	263 x 16 x 16	256 x 16 x 16	128 x 16 x 16	256 x 16 x 16	256 x 16 x 16	256 x 1

Figure 15. The "pool" architecture used for learning representations for the scenes' datasets via FCRL. The architecture is the same as was used for training GQN (Eslami et al., 2018).

ize the structure corresponding to each factor, we only vary one factor and fix the rest of them except for first degree of freedom and second degree of freedom factors. A clear structure can be seen in the learned representations.

Implementation Details. We use the GQNs 'pool' architecture with batch normalization as encoder. As mentioned in the ablation study, we did a random sweep over the range of hyperparameters and selected the best performing model. Further details on the hyperparameters is provided in Table 3.

E.2. Learning Scenes for RLScenes Dataset.

To train the FCRL encoder, we randomly sample the number of views from each scene to lie within the range [2, 20]: upper-bounded by 20 to restrict the maximum number of images per scene to be the same as that used in (Eslami et al., 2018), and lower bounded by 2 in case just the one view is not from a suitable angle. So, here, the maximum number of context points N is 20. The number of subsets J is set to 8. In the downstream reinforcement learning task, we use only one image to train the policy network, as is the usual practice, and the same as (Eslami et al., 2018). We kept the joint ranges from which joint positions are uniformly sampled to randomly reset the robot at the beginning of every episode while training the policy network to be the same as the ranges used for sampling the robot position while generating the dataset to train the FCRL encoder. These joint ranges are selected so as to ensure that there are more scenes in which the robot finger is visible. However, in order to not constrain the agent's exploration, we let the action space for training the reaching agent to be less constrained, and be able to explore the entire range from $-pi$ to pi. So, effectively, the space seen by the robot during the training of the representations is a subspace of that seen while inferring the representations from the environment used for this downstream reaching task. Interestingly, the inferred representations can also work effectively on unseen robot configurations as demonstrated by the success of the reacher.

Implementation Details. Similar to the encoder training for MPI3D scenes, we learned the encoder for RLScenes. However, since the downstream tasks is a reinforcement learning task, it was hard to judge the quality of representations. Therefore, we took some insights from the MPI3D experiments and selected the model, trained with hyperparameters, which performed the best on the RL downstream tasks. Further details on the hyperparameters is provided in Table 3.

F. Details of Experiments on 1D Functions

Implementation Details. We used the same encoder architecture for our method and the baselines (Garnelo et al., 2018a;b) in all experiments. For 1D and 2D functions, the data is fed in the form of input-output pairs (x, y), where x and y are 1D values. We use MLPs with two hidden layer to encode the representations of these inputs. The number of hidden units in each layer is $d = 50$. All MLPs have relu non-linearities except the final layer, which has no non-linearity.
Encoder: Input(2) $\rightarrow 2 \times$ (FC(50), ReLU) $\rightarrow$ FC(50).
While learning the representations of sinusoid functions with FCRL, we also scale the output scores with temperature to be 0.07. We used the following hyper-parameter settings to train an encoder with FCRL.

Downstream Tasks. To learn the subsequent task-specific decoders on the representations, we adapted the same data

Table 3. Hyperparameters Settings for Scene Representation Learning Experiments.

Parameter	Values
Batch size	64
Representation dimension	128
Temperature: τ	0.88
Number of subsets: J	2
Max number of context points: 3	
Epochs	100
Critic	Nonlinear
Objective	NCE
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1e-8
Adam: learning rate	0.0005
Learning Rate Scheduler	Cosine
Number of workers	10
Batch normalization	Yes

(a) Hyperparameters to train FCRL based encoder on the MPI3D Dataset.

Parameter	Values
Batch size	32
Representation dimension	256
Temperature: τ	0.46
Number of subsets: J	4
Max number of context points: 20	
Epochs	100
Critic	Nonlinear
Objective	NCE
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1e-8
Adam: learning rate	0.0005
Learning Rate Scheduler	Cosine
Number of workers	10
Batch normalization	Yes

(b) Hyperparameters to train FCRL based encoder on the RLScenes Dataset.

Table 4. Hyperparameters Settings for Sinusoid Experiments.

Parameter	Values
Batch size	256
Latent space dimension	50
Temperature: τ	0.07
Number of subsets: J	2
Max number of context points: N	20
Epochs	30
Critic	Nonlinear
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1e-8
Adam: learning rate	0.0003
Learning Rate Scheduler	Cosine

(a) Hyperparameters to train FCRL based encoder for 1D sinusoid functions.

Parameter	Values
Batch size	256
Epochs	30
Critic	Nonlinear
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1e-8
Adam: learning rate	0.001
Learning Rate Scheduler	Cosine

(b) Hyperparameters to train FSR Decoder on FCRL learned representations.

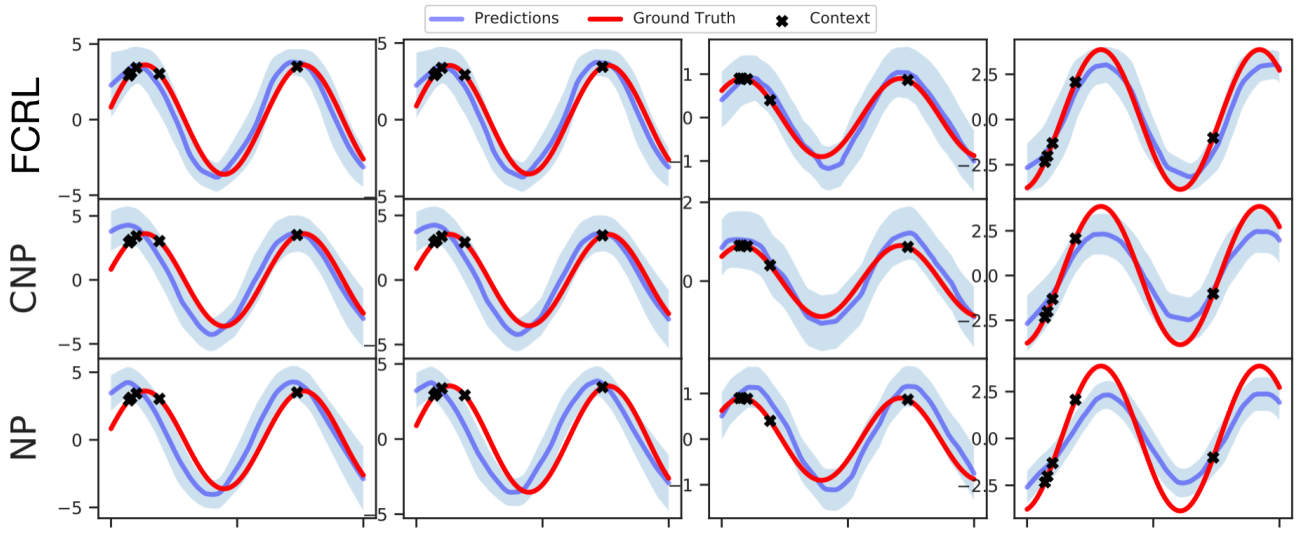


Figure 16. Additional results on 5-shot sinusoid regression. Each column corresponds to a different sinusoid function where only 5 context points are given. The predictions of the decoder trained on FCRL based encoder are closer to the groundtruth.

processing pipeline as above. For 1D functions, we train decoders for two different tasks: *few-shot regression* and *few-shot parameter identification*. The decoders for each task are trained with the same training dataset as was used to train the encoders. The training procedure for both downstream tasks on sinusoid functions is as follows

- For Few-Shot Regression (FSR), we use an MLP architecture with two hidden layers. The same architecture are used in CNP (Garnelo et al., 2018a), however in CNP the decoder and encoder are trained jointly. All the baselines and our model are trained for the same number of iterations. We used slightly higher learning rate to train the decoder as the training converges quite easily.
  FSR Decoder: Input(50) $\rightarrow$ 2 $\times$ (FC(50), ReLU) $\rightarrow$ FC(1) .

- For Few-Shot Parameter Identification (FSPI), we train a linear decoder without any activation layers on the representations learned via FCRL and the baseline methods. The decoder is trained for only one epoch.
  FSPI Decoder: Input(50) $\rightarrow$ FC(1) .

Additional Results. In Figure 16 and Figure 17, we provide additional results on 5-shot regression on test sets and compare the results with CNP and NP. The curves generated by the decoder using FCRL learned representations are closer to the groundtruth. The difference is evident in 5-shot experiments which supports the quantitative results in Table 1.

G. Details of Experiments on 2D Functions

Implementation Details. In this experiment, we treat MNIST images as 2D functions. We adapt the architectures of encoders and decoders from the previous 1D experiments. However, due to the increased complexity of the function distributions we increase the number of hidden units of MLP to $d = 128$. Moreover, the input x is 2D as it corresponds to the cartesian coordinates of an image. The hyperparameter settings to train FCRL based encoder on such 2D function is given in Table 5.

G.1. Downstream Tasks.

We consider two downstream tasks to evaluate the quality of the representations learned on 2D functions: *few-shot image completion* and *few-shot content classification*. A separate decoder is trained for both of these tasks.

- For Few-Shot Image Completion (FSIC), we use an MLP based decoder with two hidden layers. The decoder is trained on the same training data for the same number of iterations. Details are given in Table 5(b).

- For Few-Shot Content Classification (FSCC), we train a linear regression on top of the representations obtained by both FCRL and the baselines. The decoder is trained for only one epoch.

Additional Results. In Figure 18 and Figure 19, we provide additional results for 50-shot and 200-shot image completion. We can see that the the results from FCRL based decoder consistently perform better than CNP in low-shot

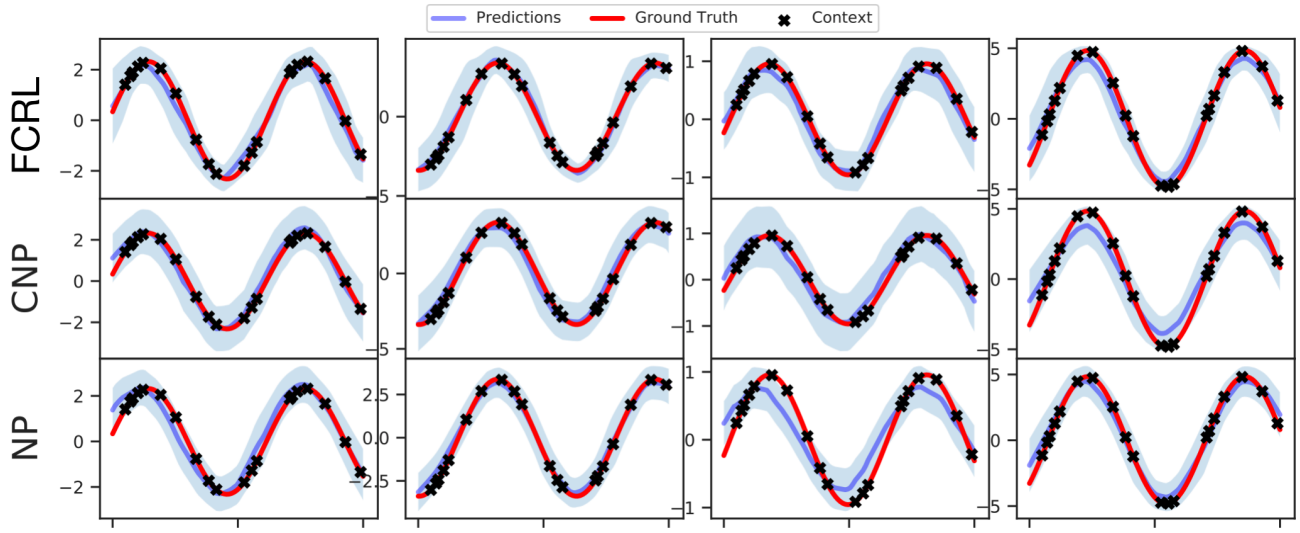


Figure 17. Additional results on 20-shot sinusoid regression. Each column corresponds to a different sinusoid function where only 20 context points are given. The predictions of the decoder trained on FCRL based encoder are comparable to CNP and better than NP.

Table 5. Hyperparameters Settings for MNIST as 2D Functions Experiment.

Parameter	**Values**
Batch size	16
Latent space dimension	128
Temperature: τ	0.007
Number of subsets: J	40
Max number of context points: N	200
Epochs	100
Critic	Nonlinear
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1e-8
Adam: learning rate	0.0006
Learning Rate Scheduler	Cosine

(a) Hyperparameters to train FCRL based encoder for 2D functions.

Parameter	**Values**
Batch size	16
Epochs	100
Critic	Nonlinear
Optimizer	Adam
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	1e-8
Adam: learning rate	0.001
Learning Rate Scheduler	Cosine

(b) Hyperparameters to train Few-Shot Image Completion (FSIC) Decoder trained on FCRL learned representations.

scenario of 50 context points. In 200-shot scenario, the results look comparable to CNP.

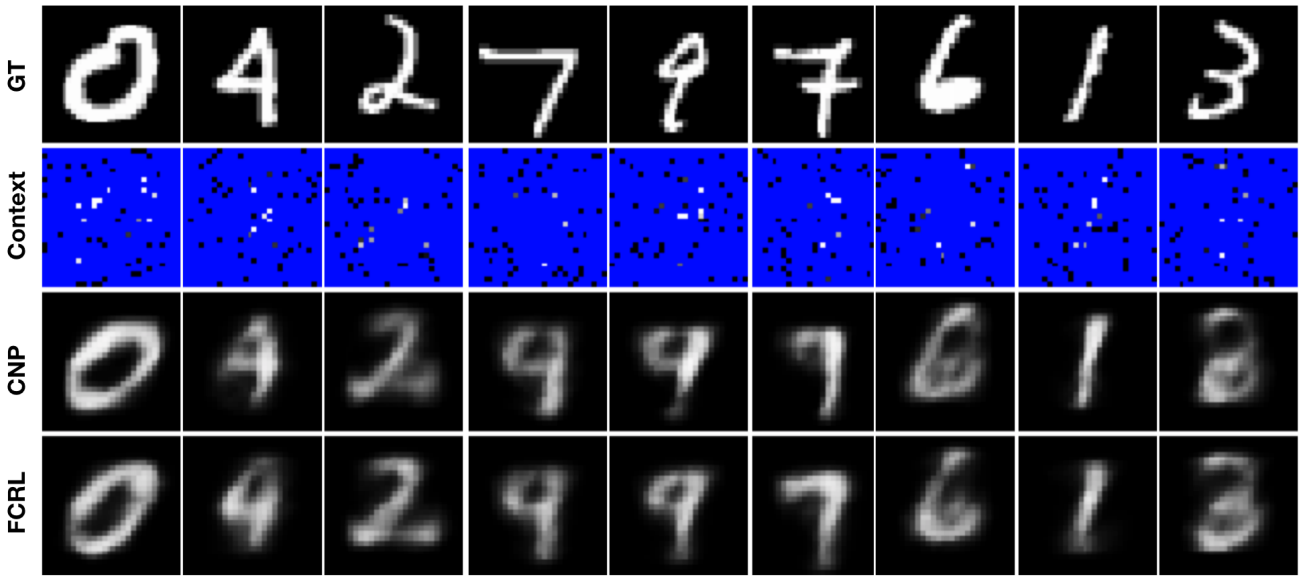


Figure 18. Additional results on 50-shot mnist image completion. The context is shown in the second row where target pixels are colored blue. Predictions made by a decoder trained on FCRL based encoder are slightly better than the CNP in terms of guessing the correct form of digits.

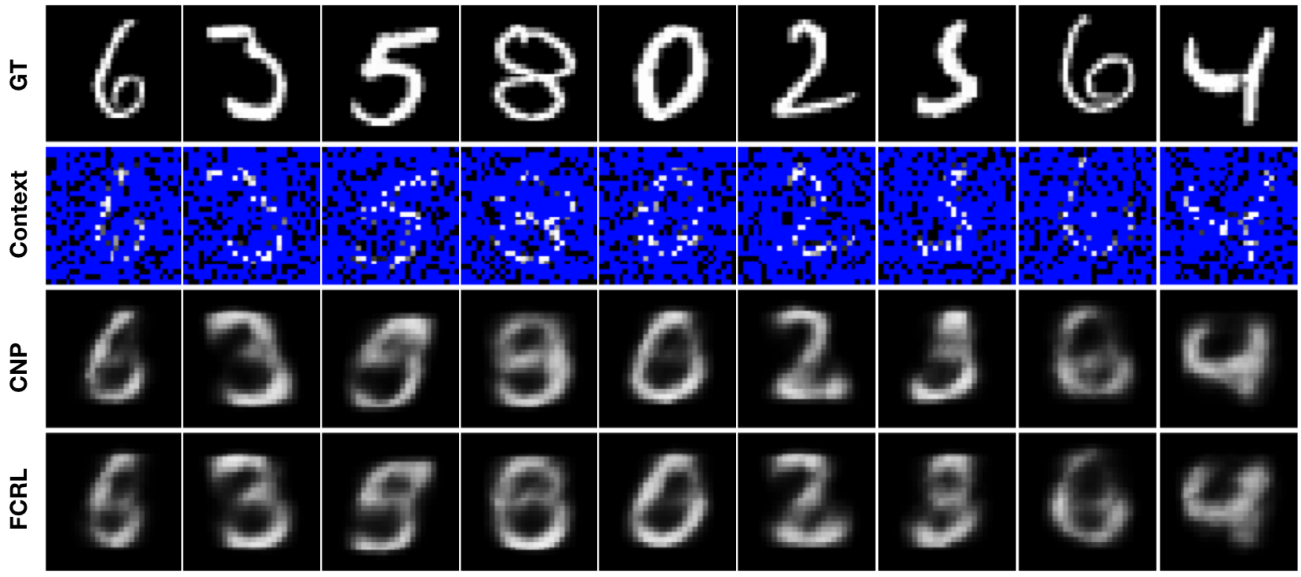


Figure 19. Additional results on 200-shot mnist image completion. The context is shown in the second row where target pixels are colored blue. Predictions made by a decoder trained on FCRL based encoder are comparable to CNP.