A. Model Details

A.1. ANP, SNP and ASNPs

In this section, we describe the model details of ANP, SNP, ASNP-W, and ASNP-RMR. All the models share the basic components such as the real context encoder and decoder. There are two context encoders for producing the latent and the deterministic representations. *i) Latent Encoder:* In this encoder, we first use a 3-layer MLP with ReLU (Nair & Hinton, 2010) activation to encode each context point. Next, we sum-pool these encoded representations to obtain a single context representation. Lastly, we provide this representation to a 2-layer MLP to compute the mean and variance of the latent. *ii) Deterministic Encoder:* In this encoder, we use a 6-layer MLP with ReLU activation to encode each context point. For SNP, we sum-pool these encodings. For attention-based models, we associate each encoding with its corresponding key to obtain a key-value set. For ANP, we directly attend to this key-value set based on the target queries. For ASNP-W and ASNP-RMR, we first augment this key-value set with a memory (i.e. RMR or context buffer) and then attend on it given the target queries.

Attention Mechanism. In the attention module, ANP, ASNP-W and ASNP-RMR can use any of the following implementations: Dot-Product, Laplace or Multihead. In our experiments, we use Multihead attention (Vaswani et al., 2017) as recommended in Kim et al. (2019). ASNP-RMR has two attention modules – one for value-flow interaction and other for the deterministic encoder. In our implementation, both attention modules share parameters since it showed better performance compared to using different parameters.

Encoding task-step into the query. Recall that in ANP, we perform sum-pooling of context points from different tasks. Similarly, in ASNP-W, we store the context points from different tasks in the same memory buffer. In these situations, if the task-step information is absent from the context points, the models cannot distinguish the task to which each point belongs. To prevent this, we concatenate each query with the corresponding task-step information. The task-step information is provided as a normalized float value $e_t = 0.25 + 0.5(t/T)$, where t is the task-step and T is the length of the task-sequence. Therefore, the augmented query is given by $x' = (x, e_t)$.

Recurrent Modules. The sequential models (SNP and ASNP) have an underlying recurrent architecture in the latent encoder and the deterministic encoder. As in SNP (Singh et al., 2019), we use the recurrent state-space model (RSSM) (Hafner et al., 2018) and we use LSTM to implement it. For RMR, we also use LSTM to implement the recurrent modules in the key and value update. The LSTM module for the key update takes the concatenated keys as an input. We use the default setting of Tensorflow (Abadi et al., 2016) for all LSTMs. The LSTM hidden unit size and representation size are the same and are denoted by h. We experiment with values of $h = 128$, 512 and 1024.

Memory Sizes. Let K denote the RMR memory size. Let K also denote the size of the context window in ASNP-W. In our experiments, we evaluate the models for $K = 9$, 25 and 100. We also evaluate the case when ASNP-W stores the entire past context. We denote this case with $K = \inf$.

Training. The initial imaginary context points are trainable parameters. For dynamic 1D regression, we train our models with a learning rate 10^{-4} and batch size 16. For moving MNIST and moving CelebA, we train with batch sizes 8 and 4, respectively.

A.2. TGQN and ATGQN

In this section, we describe the model details of TGQN, ATGQN-W and ATGQN-RMR. Each of them share the basic architecture that consists of *i)* an encoder to encode the real contexts, *ii)* the recurrent state-space model to generate the latents and *iii)* a decoder to generate the target outputs. In the encoder, we use 3-layer CNN with ReLU activation to encode the context image. We append the query to the last layer to provide the encoder with the query information similar to the Tower Network in Eslami et al. (2018). As in TGQN Singh et al. (2019), we sum-pool the representations to obtain a global representation. To obtain the latents $z_{1:T}$, we use Temporal-ConvDRAW (Singh et al., 2019), which is implemented using a ConvLSTM (Xingjian et al., 2015). For the deterministic representation also, we use ConvLSTM. Similar to CGQN (Kumar et al., 2018), our decoder renders the image recurrently.

In ATGQN-W, we append the task-step information to the queries as described above for ANP and ASNP-W. We implement attention in ATGQN same as ANP and ASNP-W. To generate the imaginary keys we use LSTM with the default Pytorch settings (Paszke et al., 2017). To maintain a convolutional architecture, we implement the value flow-tracker using a ConvLSTM.

In our experiments, the patch-size is 8 and $h = 128$. Number of DRAW steps for Temporal-ConvDRAW is 3 and the dimension of z is 4. At each task-step, we take z_t to be the latent obtained at the last iteration of ConvDRAW. The ConvLSTM hidden unit size in latent and deterministic encoding is 40. In the decoder, the hidden unit size is 32 and

the number of steps for auto-regressive rendering is 2. The query size is 2. For ATGQN-W, the query size is 3 since we concatenate the query and the task-step. We take the imaginary context size and the context window size as 100. We train the models with a learning rate 10^{-4} and batch size 4.

B. Setting Details

B.1. Dynamic 1D Regression

To generate a task-sequence, we first choose the kernel parameters for the Gaussian Process setting randomly at $t = 1$. In our experiments, the GPs use a squared-exponential kernel that are characterized by a length-scale l and a kernel-scale σ. Therefore, at each task-step, the kernel parameters are updated as $l \leftarrow l + \Delta l + \epsilon_l$ and $\sigma \leftarrow \sigma + \Delta\sigma + \epsilon_\sigma$ where ϵ_l and ϵ_σ are a small Gaussian noise. At $t = 1$, we randomly choose Δl from $[-0.03, 0.03]$ and $\Delta\sigma$ from $[-0.05, 0.05]$. We sample the noise values at every task-step from a Gaussian distribution $\mathcal{N}(0, 0.1)$.

In the sparse-context regime, we choose the initial values of l from $[0.7, 1.2]$ and σ from $[1.0, 1.6]$. We then randomly choose 45 task-steps out of 50 and provide a small-sized context in them. For the remaining tasks, we provide an empty context. Let n denote the number of contexts and m denote the number of targets. At the task-steps for providing a small-sized context, we provide 1 context, $n = 1$. At every task-step, we randomly select m from $[1, 11 - n]$. In the transfer-prediction regime, we select the initial values of l from $[1.2, 1.9]$ and σ from $[1.6, 3.1]$. We provide a large-sized context in the first 10 task-steps out of 20 and an empty context in the remaining. At the task-steps chosen for providing a non-empty context, we randomly choose n from $[5, 50]$. At every task-step, we randomly choose m from $[1, 51 - n]$.

B.2. Dynamic 2D Image Completion

In this setting, we experiment on the moving MNIST and moving CelebA dataset. For each task-step, we take a white canvas of size 42×42 that consists of a moving image. The size of the moving MNIST image is 28×28 and the size of the moving CelebA face image is 32×32. At $t = 1$, the digits or the faces start to move from a random location and head towards a direction chosen randomly. The motion speed is 3 pixels per task-step. At every task-step, we also add a small Gaussian noise to the transition to introduce stochasticity similar to the dynamic 1D regression setting. When a wall is encountered, the image performs a perfectly elastic bounce. As in the dynamic 1D regression setting, we take the same sequence length and choose the same number of task-steps for providing a non-empty context. In the sparse-context regime, we take $n = 30$ for the tasks with non-empty context. We randomly choose m from $[1, 51 - n]$ for every task. In the transfer-prediction regime, for the tasks chosen for a non-empty context, we randomly choose n from $[5, 500]$. For every task, we randomly choose m from $[1, 501 - n]$.

B.3. Dynamic 2D Image Rendering

In this setting, we experiment on the moving CelebA dataset. We take the size of the white canvas as 80×80 and the size of the moving face image as 64×64. The image-patch size is 8×8. The moving image starts to move from a random location in a random direction at a speed of 13 pixels per task-step. When a wall is encountered, the image performs a perfect bounce. We add a small Gaussian noise on each transition. We take sequences of length 6. In the sparse-context regime, we provide a small-sized context in 5 task-steps and an empty context in the remaining. At the task-steps chosen for providing a small-sized context, we provide 100 contexts i.e. $n = 100$. At every task-step, we randomly choose m from $[1, 151 - n]$. In the transfer-prediction regime, we provide a large-sized context in the first 3 task-steps out of 6 and an empty context in the remaining. At the task-steps chosen for non-empty context, we randomly choose n from $[300, 350]$. At every time-step, we randomly choose m from $[1, 351 - n]$.

C. ELBO Derivations

Although our proposed framework allows a stochastic RMR, for simplicity, we use a deterministic RMR as $P(\tilde{C}_t|\tilde{C}_{<t}, C_{\leq t}) = \delta[\tilde{C}_t = \mathrm{RMR}(\tilde{C}_{<t}, C_{\leq t})]$ where δ is a Dirac delta function. With this, we can describe the generative process of ASNP as follows.

$$P(Y, Z, \tilde{C}|X, C) = \prod_{t=1}^{T} P(y_t|x_t, z_t, \bar{C}_t)P(z_t|z_{<t}, \tilde{C}_{\leq t}, C_{\leq t}). \tag{12}$$

For this generative process, we derive an ELBO as follows.

$$
\begin{aligned}
&\log P(Y|X,C) \\
&= \log \mathbb{E}_{Q(Z|\tilde{C},C,D)}\left[\frac{P(Y,Z,\tilde{C}|X,C)}{Q(Z|\tilde{C},C,D)}\right] \\
&= \log \mathbb{E}_{Q(Z|\tilde{C},C,D)}\left[\prod_{t=1}^{T}\frac{P(y_t|x_t,z_t,\bar{C}_t)P(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t})}{Q(z_t|z_{<t},\tilde{C}_{\leq t},C,D)}\right] \\
&\geq \mathbb{E}_{Q(Z|\tilde{C},C,D)}\left[\log\prod_{t=1}^{T}\frac{P(y_t|x_t,z_t,\bar{C}_t)P(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t})}{Q(z_t|z_{<t},\tilde{C}_{\leq t},C,D)}\right] \\
&= \mathbb{E}_{Q(Z|\tilde{C},C,D)}\sum_{t=1}^{T}\left[\log\frac{P(y_t|x_t,z_t,\bar{C}_t)P(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t})}{Q(z_t|z_{<t},\tilde{C}_{\leq t},C,D)}\right] \\
&= \mathbb{E}_{Q(Z|\tilde{C},C,D)}\sum_{t=1}^{T}\left[\log P(y_t|x_t,z_t,\bar{C}_t) - \log\frac{Q(z_t|z_{<t},\tilde{C}_{\leq t},C,D)}{P(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t})}\right] \\
&= \sum_{t=1}^{T}\mathbb{E}_{\prod_{t'=1}^{t}Q(z_{t'}|z_{<t'},\tilde{C}_{\leq t'},C,D)}\left[\log P(y_t|x_t,z_t,\bar{C}_t)\right] \\
&\quad - \mathbb{E}_{\prod_{t'=1}^{t}Q(z_{t'}|z_{<t'},\tilde{C}_{\leq t'},C,D)}\left[\log\frac{Q(z_t|z_{<t},\tilde{C}_{\leq t},C,D)}{P(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t})}\right] \\
&= \sum_{t=1}^{T}\mathbb{E}_{\prod_{t'=1}^{t}Q(z_{t'}|z_{<t'},\tilde{C}_{\leq t'},C,D)}\left[\log P(y_t|x_t,z_t,\bar{C}_t)\right] \\
&\quad - \mathbb{E}_{\prod_{t'=1}^{t-1}Q(z_{t'}|z_{<t'},\tilde{C}_{\leq t'},C,D)}\mathbb{KL}\left[(Q(z_t|z_{<t},\tilde{C}_{\leq t},C,D)\parallel P(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t}))\right] \\
&= \sum_{t=1}^{T}\mathbb{E}_{Q_\phi(z_t|C,D)}\left[\log P_\theta(y_t|x_t,z_t,\tilde{C}_t,C_t)\right] \\
&\quad - \mathbb{E}_{Q_\phi(z_{<t})}\left[\mathbb{KL}\left(Q_\phi(z_t|z_{<t},\tilde{C}_{\leq t},C,D)\parallel P_\theta(z_t|z_{<t},\tilde{C}_{\leq t},C_{\leq t}))\right)\right]
\end{aligned}
$$

where $Q_\phi(z_t|C,D)=\prod_{t'=1}^{t}Q(z_{t'}|z_{<t'},\tilde{C}_{\leq t'},C,D)$ and $Q_\phi(z_{<t})=\prod_{t'=1}^{t-1}Q(z_{t'}|z_{<t'},\tilde{C}_{\leq t'},C,D)$ for simplicity.

D. Qualitative Results

D.1. Dynamic 1D regression

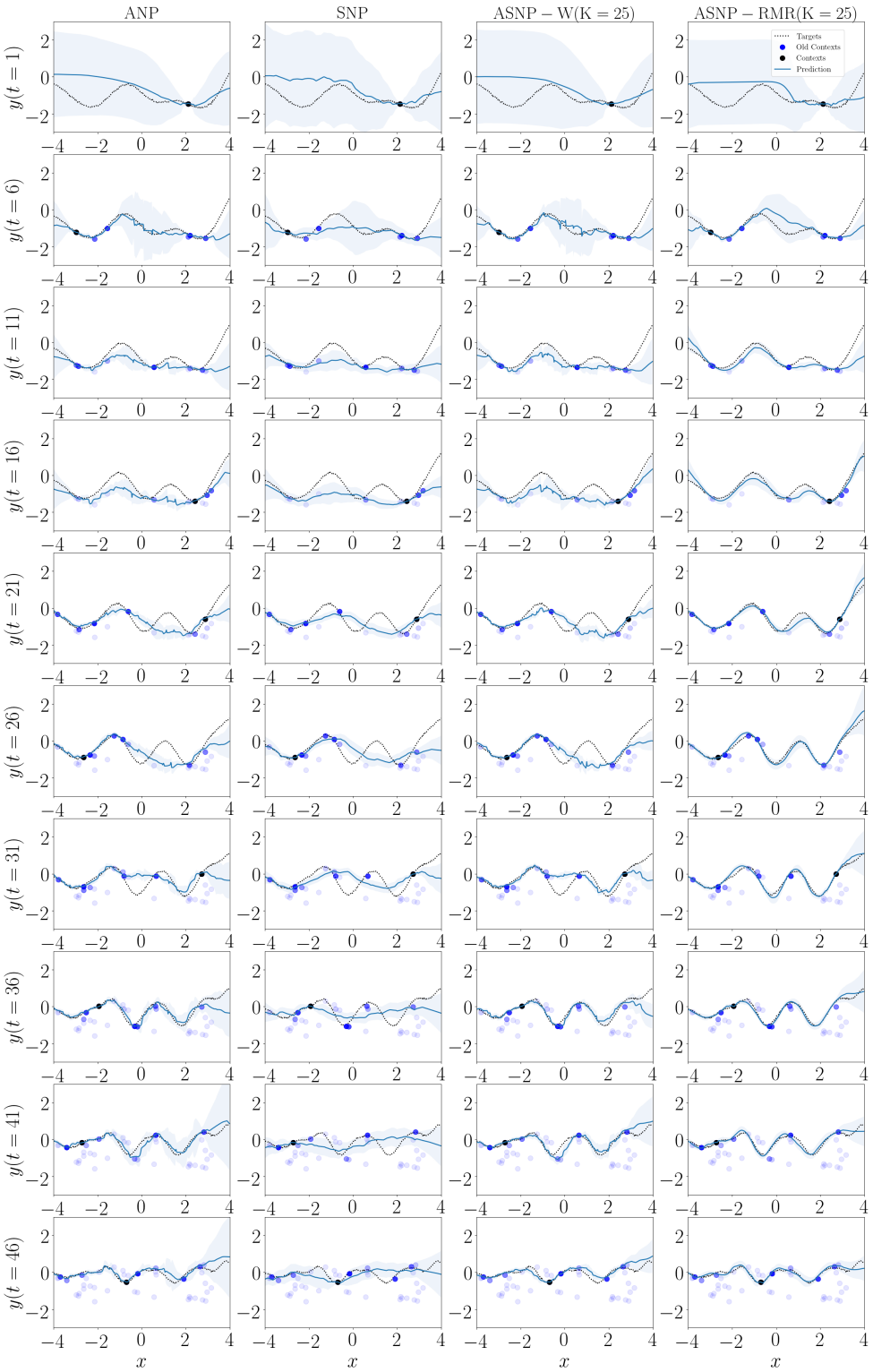

Figure 11: Dynamic 1D regression samples in sparse-context regime. Columns are ANP, SNP, ASNP-W and ASNP-RMR, respectively. Each row shows examples at each task-step. Every 5^{th} task-step is shown.

D.2. Dynamic 2D image completion

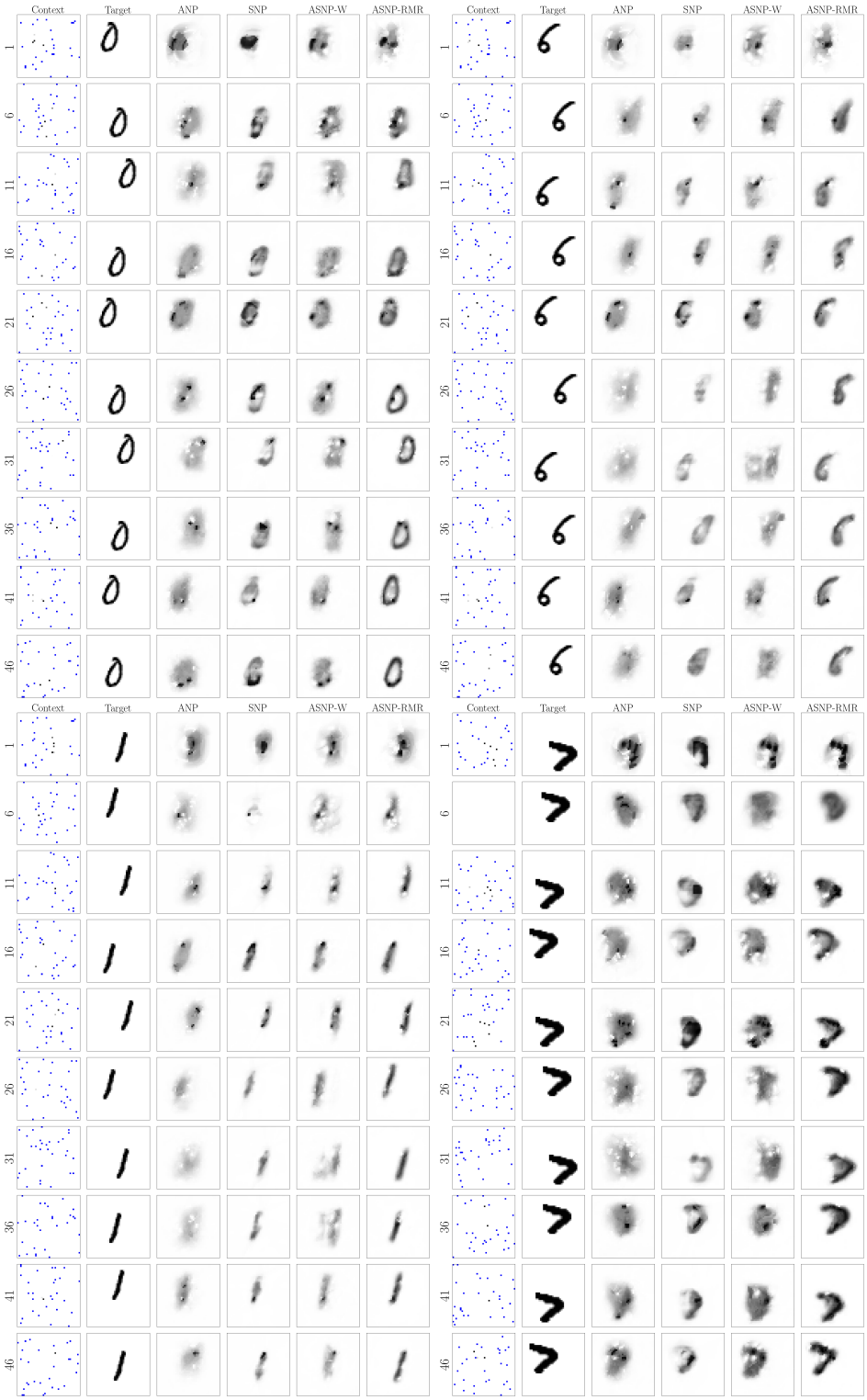

Figure 12: Dynamic moving MNIST completion samples in sparse-context regime. Columns are Context, Target, ANP, SNP, ASNP-W and ASNP-RMR, respectively. Each row shows examples at different task-steps. We show every 5^{th} task-step.

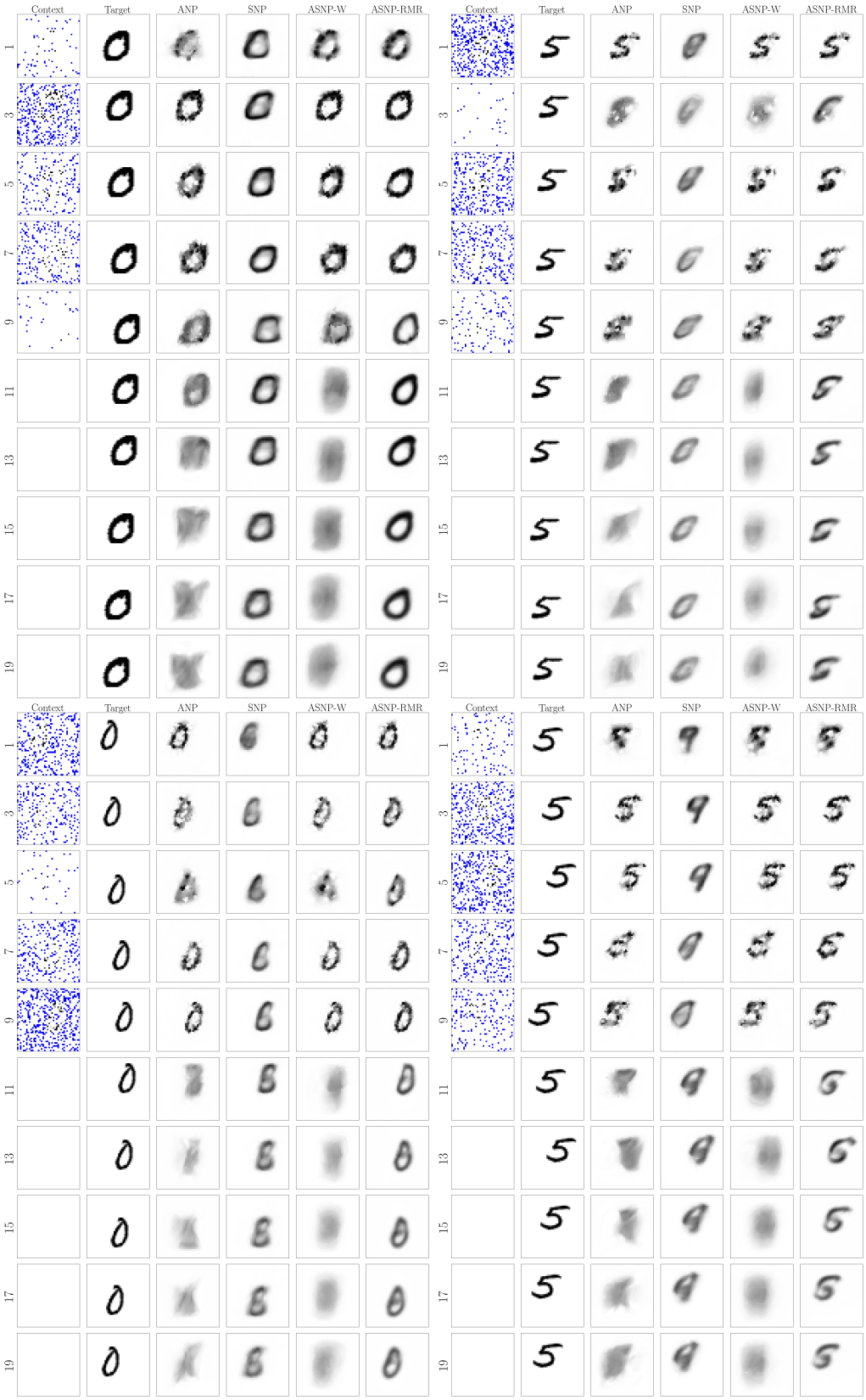

Figure 13: Dynamic moving MNIST completion samples for transfer-prediction. Columns are Context, Target, ANP, SNP, ASNP-W and ASNP-RMR, respectively. Each row shows examples at different task-steps. We show every 2nd task-step.

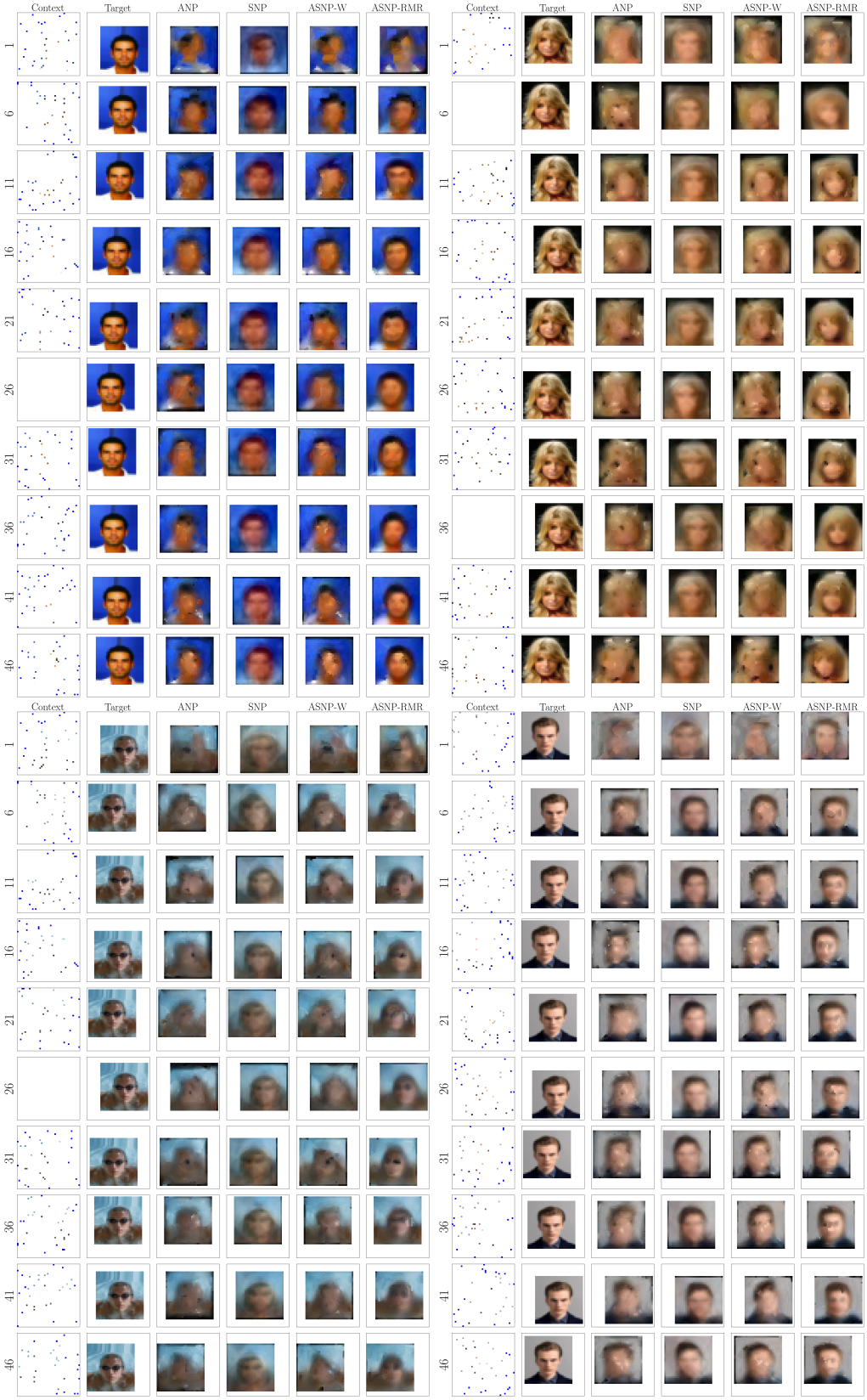

Figure 14: Dynamic moving CelebA completion samples in sparse-context regime. Columns are Context, Target, ANP, SNP, ASNP-W and ASNP-RMR, respectively. Each row shows examples at each task-step. Every 5^{th} task-step is shown here.

Figure 15: Dynamic moving CelebA completion samples for transfer-prediction. Columns are Context, Target, ANP, SNP, ASNP-W and ASNP-RMR, respectively. Each row shows examples at each task-step. Every 2nd task-step is shown here.