

1 A.1 Appendix

2 A.2 Discovery of Objects

3 In contrast to other algorithms, DAF-Net learns to predict both a set of object properties y_k of objects
 4 and a set of confidences c_k for each object. This corresponds to the task of both predicting the number
 5 of objects in set of observations, as well as associated object properties. We evaluate the ability to
 6 regress object number in DAF-Net in scenarios where the number of objects is different than that of
 7 training. We evaluate on the Normal distribution with a variable number of component distributions,
 8 and measure inferred component through a threshold confidence. DAF-Net is trained on a dataset
 9 with 3 underlying components.

10 We find in Figure A1 that DAF-Net is able to infer the presence of more component distributions (as
 11 they vary from 3 to 6), with improved performance when cluster centers are sharply separated (right
 12 figure of Figure A1).

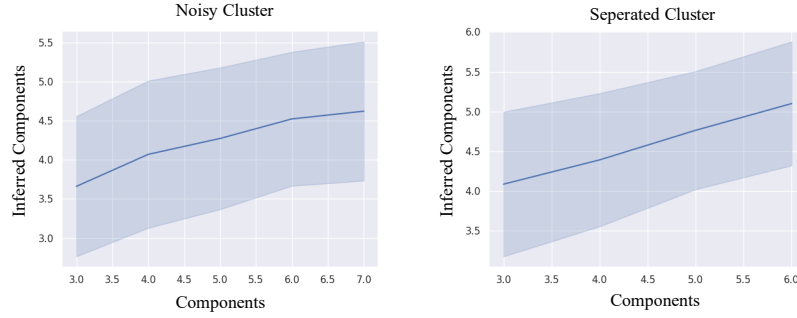


Figure A1: Plots of inferred number of components using a confidence threshold in DAF-Net compared to the ground truth number of clusters (DAF-Net is trained on only 3 clusters). We consider two scenarios, a noisy scenario where cluster centers are randomly drawn from -1 to 1 (left) and a scenario where all added cluster components are well separated from each other (right). DAF-Net is able to infer more clusters in both scenarios, with better performance when cluster centers are more distinct from each other.

13 A.3 Qualitative Visualizations

14 We provide an illustration of our results on the *Normal* clustering task in Figure A2. We plot the
 15 decoded values of hypothesis slots in red, with size scaled according to confidence, and ground-truth
 16 cluster locations in black. DAF-Net is able to selectively refine slot clusters to be close to ground
 17 truth cluster locations even with much longer observation sequences than it was trained on.

18 We find that each component learned by DAF-Net is interpretable. We visualize attention weights of
 19 each hypothesis slot in Figure A3 and find that each hypothesis slot learns to attend to a local region
 20 next to the value it decodes to. We further visualize a plot of relevance weights in Figure A4 across
 21 increasing number of observations over different levels of noise in each distribution. We find that as

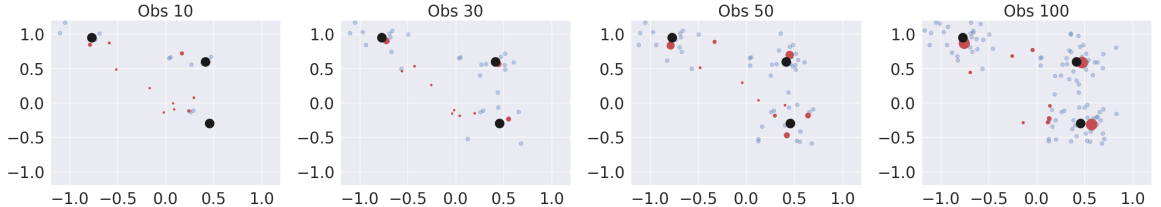


Figure A2: Illustration of the clustering process. Decoded value of hypothesis (with size corresponding to confidence) shown in red, with ground truth clusters in black. Observations are shown in blue.

we see more observations, the relevance weight of new observations decreases over time, indicating that DAF-Net learns to pay the most attention towards the first set of observations it sees. In addition, we find that in distributions with higher variance, the relevance weight decreases more slowly, as later observations are now more informative in determining cluster centers.

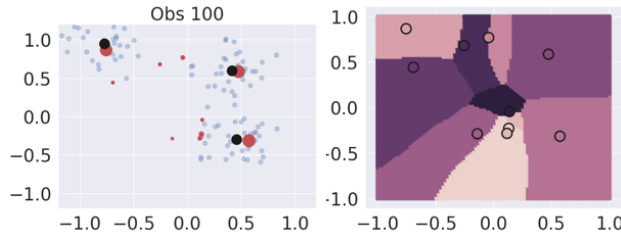


Figure A3: Plot of slots (left), and what slot each input assigns the highest attention towards (right) (each slot is colored differently, with assigned inputs colored in the same way). Note alignment of regions on the right with point density on the left.

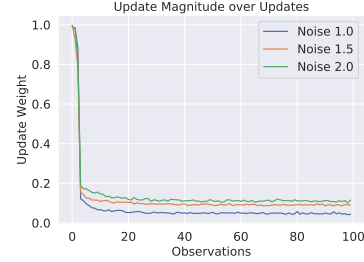


Figure A4: Plots of the magnitude of relevance weights with increased observation number on different distributions with higher standard deviation (noise).

26 A.4 Quantitative Results

We report full performance of each different model across different distributions in Table 1. We find that DAF-Net is able to obtain better performance with increased number of observations across different distributions. In addition DAF-Net out-performs neural network baselines when evaluated on 30 observations across distributions except for rotation. For rotation we find that when training with 10,000 different distribution, DAF-Net exhibits better performance of 0.555 compared to Set Transformer Online performance of 0.647 and LSTM performance of 0.727.

33 A.5 Distributions Details

We provide detailed quantitative values for each distribution below. Gaussian centers are drawn uniformly from -1 to 1.

1. *Normal*: Each 2D Gaussian has standard deviation 0.2.
2. *Mixed*: Each distribution is a 2D Gaussian, with fixed identical variance across each individual dimension, but with the standard deviation of each distribution drawn from a uniform distribution from (0.04, 0.4).
3. *Elongated*: Each distribution is a 2D Gaussian, where the standard deviation along each dimension is drawn from a uniform distribution from (0.04, 0.4), but fixed across distributions.
4. *Angular*: Each distribution is a 2D Gaussian with identical standard deviation across dimension and distribution, but points above π are wrapped around to $-\pi$ and points below $-\pi$ wrapped to π . Gaussian means are selected between $(-\pi, -2\pi/3)$ and between $(2\pi/3, \pi)$. The standard deviation of distributions is $0.3 * \pi$.
5. *Noise*: Each distribution has 2 dimensions parameterized by Gaussian distributions with standard deviation 0.5, but with the values of the remaining 30 dimensions drawn from a uniform distribution from $(-1, 1)$.

Type	Model	Online	Observations			
			10	30	50	100
Normal	DAF-Net	+	0.235 (0.001)	0.162 (0.001)	0.146 (0.001)	0.128 (0.001)
	Set Transformer	+	0.390 (0.002)	0.388 (0.002)	0.388 (0.002)	0.389 (0.001)
	LSTM	+	0.288 (0.001)	0.260 (0.001)	0.269 (0.001)	0.288 (0.001)
	VQ	+	0.246 (0.001)	0.172 (0.001)	0.147 (0.001)	0.122 (0.001)
	Set Transformer	+	0.295 (0.003)	0.261 (0.001)	0.253 (0.001)	0.247 (0.001)
	K-means++	-	0.183 (0.002)	0.107 (0.001)	0.086 (0.001)	0.066 (0.001)
	GMM	-	0.189 (0.002)	0.118 (0.001)	0.087 (0.001)	0.067 (0.001)
Mixed	DAF-Net	+	0.255 (0.002)	0.184 (0.001)	0.164 (0.001)	0.147 (0.001)
	LSTM	+	0.306 (0.002)	0.274 (0.001)	0.284 (0.001)	0.290 (0.001)
	Set Transformer	+	0.415 (0.002)	0.405 (0.001)	0.407 (0.001)	0.408 (0.001)
	VQ	+	0.262 (0.002)	0.192 (0.001)	0.169 (0.001)	0.145 (0.001)
	Set Transformer	-	0.309 (0.002)	0.274 (0.001)	0.266 (0.001)	0.261 (0.001)
	K-means++	-	0.206 (0.003)	0.135 (0.001)	0.105 (0.001)	0.088 (0.001)
	GMM	-	0.212 (0.003)	0.136 (0.001)	0.105 (0.001)	0.079 (0.001)
Enlongated	DAF-Net	+	0.258 (0.002)	0.192 (0.001)	0.173 (0.001)	0.161 (0.001)
	LSTM	+	0.314 (0.003)	0.274 (0.002)	0.288 (0.001)	0.300 (0.001)
	Set Transformer	+	0.394 (0.003)	0.391 (0.003)	0.394 (0.003)	0.394 (0.003)
	VQ	+	0.265 (0.003)	0.194 (0.002)	0.172 (0.001)	0.149 (0.001)
	Set Transformer	-	0.309 (0.002)	0.244 (0.002)	0.240 (0.001)	0.232 (0.001)
	K-means++	-	0.213 (0.002)	0.139 (0.001)	0.113 (0.001)	0.092 (0.001)
	GMM	-	0.214 (0.002)	0.141 (0.001)	0.112 (0.001)	0.086 (0.001)
Rotation	DAF-Net	+	0.892 (0.001)	0.794 (0.001)	0.749 (0.002)	0.736 (0.001)
	LSTM	+	0.799 (0.003)	0.796 (0.002)	0.795 (0.002)	0.794 (0.002)
	Set Transformer	+	0.793 (0.003)	0.794 (0.002)	0.782 (0.002)	0.782 (0.002)
	VQ	+	0.956 (0.003)	1.00 (0.003)	1.00 (0.003)	0.984 (0.003)
	Set Transformer	-	0.815 (0.003)	0.784 (0.002)	0.779 (0.002)	0.772 (0.002)
	K-means++	-	0.827 (0.004)	0.834 (0.003)	0.823 (0.002)	0.802 (0.001)
	GMM	-	0.842 (0.004)	0.875 (0.001)	0.867 (0.003)	0.848 (0.002)
Noise	DAF-Net	+	0.375 (0.001)	0.343 (0.001)	0.338 (0.001)	0.334 (0.001)
	LSTM	+	0.419 (0.001)	0.406 (0.001)	0.405 (0.001)	0.407 (0.001)
	Set Transformer	+	0.434 (0.001)	0.424 (0.001)	0.425 (0.001)	0.424 (0.001)
	VQ	+	1.479 (0.002)	0.948 (0.002)	0.826 (0.001)	0.720 (0.001)
	Set Transformer	-	0.436 (0.001)	0.407 (0.002)	0.398 (0.001)	0.394 (0.001)
	K-means++	-	1.836 (0.002)	1.271 (0.002)	1.091 (0.002)	0.913 (0.002)
	GMM	-	1.731 (0.002)	1.215 (0.002)	1.056 (0.002)	0.856 (0.002)

Table 1: Comparison of performance under different settings after training on different distribution for a thousand iterations. We use a total of 3 components, and train models with 30 observations. We report standard error in parentheses.

49 A.6 Model/Baseline Architectures

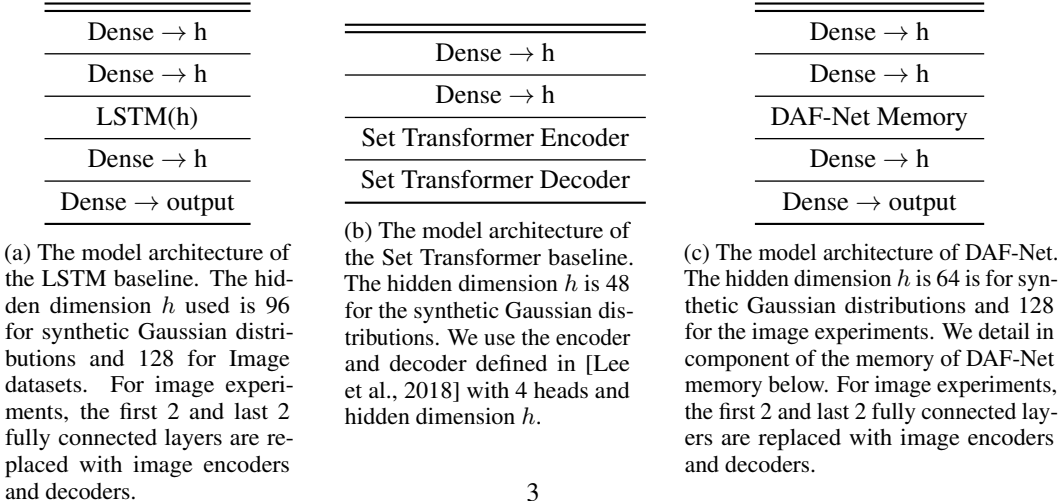


Figure A5: Architecture of different models.

5x5 Conv2d, 32, stride 2, padding 2
3x3 Conv2d, 64, stride 2, padding 1
3x3 Conv2d, 64, stride 2, padding 1
3x3 Conv2d, 64, stride 2, padding 1
3x3 Conv2d, 128, stride 2, padding 1
Flatten
Dense \rightarrow h

(a) The model architecture of the convolutional encoder for image experiments.

Dense \rightarrow 4096
Reshape (256, 4, 4)
4x4 Conv2dTranspose, 128, stride 2, padding 1
4x4 Conv2dTranspose, 64, stride 2, padding 1
4x4 Conv2dTranspose, 64, stride 2, padding 1
4x4 Conv2dTranspose, 64, stride 2, padding 1
3x3 Conv2d, 3, stride 1, padding 1

(b) The model architecture of the convolutional decoder for image experiments.

Figure A6: Architectures of encoder and decoder models on image experiments.

We provide overall architecture details for LSTM in Figure A5a, for the set-transformer in Figure A5b and DAF-Net in Figure A5c. For image experiments, we provide the architecture of the encoder in Figure A6a and decoder in Figure A6b. Both LSTM, DAF-Net, and autoencoding baselines use the same image encoder and decoder.

In DAF-Net memory, the function $\text{update}(s_k, n_k, e)$ is implemented by applying a 2 layer MLP with hidden units h which concatenates the vectors s_k, n_k, e as input and outputs a new state u_k of dimension h . The value n_k is encoded using the function $\frac{1}{1+n_k}$, to normalize the range of input to be between 0 and 1. The function $\text{attend}(s_k, n_k, e)$ is implemented in an analogous way to update, using a 2 layer MLP that outputs a single real value for each hypothesis slot.

For the function $\text{relevance}(s_k, n_k, e)$, we apply NN_1 per hypothesis slot, which is implemented as a 2 layer MLP with hidden units h that outputs a intermediate state of dimension h . (s_k, n_k, e) are fed into NN_1 in an analogous manner to update. NN_2 is applied to average of the intermediate representations of each hypothesis slot and is also implemented as a 2 layer MLP with hidden unit size h , followed by a sigmoid activation. We use the ReLU activation for all MLPs.

A.7 Baseline Details

All baseline models are trained using prediction slots equal to the ground truth of components. To modify the set transformer to act in an online manner, we follow the approach in [Santoro et al., 2018] and we apply the Set Transformer sequentially on the concatenation of an input observation with hypothesis slots. Hypothesis slots are updated based off new values of the slots after applying self-attention (Set Transformer Encoder). We use the Chamfer loss to train baseline models, with confidence set to 1.

A.8 Ablation

We investigate ablations of our model in Table 2. We ablate the components of sparsity loss, learned memory update, suppression of attention weights and relevance weights. We find that each component of our model contributes to improved performance.

Sparsity	Learned Memory	Suppression	Relevance	Observations			
				10	30	50	100
−	−	−	−	0.382 (0.003)	0.452 (0.003)	0.474 (0.003)	0.487 (0.003)
+	−	−	−	0.384 (0.001)	0.412 (0.001)	0.423 (0.001)	0.430 (0.003)
+	+	−	−	0.335 (0.002)	0.357 (0.002)	0.366 (0.003)	0.387 (0.001)
+	+	+	−	0.279 (0.001)	0.274 (0.001)	0.278 (0.001)	0.282 (0.001)
+	+	+	+	0.238 (0.001)	0.157 (0.001)	0.137 (0.001)	0.131 (0.001)

Table 2: We ablate each components of DAF-Net on the *Normal* distribution . When learned memory is ablated, DAF-Net updates states based on observed values (appropriate in the Normal Distribution dataset). We report $L_{cluster}$ of predictions and report standard error in parentheses. We find that each proposed component of our model is important for improved performance.

References

- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. *arXiv preprint arXiv:1810.00825*, 2018.
- Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. In *Advances in neural information processing systems*, pages 7299–7310, 2018.