An Explicitly Relational Neural Network Architecture: Supplementary Material

S8. Hyperparameters

Table S2 shows the default hyperparameters used for the experiments reported in the main text.

S9. Supplementary Analysis

S9.1. Dimensionality reduction on intermediate representations

To qualitatively assess the nature of the representations produced by each architecture, we performed a dimensionality reduction analysis on the outputs of the central module of each architecture trained on the ‘colour / shape’ task. After training, a batch of 10,000 images (pentominoes) was passed through the network and principal component analysis (PCA) was performed on the resulting representations, which were then projected onto the two largest principal components for visualisation. The projected representations were then colour-coded by the labels for the corresponding images (i.e. different/same shape, different/same colour).

PCA on the full representations (concatenating the head outputs in the case of the PrediNet and MHA models) did not yield any clear clustering of representations according to the labels for any of the models (Figure S8).

For the PrediNet and MHA models, we also ran separate PCAs on the output relations of each head in order to see how distributed / disentangled the representations were. While in the MHA model there was no evidence of clustering by label on any of the heads, reflecting a heavily distributed representation, there were several heads in the PrediNet architecture that individually clustered the different labels (Figure S9). In some heads, colour and shape seemed to be projected along separate axes (e.g. heads 5, 26, and 27), while in others objects with different colours seemed to be organised in a hexagonal grid (e.g. heads 9 and 14).

We noted that the clustering was preserved (though slightly compressed in PC space) when the held-out set of images (hexominoes) was passed through the PrediNet and projected onto the same principal components derived using the training set. (In Section S9.2, we show that the PrediNet heads that seem to cluster the labels also attend to the two objects in the image rather than the background.)

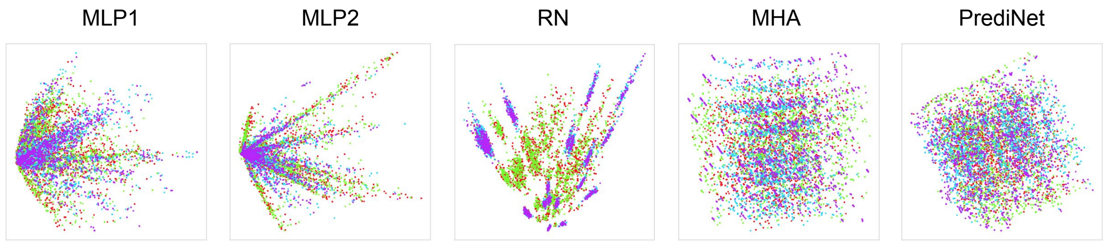
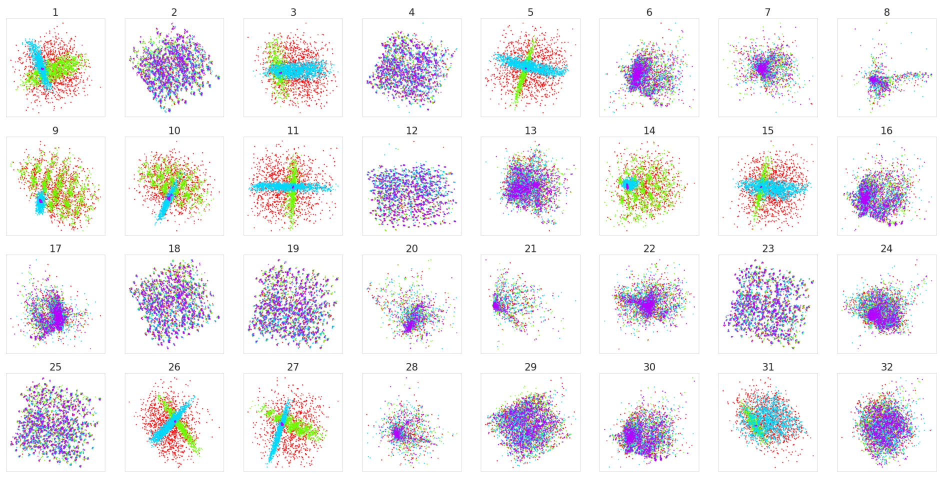


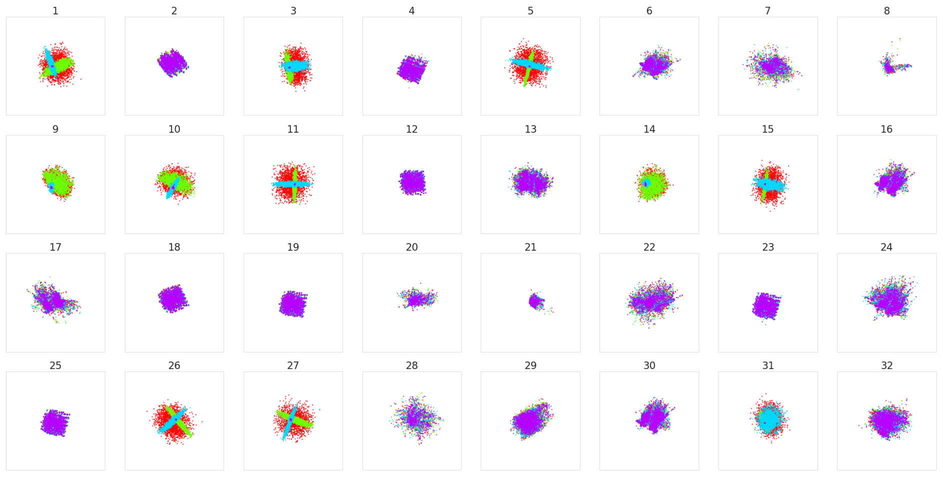
Figure S8. Representative central module outputs for networks trained on the ‘colour / shape’ task when projected onto the two largest principal components.

Table S2. Default hyperparameters

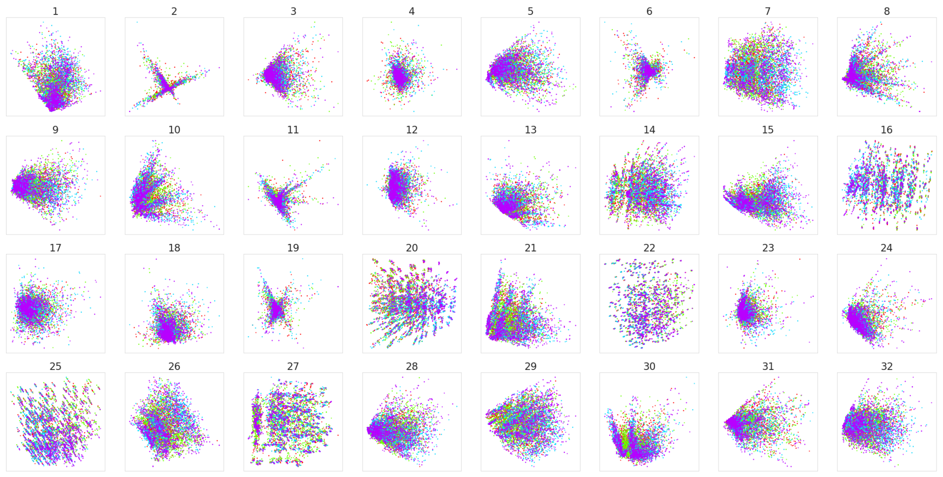
Parameter	Value
Input images size	$36 \times 36 \times 3$
L size	25×34
Runs per experiment	10
Optimiser	Gradient descent
Learning rate	0.01
Batch size	10
Input CNN output channels	32
Input CNN filter size	12
Input CNN stride	6
Input CNN activation	ReLu
Bias	Yes
Output MLP hidden layer size	8
Output MLP output size	2 (4)
Output MLP activation	ReLu
Bias	Yes (both)
MLP1 output size	$k(j+4)$
MLP1 activation	ReLu
Bias	Yes
MLP2 hidden layer size	1024
MLP2 activations	ReLu
MLP2 output size	$k(j+4)$
Bias	Yes (both)
RN MLP hidden layer size (pre-aggregation)	256
RN output size	$k(j+4)$
RN activation	ReLu
RN aggregation	Element-wise mean
Bias	No
MHA no. of heads	k
MHA key / query size	16
MHA value size	$j+4$
MHA output size	$k(j+4)$
MHA attention mechanism	$\mathrm{softmax}(QK^\top)V$
Bias	No
PrediNet no. of heads	$k=32$
PrediNet key / query size	$g=16$
PrediNet relations	$j=16$
PrediNet output size	$k(j+4)$
Bias	n/a



(a) PrediNet training set



(b) PrediNet test set



(c) MHA training set

● Same colour, same shape ● Different colour, same shape ● Same colour, different shape ● Different colour, different shape

Figure S9. Per-head PCA on the heads of a PrediNet and an MHA trained on the ‘colour / shape’ task. For all networks, PCA was performed using the training data (pentominoes). In (a) and (c), the training data are projected onto the two largest PCs and in (b) the test data (hexominoes) was used.

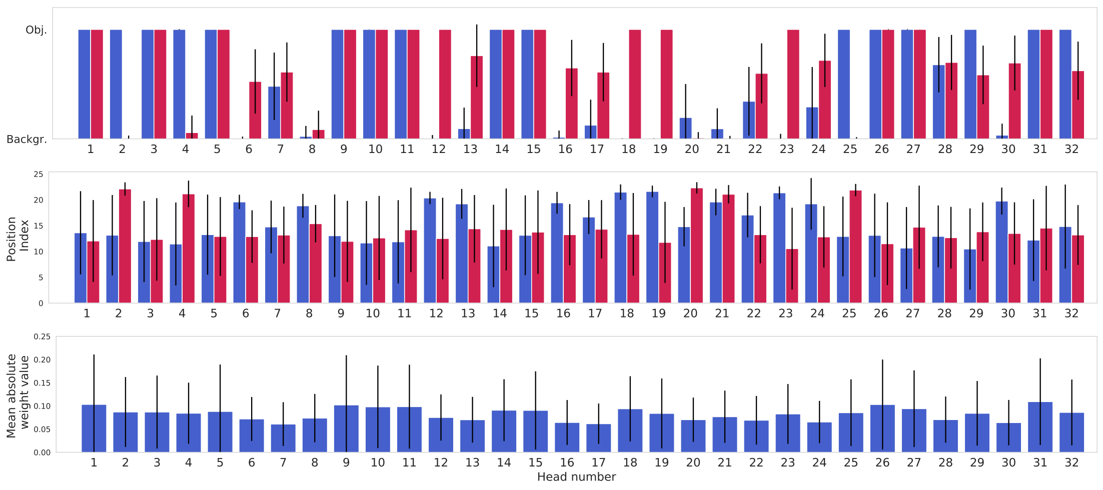


Figure S10. Top: The extent to which the two attention masks of the different heads attend to objects rather than the background. Middle: The extent to which the two attention masks of the different heads attend to specific locations in the image. Bottom: Mean absolute value of the weights from the different PrediNet heads to the output MLP.

S9.2. Attention analysis

To assess the extent to which the various PrediNet heads attend to actual objects as opposed to the background, we produced a lower resolution *content* mask for each image (with the same resolution as the attention mask) containing 0.0s at locations where there are no objects in the corresponding pixels of the full image, 1.0s where more than 90% of the pixels contain an object, and 0.5s otherwise. By applying the attention mask to the content mask, and summing the resulting elements, we produced a scalar indicating whether the attention mask was selecting a region of the image with an object (value close to 1.0), or the background (value close to 0.0). This was tested over 1000 images from the training set (Fig. S10), but similar results are obtained if the held-out set images are used instead. The top plot in Fig. S10 shows that both attention masks of some heads consistently attend to objects, while others to a combination of object and background. Importantly, the heads for which the PCA meaningfully clusters the labels are also the ones in which both attention masks attend to objects (Fig. S9).

We additionally provide a similar analysis with a *position* mask, where each pixel in the mask contains a unique location index. The middle plot in Fig. S10 shows that the attention masks in the majority of the heads do not consistently attend to specific locations. Finally, the mean absolute values of the per-head input weights to the output MLP are shown in the bottom plot of the same figure. Interestingly, the heads that consistently attend only to objects have higher weighting than the rest.

S10. Experimental Variations

Further experimental results are provided in this section, including variations in hyper-parameters. Fig. S11 presents test accuracy curves for the ‘stripes’ object set, for which a summary is presented in Table 1 of the main text. Fig. S12 shows the results on the same experiment but using the Adam optimiser instead of SGD, with a learning rate of 10^{-4}. The TensorFlow default values for all other Adam parameters were used. While other learning rate values were also tested, a value of 10^{-4} gave the best overall performance for all architectures. Multi-task experiments were also performed using Adam with the same learning rate (Fig. S14 & S15), yielding an overall similar performance to SGD with a learning rate of 10^{-2}.

To assess the extent to which the number of heads and relations plays a role in the performance, we ran experiments with $k = 64$ heads and $j = 16$ relations (Fig. S16 & S17), as well as $k = 16$ heads and $j = 32$ relations (Fig. S18 & S19). The results indicate that having a greater number of heads leads to better performance than having a greater number of relations, because they provide more stability during training and, perhaps, a richer propositional representation.

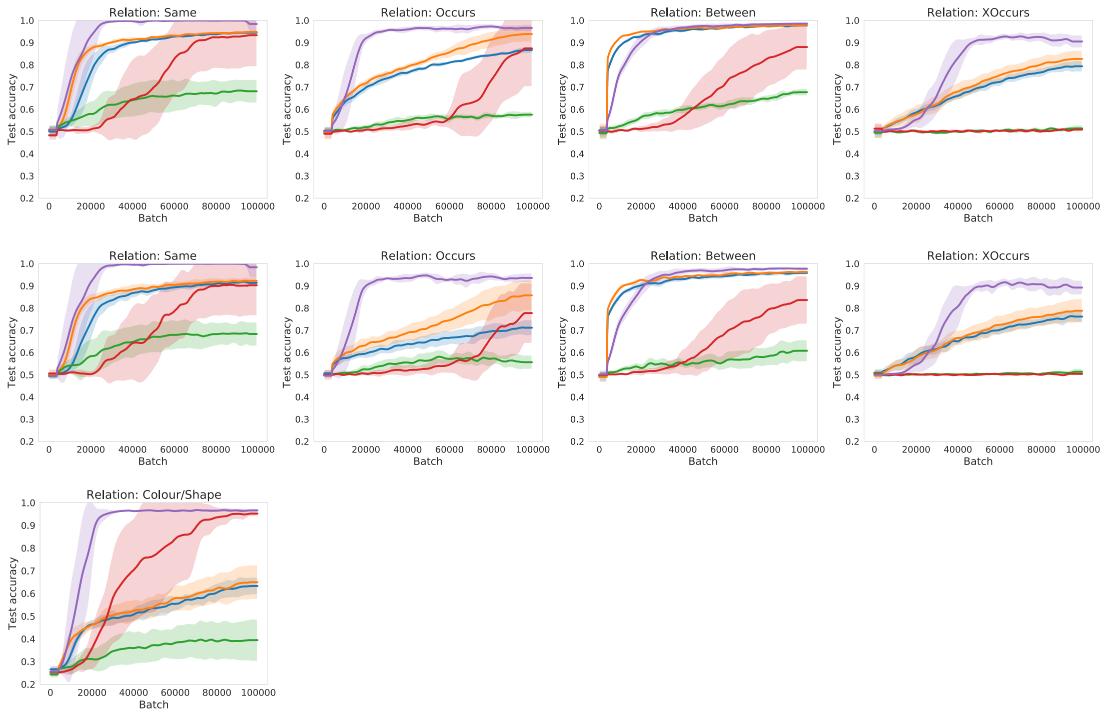


Figure S11. Relations Game learning curves for the different models. SGD with a learning rate of 0.01 was used with a PrediNet of $k = 32$ heads and $j = 16$ relations. The top and bottom rows show results for the 'hexominoes' held-out object set, while the middle row is for the 'stripes' held-out object set. The results for the top 10 batches are summarised in Table 1 of the main manuscript.

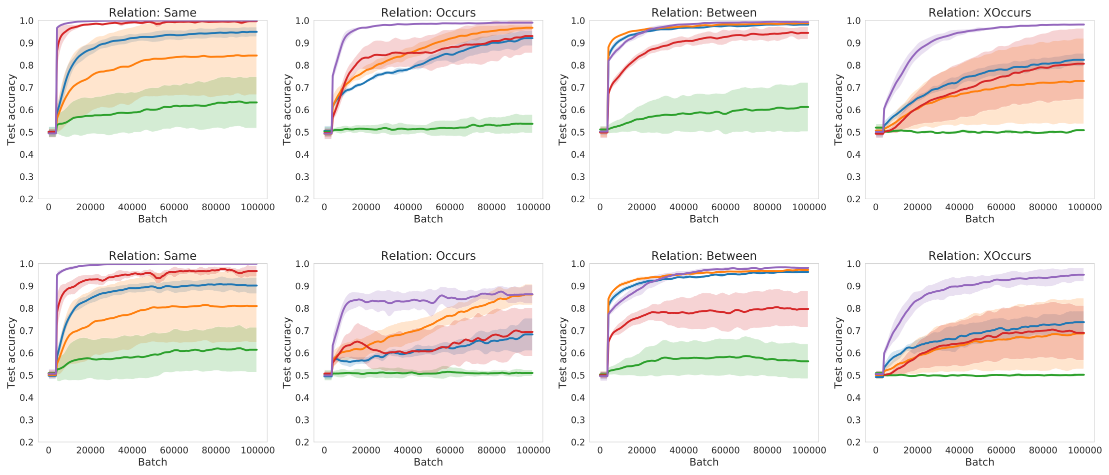


Figure S12. Relations Game learning curves for the different models trained with the Adam optimiser (learning rate: 10^{-4}). All other experimental parameters are the same as Fig. S11. The top row shows results for the 'hexominoes' held-out object set, while the bottom row is for the 'stripes' held-out object set

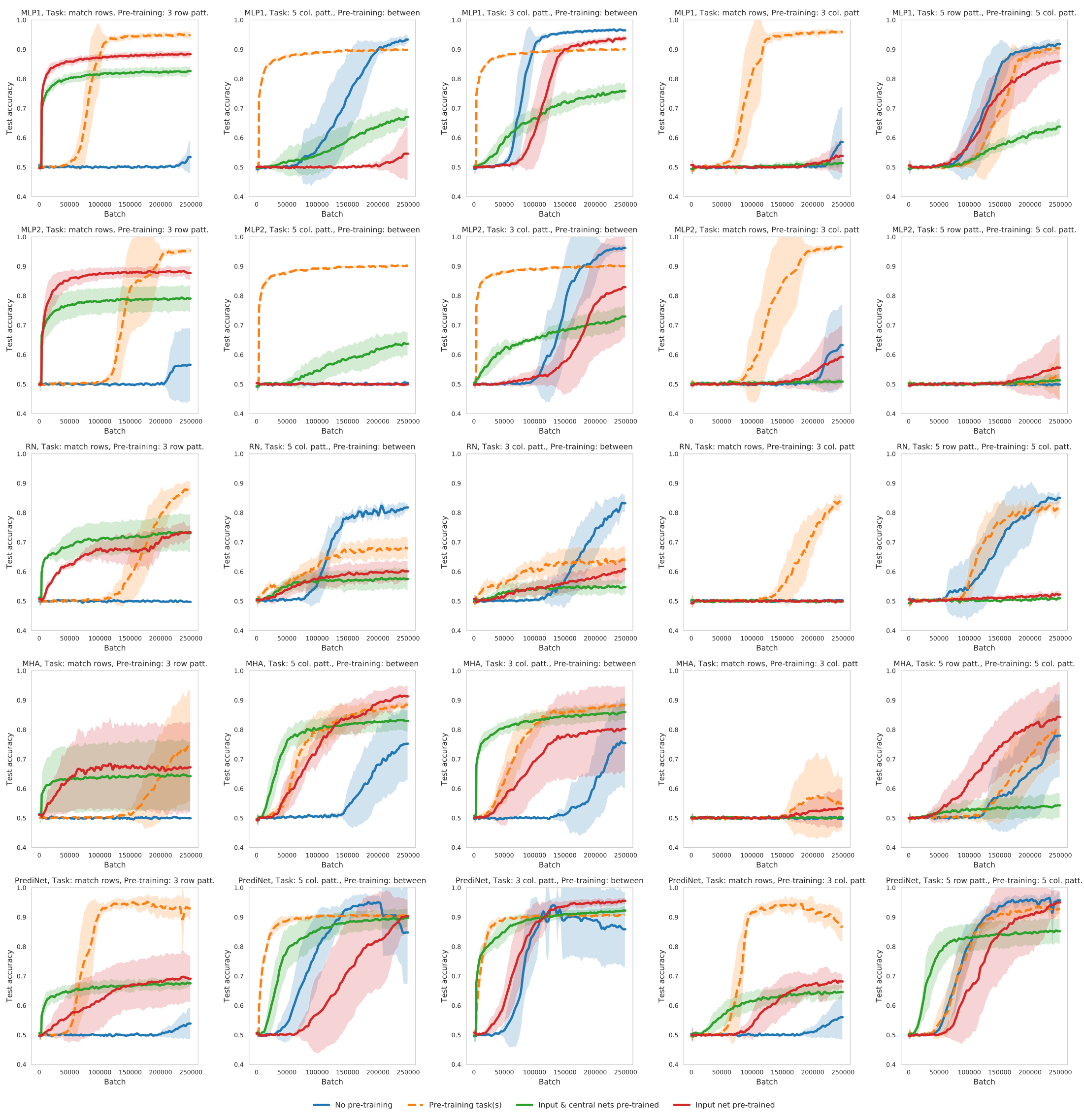


Figure S13. Multi-task curriculum training. The columns correspond to different target / pre-training task combinations, while the rows correspond to the different architectures. SGD with a learning rate of 0.01 was used, with $k = 32$ and $j = 16$. Training was performed using the pentominoes object set and testing using the 'stripes' object set. From left to right, the combinations of target / pre-training tasks are: ('match rows', '3 row patterns'), ('5 column patterns', 'between'), ('3 column patterns', 'between'), ('match rows', '3 column patterns') and ('5 row patterns', '5 column patterns'). From top to bottom, the different architectures are: MLP1, MLP2, relation net (RN), multi-head attention (MHA) and PrediNet.

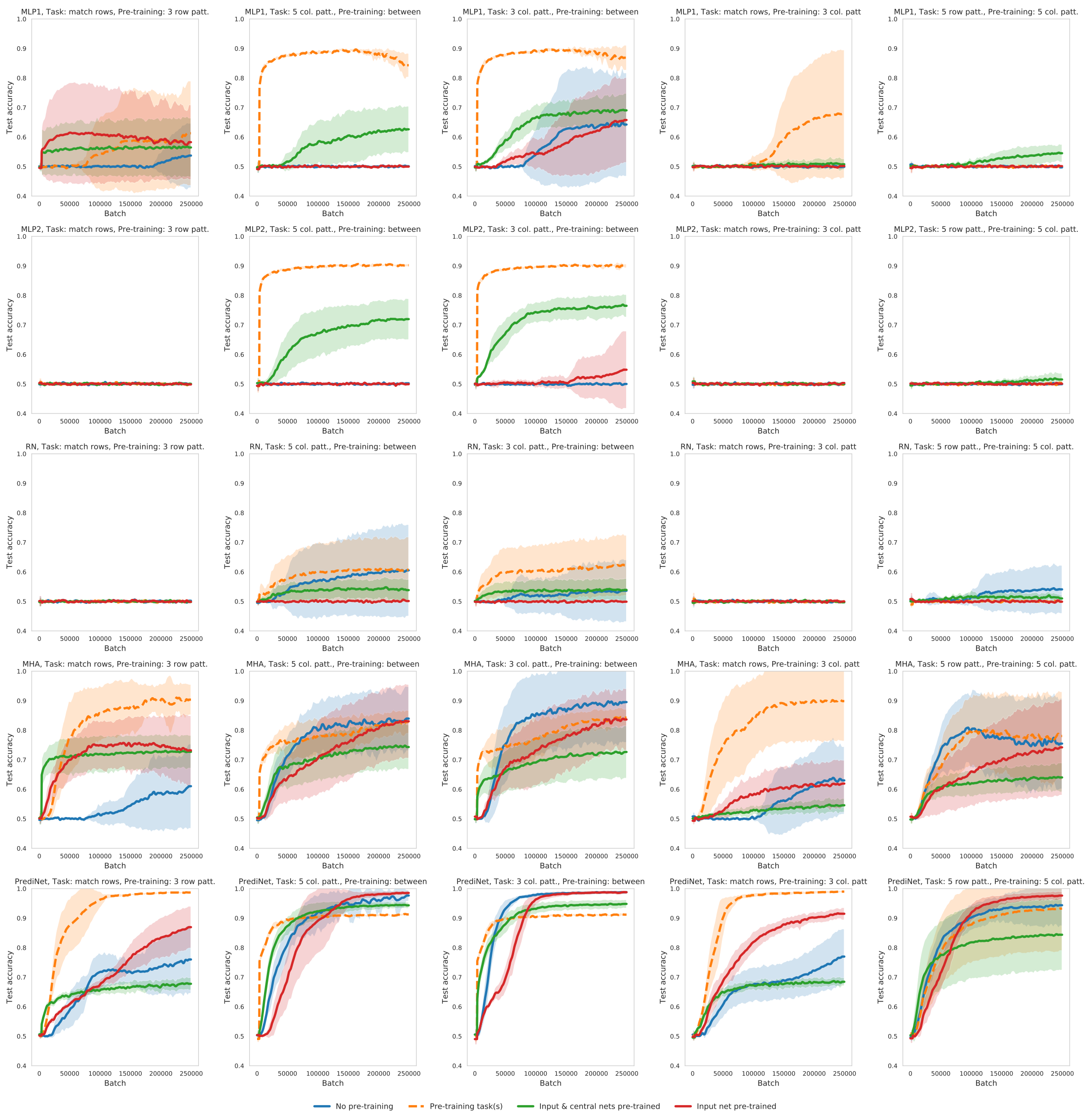


Figure S14. Multi-task curriculum training. The columns correspond to different target / pre-training task combinations, while the rows correspond to the different architectures, as in Fig. S13. The Adam optimiser with a learning rate of 10^{-4} was used. Training was performed using the pentominoes object set and testing using the 'stripes' object set.

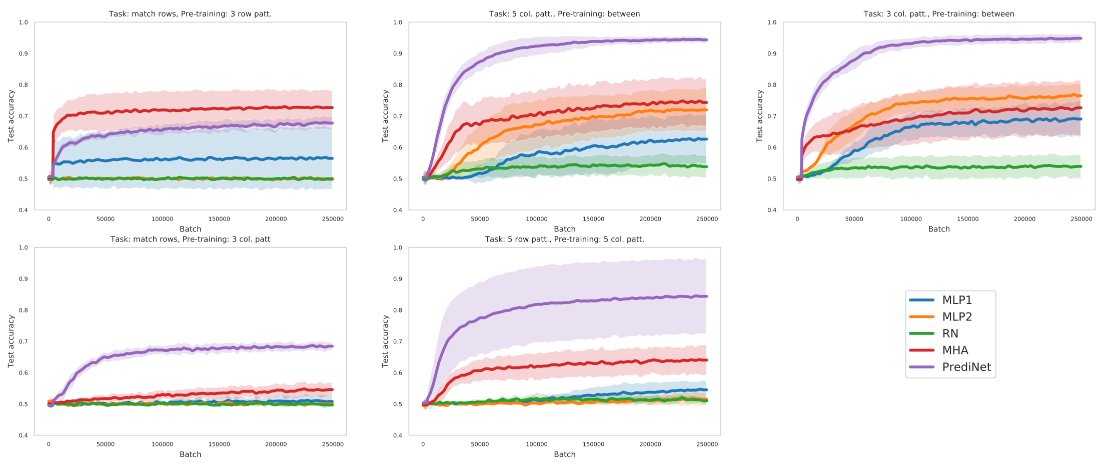


Figure S15. Reusability of representations learned with a variety of target and pre-training tasks, using the 'stripes' object set. All architectures were trained using Adam, with a learning rate of 10^{-4}. The experimental setup is the same as in Fig. S14.

S11. Atari Experiments

To show that the PrediNet serves as a generic neural network module, we evaluated it in a reinforcement learning (RL) setting. In the longer term, we envisage embedding explicitly relational neural network modules like the PrediNet in architectures that are tailored to bring out their advantages. But for the present evaluation, we built a strong baseline RL agent based on an established architecture (Espeholt et al., 2018). See Table S3 for details. We compared this to an equivalent agent with the core parameter-heavy, fully connected layer replaced by a PrediNet module, while preserving input and output sizes and all other parameters (Fig. S20). The PrediNet module we used is a slightly modified version of the one in Fig. 1. Specifically, to cope with higher-dimensional input, the queries Q_1^h and Q_2^h are computed via a three-layer convolutional net rather than by a single linear layer. In all other respects, the modified PrediNet module conforms to the blueprint of Fig. 1.

We ran the resulting two agents on the standard suite of 57 Atari games (three runs each). The performance of the two agents after 1e9 steps was comparable, producing very similar mean human-normalised scores for 53 out of 57 games (Fig. S21). The baseline agent outperformed the PrediNet agent on three games (Frostbite, James Bond, and Montezuma's Revenge), while the PrediNet agent outperformed the baseline on Amidar. However, the PrediNet agent used significantly fewer parameters than the baseline (approx. 1.6 million versus approx. 5.1 million), and generated interpretable attention masks comparable to those of (Mott et al., 2019) (Fig. S22).

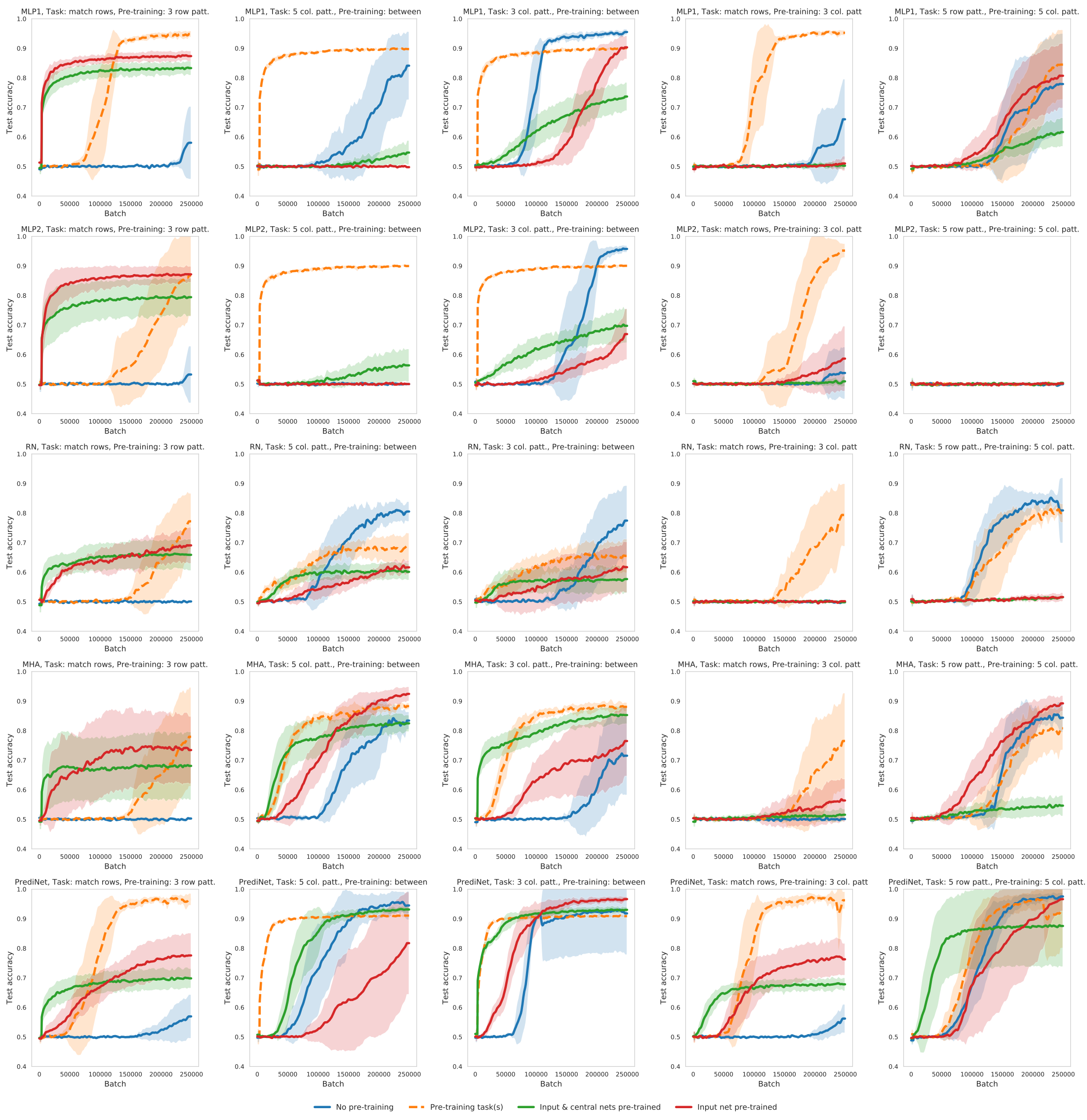


Figure S16. Multi-task curriculum training. The columns correspond to different target/pre-training task combinations, while the rows correspond to the different architectures. SGD with a learning rate of 0.01 was used. Training was performed using the pentominoes object set and testing using the ‘stripes’ object set. The experimental setup is the same as for Fig. S13, except that $k = 64$ and $j = 16$. Increasing the number of heads for the PrediNet increases the stability during training and overall performance.

Table S3. RL architecture parameters

Module/Parameters	Baseline	PrediNet
Optimiser	RMS Prop	
Learning rate	10^{-4}	
Reward discount factor	0.99	
Action Repeat	4	
Vision Module		
Number of channels	[16, 32 , 32]	
Filter size	[3, 3 , 3]	
Stride	[1, 1 , 1]	
Pooling type	Max pooling	
Residual blocks after pooling layer	[2, 2, 2]	
Activation	ReLu	
Padding	SAME	
Bias	Yes	
Central Module		
Fully connected layer output size	256	–
Query CNN filter sizes	–	[32, 32, 32]
Query CNN filter size	–	[3, 3, 3]
Query CNN strides	–	[2, 2, 2]
Query CNN activation	–	ReLu
Query CNN padding	–	VALID
Query CNN Bias	–	Yes
Query Linear layer output size	–	16
Key size	–	16
Number of heads	–	16
Number of relations	–	64
LSTM Layer		
Hidden size	256	
Value Head		
MLP output size	[128, 1]	
Activation	ReLu	
Bias	Yes	
Policy Head		
MLP output size	[64, 18]	
Activation	ReLu	
Bias	Yes	
Total Number of Parameters	**5,098,515**	**1,590,355**

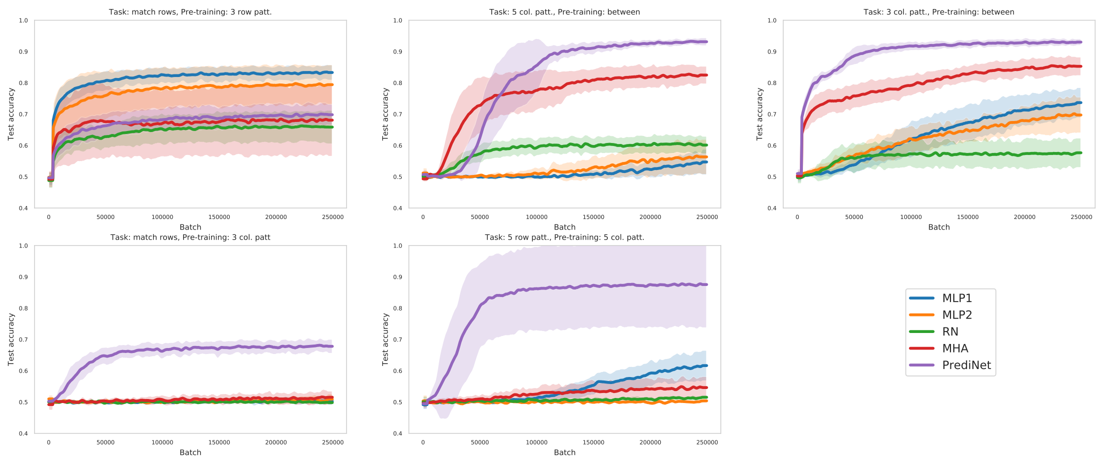


Figure S17. Reusability of representations learned with a variety of target and pre-training tasks, using the 'stripes' object set. All experimental parameters are as in Fig. S16.

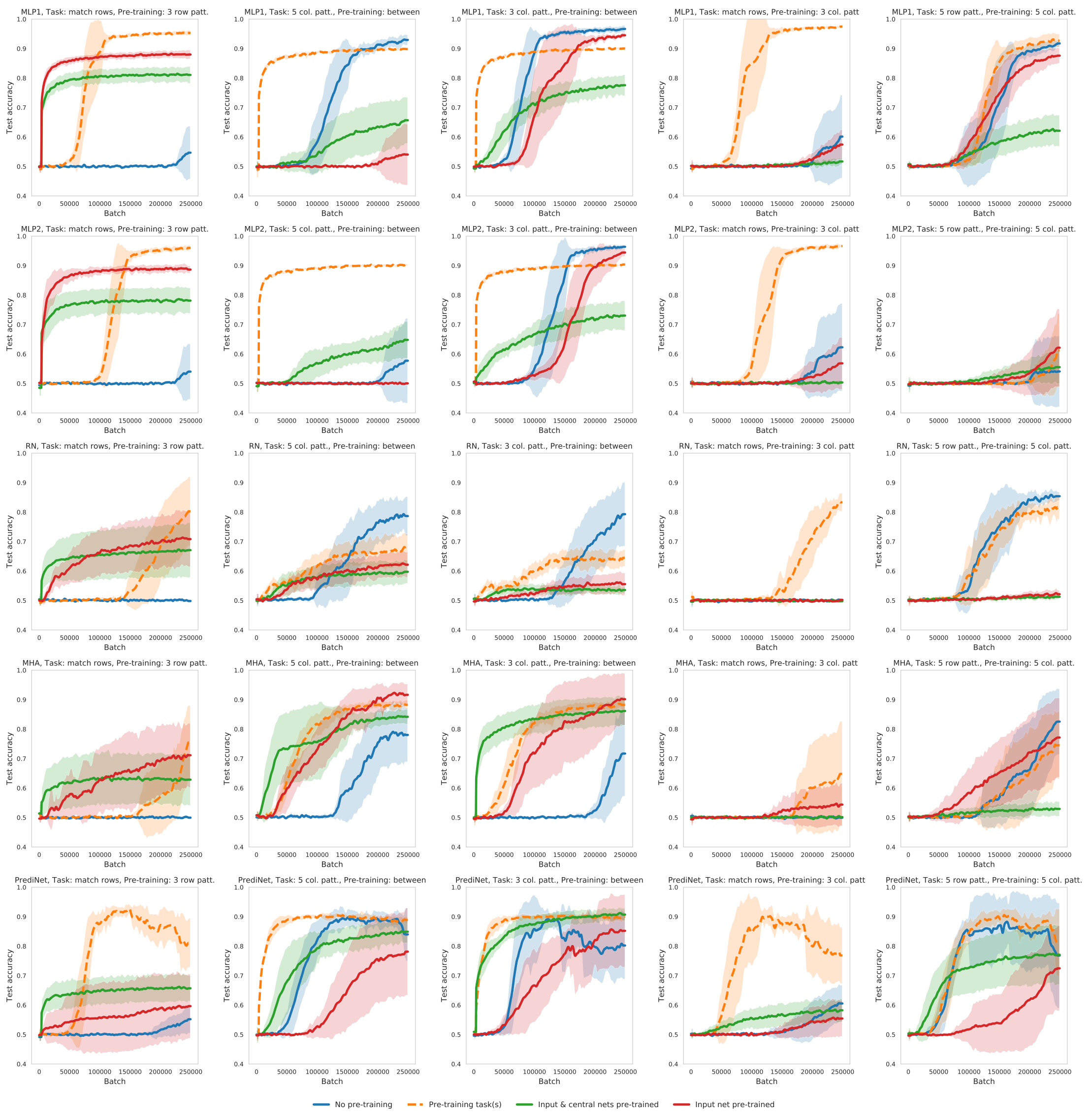


Figure S18. Multi-task curriculum training. The columns correspond to different target / pre-training task combinations, while the rows correspond to the different architectures. SGD with a learning rate of 0.01 was used. Training was performed using the pentominoes object set and testing using the 'stripes' object set. The experimental setup is the same as for Fig. S13, except that $k = 16$ and $j = 32$. Having fewer heads leads to a decrease in performance, even if the number of relations is increased to maintain network size.

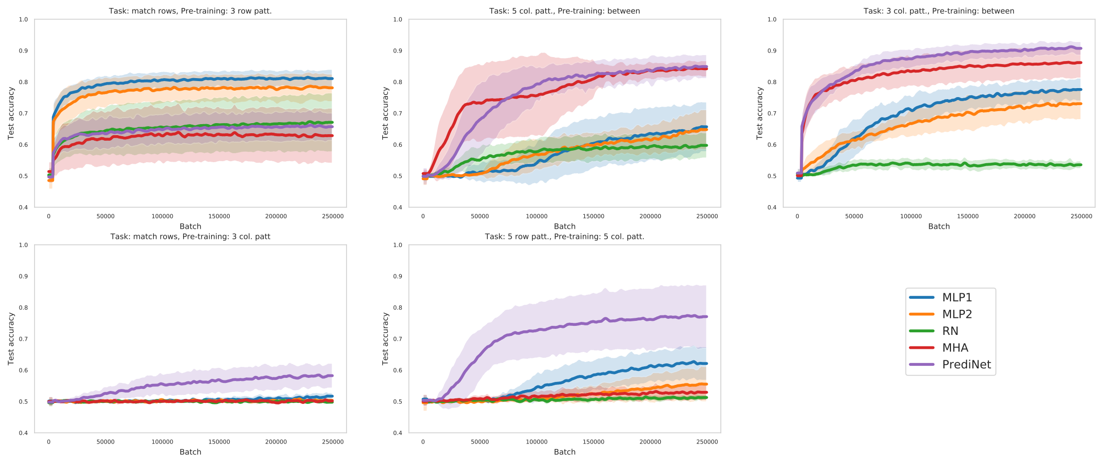


Figure S19. Reusability of representations learned with a variety of target and pre-training tasks, using the ‘stripes’ object set. All experimental parameters are as in Fig. S18.

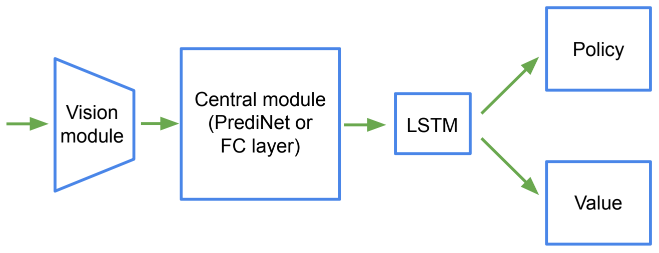


Figure S20. RL agent architecture. The PrediNet agent is obtained by taking a strong baseline and replacing its central fully-connected layer with an equivalent PrediNet module, preserving all other parameters. See Table S3.

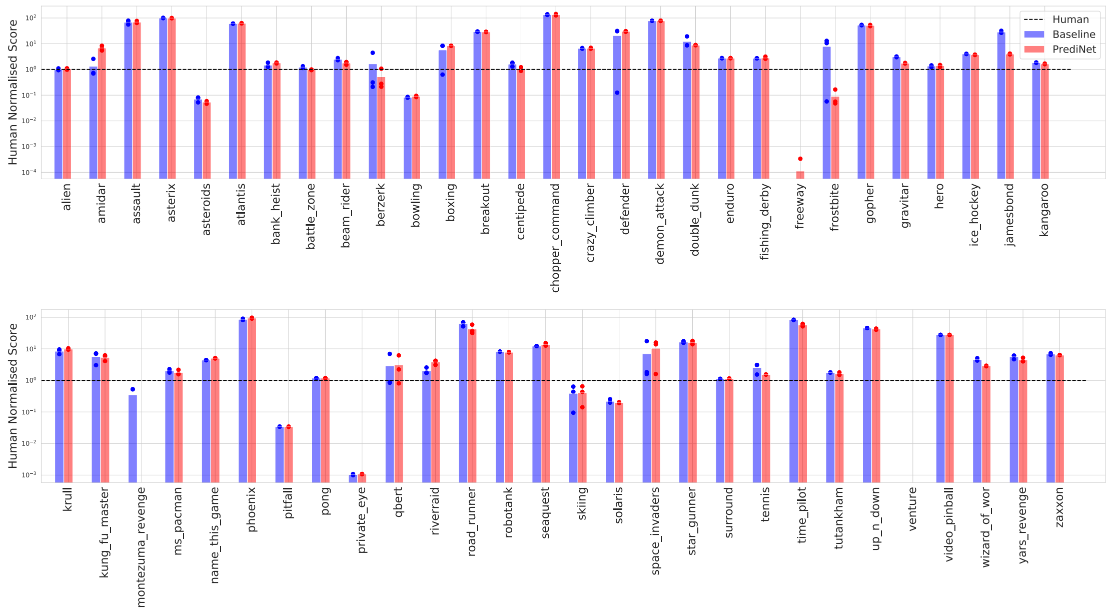


Figure S21. Human-normalised Atari scores of the PrediNet agent compared to a strong baseline (see Fig S20 and Table S3). Means of three runs per game are shown. Individual runs are depicted as dots. The performance of the two agents is comparable.

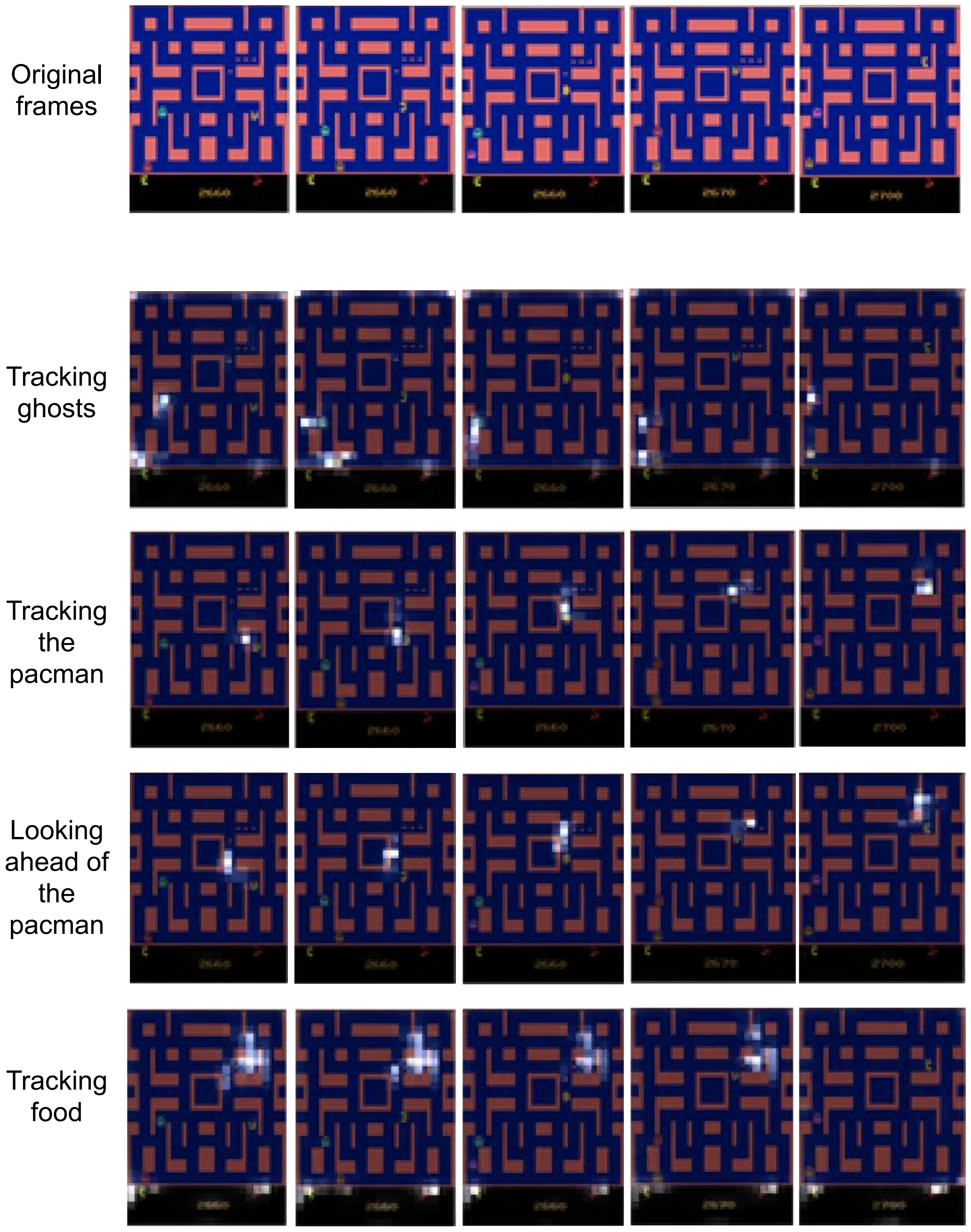


Figure S22. Attention on Ms Pacman. The top row shows a sequence of five frames, approximately 20 frames apart. The other rows depict the attention masks computed by 4 selected PrediNet heads (out of 16) of a selected PrediNet agent. Each head computes two attention masks, picking out a pair of objects, which are here overlaid. The heads are clearly attending to meaningful objects with respect to the gameplay (cf. (Mott et al., 2019)).