Visual Knowledge Tracing - Supplementary Material

Neehar Kondapaneni[1] Pietro Perona[1] Oisin Mac Aodha[2]

[1]Caltech [2]University of Edinburgh

A Additional Experiments

In this section, we present several additional models and also consider the impact of embedding dimension on performance.

A.1 RNN Variants

We can vary the input information to both of the recurrent models in three ways. The notation in parentheses maps to the entries in the later supplementary tables.

(base): The models only receive the history of images, ground truth class labels, and learner responses

$$\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t})). \tag{1}$$

(y): In addition to the history, the model receives the ground truth class label of the image shown to the learner at the current time-step

$$\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t}, y)). \tag{2}$$

(y, $\mathbf{z}$): Finally, as in the main paper, the model can include both the ground truth class of the image and the representation of the image from the learned CNN

$$\psi_{rnn}(\mathbf{z}^k_{1:t}, y^k_{1:t}, r^k_{1:t}, \mathbf{z}, y)). \tag{3}$$

The results of the variants are presented in Tables A1, A2, and A3.

A.2 DKT

Next, we adapt Deep Knowledge Tracing (DKT) [5] to our setting. We deviate from the original DKT method in two main ways. First, the types of queries (e.g. math problems) in educational datasets do not allow for instance-level representations. Instead, skills (i.e. question types) were jointly encoded with information about whether the problem was answered correctly by the learner. Second, the output of DKT was the learner's probability of being correct for each skill, not a particular question instance.

We modify the DKT algorithm to make it appropriate for the setting described in our work. We replace skills with the class-level label for an image and convert the output into a probability distribution over the class labels such that it can be trained with the cross-entropy loss

$$\phi_{dkt}(y) = \sigma(\psi_{rnn}(y^k_{1:t}, r^k_{1:t}, y))). \tag{4}$$

At a high-level, this model variant encodes no instance-level (i.e. image) information to make it's predictions.

We observe that this DKT model (ϕ_{dkt}) performs slightly worse in all cases, indicating that image information is valuable to enable the models to better trace learner performance. The results of the DKT variant are presented in Tables A1, A2, and A3.

A.3 Cognitive Models

Cognitive models make stronger assumptions on how humans learn. We modify the prototype and exemplar models described in the cognitive science literature [4] and evaluate them on our datasets.

Prototype. The prototype model proposes that learners store a prototypical image for each class. Each new image is compared to the learner's class prototypes and the highest similarity class is selected. In our formulation, the class prototype is the average feature representation of previously seen images of that class. In the following equations, τ is the current time-step, $P^k_\tau(c)$ is the prototype of class c for learner k at time-step τ, and δ is the dirac-delta function and acts as a selector for images from class c,

$$P^k_\tau(c) = \frac{1}{\tau - 1} \cdot \sum_t^{\tau-1} z^k_t * \delta(y^k_\tau - c), \tag{5}$$

$$\hat{r}^k_\tau(c) = \frac{sim(P^k_\tau(c), z^k_\tau)}{\sum_c sim(P^k_\tau(c), z^k_\tau)}. \tag{6}$$

Exemplar. The exemplar model proposes that learners store previously seen images in a memory bank of exemplars. Query images are compared to all of the exemplars. The learner chooses the class with the highest total similarity to the query image. In the following equations, $E^k_\tau(c)$ is the sum of the class c similarity scores for learner k at time-step τ with respect to the current image z^k_τ. Following [4], we introduce a learnable parameter γ to scale the similarities (this value is fixed to 1 in the prototype model),

$$E^k_\tau(c) = \sum_t^{\tau-1} sim(z^k_t, z^k_\tau) \cdot \delta(y^k_\tau - c), \tag{7}$$

$$\hat{r}_{\tau}^{k}(c) = \frac{sim(E_{\tau}^{k}(c), z_{\tau}^{k})^{\gamma}}{\sum_{c} sim(E_{\tau}^{k}(c), z_{\tau}^{k})^{\gamma}}. \quad (8)$$

Both models compute similarity by using an exponential decay function over the Euclidean distance between feature representations of the images,

$$sim(z_i, z_j) = e^{-c*d(z_i, z_j)}. \quad (9)$$

Finally, instead of learning the feature space separately with visual similarity experiments, we jointly estimate a CNN along with the model parameters to discover the feature space.

We find that these models perform worse than the models presented in the main paper. However, simple modifications (like weighting the history of exemplars or images in the prototype) may help. Exploring the space of cognitive models is an interesting direction for future work. The results of these variants are presented in Tables A1, A2, and A3.

A.4 Transformers

Recently, the knowledge tracing community has found the Transformer architecture to be an effective model for tracing human learners in non-visual tasks. We modify the SAINT model [1] for our visual learning setting. First, we introduce a CNN-based feature extraction stage to embed images. The encoder receives the current image's embedding and its ground truth label. The decoder gets the previous learner response. The decoder predicts the learner's response to the image (also passed to the encoder).

The Transformer model does surprisingly poorly on these datasets. We expect that future work exploring Transformer architectures designed for this task will demonstrate performance on par with the recurrent models. The results of the Transformer model are presented in Tables A1, A2, and A3.

A.5 ResNet Backbone Experiments

We swap out our CNN backbone with a ResNet-18 [2] pre-trained on ImageNet. We freeze the weights in layers 1, 2 and 3, but leave layer 4 to be learned. The output of layer 4 is passed to a fully-connected layer that reduces the output of the layer to the desired dimensionality, as opposed to the final classifier used for the ImageNet classification task. The results of these experiments are presented in Table A4.

A.6 Including Per-Class Accuracy as Input

We find that including some meta-information can help with tracing performance. To do this, we compute a learner's accuracy on each class at each timestep and concatenate this vector to the input of three tracing models (ϕ_{static}, ϕ_{direct} and ϕ_{cls_pred}). We find a boost in performance across all models. The results are presented in Table A5, where we observe a boost in performance. It is likely that other sources of meta-information (such as time-taken on an example) will also help [6].

A.7 Varying Embedding Dimension

We demonstrate that varying the embedding dimension of the feature extractor has little effect on the performance of the direct response model (Table A6).

Table A1. Performance of all model variants on the Butterflies dataset. The model variant is denoted in the subscript corresponding to the same subscripts in A.1. One can see that $\phi_{direct(base)}$ performs poorly for a recurrent model. This model does not have access to any information about the current time-step and is effectively guessing both the image that will be shown and the associated response. We also show the per-class average precision scores on the train sequence in addition to the micro and macro scores from before. These scores show that the benefit of the recurrent models appear primarily in classes that have large changes in average performance (e.g. Red Admiral) over the training period. The models with † are models presented in Table 1 of the main paper. The scores are reported with their standard deviations and the top average performers in each column are in bold.

	Butterflies								
	Train							Test	
	Cabbage White	Monarch	Queen	Red Admiral	Viceroy	Micro	Macro	Micro	Macro
GT Label†	0.94±0.03	0.32±0.01	0.36±0.04	0.65±0.04	0.27±0.01	0.48±0.02	0.51±0.02	0.58±0.02	0.61±0.02
ϕ_{static}†	0.95±0.01	0.37±0.02	0.34±0.06	0.66±0.03	0.27±0.03	0.63±0.02	0.52±0.02	0.67±0.02	0.58±0.03
ϕ_{static_time}†	0.97±0.01	0.34±0.03	0.29±0.03	0.35±0.03	0.24±0.02	0.52±0.02	0.44±0.01	0.49±0.01	0.40±0.01
ϕ_{dkt}	0.95±0.02	0.43±0.02	0.45±0.04	0.72±0.05	0.33±0.06	0.67±0.02	0.57±0.02	0.74±0.02	0.64±0.02
$\phi_{transformer}$	0.96±0.01	0.36±0.03	0.34±0.02	0.69±0.06	0.26±0.01	0.62±0.02	0.52±0.02	0.68±0.05	0.58±0.04
$\phi_{prototype}$	0.95±0.02	0.32±0.03	0.30±0.05	0.59±0.08	0.27±0.03	0.53±0.03	0.48±0.03	0.63±0.01	0.54±0.02
$\phi_{exemplar}$	0.90±0.03	0.30±0.03	0.27±0.04	0.35±0.13	0.26±0.03	0.44±0.05	0.42±0.04	0.53±0.07	0.45±0.07
$\phi_{direct(base)}$	0.34±0.02	0.29±0.03	0.23±0.02	0.26±0.02	0.22±0.03	0.27±0.01	0.27±0.01	0.20±0.01	0.20±0.01
$\phi_{direct(y)}$	0.97±0.02	0.43±0.03	**0.48±0.05**	0.76±0.07	**0.38±0.04**	**0.71±0.02**	0.60±0.02	**0.78±0.02**	**0.66±0.01**
$\phi_{direct(y,\mathbf{z})}$†	0.97±0.01	0.41±0.04	0.44±0.06	0.77±0.07	0.36±0.03	0.70±0.03	0.59±0.02	0.77±0.03	0.64±0.03
$\phi_{cls\_pred(base)}$	0.96±0.01	0.41±0.06	0.32±0.05	0.59±0.14	0.25±0.02	0.59±0.06	0.51±0.05	0.61±0.06	0.52±0.05
$\phi_{cls\_pred(y)}$†	**0.98±0.01**	**0.46±0.02**	**0.48±0.04**	**0.78±0.05**	**0.38±0.02**	**0.71±0.02**	**0.62±0.01**	0.77±0.02	0.65±0.02
$\phi_{cls\_pred(y,\mathbf{z})}$	**0.98±0.01**	0.45±0.01	0.47±0.04	**0.78±0.05**	0.37±0.02	0.70±0.02	0.61±0.01	0.77±0.03	**0.66±0.02**

Table A2. Performance of all model variants on the Eyes dataset. Please see the caption of Table A1 for more details.

	Eyes						
	Train					Test	
	DME	Drusen	Normal	Micro	Macro	Micro	Macro
GT Label†	0.56±0.02	0.54±0.03	0.58±0.02	0.56±0.02	0.56±0.02	0.69±0.02	0.69±0.01
ϕ_{static}†	0.63±0.03	0.53±0.05	0.62±0.02	0.60±0.03	0.59±0.03	0.67±0.02	0.68±0.02
ϕ_{static_time}†	0.32±0.01	0.35±0.01	0.36±0.01	0.34±0.00	0.34±0.01	0.33±0.01	0.34±0.02
ϕ_{dkt}	0.63±0.02	0.60±0.03	0.65±0.02	0.63±0.02	0.63±0.02	0.74±0.03	0.73±0.03
$\phi_{transformer}$	0.41±0.09	0.41±0.05	0.42±0.09	0.41±0.08	0.41±0.08	0.37±0.02	0.39±0.02
$\phi_{prototype}$	0.61±0.04	0.50±0.04	0.56±0.04	0.56±0.03	0.56±0.03	0.65±0.04	0.65±0.05
$\phi_{exemplar}$	0.57±0.02	0.48±0.02	0.59±0.03	0.54±0.02	0.55±0.02	0.68±0.04	0.67±0.04
$\phi_{direct(base)}$	0.38±0.02	0.40±0.01	0.38±0.02	0.38±0.01	0.39±0.01	0.35±0.01	0.34±0.01
$\phi_{direct(y)}$	**0.65±0.03**	**0.62±0.03**	0.68±0.03	**0.66±0.02**	**0.65±0.02**	**0.75±0.01**	0.73±0.02
$\phi_{direct(y,\mathbf{z})}$†	0.64±0.02	**0.62±0.03**	**0.69±0.02**	**0.66±0.02**	**0.65±0.02**	**0.75±0.01**	**0.74±0.02**
$\phi_{cls\_pred(base)}$	0.48±0.05	0.39±0.02	0.48±0.05	0.45±0.04	0.45±0.03	0.44±0.02	0.44±0.02
$\phi_{cls\_pred(y)}$†	**0.65±0.02**	**0.62±0.03**	**0.69±0.01**	0.65±0.03	**0.65±0.02**	0.74±0.02	**0.74±0.04**
$\phi_{cls\_pred(y,\mathbf{z})}$	0.64±0.01	**0.62±0.03**	**0.69±0.02**	0.65±0.02	**0.65±0.02**	**0.75±0.01**	**0.74±0.02**

Table A3. Performance of all model variants on the Greebles dataset. Please see the caption of Table A1 for more details.

	Greebles						
	Train					Test	
	Agara	Bari	Cooka	Micro	Macro	Micro	Macro
GT Label†	0.51±0.02	0.37±0.03	0.43±0.03	0.45±0.02	0.44±0.02	0.50±0.01	0.49±0.01
ϕ_{static}†	0.63±0.03	**0.43±0.04**	0.55±0.05	**0.55±0.03**	**0.53±0.03**	**0.64±0.01**	**0.61±0.01**
ϕ_{static_time}†	**0.64±0.04**	0.39±0.02	0.54±0.04	0.54±0.03	0.52±0.03	0.61±0.01	0.59±0.02
ϕ_{dkt}	0.59±0.03	0.41±0.03	0.49±0.05	0.52±0.02	0.50±0.02	0.59±0.02	0.55±0.02
$\phi_{transformer}$	0.52±0.11	0.36±0.03	0.45±0.05	0.46±0.07	0.45±0.06	0.44±0.08	0.43±0.08
$\phi_{prototype}$	0.58±0.03	0.42±0.01	0.54±0.05	0.52±0.02	0.51±0.02	0.58±0.02	0.57±0.02
$\phi_{exemplar}$	0.59±0.02	**0.43±0.03**	0.52±0.05	0.52±0.03	0.51±0.03	0.63±0.01	0.60±0.01
$\phi_{direct(base)}$	0.37±0.02	0.35±0.02	0.36±0.01	0.36±0.00	0.36±0.00	0.34±0.02	0.34±0.02
$\phi_{direct(y)}$	0.59±0.02	0.41±0.03	0.51±0.04	0.52±0.02	0.50±0.03	0.59±0.02	0.55±0.02
$\phi_{direct(y,\mathbf{z})}$†	0.62±0.03	0.42±0.03	0.55±0.05	**0.55±0.03**	**0.53±0.03**	0.60±0.02	0.57±0.03
$\phi_{cls\_pred(base)}$	0.63±0.03	0.40±0.02	**0.56±0.06**	**0.55±0.02**	**0.53±0.03**	0.61±0.02	0.60±0.03
$\phi_{cls\_pred(y)}$†	0.62±0.03	0.41±0.03	0.54±0.03	0.54±0.02	0.52±0.03	0.60±0.02	0.57±0.03
$\phi_{cls\_pred(y,\mathbf{z})}$	0.63±0.03	0.41±0.03	0.55±0.03	**0.55±0.02**	**0.53±0.01**	0.61±0.02	0.59±0.02

Table A4. Performance of models trained using a pre-trained ResNet with partially frozen weights (as described in Sec. A.5). We only compare model variants that appear in the main text. Similar to the original experiment results the classifier prediction model (ϕ_{cls_pred}) performs the best. However, the overall performance decreases slightly across the board. We observe a larger decrease for the direct response model (ϕ_{direct}), likely due to the larger dependence that it has on the feature space.

	Butterflies								
	Train							Test	
	Cabbage White	Monarch	Queen	Red Admiral	Viceroy	Micro	Macro	Micro	Macro
GT Label	0.94±0.03	0.32±0.01	0.36±0.04	0.65±0.04	0.27±0.01	0.48±0.02	0.51±0.02	0.58±0.02	0.61±0.02
ϕ_{static}	0.95±0.01	0.36±0.02	0.32±0.04	0.65±0.04	0.27±0.03	0.62±0.02	0.51±0.02	0.67±0.03	0.57±0.03
ϕ_{static_time}	0.44±0.02	0.25±0.02	0.22±0.01	0.33±0.03	0.19±0.02	0.28±0.01	0.29±0.00	0.27±0.02	0.34±0.02
ϕ_{direct}	**0.97±0.01**	**0.41±0.03**	0.38±0.06	0.74±0.07	0.28±0.03	0.66±0.03	0.55±0.02	0.70±0.03	0.58±0.04
$\phi_{cls_{pred}}$	**0.97±0.01**	0.39±0.02	**0.49±0.02**	**0.75±0.06**	**0.32±0.03**	**0.69±0.01**	**0.59±0.01**	**0.75±0.01**	**0.65±0.02**

Table A5. Performance of models after concatenating per-class accuracy information to the input vector for the tracing model (Sec. A.6). We only compare model variants that appear in the main text. We observed a boost for all models.

	Greebles				**Eyes**				**Butterflies**			
	Train		Test		Train		Test		Train		Test	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
ϕ_{static}†	0.63	0.52	0.67	0.58	0.60	0.59	0.67	0.68	0.55	0.53	0.64	0.61
$\phi_{static+perClAcc}$	0.65	0.54	0.69	0.58	0.63	0.62	0.69	0.70	**0.59**	**0.57**	**0.66**	**0.65**
ϕ_{direct}†	0.70	0.59	0.77	0.64	0.66	0.65	0.75	0.74	0.55	0.53	0.60	0.57
$\phi_{direct+perClAcc}$	**0.71**	**0.62**	0.78	**0.67**	**0.69**	**0.69**	0.78	0.78	0.53	0.51	0.61	0.57
ϕ_{cls_pred}†	**0.71**	0.62	0.77	0.65	0.65	0.65	0.74	0.74	0.54	0.52	0.60	0.57
$\phi_{cls\_pred+perClAcc}$	**0.71**	0.61	**0.79**	**0.67**	**0.69**	**0.69**	**0.79**	**0.79**	0.53	0.51	0.60	0.58

Table A6. We train the ϕ_{direct} tracing model on the butterflies dataset with different embedding dimensions. We find that embedding dimension has no impact on performance.

	Butterflies								
	Train							Test	
	Cabbage White	Monarch	Queen	Red Admiral	Viceroy	Micro	Macro	Micro	Macro
ϕ_{direct_dim8}	0.98±0.01	0.41±0.03	0.45±0.06	0.77±0.07	0.35±0.03	0.70±0.03	0.59±0.03	0.77±0.03	0.65±0.04
ϕ_{direct_dim16}	0.98±0.01	0.42±0.05	0.46±0.04	0.77±0.07	0.37±0.03	0.70±0.02	0.60±0.02	0.78±0.03	0.66±0.03
ϕ_{direct_dim32}	0.98±0.01	0.43±0.04	0.46±0.03	0.77±0.06	0.35±0.02	0.70±0.02	0.60±0.01	0.78±0.02	0.67±0.02
ϕ_{direct_dim64}	0.97±0.01	0.43±0.03	0.48±0.04	0.78±0.06	0.37±0.03	0.71±0.02	0.61±0.02	0.79±0.01	0.66±0.01

B CNN Architecture Details

In Table B7 we describe the architecture of the CNN used to encode images for all of the models.

Table B7. Structure of the CNN backbone used to learn the image representation. The bolded and italicized entries are variable and depend on the experiment and dataset. The number of image channels (***img_chns***) is three for the Butterflies and Greebles dataset, but is one for Eyes. The Butterflies and OCT datasets contain larger images (144 x 144), and so ***img_feats*** is set to 1296. For the Greebles dataset, the images are (128 x 128) and ***img_feats*** is set to 1204. Finally, the output of the model is the size of the embedding dimension and is set to 16 for all experiments.

CNN backbone						
layer	**in channels**	**out channels**	**k**	**s**	**p**	**activation**
conv1	***img_chns***	8	5	1	2	PReLU
maxpool1	-	-	4	-	-	PReLU
conv2	8	16	5	1	2	PReLU
maxpool2	-	-	4	-	-	-
flatten	-	-	-	-	-	-
linear	***img_feats***	512	-	-	-	PReLU
linear	512	256	-	-	-	PReLU
linear	256	256	-	-	-	PReLU
linear	256	16	-	-	-	PReLU

C Additional Results

We recreate Fig. 4 for all datasets and include results from the Direct Response and Time-Sensitive Model. For the Greebles dataset, we include the histograms of the features of the classes to demonstrate the difficulty of the task.

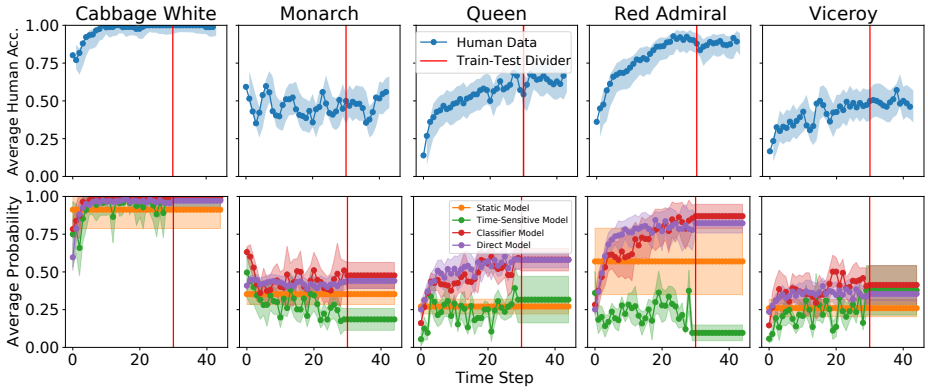

Fig. C1. (Top) The smoothed average human learner accuracy for each class over time on the Butterflies dataset. The shadowed regions indicate confidence intervals as the number of samples in each time and class bin are not guaranteed to be the same. **(Bottom)** The average probability of getting a class correct predicted by the static ϕ_{static} model (orange), ϕ_{static_time} model (green), the direct response ϕ_{direct} model (purple), and the classifier prediction ϕ_{cls_pred} model (red). At each time-step, for each learner in the test set, the models predict class probabilities for ~ 50 images per class. The probabilities are averaged (solid line) and the shadows indicate one standard deviation. While both recurrent models have similar traces, the ϕ_{direct} produces smoother average probabilities.

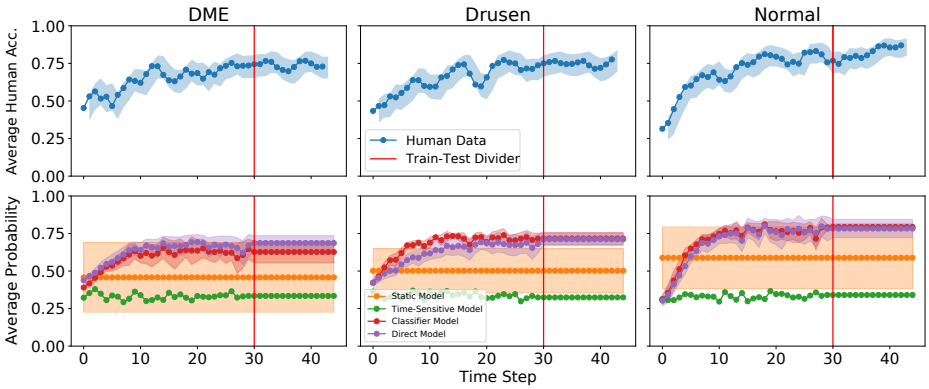

Fig. C2. Human and model performance on the Eyes dataset. See Fig. C1 for a detailed caption.

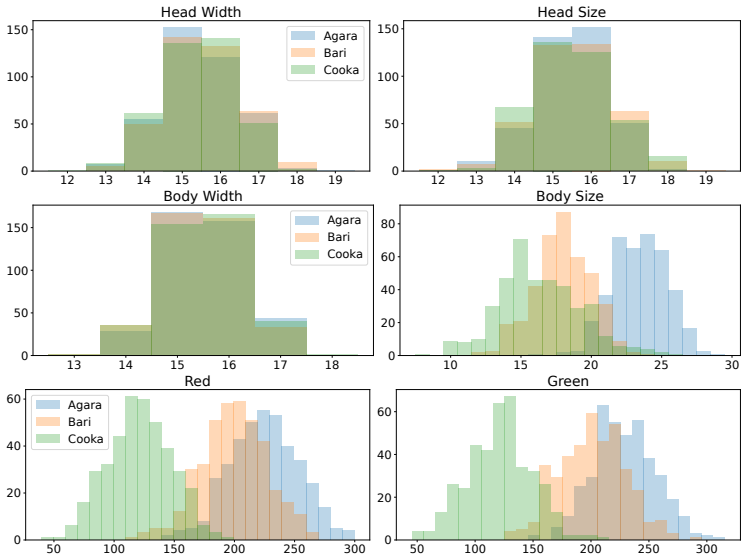

Fig. C3. The Greebles dataset was inspired by the one used in [7]. In our version, the three classes vary in Head Width and Size (top row), Body Width and Size (middle row), and the Red and Green channel for the RGB color (bottom row). The histograms overlap completely for Head Width, Head Size, and Body Width. These variations serve as distractors since they provide no information about which class an image belongs to. The other features, Body Size, the Red channel, and the Green channel have different distributions and can be used to estimate the class. Agara and Bari are most separable by Body Size, Cooka is most separable from both Agara and Bari in the two color channels. However, note that they are not perfectly separated and it is possible, although less likely, for two images from different classes to take on the same properties. This makes the Greebles dataset particularly challenging, since the important features are both subtle and imperfect for distinguishing between classes.

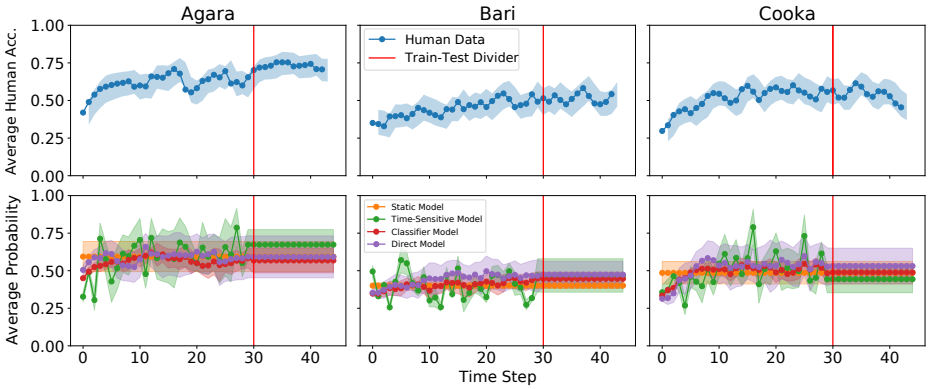

Fig. C4. Human and model performance on the Greebles dataset. See Fig. C1 for a detailed caption.

D Learned Representations

Here we explore the representations learned by the the classifier prediction model (ϕ_{cls_pred}) on the Butterflies dataset. In Fig. D5 we visualize the internal state of the model and in Fig. D6 we provide an in depth comparison for two different learners.

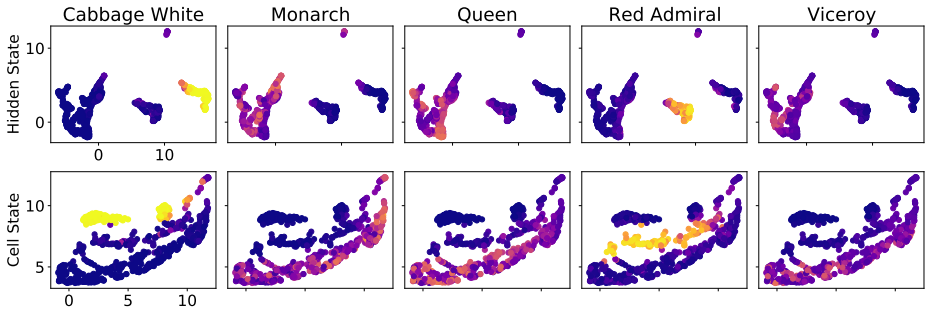

Fig. D5. The hidden states and cell states of the LSTM for ϕ_{cls_pred} while tracing 25 test-set learners are plotted in 2D using the UMAP dimensionality reduction algorithm [3]. **(Top)** The hidden state representations are colored according to the probability (purple to yellow) that a response produced with that hidden state would correctly classify an image of the class in the panel title. We can see that the classes that correspond to the best average performance by humans are in well-defined clusters (e.g. Cabbage White), whereas the classes that are commonly mistaken for each other are grouped together and have much weaker probabilities of being correct. **(Bottom)** The cell states are visualized in the same manner. For the cell states, we can see that the clusters seem to be dragged across a single dimension. The Cabbage White and Red Admiral cluster is split in two pieces in the cell state, which we explore in Fig. D6.

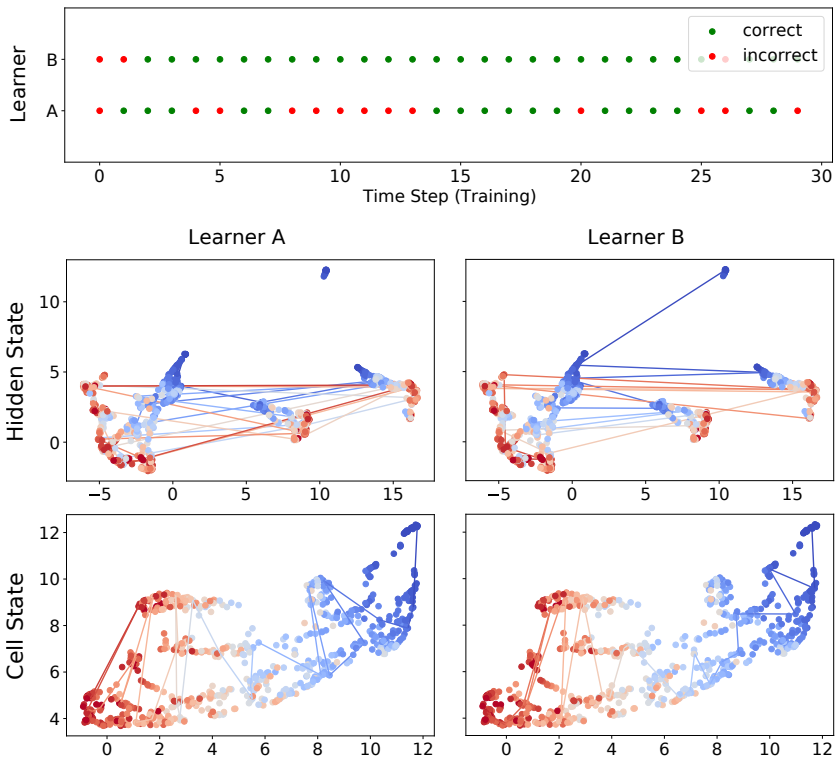

Fig. D6. (Top) The sequence of correct and incorrect responses made by two human learners during training. We selected these two learners as they demonstrate different learning behaviors. It seems Learner B may already be familiar with butterflies. **(Bottom)** We overlay each learners' trajectory through the hidden and cell states. The colors represent the time-step, where dark blue is the beginning of training, light-grey is the middle, and dark red is the end. We see that Learner B's trajectory quickly skips to the left of the cell state, suggesting the LSTM encodes the learners skill level on all classes in certain dimensions of the cell state and uses the hidden state to translate the skill level into an appropriate response for the image shown to the learner.

E Feature Space

In Fig. E7 we visualize the feature space learned by the CNN for the classifier prediction model (ϕ_{cls_pred}) on the Butterflies dataset.

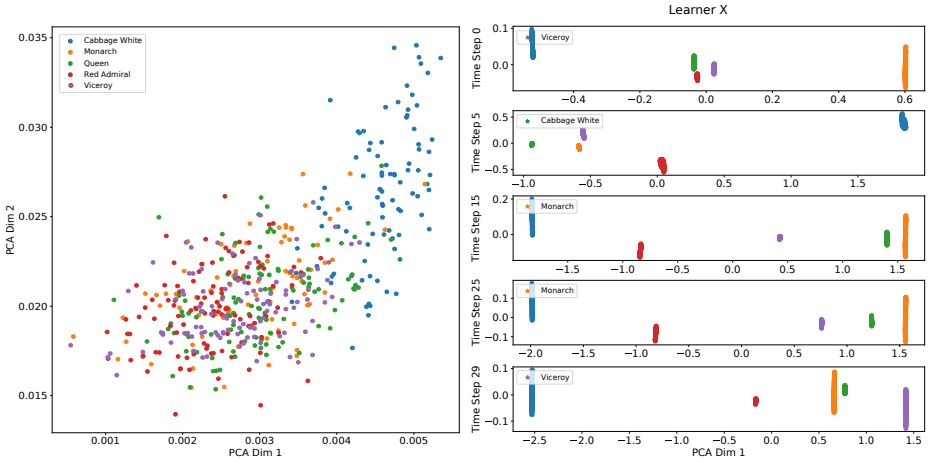

Fig. E7. The feature space learned by the CNN must support several types of behaviors, since behavior changes between learners and over time. We use PCA to reduce the learned feature space into two dimensions. **(Left)** We show a subset of images in the Butterflies dataset colored by the ground truth label. Aside from the Cabbage White class, which is the easiest to identify, the representations are difficult to separate. **(Right)** We use the hyperplanes predicted while tracing a single learner X to induce a subspace and visualize the features in that subspace. Within the subspaces, the classes are much better separated. Each row shows a subspace induced by a hyperplane for different time-steps - where the time-step is indicated on the left. The colors represent the class and the labelled color is the target class for the image being evaluated in that time-step. We see that, over time, the target class is pushed further to the right and is better separated from the other classes (see orange cluster in time-step 15 vs. 25). Classes that are confused for each other have less separation, whereas classes like Cabbage White, that are rarely confused, are extremely well-separated from the other classes. Also, note that the subspace orientation (target class moved to the right) matches how the dot product between the hyperplanes and features is translated into probabilities in the model.

F Additional Implementation Details

F.1 Types of Learner Responses

There are several ways to request information from the learner: they can provide their best guess, a ranked list of guesses, or confidence scores for each class. In these datasets, we ask for a ranking of each learner's top 3 classes as a balance between time-spent and informativeness. While our models are trained on their top choice (equivalent to their best guess), we hypothesize that the extra information available in the ranked responses can be leveraged to improve response prediction performance. We leave this to future work.

F.2 Recurrent Neural Networks

Here we elaborate on the details of the recurrent neural network based models.

Direct Response Model. At each time-step, this model receives the hidden states, cell states, the learner's response to the previous interaction, the embedding of the current image, and the true label of the current image. The model predicts the response of the learner with respect to the input image.

Classifier Prediction Model. At each time-step, this model receives the hidden state, cell state, the embedding of the image from the previous interaction, the learner's response to the previous interaction, and the true label of the current image. The model predicts a classifier that is used to classify the embedding of the input image such that it matches the response of the learner to that input image at that time-step.

Ground truth labels and learner responses are represented as one-hot-encoded vectors. For both models, at the first time-step, some of the values that make up the input vector are not available to the model. For example, the hidden state, cell state, the response to previous interaction, etc. These vectors are initialized to zeros.

References

1. Choi, Y., Lee, Y., Cho, J., Baek, J., Kim, B., Cha, Y., Shin, D., Bae, C., Heo, J.: Towards an appropriate query, key, and value computation for knowledge tracing. In: Proceedings of the Seventh ACM Conference on Learning@ Scale (2020)
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
3. McInnes, L., Healy, J., Saul, N., Grossberger, L.: Umap: Uniform manifold approximation and projection. The Journal of Open Source Software (2018)
4. Nosofsky, R.M., Meagher, B., Kumar, P.: Contrasting exemplar and prototype models in a natural-science category domain. In: CogSci (2020)
5. Piech, C., Spencer, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., Sohl-Dickstein, J.: Deep knowledge tracing. NeurIPS (2015)
6. Shin, D., Shim, Y., Yu, H., Lee, S., Kim, B., Choi, Y.: Saint+: Integrating temporal features for ednet correctness prediction. In: LAK21: 11th International Learning Analytics and Knowledge Conference (2021)
7. Welinder, P., Branson, S., Perona, P., Belongie, S.: The multidimensional wisdom of crowds. NeurIPS (2010)