A Neurobiological Evaluation Metric for Neural Network Model Search: Supplemental Material

Nathaniel Blanchard
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
nblancha@nd.edu

Jeffery Kinnison
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
jkinniso@nd.edu

Brandon RichardWebster
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
brichar1@nd.edu

Pouya Bashivan
McGovern Institute for Brain Research and
Dept. of Brain and Cognitive Sciences, MIT
Cambridge, MA 02142
bashivan@mit.edu

Walter J. Scheirer
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
walter.scheirer@nd.edu

1. Datasets

In this section we provide additional details about the datasets that we used for our experiments, as well as information about the pre-processing of each dataset. In the cases where datasets were composed of images, rather that video, we presented the stimuli to PredNet formatted as though it were a static video in order to match the network's input layer. The activation patterns from PredNet for this style of stimulus presentation mimic biological neural responses and perception [4].

1.1. Stimuli from fMRI experiments

We used fMRI recordings of four participants' inferior temporal (IT) and early visual cortex recorded by Kriegeskorte *et al*. [3], who abstracted each recording into an RDM. Full experimental details can be found in [3]. The stimuli used for experiments are presented in Fig. 1. Kriegeskorte *et al*. [3] provide valid fMRI recordings for 92 out of 96 of these images. The stimuli were selected to represent neural behavior through activation similarity, and are thus composed of three tiers of similar and dissimilar objects. We use these same stimuli in order to obtain network activations from each PredNet network.

1.2. KITTI

The KITTI dataset is a collection of video recordings taken by a car driving through various locales: city, residential, and road [2]. We follow the methods described by Lotter *et al*. [5], who designed the predictive coding network we deploy, to extract 10 frame sequences from KITTI videos. We utilize this data for unsupervised training and next frame prediction evaluation performance. We split the data into training, validation, and testing sets that remain constant over all experiments. In Fig. 2 we show an example of the KITTI dataset and a random network's next frame prediction for that series.

1.3. VLOG

The VLOG dataset is a recent action-annotation dataset [1]. This dataset was selected to allow for a more fine-grained measure of prediction. The task the dataset was designed for is the prediction of human gestures and actions, at times from an

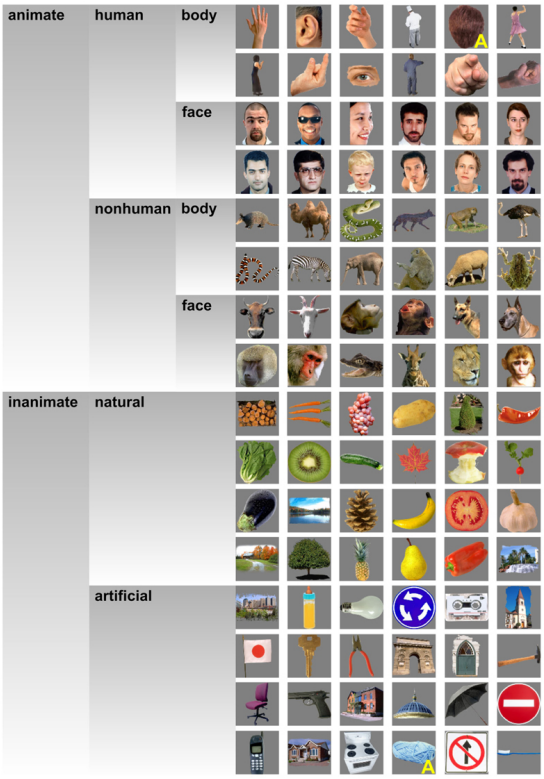

Figure 1. Image from Mur *et al*. [6] showing the stimuli that we used to compare humans and networks. Stimuli were shown one at a time.

Figure 2. Qualitative evaluation of next frame prediction using KITTI for a random PredNet network [2]. Image created with code from Lotter *et al*. [5].

unstable camera vantage-point. When used for the next frame prediction, such conditions in the dataset increase the difficulty of that task. We pre-process VLOG data in a manner that is similar to the methodology associated with the KITTI dataset. We downsample the VLOG videos to 10 frames per second in order to match the KITTI frames per second and reshape the videos to match KITTI pre-processing sizes set by Lotter *et al*. [5]. The images are organized into separate training, testing, and validation sets, which are further split by subject. This data is utilized in experiments to provide an additional test of network training variability and as a cross-dataset validation of a network's next frame prediction performance. Details for those experiments are provided below in Sec. 3.

1.4. Gazoobian objects

These "alien" objects were originally proposed by Tenenbaum *et al*. [7] in order to assess novel object hierarchy. We created an original dataset of hierarchical objects in order to assess the object matching capabilities of predictive coding networks. Objects are randomly generated via procedural graphics and posed at random angles. Two third of the objects were presented with both random lighting position and intensity. One third of the objects were randomly colored. While Gazoobian objects exist within broader hierarchical categories, for all experiments we ignore the hierarchy and treat each object as a unique class. This is done to enhance the difficulty of the object matching task to ensure that we do not hit an

Figure 3. Qualitative evaluation of next frame prediction using VLOG for a random PredNet network [1]. The VLOGer in this clip shows the camera a container of makeup, indicative of the fine-grained scenes annotated within the VLOG dataset. Image created with code from Lotter *et al*. [5].

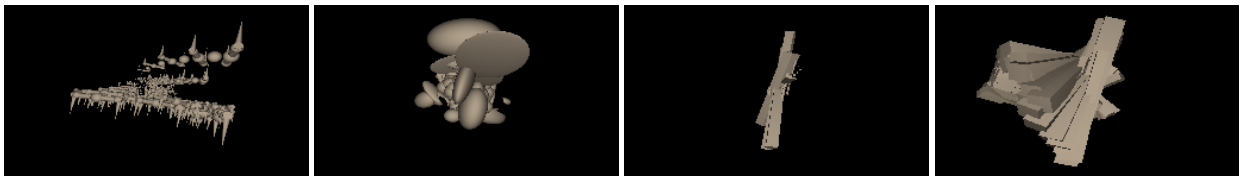

Figure 4. Randomly generated Gazoobian objects. These objects are downsampled to match the KITTI input and are used to test a network's object matching accuracy. Gazoobian objects are hierarchical by design and thus an object matching task can be made difficult by using them. The two stimuli on the right are examples from the same class of objects, but they are not matching instances of the same exact object at different view points.

accuracy ceiling. The major benefit of Gazoobian stimuli for object recognition is that the other-worldliness of the objects allows a network to be tested on unseen objects regardless of training data. We down-sample the objects to match KITTI. The objects in Fig. 4 are all examples of Gazoobian objects. We consider the two stimuli on the right to be difficult to distinguish because they are a part of the same hierarchical class, but are not the exact same object at different view points.

2. Details of hyperparameter ranges for network sampling

In Sec. 4.1 of the main paper, we describe experiments that evaluate our proposed HMS metric over a range of networks. In this section we describe the full experimental details related to training and evaluating networks across those experiments. Although we focused the majority of the analysis on the set of 95 networks, introduced in Sec. 4.1, we also trained a larger set of 1811 networks, which we briefly utilized in Sec. 4.1 to study the effects of the learning rate hyperparameter. The key difference between the two samples is the range and type of hyperparameters the models were sampled from. The hyperparameters from the 95 network sample largely consist of *training hyperparameters*, such as batch size, while the set of 1811 networks is largely composed of *architectural hyperparameters*, such as the number of layers. These samples, and their overlap, are illustrated in Fig. 5. The overlap results when changing a network's filter size while simultaneously varying a small range of training hyperparameters to avoid over-fitting our search. Within each set of hyperparameters we employ Monte Carlo sampling to obtain a sample of the search space.

2.1. Hyperparameters of the 95 network sample (training hyperparameters)

Training hyperparameters include the batch size, the number video sequences used for training, the number of video sequences used for validation loss, the number of training epochs, and the learning rate. In Table 1 we list the hyperparameter ranges we sampled from.

2.2. Hyperparameters of the 1811 network sample (architectural hyperparameters)

We also performed a series of experiments to assess how architectural hyperparameter variations affect PredNet network performance. Based on the experiments conducted in Sec. 4 of the main paper we made two network training assumptions in order to minimize training time and limit the influence of non-architectural hyperparameters. First, we limited the number of epochs used for training to vary between 20 and 60 because of the stability of HMS with minimal training, as discussed in Sec. 4.2 of the main paper. Second, we minimized the search space of all training hyperparameters except for learning rate. We kept the search space for learning rate the same due to its strong correlation with network performance, as discussed in Sec. 4.1 of the main paper. No training hyperparameters, defined in Table 1, were fixed at a specific value to avoid over-fitting

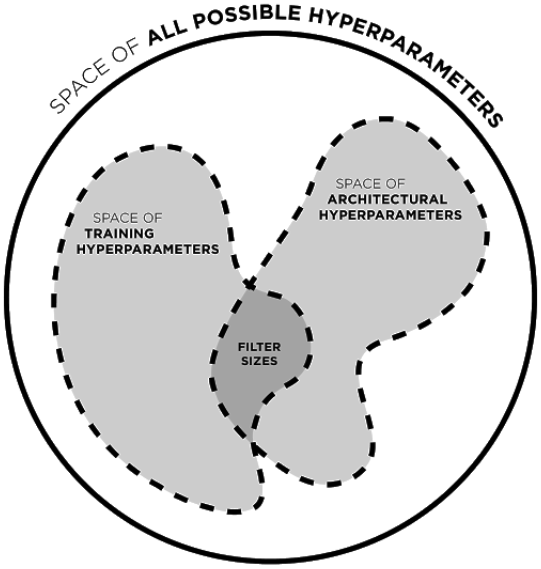

Figure 5. A conceptual overview of each of our model search types, architectural and training based, as well as the overlap between the two types. The overlap results from changing the network's filter size when varying architectural hyperparameters while also varying a small range of training hyperparameters to avoid over-fitting in the search process. We use a Monte Carlo method to sample networks from within each space in order to study how networks, and the metrics we evaluate those networks with, change within that space.

Table 1. The range of training hyperparameters varied in our experiments. We include one non-training hyperparameter, filter size, to avoid over-fitting our experiments to one architecture. All remaining architectural features were set to the PredNet default (48 filters, four layers).

Hyperparameter	**Range**
Batch size	[2, 14]
Number of video sequences for training	[100, 2000]
Number of video sequences for validation	[10, 500]
Number of epochs	[10, 500]
Learning Rate	[0.1, 0.00001]
Filter size	$[1 \times 1, 6 \times 6]$

to one training regime. We used a range of epochs in order to: balance our need for short training times, obtain a large sample, and to allow next frame prediction and accuracy more time to converge. This allowed us to confirm the correlations found in Sec. 4.1 of the paper.

Different architectural hyperparameters were used to assess the stability of HMS across architectures. Table 2 lists the full set of hyperparameters and their ranges. Architectural hyperparameters included filter sizes for each of the PredNet layers, the number of PredNet layers, and the number of filters per layer. The number of hyperparameters in each network varied between 6 and 20 depending on the number of layers in the network. Thus, we also exponentially increased the number of networks we trained to assess these parameters in order to maximize architectural variability. In all, we trained 1811 networks. We found the evaluation metrics for this network sample were within the range of the metrics from the training hyperparameters sample, listed in Table 3.

For our 1811 network sample, we note that next frame prediction and object matching were both, on average, not as good as the metrics reported the training hyperparameter sample (Table 1 in the main paper), in spite of an increase in HMS. We suspect that this occurs because object matching and next frame prediction performance have not converged, since they have not been trained for enough epochs, as discussed in Sec. 4.2 of the main paper. As further evidence, for this sample, the correlation between object matching and next frame prediction is -0.316 ($p < 0.001$), a substantial decrease in correlation strength from -0.791 ($p < 0.001$) in the sample of 95 networks.

The lower standard deviation of HMS in Table 3 compared to the 95 model sample from the main paper indicates the 1811 model sample of HMS is likely a more accurate representation. To search for further evidence for this hypothesis, we examined the average HMS of 3-layer networks from this sample, which converge more quickly than larger models and should thus be the most accurate representation from this sample, and find the mean to be 0.117 ($\text{SD} = 0.047; N = 525$). This gives us confidence that the average HMS for the full space of possible models is likely around 0.12 ($\text{SD} = 0.047; N = 525$).

As an aside on the reliability of Monte Carlo sampling, we note that the threshold we employ in Sec. 4.3 of the main text (mean plus one standard deviation) is 0.161, a minuscule 0.003 difference from the threshold of 0.164 we would have used if we had used this much larger sample to set the threshold.

Lastly, we use this sample of models to confirm that learning rate is not overly influential of the link between HMS and the other evaluations (see Sec. 4.1 of the main paper). We find HMS is correlated ($p < 0.001$) with next frame prediction error (-0.211) and object matching accuracy (0.245) when performing a partial correlation to control for the effects of learning rate. These correlations are weaker than presented in Table 4, confirming that learning rate is an influential hyperparameter in the training process. But it is not the sole variable responsible for the success of HMS.

This sample of models confirms HMS to still be predictive of performance across architectural variations, and provides further evidence that HMS converges quickly in training.

Table 2. The range of architectural hyperparameters evaluated in our experiment. For information on the PredNet architecture, see Lotter *et al*. [5]. To avoid over-fitting our experiments to a particular training pattern, we vary training hyperparameters but scale some of their ranges back, compared to those in Table 1. In the training hyperparameter search, we varied all filters within PredNet in parallel. However, in this search we varied filters individually. There is one representational target: and prediction filter size per layer.

Hyperparameter	**Range**
Representation modules filter size	$[1 \times 1, 9 \times 9]$
Target modules filter size	$[1 \times 1, 9 \times 9]$
Prediction modules	$[1 \times 1, 9 \times 9]$
Number of layers	[3, 6]
Filters in first stack	[2, 56]
Batch size	[2, 14]
Number of video sequences for training	[500, 600]
Number of video sequences for validation	[10, 200]
Number of epochs	[20, 60]
Learning Rate	[0.1, 0.00001]

3. Alternative next frame prediction dataset: VLOG

We experimented with the VLOG dataset to further test the generalization of trained networks — using it to inform us about the the cross-dataset generalization of predictive coding networks in the next frame prediction task. We also considered how training on other datasets may affect performance. We trained 88 models using VLOG and 112 models with KITTI, and evaluated the models on next frame prediction error, object matching accuracy, and HMS on both datasets. We present the numbers for VLOG as the training dataset in Table 5 and KITTI as the training dataset in Table 6, obtaining similar results to those from the 95 model sample (Table 1 in the main paper). We found object matching accuracy was consistent whether trained on KITTI or VLOG. The cross-dataset performance appears unaffected by training data – Spearman's rho for evaluating next frame prediction performance of VLOG and KITTI was almost 1, indicating nearly perfect correlation, regardless of whether the network was trained on KITTI ($0.965, p < 0.001$) or VLOG ($0.967, p < 0.001$). These results indicate that PredNet is largely unaffected by the specific training data used, at least with respect to video recordings.

The size of the VLOG dataset is very large, thus we were unable to load the entire dataset into memory and instead had to load each batch as needed. This was computationally intensive. With the computational overhead of VLOG and the finding that PredNet was largely dataset invariant, we chose not include VLOG in the experimental analysis found in Secs. 4.2 and 4.3 of the main paper.

We downsampled the VLOG dataset to match KITTI pre-processing described by Lotter *et al*. [5]. One direction for future work with datasets is testing performance on high resolution videos, as our samples indicate we are nearing the ceiling on next-frame prediction performance with regard to video quality.

Table 3. Summary statistics of evaluation scores and standard deviations for a sample of 1811 randomly selected architectural hyperparameters from PredNet. These scores are similar to the scores reported for the training parameters in the main paper (Table 1), although the networks from this sample are under-trained on the next frame prediction and object matching evaluations. The lower standard deviation (SD) for HMS indicates that this average is more representative of average of HMS scores.

Evaluation Task	**Metric**	**Mean**	**SD**
Next Frame Prediction	MSE	0.0630	0.1149
Object Matching	Accuracy	0.3363	0.1493
Human-Model Similarity	Correlation	0.1201	0.0412

Table 4. Spearman's Rho of evaluation metrics for 1811 trained PredNet networks with randomly selected architectural hyperparameters. Correlations were lower than those reported for varying the training hyperparameters in Table 2 from the main paper because the models are trained for fewer epochs, meaning accuracy and next frame prediction have less time to converge.

Variable	**Accuracy**	**Human-Model Similarity**
Next Frame Prediction Error	-0.316**	**-0.612****
Object Matching Accuracy	.	**0.323****

$^{**}p < 0.001$

Table 5. Summary statistics for metrics when 88 models are trained on the VLOG dataset across random architectures.

Evaluation Task	**Mean (SD)**
Next Frame Prediction (VLOG)	0.093 (0.115)
Next Frame Prediction (KITTI)	0.102 (0.115)
Object Matching	0.373 (0.141)
Human-Model Similarity	0.129 (0.042)

Table 6. Summary statistics for metrics when 112 models are trained on the KITTI dataset across random architectures.

Evaluation Task	**Mean (SD)**
Next Frame Prediction (VLOG)	0.088 (0.119)
Next Frame Prediction (KITTI)	0.097 (0.131)
Object Matching	0.363 (0.123)
Human-Model Similarity	0.130 (0.049)

4. Human-model similarity on a per-layer basis

We compared measurements for HMS at the layer level instead of the model level across 1302 models with varying architectural hyperparameters. The results are shown in Table 7. We found that, on average, HMS tends to be highest when concatenating all layers of the predictive coding network and comparing them simultaneously. We also found that earlier layers are more indicative of HMS performance than later layers. Previously, Yamins *et al*. [8] found that different layers of CNNs correspond to different ventral cortical regions (V4 and IT). However, that work focused on more targeted recordings on neural activity via electrophysiology. Since the fMRI data we use reflects measurements across several visual areas, and not just early or late regions, it follows that the full model concatenating all layers would correlate best.

Given these results, we investigated the use of specific layers when calculating HMS, and their effect on its prediction capabilities for the two computer vision tasks. Over the set of all models, both accuracy and MSE are more strongly correlated with the last layer than the first layer. However, when considering just deeper 6-layer models, the correlations patterns are different, and weaker, implying the models have been undertrained.

5. Human-model similarity stability

In this section we supplement the discussion from Sec. 4.2 in the main paper with further experiments and analysis exploring the stability of the behavior of HMS in relation to predictive coding networks.

Depth of network. Note that our original experiment was conducted on a 4-layer network, but we also trained the set of 1811 networks with three to six layers. We investigated whether an increased number of layers would further increase

Table 7. A comparison of the HMS metric when calculated using a full model (the original evaluation source, labeled "concatenated" below), an alternative full-view model (averaging the calculated similarity scores for each individual layer), and the first and last layers alone. We compute averages for a set of 1302 models. SD is standard deviation.

PredNet Reference for RDM Construction	Mean HMS (SD)
All Layers (Concatenated)	**0.130 (0.034)**
All Layers (Averaged)	0.103 (0.033)
First Layer	0.114 (0.037)
Last Layer	0.090 (0.048)

required training time for HMS to stabilize. We conducted a small ten-model experiment using a 6-layer PredNet network trained for various numbers of epochs. We determined that HMS is more inconsistent when a larger network is trained for fewer epochs. In this case, we find that as many as 50 epochs may be needed to accurately report HMS. Comparatively, we needed only 10 in the across-network stability experiment. We only trained the network for as many as 150 epochs, but we found that the object matching standard deviation was 0.162 and that the next frame prediction mean was still decreasing at 150 epochs of training. This indicates, despite the increased time to stability over the 4-layer network from the across-model experiment from Section 4.2, that HMS is still stabilizing much more quickly than the other metrics. Finally, we found that the similarity score can be monitored at every epoch, which can be used to detect when HMS has stability.

Within-network stability. We considered how HMS score at the point of stability compared with the HMS score after 150 epochs. We identify the point of stability as the time when the standard deviation of the last 25 epochs dropped to 0.01. For seven of the models, HMS remained within 0.01 of its point of stability score, indicating the stability of the metric was consistent after it converged. The other three networks were cases where HMS dropped between 0.05 to 0.06 points. Further examination of these instances indicated that although MSE had continued to decrease, object matching accuracy had failed to improve, and was matching at or below the performance of random weights. These findings strongly indicate that these networks were being overfit to the next frame prediction task and were failing at the cross-domain object matching task. HMS had decreased as the network's internal behavior became less and less human-like. Thus HMS can be used as an indicator of network generalization capability during individual training instances.

Across-network stability. We also investigated how quickly all scores stabilized within a network architecture. The goal was to find out how much training is required for a network architecture to achieve a consistent result. We investigated this by training multiple sets of networks with the same architecture and epochs, and we varied the number of epochs over the sets. This analysis allowed us to identify how the behavior of metrics varied across a series of randomly instantiated networks. Ideally, such randomness should not significantly impact the reliability of a good metric. We discovered that for networks that had not converged yet, the metrics reflected the lack of convergence through their behavior. Early in training, predictable non-stable behavior occurred for each metric. We define non-stable behavior as behavior that occurs early in training but eventually disappears, given enough epochs. When the networks were trained for fewer than 100 epochs, object matching accuracy was inconsistent across networks, indicating randomness in training was affecting the results. Next frame prediction error continuously decreased, but after the network converged at 100 epochs, the error plateaued. Impressively, HMS was consistent across all the networks in our experiment, even in networks trained for as few as 10 epochs. The reliability and predictiveness of HMS make it a strong early indicator of network performance for a given hyperparameter set. In this case, one would only need to train for 10 epochs to stabilize, compared with 100 epochs for the other metrics.

We further confirmed this estimate with the sample of 1811 networks trained for 20 - 60 epochs, which was discussed above in Sec. 2.2. For this sample, we would expect that accuracy and next frame prediction would not be correlated, since neither has converged. We find that this is the case. HMS, however, was moderately correlated with both metrics ($p < 0.001$), but weaker than what was reported in Table 2 of the main paper. This occurs because we are only training the networks for 20 - 60 epochs and accuracy and next frame prediction are not trained enough to be consistent.

6. A mechanism for early stopping

This section provides additional detail and analysis for the experiments described in Sec. 4.3 of the main paper. An outcome of the findings in Sec. 4.2 of the main paper and Sec. 5 above is that our proposed HMS metric can be employed during network training as a way to discard (*i.e.*, stop training) networks that will ultimately perform below the optimal network or networks. To demonstrate this, we performed two *post hoc* analyses of the 95 networks from Sec. 4.1 of the main paper. First, we analyzed the discarded networks for false rejections. Second, we quantified the amount of time saved if certain networks had been discarded during training. We considered training "stopped" if the HMS score for a network

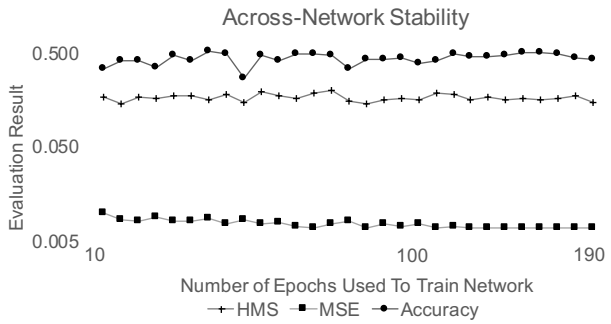

Figure 6. Mean across-network stability for 66 models trained with a fixed hyperparameter configuration, but for different numbers of epochs. Before 100 epochs, object matching accuracy is unstable (SD $= 0.07$) and next frame prediction error (MSE) drops continuously. After 100 epochs, object matching accuracy is largely stable (SD $= 0.03$) and MSE largely plateaus, indicating convergence. HMS is the most stable metric ($<$ 100 epochs SD $= 0.02$; $>$ 100 epochs SD $= 0.01$) and is stable early in training. HMS is thus a stable indication of the potential of a particular combination of hyperparameters across models. All metrics are not strongly impacted by random initialization during training.

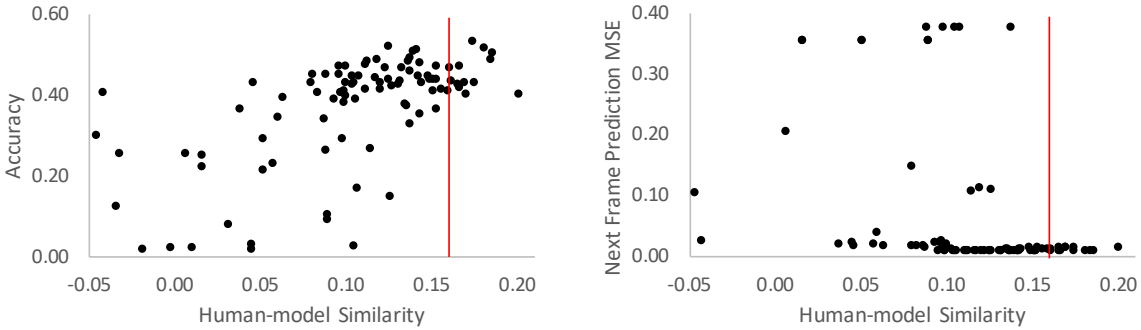

Figure 7. Scatter plots of HMS for accuracy (left) and mean-squared error (right). The red line is the threshold.

snapshot at a specific epoch was below a threshold of 0.161 (the mean HMS from Table 1 in the main paper plus one standard deviation). Only 13 of the 95 networks (13.7%) were above this threshold. These results, along with results for next frame prediction after thresholding, are shown in Fig. 6.

Our analysis shows that even with a high threshold for discarding networks, and the loss of some networks with high performance, most retained networks are high performing and the network with the highest performance on both computer vision tasks is retained. Additionally, retained networks were more likely to be better at both tasks than high performing networks that were discarded. Overall, employing early stopping would reduce the training time by 52% at no cost to final performance when selecting networks.

Performance cost of early stopping. The possibility remains that there is some performance cost to early stopping. To look into this question in more detail, we considered high performance networks (HPMs) to have one or both cross-domain evaluation metrics (accuracy and MSE) in the top 10% of each metric. We identified 15 HPMs from the set of 95 networks described in Sec. 4.1 of the main paper. With early stopping, 10 are discarded and 5 are retained. HPMs make up 38% of the 13 retained networks from the above experiment. One retained network had an object matching accuracy of 53%, making it the best object matching network, and a next frame prediction MSE of 0.0068, trivially only 0.0003 higher than the lowest MSE network (0.0065), which was discarded. Additionally, performance was similar between all HPMs (see Sec. 4.1 of the main paper), indicating that even if the top network was discarded, some retained network would have similar performance. Finally, 3 of 4 HPMs with both metrics in the top 10% were retained, making up 60% of the retained HPMs, indicating high performance networks that generalize well are more likely to be retained. For this set of networks an early stop comes at no performance cost. Using the same threshold on the set of 10 networks analyzed for the within-model stability experiment produces similar results, with retained networks including a high accuracy (43%) and low next frame prediction error (0.006) network, with an HMS score of 0.19, at no performance cost.

Explicit test of time saved with early stopping. The details of the performance cost from the early stopping analysis above can be used to estimate how much computation time we would have saved, had we utilized this strategy during training.

We estimated the amount of computation time saved by discarding 82 networks during training, including the time required to appropriately evaluate HMS. A 4-layer predictive coding network approximately requires three hours to train 150 epochs, the default number of epochs in the PredNet code, on the KITTI dataset with an NVIDIA GeForce GTX Titan Xp. Therefore, the baseline time to train 95 networks was 285 GPU hours. Only training 13 networks required 39 hours. However, the other 82 networks must still be trained until the similarity metric becomes stable. Stability occurs when the metric does not deviate more than ~ 0.01 across 25 epochs. We estimate on average stability occurs after 33 epochs, the average number of epochs it took for HMS to remain converged for 25 epochs (see Sec. 4.2 in the main paper). From our total per network estimate, on average, it would take 39.6 minutes to train a network for 33 epochs, equaling 54.1 hours to train the 82 discarded networks. In total, we estimate it required 93.16 hours of GPU time to both fully train the 13 networks and partially train the other 82 before discarding them. Ultimately, this uses 191.8 fewer hours of GPU time than the baseline — 67% of the time required to train all networks.

References

[1] D. F. Fouhey, W.-C. Kuo, A. A. Efros, and J. Malik. From lifestyle Vlogs to everyday interactions. In *IEEE/CVF CVPR*, 2018. 1, 3

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1, 2

[3] N. Kriegeskorte, M. Mur, D. A. Ruff, R. Kiani, J. Bodurka, H. Esteky, K. Tanaka, and P. A. Bandettini. Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey. *Neuron*, 60(6):1126–1141, Dec. 2008. 1

[4] W. Lotter, G. Kreiman, and D. Cox. A neural network trained to predict future video frames mimics critical properties of biological neuronal responses and perception. *arXiv preprint arXiv:1805.10734*, 2018. 1

[5] W. Lotter, G. Kreiman, and D. D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017. 1, 2, 3, 5

[6] M. Mur, M. Meys, J. Bodurka, R. Goebel, P. A. Bandettini, and N. Kriegeskorte. Human Object-Similarity Judgments Reflect and Transcend the Primate-IT Object Representation. *Frontiers in Psychology*, 4, 2013. 2

[7] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011. 2

[8] D. Yamins, H. Hong, C. Cadieu, and J. J. DiCarlo. Hierarchical modular optimization of convolutional networks achieves representations similar to macaque IT and human ventral stream. In *Advances in neural information processing systems*, pages 3093–3101, 2013. 6