Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning

William Lotter
Harvard University
lotter@fas.harvard.edu

Gabriel Kreiman
Harvard University

David Cox
Harvard University
davidcox@fas.harvard.edu

Abstract

While great strides have been made in using deep learning algorithms to solve supervised learning tasks, the problem of unsupervised learning — leveraging unlabeled examples to learn about the structure of a domain — remains a difficult unsolved challenge. Here, we explore prediction of future frames in a video sequence as an unsupervised learning rule for learning about the structure of the visual world. We describe a predictive neural network ("PredNet") architecture that is inspired by the concept of "predictive coding" from the neuroscience literature. These networks learn to predict future frames in a video sequence, with each layer in the network making local predictions and only forwarding deviations from those predictions to subsequent network layers. We show that these networks are able to robustly learn to predict the movement of synthetic (rendered) objects, and that in doing so, the networks learn internal representations that are useful for decoding latent object parameters (e.g. pose) that support object recognition with fewer training views. We also show that these networks can scale to complex natural image streams (car-mounted camera videos), capturing key aspects of both egocentric movement and the movement of objects in the visual scene, and generalizing across video datasets. These results suggest that prediction represents a powerful framework for unsupervised learning, allowing for implicit learning of object and scene structure.

1 Introduction

The most successful current deep learning architectures for vision rely on supervised learning from large sets of labeled training images [18, 23, 44, 51]. While the performance of these networks is undoubtedly impressive, reliance on such large numbers of training examples limits the utility of deep learning in many domains where such datasets are not available. Furthermore, the need for large numbers of labeled examples stands at odds with human visual learning, where a few or one view is often all that is needed to enable robust recognition of that object across a wide range of different views, lightings and contexts. The development of a representation that facilitates such abilities, especially in an unsupervised way, is a largely unsolved problem.

In addition to the unnatural extent of supervised learning, computer vision models are typically trained using single images as training examples, whereas, in the real world, visual objects are rarely experienced as disjoint, static snapshots. Instead, the visual world is alive with movement, driven both by self-motion of the viewer and the movement of objects within the scene. Many have suggested that temporal experience with objects as they move and undergo transformations can serve as an important signal for learning about the structure of objects [1, 10, 13, 15, 26, 28, 30, 32, 33, 38, 46, 48, 55, 57, 58]. For instance, Wiskott and Sejnowski proposed "slow feature analysis" as a framework for exploiting temporal structure in video streams [58]. Their approach attempts to build feature

Code and video examples can be found at: https://coxlab.github.io/prednet/

representations that extract slowly-varying parameters, such as object identity, from parameters that produce fast changes in the image, such as movement of the object. While not rivaling supervised methods, this approach, and others that rely on temporal coherence, have achieved some measure of success, pointing to the potential of learning useful representations from video [15, 27, 30, 50, 55].

Here, we explore another potential principle for exploiting video for unsupervised learning: prediction of future image frames [16, 28, 32, 33, 34, 46, 48]. A key insight here is that in order to be able to predict how the visual world will change over time, an agent must have at least some implicit model of the object structure and the possible transformations objects can undergo. To this end, we have designed a neural network architecture, which we informally call a "PredNet," that attempts to continually predict the appearance of future video frames, using a deep, recurrent convolutional network with both bottom-up and top-down connections. Our work here builds on previous work in next-frame video prediction [26, 28, 29, 38, 48], but we take particular inspiration from the concept of "predictive coding" from the neuroscience literature [2, 5, 7, 9, 11, 21, 25, 32, 39, 40, 47, 49]. Predictive coding posits that the brain is continually making predictions of incoming sensory stimuli [11, 13, 39]. Top-down (and perhaps lateral) connections convey these predictions, which are compared against actual observations to generate an error signal. The error signal is then propagated back up the hierarchy, eventually leading to an update of predictions.

We demonstrate the effectiveness of our model for both synthetic sequences, where we have access to the underlying generative model and can investigate what the model learns, as well as natural videos. Consistent with the idea that prediction requires knowledge of object structure, we find that these networks successfully learn internal representations that are well-suited to subsequent recognition and decoding of latent object parameters (e.g. identity, view, rotation speed, etc.). We also find that our architecture can scale effectively to natural image sequences, by training the architecture using car-mounted camera videos from the KITTI dataset [12]. The network is able to successfully learn to predict both the movement of the camera and the movement of objects in the camera's view, and it shows excellent generalization performance to a different car-cam dataset.

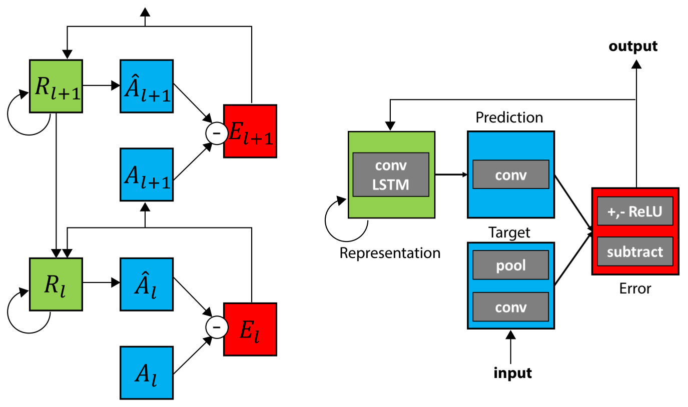

Figure 1: Predictive Coding Network (PredNet). Left: Illustration of information flow within two layers. Each layer consists of representation neurons (R_l), which output a layer-specific prediction at each time step ($\hat{A}_l$), which is compared against a target (A_l) to produce an error term (E_l), which is then propagated laterally and vertically in the network. (Right: Module operations for case of video sequences.)

2 The PredNet Model

The PredNet architecture is diagrammed in Figure 1. The network is made up a series of repeating stacked modules that attempt to make local predictions of the input to the module, which is then subtracted from the actual input and passed along to the next layer. Briefly, each module of the network consists of four basic parts: an input convolutional layer (A_l), a recurrent representation layer (R_l), a prediction layer ($\hat{A}_l$), and an error representation (E_l). The representation layer, R_l, is a recurrent convolutional network that generates a prediction, $\hat{A}_l$, of what the layer input, A_l, will be on the next frame. The network takes the difference between A_l and $\hat{A}_l$ and outputs an error representation, E_l, which is split into separate rectified positive and negative error populations. The error, E_l, is then passed forward through a convolutional layer to become the input to the next layer (A_{l+1}). The recurrent prediction layer R_l receives a copy of the error signal E_l, along with top-down input from the representation layer of the next level of the network (R_{l+1}). The organization of the network is such that on the first time step of operation, the "right" side of the network (A_l's and E_l's) is equivalent to a standard deep convolutional network. Meanwhile, the "left" side of the network (the R_l's) is equivalent to a generative deconvolutional network with local recurrence at each stage. The architecture described here is inspired by that originally proposed by [39], but is formulated in a modern deep learning framework and trained end-to-end using gradient descent, with a loss function implicitly embedded in the network as the firing rates of the error neurons.

While the architecture is general with respect to the kinds of data it models, here we focus on image sequence (video) data. Consider a sequence of images, x_t. The target for the zeroth layer is set to the the actual sequence itself, i.e. $A_0^t = x_t \; \forall t$. The targets for higher layers, A_l^t for $l > 0$, are computed by a convolution over the error units from the layer below, E_{l-1}^t, followed by rectified linear unit (ReLU) activation and max-pooling. Thus, the targets are designed to compute higher level features as the error propagates up the network [4]. For the representation neurons, we specifically use convolutional LSTM units [19, 43]. A convolutional LSTM replaces the dense matrix multiplication in a vanilla LSTM with a sparse convolutional matrix. In our setting, the R_l^t hidden state is updated according to R_l^{t-1}, E_l^{t-1}, as well as R_{l+1}^t, which is first spatially upsampled (nearest-neighbor), due to the pooling present in the feedforward path. The predictions, $\hat{A}_l^t$ are made through a convolution of the R_l^t stack followed by a ReLU non-linearity. For the zeroth (pixel) layer, $\hat{A}_l^t$ is also passed through a saturating non-linearity set at the maximum pixel value: $\text{SatLU}(x; p_{max}) := \min(p_{max}, x)$. Finally, the error response, E_l^t, is calculated from the difference between $\hat{A}_l^t$ and A_l^t and is split into ReLU-activated positive and negative prediction errors, which are concatenated along the feature dimension. As discussed in [39], although not explicit in their model, the separate error populations are analogous to the existence of on-center, off-surround and off-center, on-surround neurons early in the visual system.

The full set of update rules are listed in Equations (1) to (4). Altogether, the model is trained to minimize the (weighted) sum of the firing rates of the error units. With error units consisting of subtraction followed by ReLU activation, this is equivalent to an L_1 error. Although not explored here, other error unit implementations, potentially even probabilistic or adversarial [14, 37], could also be used.

$$A_l^t = \begin{cases} x_t & \text{if } l = 0 \\ \textsc{MaxPool}(\textsc{ReLU}(\textsc{Conv}(E_{l-1}^t))) & l > 0 \end{cases} \tag{1}$$

$$\hat{A}_l^t = \textsc{ReLU}(\textsc{Conv}(R_l^t)) \tag{2}$$

$$E_l^t = [\textsc{ReLU}(A_l^t - \hat{A}_l^t); \textsc{ReLU}(\hat{A}_l^t - A_l^t)] \tag{3}$$

$$R_l^t = \textsc{ConvLSTM}(E_l^{t-1}, R_l^{t-1}, \textsc{Upsample}(R_{l+1}^t)) \tag{4}$$

The order in which each unit in the model is updated must also be specified, and our implementation is described in Algorithm 1. Updating of states occurs through two passes: a top-down pass where the R_l^t states are computed and then a forward pass to calculate the predictions, errors, and higher level targets. A last detail of note is that R_l and E_l are initialized to zero, which, due to the convolutional nature of the network, means that the initial prediction is spatially uniform.

As a concrete example, consider input images of shape $(64, 64, 3)$, representing the height, width, and number of channels, respectively. In this case, A_0^t and $\hat{A}_0^t$ will also have a shape of $(64, 64, 3)$. Because of the splitting of positive and negative errors, E_0^t will have a shape of $(64, 64, 6)$. Using same-size convolution and max-pooling with a stride of 2, A_1^t will have spatial dimensions of 32, but can have any given number of channels. All of the convolutions in the network are zero-padded such that the spatial size is constant within a layer and across time. Thus, in this case, R_1^t would also have spatial size of 32. Although the number of channels for representational units is unconstrained, it was usually set at the same number as the other units in the layer.

Algorithm 1 Calculation of PredNet states

```
Require: $x_t$
 1: $A_0^t \leftarrow x_t$
 2: $E_l^0, R_l^0 \leftarrow 0$
 3: for $t = 1$ to $T$ do
 4:     for $l = L$ to $0$ do                                    ▷ Update $R_l^t$ states
 5:         if $l = L$ then
 6:             $R_L^t = \text{ConvLSTM}(E_L^{t-1}, R_L^{t-1})$
 7:         else
 8:             $R_l^t = \text{ConvLSTM}(E_l^{t-1}, R_l^{t-1}, \text{Upsample}(R_{l+1}^t))$
 9:     for $l = 0$ to $L$ do                                    ▷ Update $\hat{A}_l^t, A_l^t, E_l^t$ states
10:         if $l = 0$ then
11:             $\hat{A}_0^t = \text{SatLU}(\text{ReLU}(\text{Conv}(R_0^t)))$
12:         else
13:             $\hat{A}_l^t = \text{ReLU}(\text{Conv}(R_l^t))$
14:         $E_l^t = [\text{ReLU}(A_l^t - \hat{A}_l^t); \text{ReLU}(\hat{A}_l^t - A_t^l)]$
15:         if $l < L$ then
16:             $A_{l+1}^t = \text{MaxPool}(\text{Conv}(E_t^l))$
```

3 Experiments

3.1 Rendered Image Sequences

To gain an understanding of the representations learned in the proposed framework, we first trained PredNets using synthetic images, for which we have access to the underlying generative stimulus model and all latent parameters. We created sequences of rendered faces rotating with two degrees of freedom, along the "pan" (out-of-plane) and "roll" (in-plane) axes. The faces started at a random orientation and rotated at a random constant velocity for a total of 10 frames. A different face was sampled for each sequence. Images were 64x64 pixels in size and grayscale, with values normalized between 0 and 1. We used 16K sequences for training and 800 for both validation and testing.

Predictions generated by a PredNet model are shown in Figure 2. The model was trained to predict one time step ahead and, as the goal was prediction itself, the loss was taken as the sum of the firing rates of the error neurons in the zeroth (pixel) layer at time steps 2-10. A random hyperparameter search was performed over four and five layer models with the top model, the model shown, chosen with respect to performance on the validation set. This model consists of 5 layers (including the pixel layer) with 3x3 filter sizes for all convolutions and stack sizes per layer of $(1, 32, 64, 128, 256)$. Model weights were optimized using the Adam algorithm [22] with default parameters. All models were developed using Theano [52] and Keras [6].

As illustrated in Figure 2, the PredNet model is able to accumulate information over time to make accurate predictions of future frames. Since the representation neurons are initialized to zero, the prediction at the first time step is uniform. On the second time step, with no motion information yet, the prediction is a blurry reconstruction of the first time step. After further iterations, the model learns the underlying dynamics to generate predictions that closely match the incoming frame.

Quantitative evaluation of generative models is a difficult, unsolved problem [53], but we report prediction error in terms of mean-squared error (MSE) and the Structural Similarity Index Measure (SSIM) [56]. SSIM is designed to be more correlated with perceptual judgments, and ranges from -1 and 1, with a larger score indicating greater similarity. We compare the PredNet to the trivial solution of copying the last frame, as well as a control model that shares the overall architecture and training scheme of the PredNet, but that sends forward the layer-wise activations (A_l) rather than the errors (E_l). This model thus takes the form of a more traditional encoder-decoder pair, with the addition of lateral skip connections (between A_l's and R_l's). On the rotating faces, the PredNet outperformed both baselines on both measures, with a MSE of 1.53×10^{-2}, compared to 1.80×10^{-2} and 1.25×10^{-1} for the control model and last frame respectively, and an SSIM of 0.937 compared to 0.907 and 0.631. The measures were calculated as an average over all predictions after the first frame.

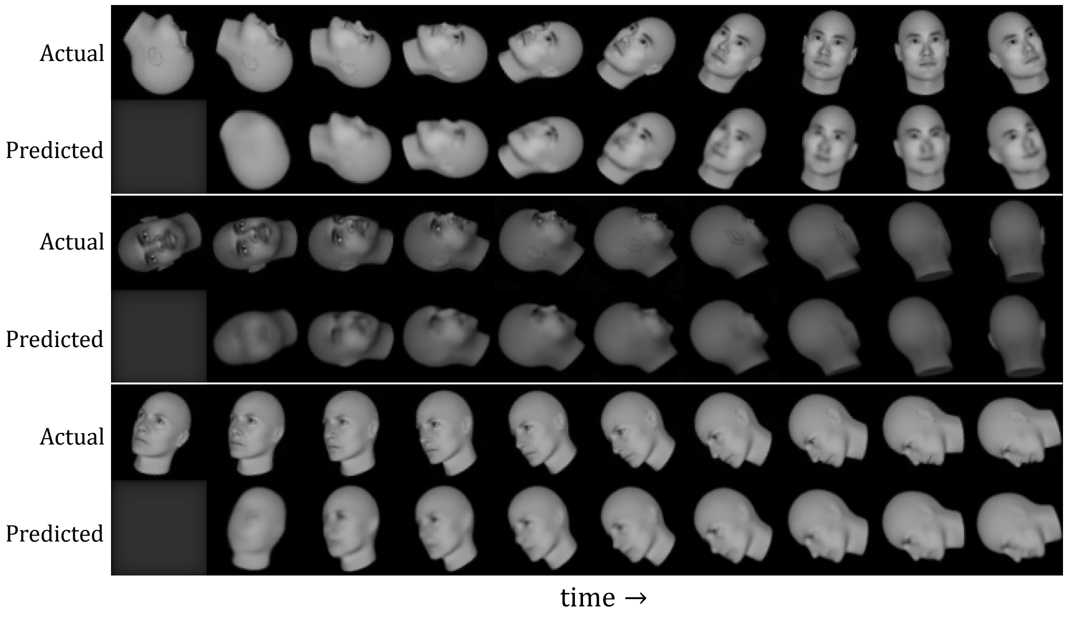

Figure 2: PredNet predictions for sequences of rendered faces rotating with two degrees of freedom. Faces shown were not seen during training.

Synthetic sequences were chosen as the initial training set in order to better understand what is learned in different layers of the model, specifically with respect to the underlying generative model [24]. The rotating faces were generated using the FaceGen software package [45], which internally generates 3D face meshes by a principal component analysis in "face space", derived from a corpus of 3D face scans. Thus, the latent parameters of the image sequences used here consist of the initial pan and roll angles, the pan and roll velocities, and the principal component (PC) values, which control the "identity" of the face. To understand the information contained in the model, we decoded the latent parameters from the representation neurons (R_l) in each layer, using a ridge regression. The R_l states were taken at the earliest possible informative time steps, which, in the our notation, are the second and third steps, respectively, for the static and dynamic parameters. The regression was trained using $4K$ sequences with 500 for validation and $1K$ for testing. For a baseline comparison of the information implicitly embedded in the network architecture, we compare to the decoding accuracies of an untrained network with random initial weights. Note that in this randomly initialized case, we still expect above-chance decoding performance, given past theoretical and empirical work with random networks [20, 36, 42].

Latent variable decoding accuracies of the pan and roll velocities, pan initial angle, and first PC are shown in the left panel of Figure 3. Generally, the accuracies increase as the layer depth increases, and the model appears to learn a representation of rotation speeds particularly well. For instance, the pan rotation speed (α_{pan}) could not be decoded from the initial weights, but the trained model had a r^2 of over 0.85 in both of the last two layers.

Beyond predicting pixels, the goal is to learn a useful representation that generalizes well to other tasks. We thus tested the model in an orthogonal task, face classification from single, static images. We created a dataset of 25 previously unseen faces at 7 pan angles, equally spaced between $[-\frac{\pi}{2}, \frac{\pi}{2}]$, and 8 roll angles, equally spaced between $[0, 2\pi)$. There were therefore $7 \cdot 8 = 56$ orientations, which were tested in a cross-validated fashion. A linear SVM to decode face identity was fit on a model's representation of a random subset of orientations and then tested on the remaining angles. For each size of the SVM training set, ranging from 1-40 orientations per face, 50 different random splits were generated, with results averaged over the splits.

For the static face classification task, we compare the PredNet to a standard autoencoder and a variant of the Ladder Network [41, 54]. Both models were constructed to have the same number of layers and stack shapes as the PredNet, as well as a similar alternating convolution/max-pooling, then upsampling/convolution scheme. As both networks are autoencoders, they were trained with a

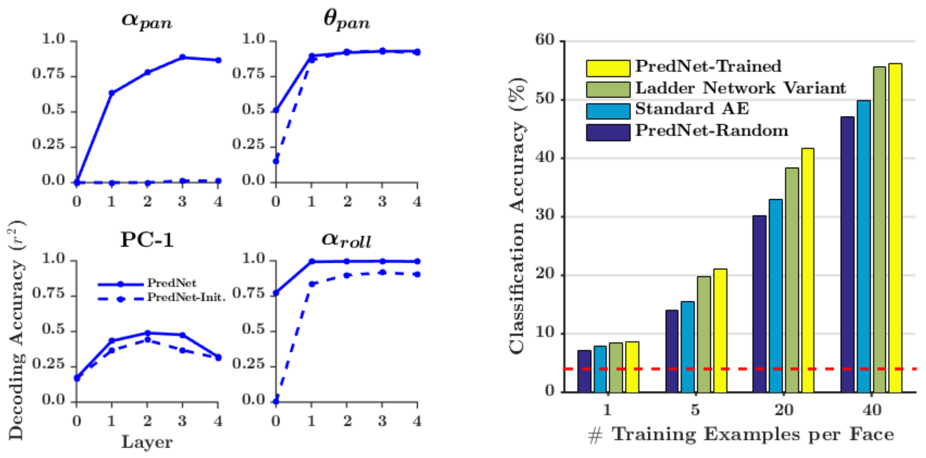

Figure 3: Information contained in PredNet representation for rotating faces sequences. Left: Decoding of latent variables using a ridge regression (α_{pan}: pan (out-of-frame) angular velocity, θ_{pan}: pan angle, PC-1: first principal component of face, α_{roll}: roll (in-frame) angular velocity). Right: Orientation-invariant classification of static faces. Dataset contained 25 static computer-generated faces at 56 orientations. A linear SVM was fit on a subset of orientations and then tested on the remaining orientations.

reconstruction loss, with a dataset consisting of all of the individual frames from the sequences used to train the PredNet. For the Ladder Network, which is a denoising autoencoder with lateral skip connections, one must also choose a noise parameter, as well as the relative weights of each layer in the total cost. We tested noise levels ranging from 0 and 0.5 in increments of 0.1, with a loss that was either evenly distributed across layers or solely concentrated at the pixel layer. Shown is the model that performed best for classification, which consisted of 0.4 noise and only pixel weighting. Lastly, as in our architecture, the Ladder Network has lateral and top-down streams that are combined by a combinator function. Inspired by [35], where a learnable MLP improved results, and to be consistent in comparing to the PredNet, we used a purely convolutional combinator. Given the distributed representation in both networks, we decoded from a concatenation of the feature representations at all layers, except the pixel layer, and trace-normalized by the number of units in each layer. For the PredNet, the representation units were used and features were extracted after processing one input frame.

Face classification accuracies using the representations learned by a PredNet, standard autoencoder, and Ladder Network are shown in the right panel of Figure 3. The PredNet compares favorably to the other models at all of the training set, suggesting it learns a representation that is relatively tolerant to object transformations. Thus, predictive training with the PredNet model can be a viable alternative to other models trained with a more traditional reconstructive or denoising loss.

3.2 Natural Image Sequences

We next sought to test the PredNet architecture on complex, real-world sequences. As a testbed, we chose car-mounted camera videos, since these videos span across a wide range of settings and are characterized by rich temporal dynamics, including both self-motion of the vehicle and the motion of other objects in the scene [1]. Models were trained using the raw videos from the KITTI dataset [12], which was captured by a roof-mounted camera on a car driving around an urban environment in Germany. Sequences of 10 frames were sampled from the "City", "Residential", and "Road" categories, with 57 recording sessions used for training and 4 used for validation. Frames were center-cropped and downsampled to 128x160 pixels. In total, the training set consisted of roughly 41K frames.

A random hyperparameter search, with model selection based on the validation set, resulted in a 4 layer model with 3x3 convolutions and layer stack sizes of $(3, 48, 96, 192)$. Models were again trained with Adam [22] using a loss only on the pixel layer. Adam parameters were initially set to their default values ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) with the learning rate, α, decreasing by a factor of 10 halfway through training. To assess that the network had indeed learned a robust representation, we tested on the Caltech Pedestrian dataset [8], which consists of videos from a dashboard-mounted camera on a vehicle driving around Los Angeles. Testing sequences were made to match the frame rate of the KITTI dataset and again cropped to 128x160 pixels. Quantitative evaluation was performed on the entire Caltech test partition, split into sequences of 10 frames.

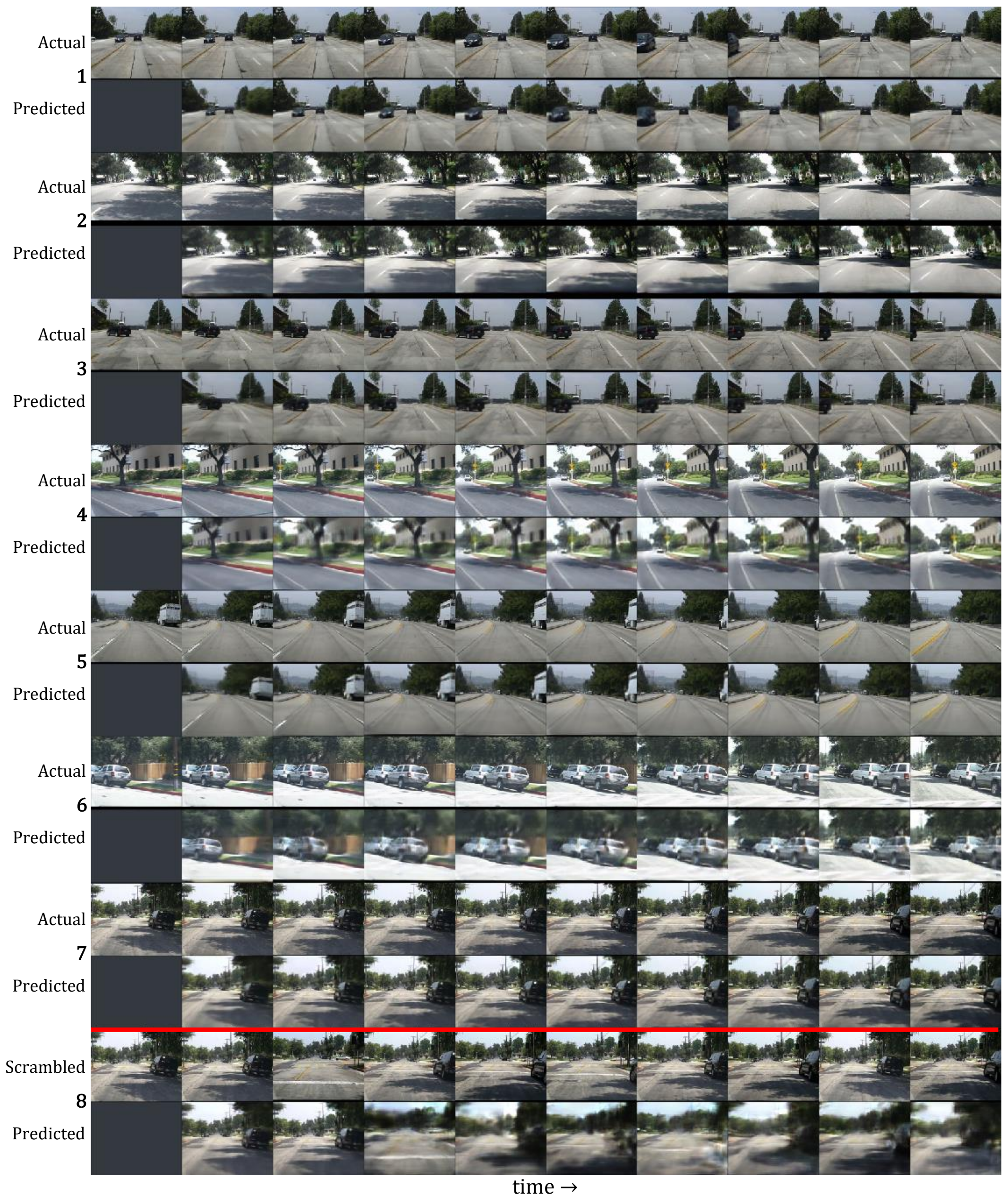

Figure 4: PredNet predictions for car-cam videos. The first rows contain ground truth and the second rows contain predictions. The sequence below the red line was temporally scrambled. The model was trained on the KITTI dataset and sequences shown are from the Caltech Pedestrian dataset.

Sample PredNet predictions for the Caltech Pedestrian dataset are shown in Figure 4, with additional enlarged views provided in the supplementary material. The model is able to make fairly accurate predictions in a wide range of scenarios. In the top sequence, a car is passing in the opposite direction, and the model, while not perfect, is able to predict its trajectory, as well as fill in the ground it leaves behind. Similarly in Sequence 3, the model is able to predict the motion of a vehicle completing a left turn. Sequences 2 and 5 illustrate that the PredNet can judge its own movement, as it predicts the appearance of shadows and a stationary vehicle as they approach. The model makes reasonable predictions even in difficult scenarios, such as when the camera-mounted vehicle is turning. In Sequence 4, the model predicts the position of a tree, as the vehicle turns onto a road. The turning sequences also further illustrate the model's ability to "fill-in", as it is able to extrapolate sky and tree textures as unseen regions come into view. For more evidence that the model's predictions are not trivial, we show a control sequence at the bottom of Fig. 4, where the input has been temporally scrambled. In this case, the model generates blurry frames, which mostly just resemble the previous frame.

Quantitatively, the PredNet again outperformed the baselines. Averaged over all predictions after the first frame, the PredNet had a MSE of 3.13×10^{-3} and SSIM of 0.884, compared to 7.95×10^{-3} and 0.762 for copying the last frame, and 3.67×10^{-3} and 0.865 for the control model. Note, the control model was trained with precisely the same structure and hyperparameters, except the layer-wise activations (A_l) are passed, as in traditional architectures, instead of the errors (E_l). To ensure that the difference in performance was not simply because of the choice of hyperparameters, we trained models with four other sets of hyperparameters, which were sampled from the initial random search over the number of layers, filter sizes, and number of filters per layer. For each of the four additional sets, the PredNet again outperformed the control, with an average error reduction of 14.7% and 14.9% for MSE and SSIM, respectively. In fact, every PredNet model tested (5 total) had a better MSE than every control model and only one control model had a better SSIM than the worst PredNet model.

While the models were trained to predict one frame ahead, they can be made to predict multiple frames by treating predictions as actual input and recursively iterating, an approach common in natural language processing tasks [17, 31]. Results from this procedure with the PredNets are shown in Figure 5.

Although the next frame predictions are reasonably accurate, the model tends to break down fairly quickly when extrapolating. This is not surprising since the predictions will unavoidably have different statistics than the natural images for which the model was trained to handle, and these deviations accumulate [3]. If we additionally train the model to process its own predictions, the model is better able to extrapolate. The third row for every sequence shows the output of the original PredNet fine-tuned for extrapolation. Starting from the trained weights, the model was trained with a loss over 15 time steps, where the actual frame was inputted for the first 10 and then the model's predictions were used as input to the network for the last 5. Despite eventual blurriness (which might be expected to some extent due to uncertainty), the model captures some key structure in its extrapolations after the tenth time step. For instance, in the first sequence, the model estimates the general shape of an upcoming shadow, despite minimal information in the last seen frame. In the second sequence, the model is able to extrapolate the motion of a car moving to the right.

4 Discussion

Above, we have demonstrated a predictive coding inspired architecture that is able to predict future frames in both synthetic and natural image sequences. Importantly, we have shown that learning to predict how an object or scene will move in a future frame confers advantages in decoding latent parameters (such as viewing angle) that give rise to an object's appearance, and can improve recognition performance. More generally, we argue that prediction can serve as a powerful unsupervised learning signal, since accurately predicting future frames requires at least an implicit model of the objects that make up the scene and how they are allowed to move. Developing a deeper understanding of the nature of the representations learned by the networks, and extending the architecture, by, for instance, allowing sampling, are important future directions.

Acknowledgments

We would like to thank Rasmus Berg Palm for fruitful discussions and early brainstorming. This work was supported by IARPA (contract D16PC00002), the National Science Foundation (NSF IIS 1409097), and the Center for Brains, Minds and Machines (CBMM, NSF STC award CCF-1231216).

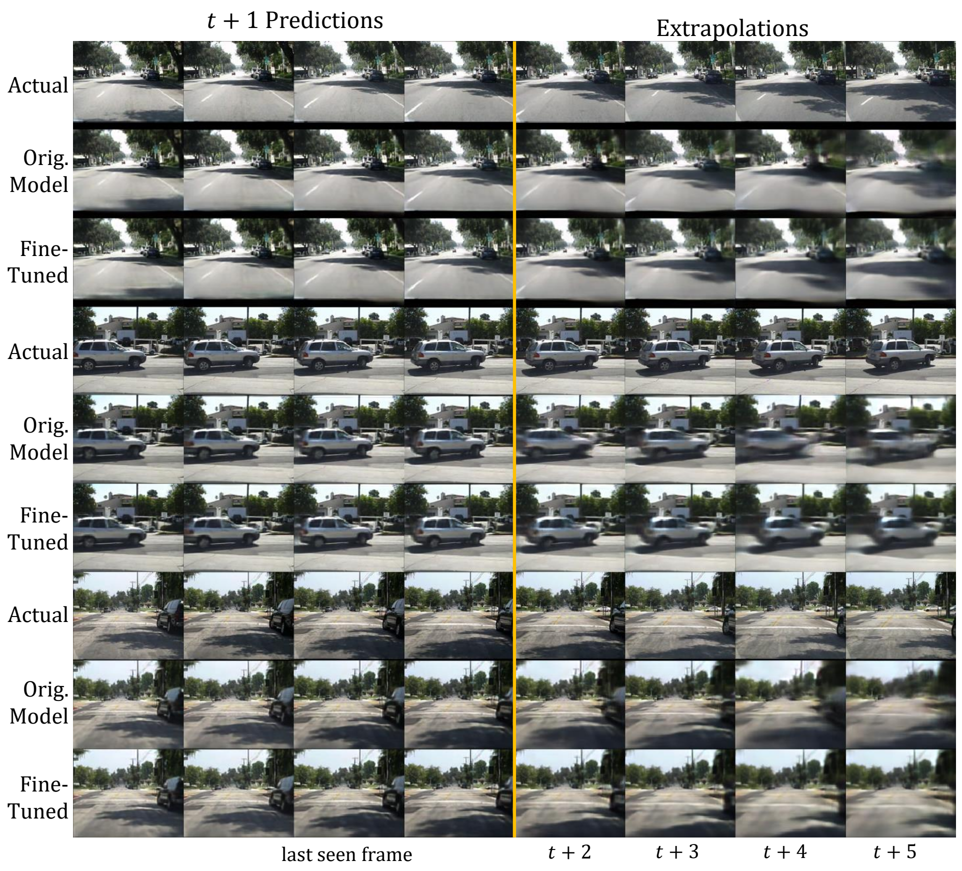

Figure 5: Extrapolation sequences generated by feeding PredNet predictions back into model. The images to the left of the orange line are normal $t + 1$ predictions, whereas the images on the right were generated by recursively using the predictions as input. The first row contains the ground truth sequences. The second row shows the generated frames of the original model, trained to solely predict one time step ahead. The third row contains the sequences of the original model fine-tuned to perform the extrapolation task.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. *CoRR*, 2015.

[2] A. M. Bastos, W. M. Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 2012.

[3] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, 2015.

[4] Y. Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *CoRR*, 2014.

[5] R. Chalasani and J. C. Principe. Deep predictive coding networks. *CoRR*, 2013.

[6] F. Chollet. Keras, 2016.

[7] A. Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 2013.

[8] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *CVPR*, 2009.

[9] T. Egner, J. M. Monti, and C. Summerfield. Expectation and surprise determine neural population responses in the ventral visual stream. *J Neurosci*, 2010.

[10] P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 1991.

[11] K. Friston. A theory of cortical responses. *Philos Trans R Soc Lond B Biol Sci*, 2005.

[12] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[13] D. George and J. Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of the International Joint Conference on Neural Networks. IEEE*, 2005.
[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*. 2014.
[15] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. *CoRR*, 2015.
[16] R. Goroshin, M. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. *CoRR*, 2015.
[17] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, 2013.
[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, 2015.
[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
[20] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*. 2009.
[21] R. Kanai, Y. Komura, S. Shipp, and K. Friston. Cerebral hierarchies : predictive processing , precision and the pulvinar. *Philos Trans R Soc Lond B Biol Sci*, 2015.
[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.
[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.
[24] T. D. Kulkarni, W. Whitney, P. Kohli, and J. B. Tenenbaum. Deep convolutional inverse graphics network. *CoRR*, 2015.
[25] T. S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *J Opt Soc Am A Opt Image Sci Vis*, 2003.
[26] W. Lotter, G. Kreiman, and D. Cox. Unsupervised learning of visual structure using predictive generative networks. *CoRR*, 2015.
[27] D. Maltoni and V. Lomonaco. Semi-supervised tuning from temporal coherence. *CoRR*, 2015.
[28] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, 2015.
[29] V. Michalski, R. Memisevic, and K. Konda. Modeling deep temporal dependencies with recurrent "grammar cells". In *NIPS*. 2014.
[30] H. Mohabi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *ICML*. 2009.
[31] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. P. Singh. Action-conditional video prediction using deep networks in atari games. *CoRR*, 2015.
[32] R. C. O'Reilly, D. Wyatte, and J. Rohrlich. Learning through time in the thalamocortical loops. *CoRR*, 2014.
[33] R. B. Palm. Prediction as a candidate for learning deep hierarchical models of data. *Master's thesis, Technical University of Denmark*, 2012.
[34] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. *CoRR*, 2015.
[35] M. Pezeshki, L. Fan, P. Brakel, A. C. Courville, and Y. Bengio. Deconstructing the ladder network architecture. *CoRR*, 2015.
[36] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol*, 2009.
[37] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, 2015.
[38] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, 2014.
[39] R. P. N. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 1999.
[40] R. P. N. Rao and T. J. Sejnowski. Predictive sequence learning in recurrent neocortical circuits. *NIPS*, 2000.
[41] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-supervised learning with ladder network. *CoRR*, 2015.
[42] A. Saxe, M. Bhand, Z. Chen, P. W. Koh, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *Workshop: Deep Learning and Unsupervised Feature Learning (NIPS)*. 2010.
[43] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, 2015.
[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.
[45] Singular Inversions, Inc. FaceGen. `http://facegen.com`.

[46] W. R. Softky. Unsupervised pixel-prediction. *NIPS*, 1996.
[47] M. W. Spratling. Unsupervised learning of generative and discriminative weights encoding elementary image components in a predictive coding model of cortical function. *Neural Computation*, 2012.
[48] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, 2015.
[49] C. Summerfield, T. Egner, M. Greene, E. Koechlin, J. Mangels, and J. Hirsch. Predictive codes for forthcoming perception in the frontal cortex. *Science*, 314, 2006.
[50] L. Sun, K. Jia, T.-H. Chan, Y. Fang, G. Wang, and S. Yan. Dl-sfa: Deeply-learned slow feature analysis for action recognition. *CVPR*, 2014.
[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, 2014.
[52] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv*, 2016.
[53] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *ICLR*, 2016.
[54] H. Valpola. From neural pca to deep unsupervised learning. *CoRR*, 2015.
[55] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. *CoRR*, 2015.
[56] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.
[57] W. F. Whitney, M. Chang, T. D. Kulkarni, and J. B. Tenenbaum. Understanding visual concepts with continuation learning. *CoRR*, 2016.
[58] L. Wiskott and T. J. Sejnowski. Learning invariance from transformation sequences. *Neural Computation*, 2002.

5 Supplementary Material

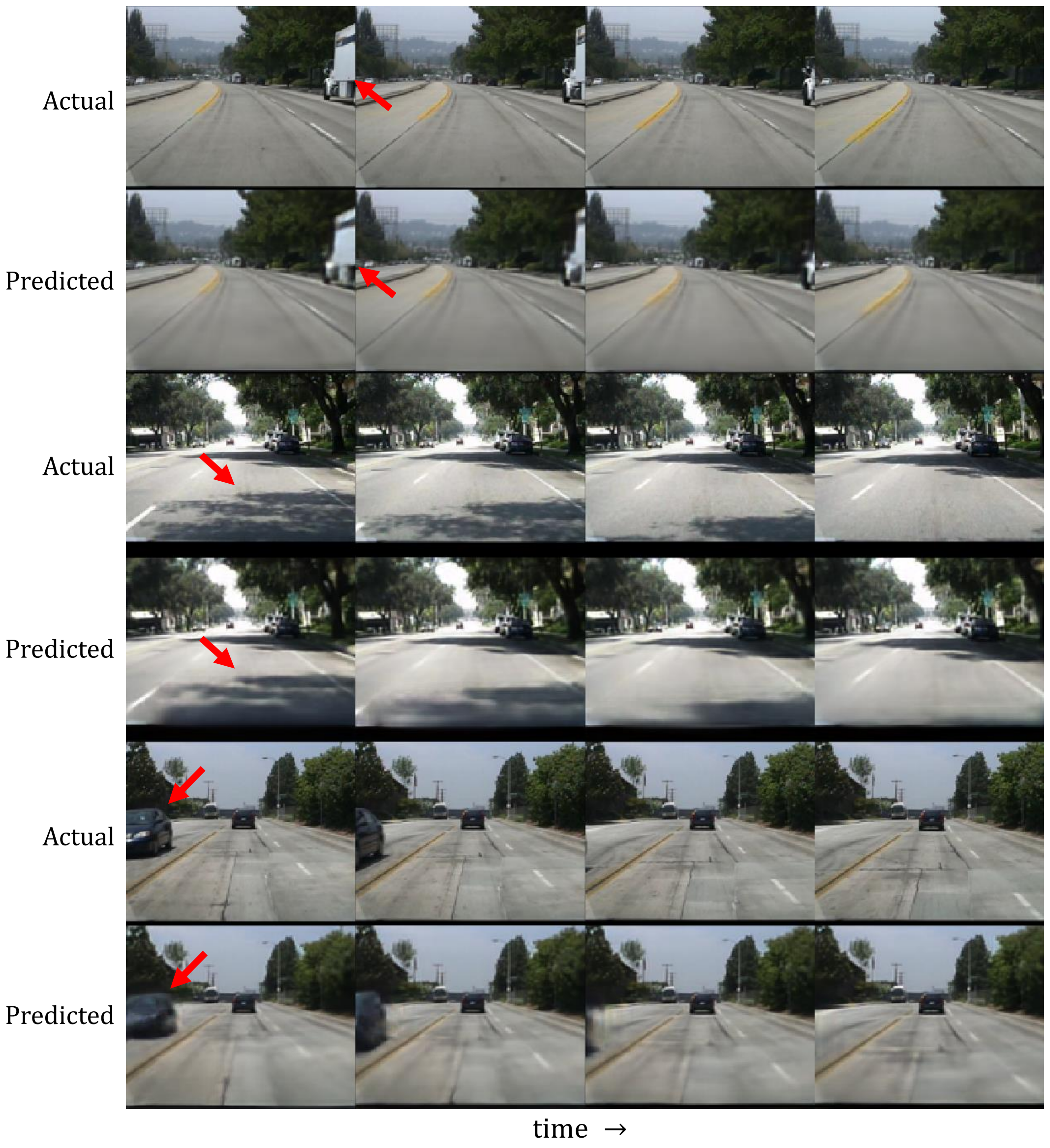

Figure 6: Larger display of PredNet predictions for car-cam sequences. The first rows contain ground truth and the second rows contain predictions. The arrows indicate interest points to compare in consecutive frames. The model was trained on the KITTI dataset and sequences shown are from the Caltech Pedestrian dataset.