HARVARD UNIVERSITY
Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

Committee on Higher Degrees in Biophysics

have examined a dissertation entitled

Prediction as a Rule for Unsupervised Learning in Deep Neural Networks

presented by **William E. Lotter**

candidate for the degree of Doctor of Philosophy and hereby certify that it is worthy of acceptance.

Signature ________________________
Typed name: Prof. Mark L. Andermann

Signature ________________________
Typed name: Prof. Jan Drugowitsch

Signature ________________________
Typed name: Prof. Samuel J. Gershman

Signature ________________________
Typed name: Prof. James M. Hogle

Date: July 6, 2017

Prediction as a Rule for Unsupervised Learning in Deep Neural Networks

A DISSERTATION PRESENTED
BY
WILLIAM E. LOTTER
TO
THE DEPARTMENT OF BIOPHYSICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
BIOPHYSICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
JULY 2017

Thesis advisors: Professors David Cox & Gabriel Kreiman William E. Lotter

Prediction as a Rule for Unsupervised Learning in Deep Neural Networks

Abstract

While machine learning systems have recently achieved impressive, (super)human-level performance in several tasks, they have often relied on unnatural amounts of supervision – e.g. large numbers of labeled images or continuous scores in video games. In contrast, human learning is largely unsupervised, driven by observation and interaction with the world. Emulating this type of learning in machines is an open challenge, and one that is critical for general artificial intelligence. Here, we explore prediction of future frames in video sequences as an unsupervised learning rule. 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 object structure and the possible transformations objects can undergo. To this end, we have designed several models capable of accurate prediction in complex sequences.

Our first model consists of a recurrent extension to the standard autoencoder framework. Trained end-to-end to predict the movement of synthetic stimuli, we find that the model learns a representation of the underlying latent parameters of the 3D objects themselves. Importantly, we find that this representation is naturally tolerant to object transformations, and generalizes well to new tasks, such as classification of static images. Similar models trained solely with a reconstruction loss fail to generalize as effectively. In addition, we explore the use of an adversarial loss, as in a Generative Adversarial Network, illustrating its complementary effects to traditional pixel losses for the task of next-frame prediction.

Next, we propose a novel architecture based on the concept of predictive coding from the neuroscience literature. The model, which we informally call the "PredNet", is trained to continually make hierarchical predictions of future video frames. Top-down and lateral connections convey these predictions, and residual errors are propagated forward. We again find that the model learns a robust representation of the underlying stimuli in artificial video sequences. The model can also 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. In this setting, the representation learned is useful for estimating the steering angle of the car.

Finally, we examine a variety of neural phenomena through the lens of our predictive coding model. First, we demonstrate that our model exhibits extra-classical receptive field effects commonly observed in biological visual processing, specifically end-stopping and surround suppression. These effects are disrupted when the recurrent connections in the model are silenced. Going beyond simple stimuli, we find that our model expresses a norm-based coding of faces, akin to neurophysiology findings in macaques. Lastly, our model provides insight to the well-studied flash-lag illusion. Trained on natural stimuli, the model's outputted predictions align with the common percept of the illusion, providing an empirical explanation of the effect. Altogether, our results suggest that prediction is a prominent component of neural processing. Combined with the machine learning experiments, our efforts demonstrate the potential of prediction as powerful source of unsupervised learning in artificial and biological deep neural networks.

Contents

0 Introduction 1

1 Predictive Generative Networks 12
- 1.1 Predictive Generative Network Formulation 13
- 1.2 Prediction Performance 16
- 1.3 Exploring Latent Representation Learning 19

2 PredNet: A Deep Predictive Coding Network 26
- 2.1 The PredNet Model 27
- 2.2 Experiments on Rendered Image Sequences 31
- 2.3 Experiments on Natural Image Sequences 37

3 Reproducing Neural Phenomena using the PredNet Model 50
- 3.1 Surround Suppression 51
- 3.2 End-Stopping 53
- 3.3 Norm-Based Coding of Faces 55
- 3.4 Flash-Lag Effect 57

4 Conclusion 60

Appendix A Deep Learning for Medical Imaging: Application to Screening Mammography 63
- A.1 Background on Computer Vision for Mammography 64
- A.2 Multi-Scale CNN with a Curriculum Learning Approach 66
- A.3 Experiments on the Digital Database for Screening Mammography 68
- A.4 The Digital Mammography DREAM Challenge 73

References 85

To my Mom, the strongest person I know and who taught me to never give up.

Acknowledgments

There are many people whom I'd like to thank for making this dissertation possible.

First, I'd like to thank my advisors, Dave and Gabriel. I'll forever be grateful for the opportunities and enduring support you have given me. You provided an atmosphere that allowed me to explore many projects, giving me the freedom to ultimately find my interests. Dave, thanks especially for answering all of my odd-hour Slack messages.

I'd also like to thank the members of my dissertation and defense committees, including Sam Gershman, Rick Born, Josh McDermott, Mark Andermann, and Jan Drugowitsch. It's been a privilege to talk science with people as knowledgeable and intelligent as you. Your input has been indispensable in helping me develop the scientific narrative around my work.

To Jim and Michele, the support you give to Biophysics students was evident the day of my first visit through the day after my defense. The flexibility of the program, coupled with your constant support, creates a ideal environment for young scientists.

Thank you to everyone in the Cox and Kreiman labs for your support and friendship throughout the years. I want to especially thank Hanlin Tang and Kenyon Tsai. You both took me under your wings, patiently answering all my questions and teaching me valuable lessons in both science and life. Thank you also to Julie Rhee. Starting from the days of our shared recruitment visits, it's been fun going through the whole process together. Thanks also for being adamant that I rotate in Dave's lab.

Last but not least, I want to thank my family and friends. Thanks Mom, for teaching me to always follow my dreams. Thanks Makayla, for putting up with my late night trips to lab and my urge to read papers on vacation – but for real, thanks for always being supportive and helping me stay balanced.

0
Introduction

A defining characteristic of the brain is its ability to learn meaningful representations from noisy, high-dimensional sensory data. It has been estimated that the human retina alone transmits about 10 million bits per second to the brain (Koch et al., 2006). This deluge of information is rapidly transformed into a behaviorally useful format, representing underlying latent factors such as the identity, spatial configuration, and physical properties of objects in the scene. While the characterizations of these neural representations have been heavily studied, less is known about how they are learned, or the cost functions employed to guide their development.

From the perspective of artificial intelligence, developing systems that represent the world in a fashion that mimics the brain is a holy grail. For highly specific tasks, the field of deep learning has begun to produce solutions that rival (or perhaps surpass) the performance of humans (Mnih et al., 2015; Silver et al., 2016; Esteva et al., 2017). Inspired by principles of neural computation, these models learn representations that enable object categorization, speech recognition, and even policies for mastering video games (LeCun et al., 2015). Unlike the brain, in these systems, we know exactly the cost functions employed. By far the most successful approach, when applicable, has been supervised learning. Here the loss is tightly coupled to the task at hand, and training involves large amounts of labeled data. A canonical example of this is object recognition using the ImageNet dataset (Russakovsky et al., 2014). Consisting of 1.2 million images of 1000 object categories, with an object label for each image, ImageNet has been a driving force for the development of modern deep learning architectures. After training on ImageNet, the representation learned in these models is useful for many other tasks, such as object detection, scene classification, and object localization (Oquab et al., 2015; Huh et al., 2016).

While deep learning shares inspiration from neuroscience, the reliance on supervised learning is quite a divergence. Infants may receive occasional labels from a caretaker, but much of human learning is guided simply by observing and interacting with the world. That is, humans learn in largely an unsupervised fashion. Determining the cost functions that drive this learning would have a profound effect on our understanding of the brain, as well as the development of intelligent machines.

In a sense, describing human learning as "unsupervised", is perhaps actually misleading. Although it is clear that strong supervision (i.e. "This is a cat. That is a dog.") is minimal, there are many other forms of supervisory signals in the world. For instance, from an occluded view of a face, the locations of unseen facial landmarks are easy to estimate, given our heavy experience with faces. From a side view of someone's face, it's easy to picture the appearance of a frontal view. One could even envision creating a training set of such side and frontal view pairs, where the goal is to map from one to the other, training in a purely supervised manner. While this may seem unnatural, it's conceivable how such pairs could arise in real life, e.g., by observing someone rotate their head. In this fashion, it's also easy to imagine how one could build a view-invariant representation, learning a feature space that only slowly changes while observing this rotation. This is in fact the intuition behind the influential approach known as "Slow Feature Analysis (SFA)" (Földiák, 1991; Wiskott & Sejnowski, 2002). While not yet yielding representations as powerful as those learned by strongly supervised methods, deep learning implementations of SFA have indeed pointed to the potential of learning useful representations from video (Mohabi et al., 2009; Goroshin et al., 2015a; Wang & Gupta, 2015; Sun et al., 2014; Jayaraman & Grauman, 2015).

Here, we explore another potential approach for unsupervised learning from video: prediction of future image frames. At its core, effective prediction requires an internal model of the world and an understanding of the rules by which the world changes. For instance, in the rotating face example, accurate predictions rely on knowledge of faces (i.e. they tend to have two ears and eyes), and general physical properties of world (i.e. 3D geometry, symmetry, lighting). Hence, we argue that prediction can serve as a powerful training signal in biological and artificial neural networks, a claim we support through a series of modeling experiments.

We begin in Chapter 1 by extending the framework of an autoencoder to the next-frame prediction task. Autoencoders, the traditional approach to unsupervised learning in neural networks, are trained with a reconstruction loss (Hinton & Salakhutdinov, 2006). Using a dataset of computer-generated faces, we demonstrate that prediction is a more powerful loss, allowing for better decoding of latent parameters and an increase in classification performance. We also explore the use of an adversarial loss (AL), as in Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), for next-frame prediction. We find that the AL effectively combats common issues found with traditional pixel losses, such as blurring.

Next, in Chapter 2, we take direct inspiration from the neuroscience literature, specifically the idea of predictive coding, to propose a novel deep learning architecture (the “PredNet”). Predictive coding suggests that the brain is constantly making predictions of incoming stimuli, where feedback connections convey these predictions and feedforward connections relay the residual errors (Rao & Ballard, 1999; Friston, 2005). We implement a deep learning formulation of predictive coding, which proves to be very effective in making predictions for both artificial (generated) and natural stimuli. Again 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.).

Finally, in Chapter 3, we demonstrate that our PredNet model can reproduce a variety of neural and psychophysical phenomenon. In the seminal paper of Rao & Ballard (1999), the authors demonstrate that their predictive coding model trained on natural images exhibits several extra-classical receptive field effects observed in visual cortex. We show that our model behaves similarly, reproducing effects of end-stopping and surround suppression. These effects are mitigated when the recurrent connections, carrying predictions based on natural statistics, are effectively “cooled” (Nassi et al., 2014). Thus, our results support the notion that these effects result from the efficient representation of natural stimuli, formulated as predictive coding. Interestingly, we note that these static, spatially dependent phenomena arise in our model trained for temporal prediction with dynamic stimuli, as opposed to the static images in Rao & Ballard (1999). Next, we demonstrate that our model also exhibits properties of the norm-based coding of faces, where the overall feedforward response is correlated with the amount of deviation from the average face (Rhodes & Jeffery, 2006; Leopold, 2017). Lastly, we explore the behavior of our model in response to the flash-lag illusion (Mackay, 1958; Nijhawan, 1994). We find that the model’s predictions match the common percept of the illusion, which is in fact quite different from the ground truth in pixel space.

In summary, the (super)human-level performance of deep learning on a number of tasks is quite impressive. However, although the overall framework is inspired by the brain, the reliance on large number of labeled examples stands at odds with biological learning. A potential driver of this learning is prediction. To explore prediction as unsupervised learning, we have developed models that can make robust predictions of future frames in complex video sequences. Importantly, the representations learned in this paradigm support the decoding of underlying latent variables, and tasks such as classification. In addition, our models reproduce a wide variety of neural phenomena, suggesting that prediction may indeed be important in neural processing and learning.

0.1 Related Work

Our approach is closely related to other works beyond which we have already mentioned, as we elaborate below.

"Time as a Teacher"

The notion of "Time as a Teacher" has strong roots in both computational neuroscience and machine learning. Early efforts in this field demonstrated how invariances to particular transformations can be learned through temporal exposure (Földiák, 1991). As mentioned above, Slow Feature Analysis (SFA) shares a similar motivation and aims to build feature representations that extract slowly-varying parameters, such as object identity, from parameters that produce fast changes in the image, such as pose and position (Wiskott & Sejnowski, 2002). Deep learning instantiations of SFA have been successful for tasks such as action recognition (Sun et al., 2014; Jayaraman & Grauman, 2015). A slightly different formulation of SFA, temporal coherence, seeks to learn a feature space where the distance between frames from the same video is minimal, and maximal for frames from different videos (Mohabi et al., 2009; Goroshin et al., 2015a; Wang & Gupta, 2015).

Also related to temporal learning, especially in the context of rotating objects, is the field of relational feature learning (Memisevic & Hinton, 2007; Taylor & Hinton, 2009). This posits modeling time-series data as learning representations of the *transformations* that take one frame to the next. Recently, Michalski et al. (2014) proposed a predictive training scheme where a transformation is first inferred between two frames and is then applied again to obtain a prediction of a third frame. They reported evidence of a benefit of using predictive training versus traditional reconstruction training.

Several works have explored precisely the idea of learning from temporal prediction in deep networks. Palm (2012) coined the term "Predictive Encoder" to refer to an autoencoder that is trained to predict future input instead of reconstructing current stimuli. In preliminary experiments, it was shown that such a model could learn Gabor-like filters in training scenarios where traditional autoencoders failed to learn useful representations. George & Hawkins (2005) provide a hierarchical bayesian model that learns the probability of temporal sequences, which they use for motivating invariant pattern recognition. O'Reilly et al. (2014) argue that a primary function of cortex is temporal prediction, which they model with their LeabraTI system. Goroshin et al. (2015b) propose training models to linearize transformations observed over sequences of frames in natural video.

Next-Frame Video Prediction

Tightly coupled to representation learning from video, there has been a recent flurry of models designed for next-frame prediction (Ranzato et al., 2014; Srivastava et al., 2015; Patraucean et al., 2015; Mathieu et al., 2016; Kalchbrenner et al., 2016; Finn et al., 2016; Xue et al., 2016; Brabandere et al., 2016; Villegas et al., 2017; Villegas et al., 2017). Ranzato et al. (2014) formulated the problem closely to character prediction in natural language processing (Graves, 2013) by discretizing image patches into a dictionary set. Srivastava et al. (2015) proposed a Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) encoder-decoder model for next frame prediction and demonstrated that predictive pre-training improved performance on an action recognition task. Patraucean et al. (2015) combine an encoder-decoder framework with a convolutional LSTM unit to estimate optical flow, which is used to transform the previous frame to the proposed next frame. To address the blurriness inherent with traditional losses, Mathieu et al. (2016) incorporate an adversarial GAN loss. Vondrick et al. (2016) also use a GAN paradigm to generate entire video sequences from scratch. They demonstrate that the representation learned in this framework is useful for action recognition. Xue et al. (2016) use a conditional variational autoencoder (Kingma & Welling, 2013) to propose a model that samples future frames using separate image and motion streams. In a pair of papers, Villegas and coauthors (Villegas et al., 2017; Villegas et al., 2017) similarly add more structure and composition to video prediction models, using hierarchical predictions, as well as decomposing motion and content. Finally, it should be noted that, before the advent of modern deep learning, Softky (1996) proposed to use pixel prediction as a means for unsupervised learning, building a cortically-inspired model that could make predictions in simple moving stimuli.

Self-Supervised Learning in Deep Neural Networks

A term that is becoming increasingly popular is "self-supervised learning". This term has come to represent methods that are unsupervised in the sense that they don't rely on explicit labels, but supervised in that they use a loss function depending on input-output pairs. The key is that these input-output pairs are essentially generated for free. Next-frame prediction, as posed in our work, falls under this category because models are trained end-to-end to predict future frames given previous frames, but these targets are trivial to obtain from unlabeled video.

Many other proposed self-supervised methods also rely on temporal dynamics (Fernando et al., 2016; Agrawal et al., 2015; Purushwalkam & Gupta, 2016; Luo et al., 2017). For instance, Misra et al. (2016) pose a sequential verification task where the goal is to determine if a set of frames is in the correct temporal order. They demonstrate that the representation learned in this framework is useful for action recognition. In a different approach relying on motion, Pathak et al. (2016) use unsupervised motion cues to create approximate segmentation maps of objects in videos. Training a CNN to then reproduce these segmentation maps allows for effective transfer learning on object detection.

Spatial statistics are another property that is often exploited in self-supervised learning (Wu et al., 2017; Cruz et al., 2017). For instance, Doersch et al. (2015) train CNNs to estimate the relative location of one patch of an image with respect to another patch. The resulting representation is useful for object detection. Similarly, Noroozi & Favaro (2016) use CNNs to solve jigsaw puzzles, created by scrambling patches of an image, which later helps for detection and classification. Other approaches that have been proposed for self-supervised learning include colorization (Larsson et al., 2017; Zhang et al., 2016a), cross-channel prediction (Zhang et al., 2016b), and audio-visual correspondence (Arandjelovic & Zisserman, 2017),

There have been a number of recent works that have used prediction as a self-supervision mechanism in reinforcement learning settings. Oh et al. (2015) propose an action-conditional visual prediction model for the Atari game evaluation suite, which they show can be used for more efficient exploration. Dosovitskiy & Koltun (2016) develop an agent that can make predictions of future measurements (e.g. health, score) conditioned on all possible actions, enabling efficient action selection. Pathak et al. (2017) demonstrate that prediction error (in a learned feature space of the environment) can be used as an intrinsic, curiosity reward signal to drive exploration, which is ultimately useful in later tasks.

Predictive Coding

As mentioned above, our work is heavily inspired by Rao & Ballard (1999). There are several other compelling papers by Rajesh Rao, where he suggests predictive coding could also play a role in visual attention (Rao, 1998b) and even object segmentation (Rao, 1998b). Overall, there has been much work supporting the biological relevance of predictive coding (Summerfield et al., 2006; Clark, 2013; Egner et al., 2010; Issa et al., 2016; Kanai et al., 2015; Spratling, 2012; Bastos et al., 2012). One particularly interesting example is the experiment of Murray et al. (2002). The authors present visual stimuli that are either grouped into objects or randomly arranged to human subjects while monitoring activity using fMRI. They observed significant increases in activity in the lateral occipital complex, a higher object processing area, along with significant decreases in activity in the primary visual cortex when presenting the stimuli as grouped objects. A predictive coding view is a potential explanation, where inferences of higher areas explain-away activity in lower areas through feedback.

0.2 Additional Background Material

Long Short-Term Memory (LSTM)

A standard deep learning component for time-series problems is the recurrent neural network (RNN). In it's most basic form, a recurrent neural network consists of a hidden state that updates at each discrete time step based on the value of the hidden state at the previous time step, and the input into the network at the current time step. Explicitly, an RNN is generally specified as,

$$h_t = f(h_{t-1}, x_t) \tag{1}$$

where h is the hidden state vector and x is the input to the network. Different forms of RNNs correspond to different specifications of the function f. A vanilla RNN has the form of

$$h_t = \sigma(W_h h_{t-1} + W_x x_t) \tag{2}$$

where W_h and W_x correspond to weight matrices for the hidden state and input, respectively, and σ is a non-linearity, such as a sigmoid.

Depending on the task, the RNN hidden state is eventually used to estimate a target, y_T, using an output $\hat{y_T} = g(h_T)$, where g is another function and T is the time step of interest. The standard way of training RNNs is via backpropagation thru time (BPTT), which amounts to unrolling the network in time and then applying the standard backprop algorithm. For long sequences, however, this often results in poor training due to vanishing and exploding gradient issues. There have been many attempts to overcome these issues, which often amounts to choosing different state update functions f. In practice, the most popular RNN variant has been the Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997).

LSTMs contain a cell state and several gates designed to alleviate the long-term dependency issues inherent in BPTT. The cell, c_t, can be thought of as a memory state. Access to the cell is controlled through an input gate, i_t, and a forget gate, f_t. The final output of the LSTM unit, h_t, is a function of the cell state, c_t, and an output gate, o_t. A traditional LSTM has the following update equations:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
h_t &= o_t \tanh(c_t)
\end{aligned}$$

where x_t is the input to the LSTM network, $W_{\bullet\bullet}$ are the weight matrices, and σ is the elementwise logistic sigmoid function. Convolutional LSTMs, useful when the input consists of images, involve simply replacing the matrix multiplication operators with a convolutional operators (Patraucean et al., 2015; Shi et al., 2015).

Generative Adversarial Networks

In the trend of learning as much as possible from the data instead of hand-specifying, Generative Adversarial Networks (GANs) are a clever method for learning a loss function for generative models (Goodfellow, 2017). In many applications, such as generating natural images, it is difficult to precisely define a loss function in a concise, analytical form. GANs provide a way around this by *learning* a loss function based on comparing natural images from the dataset and synthetically generated samples. Introduced by Goodfellow et al. (2014), GANs consist of a generator (G) and a discriminator (D) with a game theoretic formulation. Given a set of real data $\{x_{data}\}$, the generator G is trained to generate realistic samples $\{x_{model}\}$ and the discriminator D is trained to discern between x_{data} and x_{model}. Samples are generated using G by passing a random vector z, sampled from a specified distribution (i.e. uniform), through a deterministic function, like a deep neural network, to produce an output with same dimensions of the data. D is another neural network that is trained to output 1 for real samples, x_{data}, and 0 for generated samples, $G(z)$. Training of G and D typically entails alternating, minimax-like updates. Given a mini-batch size of n the loss functions are specified below:

$$L_D = -\frac{1}{2n}\sum_{i=1}^{n}[\log D(x^i_{data}) + \log(1 - D(G(z_i))]$$

$$L_G = \frac{1}{n}\sum_{i=1}^{n}\log(1 - D(G(z_i)))$$

1

Predictive Generative Networks

The traditional approach for unsupervised learning with neural networks is through the use of autoencoders (Hinton & Salakhutdinov, 2006; Vincent et al., 2008b; Erhan et al., 2010). An autoencoder consists of an encoder, a latent code, and a decoder, where the goal is to reconstruct the input into the network. Through regularization and/or a compressive scheme, the hope is that the

Material contained in this chapter has been published in Lotter et al. (2016).

model learns a meaningful representation of its inputs, expressed in the learned latent code. While autoencoders have had some success in pre-training of neural networks (Erhan et al., 2010), pure supervised learning has been proven to be much more effective at learning useful representations, at least when given enough data (Huh et al., 2016). A potential reason for this could be that reconstruction is simply not an effective loss. Indeed, at one extreme, without proper regularization, an autoencoder can learn to simply copy its input, thus learning a trivial representation. Augmenting training with a denoising criteria (Vincent et al., 2008b), for instance, can offer improvements, but here we are interested in going beyond reconstruction, and instead, training for prediction.

1.1 Predictive Generative Network Formulation

The first model for which we explore the use of prediction for unsupervised learning is a simple extension to an autoencoder, where we make the latent code recurrent, specifically an LSTM (Hochreiter & Schmidhuber, 1997). The architecture, which we refer to as a Predictive Generative Network (PGN), is illustrated in the left side of Fig 1.1. The model consists of a convolutional neural network (CNN), followed by an LSTM, which outputs to deconvolutional neural network (deCNN). Falling into the class of Encoder-Recurrent-Decoder architectures (Fragkiadaki et al., 2015), the model is trained end-to-end to combine feature representation learning with the learning of temporal dynamics. The idea is that the CNN transforms the images into a feature space where the LSTM can learn the dynamics, and then the deCNN inverts the latent code at the last time step to the predicted next frame.

In next-frame prediction, choosing a loss between the predicted frame and the actual frame is a nontrivial task. Typical losses such as mean-squared error (MSE) don't reflect perceptual similarity well, and often result in blurring, as any uncertainty is averaged-out. A promising alternative to this, at least for combatting blurring, is using an adversarial loss, as in Generative Adversarial Networks

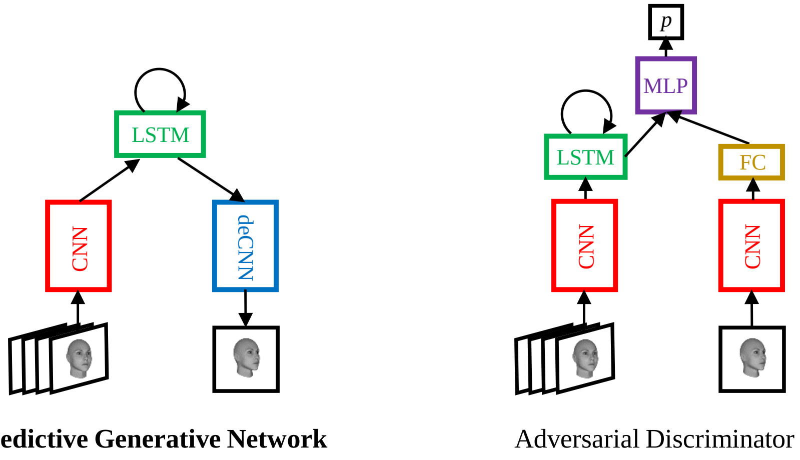

Figure 1.1: Predictive Generative Network (PGN)

(GANs) (Goodfellow et al., 2014). As described in the Introduction, GANs consist of a generator and a discriminator, where the generator is trained to generate realistic samples, and the discriminator is trained to discern between real and fake samples. In the current scenario, if the PGN outputs blurry predictions, the discriminator could learn these artifacts, causing the PGN to better match natural statistics to fool the discriminator.

Our proposed form of an adversarial discriminator for a PGN is shown on the right side of Fig 1.1. For next-frame prediction, there are two basic modes for which a generated prediction can fail. First, the image could simply not resemble a natural image (i.e. because of blurriness). Second, the predicted next frame could not correspond with the previous frames in the sequence. Our discriminator was designed with both of these failure modes in mind. The previous frames and the proposed next frame are both inputs into the network. All of the frames are processed by a CNN, where the CNN features for the previous frames are then condensed by an LSTM into a single vector. The dimensionality of the features for the proposed next frame are reduced by a fully-connected layer, and then concatenated with the feature vector representing the previous frames. A final multi-layer perceptron (MLP) readout outputs the predicted probability that the frame is the true next frame.

Overall, the discriminator and PGN form a conditional GAN pair. In some conditional GANs, a random vector is also an input into the generator, allowing a form of sampling (Mirza & Osindero, 2014; Gauthier, 2014). We attempted concatenating the LSTM output with a random vector before passing to the deCNN, but this did not lead to any noticeable effects in the trained network. This is likely because the distribution of possible next frames is highly peaked in our datasets, adding to the commonly observed problem of mode collapse in GANs. Further methods have since been suggested to ameliorate the mode collapse in GANs, which may be useful in our model, but in the presented results, we discarded the random vector input.

We use the original formulation of the adversarial loss function (Goodfellow et al., 2014). Let $x^i_{1:t}$ be an input sequence of t frames and x^i_{t+1} be the true next frame. Let the proposed frame from the generator be $G(x^i_{1:t})$ and $D(\bullet, x^i_{1:t})$ be the discriminator's output. Given a mini-batch size of n sequences, the loss of the discriminator, $L_D^{(AL)}$, and of the generator, $L_G^{(AL)}$, have the form:

$$\begin{aligned} L_D^{(AL)} &= -\frac{1}{2n}\sum_{i=1}^{n}[\log D(x^i_{t+1}, x^i_{1:t}) + \log(1 - D(G(x^i_{1:t}), x^i_{1:t}))] \\ L_G^{(AL)} &= \frac{1}{n}\sum_{i=1}^{n}\log(1 - D(G(x^i_{1:t}), x^i_{1:t})) \end{aligned}$$

As in the original paper (Goodfellow et al., 2014), we actually train the generator to maximize $\log(D(G(x^i_{1:t}), x^i_{1:t}))$ and not minimize $\log(1 - D(G(x^i_{1:t}), x^i_{1:t}))$, as the latter tends to saturate early in training.

In the end, we noticed best results when we combined MSE and the AL into a weighted loss, $L_G^{(tot)}$, controlled by a hyperparameter λ:

$$L_G^{(tot)} = L_G^{(MSE)} + \lambda L_G^{(AL)}$$

Even for high values of λ, the MSE loss proved to be useful as it stabilizes and accelerates training.

1.2 Prediction Performance

We evaluated the Predictive Generative Networks (PGNs) on two datasets of synthetic video sequences. As a baseline to compare against other architectures, we first report performance on a standard bouncing balls paradigm (Sutskever et al., 2009a). We then proceed to a dataset containing out-of-the-plane rotations of computer-generated faces, where we thoroughly analyze the learned representations.

For both datasets, the CNNs in the model consist of two layers of alternating convolution, ReLU activation, and max-pooling. The output is flattened and passed to a LSTM of 1568 units for the bouncing balls and 1024 for the rotating faces. The deCNN consists of two-layers of nearest-neighbor unsampling, convolution, and ReLU activation. The last layer also contains a saturating non-linearity set at the maximum pixel value. Due to the higher dimensional size of the rotating faces, an additional fully-connected (FC) layer is used between the LSTM and deCNN. When the adversarial discriminator is used, it consists of similar implementations of the CNN and LSTM, with three FC layers in the MLP readout.

1.2.1 Bouncing Balls

The bouncing balls dataset is a common test set for models that generate high dimensional sequences. It consists of simulations of three balls bouncing in a box. We followed standard procedure to create 4000 training videos and 200 testing videos (Sutskever et al., 2009b) and used an additional 200 videos for validation. Our networks were trained to take a variable number of frames as input, selected randomly each epoch from a range of 5 to 15, and output a prediction for the next timestep. Training with MSE was very effective for this dataset, so AL was not used. Models were optimized using RMSprop (Tieleman & Hinton, 2012) with a learning rate of 0.001. In Table 1.1, we report the average squared one-step-ahead prediction error per frame. Our model compares fa-

Table 1.1: Average prediction error for the bouncing balls dataset. †(Gan et al., 2015) ◇(Mittelman et al., 2014)

Model	Error
PGN (MSE)	$\mathbf{0.65 \pm 0.11}$
DTSBN †	2.79 ± 0.39
SRTRBM ◇	3.31 ± 0.33
RTRBM ◇	3.88 ± 0.33
Frame t-1	11.86 ± 0.27

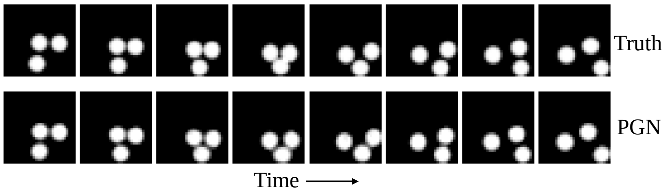

Figure 1.2: Example prediction sequence for bouncing balls dataset. Predictions are repeatedly generated one step ahead using the prior ten frames as input.

vorably to the recently introduced Deep Temporal Sigmoid Belief Network (Gan et al., 2015) and restricted Boltzmann machine (RBM) variants, the recurrent temporal RBM (RTRBM) and the structured RTRBM (SRTRBM) (Mittelman et al., 2014). An example prediction sequence is shown in Figure 1.2, where each prediction is made one step ahead by using the ten previous frames as input.

1.2.2 Rotating Faces

For the rotating faces dataset, each video consists of a unique, randomly generated face rotating about the vertical axis with a random speed and initial angle. Speed is sampled uniformly from $[0, \pi/6]$ rad/frame with an initial angle sampled from $[-\pi/2, \pi/2]$, where 0 corresponds to a frontal view. Input sequences consist of 5 frames of size 150x150 pixels. We use 4000 clips for training and 200 for validation and testing.

Generative models are often evaluated using a Parzen window estimate of the log-likelihood (Breuleux et al., 2011), but due to the deficiencies of this approach for high dimensional images, we chose values of the weighting parameter between MSE and AL, λ, based on qualitative assessment. Adversarial models are notoriously difficult to train and we empirically found benefits in giving the generator and discriminator a “warm start”. For the generator, this corresponded to initializing from a solution trained solely with MSE. This is analogous to increasing the value of λ over training, thus ensuring that the models learn the low frequency components first, and then the high frequency components, akin to the LAPGAN approach (Denton et al., 2015). For the discriminator, we used a pre-training scheme where it was first trained against a generator with a high value of λ. This exposes the discriminator to a wide variety of stimuli in pixel space early in training, which helps it quickly discriminate between real and generated images when it is subsequently paired with the MSE-initialized generator. For the data shown in this paper, these initialization schemes are used and λ is set to 0.0002. The generator is optimized using RMSprop (Tieleman & Hinton, 2012) with a learning rate of 0.001. The discriminator is trained with stochastic gradient descent (SGD), with a learning rate of 0.01 and momentum of 0.5.

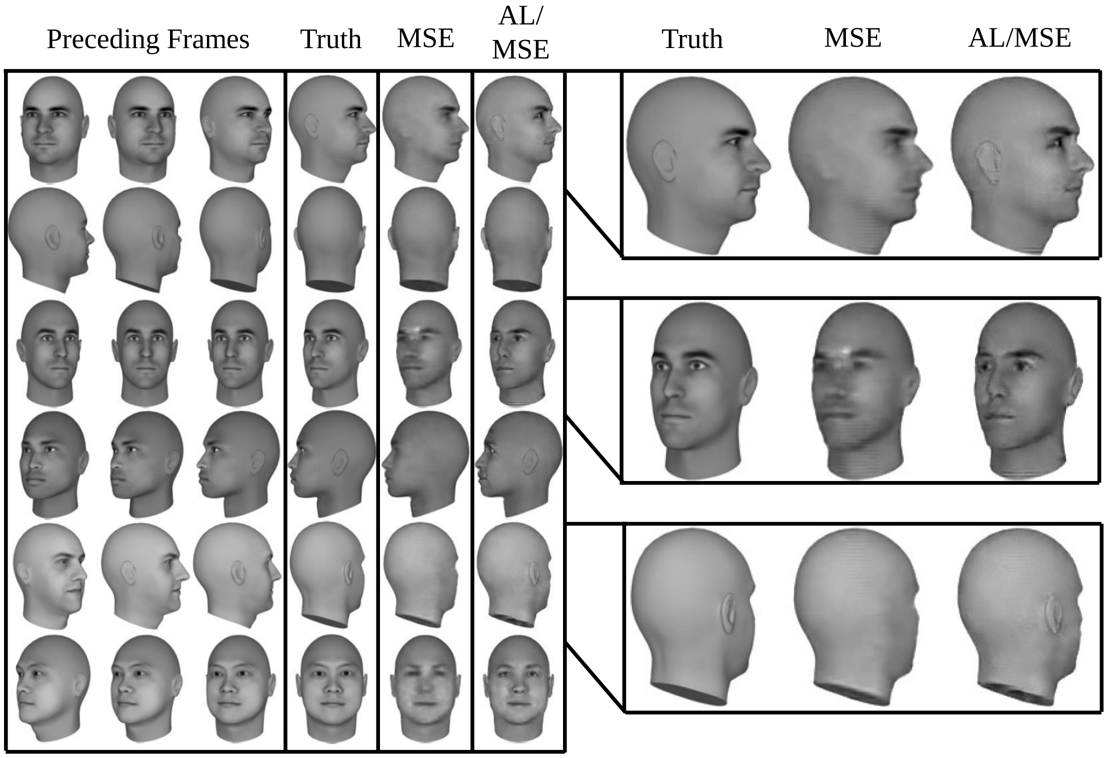

Figure 1.3: Example predictions for the rotating faces dataset. Predictions for models trained with MSE and a weighted MSE and adversarial loss (AL) are shown.

Example predictions are shown in Figure 1.3. We compare the results of training with MSE to the weighted AL/MSE model. Predictions are for faces not seen during training. Both models successfully estimate the angle and basic shape of the face very well. However, the MSE model produces blurred, low-passed versions as expected, whereas the AL/MSE model generates images with high detail. Most notably, the AL/MSE model has learned that faces contain conspicuous eyes and ears, which are largely omitted by the MSE model. When the AL/MSE model does make mistakes, it's often through generating faces that notably look realistic, but seem slightly inconsistent with the identity of the face in the preceding frames. This can be seen in the second row in the right panel of Figure 1.3. Weighting AL higher exaggerates this effect. One would hope that the discriminator would be able to discern if the identity changed for the proposed rotated view, but interestingly, even humans struggle with this task (Wallis & Bulthoff, 2001).

1.3 Exploring Latent Representation Learning

Beyond generating realistic predictions, we are interested in understanding the representations learned by the predictive models, especially in relation to the underlying generative model. The faces are created according to a principal component analysis (PCA) in "face space", which was estimated from real-world faces. In addition to the principal components (PCs), the remaining latent variables are the initial angle and rotation speed.

A decoding analysis was performed in which an L2-regularized regression was used to estimate the latent variables from the LSTM representation. We decoded from the hidden unit responses after five time steps, i.e. the last time step before the hidden representation is outputted to the deCNN to produce a predicted image. The regression was fit, validated, and tested using a different dataset than the one used to train the model.

As a baseline, we compare decoding from the predictive models to a model with the same architecture, trained on precisely the same stimulus set, but with a reconstruction loss. Here, the input sequence is all six frames, for which the model is trained to reconstruct the last. Note, the model cannot simply copy the input, but must learn a low dimensional representation, because the LSTM has a dimension size much less than the input (1024 v.s. $150\text{x}150 = 22.5\text{K}$), e.g. the common autoencoder scenario.

The decoding results for the initial angle, rotation speed, and first four principal components are contained in Table 1.2. Although they produce visually distinct predictions, the MSE and AL/MSE PGNs show similar decoding performance. This is not surprising since the PCs dictate the shape of the face, which the MSE model estimates very well. Nevertheless, both predictive models strongly outperform the autoencoder. There are more sophisticated ways to train autoencoders, including denoising criteria (Vincent et al., 2008a), but here we show that, for a fixed training set, a predictive loss can lead to a better representation of the underlying generative model than a reconstruction loss.

Table 1.2: Decoding accuracy (r^2) of latent variables from the LSTM hidden unit representation.

Model	Angle	Speed	PC1	PC2	PC3	PC4
PGN (MSE)	0.994	0.986	0.877	0.826	0.723	0.705
PGN (AL/MSE)	0.994	0.990	0.873	0.828	0.724	0.686
Autoencoder (MSE)	0.943	0.927	0.834	0.772	0.655	0.635

To gain insight into the learning dynamics, we show decoding performance for both the hidden state and cell state as a function of training epoch for the MSE model in Figure 1.4. Epoch 0 corresponds to the random initial weights, from which the latent variables can already be decoded fairly well, which is expected given the empirical evidence and theoretical justifications for the success of random weights in neural networks (Jarrett et al., 2009; Pinto et al., 2009; Saxe et al., 2010). Still, it is clear that the ability to estimate all latent variables increases over training. The model quickly peaks at its ability to linearly encode for speed and initial angle. The PCs are learned more slowly, with decoding accuracy for some PCs actually first decreasing while speed and angle are rapidly learned. The sequence in which the model learns is reminiscent of theoretical work supporting the notion that modes in the dataset are learned in a coarse-to-fine fashion (Saxe et al., 2013).

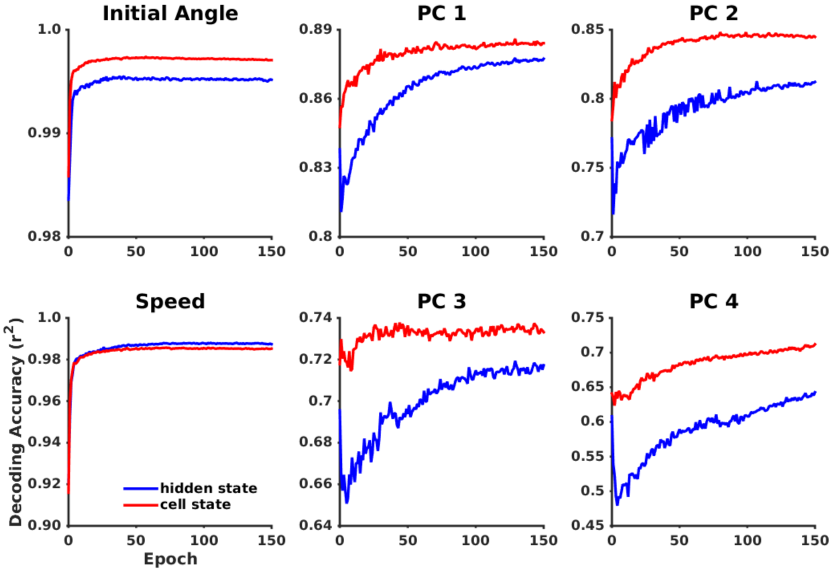

Figure 1.4: Dynamics of latent variable decoding from internal representation of PGN (MSE)

To understand the representational changes accompanying the increase in decoding performance, we provide visualizations of the hidden unit feature space over training in Figures 1.5 and 1.6. Figure 1.5 contains a multidimensional-scaling (MDS) plot for the initial random weights and the weights after Epoch 150 trained with MSE. Points are colored by PC1 value and rotation speed. Although a regression on this feature space at Epoch 0 produces an r^2 of ~0.83, it is apparent that the structure of this space changes with training. To have a more clear understanding of these changes, we linearized the feature space in two dimensions with axes pointing in the direction of the regression coefficients for decoding PC1 and rotation speed. Points from a held-out set were projected on these axes and plotted in Figure 1.6 and we show the evolution of the projection space, with regression coefficients calculated separately for each epoch. Over training, the points become more spread out over this manifold. This is not due to an overall increase in feature vector length, as this does not increase over training. Thus, with training, the variance in the feature vectors become more aligned with the latent variables of the generative model.

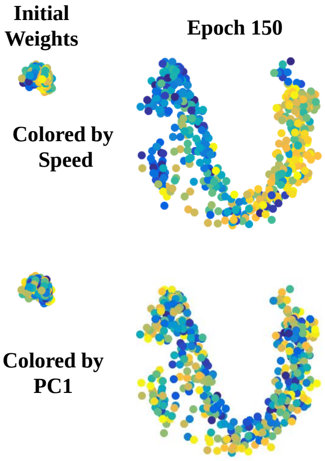

Figure 1.5: Multidimensional scaling plot of the LSTM representation demonstrating changes with training.

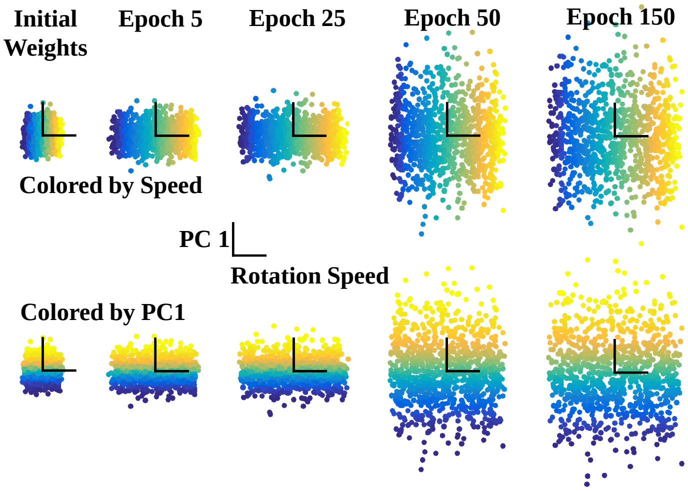

Figure 1.6: Projection of LSTM feature space on latent variables axes. Axes are in the direction of regression coefficients. A different regression was fit for each epoch.

The previous analyses suggest that the PGNs learn a low dimensional, linear representation of the face space. This is further illustrated in Figure 1.7. Here, the feature representation of a given seed face is calculated and then the feature vector is perturbed in the direction of a principal component axis, as estimated in the decoding analysis. The new feature vector is then passed to the pre-trained deCNN to produce an image. The generated image is compared with changing the PC value directly in the face generation software. Figure 1.7 shows that the extrapolations produce realistic images, especially for the AL/MSE model, which correlate with the underlying model. The PC dimensions do not precisely have semantic meanings, but differences can especially be noticed in the cheeks and jaw lines. The linear extrapolations in feature space generally match changes in these features, demonstrating that the models have learned a representation where the latent variables are linear.

While the generation of frame-by-frame future predictions is useful *per se*, we were especially

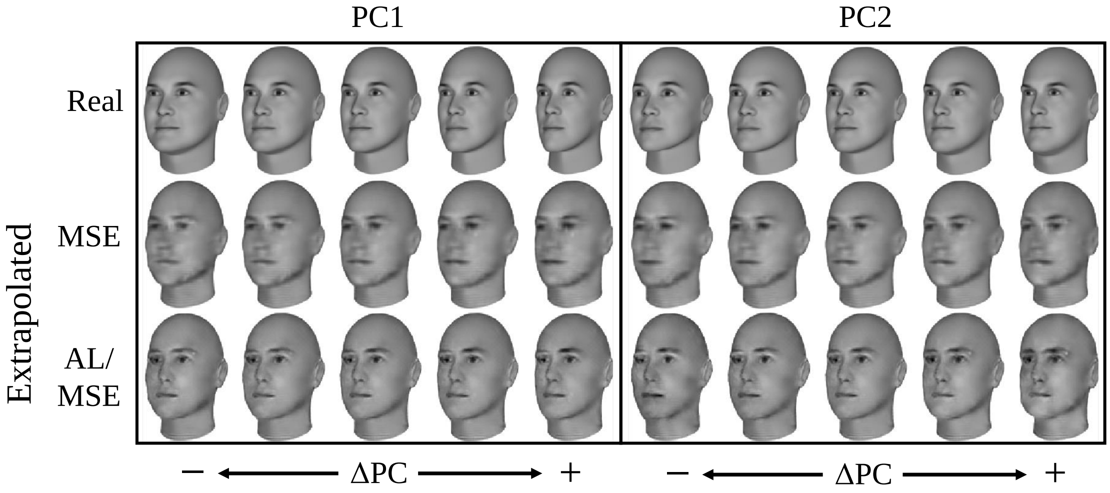

Figure 1.7: Linearly moving through LSTM feature space along principal component axes.

interested in the extent to which prediction could be used as an unsupervised loss for learning representations that are suited to other tasks. We tested this hypothesis through a task completely orthogonal to the original loss function, namely classification of static images. As a control, to specifically isolate the effect of the loss itself, we trained comparable models using a reconstruction loss and either dynamic or static stimuli. The first control was carried over from the latent variable decoding analysis and had the same architecture and training set of the PGN, but was trained with a reconstruction loss (denoted as AE LSTM (dynamic) in Fig. 1.8). The next model again had the same architecture and a reconstruction loss, but was trained on static videos [AE LSTM (static)]. A video was created for each unique frame in the original training set. For the last two models, the LSTM was replaced by a fully-connected (FC) layer, one with an equal number of weights [AE FC (= # weights)] and the other with an equal number of units [AE FC (= # units)] as the LSTM. These were trained in a more traditional autoencoder fashion to simply reconstruct single frames, using every frame in the original video set. All control models were trained with MSE since AL is more sensitive to hyperparameters.

The classification dataset consisted of 50 randomly generated faces at 12 equally-spaced angles between $[-\frac{\Pi}{2}, \frac{\Pi}{2}]$. A support vector machine (SVM) was fit on the feature representations of each model. For the models containing the LSTM layer, the feature representation at the fifth time step was chosen. To test for transformation tolerance, training and testing were done with separate sets of angles.

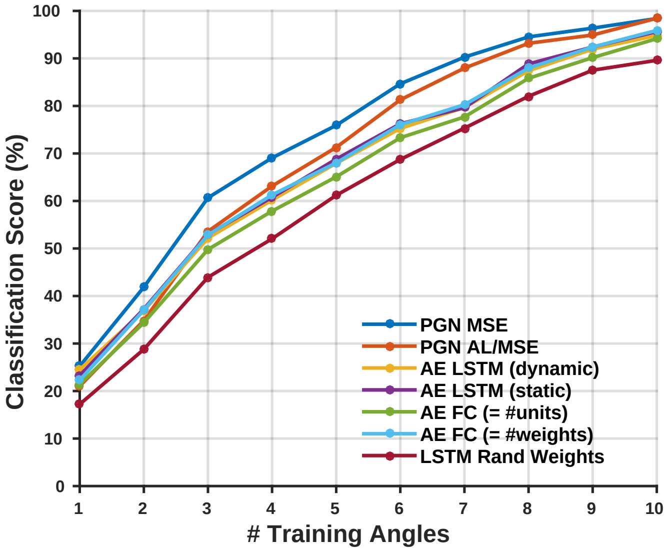

Figure 1.8: Classification accuracy on a 50-way face identification task. AE: Autoencoder.

The classification performance curves are shown in Figure 1.8. While all models show improvement compared to random initial weights, the predictive models outperform the controls. The MSE PGN has the highest score for each size of the training data. The AL/MSE PGN performs moderately worse, but still better than the control models. This is likely because, as previously mentioned, the AL/MSE model can tend to produce realistic faces, but are somewhat unfaithful to the underlying identity. While this is a relatively simple task compared to modern machine learning datasets, it provides a proof-of-principle that a model trained with a unsupervised, predictive loss on dynamic sequences can learn interesting structure, which is even useful for other tasks.

1.4 Conclusions

Above, we have demonstrated that an unsupervised, predictive loss can result in a rich internal representation of visual objects. Our CNN-LSTM-deCNN models trained with such a loss function successfully learn to predict future image frames in several contexts, ranging from the physics of simulated bouncing balls to the out-of-plane rotations of previously unseen computer-generated faces. However, importantly, models trained with a predictive unsupervised loss are also well-suited for tasks beyond the domain of video sequences. For instance, representations trained with a predictive loss outperform other models of comparable complexity in a supervised classification problem with static images. This effect is particularly pronounced in the regime where a classifier must operate from just a few example views of a new object (in this case, face). Taken together, these results support the idea that prediction can serve as a powerful framework for developing transformation-tolerant object representations of the sort needed to support one- or few-shot learning.

The experiments presented here are all done in the context of highly-simplified artificial worlds, where the underlying generative model of the stimuli is known, and where the number of degrees of freedom in the data set are few. We nonetheless argue that experiments with highly controlled stimuli hold the potential to yield powerful guiding insights. In the next chapter, we use these insights to develop a model that can indeed scale to natural videos.

2

PredNet: A Deep Predictive Coding Network

While our work in the previous chapter using an Encoder-Recurrent-Decoder framework was a promising illustration of predictive training, the basis for the architecture stemmed more from

Material contained in this chapter has been published in Lotter et al. (2017a).

the autoencoder machine learning literature and less from neuroscience. Compared to biological neural networks, currently our only example general artificial intelligence, the previous architecture is missing several ingredients. Most prominently, biological neural networks are heavily recurrent, containing abundant lateral and feedback connections (Lamme, 1998; Felleman & Van Essen, 1991). Our previous model contained only one recurrent layer, as opposed to biological visual hierarchies, where recurrence is present throughout. One very influential theory on a role for these lateral and feedback connections is the idea of "predictive coding".

Made particularly popular by works such as Rao & Ballard (1999) and Friston (2005), predictive coding suggests that the brain is constantly making hierarchical predictions of incoming stimuli. Feedback and lateral connections convey these predictions, and the residual error between the predictions and observations is passed on through feedforward connections. The propagated signal allows for the updating of hypotheses, leading to new predictions.

We have developed a deep learning implementation of predictive coding. Extending beyond a mechanism of efficient coding, we propose that predictive coding can be a powerful framework for representational learning. Trained to minimize prediction error, our model, which we informally call the "PredNet", is able to make accurate predictions in artificial and natural image sequences, and in doing so, learns a useful representation of higher level latent variables. Compared to our previous PGN model, the PredNet can make accurate predictions on much more complex sequences, even without depending on a difficult to train adversarial loss.

2.1 The PredNet Model

The PredNet architecture is diagrammed in Figure 2.1. The network consists of 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

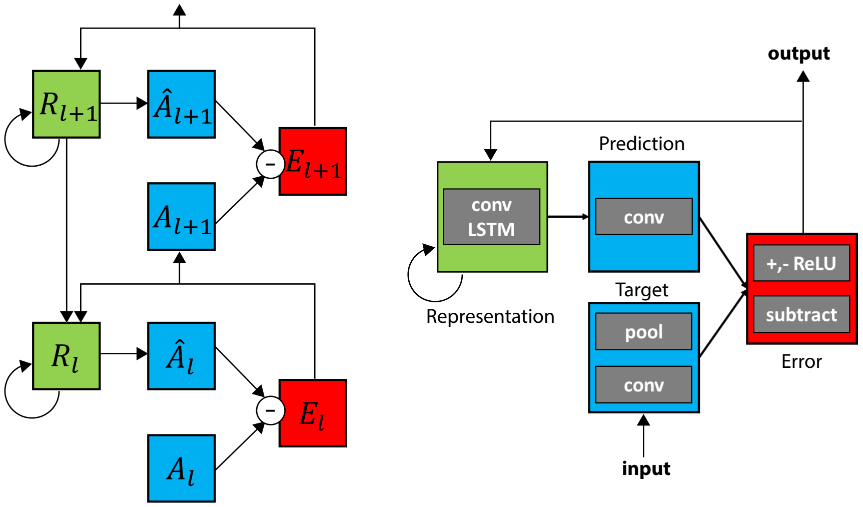

Figure 2.1: Predictive Coding Network (PredNet). Left: Illustration of information flow within two layers. Each layer consists of representation neurons (R_l), which output a layer-specific prediction at each time step ($\hat{A}_l$), which is compared against a target (A_l) (Bengio, 2014) to produce an error term (E_l), which is then propagated laterally and vertically in the network. Right: Module operations for case of video sequences.

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

(Rao & Ballard, 1999), 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. Our work also shares motivation with the Deep Predictive Coding Networks of Chalasani & Principe (2013); however, their framework is based upon sparse coding and a linear dynamical system with greedy layer-wise training, whereas ours is rooted in convolutional and recurrent neural networks trained with backprop.

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 lowest 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. For the representation neurons, we specifically use convolutional LSTM units (Hochreiter & Schmidhuber, 1997; Shi et al., 2015). 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 lowest 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 (Rao & Ballard, 1999), 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 (2.1) to (2.4). The model is trained to minimize the weighted sum of the activity of the error units. Explicitly, the training loss is formalized in Equation 2.5 with weighting factors by time, λ_t, and layer, λ_l, and where n_l is the number of units in the lth layer. With error units consisting of subtraction followed by ReLU activation, the loss at each layer is equivalent to an L1 error. Empirically, we find that this loss is sufficient for making compelling predictions with the current model, although incorporating a GAN adversarial objective could also be useful, which we leave for future work.

$$A_l^t = \begin{cases} x_t & \text{if } l = 0 \\ \textsc{MaxPool}(\textsc{ReLU}(\textsc{Conv}(E_{l-1}^t))) & l > 0 \end{cases} \tag{2.1}$$

$$\hat{A}_l^t = \textsc{ReLU}(\textsc{Conv}(R_l^t)) \tag{2.2}$$

$$E_l^t = [\textsc{ReLU}(A_l^t - \hat{A}_l^t); \textsc{ReLU}(\hat{A}_l^t - A_l^t)] \tag{2.3}$$

$$R_l^t = \textsc{ConvLSTM}(E_l^{t-1}, R_l^{t-1}, \textsc{Upsample}(R_{l+1}^t)) \tag{2.4}$$

$$L_{train} = \sum_t \lambda_t \sum_l \frac{\lambda_l}{n_l} \sum_{n_l} E_l^t \tag{2.5}$$

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

Algorithm 1 Calculation of PredNet states

```
Require: $x_t$
$A_0^t \leftarrow x_t$
$E_l^0, R_l^0 \leftarrow 0$
for $t = 1$ to $T$ do
    for $l = L$ to $0$ do                                        ▷ Update $R_l^t$ states
        if $l = L$ then
            $R_L^t = \textsc{ConvLSTM}(E_L^{t-1}, R_L^{t-1})$
        else
            $R_l^t = \textsc{ConvLSTM}(E_l^{t-1}, R_l^{t-1}, \textsc{Upsample}(R_{l+1}^t))$
    for $l = 0$ to $L$ do                          ▷ Update $\hat{A}_l^t, A_l^t, E_l^t$ states
        if $l = 0$ then
            $\hat{A}_0^t = \textsc{SatLU}(\textsc{ReLU}(\textsc{Conv}(R_0^t)))$
        else
            $\hat{A}_l^t = \textsc{ReLU}(\textsc{Conv}(R_l^t))$
        $E_l^t = [\textsc{ReLU}(A_l^t - \hat{A}_l^t); \textsc{ReLU}(\hat{A}_l^t - A_t^l)]$
        if $l < L$ then
            $A_{l+1}^t = \textsc{MaxPool}(\textsc{Conv}(E_t^l))$
```

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.

2.2 Experiments on Rendered Image Sequences

2.2.1 Prediction Performance

To gain an understanding of the representations learned in the proposed framework, we first trained PredNet models using synthetic images, similar to the PGN models. With the higher prediction capacity than the PGNS, we were able to use sequences of rendered faces rotating with two degrees of freedom, along the “pan” (out-of-plane) and “roll” (in-plane) axes. The faces start at a random orientation and rotate at a random constant velocity for a total of 10 frames. A different face was sam-

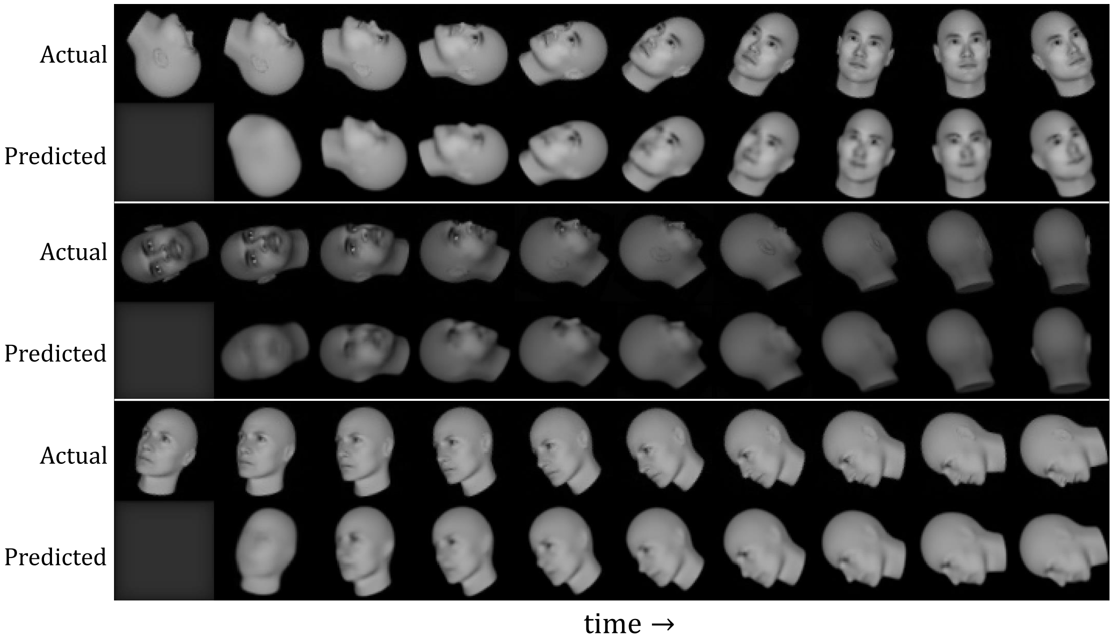

Figure 2.2: PredNet next-frame predictions for sequences of rendered faces rotating with two degrees of freedom. Faces shown were not seen during training.

pled for each sequence. The images were processed to be grayscale, with values normalized between 0 and 1, and 64x64 pixels in size. We used 16K sequences for training and 800 for both validation and testing.

Predictions generated by a PredNet model are shown in Figure 2.2. The 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 adapts to the underlying dynamics to generate predictions that closely match the incoming frame.

For choosing the hyperparameters of the model, we performed a random search and chose the model that had the lowest L1 error in frame prediction averaged over time steps 2-10 on a validation set. Given this selection criteria, the best performing models tended to have a loss solely concentrated at the lowest layer (i.e. $\lambda_0 = 1, \lambda_{l>0} = 0$), which is the case for the model shown. Using an equal loss at each layer considerably degraded predictions, but enforcing a moderate loss on upper layers that was one magnitude smaller than the lowest layer (i.e. $\lambda_0 = 1, \lambda_{l>0} = 0.1$) led to only slightly worse predictions, as illustrated in Figure 2.3. In all cases, the time loss weight, λ_t, was set to zero for the first time step and then one for all time steps after. As for the remaining hyperparameters, the model shown has 5 layers with 3x3 filter sizes for all convolutions, max-pooling of stride 2, and number of channels per layer, for both A_l and R_l units, of $(1, 32, 64, 128, 256)$. Model weights were optimized using the Adam algorithm (Kingma & Ba, 2014).

Quantitative evaluation of generative models is a difficult, unsolved problem (Theis et al., 2016), but here we report prediction error in terms of mean-squared error (MSE) and the Structural Similarity Index Measure (SSIM) (Wang et al., 2004). 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 a CNN encoder that has lateral skip connections to a convolutional LSTM decoder. The performance of all models on the rotating faces dataset is summarized in Table 2.1, where the scores were calculated as an average over all predictions after the first frame. We report results for the PredNet model trained with loss only on the lowest layer, denoted as PredNet L_0, as well as the model trained with an 0.1 weight on upper layers, de-

Table 2.1: Evaluation of next-frame predictions on Rotating Faces Dataset (test set).

	MSE	SSIM
PredNet L_0	0.0152	0.937
PredNet L_{all}	0.0157	0.921
CNN-LSTM Enc.-Dec.	0.0180	0.907
Copy Last Frame	0.125	0.631

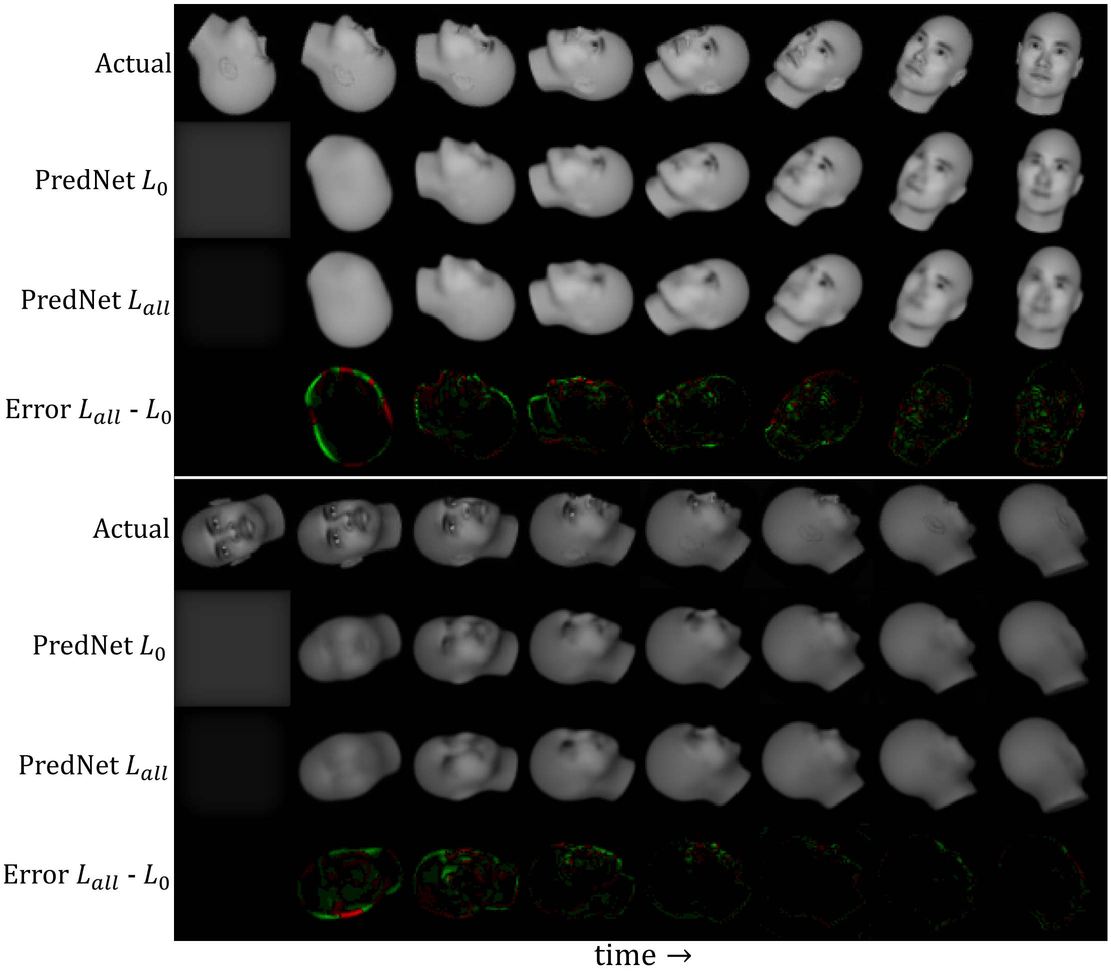

Figure 2.3: Next-frame predictions of PredNet L_{all} model on the rotating faces dataset and comparison to L_0 version. The "Error $L_{all} - L_0$" visualization shows where the pixel error was smaller for the L_0 model than the L_{all} model. Green regions correspond to where L_0 was better and red corresponds to where L_{all} was better.

noted as PredNet L_{all}. Both PredNet models outperformed the baselines on both measures, with the L_0 model slightly outperforming L_{all}, as expected for evaluating the pixel-level predictions.

2.2.2 Exploring Representation Learned

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

(Kulkarni et al., 2015). The rotating faces were generated using the FaceGen software package (Singular Inversions, Inc.), 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 trained models, we decoded the latent parameters from the representation neurons (R_l) in different layers, 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 (Pinto et al., 2009; Jarrett et al., 2009; Saxe et al., 2010).

Latent variable decoding accuracies of the pan and roll velocities, pan initial angle, and first PC are shown in the left panel of Figure 2.4. There are several interesting patterns. First, the trained models learn a representation that generally permits a better linear decoding of the underlying latent factors than the randomly initialized model, with the most striking difference in terms of the the pan rotation speed (α_{pan}). Second, the most notable difference between the L_{all} and L_0 versions occurs with the first principle component, where the model trained with loss on all layers has a higher decoding accuracy than the model trained with loss only on the lowest layer.

The latent variable decoding analysis suggests that the model learns a representation that may generalize well to other tasks for which it was not explicitly trained. To investigate this further, we assessed the models in a classification task from single, static images. We created a dataset of 25 previously unseen FaceGen faces at 7 pan angles, equally spaced between $[-\frac{\Pi}{2}, \frac{\Pi}{2}]$, and 8 roll angles,

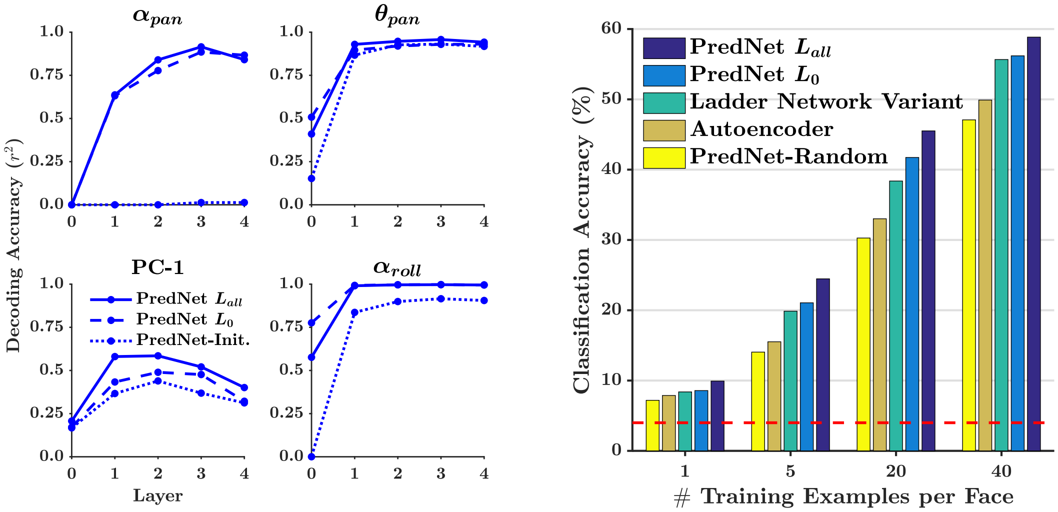

Figure 2.4: Information contained in PredNet representation for rotating faces sequences. Left: Decoding of latent variables using a ridge regression (α_{pan}: pan (out-of-frame) angular velocity, θ_{pan}: pan angle, PC-1: first principal component of face, α_{roll}: roll (in-frame) angular velocity). Right: Orientation-invariant classification of static faces.

equally spaced between $[0, 2\pi)$. There were therefore $7 \cdot 8 = 56$ orientations per identity, 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 PredNets to a standard autoencoder and a variant of the Ladder Network (Valpola, 2015; Rasmus et al., 2015). Both models were constructed to have the same number of layers and channel sizes as the PredNets, as well as a similar alternating convolution/max-pooling, then upsampling/convolution scheme. As both networks are autoencoders, they were trained with a reconstruction loss, with a dataset consisting of all of the individual frames from the sequences used to train the PredNets. 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 to 0.5 in increments of 0.1, with loss weights either evenly distributed across layers, solely concentrated at the pixel layer, or 1 at the bottom layer and 0.1 at upper layers (analogous to the PredNet L_{all} model). 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 (Pezeshki et al., 2015), 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. For the PredNets, the representation units were used and features were extracted after processing one input frame.

Face classification accuracies using the representations learned by the L_0 and L_{all} PredNets, a standard autoencoder, and a Ladder Network variant are shown in the right panel of Figure 2.4. Both PredNets compare favorably to the other models at all sizes of the training set, suggesting they learn a representation that is relatively tolerant to object transformations. Similar to the decoding accuracy of the first principle component, the PredNet L_{all} model actually outperformed the L_0 variant. Altogether, these results suggest that predictive training with the PredNet can be a viable alternative to other models trained with a more traditional reconstructive or denoising loss, and that the relative layer loss weightings (λ_l's) may be important for the particular task at hand.

2.3 Experiments on Natural Image Sequences

2.3.1 Prediction Performance

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 (Agrawal et al., 2015). Models were trained using the raw videos from the KITTI dataset (Geiger et al., 2013), which were 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 channel sizes of $(3, 48, 96, 192)$. Models were again trained with Adam (Kingma & Ba, 2014) using a loss either solely computed on the lowest layer (L_0) or with a weight of 1 on the lowest layer and 0.1 on the upper layers (L_{all}). 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 (Dollár et al., 2009), 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.

Sample PredNet predictions (for the L_0 model) on the CalTech Pedestrian dataset are shown in Figure 2.5, and example videos can be found at `https://coxlab.github.io/prednet/`. The model is able to make fairly accurate predictions in a wide range of scenarios. In the top sequence of Fig. 2.5, 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

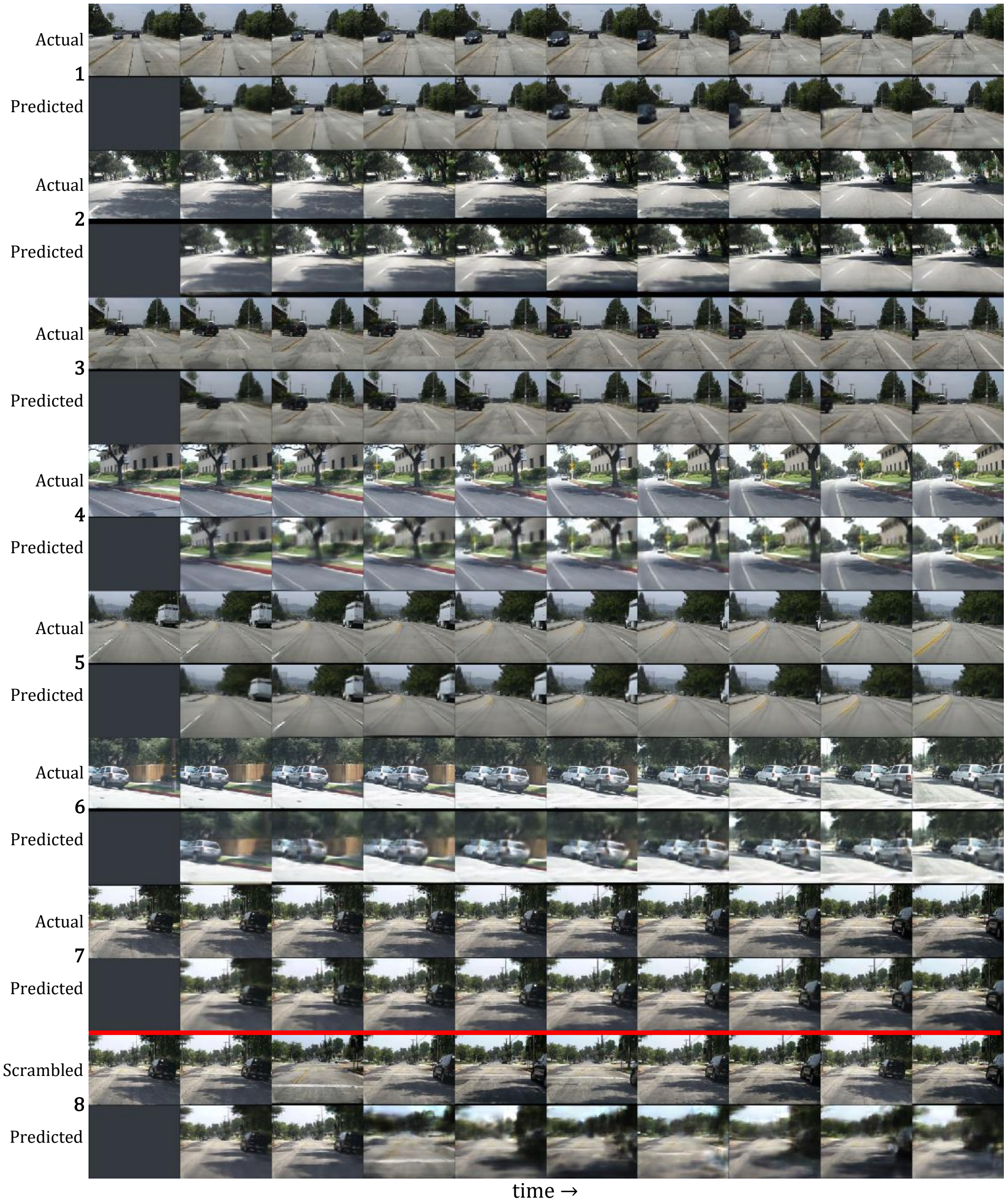

Figure 2.5: PredNet predictions for car-cam videos. The first rows contain ground truth and the second rows contain predictions. The sequence below the red line was temporally scrambled. The model was trained on the KITTI dataset and sequences shown are from the CalTech Pedestrian dataset.

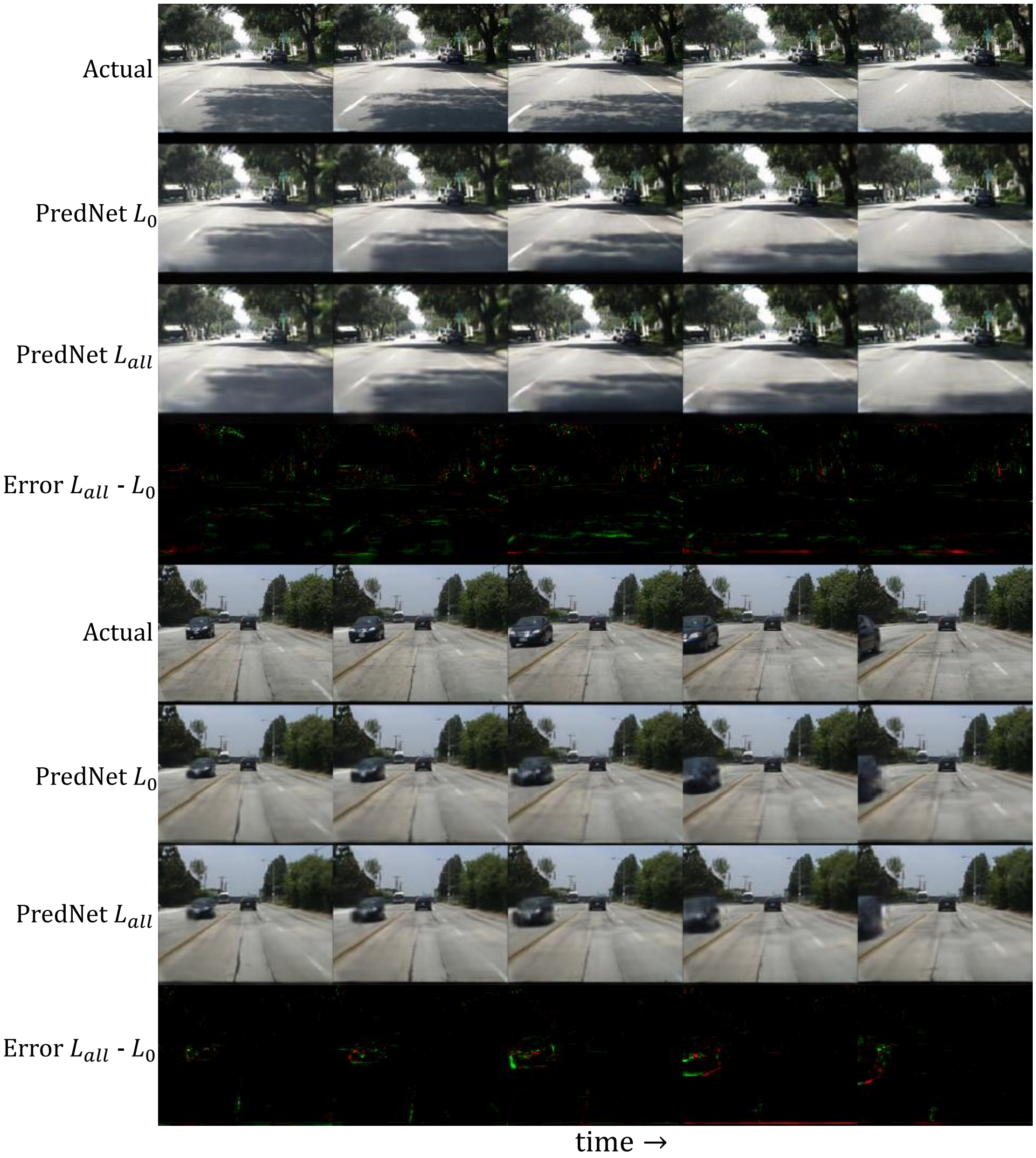

Figure 2.6: Next-frame predictions of PredNet L_{all} model on the CalTech Pedestrian dataset and comparison to L_0 version. The ”Error $L_{all} - L_0$” visualization shows where the pixel error was smaller for the L_0 model than the L_{all} model. Green regions correspond to where L_0 was better and red corresponds to where L_{all} was better.

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. As an additional control, we show a sequence at the bottom of Fig. 2.5, where the input has been temporally scrambled. In this case, the model generates blurry frames, which mostly just resemble the previous frame. Finally, a comparison between predictions by the L_0 and L_{all} model are shown in Fig. 2.6. At first glance, the difference in predictions between the models seem fairly minor, but upon careful inspection, however, it is apparent that the L_{all} predictions lack some of the finer details of the L_0 predictions and are more blurry in regions of high variance.

Table 2.2: Evaluation of Next-Frame Predictions on CalTech Pedestrian Dataset.

	MSE	SSIM
PredNet L_0	$\mathbf{3.13 \times 10^{-3}}$	0.884
PredNet L_{all}	3.33×10^{-3}	0.875
CNN-LSTM Enc.-Dec.	3.67×10^{-3}	0.865
Copy Last Frame	7.95×10^{-3}	0.762

Quantitatively, the PredNet models again outperformed the CNN-LSTM Encoder-Decoder. 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 L_0 had the best performance, with an average error reduction of 14.7% and 14.9% for MSE and SSIM, respectively, compared to the CNN-LSTM Encoder-Decoder.

For a more thorough investigation of the difference in performance between the PredNet and the CNN-LSTM Encoder-Decoder, we performed an ablation study with results in Table 2.3. We evaluate the models in terms of pixel prediction, thus using the PredNet model trained with loss only on the lowest layer (PredNet L_0) as the base model. In addition to mean-squared error (MSE) and the Structural Similarity Index Measure (SSIM), we include calculations of the Peak Signal-To-Noise Ratio (PSNR). For each model, we evaluate it with the original set of hyperparameters (controlling the number of layers, filter sizes, and number of filters per layer), as well as with the four additional sets of hyperparameters that were randomly sampled from the initial random search. Below is an explanation of the additional control models:

- PredNet (no E split): PredNet model except the error responses (E_l) are simply linear ($\hat{A}_l - A_l$) instead of being split into positive and negative rectifications.
- CNN-LSTM Enc.-Dec. (2x A_l filts): CNN-LSTM Encoder-Decoder model (A_l's are passed instead of E_l's) except the number of filters in A_l is doubled. This controls for the total number of filters in the model compared to the PredNet, since the PredNet has filters to produce $\hat{A}_l$ at each layer, which is integrated into the model's feedforward response.
- CNN-LSTM Enc.-Dec. (except pass E_0): CNN-LSTM Encoder-Decoder model except the error is passed at the lowest layer. All remaining layers pass the activations A_l. With training loss taken at only the lowest layer, this variation allows us to determine if the "prediction" subtraction operation in upper layers, which is essentially unconstrained and learnable in the L_0 case, aids in the model's performance.
- CNN-LSTM Enc.-Dec. (+/- split): CNN-LSTM Encoder-Decoder model except the activations A_l are split into positive and negative populations before being passed to other layers in the network. This isolates the effect of the additional nonlinearity introduced by this procedure.

Equalizing the number of filters in the CNN-LSTM Encoder-Decoder (2x A_l filts) cannot account for its performance difference with the PredNet, and actually leads to overfitting and a decrease in performance. Passing the error at the lowest layer (E_0) in the CNN-LSTM Enc.-Dec. improves performance, but still does not match the PredNet, where errors are passed at all layers. Finally, splitting the activations A_l into positive and negative populations in the CNN-LSTM Enc.-Dec. does not help, but the PredNet with linear error activation ("no E_l split") performs slightly

Table 2.3: Quantitative evaluation of additional controls for next-frame prediction in CalTech Pedestrian Dataset after training on KITTI. First number indicates score with original hyperparameters. Number in parenthesis indicates score averaged over total of five different hyperparameters.

	MSE (x 10^{-3})	PSNR	SSIM
PredNet	3.13 (3.33)	25.8 (25.5)	0.884 (0.878)
PredNet (no E_l split)	3.20 (3.37)	25.6 (25.4)	0.883 (0.878)
CNN-LSTM Enc.-Dec.	3.67 (3.91)	25.0 (24.6)	0.865 (0.856)
CNN-LSTM Enc.-Dec. (2x A_l filts)	3.82 (3.97)	24.8 (24.6)	0.857 (0.853)
CNN-LSTM Enc.-Dec. (except pass E_0)	3.41 (3.61)	25.4 (25.1)	0.873 (0.866)
CNN-LSTM Enc.-Dec. (+/- split)	3.71 (3.84)	24.9 (24.7)	0.861 (0.857)
Copy Last Frame	7.95	20.0	0.762

worse than the original split version. Together, these results suggest that the PredNet's error passing operation can lead to improvements in next-frame prediction performance.

While the previous controls and comparisons between the PredNet and the CNN-LSTM Enc.-Dec. isolate the effects of the more unique components in the PredNet, we also directly compared against other published models. We report results on a 64x64 pixel, grayscale car-cam dataset and the Human3.6M dataset (Ionescu et al., 2014) to compare against the two concurrently developed models by Brabandere et al. (2016) and Finn et al. (2016), respectively. For both comparisons, we use a model with the same hyperparameters (# of layers, # of filters, etc.) of the PredNet L_0 model trained on KITTI, but train from scratch on the new datasets. The only modification we make is to train using an L2 loss instead of the effective L1 loss, since both models train with an L2 loss and report results using L2-based metrics (MSE for Brabandere et al. (2016) and PSNR for Finn et al. (2016)). That is, we keep the original PredNet model intact but directly optimize using MSE between actual and predicted frames. We measure next-frame prediction performance after inputting 3 frames and 10 frames, respectively, for the 64x64 car-cam and Human3.6M datasets, to be consistent with the published works. We also include the results using a feedforward multi-scale network, similar to the model of Mathieu et al. (2016), on Human3.6M, as reported by Finn et al. (2016).

Table 2.4: Evaluation of Next-Frame Predictions on 64x64 Car-Cam Dataset.

	MSE (per-pixel)
DFN (Brabandere et al., 2016)	1.71×10^{-3}
PredNet	$\mathbf{1.16 \times 10^{-3}}$
Copy Last Frame	3.58×10^{-3}

Table 2.5: Evaluation of Next-Frame Predictions on Human3.6M

	PSNR
DNA (Finn et al., 2016)	42.1
PredNet	38.9
FF multi-scale (Mathieu et al., 2016)	26.7
Copy Last Frame	32.0

2.3.2 Multi-Step Prediction

The results for the PredNet so far have focused on one-step-ahead prediction, but the model can be made to predict multiple frames by treating predictions as actual input and recursively iterating. Examples of this process are shown in Figure 2.7 for the PredNet L_0 model. Although the next frame predictions are reasonably accurate, the model naturally breaks down when extrapolating further into the future. This is not surprising since the predictions will unavoidably have different statistics than the natural images for which the model was trained to handle (Bengio et al., 2015). 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. For the first 10 time steps, the training loss was calculated on the E_l activations as usual, and for the last 5, it was calculated directly as the mean absolute error with respect to the ground truth frames. Despite eventual blurriness (which might be expected to some extent due to uncertainty), the fine-tuned 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

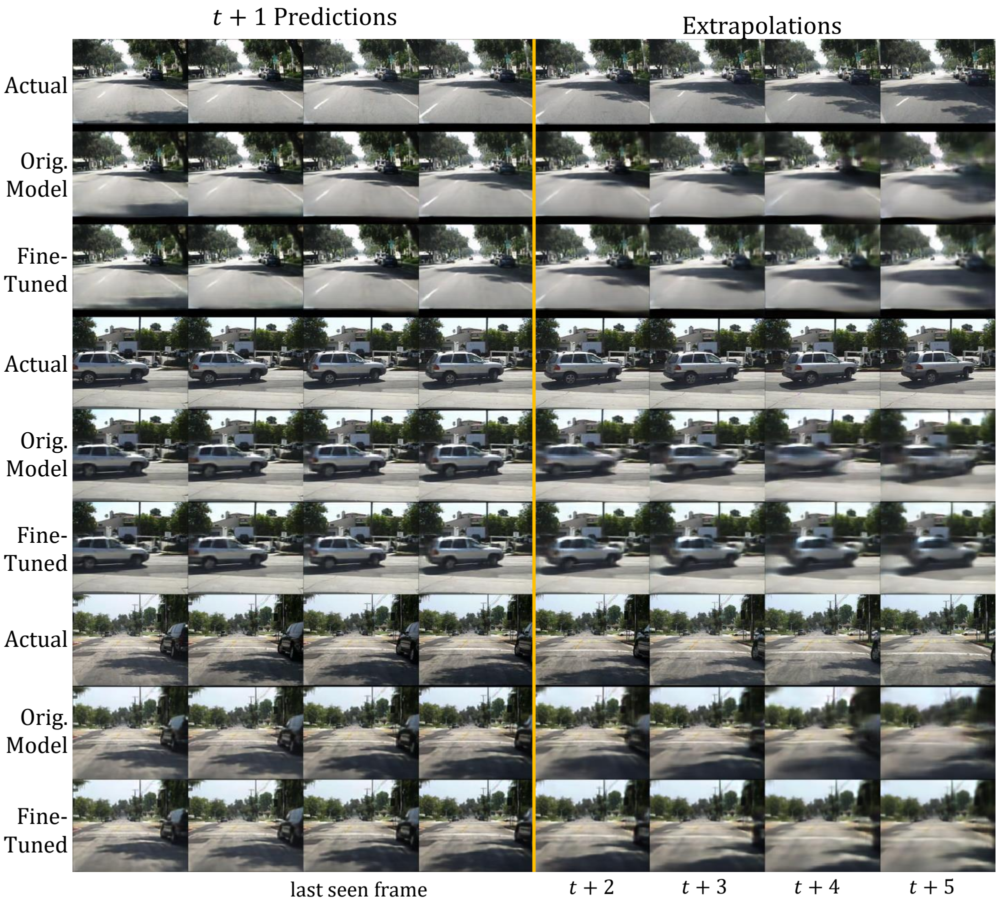

Figure 2.7: Extrapolation sequences generated by feeding PredNet predictions back into model. Left of the orange line: Normal $t+1$ predictions; Right: Generated by recursively using the predictions as input. First row: Ground truth sequences. Second row: Generated frames of the original model, trained to solely predict $t + 1$. Third row: Model fine-tuned for extrapolation.

model is able to extrapolate the motion of a car moving to the right. The reader is again encouraged to visit `https://coxlab.github.io/prednet/` to view the predictions in video form. Quantitatively, the MSE of the model's predictions stay well below the trivial solution of copying the last seen frame, as illustrated in Fig 2.8. The MSE increases fairly linearly from time steps 2-10, even though the model was only trained for up to $t + 5$ prediction.

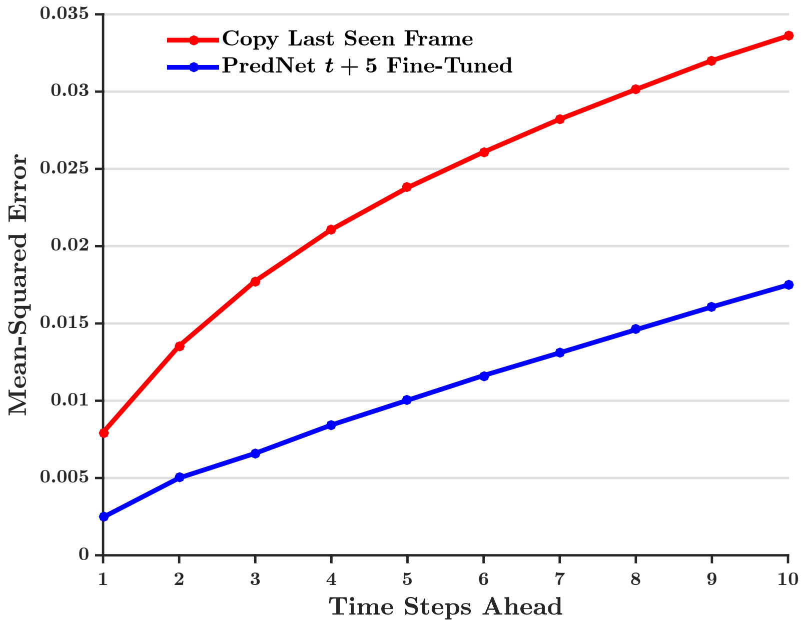

Figure 2.8: MSE of PredNet predictions as a function of number of time steps ahead predicted. Model was fine-tuned for up to $t + 5$ prediction.

2.3.3 Exploring Representation Learned

To test the implicit encoding of latent parameters in the car-cam setting, we used the internal representation in the PredNet to estimate the car's steering angle (Bojarski et al., 2016; Biasini et al., 2016). We used a dataset released by Comma.ai (Biasini et al., 2016) consisting of 11 videos totaling about 7 hours of mostly highway driving. We first trained networks for next-frame prediction and then fit a linear fully-connected layer (e.g. a linear regression) on the learned representation to estimate the steering angle, using a MSE loss. We again concatenate the R_l representation at all layers, but first spatially average pool lower layers to match the spatial size of the upper layer, in order to reduce dimensionality. Steering angle estimation results, using the representation on the 10^{th} time step, are shown in Figure 2.9. Given just 1K labeled training examples, a simple linear readout on the PredNet L_0 representation explains 74% of the variance in the steering angle and outperforms the

CNN-LSTM Enc.-Dec. by 35%. With 25K labeled training examples, the PredNet L_0 has a MSE (in *degrees*^{2}) of 2.14. Interestingly, in this task, the PredNet L_{all} model actually underperformed the L_0 model and slightly underperformed the CNN-LSTM Enc.-Dec, again suggesting that the λ_l parameter can affect the representation learned, and different values may be preferable in different end tasks. Nonetheless, the readout from the L_{all} model still explained a substantial proportion of the steering angle variance and strongly outperformed the random initial weights.

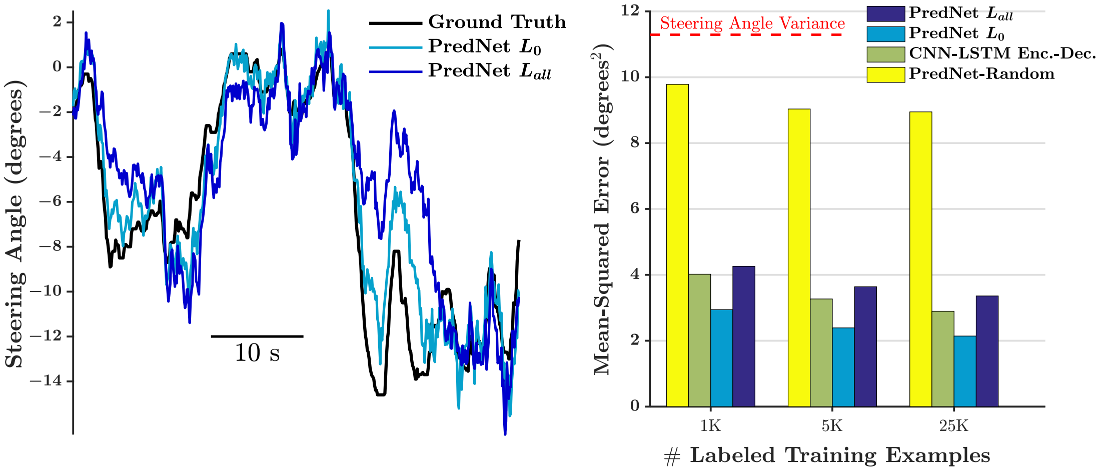

Figure 2.9: Steering angle estimation accuracy on the Comma.ai dataset (Biasini et al., 2016). Left: Example steering angle curve with model estimations for a segment in the test set. Decoding was performed using a fully-connected readout on the PredNet representation trained with 25K labeled training examples. PredNet representation was trained for next-frame prediction on Comma.ai training set. Right: Mean-squared error of steering angle estimation.

In Figure 2.10, we show the steering angle estimation accuracy on the Comma.ai (Biasini et al., 2016) dataset using the representation learned by the PredNet L_0 model, as a function of the number of frames inputted into the model. The PredNet's representation at all layers was concatenated (after spatially pooling lower layers to a common spatial resolution) and a fully-connected readout was fit using MSE. For each level of the number of training examples, we average over 10 cross-validation splits. To serve as points of reference, we include results for two static models. The first model is an autoencoder trained on single frame reconstruction with appropriately matching hyperparameters. A fully-connected layer was fit on the autoencoder's representation to estimate the steering angle in the same fashion as the PredNet. The second model is the default model in the posted Comma.ai code (Biasini et al., 2016), which is a five layer CNN. This model is trained end-to-end to estimate the steering angle given the current frame as input, with a MSE loss. In addition to 25K examples, we trained a version using all of the frames in the Comma dataset (~396K). For all models, the final weights were chosen at the minimum validation error during training. Given the relatively small number of videos in the dataset compared to the average duration of each video, we used 5% of each video for validation and testing, chosen as a random continuous chunk, and discarded the 10 frames before and after the chosen segments from the training set.

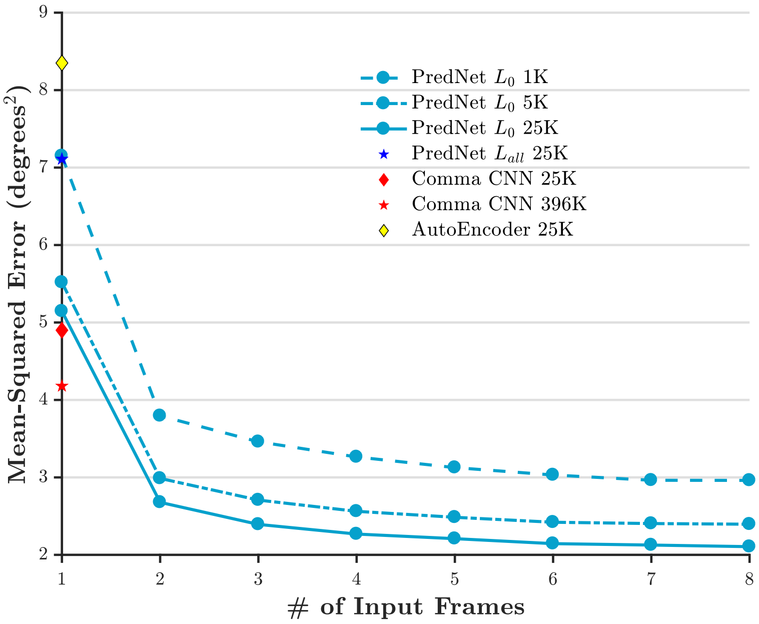

Figure 2.10: Steering angle estimation accuracy as a function of the number of input frames.

As illustrated in Figure 2.10, the PredNet's performance gets better over time, as one might expect, as the model is able to accumulate more information. Interestingly, it performs reasonably well after just one time step, in a regime that is orthogonal to the training procedure of the PredNet where there are no dynamics. Altogether, these results again point to the usefulness of the model in learning underlying latent parameters.

2.4 Conclusions

Motivated by ideas of predictive coding, we have presented a model that is able to predict future frames in both synthetic and natural image sequences. Scaling up the experiments from the previous chapter, we have provided more evidence that learning to predict how an object or scene will move in a future frame confers advantages in decoding latent parameters that give rise to an object's appearance, and can improve recognition performance. In particular, we have demonstrated that passing errors can be beneficial for both prediction performance and representation learning.

3

Reproducing Neural Phenomena using the PredNet Model

In the previous chapter, we presented our PredNet model, a deep predictive coding network that is trained for next-frame video prediction. The model is inspired by the seminal work of Rao & Ballard (1999). They propose a hierarchical model designed with the principles of predictive coding, where higher layers are trained to predict the responses of lower layers, and the lower layers pass forward the residual deviations from these predictions. One of the major findings that they report, when training their model on natural images, is the emergence of extra-classical receptive field effects. For instance, some error units in their model exhibited patterns of end-stopping, resembling neurons commonly found in early visual cortex (Hubel & Wiesel, 1965, 1968). These units, which had developed orientation selectivity, preferentially responded to bars of certain length, with a decrease in firing rate as the bar extended beyond the classical receptive field. Their proposed explanation is that, in natural scenes, short line segments are abnormal. Instead, lines tend to be continuous, and the inner segments are predictable by the surrounding segments. Thus, an end-stopping neuron can be seen as a residual error detector, signaling a short segment as a deviation from expected statistics.

In addition to extra-classical receptive field effects, predictive coding models have been used to explain a variety of neural phenomena, such as spatial and temporal receptive fields in the retina (Srinivasan et al., 1982; Atick, 1992; Hosoya et al., 2005), tuning to optical flow in medial superior temporal area cells (Jehee et al., 2006), and even correlates of visual attention (Rao, 1998a). We sought to reproduce some of these findings in our model, as well provide new examples that our model can explain. For the experiments, we use the PredNet (L_{all}) model trained on natural videos (the KITTI car-mounted camera dataset (Geiger et al., 2013)), unless otherwise noted.

3.1 Surround Suppression

We began by testing for surround suppression effects in the zeroth layer of the model, analogous to an LGN layer. Surround suppression refers generally to a decrease in response caused by stimuli extending beyond the classical receptive field, and has been observed in various visual stages, such as the retina (Solomon et al., 2006; Nolt et al., 2004), LGN (Solomon et al., 2002; Fisher et al., 2016), and V1 (Jones et al., 2001; Sceniak et al., 1999). We presented centered squares of different sizes on a gray background and measured the response of the E_0 units with corresponding receptive fields. The red curve in Figure 3.1 shows the average response of the E_0 units as a function of block width. As the error units are split into positive and negative errors by construction, the units respond to either lighter or darker central contrasts, so we present both white and black squares and take the maximum response for each unit. We again initially quantify the responses as a "rate code", totaling the activation over the first 10 timesteps. Evident in the red curve, the model exhibits surround suppression. The maximum average response occurs at a block width of 1 pixel, and then decreases sharply as the width increases. Interestingly, if the feedback from the layer above is"cooled" (connections from R_1 to R_0 are set to zero), the surround suppression effect is strongly diminished. These results are qualitatively very similar to the findings of Nassi et al. (2014), where they noticed a decrease in surround suppression effects in macaque V1 when feedback from V2 was cooled.

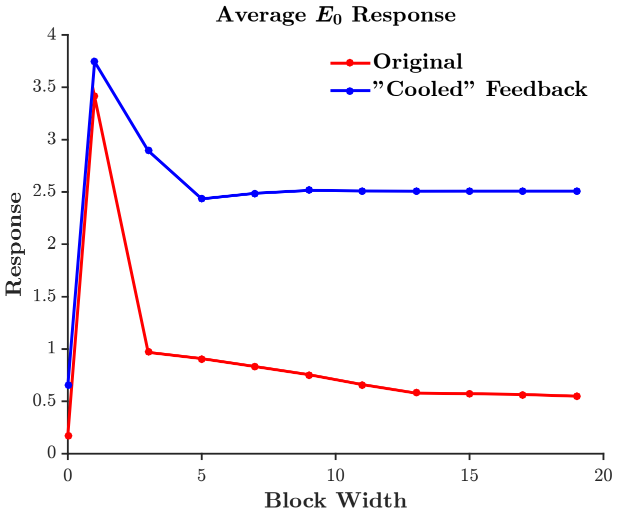

Figure 3.1: Response of E_0 units presented with a square stimulus of different widths. Surround suppression is evident, which is diminished when top-down feedback is "cooled" to zero.

Figure 3.2 displays the temporal response curves for the surround suppression experiment. As the classical receptive field of the E_0 units has a size of 1 by construction, the original and "cooled" models both show the same magnitude in response for the first timestep for widths $>=\ 1$. In the original model, the response drops rapidly for large stimulus widths, but only slowly decays for the width of one pixel. When the feedback is cooled, the decay in activation is largely diminished. Thus, in the original model, the feedback connections help convey a prediction of the central pixel, which is influenced by the surround. Since the model is trained on natural stimuli, we can interpret the strength of the feedforward residual response as the amount of deviation from natural statistics.

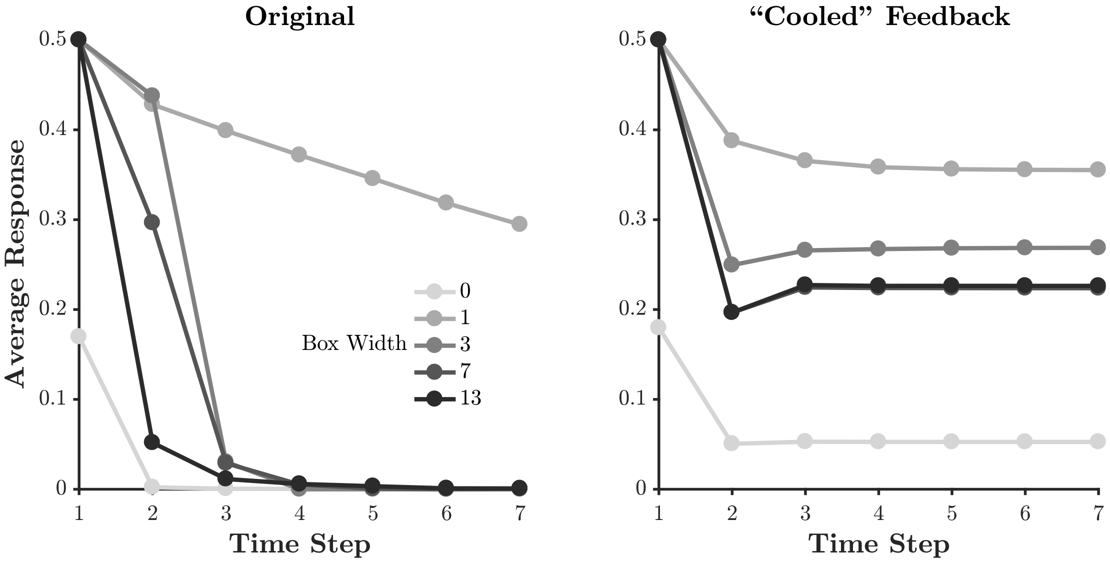

Figure 3.2: Temporal response curves for th surround suppression experiment. Left: Original PredNet model. Right: Prednet model with feedback from R_1 to R_0 removed. In the original model, the responses to stimuli with widths larger than 1 pixel decays sharply, whereas in the perturbed model, the responses are sustained.

3.2 End-Stopping

To investigate end-stopping in the network, we first present static gabor stimuli to find the optimal orientation for each unit in the E_1 layer. For units that have sufficient orientation tuning (using a threshold of 0.8 in circular variance (Ringach et al., 2002)), we then present static bars at the optimal orientation and record the response for bars of different length. We again quantify the response in terms of a "rate code". Within the population, there are many neurons that exhibit end-stopping.

Two example units are shown in Fig. 3.3. In fact, if we average the normalized response over all units, this aggregate curve also has a end-stopping pattern. Thus, our model aligns with the model of Rao & Ballard (1999) in suggesting that end-stopping behavior can be interpreted as an error signal.

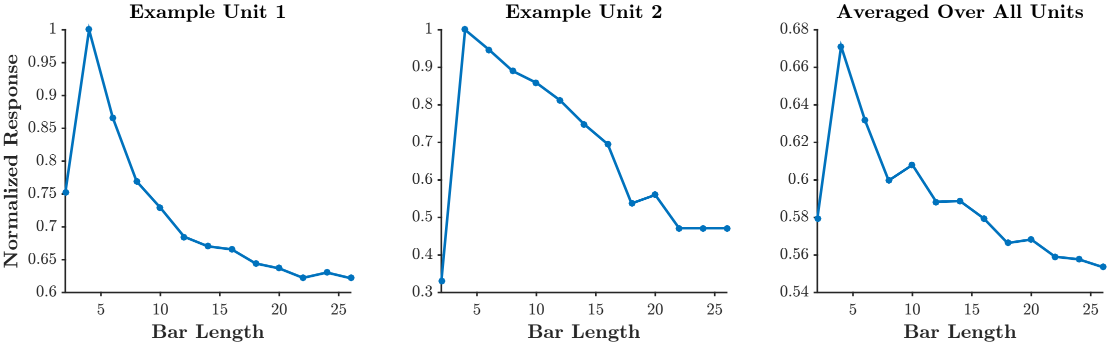

Figure 3.3: End-stopping behavior in the E_1 layer. The left two subpanels contain example units, and the right panel is averaged over all orientation-selective units in the E_1 layer. The average is taken over normalized responses, where the tuning curve is normalized to have a peak of 1.

Figure 3.4 demonstrates the effect of cooling on end-stopping, again using the average normalized response for Gabor-selective units in each condition. Interestingly, cooling only the feedback from R_2 to R_1 above doesn't completely diminish the end-stopping effect. However, cooling the lateral recurrent connections (the R_0 to R_0 recurrence) in addition to the feedback does diminish the effect.

Another interesting aspect to note, is that surround suppression and end-stopping are essentially spatial predictive coding effects (Huang & Rao, 2011). Our networks were trained with an emphasis on temporal predictive coding, with the goal of predicting the next input frame. This is necessarily intertwined with spatial predictive coding, however, as the world tends to evolve slowly over time, with a static input as an extreme case. Nevertheless, it is notable that our results align with the work of Rao & Ballard (1999), where their model was trained on static stimuli.

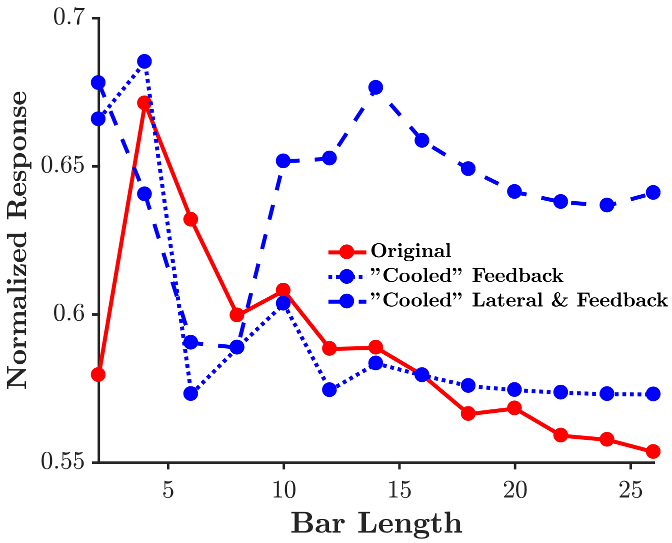

Figure 3.4: Effects of "cooling" on end-stopping. Removing both feedback (from R_2 to R_1) and lateral recurrence (from R_1 to R_1) is necessary to abolish the effect.

3.3 Norm-Based Coding of Faces

Beyond simple images like bars and squares, we're interested in exploring error-like mechanisms with complex stimuli. One interesting theory that could be consistent with a predictive coding framework is the norm-based coding of faces (Rhodes & Jeffery, 2006; Leopold, 2017). For instance, in presenting images of conspecific faces to macaques, Leopold (2017) found that faces which deviated more from the average face generally elicited stronger responses in IT cortex. They demonstrated this consistently by morphing faces along various axes. One could imagine that more caricature-like features deviate from overall expectations developed from natural exposure during development, and so the larger responses could be interpreted as error signals.

We tested the norm-based coding of faces using our model trained on FaceGen-generated faces (Singular Inversions, Inc.). The generative model employed in this software is a principal component analysis (PCA) using a corpus of natural faces. Sampling a random face consists of sampling 130 principal components with zero mean and unit variance. Our model was trained on faces generated randomly in this fashion. To test the norm-based coding effect, we generated an additional 200 faces, morphing the distance away from the normal face in different levels. An example is shown in Fig. 3.5, where a morph level of 1 corresponds to having a value of magnitude 1 for all principal components. Negative morph levels correspond to morphing in the opposite direction. To quantify responses to different morph levels, we averaged the responses over all error units in each layer (again expressed as a rate code), for each of the 200 faces. As demonstrated in Fig. 3.5, the minimum response occurs at the mean face (morph level 0) for all error layers of the network. The responses increase monotonically as the magnitude of the morph level is increased. These results are consistent with a norm-based coding theory of faces.

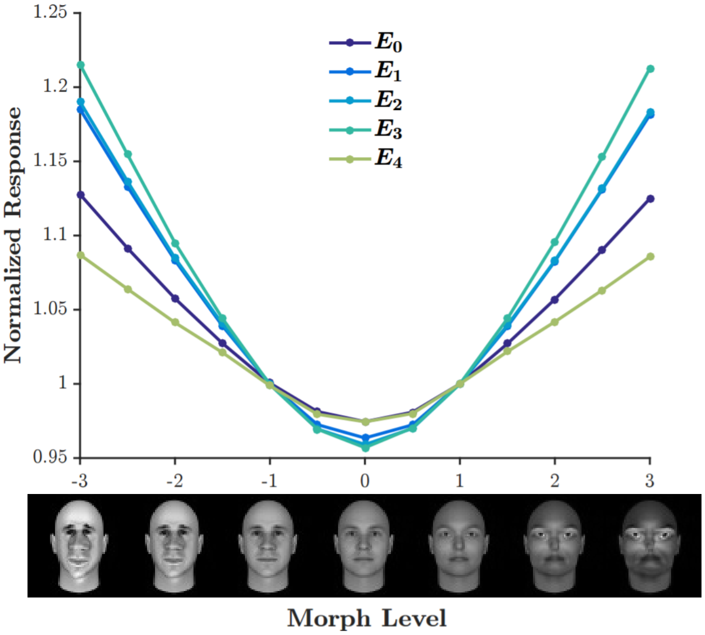

Figure 3.5: Average response in different layers in the model over different levels of face morphing. An example morph sequence is shown. Each curve is normalized such that the response is 1 at a morph level of 1.

In the work of Leopold (2017), the higher responses to abnormal faces manifested in slower decaying temporal responses. That is, the initial response was similar for different faces, but abnormal faces resulted in longer sustained responses. In testing this in our model, we have mixed results, as illustrated in Fig. 3.6. For instance, in the E_0 layer, the initial feedforward response is largely consistent across different morph levels, and the difference for abnormal faces is evident at later timesteps. However, this pattern is not true for all layers, for example, the uppermost layer actually exhibits a larger response on the initial timestep for the abnormal faces.

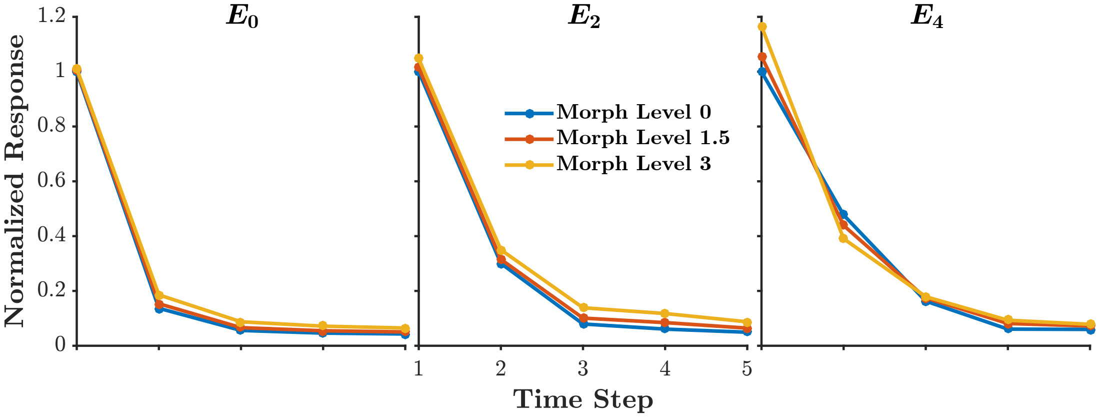

Figure 3.6: Average temporal response to faces at different morph levels for several layers in the PredNet model. The temporal pattern of responses varies across layers, although the total summed response is greater for higher morph levels (as shown in Fig. 3.5.

3.4 Flash-Lag Effect

One of the most compelling examples of predictive sensory processing is the flash-lag illusion (Mackay, 1958; Nijhawan, 1994). An example version of this illusion consists of a central bar rotating at constant speed, and a peripheral bar that periodically flashes on the screen (e.g. `http://www.michaelbach.de/ot/mot-flashLag/`). In the pixel space, the two bars are perfectly collinear when presented, however, the standard perception is that the flashed line lags the rotating line and is a different angle. Although there are other proposed explanations (Eagleman & Sejnowski, 2000), one potential explanation is that the visual system attempts to account for sensory processing delays through predictive mechanisms.

Fig. 3.7 contains an example prediction from our model when presented with a flash-lag stimulus. The video is created such that the inner bar rotates at 6° per frame and the outer bar appears every 8 frames. The displayed image sequence is taken after 95 frames, where the model has essentially reached a steady state. Notably, the predicted frame after the flashed bar resembles the typical perception of the illusion. That is, the outer bar in the prediction is behind and at a different angle than the inner bar.

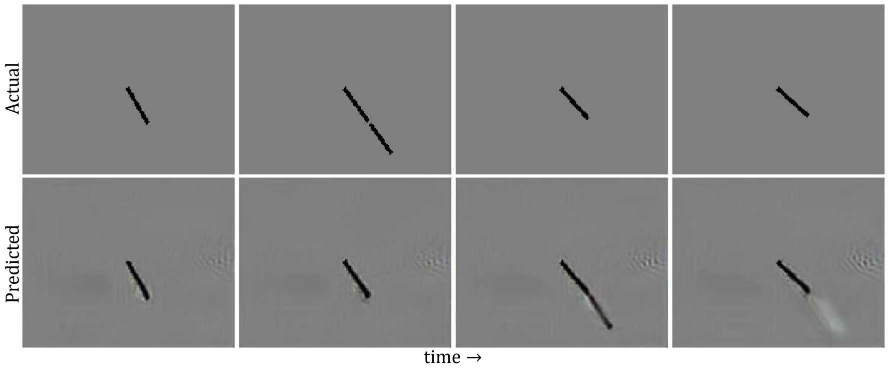

Figure 3.7: PredNet model predictions under the Flash-Lag Illusion.

While the neural phenomena previously presented rely specifically on a predictive coding framework with error computations, the results with the flash-lag effect have a different flavor. Here, we are looking at the actual predictions of the model, which, in this case, match perception. These predictions were trained to minimize errors in natural video sequences, suggesting that the percept matches more closely to the statistically predicted next frame than the actual frame. This natural statistics interpretation of the flash lag illusion has, in fact, been similarly suggested by Wojtach et al. (2008).

3.5 Conclusions

We have demonstrated that our PredNet model, trained end-to-end to predict future frames in natural videos, can replicate a variety of observations from neuroscience. Starting with extra-classical receptive field effects, our results align with the influential work of Rao & Ballard (1999). Importantly, however, our contributions extend beyond reproducing these efforts. In addition to being implemented in a modern deep learning framework with CNNs and LSTMs, a critical aspect of our approach is temporal prediction. Our models go beyond linear systems to the prediction of complex, natural videos. This manifests in the ability to explain higher level and temporal phenomena, such as the norm-based coding of faces and the flash-lag illusion. What is particularly striking is that a model trained to predict the next frame in videos of cars moving in Germany has units that resemble those recorded in macaques during a variety of simple visual stimulation protocols. Altogether, these results suggest that (temporal) prediction plays in important role in neural computation and representation development.

4

Conclusion

The brain has a remarkable ability to parse the world into behaviorally relevant concepts. Building machines with this level of understanding would have profound effects on society. In the limited domains where deep learning is reaching human-level performance, much of it has been through unnatural amounts of supervision, whether it is image labels in ImageNet (Russakovsky et al., 2014), or constant score updates in Atari games (Mnih et al., 2015). Relying less on these types of cues, and more on unsupervised signals, will be necessary for general artificial intelligence.

Here we argue that prediction is a powerful unsupervised loss function for training deep neural networks. Specifically, we demonstrate that models trained for next-frame prediction in video develop useful representations of underlying latent parameters, supporting tasks such as classification with fewer labeled examples. The first model we propose is an extension to a static autoencoder. We explore training this model with an adversarial loss, as in a Generative Adversarial Network (Goodfellow et al., 2014), illustrating its complementary effects to training with traditional pixel losses. For our next model, we take inspiration from the theory of predictive coding from the neuroscience literature. Trained with a hierarchical error passing scheme, the model proves to be very effective in next-frame prediction and represention learning, even for complex natural videos. In addition, this model is able to replicate a variety of phenomena found in neuroscience, suggesting prediction is an essential component of cortical processing.

While our results are promising, there are many avenues for future work. One significant limitation in the current PredNet model is that it is deterministic. The real world, on the other hand, is probabilistic. If an agent is to have a true understanding of the world, which we have argued is reflective in the ability to make predictions, the agent must also have a sense of this probabilistic nature. For instance, if a car approaches a stop sign at a "T" intersection, valid predictions would be that the car would either go left *or* right, not some combination of left *and* right, which would be the prediction of a model trained with a standard MSE/MAE loss. To implement a sampling mechanism of this kind in the PredNet, a GAN approach could be used. One particular formulation that could be effective would be using a "Siamese" discriminator, that takes both the predicted next frame and the actual next frame as input, and decides which was the true observation. We leave this approach for future exploration.

In addition to improvements of our model, there are also interesting neuroscience experiments that could be performed to elucidate the mechanisms and extent of probabilistic prediction in the brain. For instance, presenting a variant of a car going either left or right at a stop sign and recording neural responses in an animal model could be a means for measuring a predictive sampling signal. If the brain is truly sampling possible futures, one might except to see neural dynamics oscillating between two states. If this is the case, one might expect that varying the prior probability of the "left" state and "right" state would effect the relative time spent in either state.

Overall, we have motivated prediction as a rule for unsupervised learning. We demonstrated its usefulness in training artificial deep networks, as well as showing that a wide array of neural phenomena can be reproduced in a predictive framework. Our models, as well as other present AI models, are still far from having human-level intelligence. Distilling the unsupervised loss functions implemented by the brain into machines will help us close this gap.

A

Deep Learning for Medical Imaging: Application to Screening Mammography

While my previously presented work is oriented more towards research, my excitement for artificial intelligence and deep learning ultimately stems from the belief that it will have a dramatic

Material contained in this chapter has been published in Lotter et al. (2017b).

impact on society. One of the most significant applications of artificial intelligence is improving healthcare. As a computer vision scientist, an immediate domain of interest is medical imaging. Given the complexity of interpreting medical images, relying on tasks ranging from segmentation and detection, to reasoning and internal model building, it is ripe with interesting problems for the machine learning community. Paired with the potential for enormous societal benefit and the copious amounts of existing data, it is challenging to find a more promising domain of application.

While the potential of AI in medical imaging is clear, navigating the complex ecosystem of healthcare is challenging, especially as an outsider. This is just one of the many reasons why I have been fortunate enough to meet and work with Dr. Greg Sorensen. A neuroradiologist by trade, with decades-worth of experience in the highest echelons of healthcare, he has been an invaluable resource and mentor. In thinking about the most fruitful initial applications of deep learning to medical imaging, the problem that has stood out to us is screening mammography.

A.1 Background on Computer Vision for Mammography

A screening mammogram typically consists of two x-ray views of each breast. The American Cancer Society currently recommends the procedure at an annual or bi-annual basis (depending on age) for women over 45 (ACS). The purpose of screening mammography is to detect potential cancers in their early stages. Roughly one eighth of women in the United States will develop breast cancer during their lifetimes (Breastcancer.org, 2017), and early intervention is critical — five-year relative survival rates can be up to 3-4 times higher for cancers detected at an early stage, as compared to those detected at later stages (SEER). Although screening mammography has been associated with a 30% drop in mortality of breast cancer, it's overall value is limited by several factors. Interpreting mammograms is a tedious and error-prone process, and not all radiologists achieve uniformly high levels of accuracy (Elmore et al., 2009). In particular, empirically high false positive rates in screening mammography lead to significant unnecessary cost and patient stress (Brewer et al., 2007; ER et al., 2015). It is for all these reasons that effective machine vision-based solutions for reading mammograms hold significant potential to improve patient outcomes.

Traditional computer-aided diagnosis (CAD) systems for mammography have typically relied on hand-engineered features (Nishikawa, 2007). With the recent success of deep learning in other fields, there have been several promising attempts to apply these techniques to mammography (Arevalo et al., 2016; Mordang et al., 2016; Carneiro et al., 2015; Neeraj Dhungel, 2015; Dhungel et al., 2014; Geras et al., 2017; Kooi et al., 2016; Lévy & Jain, 2016; Yi et al., 2017; Zhu et al., 2016). Many of these approaches have been designed for specific tasks or subtasks of a full evaluation pipeline, for instance, mass segmentation (Zhu & Xie, 2016; Dhungel et al., 2014) or region-of-interest (ROI) microcalcification classification (Mordang et al., 2016). We have been working on the full problem of binary cancer status classification: given an entire mammogram image, we seek to classify whether cancer is present (Carneiro et al., 2015; Kooi et al., 2016; Zhu et al., 2016). As recent efforts have shown (Geras et al., 2017), creating an effective end-to-end differentiable model, the cornerstone of supervised deep learning, is very difficult given the ”needle in a haystack” nature of lesion detection in mammograms. To overcome this problem, we have developed a two-stage, curriculum learning-based approach (Bengio et al., 2009) consisting of first training patch-level CNN classifiers at multiple scales, followed by image-level aggregation and end-to-end training. The first level of training relies on location-specific lesion annotations, for which we use the Digital Database for Screening Mammography (DDSM) (Heath et al., 2001), the largest public mammography database. After training the full image-level model, we show that it is no longer necessary to require location annotated data for new datasets, which we demonstrate on the Digital Mammography DREAM Challenge, a public competition where we were amongst the top performers.

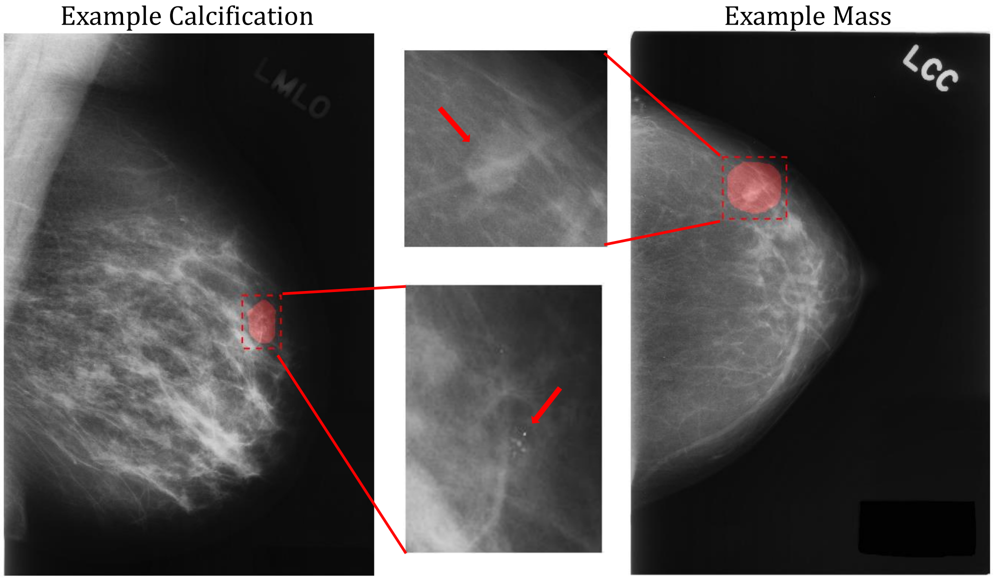

Figure A.1: Examples of the two most common categories of lesions in mammograms, calcifications and masses, from the DDSM dataset (Heath et al., 2001). Radiologist-annotated segmentation masks are shown in red. The examples were chosen such that the mask sizes approximately match the median sizes in the dataset, which is only about 0.5% and 1.2% of the image, for calcifications and masses respectively.

A.2 Multi-Scale CNN with a Curriculum Learning Approach

Figure A.1 shows typical examples of the two most common classes of lesions found in mammograms, masses and calcifications. Segmentation masks drawn by radiologists are shown in red (Heath et al., 2001). Even though the masks often encompass the surrounding tissue, the median size is only around 0.5% of the entire image area for calcifications, and 1.2% for masses. The insets shown in Fig. A.1 illustrate the high level of fine detail required for detection. As noted in (Geras et al., 2017), the requirement to find small, subtle features in large high resolution images (e.g. ~5500x3000 pixels in the DDSM dataset) means that the standard practice of downsampling images, which has proven effective in working with many standard natural image datasets (Russakovsky et al., 2014), is unlikely to be successful for mammograms. It is for these reasons that traditional mammogram classification pipelines typically consist of a sequence of steps, such as candidate ROI proposals, followed by feature extraction, perhaps segmentation, and finally classification (Kooi et al., 2016). While deep learning could in principle be used for any or all of these individual pieces, a variety of studies have suggested that the greatest gains from deep learning are seen when the system is trained "end-to-end" such that errors are backpropagated uninterrupted through the pipeline.

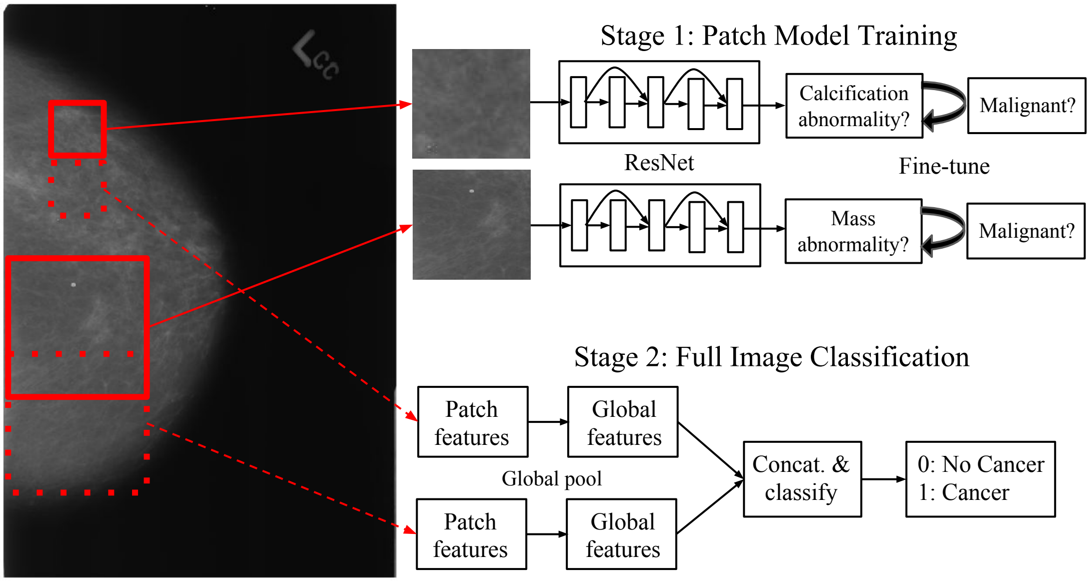

Figure A.2: Schematic of our approach, which consists of first training a patch classifier, followed by image-level training using a scanning window scheme. We train separate patch classifiers for calcifications and masses, at different scales, using a form of a ResNet CNN (He et al., 2015; Zagoruyko & Komodakis, 2016). For image training, we globally pool the last layer ResNet features at each scale, followed by concatenation and classification, with end-to-end training on binary cancer labels.

Our training strategy is illustrated in Fig A.2. The first stage of our approach consists of training a classifier to estimate the probability of the presence of a lesion in a given image patch. For training, we randomly sample patches from a set of training images, relying on segmentation maps to create labels for each patch. Given the different typical scales of calcifications and masses, and because DDSM has the corresponding annotations, we train a separate classifier for each. For the classifiers themselves, we use ResNets (He et al., 2015), specifically with the "Wide ResNet" formulation (Zagoruyko & Komodakis, 2016). We first train for abnormality detection (i.e. is there a lesion present), followed by fine-tuning for malignancy, as determined by the pathology outcomes. For the calcification classifier, we use a normal/benign/malignant labeling scheme for the fine-tuning, whereas we simply use a binary malignant/not-malignant scheme for masses, as the three-way scheme led to overfitting issues.

The second stage of our approach consists of image-level training, using the patch classifiers in a sliding window fashion. Instead of having a fixed stride with zero-padding, we partition the image into a set of patches such that each patch is contained entirely within the image, and the image is completely tiled, but there is as minimal overlap and number of patches as possible. The last layer before classification of the patch model is used for a set of features, which are aggregated across patches into a set of global features, using global average pooling. Max pooling was also explored, but didn't perform as well in our tests. For final classification, the globally pooled features at each of the two scales are simply concatenated, followed by a single fully-connected layer with sigmoidal activation. Using more fully-connected layers, either before or after concatenation, did not appear to improve performance. When training at the image level, we train end-to-end, updating the patch model weights as well.

A.3 Experiments on the Digital Database for Screening Mammography

We first evaluate our approach on the original version of the Digital Database for Screening Mammography (DDSM) (Heath et al., 2001), which consists of 10480 images from 2620 cases. Each case consists of the standard two views for each breast, craniocaudal (CC) and mediolateral-oblique (MLO). As there is not a standard cross validation split, we split the data into an 87%/5%/8% training/validation/testing split, where cross validation was done by patient.

For the first stage of training, we create a large dataset of image patches by sampling from the training images, enforcing that the majority of the patches come from the breast, by first segmenting using Otsu's method (Otsu, 1979). Before sampling, we resize the images (using bilinear interpolation) with different resize factors for calcification and mass patches. Instead of using a fixed resizing, which would cause distortions because the aspect ratio varies over the dataset, or cropping, which could cause a loss of information, we resize such that the resulting image falls within a particular range. We set the target size to 2750x1500 and 1100x600 pixels, for the calcification and mass scales respectively. Given an input image, we calculate a range of allowable resize factors as the min and max resize factors over the two dimensions. That is, given an example of size, say 3000x2000, the range of resize factors for the calcification scale would be $[1500/2000 = 0.75\,, 2750/3000 = 0.92]$, from which we sample uniformly. We then sample patches of 256x256 for input into the patch classifier. When creating patches, we also use data augmentation of horizontal flipping and rotation of up to 30°, as well as additional size augmentation by a factor 0.75 and 1.25, after the initial resizing. In the first stage of patch classification training, lesion detection without malignancy classification, we create 800K patches for each lesion category, split equally between positive and negative samples. In the second stage, we create 900K patches split equally between normal, benign, and malignant.

As mentioned above, for the patch classifiers, we use ResNets (He et al., 2015) with the "Wide ResNet" formulation (Zagoruyko & Komodakis, 2016), although our networks are not particularly wide, for the sake of training speed and to avoid overfitting. The Wide ResNet consists of groups of convolutional blocks with residual connections, and 2x2 average pooling between the groups. Each convolution in a block is proceeded by batch normalization followed by ReLU activation. After the final group, features are globally average pooled, followed by a single classification layer. The main hyperparameters of the model are the number of filters per layer and the number of residual blocks per group, N. For our models, we use five groups with the number of filters per group of (16, 32, 48, 64, 96) and an N of 2 and 4 for the calcification and mass models, respectively. For more details of the architecture, the reader is referred to (Zagoruyko & Komodakis, 2016). The only deviation we make is using 5x5 convolutions with a 3x3 stride for the initial convolutional layer, accounting for the relatively large input size we use of 256x256.

For training the patch models, we use RMSprop Tieleman & Hinton (2012) with a learning rate of 2×10^{-4} and batch size of 32. We train for 50 epochs with 10K patch samples per epoch and an equal proportion of positive and negative samples, followed by 125 epochs with 15K per epoch and a positive sample rate of 25%, for the initial abnormality detection stage. We then fine-tune for malignancy for 150 epochs with 15K per epoch for the masses, with a 20/40/40 normal/benign/malignant ratio and binary malignant/non-malignant labeling scheme. For calcifications, we fine-tune for 225 epochs, with an equal proportion of the three classes and 3-way classification. The difference in training between the two lesion classes is because masses were more prone to overfitting, as assessed with the validation set. To illustrate the information learned by the patch classifiers, we show several of the highest scoring patches for malignancy in the test set in Fig. A.3.

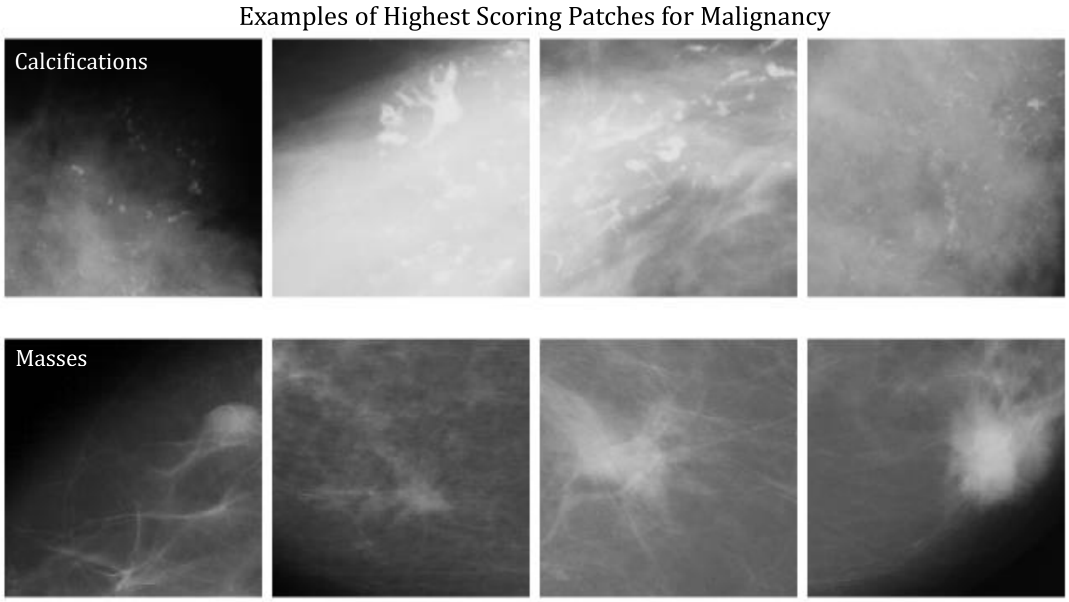

Figure A.3: Examples of the highest scoring patches for malignancy in the test set. The top row is for the calcification model and the bottom row is for the mass model.

For the image level training, we initialize the model with the final patch weights and follow a similar resizing scheme. For data augmentation, we again use horizontal reflections and resize by a factor chosen between 0.8 to 1.2, after the initial resizing. At each scale, we divide the image into 256x256 patches, using the stride strategy explained earlier. We also keep track of the regions of overlap between patches, and normalize these areas when global pooling, since otherwise the final features would be biased towards these locations. For image-level labels, we categorize according to if there is a malignant lesion in either view of the breast. Due to the possible different number of patches per image and because of the high memory footprint, we use a batch size of 1 during training. We train for 100K iterations using RMSprop with a learning rate of 2×10^{-4}, followed by 4×10^{-5} for 50K iterations, with final weights chosen by monitoring the area-under-the-curve (AUC) for a receiver operating characteristic (ROC) curve on the validation set. While we train on a per image basis, we report final results on a (patient, laterality) basis by averaging final scores across the CC and MLO views of the breast. For final test results, we average predictions across five resizing factors, chosen equally spaced between the allowable factors per image, and the two possible horizontal orientations.

Fig. A.4 contains the ROC curve on the DDSM test set for our proposed model. We obtain error bars using a bootstrapping estimate. Our model achieves an AUC of 0.92 ± 0.02. To provide a baseline of performance, we compare to results using a state-of-the-art CNN designed for ImageNet classification. We use the GoogLeNet (Szegedy et al., 2015a), specifically the InceptionV3 version (Szegedy et al., 2015b), choosing this model over alternatives because its input size is relatively large at 299x299. Because InceptionV3 is designed for a fixed input size, training with resizing augmentation isn't feasible, but we do train with horizontal flip augmentation. Consistent with many other results in the literature (Zhu et al., 2016; Carneiro et al., 2015), we find that ImageNet pre-training of InceptionV3 helps for eventual training on the DDSM mammogram dataset, though in the end, InceptionV3 still underperforms our model, achieving an AUC of 0.77 ± 0.03. Without ImageNet pre-training, the InceptionV3 model achieves an AUC of 0.59 ± 0.04. In both cases, results are still

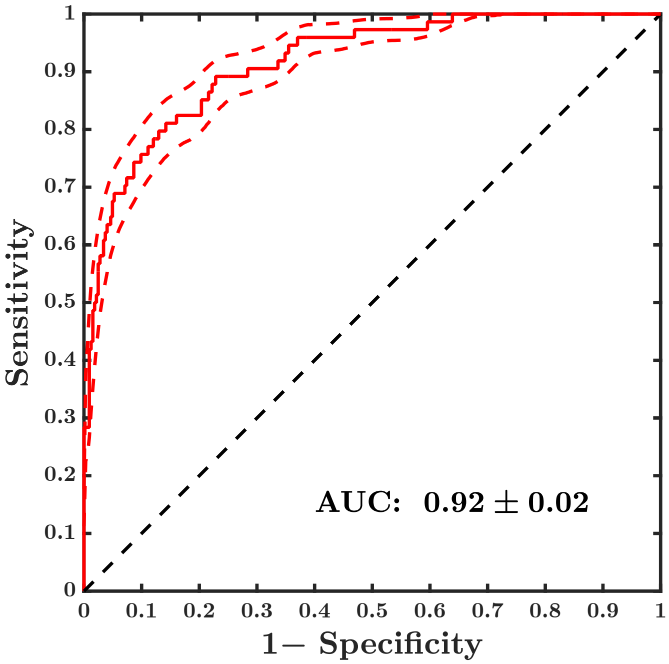

Figure A.4: ROC curve for our model on a test partition of the DDSM dataset. Predictions and ground-truth are compared at a breast-level basis.

reported on a (patient, laterality) basis with averaging across views and possible horizontal orientations. A summary of our results is contained in Table A.1.

	Pre-Training	Data Augment.	AUC
Multi-scale CNN	DDSM lesions	size, flips	0.92 ± 0.02
Multi-scale CNN	DDSM lesions	flips	0.89 ± 0.02
Multi-scale CNN	none	flips	0.65 ± 0.04
InceptionV3	ImageNet	flips	0.77 ± 0.03
InceptionV3	none	flips	0.59 ± 0.04

Table A.1: ROC AUC by pre-training and data augmentation. InceptionV3 assumes a fixed input size, so "size" augmentation, i.e. random input image resizing, isn't directly feasible. ROC curve on left corresponds to the top row.

To make a more controlled comparison, we also report the results for our model without size augmentation training, e.g. having a fixed resize factor, chosen as the mean of the allowable resize factors calculated for each image. The performance drops slightly to 0.89 ± 0.02, but is still significantly higher than the InceptionV3 model, indicating that our better performance cannot simply be explained by different dataset augmentation procedures. The third row of the table also contains results for our model without the DDSM lesion pre-training, which substantially decreases performance to 0.65 ± 0.04, however the model still performs somewhat better than the InceptionV3 model without pre-training (last row). Altogether, these results suggest that all elements of our approach — including model formulation and pre-training scheme — are important for accurate full image mammogram classification performance.

A.4 The Digital Mammography DREAM Challenge

With our promising results on DDSM, we were excited to participate in The Digital Mammography DREAM Challenge (Sage Bionetworks), an open data science competition organized. The data for the competition consisted of over 640K digital mammograms representative of a screening population. There competition was organized into two sub-challenges. Sub-challenge 1 consisted of predicting cancer status of each patient given only the imaging data for their most recent exam. In Sub-challenge 2, previous mammogram exams, as well as meta-data (i.e. age, family history, etc.), were available to make final predictions. The most unique and challenging aspect of the competition was that participants were not actually given the training data. Access was granted only through the use of Docker containers, which were run on the challenge servers, where the data was mounted. Thus, a large part of the competition was navigating the IT infrastructure and intelligently using the alotted resources, to which there was a maximum time usage quota for each round.

Using a similar model presented above on the DDSM data, we were a top performing team in the competition. Starting with the pre-trained weights from the DDSM data, the model was able to rapidly reach respectable levels of performance. Competing against over 1200 other registered participants, we won the second round of the challenge, and ultimately tied for second in both the third and validation rounds. In Sub-challenge 1, our best model achieved an AUROC of 0.872. Combining an AdaBoost classifier trained on the patient metadata with our image model, we achieved a mild boost in performance for Sub-challenge 2, obtaining an AUROC of 0.873. The biggest issues we had were finding a good learning schedule for stable training (choosing learning rates, dropout ratios, etc.). Given the training time quota, our final model ultimately didn't even see a good proportion of the training set. In the future, we hope to remedy some of these issues and continue to improve our model.

A.5 Conclusions

Computer-aided diagnosis for mammography is a heavily studied problem given its potential for large real-world impact. This field, like many others, is transitioning from hand-engineered features to features learned in a deep learning framework. While there have been many works applying deep learning to subcomponents of the mammography pipeline, here we are concerned with full image classification. Given the high resolution and relatively small ROIs, effectively designing an end-to-end solution is challenging. We have presented a multi-scale CNN scanning window scheme with a lesion-specific curriculum learning strategy that achieves promising results, which we have demonstrated on the largest public mammogram dataset, as well as in a heavily participated open competition. Overall, we believe mammogram classification, and medical imaging analysis in general, is a fruitful task for developing novel machine learning methods, and one in which improvements could have dramatic societal impact.

References

Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. *CoRR*, 2015.

Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. *CoRR*, 2017.

J. Arevalo, F. González, R. Ramos-Pollan, and et al. Representation learning for mammography mass lesion classification with convolutional neural networks. *Computer Methods and Programs in Biomedicine*, 2016.

Joseph J Atick. Could information theory provide an ecological theory of sensory processing? *Network: Computation in neural systems*, 1992.

Andre M. Bastos, W. Martin Usrey, Rick A. Adams, George R. Mangun, Pascal Fries, and Karl J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 2012.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, 2015.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. *ICML*, 2009.

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

Riccardo Biasini, George Hotz, Sam Khalandovsky, Eder Santana, and Niel van der Westhuizen. Comma.ai research, 2016. URL https://github.com/commaai/research.

Sage Bionetworks. The digitial mammography dream challenge, 2017. URL https://www.synapse.org/#!Synapse:syn4224222/wiki/.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, 2016.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. *CoRR*, 2016.

Breastcancer.org. U.s. breast cancer statistics, 2017. URL http://www.breastcancer.org/symptoms/understand_bc/statistics.

Olivier Breuleux, Yoshua Bengio, and Pascal Vincent. Quickly generating representative samples from an rbm-derived process. *Neural Computation*, 2011.

N.T. Brewer, T. Salz, and S.E. Lillie. Systematic review: The long-term effects of false-positive mammograms. *Annals of Internal Medicine*, 2007.

Gustavo Carneiro, Jacinto C. Nascimento, and Andrew P. Bradley. Unregistered multiview mammogram analysis with pre-trained deep learning models. In *MICCAI*, 2015.

Rakesh Chalasani and Jose C. Principe. Deep predictive coding networks. *CoRR*, 2013.

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

Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. Deeppermnet: Visual permutation learning. *CoRR*, 2017.

Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, 2015.

Neeraj Dhungel, Gustavo Carneiro, and Andrew P. Bradley. Deep structured learning for mass segmentation from mammograms. *CoRR*, 2014.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, 2015.

Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *CVPR*, 2009.

Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *CoRR*, 2016.

David M. Eagleman and Terrence J. Sejnowski. Motion integration and postdiction in visual awareness. *Science*, 2000.

Tobias Egner, Jim M. Monti, and Christopher Summerfield. Expectation and surprise determine neural population responses in the ventral visual stream. *J Neurosci*, 2010.

Joann G. Elmore, Sara L. Jackson, Linn Abraham, Diana L. Miglioretti, Patricia A. Carney, Berta M. Geller, Bonnie C. Yankaskas, Karla Kerlikowske, Tracy Onega, Robert D. Rosenberg, Edward A. Sickles, and Diana S. M. Buist. Variability in interpretive performance at screening mammography and radiologists' characteristics associated with accuracy. *Radiology*, 2009.

Myers ER, Moorman P, Gierisch JM, and et al. Benefits and harms of breast cancer screening: A systematic review. *JAMA*, 2015.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 2010.

Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 2017.

Daniel J Felleman and David C Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1991.

Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. *CoRR*, 2016.

Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, 2016.

Tucker G. Fisher, Henry J. Alitto, and W. Martin Usrey. Retinal and non-retinal contributions to extraclassical surround suppression in the lateral geniculate nucleus. *Journal of Neuroscience*, 2016.

Peter Földiák. Learning invariance from transformation sequences. *Neural Computation*, 1991.

Katerina Fragkiadaki, Sergey Levine, and Jitendra Malik. Recurrent network models for kinematic tracking. *CoRR*, 2015.

Karl Friston. A theory of cortical responses. *Philos Trans R Soc Lond B Biol Sci*, 2005.

Zhe Gan, Chunyuan Li, Ricardo Henao, David Carlson, and Lawrence Carin. Deep temporal sigmoid belief networks for sequence modeling. *CoRR*, 2015.

Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition*, 2014.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Dileep George and Jeff 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.

Krzysztof J. Geras, Stacey Wolfson, S. Gene Kim, Linda Moy, and Kyunghyun Cho. High-resolution breast cancer screening with multi-view deep convolutional neural networks. *CoRR*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*. 2014.

Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, 2017.

Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. *CoRR*, 2015a.

Ross Goroshin, Michaël Mathieu, and Yann LeCun. Learning to linearize under uncertainty. *CoRR*, 2015b.

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, 2015.

Michael Heath, Kevin Bowyer, Daniel Kopans, Richard Moore, and W. Philip Kegelmeyer. The digital database for screening mammography. *Proceedings of the Fifth International Workshop on Digital Mammography*, 2001.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Toshihiko Hosoya, Stephen A Baccus, and Markus Meister. Dynamic predictive coding by the retina. *Nature*, 2005.

Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2011.

D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 1968.

David H. Hubel and Torsten N. Wiesel. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology*, 1965.

Mi-Young Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning? *CoRR*, 2016.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

Elias B. Issa, Charles F. Cadieu, and James J. DiCarlo. Evidence that the ventral stream codes the errors used in hierarchical inference and learning. *bioRxiv*, 2016.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*. 2009.

Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: Higher order temporal coherence in video. *CoRR*, 2015.

Janneke FM Jehee, Constantin Rothkopf, Jeffrey M Beck, and Dana H Ballard. Learning receptive fields using predictive feedback. *Journal of Physiology-Paris*, 2006.

HE Jones, KL Grieve, W Wang, and AM Sillito. Surround suppression in primate v1. *Journal of neurophysiology*, 2001.

Nal Kalchbrenner, Aaron van den Oord, , Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. *CoRR*, 2016.

Ryota Kanai, Yutaka Komura, Stewart Shipp, and Karl Friston. Cerebral hierarchies : predictive processing , precision and the pulvinar. *Philos Trans R Soc Lond B Biol Sci*, 2015.

D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *CoRR*, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

Kristin Koch, Judith McLean, Ronen Segev, Michael A Freed, Michael J Berry, Vijay Balasubramanian, and Peter Sterling. How much the eye tells the brain. *Current Biology*, 2006.

Thijs Kooi, Geert Litjens, Bram van Ginneken, Albert Gubern-Mérida, Clara I. Sánchez, Ritse Mann, Ard den Heeten, and Nico Karssemeijer. Large scale deep learning for computer aided detection of mammographic lesions. *Medical Image Analysis*, 2016.

Tejas D. Kulkarni, Will Whitney, Pushmeet Kohli, and Joshua B. Tenenbaum. Deep convolutional inverse graphics network. *CoRR*, 2015.

Victor A.F. Lamme. Feedforward , horizontal , and feedback processing in the visual cortex. *Curr Opin Neurobiol.*, 1998.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a proxy task for visual understanding. *CoRR*, abs/1703.04044, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

David Leopold. High-level visual specialization in the brain: Linking single neurons to fmri studies. Talk given at Harvard University, March 28, 2017, 2017.

Daniel Lévy and Arzav Jain. Breast mass classification from mammograms using deep convolutional neural networks. *CoRR*, 2016.

William Lotter, Gabriel Kreiman, and David Cox. Unsupervised learning of visual structure using predictive generative networks. *ICLR (Workshop Track)*, 2016.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *ICLR*, 2017a.

William Lotter, Greg Sorensen, and David Cox. A multi-scale cnn and curriculum learning strategy for mammogram classification. *MICCAI Workshop on Deep Learning in Medical Image Analysis*, 2017b.

Zelun Luo, Boya Peng, De-An Huang, Alexandre Alahi, and Li Fei-Fei. Unsupervised learning of long-term motion dynamics for videos. *CoRR*, 2017.

Donald M Mackay. Perceptual stability of a stroboscopically lit visual field containing self-luminous objects. *Nature*, 1958.

Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016.

Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. In *CVPR*. 2007.

Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent ”grammar cells”. In *NIPS*. 2014.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, 2014.

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Unsupervised learning using sequential verification for action recognition. *CoRR*, 2016.

Roni Mittelman, Benjamin Kuipers, Silvio Savarese, and Honglak Lee. Structured recurrent temporal restricted boltzmann machines. In *ICML*. 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Hossein Mohabi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *ICML*. 2009.

Jan-Jurre Mordang, Tim Janssen, Alessandro Bria, Thijs Kooi, Albert Gubern-Mérida, and Nico Karssemeijer. Automatic microcalcification detection in multi-vendor mammography using convolutional neural networks. In *Proceedings of the 13th International Workshop on Breast Imaging - Volume 9699*, 2016.

Scott O. Murray, Daniel Kersten, Bruno A. Olshausen, Paul Schrater, and David L. Woods. Shape perception reduces activity in human primary visual cortex. 2002.

Jonathan J Nassi, Camille Gómez-Laberge, Gabriel Kreiman, and Richard T Born. Corticocortical feedback increases the spatial extent of normalization. *Frontiers in systems neuroscience*, 2014.

Andrew P Bradley Neeraj Dhungel, Gustavo Carneiro. Automated mass detection in mammograms using cascaded deep learning and random forests. *DICTA*, 2015.

Romi Nijhawan. Motion extrapolation in catching. *Nature*, 1994.

Robert M Nishikawa. Current status and future directions of computer-aided diagnosis in mammography. *Computerized Medical Imaging and Graphics*, 2007.

Mark J Nolt, Romesh D Kumbhani, and Larry A Palmer. Contrast-dependent spatial summation in the lateral geniculate nucleus and retina of the cat. *Journal of neurophysiology*, 2004.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, 2016.

SEER Program of the National Cancer Institute. Cancer stat facts: Female breast cancer, 2013. URL https://seer.cancer.gov/statfacts/html/breast.html.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder P. Singh. Action-conditional video prediction using deep networks in atari games. *CoRR*, 2015.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *CVPR*, 2015.

Randall C. O'Reilly, Dean Wyatte, and John Rohrlich. Learning through time in the thalamocortical loops. *CoRR*, 2014.

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1979.

Rasmus Berg Palm. Prediction as a candidate for learning deep hierarchical models of data. *Master's thesis, Technical University of Denmark*, 2012.

Deepak Pathak, Ross B. Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. *CoRR*, 2016.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *CoRR*, 2015.

Mohammad Pezeshki, Linxi Fan, Philemon Brakel, Aaron C. Courville, and Yoshua Bengio. Deconstructing the ladder network architecture. *CoRR*, 2015.

Nicolas Pinto, David Doukhan, James J. DiCarlo, and David D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol*, 2009.

Senthil Purushwalkam and Abhinav Gupta. Pose from action: Unsupervised learning of pose features based on motion. *CoRR*, 2016.

Marc'Aurelio Ranzato, Arthur Szlam, Joan Bruna, Michaël Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, 2014.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 1999.

Rajesh PN Rao. Correlates of attention in a model of dynamic visual recognition. In *NIPS*, 1998a.

Rajesh PN Rao. Correlates of attention in a model of dynamic visual recognition. In *NIPS*, 1998b.

Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder network. *CoRR*, 2015.

Gillian Rhodes and Linda Jeffery. Adaptive norm-based coding of facial identity. *Vision Research*, 2006.

Dario L Ringach, Robert M Shapley, and Michael J Hawken. Orientation selectivity in macaque v1: diversity and laminar dependence. *Journal of Neuroscience*, 2002.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, 2014.

Andrew Saxe, Maneesh Bhand, Zhenghao Chen, Pang Wei Koh, Bipin Suresh, and Andrew Y. Ng. On random weights and unsupervised feature learning. In *Workshop: Deep Learning and Unsupervised Feature Learning (NIPS)*. 2010.

Andrew M. Saxe, James L. Mcclelland, and Surya Ganguli. Learning hierarchical category structure in deep neural networks. *Proc. of the Cognitive Science Society*, 2013.

Michael P Sceniak, Dario L Ringach, Michael J Hawken, and Robert Shapley. Contrast's effect on spatial summation by macaque v1 neurons. *Nature neuroscience*, 1999.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Singular Inversions, Inc. FaceGen. http://facegen.com.

American Cancer Society. American cancer society recommendations for the early detection of breast cancer, 2016. URL https://www.cancer.org/cancer/breast-cancer.

William R. Softky. Unsupervised pixel-prediction. *NIPS*, 1996.

Samuel G Solomon, Andrew JR White, and Paul R Martin. Extraclassical receptive field properties of parvocellular, magnocellular, and koniocellular cells in the primate lateral geniculate nucleus. *Journal of Neuroscience*, 2002.

Samuel G Solomon, Barry B Lee, and Hao Sun. Suppressive surrounds and contrast gain in magnocellular-pathway retinal ganglion cells of macaque. *Journal of Neuroscience*, 2006.

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.

Mandyam V Srinivasan, Simon B Laughlin, and Andreas Dubs. Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London B: Biological Sciences*, 1982.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015.

Christopher Summerfield, Tobias Egner, Matthew Greene, Etienne Koechlin, Jennifer Mangels, and Joy Hirsch. Predictive codes for forthcoming perception in the frontal cortex. *Science*, 314, 2006.

Lin Sun, Kui Jia, Tsung-Han Chan, Yuqiang Fang, Gang Wang, and Shuicheng Yan. Dl-sfa: Deeply-learned slow feature analysis for action recognition. *CVPR*, 2014.

Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted boltzmann machine. In *NIPS*. 2009a.

Ilya Sutskever, Geoffrey E. Hinton, and Graham W. Taylor. The recurrent temporal restricted boltzmann machine. In *NIPS*. 2009b.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015a. URL http://arxiv.org/abs/1409.4842.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, 2015b.

Graham W. Taylor and Geoffrey Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of The 26th International Conference on Machine Learning*, pp. 1–8. 2009.

Lucas Theis, Aaron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *ICLR*, 2016.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - rmsprop, coursera. *Neural networks for machine learning*, 2012.

Harri Valpola. From neural pca to deep unsupervised learning. *CoRR*, 2015.

R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing Motion and Content for Natural Video Sequence Prediction. *CoRR*, 2017.

Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. *CoRR*, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*. 2008a.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008b.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *CoRR*, 2016.

Guy Wallis and Heinrich H. Bulthoff. Effects of temporal association on recognition memory. *PNAS*, 2001.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. *CoRR*, 2015.

Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

Laurenz Wiskott and Terrence J. Sejnowski. Learning invariance from transformation sequences. *Neural Computation*, 2002.

William T Wojtach, Kyongje Sung, Sandra Truong, and Dale Purves. An empirical explanation of the flash-lag effect. *Proceedings of the National Academy of Sciences*, 2008.

Zuxuan Wu, Larry S. Davis, and Leonid Sigal. Weakly-supervised spatial context networks. *CoRR*, 2017.

Tianfan Xue, Jiajun Wu, Katherine L. Bouman, and William T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *CoRR*, 2016.

Darvin Yi, Rebecca Lynn Sawyer, David Cohn III, Jared Dunnmon, Carson Lam, Xuerong Xiao, and Daniel Rubin. Optimizing and visualizing deep learning for benign/malignant classification in breast tumors. *CoRR*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, 2016.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, 2016a.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *CoRR*, 2016b.

Wentao Zhu and Xiaohui Xie. Adversarial deep structural networks for mammographic mass segmentation. *CoRR*, 2016.

Wentao Zhu, Qi Lou, Yeeleng Scott Vang, and Xiaohui Xie. Deep multi-instance networks with sparse label assignment for whole mammogram classification. *CoRR*, 2016.