PreCNet: Next-Frame Video Prediction Based on Predictive Coding

Zdenek Straka*, Tomáš Svoboda*, Member, IEEE, and Matej Hoffmann*, Member, IEEE

Abstract—Predictive coding, currently a highly influential theory in neuroscience, has not been widely adopted in machine learning yet. In this work, we transform the seminal model of Rao and Ballard (1999) into a modern deep learning framework while remaining maximally faithful to the original schema. The resulting network we propose (PreCNet) is tested on a widely used next-frame video prediction benchmark, which consists of images from an urban environment recorded from a car-mounted camera, and achieves state-of-the-art performance. Performance on all measures (MSE, PSNR, and SSIM) was further improved when a larger training set (2M images from BDD100k) pointed to the limitations of the KITTI training set. This work demonstrates that an architecture carefully based on a neuroscience model, without being explicitly tailored to the task at hand, can exhibit exceptional performance.

Index Terms—Deep neural networks, next-frame video prediction, predictive coding, self-supervised learning.

I. INTRODUCTION

Predicting near future is a crucial ability that every agent—humans, animals, or robots—needs for survival in a dynamic and complex environment. Just to safely cross a busy road, one needs to anticipate the future position of cars and pedestrians, as well as the consequences of own actions. Machines are still lagging in this ability. For deployment in such environments, it is necessary to overcome this gap and develop efficient methods for foreseeing the future.

One candidate approach for predicting the near future is predictive coding—a popular theory from neuroscience. The basic idea is that a brain is a predictive machine that anticipates incoming sensory inputs, and only the prediction errors—unpredicted components—are used for updating an internal representation. In addition, predictive coding tackles another important aspect of perception: how to efficiently encode redundant sensory inputs [1]. Rao and Ballard [2] proposed and implemented a hierarchical architecture—which we will refer to as predictive coding schema (see Section III-A for details)—that explains certain important properties of the visual cortex: the presence of oriented edge/bar detectors and extra-classical receptive field effects. This schema has influenced several works on human perception and neural information processing in the brain (see, e.g., [3], [4], [5], [6]; for reviews, see [1], [7], [8]).

In this work, our goal was to remain as faithful as possible to the predictive coding schema but cast it into a modern deep learning framework. We thoroughly analyze how the conceptual architecture is preserved. To demonstrate the performance, we chose a widely used benchmark—next-frame video prediction—for the following reasons. First, large datasets of unlabeled sequences are available and this task bears direct application potential. Second, this task is an instance of unsupervised representation learning, which is currently actively researched (e.g., [9]). Third, the complexity of the task can be scaled, for example, by performing multiple-frame prediction (frames are anticipated more steps ahead). On a popular next-frame video prediction benchmark, our—strongly biologically grounded—network achieves state-of-the-art performance. In addition to the commonly used training dataset (KITTI), we trained the model on a significantly bigger dataset, which improved the performance even further.

We summarize our contributions as follows. First, in this work, the seminal predictive coding model of Rao and Ballard [2] has been cast into a modern deep learning framework, while remaining as faithful as possible to the original schema. Second, we tested our architecture (PreCNet) on a widely used next-frame video prediction benchmark (KITTI with 41k images for training, Caltech Pedestrian Dataset for testing) and outperformed most of the state-of-the-art methods and achieved 2nd–3rd rank when measured with the structural similarity index (SSIM)—a performance measure that should best correlate with human perception. Third, performance on all three measures (MSE, PSNR, and SSIM) was significantly improved when a larger training set (BDD100k with 2M images) than usually used by the community (KITTI) was employed.

This article is structured as follows. The related work section overviews models inspired by predictive coding and state-of-the-art methods for video prediction. This is followed by the architecture section where we describe our model and compare it in detail with the original Rao and Ballard schema [2] and PredNet [10]—a model for next-frame

*Corresponding author: Matej Hoffmann.

The authors are with the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, 12135 Prague, Czech Republic (e-mail: straka.zdenek@fel.cvut.cz; svobodat@fel.cvut.cz; matej.hoffmann@fel.cvut.cz).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2023.3240857, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2023.3240857
video prediction inspired by predictive coding. In Section IV, we detail the datasets, performance metrics, and our experiments in next and multiple-frame video prediction. This is followed by a conclusion, discussion, and future work. All code and trained models used in this work are available at [11].

II. RELATED WORK

This section starts with a summary of predictive coding-inspired machine-learning models. This is followed by an overview of state-of-the-art methods for video prediction.

A. Predictive Coding Models

In this section, we will focus on predictive coding-inspired machine-learning models. A reader who is interested in the application in computational and theoretical neuroscience may find useful reviews [1], [8], [12] and references [2], [3], [4], [5], [6], [13]. Predictive coding, a theory originating in neuroscience, is more a general schema (with certain properties) than a concrete model. Therefore, no “correct” model of predictive coding is available to date. In this work, by predictive coding, we will understand a well-defined schema proposed by Rao and Ballard [2], which was also implemented as a computational model (see Section III-A for a description of the schema). This schema, which is highly influential in neuroscience, embodies crucial ideas of the predictive coding theory.

We will relate predictive coding-inspired machine-learning models to the schema by Rao and Ballard and analyze which properties of the original are preserved and which are not. A detailed comparison of our deep neural network—intended to be as faithful as possible to the Rao and Ballard schema—will be presented in a separate Section III-C1. The models with static inputs and sequences will be presented separately.

1) Models With Static Inputs: Song et al. [14] proposed fast inference predictive coding (FIPC) model for image representation and classification, which extends the schema by Rao and Ballard by: 1) a regression procedure with fast inference during testing and 2) a classification layer which directly represents representation learning to achieve discriminative features. An important part of predictive coding theory is the existence of prediction error neurons along with representational neurons (see [2], [8]). Models [15], [16], [17] intended for object recognition in natural images have these two distinct neural populations, however, their training is not based on the prediction error minimization used in predictive coding. A generative model by Dora et al. [18] for inferring causes underlying visual inputs does not follow the division into the error and representational neurons. However, the model is trained, in accordance with predictive coding, to minimize prediction errors. The same authors contributed to the model that extends the predictive coding approach to the inference of latent visuo-tactile representations [19], used for place recognition of a biomimetic robot in a simulated environment.

2) Models With Sequences as Inputs: Ahmadi and Tani proposed the predictive-coding-inspired variational recurrent neural network [20] (P-RVNN). The network works in a three-stage processing cycle: 1) producing prediction; 2) backpropagating the prediction errors across the network hierarchy; and 3) updating the internal states of the network to minimize future prediction errors. The network was used for synchronous imitation between two robots—joint angles and XYZ coordinates of a hand tip were used—and for extracting latent probabilistic structure from a binary output of a simple probabilistic finite state machine. Using the same three-stage predictive coding processing cycle, Choi and Tani [21] developed a predictive multiple spatiotemporal recurrent neural network (P-MSTRNN) for predicting binary image (36 × 36 pixels) sequences of human whole-body cyclic movement patterns. They also explored how the inferred internal (latent) states can be used for the recognition of movement patterns. Chalasani and Principe [22] proposed a hierarchical linear dynamical model for feature extraction. The model took inspiration from predictive coding and used higher-level predictions for inference of lower-level predictions. However, all three models do not use the division into the error and representational neurons and consequently use a different schema than Rao and Ballard [2].

Lotter et al. [10] proposed a predictive neural network (PredNet) for next-frame video prediction. The network follows the division into error and representational neurons, but the processing schema is different from the one proposed by Rao and Ballard [2] and consequently to our model (see Section III-C3 for details). Despite the architectural differences from the schema by Rao and Ballard, the network could mimic certain features of biological neurons and perception [23].

B. Video Prediction Models

Video prediction is an important task in computer vision with a long history. A sequence of images is given and one or multiple following images are predicted (i.e., next- and multiple-frame video prediction task, respectively). As prediction of the next sensory input is inherent to predictive coding, next-frame video prediction provides a natural use case to benchmark the performance of our neural network architecture. Therefore, we will focus predominantly on a brief review of recent work with state-of-the-art performance on next-frame video prediction and—wherever feasible—we will quantitatively compare the performance (see Section IV-C4).

Many of the methods for video prediction produce blurred predictions. As blurriness is undesirable, Matthieu et al. [9] proposed a gradient difference loss function that is minimized when the gradient of the actual and predicted image is the same. This loss function was then combined with adversarial learning. Byeon et al. [24] showed with their LSTM-based architecture that the direct connection of each predicted pixel with the whole available past context led to decreasing prediction uncertainty on pixel level and therefore also reduced blurriness. Reda et al. [25] suggested that blurriness is amplified by using datasets with a lack of large motion and small resolution. Therefore, they used video games (GTA-V and Battlefield-I) for the generation of a large high-resolution dataset with large enough motion (testing was performed on natural sequences). The dataset was then used for training a model which combines a kernel-based approach with the usage of optical flow. In addition to optical flow estimation, a model by Lu et al. [26] used pixel generation and adversarial
training. Gao et al. [27] proposed a model which performed the generation of the future frames in two steps. First, a flow predictor was used for warping the nonoccluded regions. Then, the occluded regions were inpainted by the same network. A method by Liu et al. [28] did not use optical flow directly, however, a deep network was trained to synthesize a future frame by flowing pixel values from the given video frames. This self-supervised method was also used for interpolation. Similar to Gao et al. [27], Hao et al. [29] proposed a two-stage architecture. However, the input of a network contained, in addition, sparse motion trajectories (automatically extracted for video prediction). First, the network produced a warped image that respected the given motion trajectories. In the second stage, occluded parts of the image were hallucinated and color change was compensated.

Villegas et al. [30] introduced a model that first performed human pose detection and its future evolution. Then, the predicted human poses were used for future frame generation.

Finn et al. [31] proposed a model that next to visual inputs takes actions of the robot into account. This action-conditioned model learned to anticipate pixel motions relatively to the previous frame.

A conditionally reversible network (CrevNet) proposed by Yu et al. [32] uses a bijective two-way autoencoder, based on convolutional networks, for encoding and decoding input frames. Feature maps obtained from the autoencoder are then used as input to a ConvRNN-based predictor. The transformed feature maps by the predictor are then decoded by the autoencoder and outputted as predicted frames. Chang et al. [33] proposed an information-preserving spatiotemporal predictive model (IPRNN) which used skip connections from encoders to decoders. As a decoder has access to information from an encoder, the information loss is reduced. The model used stacked spatiotemporal gated recurrent units which took the encoded states as input. Yuan et al. [34] integrated the attention mechanism into a convolutional LSTM network. This model was further extended by pixel restoration of the input images to the predictions and denoted as the deep pixel restoration AttConvLSTM (DPRAConvLSTM) model. An attention mechanism was also effectively used for human-skeleton motion prediction [35], [36].

Some other state-of-the-art architectures are based on generative adversarial networks (GANs). The GAN by Kwon and Park [37] is trained to anticipate both future and past frames. The GAN proposed by Liang et al. [38] is trained to consistently predict future frames and pixel-wise flows using a dual learning mechanism. Vondrick et al. [39] proposed GAN for the generation of image sequences that unravels the foreground from the background of the images. A video prediction network proposed by Jin et al. [40] integrates generative adversarial learning with the usage of spatial and temporal wavelet analysis modules.

The stochastic nature of natural video sequences makes it impossible to predict the future sequence perfectly. Models such as [41], [42], [43] attempt to deal with that by generating multiple possible futures. The objective is to predict frame sequences that are: 1) diverse; 2) perceptually realistic; and 3) a plausible continuation of the given input sequence or image [41]. Therefore, this task is different from deterministic video frame prediction whereby the model is intended to produce only a single frame or sequence best fitting the actual future.

Some of the mentioned works [10], [32], [38], [39] also demonstrated that the representations which were learned during next-frame video prediction training could be used for supervised learning tasks (e.g., human action recognition).

III. Architecture

This section starts with a description of the predictive coding schema which was proposed by Rao and Ballard [2]. This is followed by a detailed description of our model. The section is closed by a comparison of our model with related models:

1) A hierarchical network for predictive coding proposed by Rao and Ballard and 2) PredNet—a deep network for next-frame video prediction inspired by predictive coding.

A. Predictive Coding Schema

Motivated by crucial properties of the visual cortex, Rao and Ballard have proposed a hierarchical predictive coding schema with its implementation [2]. According to this schema, throughout the hierarchy of visual processing, “feedback connections from a higher- to a lower-order visual cortical area carry predictions of lower-level neural activities, whereas the feedforward connections carry the residual errors between the predictions and the actual lower-level activities” [2]. The residual errors are used to reduce the prediction error in the following moment [see Fig. 1(b)].

This schema was directly turned into a computational model in [2] [see Fig. 1(a)]. The feedback connection from the higher-level to lower-level predictive estimator (PE) carries the top-down prediction $r^{td}$ of the lower-level PE activity $r$. The residual error $r - r^{td}$ is sent back via feedforward connections to the higher-level PE. The same error with opposite sign, $r^{td} - r$, affects the following PE activity $r$ [see Fig. 1(a)]. The bottom-level PE produces a prediction of the visual input.

Drawing on the predictive coding schema, we propose the predictive coding network (PreCNet) [see Fig. 1(c)]. In contrast with the model by Rao and Ballard (compare parts (a), (c) of Fig. 1), PreCNet uses a modern deep learning framework (see Section III-B for details of PreCNet architecture and Section III-C1 for a more detailed comparison of both models). This has enabled us to create a model based on the predictive coding schema with state-of-the-art performance, as demonstrated on the next-frame video prediction benchmark.

B. Description of PreCNet Model (Ours)

The structure, computation of prediction and states, and training of the model are detailed below.

1) Structure of the Model: The model, shown in Fig. 2, consists of $N$ +1 hierarchically organized modules. A module $i \in \{0, 1, \ldots, N\}$ consists of the following components.

1The model is the same as in Fig. 1(c). However, to enable direct comparison with Rao and Ballard model, it was redrawn in a different arrangement for Fig. 1(b). The PE from the model of [2] is not equivalent to the “Module” in Fig. 2. See Fig. 3(b), (e) for a comparison.
1) A representation layer is a convolutional LSTM (convLSTM) layer (see [44], [45]) with output state $R_i$ (alternatively $\hat{r}$). The convLSTM followed dynamics from commonly used “No peepholes” LSTM variant [46] (see Supplementary materials—convLSTM for details). Technically, it consists of two convolutional LSTM layers (convLSTM$^{\text{up/down}}$) which share hidden and cell states ($R_i, C_i$) but differ in the input ($E_i$ vs. $E_{i+1}$). The input, forget, and output gates use a hard sigmoid as an activation function. During the calculation of the final (hidden) and cell states, a hyperbolic tangent is used.

2) An error representation consists of the rectified linear units (ReLU) whose input is obtained by merging errors PREDICTION – ACTUAL STATE and ACTUAL STATE – PREDICTION. The state of the error representation is denoted as $E_i$.

3) A decoding layer is a convolutional (conv$^l$) layer with output state $\hat{A}_i$. It uses ReLU as an activation function.

4) An upsample layer, which uses the nearest-neighbor method, upscales its input by factor 2. This layer is not present in module 0.

5) A max-pooling layer which down scales its input by a factor of 2. This layer is not present in module 0.

2) Computation of the Prediction and States: In every timestep, PreCNet outputs a prediction of the incoming image. The error of the prediction is then used for the update of the states (see also Fig. 4). The computation in every timestep can be divided into two phases.

1) Prediction phase: The information flow goes iteratively from a higher to a lower module. At the end of this phase (at Module 0), the prediction of the incoming input image $A_0$ is outputted.

2) Correction phase: In this phase, the information flow goes iteratively up. The error between the prediction and actual input is propagated upward.

In a nutshell, a representational layer (with state $R_i$) represents a prediction of the image $I_i$ or a pooled convLSTM state $R_{i-1}$ from the module below ($i > 0$). The decoding layer transforms the representation $R_i$ into the prediction $\hat{A}_i$. The error representation units $E_i$ then depend on the error of the prediction $\hat{A}_i$ (difference between the prediction $\hat{A}_i$ and the image $I_i$ or the pooled state $R_{i-1}$). The computation is completely described in Algorithm 1.

3) Training of the Model: The model is trained by minimizing weighted prediction errors through the time and hierarchy [10]. The loss function is defined as

\begin{equation}
L_{\text{train}} = \sum_{m=1}^{M} L_{\text{seq}}(m)
\end{equation}

\begin{equation}
L_{\text{seq}}(j) = \sum_{t=1}^{l} \mu_j \sum_{i=0}^{n_t} \bigg[ \sum_{l=0}^{n_t} \lambda_j \sum_{l=1}^{n_l} E^j_l(i) \bigg]
\end{equation}

where $L_{\text{seq}}(m)$ is loss of the $m$th sequence, $E^j_l(i)$ is the error of the $i$th unit in the module $l$ at time $t$, $M$ is a number of image sequences, $l_j$ is a length of a sequence, $N + 1$ is a
number of modules, \( \mu_t \) and \( \lambda_t \) are time and module weighting factors, respectively, and \( n_t \) is the number of error units in the \( l \)th module. The mini-batch gradient descent was used for the minimization.

C. Comparison of PreCNet With Other Models

We will compare our model with the predictive coding schema [2], the FIPC model [14], and PredNet [10].

1) Comparison of PreCNet and Rao and Ballard Model: PreCNet uses the same schema as the model by Rao and Ballard (see Fig. 1 and Section III-A). However, as PreCNet is couched in a modern deep learning framework and uses video sequences as inputs, there are inevitably some differences. The crucial differences are as follows.

1) Dynamic versus static inputs: In contrast with PreCNet and image sequences as inputs, the model by Rao and Ballard takes static images as inputs. An extension to next-frame video prediction should be possible [47].

Algorithm 1: Calculate PreCNet States at Time \( t \), Assume \( N > 0 \). Merging of States A and B is Denoted by Putting Them Into Curly Brackets \{A, B\}. See Supplementary Materials—convLSTM for a Detailed convLSTM Description

Require: Image \( I_t \), previous \( (t-1) \) hidden and cell states \( R^{t-1}_l, C^{t-1}_l \) of the representation layers \( l \in \{0,1,\ldots,N\} \), previous error state \( E^t_l \) of the (top) module \( N \), maximum pixel value \( \text{pix}^{\text{max}} \). For \( t = 1 \), the previous states \( R^0_l, C^0_l, E^0_N \) are initialized to zero.

for \( l = N, N - 1, \ldots, 0 \) \{Iterate top-down through the modules\} do
  if \( l == N \) \{Update the states in the top module\} then
    \( R^l_l, C^l_l \leftarrow \text{convLSTM}_{\text{down}}^l(R^{l-1}_l, C^{l-1}_l, E^{t-1}_l) \)
    \( \hat{A}^l_l \leftarrow \text{conv}_V(R^l_l) \)
    \( E^l_l \leftarrow \text{ReLU}(\hat{A}^l_l - \text{pool}(R^{l-1}_l), \text{pool}(R^{l-1}_l) - \hat{A}^l_l) \)
  if \( l \neq N \) and \( l \neq 0 \) \{Update the states in the “middle” module \( l \)\} then
    \( R^l_l, C^l_l \leftarrow \text{convLSTM}_{\text{down}}^l(R^{l-1}_l, C^{l-1}_l, \text{upsample}(E^{t+1}_l)) \)
    \( \hat{A}^l_l \leftarrow \text{conv}_V(R^l_l) \)
    \( E^l_l \leftarrow \text{ReLU}(\hat{A}^l_l - \text{pool}(R^{l-1}_l), \text{pool}(R^{l-1}_l) - \hat{A}^l_l) \)
  if \( l == 0 \) \{Update the states in the bottom module\} then
    \( R^l_l, C^l_l \leftarrow \text{convLSTM}_{\text{down}}^l(R^{l-1}_l, C^{l-1}_l, \text{upsample}(E^{t+1}_l)) \)
    \( \hat{A}^l_l \leftarrow \text{min}(\text{conv}_V(R^l_l), \text{pix}^{\text{max}}) \)
    \( E^l_l \leftarrow \text{ReLU}(\hat{A}^l_l - I_t, I_t - \hat{A}^l_l) \)

for \( l = 0, 1, \ldots, N \) \{Iterate bottom-up through the modules\} do
  if \( l == 0 \) then
    \( R^l_l, C^l_l \leftarrow \text{convLSTM}_{\text{up}}^l(R^l_l, C^l_l, E^l_l) \)
  if \( l \neq 0 \) and \( l \neq N \) then
    \( E^l_l \leftarrow \text{ReLU}(\hat{A}^l_l - \text{pool}(R^{l-1}_l), \text{pool}(R^{l-1}_l) - \hat{A}^l_l) \)
    \( R^l_l, C^l_l \leftarrow \text{convLSTM}_{\text{up}}^l(R^l_l, C^l_l, E^l_l) \)
  if \( l == N \) then
    \( E^l_l \leftarrow \text{ReLU}(\hat{A}^l_l - \text{pool}(R^{l-1}_l), \text{pool}(R^{l-1}_l) - \hat{A}^l_l) \)
3) **Different update of representation states:** Representation layer states of the Rao and Ballard model are determined by a first-order differential equation. The states are updated until they converge. In contrast, PreCNet’s representation layer states are calculated by convLSTM. To update the representation states $r$ of the model by Rao and Ballard, the “bottom-up” difference between the prediction of the PE and the actual input $[I - f(Ur)]$ in Fig. 1(a) followed by fully connected layer and the “top-down” difference between the predicted state by the higher PE and the actual state of the PE [$r^{d} - r$ in Fig. 1(a)] are used simultaneously. PreCNet also uses both differences for computation of the new representation states $r$, however, not simultaneously; one difference is used by the convLSTM$^{\text{down}}$ during the prediction phase, the second is used by the convLSTM$^{\text{up}}$ during the correction phase (notice that the convLSTM$^{\text{down}}$ and convLSTM$^{\text{up}}$ share cell and hidden unit states).

4) **One versus multiple PEs on one level:** There are multiple PEs in one level of the model by Rao and Ballard. Higher-level PEs progressively operate on bigger spatial areas than lower-level PEs. PreCNet has one PE at each level of the hierarchy.

5) **Intensity of interaction between the PEs:** Each PE of PreCNet is updated just two times during one timestep (one input image). Once during the prediction (top-down) phase and once during the correction (bottom-up) phase. This means that each PE interacts with its neighbor just two times during one timestep. On the contrary, the PEs of the model by Rao and Ballard interact with each other many times (until their representation states converge) during one timestep. As PreCNet uses the deep learning approach, which is more computationally demanding, such intensive interaction between the PEs is not possible.

6) **Minimizing error in all levels versus only the bottom level error:** Errors in all levels of the model by Rao and Ballard are minimized. However, PreCNet has achieved better results when only the bottom level error—the difference between the predicted and the actual image—was minimized (see the setting of parameter $\lambda_i$ in Section IV-C2).

2) **Comparison of PreCNet and FIPC:** As FIPC [14] is mainly an extension of the original Rao and Ballard model by a procedure for regression mapping with fast inference at test time, it shares many properties with the Rao and Ballard model. The most important differences between PreCNet and FIPC are as follows.

1) **Different building blocks:** Main building blocks of PreCNet are convolutional and convolutional LSTM networks. FIPC’s main basic building blocks, similar to the Rao and Ballard model, are simple feedforward networks. This might be limiting for usage on large-scale images and video sequences.

2) **Dynamic versus static inputs:** PreCNet takes image sequences as inputs and predicts the next frames. FIPC, identically to Rao and Ballard model, works with static images and is trained for their classification and feature representation.

3) **Fast inference at test time:** During testing, the trained network works as a feedforward network (a subset of weights is used) with class labels as outputs. Therefore, during test time, the network does not follow the predictive coding schema.

4) **Intensity of interaction between the PEs during training:** FIPC PEs, similar to the Rao and Ballard model, interact with each other many times during one training timestep. For PreCNet, it is only two times (see Section III-C1).

5) **Classification layer:** FIPC added to the predictive coding schema, a classification layer which helps to learn more discriminative features for a given classification task. On the other hand, PreCNet is completely self-supervised.

3) **Comparison of PreCNet and PredNet:** PredNet, a state-of-the-art deep network for next-frame video prediction [10], is also inspired by the model by Rao and Ballard. PredNet and PreCNet (which we propose) are similar in these aspects.

1) **Building blocks:** Error representations, convolutional, and convolutional LSTM networks.

2) **Training procedure:** For the next-frame video prediction task, most training parameters, such as input sequence length and batch size, of PreCNet are taken from PredNet.\(^4\)

3) **Number of trainable parameters:** For training on the KITTI Dataset, PreCNet had approximately 7.6M and PredNet 6.9M trainable parameters. We tested also PreCNet-small with 0.8M parameters (see Table III for results).

However, there are two crucial properties in which PredNet departs from the predictive coding schema [see Fig. 3(a), (c), and (d)]:

1) According to the predictive coding schema, except for the bottom PE, each PE outputs a prediction of the next lower-level PE activity $r$ (representation layer state). See Section III-A.

2) No direct connection between two neighboring PE activities $r_i$ and $r_{i-1}$ (representation layer states $R_i$ and $R_{i-1}$ in formalism of [10]).

Instead, to remain faithful to the predictive coding schema, the building blocks of PreCNet were connected in a significantly different way (see Fig. 3 for comparison). These modifications have led to considerably better performance of PreCNet in comparison with PredNet (see Section IV-C4).

**IV. Experiments**

In this section, the datasets and performance measures are introduced, followed by experiments on the next-frame and multiple-frame video prediction. Trained models and code needed for replication of all the results presented in the article.

\(^4\)The motivation was twofold. First, we wanted to make it clear that the significant improvement of PreCNet over PredNet is not caused by a better choice of training parameters. Second, few trials with other parameter values that we tried did not lead to significantly better results.
Fig. 3. Comparison of PredNet and PreCNet. In (a) and (b), the differences (connections between the blocks, some building blocks) are highlighted. In (c), (d), and (e), there is a comparison of the PEs of PredNet, PreCNet, and the model by Rao and Ballard. Notice that the input from above in (d), (e)—prediction $r^{kd}$ of $r$—is compared with the representation state $r$ and the error is used for the update of the $r$. The corresponding upper input (blue) of the PredNet is a different entity; it is not related to $r$ and is also compared with a different entity (Conv$(E_0)$). There is also one more input from above—representation layer state from above—which goes directly into ConvLSTM block of PredNet. PreCNet [see (e)] has overcome these differences and follows the same predictive coding scheme as the model by Rao and Ballard. Notice the correspondence of the olive, purple polygons [(a), (b)] and the PEs of PredNet and PreCNet [(c), (e)]. To enhance comprehensibility, some of the labels from (a), (b) were added to (c), (d), (e) and v.v. See Supplementary materials—Schema transformation to check the correspondence between both [(a), (b) and (c), (e)] ways of visualization.

A. Datasets

All datasets used are visual sequences obtained from a car-mounted camera. These scenes include fast movements of complex objects (e.g., cars, pedestrians), new objects coming unexpectedly to the scene, as well as the movement of the urban background.

For training, we used two different datasets: KITTI [49] and BDD100K [50]. For evaluation, we used the Caltech Pedestrian Dataset [51], [52], employing Piotr’s Computer Vision MATLAB Toolbox [53] during preprocessing. Using the Caltech Pedestrian Dataset for establishing performance enables direct comparison of the models from both training variants.

1) KITTI Dataset and its preprocessing: We followed the preprocessing procedure from [10]. The frames were center-cropped and resized with the bicubic method to 128 by 160 pixels size (see the repository for code). We also followed the division categories “city,” “residential,” and “road” of the KITTI Dataset to training (57 recording sessions, approx. 41K of frames) and validation parts in the same way as in [10]. The dataset has a 10 fps frame rate.

2) Caltech Pedestrian Dataset and its preprocessing: Frames were preprocessed in the same way as the frames of the KITTI Dataset (see above). Videos were downsampled from 30 to 10 fps (every third frame was taken). As this dataset was used only for evaluation of the performance, only testing parts (set06–set10) were used (approx. 41K of frames).

3) BDD100K and its preprocessing: The preprocessing of the dataset was analogous to the preprocessing of the Caltech Pedestrian Dataset, including reducing the frame rate from 30 to 10 fps. As the size of the whole dataset is very large (roughly 40M frames if 10 fps is used), we had to randomly choose training and validation subsets of the dataset—see the repository for details and chosen videos. We created two variants of the training dataset; a big one with roughly 2M frames (5000 recording sessions) and a small one with a similar size to the KITTI training dataset (approx. 41K frames, 105 recording sessions). As a validation dataset, we randomly selected a subset of the validation part of BDD100K with approx. 9K frames.

B. Performance Measures

For comparison of a predicted with the actual frame, we use standard measures: mean square error (MSE), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM) [54]. MSE is a simple measure whose low values indicate high similarity between frames. PSNR is a related measure to MSE whose value is desired to be as high as possible. A significant limitation of these two is that their evaluation of similarity between two images does not correlate very well with human judgment (e.g., [55], [56]). SSIM was created to be more correlated with human perception. SSIM values are bounded to $[-1, 1]$ and a higher value signifies higher similarity.

C. Next-Frame Video Prediction

First, the settings of experiments and parameters will be described. This is followed by quantitative results and qualitative analysis. Results, achieved by PreCNet, presented in this subsection can be generated by publicly available code [11]. Summary and details of the network parameters and training are in Supplementary materials—1 Network and training parameters summary and details.
1) Experimental Settings: We performed experiments with two settings. In both, the performance of trained models was measured using Caltech Pedestrian Dataset (see Section IV-A) which is commonly used for evaluating the next-frame video prediction task. This also enabled a direct comparison of training on both datasets. The training was done on the following.

1) KITTI Dataset: This setting (i.e., KITTI for training, and the Caltech Pedestrian Dataset for evaluation) is popular for evaluation of the next-frame video prediction task and enables good comparison with other state-of-the-art methods.

2) BDD100K Dataset: Randomly chosen subset of the dataset (approx. 2M of frames) was used. The training dataset is significantly larger than the KITTI Dataset which enables avoiding overfitting. We also performed training on a smaller BDD100K subset with the roughly same size as the KITTI training dataset.

2) Network Parameters: The main parameters of the network are summarized in Table I. In the table, the parameters of each module in the hierarchy are described in a row. Module weights are in the second column. The following columns contain the number of channels #chan, (layer size) and filter sizes of decoding (conv) and representation (convLSTM) layers. For a detailed explanation, see Section III-B.

For choosing a suitable number of hierarchical modules, layer sizes (number of channels), and module weight factors ($\lambda_i, i \in \{0, 1, 2\}$), the KITTI Dataset was used for training. We performed a manual heuristic parameter search to minimize mean absolute error (between the predicted and actual frames) on the validation set. Padding was used to preserve the size in all convolutional layers (including convLSTM). Values of the pixels of the input frames were divided by 255 to make them in the range [0, 1]. The filter sizes were taken from [10] (for explanation of this choice, see Section IV-C3).

To better understand how the number of trainable parameters affects the performance and better comparison with PredNet, we proposed the same model but changed the number of channels in the modules from 60, 120, 240 to 20, 40, and 80, respectively (PreCNet-small). The number of parameters was reduced from 7.6M to 0.8M. Moreover, we simplified the architecture by: i) replacing all pairs of convolutional LSTMs with shared hidden and cell states—convLSTM up, convLSTM down—by single convolutional LSTMs—convLSTM (PreCNet-single-LSTMs) and ii) by simplifying error blocks ReLU([PREDICTION–ACTUAL, ACTUAL–PREDICTION]) (see Algorithm 1) to residual errors PREDICTION–ACTUAL only (PreCNet-residual-error). It was also necessary to modify the sequence loss (2) by putting the error values $E_i(i)$ into absolute value.

3) Training Parameters: Except for training length and learning rate, all the values of the training parameters were the same as in [10] (see Section III-C3 for the explanation). The network was trained on input sequences with length $l_i = 10$. In the sequences used for training and validation, a frame was generally present in more sequences, meaning that the sequences overlap. During learning, the error related to the first predicted input is ignored ($\mu_{t=1} = 0$), since the first prediction is produced before seeing any input frame. Prediction errors related to the following timesteps are equally weighted ($\mu_t = (1/(l_i - 1), \text{for } t \in \{2, \ldots, l_i\})$).

In each epoch, 500 sequences from the training set were randomly selected to form batches of size 4 and used for weight updates. For validation, 100 randomly selected sequences from the validation set were used in each epoch. We used Adam [57] as an optimization method for stochastic gradient descent on the training loss (1). The values of the Adam parameters $\beta_1, \beta_2$ were set to their default values ($\beta_1 = 0.9, \beta_2 = 0.999$).

Training parameters for training on both datasets were very similar except for the number of training epochs and learning rate setting. For the KITTI and BDD100K training, the learning consists of 1000 and 10000 epochs, respectively. The learning rate was set to 0.001 and 0.0005 for the first 900 and 9900 epochs, respectively. Then, it was decreased to 0.0001 for last 100 epochs. As the BDD100K training set is significantly larger than the KITTI training set, the training was longer for BDD100K. The choice of the length of the training and learning rate was based on the evolution of validation loss and limited computational resources. It means that validation loss still slightly decreased at the final epochs, however, the benefit was not so significant to continue training and use (limited) computational resources.

4) Quantitative Results: For a quantitative analysis of the performance of the model, we used a standard procedure and measures for evaluating the next-frame video prediction. The network obtained a sequence (from the Caltech Pedestrian Dataset) of length 10 and then predicted the next frame (see Fig. 4 for details). Contrary to training and validating sequences, there was no overlap between the two testing sequences of length 11. This frame is compared to the actual frame using MSE, PSNR, and SSIM (see Section IV-B). The overall value of each measure is then obtained as a mean of the calculated values for each predicted frame.

We performed ten training repetitions on the KITTI Dataset (see Section IV-C1). The results are summarized in Table II. The results show that the learning is stable. Moreover, we carried out one training repetition of PreCNet-small, PreCNet-residual-error, and PreCNet-single-LSTMs.

We took the best model of ten repetitions (according to SSIM) and compared it with state-of-the-art methods

| TABLE I | NETWORK PARAMETERS SUMMARY. SEE TEXT FOR A DESCRIPTION |
|---------|---------------------------------------------------|
| module | weight $\lambda_i$ | #chan | filter size | #chan | filter size |
| i=0    | 1                  | 3      | 3           | 60    | 3           |
| i=1    | 0                  | 60     | 3           | 120   | 3           |
| i=2    | 0                  | 120    | 3           | 240   | 3           |

---

4) If $\lambda_0 = 1, \lambda_{1,2} = 0$, then the mean absolute error between the predicted and actual frames corresponds to $2^{\text{loss value (2)}}$. This is a consequence of the division of error representation into negative and positive parts and using of ReLU. For nonzero $\lambda_{1,2}$, this does not hold.

5) Learning rate setting 0.001 for BDD100K training led in two of four cases to the rapid increase of training loss in later stages of training. Therefore, the learning rate was changed to 0.0005.
of next-frame video prediction (see Section IV-C).

This schema was used for quantitative and qualitative analyses where the predicted frame is compared—using MSE, PSNR, and SSIM—with the actual input. However, “PreCNet model were calculated after only seven, four, three, and nine (2–10) timesteps. Values for “PreCNet 7 input frames” and RC-GAN had better performance in this case than for input sequences of length ten. If this is true also for DPRACconvLSTM and Lu et al. [26] model is not known.

PreCNet achieved second–third position in SSIM. In MSE and PSNR, it was outperformed by four and seven other methods, respectively. The number of trainable parameters—for the models where it is available—is similar for all except DM-GAN (113M) and PreCNet-small (0.8M). PreCNet with a small number of parameters still had comparable performance to other models and outperformed PredNet. Replacement of the pairs of convLSTM$^{up/dn}$ by single convLSTM, and simplification of error blocks degraded the performance only slightly.

Moreover, we took the best-trained model and investigated its performance for shorter input testing sequences. The results are shown in Table IV. The network received input sequences with the given input length and predicted the next frame. Except for the input sequence length, the experiential setting and the PreCNet network are the same as in Table III (the values for input length 10 are the same as in Table III). Copy last MSE is not the same for all input lengths because the test set was split into nonoverlapping sequences with different lengths. The performance was significantly worse for input length 3 and slightly worse also for length 4. For longer sequences, it was stable. For input length 10, the performance even slightly decreased. A possible reason could be that the network was trained on sequences with length 10 and, therefore, it was not directly trained to predict the 11th frame after inputting ten frames. To investigate the influence of sequence length during training, we trained a network with $l_s = 5$—half of the basis sequence length—and evaluated it on the sixth frame after inputting five frames. We performed two repetitions with 1000 epochs and two repetitions with 2000 epochs. The results in all four cases were similar; SSIM was 0.924 in all cases, PSNR varied from 28.2 to 28.4, and MSE was between 0.00169 and 0.00209. The usage of a larger dataset led to significant performance improvement in all three measures. Comparing PreCNet trained on a large

(see Table III). In the table, the methods are sorted according to their SSIM values. If not stated otherwise, a network received ten input images and predicted the next one which was used during performance evaluation. Unless otherwise stated, the values were taken from the original articles. Values for BeyondMSE were taken from [38]; values for DVF and CtrlGen were taken from [27]. Values for PredNet were taken from [24], because in [10], the values were averaged over nine (2–10) timesteps. Values for “PreCNet 7 input frames” (see Table IV), RC-GAN, DPRACconvLSTM, and Lu et al. [26] model were calculated after only four, three, and two input images (not ten), respectively. However, “PreCNet 7 input frames” and RC-GAN had better performance in this case than for input sequences of length ten.
Fig. 5. Qualitative comparison of PreCNet with other state-of-the-art methods on the Caltech Pedestrian Dataset. All models were trained on the KITTI Dataset. Ten input frames were given (see frames for $t = 8$, $t = 10$), and the next one ($t = 11$) was predicted (RC-GAN used only four input frames—see Section IV-C4 for explanation) by the models (for references, see Table III). The images of predictions of other models are copied from original or other cited articles (see references in Table III). The position of the sequences in the Caltech Pedestrian Dataset by rows; set06-v013, set06-v000, set07-v011, set10-v010, set10-v009, set10-v010, and set06-v009.

TABLE V

| Training Set | Caltech Pedestrian Dataset | MSE | PSNR | SSIM |
|--------------|---------------------------|-----|------|------|
| BDD100K      | 2M | 10000  | 0.00167 | 29.4 | 0.938 |
| BDD100K      | 41K | 1000  | 0.00201 | 28.6 | 0.926 |
| KITTI        | 41K | 1000  | 0.00205 | 28.4 | 0.929 |

BDD100K subset (2M) with the models trained on the KITTI Dataset (see Table III), our model was second in SSIM and third in PSNR and MSE.

To evaluate the effect of different properties of the BDD100K and KITTI Datasets on performance, we created a small version of the BDD100K Dataset with only approx. 41K frames (similar size as the size of KITTI) and used the same training parameters which were used for training on KITTI. The performance on this dataset was similar to the performance on KITTI. This suggests that the “quality” of the training set (BDD100K vs. KITTI) is not the key factor for obtaining better performance in this case. We studied the effect of the number of training epochs as well. Validation loss on the small subset of BDD100K (41K frames) started to increase during training (1K epochs), indicating overfitting. Thus, we can exclude the possibility that training for 10K epochs would further improve performance. Hence, we claim that it is really the
dataset size that is the enabling factor for performance and that permitted the results obtained for BDD100K (2M frames, 10K epochs).

5) **Qualitative Analysis:** In Fig. 5, there is a qualitative comparison of PreCNet with other state-of-the-art methods trained on the KITTI Dataset (see Table III). The way of obtaining the predicted frames used for the analysis is the same as for quantitative analysis (see the predicted frame at \( t = 11 \) in Fig. 4). For a qualitative comparison of PreCNet with the model by Jin et al. [40], see Section IV-D2 (the predicted frame at \( t = 11 \)).

To assess which of the methods is best through visual inspection is not straightforward, none of the models is better than the others in all aspects and shown frames (excluding PredNet which produced significantly worse predictions). For example, in the last row of Fig. 5, DPG has generally the sharpest prediction but PreCNet predicted the street lamp significantly better. To compare our model with the IPRNN, which significantly outperformed all models in all metrics used, we used the sequences from the IPRNN article [33]. On these sequences (see Fig. 5, first two rows), there is no apparent qualitative difference between the predictions by IPRNN and PreCNet. For example, “STOP” sign in the sequence from the first row is predicted sharper by PreCNet than by IPRNN and the other models.

In Fig. 6, KITTI and BDD100K (both 2M and 41K) training variants (see Table V) are compared. Usage of a large BDD100K Dataset (with approx. 2M frames) for training led to significant improvement of all the measures (see Table V) in comparison with training on KITTI Dataset. It manifested also in the visual quality of the prediction of fast-moving cars as you can see in the second and third columns of the figure. The phantom parts of the predicted cars were reduced. It also led to better shapes of the predicted cars as you can see in the prediction in the first column (focus on the front part of the van). On the other hand, in some cases, training on the BDD100K Dataset led to blurrier predictions than on KITTI (see the last column).

**D. Multiple-Frame Prediction**

For multiple-frame prediction, we used the same trained models which we used for next-frame video prediction (see Section IV-C). The network had access to the first ten frames—same as in the next-frame video prediction. Then, in each timestep, the network produced the next frame and this next frame was used as the actual input (as illustrated in Fig. 7). Therefore, the prediction error between the prediction and input frame was zero.

We briefly explored fine-tuning the network for multiple-frame prediction [10]. To generate multiple future frames, the predicted frames—produced by the pretrained network for next-frame prediction—were used as inputs after inputting ten frames. The network was trained to minimize the mean absolute error between the predicted and ground-truth frames. According to our preliminary results, this did not significantly improve multiple-frame prediction performance.

Note the different meaning of timestep labels \( t \) and \( T \): small \( t \) starts at the beginning of a sequence, in contrast with a capital \( T \), which starts at the beginning of a predicted sequence (see the timestep labels in Fig. 7). The code needed for the generation of the results presented is publicly available [11].

1) **Quantitative Results:** In Table VI, there is a quantitative comparison of PreCNet, PredNet, CrevNet, and RC-GAN for multiple-frame prediction. The methods obtained sequences with a fixed length (ten for PredNet, CrevNet, and PreCNet; four for RC-GAN; see Section IV-C4 for explanation) of Caltech Pedestrian Dataset and outputted predictions 15 steps ahead (CrevNet only 12). CrevNet, RC-GAN, PredNet, and PreCNet (KITTI) were trained on KITTI. PreCNet was also trained on a subset of BDD100K with size 2M. Values for PredNet and RC-GAN were copied from [37]. Values for CrevNet were taken from [32]. Some values for \( T = 1 \) from Tables III and V are slightly different because the test set used there was split into nonoverlapping sequences with different length (11 vs. 25).
Fig. 7. Multiple-frame video prediction evaluation schema. After inputting ten frames, the predicted frames are inputted instead of the actual frames. The prediction errors are therefore zeros. The predicted frames are compared—using MSE, PSNR, and SSIM—with the actual inputs. We used this schema for both quantitative and qualitative analyses of multiple-frame prediction (see Section IV-D).

Fig. 8. Qualitative comparison of selected methods for multiple-frame prediction. The methods obtained sequence with fixed length (ten for PredNet, CrevNet, and PreCNet, four for RC-GAN; see Section IV-C4 for explanation) of the Caltech Pedestrian Dataset and outputted predictions 15 steps ahead. RC-GAN, PredNet, CrevNet, and PreCNet (KITTI) were trained on KITTI. PreCNet was also trained on a subset of BDD100K with size 2M. This should be noticed during comparison with the other four trained models. This figure was obtained from the figure from [37] by adding sequences for PreCNet and CrevNet (taken from [32]). The location of the sequence in the Caltech Pedestrian Dataset is set10-v009. Another qualitative comparison (without CrevNet), with a different sequence, is in Supplementary materials—multiple-frame video prediction sequence.

For SSIM, PreCNet trained on KITTI outperformed PredNet until timestep \( T = 9 \) (\( t = 19 \)) when the values became equal and then PreCNet started to lose. For PSNR, PreCNet started to lose earlier (\( T = 6 \)). RC-GAN and CrevNet outperformed PreCNet in nearly all timesteps for SSIM\(^{10}\) and RC-GAN also in all timesteps for PSNR.

\(^{10}\)In \( T = 1 \), SSIM for PreCNet was 0.930 and for CrevNet 0.925.
We also added PreCNet trained on the large subset of BDD100K to the comparison. Then PreCNet outperformed PredNet in all timesteps for SSIM and most timesteps for PSNR; in timestep $T = 15$, it reversed. However, CrevNet and RC-GAN still outperformed PreCNet in most timesteps. For SSIM, PreCNet had better results than CrevNet and RC-GAN only for predicted frames in $T \in \{1, 3\}$. For PSNR, RC-GAN was outperformed by PreCNet only for $T = 1$.

In summary, PreCNet started with mostly better predictions than its competitors; however, its performance tended to degrade faster for prediction further ahead.

2) Qualitative Analysis: The methods were compared using the sequences used in [37]. Fig. 8 provides one example (for another illustration, see Supplementary Materials—Multiple-frame video prediction sequence). Moreover, PreCNet was separately compared to the model by Jin et al. [40] (Fig. 9). Predictions by PreCNet appear less blurred than those by PredNet and by the model of Jin et al. [40] (see Fig. 9). This is especially apparent for the later predicted frames. Compared to RC-GAN, predicted frames by PreCNet trained on KITTI seem to have more natural colors and the background is mostly less blurred (focus on the buildings in the background). PreCNet trained on a large subset of BDD100K (2M of frames) produced even less blurred frames. Comparison with CrevNet is not straightforward. For example, CrevNet captured the geometry of the shadow of the building on the road better than PreCNet. On the other hand, it produced a phantom object (see right side of the road in timesteps 9, 11) which is not present (or negligible) in the corresponding frames by PreCNet.

V. CONCLUSION AND DISCUSSION

In this work, the seminal predictive coding model of Rao and Ballard [2]—here referred to as predictive coding schema—has been cast into a modern deep learning framework, while remaining as faithful as possible to the original schema. The similarities and differences are elaborated on in detail. We also claim and explain that the network we propose (PreCNet) is more congruent with [2] than others based on the deep learning framework that takes inspiration from predictive coding; the case of PredNet [10] is studied explicitly. PreCNet was tested on a widely used next-frame video prediction benchmark—KITTI for training (41k images), and the Caltech Pedestrian Dataset for testing—which consists of images from an urban environment recorded from a car-mounted camera. On this benchmark, we outperformed most of the state-of-the-art methods and achieved second–third rank when measured with the structural similarity index (SSIM)—a performance measure that should best correlate with human perception. Performance on all three measures was further improved when a larger training set (2M images from BDD100k; to our knowledge, the biggest dataset ever used in this context) was employed. This may suggest that the current practice based on the rather small KITTI Dataset used for training may be limited in the long run. At the same time, the task itself seems highly relevant, as a virtually unlimited amount of data and without any need for labeling is readily available.

Below, we discuss some limitations of this work. For some fast-moving objects in the scene, PreCNet could not restore their structure precisely (see, e.g., the third column of Fig. 6 where the car contours are not preserved). This may be a drawback of the cost function that minimizes the per-pixel loss. Perceptual loss (e.g., [58]) based on high-level feature differences between frames might alleviate this problem.

In multiple-frame video prediction, qualitatively, the frames predicted by PreCNet look reasonable and in some aspects better than some of the competitors. However, a quantitative comparison reveals that PreCNet performance degrades slightly faster than that of its competitors when predicting up to 15 frames ahead. We speculate that architectures that achieve multiple-frame prediction by recurrent feeding of previous predictions may not achieve their best performance for next- and multiple-frame predictions at the same time. Increasing performance for multiple-frame prediction may decrease performance for next-frame prediction and vice versa. We performed fine-tuning of our network for multiple-frame prediction, but preliminary results did not show any significant improvement. PreCNet and predictive coding, in general, is perhaps intrinsically more suited for next-frame prediction. This remains to be further analyzed.

In the future, we plan to analyze the representations formed by the proposed network. It would be interesting to study how much of the semantics of the urban scene has the network “understood” and how that is encoded. For example, our
A network has not quite figured out that every car has a finite length and its end should be predicted at some point when it is not occluded anymore. In our model, the best results on the task were achieved when only prediction error on the bottom level—the difference between the actual frame and the predicted one—was minimized during learning. Rao and Ballard [2], on the other hand, minimized this error on every level of the network hierarchy, which may have an impact on the representations formed. Testing on a different task, like human action recognition (e.g., [10], [38], [39]) is also a possibility. Finally, some datasets feature also other signals apart from the video stream. Adding inertial sensor signals or the car’s steering wheel angle or throttle level is another avenue for future research.

We want to close with a discussion of the implications of our model for neuroscience. Casting the predictive coding schema into a deep learning framework has led to an exceptional performance on a contemporary task, without being explicitly designed for it. In the future, we plan to analyze the consequences of computational neuroscience. While receptive field properties in sensory cortices remain an active research area (e.g., [59]), a question remains whether the deep learning approach can lead to a better model than, for example, that of Rao and Ballard [2]. Richards et al. [60] and Lindsay [61] provide recent surveys of this perspective. An investigation of this kind has recently been performed for PredNet [23].

ACKNOWLEDGMENT

The authors gratefully acknowledge access to the computational infrastructure available through the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 Research Center for Informatics. They would also thank the authors of PredNet [10] for making their source code public which significantly accelerated the development of PreCNet.

REFERENCES

[1] Y. Huang and R. P. N. Rao, “Predictive coding,” Wiley Interdiscip. Rev.: Cognit. Sci., vol. 2, no. 5, pp. 580–593, Sep./Oct. 2011.
[2] R. P. Rao and D. H. Ballard, “Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects,” Nature Neurosci., vol. 2, no. 1, p. 79, 1999.
[3] M. W. Spratling, “Reconciling predictive coding and biased competition models of cortical function,” Frontiers Comput. Neurosci., vol. 2, p. 4, Oct. 2008.
[4] G. Stefanics, J. Kremlácek, and I. Cigzler, “Visual mismatch negativity: A predictive coding view,” Frontiers Hum. Neurosci., vol. 8, p. 666, Sep. 2014.
[5] C. Summerfield and T. Egner, “Expectation (and attention) in visual cognition,” Trends Cognit. Sci., vol. 13, no. 9, pp. 403–409, Sep. 2009.
[6] M. W. Spratling, “Predictive coding as a model of response properties in cortical area V1,” J. Neurosci., vol. 30, no. 9, pp. 3531–3543, 2010.
[7] K. Friston, “Does predictive coding have a future?” Nature Neurosci., vol. 21, no. 8, pp. 1019–1021, Aug. 2018.
[8] A. Clark, “What next? Predictive brains, situated agents, and the future of cognitive science,” Behav. Brain Sci., vol. 36, no. 3, pp. 181–204, 2013.
[9] M. Mathieu, C. Couprie, and Y. LeCun, “Deep multi-scale video prediction beyond mean square error,” in Proc. 4th Int. Conf. Learn. Represent. (ICLR), 2016, pp. 1–14.
[10] W. Lotter, G. Kreiman, and D. Cox, “Deep predictive coding networks for video prediction and unsupervised learning,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR), 2017, pp. 1–18.
[11] (2020). Precnet Github Repository. [Online]. Available: https://github.com/ZeDenekStraka/precnet
[12] M. W. Spratling, “A review of predictive coding algorithms,” Brain Cogn., vol. 112, pp. 92–97, Mar. 2017.
[13] K. Friston, “A theory of cortical responses,” Philos. Trans. Roy. Soc. B, Biol. Sci., vol. 360, no. 1456, pp. 815–836, Apr. 2005.
[14] Z. Song, J. Zhang, G. Shi, and J. Liu, “Fast inference predictive coding: A novel model for constructing deep neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 4, pp. 1150–1165, Apr. 2019.
[15] M. W. Spratling, “A hierarchical predictive coding model of object recognition in natural images,” Cognit. Comput., vol. 9, no. 2, pp. 151–167, Apr. 2017.
[16] K. Han, H. Wen, Y. Zhang, D. Fu, E. Culurciello, and Z. Liu, “Deep predictive coding network with local recurrent processing for object recognition,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 9201–9213.
[17] H. Wen, K. Han, J. Shi, Y. Zhang, E. Culurciello, and Z. Liu, “Deep predictive coding network for object recognition,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 5266–5275.
[18] S. Dora, C. Pennartz, and S. Bohle, “A deep predictive coding network for inferring hierarchical causes underlying sensory inputs,” in Proc. Int. Conf. Artif. Neural Netw. Cham, Switzerland: Springer, 2018, pp. 457–467.
[19] O. Struckmeier et al., “MuPNet: Multi-modal predictive coding network for place recognition by unsupervised learning of joint visuo-tactile latent representations,” 2019, arXiv:1909.07201.
[20] A. Ahmad and J. Tani, “A novel predictive-coding-inspired variational RNN model for online prediction and recognition,” Neural Comput., vol. 31, no. 11, pp. 2035–2057, Nov. 2019.
[21] M. Choi and J. Tani, “Predictive coding for dynamic visual processing: Development of functional hierarchy in a multiple spatiotemporal scales RNN model,” Neural Comput., vol. 30, no. 1, pp. 237–270, Jan. 2018.
[22] R. Chalasani and J. C. Principe, “Deep predictive coding networks,” in Proc. Workshop Int. Conf. Learn. Represent., 2013, pp. 1–13.
[23] W. Lotter, G. Kreiman, and D. Cox, “A neural network trained for prediction mimics diverse features of biological neurons and perception,” Nature Mach. Intel., vol. 2, no. 4, pp. 210–219, Apr. 2020.
[24] W. Byeon, Q. Wang, R. Kumar Srivastava, and P. Koumoutsakos, “ContextVP: Fully context-aware video prediction,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 753–769.
[25] F. A. Reda et al., “SDC-Net: Video prediction using spatially-displaced convolution,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 718–733.
[26] X. Hu, J. Cui, Y. Zhang, and L. Zhang, “A video prediction method based on optical flow estimation and pixel generation,” IEEE Access, vol. 9, pp. 100395–100406, 2021.
[27] H. Gao, H. Xu, Q.-Z. Cai, R. Wang, F. Yu, and T. Darrell, “Disentangling propagation and generation for video prediction,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 9006–9015.
[28] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, “Video frame synthesis using deep voxel flow,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 4463–4471.
[29] Z. Hao, X. Huang, and S. Belongie, “Controllable video generation with sparse trajectories,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 7854–7863.
[30] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, “Learning to generate long-term future via hierarchical prediction,” in Proc. 34th Int. Conf. Mach. Learn., 2017, pp. 3560–3569.
[31] C. Finn, I. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 64–72.
[32] W. Yu, Y. Lu, S. Easterbrook, and S. Bidler, “Efficient and information-preserving future frame prediction and beyond,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–14.
[33] Z. Chang, X. Zhang, S. Wang, S. Ma, and W. Gao, “IPRNN: An information-preserving model for video prediction using spatiotemporal GRUs,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2021, pp. 2703–2707.
[34] M. Yuan and Q. Dai, “A novel deep pixel restoration video prediction algorithm integrating attention mechanism,” Appl. Intell., vol. 52, pp. 5015–5033, 2022.
[35] X. Shu, L. Zhang, G.-J. Qi, W. Liu, and J. Tang, “Spatiotemporal co-attention recurrent neural networks for human-skeleton motion prediction,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 6, pp. 3300–3315, Jun. 2022.
[36] R. Zhang, X. Shu, R. Yan, J. Zhang, and Y. Song, “Skip-attention encoder–decoder framework for human motion prediction,” Multimedia Syst., vol. 28, no. 2, pp. 413–422, Apr. 2022.

[37] Y.-H. Kwon and M.-G. Park, “Predicting future frames using retrospective cycle GAN,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1811–1820.

[38] X. Liang, L. Lee, W. Dai, and E. P. Xing, “Dual motion GAN for future-flow embedded video prediction,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 1744–1752.

[39] C. Vondrick, H. Pirsiavash, and A. Torralba, “Generating videos with scene dynamics,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 613–621.

[40] B. Jin et al., “Exploring spatial–temporal multi-frequency analysis for high-fidelity and temporal-consistency video prediction,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 4554–4563.

[41] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, “Stochastic adversarial video prediction,” 2018, arXiv:1804.01523.

[42] M. Babaeizadeh, C. Finn, D. Erhan, R. Campbell, and S. Levine, “Stochastic variational video prediction,” in Proc. 6th Int. Conf. Learn. Represent. (ICLR), 2018, pp. 1–15.

[43] E. Denton and R. Fergus, “Stochastic video generation with a learned prior,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1174–1183.

[44] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 802–810.

[45] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[46] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space Odyssey,” IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[47] R. P. N. Rao and D. H. Ballard, “Dynamic model of visual recognition predicts neural response properties in the visual cortex,” Neural Comput., vol. 9, no. 4, pp. 721–763, 1997.

[48] R. P. N. Rao, “An optimal estimation approach to visual perception and learning,” Vis. Res., vol. 39, no. 11, pp. 1963–1989, Jun. 1999.

[49] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” Int. J. Robot. Res., vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[50] F. Yu et al., “BDD100K: A diverse driving dataset for heterogeneous multitask learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 2633–2642.

[51] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: A benchmark,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 304–311.

[52] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 4, pp. 743–761, Aug. 2012.

[53] P. Dollar. Piotr’s Computer Vision MATLAB Toolbox (PMT). [Online]. Available: https://github.com/pdollar/toolbox

[54] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, Apr. 2004.

[55] Z. Wang, A. C. Bovik, and L. Lu, “Why is image quality assessment so difficult?” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., vol. 4, May 2002, pp. 1–4.

[56] S. Winkler, “Perceptual distortion metric for digital color video,” Proc. SPIE, vol. 3644, pp. 175–184, May 1999.

[57] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–15.

[58] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 694–711.

[59] Y. Singer, Y. Teramoto, B. D. Willmore, J. W. Schnupp, A. J. King, and N. S. Harper, “Sensory cortex is optimized for prediction of future input,” elife, vol. 7, Jun. 2018, Art. no. e31557.

[60] B. A. Richards et al., “A deep learning framework for neuroscience,” Nature Neurosci., vol. 22, no. 11, pp. 1761–1770, Nov. 2019.

[61] G. Lindsay, “Convolutional neural networks as a model of the visual system: Past, present, and future,” J. Cognit. Neurosci., vol. 33, no. 10, pp. 1–15, 2020.