Modelling the Sequential Metacontrast Paradigm with Recurrent Neural Networks

Neil Scheidwasser-Clow
EPFL, Lausanne, Switzerland
January 11, 2020

Abstract—**The Sequential Metacontrast Paradigm (SQM) is a practical tool to study temporal feature integration by the human visual system during object detection. As more realistic to model biological object detection than feedforward convolutional neural networks (ffCNNs), recurrent neural networks (RNNs) were explored to simulate human feature integration using the SQM. In this project, five RNNs were tested: vanilla RNN, LSTM, Gated recurrent unit (GRU), convolutional LSTM and PredNet. To model the SQM, a reconstruction-decoder framework was implemented. Combining both reconstruction and decoding, convolutional LSTMs and PredNet were found to be the best performing recurrent networks. This suggests that convolutional layers remain necessary to predict object motion. That said, PredNet seemed to be most human-like when testing the network with the SQM.**

I. Introduction

Starting from the retina, the visual system can detect and interpret photon patterns to analyze the surrounding environment. Although most progress to automate human vision was recently made through deep feedforward convolutional neural networks (ffCNNs) [1] [2], their architecture is unable to simulate how the human visual system processes information [3]. In particular, object detection naturally involves neural feedback connections, e.g., through the ventral visual pathway [4]. In addition, object perception requires temporal integration to estimate features such as shape or motion direction [5] [6]. Thus, using recurrent neural networks seems like a more realistic approach to model human object detection. To study temporal feature integration, the sequential metacontrast paradigm (SQM) can be used [7] [8]. In this model, a central vernier is followed by a pair of flanking verniers, which then propagate in discrete time windows from the center in opposite directions (see Fig. 1). As a consequence, two diverging line streams are perceived.

With that in mind, this project aimed at building a recurrent neural network pipeline to simulate feature integration using the SQM. For that purpose, four "classical" recurrent networks (fully-connected RNN, LSTM [9], Gated Recurrent Unit (GRU) [10], and convolutional LSTM [11]) were tested and compared to PredNet [12], a recurrent convolutional neural network motivated by predictive coding [13]. Finally, the obtained predictions of the recurrent models to the SQM could be compared with human responses to the SQM, notably those experimented by Drissi-Daoudi et al. [14].

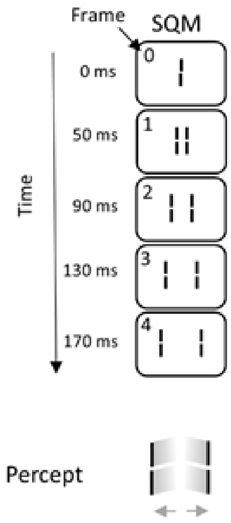


Fig. 1: The Sequential Metacontrast Paradigm (SQM). A central vernier (i.e., pair of lines) is followed by two flanking verniers, which propagate from the center in opposite directions. The resulting percept is two diverging streams of lines. Source: Adapted from [14]

II. Methods

To model the Sequential Metacontrast Paradigm (SQM), the following pipeline was designed:

- A wrapper class including:
  - A recurrent neural network chosen among the five described in II-B
  - A *reconstructor*, to learn on object motion (in this case, on motion of a pair of objects)
  - A *decoder*, to learn on vernier offset values
- A testing framework, to assess the performance of the whole pipeline using the SQM.

The pipeline was implemented using Tensorflow's Keras API [15], using Google Colab's free GPU to accelerate training.

A. Datasets

For each compartment of the pipeline, a different dataset was generated using Scikit-image, a Python collection of image processing algorithms [16]. Each dataset example had 10 frames representing successive motion of one or two pairs of objects (depending on the pipeline compartment). More specifically, each frame was implemented as a 32x64-pixel binary array, where one-pixels corresponded to coordinates of a shape.

1) General object motion: To simulate motion of a pair of objects along a given axis, the array containing the pair could be shifted using NumPy's *roll*. If the frame had two object pairs, both could be stored in separate arrays, and shifted simultaneously (in opposite directions) before being merged to form the next image. However, NumPy's function *roll* re-introduces elements that go beyond array boundaries, which is incompatible with natural object motion. To overcome this effect, object motion was stopped when a shape reached the image boundaries. For that purpose, image boundaries were zero-padded by a 1-pixel layer after each rolling. As a consequence, the number of non-zero pixels at each frame could be used. Indeed, if that count decreased after a new motion iteration compared to the previous count, it would mean that a shape reached the boundary. Thus, motion could be stopped, and empty frames could be added until the end of the sequence.

2) Data augmentation: Additive white Gaussian noise (AWGN) ($\sigma = 0.1$) was added to each frame as a data augmentation technique to reduce overfitting.

3) Reconstructor data: To learn on object motion, training the reconstructor was performed with four different shapes: circles, diamonds, triangles and lines. For each shape, 2,500 examples were generated with random shape size and initial position (uniformly generated at the vicinity of the image center), thus amounting to 10,000 images in total. Finally, object motion could occur in both horizontal and vertical axes, where speed along each axis was generated uniformly. Fig. 2 shows below an example from the reconstruction dataset.

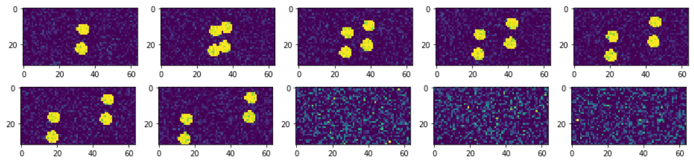


Fig. 2: Reconstruction data example, with circles (without the three initial blank frames). Here, the circles reached the image boundary at the seventh frame. Thus, three empty frames are present at the end.

Finally, to render the task less obvious to learn, the offset value between shape pairs oscillated around a given value (making the motion "fuzzy"), as shown in Fig. 3.

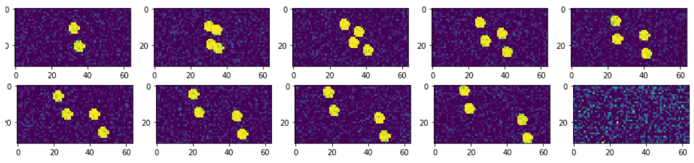


Fig. 3: Reconstruction data example, with circles (without the three initial blank frames). Here, the offset value oscillates at each frame, creating a "fuzzy" motion.

4) Decoder data: To train the decoder, only lines were used. A combination of two datasets was used to recognize both single verniers and vernier pairs:

- 10 empty frames, except 5 random frames with the same offset vernier.
- 10 empty frames, except 5 random frame(s) with the same vernier pair (see Fig. 4). More specifically, any of the three configurations could occur:
  - Left vernier is offset
  - Right vernier is offset
  - Both verniers are offset

Thus, at each epoch, the decoder would face either several frames of the same single vernier or the same vernier pair.

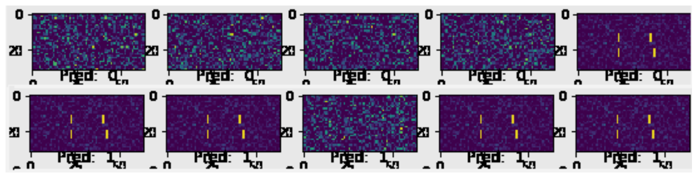


Fig. 4: Decoder data example, with a pair of verniers (without the three initial blank frames). Here, the same pair of verniers was shown at frames 5-6-7-9-10.

5) Testing data: The testing framework corresponded to the Sequential Metacontrast Paradigm (SQM). As explained earlier, a central vernier is followed by a pair of flanking verniers, which then propagate in discrete time windows from the center in opposite directions (see Fig. 1).

B. Models

To model the SQM, five models were tested in this project.

- A fully connected RNN
- A long-short term memory (LSTM): similar to RNN, but includes a memory cell state both to improve gradient flow and remember long-term information [9].
- A gated recurrent unit (GRU): a simplified version of LSTM [10] with two gating units (instead of four in the LSTM)
- A convolutional LSTM: an LSTM that where input and recurrent transformations are convolutional.
- PredNet, a recurrent convolutional network inspired by predictive coding [12].

In addition, two versions of PredNet were tested: a more "complex" PredNet (*large-prednet* in Fig. 6 and Fig. 7), with respectively 32 and 64 channels per layer in representation (R_l) and input layers (A_l), and a simpler version (*large-prednet-2* in Fig. 6 and Fig. 7), with respectively 8 and 16 channels per layer in representation and input layers.

C. Reconstructor

Excepting PredNet, which produces its own reconstructions, a reconstruction framework was necessary for the other recurrent networks to predict object motion. Here, the reconstruction model was implemented as a deconvolution network. In addition, the reconstruction framework was trained against a squared L2-norm function:

$$\mathcal{L}_{\mathcal{R}} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{R}_n)^2 \tag{1}$$

where $\boldsymbol{R}_n$ and $\boldsymbol{x}_n$ are respectively the predicted reconstruction and the real image at frame n and N the number of frames.

D. Decoder

To predict the offset direction (left or right) at each frame, the decoder was implemented as a two-layer densely connected network, with the latter layer being activated by SoftMax. Thus, output consisted of prediction probabilities for each offset orientation. Batch normalization was used prior to the dense layers to normalize the input layer activations. Training the decoder was performed against categorical cross-entropy loss:

$$\mathcal{L}_{\mathcal{D}} = -\sum_{i=1}^{N}\sum_{j=1}^{2} y_{i,j} log(s_{i,j}) \tag{2}$$

where $y_{i,j}$ and $s_{i,j}$ are respectively the ground truth the predicted probability for class j (0 = left, 1 = right) at frame i.

E. Training

Training was performed during 300 epochs for both reconstruction and decoding. For each network configuration, the best learning rate was determined using a learning rate finder inspired by Smith [17]. Adam [18] was used to optimize gradient descent.

F. Testing

After training both the reconstructor and decoder, testing of the pipeline was performed using the SQM. To evaluate performance, *central vernier dominance* (CVD) was calculated at each frame as the model tried to predict offset directions (left or right):

$$\forall i \in [\![1, N]\!]\; CVD_i = \frac{1}{D}\sum_{j=1}^{D} \mathbb{1}_{y_{i,0}=s_{i,j}} \tag{3}$$

Here, $\mathbb{1}$ is the indicator function, and $y_{i,j}$, $s_{i,j}$ are respectively the ground truth label and the predicted label at frame i for example j. In other words, CVD denotes the percentage of predicted labels agreeing with the ground truth label of the central vernier.

In particular, three SQM conditions (illustrated in Fig. 5) were assessed:

- Vernier (V): only the central vernier is offset.
- Vernier-antivernier (VAV): the central vernier and the left vernier at another frame (fifth frame) are offset in *opposite* directions. The other frames do not have any offsets.
- Vernier-provernier (VPV): the central vernier and the left vernier at another frame (fifth frame) are offset in *identical* directions. The other frames do not have any offsets.

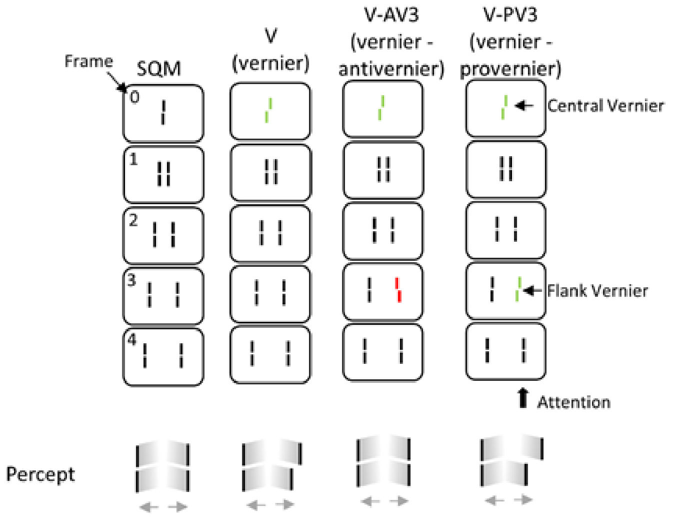


Fig. 5: The Metacontrast Paradigm (SQM): Vernier (V), Vernier-antivernier (VAV), and Vernier-provernier (VPV) paradigms. In the Vernier configuration, only the central vernier is offset. In the other paradigms, another pair of lines is offset, with the same value as the central vernier (VPV), or the opposite value (VAV). Source: Adapted from [14].

To have a good estimation of CVD, a network had to predict vernier directions for 1000 examples.

Thus, simulated CVD at the last frame could then be compared with the results obtained by Drissi-Daoudi et al. [14] with human subjects. At last, the entropy of the output layer of each model was also plotted as a marker for neural activity of the studied model.

III. Results

To begin with, Fig. 6 shows that reconstruction losses generally decreased for each network. More specifically, PredNet achieved lowest training loss after 300 epochs. As a matter of fact, losses were quite similar between both PredNet versions. Besides, while the convolutional LSTM, simple RNN and simple GRU reconstruction losses were close to PredNet, the simple LSTM loss was in general 50% to 100% greater than the other networks.

Subsequently, Fig. 7 shows that only the convolutional LSTM and both PredNets had a decreasing loss. Among these networks, convolutional LSTM performed best decoding. Different from reconstruction, the simpler PredNet network was less accurate than the more ”complex” PredNet (with

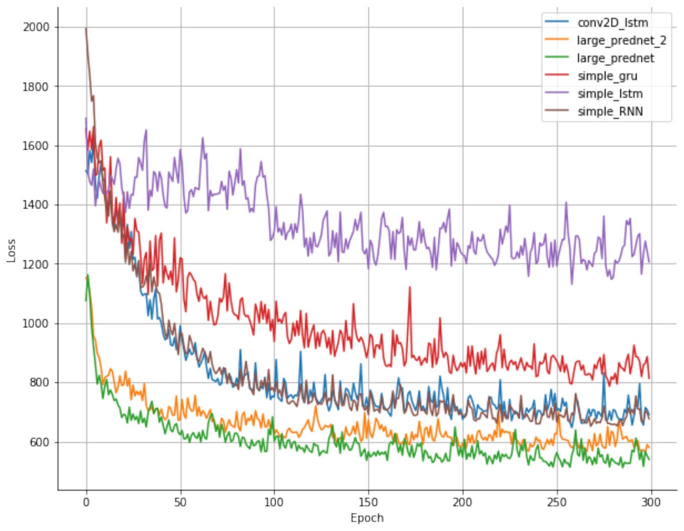


Fig. 6: Reconstruction loss at each epoch for six recurrent networks: RNN, LSTM, GRU, convolutional LSTM and two versions of PredNet. PredNet networks achieved lowest losses.

more channels in both the representation and input layers). By contrast, losses for the ”simple” RNN, LSTM and GRU were quasi-constant over 300 epochs of training. Thus, only the convolutional LSTM and the ”complex” PredNet were used in the testing framework.

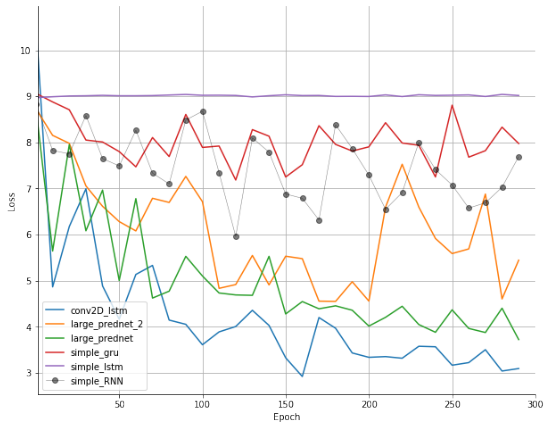
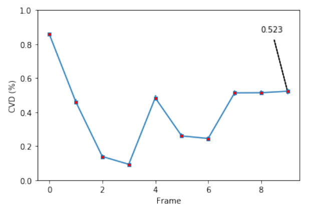


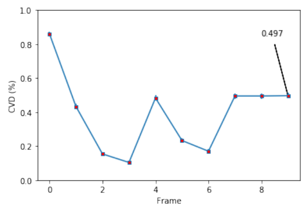
Fig. 7: Decoder loss at each epoch for four recurrent networks: RNN, convolutional LSTM and two versions of PredNet. For easier visualization, only 30 points were plotted (instead of 300), where each point is the mean loss over 10 epochs. Convolutional LSTM achieved lowest loss.

Finally, the selected recurrent networks (i.e., convolutional LSTM and PredNet) responded very differently when predicting the SQM conditions (i.e., vernier-only (V), vernier-provernier (VPV), vernier-antivernier (VAV)). For the convolutional LSTM, central vernier dominance seemed to fade until the fifth frame (where a flanking vernier with non-zero offset), and then settled down after the eighth frame around 50% dominance (Fig. 8). For PredNet, CVD drastically changed at the sixth frame for both VAV and VPV models, before converging around 50% like the convolutional LSTM (Fig. 9). Meanwhile, CVD barely changed at the same frame for the vernier-only model with PredNet (Fig. 9). At the last frame, the CVD increased from VAV to VPV to V for both networks.

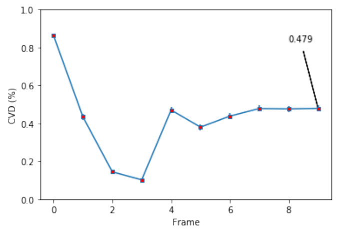
In addition, the entropy plots also differed significantly between the convolutional LSTM and PredNet (see Fig. 10). First, the entropy values for the convolutional LSTM (Fig. 10a) were generally higher than those of PredNet (Fig. 10a), albeit having lower variability. Subsequently, PredNet had a significant drop at the fifth frame, while a smaller trough could be observed at the ninth frame for the convolutional LSTM.



(a) Vernier only (V). CVD at last frame: **0.523**

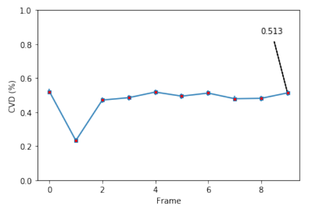


(b) Vernier-provernier (VPV). CVD at last frame: **0.497**

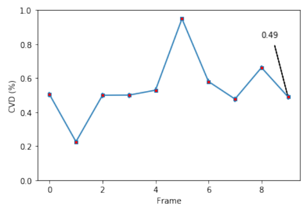


(c) Vernier-antivernier (VAV). CVD at last frame: **0.479**

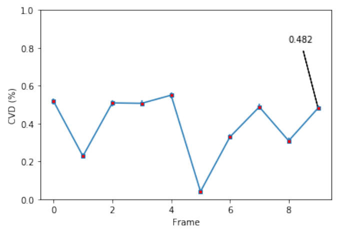
Fig. 8: Central Vernier Dominance (CVD) predicted by a convolutional LSTM over 10 SQM frames (see Eq. 3). For vernier-provernier (VPV) and vernier-antivernier (VAV) paradigms, the second vernier appeared at the fifth frame.



(a) Vernier only (V). CVD at last frame: **0.513**

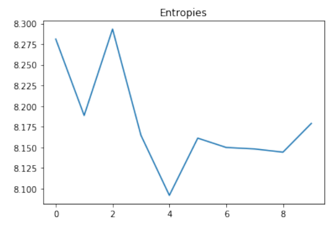


(b) Vernier-provernier (VPV). CVD at last frame: **0.49**

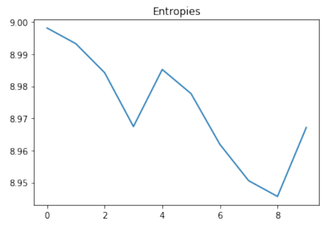


(c) Vernier-antivernier (VAV). CVD at last frame: **0.482**

Fig. 9: Central Vernier Dominance (CVD) predicted by the ”complex” PredNet over 10 SQM frames (see Eq. 3). For vernier-provernier (VPV) and vernier-antivernier (VAV) paradigms, the second vernier appeared at the fifth frame.



(a) ”Complex” PredNet



(b) Convolutional LSTM

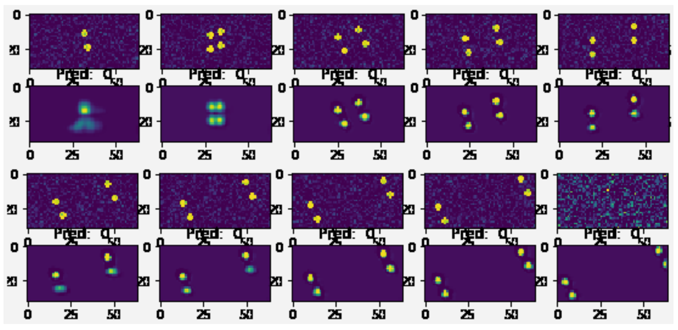
Fig. 10: Mean entropy of PredNet and convolutional LSTM while predicting SQM frames in the VPV configuration. Note that the plots are very similar for the other configurations (VAV, V).

IV. Discussion

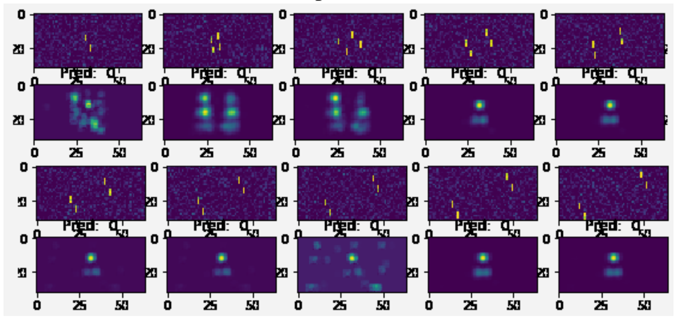
Combining results from reconstruction and decoding, the convolutional LSTM and PredNet proved to be the best performing networks. For reconstruction, this can be considered as an expected result, as both networks are commonly used for next-frame prediction [12] [19]. Furthermore, the non-convolutional networks (simple RNN, GRU and LSTM) did not manage to ”learn” decoding correctly. Indeed, their decoding loss barely decreased compared to PredNets and the convolutional LSTM (Fig. 7). Thus, as PredNet also uses convolutional LSTM layers [12], one could at least suggest that convolutional layers remain necessary for predicting object motion.

Furthermore, loss graphs generally agreed with image predictions for reconstruction. For instance, accurate frame predictions were made from the ”complex” PredNet (Fig. 11a) compared to the poor predictions from LSTM (Fig. 11b). However, the image predictions were quite weak during decoding (e.g., see Fig. 12) albeit the predictions for offset directions were accurate. A first explanation is that the lines were small, and thus maybe less easily detectable. Furthermore, the weak image predictions could also be due to the fact that data is non-sequential compared to the reconstruction framework.

Finally, the convolutional LSTM and PredNet behaved very differently when predicting SQM-like sequences. On the one hand, one could argue that PredNet modelled best human responses to both VAV and VPV configurations. Indeed, CVD drastically changed after a flanking offset was shown, and followed the human results found by Drissi-Daoudi et al. [14] - in particular, that the central vernier dominated predictions for VPV, while the flanking vernier was dominant for VAV. Moreover, the entropy drop observed at the fifth frame (where a flanking vernier is offset) suggests that PredNet ”made a decision” at this frame, as lower entropy correlates with neurons responding more similarly. On the other hand, convolutional LSTM could be considered as more adequate for the vernier-only configuration (V). Indeed, high central vernier dominance was observed at the first frame, which was also observed with human subjects [14]. This could be a consequence of the decoding training, where convolutional LSTM performed best (Fig. 7). On the contrary, PredNet tended to random chance for this case. Nevertheless, the convolutional LSTM was in general less precise to model the SQM. Looking at its entropy plot (Fig. 10a), the minimum entropy reached at the 9th frame corresponded to approximately 50% CVD for each configuration, i.e., close to random chance. This could suggest that offset integration was quickly lost for this network, thus leading to near-random guesses of offset directions.



(a) ”Complex” PredNet



(b) LSTM

Fig. 11: Comparison of reconstruction predictions of two recurrent networks (PredNet and LSTM, in second and fourth rows) against ground truth (first and third rows) after 300 epochs. The first three empty frames are not displayed.

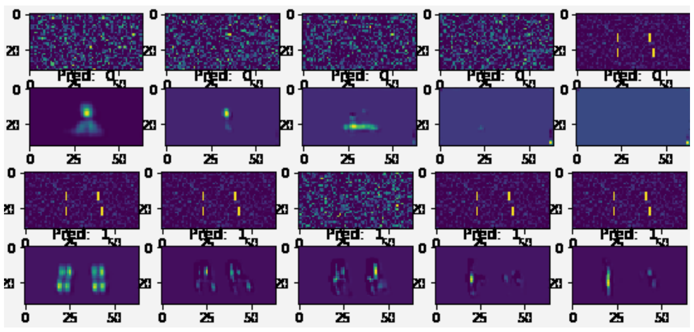


Fig. 12: Comparison of decoding predictions of the ”complex” PredNet (second and fourth rows) against ground truth (first and third rows) after 300 epochs. The first three empty frames are not displayed.

V. Conclusion

This project aimed at modelling the SQM using a reconstructor-decoder framework with five recurrents networks: RNN, LSTM, GRU, convolutional LSTM and PredNet. While PredNet proved best at reconstruction (i.e., predicting object motion), the convolutional LSTM was most accurate at decoding (i.e., predicting offset directions). As simple recurrent networks generally failed at decoding, convolutional layers still seem necessary for these tasks. Finally, PredNet seemed to behave more in a human-like fashion, especially when being shown flanking verniers with offsets (VAV and VPV configurations). As the simple LSTM was the worst recurrent network for both reconstruction and decoding, one may wonder if replacing LSTM-connections by RNN-connections in PredNet would model better the SQM. A future direction would be to test PredNet against other networks for next-frame prediction, e.g., ROLO [20], Inception-inspired LSTMs [21] or a deep predictive coding network [22].

VI. Code availability

The pipeline is available at: https://github.com/Neclow/model_sqm_with_rnns/

VII. Acknowledgements

I wish to thank Alban Bornet, Oh-Hyeon Choung and Adrien Doerig for their supervision throughout the whole project.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[3] Adrien Doerig et al. "Crowding Reveals Fundamental Differences in Local vs. Global Processing in Humans and Machines". In: *bioRxiv* (2019), p. 744268.

[4] Daniel J Felleman and David C Van Essen. "Distributed hierarchical processing in the primate cerebral cortex." In: *Cerebral cortex (New York, NY: 1991)* 1.1 (1991), pp. 1–47.

[5] Radoslaw Martin Cichy, Dimitrios Pantazis, and Aude Oliva. "Resolving human object recognition in space and time". In: *Nature neuroscience* 17.3 (2014), p. 455.

[6] Alex Clarke et al. "Predicting the time course of individual objects with MEG". In: *Cerebral Cortex* 25.10 (2014), pp. 3602–3612.

[7] Thomas U Otto, Haluk Ögmen, and Michael H Herzog. "The flight path of the phoenix—The visible trace of invisible elements in human vision". In: *Journal of Vision* 6.10 (2006), pp. 1079–1086.

[8] Michael H Herzog, Thomas Otto, and Haluk Ögmen. "The fate of visible features of invisible elements". In: *Frontiers in psychology* 3 (2012), p. 119.

[9] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[10] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 `[cs.CL]`.

[11] SHI Xingjian et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in neural information processing systems*. 2015, pp. 802–810.

[12] William Lotter, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *arXiv preprint arXiv:1605.08104* (2016).

[13] Rajesh PN Rao and Dana H Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects". In: *Nature neuroscience* 2.1 (1999), p. 79.

[14] Leila Drissi-Daoudi, Adrien Doerig, and Michael H Herzog. "Feature integration within discrete time windows". In: *Nature communications* 10.1 (2019), pp. 1–8.

[15] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[16] Stefan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: 10.7717/peerj.453. URL: https://doi.org/10.7717/peerj.453.

[17] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 464–472.

[18] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv* (2014).

[19] François Chollet et al. *Convolutional LSTM - Keras Documentation*. 2016. URL: https://keras.io/examples/conv_lstm/.

[20] Guanghan Ning et al. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[21] Matin Hosseini et al. "Inception-inspired LSTM for Next-frame Video Prediction". In: *arXiv preprint arXiv:1909.05622* (2019).

[22] Haiguang Wen et al. "Deep predictive coding network for object recognition". In: *arXiv preprint arXiv:1802.04762* (2018).