Bachelor Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Application of Predictive Coding for Visuo-Tactile Sensory Integration

Adrián Pitoňák

Supervisor: Ing. Zdeněk Straka
Supervisor–specialist: Mgr. Matěj Hoffmann, Ph.D.
Field of study: Cybernetics and Robotics
May 2020

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Pitoňák Adrián**	Personal ID number:	**474380**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Application of Predictive Coding for Visuo-Tactile Sensory Integration

Bachelor’s thesis title in Czech:

Aplikace prediktivního kódování pro vizuo-taktilní integraci

Guidelines:

1. Study selected literature on predictive coding, multisensory integration and peripersonal space representations.
2. Create suitable dataset with visuo-tactile stimuli in Neurorobotics platform - objects approaching or receding from a simulated sensory array and observed by the robot camera.
3. Train a predictive coding based neural network (you will receive it from your supervisor) on this dataset with only visual inputs (next frame video prediction).
4. Propose how the neural network can be extended to the tactile modality: predicting the contact through the activation of appropriate tactile sensors.
5. Evaluate and analyze the proposed extensions.
6. Compare and discuss the properties of the neural network employed in this work with selected multisensory integration models and peripersonal space representations.

Bibliography / sources:

[1] Ursino, Mauro, Cristiano Cuppini, and Elisa Magosso. "Neurocomputational approaches to modelling multisensory integration in the brain: a review." Neural Networks 60 (2014)
[2] Cléry, Justine, et al. "Neuronal bases of peripersonal and extrapersonal spaces, their plasticity and their dynamics: knowns and unknowns." Neuropsychologia 70 (2015)
[3] Noel, Jean-Paul, et al. "Neural adaptation accounts for the dynamic resizing of peripersonal space: evidence from a psychophysical-computational approach." Journal of neurophysiology 119.6 (2018)
[4] Rao, Rajesh PN, and Dana H. Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." Nature neuroscience 2.1 (1999)
[5] Straka, Zdenek, and Matej Hoffmann. "Learning a peripersonal space representation as a visuo-tactile prediction task." International Conference on Artificial Neural Networks. Springer, Cham, (2017)

Name and workplace of bachelor’s thesis supervisor:

Ing. Zdeněk Straka, Vision for Robotics and Autonomous Systems, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Mgr. Matěj Hoffmann, Ph.D., Vision for Robotics and Autonomous Systems, FEE

Date of bachelor’s thesis assignment: **19.12.2019** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. Zdeněk Straka	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

First of all, I wish to express my sincere thanks to my supervisor Ing. Zdeněk Straka and my supervisor-specialist Mgr. Matěj Hoffmann, Ph.D. for their enthusiasm for the project, for their support, encouragement and patience. Further, I would like to thank the faculty, for providing me with all the necessary facilities for the research. To conclude, I cannot forget to thank my family and close friends for all the unconditional support throughout the study.

Declaration

Abstract

The main goal of this thesis is to train a neural network based on predictive coding, that is capable of representing peripersonal space by learning from images and tactile sensors. The PreCNet neural network for next frame video prediction, based on predictive coding, was used for this task and extended with the tactile modality. For training the network, I designed experiments in the Neurorobotics Platform, in which an object is approaching the humanoid robot iCub with tactile sensors on the torso. This object consequently hits or misses the robot. Trained neural networks were evaluated by their ability to predict looming stimulus temporally and spatially, both quantitatively and qualitatively. In the designed experiments the neural network was able to implement visuo-tactile integration. I analyzed the drawbacks of this model and put forward improvements for future work. Achieved results indicate that a network based on predictive coding is capable of multisensory integration, which is necessary for the representation of peripersonal space.

Keywords: peripersonal space, predictive coding, multisensory integration

Abstrakt

Cieľom tejto práce je natrénovať neurónovú sieť založenú na prediktívnom kódovaní, ktorá je schponá reprezentovať peripersonálny priestor, učením z obrázkov a taktilných senzorov. Neurónová sieť PreCNet, určená na predickiu ďalšieho snímku videa, založená na prediktívnom kódovaní, bola po rozšírení o taktilnú modalitu, použitá pre túto úlohu. Na učenie tejto siete som vytvoril datasety v Neurorobotickej Platforme, v ktorých sa na humanoidného robota iCuba s taktilnými senzormi na trupe, posutpne približuje objekt. Následne tento objekt robota zasiahne alebo minie. Naučené neurónvé siete boli kvantitatívne a kvalitatívne vyhodnotené, na základe ich temporálnej a priestorovej schopnosti predikovať prichádzajúci stimul. V navrhnutých experimentoch neurónová sieť dokázala implementovať vizuo-taktilnú modalitu. Zanalyzoval som nedostatky siete a navrhol možné riešenia pre budúcu prácu. Dosiahnuté výsledky naznačujú, že sieť založená na prediktívnom kódovaní je schopná multisenzornej integrácie, ktorá je nevyhnutná pre reprezentáciu peripersonálneho priestoru.

Klíčová slova: peripersonální prostor, prediktivní kódování, multisenzorní integrace

Překlad názvu: Aplikace prediktivního kódování pro vizuo-taktilní integraci

Contents

Chapter 1

Introduction

This thesis focuses on the implementation of visuo-tactile multisensory integration in neural network PreCNet [SSH20] based on Predictive Coding—a popular theory from neuroscience—extended by tactile modality. Since multisensory integration, the process of combining information from different modalities, of tactile and visual senses stands in the root of peripersonal space (space immediately surrounding an agent's body) representation such a neural network could be used to represent peripersonal space in humanoid robots. Creating a real-world dataset might be very time consuming and very complex from the beginning when one cannot estimate the behaviour of the network. Therefore I designed an experiment in the Neurorobotics Platform [Pro] [FVA+17], which is highly adjustable and preserves the structure of tactile sensors on iCub's [MNN+10] torso. In the experiment, the camera in the robot's eye is observing the approaching ball, until it hits tactile sensors located on the torso, thus activating them. Images and tactile activations obtained from this experiment are then used as a dataset to train the neural network. Simulation of the experiment in a virtual environment makes it possible to create several datasets, that could explain strategies for predicting incoming stimuli.

The thesis is structured as follows. In Chapter 2, I will describe the peripersonal space representation and the importance of multisensory integration in it, following dynamical properties of peripersonal space representation. After that, I will outline some computational models of peripersonal space. In Chapter 3, I will describe related predictive coding models, such as the general schema of predictive coding proposed by Rao and Ballard, the used neural network PreCNet and outline some other networks that use predictive coding. At the end of this chapter, I will introduce the extension of PreCNet network for tactile modality. Chapter 4 describes the task of this work, experiments from which datasets were generated and structure of these datasets, learning parameters of the network and the system I proposed for the evaluation of the results. I will present the results and their interpretation in Chapter 5. In Chapter 6, I will discuss the results of trained networks and provide suggestions for future improvements. Finally, in Chapter 7, I will summarize the achieved results of the whole work.

Chapter 2

Multisensory integration and peripersonal space representation

Multisensory integration is the process of combining information from various sensory channels (e.g., [SSR09]). Visuo-tactile neurons are activated when stimulated with tactile, but also with visual stimuli, close to the same body part independently (see, for example, [RSMG81] [DCG98]). Multisensory integration is very closely related to peripersonal space, that is the space immediately surrounding the body (see Figure 2.1), where the interactions between the body and the environment occur (see e.g., [NBMS18] [GC06] [SNG+15]).

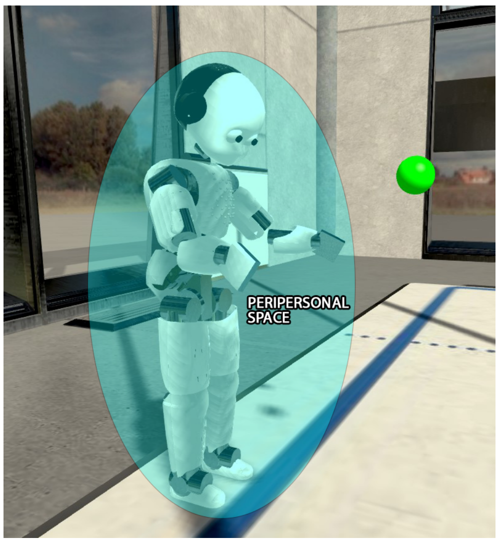


Figure 2.1: Peripersonal space is the space immediately surrounding the body.

Protecting the space close to one's body from external stimuli like objects or living beings is essential for survival. It is concluded, that brain areas involved in the peripersonal space representation are also responsible for protecting the space close to our body by encoding a safety body margin [CGWH15] [NBMS18]. As an example of encoding safety body margin serves a study conducted on monkeys [GC06]. In their experiment brain areas linked to peripersonal space representation (see Section 2.2) were stimulated and triggered defensive-like behaviour.

The next section describes the main properties of peripersonal space. Further, it describes the dynamic properties and the effect of approaching objects to the peripersonal space of the subject and mentions a few experiments performed on this matter. Finally, the last section outlines some computational models of peripersonal space representation.

2.1 Properties of peripersonal space representations

At the beginning of this chapter, I will describe visuo-tactile neurons that are essential in peripersonal space representation. These neurons were identified, for example, in brain areas F4 and VIP of parietal-premotor network [RFFG97] [FGF+96] [CDG93].

Receptive fields (RFs) of visuo-tactile neurons are body-part-centered [SNG+15] [dPLF97]. In other words, peripersonal spaces around the specific body part move with this body part. Combination of these receptive fields encodes peripersonal space representation [RSMG81], "serving the definition of safety body margin contributing to the definition of self (as a whole) with respect to the external world" [CGWH15]. Vulnerable body parts as the head or arms are over-represented to protect this body margin from external aggression (e.g., [CGWH15]).

Another important property of receptive fields is that they change dynamically according to the velocity of looming stimuli (they enlarge with the increasing velocity of the stimuli) (see, for example, [NBMS18] [CGWH15]). These dynamical properties are further described in the next section.

2.2 Dynamic properties of peripersonal space

An important point is that stable stimuli in our peripersonal space are not as significant as stimuli looming towards us since they could pose a possible threat [CGWH15] [GC06] [Gib72]. Receptive fields of neurons that encode peripersonal space are dynamical, as they grow with increasing velocity of the looming object. To detect objects in space relative to the subject, both low-level cues, namely area, contrast, edges, visual depth [ZOS08] and high-level cues, like the apparent size of the detected object, based on experience must be perceived [CGWH15].

Ball and Tronick performed 2 experiments on this phenomenon [BT71]. In the first experiment, 24 infants reacted with avoidance and upset to expanding shadows, representing approaching objects, whereas ignoring shadows acting as shrinking objects or objects that could not possibly hit them. Besides that, in the second experiment, “seven infants defensively reacted to the approach of a real object except when it was on a miss path” [BT71].

Another experiment was performed on monkeys, observing multisensory integration for dynamic stimuli looming towards the face. According to this experiment multisensory integration indicates the highest activity when tactile stimulus can be predicted from the visual stimulus and these stimuli are spatially and temporally (in regard to space and time) related [CGWBH13] (for more detailed discussion see [CGWH15]). Authors of this study further emphasize that the parietal-premotor network mentioned in Section 2.1 was significantly activated during this experiment, especially when the stimuli were predicted. Furthermore, they suggest, that this network belongs most likely to a “larger functional network involving lower-level visual areas” [CGWH15].

2.3 Computational models of peripersonal space

As it was depicted in Section 2.1, visuo-tactile neurons are body specific and create peripersonal space representation around specific body parts. To simulate the peripersonal space representation around the right and left hands a neural network was proposed that implements visuo-tactile multisensory integration [MZS$^+$10]. More specifically, this neural network is composed of two network parts that represent the left and right hemisphere, meanwhile, they are reciprocally interconnected. Each of these hemispheres further consists of three populations of neurons. Two populations are unimodal—

visual and tactile. They interact with the third visuo-tactile multimodal population via feedforward and feedback connection. The model was built to better understand relations between the described populations.

Another neural network designed to represent peripersonal space by visuo-tactile integration consists of a Restricted Boltzmann Machine for encoding the position and velocity of visual stimulus and a feedforward neural network for predicting the tactile stimulus [SH17]. This model aimed to predict the tactile stimulus from the position and velocity of a looming stimulus in the 2D scenario.

The most related model to our work [RHP+16] is trained on real iCub humanoid robot [MNN+10]. It uses two cameras, joint angles and artificial skin not only on the torso but also other body parts (i.e., fingers, palms, forearms) to compute the likelihood of collision with the corresponding tactile sensor. The main goal of this model is to learn body-centred peripersonal space representation with dynamical receptive fields by self-touch or tactile interaction with other objects. After the tactile activation occurs, the model tracks the stimulus back in time to update the corresponding likelihood.

Chapter 3

Predictive coding network for visuo-tactile integration

As the network used in this experiment is a modification of the Predictive Coding Network (PreCNet) [SSH20] that is based on predictive coding, it is important to first understand the general concept of predictive coding and the basic structure of PreCNet. Apart from this, I will here introduce other related models based on predictive coding. Above all, I will introduce our proposed extension of PreCNet with tactile modality, which is used for peripersonal space representation learning in our work, in Section 3.2.

3.1 Related predictive coding networks

At the beginning of this section, I will describe the concept of predictive coding. The Predictive Coding Network (PreCNet), the state-of-the-art implementation of the general schema proposed by Rao and Ballard, is described in separate Section 3.1.2, as it is the network that was modified for this work. Next to this, I will introduce several other implementations of predictive coding.

3.1.1 Predictive coding

Predictive coding is a brain theory that helps us understand the redundancy reduction of the nervous system. By learning the statistical regularities of the environment and the data, provided through sensory channels, the network can achieve more efficient coding by removing the predictable components of the input [HR11]. Further in this section, I will describe the mechanisms of predictive coding proposed by Rao and Ballard [RB99] and presents their general schema and generative model of predictive coding.

General schema by Rao and Ballard

Rao and Ballard suggested a general schema of predictive coding [RB99], with hierarchically organized Predictive Estimators (PE). Feedback pathways (also known as top-down or higher to lower-order connections) in each level carry predictions of the neural activity of the lower level PE. On the lowest level, the neural activity corresponds to the sensory inputs. These predictions are subtracted from the actual response in each layer, producing residual errors. Feedforward (also known as bottom-up or lower to higher-order) pathways are then responsible for carrying these residual errors of lower-level activities to the higher level to improve the prediction [RB99] (see Figure 3.1b).

Predictive estimator

In the generative model [RB99] designed by Rao and Ballard, there are several levels of predictive estimators (see Figure 3.1b). This section describes one unit of a predictive estimator.

A 'top-down' prediction estimate of an internal representation vector $\mathbf{r^{td}}$ from higher level is subtracted by current internal representation vector $\mathbf{r}$ (see Figure 3.1a). This residual error is sent to higher level, as well as it is used for updating the current state of $\mathbf{r}$, along with residual error from lower level encoded with weight matrix $\mathrm{U^T}$. The state $\mathbf{r}$ is additionally sent down, decoded by weight matrix U, following f activation function (i.e., tanh) producing prediction estimate for lower level[1] (see Figure 3.1a).

[1] In the first iteration, prediction estimates are initialized randomly on each level

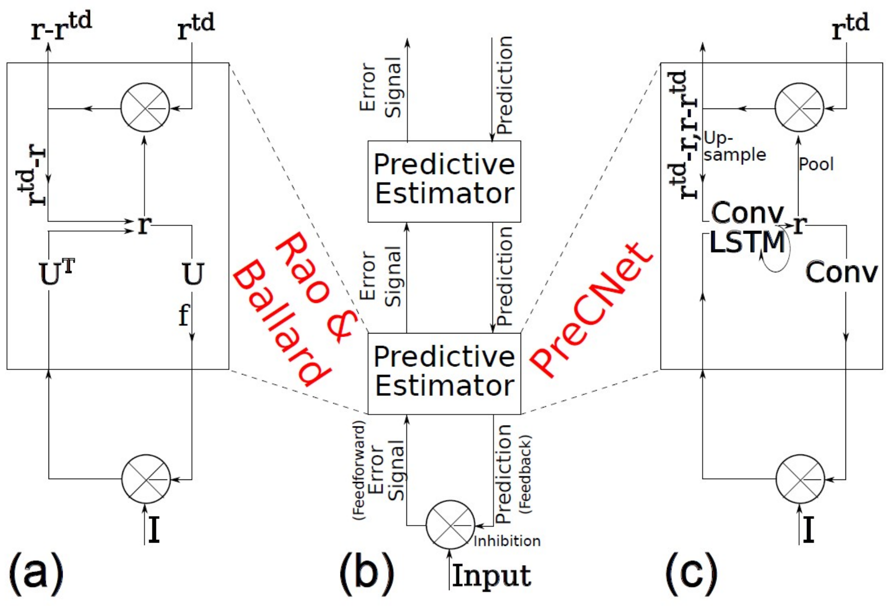


Figure 3.1: **(a) Components of a PE module**, composed of feedforward neurons encoding the synaptic weights U^T, neurons whose responses $\mathbf{r}$ maintain the current estimate of the input signal, feedback neurons encoding U and conveying the prediction f(U**r**) to the lower level, and error-detecting neurons computing the difference $(\mathbf{r} - \mathbf{r^{td}})$ between the current estimate r and its top-down prediction $\mathbf{r^{td}}$ from a higher level. **(b) General architecture of the hierarchical predictive coding model.** At each hierarchical level, feedback pathways carry predictions of neural activity at the lower level, whereas feedforward pathways carry residual errors between the predictions and actual neural activity. These errors are used by the predictive estimator (PE) at each level to correct its current estimate of the input signal and generate the next prediction. **(c) Components of a PE module of PreCNet architecture** (see Section 3.1.2). Figures **(a)**, **(b)** and **(c)** taken from [SSH20], in which **(a)** and **(b)** are redrawn from [RB99].

3.1.2 PreCNet – Predictive Coding Network

The Predictive Coding Network (PreCNet) was used for next frame video prediction [SSH20]. Importantly, to make predictions of future frames, the neural network must possess an internal representation of the scene and its dynamics, leading on to self-supervised learning (without labelled dataset) of such internal representation [MCL15]. In other words, the information about the environment is provided to the network by only using actual visual sensations. The network is trained by minimizing the weighted sum of errors through error units in each module, through modules, then through individual frames in sequences and finally through sequences (see Equations 3.1 and 3.2) using gradient descent [LKC16] [SSH20].

$$L_{train} = \sum_{m=1}^{M} L_{seq}(m) \tag{3.1}$$

$$L_{seq}(j) = \sum_{t=1}^{T} \mu_t \sum_{l=0}^{N} \frac{\lambda_l}{n_l} \sum_{i=1}^{n_l} E_l^t(i) \tag{3.2}$$

where L_{train} is the loss of one episode, L_{seq} is the loss of one sequence, M is the number of sequences in the train dataset, T is the length of each sequence, N + 1 is the number of modules and n_l is the number of error units in each module. $E_l^t(i)$ is the error of i-th unit in block l (see Algorithm 1). μ_t is time constant and λ_t is module constant.

To update the representation state, the neural network uses convolutional LSTM (convLSTM) modules [XCW⁺15] [HS97], which are defined as follows (see Equations 3.3 – 3.7).

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \tag{3.3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \tag{3.4}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3.5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \tag{3.6}$$

$$h_t = o_t \circ tanh(c_t) \tag{3.7}$$

where t denotes actual time step, t-1 denotes previous time step. W represents weight matrix, x is the input to convLSTM block and b represents the bias. The memory cell is marked as c and its two inputs are weighted with input gate i and forget gate f. Final state[2] h is influenced by the output gate o. The operations used between the matrices are convolutions, σ is sigmoid function[3] and $\circ$ is element-wise multiplication.

In the next section, I will provide a more detailed description of PreCNet.

[2]Later in the work the final state is referred to as the representation state r.

[3]PreCNet uses hard sigmoid function instead of sigmoid function

PreCNet schema

PreCNet adjusts internal representation states in order to minimize future prediction errors. For this purpose, it is organized hierarchically according to the general schema of predictive coding proposed by Rao and Ballard, yet using modern deep learning tools. As can be seen from Figure 3.1, PreCNet neural network and generative model by Rao and Ballard have a very similar structure.

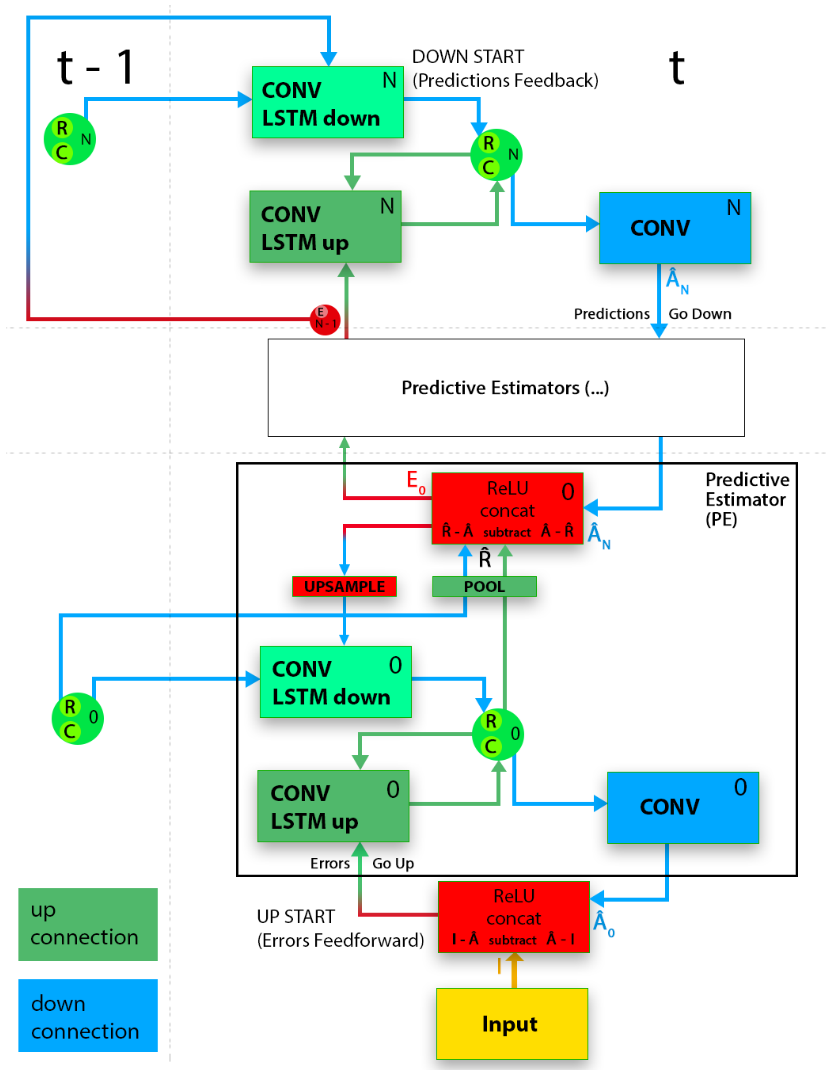


Figure 3.2: PreCNet schema based on PreCNet (see Algorithm 1). In the article [SSH20] RGB images were used as an input for the network.

On the **top-down path**, (see "Predictions Go Down" in Figure 3.2; for further details see the first for loop in Algorithm 1) a prediction estimate $\mathbf{r}^{\mathbf{td}}$ from higher layer is subtracted by pooled (downsized by 2) internal representation state $\mathbf{r}$ from previous time step. This residual error gets upsampled (by 2) and serves as input to convLSTM block [XCW+15] [HS97], together with representation states from the last time step. ConvLSTM updates the current state of $\mathbf{r}$. After that, $\mathbf{r}$ is decoded by convolutional layer, producing prediction estimate for lower layer[4] (see Figure 3.2c).

Conversely, on the **bottom-up path** (see “Errors Go Up” in Figure 3.2; for further details see the second for loop in Algorithm 1), an internal representation state $\mathbf{r}$ is updated, by inputting the residual error from lower layer to convolutional LSTM layer. This prediction estimate $\mathbf{r}$ is pooled and subtracted by prediction estimate $\mathbf{r^{td}}$ from higher level, producing the residual error for higher layer (see Figure 3.2c).

Building blocks of predictive estimator and structure

Each time step starts with the top-down flow (see “DOWN START” in Figure 3.2) and continues down, until the prediction of the input image is produced. After that, the error from the lowest layer (see “UP START” in Figure 3.2) is propagated to the highest layer.

Each layer of PreCnet consists of 5 main building blocks.

- **Representation Layer** – it consists of two convolutional LSTMs (convLSTM) [XCW+15] [HS97] (see Equations 3.3 – 3.7), which share cell state C and representation state[5] R (see green RC bubbles in Figure 3.2).

  States R and C are generated by convLSTM module on the top-down path, with the R and C states from the previous time step and upsampled error (see Upsample Layer). At the highest level, the network uses directly the highest 'bottom-up' error state from the previous time step (see Figure 3.2). On the bottom-up path are R and C states updated by another convLSTM module, using R and C states generated on the top-down path and error from the same layer.

  The convLSTM module uses hard sigmoid activation for the input, forget and output gates. Hyperbolic tangent is used for the calculation of the cell state C and representation state R.

[4]In the first iteration, prediction estimates are initialized to zero on each level.

[5]representation state R is in convLSTM layers better known as 'h'. However R corresponds to the representation state r in Figure 3.1

Algorithm 1 Calculate PreCNet states at time t, assume $N > 0$. Algorithm provided by Zdenek Straka from [SSH20].

```
Require: Image $I^t$, previous $(t-1)$ hidden and cell states $R_l^{t-1}, C_l^{t-1}$ of the
  representation layers $l \in \{0, 1, \ldots, N\}$, previous error state $E_N^{t-1}$ of the
  (top) module $N$, maximum pixel value $pix^{max}$.
  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
      $R_l^t, C_l^t \leftarrow \text{convLSTM}_l^{down}(R_l^{t-1}, C_l^{t-1}, E_l^{t-1})$
      $\hat{A}_l^t \leftarrow \text{conv}_l(R_l^t)$
      $E_l^t \leftarrow \text{ReLU}(\{\hat{A}_l^t - \text{pool}(R_{l-1}^{t-1}), \text{pool}(R_{l-1}^{t-1}) - \hat{A}_l^t\})$
    end if

    if $l \neq N$ and $l \neq 0$ {Update the states in the “middle” module $l$} then
      $R_l^t, C_l^t \leftarrow \text{convLSTM}_l^{down}(R_l^{t-1}, C_l^{t-1}, \text{upsample}(E_{l+1}^t))$
      $\hat{A}_l^t \leftarrow \text{conv}_l(R_l^t)$
      $E_l^t \leftarrow \text{ReLU}(\{\hat{A}_l^t - \text{pool}(R_{l-1}^{t-1}), \text{pool}(R_{l-1}^{t-1}) - \hat{A}_l^t\})$
    end if

    if $l == 0$ {Update the states in the bottom module} then
      $R_l^t, C_l^t \leftarrow \text{convLSTM}_l^{down}(R_l^{t-1}, C_l^{t-1}, \text{upsample}(E_{l+1}^t))$
      $\hat{A}_l^t \leftarrow \min\{\text{conv}_l(R_l^t), pix^{max}\}$
      $E_l^t \leftarrow \text{ReLU}(\{\hat{A}_l^t - I^t, I^t - \hat{A}_l^t\})$
    end if
  end for

  for $l = 0, 1, \ldots, N$ {Iterate bottom-up through the modules} do
    if $l == 0$ then
      $R_l^t, C_l^t \leftarrow \text{convLSTM}_l^{up}(R_l^t, C_l^t, E_l^t)$
    end if

    if $l \neq 0$ and $l \neq N$ then
      $E_l^t \leftarrow \text{ReLU}(\{\hat{A}_l^t - \text{pool}(R_{l-1}^t), \text{pool}(R_{l-1}^t) - \hat{A}_l^t\})$
      $R_l^t, C_l^t \leftarrow \text{convLSTM}_l^{up}(R_l^t, C_l^t, E_l^t)$
    end if

    if $l == N$ then
      $E_l^t \leftarrow \text{ReLU}(\{\hat{A}_l^t - \text{pool}(R_{l-1}^t), \text{pool}(R_{l-1}^t) - \hat{A}_l^t\})$
    end if
  end for
```

- **Error Layer** – this layer receives higher layer prediction estimate Â and pooled (see Max-pooling Layer) representation state R̂. These two states are subtracted in both orders (Â - A, A - Â) and concatenated together, followed by the ReLU activation function.

  This is done twice per time step. On the top-down path, when the network takes the representation state R from the previous time step. Then on the bottom-up path with the representation state R from the current time step is used.

- **Decoding Layer** – is a convolutional layer applied on the representation state R on the top-down path, followed by ReLU activation function. It produces prediction estimate state Â that is sent to the lower layer.

- **Upsample Layer** – this layer upscales the input by 2 by nearest neighbour interpolation.

- **Max-pooling Layer** – this layer downscales the input by 2.

3.1.3 Other predictive coding models

In this section, I will outline several other models inspired by the idea of predictive coding.

PredNet – Predictive Neural Network

PredNet was designed for predicting next frame video prediction. Inspired by Rao and Ballard model (see Section 3.1.1), each layer of PredNet makes local predictions, but unlike the model, it forwards deviations from these predictions. In other words, excluding the first input layer, PredNet tries to predict the target error from the lower layer at each level. PredNet also added connection between representation blocks from different layers contrary to Rao and Ballard model [LKC16]. These are also the main distinctions between PredNet and PreCNet networks.

Besides the above, PreCNet and PredNet have many properties in common. Mainly the structure of basic building blocks (e.g., convolutional LSTMs in representation layer and error computation) (for a more details see [SSH20]).

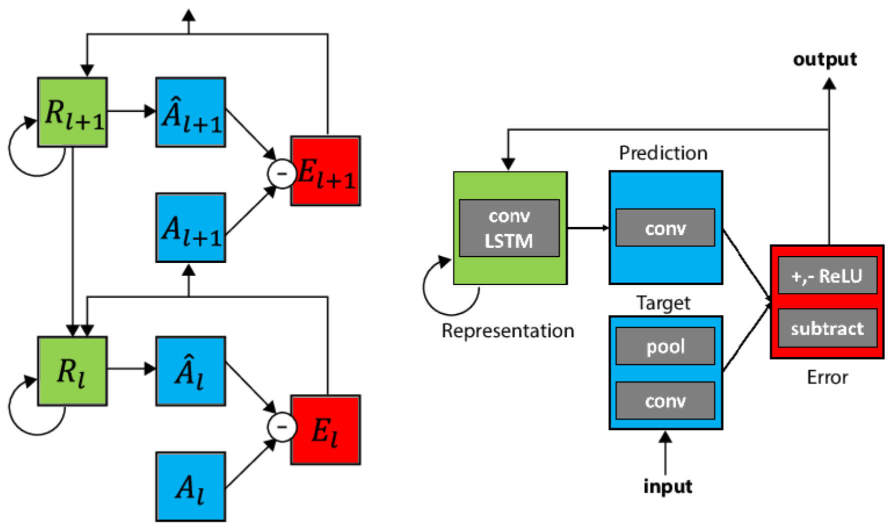


Figure 3.3: Predictive Neural Network (PredNet) schema. Left: Illustration of information flow within two layers. Each layer consists of representation neurons R_l), which output a layer-specific prediction at each time step ($\hat{A}_l$), which is compared against a target (A_l) [Ben14] to produce an error term (E_l), which is then propagated laterally and vertically in the network. Right: Module operations for the case of video sequences. Images and their captions taken from [LKC16].

PCN – Predictive Coding Network

Another neural network inspired by Rao and Ballard model (compare Figure 3.4a-Right with 3.1b) built for image classification is Predictive Coding Network (PCN) [WHS$^+$18]. Unlike PreCNet and PredNet, PCN uses labeled datasets for classifying images. Therefore the main goal of the network is to minimize the error of classification, not the prediction error.

In their article, the authors compared several models with only feedforward connections to models with feedforward and additional recurrent and feedback connections, which "always outperformed its feedforward-only counterpart" [WHS$^+$18]. Furthermore, PCN managed to deliver comparable results to networks with much more layers on classification with benchmark datasets.

DPCN – Deep Predictive Coding Networks

One other notable neural network build for processing video sequences is Deep Predictive Coding Networks (DPCN) [CP13]. DPCN was designed to classify

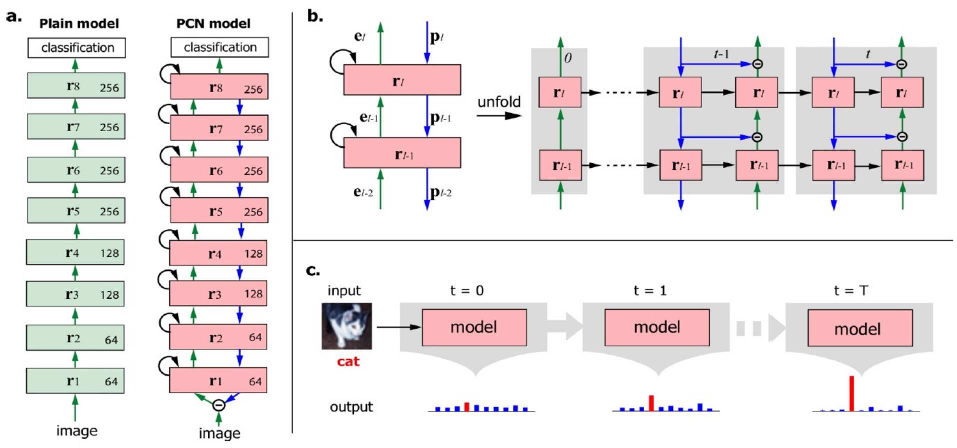


Figure 3.4: Predictive Coding Netwrok (PCN) schema. a) An example PCN with 9 layers and its CNN counterpart (or the plain model). b) Two-layer substructure of PCN. Feedback (blue), feedforward (green), and recurrent (black) connections convey the top-down prediction, the bottom-up prediction error, and the past information, respectively. c) The dynamic process in the PCN iteratively updates and refines the representation of visual input over time. PCN outputs the probability over candidate categories for object recognition. The bar height indicates the probability and the red indicates the ground truth. Images and their captions taken from [WHS+18]

short video sequences containing one of three possible shapes. The main goal of the network is to dynamically adapt to the context of the data. Top-down connections are trying to predict the representation in the layer below, "using the top-down information from the layers above and temporal information from the previous states" [CP13]. Bottom-up connections are extracting key features with sparse states and pooling them, therefore classifying the video.

3.2 Proposed architecture - extension of PreCNet with tactile modality

Since the original PreCNet model uses only images as an input to the network, the first change in the network was made by proposing the 4th tactile channel (see Figure 3.5). In the 4th channel, the state of the tactile modality is encoded as 1-channel image (see the "tactile image" in Fig. 3.5). The source code for the network is available at [PS20c] or on the enclosed CD (see Appendix A).

As our task is to predict the tactile activations caused by looming stimuli, it is much more important to penalise errors for not activating a tactile sensor, when the stimulus already occurred, than penalise errors for falsely predicting

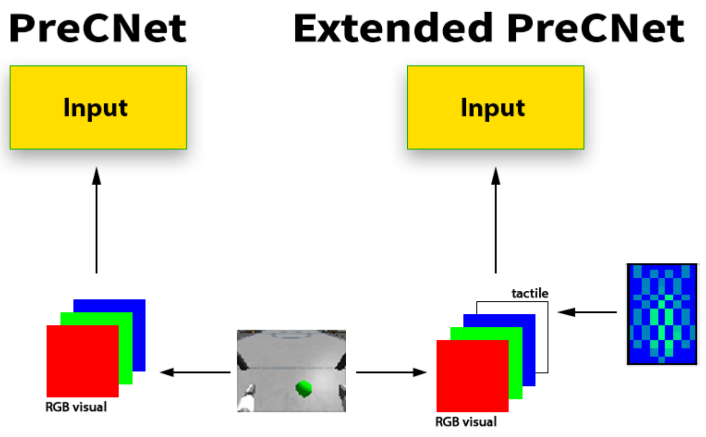


Figure 3.5: Difference between the input of PreCNet and input of extended PreCNet. - In the article [SSH20] 3-channel RGB images were used as an input for the network. The extended version of PreCNet uses 4-channel frames, adding tactile modality, encoded as a 1-channel image, to the last channel (see Section 4.1.2).

the stimulus (see Figure 3.6). In fact, the network is expected to predict the stimulus before it occurs, hence strongly penalising the network for activating the sensors, when there is no activation yet, is unwilling. Though some penalisation for making false-positive errors in predictions should limit the network in activating too many sensors.

Accordingly, the original PreCNet network [SSH20] was modified to set weights to false-negative and false-positive tactile prediction errors. This was achieved by introducing two parameters, namely *tactile false positive* (in short 'fp') and *tactile false negative* (in short 'fn'), that allow us to set weights to the corresponding tactile errors (see changes in Algorithm 2).

Algorithm 2 Changes made in PreCNet algorithm [SSH20]. Algorithm 1 describes original PreCNet. Changes are highlighted by green color. Subscript img denotes that first 3 channels from the array were used, whereas subscript tact denotes the last channel. Curly braces without function before them, symbolize concatenation along the third channel axis. 'fp' is *tactile false positive* parameter and 'fn' is *tactile false negative* parameter.

```
if $l == 0$ {Update the states in the bottom module} then
    $R_l^t, C_l^t \leftarrow \text{convLSTM}_l^{down}(R_l^{t-1}, C_l^{t-1}, \text{upsample}(E_{l+1}^t))$
    $\hat{A}_l^t \leftarrow \min\{\text{conv}_l(R_l^t), pix^{max}\}$
    $E_{l,img}^t \leftarrow \text{ReLU}(\{\hat{A}_{l,img}^t - I_{img}^t, I_{img}^t - \hat{A}_{l,img}^t\})$
    $E_{l,tact}^t \leftarrow \{\text{fp} \cdot \text{ReLU}(\hat{A}_{l,tact}^t - I_{tact)}^t, \text{fn} \cdot \text{ReLU}(I_{tact}^t - \hat{A}_{l,tact}^t)\}$
    $E_l^t \leftarrow \{E_{l,img}^t, E_{l,tact}^t\}$
end if
```

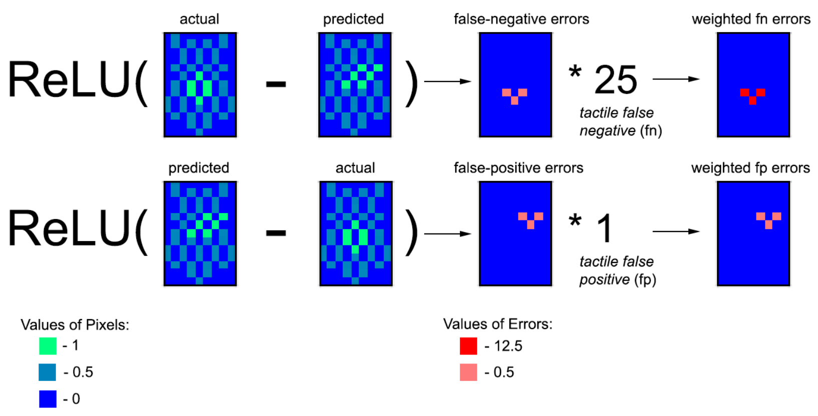


Figure 3.6: False-negative and false-positive tactile errors. The tactile modality with its activations (see the actual and predicted "tactile images") is encoded as 1-channel image (see Section 4.1.3). **Upper:** When the actual sensor was not activated (the value is 0.5) and the activation at this sensor was predicted (the value is 1) the error at this pixel is set to zero since ReLU sets all negative values to zero while preserving all positive values producing false-negative errors. **Lower:** Similar principle applies to false-positive errors. **Both:** After separating these errors their weights can be set by adjusting *tactile false negative* and *tactile false positive* parameters. In this example the network was more penalized for not predicting some of the tactile sensors, by setting the 'fn' to 25 and 'fp' to 1.

As it was described in Section 3.1.2, error layer makes the differences between the prediction and the actual response for each channel (`prediction` − `actual`, `actual` − `prediction`). Taking into account how the tactile data is represented (see Section 4.1.3), subtracting `actual` − `prediction` and applying ReLU activation function yields false-negative errors [6] (see Figure 3.6). Oppositely, subtracting `prediction` − `actual` yields false-positive errors which can be interpreted as falsely identified stimulus or in other words redundant activated sensors (see Figure 3.6). As these errors are now separated, we can set weights to these errors by multiplying them with *tactile false positive* and *tactile false negative* parameters and concatenate them to the image errors. In this way, the network can be more penalized for not predicting the incoming stimuli than for falsely identifying incoming stimuli (see higher value of *tactile false negative* than *tactile false positive* parameter in Figure 3.6).

[6]When the actual sensor was not activated (the value is 0.5) and the activation at this sensor was predicted (the value is 1) the error at this pixel is set to zero since ReLU sets all negative values to zero

Chapter 4

Task description

The main motivation for this work is to learn peripersonal space representation through self-supervised learning, that is to say, using only unlabeled data from tactile sensors and camera. More specifically, a predictive coding based network PreCNet is trained to predict tactile stimulation, closely before an object hits the torso, using visual and tactile inputs (see Figure 4.1).

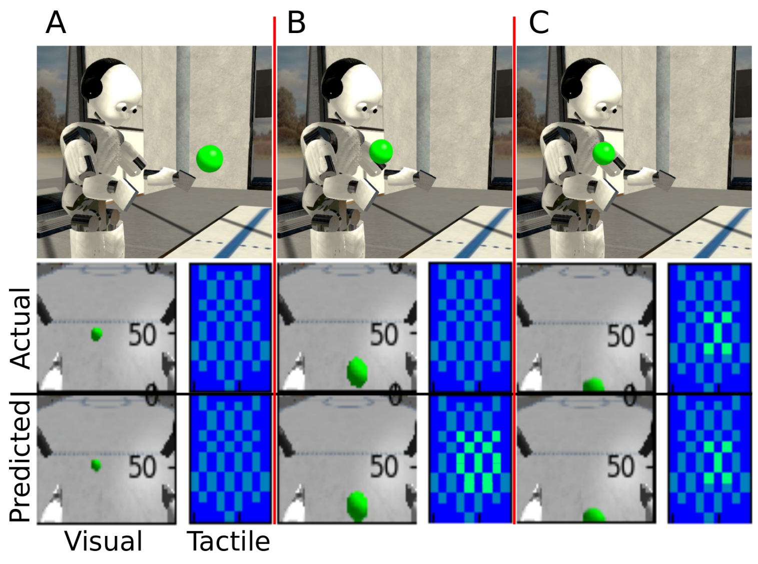


Figure 4.1: Experiment illustration. The ball is looming towards iCub and hits the torso. (upper sequence). Information about the environment is gathered using a camera implemented in iCub's eye and tactile sensors on the torso (Actual – middle sequence). PreCNet neural network is predicting the actual visual and tactile response in each frame (Predicted – lower sequence). In the best-case scenario, the actual activation is predicted, before it occurs (Frame B). Image provided by Zdenek Straka.

To receive input from tactile and visual modalities, I created an experiment in a simulated environment, namely in the Neurorobotics Platform. In this experiment, balls are looming towards the model of humanoid robot iCub [MNN$^+$10], extended with tactile sensors on the torso. These balls can either hit its torso or miss it, eventually hit its other part of the body (i.e., arm).

Next section describes, how data generated in the Neurorobotics Platform are processed to be compatible with PreCNet neural network. After that, I will describe the parameters of the neural network used in this experiment. In the final section, I will describe the procedure for evaluating the results

4.1 Description of the experiment

Motivated by the experiment conducted on monkeys, according to which the highest activity was observed when tactile stimulus could be predicted from the visual stimulus and both stimuli were spatially and temporally related [CGWBH13] (see Section 2.2 for details), I created a framework for designing experiments in the Neurorobotics Platform (online version) [Pro] [FVA$^+$17]. The source code[1] for the experiment is available at [PS20b] or on the enclosed CD (see Appendix A).

The basic model of iCub, available in the Neurorobotics Platform, was extended with tactile sensors floating above its torso, preserving the same layout as in the real model with artificial tactile modality[2] [CGN17]. Yet the complicated high-resolution sensors were simplified with spheres. Apart from this tactile modality, visual information is gathered via a 12 fps camera located in iCub's eye with resolution 120x90 (w x h). Since our model uses video sequences from only one eye, low-level cues related to the visual depth which are critical in object detection, like binocular disparity information [CGWH15], might be missing. Nevertheless, as the object—ball—is simple and its size is constant, this simplification should not influence the experiment in a significant way, as well as the memory requirements for such implementation, would be significantly higher.

In these experiments a ball is randomly spawned on a plane (see the green plane in Figure 4.2) and follows a trajectory along a line to a random point

[1]Parameters described in this Section (e.g., spawn plane, target plane, velocity, frame rate) can be changed (see 'README.md'). The online version of the Neurorobotics Platform was used, so no download nor installation is required

[2]Script for processing the tactile modality was made in cooperation with Jiří Štěpanovský

on a target plane (see the red plane in Figure 4.2) on iCub's torso. In some sequences the ball hits iCub's torso with sensors, meanwhile, in the others the ball might possibly miss the torso. After the ball hits or misses the target, it floats through the air for a while, and another sequence starts.

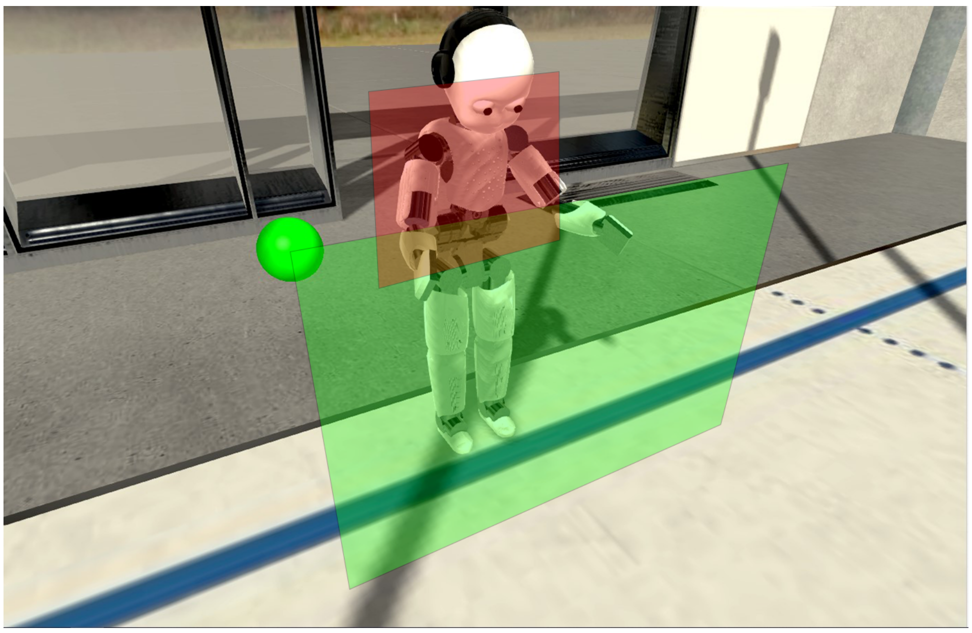


Figure 4.2: Design of the experiment. The green plane indicates possible points, where the ball can randomly spawn. The red plane indicates possible target points, where the ball will head.

Part of the data pre-processing for the network is splitting sequences that are longer than the the length of an input sequence T (see Equation 3.2) in PreCNet into smaller sequences with the length T. In figure 4.3, there is an example of splitting sequences for a network with $T = 5$.

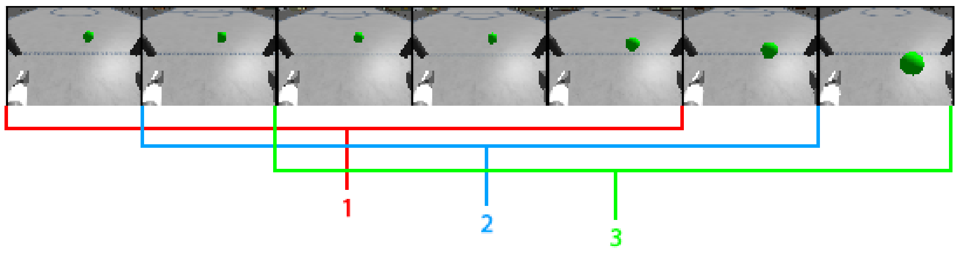


Figure 4.3: Splitting sequences from the experiment for the dataset. Let's say that the network has the length of input sequence T set to 5. A sequence from the experiment with 7 frames would be split to 3 sequences with the length of 5.

4.1.1 Variants of the experiment

The first two experiments maintain a constant velocity of the ball across all of the sequence. The sequence lengths are adjusted so that a 12 fps camera can track the movement of the ball while keeping individual sequences around 15-20 frames in the first experiment and around 15-30 frames in the second experiment. In the first experiment, the ball is slightly bigger than in the others and hits iCub's torso each time. In the second experiment, the ball misses the torso in roughly 10% of the sequences. The purpose of the second experiment is to reduce bias in time steps, where the tactile activation is expected. Therefore when the length of an input sequence T (see Equation 3.2) in PreCNet is set to 15 frames, after the extraction of all 15 frames subsequences from each 15-30 sequence (see Figure 4.3), the tactile activation can occur not only by the end of the sequence, but also at the beginning of it.

In another two experiments, the velocity of the ball across sequences was randomly generated. These experiments were designed with the goal to explore dynamical properties of peripersonal space representation, hence dynamical properties of safety body margin. The first experiment was intended for PreCNet with the sequence length of 8 frames. The second one for PreCNet with the sequence length of 15 frames. In the first experiment, the ball misses the iCub torso in around 15% of the sequences, meanwhile, in the second it is around 10%.

To sum up, four experiments were created:

- Experiments with static velocity across sequences
  - 1. Sequence lengths 15-20, all balls are looming towards iCub's torso (final dataset contains 3560 sequences). The ball is slightly bigger than in other experiments.
  - 2. Sequence lengths 15-30, in 10% of sequences the ball misses iCub's torso (final dataset contains 11100 sequences)
- Experiments with changing velocity across sequences
  - 3. Sequence lengths 8-12, in 5% of sequences the ball misses iCub's torso (final dataset contains 8260 sequences)
  - 4. Sequence lengths 15-30, in 10% of sequences the ball misses iCub's torso (final dataset contains 13520 sequences)

4.1.2 Dataset generation

Raw csv images and tactile activations, from each experiment (see Section 4.1.1), are processed using `parse_input_files.py` [PS20c]. The script synchronizes visual and tactile data by their sequence numbers, producing arrays of images and tactile activations (see Figure 4.5).

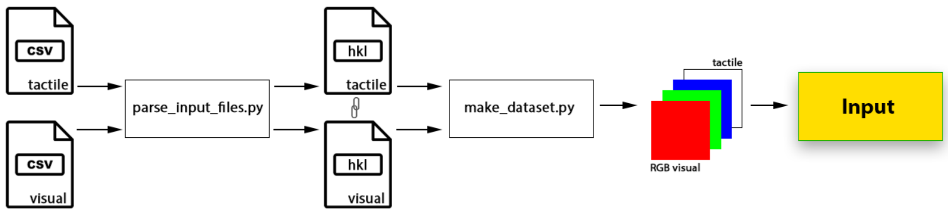


Figure 4.4: Input of the network. Raw .csv data are synchronized by `parse_input_files.py` and furthered processed with `make_dataset.py` to create input frames for the PreCNet network (see Figure 3.2)

These arrays must be further processed with `make_dataset.py` [PS20c] (see Section 4.1.3) in order to be compatible with PreCNet neural network. Each input data frame has 4 channels of 2D arrays. First 3 channels, are composed of the RGB image and optional zero paddings on the sides[3]. The last channel is used for the tactile data (see Figure 4.4).

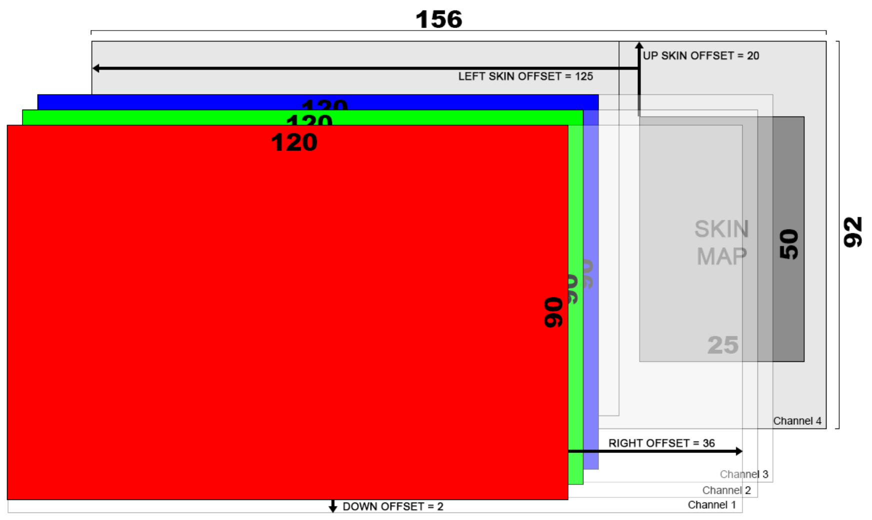


Figure 4.5: Format of one frame from the input dataset. Each sequence has 4 channels of 2D arrays. First 3 channels are composed of RGB image with offsets on the right side and down side, that are filled with zeros. The last channel has zeros everywhere, but the skin map (see Section 4.1.3). The position of the skin map is determined by the `LEFT SKIN OFFSET` and the `UP SKIN OFFSET` parameters.

[3]see `__define_basic_grid()` in `make_dataset.py` [PS20c]

Finally, each dataset is split, providing 90% of the sequences for training, 5% for validation and 5% for testing.

4.1.3 Description of tactile data

The parsing script `make_dataset.py` [PS20c] allows the user to create the 2D data representation of the tactile modality that will be referred to as 'skin map'.

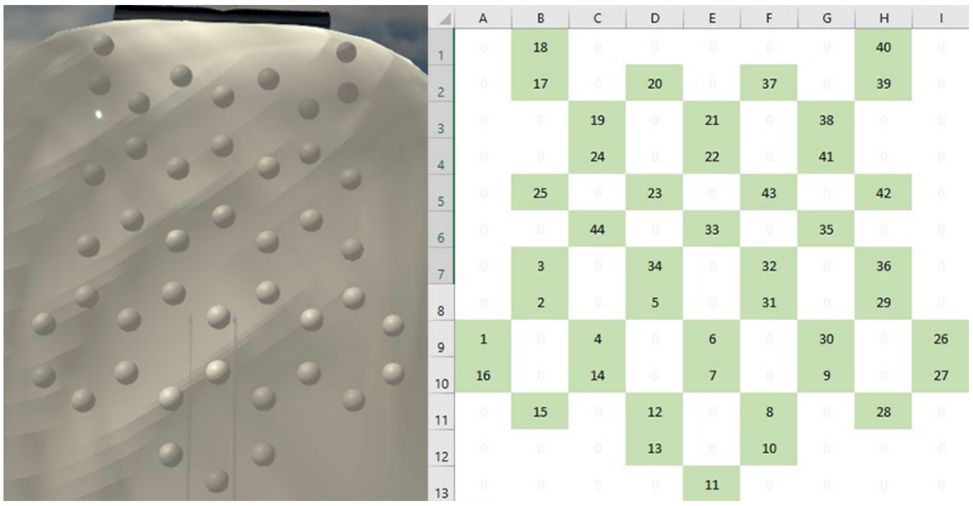
Data from tactile arrays are then filled to the 2D skin map (see the last channel in Figure 4.4) according to their numbers as follows[4]:

- 0 – no tactile sensor
- 0.5 – no activation of the sensor
- 1 – sensor was activated

At first .xlsx file is created, that maps tactile activations to a skin map. All of the experiments (see Section 4.1.1) used the default mapping (see Figure 4.6) based on the placement of sensors in the experiment. The file is then converted into 2D NumPy array and can be further modified[5]. In most of the experiment, the default mapping was resized by the nearest neighbour interpolation to be 25x50 (width x height). One can set the zero offsets to each side of the image and therefore define the positioning of the image within the grid. It is important, that both dimensions of the final grid are divisible by 4 (i.e., Figure 4.4) due to the pooling and upsampling layers. Finally, the created skin map is placed within the grid by deciding, where the left upper pixel of the skin map will be put, by modifying the `LEFT SKIN OFFSET` and `UP SKIN OFFSET` parameters.

[4]Since PreCNet normalizes the input by dividing all values in the array by 255, skin map is multiplied by 255 after mapping to their corresponding values

[5]When the original array is modified, a user must also define inverse operations, that will be performed on the output of the network. See 'README.md' [PS20c] for further details.



	A	B	C	D	E	F	G	H	I
1	0	18	0	0	0	0	0	40	0
2	0	17	0	20	0	37	0	39	0
3	0	0	19	0	21	0	38	0	0
4	0	0	24	0	22	0	41	0	0
5	0	25	0	23	0	43	0	42	0
6	0	0	44	0	33	0	35	0	0
7	0	3	0	34	0	32	0	36	0
8	0	2	0	5	0	31	0	29	0
9	1	0	4	0	6	0	30	0	26
10	16	0	14	0	7	0	9	0	27
11	0	15	0	12	0	8	0	28	0
12	0	0	0	13	0	10	0	0	0
13	0	0	0	0	11	0	0	0	0

Figure 4.6: Default mapping of tactile sensors. (a) Placement of sensors in the experiment on iCub's torso (b) Default mapping in .xlsx file

4.2 Learning parameters

To better understand the strategies of the network for tactile stimuli prediction, I ran the training on each dataset several times, while changing the *tactile false negative* parameter, described in Section 3.2. Apart from this parameter, all other parameters remained the same during all experiments. Structure of the network remained the same as in the article [SSH20] (see Table 4.1).

		conv_i		$\text{convLSTM}_\text{i}^{\text{up/down}}$	
module	weight λ_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

Table 4.1: Parameters defining the structure of PreCNet. Table taken with permission from [SSH20]. λ_i is module constant (see Equation 3.2). In all convolutions (inc. convLSTM), padding was set to keep the original shape ('same' option)

Each input array is normalized to the range [0,1] by dividing it by 255. Since the structure and dynamics of the surroundings are far simpler than in the KITTI dataset [GLSU13], using images from the real-world camera as in the article [SSH20], I trained the networks on 100 episodes, which was sufficient in each training run. The network was trained with learning rate 0.001, using Adam optimizer with default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) [KB14]. In some cases, when the loss started to diverge, the learning rate was reduced to 0.0001 after the first 15 episodes. Length of input sequences

T were 15 for all experiments, except the experiments with changing velocity, in which sequences were around 8-12 frames (see Section 4.1.1), where the length of sequences T was set to 8. The time constant (see Section 3.2) μ_t was set to 0 for the first frame and $\frac{1}{\mathrm{nt}-1}$ for all other frames.

4.3 Evaluation description

The authors of PreCNet used widely known metrics (i.e., MSE, SSIM, PSNR) to evaluate how well can PreCNet predict images [SSH20]. These metrics are still used to evaluate images, outputted by the network. However, these metrics cannot be applied to the constructed skin map (see Section 4.1.3). Increasing the *tactile false negative* parameter described in Section 3.2 strongly influences how early the network predicts the stimuli, but also how many redundant tactile sensors are activated. By trying several configurations, one cannot effectively rely only on the visualisation of the results. Since the network should predict the tactile stimuli both temporally (with respect to time), but also spatially (with respect to space), I created 2 tables described below, for each instance of the network. Finally, the network creates prediction plots for each tested sequence, for both visual and tactile data, comparing actual and predicted response in each time step. In addition, one can use visualisation application (see Section 4.3.3) with the files created by the evaluation script[6].

4.3.1 Evaluating temporally

By predicting the tactile stimuli temporally, we understand that the network should predict the tactile stimuli in the actual time of activation. If the prediction in the actual time of activation is not possible, the network should predict the stimulus rather earlier than later. To evaluate, how well can PreCNet predict the stimuli temporally, the Algorithm 3 (see Table 4.2) is proposed. It compares the frame from time step, in which the first tactile prediction occurred to the frame with the actual first tactile activation.

If the network predicts the tactile stimulus in the frame, when the actual stimulus occurred, this sequence is marked as 0. When the network activates earlier than the actual first tactile stimulus, the sequence is labelled with a negative number depicting how many frames ahead did the network activated.

[6]Evaluation script is available at 'icub_evaluate.py' [PS20c]

Algorithm 3 Temporal evaluation of the network. n is the number of sequences, function $\mathrm{fist}_{\mathrm{tact}}(\mathrm{s})$ gets the index of the time step, when the first tactile activation in the sequence s occured. $\mathrm{A_s}$ denotes the actual s-th sequence, $\hat{\mathrm{A}}_\mathrm{s}$ denotes the predicted s-th sequence. The operation '/' is elementwise division. Lastly the $\mathrm{e_{temp}}$ indicates, that the error is temporal.

```
$e_{temp} = 0$
for $s = 0, \dots, n$ do
    $i = \mathrm{first_{tact}}(\hat{\mathrm{A}}_\mathrm{s}) - \mathrm{first_{tact}}(\mathrm{A_s})$
    $e_{temp}(i) \leftarrow e_{temp}(i) + 1$
end for
$e_{temp} \leftarrow e_{temp}/n$
```

Eventually, when the network predicts activation after the actual tactile stimulus, the sequence is marked with a positive number denoting the delay of our activation. Finally, the sequences from testing dataset in each category are summed up and divided by the total number of sequences.

i	**-4**	**-3**	**-2**	**-1**	**0**	**1**
$\mathbf{e_t(i)}$	1%	5%	19%	43%	24%	9%

Table 4.2: Example of temporal tactile evaluation of a trained PreCNet network instance. In this example, the first tactile activation was predicted in the same time step as the first actual activation in 24% of the sequences. In 43% of the sequences the model predicted the first tactile activation one time step before it actually occurred (similar for -2, -3 and -4 columns). In 9% of all sequences, the model predicted the first tactile activation one time step after it happened.

For sequences, in which the activation did not occur (the ball missed iCub's torso), the evaluation script marks if any tactile sensor was activated during the sequence. This part of a table will be marked as 'MISS'.

4.3.2 Evaluating spatially

In order to find out if some redundant sensors were activated in the prediction or eventually some sensors should have been activated but were not, the network is evaluated by means of its ability to predict the tactile stimuli spatially. As it is desired that the network predicts the tactile stimulus before it actually happens, it would make no sense to compare the predicted skin map, with the actual skin map, which has no activation, in the corresponding time step. For this reason, all predictions of tactile stimulation that occur before the actual first tactile activation are compared to the first skin map, in which some activation is detected. After that, all other predicted skin maps are compared with the actual tactile skin maps from the corresponding time step (see Figure 4.7). For this task, I propose an algorithm (see Algorithm 4) that evaluates the tactile predictions spatially.

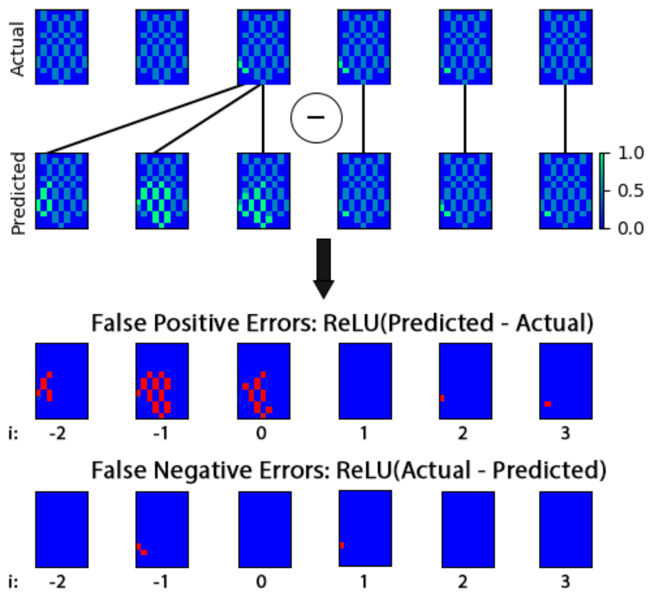

Actual
Predicted
1.0
0.5
0.0
False Positive Errors: ReLU(Predicted - Actual)
i:
-2
-1
0
1
2
3
False Negative Errors: ReLU(Actual - Predicted)
i:
-2
-1
0
1
2
3

Figure 4.7: Explanation of spatial evaluation. Predicted tactile activations, that occurred before and at the during the actual tactile activation are subtracted by the first tactile activation frame, whereas other frames are paired with the activation from corresponding time step. This produces false-positive and false-negative errors

This yields false-positive errors and false-negative errors[7] for each time step, relatively to the time step of the first actual tactile activation.

Finally, for each relative time step i, the errors are summed and divided by the counts of sequences in each of these relative time steps n_i. These mean errors for each time step relative to the first tactile activation i are put into a table (see example Table 4.3).

Type	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**
$\mathbf{e_{pos}}$	3.777	2.560	3.208	2.536	1.019	0.196	0.050	0.003
$\mathbf{e_{neg}}$	0.636	0.929	1.194	0.048	0.001	0.000	0.000	0.000

Table 4.3: Example of spatial tactile evaluation of a trained PreCNet network instance.

Important note: While the final code does the spatial evaluation after rounding the values in the tactile map to the values $\{0, 0.5, 1\}$, most of the networks were evaluated with the script without this rounding. By default, it is meant that there was no rounding in the spatial evaluation. The only networks evaluated with rounding are the networks from Table 5.4.

[7]This categorization of false-positive and false-negative errors is similar as it was described in Section 4.1.3.

Algorithm 4 Spatial evaluation of the network. n is the number of sequences, $\mathrm{n_i}$ is the number of sequences, that were summed into error e(i). Function $\mathrm{fist_{tact}(s)}$ gets the index of the time step, when the first tactile activation in the sequence s occurred. Oppositely $\mathrm{last_{tact}(s)}$ gets the index of the time step, when the last tactile activation in the sequence s occurred. $\mathrm{A_s}$ denotes the actual s-th sequence, $\mathrm{\hat{A}_s}$ denotes the predicted s-th sequence. $\mathrm{A_s(i)}$ is i-th frame in the the actual s-th sequence, $\mathrm{\hat{A}_s(i)}$ denotes the i-th frame in the predicted s-th sequence. SUM() function, sums all the values in the map. The operation '/' is division. Lastly the $\mathrm{e_{pos}}$ stands for false positive error and $\mathrm{e_{neg}}$ stands for false negative error.

```
$e_{pos} = 0$
$e_{neg} = 0$
for $s = 0, \dots, n$ do
    $a = \mathrm{first_{tact}(A_s)}$
    for $p = \mathrm{first_{tact}(\hat{A}_s)}, \dots, \mathrm{last_{tact}(\hat{A}_s)}$ do
        $i = p - a$
        if $i <= 0$ then
            $e_{pos}(i) \leftarrow e_{pos}(i) + \mathrm{SUM}(\mathrm{ReLU}(\mathrm{\hat{A}_s}(i)) - \mathrm{A_s}(a))$
            $e_{neg}(i) \leftarrow e_{neg}(i) + \mathrm{SUM}(\mathrm{ReLU}(\mathrm{A_s}(i)) - \mathrm{\hat{A}_s}(a))$
        else
            $e_{pos}(i) \leftarrow e_{pos}(i) + \mathrm{SUM}(\mathrm{ReLU}(\mathrm{\hat{A}_s}(i) - \mathrm{A_s}(i)))$
            $e_{neg}(i) \leftarrow e_{neg}(i) + \mathrm{SUM}(\mathrm{ReLU}(\mathrm{A_s}(i) - \mathrm{\hat{A}_s}(i)))$
        end if
    end for
end for
for each i do
    $e_{pos}(i) \leftarrow e_{pos}(i)/n_i$
    $e_{neg}(i) \leftarrow e_{neg}(i)/n_i$
end for
```

4.3.3 Qualitative Evaluation

After evaluating the test dataset with the metrics mentioned above, generated images and tactile activations are put into visualisation application[8]. As can be seen in Figure 4.8, the application offers side by side comparison of actual response, predicted response and difference between these 2 responses.

[8] Visualisation application is available at `visualise_data.py` [PS20c]

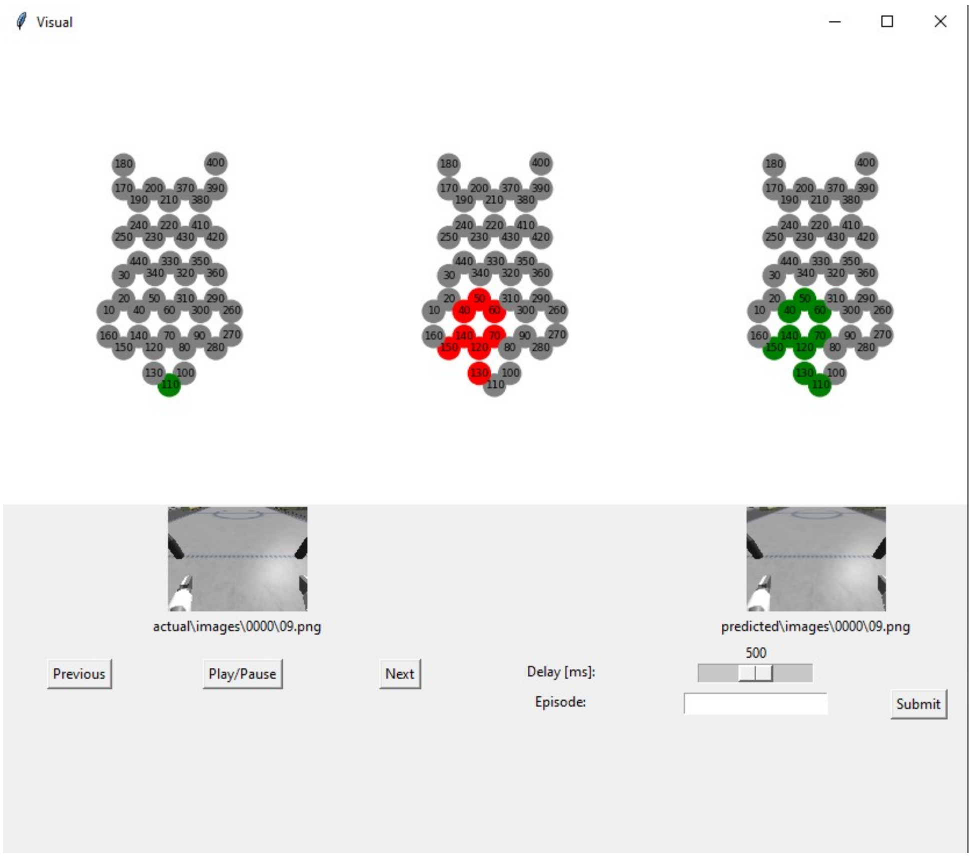


Figure 4.8: Visualisation application – actual (left), predicted (right) tactile and visual responses. Difference in tactile responses (middle)

4.3.4 Common strategies for predicting tactile stimulation

After a few trained models, I observed that three strategies are repeating very often with slight modifications. In this section, I will describe the main characteristics of these strategies that can be recognized from the quantitative and qualitative evaluation. I named these strategies consequently:

- Copy Previous Frame Strategy
- Fixed Map Strategy
- Visuo-tactile Strategy

The first strategy was named **Copy Previous Frame Strategy**, as it only copies the actual tactile response from the previous frame (see Figure 4.9).

Therefore all tactile stimuli are activated one frame after the actual tactile activation. This can be seen from the temporal evaluation (see Table 4.4)

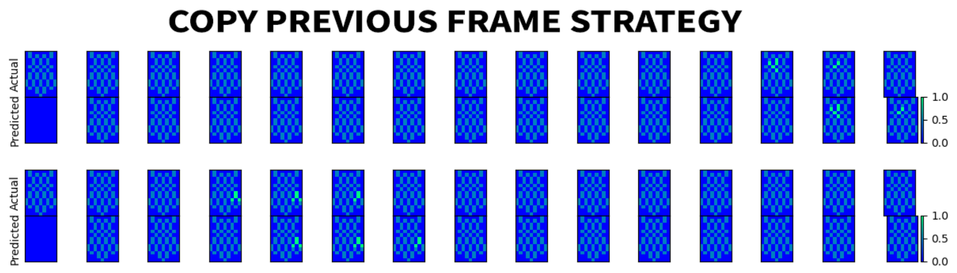


Figure 4.9: Copy Previous Frame Strategy – PreCNet only copies the previous actual frame to the output.

-2	**-1**	**0**	**1**	**2**
0%	0%	0%	100%	0%

Table 4.4: Typical values of the Copy Previous Frame Strategy – Temporal evaluation table, commonly observed in networks implementing the Copy Previous Frame Strategy.

Another strategy is called **Fixed Map Strategy**, because it applies a fixed 'map' of activations, in time steps where the activation is expected (where the activation mostly occurs), which is same for all sequences (see Figure 4.10). It is very often combined with the Copy Previous Frame Strategy. Typically it predicts the actual stimulus many frames ahead (see Table 4.5), producing big false positive errors.

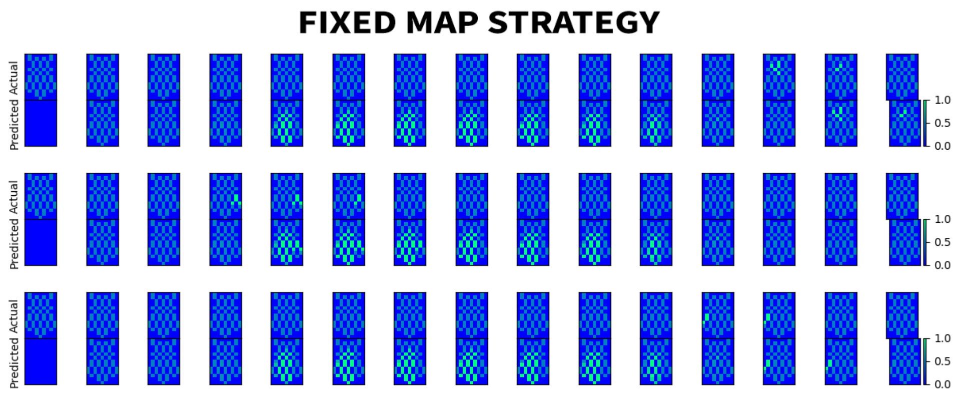


Figure 4.10: Fixed Map Strategy – PreCNet applies a fixed skin map to the time steps, where the activation is expected, besides copying the previous actual activation

-12 to -7	**-6 to -1**	**0**	**1**
1% to 7% ea.	10% to 14% ea.	6%	1%

Table 4.5: Typical values of Fixed Map Strategy – Temporal evaluation table, commonly observed in networks implementing the Fixed Map Strategy.

The last strategy was named **'Visuo-tactile Strategy'**. The name of this strategy, came from the observations since the network in most cases predicted the tactile stimulus closely before it actually occurred (see Figure 4.11), without placing fixed maps as in the 'Fixed Map Strategy'. This strategy will be furthered discussed in Section 6.

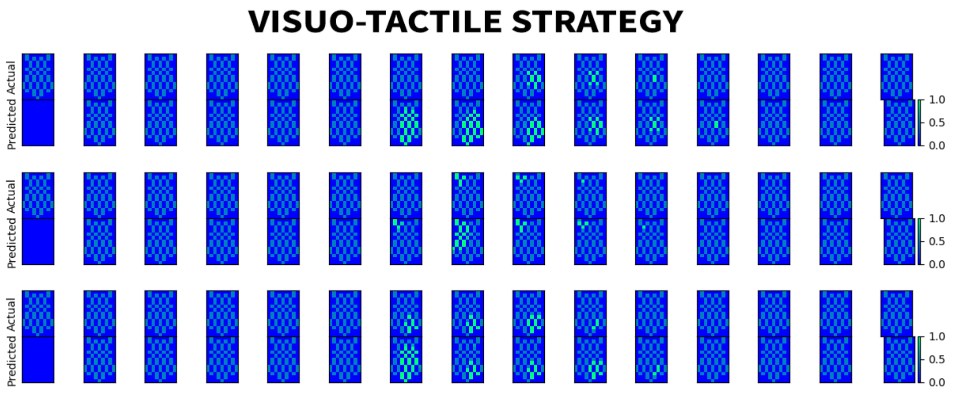


Figure 4.11: Visuo-Tactile Strategy – PreCNet predicts the incoming stimulus before it occurs. Notice that in the third sequence, the first predicted tactile activation (timestep 7) is produced before "experiencing" the actual tactile activation in this timestep.

Chapter 5

Results and their interpretation

In Section 4.1.1, I described four experiments created in the framework running on the Neurorobotics Platform [Pro] [FVA$^+$17]. These datasets were then used for training extended PreCNet neural networks with various parameters. In order to evaluate these network quantitatively, while preserving most of the information about the behaviour of the network, by means of temporal and spatial prediction, an evaluation system was proposed (see Section 4.3). Since this evaluation system is not standard, the reader may find it hard to interpret the results just by looking at the tables. Therefore, I will interpret these results after presenting them. I would like to note, that I will leave out some columns replacing them with '..' or merge them, to maintain readability. Full list of evaluated models, with prediction plots for some of them (see an example in Figure 5.1) is published online [PS20a][1].

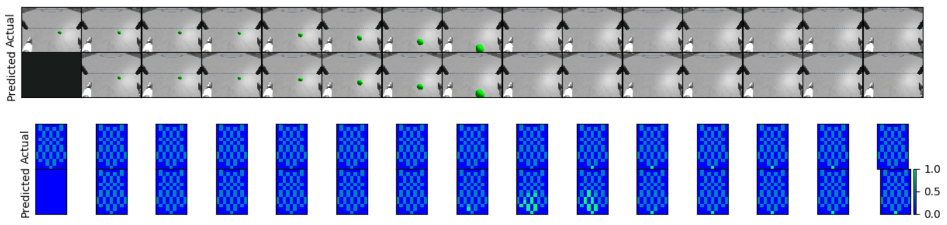


Figure 5.1: Example of prediction plots – Visual data in upper sequence containing the actual images in the upper row and predicted images in the lower row. Tactile data in lower sequence containing the actual tactile stimulation in the upper row and predicted in the lower row.

After training a few models I noticed that the predicted images are maintaining the popular structural similarity index (SSIM) [WBSS04] around

[1]Evaluation of the networks can also be found on the enclosed CD.

0.97-0.98, whereas 1 is the maximum value. Qualitative evaluation of these models confirmed that the network has no issues predicting images. For this reason, I will evaluate the trained models by their ability to predict tactile stimuli, although the network had to predict images as well. Nevertheless, for those interested these values can be found for some of the networks in the same folder as their prediction plots, published online [PS20a].

Since tactile strategies of the trained models showed very similar characteristics, I decided to put these strategies in three groups, namely the 'Previous Frame Strategy, 'Fixed Map Strategy' and the 'Visuo-Tactile Strategy'. All of these strategies are described in Section 4.3.4. The value of *tactile false positive* – 'fp' parameter (see Section 3.2) was set to 1 in all models, except the those in Experiment 4 (Section 5.2.2), where it is explicitly noted (i.e., fn80_fp2).

5.1 Experiments with constant velocity of the looming object across sequences

5.1.1 Experiment 1

Data from the first experiment was used for training PreCNet with the length of input sequences T (see Equation 3.2) set to 15. tactile map was resized to 25x50 (width x height). Neural networks produced following results for various *tactile false negative* – 'fn' parameters (see Section 3.2).

fn	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**
1						99%	1%
12						100%	
15						100%	
22		2%	6%	28%	53%	11%	
27	1%	4%	8%	34%	43%	9%	
30	1%	5%	19%	43%	24%	9%	

Table 5.1: Temporal evaluation of the first experiment. First three models with *tactile false negative* – 'fn' parameter set to 1, 12 and 15 predicted all of the tactile activations one frame later than it actually occurred. Further increasing the 'fn' parameter helped to change the strategy of the network, as it predicted the tactile stimulus when it actually occurred or even before it occurred.

fn	**type**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	..
1	pos						0.23	0.34	0.13	0.10	0.08	0.03	
	neg						0.25	0.73	0.43	0.12	0.06	0.03	
12	pos						0.86	1.74	1.93	1.53	1.17	0.79	
	neg						0.40	0.41	0.33	0.41	0.38	0.33	
15	pos						0.22	1.25	0.92	0.56	0.22	0.07	
	neg						0.26	0.07	0.05	0.06	0.04	0.04	
22	pos		1.88	1.33	0.89	5.53	1.80	1.43	1.04	0.62	0.28	0.13	
	neg		2.73	1.43	1.07	0.57	0.12	0.04	0.00	0.01	0.00	0.00	
27	pos	2.61	4.64	2.16	1.50	5.84	1.99	1.42	1.00	0.60	0.25	0.09	
	neg	2.58	1.62	1.79	1.01	0.55	0.05	0.03	0.00	0.01	0.00	0.00	
30	pos	9.61	4.93	2.81	3.89	7.42	5.71	3.67	2.10	1.23	0.70	0.45	
	neg	1.25	1.54	1.43	0.80	0.31	0.04	0.03	0.00	0.01	0.00	0.00	

Table 5.2: Spatial evaluation of the first experiment. Increasing the *tactile false negative* – 'fn' parameter increases also the false positive errors of the network, because the network activates more sensors earlier.

Here the first 3 networks implemented the 'Copy Previous Frame Strategy'. Further increasing the 'fn' parameter helped the network to change the default 'Copy Previous Frame Strategy' to the 'Visuo-Tactile Strategy'. Especially in the network with 'fn' set to 22, only 11% of all sequences were not predicted in time (see Table 5.1). Further increasing the 'fn' parameter, decreased the number of sequence, that were predicted too late, by 2% (see Table 5.1) and reduced the false-negative error. Nonetheless, it increased the false-positive error, in some cases doubling it by activating more tactile sensors (see Table 5.2). Figure 5.2 illustrates some of the episodes of the model, with 'fn' parameter set to 22. All prediction plots from test dataset can be found online at [PS20a].

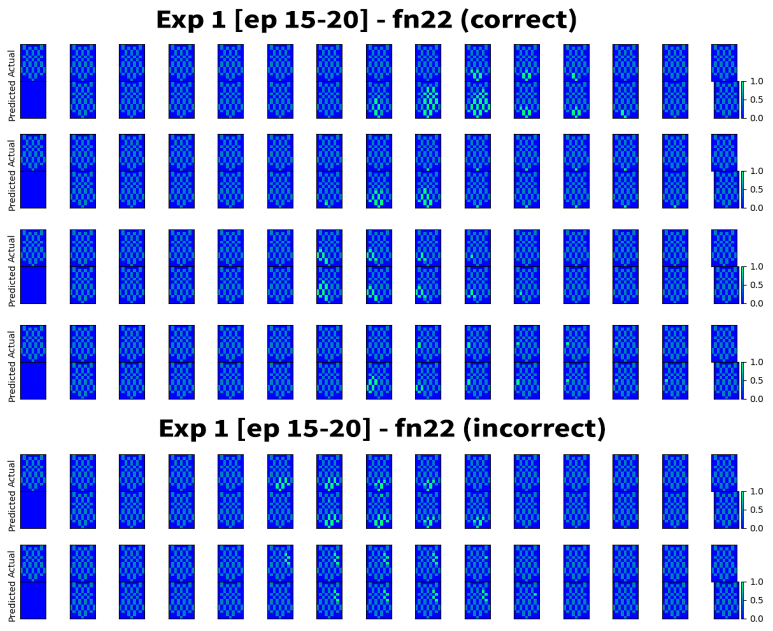


Figure 5.2: Some sequences of tactile activity of network fn22 – Actual and predicted tactile activations of fn22 network. In the upper sequences, the network predicted the tactile stimulus correctly. In the sequences below, the network made the prediction of tactile activity too late.

In most of the sequences, the model predicted the tactile stimulus closely before the actual stimulus or at the time of the actual stimulus (see Figure 5.2), without activating the same sensors in all sequences as in the 'Fixed Map Strategy'. It also activated more sensors when the first actual tactile stimulus was most likely to occur. This can be also observed from Table 5.2, as the false-positive error increases in the 0th frame.

5.1.2 Experiment 2

The same parameters of the network as in the previous experiment were used to train the network on sequences from the second experiment. These are the results of various *tactile false negative* – 'fn' parameters.

	HIT												MISS	
fn	**-12**	**-11**	**-10**	**-9**	**-8**	**-7**	**..**	**-3**	**-2**	**-1**	**0**	**1**	**FALSE**	**TRUE**
45												100%		100%
50												100%		100%
55												100%		100%
60								4%	23%	19%	16%	38%	74%	26%
70				1%	3%	8%	..	13%	14%	13%	8%	2%	100%	
80	1%	2%	8%	1%	3%	8%	..	15%	15%	12%	7%	1%	100%	

Table 5.3: Temporal evaluation of the second experiment. This table is similar as Table 5.1, but since the ball missed iCub's torso in some sequences, these sequences are evaluated separately (see end of the Section 4.3.1). Note: In the frames between -7 and -3 the last two networks had around 12% in each column.

fn	**type**	**-10**	**-9**	**-8 to -5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**..**
45	pos									0.4	0.3	0.3	0.1	
	neg									0.4	0.2	0.1	0.0	
50	pos									0.4	0.3	0.3	0.1	
	neg									0.4	0.2	0.1	0.0	
55	pos									0.4	0.3	0.3	0.1	
	neg									0.4	0.2	0.1	0.0	
60	pos				1.3	0.3	1.7	2.9	2.5	2.7	2.0	1.5	1.0	
	neg				3.0	0.9	0.4	0.4	0.1	0.2	0.1	0.0	0.0	
70	pos		5.8	3.1 – 4.3	3.8	3.4	3.9	4.5	4.5	4.3	3.0	2.3	1.4	
	neg		0.8	∼ 0.5	0.5	0.4	0.3	0.2	0.0	0.1	0.0	0.0	0.0	
80	pos	4.0	7.0	3.0-4.4	4.1	3.5	4.0	4.9	5.1	4.6	3.0	2.4	1.4	
	neg	0.0	0.6	0.3-0.4	0.4	0.4	0.3	0.2	0.1	0.0	0.0	0.0	0.0	

Table 5.4: Spatial evaluation of the second experiment – Note: Before calculating the spatial errors, predicted tactile activations were rounded to 0, 0.5, 1 (see Section 4.3.2)

The first 3 models implemented the 'Copy Previous Frame Strategy'. These models "wait" for the tactile activation to occur and after that copy it to the new predicted frame. Therefore in sequences, where the activation did not occur, the models did not activate any tactile sensor. For this reason, they predicted all of the sequences, where the activation did not happen correctly (see MISS part in Table 5.3).

The network with 'fn' parameter set to 60 did not implement any of the "undesirable" strategies, rather it implemented the 'Visuo-Tactile Strategy'.

In some sequences, predicted sensors were activated closely before the actual stimulus or at the time of the actual stimulus. The false-positive error is slightly higher in the -1st and 0th frame (see Table 5.4), thus in the frames where the activation is most likely to happen what can be seen in prediction plots as well in Figure 5.3. This is similar to the network from Experiment 1 (see Table 5.2). Moreover, this model correctly predicted 26% of sequences, which did not activate any sensor on iCub's torso (see Table 5.3) or activated only a few sensors with lower certainty as in Figure 5.4.

Finally, last 2 models with the 'fn' parameters set to 70 and 80 implemented the 'Fixed Map Strategy' (see Section 4.3.4), what can be seen from very early predictions with very high false-positive errors (many redundant sensors were activated).

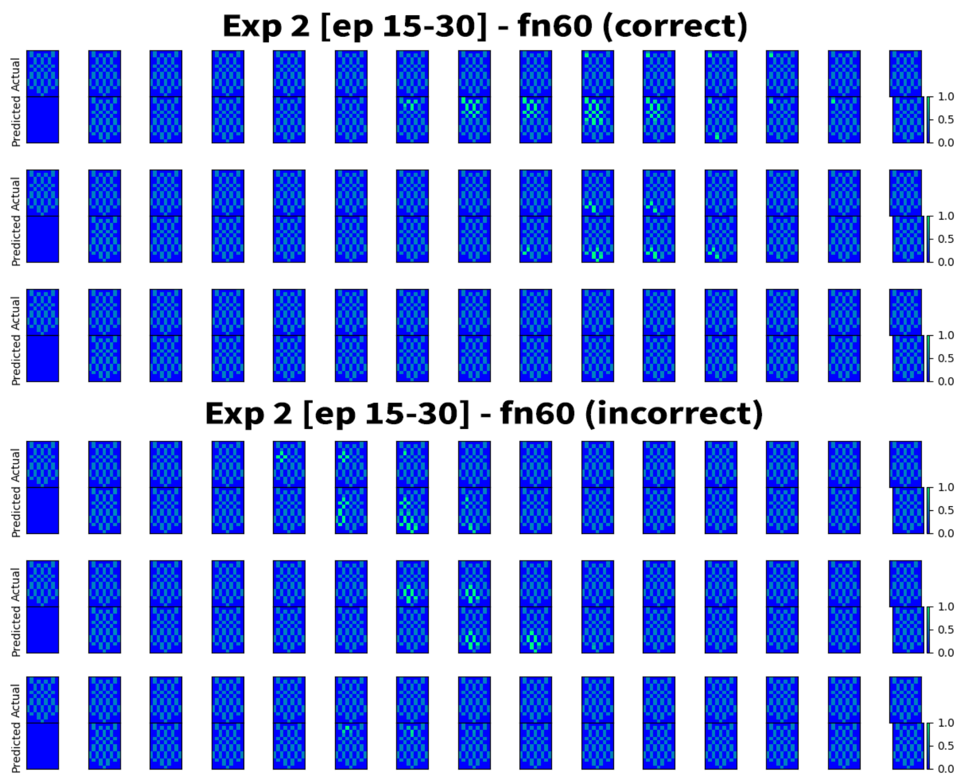


Figure 5.3: Some sequences of tactile activity of the network fn60 – Actual and predicted tactile activations of fn60 network. In the upper sequences, the network predicted the tactile stimulus correctly. In the last correct sequence, the network correctly predicted, that the ball will not hit the torso. In the sequences below, the network made the prediction of tactile activity too late. In the last sequence, network predicted a stimulus with lower certainty than in other sequences, meanwhile, the ball missed iCub's torso.

For another result of this experiment, I will use the same table as Table 5.3, but with higher precision, showing only the frames after the first actual tactile activation (see Table 5.5). With increasing 'fn' parameter by implementing the 'Fixed Map Strategy' models further minimized the false negative error of time steps after the first actual activation. This point will be considered later in the discussion (see Section 6).

fn	**type**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
45	pos		0.362	0.265	0.349	0.138	0.043	0.096	0.057	0.063
	neg		0.370	0.171	0.082	0.044	0.064	0.074	0.069	0.015
50	pos		0.362	0.265	0.349	0.138	0.043	0.096	0.057	0.063
	neg		0.370	0.171	0.082	0.044	0.064	0.074	0.069	0.015
55	pos		0.362	0.265	0.349	0.138	0.043	0.096	0.057	0.063
	neg		0.370	0.171	0.082	0.044	0.064	0.074	0.069	0.015
60	pos	2.474	2.684	2.043	1.491	0.974	0.600	0.644	0.422	0.193
	neg	0.097	0.177	0.091	0.037	0.012	0.037	0.040	0.026	0.000
70	pos	4.461	4.255	3.038	2.295	1.402	0.789	0.641	0.518	0.271
	neg	0.043	0.064	0.009	0.007	0.011	0.004	0.013	0.000	0.000
80	pos	5.147	4.636	3.022	2.351	1.440	0.999	0.759	0.471	0.197
	neg	0.060	0.031	0.015	0.013	0.002	0.010	0.010	0.002	0.000

Table 5.5: Spatial evaluation of the second experiment (part 0 to 8 frames—after the first tactile activation) – Note: Before calculating the spatial errors, predicted tactile activations were rounded to 0, 0.5, 1 (see Section 4.3.2)

5.1.3 Prediction strategies from experiment 1 and experiment 2

Results from the first two experiments indicate that increasing 'fn' parameter leads from the 'Copy Previous Frame Strategy' to 'Visuo-Tactile Strategy' and finally to the 'Fixed Map Strategy'.

To sum up the different network strategies on these two datasets, let me highlight 4 trained models in Table 5.6.

Dataset	**fn**	**-3**	**-2**	**-1**	**0**	**1**	**Strategy**
1. [ep15-20]	**15**					100%	**previous frame**
1. [ep15-20]	**22**	2%	6%	28%	53%	11%	**visuo-tactile**
2. [ep15-30]	**60**	4%	23%	19%	16%	38%	**visuo-tactile**

Dataset	**fn**	**-12 to -7**	**-6 to -1**	**0**	**1**	**Strategy**
2. [ep15-30]	**80**	1% to 7% ea.	10% to 14% ea.	6%	1%	**fixed map**

Table 5.6: Temporal evaluation of highlighted networks – The network implemented 3 different strategies (see Section 4.3.4). In the last row, some columns were merged to preserve space.

5.2 Experiments with changing velocity of the looming object across sequences

The network was also tested on a dataset, where the velocity of the ball was generated randomly within a certain range. Experiments were intended for PreCNet networks with sequence lengths 8 and 15. Additionally to the previous experiments, datasets from these experiments were created by resizing the skin map to different values not only to 25x50.

5.2.1 Experiment 3

Skin maps from the third experiment were resized to 25x50 and 63x91 (w x h). For both datasets, I used sequence length T set to 8. Training on these datasets produced the following results.

	HIT					**MISS**	
fn	**-3**	**-2**	**-1**	**0**	**1**	**FALSE**	**TRUE**
12					100%		100%
20					100%		100%
22					100%		100%
23	6%	22%	36%	34%	2%	100%	
25	6%	22%	36%	34%	2%	100%	
30	6%	22%	36%	34%	2%	100%	
40	6%	22%	36%	34%	2%	100%	

Table 5.7: Temporal evaluation of the third experiment – skin 25x50

In general, the results of trained models on both datasets were very similar. The models at first implemented the 'Copy Previous Frame Strategy' for lower 'fn' values and then switched to the 'Fixed Map Strategy' with higher 'fn' values afterwards (see Tables 5.7 and 5.9). Differing from the experiments, where the velocity of the ball was constant across episodes (see Section 5.1), the gap between these two strategies with increasing 'fn' value is much smaller (compare Tables 5.8 and 5.10 with Table 5.2 or 5.4). For example, in the dataset with skin map size 25x50 changing the value of 'fn' parameter from 22 to 23, changed the strategy completely. Similarly to the experiments with constant speed, with increasing 'fn' parameter value the models minimized false-negative errors, while increasing false positive by activating more sensors and activating them earlier.

As an illustration for the implemented strategy, let's look at Figure 5.4. Here

fn	type	-3	-2	-1	0	1	2	3	..
12	pos					1.03	0.72	0.09	
	neg					0.06	0.00	0.00	
20	pos					1.04	0.74	0.09	
	neg					0.06	0.00	0.00	
22	pos					1.15	0.87	0.18	
	neg					0.06	0.00	0.00	
23	pos	3.27	3.78	2.56	3.21	2.54	1.02	0.20	
	neg	0.27	0.64	0.93	1.19	0.05	0.00	0.00	
25	pos	5.68	5.53	4.47	5.29	4.60	2.64	0.60	
	neg	0.13	0.38	0.59	0.77	0.03	0.00	0.00	
30	pos	7.57	7.01	5.94	7.02	5.97	2.87	0.60	
	neg	0.10	0.27	0.41	0.55	0.03	0.00	0.00	
40	pos	9.86	9.05	6.79	9.18	6.67	2.72	0.58	
	neg	0.07	0.17	0.26	0.37	0.03	0.00	0.00	

Table 5.8: Spatial evaluation of the third experiment – skin 25x50

	HIT					MISS	
fn	**-3**	**-2**	**-1**	**0**	**1**	**FALSE**	**TRUE**
25					100%		100%
27	6%	22%	36%	34%	2%	100%	
30	6%	22%	36%	34%	2%	100%	
40	6%	22%	36%	34%	2%	100%	

Table 5.9: Temporal evaluation of the third experiment – skin 63x91

fn	type	-3	-2	-1	0	1	2	3	4	5
25	pos					1.01	0.73	0.07	0.01	0.00
	neg					0.06	0.00	0.00	0.00	0.00
27	pos	6.01	6.48	4.50	5.80	4.60	1.75	0.22	0.04	0.00
	neg	0.11	0.28	0.49	0.75	0.03	0.01	0.00	0.00	0.00
30	pos	6.27	5.82	5.03	5.78	5.26	3.30	0.79	0.13	0.01
	neg	0.15	0.38	0.56	0.71	0.03	0.00	0.00	0.00	0.00
40	pos	8.24	7.43	6.55	7.56	6.74	3.66	0.37	0.03	0.00
	neg	0.10	0.25	0.36	0.49	0.03	0.00	0.00	0.00	0.00

Table 5.10: Spatial evaluation of the third experiment – skin 63x91

the network placed 2 static skin maps in the middle 2 frames implementing the 'Fixed Map Strategy', combining it with the 'Copy Previous Frame Strategy'. Since the original dataset consisted of sequences with frames in the range of 8-12 frames, the actual tactile activation occurred in the 3rd frame from the left rarely and the model did not predict this tactile stimulus. As a consequence of placing a static activation frame in the 4th frame, the model never predicts the tactile activation earlier than 3 frames before it actually occurs (see Table 5.7 and 5.9).

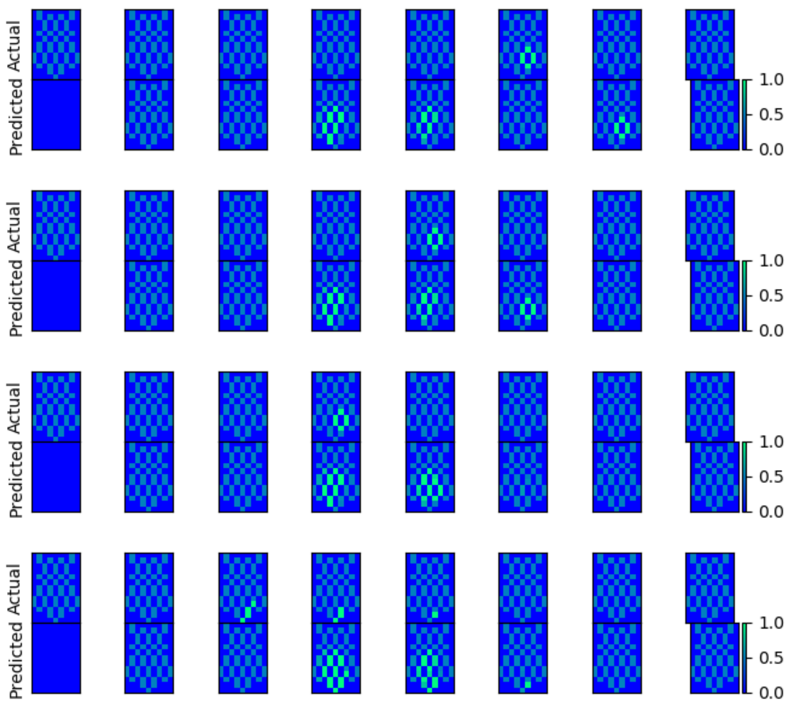


Figure 5.4: Some sequences of tactile activity of the network fn22 trained on the dataset with changning velocity and skin map resized to 25x50 – Actual and predicted tactile activations of the fn22 network. The network implemented the 'Fixed Map Strategy', with fixed skin maps in 2 middle frames, together with the 'Copy Previous Frame Strategy'.

5.2.2 Experiment 4

For this experiment, I trained networks with sequence lengths T set to 15. Skin maps were used with their default size 9x13, but they were also resized to 25x50 and 63x91 (w x h). Training of the networks yielded the following results.

	HIT							MISS	
params	**-9**	**-8**	**-7**	**-6**	**-5 to -1**	**0**	**1**	**FALSE**	**TRUE**
fn40							100%		100%
fn80	3%	6%	8%	10%	11% ea.	9%	10%	100%	
fn80_fp2							100%		100%
fn80_fp5							100%		100%
fn80_fp20							100%		100%

Table 5.11: Temporal evaluation of the fourth experiment – skin 9x13

In all three datasets the networks implemented the 'Copy Previous Frame Strategy' and changed to the 'Fixed Map Strategy' with increasing 'fn'

params	type	-9	-8 to -5	-4	-3	-2	-1	0	1	2	3	4	..
fn40	pos								0.6	1.1	0.8	0.2	
	neg								0.1	0.0	0.0	0.0	
fn80	pos	8.8	6.5-8 ea.	5.2	4.4	5.0	7.0	8.1	6.6	4.9	3.0	1.6	
	neg	0.4	0.5-0.6 ea.	0.6	0.6	0.6	0.6	0.6	0.1	0.0	0.0	0.0	
fn80__fp2	pos								0.6	1.1	0.7	0.2	
	neg								0.2	0.0	0.0	0.0	
fn80__fp5	pos								0.5	1.0	0.6	0.1	
	neg								0.2	0.0	0.0	0.0	
fn80__fp20	pos								0.5	1.0	0.6	0.1	
	neg								0.2	0.0	0.0	0.0	

Table 5.12: Spatial evaluation of the fourth experiment – skin 9x13

	HIT							MISS	
params	**-9**	**-8**	**-7**	**-6**	**-5 to -1**	**0**	**1**	**FALSE**	**TRUE**
fn60							100%		100%
fn69							100%		100%
fn80	3%	6%	8%	10%	11% ea.	9%	10%	100%	
fn80__fp2							100%		100%
fn80__fp5							100%		100%
fn80__fp20							100%		100%

Table 5.13: Temporal evaluation of the fourth experiment – skin 25x50

params	type	-9 to -5	-4	-3	-2	-1	0	1	2	3	4	..
fn60	pos							0.9	1.4	1.1	0.5	
	neg							0.1	0.0	0.0	0.0	
fn69	pos							0.7	1.2	0.9	0.3	
	neg							0.2	0.0	0.0	0.0	
fn80	pos	6.4-7.3 ea.	4.8	3.8	4.3	5.1	5.8	5.5	5.6	4.6	3.0	
	neg	0.3-0.5 ea.	0.5	0.5	0.6	0.7	0.9	0.1	0.0	0.0	0.0	
fn80__fp2	pos							0.5	1.0	0.6	0.1	
	neg							0.2	0.0	0.0	0.0	
fn80__fp5	pos							0.4	1.0	0.6	0.1	
	neg							0.2	0.0	0.0	0.0	
fn80__fp20	pos							0.4	0.6	0.2	0.0	
	neg							0.2	0.1	0.0	0.0	

Table 5.14: Spatial evaluation of the fourth experiment – skin 25x50

	HIT									MISS	
fn	**-10**	**-9**	**-8**	**-7**	**-6**	**-5 to -2**	**-1**	**0**	**1**	**FALSE**	**TRUE**
80									100%		100%
100		3%	6%	8%	10%	11% ea.	11%	9%	10%	100%	
120	3%	6%	8%	10%	11%	11% ea.	9%	7%	3%	100%	

Table 5.15: Temporal evaluation of the fourth experiment – skin 63x91

fn	type	-10	-9 to -6	-5	-4	-3	-2	-1	0	1	2	3	4	..
80	pos									0.5	1.0	0.6	0.1	
	neg									0.2	0.0	0.0	0.0	
100	pos		9.0-9.5	7.9	6.1	6.4	6.9	7.5	8.2	7.8	7.8	6.6	5.0	
	neg		∼0.2	0.3	0.3	0.3	0.3	0.4	0.5	0.0	0.0	0.0	0.0	
120	pos	10.2	10.1-10.7	9.1	7.9	8.1	8.5	8.8	9.2	8.7	8.4	7.1	5.4	
	neg	0.3	0.3-0.4	0.3	0.3	0.3	0.4	0.4	0.5	0.0	0.0	0.0	0.0	

Table 5.16: Spatial evaluation of the fourth experiment – skin 63x91

parameter (see Tables 5.11, 5.13 and 5.15). In datasets 9x13 and 25x50, I also increased the value of *tactile false positive* parameter. These models again implemented the 'Copy Previous Frame Strategy'. Spatial evaluations of these networks are provided in Tables 5.12, 5.14 and 5.16.

5.2.3 Prediction strategies from experiment 3 and experiment 4

In all instances, the network implemented either the 'Copy Previous Frame Strategy' or the 'Fixed Map Strategy'. To find out why any of the networks did not implement the 'Visuo-Tactile Strategy' as in the experiments in Section 5.1, I observed the predicted images. In contrast to the datasets, where the ball velocity remained the same across all sequences and the network predicted the ball position correctly, the ball in sequences with varying velocity was very blurry or vanished completely in some images (see Figure 5.5). Surprisingly in the case of PreCNet with 15 sequences, the network managed to predict the ball correctly, after it bounced from iCub's torso (see Figure 5.5c). This blurriness was not observed among images from experiments, where the velocity was constant. In Figure 5.5, I also included metrics for the comparison of actual and predicted image. Mean Square Error (MSE) should remain as low as possible, while the Structural Similarity Index (SSIM) [WBSS04] – whose values are between 0 and 1, and Peak signal-to-noise ratio (PSNR), the metric related to the MSE, should be as high as possible.

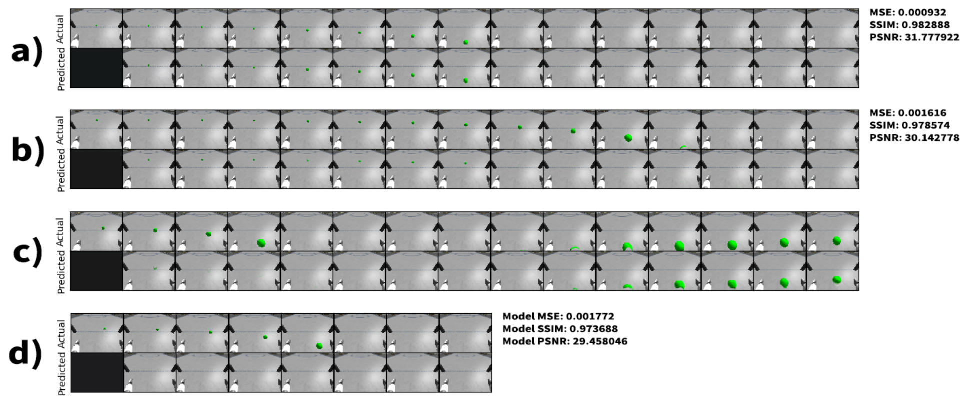


Figure 5.5: Comparison of trained networks on datasets with varying and constant velocity. On the right side, there is a network evaluation on the whole test dataset (MSE – Mean Square Error, SSIM – Structural Similarity Index [WBSS04], PSNR – Peak signal-to-noise ratio). **a)** Constant velocity throughout all sequences. **b)** Varying velocity through sequences. **c)** Same dataset as in b) – the network predicts images after the bounce from iCub's torso correctly. **d)** Varying velocity through sequences. PreCNet with sequence length 8.

Chapter 6

Discussion and future work

The main goal of this thesis was to train a neural network model based on predictive coding to predict tactile stimulation from visuo-tactile data, specifically images and tactile activations. For this purpose, the state-of-the-art neural network PreCNet [SSH20] extended for tactile modality (see Section 4.1.2) was trained with default parameters. Using default parameters led to a strategy, that predicted the tactile stimulus too late, by only copying tactile activation from the previous time step to the output (see 'Previous Frame Strategy' in Section 4.3.4).

To improve this strategy to predict the tactile stimulus earlier, the network was extended with the ability to set weights to false-positive and false-negative errors in tactile predictions (see Section 3.2). Increasing weights of false-negative errors for not predicting a tactile stimulation (*tactile false negative* or 'fn' parameter described in Section 3.2), while preserving the *tactile false negative* parameter led, in general, to earlier predictions and activation of more tactile sensors. I'd like to add that penalizing the network for not predicting the stimulus might hypothetically correspond to receiving painful stimulus after an unpredicted collision of an animal or human with an object. As the collision is predicted by the agent, it might be possible to prevent it or minimize its consequences (see Section 2.2).

In Section 5.1, I presented several models trained on a dataset with constant speed across episodes. Modifying the *tactile false negative* parameter across these models led to three different tactile prediction strategies described in Section 4.3.4, from which two strategies, namely the 'Previous Frame Strategy' and 'Fixed Map Strategy' are “undesirable”, as they predict the tactile stimulus too late or activate too many redundant sensors. Importantly in the last 'Visuo-Tactile Strategy', the neural network predicted the tactile stimulation closely before it actually occurred, additionally activating more sensors, when the first actual activation was expected to occur. I propose, that implementation of such a strategy cannot be achieved by using only the information from a tactile modality, rather the prediction must be influenced by visual modality as well. Alternatively stated, the network accomplished improvement in predicting the tactile stimulation by visuo-tactile integration.

Admittedly, the tactile stimulus was not predicted in all sequences. Some errors in predictions might be caused by bias in the datasets. By way of example, less interactions occur by the beginning of the sequence than in the middle or by the end of it. Also, in each experiment, the ball misses iCub's torso in very few sequences. Another issue observed in Section 5.2.1. In Table 5.5, it was observed that further increasing the weight of tactile false-negative error did in fact decreased the false-negative error of sequences after the initial activation by implementing the 'Fixed Map Strategy'. Considering the initial activation lasts only one frame, whereas the later activation can last roughly 2-4 frames, it is indeed better strategy to apply this strategy to minimize the false-negative error in each time step. The issue is that if the network should predict the stimulus before it hits the body or in the time of actual activation, it should prominently be penalized for not predicting the stimulus in the time of collision. One solution to this problem would be to increase the penalty of false-negative error in the frame of the first actual tactile activation.

In Section 5.2, I presented models that were tested on datasets created from experiments, in which the velocity of the ball was changing across sequences. In these models, only two common strategies (described in Section 4.3.4) for tactile predictions were observed, leaving out the 'Visuo-Tactile Strategy'. Additionally, the ball in predicted images was blurry or completely missing. I believe this blurriness is connected with uncertainty about the new position of the ball, thus the network applies the default background image instead of predicting the ball position. This uncertainty also led to much bigger fixed maps of possible activations, within the networks that implemented the fixed map strategy.

Using higher camera frame rate, bigger ball, lower velocity or changing some parameters of the network could bring improvements, especially in the dataset with changing velocity. The network could be further extended by using images from both cameras instead of just one camera. In this way, the network could use the binocular disparity information to increase confidence in visual prediction as it was outlined in Section 4.1. It is also important to note that raising some of these parameters (e.g., adding a camera, adding more layers to the network, increasing the number of input sequences T) will significantly increase memory requirements for the network.

To my knowledge, created model is the first model that applies the predictive coding model for peripersonal space representation. The proposed model uses raw images and tactile data from 3D space, in comparison to other models described in Section 2.3, which inputs require further processing to obtain physical quantities (i.e., position, velocity, distance from the tactile sensor). The model further differs from these models [SH17] [RHP$^+$16] [MZS$^+$10] in setting different weights to false-positive and false-negative tactile errors.

Chapter 7

Conclusion

In this work, I created a framework for generating datasets with visual and tactile modality, using the Neurorobotics Platform [Pro] [FVA$^+$17]. With this framework, I created several experiments and parsed them into datasets, which basically consist of frames with 4 channels with an actual image in the first 3 channels and the tactile activations in the last channel. These datasets were used to train several PreCNet [SSH20] neural networks extended for tactile modality. This network was slightly modified to set weights of the errors caused by tactile activations that were not predicted and errors caused by predicting excessive number of tactile activation (see Section 3.2). Setting different error weights yielded different tactile prediction strategies. Using proposed quantitative evaluation and observing the tactile predictions of the network I propose that the network improved its tactile prediction strategy with visual information. Therefore, the network managed to implement multisensory integration which is an important part of peripersonal space representation.

Bibliography

[Ben14] Yoshua Bengio, *How auto-encoders could provide credit assignment in deep networks via target propagation*, arXiv preprint arXiv:1407.7906 (2014).

[BT71] William Ball and Edward Tronick, *Infant responses to impending collision: Optical and real*, Science **171** (1971), no. 3973, 818–820.

[CDG93] Carol L Colby, Jean-René Duhamel, and Michael E Goldberg, *Ventral intraparietal area of the macaque: anatomic location and visual response properties*, Journal of neurophysiology **69** (1993), no. 3, 902–914.

[CGN17] Roberto Calandra, J Geukes, and M Nakatenus, *icub gazebo skin*, `https://github.com/robertocalandra/icub-gazebo-skin`, 2017.

[CGWBH13] Justine Clery, Olivier Guipponi, C Wardak, and S Ben Hamed, *Multisensory integration of dynamic stimuli in the non-human primate: a functional magnetic resonance imaging (fmri) study*, Société française des Neurosciences. Poster **142** (2013).

[CGWH15] Justine Cléry, Olivier Guipponi, Claire Wardak, and Suliann Ben Hamed, *Neuronal bases of peripersonal and extrapersonal spaces, their plasticity and their dynamics: knowns and unknowns*, Neuropsychologia **70** (2015), 313–326.

[CP13] Rakesh Chalasani and Jose C Principe, *Deep predictive coding networks*, arXiv preprint arXiv:1301.3541 (2013).

[DCG98] Jean-René Duhamel, Carol L Colby, and Michael E Goldberg, *Ventral intraparietal area of the macaque: congruent visual and somatic response properties*, Journal of neurophysiology **79** (1998), no. 1, 126–136.

[dPLF97] Giuseppe di Pellegrino, Elisabetta Ladavas, and Alessandro Farné, *Seeing where your hands are*, Nature **388** (1997), no. 6644, 730–730.

[FGF+96] Leonardo Fogassi, Vittorio Gallese, Luciano Fadiga, Guiseppe Luppino, Massimo Matelli, and Giacomo Rizzolatti, *Coding of peripersonal space in inferior premotor cortex (area f4)*, Journal of neurophysiology **76** (1996), no. 1, 141–157.

[FVA+17] Egidio Falotico, Lorenzo Vannucci, Alessandro Ambrosano, Ugo Albanese, Stefan Ulbrich, Juan Camilo Vasquez Tieck, Georg Hinkel, Jacques Kaiser, Igor Peric, Oliver Denninger, et al., *Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform*, Frontiers in neurorobotics **11** (2017), 2.

[GC06] Michael SA Graziano and Dylan F Cooke, *Parieto-frontal interactions, personal space, and defensive behavior*, Neuropsychologia **44** (2006), no. 6, 845–859.

[Gib72] James Jerome Gibson, *The ecological approach to visual perception*, Lawrence Erlbaum Associates, 1972.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, *Vision meets robotics: The kitti dataset*, The International Journal of Robotics Research **32** (2013), no. 11, 1231–1237.

[HR11] Yanping Huang and Rajesh PN Rao, *Predictive coding*, Wiley Interdisciplinary Reviews: Cognitive Science **2** (2011), no. 5, 580–593.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural computation **9** (1997), no. 8, 1735–1780.

[KB14] Diederik P Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014).

[LKC16] William Lotter, Gabriel Kreiman, and David Cox, *Deep predictive coding networks for video prediction and unsupervised learning*, arXiv preprint arXiv:1605.08104 (2016).

[MCL15] Michael Mathieu, Camille Couprie, and Yann LeCun, *Deep multi-scale video prediction beyond mean square error*, arXiv preprint arXiv:1511.05440 (2015).

[MNN+10] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes Von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, et al., *The icub humanoid robot: An open-systems platform for research in cognitive development*, Neural Networks **23** (2010), no. 8-9, 1125–1134.

[MZS+10] Elisa Magosso, Melissa Zavaglia, Andrea Serino, Giuseppe Di Pellegrino, and Mauro Ursino, *Visuotactile representation of peripersonal space: a neural network study*, Neural computation **22** (2010), no. 1, 190–243.

[NBMS18] Jean-Paul Noel, Olaf Blanke, Elisa Magosso, and Andrea Serino, *Neural adaptation accounts for the dynamic resizing of peripersonal space: evidence from a psychophysical-computational approach*, Journal of neurophysiology **119** (2018), no. 6, 2307–2333.

[Pro] Human Brain Project, *Neurorobotics platform*, `https://neurorobotics.net/`.

[PS20a] Adrián Pitoňák and Zdeněk Straka, *Evaluated networks and prediction plots*, `https://drive.google.com/drive/folders/1yfLJAIVyyLrmSxsUPYdHpqa2Ccsmj7Iy?usp=sharing`, 2020.

[PS20b] ———, *icub peripersonal experiment gitlab repository*, `https://gitlab.fel.cvut.cz/body-schema/nrp-icub-pps-experiment`, 2020.

[PS20c] ———, *Multisensory precnet gitlab repository*, `https://gitlab.fel.cvut.cz/body-schema/precnet-pps`, 2020.

[RB99] Rajesh PN Rao and Dana H Ballard, *Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects*, Nature neuroscience **2** (1999), no. 1, 79–87.

[RFFG97] Giacomo Rizzolatti, Luciano Fadiga, Leonardo Fogassi, and Vittorio Gallese, *The space around us*, Science **277** (1997), no. 5323, 190–191.

[RHP+16] Alessandro Roncone, Matej Hoffmann, Ugo Pattacini, Luciano Fadiga, and Giorgio Metta, *Peripersonal space and margin of safety around the body: learning visuo-tactile associations in a humanoid robot with artificial skin*, PloS one **11** (2016), no. 10.

[RSMG81] Giacomo Rizzolatti, Cristiana Scandolara, Massimo Matelli, and Maurizio Gentilucci, *Afferent properties of periarcuate neurons in macaque monkeys. ii. visual responses*, Behavioural brain research **2** (1981), no. 2, 147–163.

[SH17] Zdenek Straka and Matej Hoffmann, *Learning a peripersonal space representation as a visuo-tactile prediction task*, 10 2017, pp. 101–109.

[SNG+15] Andrea Serino, Jean-Paul Noel, Giulia Galli, Elisa Canzoneri, Patrick Marmaroli, Hervé Lissek, and Olaf Blanke, *Body part-centered and full body-centered peripersonal space representations*, Scientific reports **5** (2015), 18603.

[SSH20] Zdenek Straka, Tomas Svoboda, and Matej Hoffmann, *Precnet: Next frame video prediction based on predictive coding*, arXiv preprint arXiv:2004.14878 (2020).

[SSR09] Barry E Stein, Terrence R Stanford, and Benjamin A Rowland, *The neural basis of multisensory integration in the midbrain: its organization and maturation*, Hearing research **258** (2009), no. 1-2, 4–15.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE transactions on image processing **13** (2004), no. 4, 600–612.

[WHS+18] Haiguang Wen, Kuan Han, Junxing Shi, Yizhen Zhang, Eugenio Culurciello, and Zhongming Liu, *Deep predictive coding network for object recognition*, arXiv preprint arXiv:1802.04762 (2018).

[XCW+15] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo, *Convolutional lstm network: A machine learning approach for precipitation nowcasting*, Advances in neural information processing systems, 2015, pp. 802–810.

[ZOS08] S Zylinski, Daniel Osorio, and A.J. Shohet, *Perception of edges and visual texture in the camouflage of the common cuttlefish, sepia officinalis*, Philosophical transactions of the Royal Society of London. Series B, Biological sciences **364** (2008), 439–48.

Appendix A

Contents of the enclosed CD

- iCub Peripersonal Experiment – code for generating experiment in the Neurorobotics Platform. This code is accessible online via [PS20b].
- Multisensory PreCNet – code for processing csv data from the Neurorobotics Platform up to the evaluation script. This code is accessible online via [PS20c].
- networks.xlsx – Evaluated PreCNet models. This file is accessible online via [PS20a]