Cross-module learning as a first step towards a cognitive system concept

Alexander Gepperth, Jannik Fritsch, Christian Goerick

2008

Preprint:

This is an accepted article published in Proceedings of the First International Conference on Cognitive Systems. The final authenticated version is available online at: https://doi.org/[DOI not available]

Cross-module learning as a first step towards a cognitive system concept

Alexander Gepperth, Jannik Fritsch and Christian Goerick
Honda Research Institute Europe GmbH
Carl-Legien-Str.30
63779 Offenbach, Germany
{alexander.gepperth,jannik.fritsch,christian.goerick}@honda-ri.de

***Abstract*— Despite great advances in information processing, computing power and conceptual understanding of biological information processing, the goal of creating truly "intelligent" or "cognitive" technical systems still remains elusive. It seems that the mere implementation of powerful but isolated functionalities does not add up to cognitive performance on the system level: we believe that the *principles* that guide the combination of multiple functionalities into systems merit attention as well. The focus of the contribution is on a technique we term "cross-module learning": the extensive learning of statistical interdependencies between system modules and their exploitation for robustness and system performance (assuming some degree of modularity in technical systems). We describe the concept in detail and derive some consequences for the design of technical cognitive systems. These include a common system-wide data format for information exchange, a dynamical system approach to data fusion, and a system-wide learning algorithm. As a proof-of-concept, we show experimentally that the straightforward integration of the principles of cross-module learning into a large real-world object detection system (operating in complex road traffic scenes) leads to significant increases in robustness and system performance.**

I. OVERVIEW

Creating and understanding truly cognitive systems is one of the great challenges of present-day research in a variety of fields. The difficulty of the issue is best illustrated by the ongoing controversy over its definition: what, precisely, *is* a cognitive system? We deliberately want to avoid controversy over a universally valid definition which probably cannot be given anyway and which is, to our mind, not required when attempting to construct "cognitive"systems. Instead, we take a pragmatic view of the issue and just state that cognition is a (complex) property which enables humans to achieve their very formidable everyday performance in a variety of tasks. Although we thus sacrifice generality on a philosophical level, we have a working definition that we consider sufficient to guide research efforts. On the same lines, although we avoid a definition of the term "cognitive" , we can make statements of the form "less cognitive" or "more cognitive". Put in this way, the goal of this contribution is to propose principles that can make present-day technical systems more cognitive than they currently are.

Our working definition of "cognition" puts emphasis on real-world performance. Cognitive systems should therefore be benchmarked by their performance in typical tasks that humans solve daily, often without really thinking about it. This requires, by common agreement, the combination of many individual skills and functionalities, which stresses the system-level or architectural aspects of cognition. Prototypical fields that are already being heavily investigated are, e.g., personal robots, household robots or intelligent driver assistance systems in road traffic. It is not the intent of this contribution to review the state of the art in these fields: rather, we will sketch what we perceive as limitations of present-day architectures on the way to true cognitive qualities. Based on that assessment, we will propose a set of concrete guiding principles that we believe will be beneficial in moving technical systems closer to cognition. The set is not exhaustive: we focus on a specific cognitive property here, namely the issue of system-wide cross-module learning and its consequences. Other properties may be just as important, since cognition does probably not arise from a single mechanism.

We will not, at this stage, attempt to formalize the proposed principles beyond a certain point, but rather choose to implement them first in the most straightforward fashion using a simple example scenario. In this way, we are able to establish the implementability and feasibility of our ideas in principle, before going on to formalize and elaborate them in detail. This prioritization seems important to us in order to increase the acceptance among researchers oriented towards *building* technical systems, who might require experimental support before considering to rework their existing systems. In order to provide that support, we will describe how a straightforward implementation of cross-module learning is employed for improving the performance and robustness of a standard object detection system operating in road traffic. The implementation details are given fully, and the dependency of the results on parameters is analyzed and discussed.

II. INTRODUCTION

A capability that humans excel in is the detection and exploitation of interdependencies between multiple sensory or internal states. This is reflected in the physiological structure of the human brain, which exhibits a very high degree of interconnectedness between cortical areas. Although enormously complex, the connection structure is not random but follows certain architectural principles which are (more or less) universally adhered to. For a review of this issue, please see [17]. However, the interconnections between cortical areas are not only very diverse but apparently highly

plastic; thus, new corresponding sensory or internal events are quickly associated with each other and stored.

In contrast, state-of-the art technical systems exhibit quite a low degree of interconnectedness and, correspondingly, cross-module learning capacity regardless of the application domain. Only recently, the issue of extensive system-wide information integration has been raised by several researchers [12], [3], [19] with sometimes impressive results in real-world tasks. However, the interconnections between different modules within the proposed systems are not plastic but preset and therefore incapable of acquiring new information.

Changing this state of affairs in a systematic fashion, towards higher and more meaningful interconnectedness, requires that literally every part or module in a system should be, in principle, able to obtain useful information from any other part in an adaptive way. One can extrapolate several requirements from this.

First of all, there must exist a data representation in which all modules exchange their results regardless of their content: the *common data format* (CDF). Using this format, a *system-wide learning algorithm* can detect statistical dependencies in the exchanged data. The *exploitation* of the learned dependencies is an important issue and can obviously be done in many ways. From these, we will outline just one in this contribution: *dynamic data fusion*, again using the CDF as a basis.

Several consequences of these requirements may be deduced: on the one hand, the way complex data are stored in a system is affected. Such data need to be encoded into the CDF; however, in order to be understandable throughout the whole system, a CDF also needs to have a simple structure. This is why complex data representations need to be distributed to many simple ones in order to be convertible into the CDF. On the other hand, the kind of learning that is envisioned here takes place, to a significant part, between system modules, internal system states or sensory data. It is therefore improbable to obtain enough training data for supervised training methods; rather, the system-wide learning algorithm must be able to operate in an unsupervised manner, although explicit feedback (human or otherwise) during system operation should be taken into account. Furthermore, the data that is analyzed for dependencies is generated *during system operation*, strongly suggesting the use of online learning algorithms.

In the following sections, we will discuss these issues in detail and describe an implementation proposal for a concrete technical system.

III. FORMALIZATION OF KEY PROPOSALS

In order to make the ideas proposed in the previous section more precise, we need to state the assumptions about the nature of the technical systems which we are concerned with.

Primarily, we assume a certain partitioning: we assume systems to be composed of several (possibly interacting) *processing modules* which can send and receive data and which are in a certain sense independent from each other. The issue of modularity is difficult to formalize (see, e.g., [6]), but for the purposes of this contribution it is sufficient to require that a module should be able to *receive data*, perform certain computations on the data, and *produce output* that is made available to the system.

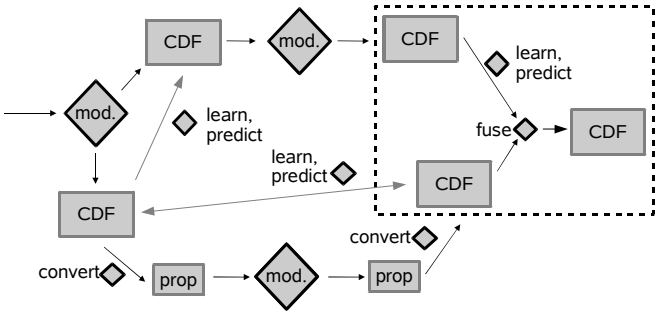

Fig. 1. Overview of proposed system properties: common data format (CDF), common learning algorithm, common fusion algorithm. Diamond-shaped boxes represent processing modules ("mod."), square boxes represent data. Small diamond-shaped boxes represent the implementation of the system properties proposed in the text ("learn,predict", "fuse"). The contents of the dashed box show how learned dependencies are exploited. It is not required that every processing module produces its output in the common data format. All that is needed is conversion ("convert") between a proprietary ("prop.") data representation and the CDF (see lower part of figure).

Furthermore, it is assumed that all relevant internal system states have a representation within the system that is accessible to the processing modules. This can imply that every internal system state is itself the output of some processing module.

Given these assumptions, the purpose of cross-module learning is then to achieve a kind of system where

- dependencies between internal system states (e.g., module outputs, sensory representations) can be detected by a system-wide learning process
- detected dependencies can be used to predict system states (generation of expectations)
- predictions can be used to influence other system states
- differences between predictions and predicted quantities (e.g., module outputs) can be used further by the system, i.e., to generate attention or learning signals.

These goals lead us to require several basic system properties, which are visualized in Fig. 1.

A. Neural map coding as a system-wide data exchange format

For detecting dependencies between internal system states by cross-module learning, the internal system states need to be in a format that can be processed by a learning algorithm. For a system-wide cross-module learning algorithm, this translates into the requirement of a system-wide common data format. However, it is not required that the learning algorithm can also *interpret* the data at its disposal, which is only necessary for modules that produce or receive the data, using appropriate assumptions. The only assumptions the learning algorithm needs to make consist of the nature of the data (distributions over numbers, explained later in this section) and their encoding into the CDF. This is sufficient

to detect dependencies, without needing to know what they actually *mean*.

We present a concrete proposal for a CDF which we term *neural map coding*. It is derived from the biological concept of *population coding*, sometimes also referred to as *space coding* [5], [22], [4]. A basic principle in neural map coding is inspired by data storage in mammalian cortical maps. Information is represented in two-dimensional *neural maps* of topologically organized elements which we term "neurons" here, although the more appropriate biological analogue are cortical columns.

The question of how information may be encoded and decoded into and from population codes has received considerable theoretical attention [22], [15]. Following previous proposals, we propose to encode probability distributions over single numerical quantities into the CDF for the experiments presented here.

An extension to multiple, ordered distributions over numbers (e.g., images, feature computation results) is straightforward, please see Fig. 2 for details. Reiterating previous work, the basic idea behind neural map coding is to assign a Gaussian "tuning curve" to each neuron which governs how strongly that particular neuron responds to a stimulus (the value to be encoded). We need to define the preferred stimulus of the neuron and the degree to which similar stimuli can still activate the neuron. Mathematically, this amounts to defining the mean and the variance parameters of the Gaussian function modeling the tuning curve.

A prerequisite for this kind of encoding is the topographical organization of neurons. This means that, at least in local neighborhoods, neurons in close proximity should respond to similar preferred stimuli. This must be ensured by an appropriate encoding method.

Fig. 2 gives an overview of the concept of neural map coding. Introducing some notation, we denote the activity of neurons contained in a two-dimensional neural map M by $u^M(\vec{x},t)$ or by $u^M_{ij}(t)$. A simple instantiation of this type of coding is given in section IV-C.

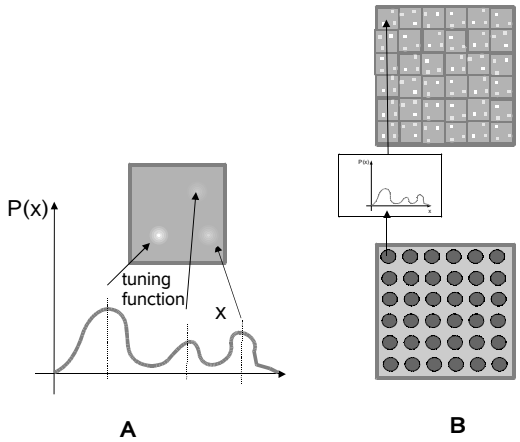

Fig. 2. Information encoding in neural maps. Figure **A** shows the basic principles involved: starting with a distribution over a real number, a two-dimensional neural map is constructed using predefined topology and tuning functions. Figure **B** visualizes how collections of distributions of real numbers may be encoded into neural maps. The point here is that, although neurons at different locations in the neural map can code for the same stimulus probability, a one-to-one relationship between the location of activation and stimulus probabilities exists locally (indicated by the rectangular partitioning of the neural map in Figure **B**).

B. A simple online learning algorithm

We propose to use an unsupervised online learning algorithm as the basis for cross-module learning. All that is required from the algorithm is that it can reliably detect simple dependencies between multiple internal system states. We expect that the benefits of exploiting such dependencies will by far exceed the performance loss to pay for the simplicity of the learning algorithm. Indeed, it is the goal of the experiments described in section IV-C to support this claim by evidence from real-world system operation.

Since the proposed learning algorithm it intended to detect dependencies between system states encoded by neural map coding, some kind of neural learning rule is required, which implies the existence of model synapses.

The transmission of information between two neural maps A and B, having discrete entries $u^A(\vec{x},t), u^B(\vec{y},t)$ at time t, using model synapses $w^{AB}_{\vec{x}\vec{y}}(t)$ is governed by the usual relationship:

$$u^B(\vec{y},t) = \sum_{\vec{x}} w^{AB}_{\vec{x}\vec{y}}(t) u^A(\vec{x},t) \quad (1)$$

An obvious learning rule which satisfies the requirements of online operation, unsupervised learning and simplicity is the Hebbian rule (see, e.g., [10]). In our notation, it reads

$$w^{AB}_{\vec{x}\vec{y}}(t+1) = w^{AB}_{\vec{x}\vec{y}}(t) + \epsilon u^A(\vec{x},t) u^B(\vec{y},t),\ \epsilon \ll 1. \quad (2)$$

In general, depending on the application, eqn. (2) is used in a slightly modified form in order to satisfy constraints, of which a very typical one is to prevent weights to grow without bounds [16]. Other constraints deal with the orthonormality of weight vectors projecting to different neurons, variance maximization [11] and similar issues.

In section IV-C, we will present a simple variant of eqn. (2) which fulfills the requirements for a cross-module learning algorithm that were outlined above.

C. A dynamical system model for decision making and data fusion

In order to use the generated predictions for influencing other system states, a flexible fusion method is required. We demand from the method that it should support the chosen CDF in a natural way. As in the case of the CDF, we argue that it is not required for the fusion algorithm to interpret the data to be fused. Any knowledge that may facilitate fusion should be acquired using the cross-module learning algorithm.

These requirements led us to choose two-dimensional neural fields evolving according to a variant of Amari dynamics [5], [21], [1], [18], [14], [9] for data fusion. Neural fields possess the advantage of an explicit time dependency, which potentially enables them to respond not only to the content but also to the time structure of incoming

data. Furthermore, their qualitative dynamic behavior can be substantially influenced by the variation of parameters. In this way, a switching between different ways of fusing data can be achieved, which provides considerable flexibility. The differential equation governing the dynamics of the neural field $u(\vec{x}, t)$ reads

$$\tau u(\vec{x}, t) = -u(\vec{x}, t) + \alpha I(\vec{x}, t) + \\ + \beta \int w(\vec{x} - \vec{x}\prime) f[u(\vec{x}\prime, t)] d\vec{x}\prime + u_{\text{rest}}, \quad (3)$$

where $\alpha, \beta, u_{\text{rest}}$ and τ are numerical constants, $w(\vec{x}) = G_{\text{on}}(\vec{x}) - G_{\text{off}}(\vec{x})$ is an interaction kernel given by the difference of two Gaussian functions having variances σ_{on} and σ_{off}, and $f[u]$ is a nondecreasing nonlinear transfer function bounded in the interval $[0, 1]$. In order to solve this equation numerically, it needs to be discretized in space. Since we use the neural field technique to determine the time evolution of neural maps, the discretization is chosen so as to coincide with the number of neurons in a neural map.

An additional point in favor of the neural field technique is that, when combined in an appropriate way with the learning algorithm described previously, it is very similar to models with self-organizing properties [2], [18].

IV. TESTING THE CONCEPTS: EXPERIMENTS IN A REAL-WORLD SCENARIO

We conducted our experiments based on a system (termed FIRST for "first integrated real scenario test") for multimodal real-time object detection and scene analysis in road traffic situations. Significant parts of the system are described in [7], [8]. The system runs on two standard notebook computers in a prototype car and is designed to initiate a braking manoevre whenever an object identified as a car comes too close. Basic implemented functionalities to solve this task include object detection by saliency maps [13], neural network-based object classification [20], visual object tracking and radar data fusion. Primary sensors are a high-resolution CCD color camera and a standard radar-based distance/relative speed sensor. The system was designed to operate in construction site scenarios but was also tested in inner-city areas with dense traffic, see Fig. 3. Indeed, the experiments described here were inspired by experiences during tests of that system.

Although we conducted all experiments based on FIRST, this was for convenience only: FIRST is treated as a black box providing data (measurements). Between these data, dependencies can be detected and exploited by the methods proposed in this contribution. Furthermore, the performance evaluation of the experiments described in section IV-C does not depend on FIRST but defines its own performance measures. If FIRST had been used to evaluate performance, the interpretation of the results would not be possible without detailed knowledge of FIRST.

A. Problem description

A formidable problem when doing real-world object classification concerns the classifier's ability to cope with unknown objects. Classifiers are usually trained offline using a finite, previously collected set of examples. When operating them in real-world scenarios which are not under the experimenter's control, it is therefore likely that unknown objects will be encountered. The problem is intensified by the fact that many objects cannot be properly characterized separately from the context in which they typically occur. Thus, a classifier that is trained offline, deciding purely on the basis of a single source of (visual) information, can be expected to run into difficulties. We believe that this will always be the case in real-world operation, regardless of the particular classification method (e.g., neural network, support vector machine, decision tree) that is used, even if care is taken to create a very large set of training data. What is more, the corresponding effort associated with ever smaller increases in performance will grow ever larger.

Fig. 3. Some performance examples of the FIRST system running on a prototype car. The left two images show operation in the construction site scenario it was envisioned for, the rightmost image shows operation in a inner-cite scenario in Offenbach, Germany. Boxes in the images represent object detections, where the object identities are shown above the boxes. Possible object classes are "car", "sigB" (signal board) and "?" (unknown object). Numbers below boxes show the distance to detected objects (if available) as measured by the radar sensor of the prototype car. In the leftmost image, the incorrect detection of a car is highlighted.

In FIRST, the issue manifested itself mainly in the spurious detection of cars, sometimes causing an emergency braking if the associated radar measurement was below a certain threshold. A type of error that occurred less frequently was the mistaking of cars for other objects, thus preventing a necessary emergency braking manoevre. The point here is that these false detection mainly occurred in unlikely places (at least to a human observer), i.e., above the road or in the sky. Please see Fig. 3 for a visualization.

In our opinion, the way to overcome the generic limitations of single-source classifiers, as well as the problems encountered in FIRST, is twofold: using online learning on the one hand, and exploiting more than one, possibly many, sources of information on the other hand. Using the concepts from section III, we claim that it is possible to realize both. For this purpose, the proposals of section III need to be concretized sufficiently to allow an implementation.

B. Implementation

What additional source of information could be exploited for object classification? A variety of possibilities comes to mind: object size, object position, the nature of the scene, the traffic situation, known ego-position, relative speed, general context cues (day/night, rainy/sunny, ..), just to name a few. In the experiments described here, we decided to go for very simple possibilities: the "retinal" position and size of detected objects. As far as position is concerned, it is intuitive that certain objects appear more frequently in certain

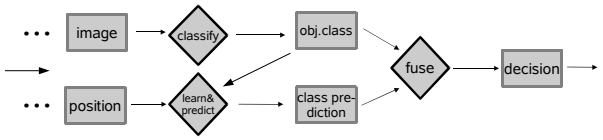

Fig. 4. Implementation of object classification using cross-module learning. Diamond-shaped boxes represent processing modules, rectangular boxes represent data. All data are encoded in the common data format, see text. The processing modules termed "learn&predict" and "fuse" implement the generic learning and fusion algorithms proposed in section III and may be used in many other parts of the system. The data fusion is here purely forward-driven, i.e., the classification module receives no feedback from a comparison of computed and predicted object class. The output of the data fusion process is a distribution over object classes. Based on this distribution, the final decision is taken.

parts of the camera image than others. For example, due to the scene geometry encountered in normal driving, cars are almost never detected in the upper half of the video image, and similar constraints hold for the retinal size of detected objects. This is a fact that can be exploited to eliminate spurious car detections. If the used classifier is able to return a distribution over classes instead of a decision, an incorrect interpretation of that distribution may still be corrected based on the prediction computed from the "hit statistics" in the position or size representations.

Based on the principles outlined in section III, we re-implemented the classification sub-system of FIRST as shown in Fig. 4. In the following text, it will be described how each proposal from section III is realized.

1) Encoding into the common data format: In order to adhere to the principles outlined in section III, it is necessary to encode all involved quantities in the common data format. Classifier outputs are converted into the CDF, as sketched in Fig. 1, in a straightforward way. The classifier produces a confidence $c_i \in [0,1], i = \{1,2,3\}$ for each object class (unknown objects, cars, signal boards). Since this is a simple learning scenario, we can afford to keep the CDF representation of the classifier outputs simple according to the philosophy outlined in section III-A. The resulting neural map should have one localized activity blob for each object class whose peak value indicates the confidence. Thus we can specify the necessary tuning functions G_σ (see section III) and the map activity resulting from the classifier results as

$$u(\vec{x},t) = \sum_i c_i G_\sigma(\vec{x} - \vec{x}_i)$$
$$G_\sigma(\vec{x}) \sim \exp -\frac{\vec{x}^2}{2\sigma^2} \tag{4}$$

with σ and $\vec{x}_i$ as given in Fig. 5. The encoding of retinal position and size into the CDF is rather trivial because the distributions obtained for these quantities are always strongly unimodal, which is also illustrated in Fig. 5. From retinal position, two representations are created: one encodes the full 2D position, the other one just its y-component. This was done in order to demonstrate that the choice of representation can strongly influence the success of learning. In total, we have three representations that are used to predict the classification output.

2) Learning: Since we want to keep things simple, we assume rate-coded neurons and synapses modeled by a single real number termed *weight*. We furthermore assume that the configuration of the weights $w^{AB}_{\vec{x}\vec{y}}$ between neurons at positions $\vec{x},\vec{y}$ in two neural maps A and B is always all-to-all. One of the simplest possible learning rules in this scenario is the Hebbian learning rule. We use a weight decay term in order to prevent the weights from growing to infinity (please see [16] for an overview of Hebbian learning rules and weight normalization techniques):

$$w^{AB}_{\vec{x}\vec{y}}(t+1) = \epsilon w^{AB}_{\vec{x}\vec{y}}(t) + \\ + (1-\epsilon)u^A(\vec{x},t)u^B(\vec{y},t), \quad \epsilon = \frac{t}{t+1}, \quad t \in \mathbb{N}^+. \tag{5}$$

Initially, all weights are set to small nonzero values between -0.01 and 0.01. From the weights that are learned during the course of system operations, predictions are generated according to eqn. (1).

3) The dynamic data fusion mechanism: For adapting to the chosen scenario, we modify eqn. (3) slightly, setting $\alpha I(\vec{x},t) \rightarrow \alpha I_M(\vec{x},t) + \nu I_P(\vec{x},t)$. The quantities I_M, I_P represent measurement (classification) and prediction inputs to the neural field. The relative magnitude of α and ν specifies the ability of the object class prediction to override the object class computed by the classifier. Except for ν which is systematically varied in the experiments, the parametrization of the Amari dynamics is fixed as follows:

$$\alpha = 1, \beta = 3, \tau = 15, u_{\text{rest}}(t=0) = -0.1$$
$$f(u) = \begin{cases} u & u \in [0,1] \\ 1 & u > 1 \\ 0 & \text{else} \end{cases} \tag{6}$$

We added a simple activity control mechanism to the dynamics: it concerns the behavior of the parameter u_{rest} which we extend to a time-dependent quantity $u_{\text{rest}}(t)$, which evolves

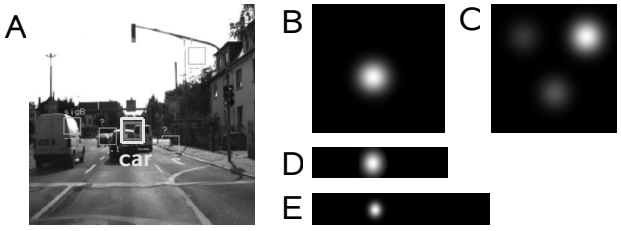

Fig. 5. Representation of object properties in neural maps. A car is detected (video image shown in **A**) and its properties are encoded into maps **B** (position), **C** (identity), **D** (y-component of position) and **E** (size). The 64x64 position map **B** is encoded by Gaussians with $\sigma = 5$. Center positions correspond to center positions of detected objects. The 64x64 identity map **C** is encoded by Gaussians with $\sigma = 5$. The center positions in the identity map are: $\vec{x}_0 = (30,41)^T$ (clutter/unknown), $\vec{x}_1 = (45,15)^T$ (cars) and $\vec{x}_2 = (15,15)^T$ (signal boards). It is taken care that the distances between center positions $\vec{x}_i$ are equal in order to prevent a bias in the competitive fusion process. The 64x20 y-position map **D** encodes the y position of detected objects, i.e., the distance in pixels from the lower image border using Gaussians of $\sigma = 5$. The 20x128 size map **E** encodes the square root of the area of detected objects using Gaussians of $\sigma = 2.5$.

according to

$$u_{\text{rest}}(t+1) = u_{\text{rest}}(t) - \eta(\mu_u(t) - \mu_u^*), \tag{7}$$

where μ_u is the average activity in map $u(\vec{x}, t)$, μ_u^* the target value for $\mu_u(t)$ and η a small constant (here we chose $\eta = 0.5$ and $\mu_u^* = 0.05$). This mechanism ensures that the map activity does not grow without bounds even if strong input is present. This is done because, in the case of too much activation, the dynamics can "blow up", i.e., converge to stable but unwanted attractor states with, e.g., globally constant activation which is undesirable for our purposes.

Each time that data fusion is performed, we iterate eqn. (3) for 50 cycles.

C. Experimental procedure

For performing the experiments, we chose a traffic video showing an extended drive through the inner city area of Offenbach am Main, Germany during the morning hours. The video comprises some 12000 frames, recorded at a rate of 10 Hz in RGB color at a resolution of 800x600 pixels. Using this video, we ran the FIRST system several times with the classification sub-system modified as described in section IV. At each run of the system, a different representation (see Fig. 5) is used for predicting the classification output.

Although we use an online learning algorithm, sufficiently many examples (data extracted from single frames) must be presented before the performance can be evaluated. We therefore distinguish three phases in the experiments:

1) Preliminary measurements: How can we benchmark, in a simple and straightforward fashion, the (potential) improvement in classification performance as a consequence using cross-module learning? Based on the problem of false detections described in section IV-A, we chose to test for the rejection of outliers, i.e., false detections that are inconsistent with previous experience. Such detections can be deliberately created, and furthermore chosen to be consistent with real ground-truth data. As an example, one could create "cars in the sky"; in this way, the performance of cross-module learning can even be roughly assessed by visual inspection.

For this purpose, we run FIRST for 5000 image frames recording all measurements concerning object size, position and identity. For object identities, a decision is computed from the obtained distributions by choosing the object class with maximal confidence. This reflects the decision FIRST would take, based on the "raw" classifier outputs not improved by predictions. We now identify, by visual inspection, certain parameter ranges of object position and size; for these ranges, we demand that FIRST should (almost) never detect cars. In the subsequent experiments, we can therefore, require that *any* car detection where object position or size fall into those ranges has to be rejected as inconsistent.

The preliminary measurements are performed on videos that are disjunct from the videos used for performance evaluation. In this simple way, we can replace ground-truth data necessary for performance evaluation, which we currently do not possess.

2) Learning phase: In all experiments, the system is allowed to run (and learn) for 1000 image frames before conducting performance evaluations.

3) Evaluation phase: In the evaluation phase (which is conducted for 4000 frames after the learning phase), we replace the classification and object position/size results from FIRST (see Fig. 5) by artificially generated detections measurements. Artificial object identity maps express uncertain decisions for "car" objects: the confidences encoded into the CDF according to section IV-B.1 are 1.2 (cars), 0.8 (signal boards) and 1.0 (clutter). Artificial object positions are located in a window W consisting of the upper 40% of the video image, which is supported by experience and the preliminary measurements. Similarly, artificial size measurements are chosen from a range of 105-180 pixels. Positions and sizes of artificial detections are drawn from a uniform probability distribution. One artificial detection per image is generated. To prevent the learning mechanism to adapt to the artificial stimuli, the learning rate is set to 0.0 in the evaluation phase.

After the fusion of prediction and artificially generated measurements, the maximum activation in the fused map is used to determine the resulting object class decision. Due to the reasoning given in section IV-C.1, *any* report of a car detection can now be considered to be incorrect. In order to assess the performance of the system, the quantity $\chi \in [0,1] \equiv 1 - \frac{\#(\text{detections})}{N}$ is calculated. χ encodes the suppression rate of spurious detections, where a value of 1.0 indicates perfect suppression. The main parameter governing χ is the relative magnitude of prediction and measurement in the fusion process. If a good suppression can be achieved even for small contributions from prediction (governed by the parameter ν, see section IV-B.3), then it is ensured that only uncertain decisions are overridden by prediction. If a large value of ν is required even for the uncertain decision that is artificially created here, then the prediction may also incorrectly override decisions that are certain and correct. This reasoning about the choice of ν could best be supported by experiments using ground-truth data for evaluation.

V. RESULTS

The results of the experiment described in chapter IV are shown in Fig. 6. Surprisingly, the representation that encodes the y component of an object's position allows the best prediction of classification results, which can be seen from the fact that for all values of ν, the suppression of incorrect detections is highest. By the reasoning from the previous section, especially small values of ν are important in this respect. In contrast, the size representation is the least feasible representation for predicting classification results, with the 2D position representation lying in-between. How can it be understood that the full two-dimensional representation of object position has weaker predictive power than the representation containing just the y component? From driving experience, we know that it is the height above ground level, here encoded by the y component of an object's (retinal) position, that is correlated with the presence of

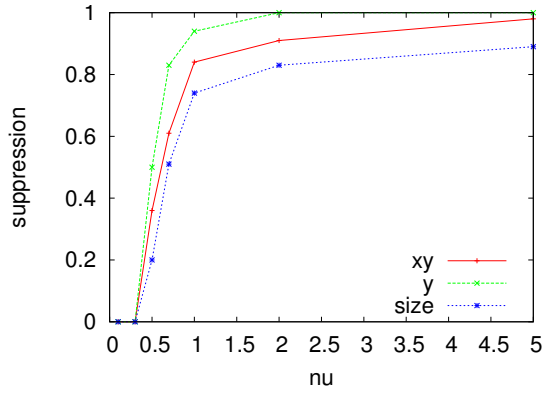

Fig. 6. Suppression of incorrect car detections using the FIRST system with the classification sub-system extended by a prediction and fusion mechanism as described in the text. Three different representations are used for predicting classification outputs: 2D object position (labeled "xy"), y-component of object position ("y") and object size ("size"). The suppression rate of incorrect detections χ mainly depends on the influence of the prediction result on the fusion mechanism, governed by the parameter ν. Surprisingly, the best predictor for the classification output is the y-component of object position.

cars. A representation just containing that relevant quantity should require fewer steps for learning dependencies than a representation where the relevant quantity is embedded into irrelevant data (like the representation of the full 2D position). This argument can be generalized further: by choosing suitably adapted representations, we expect that the effort for learning predictions can always be strongly reduced. Therefore, it is advisable to choose representations with care for any given cognitive task.

VI. DISCUSSION

How reliable are the presented results and what are the assumptions under which we expect them to generalize? We showed that the prediction of the classification output can be used to modify incorrect classifications in cases where the classifier results are uncertain (i.e., multimodal). This assumes that the classifier is able to encode the certainty of its decisions. Furthermore, the described car classification scenario implicitly assumes that the classifier results are more reliable than the prediction results (otherwise the classifier would be unnecessary). This implies that unimodal classifications, whether correct or incorrect, should never be modified. It was shown in this paper that incorrect detections of cars can be efficiently suppressed even when the prediction contributes only weakly (small ν), which validates our assumptions. What remains to be demonstrated is that correct detections are not corrupted in the process. By the argument presented above, this should not occur for certain and correct decisions in any case; the behavior for correct but uncertain decisions will be the subject of further investigations, please see section VII.

In order to provide an unbiased picture, we will now discuss possible criticisms of our work. What may be criticized? First of all, the encoding into the common data format lacks generality in the presented form. We argue that, although this is certainly true, the chosen encoding method is general enough to work within certain conditions, which are met by the chosen application scenarios and, in our opinion, in most other scenarios as well. We will discuss these conditions in the next section. Furthermore, the use of the fusion dynamics provides advantages (time dependency, flexibility) which are not required in the presented experiments. Thus, a simple addition of classification and prediction, followed by a maximum selection, might have led to the same results. Effectively, however, this is what the neural fields used for fusion do. Summation is implicit in the field dynamics, and maximum selection is achieved by lateral competition between activity blobs. We feel that the use of the presented fusion method is thus well motivated, although it is true that the *real* advantages of this method, as explained in section IV-C, will only manifest themselves in further work on this subject.

VII. FUTURE WORK

We have sketched out steps towards a cognitive system concept; since this is a formidable task, we could obviously not address all relevant issues. We have, however, a clear idea about the issues that need to be tackled in the near future.

As mentioned in the previous section, the encoding into the common data format needs to be specified in more detail in order to be generally applicable. This includes the questions of expressing (un)certainty or multiplicity, i.e., the simultaneous presence of several concepts. Especially the latter issue is challenging, since a single distribution can, by definition, just make statements about a single quantity. This is why only one car detection per frame was represented in the experiments of section IV-C; otherwise, the encoding would have become more challenging.

In addition, it needs to be determined how to autonomously choose optimal parameters for the fusion algorithm. The most obvious parameter in this respect is the relative weight of prediction and predicted quantity, ν. In the presented experiments, an optimal value was found by trial and error. A more appropriate way would be to scale each neural network weight according to an intrinsic quality measure such as the variance of predictions. In this way, varying reliabilities within a single cue (e.g., position) could be encoded and exploited autonomously.

For the construction of large systems with feedback, additional local mechanisms of keeping all activations and weight strengths finite need to be investigated.

Some extensions offer themselves at a more global level: for instance, the detection of dependencies is restricted to pairs of representations in the present formulation. This cannot easily be changed; it is more feasible to investigate how two or more representations may be *combined* or *integrated* into a single one. In this way, the learning mechanism could be extended to (effectively) more-than-pairwise dependencies. There are several candidate mechanisms for integrating representations, most notably self-organizing models or principal component algorithms. Another simple extension of the presented work is to use the error signal resulting from predictions. In the case of an object classifier, the difference

between the predicted and the actual output distribution (see section IV-C) can be fed back to the classifier in order to adapt its internal models appropriately. Thus, the need for ground-truth data could be strongly reduced. Feedback may come from any source, not only from prediction errors. Therefore an element of supervision can still be supplied.

As a last point, we hypothesize that a classifier, on its own, might not need to perform extraordinarily at all. Rather, robust performance may come about by the adaptive fusion of many sources of information using the ideas described in this contribution. It will be intriguing to test this hypothesis in FIRST, using very simple classifiers (maybe "rectangle detectors") and providing extensive and possibly complex additional information. Additional information may consist of sensor features, but also derived quantities like a scene classification or lane borders.

VIII. CONCLUSION

In this contribution, we showed that the performance of a state-of-the art classifier can be improved strongly even by a very simple method. The described integration of additional information requires little overhead since the information is already available to the system, albeit not used. This suggests to us that there may be many such isolated functionalities in technical systems (especially when performing real-world tasks) that can benefit from additional information. The key requirement is that the involved processing modules do not generate decisions but distributions. This contribution supports the exchange of distributions, and thereby the system-wide integration of information, by defining a "standard format" for distributions: the common data format.

Furthermore, we showed experimentally that not only the information that is encoded in a representation matters, but also the *way* it is encoded: not all representations are equally favorable for learning. We expect that truly task-adapted representations will strongly support the learning methods proposed here.

On a more holistic level, we proposed a number of simple principles that we believe will lead to increased cognitive capabilities in technical systems. In our opinion, the proposed ideas are not the end of the story but rather the beginning. They need to be embedded into a *cognitive system concept*, a comprehensive set of guidelines how to construct technical cognitive systems (where the term "cognitive" is used as defined in the introduction). It is our vision both to formulate such a concept, on the one hand, and to construct technical systems with cognitive abilities on the other hand. We may draw inspirations from biological information processing in this process, and we intend to do so heavily while keeping a balance between biological accuracy and the functional abstraction that is necessary in any technical realization. In the work described here, we focused mainly on local issues, i.e., properties that may be attributed to a limited region within a system. However, we propose that a key ingredient in a prospective cognitive system concept should address global system properties, i.e., an appropriate *system architecture*. A prominent (but not the only) architectural issue is, for example, the correct way of organizing hierarchical information processing, particularly when sensory information is concerned.

We believe that the principles proposed in this contribution form a first step towards a cognitive system concept. They should therefore continue to apply when defining aspects of system architecture, which is, to our mind, the next logical step to take. Especially when considering architectural issues, it seems appropriate to take inspirations from research on the biological principles of human or primate cognition.

REFERENCES

[1] S.-I. Amari. Mathematical foundations of neurocomputing. *Proceedings of the IEEE*, 78(9):1441–1463, 1990.

[2] J. Bednar, A. Kelkar, and R. Miikkulainen. Modeling large cortical networks with growing self-organizing maps. *Neurocomputing*, 44-46:315–321, 2002.

[3] B. Bolder, M. Dunn, M. Gienger, H. Janssen, H. Sugiura, and C. Goerick. Visually guided whole body interaction. In *International Conference on Robotics and Automation*. IEEE, 2007.

[4] S. Deneve, P. E. Latham, and A. Pouget. Efficient computation and cue integration with noisy population codes. *Nat Neurosci*, 4(8), 2001.

[5] W. Erlhagen and G. Schöner. Dynamic field theory of movement preparation. *Psychol Rev*, 109(3):545–572, Jul 2002.

[6] J. Fodor. *The modularity of mind*. MIT Press, 1983.

[7] J. Fritsch, T. Michalke, A. Gepperth, S. Bone, F. Waibel, M. Kleinehagenbrock, J. Gayko, and C. Goerick. Towards a human-like vision system for driver assistance. In B. De Schutter, editor, *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2008. to appear.

[8] A. Gepperth, B. Mersch, J. Fritsch, and C. Goerick. Color object recognition in real-world scenes. In J. de Sa, editor, *ICANN 2007, part II*, number 4669 in Lecture Notes in Computer Science. Springer Verlag Berlin Heidelberg New York, 2007.

[9] U. Handmann, I. Leefken, and W. von Seelen. Scene interpretation and behavior planning for driver assistance. In *Enhanced and Synthetic Vision 2000 at AEROSENSE 2000*, volume 4023 of *SPIE Proceedings Series*, pages 201–212, 2000.

[10] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall, 1999.

[11] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent component analysis*. Number 001 in Adaptive and Learning Systems for Signal Processing, Communications, and Control Series. Wiley-VCH, 2001.

[12] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 17-22 June 2007.

[13] T. Michalke, A. Gepperth, M. Schneider, J. Fritsch, and C. Goerick. Towards a human-like vision system for resource-constrained intelligent cars. In *The 5th Int. Conf. on Computer Vision Systems Conference*. Universitätsbibliothek Bielefeld, 2007.

[14] I. Mikhailova and C. Goerick. Conditions of activity bubble uniqueness in dynamic neural fields. *Biol Cybern*, 92(2):82–91, Feb 2005.

[15] A. Pouget, K. Zhang, S. Deneve, and P. E. Latham. Statistically efficient estimation using population coding. *Neural Comput*, 10(2):373–401, Feb 1998.

[16] T. Sejnowski and K. Obermayer. *Self-Organizing Map Formation: Foundations of Neural Computation*. 2001.

[17] B. Sendhoff, U. Körner, and E. Körner. On the integration of biological constraints into the evolution of artificial neural systems. In S.-Y. Lee, editor, *Seventh International Conference on Neural Information Processing – Proceedings*, pages 903–908, Taejon, Korea, 2000.

[18] J. Taylor. Neural 'bubble' dynamics in two dimensions: foundations. *Biological Cybernetics*, 80:393–409, 1999.

[19] T. Wennekers, M. Garagnani, and F. Pulvermller. Language models based on hebbian cell assemblies. *J Physiol Paris*, 100(1-3), 2006.

[20] H. Wersing and E. Körner. Learning optimized features for hierarchical models of invariant object recognition. *Neural Comp.*, 15(7), 2003.

[21] C. Wilimzig, S. Schneider, and G. Schöner. The time course of saccadic decision making: dynamic field theory. *Neural Networks*, 19(8):1059–1074, Oct 2006.

[22] R. S. Zemel, P. Dayan, and A. Pouget. Probabilistic interpretation of population codes. *Neural Comput*, 10(2):403–430, Feb 1998.