Adaptive Anomalies Detection with Deep Network

Chao Wu, Yike Guo
Department of Computing
Imperial College London, London, UK, SW7 2AZ
Email: {chao.wu, y.guo}@imperial.ac.uk

Yajie Ma
College of Information Science and Engineering
Wuhan University of Science and Technology, Wuhan, China, 430081
Email: mayajie@wust.edu.cn

Abstract—**In this paper, we try to apply inspirations from human cognition to design a more intelligent sensing and modeling system, which can adaptively detect anomalies. The target of intelligent sensing and modeling is not to get as much data as possible, or to build the most accurate model, but to establish an adaptive representation of sensing target and achieve balance between sensing performance requirement and system resource consumption. To achieve this goal, we adopt a working memory mechanism to facilitate the model to evolve with the target. We use a deep network with autoencoders as model representation, which is capable to model complex data with its nonlinear and hierarchical architecture. Since we typically only have partial observations from sensed target, we design a variance of autoencoder which can reconstruct corrupted input. We utilize attentional surprise mechanism to control model update. Training of the deep network is driven by surprises (which are also anomalies) detected (with data in working memory), which means model failure or target's new behavior. Due to partial observations, we are not able to minimize free-energy in a single round, but iteratively minimize it by keeping finding new optimization bounds. While both random and non-random sensor selection can create new optimization bounds, certain non-random methods like surprise minimization algorithm used in this paper demonstrate better performance. For evaluation, we conducted experiments on simulated data to test whether our methodology makes the model more adaptive, and got positive result. In the next step, we will try to apply the work on some real applications including ECG and EEG anomaly detection.**

***Keywords*–Cognitive sensing; deep learning; anomaly detection.**

I. Introduction

The world is always dynamic, unpredictable, ambiguous, and noisy. Such uncertainty is the only reason why we need to be equipped with intelligence. From a cognitive point of view, the intelligence of any intelligent agent (like animals), is a kind of ability to achieve equilibrium with its uncertain environment. It can sense or act (to fit with or intervene its environment), to minimize its free energy [1]. Such intelligence is under some constraints. Facing dynamic and high-dimensional world, even for the most complex systems as human brains, neural computation resource is limited [4]. Also our action capabilities on the environment (like motion capability) is limited.

To handle the uncertain world with constrained resources, intelligent agent developed several crucial cognitive mechanisms: We use attention [6] and surprise [5] to select information deserved to be processed and allocate neural resource for feature binding; we have long-term / short-term working memory to organize the knowledge hierarchically; Our brain is constituted with a large amount of deep networks, each of which act as a universal model and support ”one learning algorithm”. Such deep networks are considered to be very useful to organize and process our knowledge. Whether these mechanisms, or at least some of them, could be adopted in sensing system to make it more intelligent? This idea motivated the work in this paper: we tried to get inspirations from biological intelligence, and design an adaptive computational framework for sensing and modeling a dynamic target, under system resource constrain. The emphasis here is to detect target's anomalies, or surprises, which indicates there appear some events or new behaviors of the target. The examples of anomalies include traffic accidents (*events*) on a road (*sensing target*), and seizures (*events*) in human brain (*sensing target*).

In section 2, some related works are discussed. Detailed methodology of our framework is given in Section 3. Section 4 provides both simulated evaluation and demonstration of an application. In Section 4, we conclude the paper with a future research plan.

II. Related Works

Historically, designing computing system by learning human cognition is not new. Cognitive science discoveries had inspired a lot of researches on unsupervised learning (e.g., the Analysis by Synthesis approach [7]). Another example is the formation of Infomax principle and independent component analysis [10], [11] from the inspiration of efficient coding [8], [9]. However, it's until recent years that the advance of cognitive science and neuroscience makes it possible to build a clearer picture of how our brain works. Based these findings, we believe the meeting with Brain Informatics (BI) [12], [13] and AI will be the next drive for both developments.

Cognitive perspective also inspired new insight of anomaly. We use an unsupervised method to train our model, and this model can reconstruct the input (with low loss function output) when the input is seen before (within the range of model's representation capability). Only once the observation is new to the model and cannot be reconstructed, we detect an anomaly. It is slightly different from ordinary anomaly detection, which excludes the outliers into the model, but related to novelty detection [16], which try to detect emergent and novel patterns in the data, and incorporated into the normal model after being detected. This anomaly is also seen as an attentional surprise (more formally, 'surprisal'), induced by a mismatch between the sensory signals encountered and those predicted. Such surprise play the similar role of surprise for human, which acts as a kind of proxy for sensory information [14] (to select input for process). Similar principle has been applied in methods like predictive coding [15]

III. Methods

A. Notations and problem definition

Target: We represent the target that we want to understand as $x = (x_1, x_2, ..., x_i, ...x_n) \in R^n$ with dimension n, which is the size of spatial-temporal resolution of the target (e.g. $n = 24$, if the target is the hourly temperature in one day). x can have infinite dimensions and $n = \infty$. x is a time series : at time t, we have $x^t = (x_1, x_2, ..., x_i, ...x_n)^t$, and we denote the collection of x as $X = \{x^1, x^2, ..., x^m\}$.

Observation: In most cases, we can not observe x' directly, but only observe partial x (in lower dimension k) with noise ϵ. We define this noisy and partial observations from sensors as $X' = \{x'^1, x'^2, ..., x'^m\}$, where $\|x'\|_0 = k$ and $k \leq n$. We assume $x' = s(x)_k + \epsilon$, where function $s(x)_k$ selects k elements in x, keeps their values, and sets the other elements to 0.

Model: Model y is established based on observations. It can be viewed as a representation of x, governed by some parameters θ, just like the internal model of human can be viewed as some higher representation of its input. y can be a distribution, a learned dictionary, or a stacked autoencoder as used in this paper.

Problem definition: For a sensing system, the goal is to learn $p(x)$, which is difficult when $p(x)$ is changing. So we try to approximate $p(x)$ with the mapping from y to x (with observations x'): $q(x|y; x')$, or simply $q(x|y)$. For this approximation, there are mainly two challenges: 1) mapping from y to x should not only approximate $p(x)$, but be able to adapt to the change of $p(x)$; 2) with only partial observations, we want them to be informative, so how to design function $s(x)_k$ is then crucial, and thus becomes well-known sensor selection problem. With this adaptive mapping from y to x, we then evaluate the new observations with the established model to check whether model fails. If so, we say there is **anomaly**.

B. Cognitive approach

According to free energy principle [1], any intelligent agent will sense or act to minimize its free energy $F(x, y)$, which is the KL divergence between its internal approximation with model and real environment (i.e. sensing target, in this context):

$$F(x, y) = D_{KL}(q(x|y; x')\|p(x)) \tag{1}$$

As a result, we minimize the KL divergence between these two:

$$D_{KL}(q(x|y)\|p(x)) = \int q(x|y) \ln \frac{q(x|y)}{p(x)} dx \tag{2}$$

$$= D_{KL}(q(x|y)\|p(x|y)) - \ln p(y) \tag{3}$$

Here we get two components: 1) the first component is the KL divergence between $q(x|y)$ and $p(x|y)$. $q(x|y)$ represents the approximated distribution of x given y as the internal model; and $p(x|y)$ represents the likelihood of x if it's governed by y. We can unbiasly estimate this component by replacing x with x', then $p(x'|y)$ becomes the empirical likelihood. In this paper, we define this component (or its approximation) as **surprise**. Once we have great surprise, it means the model is too rough (not well-trained), or the target exhibits some new behavior that has not been captured by the current model. In both situations, the model cannot capture the target, and needs to be updated. Cognition model will update y and thus reduce the KL divergence between $q(x|y)$ and $p(x|y)$. However, because we only use partial observation, we actually minimize a part of original KL divergence:

$$D_{KL}(q(x|y)\|p(x|y)) \leq D_{KL}(q(x'|y)\|p(x'|y)) \tag{4}$$

After we minimizing this KL divergence with x', a very important next step is to find a new bound of KL divergence optimization, by selecting new x' through random or non-random sensor selection, and then update the model in the next iteration; 2) the second component is the negative log of $p(y)$. It measures how unlikely a representation y will happened, and will be large if the sensing target is too dynamic or noisy. The optimization of this component is out of the scope of this paper.

We can also interpret this cognitive approach with Infomax principle. Best mapping from y to x maximizes the mutual information between x and y, which can be represented as the difference between entropy of x ($H(x)$) and conditional entropy of x given y ($H(x|y)$). Following the same approach in [2], we assume x comes from an unknown distribution $p(x)$ on which θ has no influence, so $H(x)$ is constant. Therefore, the target is then to maximize $-H(x|y)$, which by definition is:

$$\underset{y}{\operatorname{argmax}} E_{p(x,y)}[\log p(x|y)] \tag{5}$$

With approximation $q(x|y)$, we have:

$$E_{p(x,y)}[\log q(x|y)] \leq E_{p(x,y)}[\log p(x|y)] \tag{6}$$

which is a lower bound of $-H(x|y)$. Assume we transform x to y with a deterministic or stochastic mapping $y = f_\theta(x)$(encoding), and reconstruct x from y by $x = g_{\theta'}(y)$ (decoding); and use empirical average over the observations as an unbiased estimation. We end up maximizing the mutual information in the following form:

$$\underset{\theta,\theta'}{\operatorname{argmax}} E_{p(x)}[\log q(x|y = f_\theta(x); \theta')] \tag{7}$$

This corresponds to the reconstruction error criterion for autoencoders.

Before describing the details in our methodology, let's give an overall workflow of the methodology. We use some sensor selection algorithm $s(x)_k$ upon target x to get observation x', and use a working memory to store the data required for model training and update. Model training component trains and updates y (its parameter θ) with data in memory. The established model is tested with new observation, to detect anomaly. Anomaly triggers actions including memory update and model update, and thus causes computational cost. Such model update optimize the free-energy within current bound, and sensor selection algorithm keeps trying to find new bound of free-energy for further optimization.

C. Model representation

We utilize a variation of stacked autoencoders [3] as the model representation. An autoencoder neural network is (unsupervisedly) trained with back propagation, setting the output values to be equal to the inputs. The result network takes an input x and transforms it to a hidden representation $y \in R^d$ through a deterministic function (with sigmoid activation function s):

$$y = f_\theta(x) = s(Wx + b) \tag{8}$$

It is parameterized by $\theta = \{W, b\}$. W is a $d \times n$ weight matrix. b is the bias vector. The resulting y is then mapped back to a reconstructed vector $z \in R^n$ in input space

$$z = g_{\theta'}(y) = g_{\theta'}(f_\theta(x^i)) = s(W'y + b') \tag{9}$$

with $\theta' = \{W', b'\}$. The weight matrix W' of the reverse mapping is constrained by $W' = W^T$. We get the optimized parameters θ^* and θ'^* by minimizing the reconstruction error (with loss function L) between x^i and z^i:

$$\theta^*, \theta'^* = \underset{\theta,\theta'}{\operatorname{argmin}} E_{p(x)}[L(x, z)] \tag{10}$$

Since:

$$L(x, z) \propto -\log p(x|z) \tag{11}$$

and thus

$$L(x, z) \propto -\log q(x|z) \tag{12}$$

We have:

$$\theta^*, \theta'^* = \underset{\theta,\theta'}{\operatorname{argmax}} \log q(x|z = g_{\theta'}(f_\theta(x))) \tag{13}$$

$$= \underset{\theta,\theta'}{\operatorname{argmax}} \log q(x|y = f_\theta(x), \theta') \tag{14}$$

which is the same form of optimization described before. With m training set, we will try to optimize:

$$= \underset{\theta,\theta'}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} L(x^i, g_{\theta'}(f_\theta(x^i))) \tag{15}$$

By placing constraints on the network, we can discover structure about the data and learn useful representation. Instead of limiting the size of hidden layer ($\|y\|_0$), we allow the size to be large, but impose sparsity constraints. An extra penalty term is added to optimization objective. let $(\hat{\rho})_j$ be the average activation of hidden unit j, averaged over the m training examples:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} [a_j(x^i)] \tag{16}$$

where a_j denotes the activation of this hidden unit when the network is given a specific input x. And let ρ to be a sparsity parameter, typically a small value close to zero (e.g. 0.05). So the penalty term is KL divergence between two (s_l is size of l-th hidden layer):

$$\sum_{j=1}^{s_l} KL(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{17}$$

Such autoencoder is used as a building block to train deep networks, with the learned representation of the k-th layer used as input for the $(k+1)$-th to learn a second level representation. Therefore, we train the $(k + 1)$-th layer after the k-th has been trained. These layers are ”stacked” in a greedy layer-wise approach as deep RBMs. This greedy layer-wise procedure has been shown to yield significantly better performance than random initialization.

D. Memory

Inspired by human short-term working memory, we enable a simple memory for the data modeling: $M = \{x^{*1}, x^{*2}, ..., x^{*k}\}$. k is the size of memory M (k is fixed currently, but can vary for specific system requirement). x^{*k} is the observation. Same historical observation x might have multiple copies in memory. They are sorted so that for any $x^{*i} = x^p$ and $x^{*j} = x^q$, if $i > j$, then $p \geq q$. So in the front of the memory, we have the oldest observations, while in the end of the memory, we have most recent observations.

The model described before will be trained or updated only with the data within this memory. We design a memory update strategy, so that the model values those new observations more than those old observations, and even forget those old observations. When new observation arrives, we apply a forget (or decay) function $M = \Phi(x^*)$ to pick out old data and replace them with new observation, we can use some naive approach for the forget function (e.g. randomly picking old observations), or a probabilistic function, so the older the data, the greater chance that it would be removed: for data at index i in M, we define its probability of being forgotten as: $\psi(i) = e^{-\delta i}$. For all i, when $\psi(i) \geq \eta$, we replace them with new observation x^*:

$$\Phi(x^*) : M(i) = x^* \tag{18}$$

The decay rate δ as well as the threshold η control the speed for memory update. If δ is large or η is small, it means the old memory would be removed quick, and the model changes fast. Such memory is crucial if the target is dynamic. And for different model training methods, memory can play different roles. For point estimation method like we used for stacked autoencoder, the memory provides the data for model training and retraining. For Bayesian estimation like Gaussian process regression , the memory provides the data for calculate the likelihood, and thus influence the model update.

E. Surprise and anomaly detection

Attentional surprise [5] can only be defined in a relative, subjective, manner and is related to the expectations of the observer, even when derived from identical observation, same data may carry different amounts of surprise at different times. So it could be seen as the subjective measurement of information (we might call it ”subjective entropy”). In this paper, surprise is defined as the KL divergence between $q(x|y)$ and $p(x|y)$. Such surprise is the trigger for model update (or retraining) as well as memory update. If surprise is larger than surprise threshold $surprise(y, x) > \xi$, it means the current model fails, and we need to update the model; also we need update the memory to include this new significant observation. In model representations, such as the stacked autoencoder used in this paper, above calculation of surprise cannot be conducted straightforwardly. Here, the surprise can be simply set as the loss value between reconstructed output and real input:

$$surprise(x) = L(x, g_{\theta'}(f_\theta(x))) \tag{19}$$

When this loss function (root mean-squared-error) output is greater than the surprise threshold ξ, it means the autoencoders cannot represent the input well at the moment. So it's necessary to update the model to fit the current input, with the data from updated memory.

This attentional surprise also enables a new method of anomaly detection. It simulates the human attention mechanism to some extent, acts as an information-processing bottleneck that allows only a small part of incoming sensory information to reach working memory and trigger model update, instead of attempting to fully process the massive sensory input. Human can maintain a certain level of alertness (e.g. when we are driving in an unknown district, we would like to pay more attention and allocation more computational resource): when it's high, more resources is prepared for attention, and even subtle signs could be detected. Similar idea is adopted here. We can find several parameters (and hyper-parameters) that provide us the chance to control surprise and model update, including: 1) Surprise threshold: model update frequency; 2) Memory size : the adaptive level of model; 3) Sensor selection parameters (pre-defines the region of sensing and change the frequency of surprise, as shown later). These top-down settings control the overall alertness of the sensing system, as well as the resource consumption. Top-down settings can be changed according to different system objectives and tasks.

F. Surprise minimization sensor selection and denoising SAE

When we only observe partial input x', the KL divergence (or surprise) is smaller than (or equal to) KL divergence with actual x. In other words, because of using partial observation, there is some space between D'_{KL} and D_{KL}. When we update the model and minimize surprise, we actually partially optimize it. Therefore, to minimize the actual KL divergence, it's necessary for sensor selection schema to keep selecting new x' to find new bound of minimization. Two different strategies can be used: one is random sensor selection, and the other is non-random sensor selection. Although random selection can reduce the space (according to our experiment shown later), it might require a large amount of iterations, especially when the sensor number is limited. So we tried to design some non-random sensor selection algorithm $s(x)_k$ that can reduce the space between D'_{KL} and D_{KL} faster. Specifically, we designed a surprise minimization sensor selection (some other methods including Markov chain are also applicable). Assume at previous observation, we have data x^1, and update model y^1 to y^2. We search for a subset of sensory space $s \subset \{m\}_k$, where:

$$surprise_{s'}(y^2, x^1) > \varrho \tag{20}$$

So it's the region that the updated model cannot well fit. We define the next sensing space s':

$$s' = (s \cup B) \cdot \omega \tag{21}$$

B is the pre-defined attentional area, which is a ”spotligh” region (subset of sensory space), indicating where we are interested in ([4]). With this pre-defined region, we can locally refine the approximation, focusing computational resources to suit the task and context at hand. And ω is to generate the randomness of sensory selection. s'^* would then used as $\{m\}_k$ for $s(x)_k$.

For a stacked autoencoders, we lower the dimension of sensing by using x' as a corrupted version of x. x' is then mapped, as with the ordinary autoencoders, to a hidden representation:

$$y = f_\theta(x') = s(Wx' + b) \tag{22}$$

from which we reconstruct $z = g_{\theta'}(y) = s(W'y + b')$. z is now a deterministic function of x' rather than x. The objective function minimized by stochastic gradient descent becomes:

$$\underset{\theta,\theta'}{\operatorname{argmin}}\, E_{p(z,x')}[L(x', g_{\theta'}(f_\theta(x')))] \tag{23}$$

IV. RESULTS

In this section, we evaluate the methodology described in the previous section with simulated experiments. The dataset contains the simulated flow count of people for a building. We generated this data based on certain generative models (Gaussian distributions and Poisson distribution). We generate the data by random-sampling these distributions with some additional noise, and get the simulated count of people hourly for a building. We then build sensing system based on the methodology described before, which tries to understand this generative model from observation. Same challenge is posed for sensing system: the target (its generative model) can change, and the observation is not complete.

1) Change of target distribution with/without memory: In the first experiment, the parameters of generative distribution changes during the observation (e.g. the mean and variance of a Gaussian change to new values). As you can see from Figure 1, when we enable the memory window, the model can quickly capture the changed distribution and minimize the KL divergence (between estimated distribution and real distribution). Using the same model without memory,the minimization takes much longer time.

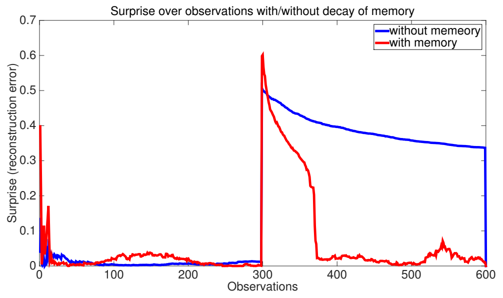

Figure 1. The target's distribution changes during the observations.

2) Change of distribution type and model representation: In the second experiment, we make more dramatic change for the generative model. So instead of changing the parameters of distribution, we change the type of distribution, from Gaussian to Poisson. As shown in Figure 2, we compared the performance of different model representations. While the Gaussian model cannot achieve KL divergence minimization. The others including dictionary learning and denoising autoencoder network work well. But we can notice that with dictionary learning method and fewer layers deep network, the KL divergence is not easy to optimize as the 5-layer deep network.

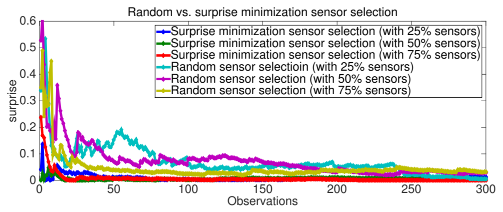

Figure 2. The distribution process changed from Gaussian to Poisson.

3) Partial observation and sensor selection: In the third experiment, we compare the random sensor selection with surprise minimization sensor selection. For a mixture Gaussian generative model with target resolution 100*100 (10000 possible sensor placements), we picked 25%, 50%, and 75% of available places for sensing, with both random and non-random sensor selection. For surprise minimization sensor selection, the initial placement is randomly picked, and then selection s' is iteratively determined by the algorithm describe before. The experiment result is shown in Figure 3, where you can find that although the random sensor selection can minimize the KL divergence between estimation and real distribution, the surprise minimization approach can achieve faster convergence.

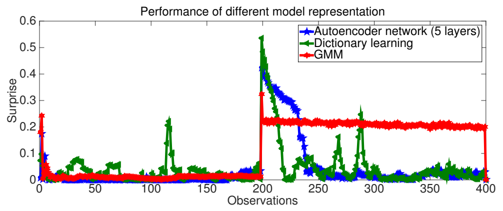

Figure 3. Surprise minimization sensor selection can better performance.

V. Conclusion

To conclude, in this paper, we 1) set a different system goal for building sensing system, which is to minimize the free energy between the model and its target; 2) adopt an attention based mechanism to detect anomaly and control the model update; 3) use a training data window as working memory mechanism; 4) utilize a deep network for model representation; 5) use partial input, based on surprise minimization sensor selection, to reduce sensing dimension.

A large amount of work is planned. we try to elaborate the components in the framework to make it more suitable for sensing intelligence: a layered or network-structured working memory will be designed to organize the data or knowledge hierarchically; model representation will be fused with external knowledges like ontology and support reasoning. Also, in the next step, we will try to apply the methodology to ECG and EEG anomaly detection. We believe for this kind of detection system, two features are highly required: firstly, it should be able to detect abnormal situation, which normally means there is something wrong or even dangerous and needs actions to be taken; secondly, it should base on personalized model, instead of an average model for large population. Therefore, the proposed methodology is believed to be suitable.

VI. Acknowledgment

This project is partly supported by the National Natural Science Foundataion of China (No. 61104215).

References

[1] Friston, K. (2010). The free-energy principle: a unified brain theory?. Nature Reviews Neuroscience, 11(2), 127-138.

[2] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research, 11, 3371-3408.

[3] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning (pp. 1096-1103). ACM.

[4] Whiteley, L., & Sahani, M. (2012). Attention in a Bayesian framework. Frontiers in human neuroscience, 6.

[5] Itti, L., & Baldi, P. F. (2005). Bayesian surprise attracts human attention. In Advances in neural information processing systems (pp. 547-554).

[6] Moran, J., & Desimone, R. (1985). Selective attention gates visual processing in the extrastriate cortex. Science, 229(4715), 782-784.

[7] Yuille, A., & Kersten, D. (2006). Vision as Bayesian inference: analysis by synthesis?. Trends in cognitive sciences, 10(7), 301-308.

[8] Barlow, H. B. (1961). Possible principles underlying the transformations of sensory messages.

[9] Olshausen, B. A., & Field, D. J. (1996). Natural image statistics and efficient coding*. Network: computation in neural systems, 7(2), 333-339.

[10] Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. Neural computation, 7(6), 1129-1159.

[11] Hyvrinen, A., Karhunen, J., & Oja, E. (2004). Independent component analysis (Vol. 46). John Wiley & Sons.

[12] Zhong, N., Liu, J., Yao, Y., Wu, J., Lu, S., Qin, Y., & Wah, B. (2007). Web intelligence meets brain informatics. In Web Intelligence Meets Brain Informatics (pp. 1-31). Springer Berlin Heidelberg.

[13] Ma, J., Wen, J., Huang, R., & Huang, B. (2011). Cyber-individual meets brain informatics. IEEE Intelligent Systems, (5), 30-37.

[14] Feldman, H., & Friston, K. J. (2010). Attention, uncertainty, and free-energy. Frontiers in human neuroscience, 4.

[15] Yun, Q. S., & Sun, H. (2000). Image and Video Compression for multimedia engineering. CRC.

[16] Markou, M., & Singh, S. (2003). Novelty detection: a reviewpart 2: neural network based approaches. Signal processing, 83(12), 2499-2521.