Machine Learning Developments around the PSQA Project

Gerardo Rubino

in MAKI's workshop on
Machine Learning in Communication Systems

Machine Learning Developments around the PSQA Project

G. Rubino

INRIA Rennes, France

Darmstadt, March 2018

Outline

Outline

2 — Measuring Perceptual Quality

- Consider any kind of application or service centered on transporting audio and/or video signals over the Internet.
- The media can be sent one-way (e.g., a video streaming service) or two-ways (e.g., an IP-telephony application).
- For many reasons (compression techniques used, congestion in the network – leading to delays, or to losses, external perturbation agents such as interferences in some networking technologies, etc.) transmission can suffer from content degradation.

Perceptual quality

- Perceptual quality refers to the (subjective) perception (the view, the feeling) the user has on the "value" of this media transmission system, on the quality of what she receives, on the impact of the degradations due to the transmission over the network.
- Two main characteristics:
  - It is, by definition, a subjective concept.
  - It lacks a formal definition.
- Critical problem (solved): how to measure it? How to measure it automatically?

Subjective testing

- Subjective testing is the standard way of measuring perceived quality.
- A panel of appropriately chosen human observers is built, and a set of media sequences is shown to the panel members.
- Following specific rules (absolute rating, comparisons...) the observers say how they perceive the quality of the sequences.
- In case of interactive applications, observers can work by pairs, or in some cases, interact with a robot.

Subjective testing

- The result is a table M where component $M_{h,\sigma}$ is the value given by human h to sequence σ.
- A statistical procedure is then followed to filter these results in order to remove outliers.
- At the end, if the set of surviving observers is $\mathcal{S}$, the quality of sequence σ, its *MOS* (Mean Opinion Score) value is

$$MOS(\sigma) = \frac{1}{|\mathcal{S}|} \sum_{h \in \mathcal{S}} M_{h,\sigma}.$$

- The procedure is surprisingly robust: with the same set of sequences but a different panel, the same subjective test gives results extremely close to the first one: if μ_σ and μ'_σ are the MOS values of sequence σ in both panels, the number $\sum_\sigma (\mu_\sigma - \mu'_\sigma)^2$ is pretty "small".

Subjective testing (cont.)

- For each type of media, there are norms specifying how to perform the corresponding subjective test. A reference here is the production of the ITU (International Telecommunication Union).
- For instance, some norms ask users to watch video sequences potentially degraded by some noisy effect, and rate them from 1 to 5 according to the following table:

MOS score	description
5	degradation imperceptible
4	degradation perceptible but not annoying
3	degradation slightly annoying
2	degradation annoying
1	degradation very annoying

Subjective testing (cont.)

- The norms specify also other constraints concerning details such as the lengths of the sequences, the timing of the subjective testing session, its own length, the experimental conditions (e.g. monitor contrast or viewing distance in case of video), the suggested panel's size, etc.
- In video, most subjective tests norms belong to the following classes:
  - Single Stimulus (SS): only the distorted signal is shown to the panel,
  - Double Stimulus Continuous Quality Scale (DSCQS): both the original and distorted sequences are shown simultaneously
  - Double Stimulus Impairment Scale (DSIS): first the original sequence and then the distorted one are shown.

Subjective testing (cont.)

- The panel must also be built following some rules (similar to those followed to make opinion polls). The idea is to build a representative sample of the users of the considered application/service, or of the subfamily of users we are interested in.
- Obviously, subjective testing is costly, takes time, and by construction is not usable for real time perceived quality assessment.
- Comment: one of the goals of my team at Inria is to use quality assessment techniques for monitoring and for control purposes. So, only methods that work automatically qualify.

Objective testing

- Objective testing means evaluating the perceived quality automatically, without using any panel of users.
- To give an example, consider video sequences. Some objective testing methods (coming from the coding area) are based on the PNSR (Peak Signal To Noise Ratio) of two images A and B:
  - for instance, assume the two images are composed of M rows and N columns of 8-bits pixels (monochrome case).
  - The MSE (Mean Squared Error) between the two images is

$$\mathrm{MSE}(A,B) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\Big(A_{i,j} - B_{i,j}\Big)^2 \in [0..255^2].$$

  - The PSNR between the two images is defined only if they are different ($\mathrm{MSE} \neq 0$) and its value (in dB) is

$$\mathrm{PSNR}_{\mathrm{dB}}(A,B) = 10\log_{10}\Big(\frac{255^2}{\mathrm{MSE}(A,B)}\Big).$$

Problems with PSNR approach

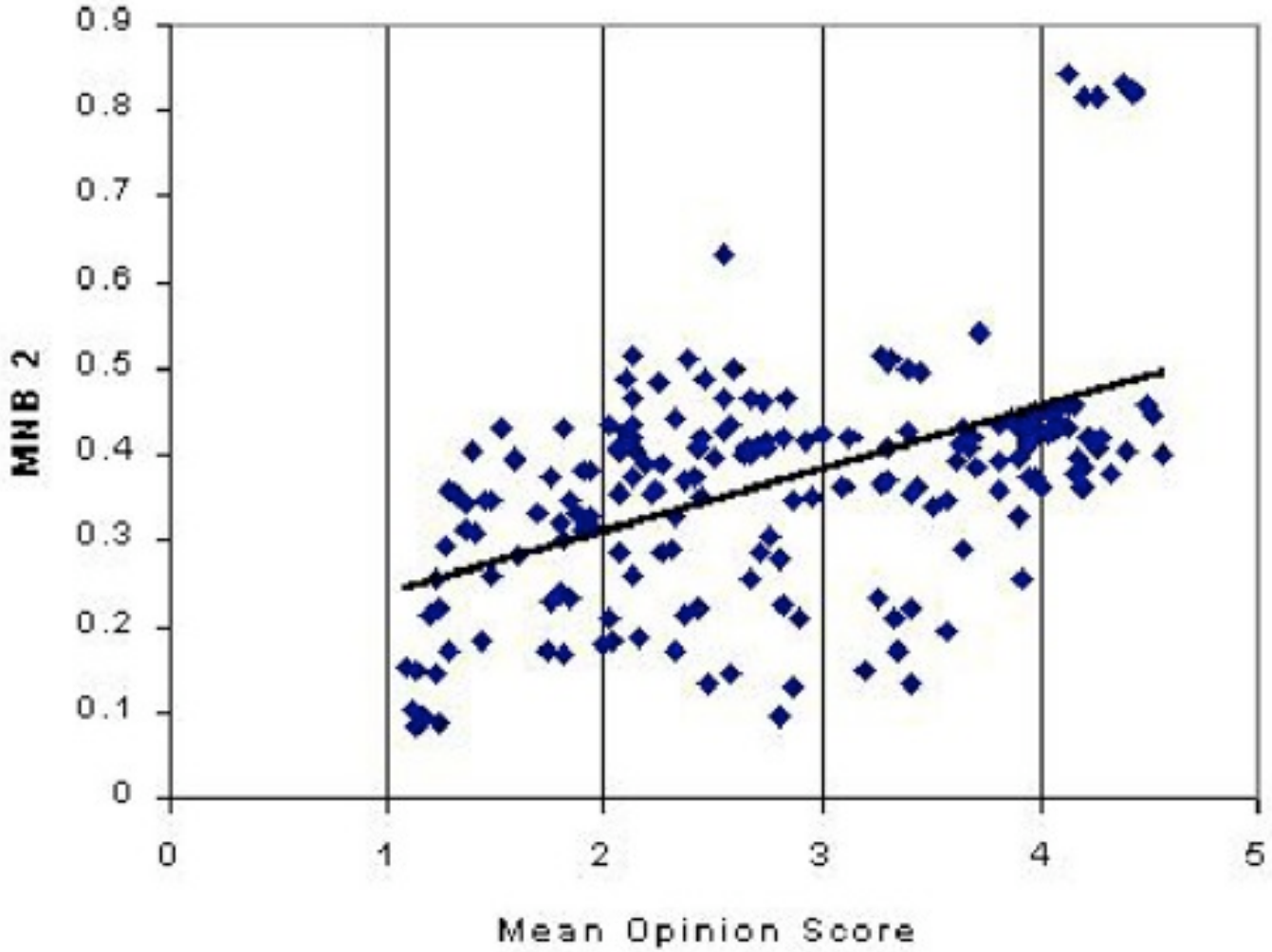
Noise in the sky

Problems with PSNR approach...

Same amount of noise in the grass

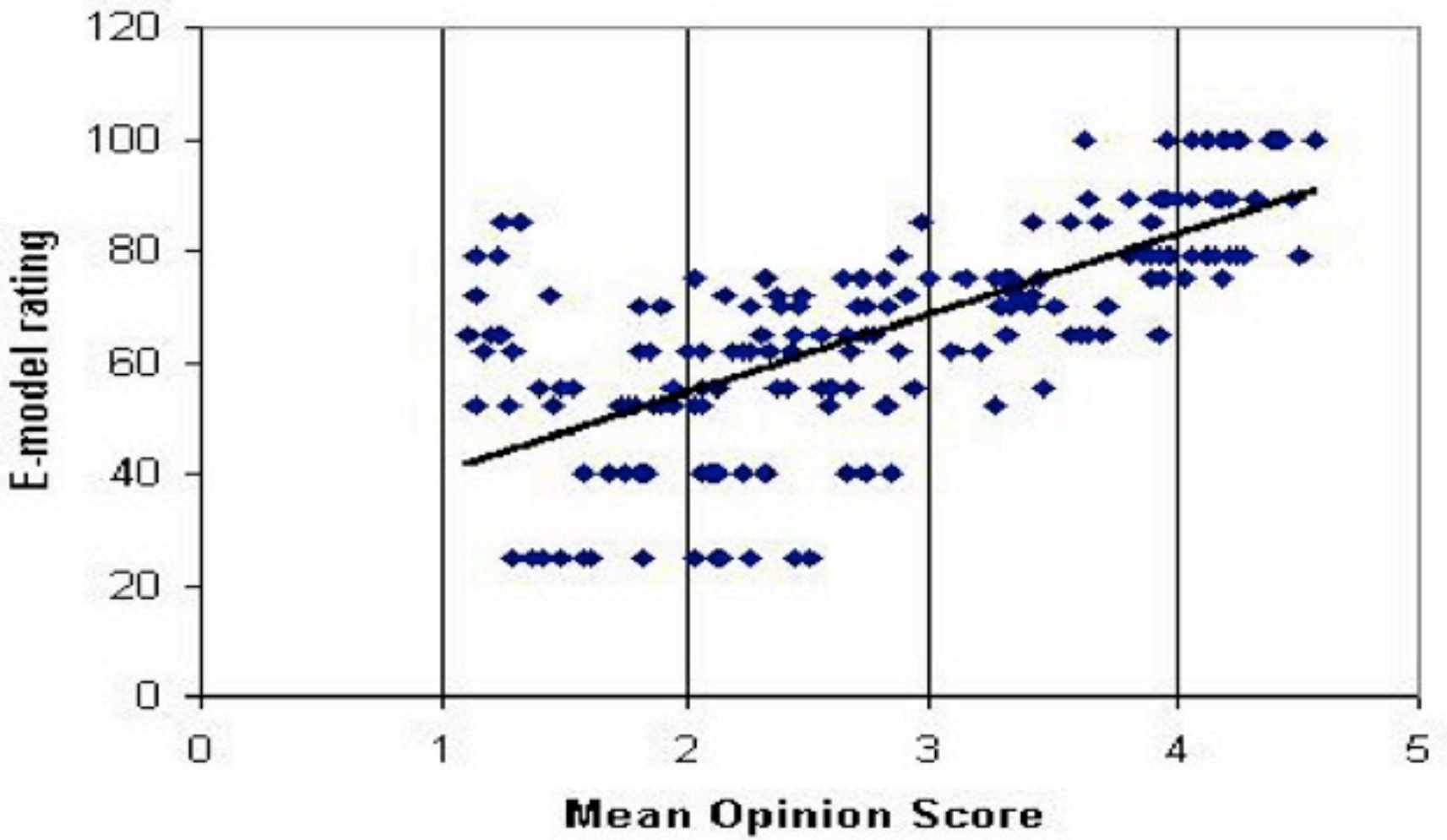
Example in audio

Bad correlation between an objective metric and subjective tests in audio.



Example in VoIP

Bad correlation between an objective metric and subjective tests in Voice over IP.



Outline

PSQA: Pseudo Subjective Quality Assessment

- In the Dionysos team we promote our solution called PSQA to all the initial problems and many more, for all types of media considered, and for both one-way and two-ways communications.
- PSQA is a metric with no reference, automatic, (so far, "optimally") accurate, and it works in real time if necessary or useful.
- PSQA is network-dependent and application-dependent.
- It is a parametric approach (a "black-box" approach), mapping QoS parameters and source-based parameters into perceptual quality (into a MOS value).

PSQA (cont.)

- The mapping is based on statistical learning tools and it is built in such a way that it has nice mathematical properties. We can use it in many manners, for instance,
  - for sensitivity analysis: which is the dominating factor having an impact on quality? where is it significantly dominating?
  - for inverse problems: which values of the considered factors lead to a high enough MOS value?
  - etc.
- More specifically, so far PSQA functions are a particular class of rational functions coming from the Random Neural Network (or G-network) area.

PSQA (cont.)

- **Example in video** (simplified version): a PSQA monitoring module measures, at the receiver's position, the instantaneous packet loss rate LR, the Bit Rate of the connexion BR and the Frame Rate FR, and calls a PSQA function ν_1(LR,BR,FR) which provides in a few msec the perceptual quality value.
- **Example in VoIP**: a PSQA monitoring module measures, at the receiver's position, the instantaneous packet loss rate LR, the average size of a burst of loss packets MLBS, the Bit Rate of the connexion BR, and the Offset of the FEC (Forward Error Correction) OFEC, and calls a PSQA function ν_2(LR,MLBS,BR,OFEC) which provides in a few msec the perceptual quality value.

How PSQA works

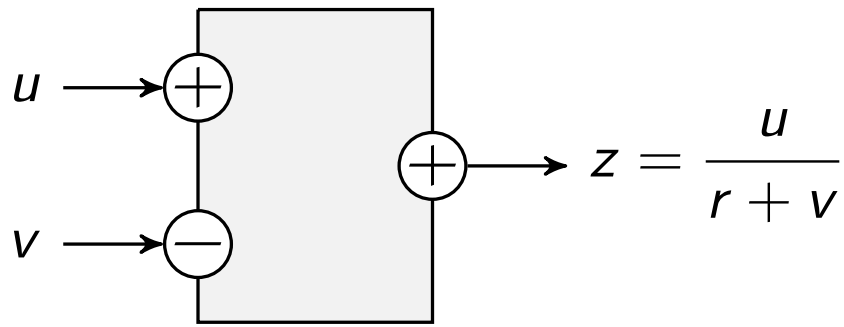
- First of all, we select
  - (i) measurable QoS metrics characterizing the state of the network (logically speaking, instantaneous measures), and assumed, a priori, to have an impact on the Perceptual Quality,
  - (ii) and metrics related to the channel or to the source, again, expected to have impact on the PQ.
- Example in video: (i) the instantaneous packet loss rate LR, (ii) the Bit Rate BR and the Frame Rate FR.
- Example in VoIP: (i) the instantaneous packet loss rate LR, the average size of a burst of loss packets $MLBS$, (ii) the Bit Rate BR and the Offset of the FEC (Forward Error Correction) $OFEC$.

- Let $\vec{x} = (x_1, \ldots, x_n)$ denote the vector whose components are the n chosen metrics. We call it configuration.
- Configurations live in some product space $S_1 \times \cdots \times S_n$.
- Our postulate: PQ depends only on $\vec{x}$ (when "things go well") and not on signal content.
- We select a few short signals (following standards) representative of the application or service target, $\sigma_1, \ldots, \sigma_K$.
- We build a pretty small set of K configurations ($K = 100, 200, \ldots$) by a mix of random sampling and quasi-Monte Carlo (or weak discrepancy sequences). Call them $\vec{\gamma}_1, \ldots, \vec{\gamma}_K$.
- Last, we build a platform allowing to send signals σ_i through a simulated or deployed network where we can simultaneously control all components of the configurations.

- Then we send different original σ_i using the platform, when the configuration is $\vec{\gamma}_j$ using one configuration at a time, and obtain a possibly degraded sequence σ'_j. We show it to a panel of humans and we obtain its PQs Q_j.
- We thus obtain a set of K sequences, with variable but known PQ (MOS values) coming from subjective testing sessions, and for each, we know which configuration was used to obtain it.
- Then, we use a RNN (a G-network), of the classical feedforward type with 3 layers, to learn the mapping from configurations to PQ (a mapping from QoS$\times$"channel state" to MOS values). Data: the M pairs $\left(\vec{\gamma}_j, Q_j\right)$.
- In the example of video, we obtain a function $\nu(LR, BR, FR)$. In the VoIP example, we obtain a function $\nu(LR, MLBS, BR, OFEC)$.

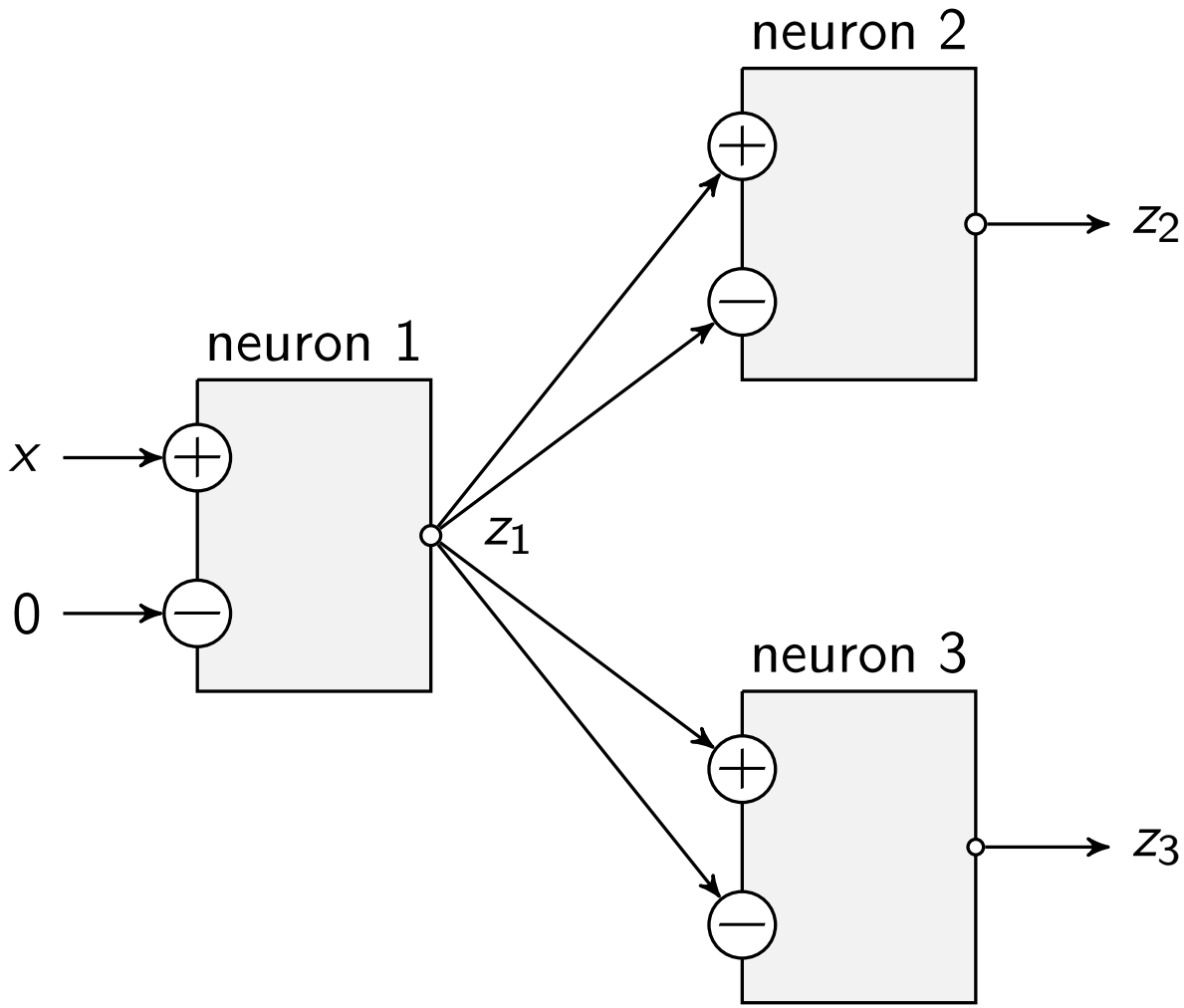
On the learning tool used

- A Random Neuron is a parametric positive real function of 2 real positive variables (we also say "ports"), one called the "positive port", the other one being the "negative port".

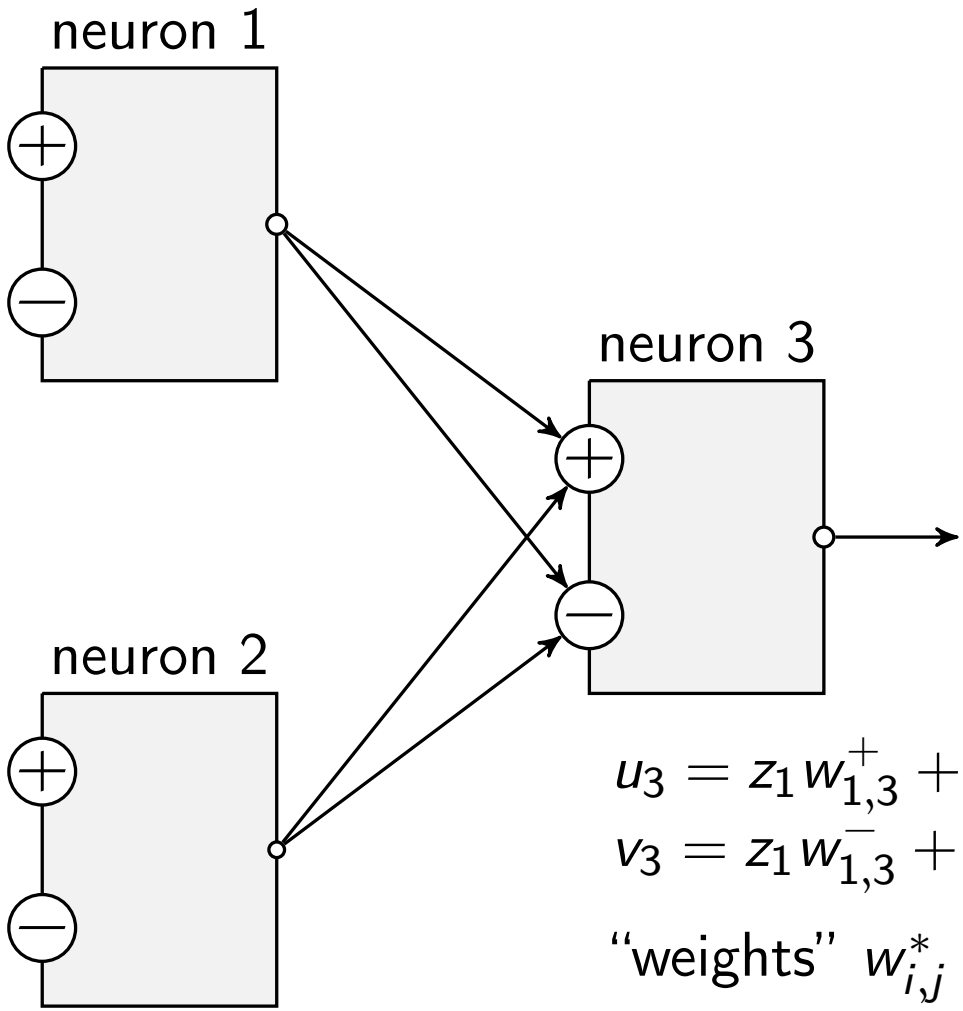


- We have $r > 0$, the *rate* of the neuron, seen as a parameter, and its two inputs, $u, v \geq 0$. The output is $z > 0$.
- There are several variants of this model. In the main and original one, we must use the function $z = \min(u/(r+v),\ 1)$.

A Random Neural Network (RNN) is a set of interconnected RNs. The connections are weighted by positive reals.



Jonctions are additive:

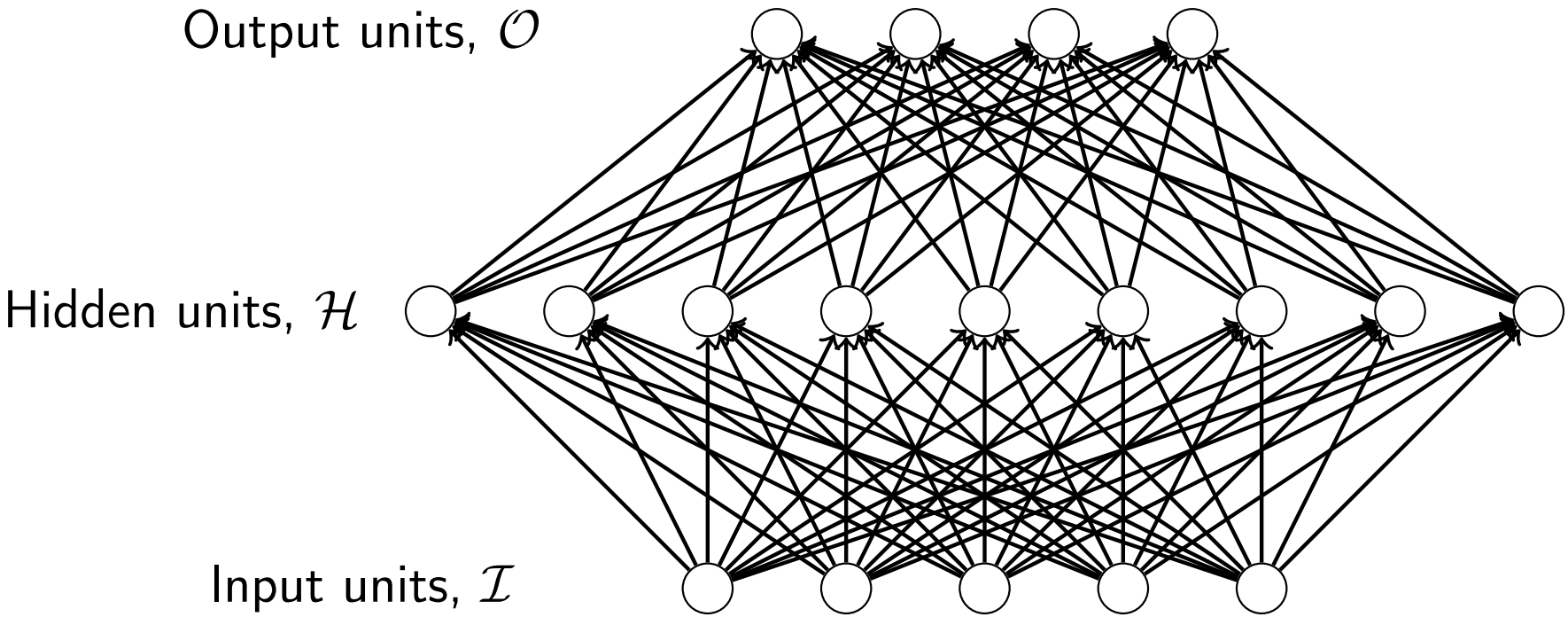


$$u_3 = z_1 w^+_{1,3} + z_2 w^+_{2,3}$$
$$v_3 = z_1 w^-_{1,3} + z_2 w^-_{2,3}$$

"weights" $w^*_{i,j}$ are nonnegative

A 3-layer Neural Network

This is the basic RNN architecture used in PSQA 1.0.



- Assume the negative ports of input neurons aren't used. Assume a single output neuron, so, a scalar network output.
- Call x_i the signal arriving at the positive port of input neuron i. Then, we can explicitly write the network output as a function of the inputs.

$$z_o = \frac{\sum_{h\in\mathcal{H}} \frac{\sum_{i\in\mathcal{I}} x_i w^+_{i,h}}{r_h + \sum_{i\in\mathcal{I}} x_i w^-_{i,h}} w^+_{h,o}}{r_o + \sum_{h\in\mathcal{H}} \frac{\sum_{i\in\mathcal{I}} x_i w^+_{i,h}}{r_h + \sum_{i\in\mathcal{I}} x_i w^-_{i,h}} w^-_{h,o}}$$

- This shows that the output is a rational function of the input. This allows many treatments. Also, for learning, costs (errors) are rational functions of weights, with many advantages.

Probabilistic origin

- Consider a dynamical system and an integer positive state variable called *potential* associated with.
- "Positive" spikes arrive to the system according to a Poisson process with some rate $\lambda^+ > 0$, and "negative" ones arrive according to another Poisson process with rate $\lambda^- \geq 0$. Both processes are independent of each other.
- When a positive spike arrives, the system's potential increases by 1.
- As far as the potential is > 0, the system sends spikes outside according to a Poisson process with some rate $\mu > 0$, which is independent of the two arrival processes.
- When a negative signal arrives, the potential is decreased by 1 unless it was already 0, in which case nothing happens.

- Call P_t the potential at time t.
- It can then be proved that when the stochastic process (P_t) (Markov) is ergodic, the probability ρ that, in equilibrium, the potential is > 0, is given by the expression

$$\rho = \frac{\lambda^+}{\mu + \lambda^-}, \text{ which is } < 1$$

(the process is topologically equivalent to the $M/M/1$ queue).

- The correspondence between this dynamical system and Random Neurons allows to derive several interesting properties that are useful for many applications.
- In the unstable case (when $\lambda^+ > \mu + \lambda^-$), at the limit (in time), the probability that the potential is strictly positive is 1, so that, in the general case, we can write

$$\rho = \min\Big(\frac{\lambda^+}{\mu + \lambda^-},\ 1\Big).$$

- Denote $p_n(t) = \mathbb{P}(P_t = n)$.
- To get an insight about the complexity of this distribution, its most compact known closed expression is the following (given for simplicity, in the case of $P_0 = 0$:[1],

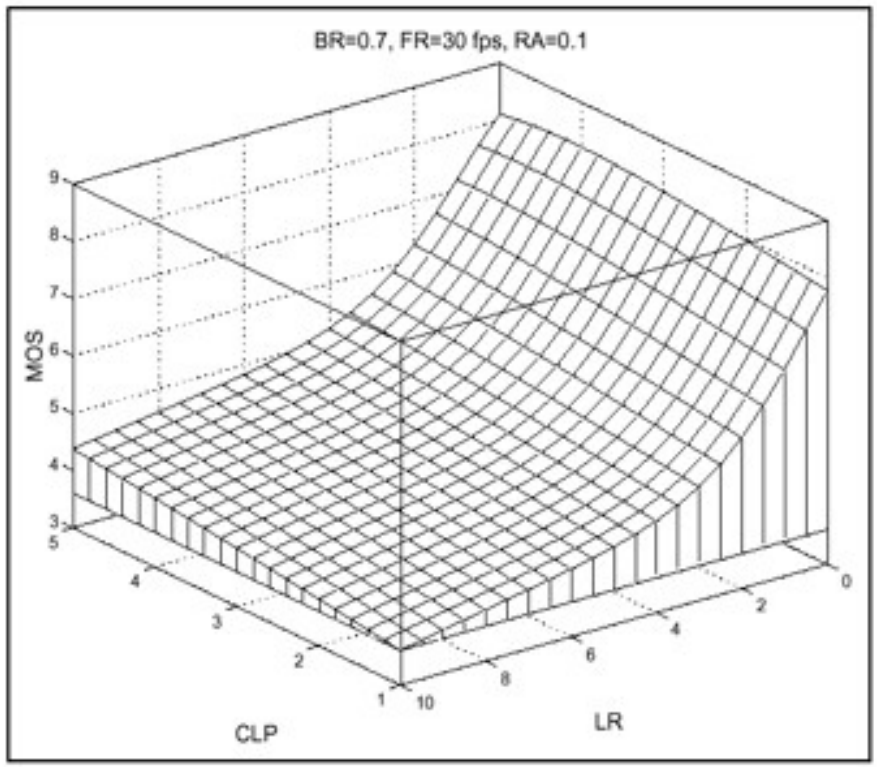
$$p_n(t) = \left(\frac{p}{q}\right)^n \sum_{j=n}^{\infty} e^{-\psi t} \frac{(\psi t)^j}{j!} \sum_{k=0}^{\lfloor \frac{j-n}{2} \rfloor} \frac{j+1-2k}{j+1} \binom{j+1}{k} p^k q^{j-k}$$

  where $p = \lambda^+/(\mu + \lambda^-)$, $q = 1 - p$ and $\psi = \lambda^+ + \lambda^- + \mu$.
- The general case is written as a function of the case of $P_0 = 0$, using modified Bessel functions of the first kind.

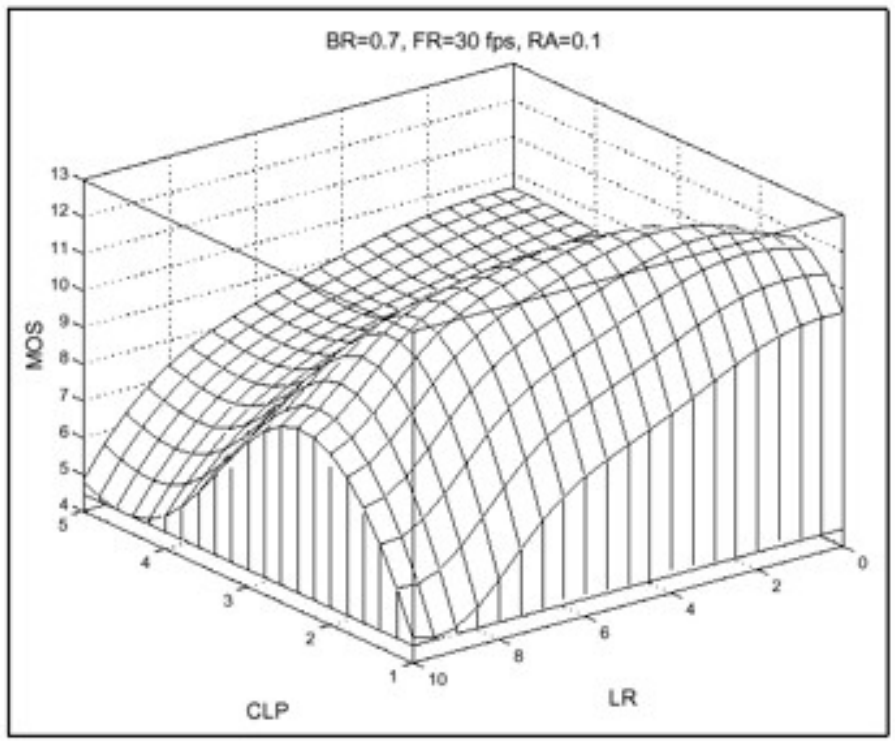
[1] P. Leguesdron, J. Pellaumail, G. Rubino, B. Sericola, *"Transient analysis of the M/M/1 queue"*, Advances in Applied Probability 25, pages 702–713, 1993.

Why using RNNs in PSQA project?

Because of the favorable comparison with standard software (at the time of the beginning, several years ago). An example: with the same data and the same neural network size, on the left, PSQA, and on the right, Matlab (an old version of the ToolBox on learning techniques).

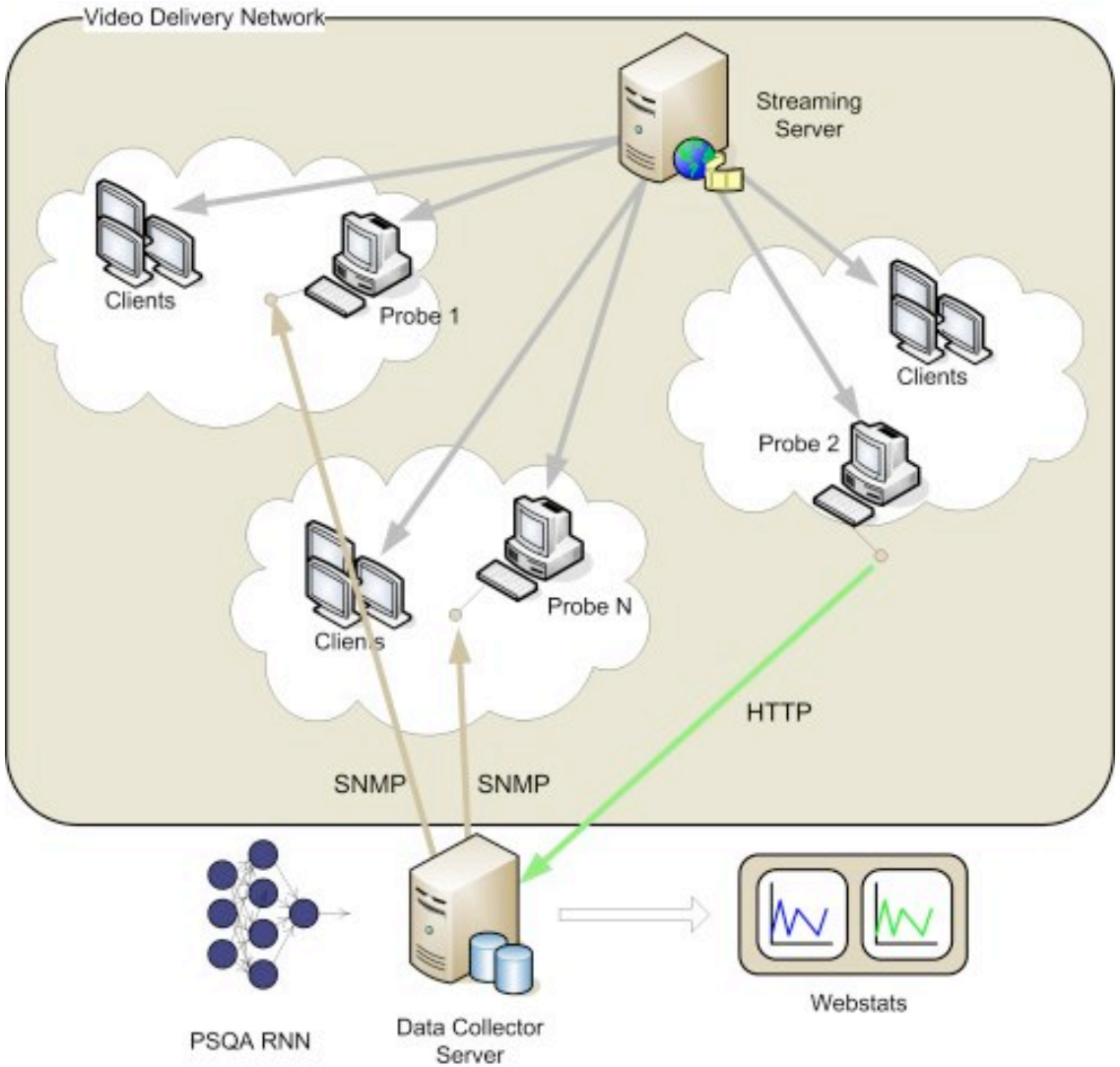


(a) Correctly trainined



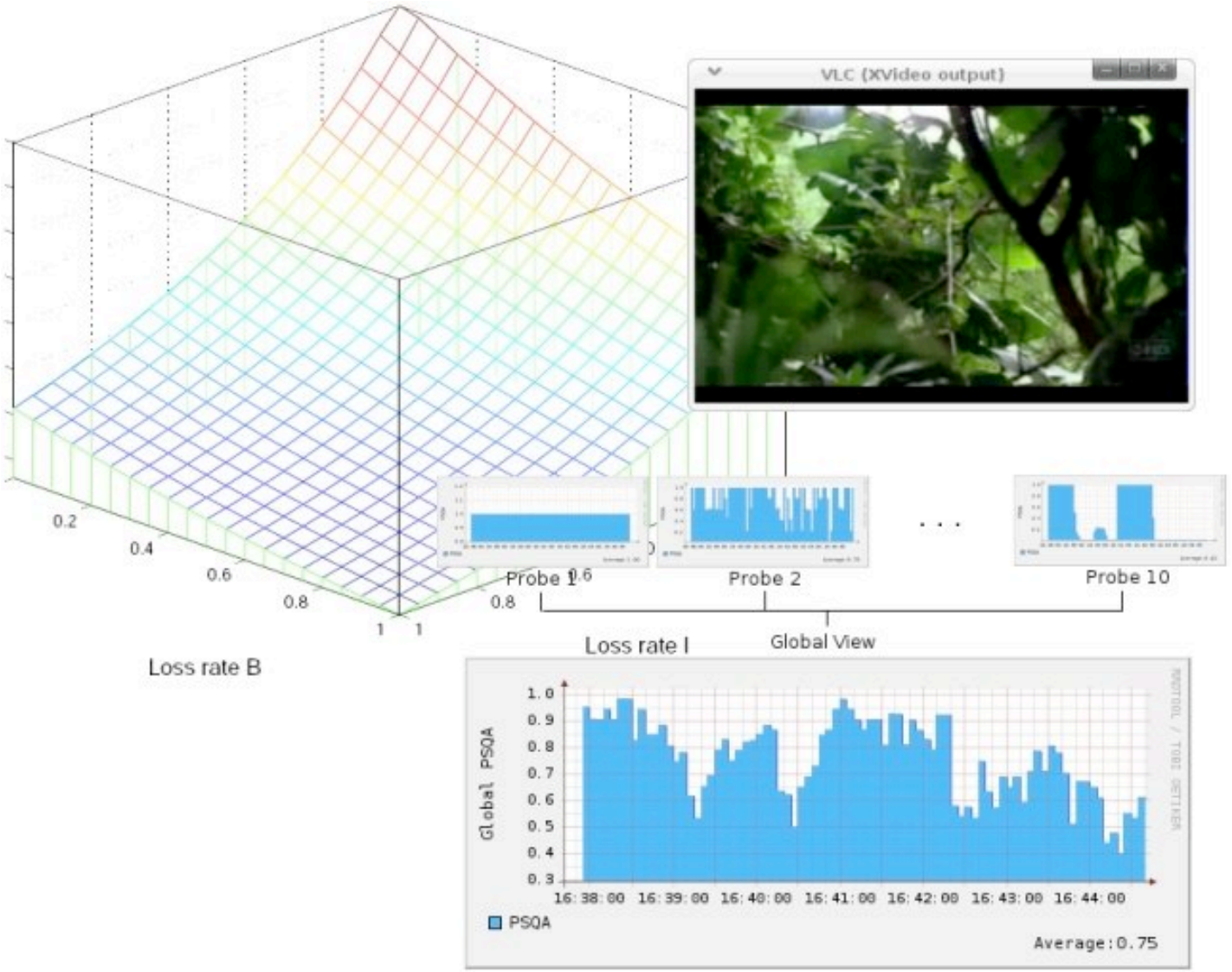
(b) Example of an over-trained ANN

Auditing the network with PSQA



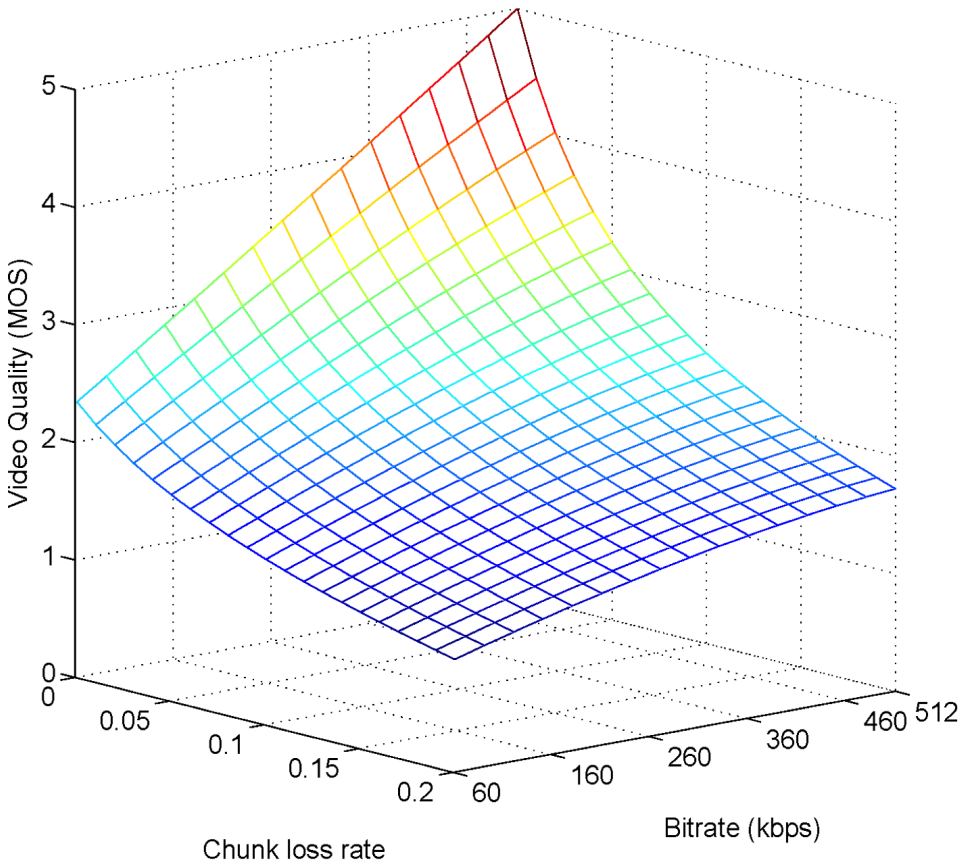
Auditing the network with PSQA (cont.)

A "graphical" overview:



Usage examples

Video quality analysis (P2P)



Analytical developments in VoIP applications

- Consider a bottleneck modeled as an $M/M/1/N$ with load $\rho \neq 1$.
- Classical approach: choose N and ρ such that $p_L < \varepsilon$, using

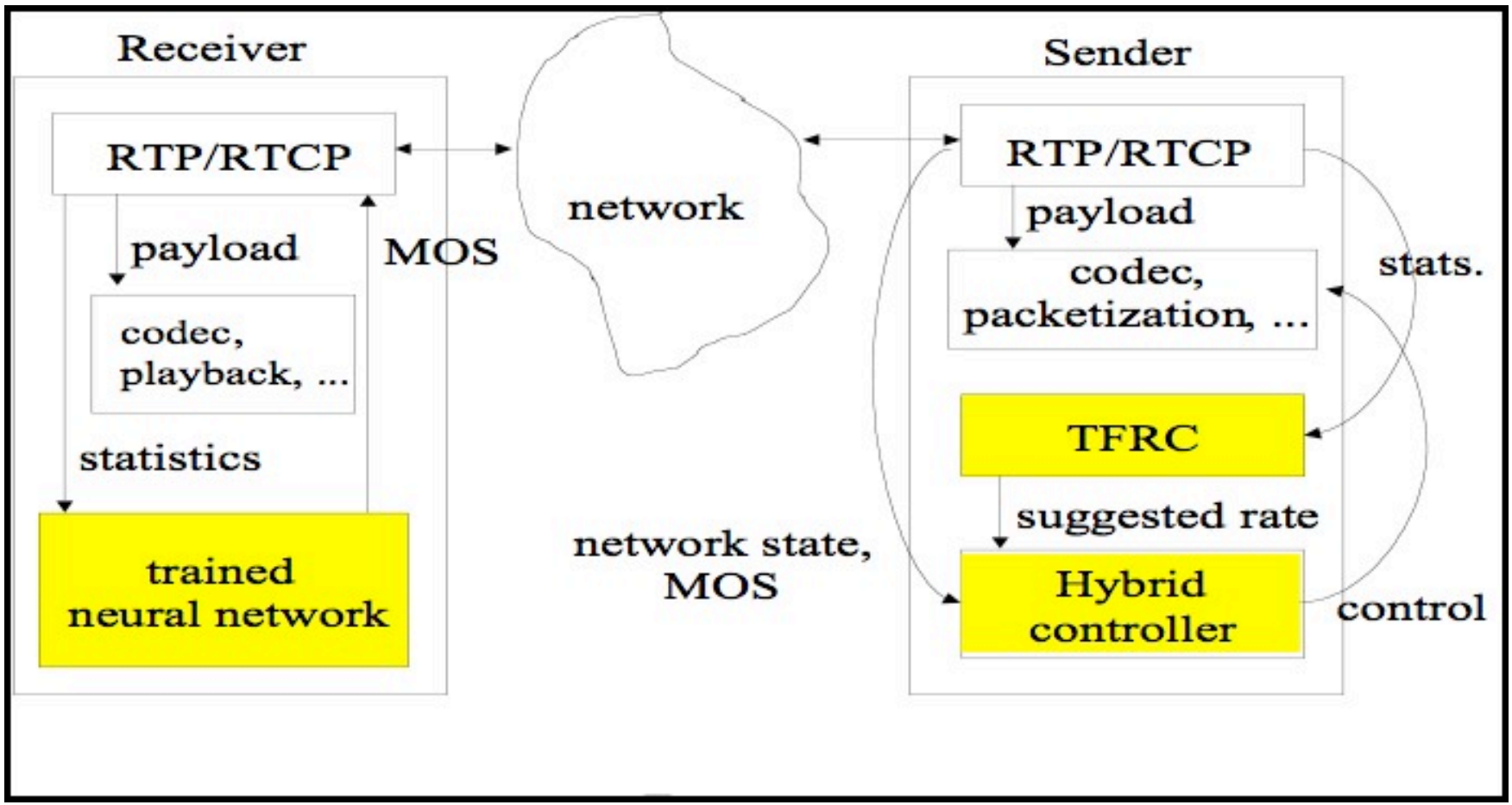
$$p_L = \frac{1-\rho}{1-\rho^{N+1}}\rho^N$$

- PSQA-based approach: choose N and ρ such that *the perceived quality* Q satisfies $Q \geq Q_0$, using

$$Q = \frac{13.3 + 2.01\rho + 6.74\rho^N - 20.0\rho^{N+1} - 2.01\rho^{N+2}}{16.6 + 3.99\rho + 93.3\rho^N - 110\rho^{N+1} - 3.99\rho^{N+2}}$$

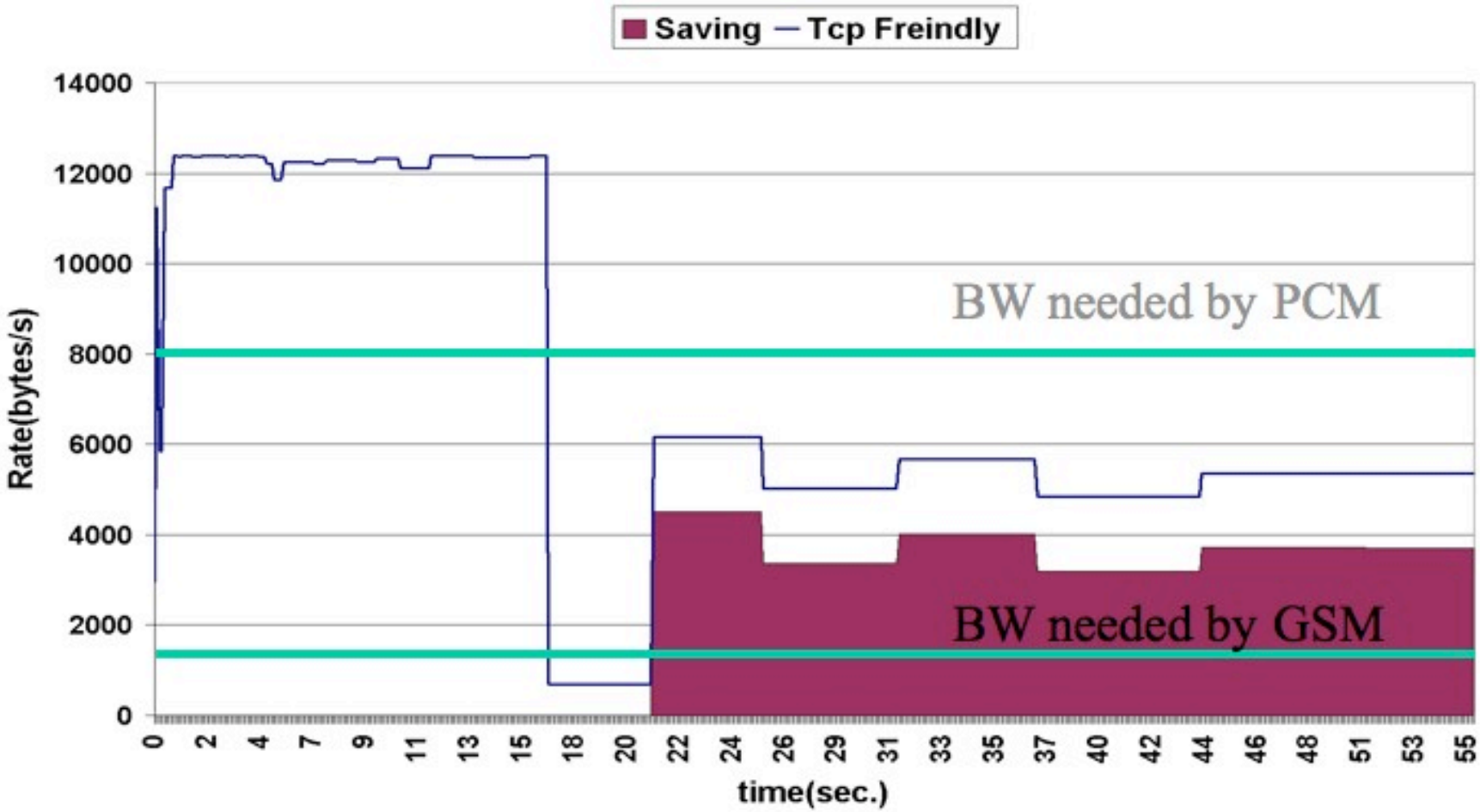
Controlling voice transport: a toy example

The general setting:



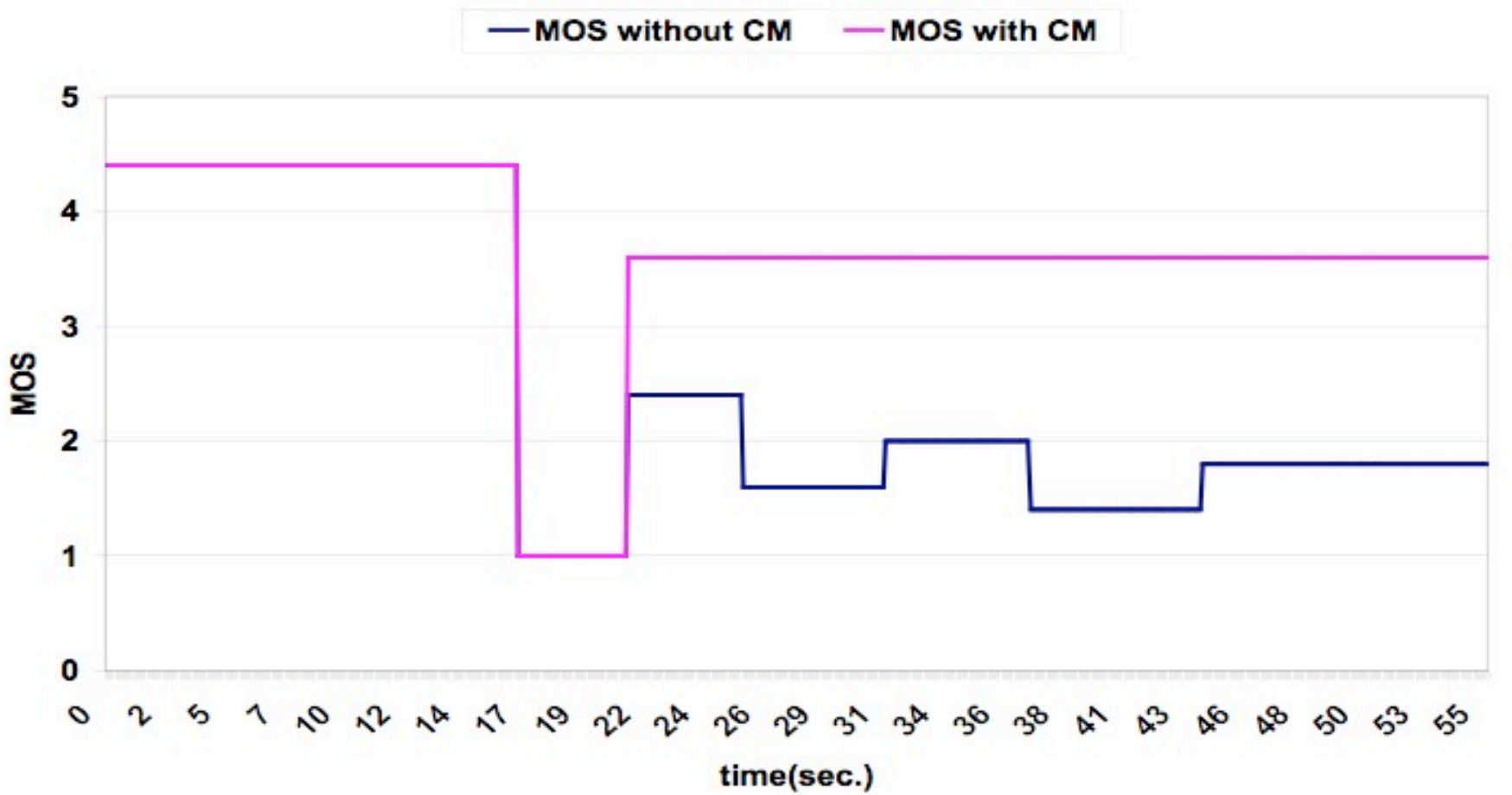
Controlling voice transport: a toy example (cont'd)

Bandwidth evolution:



Controlling voice transport: a toy example (cont'd)

Using PSQA:



Outline

Reservoir Computing

- They are an attempt to develop models that uses the potential for *memorization* of recurrent neural networks without the difficulties in the training process of these networks.
- They appeared at the beginning of the 2000s, and they are known today under the name of *Reservoir Computing* (RC) paradigm.
- The two most popular RC models are
  - the *Echo State Network (ESN)*
    (see H. Jaeger, "The *echo state* approach to analysing and training recurrent neural networks, "German National Research Centre for Information Technology, Tech. Rep. 148, 2001)
  - and the *Liquid State Machine (LSM)*
    (see W. Maass, "Liquid state machines: Motivation, theory, and applications," in *Computability in Context: Computation and Logic in the Real World*, Imperial College Press, 2010, pp. 275-296).

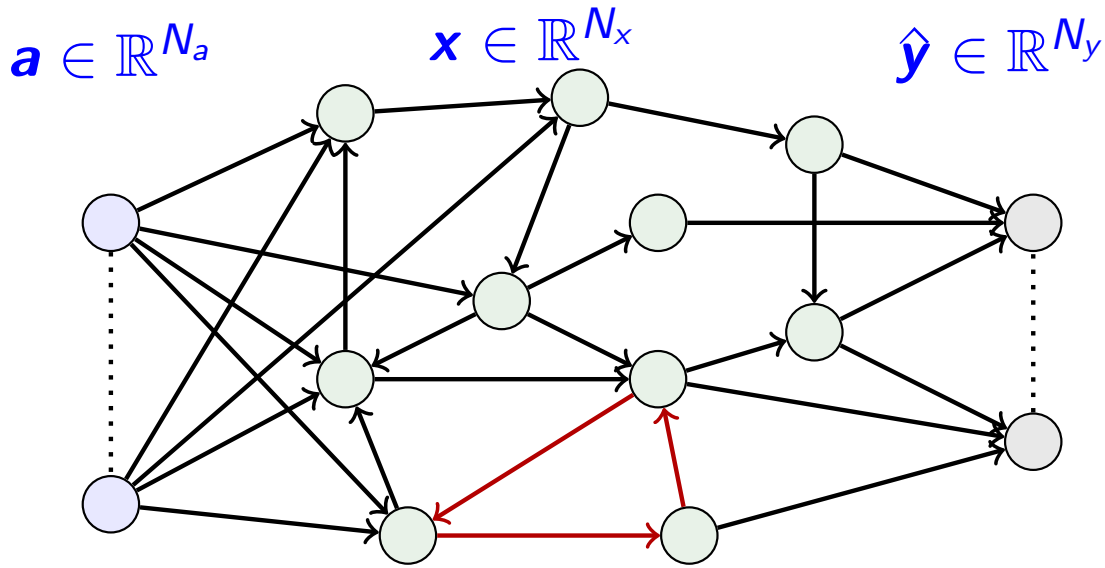
Echo State Networks (ESNs)

The main representative of the family.

- An untrained recurrent part called *reservoir*.
- A memory-less supervised learning tool called *readout*.

Three-layered neural networks:

- Input layer
- Hidden layer
- Output layer



The learning process is restricted to the output weights (readout).

Echo State Queueing Networks (ESQNs)

Echo State Queueing Networks: applying the RC idea to a G-network.

- Three sets of neurones, recurrent topology "in the middle".
- Input at time t, $\boldsymbol{a}(t) = (a_1(t), \ldots, a_{N_a}(t))$:
  $\rho_u(t) = a_u(t)/r_u$, (in general, $a_u(t) < r_u$), for $u \in (1..N_a)$.
- For all reservoir units $u = N_a + 1, \ldots, N_a + N_x$,

$$\rho_u(t) = \frac{\sum_{v=1}^{N_a} \frac{a_v(t)}{r_v} w^+_{u,v} + \sum_{v=N_a+1}^{N_a+N_x} \rho_v(t-1) w^+_{u,v}}{r_u + \sum_{v=1}^{N_a} \frac{a_v(t)}{r_v} w^-_{u,v} + \sum_{v=N_a+1}^{N_a+N_x} \rho_v(t-1) w^-_{u,v}}. \quad (1)$$

Weights notation as classical NNs: u sending spikes to v is denoted by $w^{+/-}_{v,u}$. Notation r_u is for the rate of unit (neurone) u.

- The input space in then projected into a new "larger" space.
- We compute a linear regression from the projected space to the output space.
- Thus, the network output $\hat{\boldsymbol{y}}(t) = (\hat{y}_1(t), \ldots, \hat{y}_{N_b}(t))$ is computed for any $m \in [1..N_b]$:

$$y_m(t) = w^{\text{out}}_{m,0} + \sum_{i=1+N_a}^{N_a+N_x} w^{\text{out}}_{m,i} \rho_i(t). \tag{2}$$

- Learning process:
  The output weights w^{out}_* can be computed using any fast procedure, for instance, Least Mean Square algorithms.
- Remark: we can replace this simple structure by, for instance, a classical feedforward 3-level RNN (to be explored).

Outline

On PSQA 2.0

- In the context of some preliminary work being done with industry, we are now exploring the relevance of a new set of tools, for a 2.0 version of our technology.
- One of the goals is to predict the Perceptual Quality in a close future (instead of evaluating it at present time).
- Once again, the idea of PSQA is to estimate the Perceptual Quality by a function $\nu(x_1, \ldots, x_n)$ where the x_is are metrics characterizing QoS aspects of the network and metrics related to properties of the connection (the channel).
- Some of the x_is don't change with time, but some of them evolve with time. Predicting the Perceptual Quality translates into predicting those time series.

Transfert to industry

- The technology is now mature for industry transfert.
- It has been already tested by many companies in research projects. We also deployed a P2P network in South America, for extending an operator's network of boxes, successfully.
- We are now looking for companies working in measuring or related areas, to implement an operational version of our experimental Network Perceptual Quality Analysis System.
- We intend also to add some other services to the initial measuring tools such as
  - the predictor previously described,
  - computation of sensitivities (how important is this factor, say the Packet Loss Rate, to Perceptual Quality?),
  - inverter: given a quality level Q_0, how far are our selected factors from the area corresponding to a quality level $< Q_0$?

Conclusions

- Evaluating the Perceived Quality of video/audio/voice contents on the Internet is a fundamentally subjective task.
- Our PSQA technique is able to do it with the same accuracy but automatically and in real time.
- It is based on the evaluation of a sample of sequences of video/voice/... whose quality is evaluated by humans, but that where obtained from an original sequence after being possibly degraded by its traversal of a network between the sender and the receiver.
- The characteristics of the network and the communication channel are controlled by a set of parameters $(x_1, \ldots, x_n) = \vec{x}$.
- Then, a Machine Learning tool (called Random Neural Network) is used to map $\vec{x}$ to the quality value given by humans to the sequences.

On the used learning tools

- We are now exploring the capability of the Random Neuron idea for building Deep networks. The reason is another idea but now using massive data coming from two sources: users (crowdsourcing) and objective tests.
- We have obtained some preliminary results but their interpretation resists analysis.
- On the ESQN model, the main current task is the theoretical analysis of the model (e.g., stability issues), which is hard because of its strongly nonlinear structure.
- Last, up-to-date machine learning tool with massive data volumes appear now when following the PSQA approach on encrypted data.