Artificial Neural Networks

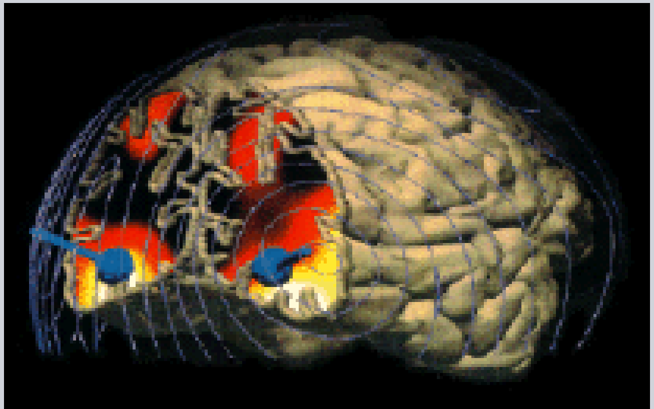
Dr. Hans Georg Zimmermann

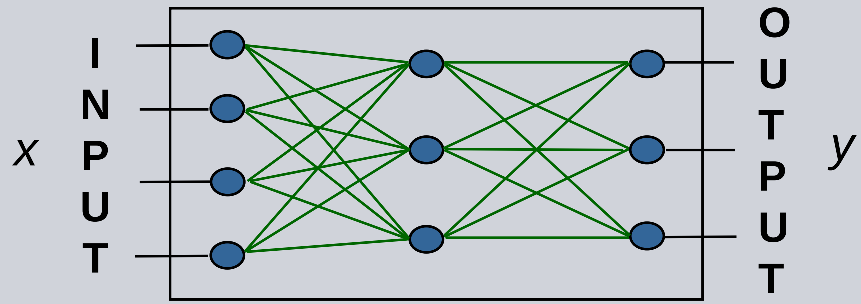
Fraunhofer IIS, Supply Chain Services, Nürnberg

Email : hans.georg.zimmermann@scs.fraunhofer.de

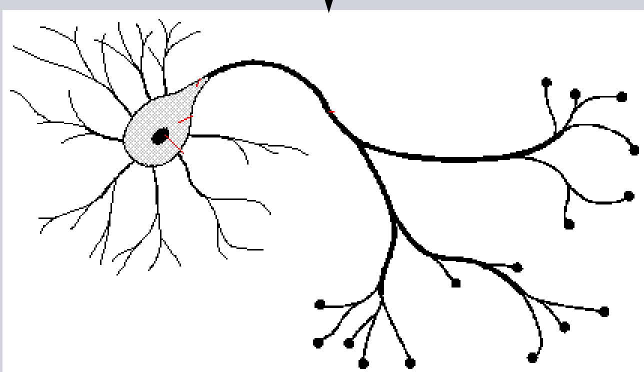
Fraunhofer IIS

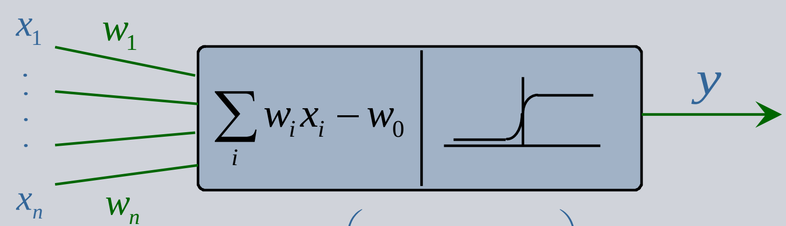
Neural Networks - from Biology to Mathematics





$$y = f\left(W_2\, f\left(W_1 x - w_1\right) - w_2\right)$$





$$y = f\left(\sum_i w_i x_i - w_0\right)$$

Neural Networks are nested expressions of alternating linear- and nonlinear functions.

The concept was born in 1943 to show an equivalence between brains & computers (logic)

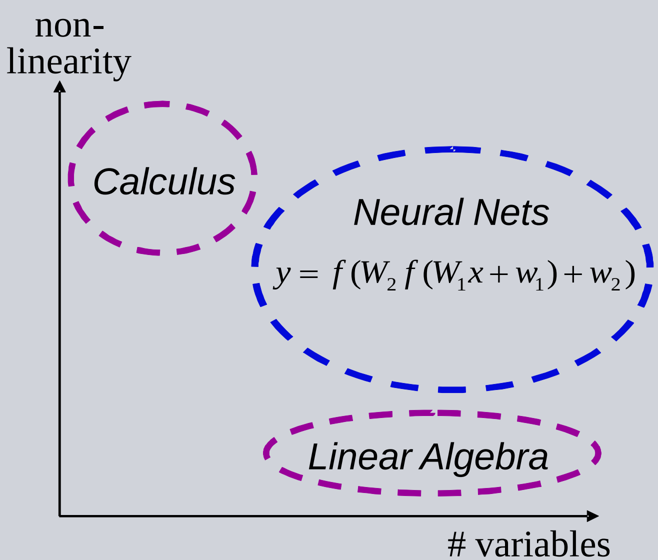
In fact, it shows that biology can mimic computers – but biology might not be limited to this.

Neural networks are universal approximators of high dimensional. nonlinear systems.

Fraunhofer IIS

Mathematical Neural Networks

Complex Systems



Existence Theorem:

(Hornik, Stinchcombe, White 1989)

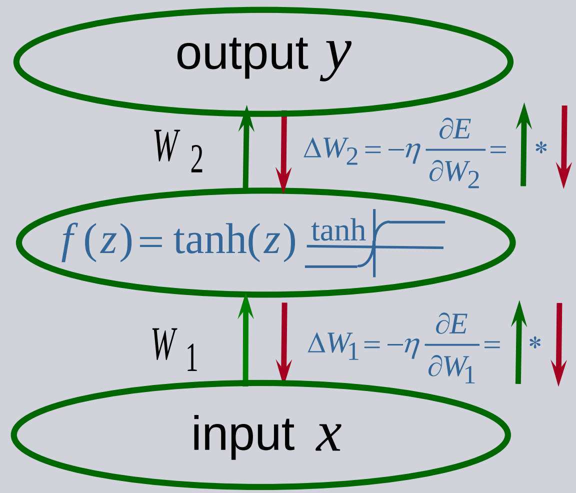
3-layer neural nets $y = W_2 f(W_1 x + w_1) + w_2$ can approximate any continuous function on a compact domain.

Nonlinear Regression

Based on data identify an input-output relation

$$y = W_2 f(W_1 x)$$

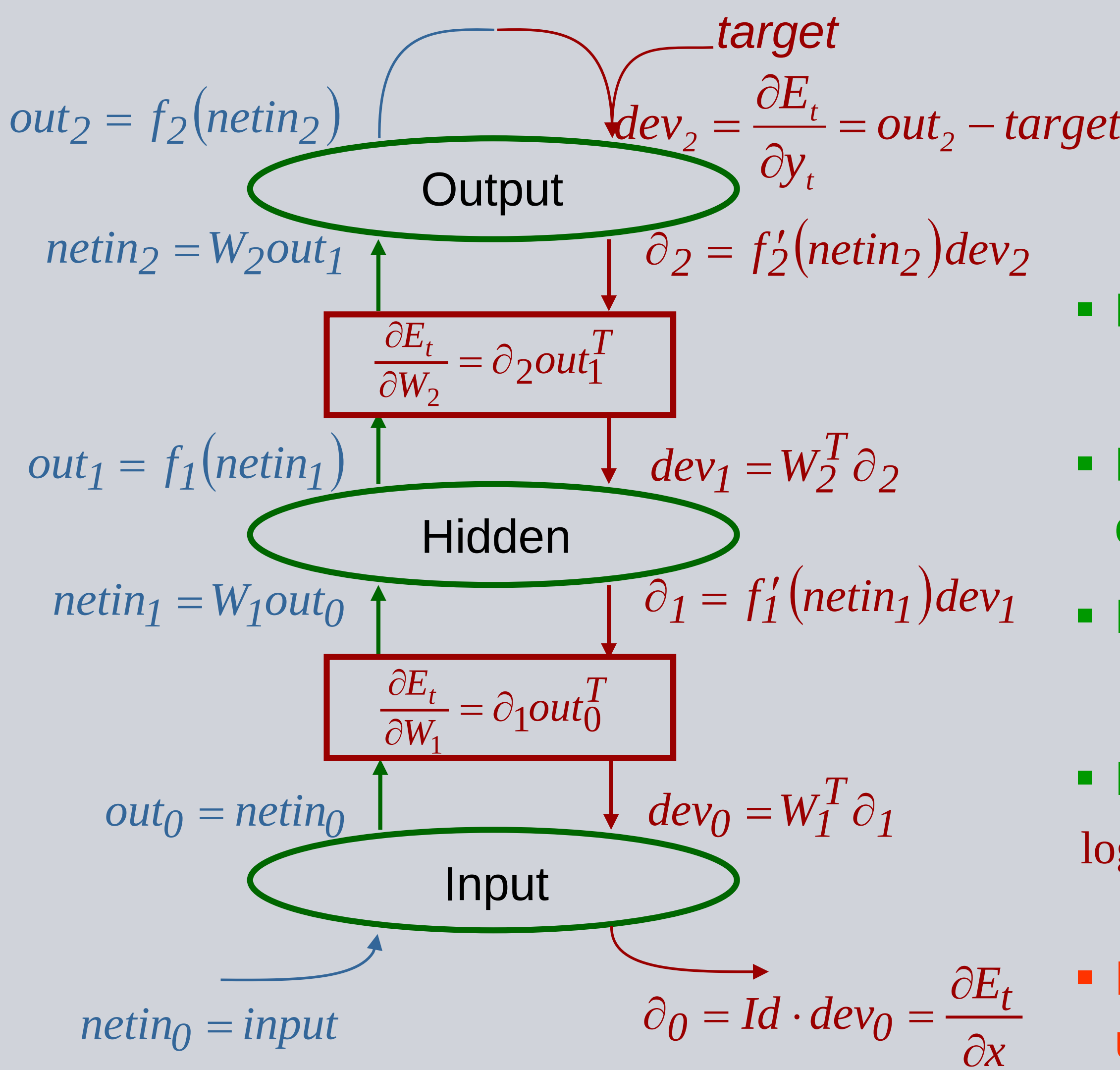
$$E = \sum_{t=1}^{T} \left(y_t - y_t^d\right)^2 \rightarrow \min_{W_1, W_2}$$



Neural networks imply a **Correspondence** of ***Equations***, ***Architectures***, ***Local Algorithms***.

Fraunhofer IIS

Error Backpropagation - Correspondence between Architecture & Algorithm



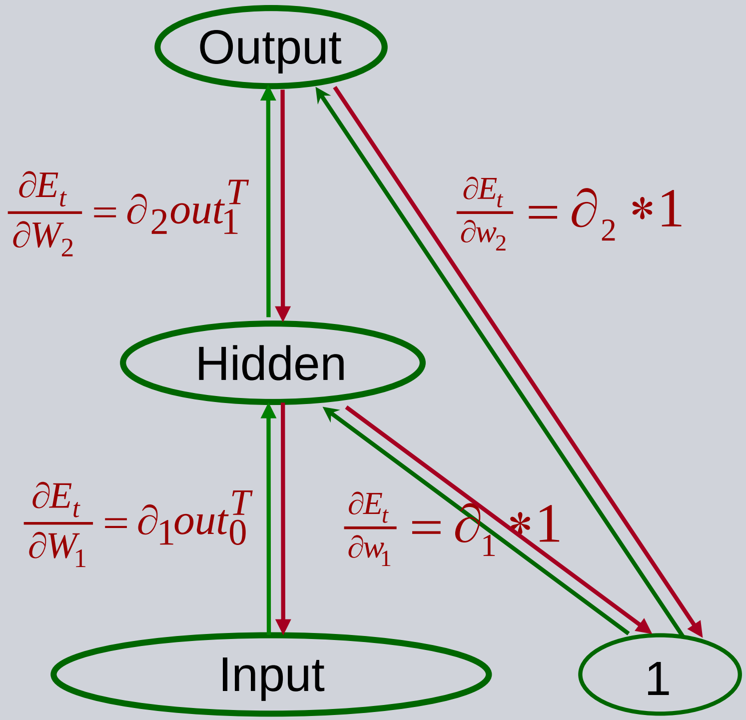
$$E = \frac{1}{T}\sum_{t=1}^{T} E_t = \frac{1}{T}\sum_{t=1}^{T} \tfrac{1}{2}(y_t - y_t^d)^2$$

$$y = f_2(W_2\, f_1(W_1 x))$$

- By the forward & backward flows, $\frac{\partial E_t}{\partial W_1}, \frac{\partial E_t}{\partial W_2}$ are efficiently computed.
- Because of the local algorithm, we can easily extend the network.
- In case of $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ we get
  $\tanh'(netin) = 1 - (\tanh(netin))^2 = 1 - out^2$
- In case of $f(z) = \text{logistic}(z) = \frac{1}{1+e^{-z}}$ we get
  $\text{logistic}'(netin) = \text{logistic}(netin)(1 - \text{logistic}(netin))$
  $= out(1 - out)$
- In case of large/sparse matrices W the use of W^T causes problems.

Fraunhofer IIS

The Re-invention of Thresholds in Neurons



Up to now we have shown algorithms for

$$y = f_2(W_2 \; f_1(W_1 x))$$

instead of

$$y = f_2(W_2 \; f_1(W_1 x + w_1) + w_2)$$

This can be fixed with one additional input = 1 together with the locality of the algorithms.

$$y = f_2(W_2 \; f_1(W_1 x + w_1 * 1) + w_2 * 1)$$

Learning Structure from Data - Learning Rules for Stochastic Search

Task: $E = \frac{1}{T}\sum_{t=1}^{T} E_t = \frac{1}{T}\sum_{t=1}^{T}\left(NN(x_t, w) - y_t^d\right)^2 \rightarrow \min_w$

Notation: $g_t = \frac{\partial E_t}{\partial w}$, $g = \frac{1}{T}\sum_{t=1}^{T} g_t$

Steepest descent learning: $\Delta w = \eta \cdot (-g)$ = step length · search direction

$$E(w + \Delta w) = E(w) + g^T \Delta w + \frac{1}{2}\Delta w^T G \Delta w$$
$$= E(w) - \eta g^T g + \frac{\eta^2}{2} g^T G g < E(w) \quad \textit{for } \eta \textit{ small}$$

Pattern by pattern learning: $\Delta w_t = -\eta g_t = \underbrace{-\eta g}_{\text{steepest descent}} - \underbrace{\eta (g_t - g)}_{\text{stochastic search}}$

Noise on weights act as curvature penalty

$$\langle E(w) \rangle = \frac{1}{T}\sum_t E(w + \Delta w_t) = E(w) + \sum_i \underbrace{\left(\frac{1}{T}\sum_t \Delta w_{it}\right)}_{\approx 0} \frac{\partial E}{\partial w_i} + \frac{1}{2}\sum_i \operatorname{var}(\Delta w_{it}) \frac{\partial^2 E}{\partial w_i^2}$$

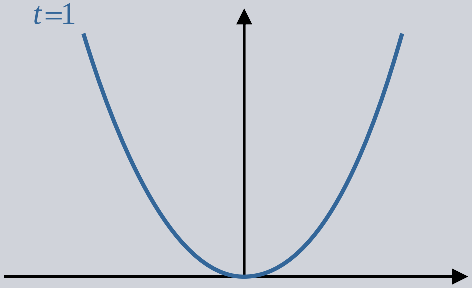
P-by-P Learning $\Delta w_t = -\eta\, g_t$ induces a local penalty on *w:*

$$\langle E(w) \rangle = \frac{1}{T}\sum_t E(w + \Delta w_t) = E(w) + \frac{\eta^2}{2}\sum_i \operatorname{var}(g_{it}) \frac{\partial^2 E}{\partial w_i^2}$$

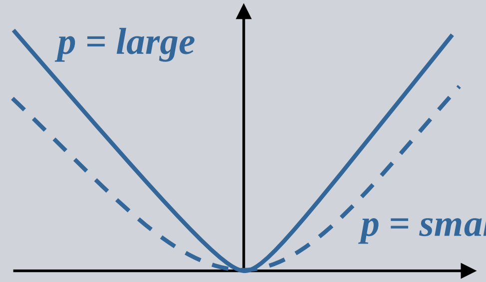
Fraunhofer IIS

Outlier Handling on Targets - Robust Error Functions & Derivatives

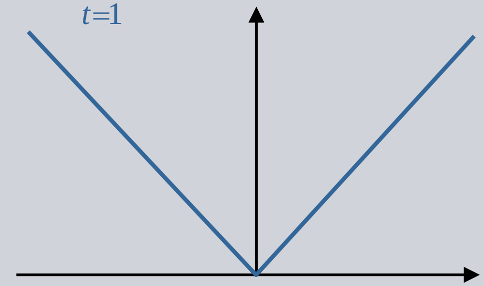
$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{2}(y_t - y_t^d)^2 \rightarrow \min$$



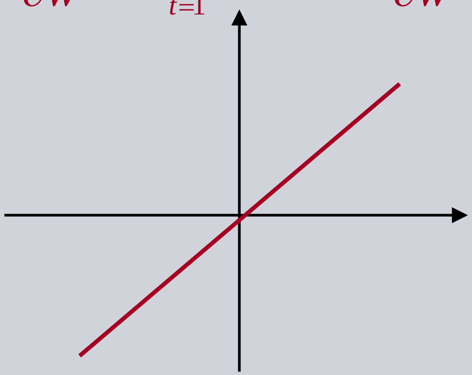
$$\frac{1}{T}\sum_{t=1}^{T}\frac{1}{p}\ln\cosh(p(y_t - y_t^d)) \rightarrow \min$$



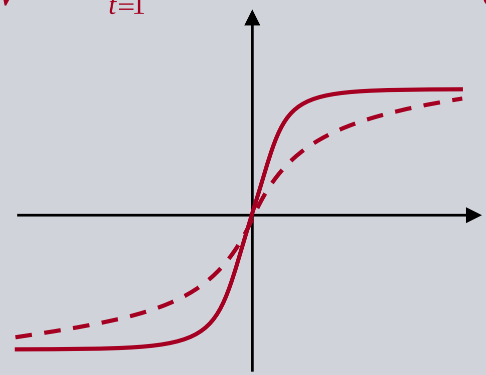
$$\frac{1}{T}\sum_{t=1}^{T}\left\| y_t - y_t^d \right\| \rightarrow \min$$



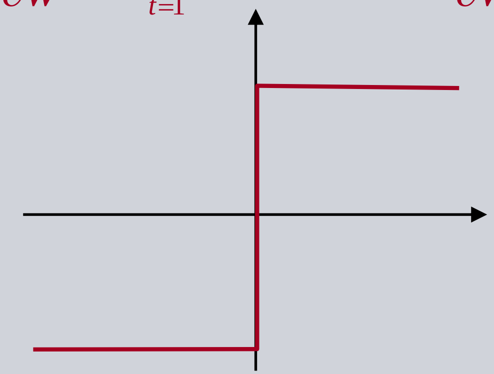
$$\frac{\partial E}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}(y_t - y_t^d)\frac{\partial y_t}{\partial w}$$



$$\frac{\partial E}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}\tanh(p(y_t - y_t^d))\frac{\partial y_t}{\partial w}$$



$$\frac{\partial E}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}sign(y_t - y_t^d)\frac{\partial y_t}{\partial w}$$



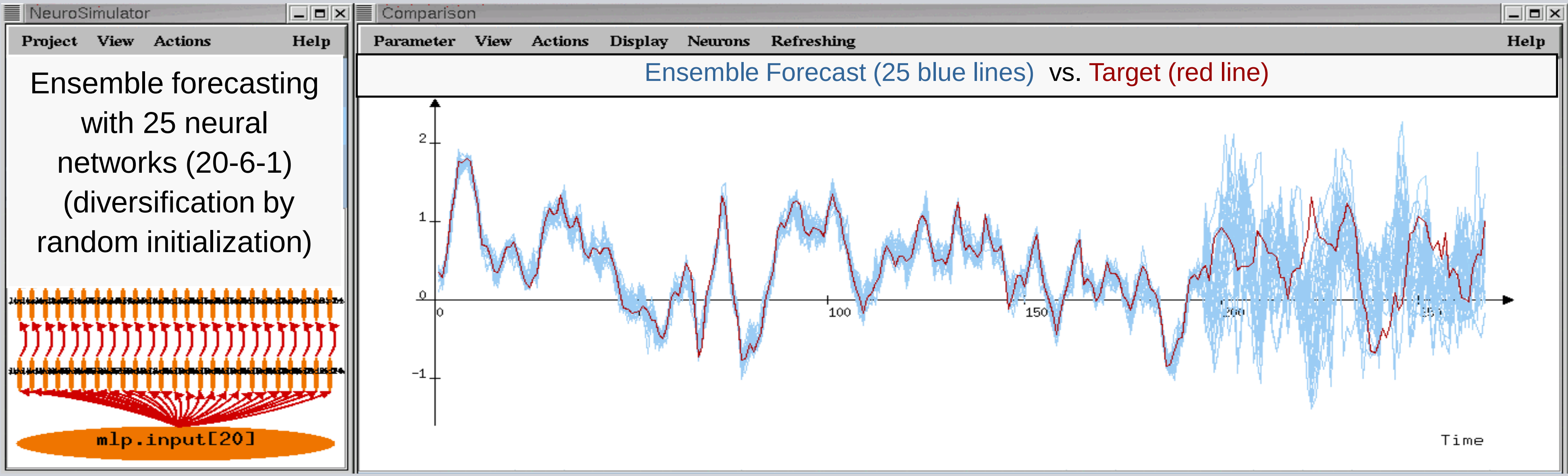
- easy to use derivative
- large impact of outliers

- easy to use derivative
- no impact of outliers (robust)

- complicated derivative
- no impact of outliers (robust)

Fraunhofer IIS

Local Minima versus Overparameterization in Nonlinear Regression



Observation: The experiment with 25 parallel networks shows a very diverse generalization.

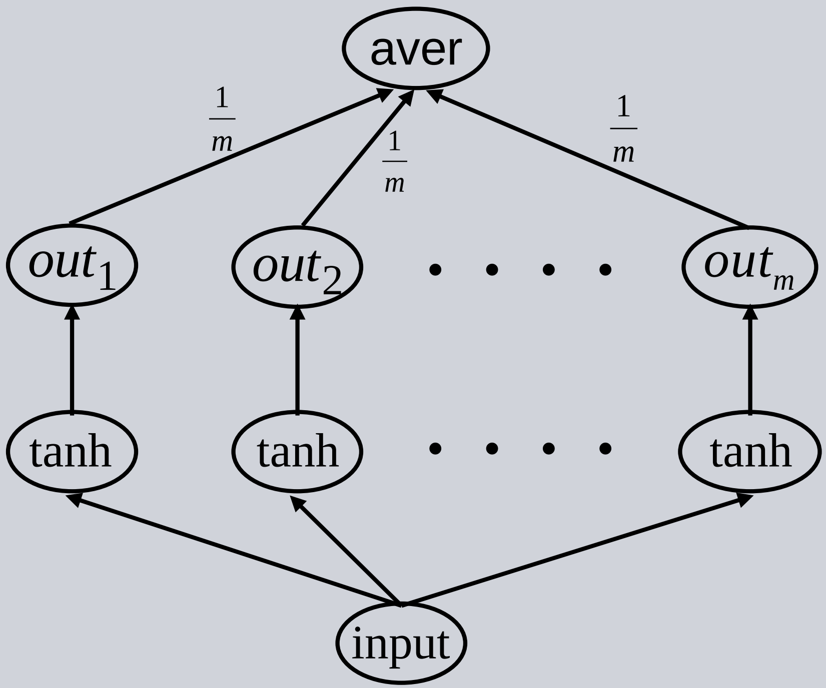
Explanation: (1) could be caused by convergence of nonlinear models to different local minima

(2) could be caused by non-unique solutions dependent on random initialization.

Dilemma: Parsimonious models are prone to type (1) problems, over parameterized models are prone to type (2) difficulties – but have less trouble with (1) !!

Fraunhofer IIS

Decreasing Model Uncertainty by Averaging



The sub-networks learn different solutions of the same task. In case of large averages (m > 20) an equal weighting is superior, in case of small averages it is superior to freeze the subnets and optimize the weighting factors.

$$\boxed{E_{aver}} = \frac{1}{T}\sum_T \left[out_{aver} - tar\right]^2$$

$$= \frac{1}{T}\sum_T \left[\left(\frac{1}{m}\sum_i out_i\right) - tar\right]^2$$

$$= \frac{1}{T}\sum_T \left[\frac{1}{m}\sum_i \left(out_i - tar\right)\right]^2$$

$$\overset{!}{=} \frac{1}{m^2}\frac{1}{T}\sum_T \sum_i \left(out_i - tar\right)^2$$

$$= \frac{1}{m}\frac{1}{m}\sum_i \frac{1}{T}\sum_T \left(out_i - tar\right)^2$$

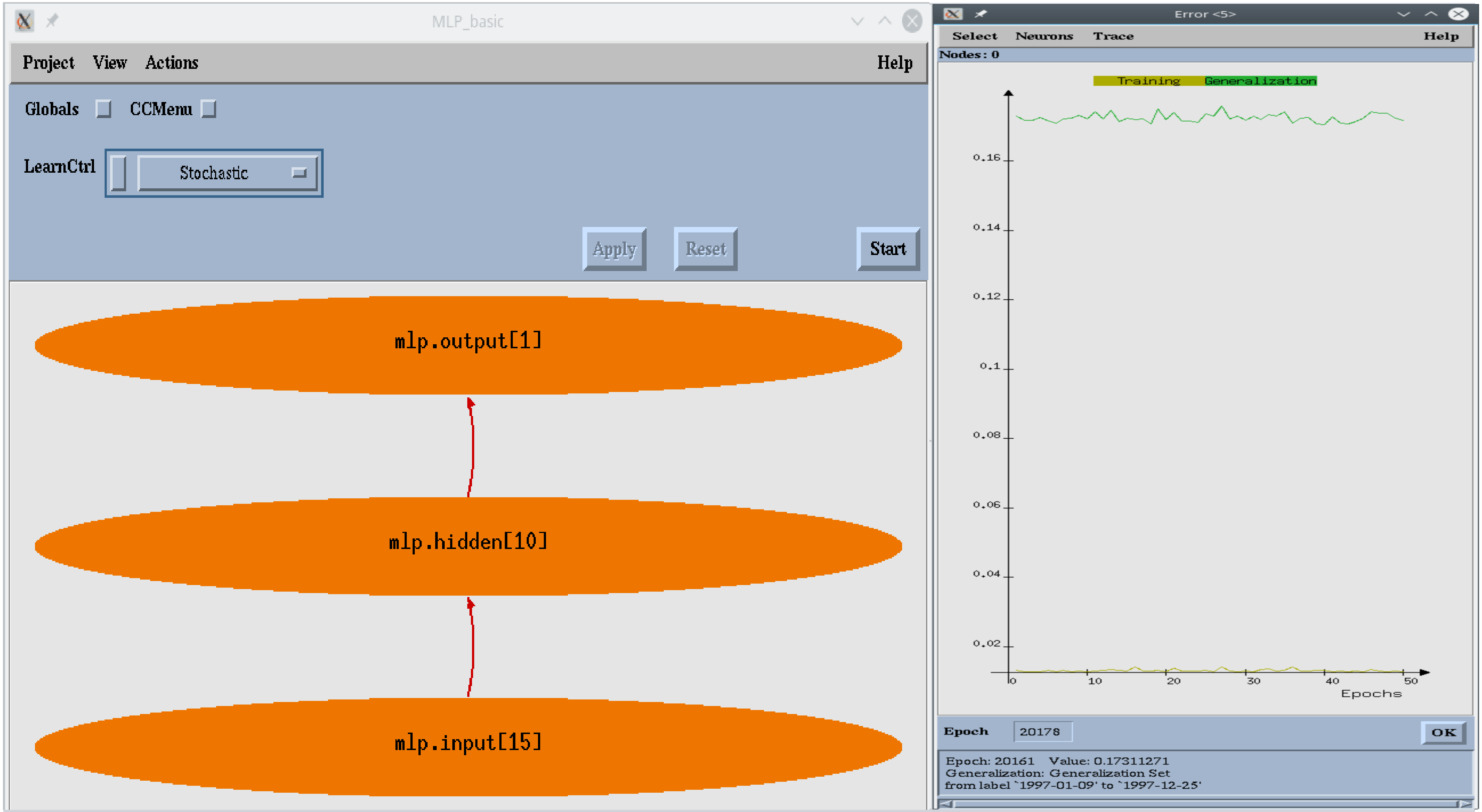
$$= \boxed{\frac{1}{m}\, aver\left(E_i\right)}$$

$$! \quad \frac{1}{T}\sum_T \left(out_i - tar\right)\cdot\left(out_j - tar\right) = 0 \quad \forall\, i \neq j$$

(covariance of the errors of the submodels)

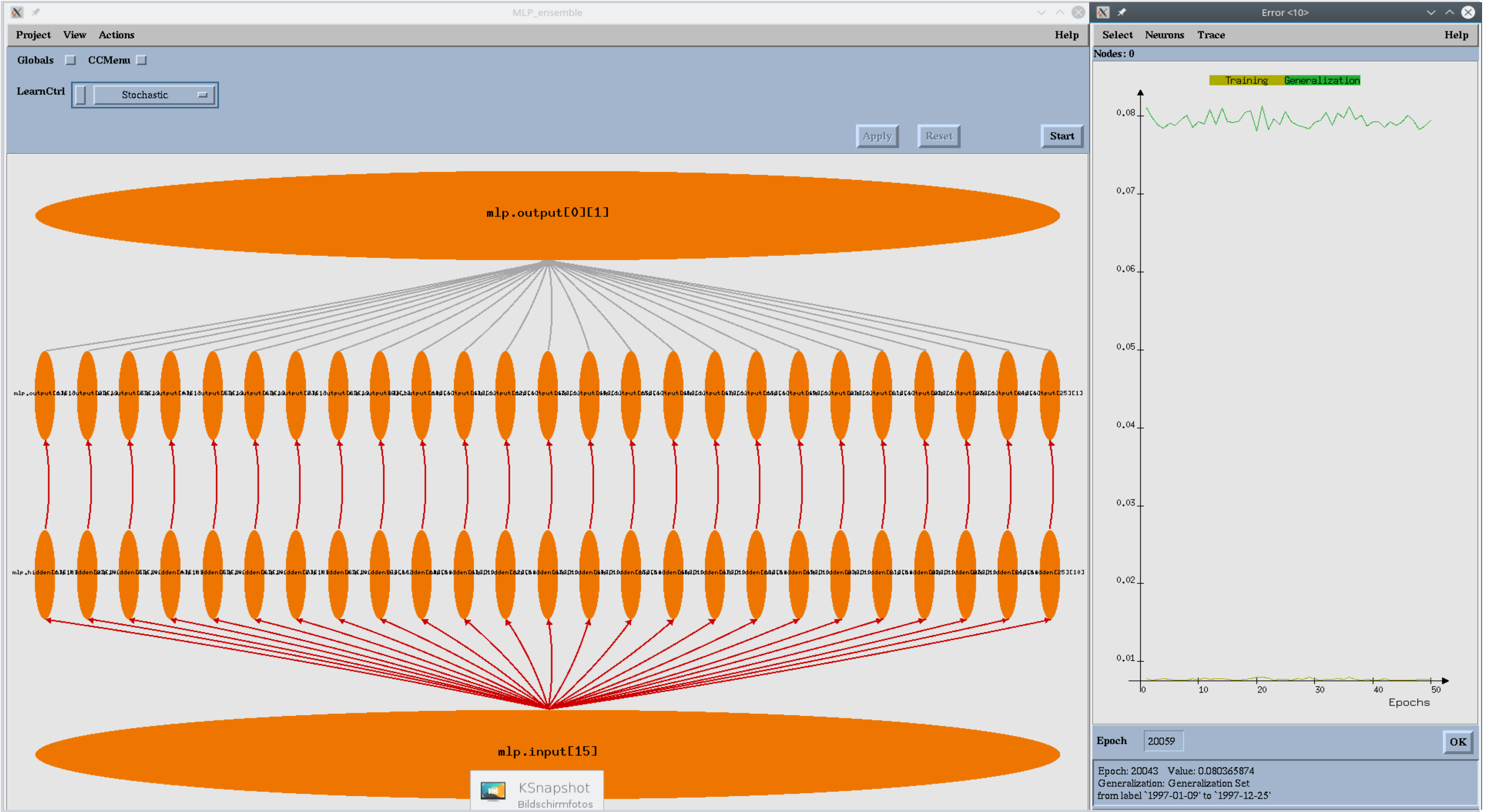
Fraunhofer IIS

Shallow Neural Networks



Fraunhofer IIS

04_MLP_ensemble



Fraunhofer IIS

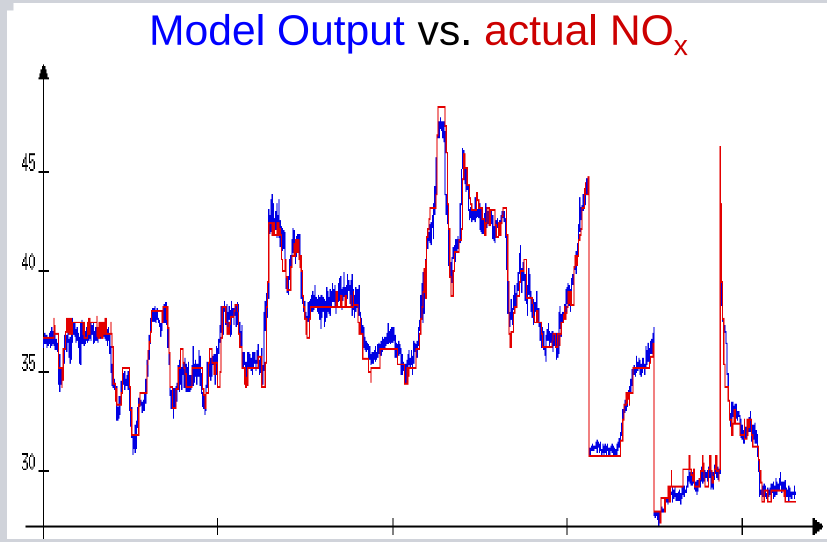
Neural Networks are No Black Boxes (Siemens)

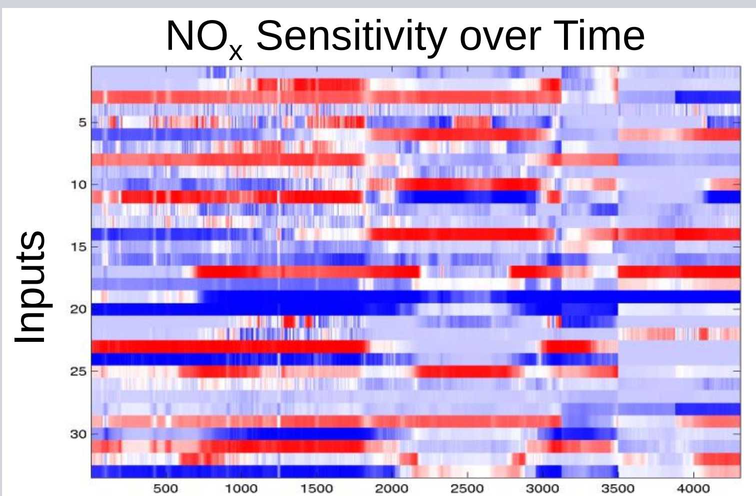
Application: Modeling of a Gas Turbine

- Inputs: 35 sensor measures and control variables of the turbine
- Output: NO_x emission of the gas turbine

Sensitivity Analysis: Compute the first derivatives along the time series:

$$\frac{\partial\, output}{\partial\, input_i} > 0 \qquad \frac{\partial\, output}{\partial\, input_i} < 0$$



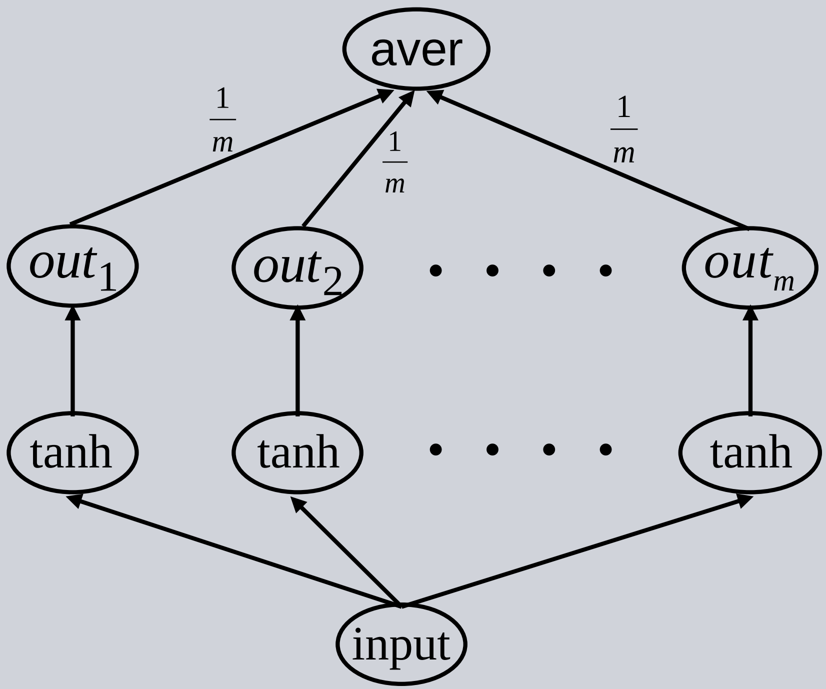


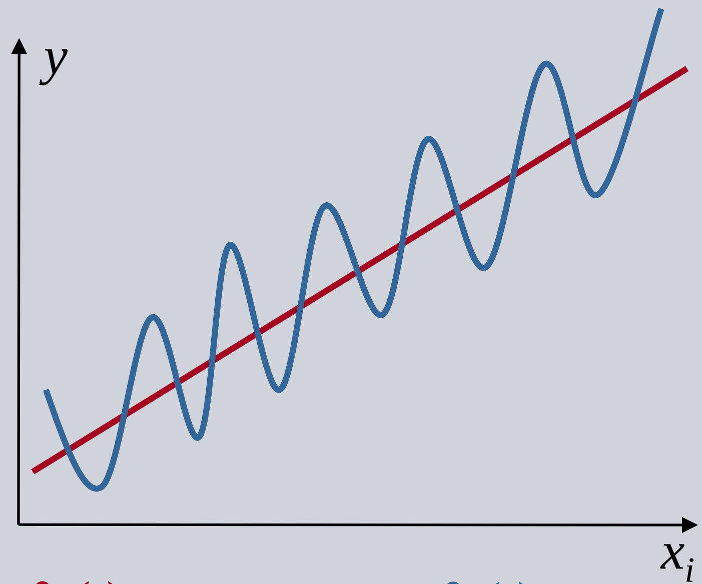
A classification of input-output sensitivities:

- ***linear relationship*** (= constant first derivative)
- ***monotone*** (input can be used in 1dim. control)
- ***non-monotone*** (only multi-dim control possible)
- ***~ zero*** (input useless in modeling and control)

Fraunhofer IIS

Identification of Monotonic Functions with Ensemble Neural Networks





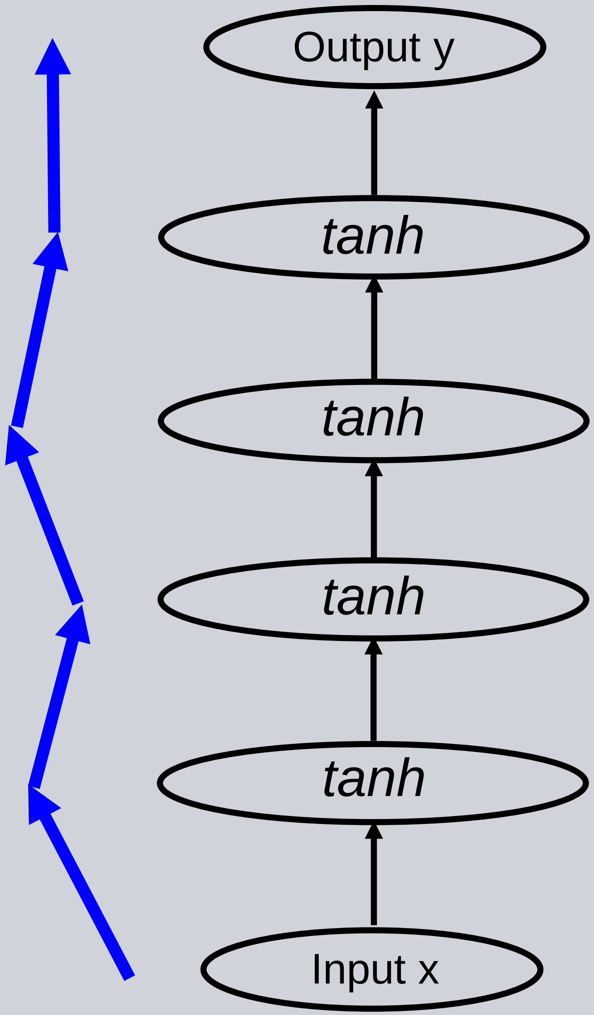
$$\frac{\partial y(t)}{\partial x_i} \geq 0 \quad \text{but} \quad \frac{\partial y(t)}{\partial x_i} <> ?$$

Ensemble networks are an efficient way to detect a monotonic input-output relation.

Assume, that all sub-models are good approximations of a monotonic target line. Even then, the individual sub-models might meander around this target in a non-monotonic way. Ensemble averaging smoothes out valleys and hills.

Fraunhofer IIS

Why Deep Feedforward Neural Networks



The learning of deep neural networks is difficult, because…

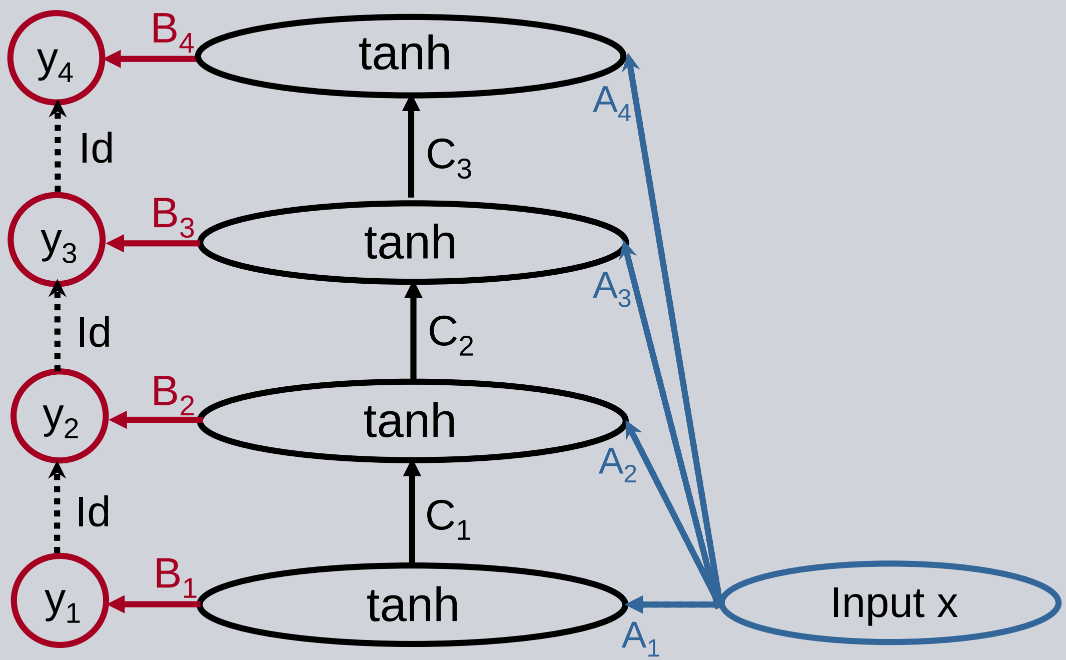
- first layers can do nonsense, while the last hidden layer corrects everything – why to use a deep structure?
- *forward path:* relevant input information may get lost in the hierarchy of hidden layers,
- *backward path:* the error signal decays passing through many hidden layers.

Challenge: Define explicit tasks for the intermediate layers!

+ to allow a sequential computation of very complicated input-output relations,
+ to do a dimensionality reduction for very large input layers,
+ to exploit neighboring relationships in the inputs (images, spectra, time series).

Error Correction Learning in Deep Feedforward Neural Networks

This architecture can be scaled to very deep topologies.



For a qualitative analysis use sensitivities $\frac{\partial y_4}{\partial x_i}$

Level1 acts as a standard feedforward neural network.

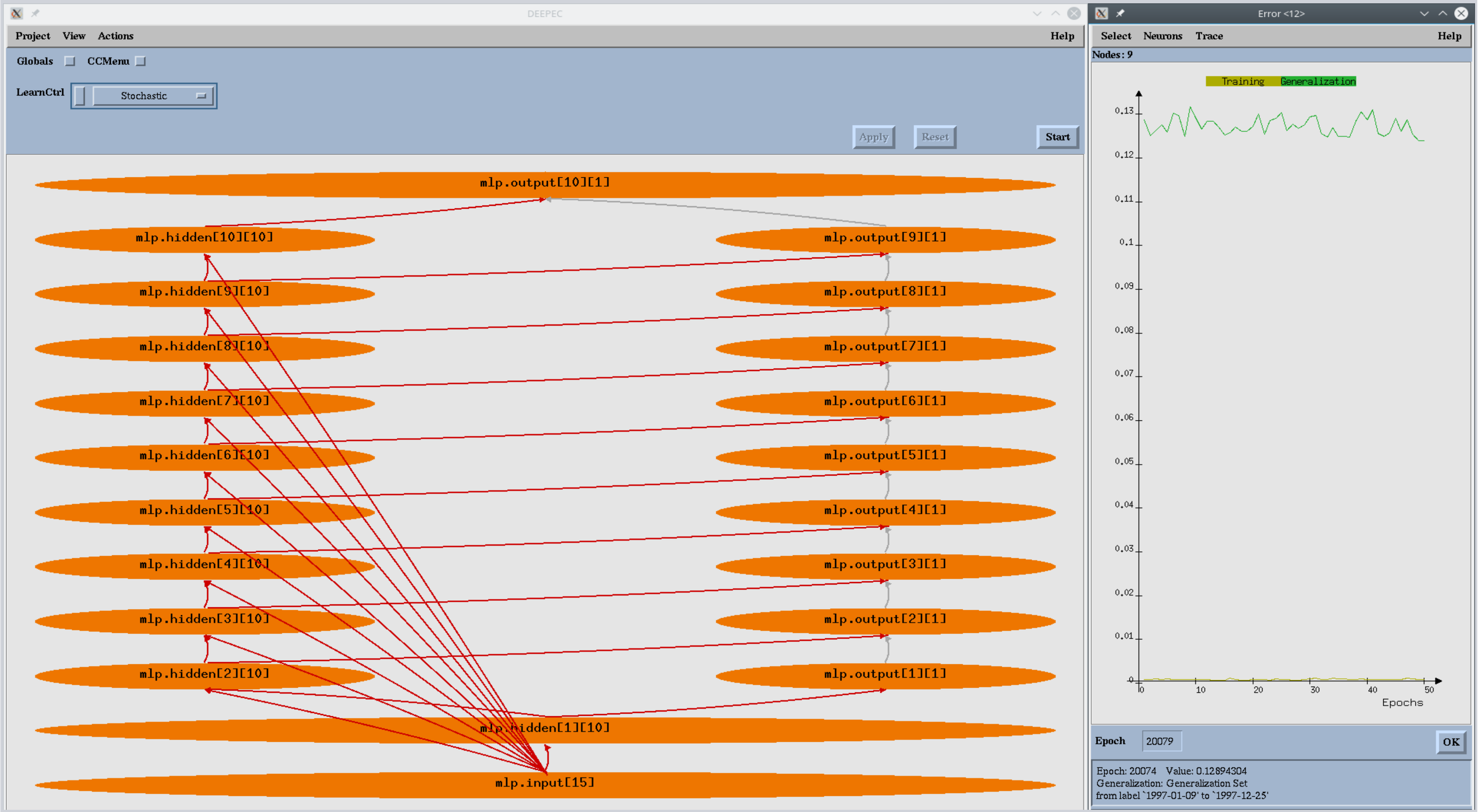
From Level2 on, all levels have to learn only the residual error from the levels below.

········➝ *forward only connections*

The correction of the residual errors may act on all inputs or can focus on a subset of the inputs.

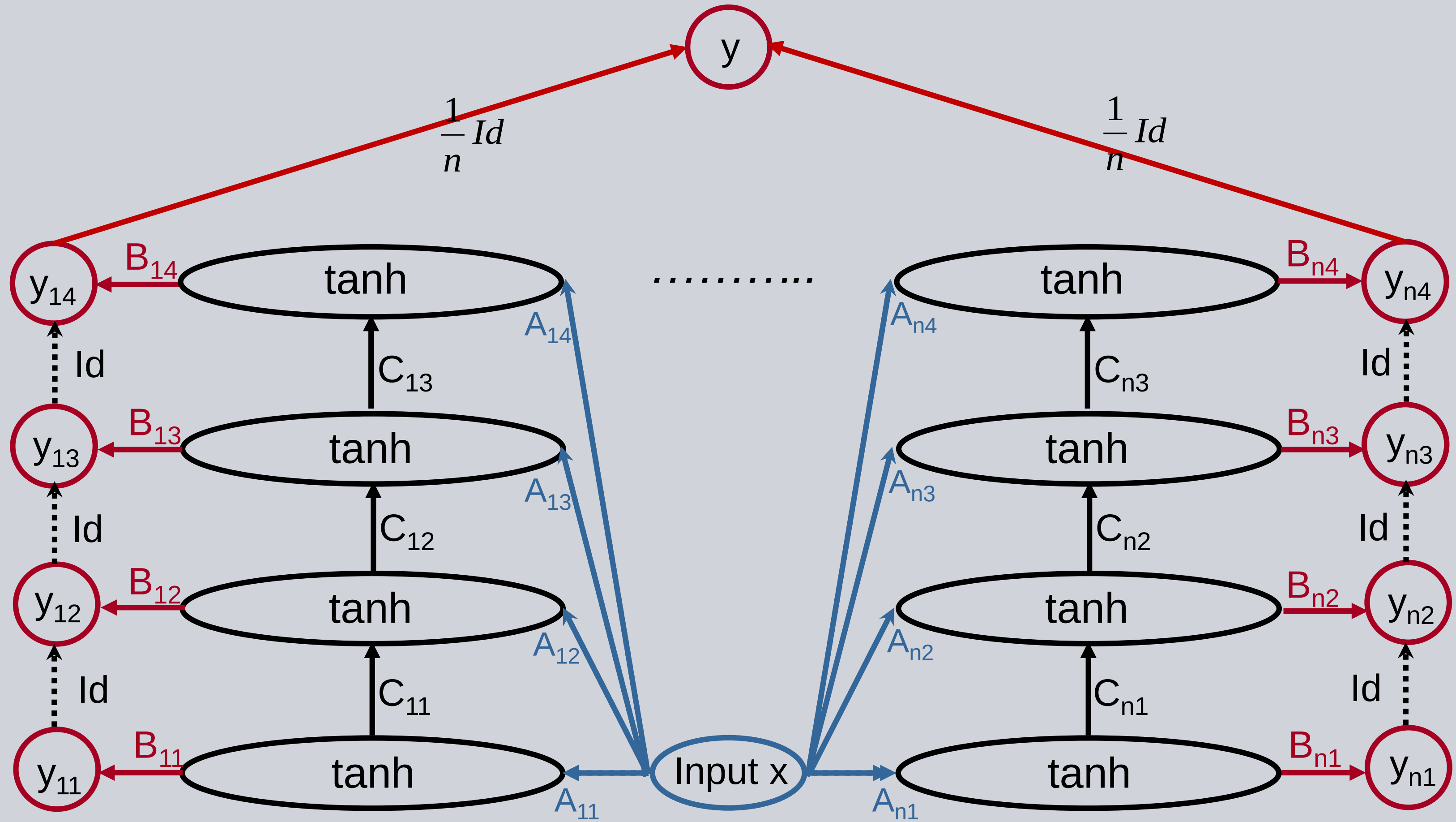
As before all output neurons need an output – target error function.

08_DEEPEC (DEEP including Error Correction)

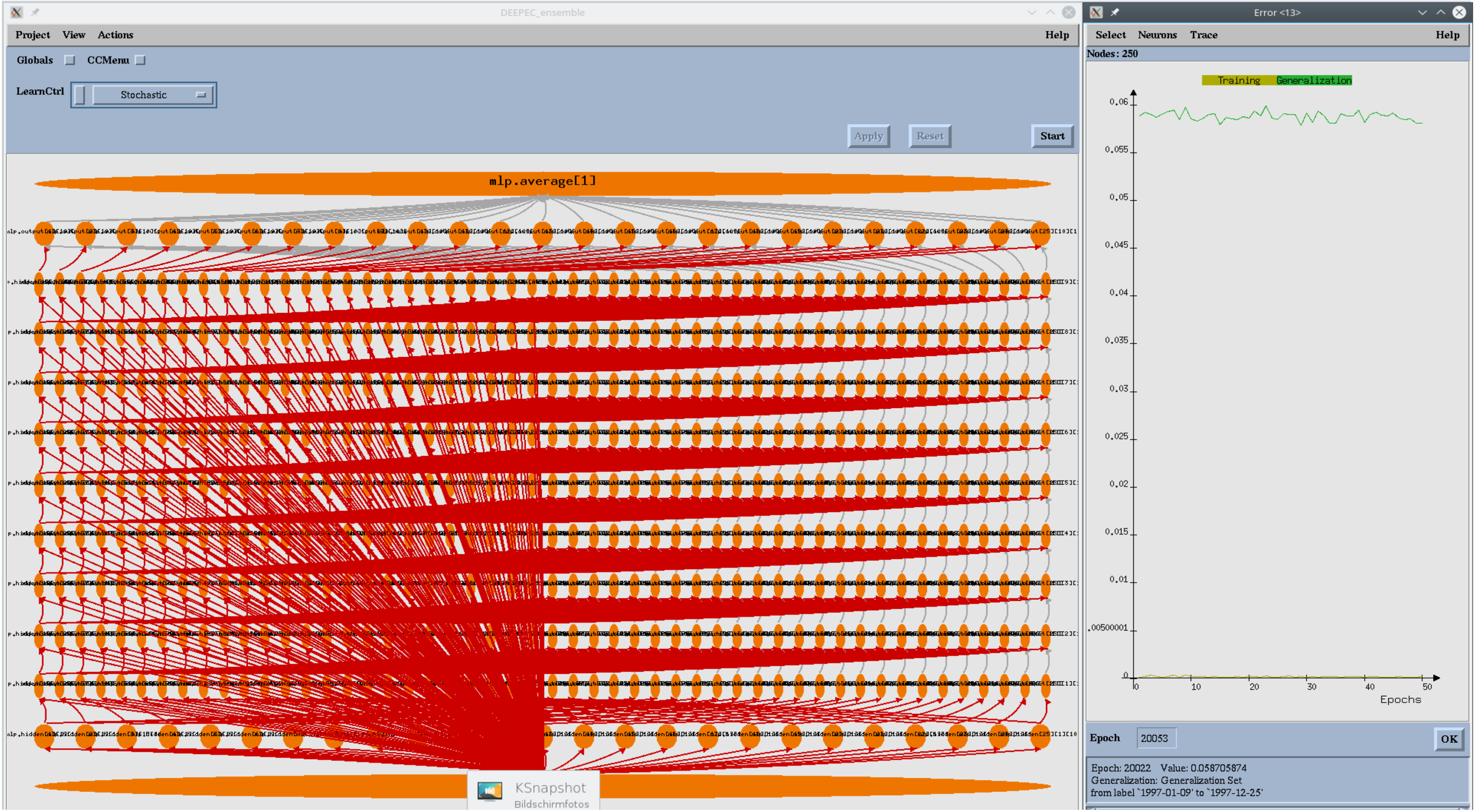


Fraunhofer IIS

Ensemble Deep Feedforward Neural Networks



09_DEEPEC_ensemble



Probability Classification with Neural Networks

n-class classification

$$\mathrm{target}_k \in \{0,1\}, \quad \sum_{k=1}^{n} \mathrm{target}_k = 1$$

$$output_k \in (0,1), \quad output_k \geq 0\,, \quad \sum_{k=1}^{n} output_k = 1$$

Error measurement on the test data:

$$\mathrm{p}(out_k \mid tar_k) = \frac{\sum_{t \in testset} out_{t,k} \cdot tar_{t,k}}{\sum_{t \in testset} tar_{t,k}} \in (0,1)$$

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{n} \mathrm{target}_{t,k} \ln(\mathrm{output}_{t,k}) \rightarrow \min$$

$$output_k = \frac{e^{netin_k}}{\sum_j e^{netin_j}}$$

$$\partial = \mathrm{out} - \mathrm{tar}$$

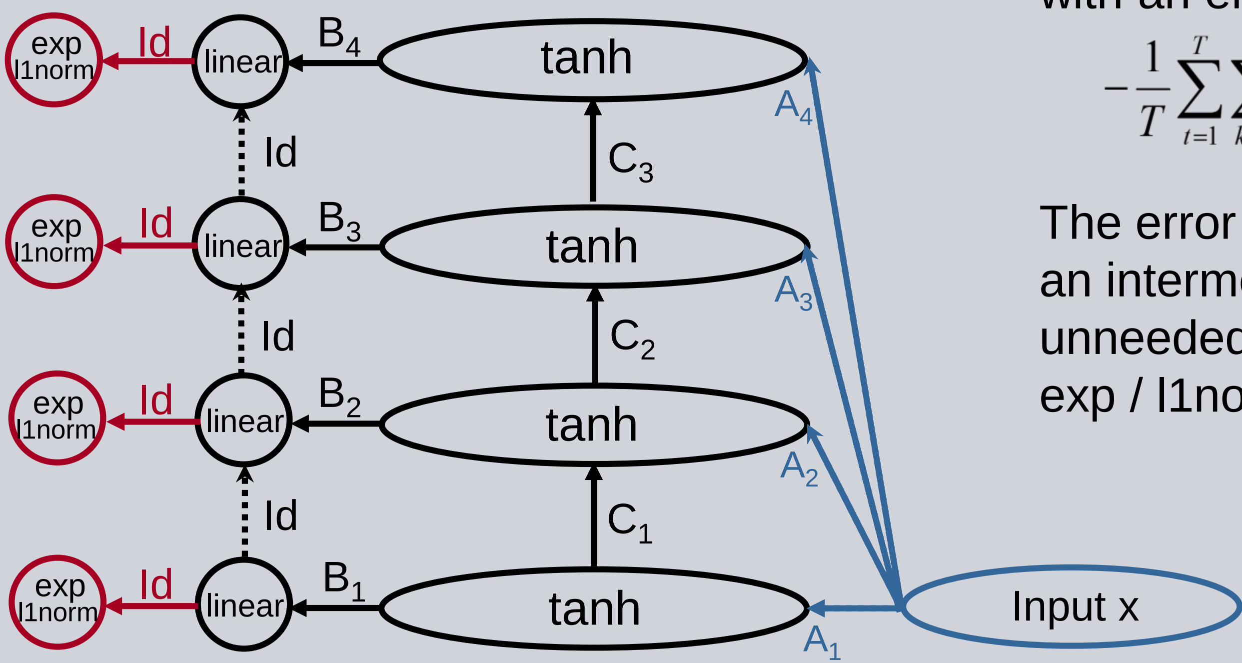
Neural Network

The combination of exp-nonlinearity, the softmax operator (L1-norm) and the entropy target function not only fulfill the output conditions, but result in an efficient computation of the learning gradients.

$$\frac{\partial output_k}{\partial netin_j} = output_k\left(1_{kj} - output_j\right), \quad \frac{\partial \ln\left(output_k\right)}{\partial netin_j} = \left(1_{kj} - output_j\right)$$

$$\frac{\partial E_t}{\partial netin_j} = -\sum_{k=1}^{n} \mathrm{target}_k \frac{\partial \ln(\mathrm{output}_k)}{\partial netin_j} = \mathrm{output}_j - \mathrm{target}_j$$

Classification in Deep Feedforward Neural Networks



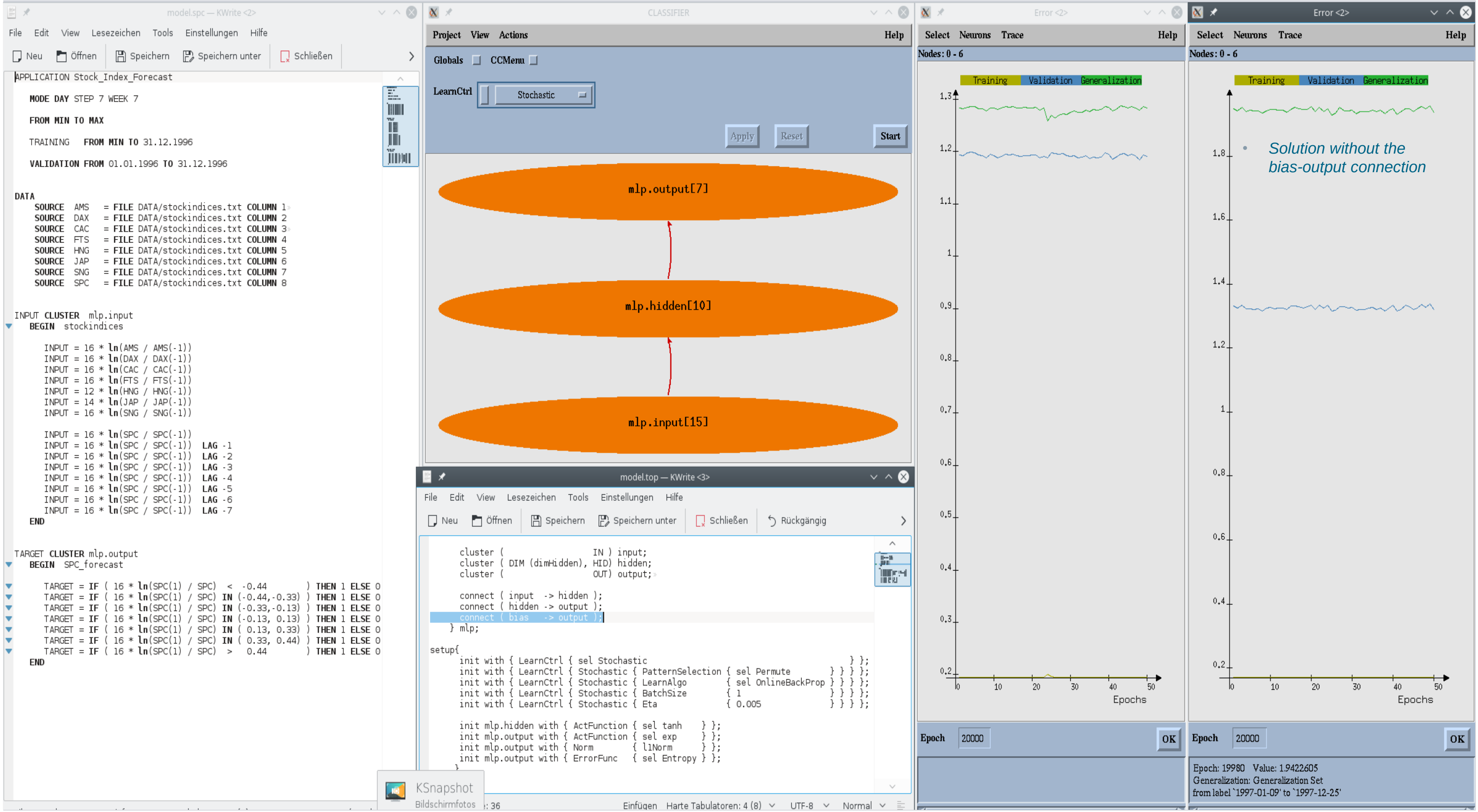
The left output clusters should be trained with an entropy error function.

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{n} \text{target}_{t,k} \ln(\text{output}_{t,k}) \rightarrow \min$$

The error correction should be done in an intermediate linear step to avoid an unneeded complexity with iterated exp / l1norm operations.

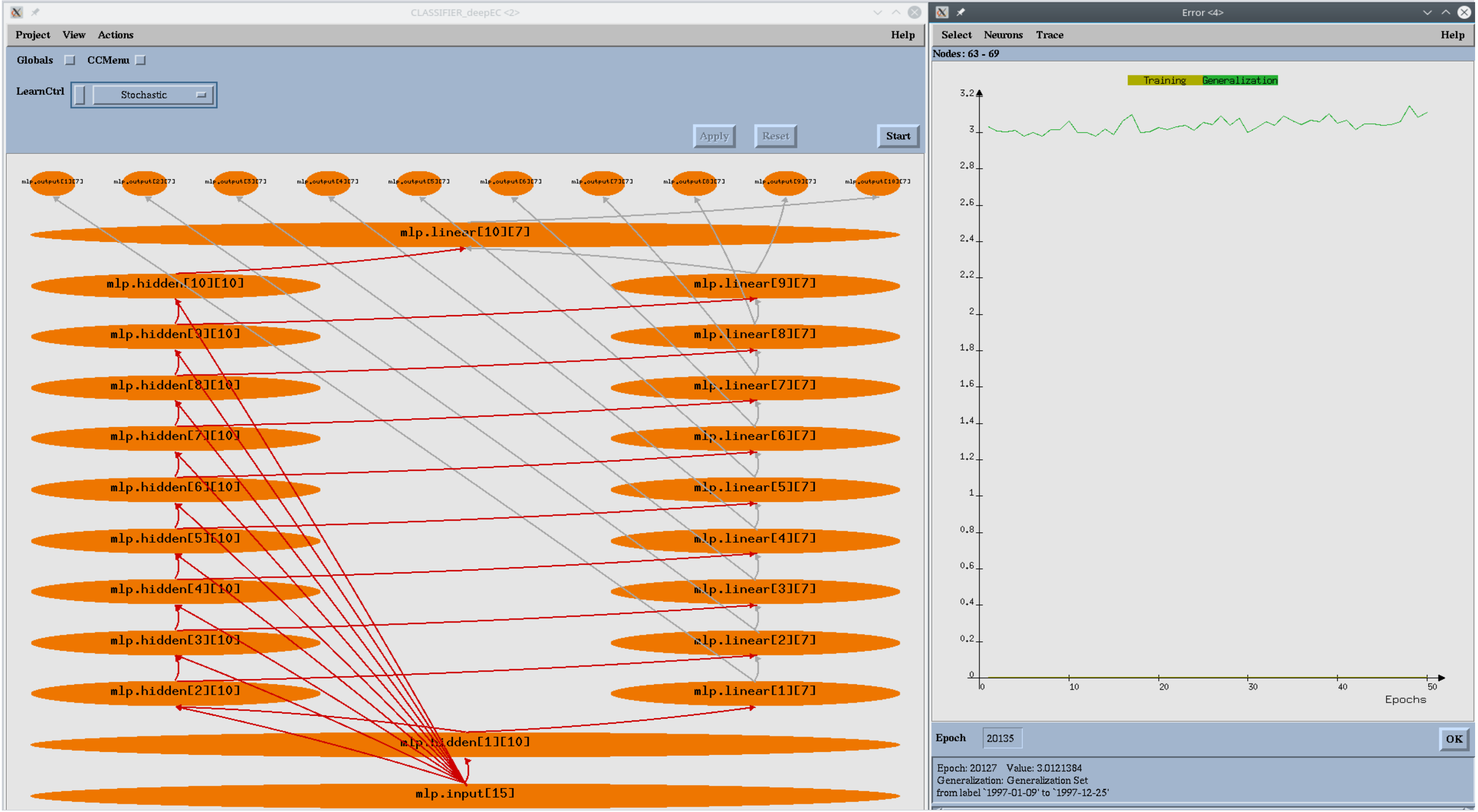
Fraunhofer IIS

10_CLASSIFIER



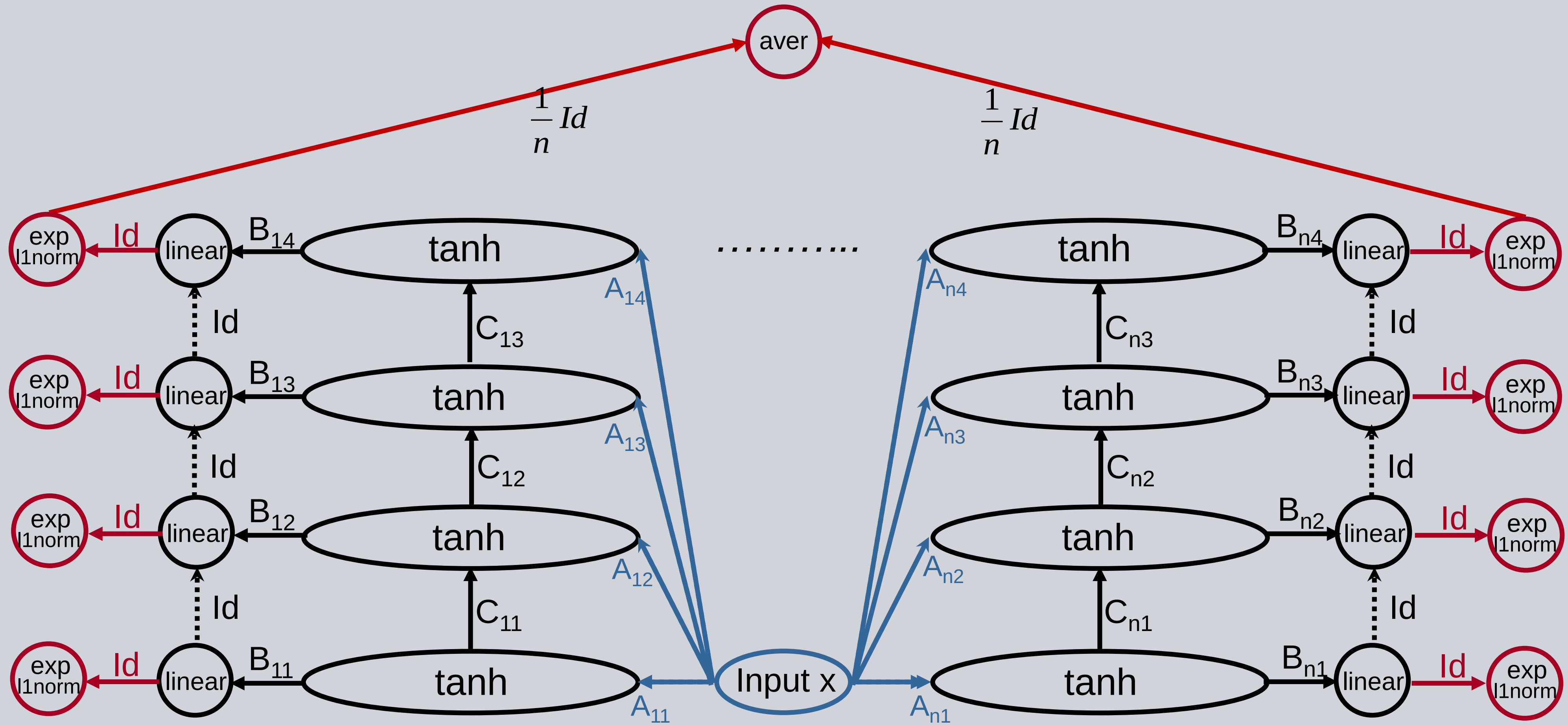
Fraunhofer IIS

11_CLASSIFIER_deepEC



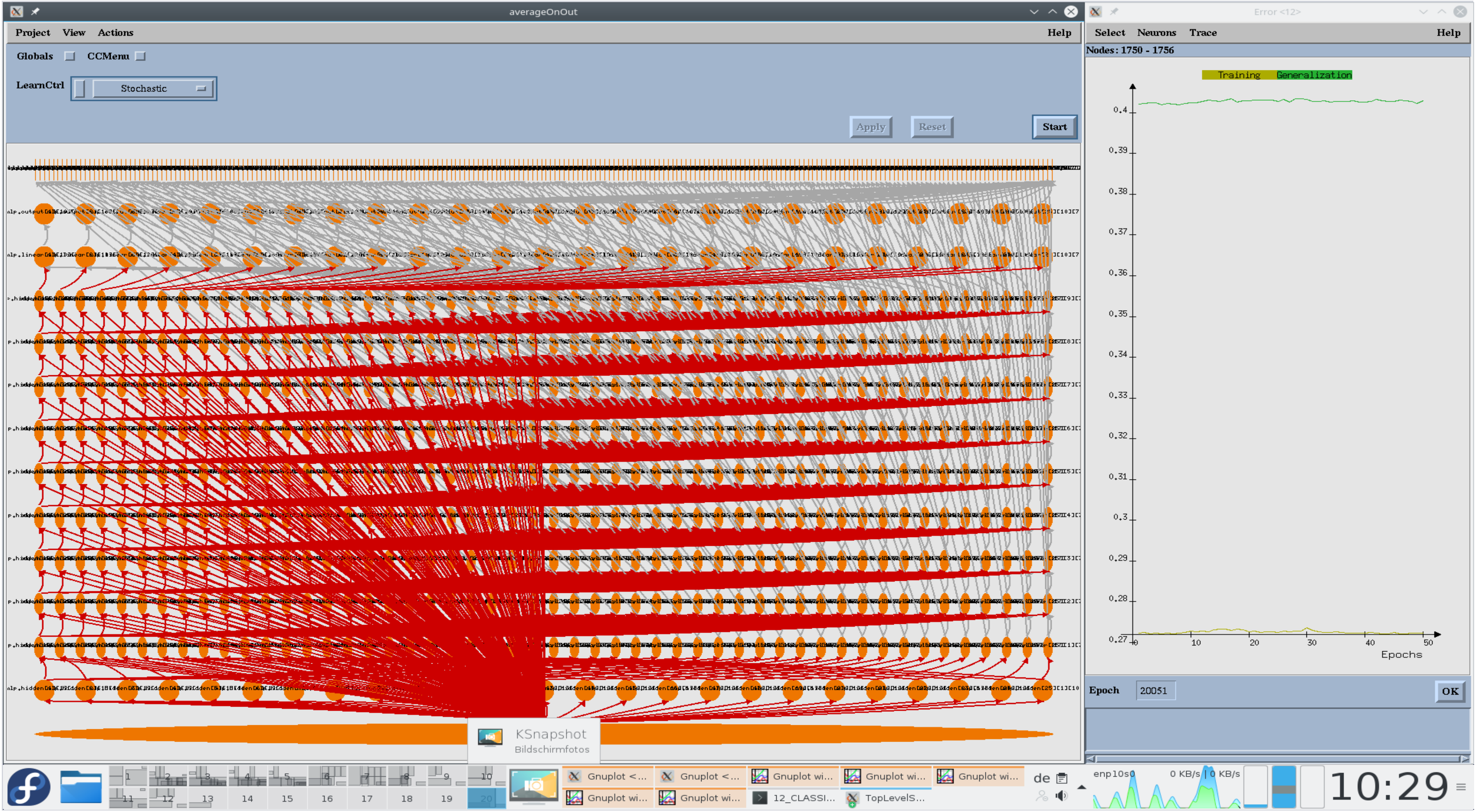
Fraunhofer IIS

Classification with Ensemble Deep Feedforward Neural Networks



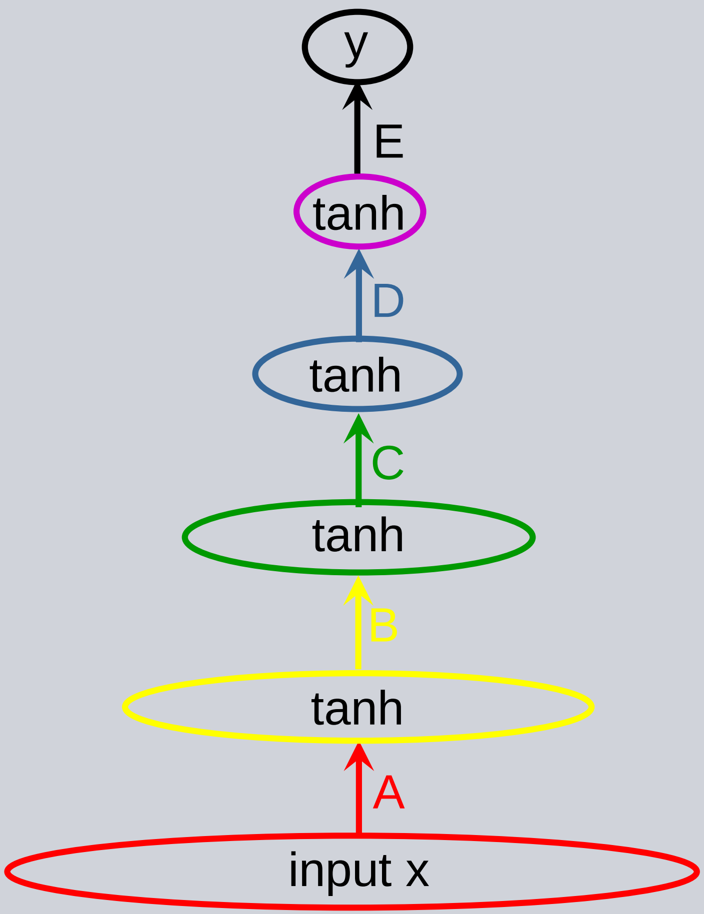
The final averaging has to be done on the linear clusters and followed by a exp / l1norm transformation. This is a guaranty of a correct classifier at the final output.

CLASSIFIER_deepEC_ensemble



Fraunhofer IIS

Deep Feedforward Neural Networks: Hierarchy of Unsupervised Learning



Design a deep feedforward neural network to realize a sequence of increasing abstract features. Finally, the output (regression or classification) is computed.

Greedy learning from bottom to top:

Connectors A, B, C, D have to be backward false (there is no backward error flow though the net).

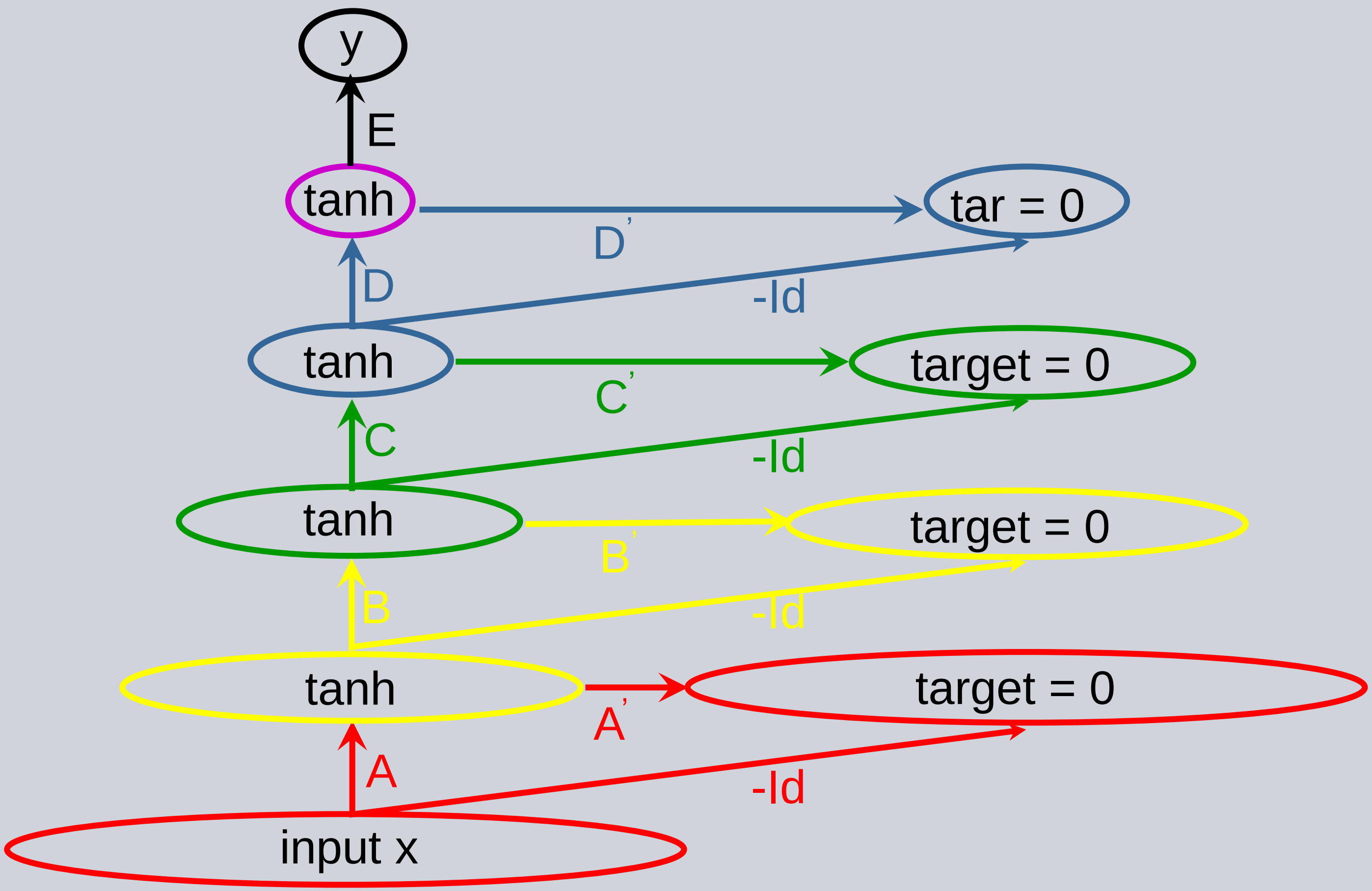
At start, apply an unsupervised learning rule only to matrix A.

Repeat this procedure level by level from bottom to top.

Finally learn the output y with supervised learning.

For references see also the work of the groups of Yoshua Bengio and Geoffrey Hinton from 2006 on

Deep Feedforward Neural Networks: A Hierarchy of Auto-Encoders



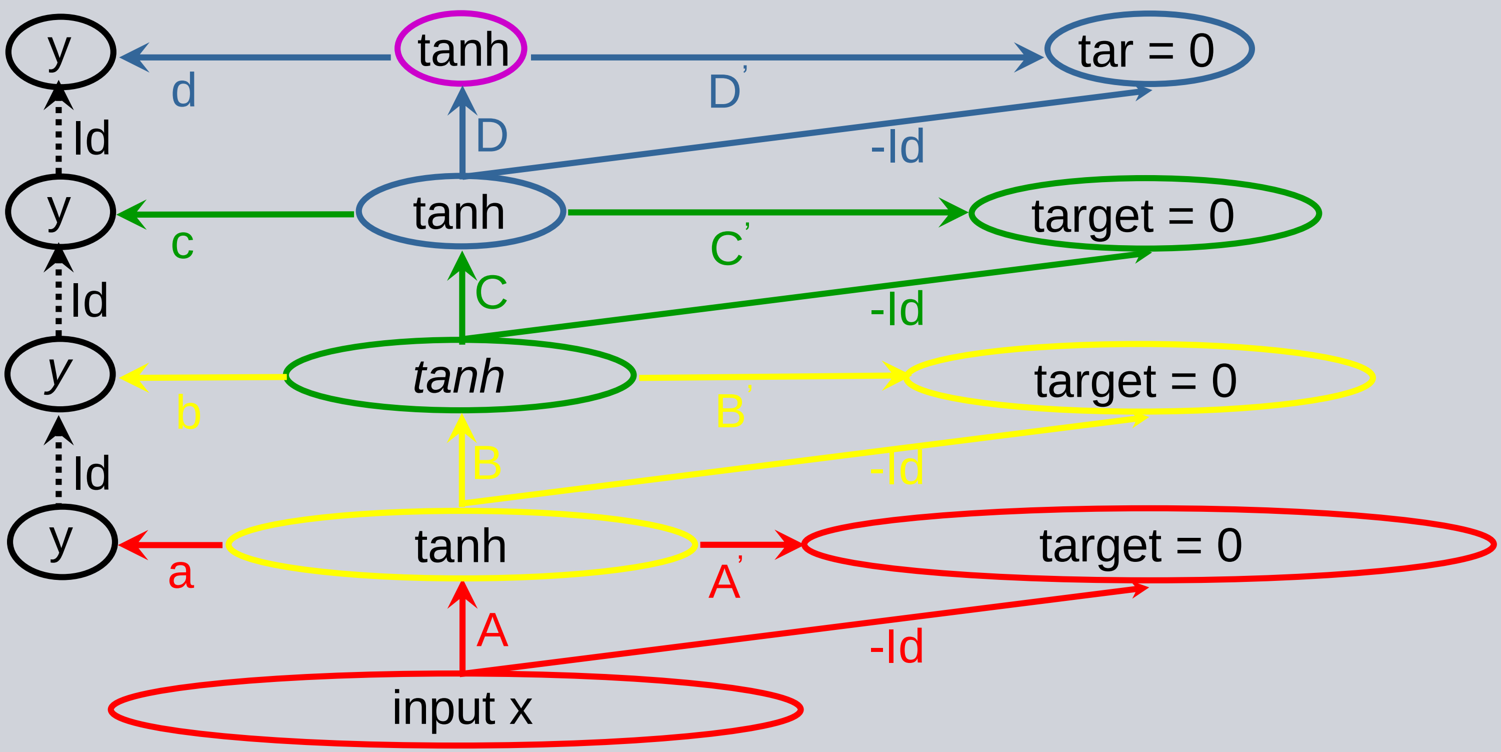
The stacked auto-encoder squeeze out redundant input information – with a focus on the final target.

In a final step we need a general regression / classifier to evaluate the task.

Do the learning level by level from bottom to top.

Instead of (A, A´)…, one can use pairs of (A, A^T)..

Don’t Forget Your Focus of Interest in the Compression Procedure

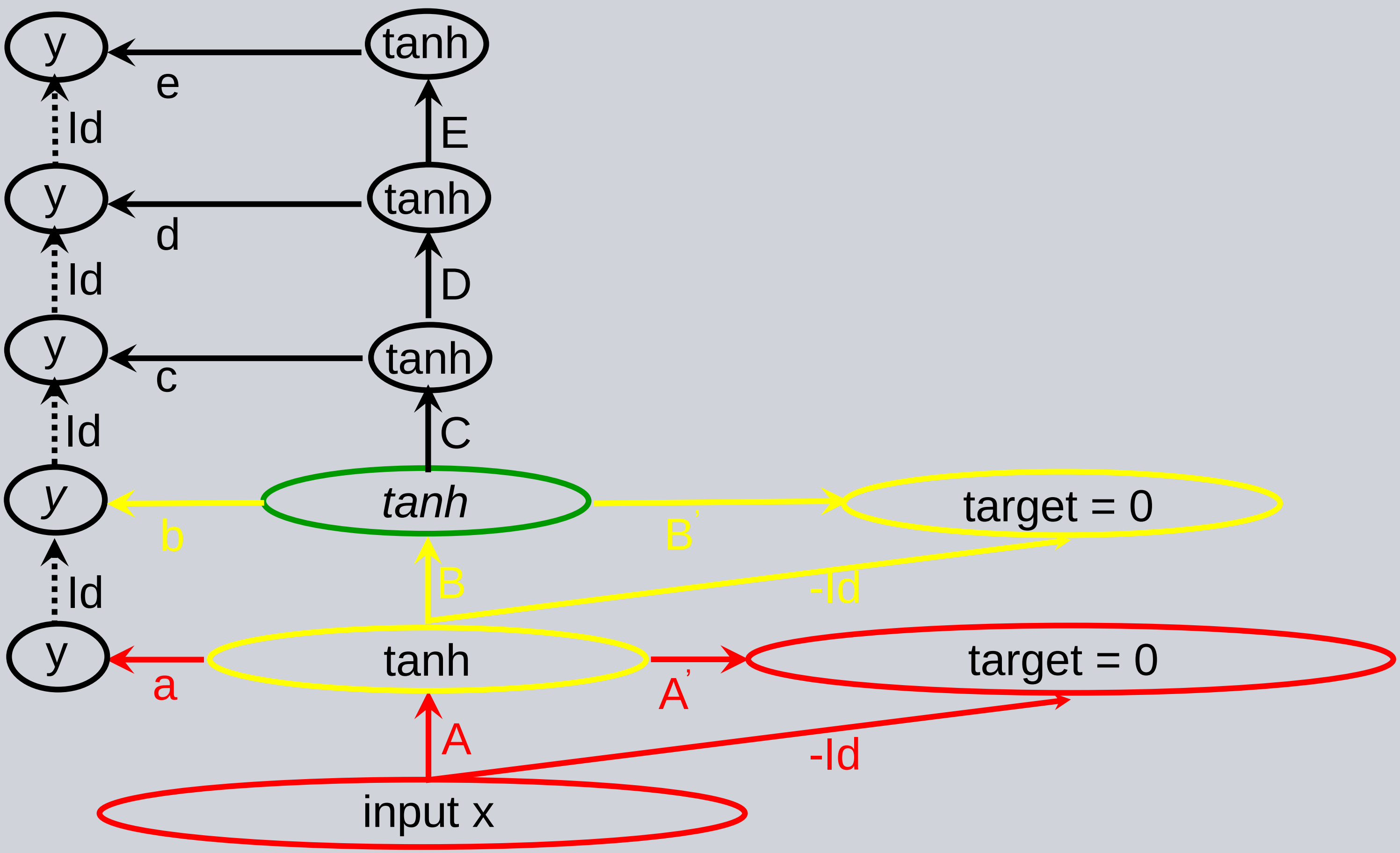


The input might have a simple main structure but the important information is coded in non- obvious parts of the input (e.g. analysis of spectra).

The hierarchy should not only focus on the compression task but also on the final target.

The embedding of clustering should be done with an additional linear branch.

Sequential Combination of Deep Compression and Deep Regression



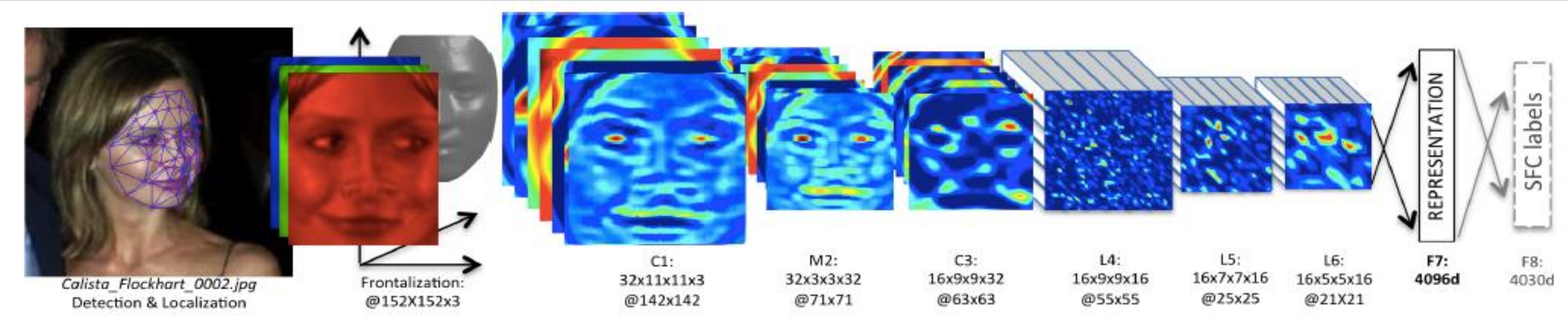
The input might have a simple main structure but the important information is coded in non- obvious parts of the input (e.g. analysis of spectra).

The hierarchy should not only focus on the compression task but also on the final target.

The embedding of clustering should be done with an additional linear branch.

Fraunhofer IIS

Image Analysis with Deep Neural Networks



y_t

(A_1, A_2, A_3)

(x_{t-1}, x_t, x_{t+1})

Traditional image analysis would measure a vector of features from the original image and apply a classification network.

Convolutions are local transformations of an image. The learning of convolutions allows the generation of new local features. Deep nets generate global features.

The following 1dimensional explanation can be extended to multidimensional inputs.

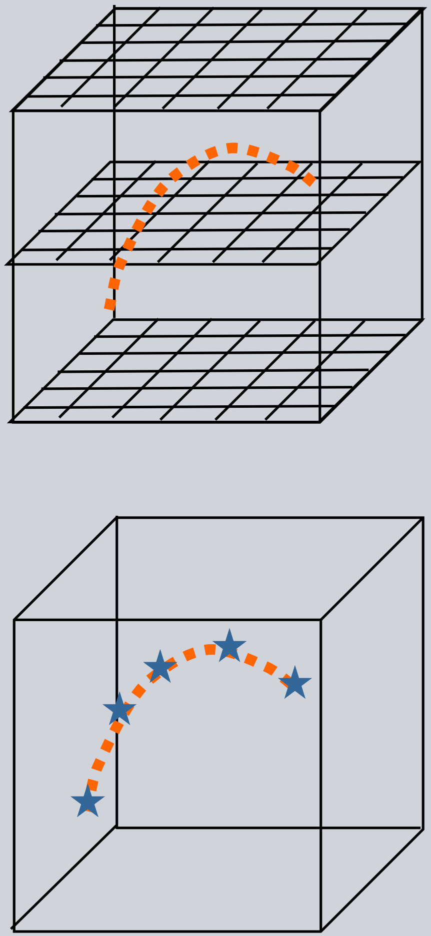
if $\sum_i A_i \approx 1$ the convolution acts as a smoothing, e.g. $y_t = \frac{1}{3}x_{t-1} + \frac{1}{3}x_t + \frac{1}{3}x_{t+1}$

if $\sum_i A_i \approx 0$ the convolution acts as an edge detection, e.g. $y_t = 1x_{t-1} - 2x_t + 1x_{t+1}$

if the A_i are asymmetric the convolution acts as a shift operator, e.g. $y_t = 1x_{t-1} + 0x_t + 0x_{t+1}$

Fraunhofer IIS

The Curse of Dimensionality in Approximation Theory



The curse of dimensionality in Standard Approximation:

$$f(x) \approx \sum_{j=1}^{m} v_j \, b_j(x) \quad \text{with} \quad \left\| \{v_j\} \right\| \approx c^{\dim(x)}$$

This is a linear superposition of basis functions – their number & the number of parameters increase exponentially with dim(x).

Neural Networks escape the curse of dimensionality:

$$f(x) \approx \sum_{j=1}^{m} v_j \, b(w_j, x) \quad \text{with} \quad \left\| \{v_j, w_j\} \right\| \approx Var(f)$$

The independence of the number of parameters from the input dimension is paid with nonlinear optimization.

Support Vector Machines offer an alternative remedy:

$$f(x) \approx \sum_{j=1}^{m} v_j \, b(x - x_j) \quad \text{with} \quad \left\| \{v_j\} \right\| \approx \left\| data\,\{x_j\} \right\|$$

This is a linear superposition of basis functions, using data as internal parameters → essentially, this is data interpolation.

On Model Building: What is a Simple Model?

Often linear models are seen as a simple starting point for model building (Taylor expansion argument, but this is true only nearby the expansion point). **Opposite, we should start with a model class which contains no unjustified a priori structure!**

Start with a simple = universal framework	e.g. - neural networks - others …
↓	
Add reasonable a priori structure	e.g. - monotonic input-output relations - diversity / similarity analysis - dynamical systems - dynamics on manifolds - linearity
↓	
Add data	e.g. - temporal / cross sectional - continuous / ordered / nominal
↓	
A posteriori model interpretation	On past data we can detect correlations only, the interpretation as causality is an intellectual insight.

Fraunhofer IIS

Introduction to Artificial Intelligence

Foundations: Turing (computability), McCulloch / Pitts (computer ~ brain), Hebb (learning), Wiener (cybernetics)

Artificial Intelligence: What To Do? Perception + Understanding + Action	**1955 / 56**	***Artificial Intelligence: How To Do?*** Brain like Information Processing
Perception → **Processing of Heterogeneous Data**		**Decision Support** → **Logical & Fuzzy Rules for Expert Systems**
Understanding → **Model Building based on Observations**		**Stimulus-Response Models** → **Feedforward Neural Nets for Regression & Classification**
Action → **Decision Support, Optimization & Control**		**Dynamical Systems** → **Recurrent Neural Nets for Forecasting & Control**
focus on learning instead of engineering	**today**	deep learning & recurrent neural networks

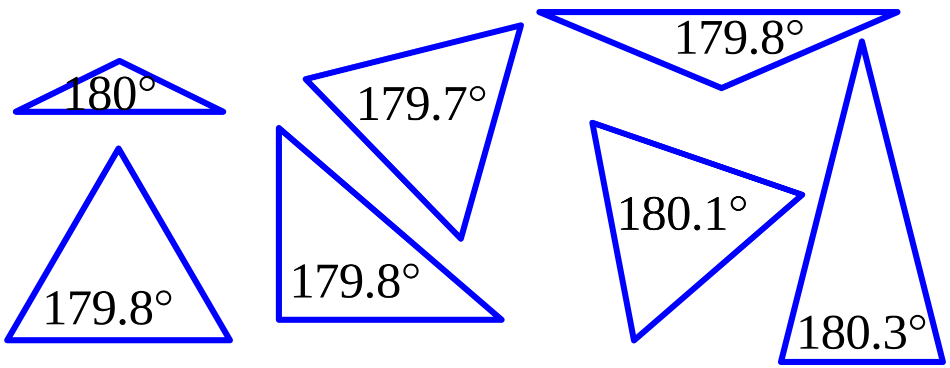
Future : **Weak AI**: Learning of increasing Complex Systems / **Strong AI:** Search for Insight and Consciousness

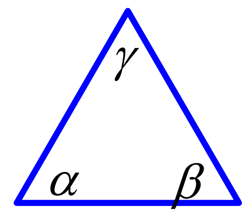
Limits of Artificial Intelligence: Learning from Data versus Insight II

The sum of all angles in a triangle is a constant, equal to 180°.

$$\alpha + \beta + \gamma = 180^\circ$$

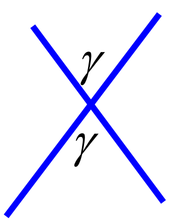
Pseudo-Proof by **Learning from Data** in form of measurements:

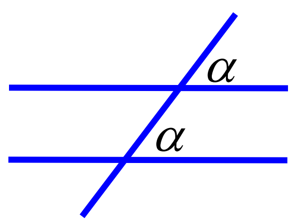


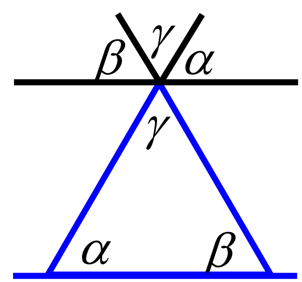


$$\alpha + \beta + \gamma \approx 180^\circ$$

Proof based on Geometric Axioms and **Insight:**







parallel of the base line

$$\alpha + \beta + \gamma = 180^\circ$$

Insight is an Interaction of an Insightful Person and an Area of Interest

The most insightful theories in physics are found in the first place with nearly no data:

Elektrodynamics (Maxwell)
Thermodynamics (Boltzmann)
Relativity Theory (Einstein)
Quantum Theory (was formulated by Schrödinger, Heisenberg, Bohr, Einstein, v.Neumann, Feynman, Bohm in different ways).

The description of an area of interest depends on a person formulating her/his insights.

But in Artificial Intelligence there is no person!