UPPSALA
UNIVERSITET

Deep Learning: Pixels to Torques

Niklas Wahlström

Department of Information Technology, Uppsala University, Sweden

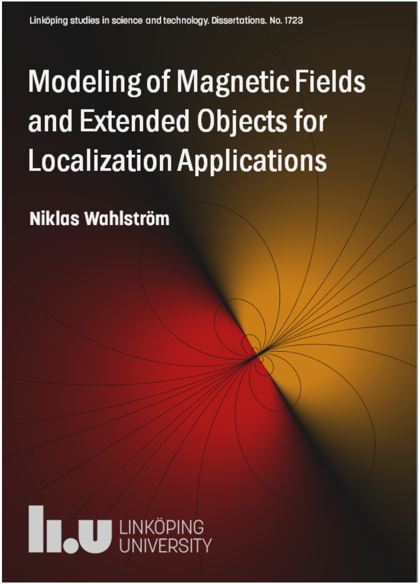
June 16, 2016

UPPSALA UNIVERSITET

Short about me

- 2005 - 2010: Applied Physics and Electrical Engineering - International, Linköping University.
  - 2007-2008: Exchange student, ETH Zürich, Swizerland
- 2010-2015 : PhD student in Automatic Control, Linköping University
  - Spring 2014, Research visit, Imperial College, London, UK
- *2016- : Researcher at Department of Information Technology, Uppsala University*

My thesis



Three areas:

- Magnetic tracking
- Extended target tracking
- Deep dynamical models for control

Magnetic tracking

Advantages

- Cheap sensors

Magnetic tracking

Advantages

- Cheap sensors
- Small sensors

Magnetic tracking

Advantages

- Cheap sensors
- Small sensors
- Low energy consumption

UPPSALA UNIVERSITET

Magnetic tracking

Advantages

- Cheap sensors
- Small sensors
- Low energy consumption
- No weather dependency

Magnetic tracking

Advantages

- Cheap sensors
- Small sensors
- Low energy consumption
- No weather dependency
- Passive unit, requires no batteries

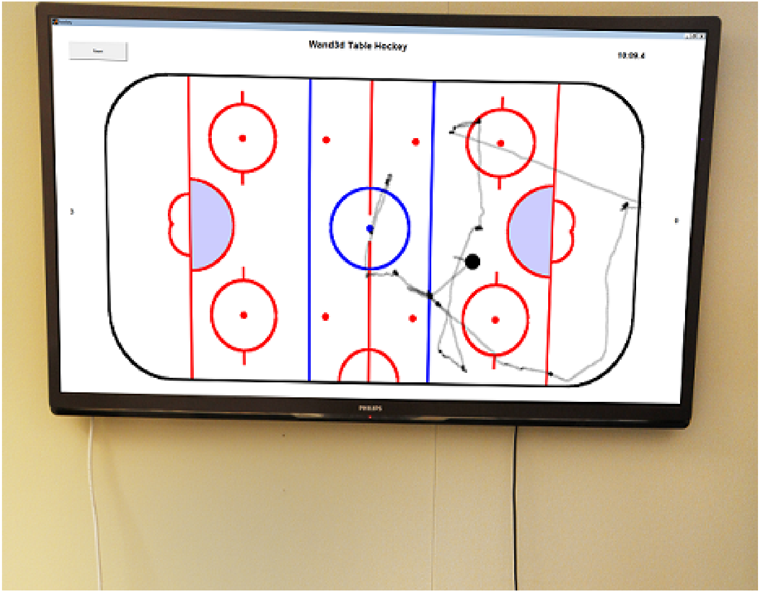
Application 1: Digital pathology

Application 2: Digital water colors

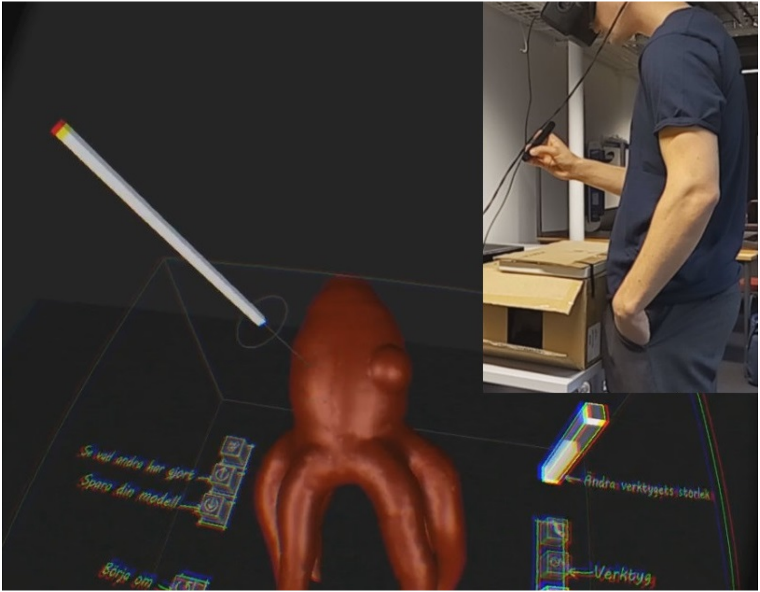
Application 3: Digital 3D coloring book

UPPSALA UNIVERSITET

Application 4: Digital table hockey

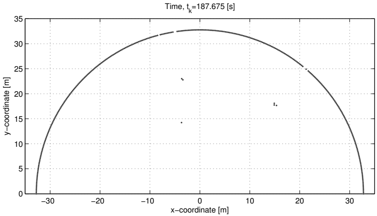


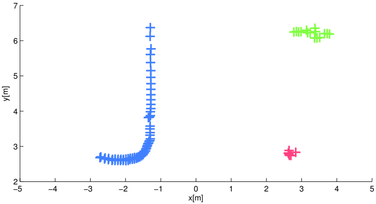
Application 5: Interactive 3D modeling



Extended target tracking

Many sensors generate more than one measurement/target



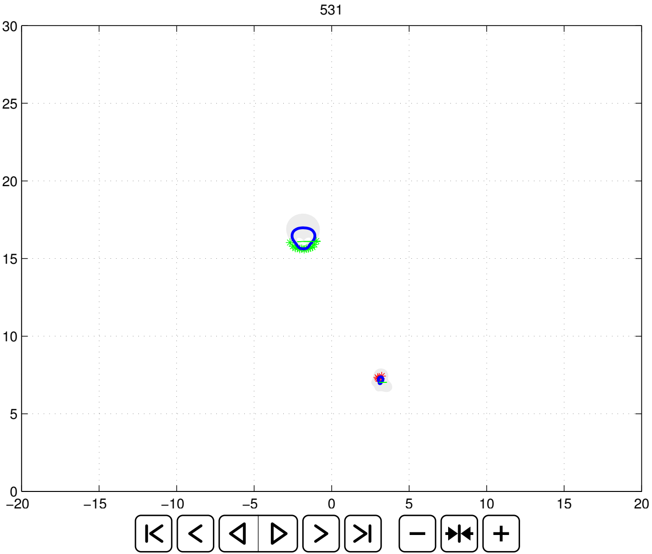


Extended target (definition)

Targets that potentially give rise to multiple measurements at each time step

Goal: We want to estimate target position, target orientation and target extent *jointly*.

Real data experiment -result



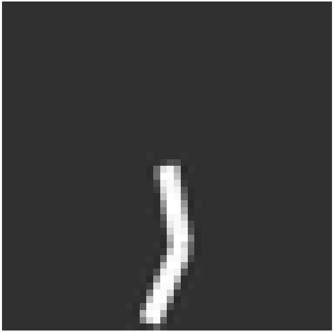
Deep Learning - Outline

1. **Introduction via three recent applications**
2. What is a neural network (NN)?
3. What is a deep neural network?
4. Some model architectures
   a) Auto-encoder
   b) Recurrent neural network
   c) Convolutional neural network
5. Developing and learning a deep dynamical model
   a) Problem formulation
   b) Deep dynamical model
6. Some pointers, summary and the future

Deep Learning: A recent example

First steps towards an autonomous system that learns by itself from raw pixel data.

Trial: 3 Frame: 94



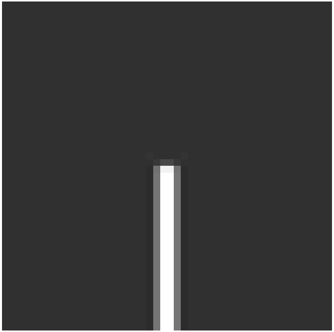
- Deep autoencoder network + nonlinear dynamical model
- Model predictive control (MPC)
- Ref. value: $\mathbf{z}_{\text{ref}} = f_{\text{d}}(\mathbf{y}_{\text{ref}})$
- The model is automatically improved (in an iterative manner)

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. **Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models**. In *Deep Reinforc. Learning WS at the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2015.

Deep Learning: A recent example

First steps towards an autonomous system that learns by itself from raw pixel data.

Trial: 1 Frame: 1



- Deep autoencoder network + nonlinear dynamical model
- Model predictive control (MPC)
- Ref. value: $\mathbf{z}_{\text{ref}} = f_{\text{d}}(\mathbf{y}_{\text{ref}})$
- The model is automatically improved (in an iterative manner)

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. **Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models**. In *Deep Reinforc. Learning WS at the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2015.

Deep Learning: Another recent example

Automatically learn how to describe the contents of images.

Illustrates the **modularity** of the autoencoder, consisting of an **encoder** (vision deep CNN) and a **decoder** (language generating RNN).

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

Xu, K., Lei Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R. Richard S. Zemel, R. S., and Bengio, Y. **Show, attend and tell: neural image caption generation with visual attention**. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, July, 2015.

A few examples where it failed

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Deep learning: On more recent example

A computer defeated the world champion for the first time in the game of Go

Silver, D. et al. **Mastering the game of Go with deep neural networks and tree search**, *Nature*, Vol 529, 484–489 (2016)

Outline

Constructing an NN for regression

A **neural network (NN)** is a nonlinear function $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{u})$ from an input variable $\mathbf{u}$ to an output variable $\mathbf{y}$ parameterized by $\boldsymbol{\theta}$.

Linear regression

Constructing an NN for regression

A **neural network (NN)** is a nonlinear function $\mathbf{y} = \mathrm{g}_{\boldsymbol{\theta}}(\mathbf{u})$ from an input variable $\mathbf{u}$ to an output variable $\mathbf{y}$ parameterized by $\boldsymbol{\theta}$.

Linear regression models the relationship between a continuous target variable y and an input variable $\mathbf{u}$,

$$y = \sum_{i=1}^{D} w_i u_i + b + \epsilon = \boldsymbol{\theta}^{\mathsf{T}} \mathbf{u} + \epsilon,$$

where ϵ is noise and $\boldsymbol{\theta}$ is the parameters composed by the "weights" w_i and the offset ("bias") term b,

$$\boldsymbol{\theta} = \begin{pmatrix} b & w_1 & w_2 & \cdots & w_D \end{pmatrix}^{\mathsf{T}},$$

$$\mathbf{u} = \begin{pmatrix} 1 & u_1 & u_2 & \cdots & u_D \end{pmatrix}^{\mathsf{T}}.$$

Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\boldsymbol{\theta}^\mathsf{T}\mathbf{u}$,

$$y = f(\boldsymbol{\theta}^\mathsf{T}\mathbf{u}).$$

Generalized linear regression

We can generalize this by introducing nonlinear transformations of the predictor $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{u}$,

$$y = f(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{u}).$$

Let us consider an example of a **feed-forward NN**, indicating that the information flows from the input to the output layer.

UPPSALA UNIVERSITET

NN for regression – an example

1. Form M linear combinations of the input $\mathbf{u} \in \mathbb{R}^D$

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}, \qquad j = 1, \dots, M.$$

UPPSALA UNIVERSITET

NN for regression – an example

1. Form M linear combinations of the input $\mathbf{u} \in \mathbb{R}^D$

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}, \qquad j = 1, \dots, M.$$

2. Apply a nonlinear transformation

$$z_j = f\left(a_j^{(1)}\right), \qquad j = 1, \dots, M.$$

niklas.wahlstrom@it.uu.se

NN for regression – an example

1. Form M linear combinations of the input $\mathbf{u} \in \mathbb{R}^D$

$$a_j^{(1)} = \sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}, \qquad j = 1, \dots, M.$$

2. Apply a nonlinear transformation

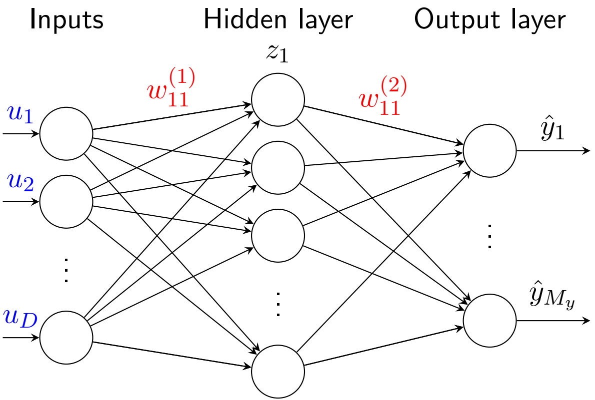
$$z_j = f\left(a_j^{(1)}\right), \qquad j = 1, \dots, M.$$

3. Form M_y linear combinations of $\mathbf{z} \in \mathbb{R}^M$

$$y_k = \sum_{j=1}^{M_y} w_{kj}^{(2)} z_j + b_k^{(2)}, \qquad k = 1, \dots, M_y.$$

NN for regression – an example

$$\hat{y}_k(\theta) = \sum_{j=1}^{M} w_{kj}^{(2)} f\left(\sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}\right) + b_k^{(2)}$$



Multi-layer neural networks

We can think of the neural network as a sequential/recursive construction of several generalized linear regressions.

Each layer in a multi-layer NN is modelled as

$$\mathbf{z}^{(l+1)} = \mathbf{f}\left(W^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}\right),$$

starting with the input $\mathbf{z}^{(0)} = \mathbf{u}$. (The nonlinearity operates element-wise.)

UPPSALA UNIVERSITET

Multi-layer neural networks

We can think of the neural network as a sequential/recursive construction of several generalized linear regressions.

Each layer in a multi-layer NN is modelled as

$$\mathbf{z}^{(l+1)} = \mathbf{f}\left(W^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}\right),$$

starting with the input $\mathbf{z}^{(0)} = \mathbf{u}$. (The nonlinearity operates element-wise.)

The scalar nonlinear function $f(\cdot)$ is what makes the neural network nonlinear. Common functions are $f(z) = 1/(1+e^{-z})$, $f(z) = \tanh(z)$ and $f(z) = \max(0, z)$.

The so-called **rectified linear unit (ReLU)** $f(z) = \max(0, z)$ is heavily used for deep architectures.

Training a NN

The final layer $\mathbf{z}^{(L)}$ of the network is used for making a prediction $\hat{\mathbf{y}}(\boldsymbol{\theta}) = \mathbf{z}^{(L)}$ and we train the network by employing:

1. A set of training data.
2. A cost function $\mathcal{L}\left(\hat{\mathbf{y}}(\theta), \mathbf{y}\right)$.
3. An iterative scheme to optimize the cost function

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} \mathcal{L}\left(\hat{\mathbf{y}}_n(\theta), \mathbf{y}_n\right).$$

Training a NN

The final layer $\mathbf{z}^{(L)}$ of the network is used for making a prediction $\hat{\mathbf{y}}(\boldsymbol{\theta}) = \mathbf{z}^{(L)}$ and we train the network by employing:

1. A set of training data.
2. A cost function $\mathcal{L}\left(\hat{\mathbf{y}}(\theta), \mathbf{y}\right)$.
3. An iterative scheme to optimize the cost function

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} \mathcal{L}\left(\hat{\mathbf{y}}_n(\theta), \mathbf{y}_n\right).$$

Training a NN does involve a lot of **engineering skill** and is more of an art than a mathematically rigorous exercise.

Backpropagation

Recall our example network again:

$$\hat{y}_k(\theta) = \sum_{j=1}^{M} w_{kj}^{(2)} f\left(\sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}\right) + b_k^{(2)}$$

Backpropagation

Recall our example network again:

$$\hat{y}_k(\theta) = \sum_{j=1}^{M} w_{kj}^{(2)} f\left(\sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}\right) + b_k^{(2)}$$

In solving the optimization problem

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \; J(\boldsymbol{\theta})$$

we typically employ gradient methods using $\nabla J(\boldsymbol{\theta})$.

Backpropagation

Recall our example network again:

$$\hat{y}_k(\theta) = \sum_{j=1}^{M} w_{kj}^{(2)} f\left(\sum_{i=1}^{D} w_{ji}^{(1)} u_i + b_j^{(1)}\right) + b_k^{(2)}$$

In solving the optimization problem

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

we typically employ gradient methods using $\nabla J(\boldsymbol{\theta})$.

Backpropagation amounts to computing the gradients via (recursive) use of the **chain rule**, combined with **reuse** of information that is needed for more than one gradient.

Outline

1. Introduction via three recent applications
2. What is a neural network (NN)?
3. **What is a deep neural network?**
4. Some model architectures
   a) Auto-encoder
   b) Recurrent neural network
   c) Convolutional neural network
5. Developing and learning a deep dynamical model
   a) Problem formulation
   b) Deep dynamical model
6. Some pointers, summary and the future

Deep neural networks

Deep learning methods allow a machine to make use of raw data to automatically discover the representations (abstractions) that are necessary to solve a particular task.

Deep neural networks

Deep learning methods allow a machine to make use of raw data to automatically discover the representations (abstractions) that are necessary to solve a particular task.

It is accomplished by using **multiple levels of representation**. Each level transforms the representation at the previous level into a new and more abstract representation,

$$\mathbf{z}^{(l+1)} = \mathbf{f}\left(W^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}\right),$$

starting from the input (raw data) $\mathbf{z}^{(0)} = \mathbf{u}$.

Deep neural networks

Deep learning methods allow a machine to make use of raw data to automatically discover the representations (abstractions) that are necessary to solve a particular task.

It is accomplished by using **multiple levels of representation**. Each level transforms the representation at the previous level into a new and more abstract representation,

$$\mathbf{z}^{(l+1)} = \mathbf{f}\left(W^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}\right),$$

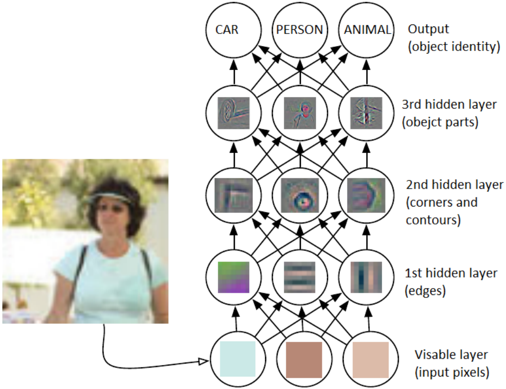
starting from the input (raw data) $\mathbf{z}^{(0)} = \mathbf{u}$.

Key aspect: The layers are **not** designed by human engineers, they are generated from (typically lots of) data using a learning procedure and lots of computations.

Hierarchy of features

Example: Image classification

The input layer represents an **image** and the output layer an **object identity**. Each hidden layer extracts increasingly abstract features.



Zeiler, M. D. and Fergus, R. **Visualizing and understanding convolutional networks**

Computer Vision - ECCV (2014).

UPPSALA UNIVERSITET

Training deep neural networks

The main problem with a deep architecture is the training. The strategy sketched above will not work.

The breakthrough came 10 years ago:

Hinton, G. E., Osindero, S. and Teh, Y-W. **A Fast Learning Algorithm for Deep Belief Nets**. *Neural Computation*, 18, 1527–1554, 2006.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. **Greedy layer-wise training of deep networks**. In *Proc. Advances in Neural Information Processing Systems (NIPS)* 19, 153–160, 2006.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. **Efficient learning of sparse representations with an energy-based model**. In *Proc. Advances in Neural Information Processing Systems (NIPS)* 19, 1137–1144, 2006.

UPPSALA UNIVERSITET

Training deep neural networks

The main problem with a deep architecture is the training. The strategy sketched above will not work.

The breakthrough came 10 years ago:

Hinton, G. E., Osindero, S. and Teh, Y-W. **A Fast Learning Algorithm for Deep Belief Nets**. *Neural Computation*, 18, 1527–1554, 2006.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. **Greedy layer-wise training of deep networks**. In *Proc. Advances in Neural Information Processing Systems (NIPS)* 19, 153–160, 2006.

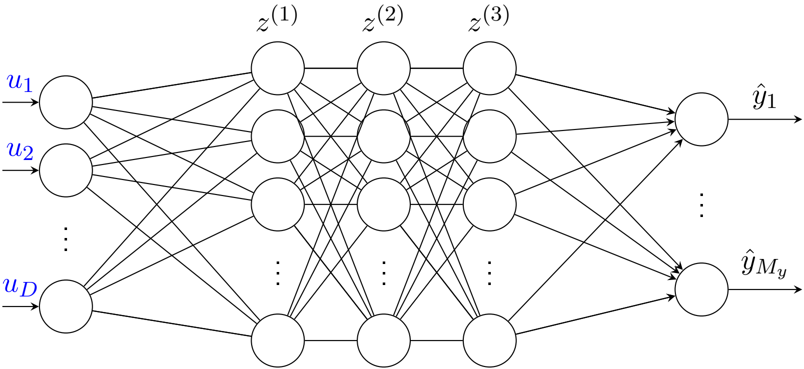
Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. **Efficient learning of sparse representations with an energy-based model**. In *Proc. Advances in Neural Information Processing Systems (NIPS)* 19, 1137–1144, 2006.

Key idea: Careful initialization by training each layer individually using an unsupervised algorithm. Referred to as **pre-training**.

Finally, a supervised algorithm (e.g. backpropagation) is used to fine-tune the parameters $\boldsymbol{\theta}$ using the result from the pre-training as initial values.

UPPSALA UNIVERSITET

Pre-training



Pre-training evolves sequentially from input to output. Here:

- 3 stages of unsupervised training
- 1 stage of supervised training

UPPSALA UNIVERSITET

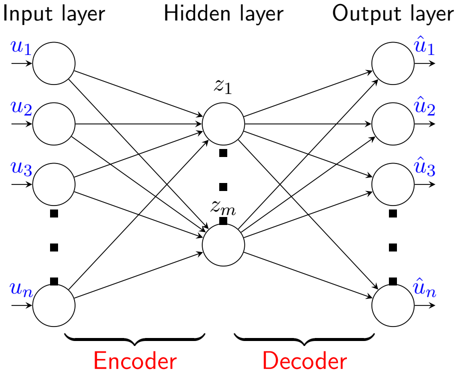
Outline

1. Introduction via three recent applications
2. What is a neural network (NN)?
3. What is a deep neural network?
4. **Some model architectures**
   a) Auto-encoder
   b) Recurrent neural network
   c) Convolutional neural network
5. Developing and learning a deep dynamical model
   a) Problem formulation
   b) Deep dynamical model
6. Some pointers, summary and the future

Autoencoder

The autoencoder is an unsupervised learning procedure for **dimensionality reduction**.

It is a NN that learns compressed representations $\mathbf{z}$ of high-dimensional data $\mathbf{u}$, where $\dim(\mathbf{u}) \gg \dim(\mathbf{z})$.



Encoder: $\mathbf{z} = \mathbf{f}_{\mathsf{e}}(\mathbf{u}) = \mathbf{f}(W^{\mathsf{T}}\mathbf{u} + \mathbf{b}).$

Decoder: $\hat{\mathbf{u}} = \mathbf{f}_{\mathsf{d}}(\mathbf{z}) = \mathbf{f}(\bar{W}^{\mathsf{T}}\mathbf{z} + \bar{\mathbf{b}}).$

niklas.wahlstrom@it.uu.se

Training the autoencoder

The unknown parameters

$$\boldsymbol{\theta} = \{W, \mathbf{b}, \bar{W}, \bar{\mathbf{b}}\}$$

are estimated by minimizing the reconstruction error

$$\mathbf{e} = \mathbf{u} - \hat{\mathbf{u}}(\boldsymbol{\theta}),$$

using some cost function $J(\boldsymbol{\theta})$, for example LS

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} \|\mathbf{u}_n - \hat{\mathbf{u}}_n(\boldsymbol{\theta})\|^2.$$

Training the autoencoder

The unknown parameters

$$\boldsymbol{\theta} = \{W, \mathbf{b}, \bar{W}, \bar{\mathbf{b}}\}$$

are estimated by minimizing the reconstruction error

$$\mathbf{e} = \mathbf{u} - \hat{\mathbf{u}}(\boldsymbol{\theta}),$$

using some cost function $J(\boldsymbol{\theta})$, for example LS

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} \|\mathbf{u}_n - \hat{\mathbf{u}}_n(\boldsymbol{\theta})\|^2.$$

After the training the encoder and the decoder will (by construction) be **approximate inverses** of each other,

$$\mathbf{f}_{\mathsf{d}}(\mathbf{f}_{\mathsf{e}}(\mathbf{u})) \approx \mathbf{u}.$$

Autoencoder

We can then easily transform either u into z or z into $\hat{u}$ using either the encoder

$$\mathbf{z} = \mathbf{f}_{\mathsf{e}}(W^{\mathsf{T}}\mathbf{u} + \mathbf{b}),$$

or the decoder,

$$\hat{\mathbf{u}} = \mathbf{f}_{\mathsf{d}}(\bar{W}^{\mathsf{T}}\mathbf{z} + \bar{\mathbf{b}}).$$

The access to both of these two mappings is important for certain applications (such as the deep dynamical model).

Autoencoder

We can then easily transform either u into z or z into $\hat{u}$ using either the encoder

$$\mathbf{z} = \mathbf{f}_{\mathsf{e}}(W^{\mathsf{T}}\mathbf{u} + \mathbf{b}),$$

or the decoder,

$$\hat{\mathbf{u}} = \mathbf{f}_{\mathsf{d}}(\bar{W}^{\mathsf{T}}\mathbf{z} + \bar{\mathbf{b}}).$$

The access to both of these two mappings is important for certain applications (such as the deep dynamical model).

If $\mathbf{f}_{\mathsf{e}}(\cdot)$ is chosen to be the identity (i.e. $\mathbf{z} = W^{\mathsf{T}}\mathbf{u} + \mathbf{b}$) and $\dim \mathbf{u} < \dim \mathbf{z}$ then the autoencoder is **equivalent to PCA.** Hence, the autoencoder is a nonlinear generalization of PCA.

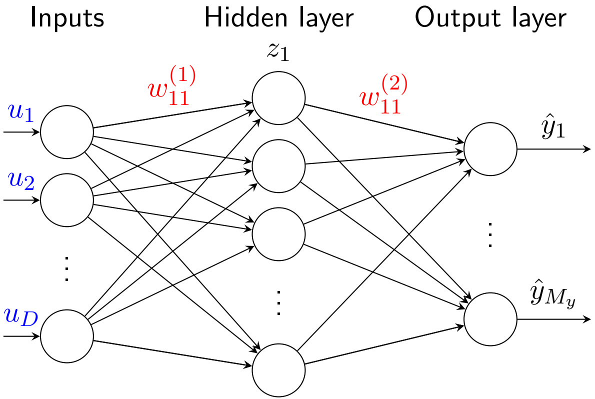
Deep autoencoder

The deep autoencoder is simply an autoencoder with several hidden layers.

Again, careful initialization is important for this to work, using the same pre-training as described before.

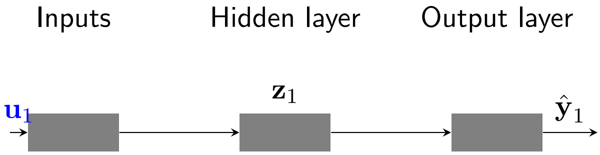
Hinton, G. E. and Salakhutdinov, R. R. **Reducing the Dimensionality of Data with Neural Networks**. Science, 313, 504–507, 2006.

Reccurent nureal networks



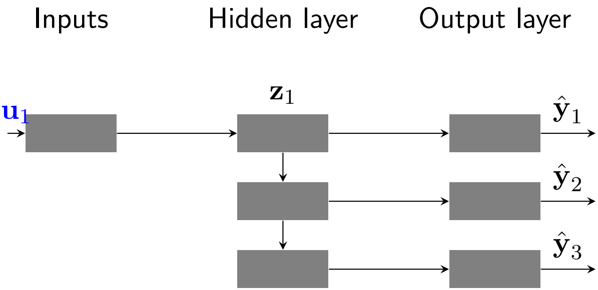
A normal feed-foreward nureal network is restrictive in the sense that they accept only a fixed-sized input and a fixed-sized output.

Reccurent nureal networks



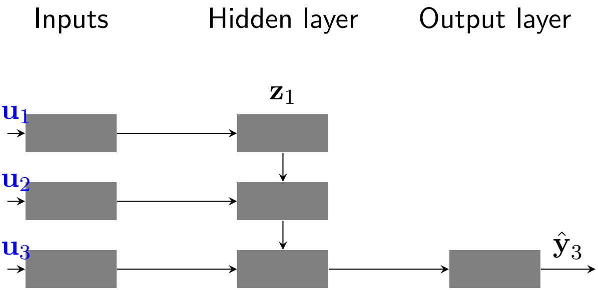
Feed-forward network, from fixed-sized input to fixed-sized output (e.g. image classification).

Reccurent nureal networks



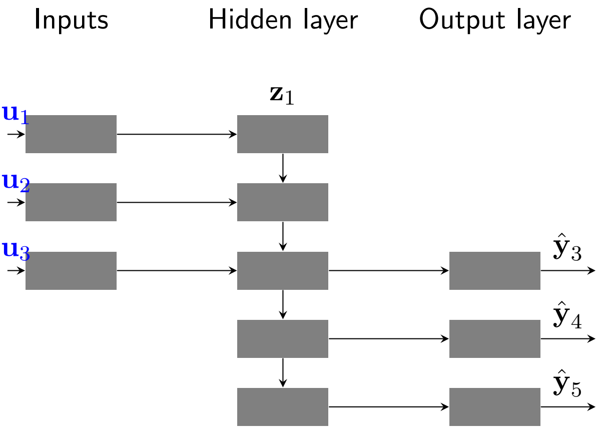
Sequence output (e.g. image captioning where an image is input and and a sentence of words is output).

Reccurent nureal networks



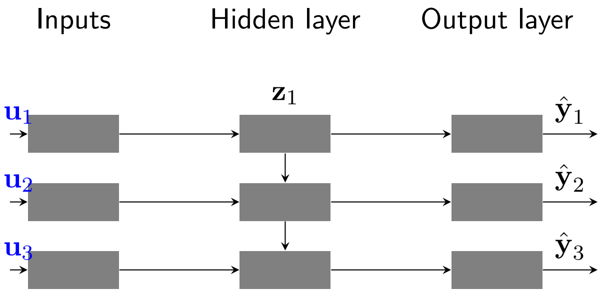
Sequence input (e.g. sentiment analysis where a given sentence is classified).

Reccurent nureal networks



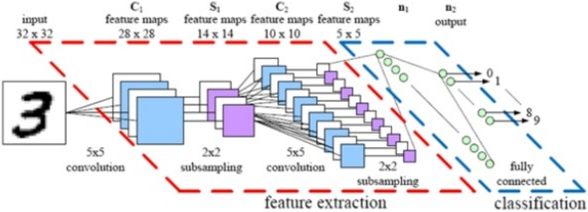
Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).

Reccurent nureal networks



Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). This looks quite similar to a state space model!

Convolutional nueral networks



Convolutional networks (ConvNets) Makes use of the weight sharing idea. Nodes forms groups of 2D arrays.

Particularly successful in machine vision.

The convNet is a notable early successful deep architecture.

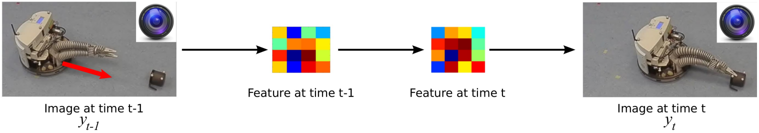
Outline

1. Introduction via three recent applications
2. What is a neural network (NN)?
3. What is a deep neural network?
4. Some model architectures
    a) Auto-encoder
    b) Recurrent neural network
    c) Convolutional neural network
5. Developing and learning a deep dynamical model
    a) Problem formulation
    b) Deep dynamical model
6. Some pointers, summary and the future

Outline

Motivation

- **Vision:** fully autonomous systems that learn by themselves from raw pixel data.
- **Strategy:** A deep dynamical model is proposed that contains a low-dimensional dynamical model.



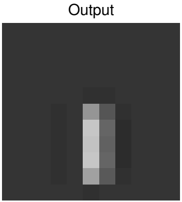
N. Wahlström, T. B. Schön, M. P. Deisenroth **Learning deep dynamical models from image pixels**
The 17th IFAC Symposium on System Identification (SYSID)

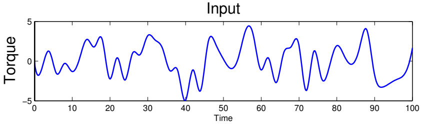
Problem Formulation

Problem formulation: Modeling of high-dimensional pixel data

Example: Video stream of a pendulum

- **Input:** Torque of a pendulum
- **Output:** Pixel values of an 11×11 image





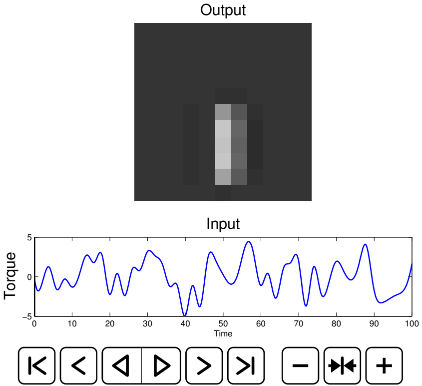
niklas.wahlstrom@it.uu.se

Problem Formulation

Problem formulation: Modeling of high-dimensional pixel data

Example: Video stream of a pendulum

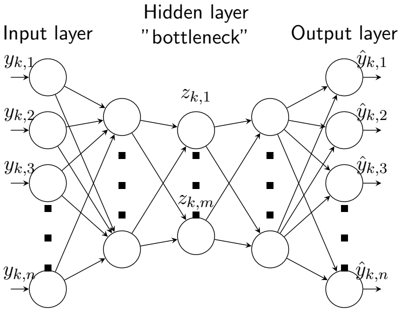
- **Input:** Torque of a pendulum
- **Output:** Pixel values of an 11×11 image



The Autoencoder

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features



niklas.wahlstrom@it.uu.se

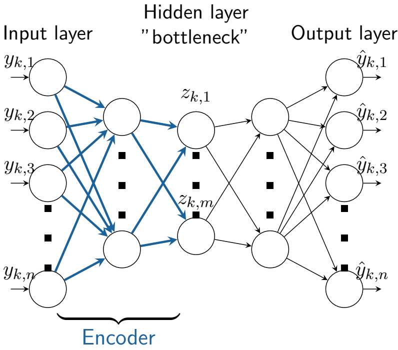
The Autoencoder

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features

Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{f}_{\mathsf{e}}(\mathbf{y}_k; \boldsymbol{\theta}_{\mathsf{E}})$



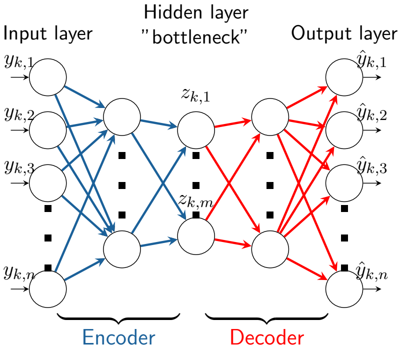
The Autoencoder

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features

Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{f}_{\mathsf{e}}(\mathbf{y}_k; \boldsymbol{\theta}_{\mathsf{E}})$
2. Decoder: $\widehat{\mathbf{y}}_k^{\mathsf{R}} = \mathbf{f}_{\mathsf{d}}(\mathbf{z}_k; \boldsymbol{\theta}_{\mathsf{D}})$



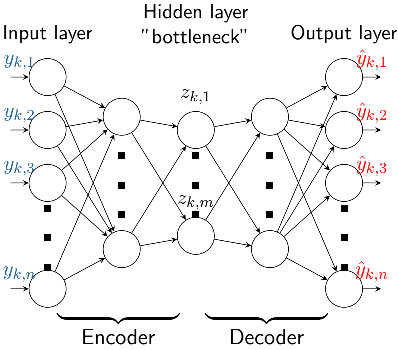
The Autoencoder

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features

Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{f}_\mathsf{e}(\mathbf{y}_k; \boldsymbol{\theta}_\mathsf{E})$
2. Decoder: $\widehat{\mathbf{y}}_k^\mathsf{R} = \mathbf{f}_\mathsf{d}(\mathbf{z}_k; \boldsymbol{\theta}_\mathsf{D})$



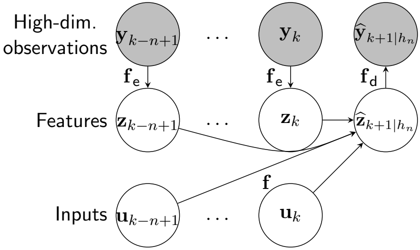
Reconstruction error:

$$V_\mathsf{R}(\boldsymbol{\theta}_\mathsf{E}, \boldsymbol{\theta}_\mathsf{D}) = \sum_{k=1}^{N} \|\mathbf{y}_k - \widehat{\mathbf{y}}_k^\mathsf{R}(\boldsymbol{\theta}_\mathsf{E}, \boldsymbol{\theta}_\mathsf{D})\|^2$$

Deep Dynamical Model

Notation:

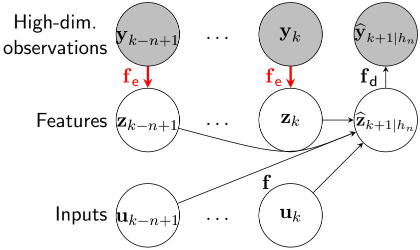
- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\mathbf{u}_k$ - Inputs



Deep Dynamical Model

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\mathbf{u}_k$ - Inputs



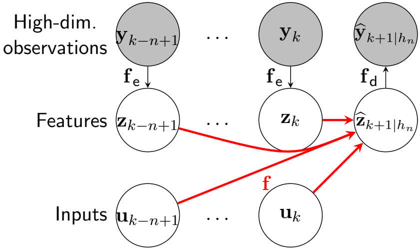
Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{f}_\mathsf{e}(\mathbf{y}_k; \boldsymbol{\theta}_\mathsf{E})$

Deep Dynamical Model

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\mathbf{u}_k$ - Inputs



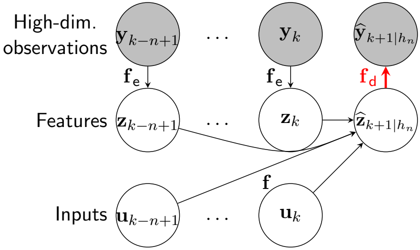
Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{f}_\mathsf{e}(\mathbf{y}_k; \boldsymbol{\theta}_\mathsf{E})$
2. Prediction model: $\widehat{\mathbf{z}}_{k+1|k} = \mathbf{f}(\mathbf{z}_k, \mathbf{u}_k, \dots, \mathbf{z}_{k-n+1}, \mathbf{u}_{k-n+1}; \boldsymbol{\theta}_\mathsf{P})$

Deep Dynamical Model

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\mathbf{u}_k$ - Inputs



Model components:

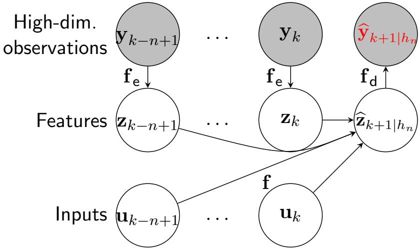
1. Encoder: $\mathbf{z}_k = \mathbf{f}_\mathsf{e}(\mathbf{y}_k; \boldsymbol{\theta}_\mathsf{E})$
2. Prediction model: $\widehat{\mathbf{z}}_{k+1|k} = \mathbf{f}(\mathbf{z}_k, \mathbf{u}_k, \dots, \mathbf{z}_{k-n+1}, \mathbf{u}_{k-n+1}; \boldsymbol{\theta}_\mathsf{P})$
3. Decoder: $\widehat{\mathbf{y}}^\mathsf{P}_{k+1|k} = \mathbf{f}_\mathsf{d}(\widehat{\mathbf{z}}_{k+1|k}; \boldsymbol{\theta}_\mathsf{D})$

niklas.wahlstrom@it.uu.se

Deep Dynamical Model

Notation:

- $\mathbf{y}_k$ - High-dim. observations
- $\mathbf{z}_k$ - Low-dim. features
- $\mathbf{u}_k$ - Inputs



Model components:

1. Encoder: $\mathbf{z}_k = \mathbf{f}_\mathsf{e}(\mathbf{y}_k; \boldsymbol{\theta}_\mathsf{E})$
2. Prediction model: $\widehat{\mathbf{z}}_{k+1|k} = \mathbf{f}(\mathbf{z}_k, \mathbf{u}_k, \ldots, \mathbf{z}_{k-n+1}, \mathbf{u}_{k-n+1}; \boldsymbol{\theta}_\mathsf{P})$
3. Decoder: $\widehat{\mathbf{y}}^\mathsf{P}_{k+1|k} = \mathbf{f}_\mathsf{d}(\widehat{\mathbf{z}}_{k+1|k}; \boldsymbol{\theta}_\mathsf{D})$

Prediction error:

$$V_\mathsf{P}(\boldsymbol{\theta}_\mathsf{E}, \boldsymbol{\theta}_\mathsf{D}, \boldsymbol{\theta}_\mathsf{P}) = \sum_{k=n}^{N-1} \|\mathbf{y}_{k+1} - \widehat{\mathbf{y}}^\mathsf{P}_{k+1|k}(\boldsymbol{\theta}_\mathsf{E}, \boldsymbol{\theta}_\mathsf{D}, \boldsymbol{\theta}_\mathsf{P})\|^2$$

Training

Key ingredient: The reconstruction error and the prediction error are minimized *simultaneously*!

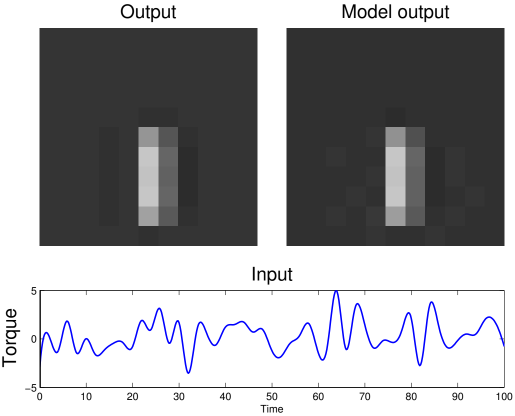
$$(\widehat{\boldsymbol{\theta}}_{\mathsf{E}}, \widehat{\boldsymbol{\theta}}_{\mathsf{D}}, \widehat{\boldsymbol{\theta}}_{\mathsf{P}}) = \underset{\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}}}{\arg\min}\ V_{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}) + V_{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}})$$

$$V_{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}) = \sum_{k=1}^{N} \|\mathbf{y}_k - \widehat{\mathbf{y}}_k^{\mathsf{R}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}})\|^2,$$

$$V_{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}}) = \sum_{k=n}^{N-1} \|\mathbf{y}_{k+1} - \widehat{\mathbf{y}}_{k+1|k}^{\mathsf{P}}(\boldsymbol{\theta}_{\mathsf{E}}, \boldsymbol{\theta}_{\mathsf{D}}, \boldsymbol{\theta}_{\mathsf{P}})\|^2.$$

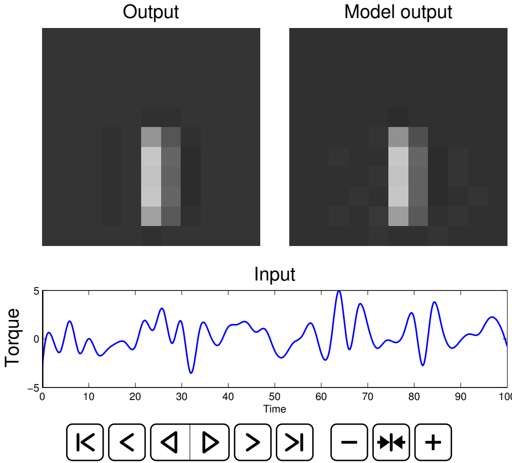
Experiment: Pendulum

- Layers in encoder/decoder: 4
- Latent dim.: $\dim(\mathbf{z}) = 1$
- Order of prediction model: $n = 4$



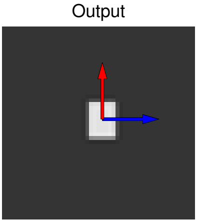
Experiment: Pendulum

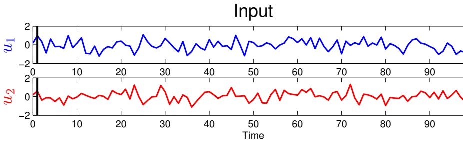
- Layers in encoder/decoder: 4
- Latent dim.: $\dim(\mathbf{z}) = 1$
- Order of prediction model: $n = 4$



Experiment: Agent in a Planar System

- **Input:** Offset in x−dir. (u_1) and y−dir. (u_2)
- **Output:** Pixel values of a 51×51 image
- **Latent dim.:** $\dim(\mathbf{z}) = 2$

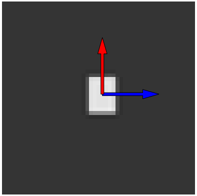




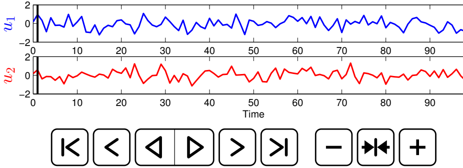
Experiment: Agent in a Planar System

- **Input:** Offset in x−dir. (u_1) and y−dir. (u_2)
- **Output:** Pixel values of a 51×51 image
- **Latent dim.:** $\dim(\mathbf{z})=2$

Output



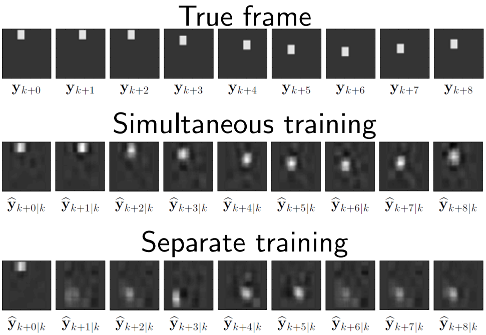
Input



 niklas.wahlstrom@it.uu.se

Experiment: Agent in a Planar System

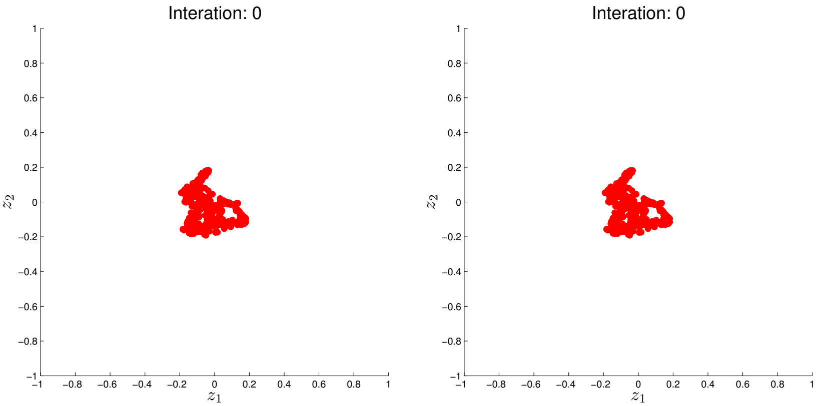
Separate vs. Simultaneous Training



Experiment: Agent in a Planar System

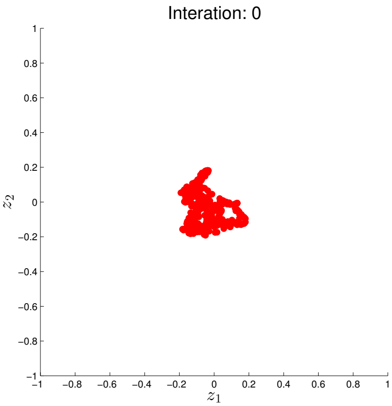
Simultaneous Training

Separate Training

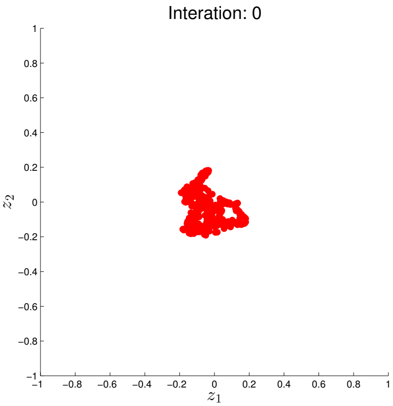


Experiment: Agent in a Planar System

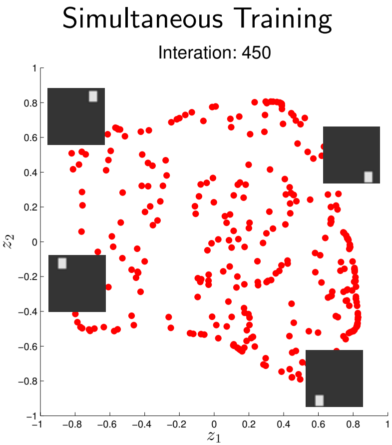
Simultaneous Training

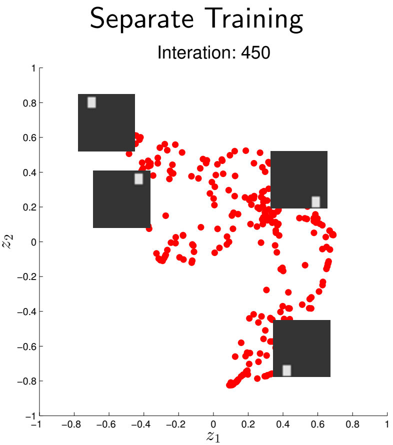


Separate Training



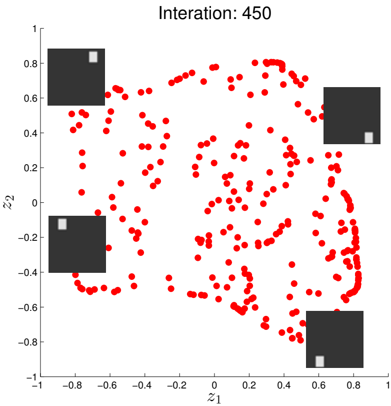
Experiment: Agent in a Planar System



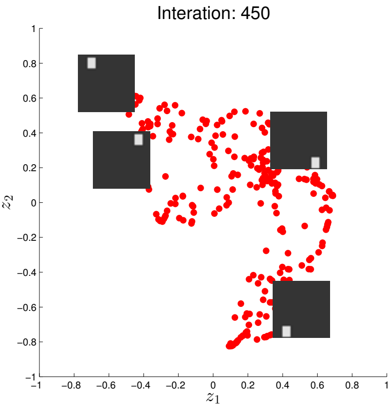


Experiment: Agent in a Planar System

Simultaneous Training



Separate Training



Deep Dynamical Models for Control

The DDM is used to learn a closed-loop policy via nonlinear **model predictive control (MPC)**. Future control signals are optimized by minimizing

$$u_0^*, \dots, u_{K-1}^* \in \operatorname*{arg\,min}_{u_{0:K-1}} \sum_{k=0}^{K-1} \|\widehat{\mathbf{z}}_k - \mathbf{z}_{\text{ref}}\|^2 + \lambda \|u_k\|^2,$$

where $\mathbf{z}_{\text{ref}} = \mathbf{f}_e(y_{\text{ref}, \boldsymbol{\theta}_e}$ is the feature of the reference image. When the control sequence $u_0^*, \dots, u_{K-1}^*$ is determined, the first control u_0^* is applied to the system.
Hence, the MPC is **only applied in the low-dimensional feature space!**

Deep Dynamical Models for Control

Proposed algorithm

Follow a random control strategy and record data

loop

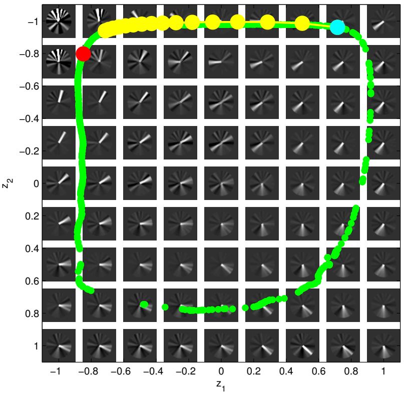
Update DDM with all data collected so far

for $k = 0$ to $N-1$ **do**

- Get $z_k, ..., z_{k-n+1}$ via encoder.
- $u_k^* \leftarrow \epsilon$-greedy MPC policy using DDM prediction.
- Apply u_k^* and record data.

end for

end loop



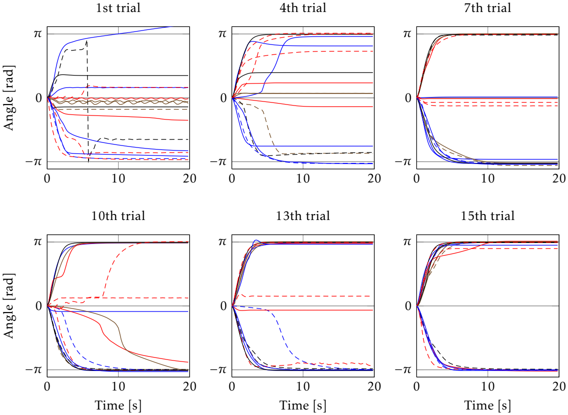
Green: Previous feature values
Cyan: Current feature
Red: Reference feature
Yellow: 15-step ahead prediction

N. Wahlström, T. B Schön, and M. P. Deisenroth
From Pixels to Torques: Policy Learning with Deep Dynamical Models. *ArXiv e-prints 1502.02251*

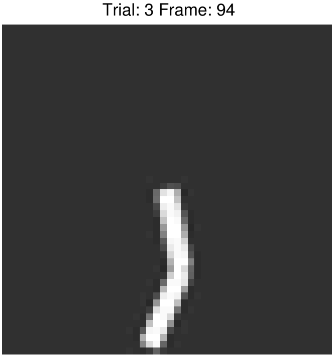
UPPSALA UNIVERSITET

Experiment: Control of a Pendulum from Pixels Only

- Ref. image: Pendulum pointing upwards
- 100 images in each trial
- After 15 trials, a good controller was learned



Application: Control of Two-Link Arm from Pixels Only

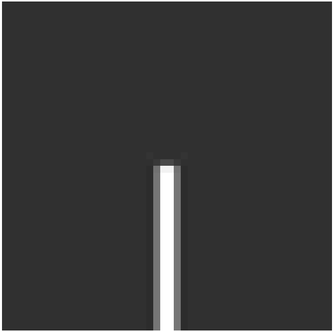


- Ref. image: Arm pointing upwards
- 1000 images in each trial
- After 8-9 trials a fairly good controller was learned.

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. **Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models**. In *Deep Reinforc. Learning WS at the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2015.

Application: Control of Two-Link Arm from Pixels Only

Trial: 1 Frame: 1



- Ref. image: Arm pointing upwards
- 1000 images in each trial
- After 8-9 trials a fairly good controller was learned.

J.-A. M. Assael, N. Wahlström, T. B. Schön, and M. P. Deisenroth. **Data-Efficient Learning of Feedback Policies from Image Pixels using Deep Dynamical Models**. In *Deep Reinforc. Learning WS at the Conference on Neural Information Processing Systems (NIPS)*, Montréal, Canada, Dec. 2015.

Outline

1. Introduction via three recent applications
2. What is a neural network (NN)?
3. What is a deep neural network?
4. Some model architectures
    a) Auto-encoder
    b) Recurrent neural network
    c) Convolutional neural network
5. Developing and learning a deep dynamical model
    a) Problem formulation
    b) Deep dynamical model
6. **Some pointers, summary and the future**

Some pointers

A book is being written at the moment

I. Goodfellow, Y. Bengio and A. Courville **Deep learning** *Book in preparation for MIT Press*,

`http://www.deeplearningbook.org/`

UPPSALA UNIVERSITET

Some pointers

A book is being written at the moment

I. Goodfellow, Y. Bengio and A. Courville **Deep learning** *Book in preparation for MIT Press*,

http://www.deeplearningbook.org/

A well written and timely introduction:

LeCun, Y., Bengio, Y., and Hinton, G. (2015) **Deep learning**, *Nature*, 521(7553), 436–444.

Some pointers

A book is being written at the moment

I. Goodfellow, Y. Bengio and A. Courville **Deep learning** *Book in preparation for MIT Press*,

`http://www.deeplearningbook.org/`

A well written and timely introduction:

LeCun, Y., Bengio, Y., and Hinton, G. (2015) **Deep learning**, *Nature*, 521(7553), 436–444.

You will also find more material than you can possibly want here

`http://deeplearning.net/`

UPPSALA UNIVERSITET

Summary (I/II)

A **neural network (NN)** is a nonlinear function $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{u})$ from an input variable $\mathbf{u}$ to an output variable $\mathbf{y}$ parameterized by $\boldsymbol{\theta}$.

Summary (I/II)

A **neural network (NN)** is a nonlinear function $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{u})$ from an input variable $\mathbf{u}$ to an output variable $\mathbf{y}$ parameterized by $\boldsymbol{\theta}$.

We can think of an NN as a sequential/recursive construction of several generalized linear regressions.

niklas.wahlstrom@it.uu.se

UPPSALA UNIVERSITET

Summary (I/II)

A **neural network (NN)** is a nonlinear function $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{u})$ from an input variable $\mathbf{u}$ to an output variable $\mathbf{y}$ parameterized by $\boldsymbol{\theta}$.

We can think of an NN as a sequential/recursive construction of several generalized linear regressions.

Deep learning refers to learning NNs with several hidden layers. Allows for data-driven models that automatically learns rep. of data (features) with multiple layers of abstraction.

Summary (I/II)

A **neural network (NN)** is a nonlinear function $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{u})$ from an input variable $\mathbf{u}$ to an output variable $\mathbf{y}$ parameterized by $\boldsymbol{\theta}$.

We can think of an NN as a sequential/recursive construction of several generalized linear regressions.

Deep learning refers to learning NNs with several hidden layers. Allows for data-driven models that automatically learns rep. of data (features) with multiple layers of abstraction.

The deep **autoencoder** makes use of a multi-layer "encoder" network to transform high-dimensional data into a low-dimensional code/feature and a similar "decoder" network is used to recover the data from the code.

Summary (II/II)

Deep dynamical model:

- Model for high-dimensional pixel data
- Simultaneous training is crucial
- Application: Control based on pixel data only

The future

The best predictive performance is obtained from **highly flexible models** (especially when large datasets are used). There are basically two ways of achieving flexibility:

1. Using models with a large number of parameters compared to the data set (e.g. deep NN).
2. Models using non-parametric components, e.g. Gaussian processes.

niklas.wahlstrom@it.uu.se

UPPSALA UNIVERSITET

The future

The best predictive performance is obtained from **highly flexible models** (especially when large datasets are used). There are basically two ways of achieving flexibility:

1. Using models with a large number of parameters compared to the data set (e.g. deep NN).
2. Models using non-parametric components, e.g. Gaussian processes.

Use the network also for “attention” and control. Use reinforcement learning to decide **where to look** for new data (resulting in new knowledge).

UPPSALA UNIVERSITET

Thank you!