Deep-Learning for Climate

25-05-2018 – SAMA Machine Learning

J. Brajard*, A. Charantonis**, P. Gallinari*

*Sorbonne Universite, Paris, France, ** ENSIEE

patrick.gallinari@lip6.fr
Outline

• Context

• Deep-Learning models used for problems in climate modeling
  • CNNs, RNNs, STNs, generative models

• Examples of Deep Learning for Climate Applications
  • Event detection
  • Spatio-temporal modeling
  • NN as dynamic models
    • Links between NNs and ODEs
Context

• Brief review of the literature on Deep learning applications to climate modeling

• Literature
  • Increasing number of application papers from both the « climate » and CS communities
    • e.g. 4 papers + 2 invited talks at « Climate Informatics 2017 »
    • Most are still preliminary work:
      • basic applications of Deep Learning methods
      • or « toy » problems
    • Several platforms available (Google-tensorFlow, Facebook PyTorch, etc) make Deep Learning experimentation easy
  • Some innovative papers from the Machine Learning community
    • Application validity?
Context

• Mainly found two application topics
  • Event detection
    • Eddy detection + following, Extreme Weather detection
    • Models: CNN, convolution-deconvolution CNNs
  • Spatio temporal modeling for different phenomena
    • SST, Precipitation Nowcasting, etc
    • Models: RNN extensions, Generative Models, Physically inspired models

• Type of data used in these papers
  • Satellite data
  • Reanalysis data
  • Simulations e.g. from atmosphere models
Deep-Learning models used for problems in climate modeling
Convolutional nets

• ConvNet architecture (Y. LeCun since 1988)
  • Deployed e.g. at Bell Labs in 1989-90
  • Character recognition
  • Convolution: non linear embedding in high dimension
  • Pooling: average, max

Fig. LeCun
Convolutions and Pooling

• Convolution, stride 1, from 3x3 image to 2x2 image, 2x2 filter

\[
\begin{array}{ccc}
  x_1 & x_2 & x_2 \\
  x_4 & x_5 & x_6 \\
  x_7 & x_8 & x_9 \\
\end{array}
\]

\[
\begin{array}{cc}
  w_1 & w_2 \\
  w_3 & w_4 \\
\end{array}
\]

Filter

\[
y_1 = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4
\]

• Pooling
  • Max pooling, stride 2

\[
\begin{array}{cccc}
  4 & 1 & 7 & 4 \\
  3 & 5 & 1 & 2 \\
  0 & 1 & 3 & 4 \\
  2 & 1 & 1 & 4 \\
\end{array}
\]

\[
\begin{array}{cc}
  5 & 7 \\
  2 & 4 \\
\end{array}
\]
Transpose convolution

• This is the reverse operation –
  • From 2x2 image to 3x3 image, 2x2 filter, Stride 1 with Padding

• Unpooling
  • Reverse pooling operation
  • Different solutions
Convolutional Nets
ResNet (He et al. 2016)

• 152 ResNet 1st place ILSVRC classification competition
• Other ResNets 1st place ImageNet detection, 1st place ImageNet localization, MS-COCO detection and segmentation
• Building block
  • Identity probably helps propagating gradients
  • $F(x)$ is called the residual

• General architecture
  • Mainly 3x3 convolutional filters

Fig. from (He 2016)
Spatial transformer networks (Jaderberg 2015)

- Proposed initially as a module for learning image transformations
  - Such as: cropping, rotations, etc
  - Differentiable module that allows image warping
    - This is the interesting mechanism for us
    - Adaptations are used e.g. in de Bezenac 2018: implements advection mechanism

- Illustration (Fig. from (Jaderberg 2015))
Spatial transformer networks
(Jaderberg 2015)

• STN implements a pointwise image transformation
• All the parameters are learned
• 2 main components
  • Sampling mechanism
    • For each target point \((x_t^i, y_t^i)\), sample a source point \((x_s^i, y_s^i)\)
    • \(T_\theta\) is a learned transformation, with parameters \(\theta = F(I_S)\), \(F\) is a NN, \(I_S\) is the Source Image
Spatial transformer networks (Jaderberg 2015)

• 2 main components
  • Transformation (warping mechanism)
    • For each sampled source point \((x_i^s, y_i^s)\), compute the value of the corresponding target point \((x_i^t, y_i^t)\)
    • Apply a kernel transformation centered on the source point \((x_i^s, y_i^s)\)
      \[
      I_T(x_i^t, y_i^t) = \sum_{(x,y) \in I_S} I_S(x, y) k(x - x_i^s, y - y_i^s)
      \]
      
      \(I(x, y)\) pixel intensity at \((x, y)\)
Recurrent neural networks - RNNs

- Basic architecture: state space model

- Up to the 90s RNN were of no practical use, too difficult to train
- Mid 2000s successful attempts to implement RNN
  - e.g. A. Graves for speech and handwriting recognition
- Today
  - RNNs SOTA for a variety of applications e.g., speech decoding, translation, language generation, etc
Google Neural Machine Translation System
(Wu et al 2016)

• **General Architecture**
  
  **Encoder:** 8 stacked LSTM RNN + residual connections

  **Decoder:** 8 stacked LSTM RNN + residual connections + Softmax output layer

  ![Diagram](image)

  **Attention mechanism**

  Figure from Wu et al. 2016

  • NMT seminal papers: Cho et al. 2014, Sutskever et al. 2014
  • Comparison and evaluation of NMT RNNs options (Fritz et al. 2017)
  • 250 k-hours GPU -> a 250 k$ paper !
Generative Adversarial Networks (Goodfellow 2014)

Generative models intuition

• Provided a sufficiently powerful model $F(z)$
  • It should be possible to learn complex mappings from latent space to real world spaces such as:
Generative Adversarial Networks (Goodfellow 2014)
Generative models intuition

- Given a probability distribution on the latent space $p_z(z)$, $G$ defines a probability distribution on the observation space $p_x(x)$
Generative Adversarial Networks (Goodfells 2014) 
Generative models intuition

- Generative latent variable model

\[ \theta \quad \begin{array}{c} \text{Z} \\ \rightarrow \text{x} \end{array} \]

- Given a simple distribution \( p(z) \), e.g. \( z \sim \mathcal{N}(0, I) \), use a NN to learn a possibly complex mapping \( p_\theta(x|z) = G(z) \)
GANs (Goodfellow, 2014)

• Principle
  • A generative network generates data after sampling from a latent distribution
  • A discriminant network tells if the data comes from the generative network or from real samples
  • The two networks are trained together
    • The generative network tries to fool the discriminator, while the discriminator tries to distinguish between true and artificially generated data
    • Formulated as a MinMax game
  • Hope: the Discriminator will force the Generator to be clever

• Applications
  • Data generation, Semi-supervised learning, super resolution, ...
GANs

- Discriminator is presented alternatively with true \((x)\) and fake \((\hat{x})\) data

\[ x \sim p_{data}(x) \]

\[ z \sim p(z) \]

\[ p(x|z) \]

\[ G(z) \]

\[ D(x) \]

1 if \(x\)

0 if \(\hat{x}\)

\[ G \] and \(D\) are typically MLPs
GAN Training

- Algorithm alternates between optimizing $D$ and $G$
GANs examples Deep Convolutional GANs (Radford 2015) - Image generation

• LSUN bedrooms dataset - over 3 million training examples
Gan example
MULTI-VIEW DATA GENERATION WITHOUT VIEW SUPERVISION (Chen 2018)

- **Objective**
  - Generate images by disantangling content and view
    - Eg. Content 1 person, View: position, illumination, etc
  - 2 latent spaces: view and content
    - Generate image pairs: same item with 2 different views
    - Learn to discriminate between generated and real pairs

![Diagram showing multi-view data generation](image)
Adversarial training: video sequence prediction

- Video prediction, (Mathieu et al. 2016)

- Predicting video future segmentations (Luc et al. 2017 << LJK Grenoble)

Figure 1: Our models learn semantic-level scene dynamics to predict semantic segmentations of unobserved future frames given several past frames.
Examples of Deep Learning applications in the Climate Domain
Event Detection

Eddy detection

Extreme weather event detection
Eddy Identification and Tracking (Lguensat 2017)

- Objective: **pixelwise** eddy classification
  - 3 classes: anticyclonic, cyclonic, no Eddy

- Data
  - SSH maps southwest Atlantic (AVISO-SSH)
  - Labeled by PET14 algorithm (Mason 2014)
    - Provides eddy center + speed and contour
    - Transformed into segmentation maps using the speed contour coordinates
    - Speed contour with the highest mean geostrophic rotational current
    - Pixels inside each contour is labeled A-eddy, C-eddy, No-eddy
  - 15 years, 1 map/ day, 14 1st years used for training, last year for testing
  - Input = 128x128 patch randomly sampled from the SSH map
    - 5 k training examples
Eddy Identification and Tracking (Lguensat 2017)

- 

Fig. 1: A snapshot of a SSH map from the Southern Atlantic Ocean with the detected eddies by PET14 algorithm, red shapes represent anticyclonic eddies while green shapes are cyclonic eddies.

Fig. 2: Example of a SSH-Segmentation training couple, anticyclonic (green), cyclonic (brown), non eddy (blue).

Patch sampling

Fig. from (Lguensat 2017)
Eddy Identification and Tracking (Lguensat 2017)

• Model
  • Convolution-Deconvolution architecture
    • Inspired from CNN for biomedical image segmentation

• Task: classification

• Training criterion
  • Cross Entropy
  • Dice-Loss = 1 – mean-softDiceCoef (better reflects segmentation...)
    • $softDiceCoef(P, T) = \frac{2 \sum_i p_i t_i}{\sum_i p_i + \sum_i t_i}$
    • $P$: predicted output (matrix), $T$: Target output (matrix)
      • T: one hot encoding (3 D) for each position, P: also 3 D for each position ($p_i \in [0,1]$)
      • $p_i$: predicted probability, $t_i = 1$ for correct label, 0 otherwise
    • mean-softDiceCoef: mean for the 3 coefficients
    • $softDiceCoef(P, T)$ should be 1 for perfect segmentation, 0 for completely mistaken segmentation
Eddy Identification and Tracking (Lguensat 2017)

Fig. 3: EddyNet architecture
Eddy Identification and Tracking (Lguensat 2017)

• Experiments
  • 2 variants of the network

• Code available, data available
• Mentioned extensions
  • 3D altimetry with 3D CNNs
  • SST as additional inputs
Extreme Weather Event Detection
(Racah 2017)

• Objective: detection of local events from earth observation
  • 4 classes: tropical depressions, tropical cyclones, extra tropical cyclones, atmospheric rivers

• Data
  • Simulated data from CAM5, a 3 D physical model of the atmosphere.
    • Generates 768x1152 images (8) per day, each with 16 channels !! (Channels: Surface temp, surface pressure, etc), for 27 years
  • Labeled with TECA (Toolkit for Extreme Climate Analysis)
    • Produces: event center coordinates in the image, bounding box for the event, event class
    • Prone to errors, + imbalanced event classes

• Method
  • Convolution-Deconvolution NN + supervision for predicting event localization, size and class
Extreme Weather Event Detection (Racah 2017)

- Model: 3D Conv – Deconv NN

Fig. from (Racah 2017)

Input image is split into a 12x18 grid of 64x64 pixels

- Location/ size of object
- Object present in the grid Y/N
- Object class

Reconstruction error
Extreme Weather Event Detection (Racah 2017)

- Exemple

Figure 3: Bounding box predictions shown on 2 consecutive (6 hours in between) simulation frames, for the integrated water vapor column channel. Green = ground truth, Red = high confidence predictions (confidence above 0.8). 3D supervised model (Left), and semi-supervised (Right).
Spatio-temporal modeling

Nowcasting
Integration of NN in numerical models
Incorporating prior physical knowledge in Deep learning models
Solving inverse problems with NNs
Precipitation Nowcasting
(Shi 2015, Shi 2017, Zhang 2017)

• Precipitation Nowcasting
  • Very short term (some hours) prediction of rainfall intensity in a local region

• Classical methods
  • Numerical Weather Prediction (NWP) methods: based on physical equations of an atmosphere model
  • Extrapolation based methods using radar data
    • Optical flow based methods inspired from vision
    • Does not fully exploit available data (Shi 2015)

• Objective
  • Learning from spatio temporal series of radar measures
    • k-step prediction
    • End to end learning

• Data
  • Local radar maps
Precipitation Nowcasting (Shi 2015)

• **Model**
  • Extension of LSTM by incorporating explicit spatial dependencies
    • ConvLSTMs
    • Inspired from early video prediction models
      • Analogy with the video prediction tasks but on dense images
      • Note: several recent papers for video prediction with NN (without optical Flow)
    • convolutions both for input to state and state to state connections

![Figure 2: Inner structure of ConvLSTM](image)

![Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting](image)
Precipitation Nowcasting
(Shi 2015)

- Data
  - Radar reflectivity maps from 97 rainy days in Hong Kong
    - 1 radar map every 6 mn, 240 frames per day
    - Small dataset
  - Radar map preprocessed into 100x100 grayscale « image » + smoothing
    - Sequences = 20 successive frames, 5 as input, 15 as prediction

- Model
  - 2 layers ConvLSTM
  - Training criterion: Cross-Entropy (rain/ no rain ?????) or MSE + thresholding ?

- Evaluation
  - Several measures
    - MSE is measured on the predicted values (regression)
    - The other measures require binary decisions: rain vs no rain, the predicted values are converted to 0/1 using a threshold of 0.5 mm/h rainfall rate
    - Rover is an optical flow based method

- Lessons
  - State to state convolutions are essential for handling spatio-temporal dependencies
  - Better than ROVER (sota Optical Flow based method) and Full LSTM

Table 2: Comparison of the average scores of different models over 15 prediction steps.

<table>
<thead>
<tr>
<th>Model</th>
<th>Rainfall-MSE</th>
<th>CSI</th>
<th>FAR</th>
<th>PDR</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvLSTM(3x3)-3x3-64</td>
<td>1.420</td>
<td>0.577</td>
<td>0.195</td>
<td>0.660</td>
<td>0.908</td>
</tr>
<tr>
<td>Rover1</td>
<td>1.712</td>
<td>0.516</td>
<td>0.308</td>
<td>0.636</td>
<td>0.843</td>
</tr>
<tr>
<td>Rover2</td>
<td>1.684</td>
<td>0.522</td>
<td>0.301</td>
<td>0.642</td>
<td>0.850</td>
</tr>
<tr>
<td>Rover3</td>
<td>1.685</td>
<td>0.522</td>
<td>0.301</td>
<td>0.642</td>
<td>0.809</td>
</tr>
<tr>
<td>FC-LSTM-2000-2000</td>
<td>1.865</td>
<td>0.236</td>
<td>0.335</td>
<td>0.351</td>
<td>0.774</td>
</tr>
</tbody>
</table>
Precipitation Nowcasting (Shi 2017)

• Extension of the ConvLSTM work
  • Based on GRUs
  • Main ideas
    • Use convolution GRUs instead of fully connected GRUs: ConvGRU
    • The spatial dependency structure between states should be context dependent and not fixed like in ConvLSTMs
    • They consider a spatial context
    • Basic unit is called TrajGRU

• New and larger dataset

• New evaluation metrics (weighted MSE)
Precipitation Nowcasting (Shi 2017)

- Selection of neighborhood at time $t$ (Warping mechanism)
  - For cell $(i,j)$ in $H_t$ select neighborhood cells at $H_{t-1}$
  - Function $\gamma(X_t, H_{t-1})$ generates a bilinear mapping which is then used to select points in $H_{t-1}$

Figure 1: Example of the encoding-forecasting structure used in the paper. In the figure, we use three RNNs to predict two future frames $I_3, I_4$ given the two input frames $I_1, I_2$. The spatial coordinates $G$ are concatenated to the input frame to ensure the network knows the observations are from different locations. The RNNs can be either ConvGRU or TrajGRU. Zeros are fed as input to the RNN if the input link is missing.

Figure 2: Comparison of the connection structures of convolutional RNN and trajectory RNN. Links with the same color share the same transition weights. (Best viewed in color)
Precipitation Nowcasting
(Shi 2017)

• Dataset: HKO-7
  • Echo radar data from 2009 to 2015 in Hong Kong
  • 1 radar map every 6 mn, 240 frames per day
  • Resolution 480x480 pixels, altitude 2 km, cover 512x512 km in Hong Kong
  • Radar images are transformed to (0, 255) pixel values + filtering
  • Rainy days: 812 days for training, 50 for validation, 131 for test
  • Prediction: radar reflectivity values are converted to rainfall intensity values

• Model
  • 3 layer Encoding – Forecasting model
  • Training criterion: weighted MSE (higher weights for heavy rainfall – compensates for data imbalance – see next slide)

• Evaluation
  • MSE and weighted MSE (regression)
  • Different measures requiring a binary decision: rain or no rain
    • Evaluation is performed at different threshold values 0.5, 5, 10, 30
    • Predicted pixel values are converted to 0/1 values for each threshold
    • Scores are computed for each threshold
Precipitation Nowcasting (Shi 2017)

- Rain statistics (dataset)

- Performance comparison

<table>
<thead>
<tr>
<th>Rain Rate (mm/h)</th>
<th>Proportion (%)</th>
<th>Rainfall Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>No/ Hardly noticeable</td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td>Light</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Light to moderate</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Moderate</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Moderate to heavy</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Rainstorm warning</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: HKO-7 benchmark result. We mark the best result within a specific setting with **bold face** and the second best result by underlining. Each cell contains the mean score of the 20 predicted frames. In the online setting, all algorithms have used the online learning strategy described in the paper. ‘†’ means that the score is higher the better while ‘‘’ means that the score is lower the better. ‘r ≥ τ’ means the skill score at the τ mm/h rainfall threshold. For 2D CNN, 3D CNN, ConvGRU and TrajGRU models, we train the models with three different random seeds and report the mean scores.
Precipitation Nowcasting (Zhang 2017)

• Number of preliminary analyses, e.g. (Zhang 2017)
  • Nowcasting based on
    • 3-D Radar maps – multiple altitudes
    • Reanalysis data from VDRAS (NCAR US)
    • Classification: rain/ no rain
      • Vertical velocity and buoyancy of an air parcel (also 3-D data)
• Objective: nowcasting, storm initiation and growth (*)
  • Argument: radar data not sufficient for (*)

Figure 1: Network Architecture of multi-channel 3D-SCN
Integration of NN in numerical models (Brajard 2018)

• Can Machine Learning (ML) techniques be used in weather and climate models to replace physical forcings

• Example

\[
\begin{align*}
\partial_t u &= + (f + \zeta) u - \partial_x \left( \frac{u^2 + v^2}{2} + g^* h \right) + \theta_u \\
\partial_t v &= - (f + \zeta) v - \partial_y \left( \frac{u^2 + v^2}{2} + g^* h \right) + \theta_v \\
\partial_t h &= - \partial_x (u(H + h)) - \partial_y (v(H + h)) \\
\end{align*}
\]

where \( \zeta \) is the vorticity.
\( \theta_u \) and \( \theta_v \) account for the effects of forcing, dissipation and diffusion.

More generally, the forcing terms mimic unresolved processes like turbulence, precipitation, radiation, clouds, friction, etc. Typically computed via complicated physical parameterizations with empirical parameters

• Question: can \( \theta(t) \) be represented by a neural network \( F(x(t)) \)?
Integration of NN in numerical models (Brajard 2018)

• Proof of concept
  • Data generated by a fully specified shallow water model
    • i.e. the $\theta$s are modeled by a physical model
  • Train a MLP to learn the $\theta$s, supervised learning

$u$: speed  
$h$: heigth of mixture level  
$\tau_x$: surface wind
Integration of NN in numerical models (Brajard 2018)

• The neural network simulation diverges after a few hundred days (kinetic and potential energy explode)

![Image of neural network simulation](image1.png)

• Solution: add a mass conservation constraint (hmean = constant) to the neural network training algorithm (physics-informed machine learning)

![Image of mass conservation constraint](image2.png)
Incorporating prior knowledge

• Motivations
  • DL SOTA for perception problems
  • Natural physical phenomenon are much more complex than problems handled by Deep Learning today
    • Can we incorporate prior knowledge from physics in statistical models?

• Challenge
  • Interaction between the Physical and the Statistical paradigms

• Illustration: Sea Surface Temperature Prediction
Incorporating prior knowledge - (de Bezenac 2018)

Physical model for fluid transport

Advection – Diffusion equation

• Describes transport of $I$ through **advection** and **diffusion**

\[
\frac{\partial I}{\partial t} + (w \cdot \nabla) I = D \nabla^2 I
\]

  - $I$: quantity of interest (Temperature Image)
  - $w = \frac{\Delta x}{\Delta t}$ motion vector, $D$ diffusion coefficient

• There exists a closed form solution
  - $I_{t+\Delta t}(x) = (k \ast I_t)(x - w(x))$

• If we knew the motion vector $w$ and the diffusion coefficient $D$ we could calculate $I_{t+\Delta t}(x)$ from $I_t$

  - $w$ and $D$ unknown
  - -> Learn $w$ and $D$
Incorporating prior knowledge - (de Bezenac 2018)
Prediction Model
Objective: predict $I_{t+1}$ from past $I_t, I_{t-1}, ...$

- 2 components: Convolution-Deconvolution NN for estimating motion vector $w_t$

  - Warping Scheme
    - Implements discretized Advection-Diffusion solution

- Past Images
- Target image
- End to End learning using only $I_{t+1}$ supervision
- Stochastic gradient optimization
- Performance on par with SOTA assimilation models
Solving inverse problems with NNs (de Bezenac et al. ongoing work)

• **Objective**
  • Given noisy observed data, and possibly some priors how to generate an approximation of the underlying true data?
  • Priors may come from a physical model

• **Applications**
  • Improve physical model predictions using observed data
  • Inpainting for physical data

• **Method**
  • Based on an extension of amiant GANs (Bora et al. 2018)
Solving inverse problems with NNs (de Bezenac et al. ongoing work)

- **Ambiant GANs (Bora et al. 2018)**
  - Train generative models from incomplete or noisy samples
  - Hyp: the noise/ measurement process is known
    - Works for some classes of measurements (theoretical results for kernels + noise distributions – empirical results for large class of processes)
  - The NN is trained to distinguish a real measurement from a simulated measurement of a generated image

Fig. from Bora et al. 2018
Solving inverse problems with NNs (de Bezenac et al. ongoing work)

- AmbiantGAN example

![Image](image.png)

Fig. 2: (Left) Samples of lossy measurements used for training. Samples produced by (middle) a baseline that trains from inpainted images, and (right) our model.
Solving inverse problems with NNs
(de Bezenac et al. ongoing work)

• Conditional amiant GANs
  • Objective
    • Given a stochastic measurement process model $F_\theta$ learn $\hat{X}$ so that $\hat{Y}$ is indistinguishable from $Y$
Solving inverse problems with NNs
(de Bezenac et al. ongoing work)

- Preliminary illustrations
  - Data from Shallow Water model
    - Left: 90% pixels eliminated (0) + noise $N(0,1)$ on remaining pixels
    - Right: « clouds »

<table>
<thead>
<tr>
<th>True State</th>
<th>Observation</th>
<th>GAN model</th>
<th>BLUE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>True State</th>
<th>Observation</th>
<th>GAN model</th>
</tr>
</thead>
</table>

2018-05-25 Deep Learning for Climate 53
NN as Dynamical Systems
NN as Dynamical Systems

- Recent papers on the interpretation of NNs as discretization schemes for differential equations
  - Links between data driven approaches (NNs) and physical models used in climate modeling
    - Allows learning efficient discretization schemes for unknown ODE
  - Motivates the alternative design of NN modules/architectures
  - Not yet a clear application to climate pb.
Resnet as a discretization scheme for ODEs

• ODE
  \[
  \frac{dX}{dt} = F(X(t), \theta(t)), \quad X(0) = X_0 \tag{1}
  \]

• Resnet module
  \[
  X_{t+1} = X_t + G(X_t, \theta_t) \tag{2}
  \]
  \[
  X_{t+1} = X_t + hF(X_t, \theta_t), \quad h \in [0,1]
  \]
  \[
  \frac{X_{t+1} - X_t}{h} = F(X_t, \theta_t)
  \]
  • Forward Euler Scheme for the ODE
  • \( h \) time step

• Note: this type of additive structure (2) is also present in LSTM and GRU units

• Resnet
  • Input \( X_t \), output \( X_{t+1} \)
  • Multiple Resnet modules implement a multi-step discretization scheme for the ODE
  \[
  X(t_1) = X(t_0) + hF(X(t_0), \theta_{t_0})
  \]
  \[
  X(t_2) = X(t_1) + hF(X(t_1), \theta_{t_1}), \ldots
  \]
Resnet as a discretization scheme for ODEs

- Alternative discretization schemes correspond to alternative Resnet like NN models
  - Backward Euler, Runge-Kutta, linear multi-step ...
- Example (Lu 2018) linear multi-step discretization scheme
  - $X_{t+1} = (1 - k_t)X_t + k_tX_{t-1} + F(X_t, \theta_t)$

Fig. (Lu 2018)

- Applications
  - Classification (a la ResNet)
  - Modeling dynamical systems
    - (Fablet 2017) Runge Kutta for dynamical systems, Toy problems
References


