

# Deep-Learning for Climate

25-05-2018 – SAMA Machine Learning

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

\*Sorbonne Université, 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

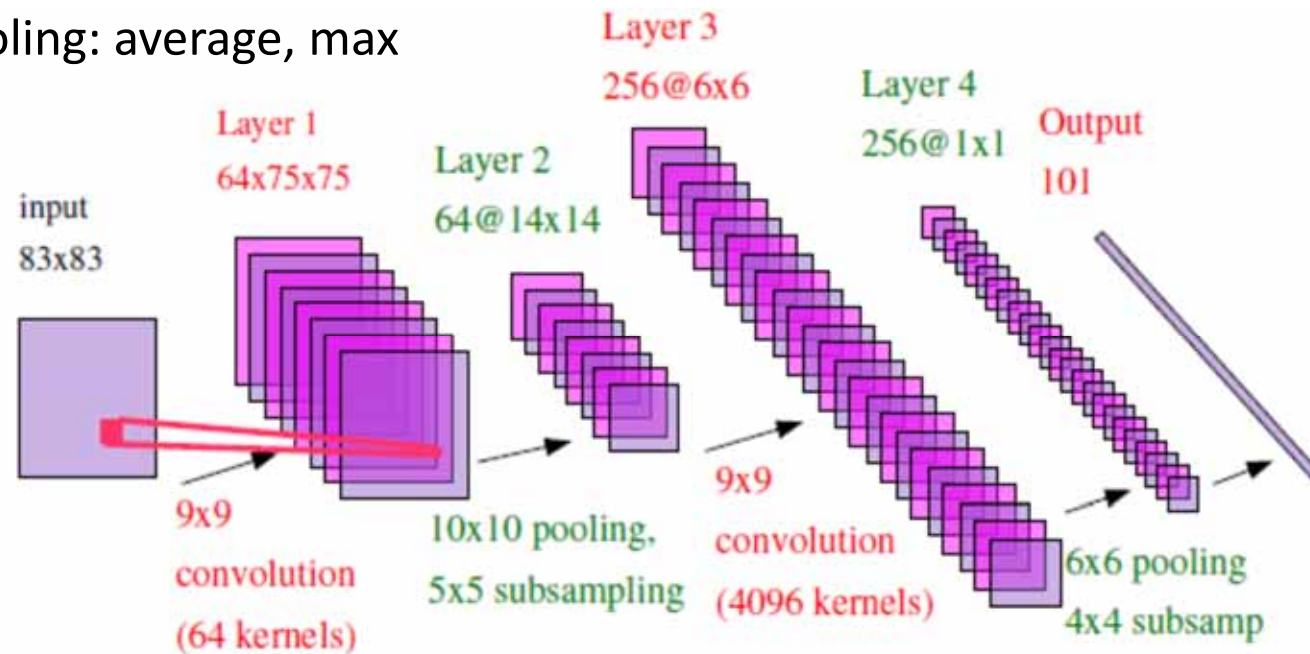
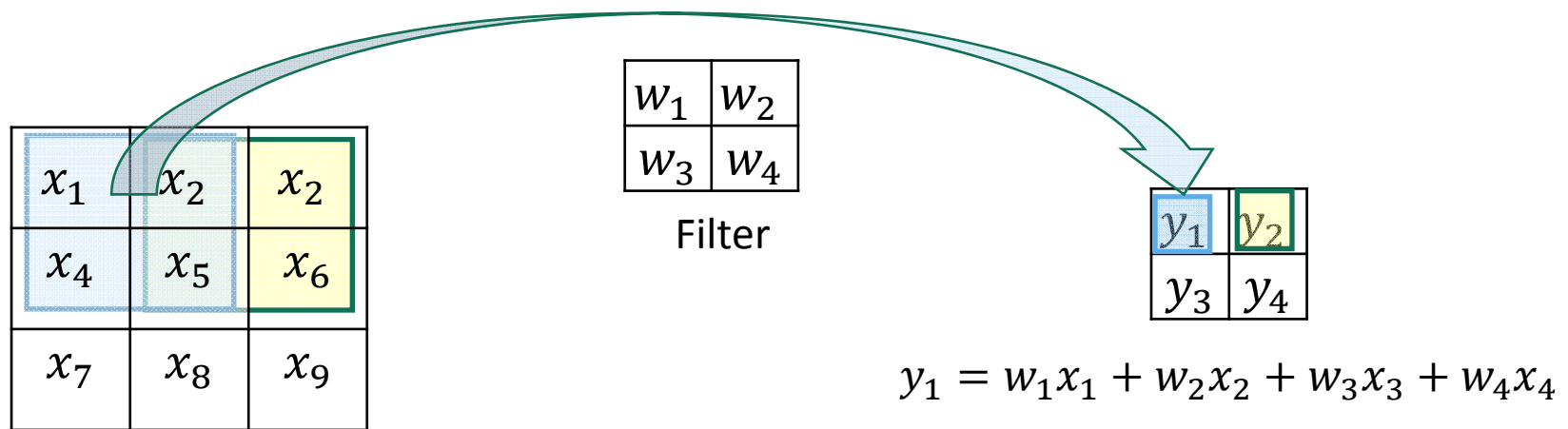


Fig. LeCun

# Convolutions and Pooling

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

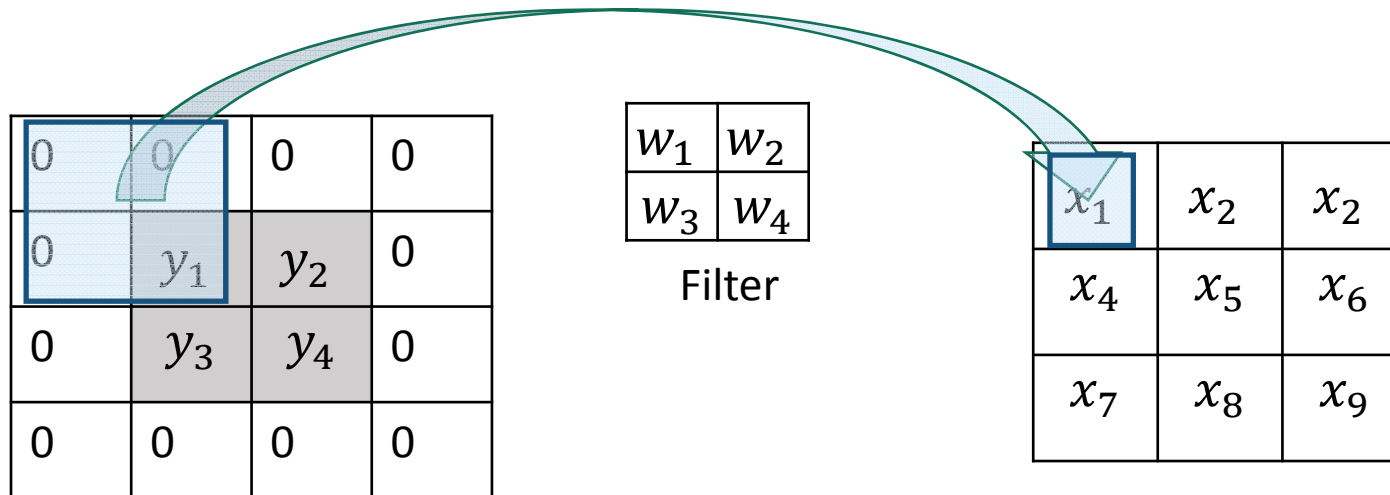


- Pooling
  - Max pooling, stride 2



# 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 gradient
  - $F(x)$  is called the residual

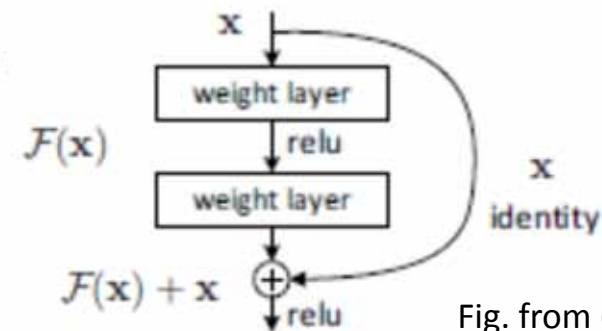
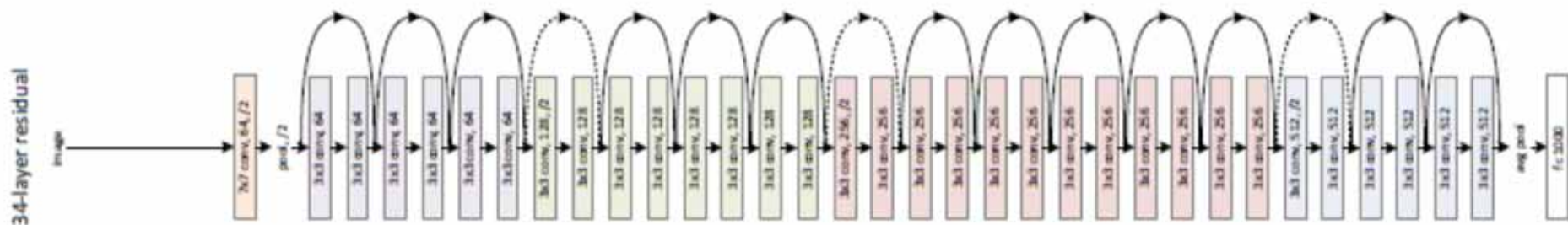


Fig. from (He 2016)

- General architecture
  - Mainly 3x3 convolutional filters



# 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))

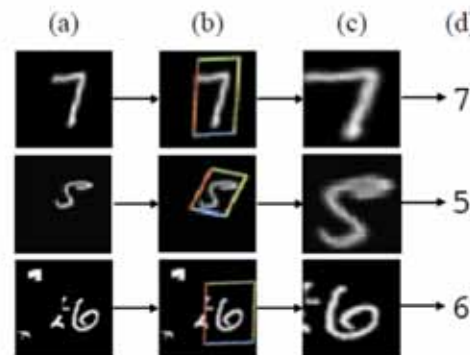
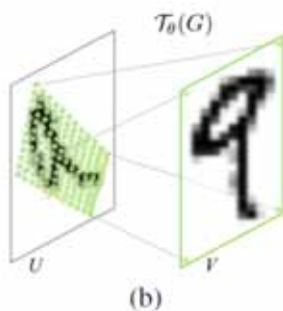
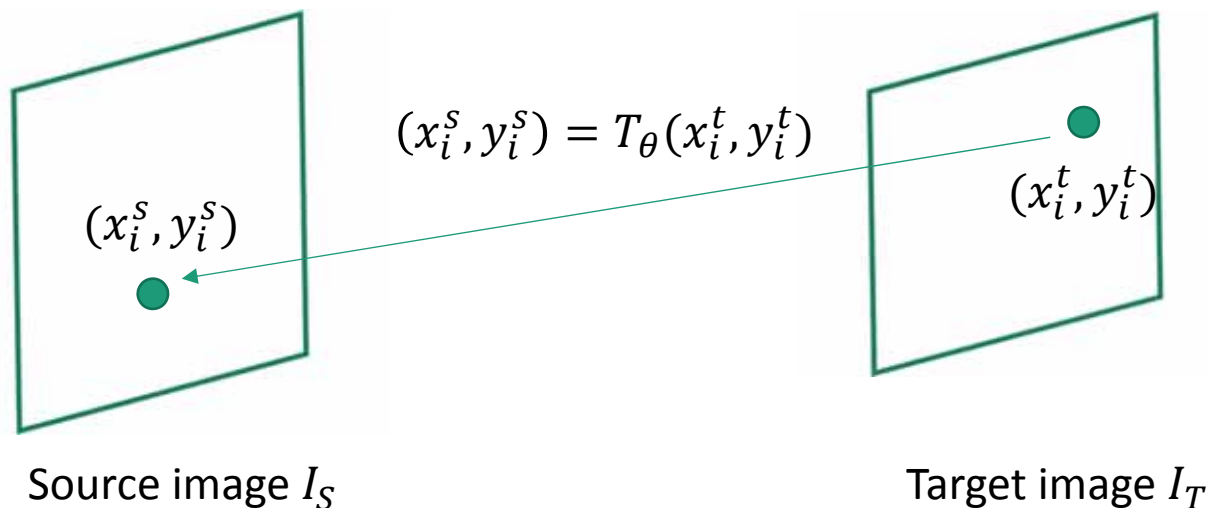


Figure 1: The result of using a spatial transformer as the first layer of a fully-connected network trained for distorted MNIST digit classification. (a) The input to the spatial transformer network is an image of an MNIST digit that is distorted with random translation, scale, rotation, and clutter. (b) The localisation network of the spatial transformer predicts a transformation to apply to the input image. (c) The output of the spatial transformer, after applying the transformation. (d) The classification prediction produced by the subsequent fully-connected network on the output of the spatial transformer. The spatial transformer network (a CNN including a spatial transformer module) is trained end-to-end with only class labels – no knowledge of the groundtruth transformations is given to the system.

# Spatial transformer networks (Jaderberg 2015)

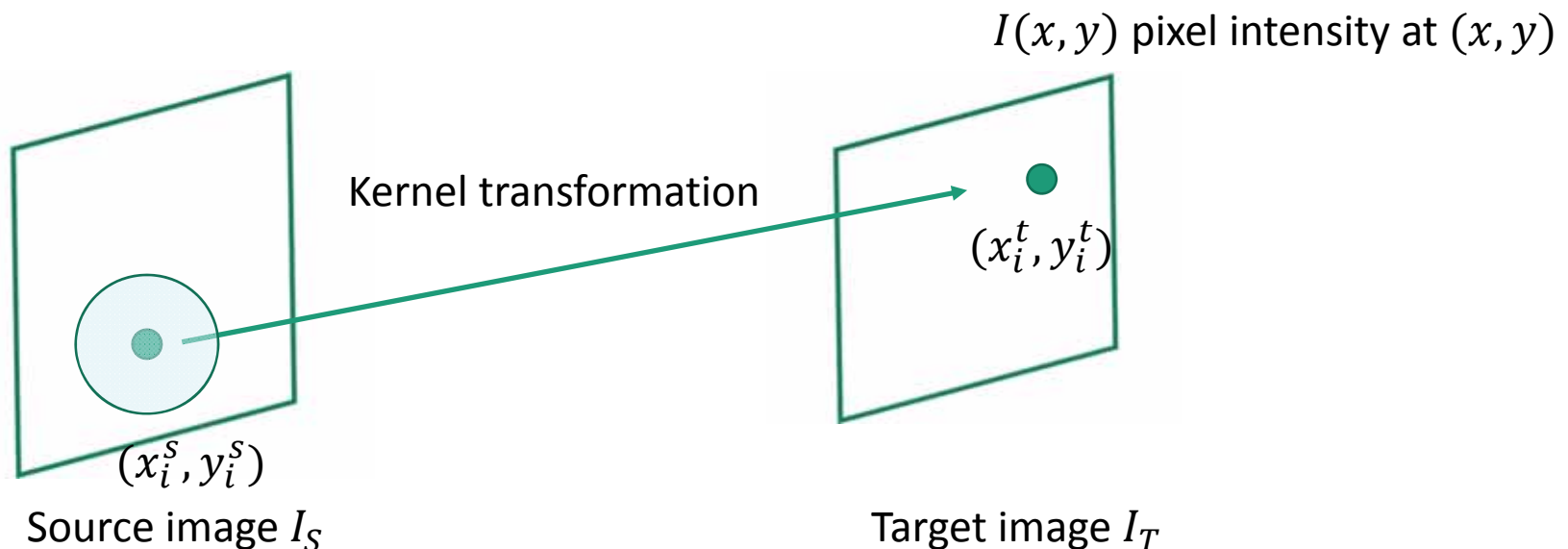
- STN implements a pointwise image transformation
- All the parameters are learned
- 2 main components
  - Sampling mechanism
    - For each target point  $(x_i^t, y_i^t)$ , sample a source point  $(x_i^s, y_i^s)$
    - $T_\theta$  is a learned transformation, with parameters  $\theta = F(I_S)$ ,  $F$  is a NN,  $I_S$  is the Source Image



# Spatial transformer networks (Jaderberg 2015)

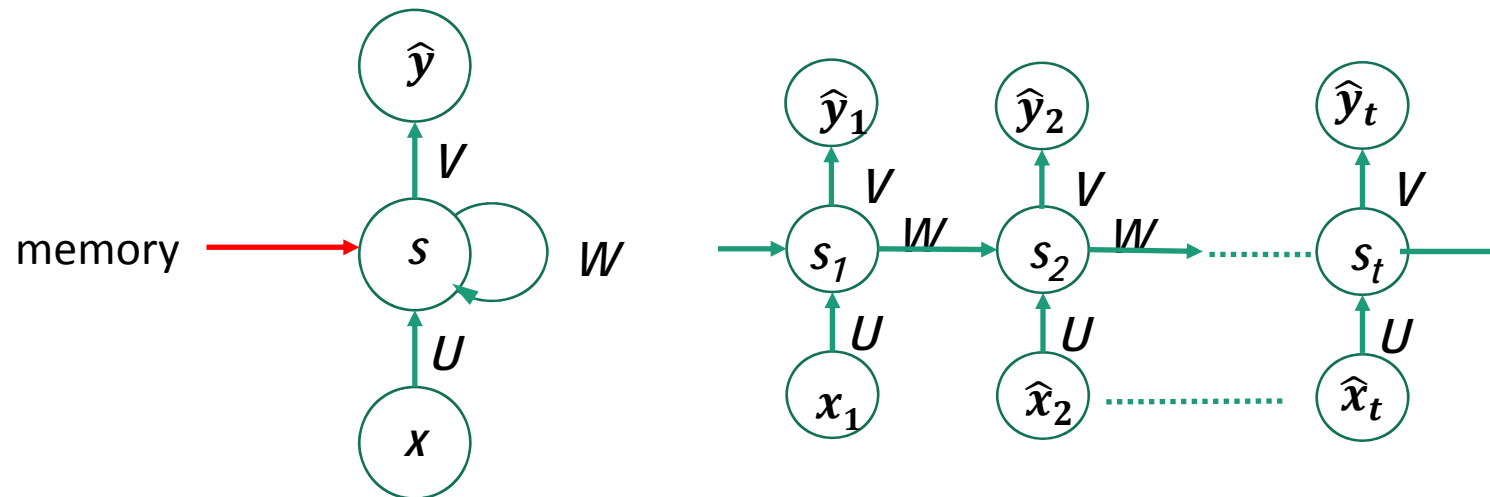
- 2 main components
  - Transformation (warping mechanism)
    - For each sampled source point  $(x_i^s, y_i^s)$ , compute the value of the corresponding target point  $(x_i^t, y_i^t)$
    - Apply a kernel transformation centered on the source point  $(x_i^s, y_i^s)$

$$I_T(x_i^t, y_i^t) = \sum_{(x,y) \in I_S} I_S(x,y) k(x - x_i^s, y - y_i^s)$$



# 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)

<https://research.googleblog.com/2016/09/a-neural-network-for-machine.html>

- General Architecture

Encoder: 8 stacked LSTM RNN  
+ residual connections

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

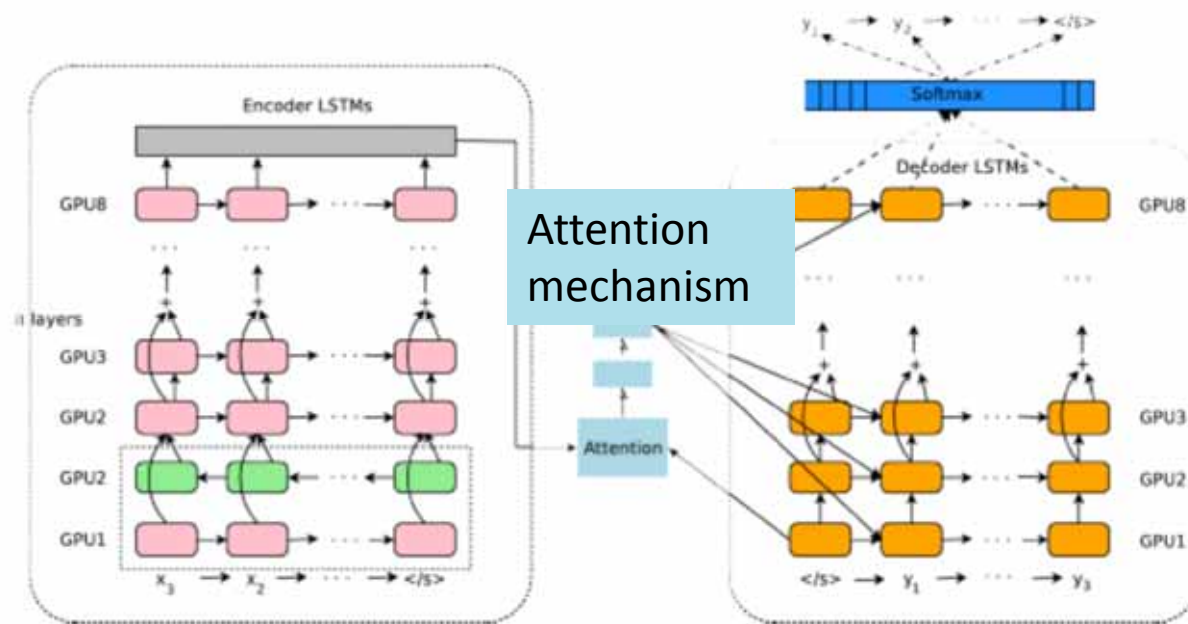


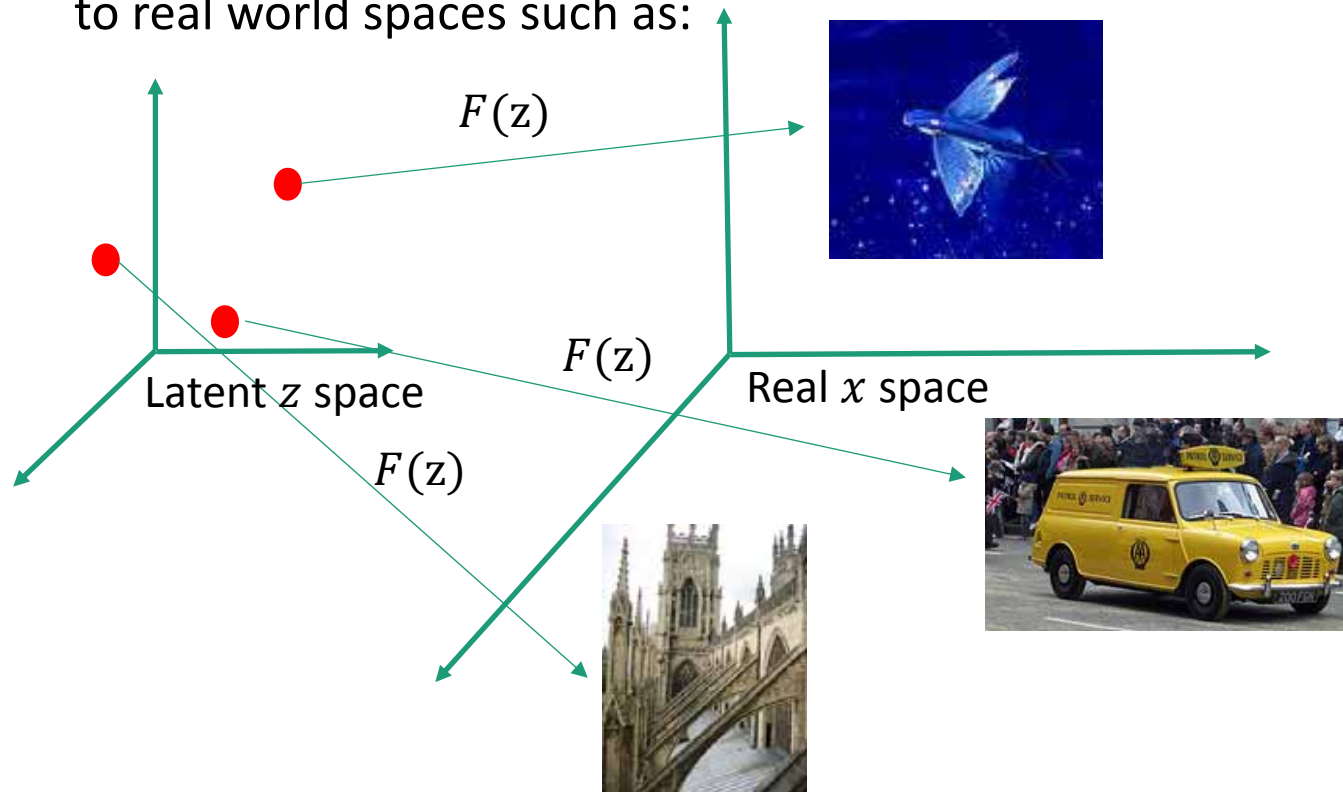
Figure from Wu et al. 2016

- 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

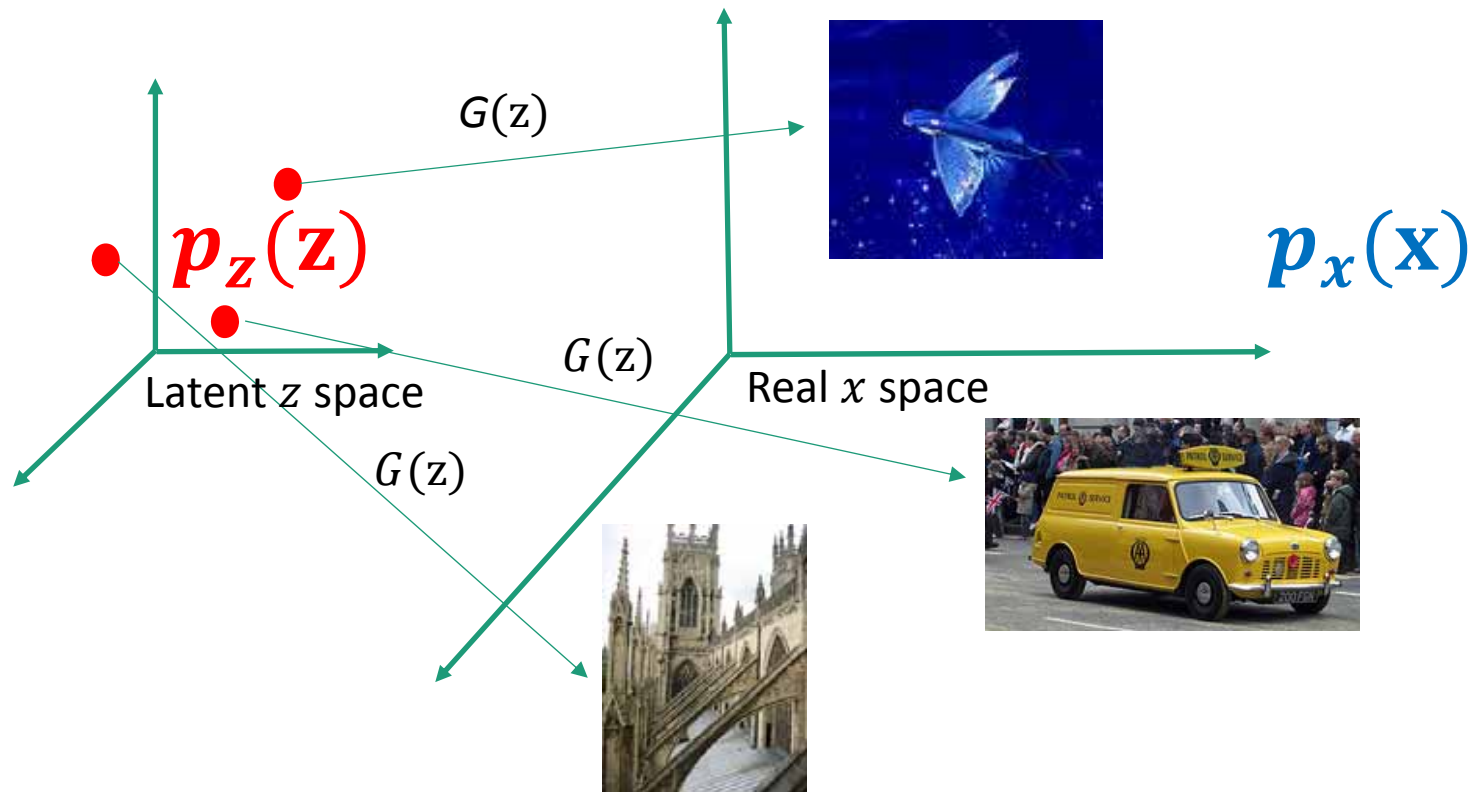
- Provided a sufficiently powerful model  $F(z)$ 
  - It should be possible to learn complex mappings from latent space to real world spaces such as:



# Generative Adversarial Networks (Goodfellow 2014)

## Generative models intuition

- Given a probability distribution on the latent space  $p_z(\mathbf{z})$ ,  $G$  defines a probability distribution on the observation space  $p_x(\mathbf{x})$

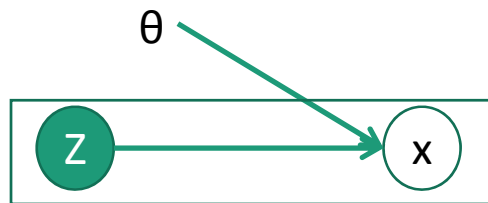




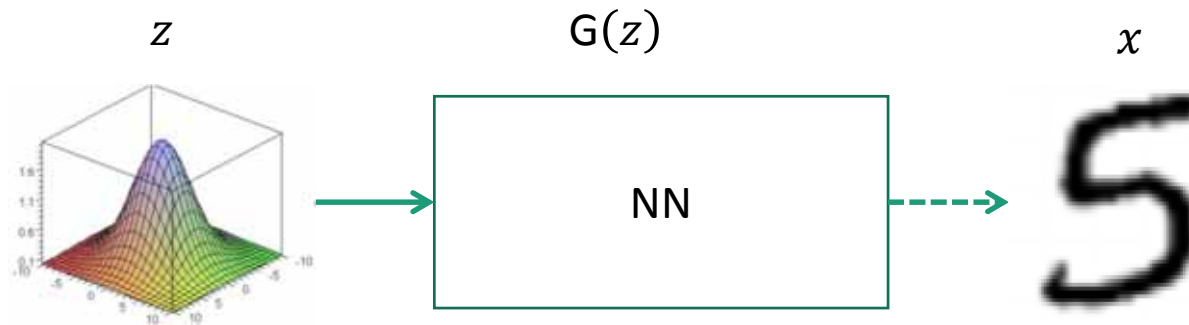
# Generative Adversarial Networks (Goodfellow 2014)

## Generative models intuition

- Generative latent variable model



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

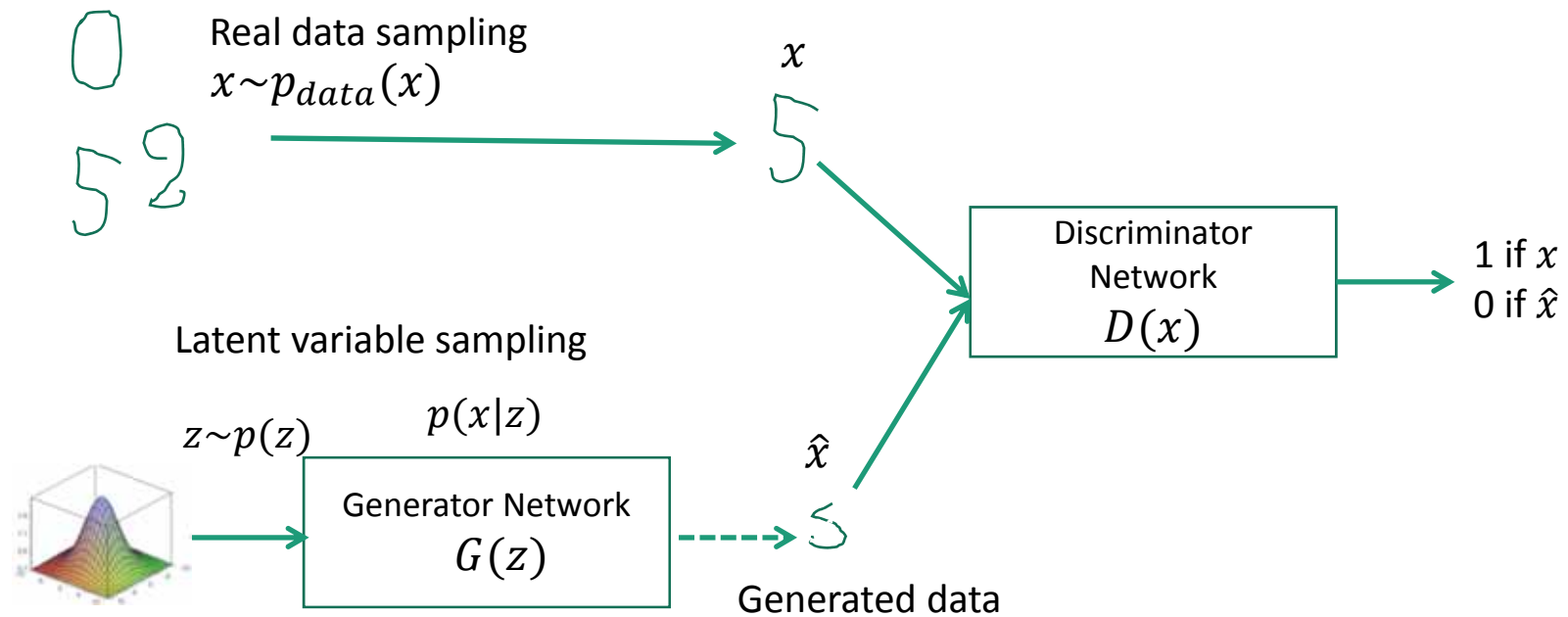


# GANs (Goodfellow, 2014)

- 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

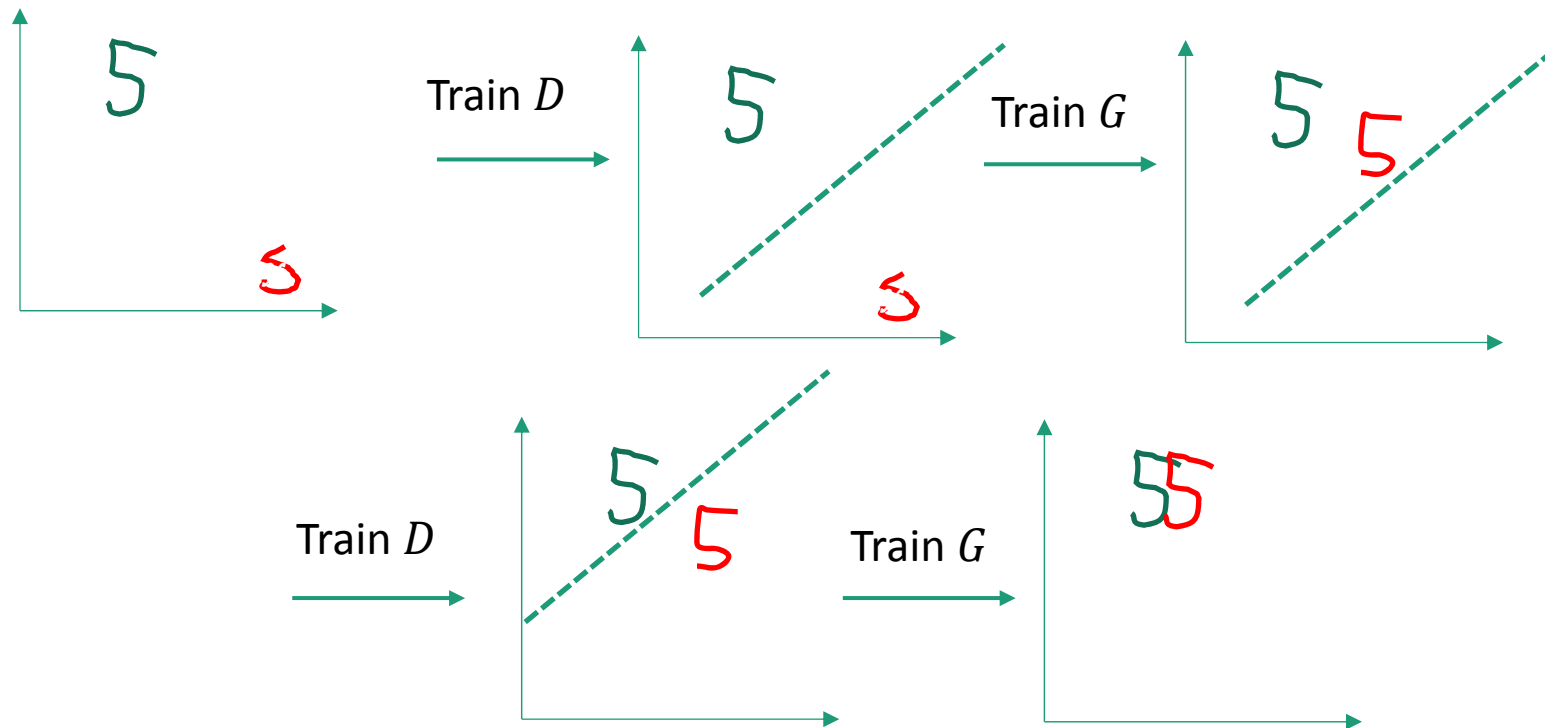
- Discriminator is presented alternatively with true ( $x$ ) and fake ( $\hat{x}$ ) data



$D$  and  $G$  are typically MLPs

# GAN Training

- Algorithm alternates between optimizing  $D$  and  $G$



# GANs examples Deep Convolutional GANs (Radford 2015) - Image generation

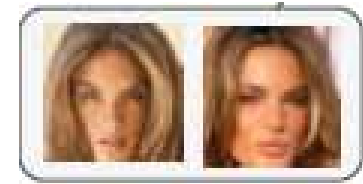
- LSUN bedrooms dataset - over 3 million training examples



Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds. Fig. Radford 2015

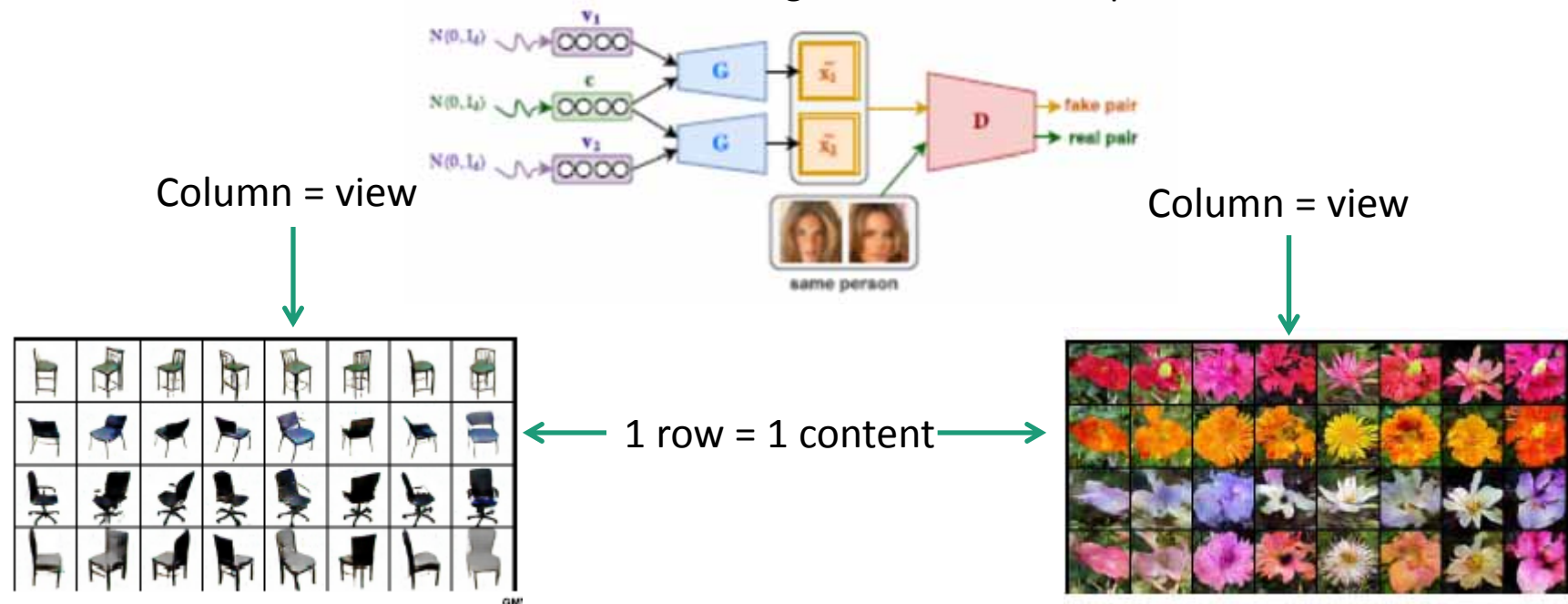
# Gan example

## MULTI-VIEW DATA GENERATION WITHOUT VIEW SUPERVISION (Chen 2018)



- Objective

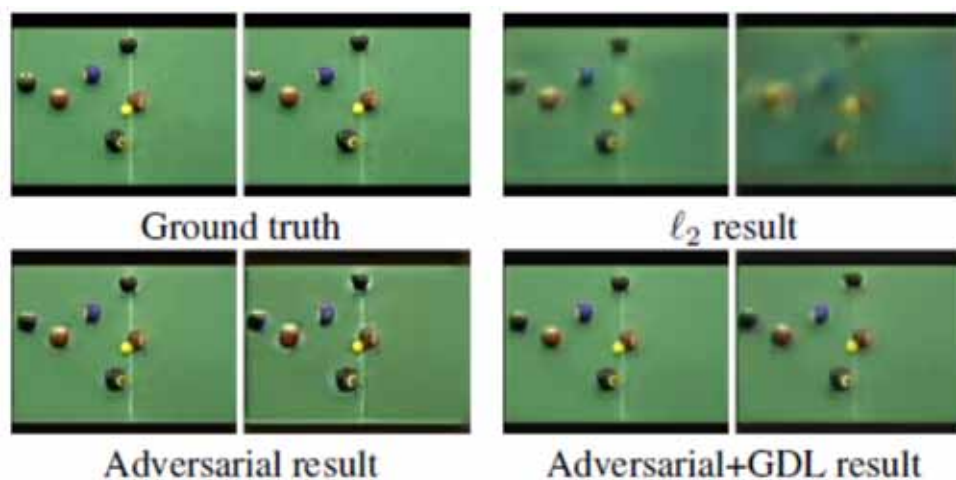
- Generate images by disentangling content and view
  - Eg. Content 1 person, View: position, illumination, etc
- 2 latent spaces: view and content
  - Generate image pairs: same item with 2 different views
  - Learn to discriminate between generated and real pairs



# Adversarial training: video sequence prediction

- 

Video prediction,  
(Mathieu et al. 2016)



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

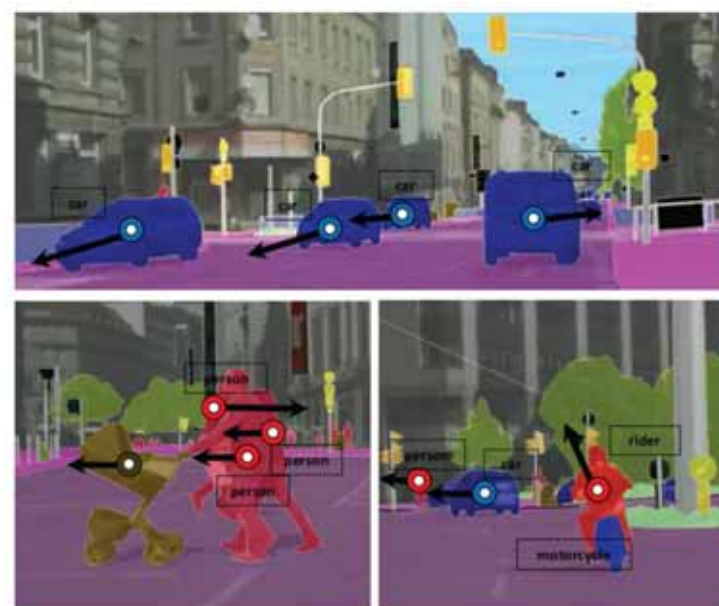


Figure 1: Our models learn semantic-level scene dynamics to predict semantic segmentations of unobserved future frames given several past frames.

# 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)

- 

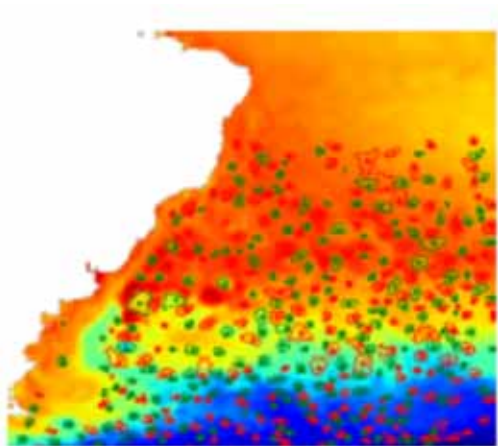


Fig. 1: A snapshot of a SSH map from the Southern Atlantic Ocean with the detected eddies by PET14 algorithm, red shapes represent anticyclonic eddies while green shapes are cyclonic eddies

Patch  
sampling →

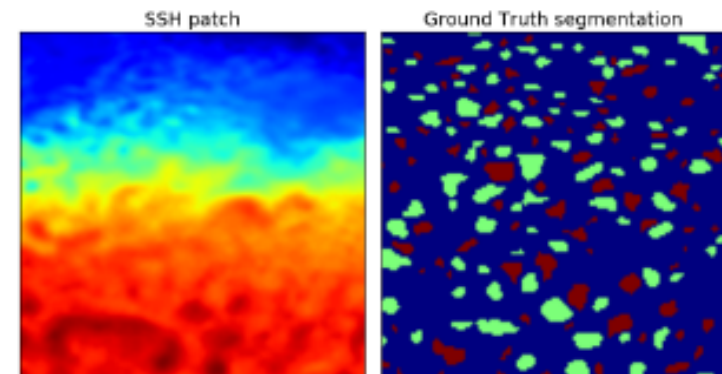


Fig. 2: Example of a SSH-Segmentation training couple, anticyclonic (green), cyclonic (brown), non eddy (blue)

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 =  $1 - \text{mean-softDiceCoef}$  (better reflects segmentation...)
    - $\text{softDiceCoef}(P, T) = \frac{2 \sum_i p_i \cdot t_i}{\sum_i p_i + \sum_i t_i}$
    - $P$ : 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
    - $\text{softDiceCoef}(P, T)$  should be 1 for perfect segmentation, 0 for completely mistaken segmentation

# Eddy Identification and Tracking (Lguensat 2017)

•

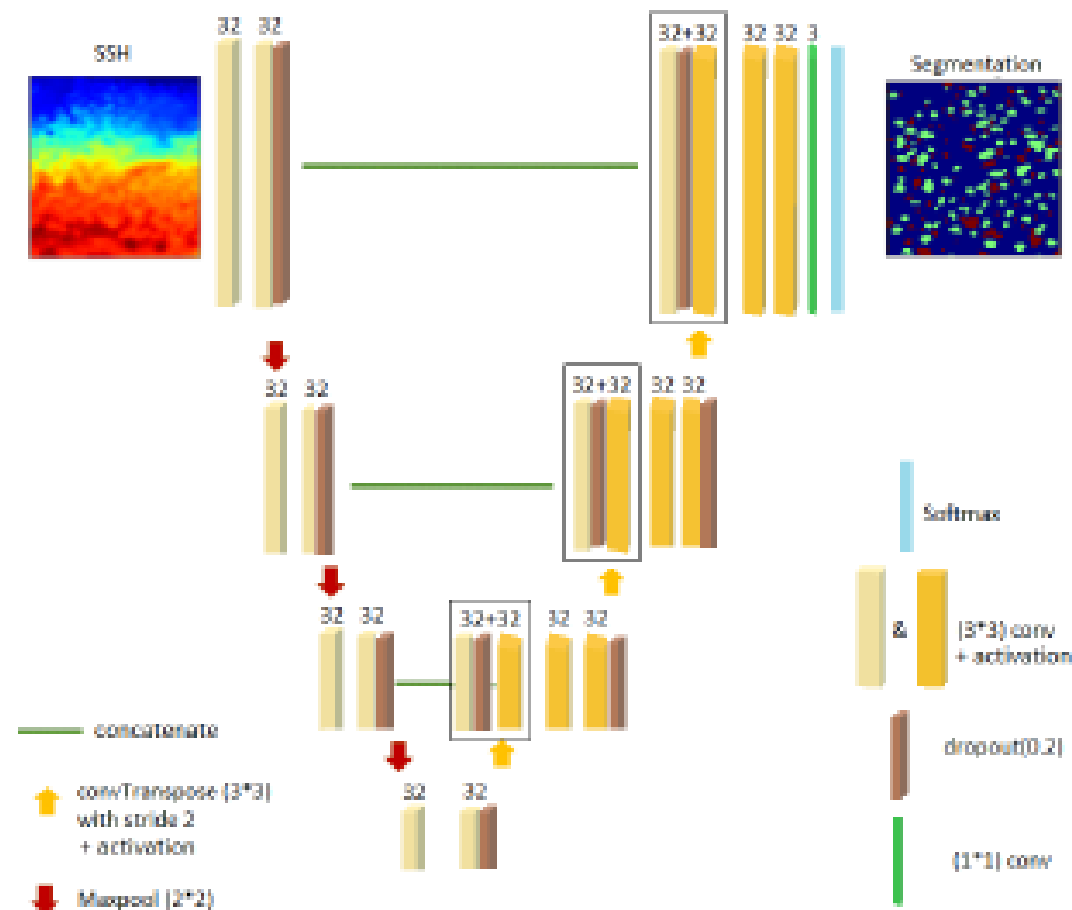


Fig. 3: EddyNet architecture

Fig. from (Lguensat 2017)

# Eddy Identification and Tracking (Lguensat 2017)

- Experiments
  - 2 variants of the network

TABLE I: Metrics calculated from the results of 50 random sets of 360 SSH patches from the test dataset, we report the mean value and put the standard variation between parenthesis.

	#Param	Epoch time	Train loss	Anticyclonic	Cyclonic	Non Eddy		
				Dice Coef			Mean Dice Coef	Global Accuracy
EddyNet	177,571	~12 min	Dice Loss	0.708 (0.002)	0.677 (0.001)	0.929 (0.001)	0.772 (0.001)	88.60% (0.10%)
			CCE	0.695 (0.003)	0.651 (0.001)	0.940 (0.001)	0.762 (0.001)	89.92% (0.07%)
EddyNet <sub>S</sub>		~7 min	Dice Loss	0.694 (0.003)	0.665 (0.001)	0.933 (0.001)	0.764 (0.001)	88.98% (0.09%)
			CCE	0.682 (0.002)	0.653 (0.002)	0.939 (0.001)	0.758 (0.001)	89.83% (0.08%)

- Code available, data available
- Mentionned 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

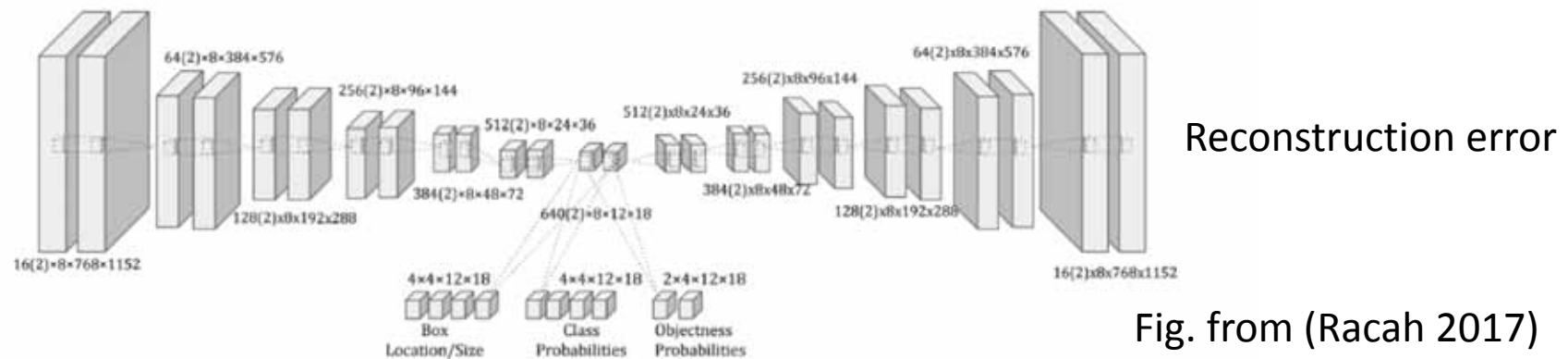


Figure 1: Diagram of the 3D semi-supervised architecture. Parentheses denote subset of total dimension shown (for ease of visualization, only two feature maps per layer are shown for the encoder-decoder. All feature maps are shown for bounding-box regression layers).

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

Location/ size of object

Object present in the grid Y/N

Object class



# Extreme Weather Event Detection (Racah 2017)

- Exemple

Fig. from (Racah 2017)

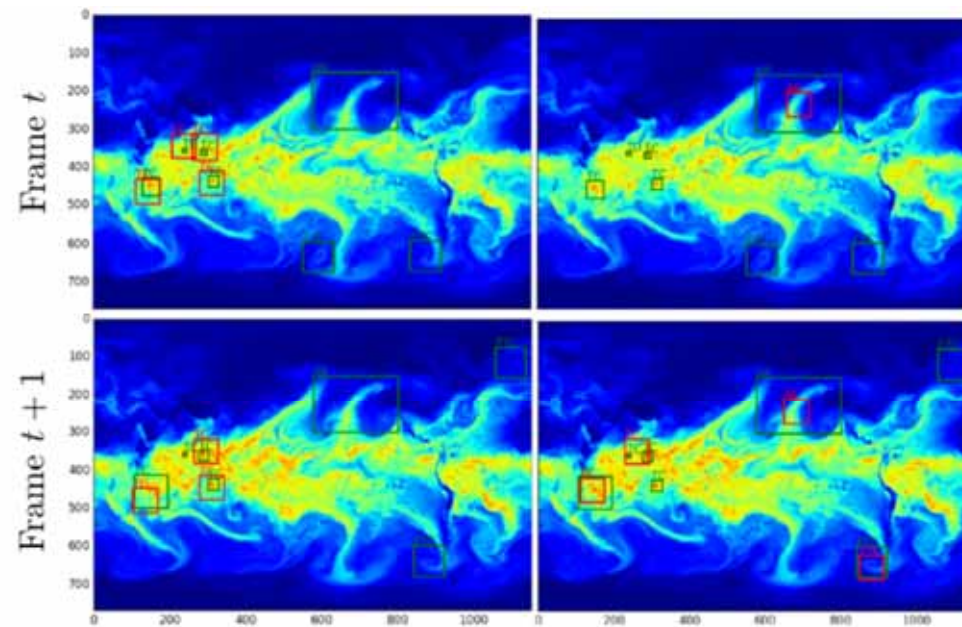


Figure 3: Bounding box predictions shown on 2 consecutive (6 hours in between) simulation frames, for the integrated water vapor column channel. Green = ground truth, Red = high confidence predictions (confidence above 0.8). 3D supervised model (Left), and semi-supervised (Right).

# 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

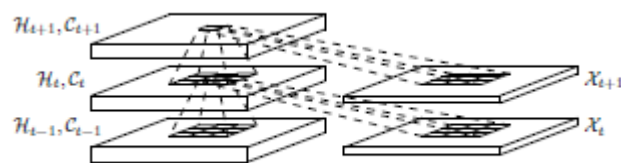


Figure 2: Inner structure of ConvLSTM

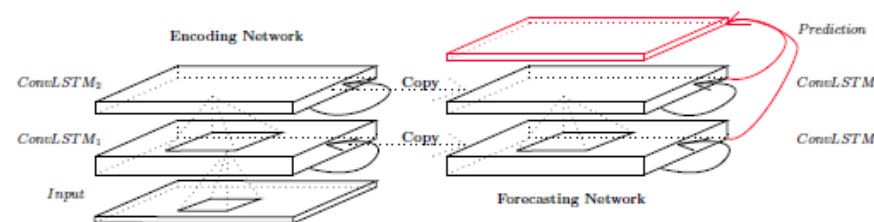


Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting

# 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

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

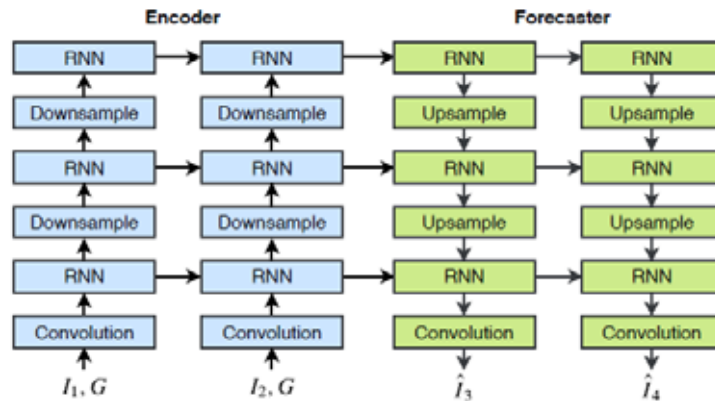
Model	Rainfall-MSE	CSI	FAR	POD	Correlation
ConvLSTM(3x3)-3x3-64-3x3-64	1.420	0.577	0.195	0.660	0.908
Rover1	1.712	0.516	0.308	0.636	0.843
Rover2	1.684	0.522	0.301	0.642	0.850
Rover3	1.685	0.522	0.301	0.642	0.849
FC-LSTM-2000-2000	1.865	0.286	0.335	0.351	0.774

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

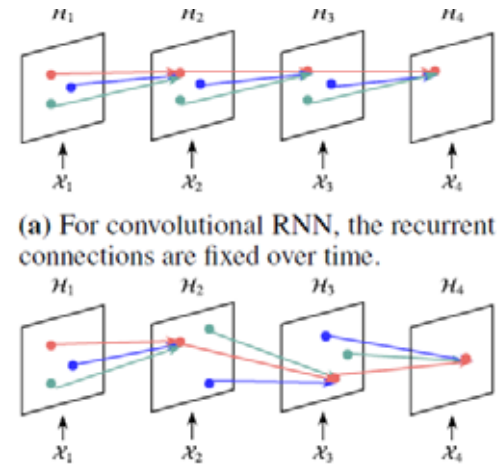
# Precipitation Nowcasting (Shi 2017)

- 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)

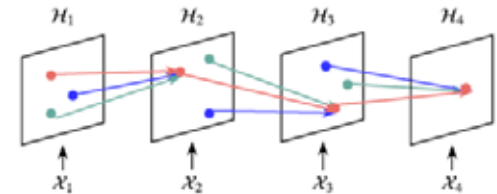
# Precipitation Nowcasting (Shi 2017)



**Figure 1:** Example of the encoding-forecasting structure used in the paper. In the figure, we use three RNNs to predict two future frames  $\hat{I}_3, \hat{I}_4$  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.



(a) For convolutional RNN, the recurrent connections are fixed over time.



(b) For trajectory RNN, the recurrent connections are dynamically determined.

**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)

- 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  $H_{t-1}$

# 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)

**Table 2:** Rain rate statistics in the HKO-7 benchmark.

Rain Rate (mm/h)	Proportion (%)	Rainfall Level
$0 \leq x < 0.5$	90.25	No / Hardly noticeable
$0.5 \leq x < 2$	4.38	Light
$2 \leq x < 5$	2.46	Light to moderate
$5 \leq x < 10$	1.35	Moderate
$10 \leq x < 30$	1.14	Moderate to heavy
$30 \leq x$	0.42	Rainstorm warning

- Performance comparison

**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. ‘ $\uparrow$ ’ means that the score is higher the better while ‘ $\downarrow$ ’ means that the score is lower the better. ‘ $r \geq \tau$ ’ means the skill score at the  $\tau$  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.

Algorithms	$r \geq 0.5$	$r \geq 2$	CSI $\uparrow$ $r \geq 5$	$r \geq 10$	$r \geq 30$	$r \geq 0.5$	$r \geq 2$	HSS $\uparrow$ $r \geq 5$	$r \geq 10$	$r \geq 30$	B-MSE $\downarrow$	B-MAE $\downarrow$
Offline Setting												
Last Frame	0.4022	0.3266	0.2401	0.1574	0.0692	<u>0.5207</u>	0.4531	0.3582	0.2512	0.1193	15274	28042
ROVER + Linear	0.4762	0.4089	0.3151	0.2146	0.1067	0.6038	0.5473	0.4516	0.3301	0.1762	11651	23437
ROVER + Non-linear	0.4655	0.4074	0.3226	0.2164	0.0951	0.5896	0.5436	0.4590	0.3318	0.1576	10945	22857
2D CNN	0.5095	0.4396	0.3406	0.2392	0.1093	0.6366	0.5809	0.4851	0.3690	0.1885	7332	18091
3D CNN	0.5109	0.4411	0.3415	0.2424	0.1185	0.6334	0.5825	0.4862	0.3734	0.2034	7202	17593
ConvGRU-nobal	0.5476	0.4661	0.3526	0.2138	0.0712	0.6756	0.6094	0.4981	0.3286	0.1160	9087	19642
ConvGRU	<u>0.5489</u>	<u>0.4731</u>	<u>0.3720</u>	<u>0.2789</u>	<u>0.1776</u>	<u>0.6701</u>	<u>0.6104</u>	<u>0.5163</u>	<u>0.4159</u>	<u>0.2893</u>	<u>5951</u>	<u>15000</u>
TrajGRU	<u>0.5528</u>	<u>0.4759</u>	<u>0.3751</u>	<u>0.2835</u>	<u>0.1856</u>	<u>0.6731</u>	<u>0.6126</u>	<u>0.5192</u>	<u>0.4207</u>	<u>0.2996</u>	<u>5816</u>	<u>14675</u>
Online Setting												
2D CNN	0.5112	0.4363	0.3364	0.2435	0.1263	0.6365	0.5756	0.4790	0.3744	0.2162	6654	17071
3D CNN	0.5106	0.4344	0.3345	0.2427	0.1299	0.6355	0.5736	0.4766	0.3733	0.2220	6690	16903
ConvGRU	<u>0.5511</u>	<u>0.4737</u>	<u>0.3742</u>	<u>0.2843</u>	<u>0.1837</u>	<u>0.6712</u>	<u>0.6105</u>	<u>0.5183</u>	<u>0.4226</u>	<u>0.2981</u>	<u>5724</u>	<u>14772</u>
TrajGRU	<u>0.5563</u>	<u>0.4798</u>	<u>0.3808</u>	<u>0.2914</u>	<u>0.1933</u>	<u>0.6760</u>	<u>0.6164</u>	<u>0.5253</u>	<u>0.4308</u>	<u>0.3111</u>	<u>5589</u>	<u>14465</u>

# 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 (\*)

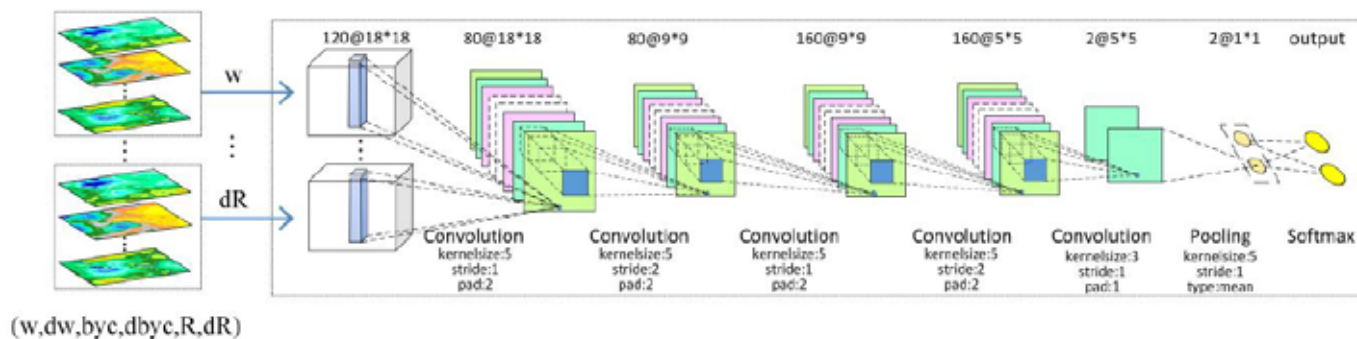


Figure 1: Network Architecture of multi-channel 3D-SCN

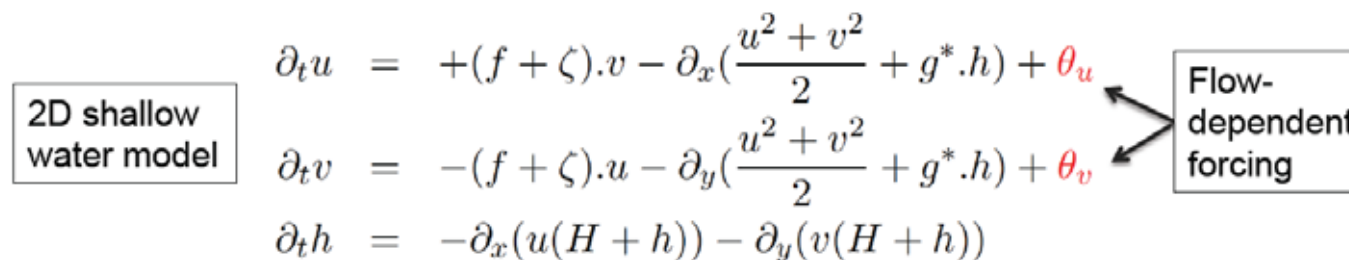
## Integration of NN in numerical models (Brajaard 2018)

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

2D shallow  
water model

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

Flow-  
dependent  
forcing



where  $\zeta$  is the vorticity.

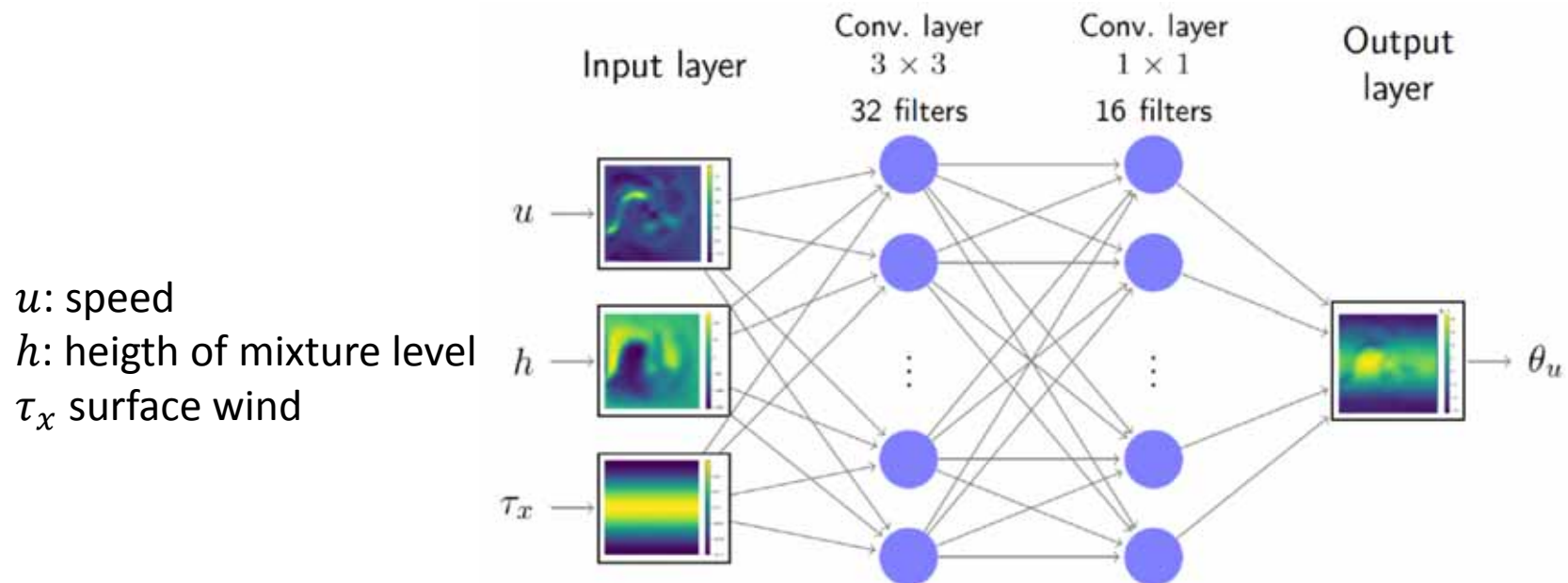
$\theta_u$  and  $\theta_v$  account for the effects of forcing, dissipation and diffusion.

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  $F(x(t))$ ?

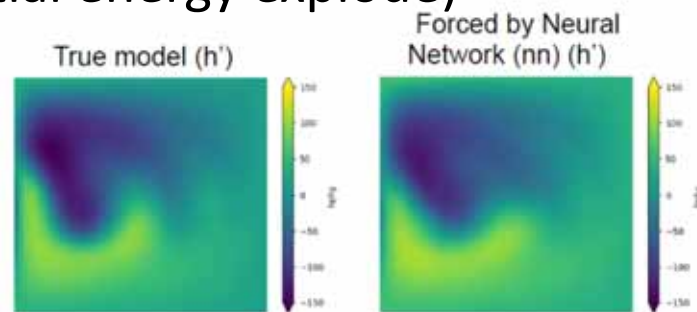
# Integration of NN in numerical models (Brajjard 2018)

- 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

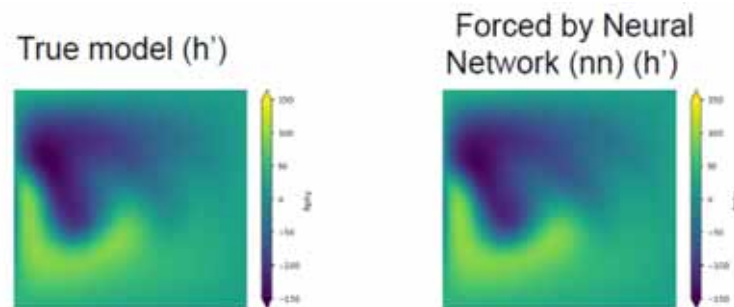


## Integration of NN in numerical models (Brajard 2018)

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



- Solution: add a mass conservation constraint ( $h_{\text{mean}} = \text{constant}$ ) to the neural network training algorithm (physics-informed machine learning)



Incorporating prior knowledge

Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge, (de Bezenac 2018)

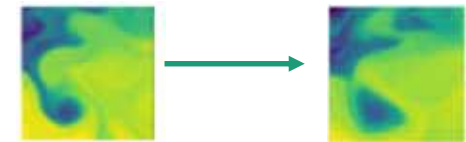
- Motivations
  - DL SOTA for perception problems
  - Natural physical phenomenon are much more complex than problems handled by Deep Learning today
    - Can we incorporate prior knowledge from physics in statistical models?
- Challenge
  - Interaction between the Physical and the Statistical paradigms
- 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
  - $I_{t+\Delta t}(x) = (k * I_t)(x - w(x))$

- 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  $D$**

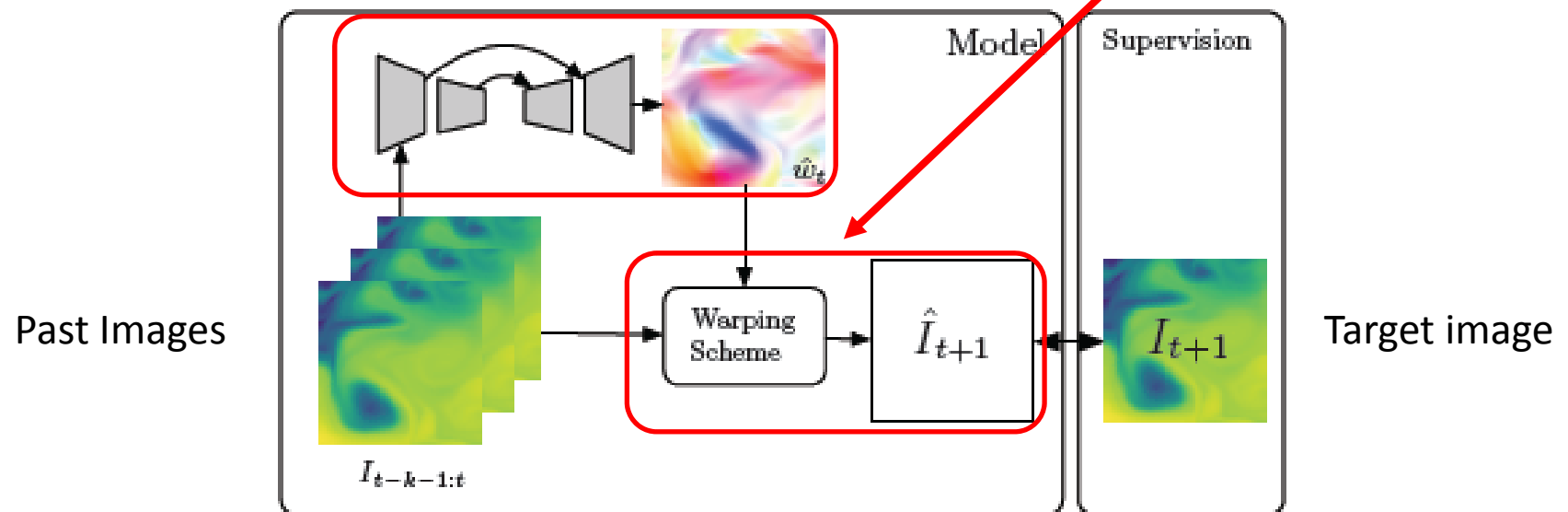
Incorporating prior knowledge - (de Bezenac 2018)

Prediction Model

Objective: predict  $I_{t+1}$  from past  $I_t, I_{t-1}, \dots$

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

Warping Scheme  
Implements discretized  
Advection-Diffusion  
solution



- 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 ambient GANs (Bora et al. 2018)

# Solving inverse problems with NNs (de Bezenac et al. ongoing work)

- Ambient 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

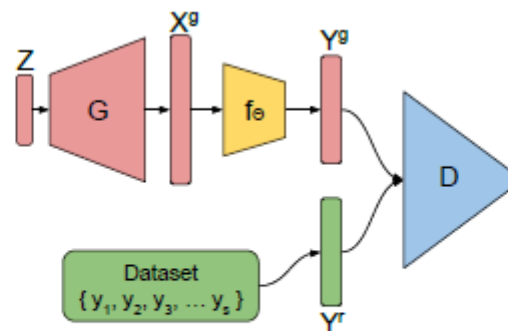


Fig. from Bora et al. 2018

Figure 1: AmbientGAN training. The output of the generator is passed through a simulated random measurement function  $f_\Theta$ . The discriminator must decide if a measurement is real or generated.

## Solving inverse problems with NNs (de Bezenac et al. ongoing work)

- AmbientGAN example



Figure 2: (Left) Samples of lossy measurements used for training. Samples produced by (middle) a baseline that trains from inpainted images, and (right) our model.

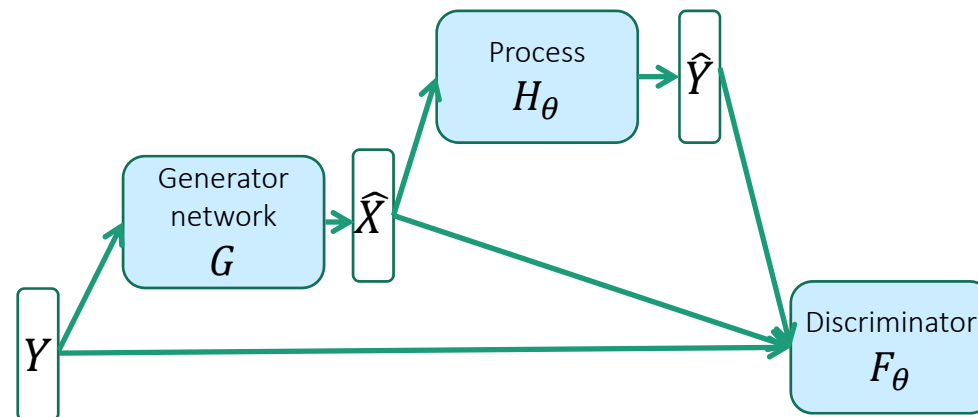
Fig. from Bora et al. 2018

# Solving inverse problems with NNs (de Bezenac et al. ongoing work)

- Conditional ambient GANs

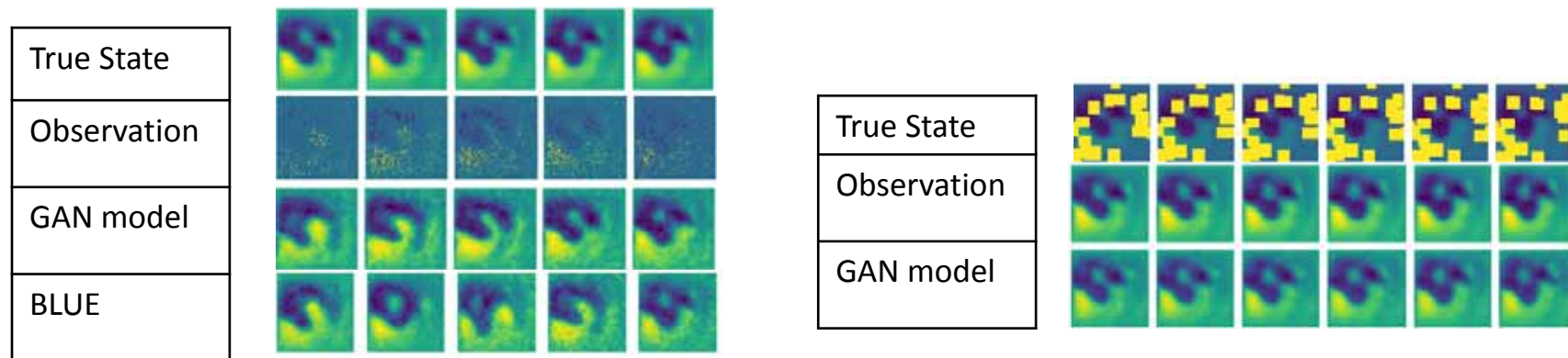
- Objective

- 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 »



# 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
  - $\frac{dX}{dt} = F(X(t), \theta(t)), X(0) = X_0$  (1)
- Resnet module
  - $X_{t+1} = X_t + G(X_t, \theta_t)$  (2)
  - $X_{t+1} = X_t + hF(X_t, \theta_t), h \in [0,1]$
  - $\frac{X_{t+1}-X_t}{h} = F(X_t, \theta_t)$ 
    - 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
    - $X(t_1) = X(t_0) + hF(X(t_0), \theta_{t_0})$
    - $X(t_2) = X(t_1) + hF(X(t_1), \theta_{t_1}), \dots$



# Resnet as a discretization scheme for ODEs

- This suggests that alternative discretization schemes will 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)

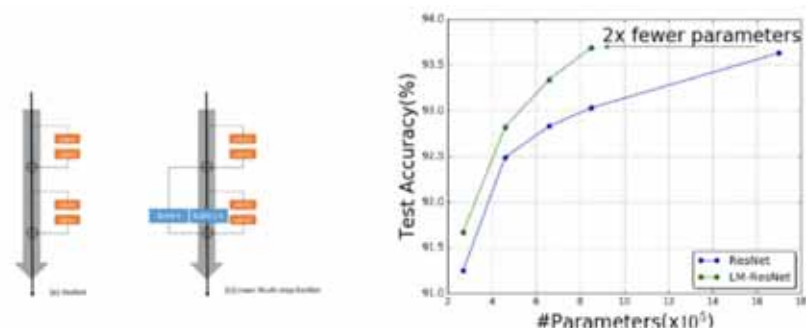


Figure 2: LM-architecture is an efficient structure that enables ResNet to achieve same level of accuracy with only half of the parameters on CIFAR10.

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

# References

- Brajard, J., (1), Charantonis A., Sirven J., Can a neural network learn a numerical model ?, Geophysical Research Abstracts, Vol. 20, EGU2018-13973, 2018
- de Bezenac, E., Pajot, A., & Gallinari, P. (2018). Deep Learning For Physical Processes: Incorporating Prior Scientific Knowledge. In *ICLR*.
- Fablet, R., Ouala, S., & Herzet, C. (2017). Bilinear residual Neural Network for the identification and forecasting of dynamical systems, 2(1). Retrieved from <http://arxiv.org/abs/1712.07003>
- Franz, K., Roscher, R., Milioto, A., & Wenzel, S. (n.d.). Ocean Eddy Identification and Tracking using Neural Networks. *ArXiv Computer Science*. <https://doi.org/10.1038/nbt.3343>
- Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial Transformer Networks. *Nips*, 2017--2025. <https://doi.org/10.1038/nbt.3343>
- Kim, S., Hong, S., Joh, M., & Song, S. (2017). DeepRain: ConvLSTM Network for Precipitation Prediction using Multichannel Radar Data, 3–6. Retrieved from <http://arxiv.org/abs/1711.02316>
- Lguensat, R., Sun, M., Fablet, R., Mason, E., Tandeo, P., & Chen, G. (2017). EddyNet: A Deep Neural Network For Pixel-Wise Classification of Oceanic Eddies (pp. 1–5). Retrieved from <http://arxiv.org/abs/1711.03954>
- Liu, Y., Racah, E., Prabhat, Correa, J., Khosrowshahi, A., Lavers, D., ... Collins, W. (2016). Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets. *ArXiv*, 1605.01156, 81–88. <https://doi.org/10.475/123>
- Lu, Y., Zhong, A., Li, Q., & Dong, B. (2017). Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations (pp. 1–15). Retrieved from <http://arxiv.org/abs/1710.10121>
- Racah, E., Beckham, C., Maharaj, T., Kahou, S. E., Prabhat, & Pal, C. (2017). ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. In *NIPS* (pp. 1–12). Retrieved from <http://arxiv.org/abs/1612.02095>
- Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W., & Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems* 28, 802–810.
- Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W., & Woo, W. (2017). Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model (pp. 1–11). Retrieved from <http://arxiv.org/abs/1706.03458>
- Zhang, W., Han, L., Sun, J., Guo, H., & Dai, J. (2017). Application of Multi-channel 3D-cube Successive Convolution Network for Convective Storm Nowcasting. *ArXiv Preprint ArXiv:1702.04517*, 1–9.