Introduction au Deep Learning

Thierry Artières

Equipe QARMA d'apprentissage automatique
Pole Science des Données - LIS lab
Ecole Centrale Marseille

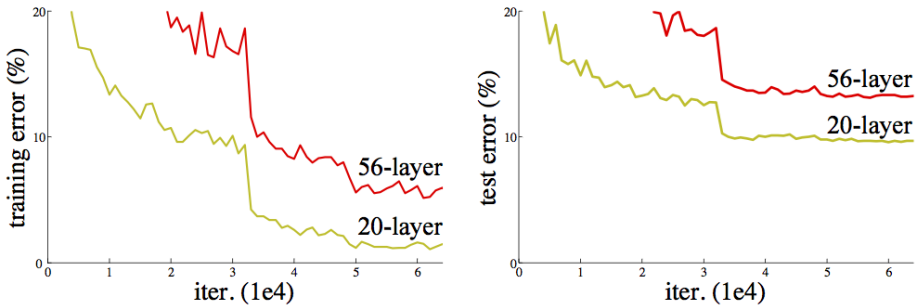
July 4, 2019

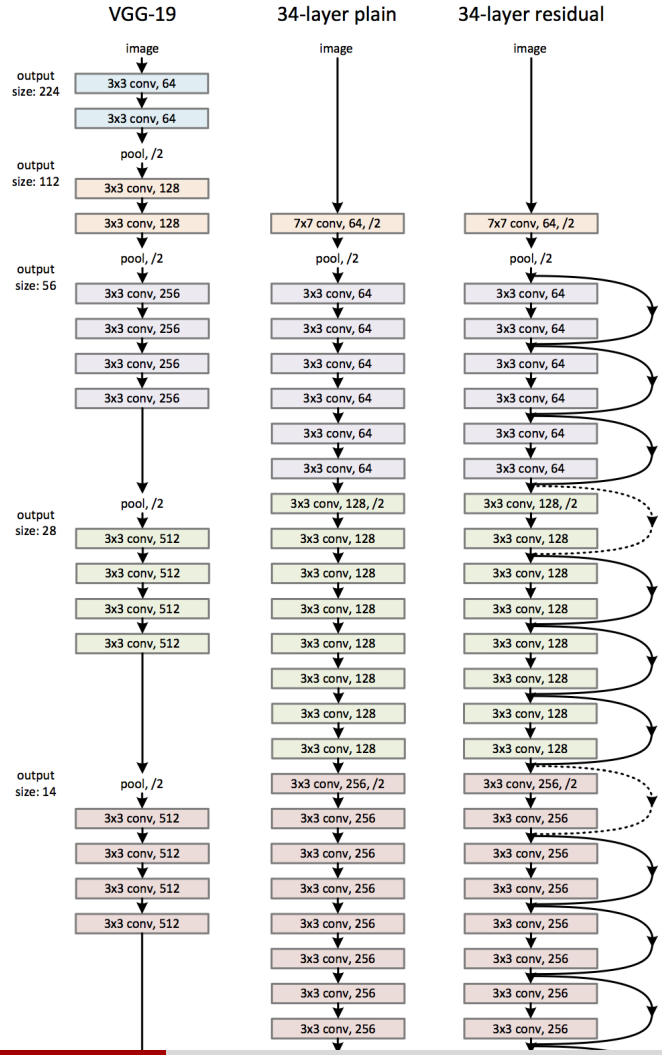
Outline

From shalow to deep

Simply stacking layers does not work (CIFAR results) ! (figures form [He and al., 2015])

...





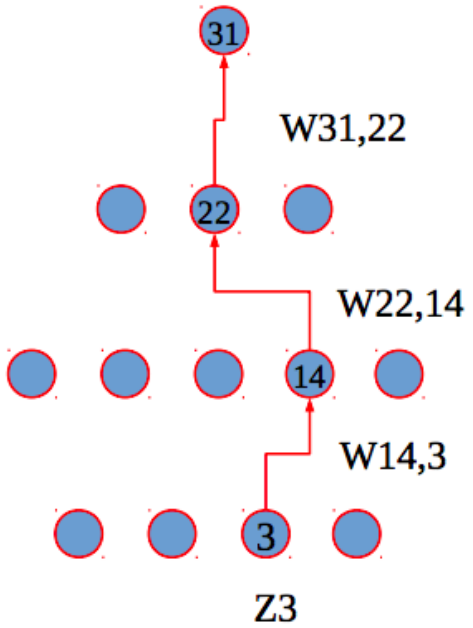
While NN optimization is reasonably easy

(Easier) analysis for ReLU DNNs [LeCun Statlearn tutorial]

- ... but expectation of generalized results to other activation functions
- With ReLu and MaxPooling operators one may formalize what happens on a path from an input to the output
- The output may be computed as :

$$\hat{y} = \sum_{P} \delta_P(W, X)(\prod_{(ij)\in P} w_{ij}) x_{j_{start}}$$

- $\delta_P(W, X)$: 1 if active path, 0 otherwise
- Implemented function is piece-wise linear



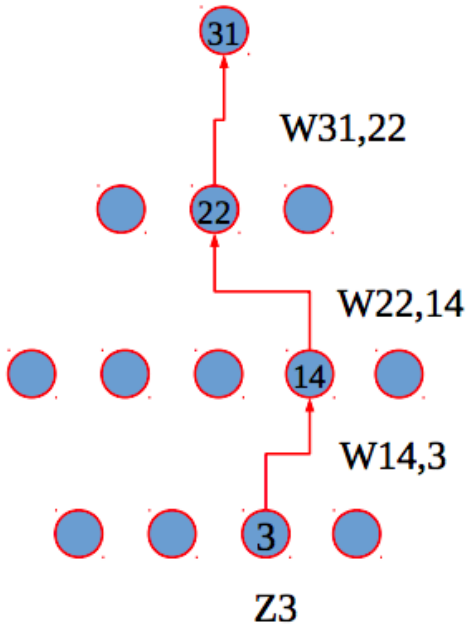
While NN optimization is reasonably easy

(Easier) analysis for ReLU DNNs [LeCun Statlearn tutorial]

- Objective function : piece wise polynomial (degree = number of hidden layers) with partially random coefficients

$$C(W) = \sum_{P} C_P(X, Y, W)\left(\prod_{(ij)\in P} w_{ij}\right)$$

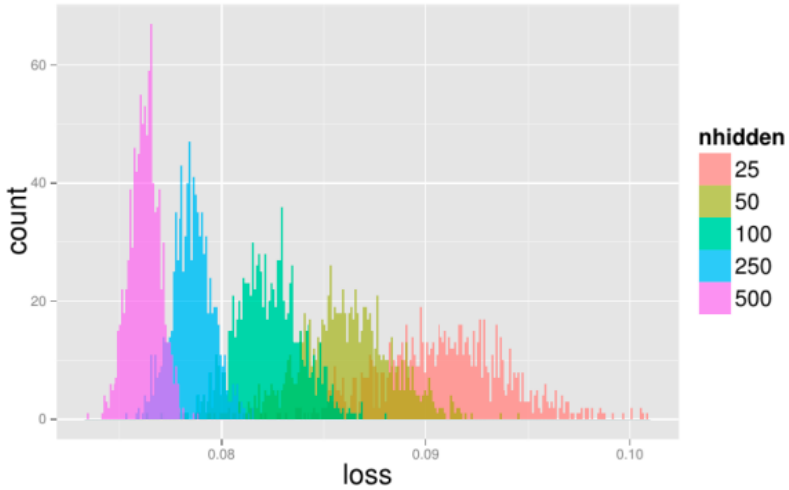
- Hint from results on distribution of critical points for polynomials with random coefficients



Deep ReLu networks

Easier analysis... [LeCun Statlearn tutorial]

- Experiments by [Choromanska and al., 2015]: Train 2-layers nets on Mnist from multiple initializations and measure loss on the test set
- Many close local minimas for large nets
- Objective function do not exhibit lots of saddle points and most local minima are good and close to globale minimas



[Choromanska, Henaff, Mathieu, Ben Arous, LeCun 2015]

Yet the depth alone is not enough

While SGD works well for NNs, the optimization of DNNs requires careful design and tricks

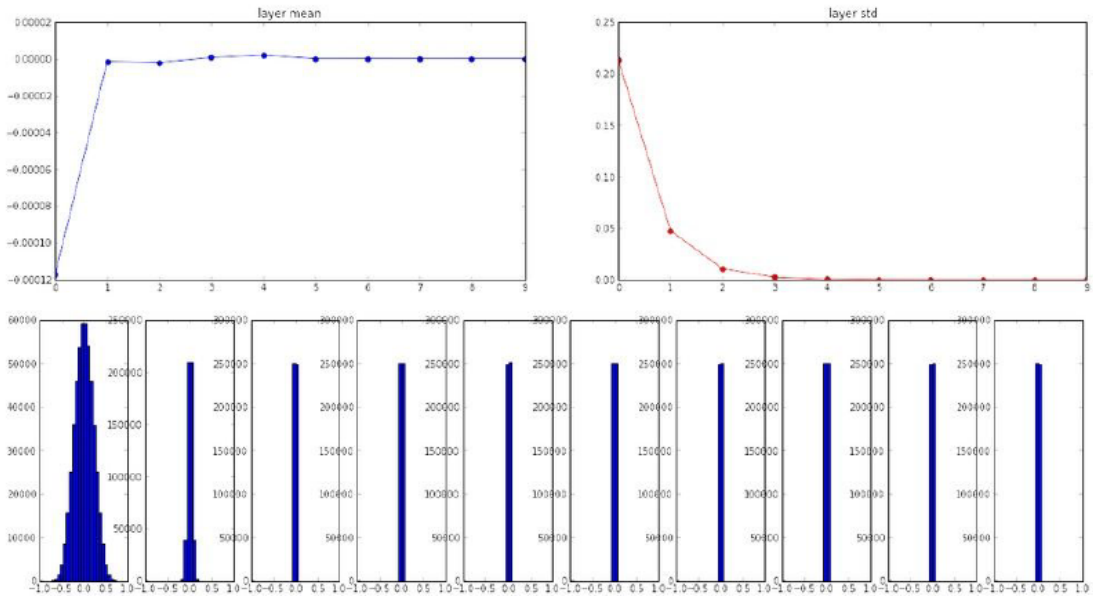
- Make the gradient flow with activation normalization (Batch Normalization)
- Make the gradient flow with structural constraints (Identity mapping)
- Regularization (Dropout)

Problems with activity propagation in deep NNs [He et al., 2016]

Few slides from *Fei Fei Li*

Standard initialization schema for MLPs

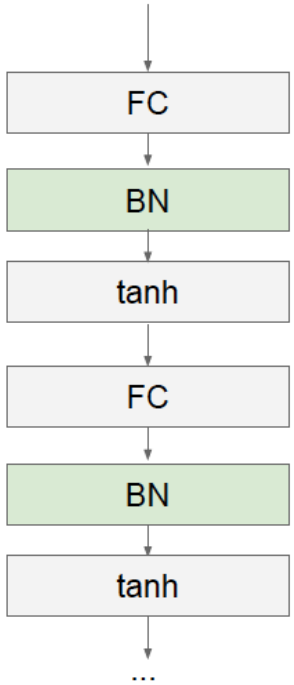
- 10 layers networks (500 neurones each, with tanh)
- Initialization : gaussian random with small (std=0.01) values (what if all null initialization?)
- All activations at 0
- What about the gradient ?



Batch Normalization

Main idea

- Usually inputs to neural networks are normalized to either the range of [0, 1] or [-1, 1] or to mean=0 and variance=1
- BN essentially performs Whitening to the intermediate layers of the networks.
- Usually placed before nonlinearities

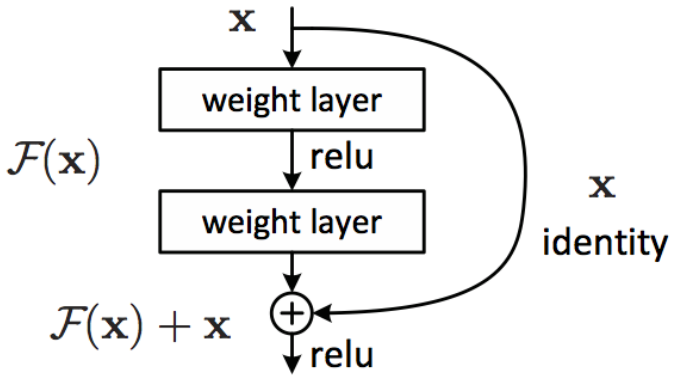


From Fei Fei Li's slides

Identity mapping in Residual Networks

Principle

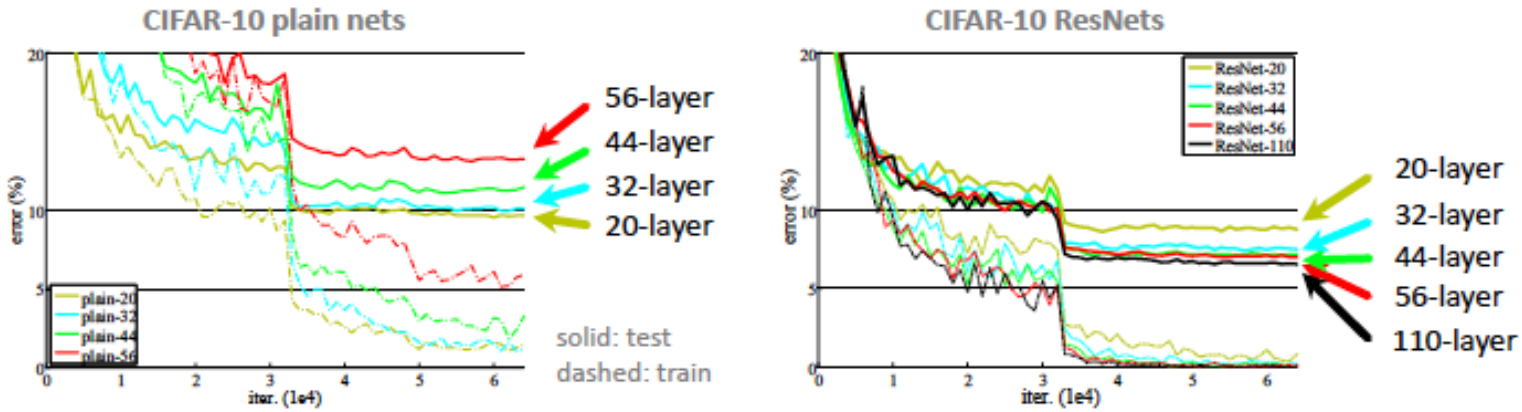
- Include identity mapping in the model
- ResNet building block [He and al., 2015]]
- Every layer becomes close to the output ($\Rightarrow$ not far in the backpropagation process)



Identity mapping in Residual Networks

Principle

- Include identity mapping in the model
- ResNet building block [He and al., 2015]]
- Every layer becomes close to the output ($\Rightarrow$ not far in the backpropagation process)



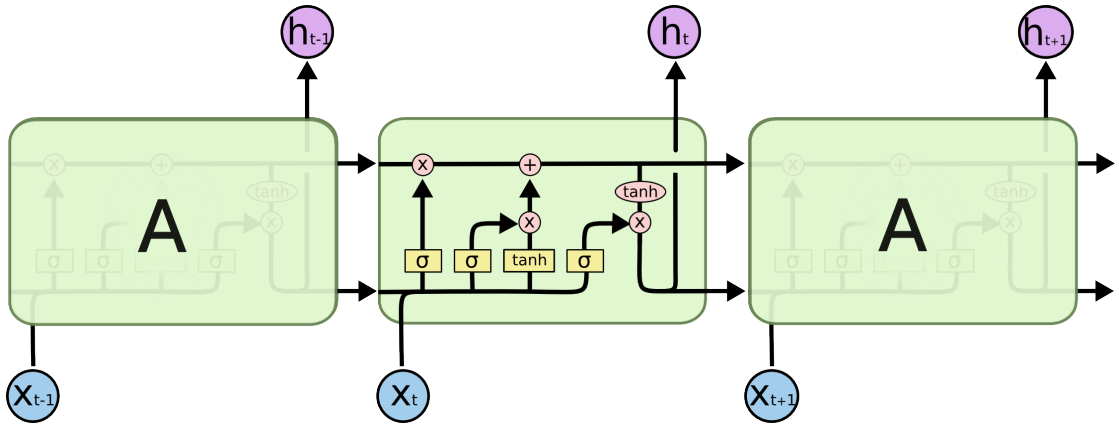
Identity mapping with LSTM units in RNNs

What it does

- Main behaviour
  - If $f_t == 1$ and $i_t == 0$ use previous cell state
  - If $f_t == 0$ and $i_t == 1$ ignore previous cell sate
  - If $o_t == 1$ output ois set to cell sate
  - If $o_t == 0$ output is set to 0

With...

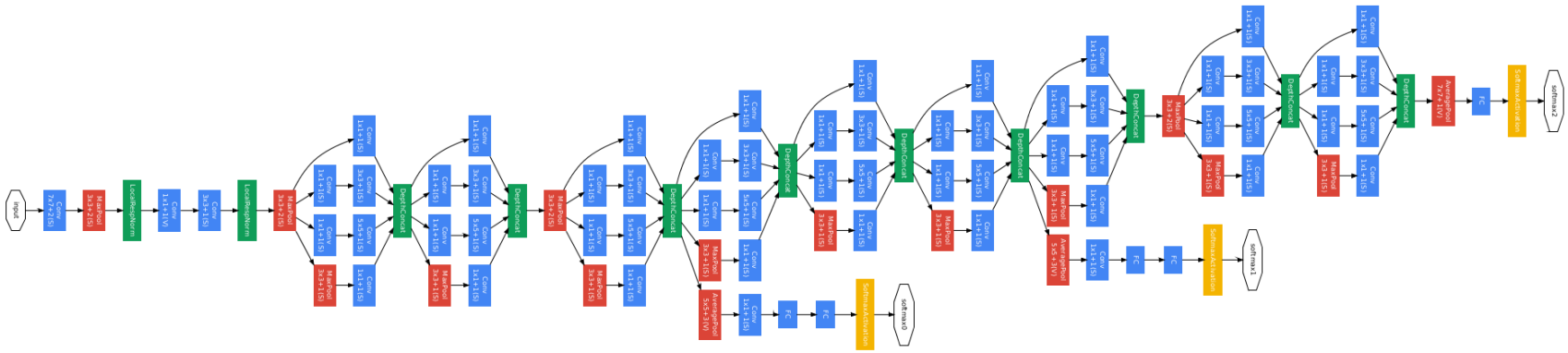
- Cell state c_t
- Forget gate f_t
- Input gate i_t
- Output o_t
- Hidden state to propagate to upper layers h_t



Auxiliary loss on intermediate layers

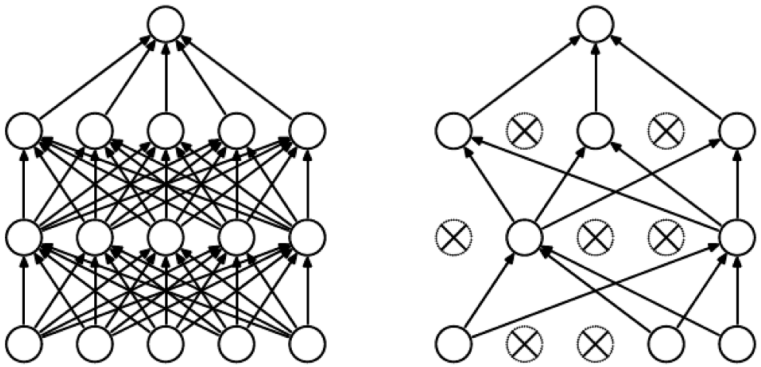
Google net

- Auxiliary loss brings some gradient to first layers



Regularizing with Dropout [Hinton 2012]

Principle

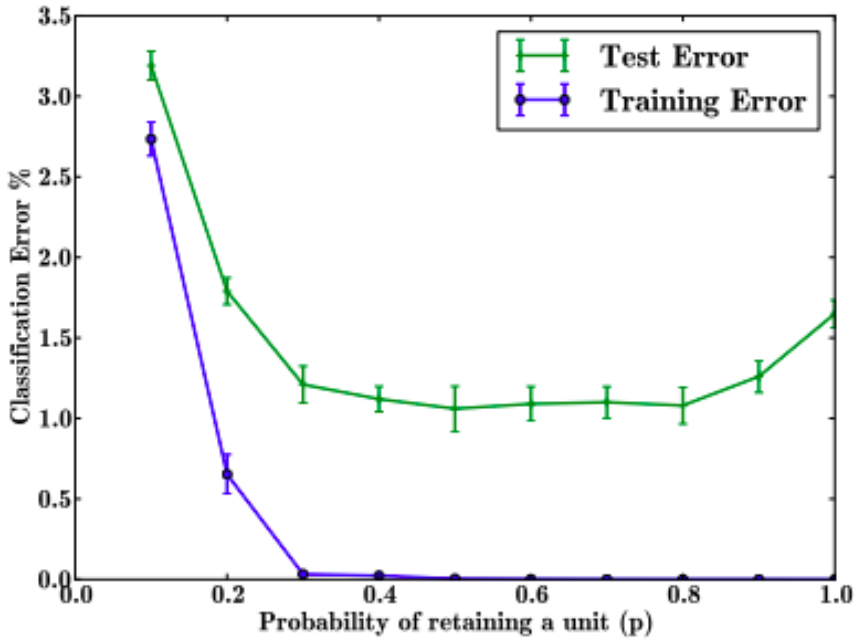


(a) Standard Neural Net (b) After applying dropout.

- First method that actually allowed learning deep networks without pretraining and smart initialization
- Related to ensemble of models
- Weights are normalized at inference time

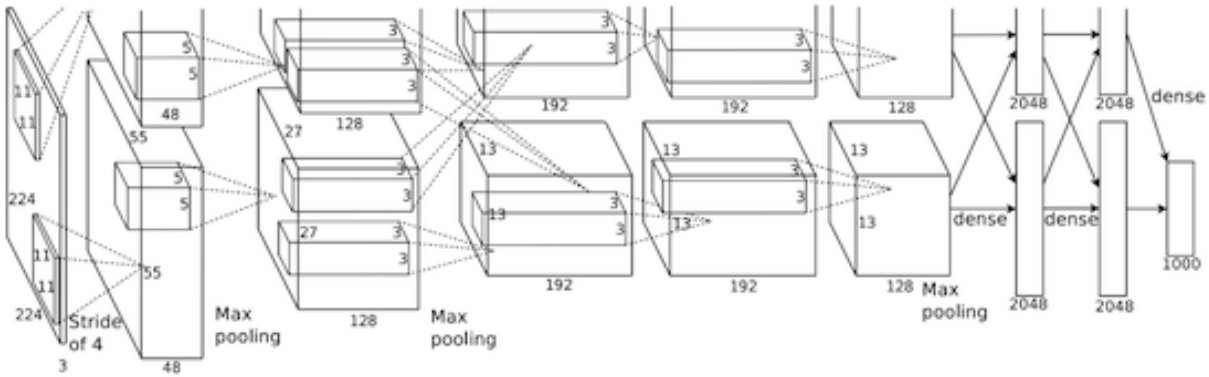
Dropout [Hinton 2012]

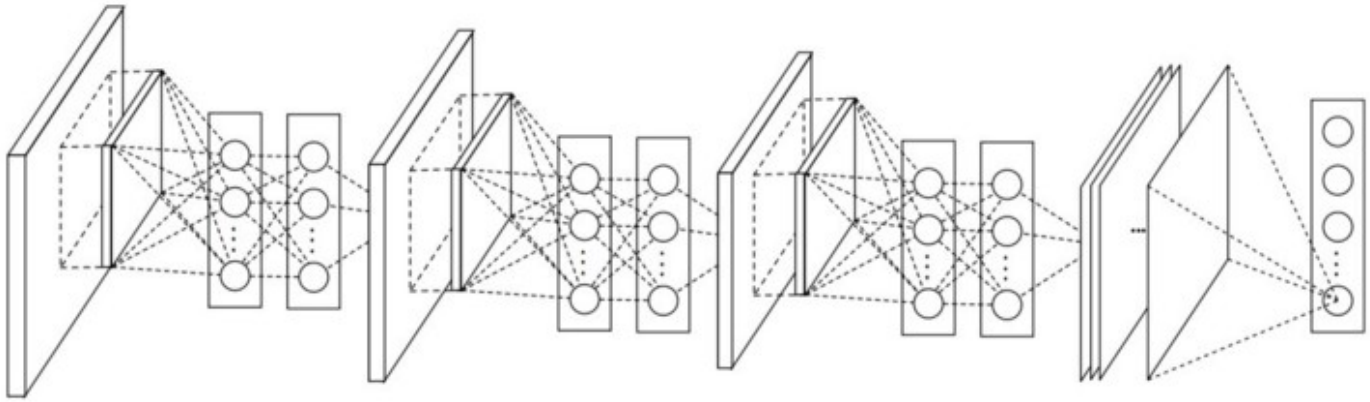
Do not ever fear overfitting !



Examples of architectures

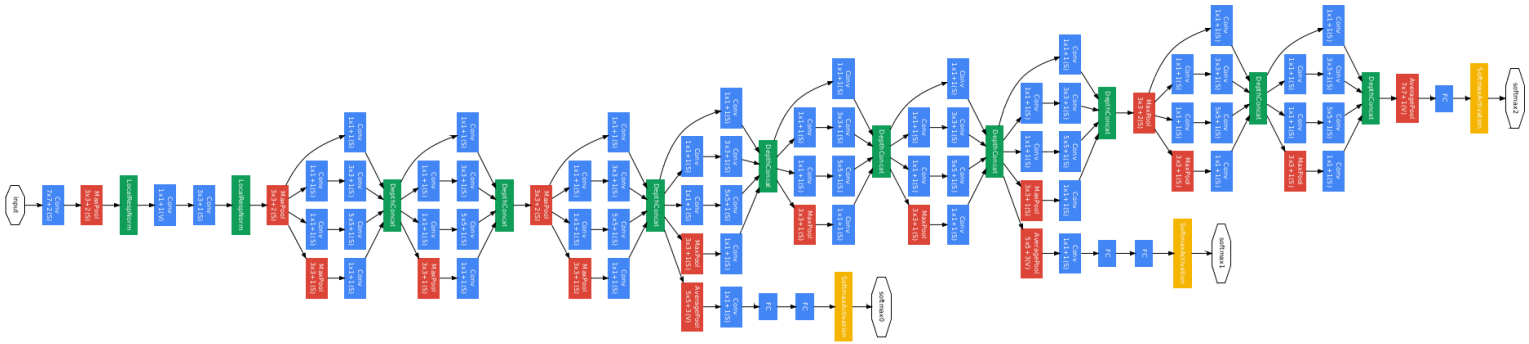
AlexNet [Krizhevsky and al., 2012] (top) and NetworkInNetwork [Lin and al.,2013] (bottom)





Looking for a good architecture: Lego game

How to reach such an architecture (GoogleNet 2014) ?



Searching for a good architecture requires making choices !!

Looking for a good architecture: Lego game

Gridsearch

- Standard Machine Learning models
  - Very few hyperparameters (regularization tradeoff, kernel width or degree etc)
  - Easier optimization problem
  - Usually much less data and much simpler models
  - $\Rightarrow$ Quite exhaustive gridsearch
- Large deep networks
  - Many choices (sequence of layers, width of layers, convolution kernel's size and strides, activation function, optimization routine and its parameters...): Not many theoretical hints
  - Harder optimization problem
  - Each try is expensive
  - $\Rightarrow$ Reuse of others' architectures whenever possible
  - $\Rightarrow$ Gain experience on **how to** design

Looking for a good architceture

Activation function

- ReLu
- Sigmoid
- Tanh
- Linear
- GRU
- LSTM

Connectivty

- Fully connected
- Convolutional, Pooling
- Dilated
- Recurrent
- Recursive
- Skip/ Residual

Criterion

- MSE
- L2 reg
- Cross entropy
- Binary cross entropy
- Likelihood

Sizes

- Layers' sizes
- Dropout rate

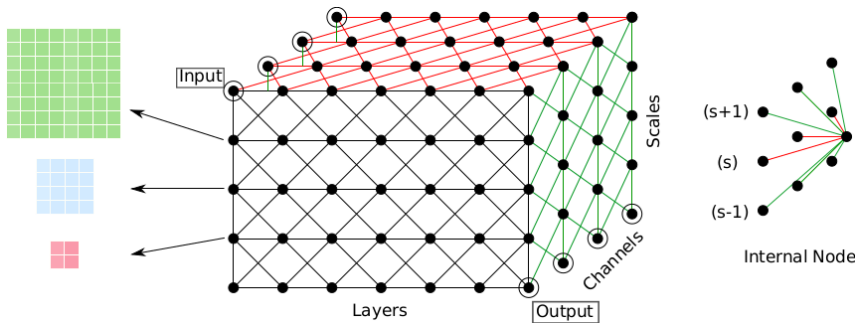
Optimization

- SGD
- Adam
- Adagrad
- Adadelta
- RMSProp
- Learning rate
- Decay
- Batch size

Looking for a good architceture

Illustration [Verbeek 2017]

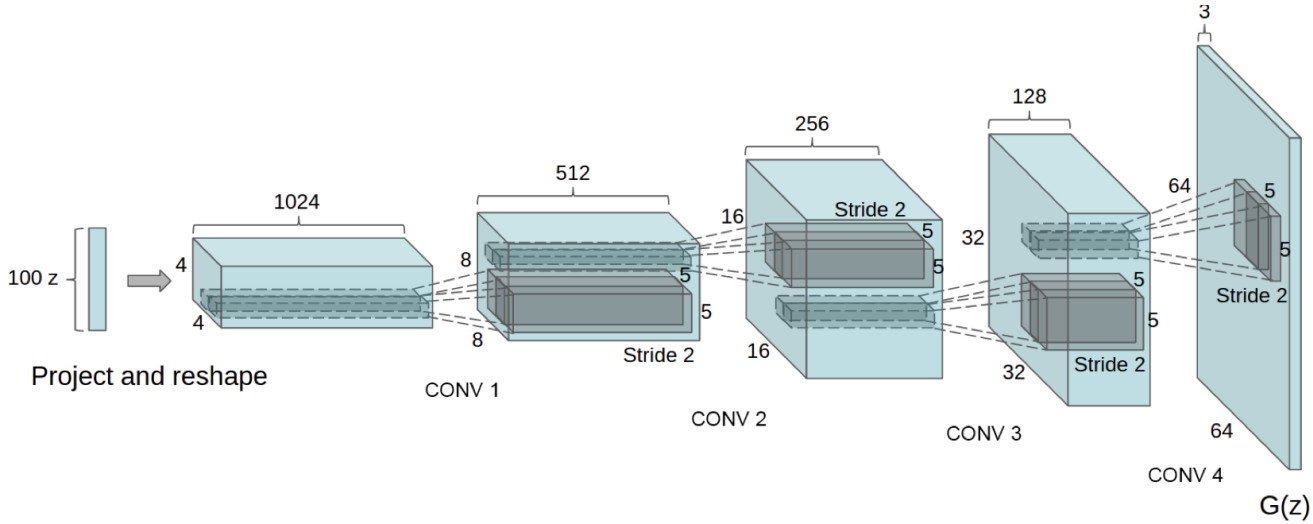
- **Simple** search (but for a large network)
  - 19 convolution layers and 5 pooling layers to set
  - Question: where to put the poooling layers? $\rightarrow$ 40 000 architectures !!
  - No question about layers' dimensions, activation function, kernels' size, pooling type etc
- Remember
  - 1 hour GPU on AWS = 1 \$
  - Learning 1 model = Few hours
    $\Rightarrow$ Expensive design !!!
- Not much alternatives



Looking for a good architecture: use others' !!

Deep Models for High resolution images [Radford 2015]

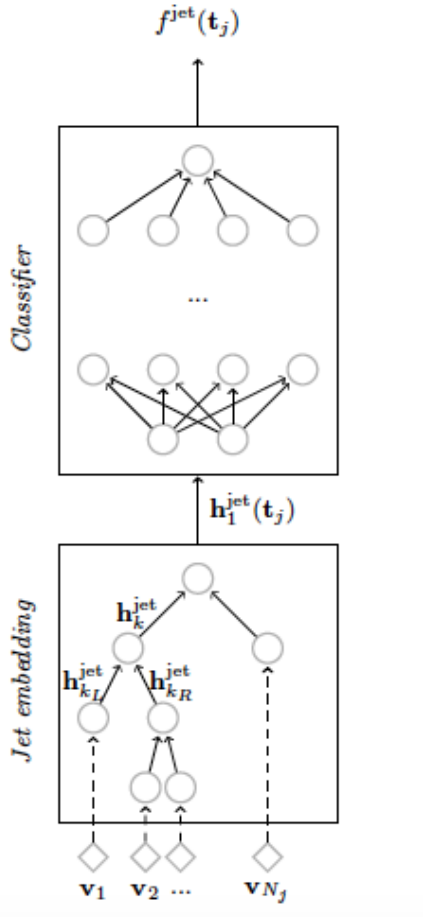
Historical attempts to scale up GANs using CNNs to model images have been unsuccessful. This motivated the authors of LAPGAN (Denton et al., 2015) to develop an alternative approach to iteratively upscale low resolution generated images which can be modeled more reliably. We also encountered difficulties attempting to scale GANs using CNN architectures commonly used in the supervised literature. However, after extensive model exploration we identified a family of architectures that resulted in stable training across a range of datasets and allowed for training higher resolution and deeper generative models.



Architecture design from prior knowledge (HEP example)

What if unstructured data: is DNN useful ?

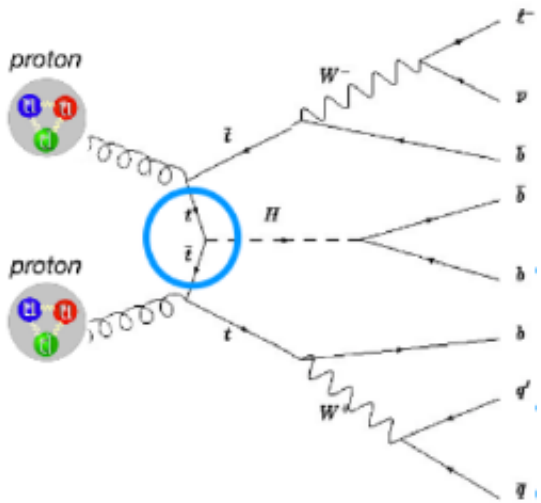
- Modeling collision at CERN : features = physical features of jets during the collision (speed, energy, angle with collision axis...)
- Main approaches
  - Use non deep machine learning models
  - Represent data as images and use Deep NNs
  - Design DNN architecture from knowledge on the considered process
- Example : Learn to aggregate features of jets using a tree structure inspired from data knowledge [Louppe et al., 2018]

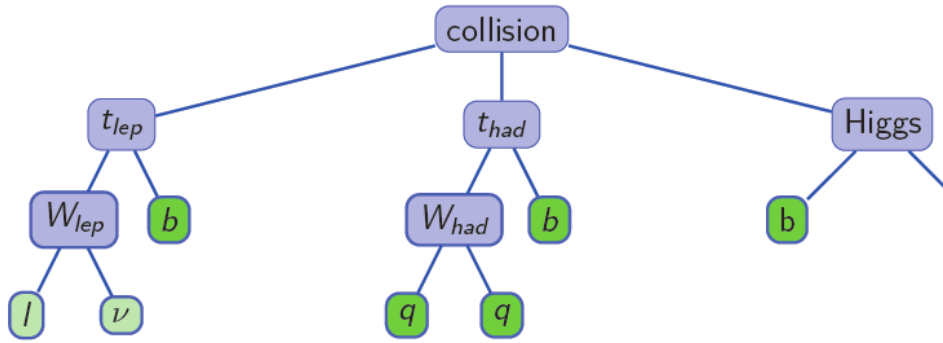


Architecture design from prior knowledge (HEP example)

Ziyu Guo's thesis (with Y. Coadou at CPPM)

- Deep learning in the search for ttH with the ATLAS experiment at the Large Hadron Collider
- Rely on the physical process to design the NN structure
- Better results than DNN, on par with state of the art models in HEP (BDTs)



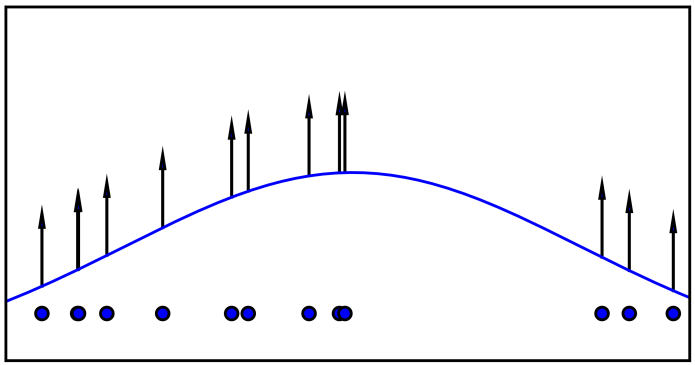


Outline

Generative models

Goal

- Learn to generate complex and realistic data
- Statistical viewpoint : learn a model of the density of data / able to sample with this density
  - Postulate a parametric model : Usually not complex enough
  - Postulate a parametric form and perform optimization (e.g. Maximum Likelihood) : Intractable for complex forms $p(x) = \frac{F(x)}{Z(x}$ with $Z(x) = \sum_x F(x)$

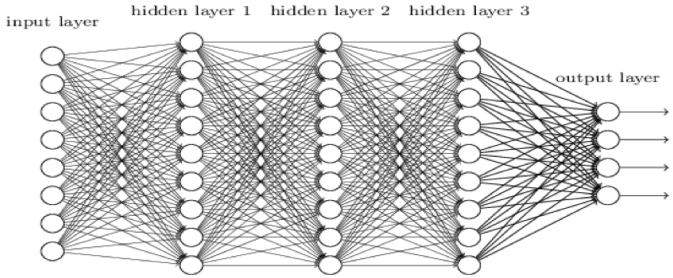


$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

Maximum Likelihood Estimation (MLE)

Adversarial Learning: Generator

Determinitic NN as a generative model



Using a deterministic NN as a generative model

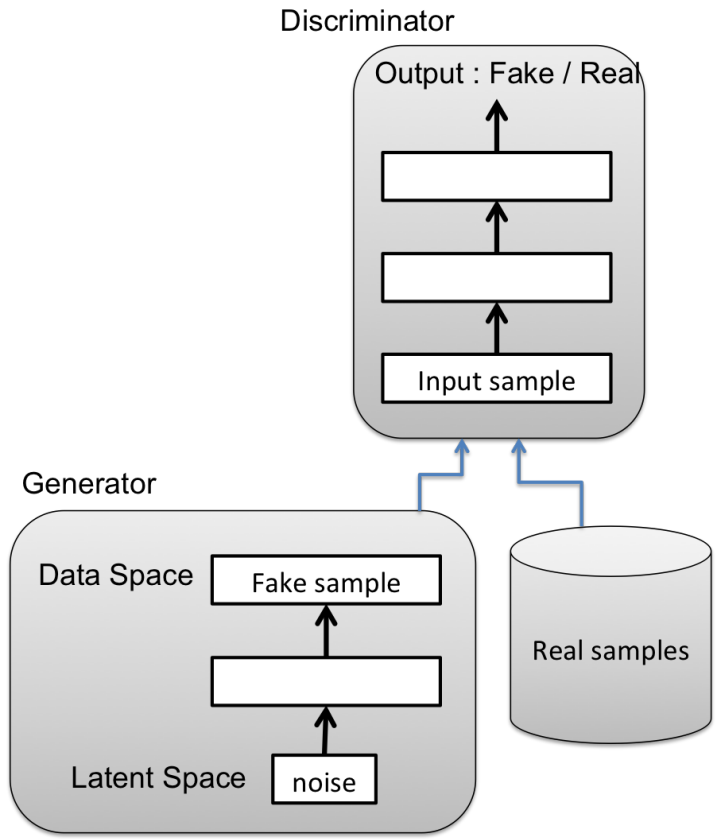
- Let note the function implemented by the model as G
- Let note the input $z \rightarrow$ The NN computes $G(z)$
- Assume z obeys a prior (noise) distribution, p_z, e.g. Gaussian distribution
- then the output x of the NN follows a distribution

$$\Rightarrow p_G(x) = \int_{z \text{ s.t. } G(z)=x} p_z(z)dz$$

Le principe de l'adversarial learning [Goodfellow and al., 2014]

Principle

- Jeu à deux joueurs: un générateur et un discriminateur
  - le discriminateur veut distinguer les exemples générés des vrais exemples
  - Le générateur veut tromper le discriminateur



Adversarial Learning criterion

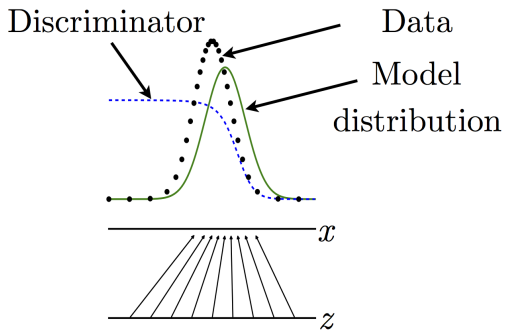
Criterion from [Goodfellow and al., 2014]

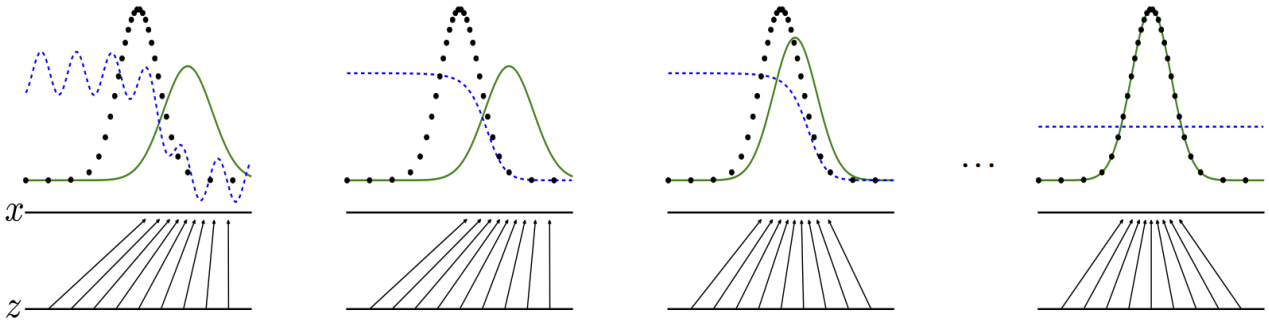
- Generator G and Discriminator D are two NNs
  - Whose parameters are noted θ_g and θ_d
- Distributions
  - p_{data} stands for the empirical distribution of the data from the training set
  - p_z is a prior noise distribution, e.g. a Gaussian distribution
  - On convergence we want $p_g = p_{data}$
- Learning criterion:

$$min_g max_d v(\theta_g, \theta_d) = \mathbf{E}_{x \sim p_{data}} [logD(x)] + \mathbf{E}_{z \sim p_z} [log(1 - D(G(z)))]$$

  - Assume G is fixed: D is trained to distinguish between fake and true samples
  - Assume D is fixed : G is trained to generate samples as realistic as possible

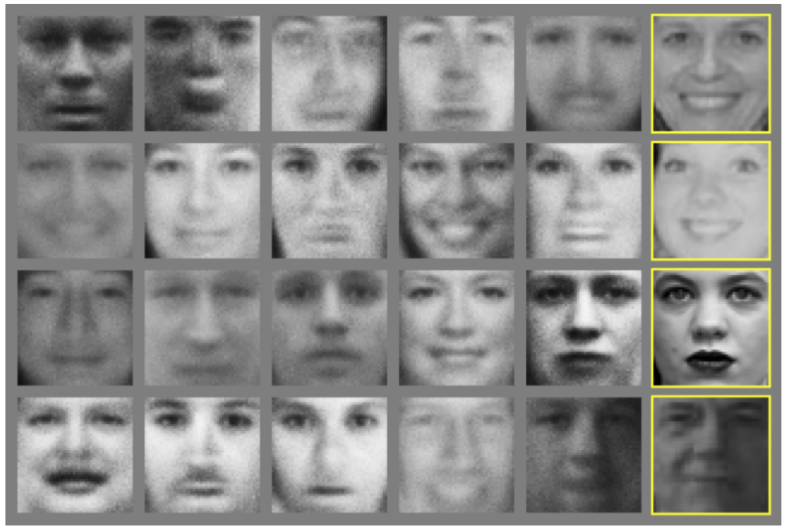
Adversarial Learning theory: What happens during Learning



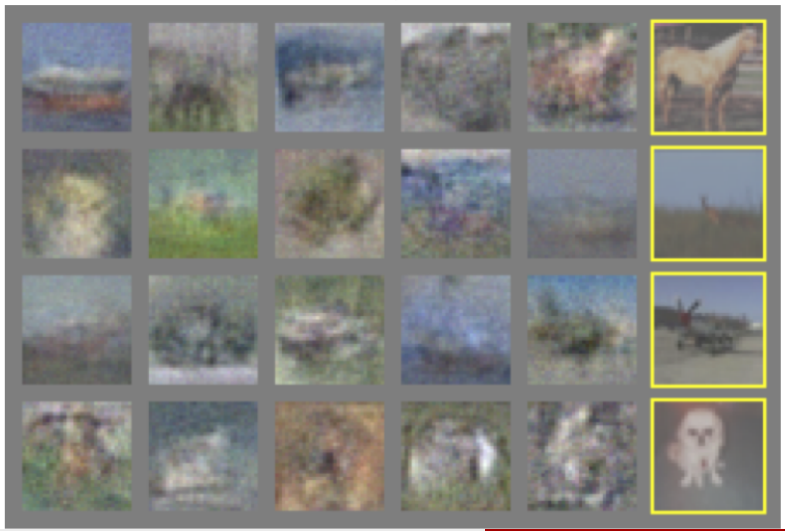


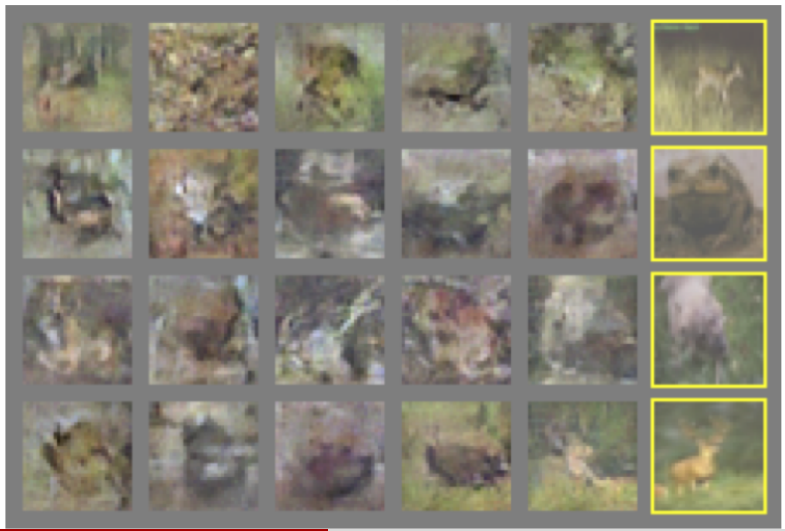
Good Examples

a)

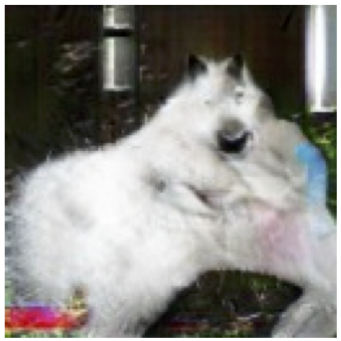
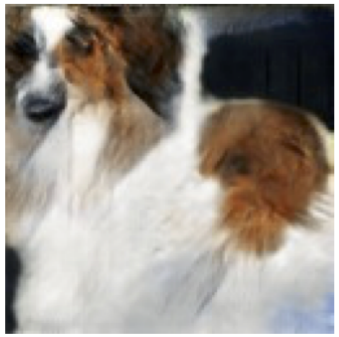


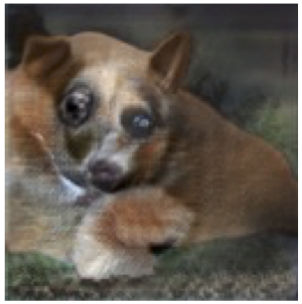
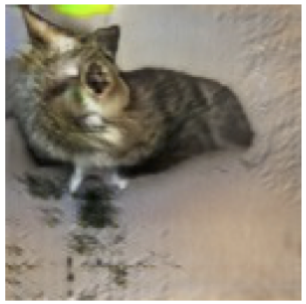
b)





Bad examples

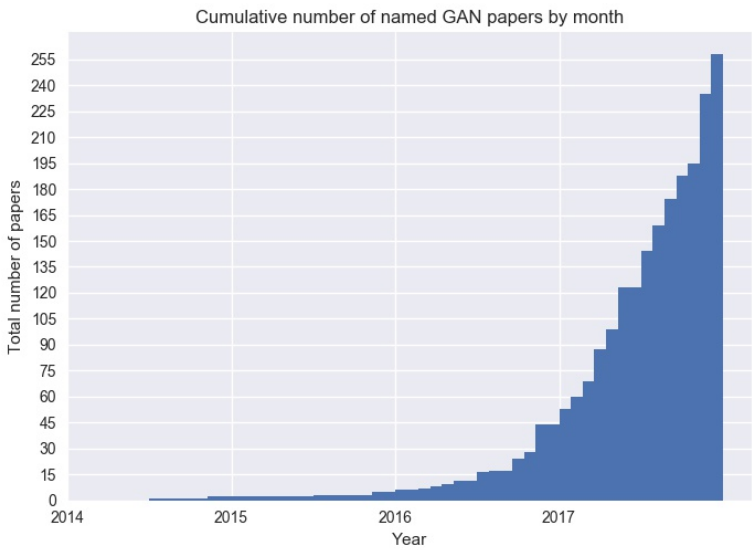
About GANs'

Known problems

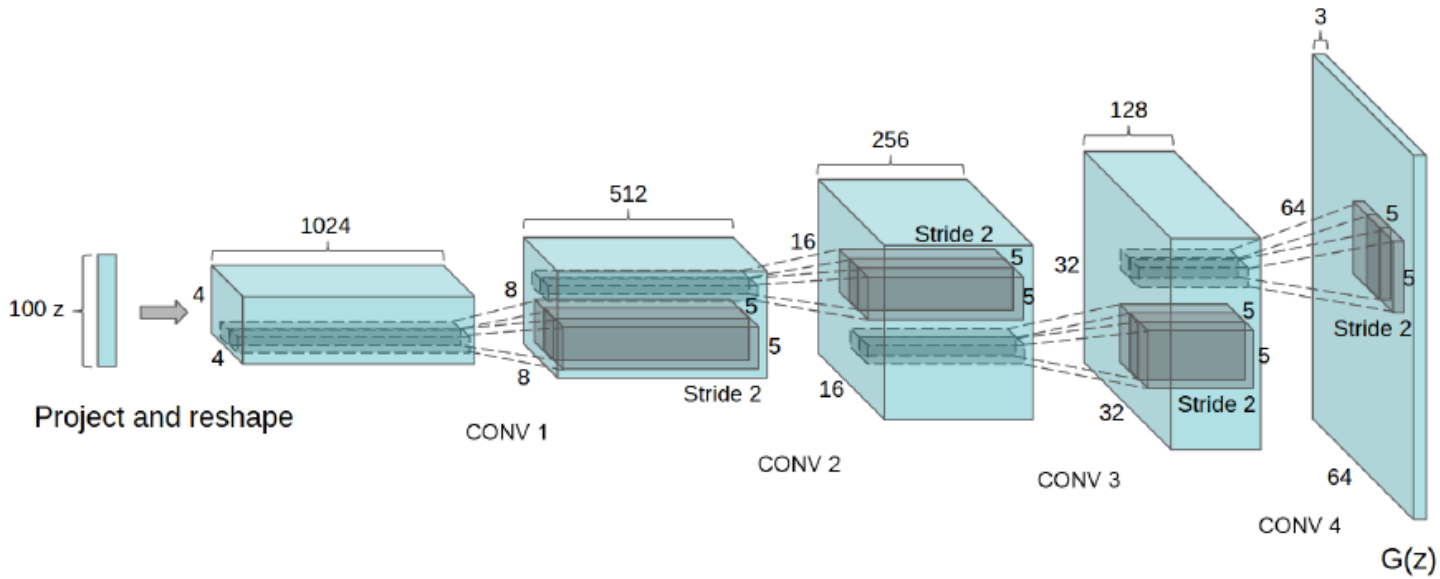
- DIfficult learning
- Very long learning
- Missing modes
- Evaluation measures

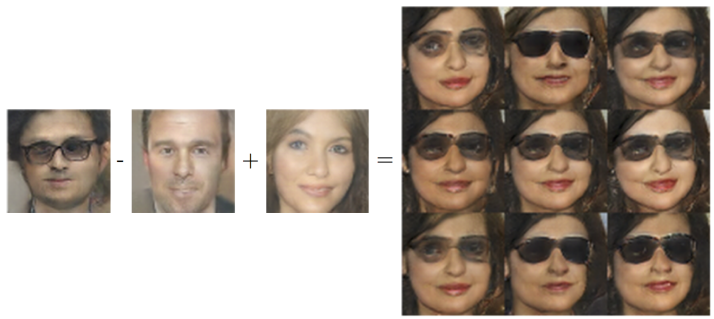
Many many variants

- Conditional
- Disantangling
- Image editing



DCGANS [Radford 2015]

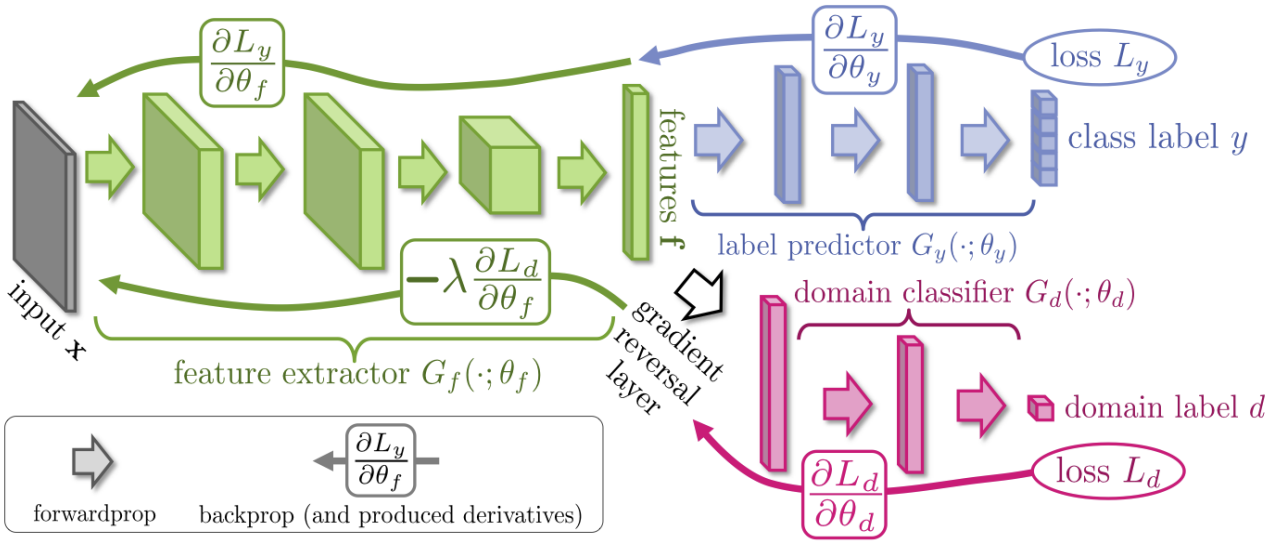




About using additional discriminators [Ganin et al, ICML 2015]

Other use of adversarial constraints

- Adversarial constraints may be used for structuring what we learn in hidden layers
- For instance : favoring some representation do not include a specific information

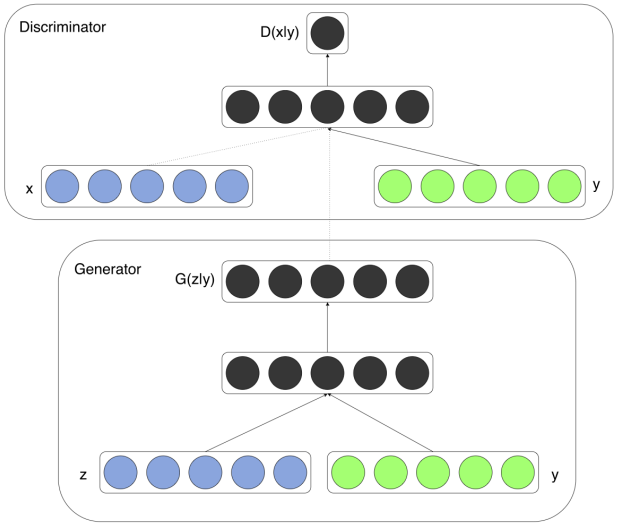


Conditional GANs [Mirza and al., 2014]

Learning criterion

- Citerion

$$min_g max_d v(\theta_g, \theta_d) = \mathbf{E}_{x,y\ p_{data}}[logD(x,y)] + \mathbf{E}_{z\ p_z, y'\ p_y}\Big[log(1 - D(G(z,y'),y'))\Big]$$



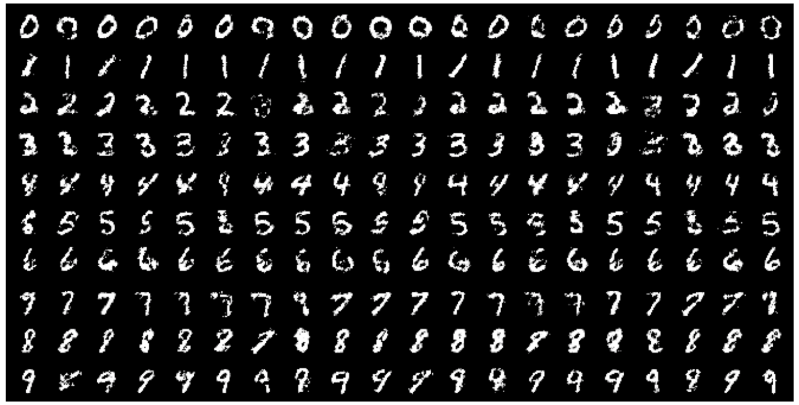
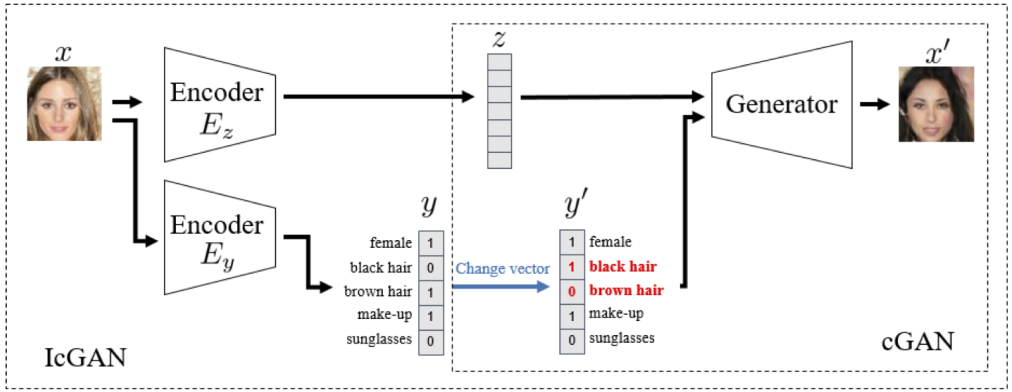
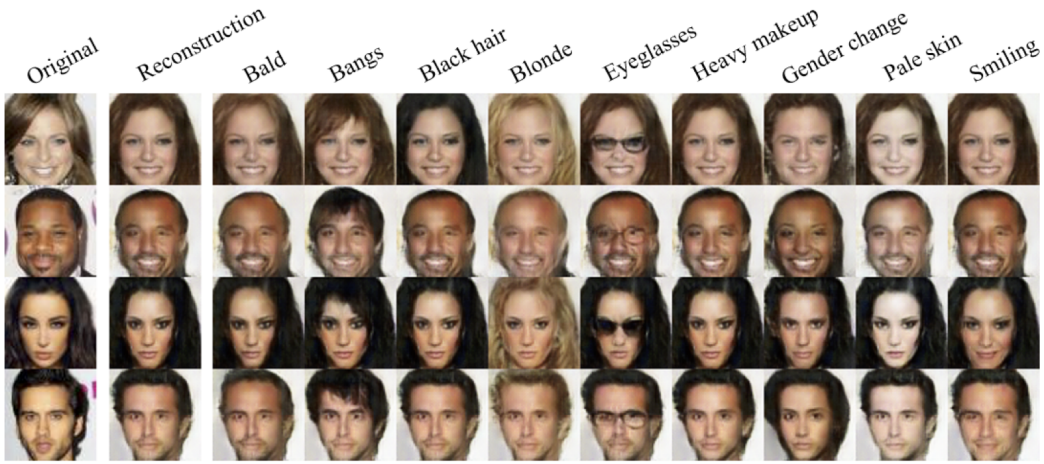


Figure 2: Generated MNIST digits, each row conditioned on one label

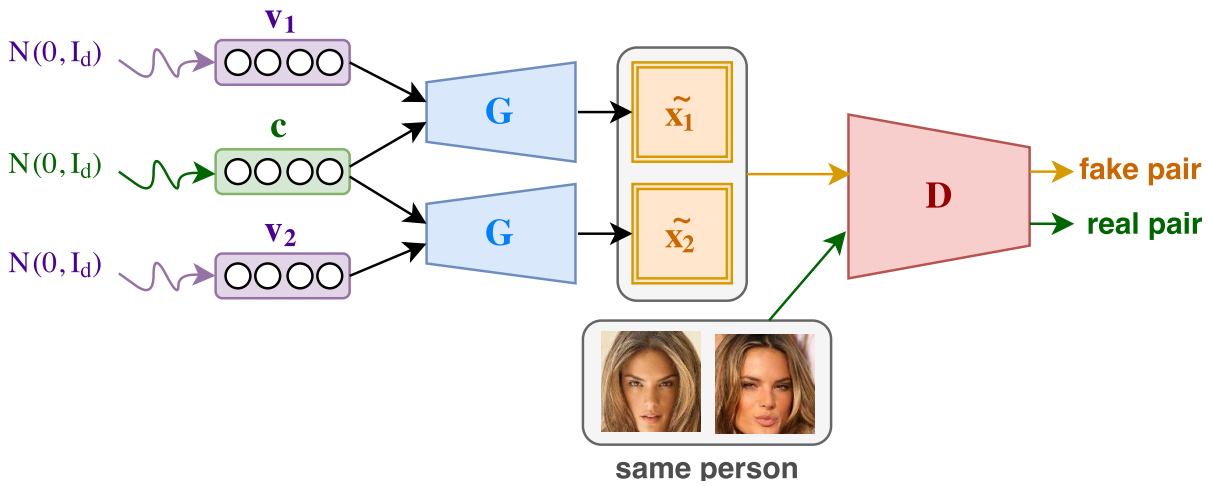
Image editing with Invertible Conditional GANs [Perarnau and al., 2016]

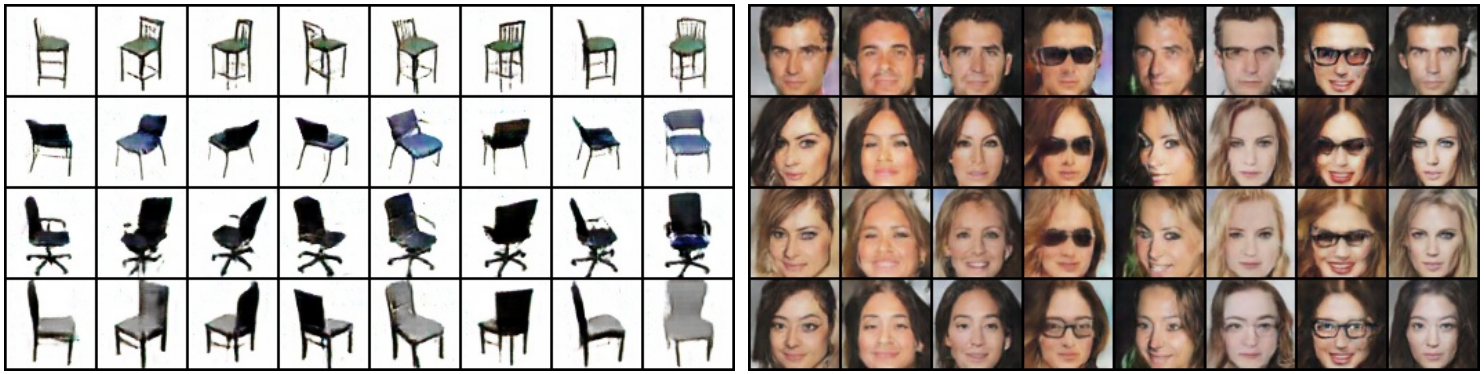




Disentangling factors of variation [Chen et al., 2018]

Generating images under various styles

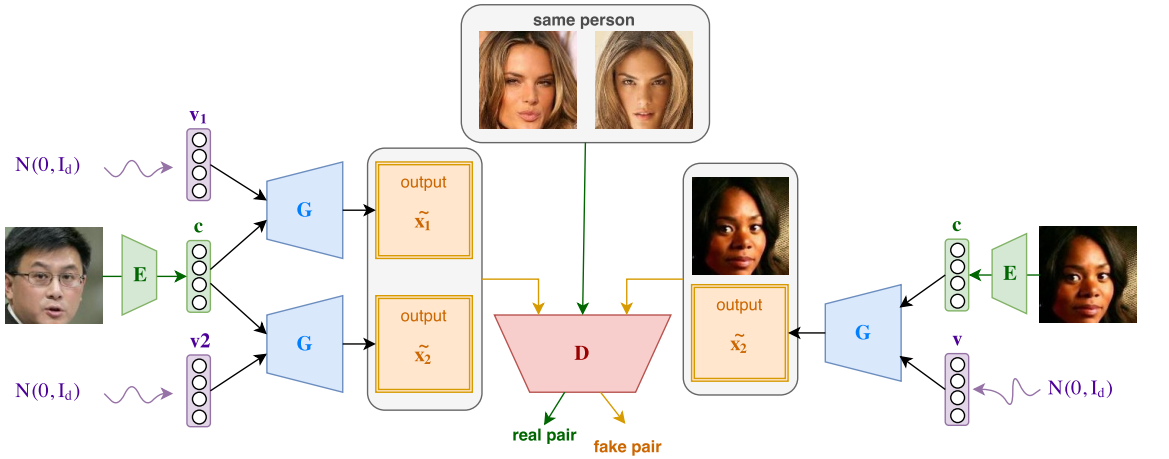


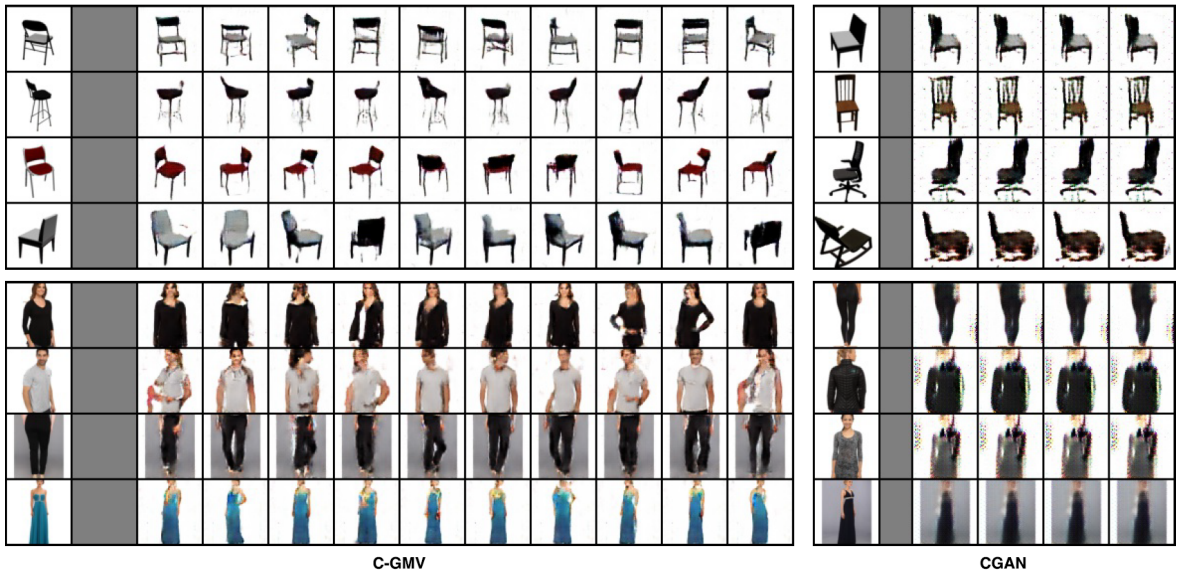


GMV

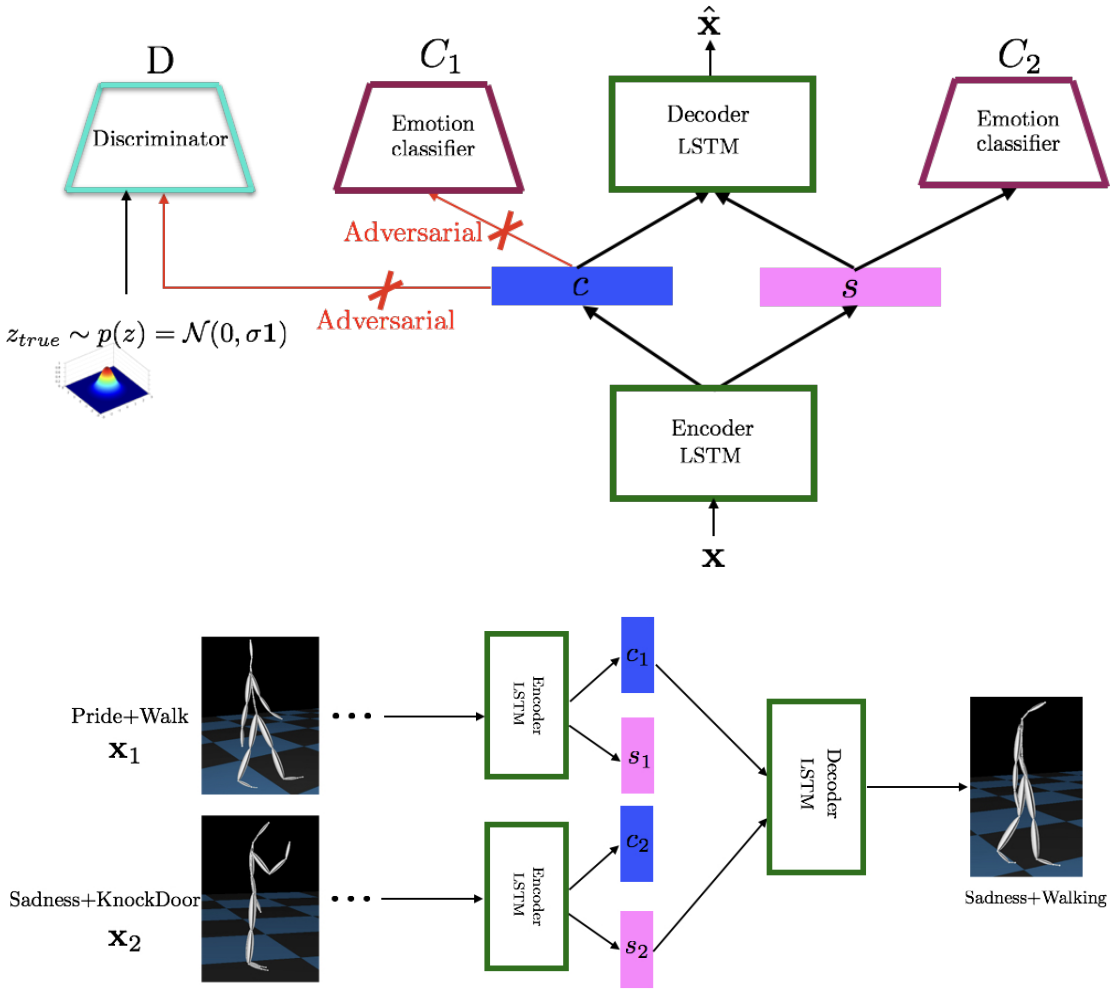
Disentangling factors of variation [Chen et al., 2018]

Transfering styles between images





Transfering styles between motion capture sequences [Qi et al., 2017]



Motion capture data