Jonas Glombitza
jonas.glombitza@fau.de

IDPASC School 2022
Olomouc, Czech Republic

Deep Learning: Advanced Techniques

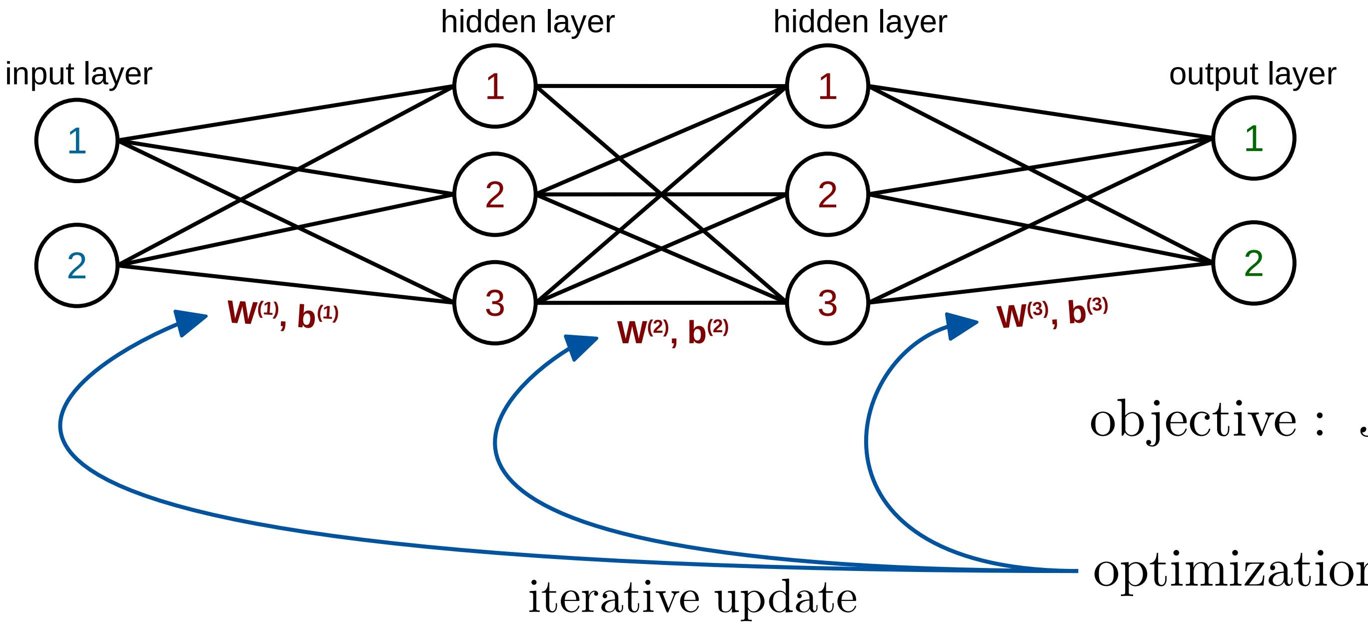
- Deep Convolutional Networks
  - normalization, shortcuts
- Unsupervised Learning
- Introspection

Recap: Deep Neural Networks

Feature Hierarchy: each new layer extract more abstract information of the data.

Probabilistic Mapping: learns to combine the extracted features

Train model (to find $\theta = \{W_i, b_i\}$ that minimizes objective) is automatic process.



adaptive parameters

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

output activation input

$$\text{objective}: \quad J(\theta) = \sum_i \left[y_m(x_i, \theta) - y_i\right]^2$$

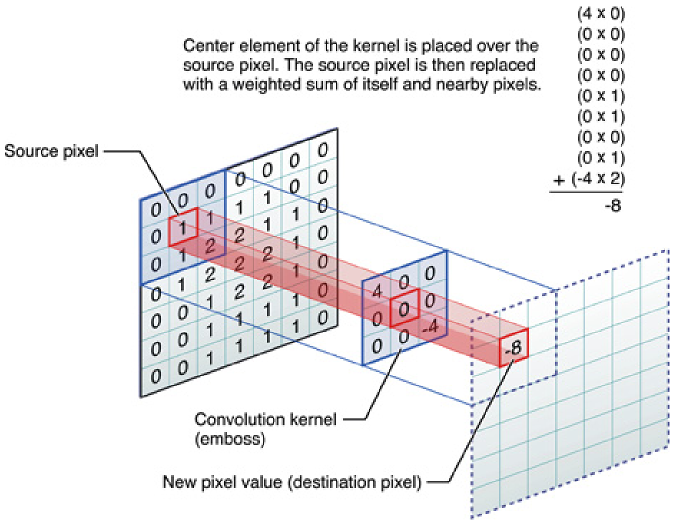
$$\text{optimization}: \quad \frac{dJ}{d\theta} \rightarrow 0$$

$$\boxed{\tilde{\theta} \rightarrow \theta - \alpha \frac{dJ}{d\theta}}$$

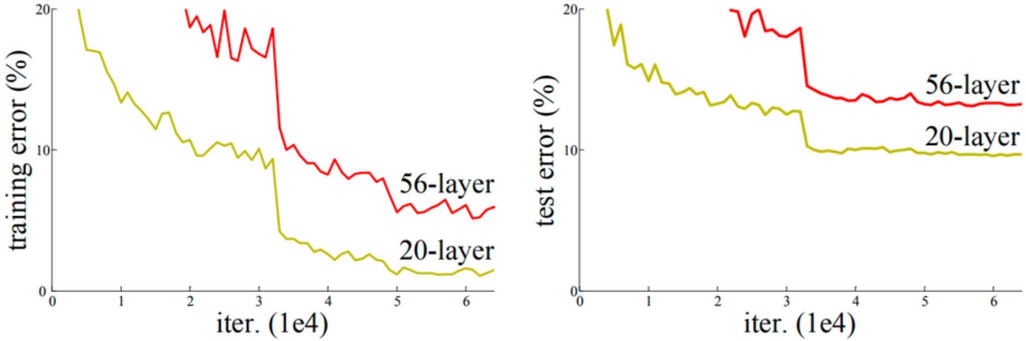
Recap: Convolutional Neural Networks

- 2D Convolution acts on 3D input (width x height x depth)
- Slide small filter over input and make linear transformation (dot product + bias)
- Hyperparameter:
  - Size of filter, typically (1 x 1), (3 x 3), (5 x 5) or (7 x 7)
  - Number of filters (feature maps)
  - **Padding** (maintain spatial extent)
  - **Striding** or **pooling** (reduce spatial extent)

- Reduction of parameters using symmetry in data:
  - Prior on **local correlations** (use small filters)
  - **Translational invariance** (weight sharing)



Obstacles when Going Deeper



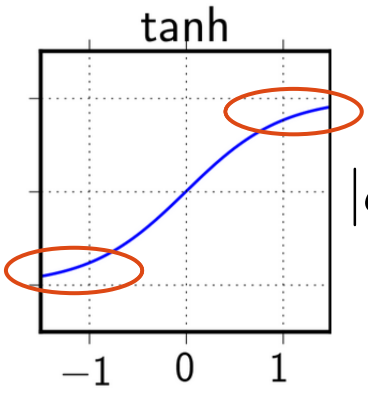
- Neural networks should get monotonously better when adding more layers
- **Problems**
  - Bad initialization
  - Vanishing gradients – gradients become too small
  - Shattered gradients – gradients become white noise
  - Internal covariate shift – need to constantly adapt changes in earlier layer
  - Convolutional filter show redundant behavior → advanced CNN operations

Vanishing Gradient Problem

$$z \xrightarrow{w_1} \overset{\sigma(x_1)}{b_1} \xrightarrow{w_2} \overset{\sigma(x_2)}{b_2} \xrightarrow{w_3} \overset{\sigma(x_3)}{b_3} \xrightarrow{w_4} \overset{\sigma(x_4)}{b_4} \xrightarrow{w_5} \overset{\sigma(x_5)}{b_5} \twoheadrightarrow y$$

$$y = \sigma(x_5) = \sigma(w_5 \cdot \sigma(x_4) + b_5) = \sigma(w_5 \cdot \sigma(w_4 \cdot \sigma(x_3) + b_4) + b_5)....$$

$$\frac{\partial y}{\partial w_1} = \frac{\partial \sigma(x_5)}{\partial x_5} \frac{\partial x_5}{\partial \sigma(x_4)} \frac{\partial \sigma(x_4)}{\partial x_4} \frac{\partial x_4}{\partial w_1} .... = \underline{\sigma'(x_5)} w_5 \cdot \underline{\sigma'(x_4)} w_4 .... \cdot \underline{\sigma'(x_1)} w_1$$



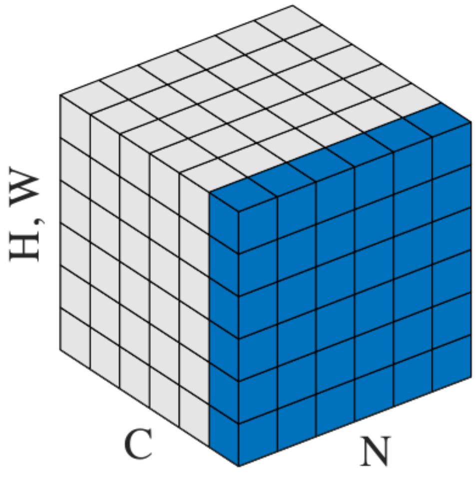
$$|\sigma'(x)| << 1$$

- Stacking many layers can lead to vanishing gradients
- Activation saturates
  - Updates in early layers become very tiny
    - **No learning**
  - Don't use sigmoids / tanh only rarely → use shortcuts
    - still useful as last layer if data ranges [0;1] or [1:-1]

Batch Normalization

- Calculate batch-wise for each channel:
  - Mean: μ_B and Variance: σ_B^2
  - Add free parameters $\gamma,\ \beta$
    - Easily control first moments of distribution

➢ $$y = \frac{x - \mu_B}{\sigma_B}\gamma + \beta$$

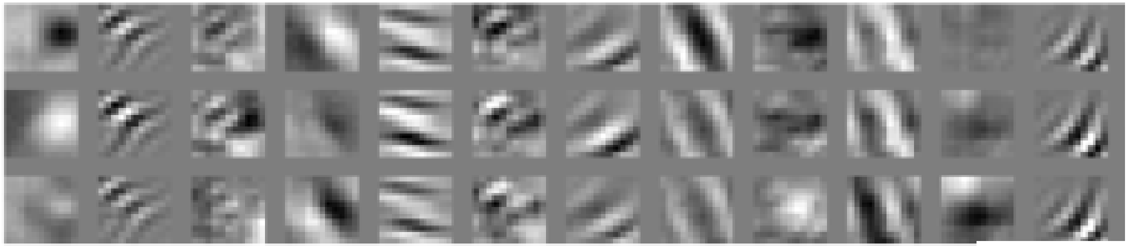


- Makes DNN robust against poor initializations
- Helps with vanishing gradient / less sensitive to high learning rates
- Has regularizing effect (no large weights, noise because of batch dependency)
- Reduce internal covariate shift

➢ **Very successful for convolutional architectures**

Inception Module

Key observation: Convolutional filters show redundant behavior

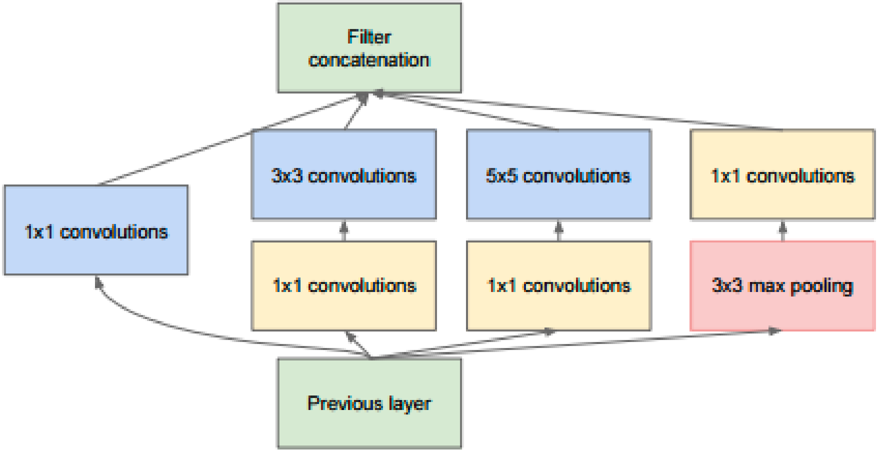


Idea: Factorize convolution operation

- Use different small convolutions in parallel and concatenate outputs
- Massive use of (1 x 1) convolutions

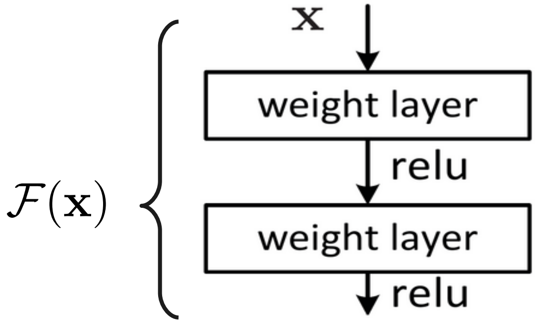
➢ Increase model complexity

➢ Make model sensitive to different scales

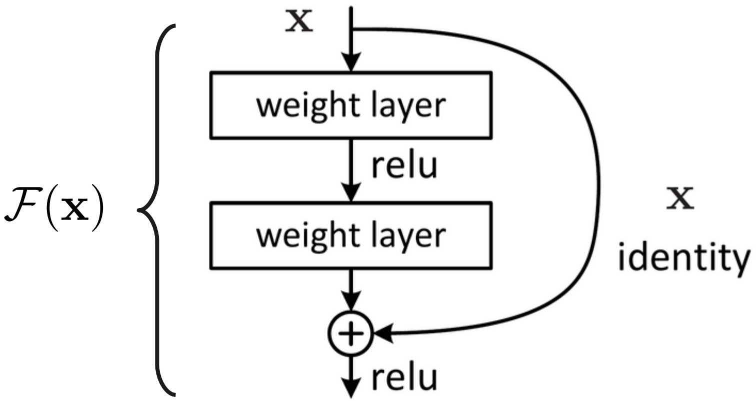


Residual Unit

Idea: Residual unit consisting of small network and a shortcut (identity mapping)

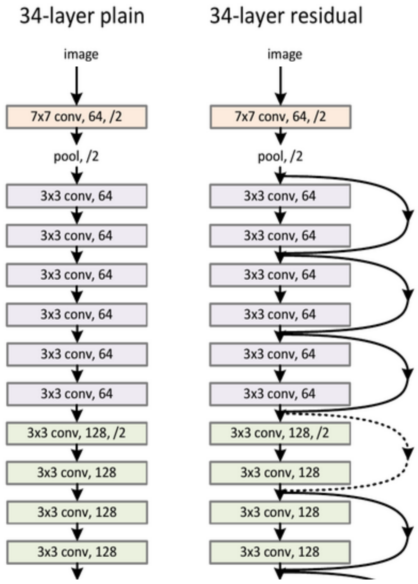


$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x})$$



$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

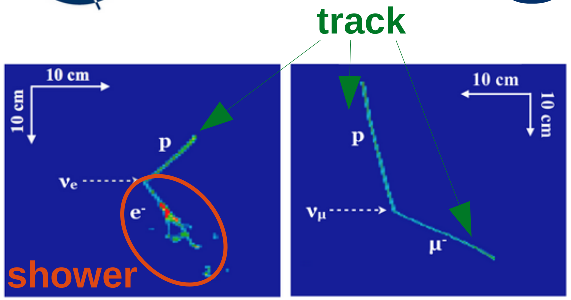
- Weight block learns small residual $\mathcal{F}(\mathbf{x})$ on top of input $\mathbf{x}$
  - Output of residual unit $\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$
- Shortcut let gradient propagate easily to earlier layers
- Later layers can easily turn weights to zero by $\mathcal{F}(\mathbf{x}) \rightarrow 0$

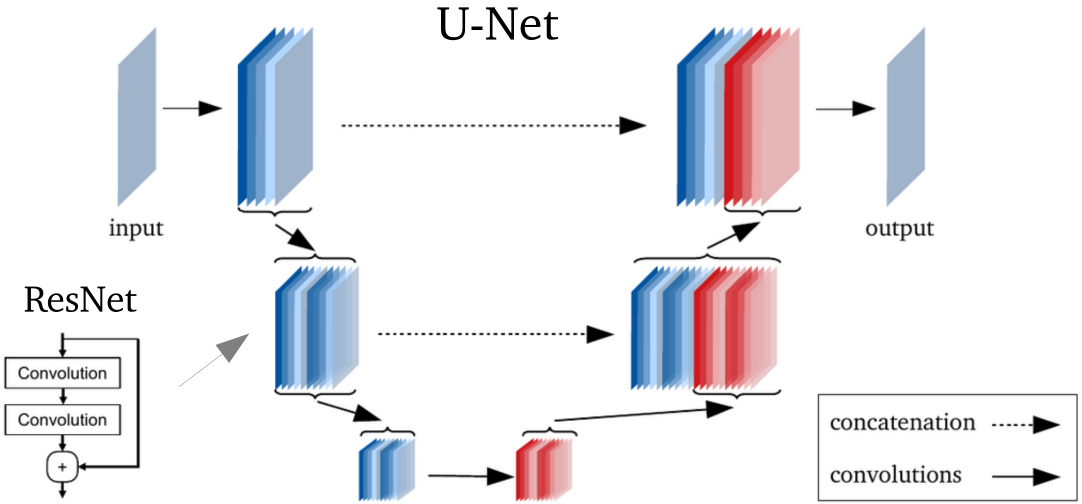


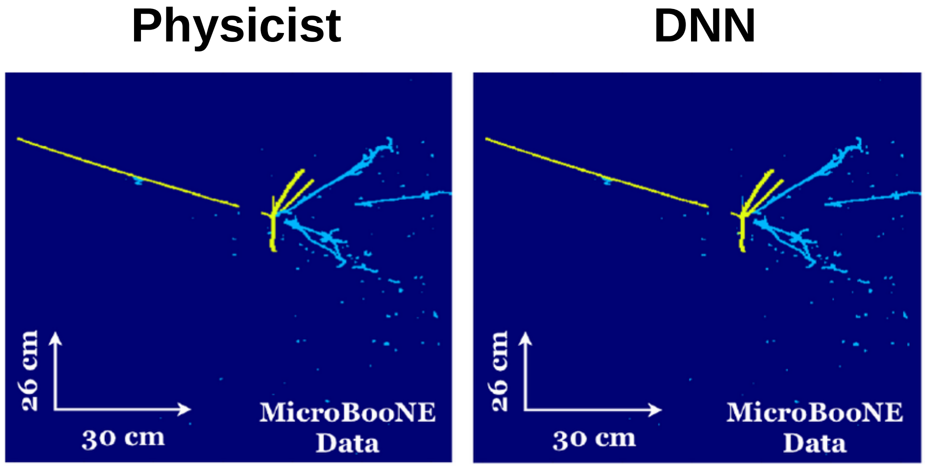
Up to several of hundreds layer deep!

Application: MircroBooNE

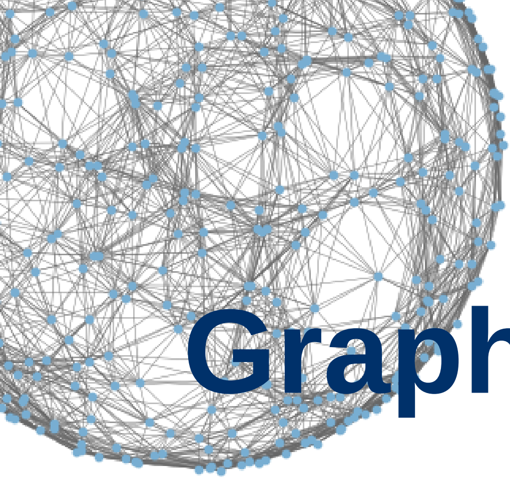
- Liquid Argon TPC for neutrino detection
- Segmentation (pixel-wise class prediction) into tracks and electromagnetic-showers
  - Architecture: combination of ResNet and U-Net
- Incorrectly classified pixel fraction per image ~ few percent





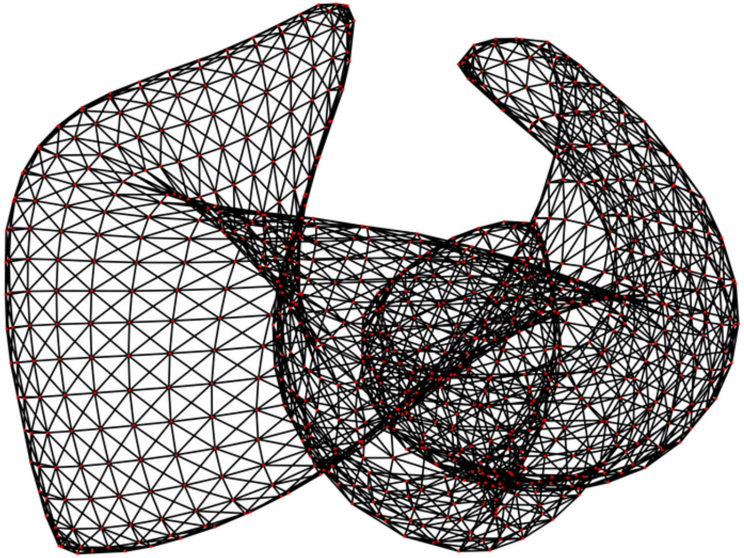


Adams et al. ArXiv: 1808.07269



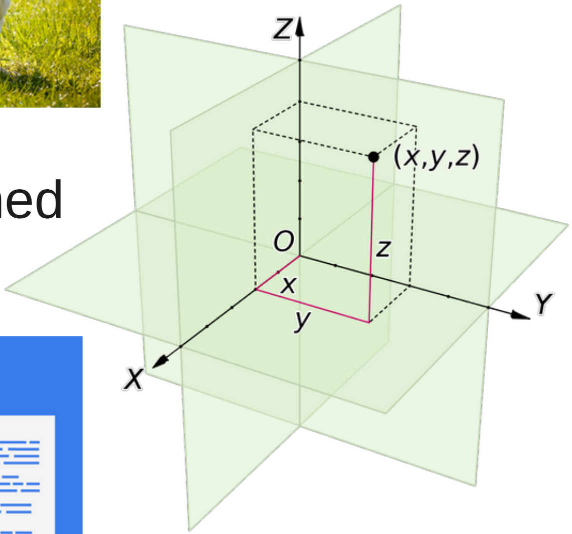
Graph Convolutional Networks

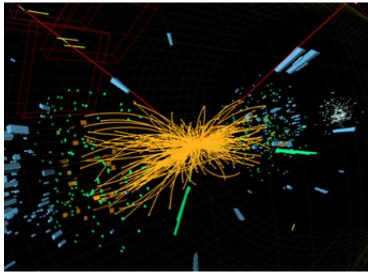
- Graphs and graph basics
- Convolutions on non-Euclidean domains
- Graph Convolutional Neural Networks
  - Spatial domain
  - Spectral domain

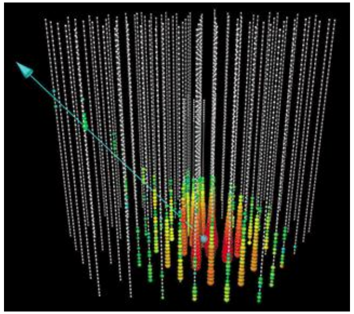


Convolutions and Datasets

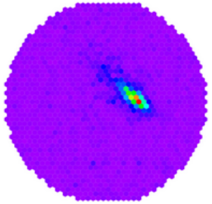
- Works in well defined euclidean space

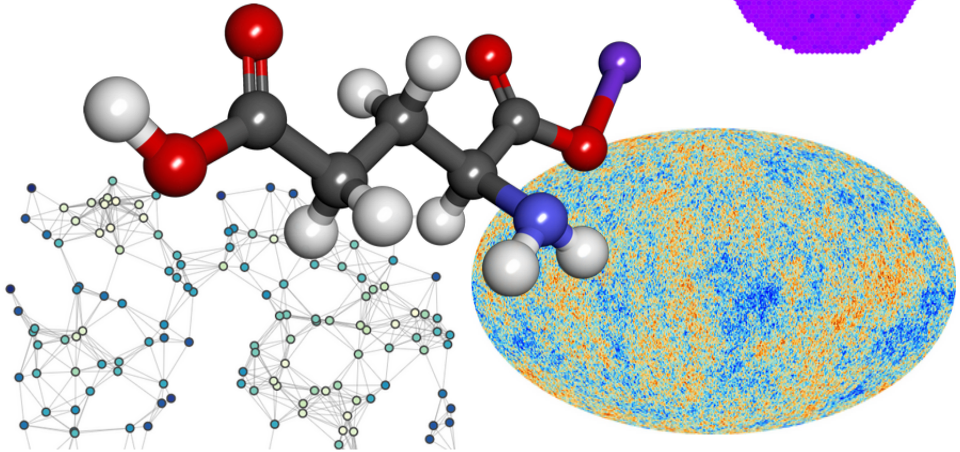






- physics data often feature different geometries

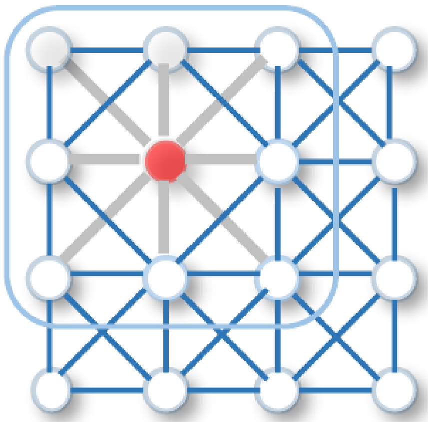




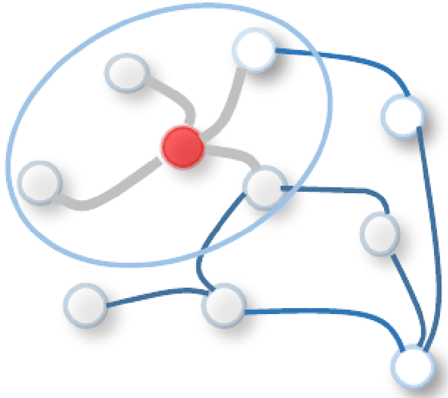
Convolution in Spatial Domain

- Images with discrete and continuous pixel coordinates

regular grid: equidistant positions



continuous grid positions

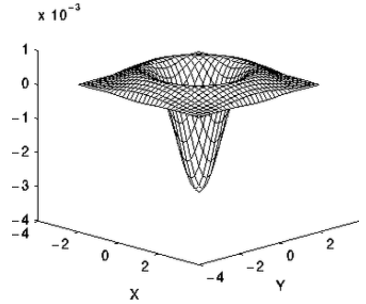


- Learned filter

$$\mathbf{D}^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

……?

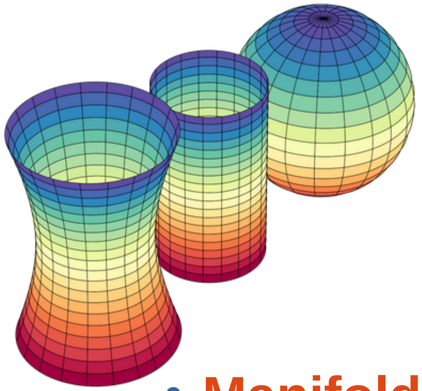
Transition of discrete filter to continuous filter



https://arxiv.org/abs/1901.00596

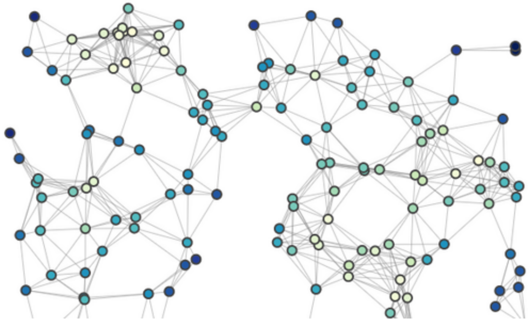
Non-Euclidean Domains

- Defining convolutions, challenging on non-euclidean domains
  - Deformation of filters, changing neighbor relations
  - Non-isometric connections on graphs



- **Manifolds**

source: wikipedia



- **Graphs**

source: Cody Marie Wild, Towards Data Science

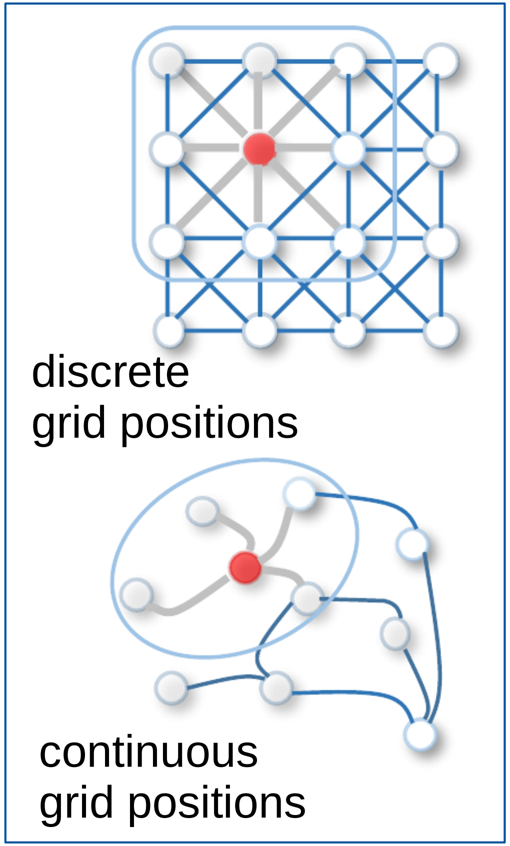
Image-like data

- collection of pixels (vector)
- coherent (rarely sparse)
- discrete, regular (symmetric)
- feature euclidean space

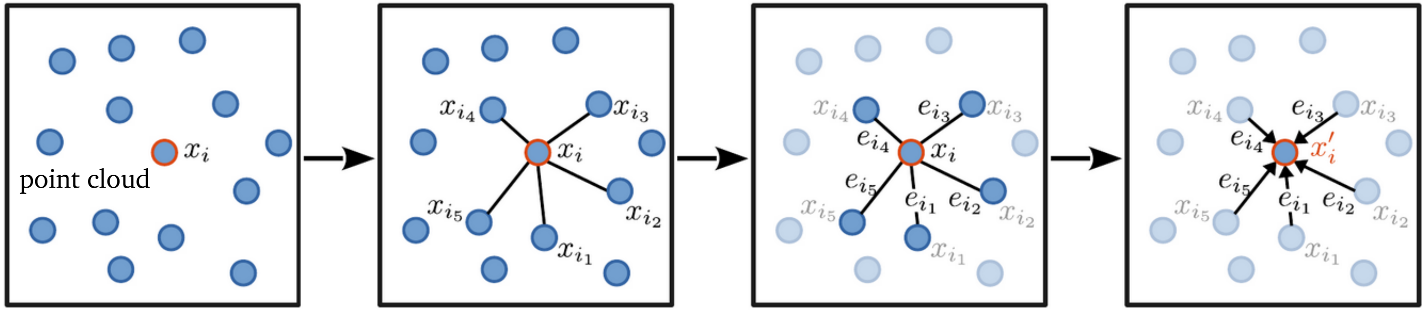
How can we generalize convolutions?

Dynamic Edge Convolution

Y.Wang et al,
https://arxiv.org/abs/1801.07829



- define continuous filter (using kNN)
- flexible for many settings: irregular structures, point clouds
- dynamically ‘adapt’ fundamental graph structure each layer



construction of directed graph

estimation of edge features

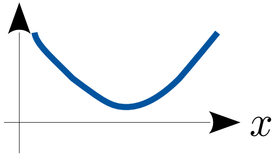
aggregation over neighborhood

→ search k nearest neighbors

$$e_{i_j} = h_\theta(x_i, x_{i_j})$$

$$x_i' = \square_{j=1}^{k} e_{i_j}$$

e.g. $x_i' = \sum_{j=1}^{k} e_{i_j}$

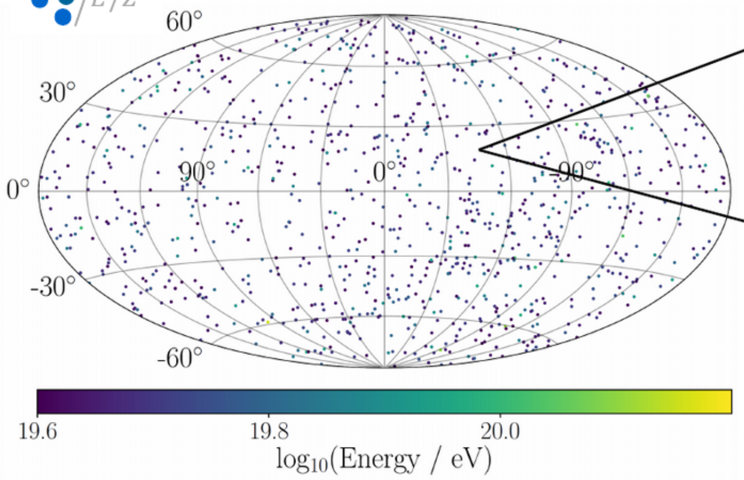


approx. by DNN

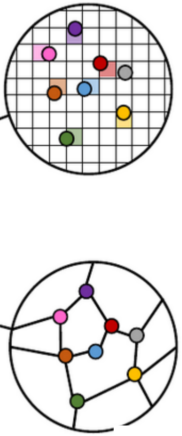
Application: Search for UHECR Origins

Slide credit: Niklas Langner

Deflection depends on energy charge and GMF



Continuously distributed on **sphere**



sparse, spherical not suited for CNN

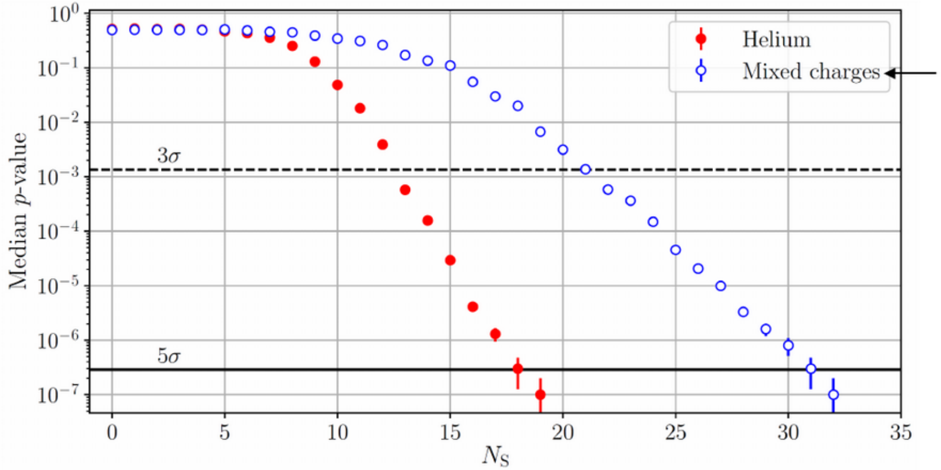
use Dynamic Graph Network

Situation:

<u>One</u> measured sky (spherical)

Learn to classify between

- isotropic sky / signal
- use dynamic edge convolutions

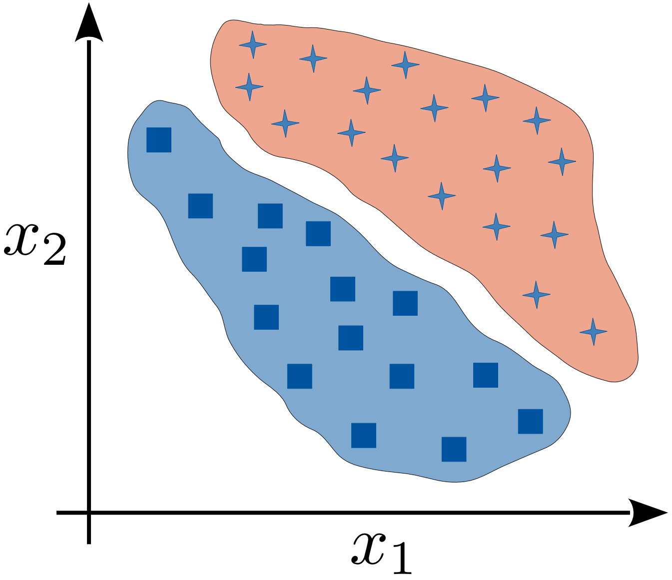


15% H
45% He
40% C, N, O

Bister et al., 10.1016/j.astropartphys.2020.102527

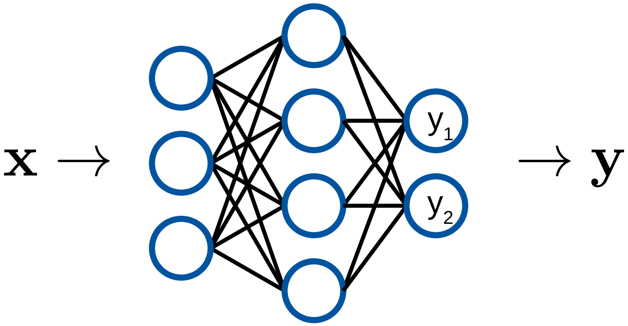
From Supervised to Unsupervised Learning

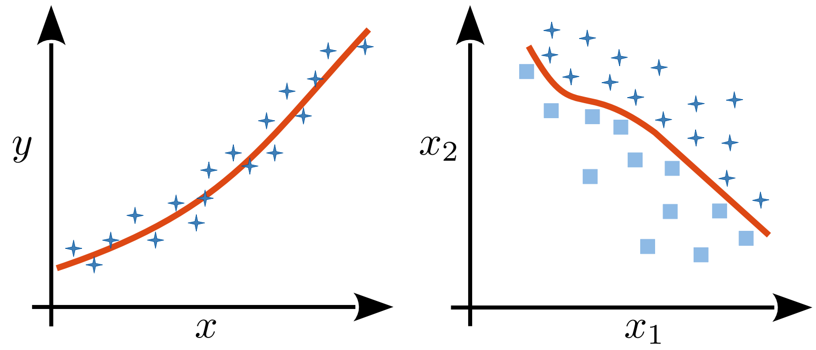
- Unsupervised Learning
    - Autoencoders
    - Generative Adversarial Networks



Supervised Learning

- Situation
  - Large labeled data set (pair of input $\mathbf{x}$ and output $\mathbf{y}$)
- Typical Task:
  - Learn function to map input to specific output
  - Train model to predict the associated label
  - Achieve best generalization performance
  - Infer *conditional* probability density $p(\mathbf{x}|\mathbf{y})$





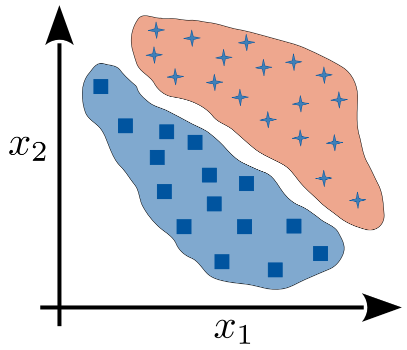
Unsupervised Learning

- Typical Situation: non labeled data set

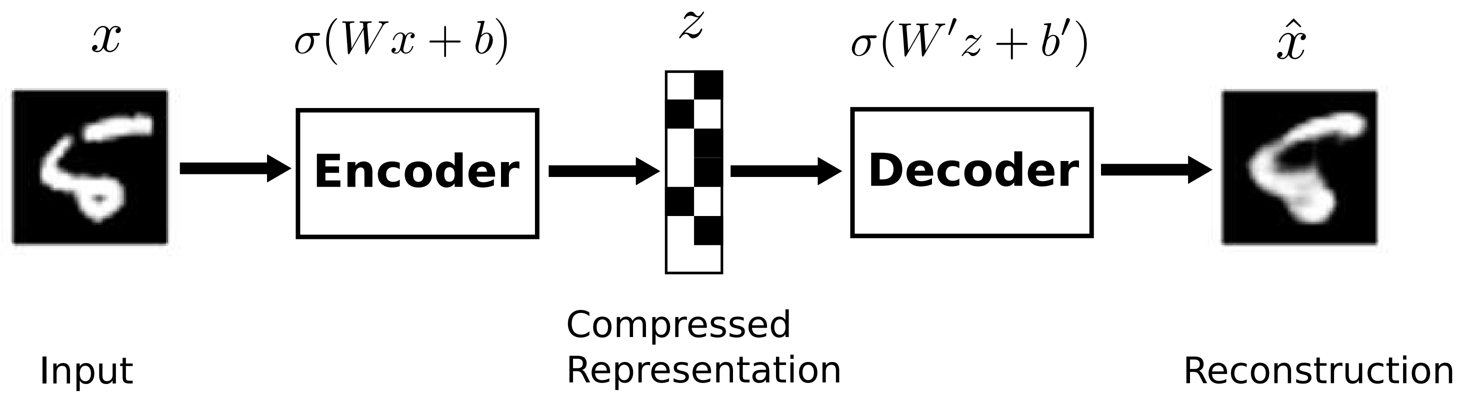
- Tasks:
  - Learn (low dimensional) data encodings → *autoencoders*
  - Estimate underlying probability density → *generative models*
  - Clustering, anomaly detection – find (non-) similar samples

- Infer *a priori* probability density $p(x)$

- Models typical trained without label information
  - Contrast: semi-supervised learning



Autoencoders



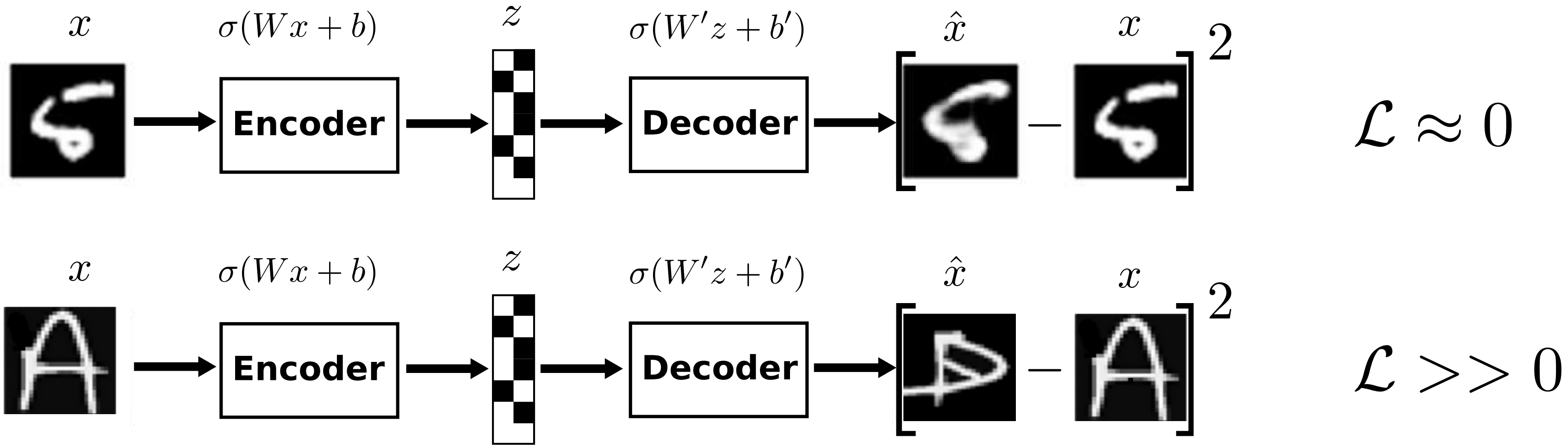
- Reconstruction of input data (approximation of identity function)
- Learning interesting representation (constraints to hidden layer)
- Objective function:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{x}}_i - \mathbf{x}_i)^2$$

- Deep autoencoders often show underfitting → use shortcuts!

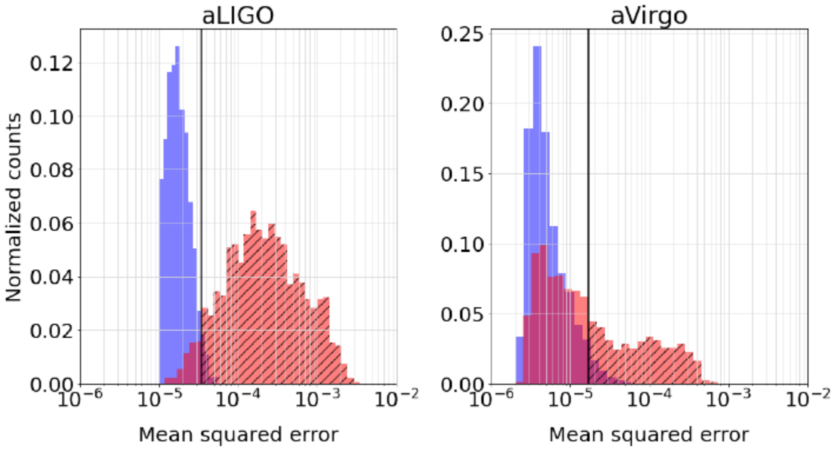
Anomaly Detection

- Train autoencoder on given data set
  - Model learns dimensionality reduction for given data
- Apply autoencoder to unknown data
  - For known phase-space → good reconstruction
  - For unknown phase-space → bad reconstruction → **anomaly**

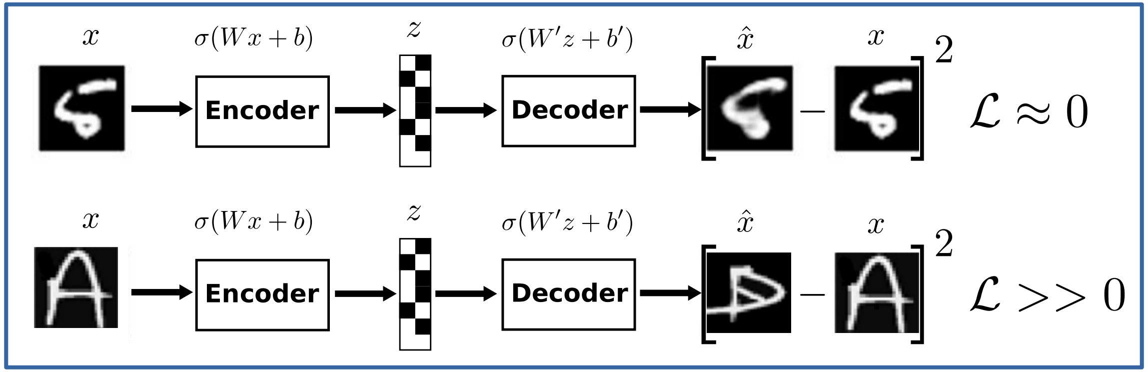


Application: Anomaly Detection

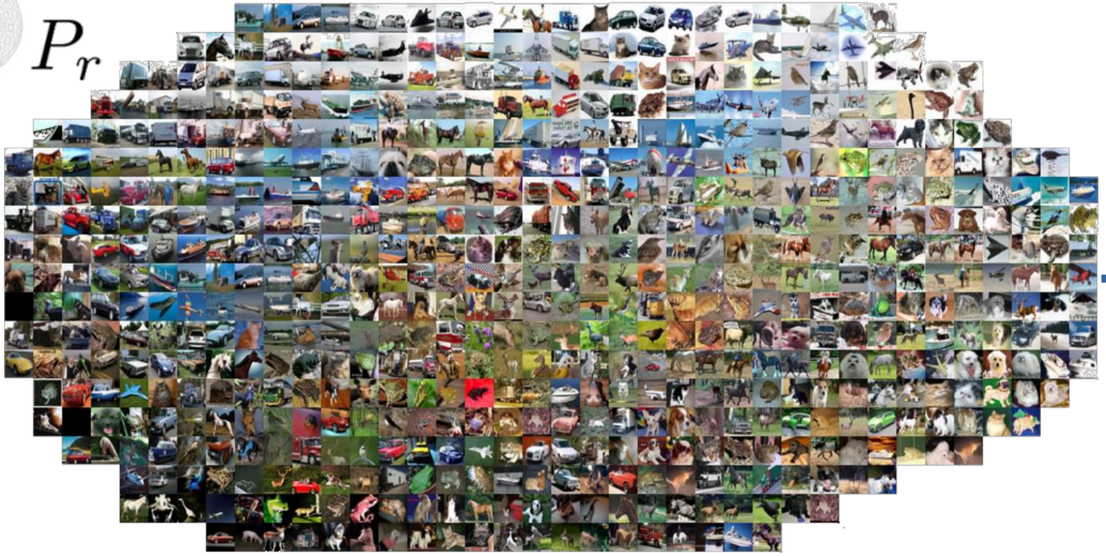
- Search for data, different than used for training, using autoencoders
- indication for new physics, proposed for BSM searches at LHC
- training without limited data (no signal labels)
  - first approaches in astroparticle physics
    - detection of gravitational waves



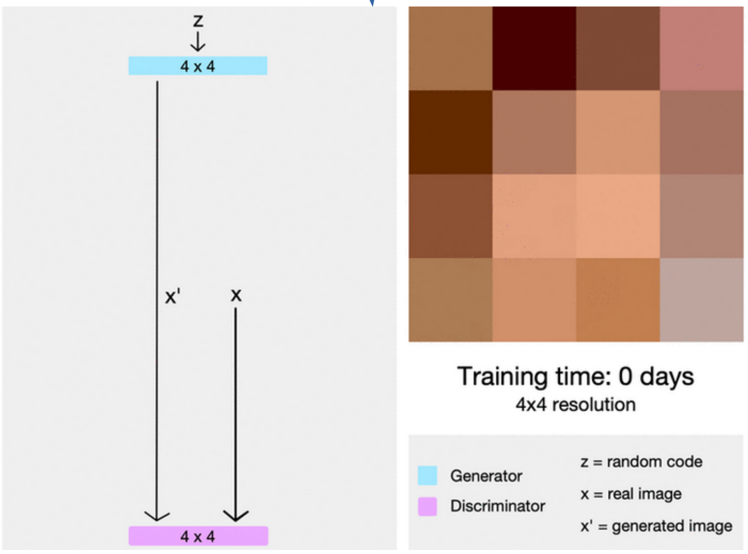
F. Morawski et al., Mach. Learn.: Sci. Technol. 2 045014



P_r



learn to generate new samples

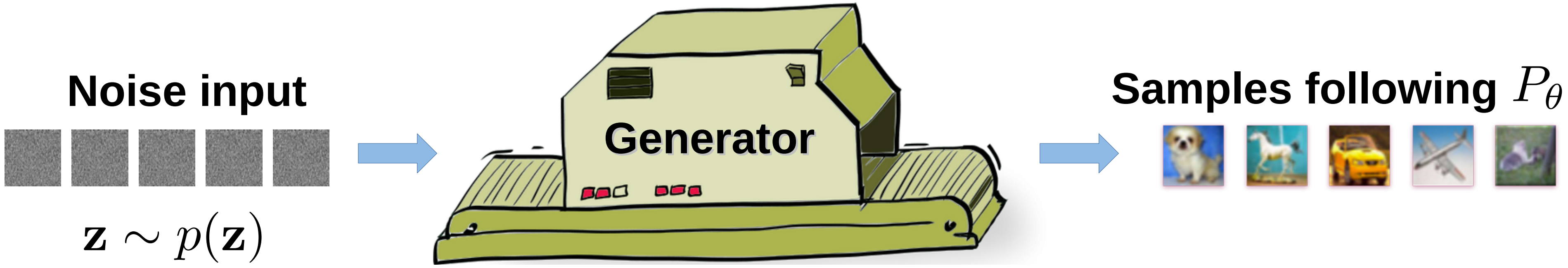


Generative Models

How to train a Generator

- Objective: learn to generate new samples following P_θ
- Learn a function that transform a distribution $p(\mathbf{z})$ into P_θ using a generator G_θ

  $\mathbf{z} \in Z \rightarrow$ latent space
- Generator G_θ is implemented as neural network with weights θ



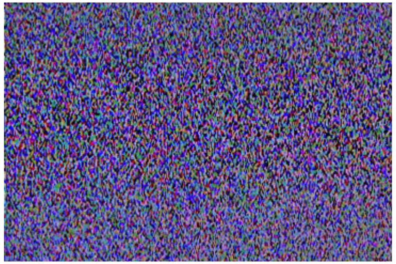
Manifold Hypothesis

Idea: Manifolds of meaningful pictures are highly concentrated with very little volume and embedded in a very high dimensional space

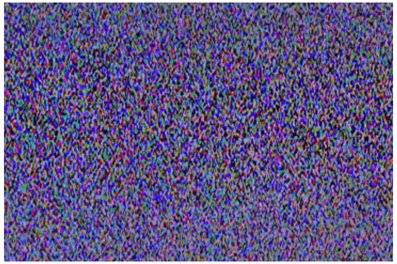
- Generation of images is a very challenging task
- Data (correlations / densities) are high dimensional

Example: Try to randomly generate images:

Goal



Sample 1



Sample 100,000

You will even never reach this “neighborhood sample”

“To deal with a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it.” - G. Hinton

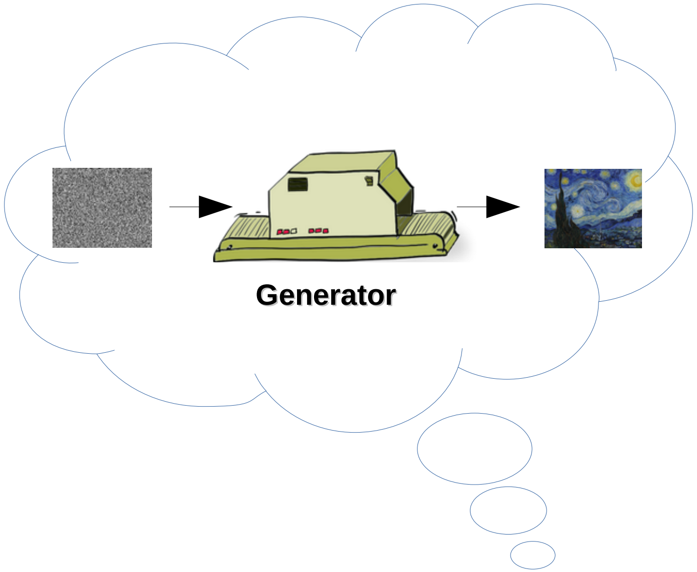
Generative Adversarial Networks

- Hard to formulate a supervised training loss

- Use **unsupervised training** to train the generator
  - Objective: $P_\theta \approx P_r$
  - Measure: given by **second neural network**

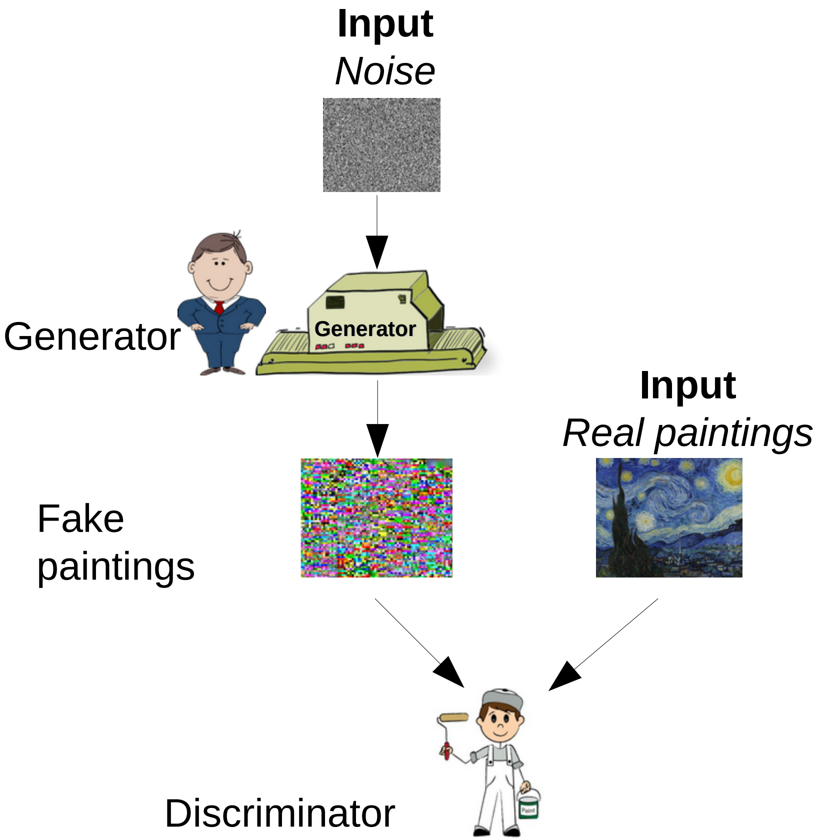
→ Generated samples of generator should be similar to real samples after training
  - without reproducing training data

→ **Adversarial approach:**
Train 2 networks adversarial (against each other)

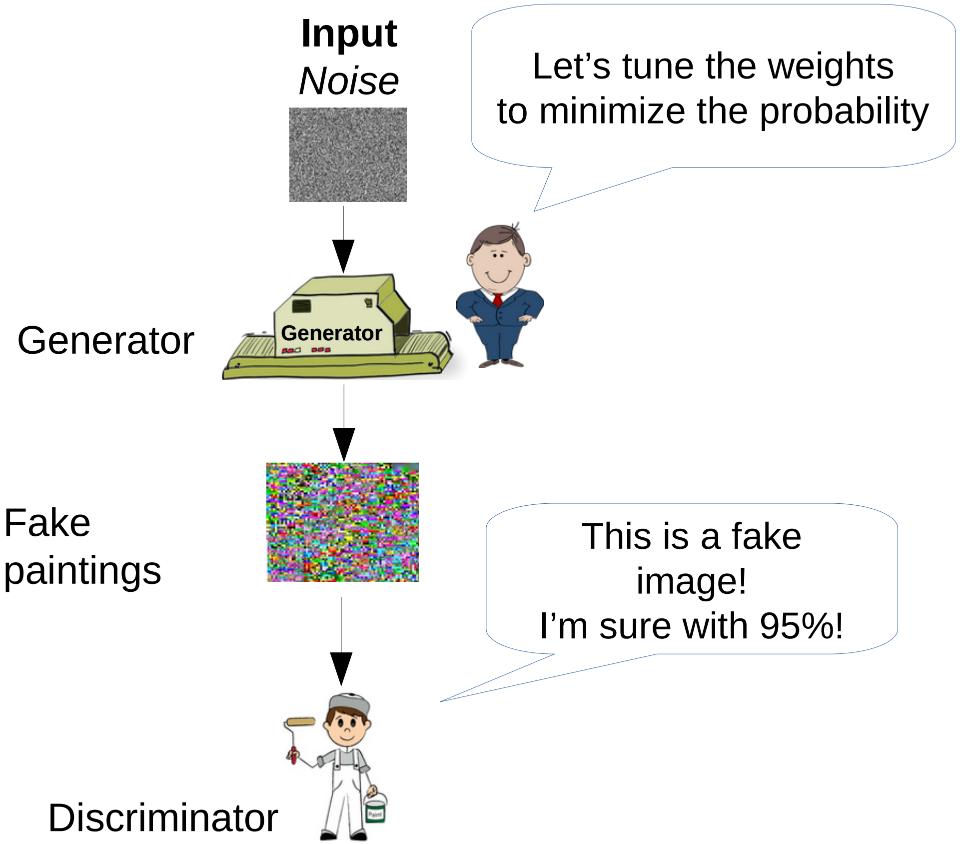


Art forger
Wants to create some fake images

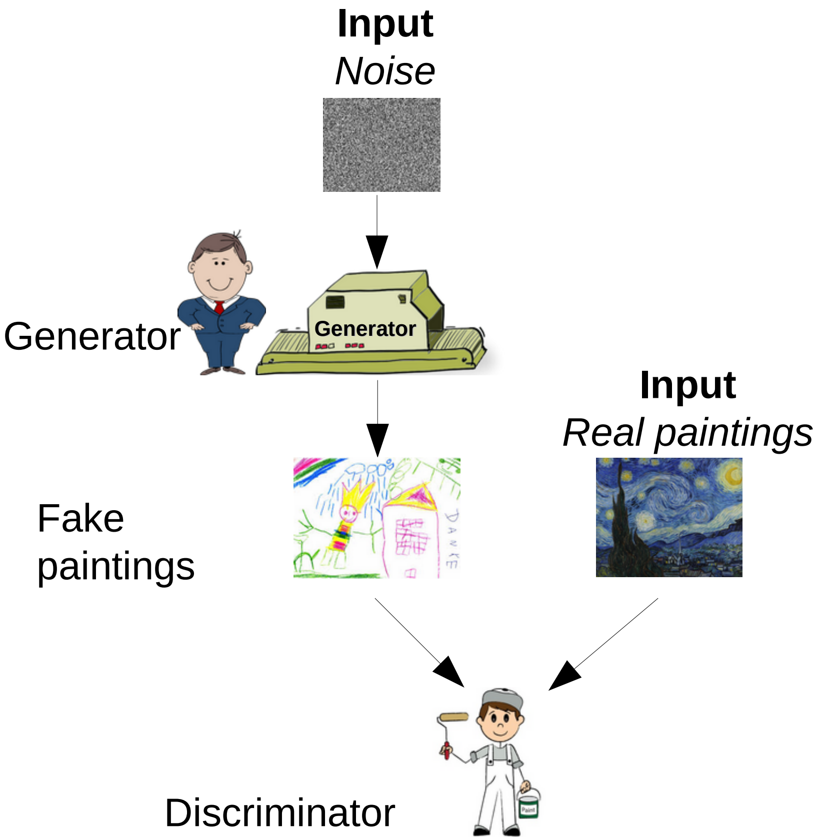
Train Discriminator



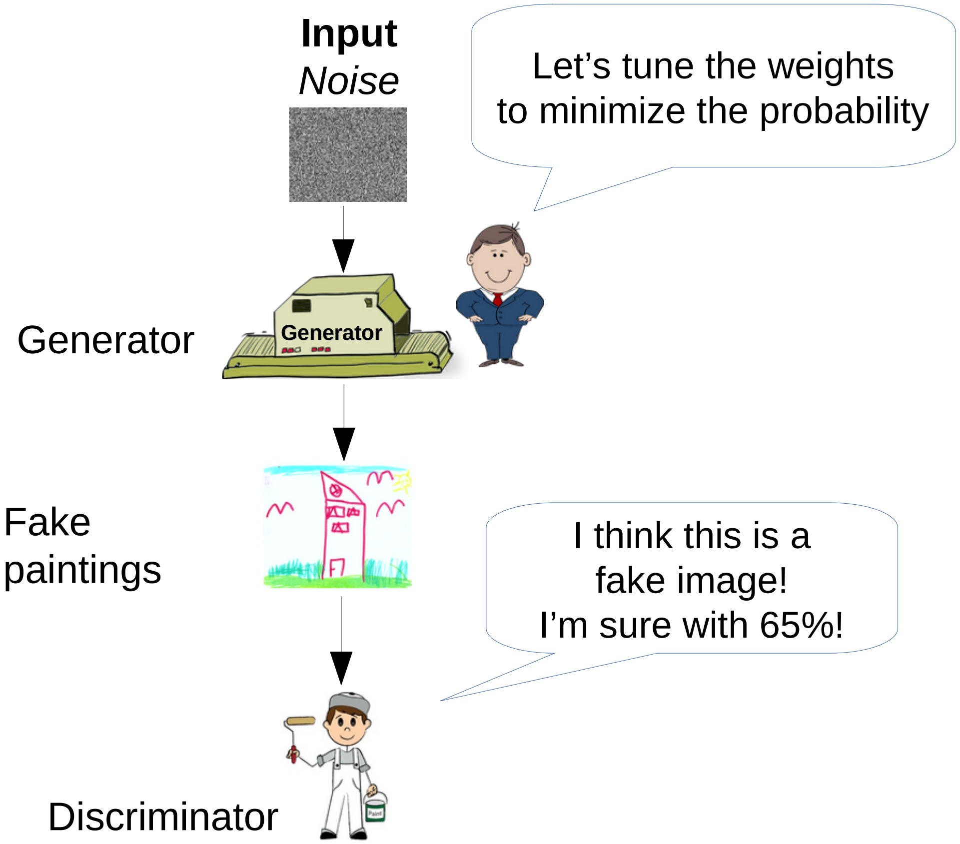
Train Generator



Train Discriminator



Train Generator

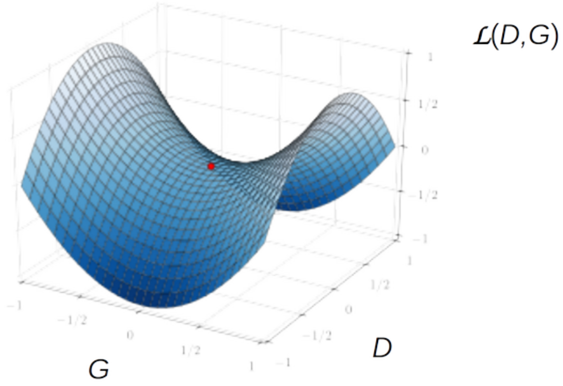


GAN Training

$$\min_G \max_D L(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1-D(G(\mathbf{z})))]$$

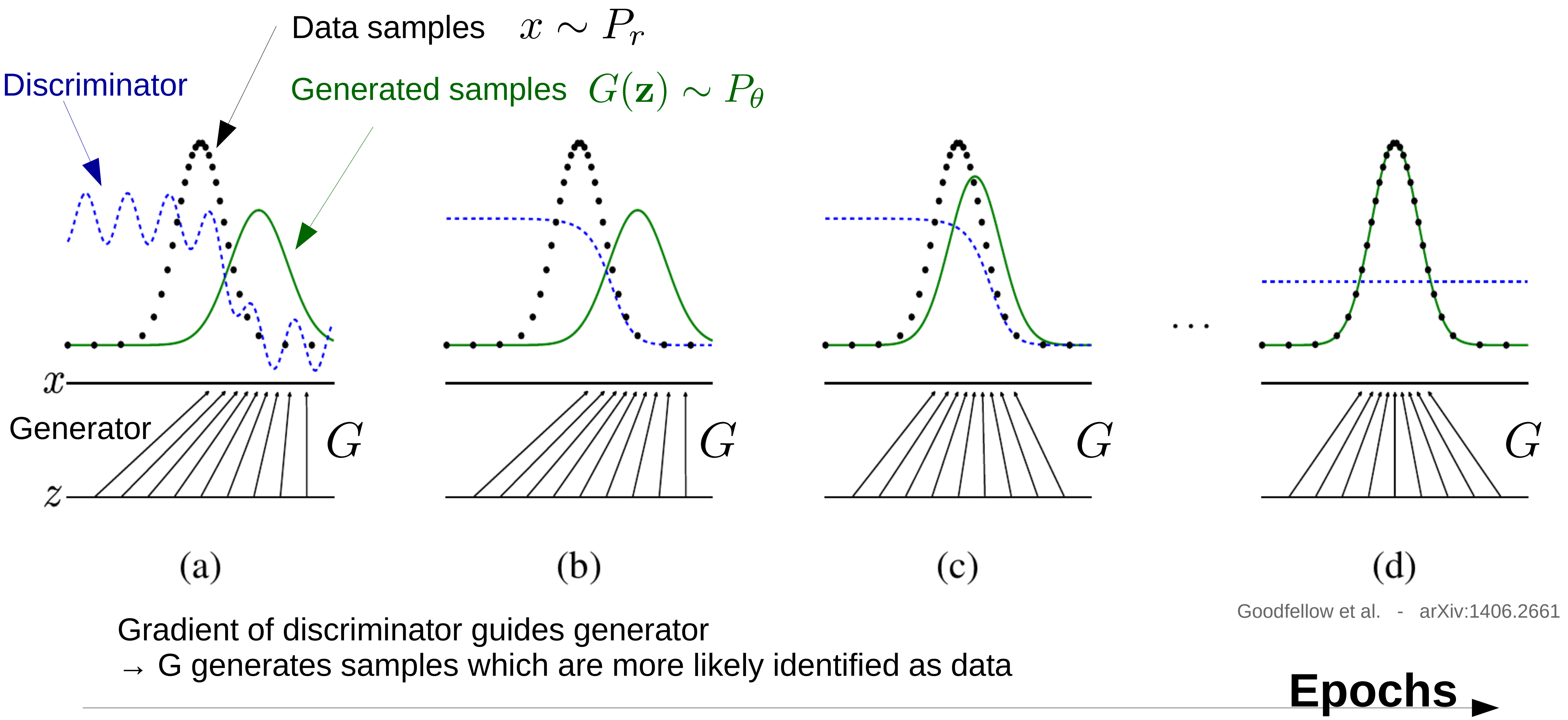
Training 2 networks at the same time is challenging
Losses of discriminator and generator are highly dependent

I. Train generator and discriminator alternating
   - Min/Max game
   - Sum of both players is zero



II. Finding Nash equilibrium is hard
   - Discriminator and generator need to have same quality
   - Minimize Jensen-Shannon divergence (assume optimal discriminator)

Optimal Evolution of GAN Training



Goodfellow et al. - arXiv:1406.2661

Gradient of discriminator guides generator
→ G generates samples which are more likely identified as data

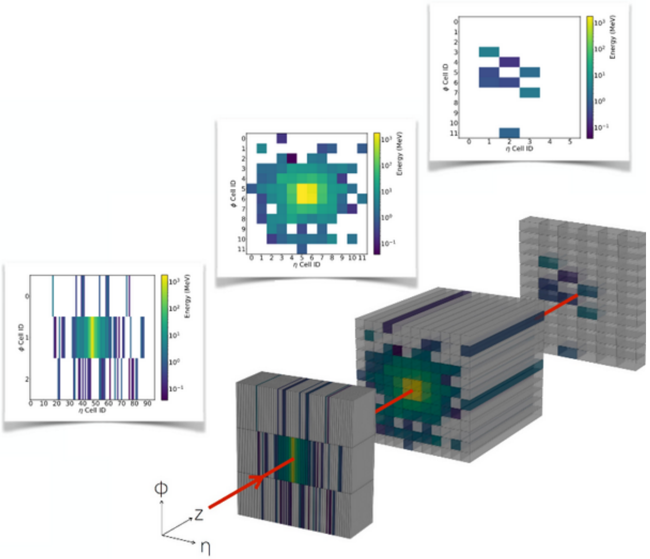
Game: Which face is real?

http://www.whichfaceisreal.com/index.php

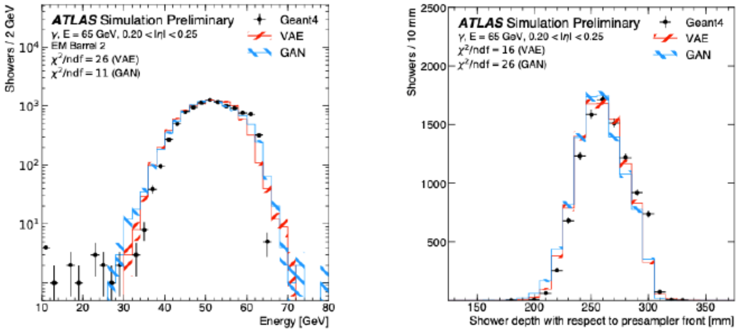
Physics Applications

- Simulation acceleration
- De-correlation
- Style transfer

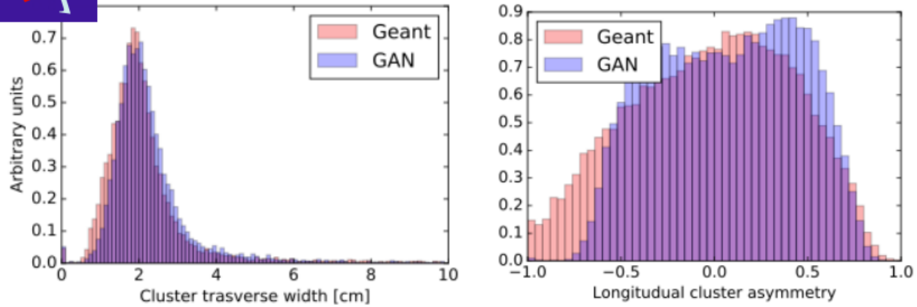


GAN applications for fast calorimeter simulation in other experiments

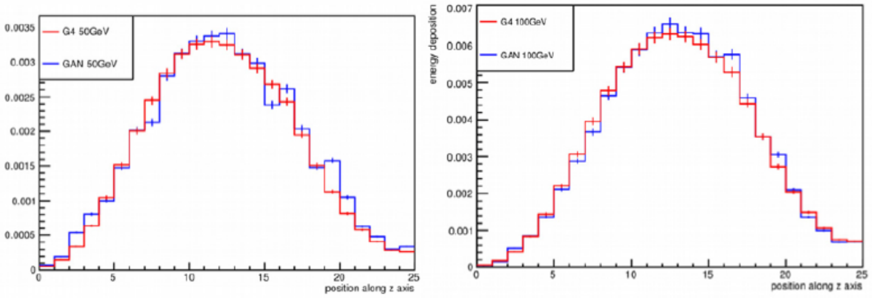
Deep generative models for fast shower simulation in ATLAS, http://cds/2680531



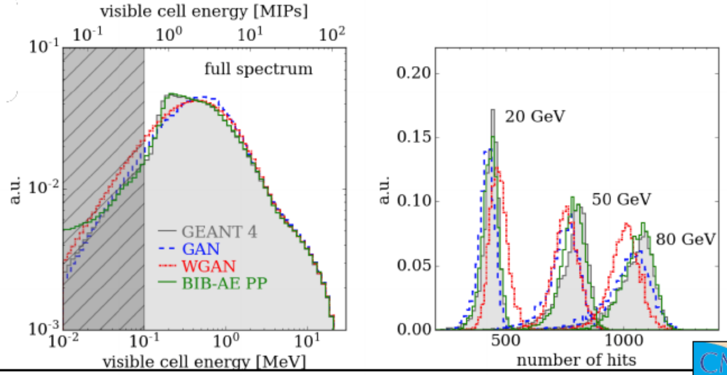
Generative Models for Fast Calorimeter Simulation - LHCb case, [1812.01319](1812.01319)



3D convolutional GAN for fast simulation, https://doi.org/10.1051/epjconf/201921402010



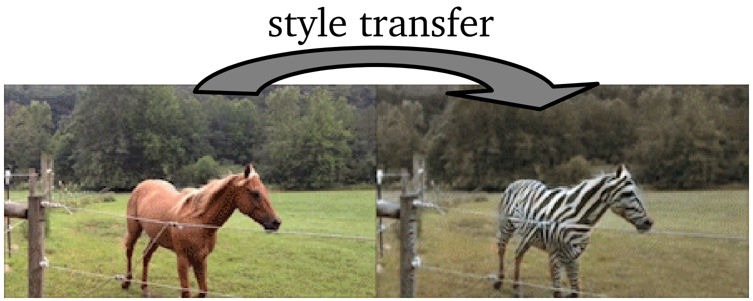
Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed, [2005.05334](2005.05334)



Generalization Capacities on Data

DNNs and Domain Adaption

- models are trained using physics simulations
- trained models are applied to data
  - ➔ can lead to reconstruction biases



https://bair.berkeley.edu/static/blog/humans-cyclegan/

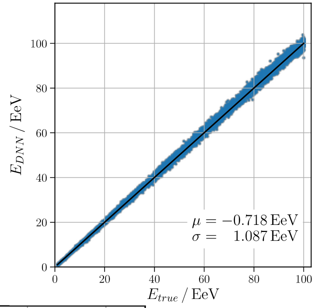
Simulation Refinement

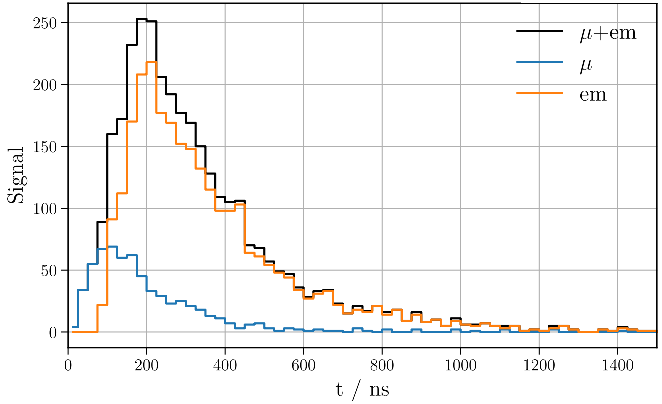
Erdmann et al.
Comput Softw Big Sci (2018) 2: 4

- Training on **simulations** but application on **data**
  - Model can be sensitive to artifacts / mismatches existing in simulation

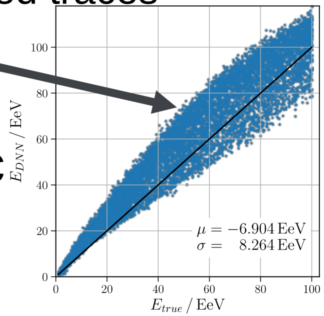
Neural network can not handle modified traces

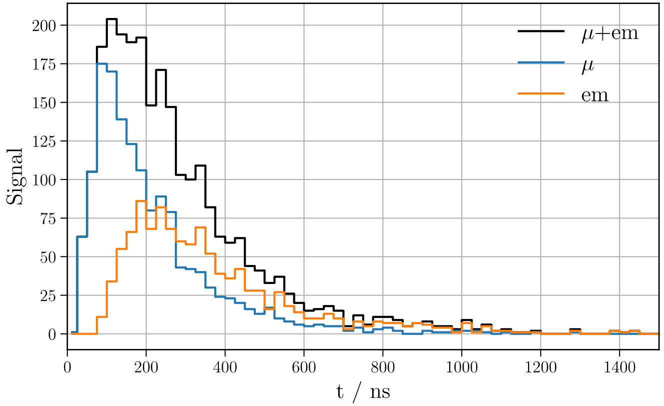
Simulation
70% electromagnetic
30% muonic





Data
30% electromagnetic
70% muonic
+ Increased noise

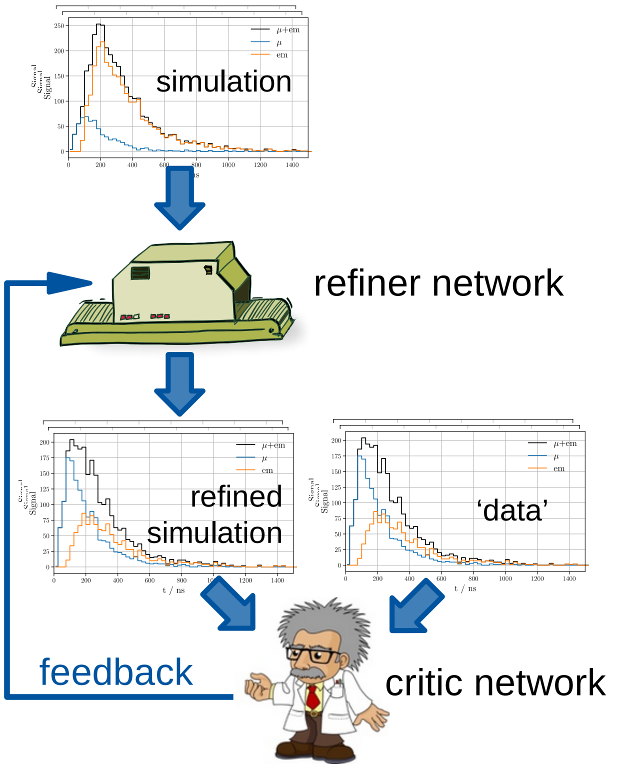




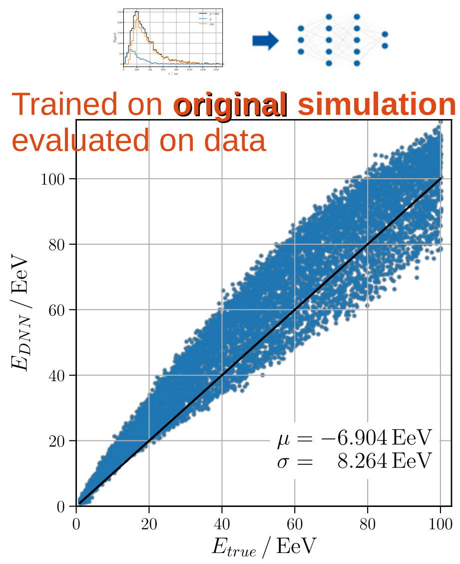
Simulation Refinement

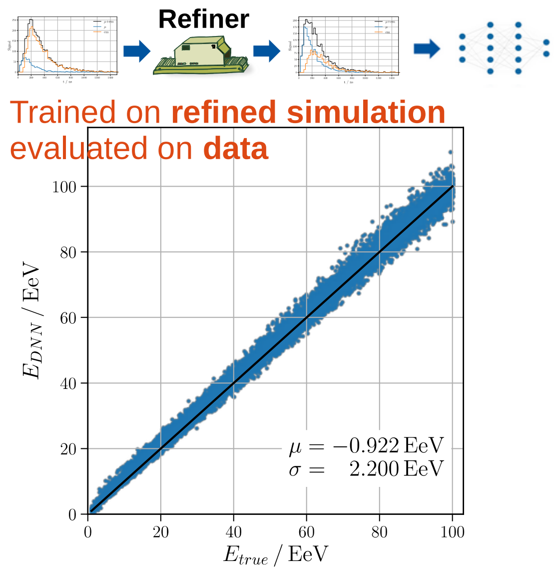
Erdmann et al.
Comput Softw Big Sci (2018) 2: 4

mitigate data / simulation mismatches → train *refiner* to refine simulated data



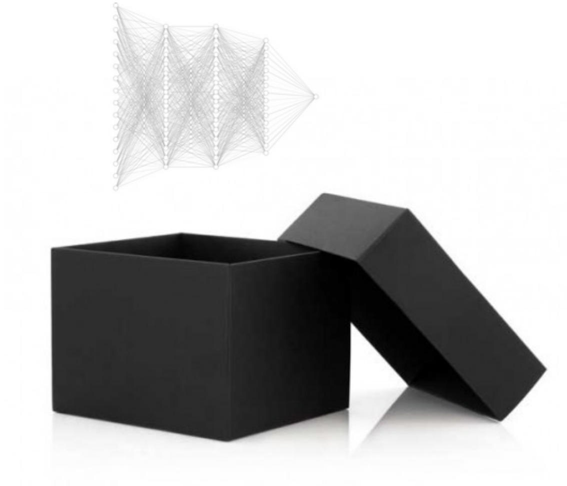
- feedback given by adversarial *critic* network, rating the refined simulation quality
  - refiner uses feedback to improve performance
- improved performance when training with refined simulation

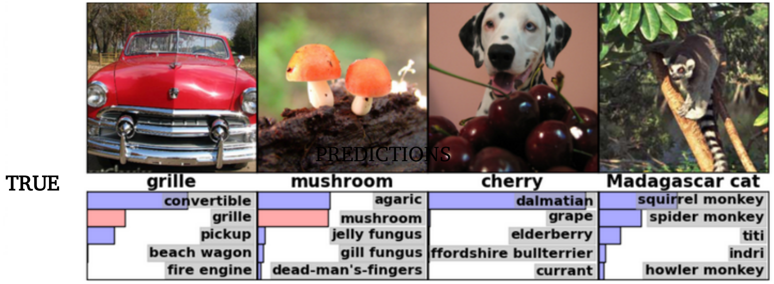




Interpretability and Deep Learning

I. – Feature Visualization
II. – Prediction Analysis





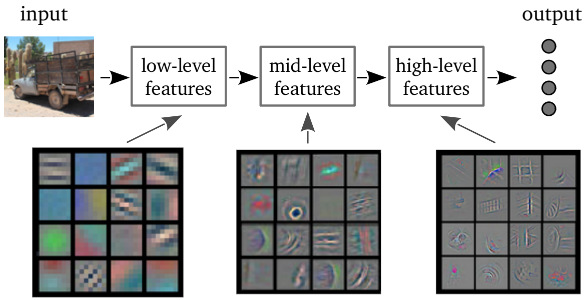
ArXiv/1602.04938

(a) Husky classified as wolf (b) Explanation

“Why is the model predicting a certain class / value?”

Predictions

Interpretability



Model

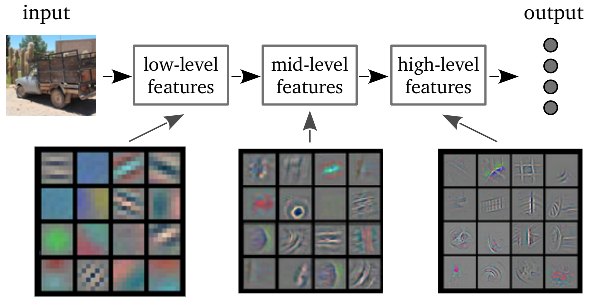
Data

“How is the model working / are features formed?”
“How do DNNs see the world?

“Which part of the data is most useful?”

Feature Visualization

Model Interpretability

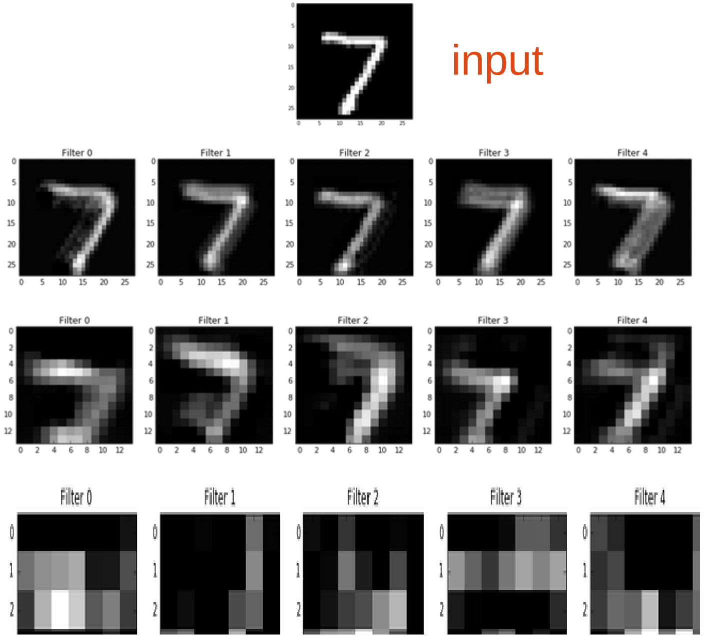


"How is the model working / are features formed?"
"How do DNNs see the world?

Visualization of an MNIST CNN

Visualization of activations

- Propagation of input through model
- Later activations hard to interpret



input

Layer 1

Layer 2

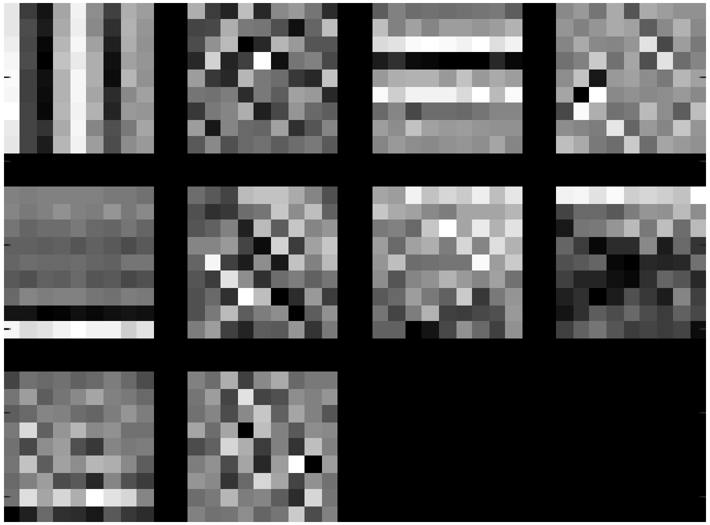
Layer 3

Arthur Juliani - Visualizing Neural Network Layer Activation (Tensorflow Tutorial), Medium

Visualization of first layer filters

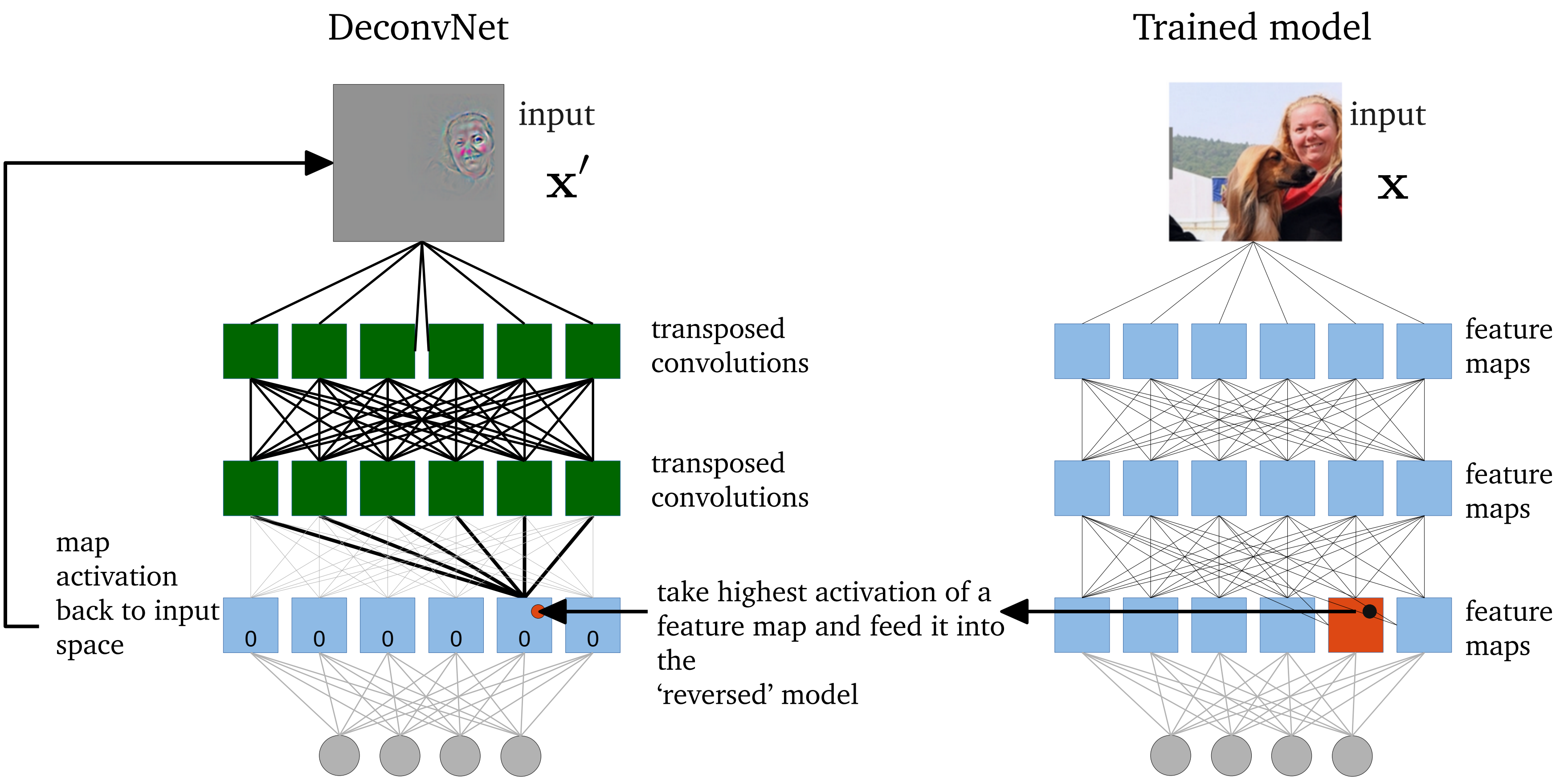
- Edge detection
- Focus on structures in the center

First layer filters learned with
mirroring, center, and dropout regularization



James Hays - http://cs.brown.edu/courses/cs143/2017_Spring/proj6a/

Deconvolutional Network (DeconvNet)



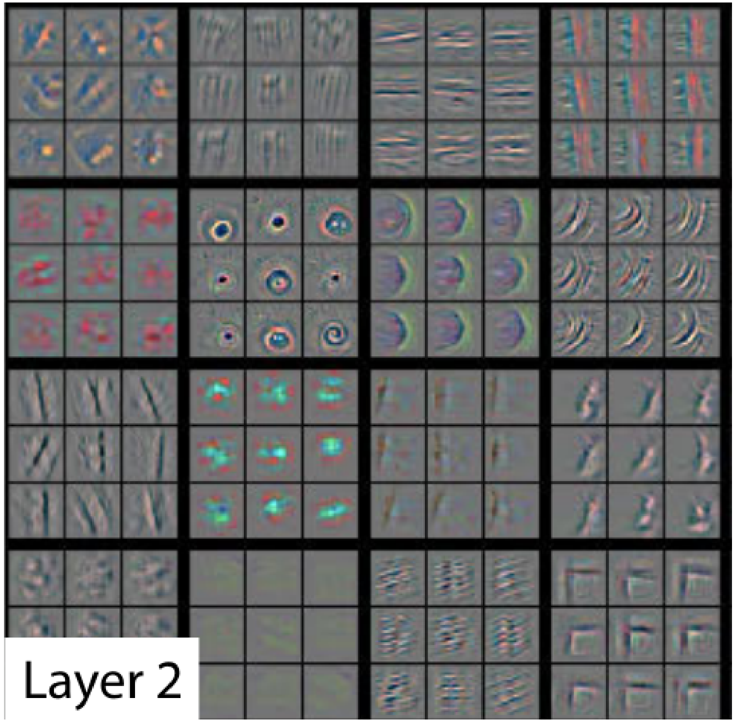
Visualization using DeconvNet

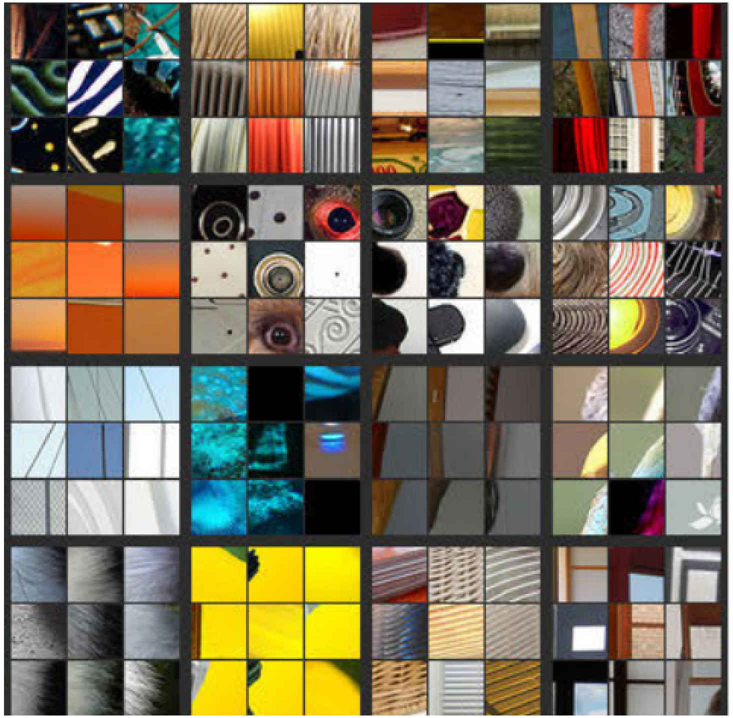
Visualized feature

Cut-outs of samples which create high activation in the specific feature map

Visualized feature

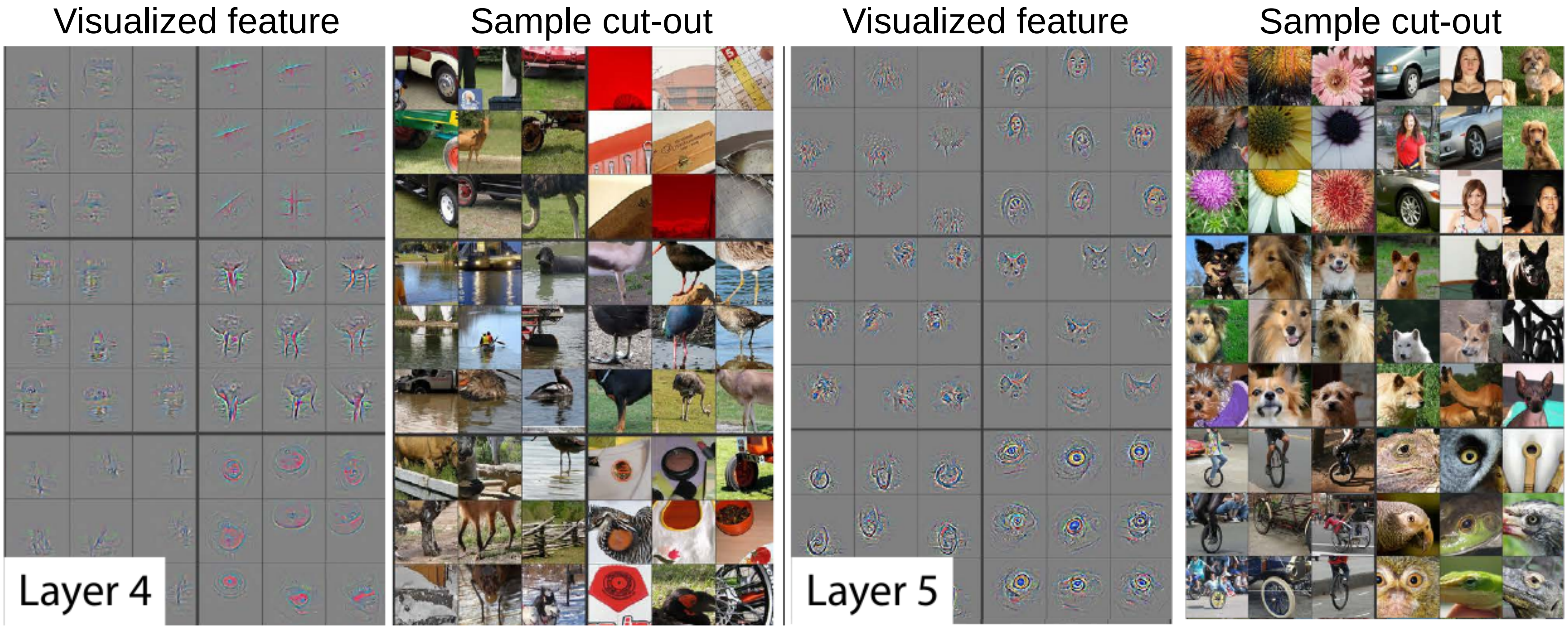
Zeiler, Fergus: Visualizing and Understanding Convolutional Networks





Cut-outs of samples which create high activation in the specific feature map

Visualization using DeconvNet



Zeiler, Fergus: Visualizing and Understanding Convolutional Networks

- Layer representation show feature hierarchy → features become more complex
  - Feature semantic becomes more specific (separation more class specific)

Visualization of Features:
https://distill.pub/2017/feature-visualization/
Model Collection with Visualization:
https://microscope.openai.com/models

Analysis of predictions & feature attribution

arXiv:1602.04938

(a) Husky classified as wolf (b) Explanation

“Why is my model predicting a certain class / value?”

“What influences the model’s reasoning most?”

Predictions

Saliency Maps

Idea:

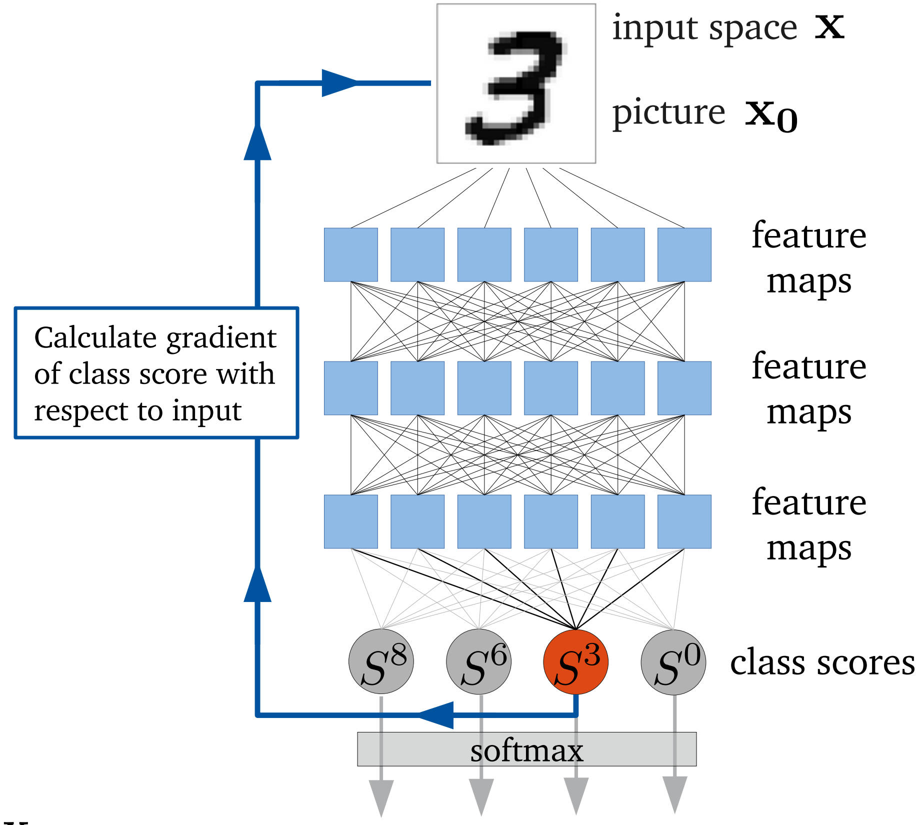
- *"What influences the class score at most?"*
- ➔ Important pixels have large gradients
  - Fix network parameters
  - Rank pixel importance of input space

- DNN $f(\mathbf{x})$ outputs score $S_c(\mathbf{x})$ for image $\mathbf{x}$
- Compute 1st order Taylor expansion

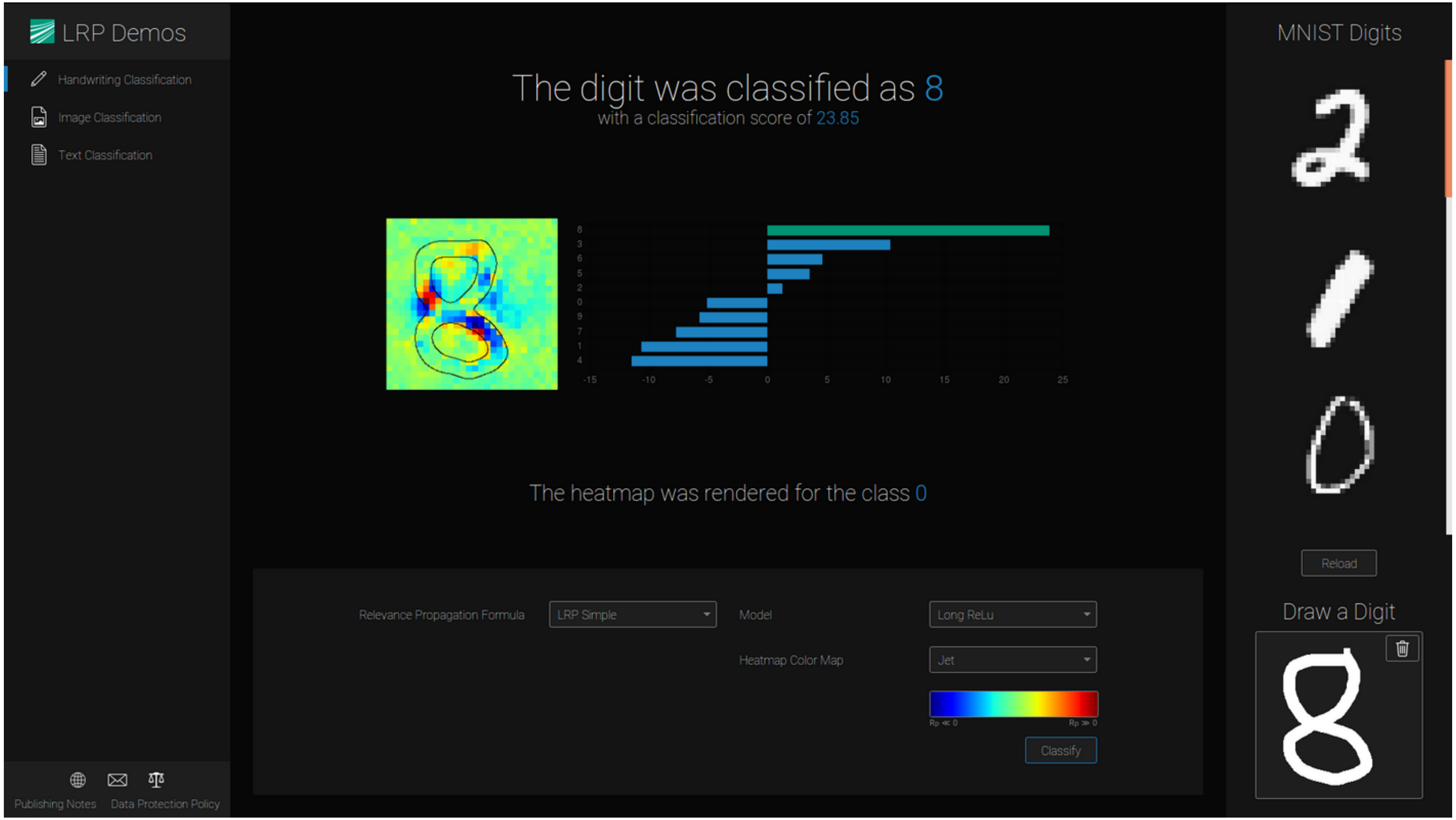
$$f(\mathbf{x}) = S_c(\mathbf{x}) \approx \mathbf{w}^T \mathbf{x} + \mathbf{b}$$

- Resulting map of gradients: $\mathbf{w} = \left.\frac{\partial S_c}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x_0}}$

Map has dimension of input image



DEMO - Handwriting



https://lrpserver.hhi.fraunhofer.de/handwriting-classification

Summary

- The field of machine learning is broad and developing fast
  - improved training strategies: shortcuts, normalization
  - advanced architectures: Graph networks, adversarial frameworks
  - interpretation of ML (DNNs are no black boxes but challenging to interpret)

- many different applications in physics research possible
  - able to outperform conventional methods with the increase in information
  - object reconstruction
  - generation of new samples
  - anomaly detection

- Still, they are **not the holy grail** but can be a powerful tool for your research!

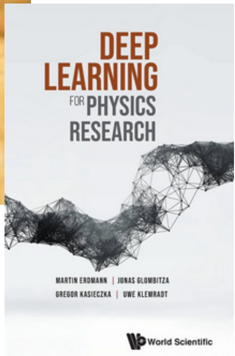
Tryout
Deep Learning
Yourself!

Find many physics examples at:

http://www.deeplearningphysics.org/

For example:

- CNNs, RNNs, GCNs
- GANs and WGANs
- Anomaly detection, Denosing AEs
- Visualization & introspection and more



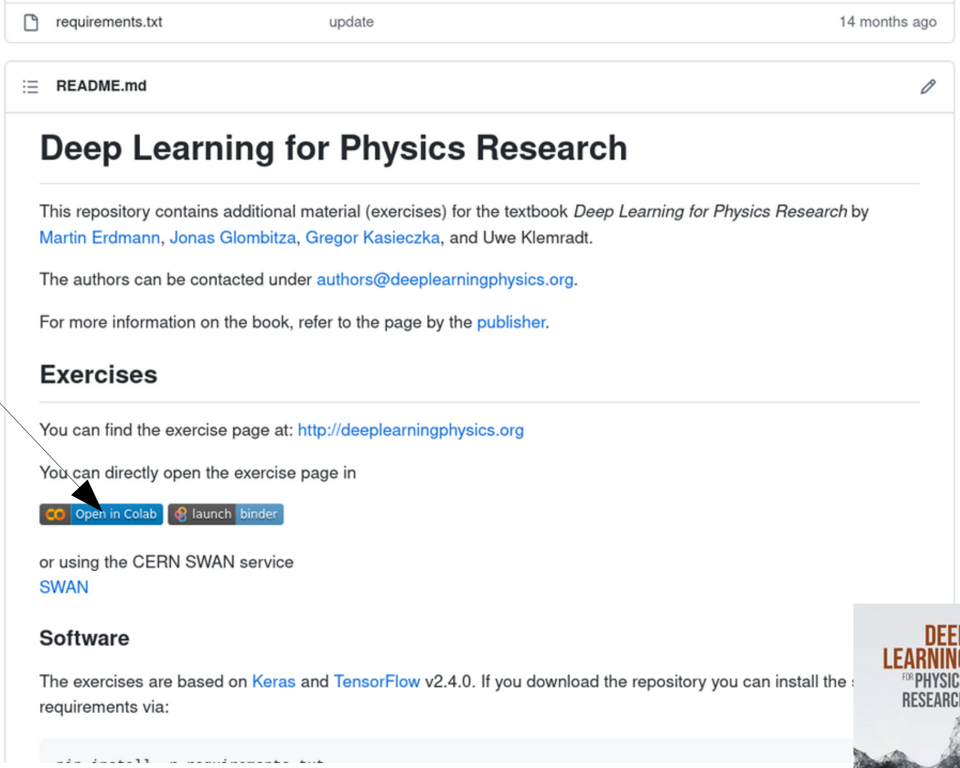
Collection of code examples → PHYSICS

https://github.com/DeepLearningForPhysicsResearchBook/deep-learning-physics

Run examples directly in GoogleColab!

- **Advanced CNNs**
  exercise 11.1
- **Graph networks**
  exercise 10.1 & exercise 16.1
- **Introspection**
  exercise 12.1, 12.2, 12.3
- **Generative Models**
  exercise 18.1, 18.2

Open in Colab


requirements.txt update 14 months ago

README.md

Deep Learning for Physics Research

This repository contains additional material (exercises) for the textbook *Deep Learning for Physics Research* by Martin Erdmann, Jonas Glombitza, Gregor Kasieczka, and Uwe Klemradt.

The authors can be contacted under authors@deeplearningphysics.org.

For more information on the book, refer to the page by the publisher.

Exercises

You can find the exercise page at: http://deeplearningphysics.org

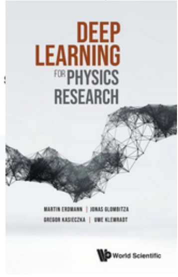
You can directly open the exercise page in

Open in Colab | launch binder

or using the CERN SWAN service
SWAN

Software

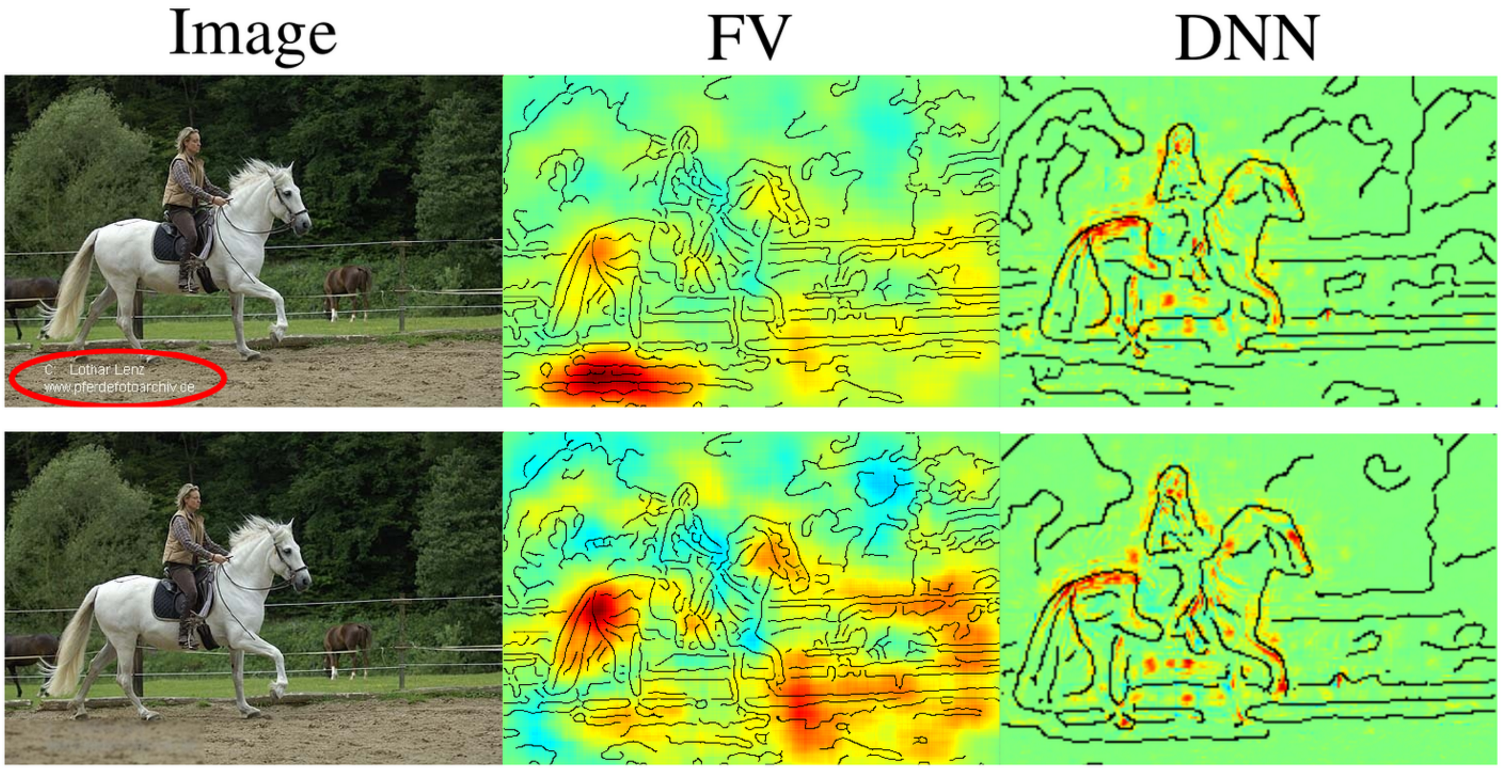
The exercises are based on Keras and TensorFlow v2.4.0. If you download the repository you can install the
requirements via:



Links & Resources

- TensorFlow Playground: https://playground.tensorflow.org
- Deep Learning (Goodfellow, Bengio, Courville), MIT Press, ISBN: 0262035618
  http://www.deeplearningbook.org/
- Neural Networks and Deep Learning (Nielson) - http://neuralnetworksanddeeplearning.com/
- CS231n - Convolutional Neural Networks for Visual Recognition (Kaparthy)
  http://cs231n.stanford.edu/syllabus.html
- Deep Learning by Google (Vanhoucke), Udacity https://www.udacity.com/course/deep-learning--ud730
- An Introduction to different Types of Convolutions in Deep Learning, Paul-Louis Pröve
  https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d
- Deep Learning with Python, Francois Chollet
- The CIFAR-10 dataset - https://www.cs.toronto.edu/~kriz/cifar.html
- Deep Learning-based Reconstruction of Cosmic Ray-induced Air Showers - Erdmann, Glombitza, Walz
  https://doi.org/10.1016/j.astropartphys.2017.10.006

Semantic Misinterpretation



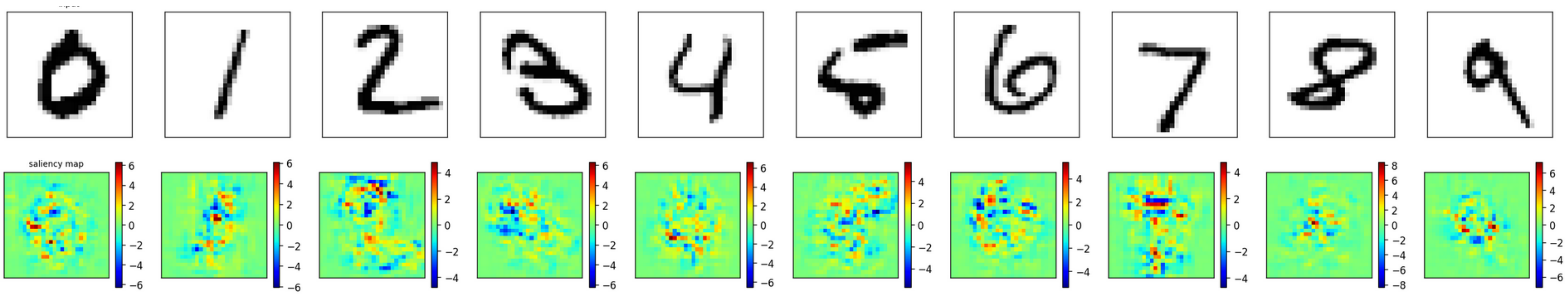
Bach et. Al. - Analyzing Classifiers: Fisher Vectors and Deep Neural Networks, arXiv:1512.00172

arXiv:1602.04938

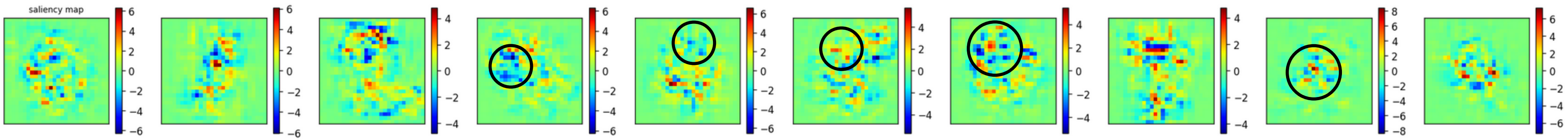
(a) Husky classified as wolf (b) Explanation

How important is the context?

Saliency Maps MNIST



- Negative gradient: intensity increase of respective pixel → reduce class score
- Positive gradient: intensity increase of respective pixel → raise class score



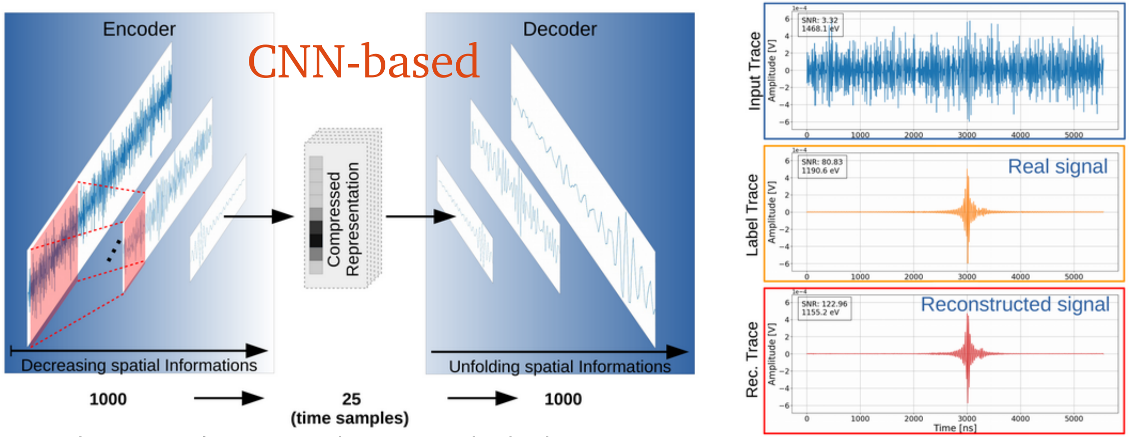
Denoising of Signal Traces (1D)

Supervised training of denoising autoencoders

- feature compressed space in between encoder and decoder
- ➔ encodes only relevant information in compressed space

Future application: brining ML close to the sensor

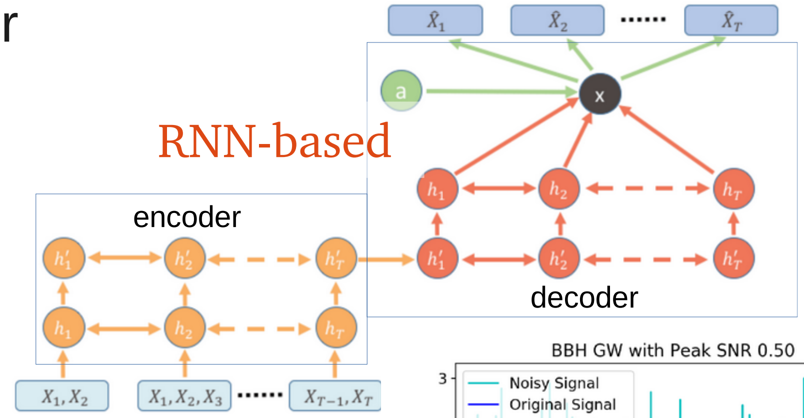
Denoising of cosmic ray radio signals

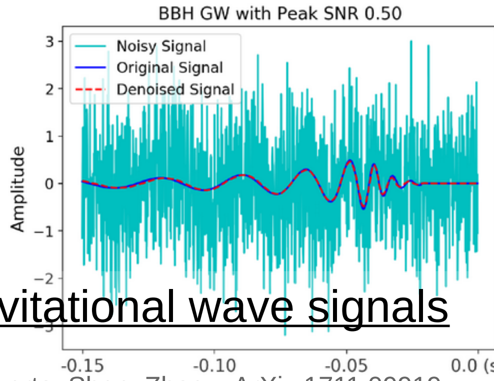


M. Erdmann et al. - 10.1088/1748-0221/14/04/P04005

A. Rehman et al., PoS ICRC2021 417

P. Bezyazeekov et al., ArXiv/2101.02943 & D. Shipilov et al., EPJ (2019) 02003



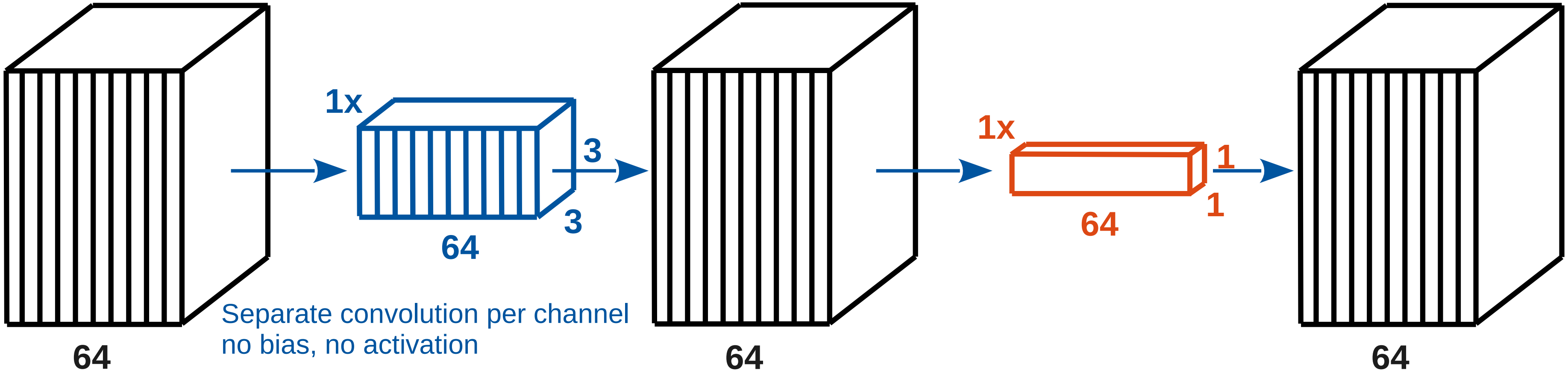


Denoising of gravitational wave signals

George, Huerta, Shen, Zhao – ArXiv 1711.09919

Xception ("Extreme Inception")

- **Idea:** If spatial correlations and cross-channel correlations are sufficiently decoupled it's better to compute them separately
- **Depthwise separable convolutions**
  - Perform depthwise separate convolution on each channel
  - Perform pointwise convolution (1 x 1) across channels



$$64(3\cdot 3\cdot 1)+64(1\cdot 1\cdot 64+1)\approx 4,700$$

Standard convolution: $64(3\cdot 3\cdot 64+1)\approx 37,000$

Tutorial

- Open tutorial page
  https://github.com/jglombitza/Introspection_tutorial
- open Colab link and login with your Google Account

- **Exercise 1:** model introspection
  - model visualization using activation maximization

  Open in Colab

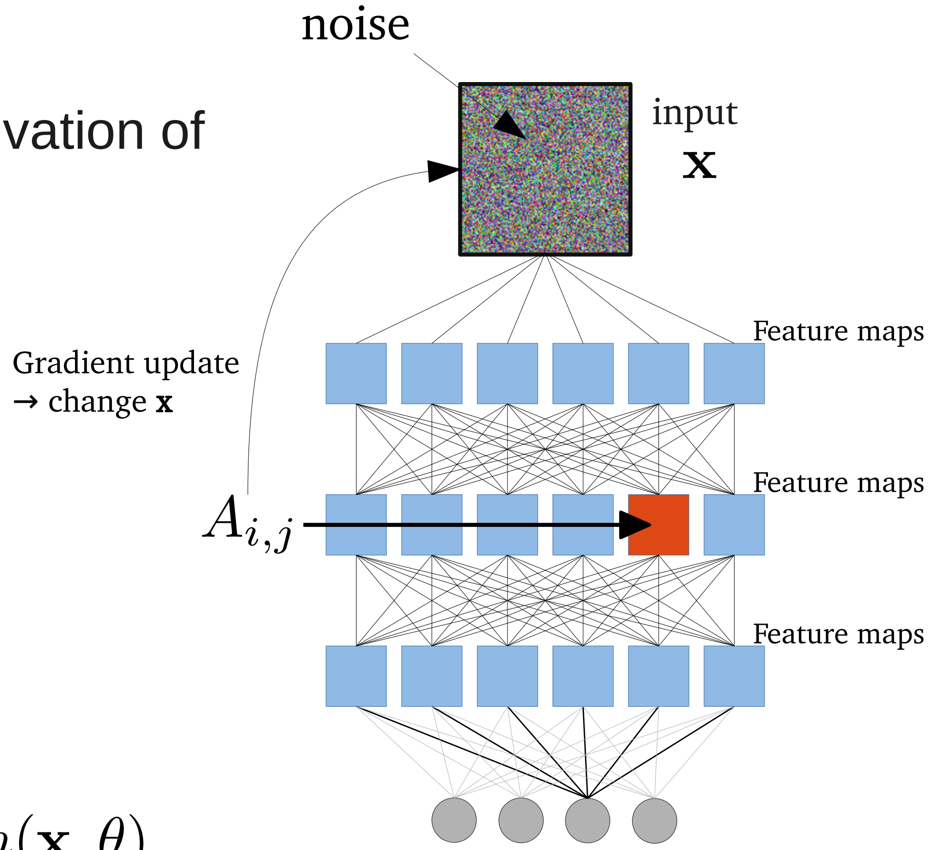
- **Exercise 2:** introspection of predictions
  - implement discriminative localization

  Open in Colab

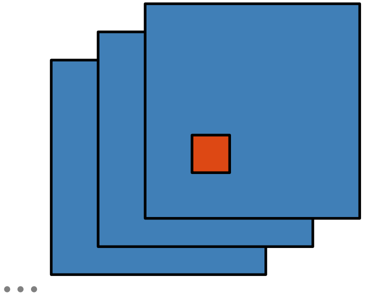
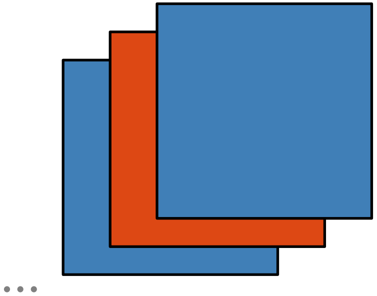
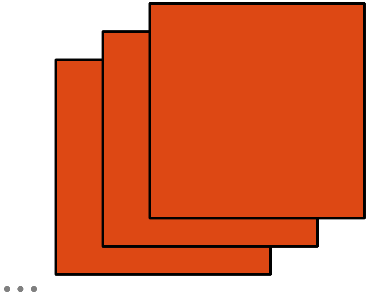
Activation maximization

Idea:

- Construct pattern which maximizes the activation of a specific feature map
- Model f_θ pre-trained, weights θ fixed
- Find $\tilde{\mathbf{x}} = \underset{\mathbf{x}}{\mathrm{argmax}}\; h(\mathbf{x}, \theta)$
- $h(\mathbf{x}, \theta) = \sum_{i,j} A_{i,j}(\mathbf{x}, \theta) + b$
- Start from noise
  - ➔ perform gradient **ascent** $\mathbf{x}' \to \mathbf{x} + \alpha \frac{dh(\mathbf{x}, \theta)}{d\mathbf{x}}$



Activation Maximization

	neuron	channel	layer (deep dream)
objective			
obtained visualizations