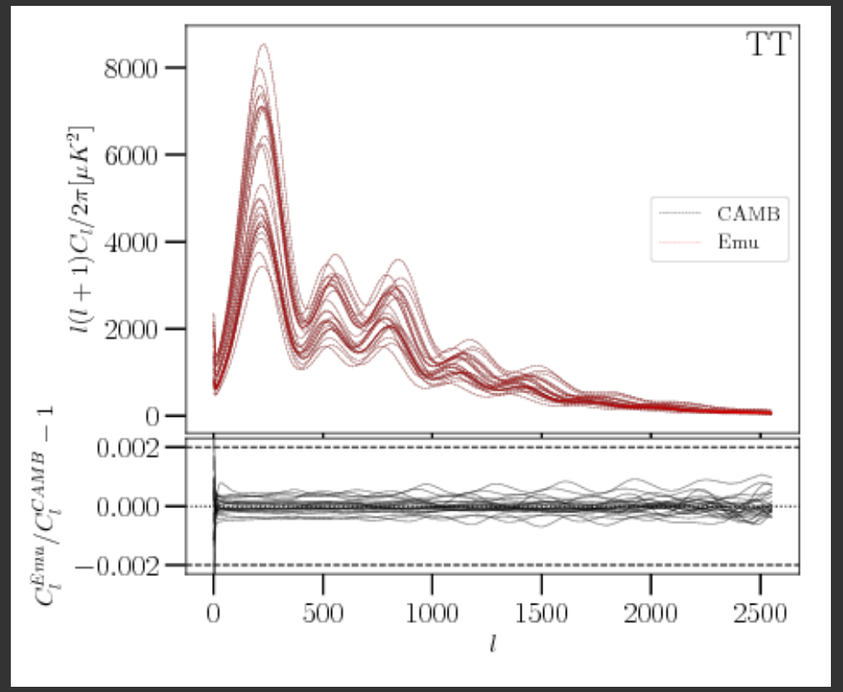
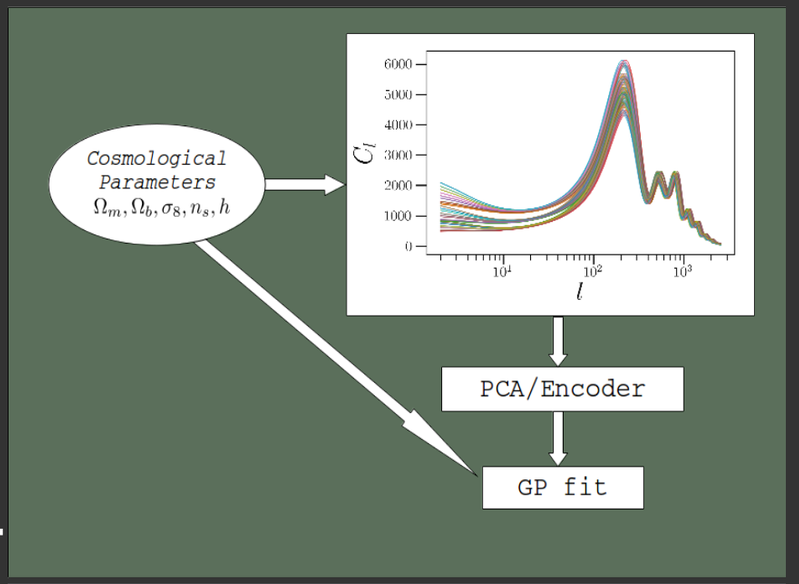
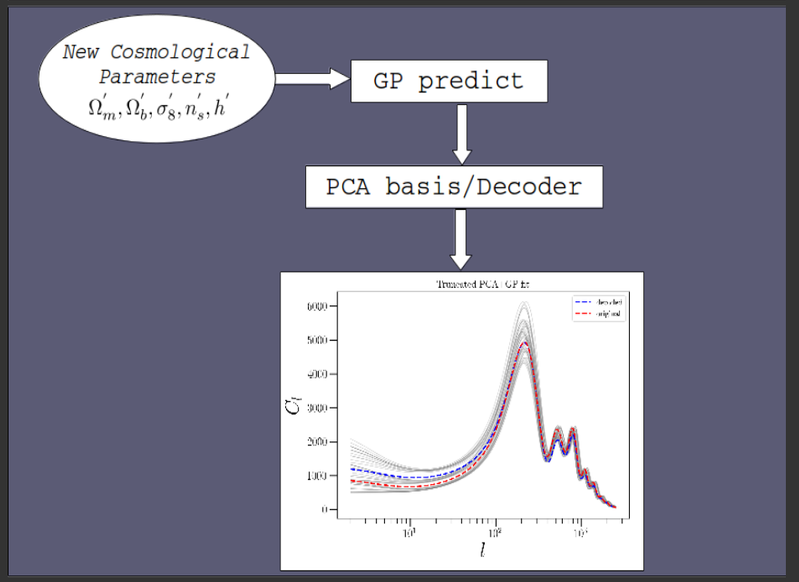
Suite of Emulators for Cosmological Inferences

Nesar Ramachandra on behalf of
Cosmological Physics and Advanced Computing Group
Argonne National Laboratory

19 March 2019





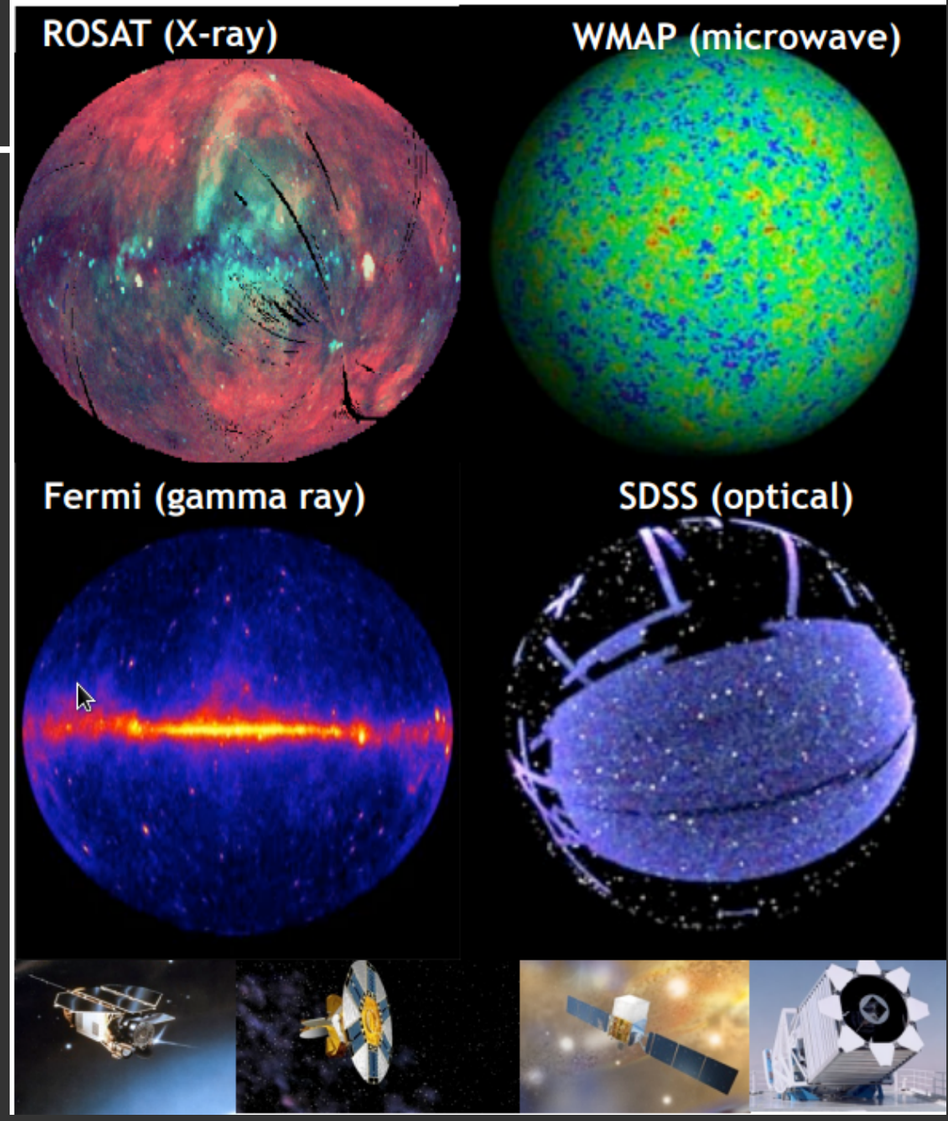


Argonne NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY | Office of Science

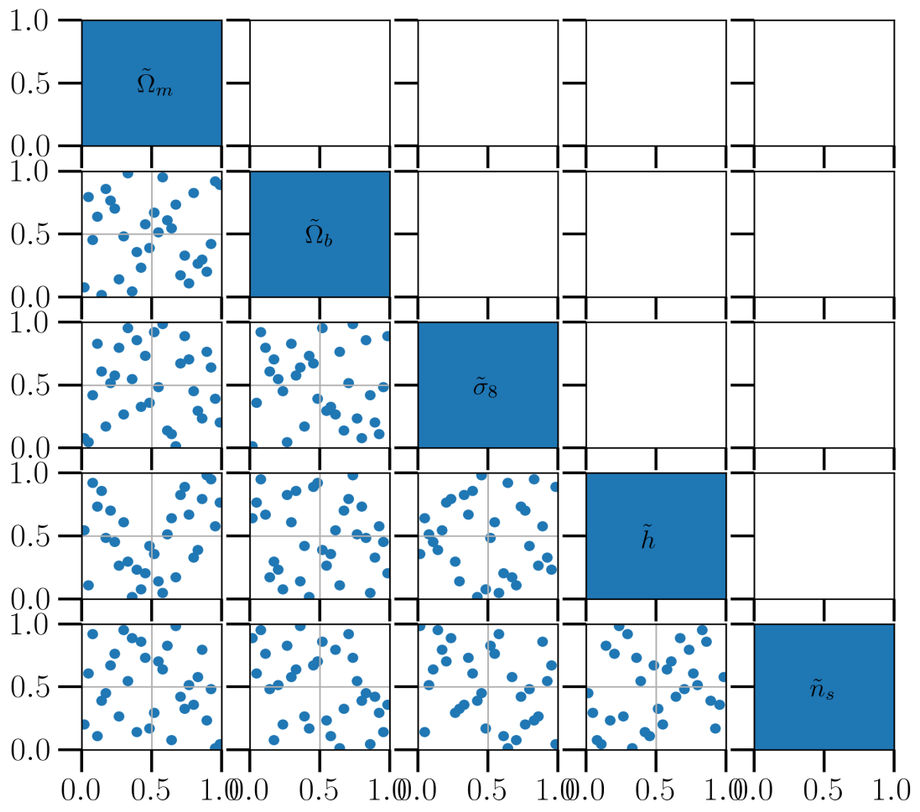
Motivations

- Modern precision Cosmology constitutes a statistical inverse problem: Sky maps → Cosmological parameters
- Goal: Infer Cosmological parameters from observations, but since we have only one Universe, we rely on computer simulations.
- But Cosmological simulators can be expensive.
  - For example: 100 simulations using all of *Summit* (the fastest supercomputer in the world) would take 24 hours per simulation.
  - 100,000 simulations for parameter calibration for MCMC or ABC ~ 3 years.
- Alternatives: Linear theory isn't enough. Non-linear regime is very difficult, Fitting functions are not accurate enough (up to 10 percent for matter power spectra)
- Possible solution: Create Emulators for various cosmological functions to explore different cosmological models efficiently



$$\{\Omega_m, \Omega_b, \sigma_8, h, n_s, \Omega_0, \Omega_a.\Omega_\nu, ...\}$$

Building an Emulator: Step 1 – Experimental design



Experimental design: Latin-hypercube sampling design

- Sampling strategy depends heavily on the interpolation scheme. Since prediction from GP emulator is a weighted average of training points, a good space-filling sampling strategy is required.
- Nested Maxmin Latin hypercube design ensures convergence of emulator accuracy, in addition to space-filling requirements.

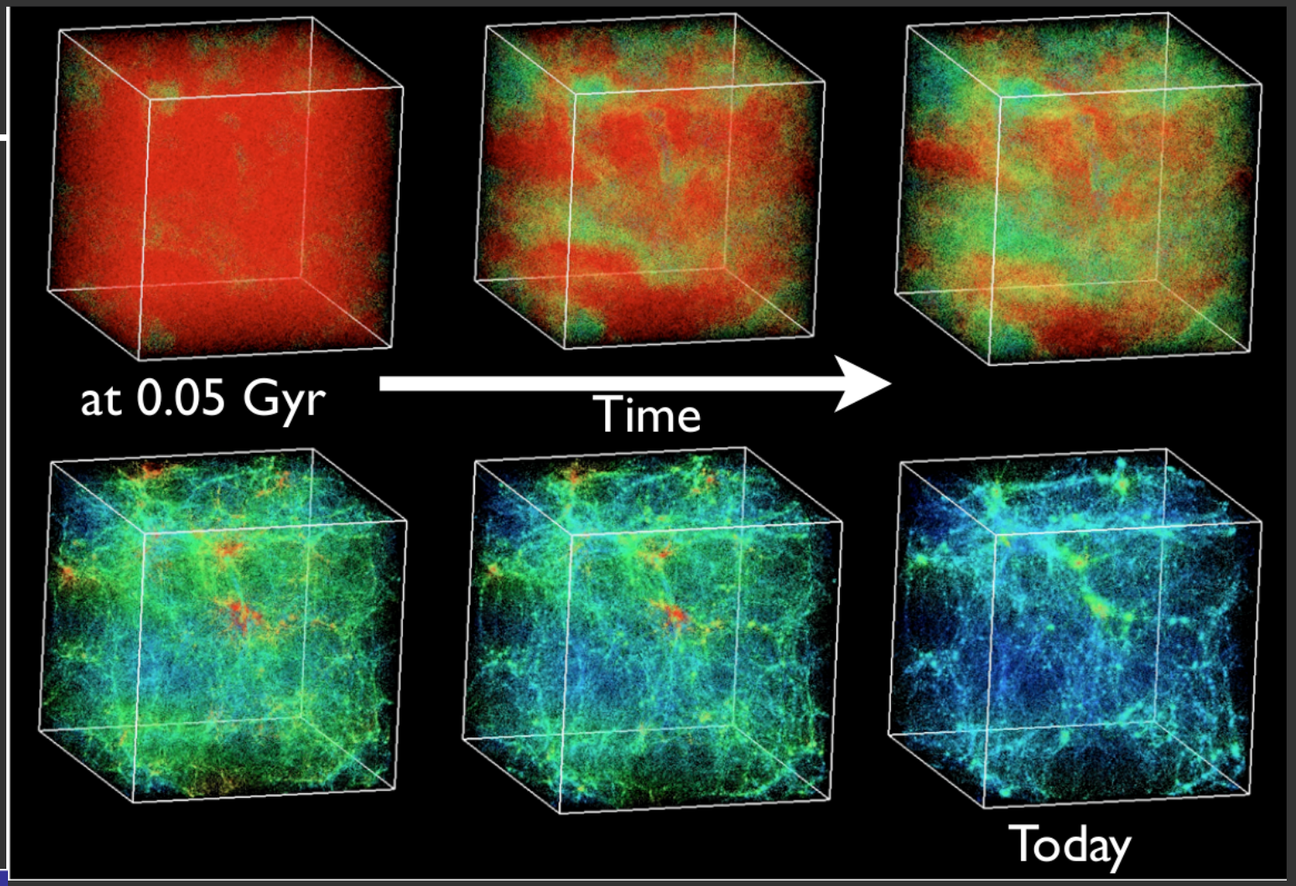
Mira-Titan Universe simulation suite and the corresponding emulators use 8 parameters

$$
\begin{aligned}
0.12 &\leq \omega_m \leq 0.155, \\
0.0215 &\leq \omega_b \leq 0.0235, \\
0.7 &\leq \sigma_8 \leq 0.9, \\
0.55 &\leq h \leq 0.85, \\
0.85 &\leq n_s \leq 1.05, \\
-1.3 &\leq w_0 \leq -0.7, \\
-1.73 &\leq w_a \leq 1.28, \\
0.0 &\leq \omega_\nu \leq 0.01.
\end{aligned}
$$

Step 2a –Training Simulations

Training data may be really expensive. For example, the Mira-Titan Universe:

- Cosmology simulations using HACC (Hardware/Hybrid Cosmology Code) for wCDM cosmology.
- Overall simulation suite of 100+ simulations is more than 1.2Pbyte
- Executed on *Mira* (at Argonne) and *Titan* (at Oak Ridge) supercomputers:

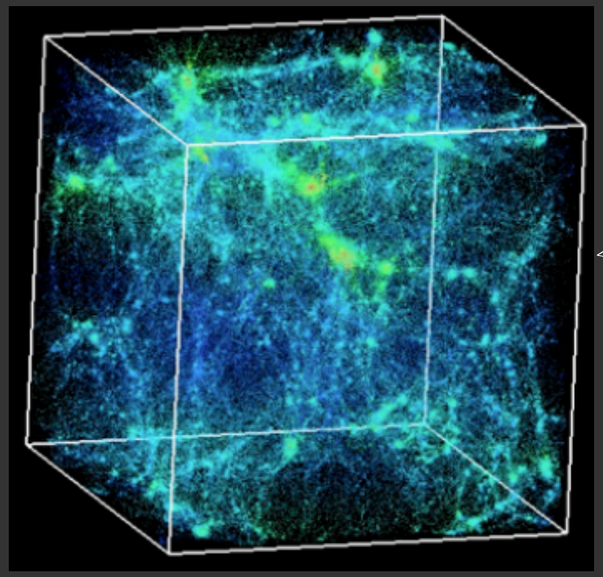


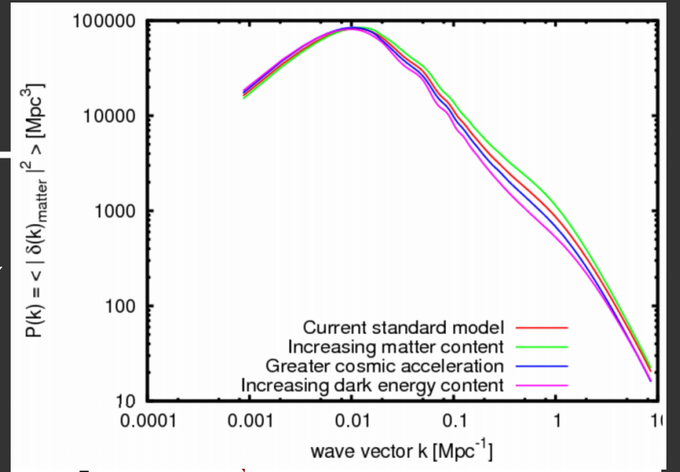
First ExaFLOP supercomputer (will be operational in 2021; was announced yesterday)

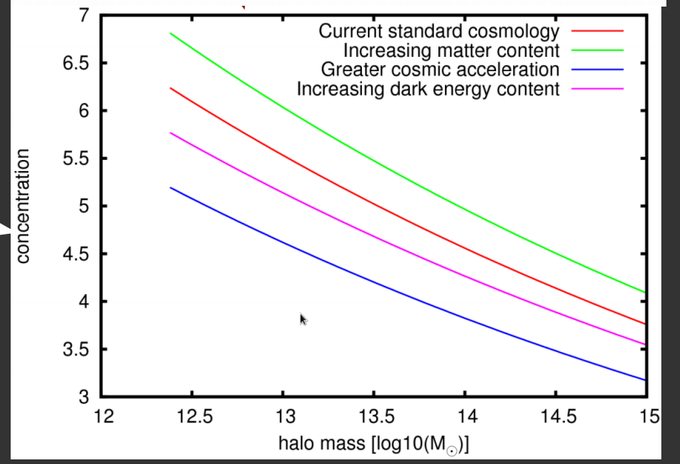
https://www.anl.gov/article/us-department-of-energy-and-intel-to-deliver-first-exascale-supercomputer

Step 2b –Training Simulations

- Cosmological Simulation → Extract quantity of interest

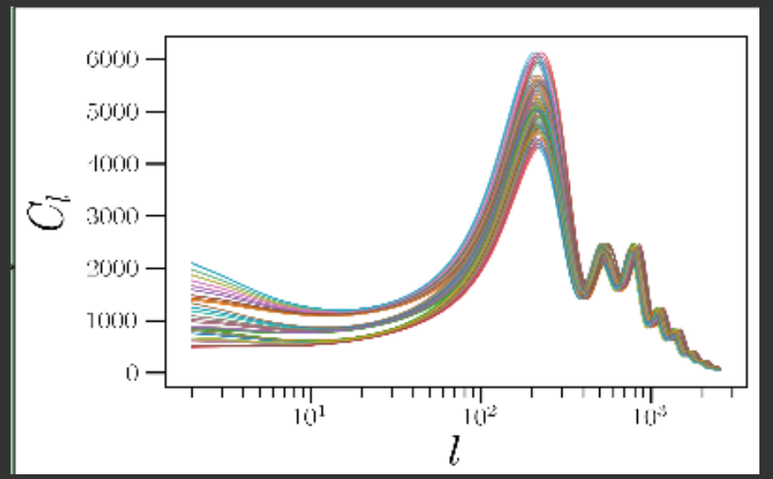






- Functions like the CMB power spectra can be much easier to compute using theoretical models.

CAMB



Step 3 – Interpolation in parameter space

- Gaussian Processes (GPs) provide an effective non-parametric method for interpolation.

- GPs can be applied directly to data or to weights of basis functions used to represent the data (e.g., to a Principal Components basis)

  - Used in Cosmic Emu (Heitmann et al 2006 and several other papers) with PCA

Cosmological function:

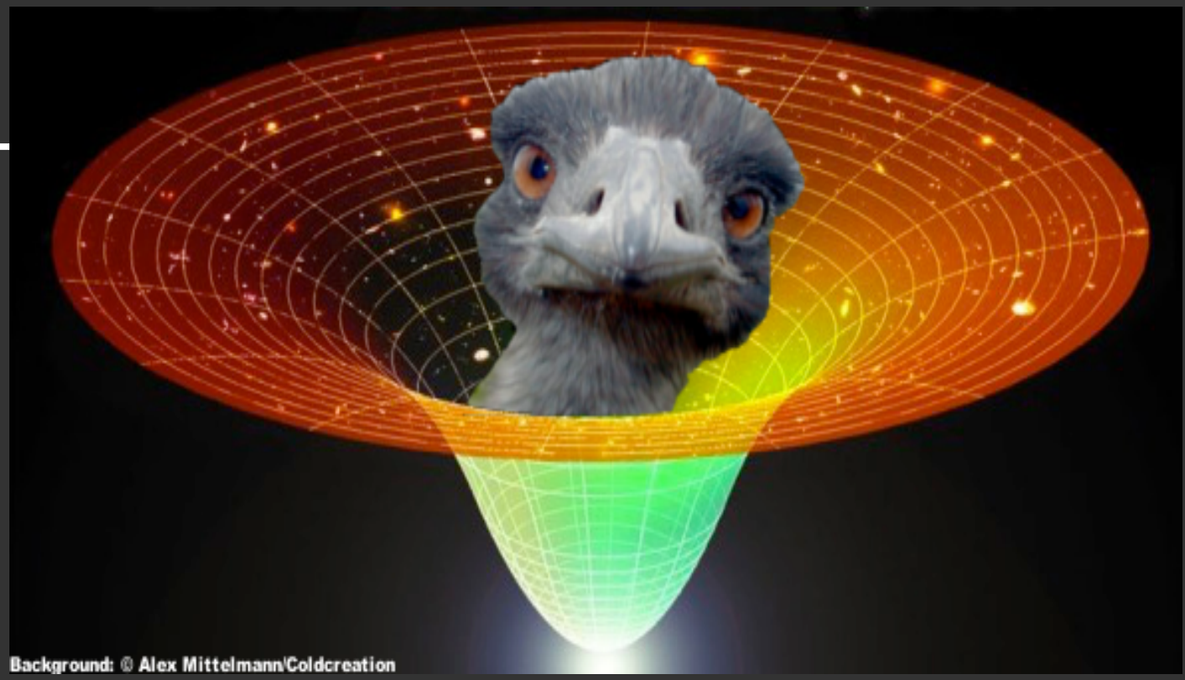
$$\eta(k;\theta) = \sum_{i=1}^{p_n} \phi_i(k) w_i(\theta) + \epsilon$$

PCA basis — GP weights — Error

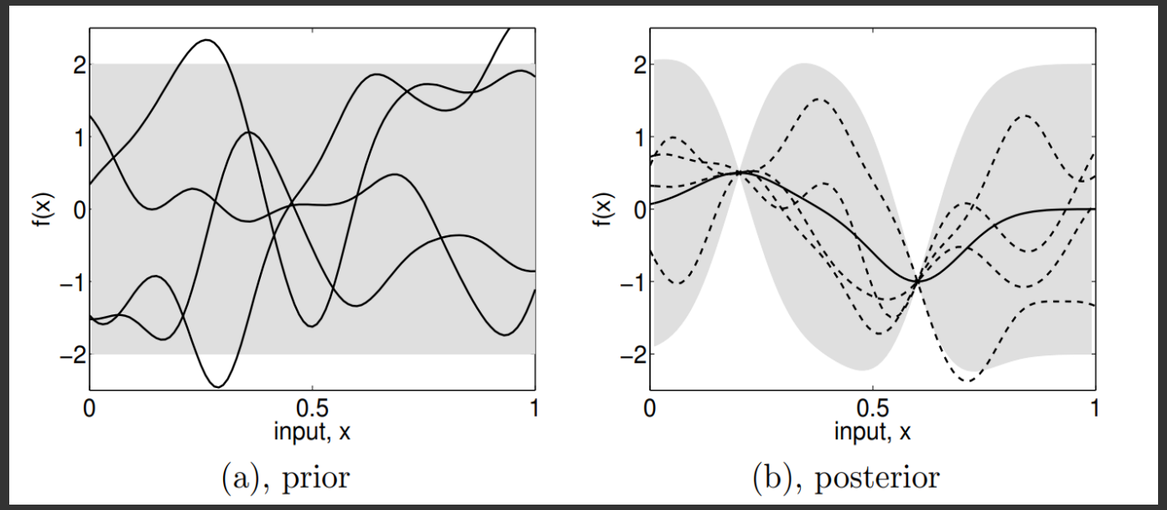
Cosmological parameters:

$$\theta \in [0,1]^{p_\theta}$$

$$w_i(\theta) = GP(\mu(\theta), K(\theta,\theta'))$$



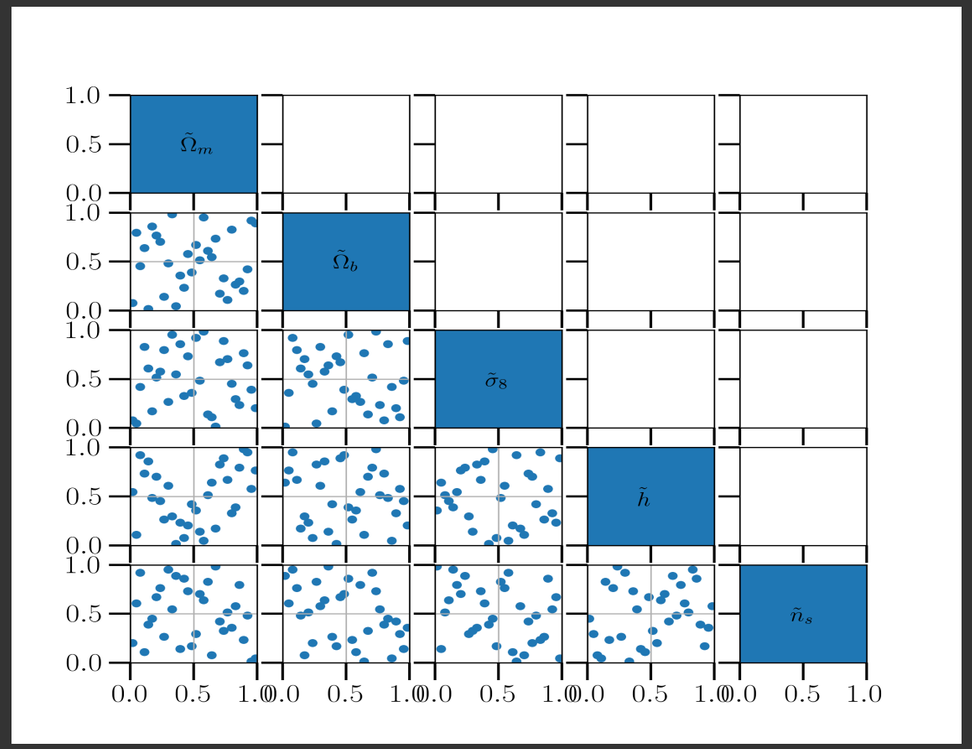
http://www.hep.anl.gov/cosmology/CosmicEmu



Rasmussen & Williams 2006

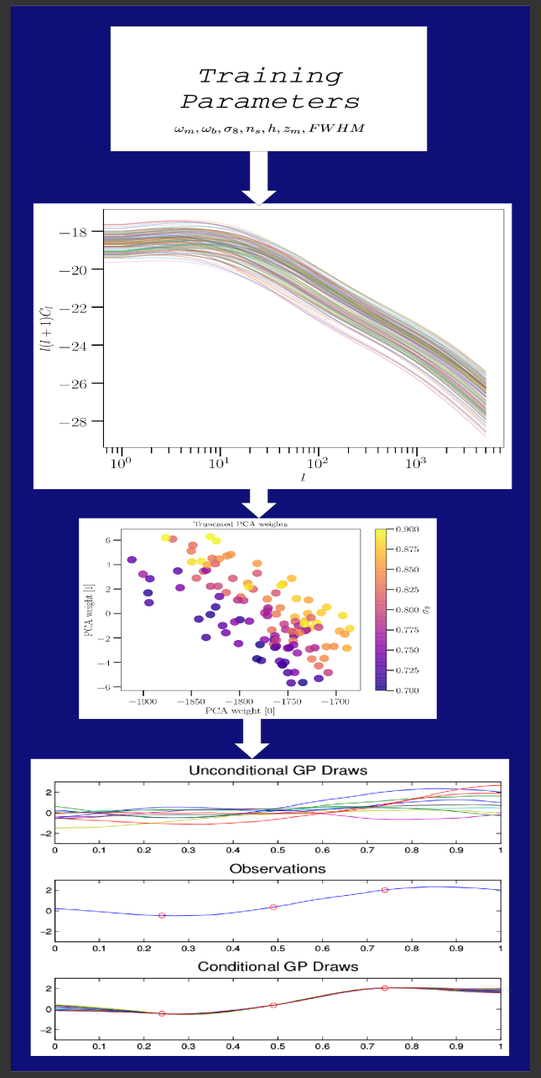
Summary: The Emulation pipeline

$$ln\left\{\frac{\Delta^2(k,z;\theta)}{2\pi k^{3/2}}\right\} = \sum_{i=1}^{p_n} \phi_i(k) w_i(\theta) + \epsilon$$

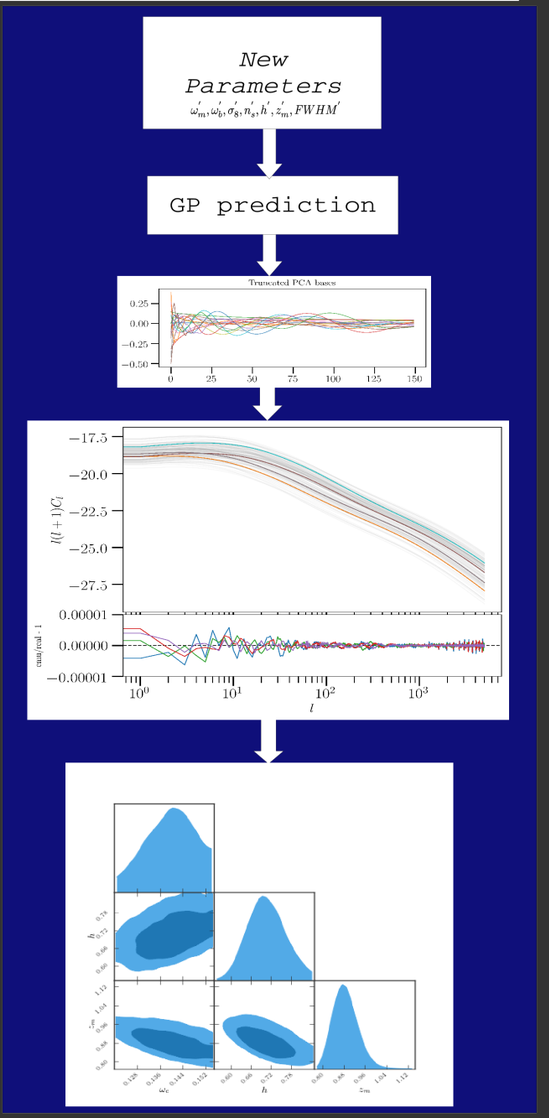


Experimental design: Training data generated from CAMB on latin-hypercube sampling design

→



Training: Cl → SVD → GP interpolation for Weights



Testing: GP prediction for new Weights → multiple by basis → new Cls

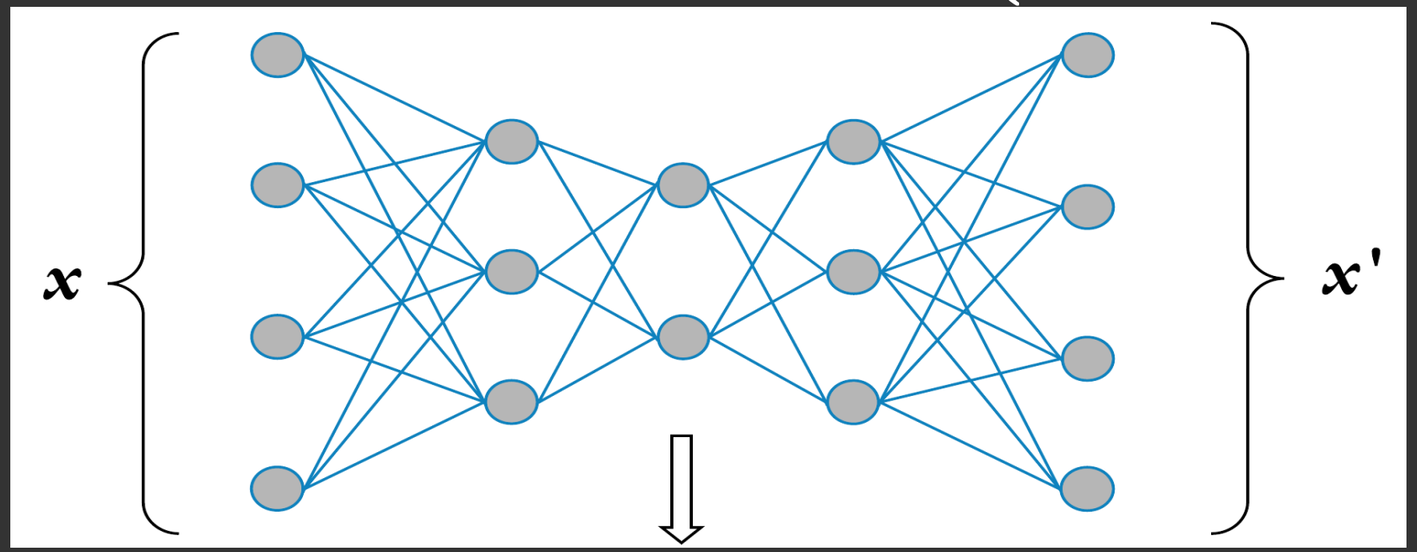
Alternatives to PCA: Basic Autoencoder (AE)

Encoder

$$z = \phi\left\{\sum_{i=1}^{n}(W_i x + b_i)\right\}$$

Decoder

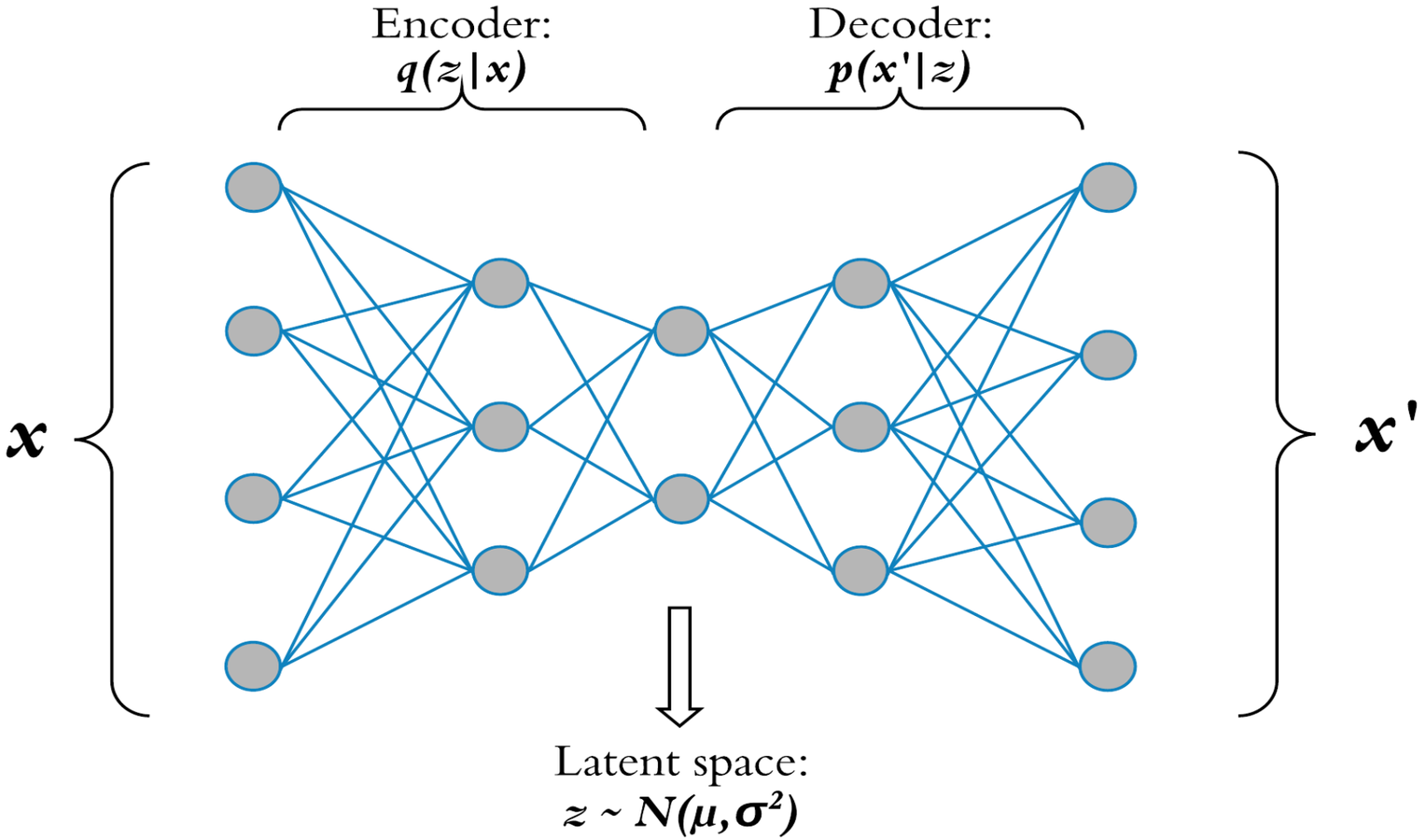
$$x' = \phi'\left\{\sum_{i=1}^{N'}(W_i' z + b_i')\right\}$$



- AE is an unsupervised learning algorithm
- 2 parts – Encoder and Decoder. Both are trained simultaneously
- The input and output of the AE are the same. The training is done with the AE successfully replicates the input.
- AE with just one fully connected layer (with linear activations) for encoder and decoder networks reduces to PCA

Reconstruction Loss: $\mathcal{L}_{recon}(x, x') = ||x - x'||$

Variational Autoencoder (VAE)

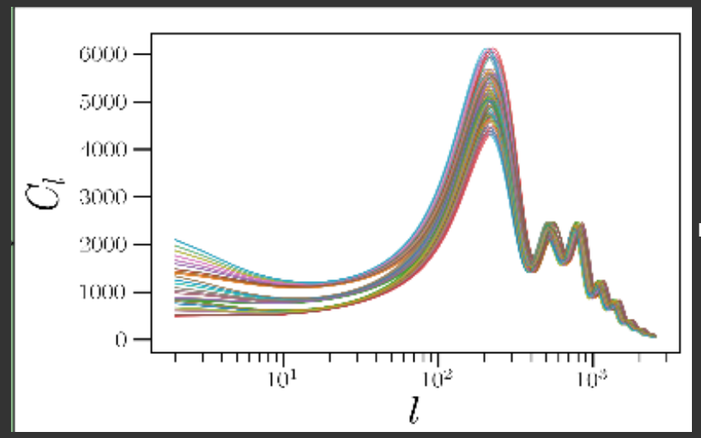


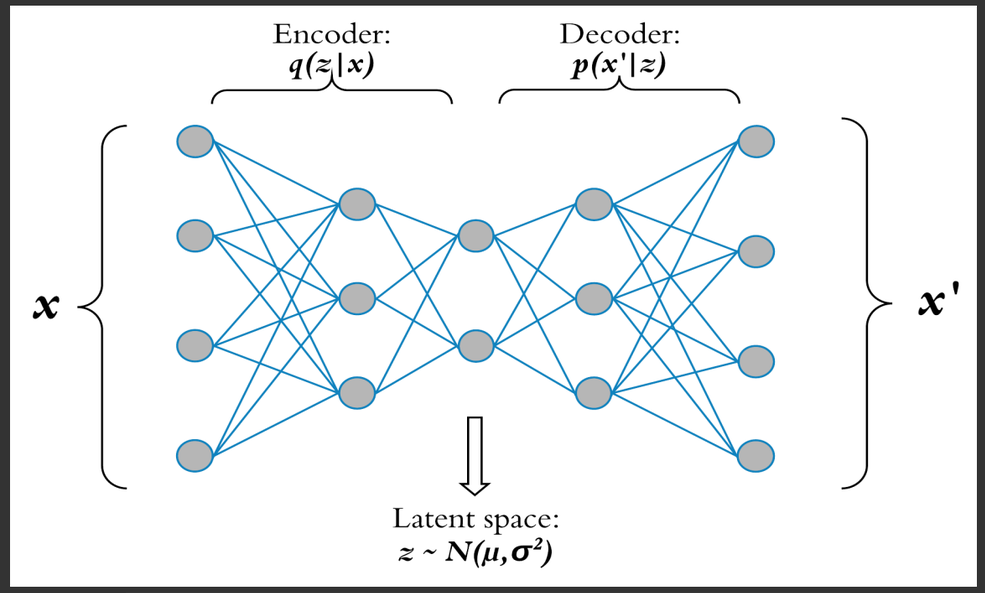
- Variational AEs are used for non-linear dimensionality reduction.
- Can be used a generative models since they facilitate easy sampling in latent space
- Less noisy than regular AEs

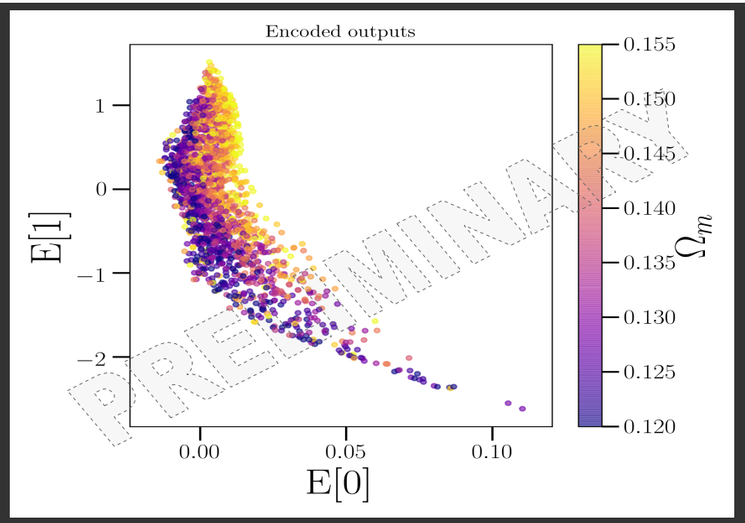
Loss: $\mathcal{L}_{total} = \mathcal{L}_{recon} - \mathcal{D}_{KL}(q_\theta)(z|x)||p_\theta(x|z))$

KL divergence loss: $\mathcal{D}_{KL}(q_\theta)(z|x)||p_\theta(x|z))$

Variational Autoencoders – information bottleneck





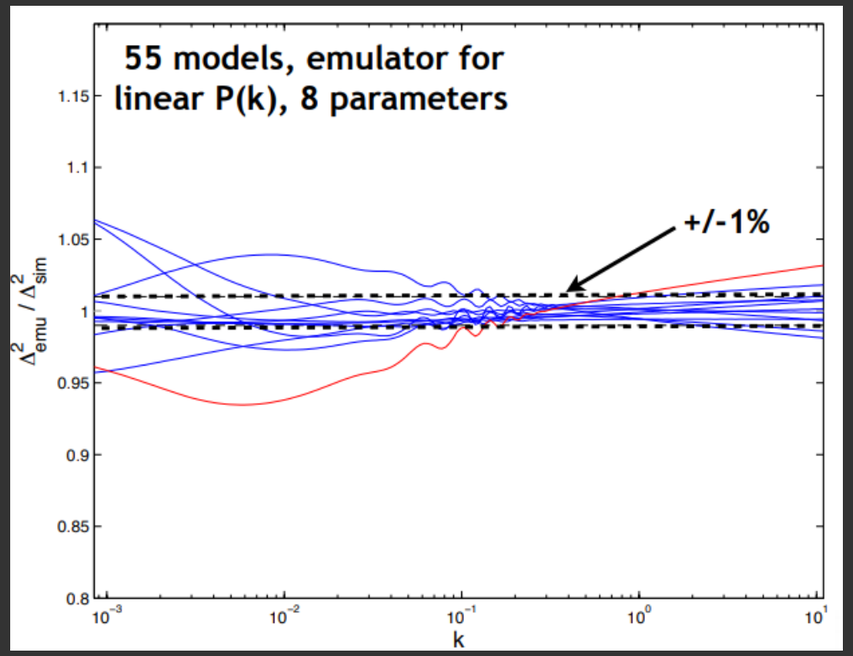


Distribution in latent space. Colorbar refers to one of the Cosmological parameters

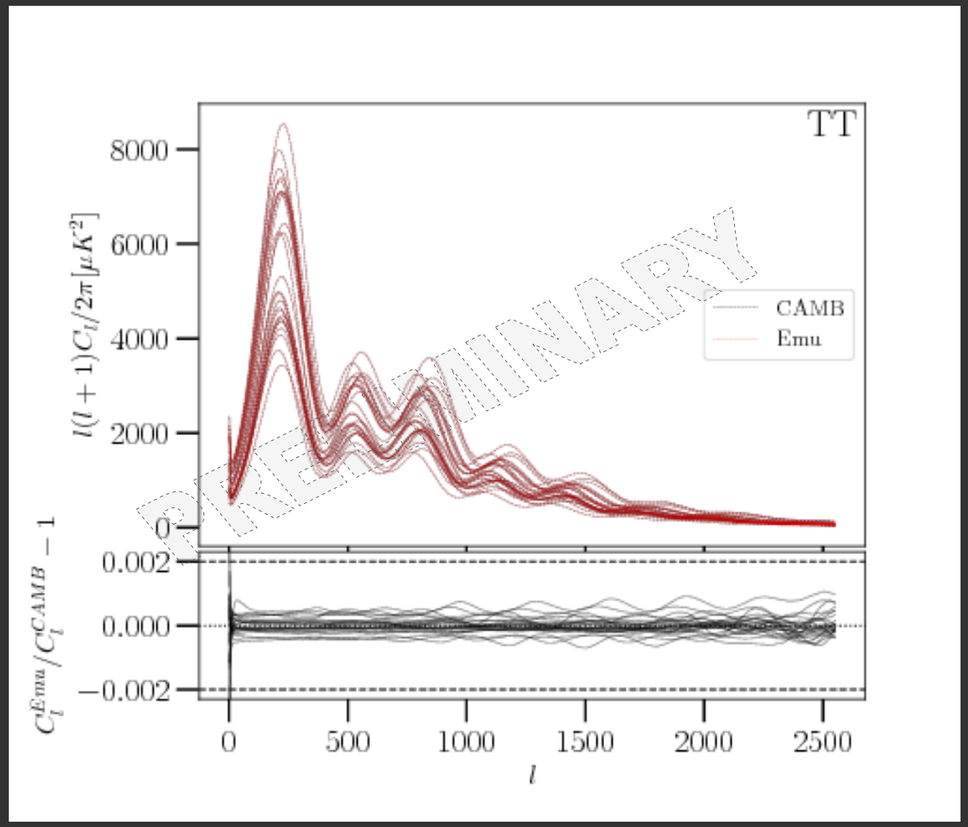
- The encoder output (latent space) at the bottleneck could be used for non-linear dimensionality reduction
- GP can then be used for *latent space-walk* (as a local interpolator in latent space)

Finally – the Emulators

- Emulation of CMB TT spectrum at > 99-*percent* accuracy – for both PCA and VAE used for data reduction
- CMB Emulators are 10-20 times faster than CAMB realizations
- Dark Matter power spectra is several orders of magnitude faster than Cosmological Simulations



CosmicEmu for Dark Matter Power spectra



CMB temperature anisotropy power spectra Emulator. Performance of Emulator on hold-out tests

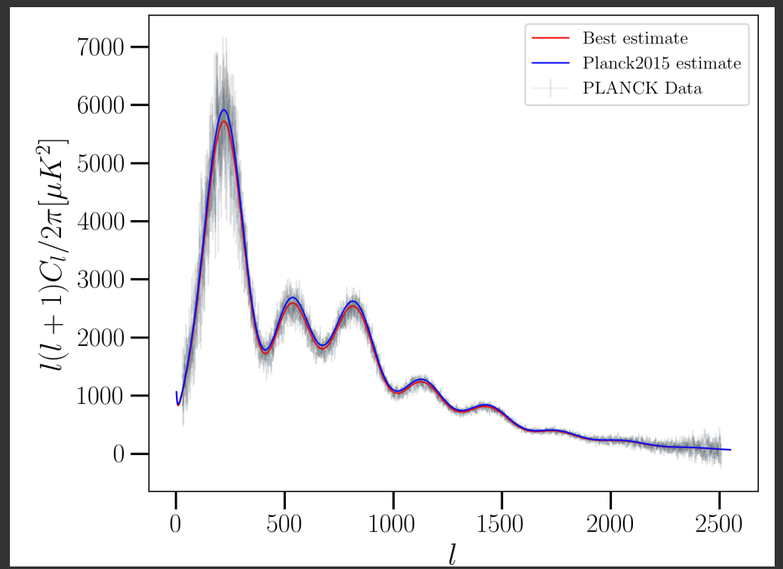
Cosmological Parameter calibration using Emulators in Likelihood methods

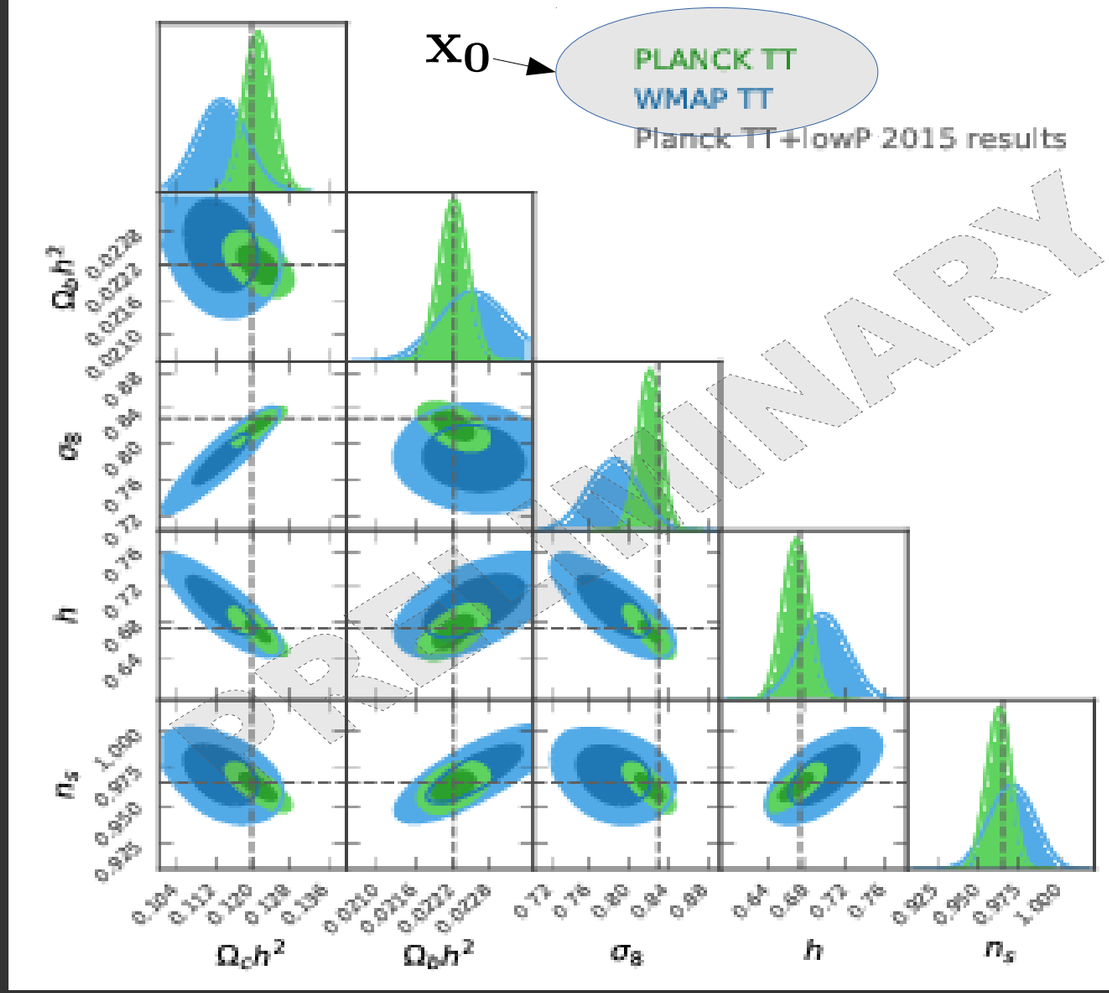
Posterior: $p(\theta|\mathbf{x}) = p(\mathbf{x}|\theta)p(\theta)$

$$P(x|\theta) \propto exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x_0})^T C^{-1}(\mathbf{x}-\mathbf{x_0})\right\}$$

From GP Emulator

- Covariance matrices may be difficult to compute.
- Likelihood may not be Gaussian.





Posteriors inferred from Cl Emulator-based MCMC for PLANCK and WMAP Satellite data

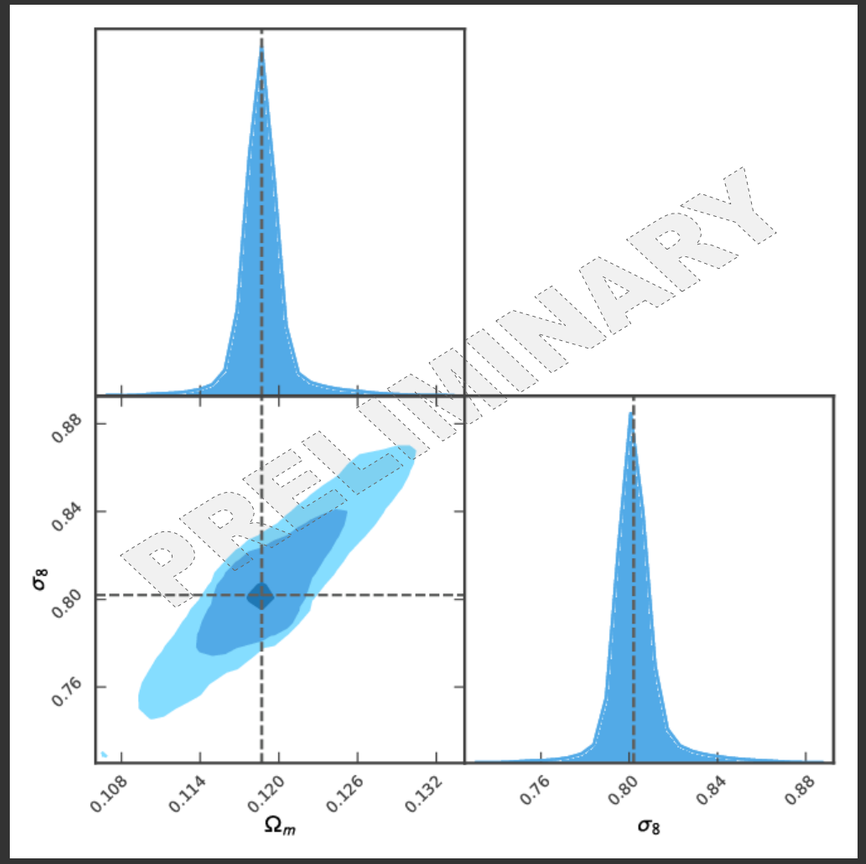
Emulators in Likelihood-free methods

Implemented on Sequential Monte Carlo sampler (astroABC: https://github.com/EliseJ/astroABC)

From GP Emulator

Metric: $$\rho(x, x_{sim}) = \sum_i \left\{ \frac{(x_i - x_{sim\,i})^2}{2\sigma_i^2} \right\}$$

My first LFI using GP emulator!

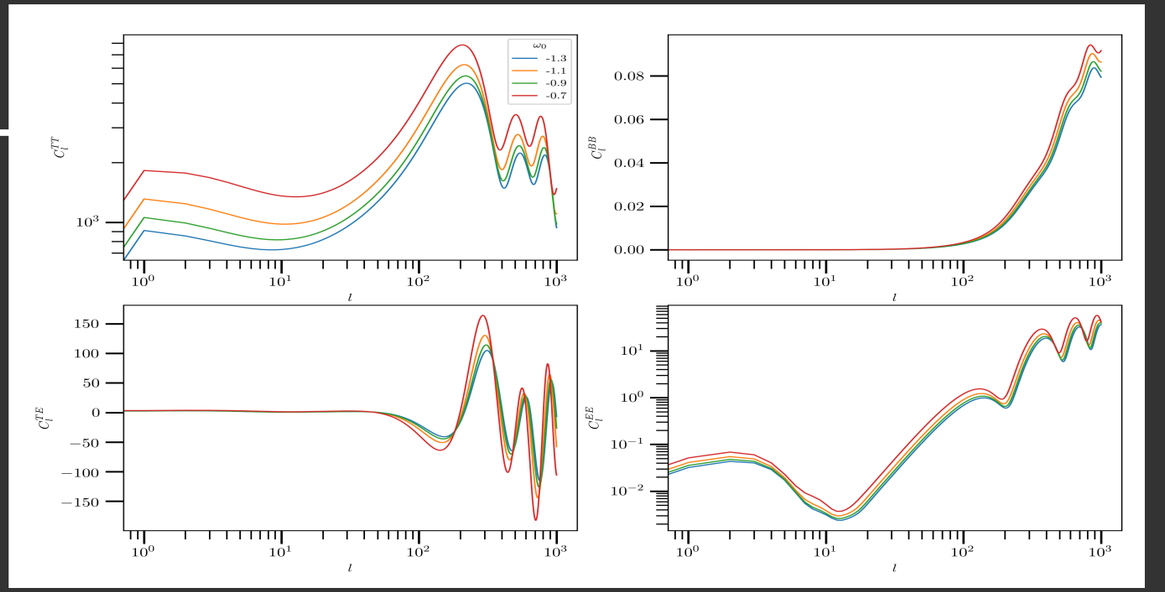


Parametric inference of 2 cosmological parameters using PLANCK Satellite data. With the emulator, the computation took a little over 30 mins on my laptop.

Suite of Emulators:

Currently we have several emulators:

- Dark matter power spectra
- Halo mass function
- Concentration-Mass relation
- Galaxy clustering power spectra
- CMB power spectra
  - TT, BB, EE, TE
  - using PCA and VAE
- Weak lensing shear power spectra
  - User- customizable emulators
- Modified Gravity emulators
  - for new cosmological models



Emulators are being constructed for the polarization and cross-power spectra for larger parameter space and will be used for analysis of South Pole Telescope (SPT) datasets.

Publicly available Emulators:
http://www.hep.anl.gov/cosmology/CosmicEmu/emu.html

Cosmic Emulators used in LSST DESC Core Cosmology Library: https://github.com/LSSTDESC/CCL

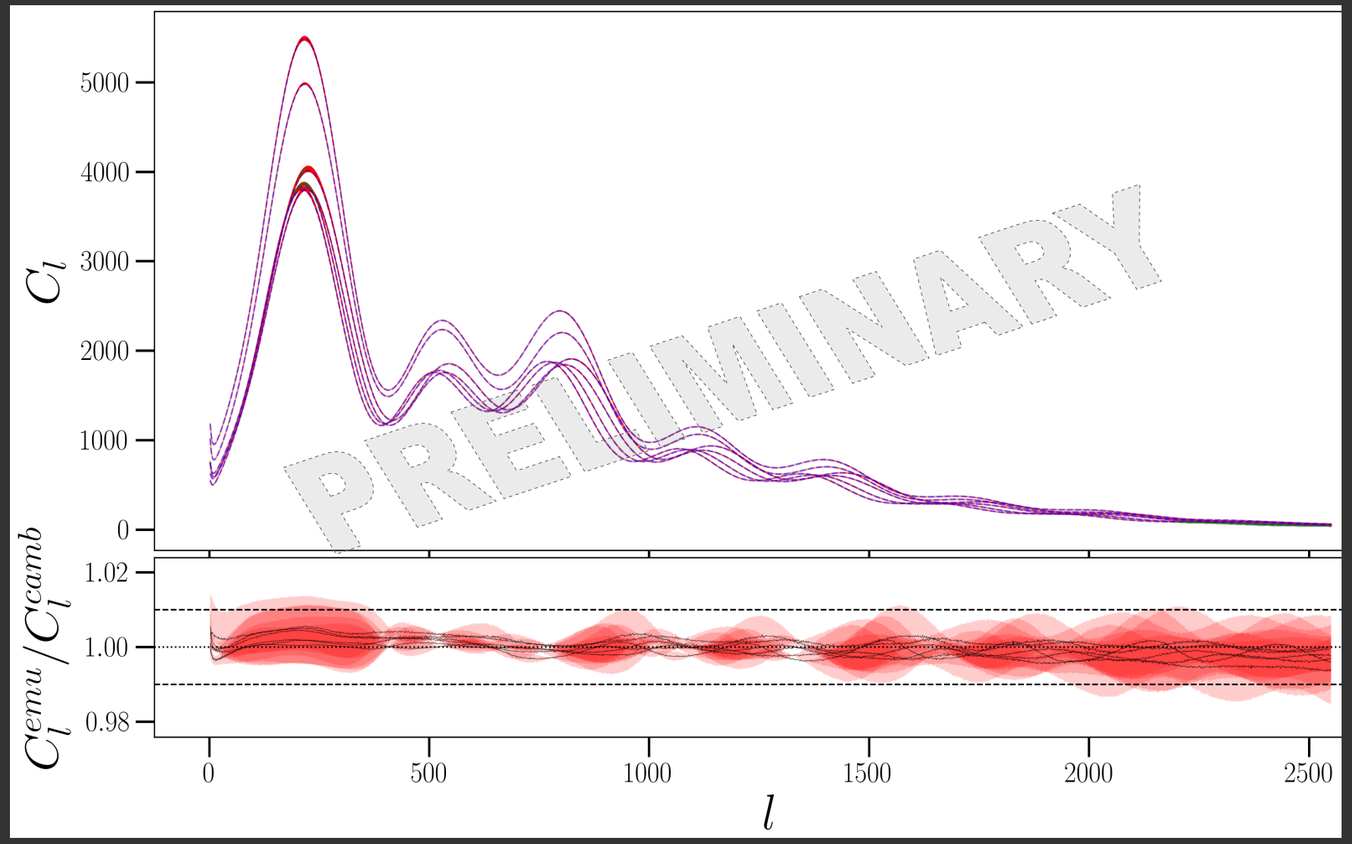
Ask me for CAMB emulators, weak lensing shear power spectra (or just to create new GP emulators).

Emulator Parameters

$\omega_m : [0.12, 0.155]$
$\omega_b : [0.0215, 0.0235]$
$\sigma_8 : [0.7, 0.9]$
$n_s : [0.85, 1.05]$
$h : [0.55, 0.85]$
$\tau : [0.01, 0.8]$
$N_{eff} : [1, 5]$
$\sum m_\nu : [0, 3]$
$r_{0.05} : [0, 2]$
$\omega_0 : [-1.3, -0.7]$
$\omega_a : [-1.73.1.28]$
$\omega_\nu : [0.0, 0.01]$

Finally:

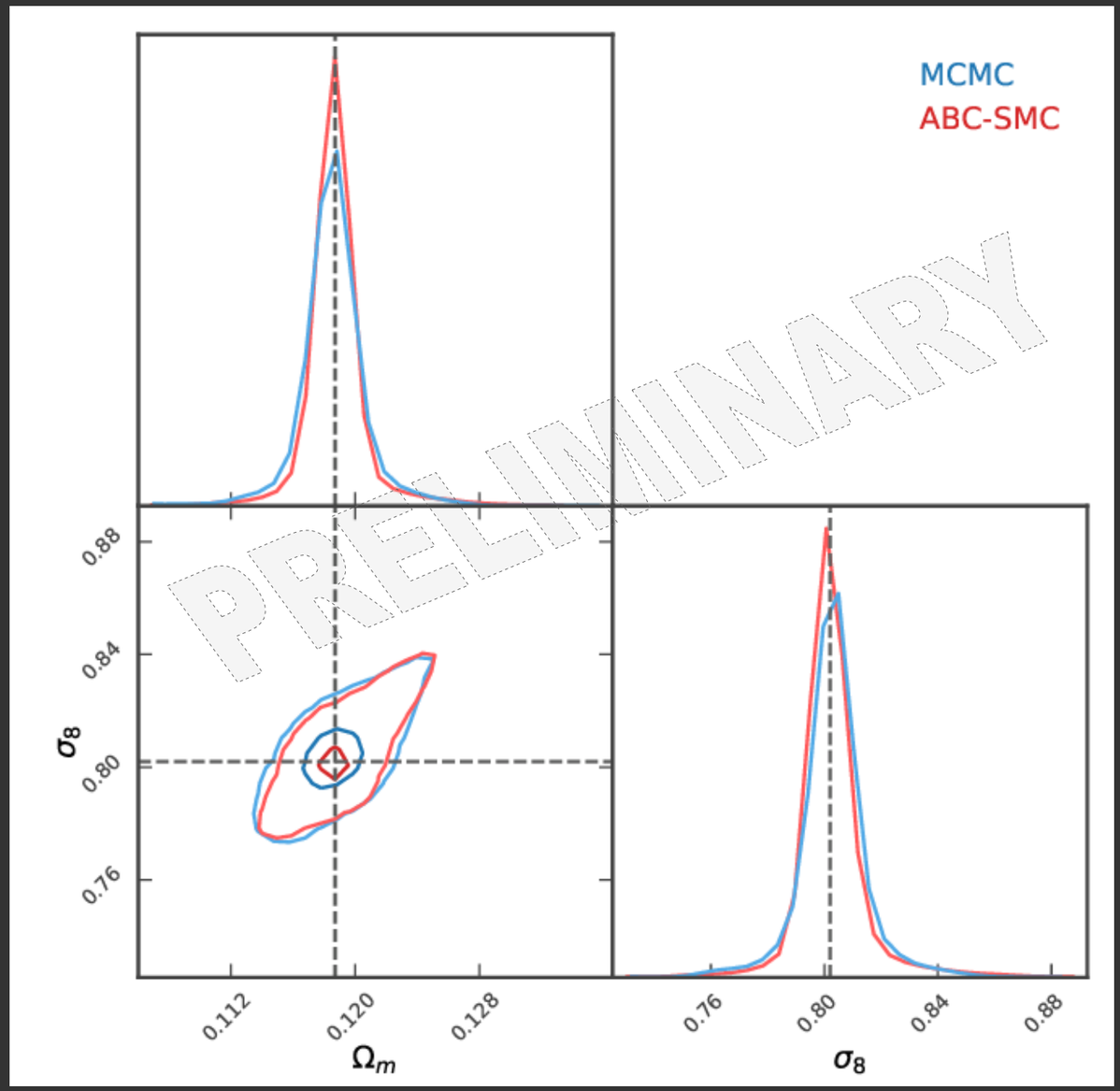
- Emulation of CMB TT spectrum at > 99-*percent* accuracy.
  - Variational AEs are used for nonlinear data reduction
  - Error bars calculated from GP covariance prediction
  - GP interpolation gives a unique insight into the hidden space of autoencoder
- Similar emulation techniques can be extended to polarization data, images and more.



Next steps (possible hack pitches?)

- Quick testing and implementations of MCMC vs LFI:
  - Choice of LFI methods/packages?
  - How does choice of summary statistics affect posteriors?

- Convergence tests of active learning techniques vs. nested sampling.
- Parallel active learning methods?
- How can we validate Emulator covariances?



Key References: http://www.hep.anl.gov/cosmology/CosmicEmu/emu.html and papers therein.

K. Heitmann, D. Higdon, C. Nakhleh, and S. Habib, ApJ Lett 646, 1 (2006)

S. Habib, K. Heitmann, D. Higdon, C. Nakhleh, and B. Williams, Phys. Rev. D 76, 083503 (2007)

K. Heitmann, D. Bingham, E. Lawrence, S. Bergner, S. Habib, D. Higdon, A. Pope, R. Biswas, H. Finkel, N. Frontiere, and S. Bhattacharya, ApJ 820, 108 (2016) [nested sampling, strong convergence]

Power Spectra: K. Heitmann, M. White, C. Wagner, S. Habib, and D. Higdon, ApJ 715, 104 (2010) [Coyote Universe I];

K. Heitmann, D. Higdon, M. White, S. Habib, B.J. Williams, and C. Wagner, ApJ 705, 156 (2009) [Coyote Universe II];

E. Lawrence, K. Heitmann, M. White, D. Higdon, C. Wagner, S. Habib, and B. Williams, ApJ 713, 1322 (2010) [Coyote Universe III];

K. Heitmann, E. Lawrence, J. Kwan, S. Habib, and D. Higdon, ApJ 780, 111 (2014) [Coyote Universe IV]

J. Kwan, K. Heitmann, S. Habib, N. Padmanabhan, E. Lawrence, H. Finkel, N. Frontiere, and A. Pope, Phys. Rev. D 810, 35 (2015) [galaxy power spectrum]

Other Examples: • J. Kwan, S. Bhattacharya, K. Heitmann, and S. Habib, ApJ 768, 123 (2013) [Halo shape emulation]

Auto-Encoding Variational Bayes, Kingma & Welling 2013 (arXiv: 1312.6114)

D. Higdon, K. Heitmann, C. Nakhleh, and S. Habib, in the Oxford Handbook of Applied Bayesian Analysis edited by O' Hagan and West (Oxford, 2010)

Gaussian Processes for Machine Learning, Rasmussen and Williams 2006 (http://www.gaussianprocess.org/gpml/)

Backup slides

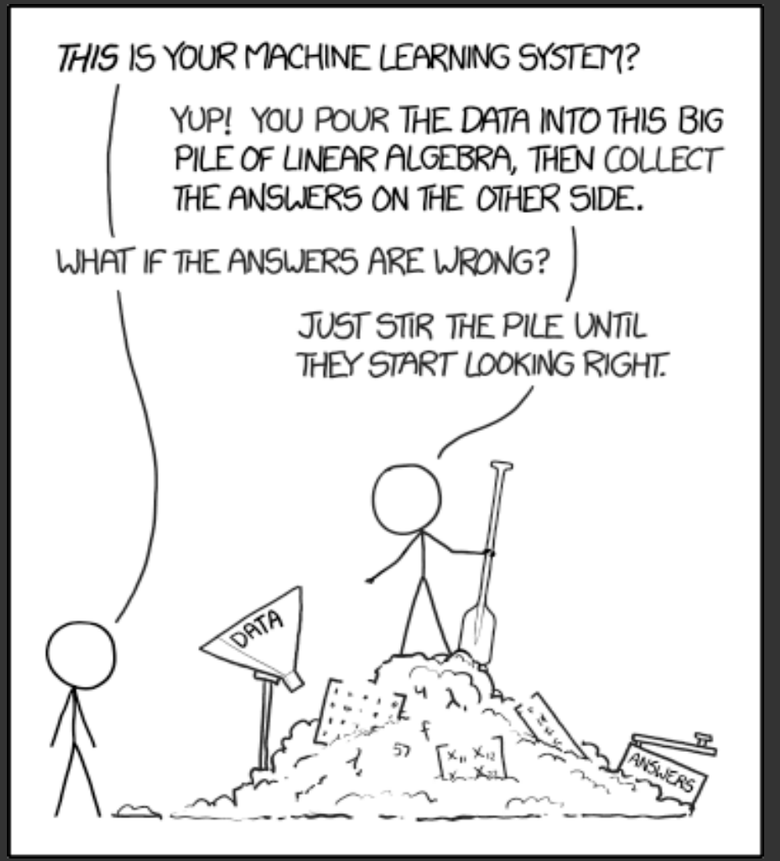
Deep concerns

- Garbage in, garbage out
  - Usually require large quantity of high fidelity data and simulation
  - There are tricks for small data sizes (Transfer learning, data augmentation)

- Too many hyper-parameters to tune
  - No straightforward way to find the best network parameters
  - Bayesian optimization techniques are currently being explored.

- Key features usually omitted: Uncertainty quantification, sensitivity analyses, prior knowledge integration.

- Deep Learning networks are not sufficiently transparent
  - Often perceived as ‘black-boxes’ - Weights, activations work under-the-hood



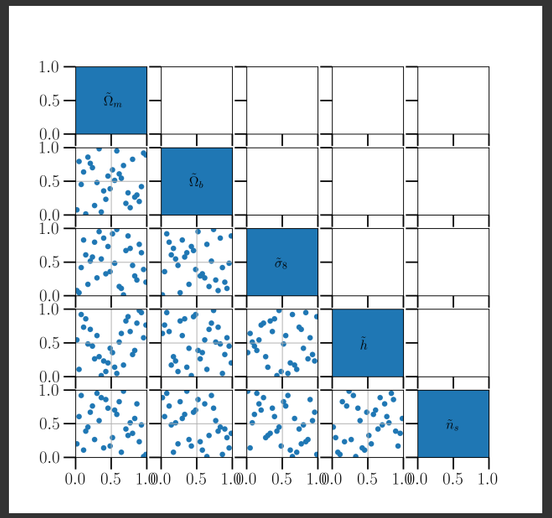
https://xkcd.com/1838

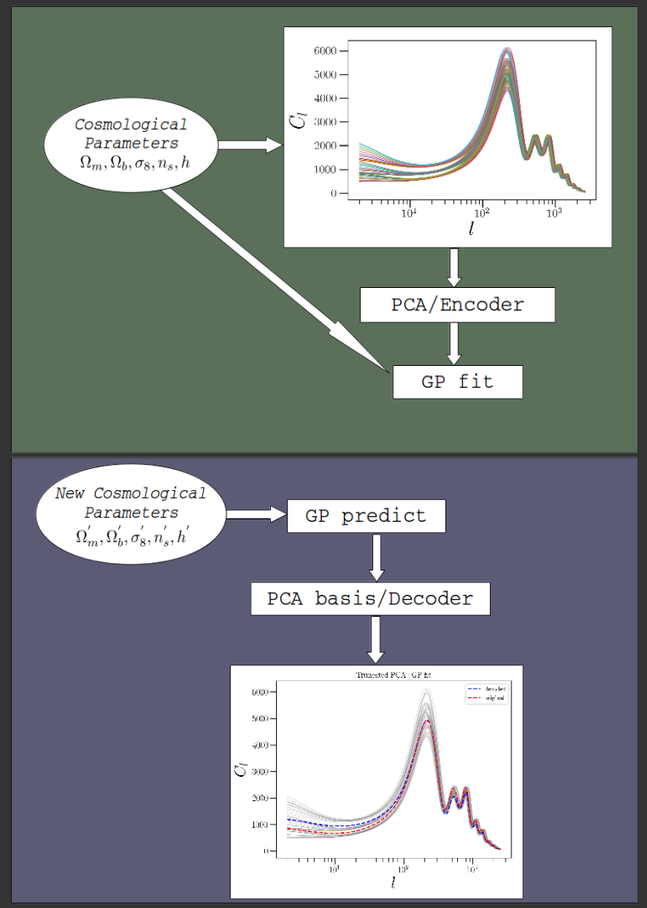
Variational Autoencoders + GP

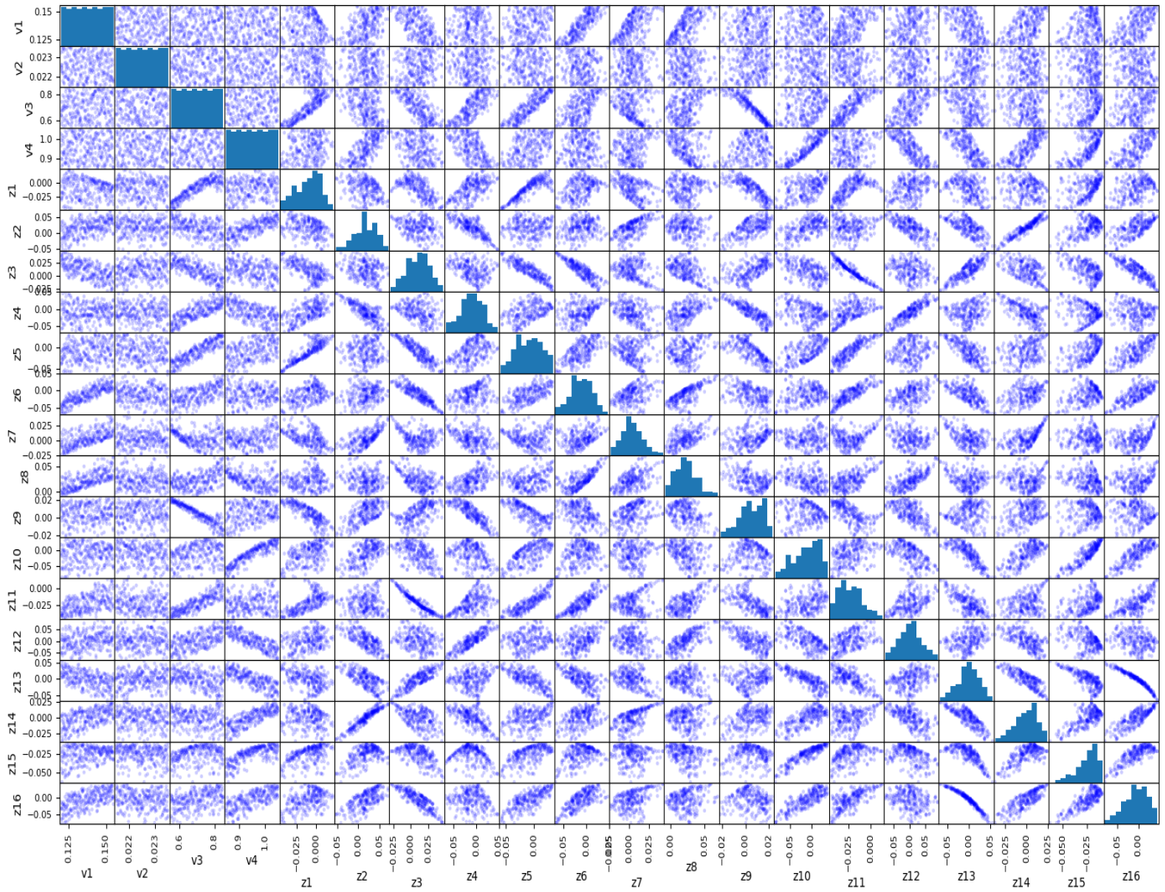
Training data generated from CAMB, using a latin hypercube design.

(v1, v2, v3, v4): $(\Omega_m, \Omega_b, n_s, h)$

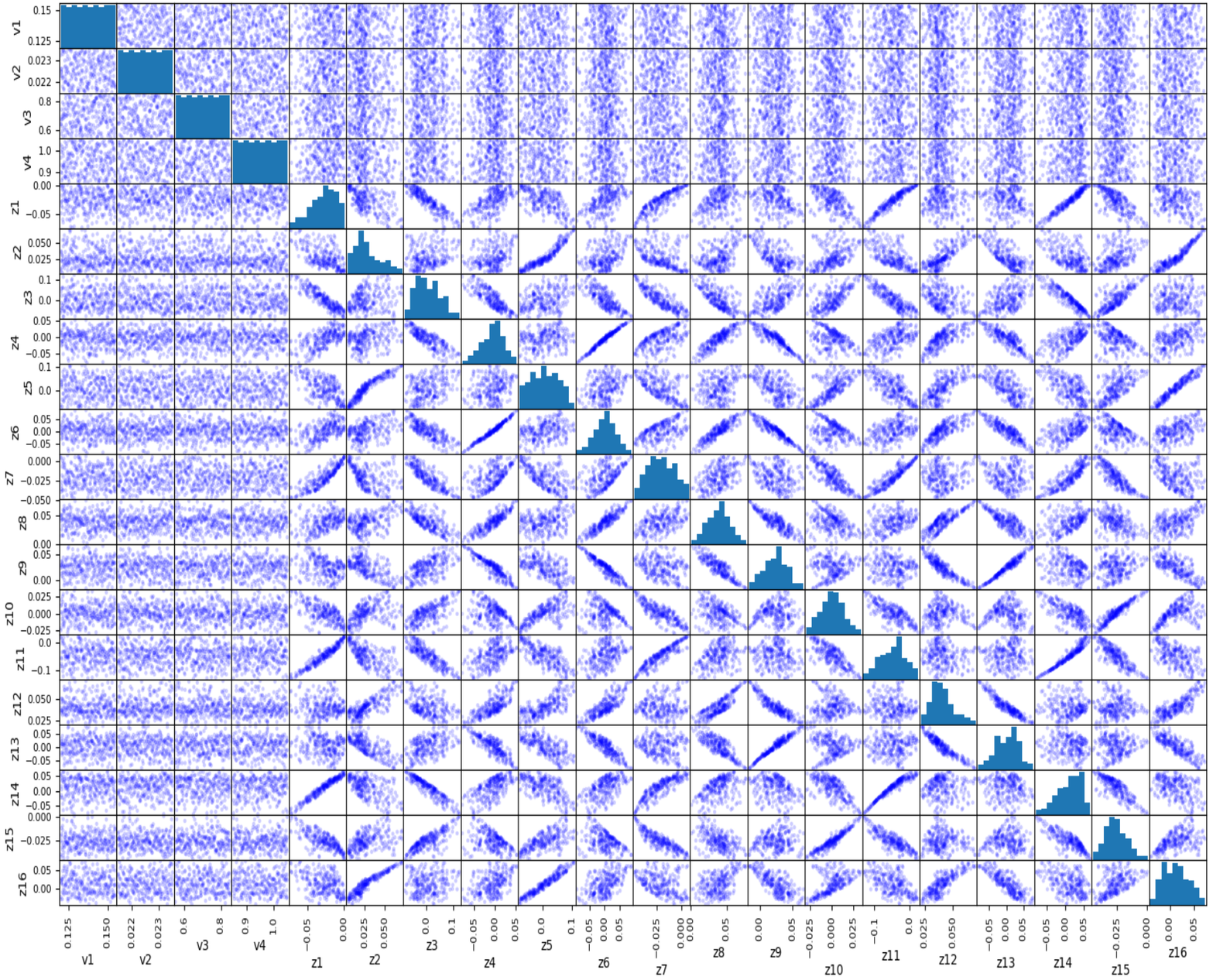
(z1, z2, ...): latent space parameters





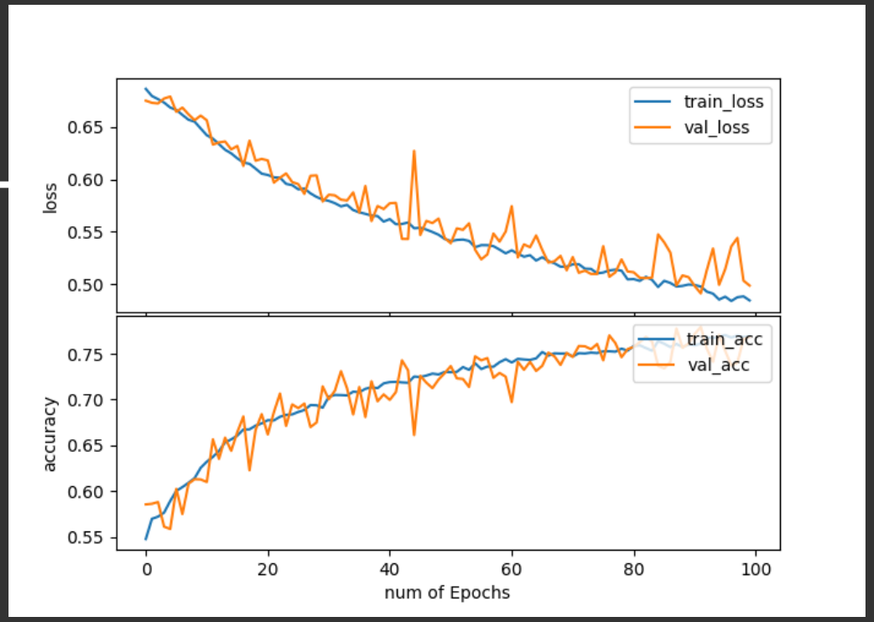


How to fool your network ?

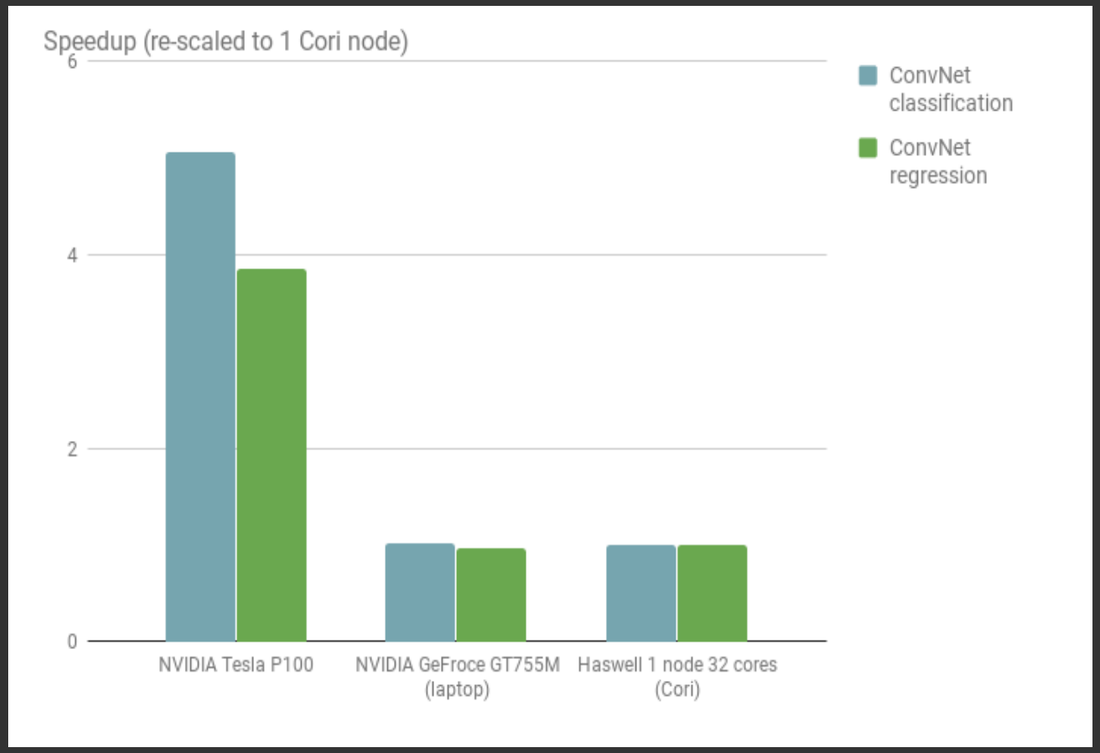


Sample training

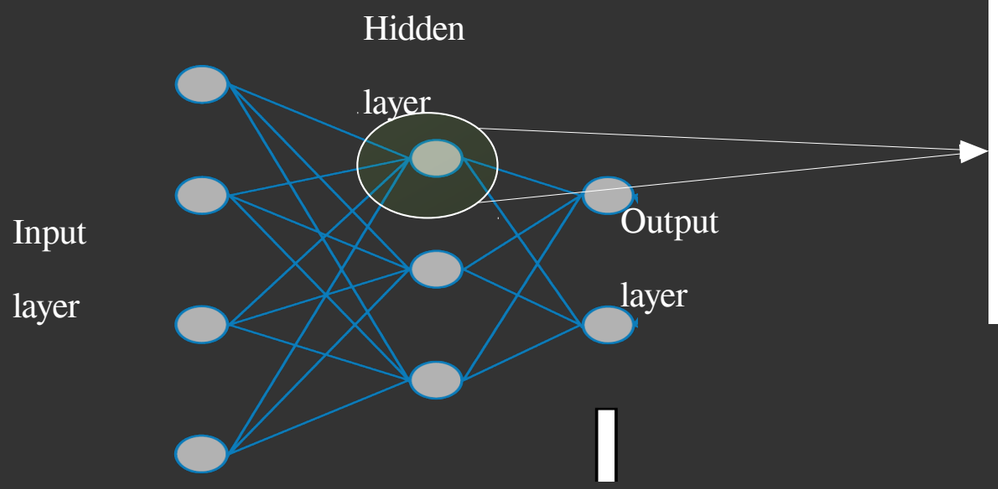
- Optimization problem of reducing loss, maximizing accuracy
- Hyper-parameters
  - Learning rate, Decay rate, Total epochs, Batch size
  - Choice of loss function, optimizers, regularization
  - Network architectures



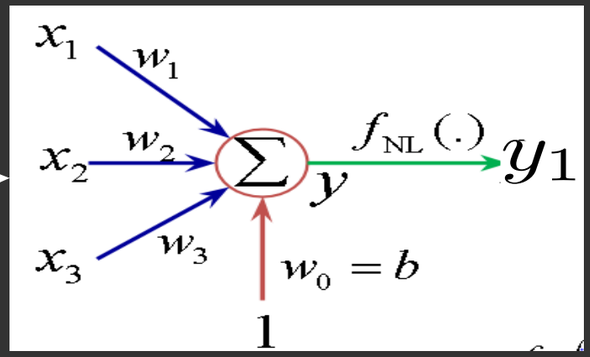
- 80-90 per cent accuracy in 30 mins on 1 Intel-Haswell node with 32 CPU cores (Cori supercomputer at NERSC)
- About the same time on NVIDIA GeForce GT 755M with 384 GPU cores.
- Speedup on state-of-the-art GPUs (NVIDIA Pascals/Tesla) is about 10-15 times faster.
- Testing is very fast: Classification time is $O(10^{-5})$ seconds per image.



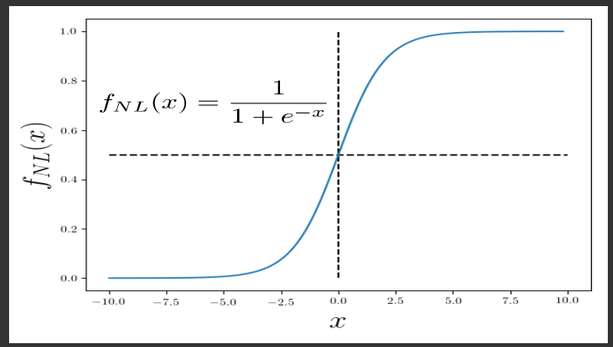
Neural Network is just one of many ML techniques



Multi-layer neural network



One neuron



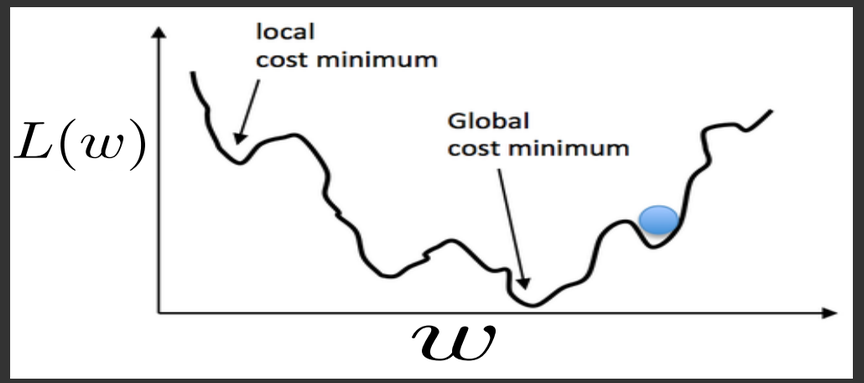
Activation function

Output:

$$y_j = f_{NL}\left(\sum_{i=1}^{n}(W_i x + b_i)\right)$$

$$L(w) = \sum_i (y_i^{pred}(w) - y_i^{real})$$

Loss function



Minimizing loss is the objective, achieved through **gradient descent** methods. The weights are updated at every **epoch** via **backpropagation**