UNIVERSITY OF CAMBRIDGE

Lecture 15
Signal Emulation for Astrophysics and Cosmology

Lecturer: **Dr Harry Bevins** (htjb2)

Overview

- Why we need signal emulators?
- Example of a 21-cm signal emulator
- Example Dimensionality Reduction
- Brief discussion of VAEs for SKA tomography and power spectrum

Slides available at https://github.com/htjb/Talks (and Moodle!)

Example codes on github too!

UNIVERSITY OF CAMBRIDGE

Why we need signal emulators?

Recap:Bayes theorem

Posterior: $P(\theta)$

Evidence: Z

Likelihood: $L(\theta)$

Prior: $\pi(\theta)$

$$\underbrace{P(\theta|D)P(D)}_{\text{Outputs}} = \underbrace{P(D|\theta)P(\theta)}_{\text{Inputs}}$$

For Nested Sampling and other MCMC algorithms

The likelihood function

- The likelihood function gives the probability of the *data given the model and parameters* $L(\theta) = P(D|\theta, M)$

- Defined as an input to Nested Sampling and other MCMC algorithms

- A lot of research goes into the form of the likelihood which varies for different analysis problems (e.g. Scheutwinkel et al. [2204.04491], Lovick et al. [2312.02075])

UNIVERSITY OF CAMBRIDGE

An example likelihood

$$\log_e L(\theta) = \sum_i -\frac{1}{2}\log_e |2\pi\Sigma| - \frac{1}{2}\left(D_i - M_i(\theta)\right)^T \Sigma^{-1}(D_i - M_i(\theta))$$

- Work in log space to make things more computationally stable
- The functional form of the likelihood defines the noise distribution in your data
- Here we are assuming the noise is Gaussian and allowing for a non-diagonal covariance

The issue

$$\log_e L(\theta) = \sum_i -\frac{1}{2}\log_e|2\pi\Sigma| - \frac{1}{2}\big(D_i - M_i(\theta)\big)^T \Sigma^{-1}(D_i - M_i(\theta))$$

- In a Nested Sampling run the likelihood function is evaluated for different θ hundreds of thousands to millions of times
- This means we need to evaluate the model $M(\theta)$ millions of times
- But the models are often computationally expensive taking minutes to hours to days per realization

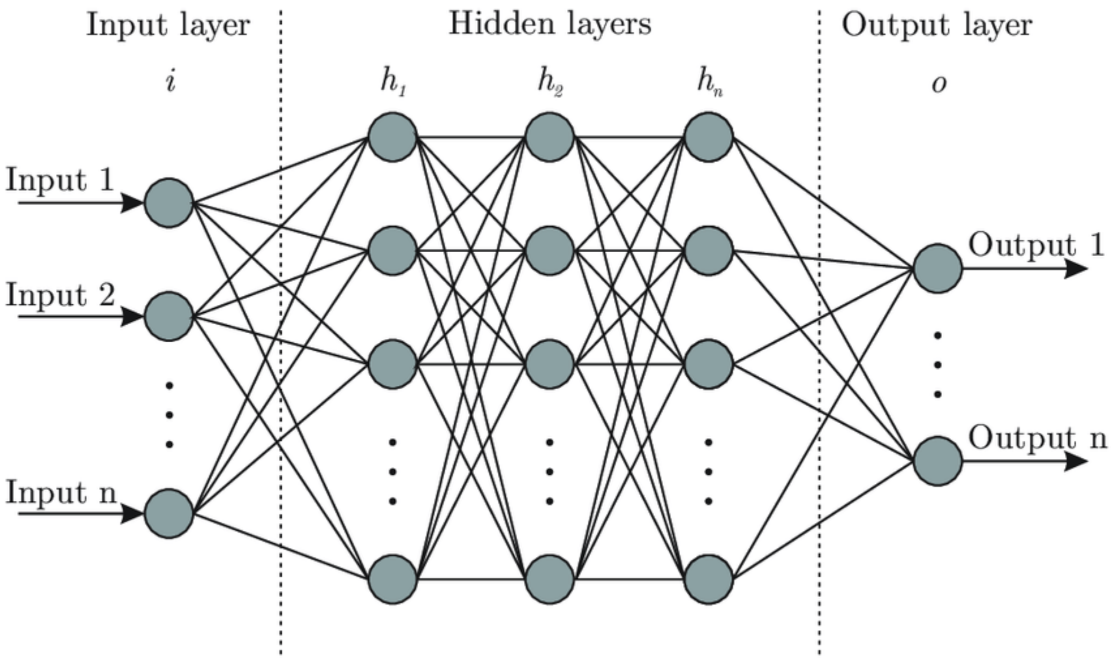
The issue

$$\log_e L(\theta) = \sum_i -\frac{1}{2}\log_e|2\pi\Sigma| - \frac{1}{2}\big(D_i - M_i(\theta)\big)^T \Sigma^{-1}(D_i - M_i(\theta))$$

- How do we get around this?
- Look towards emulators which can approximate $M(\theta)$ in less than a few milliseconds
- Go from millions of hours to hours per Nested Sampling run

What actually is an emulator?

- Some approximation of a complex astrophysical model (think N-body, hydrodynamical or semi-numerical simulations)
- Often built with machine learning tools
- Artificial neural networks and Convolutional neural networks



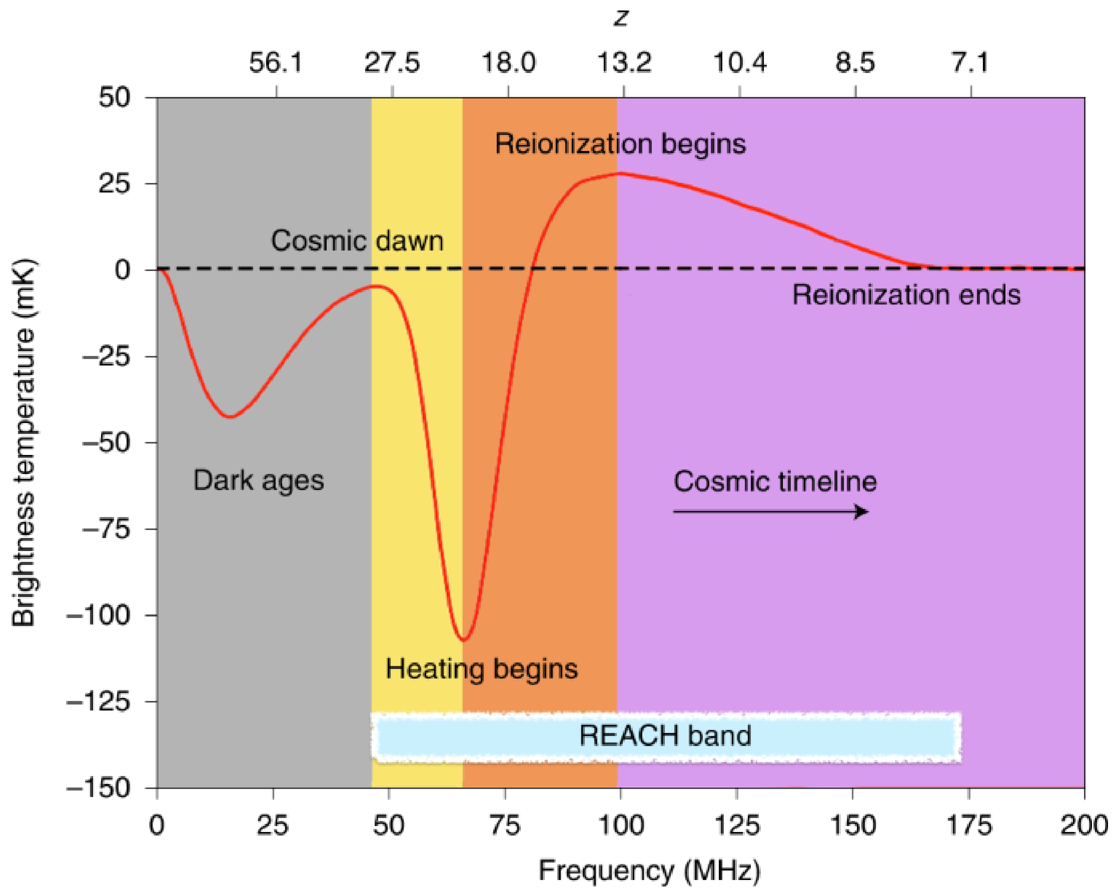
UNIVERSITY OF CAMBRIDGE

Sky-averaged 21-cm Cosmology

Sky averaged 21-cm Cosmology

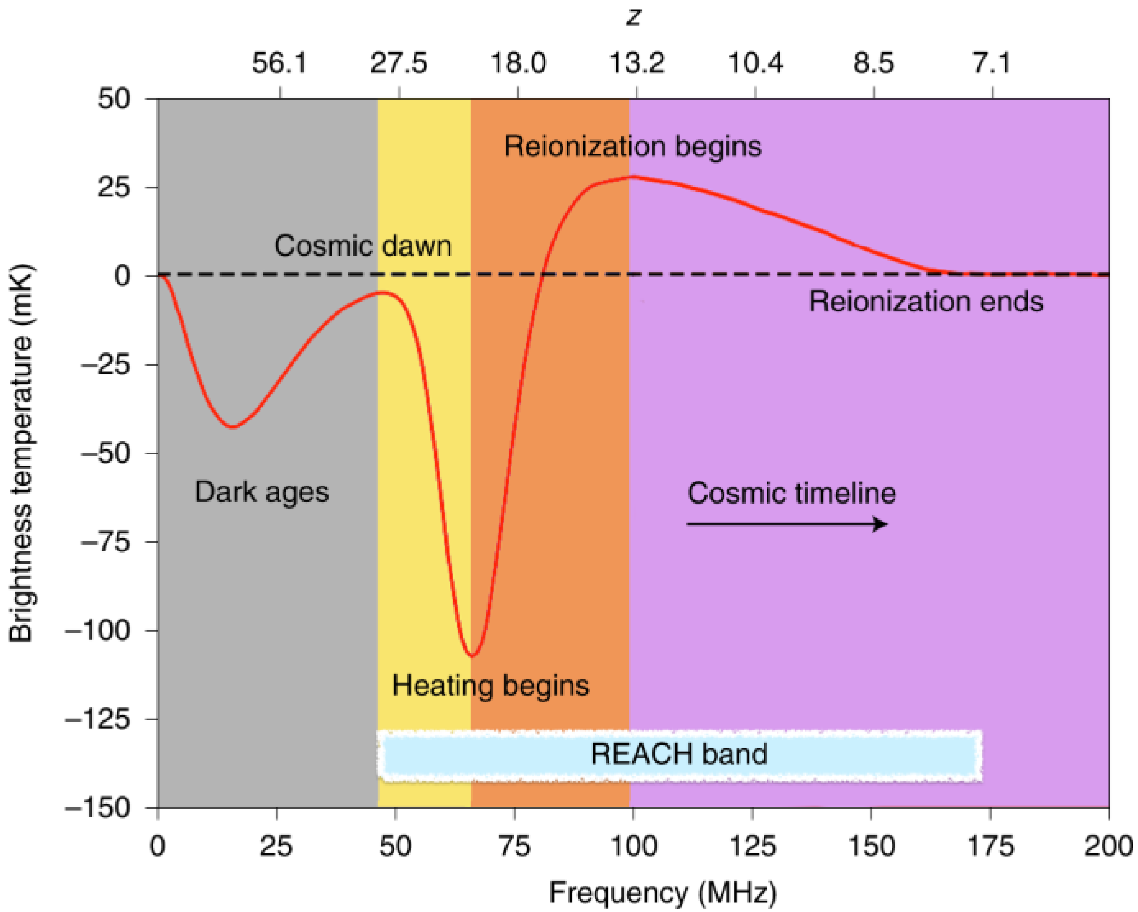
- Looking for a spectral distortion in the CMB temperature caused by neutral hydrogen

- A detection of the signal will help us understand
  - When the first stars formed ad how bright they were
  - The nature of dark matter
  - The abundance and brightness of X-ray emitting objects
  - When the universe transformed from neutral to ionized



Sky averaged 21-cm Cosmology

- Complex dependence between T_{21} and the parameters of our models θ
- We use semi-numerical simulations to model and study how the signal varies over space and time for given θ
- One signal realization takes $\approx$ 3hrs



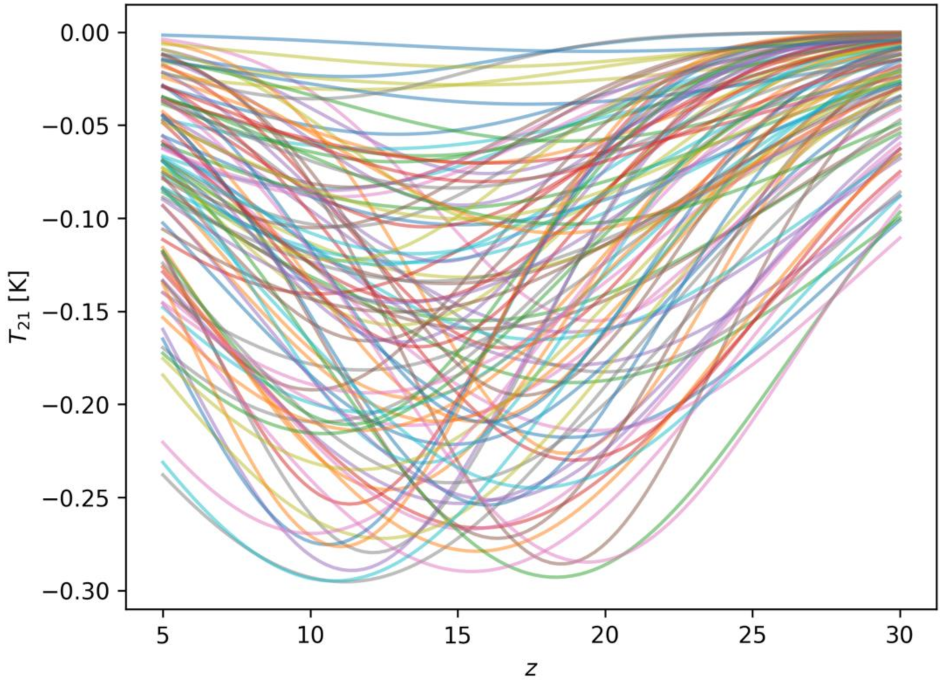
A toy 21-cm Cosmology example

- We can approximate the 21-cm signal with a Gaussian absorption feature

- Parameterised by
    - An amplitude A
    - A width σ
    - A central redshift z_c

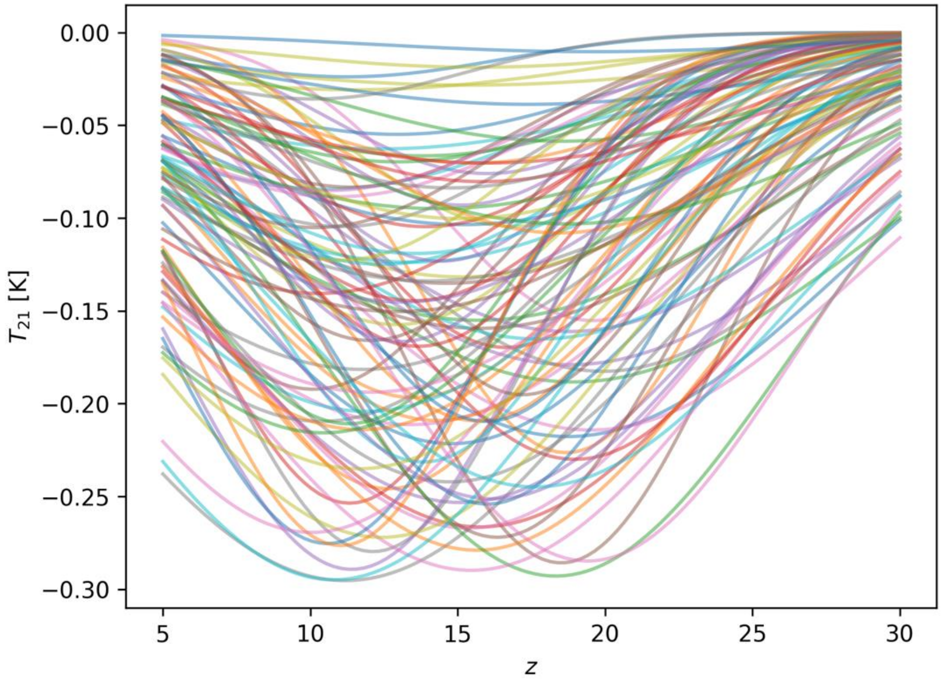
- We want to approximate the simulation with a neural network $\mathrm{T}_{21}(\theta) \approx f_{\phi}(\theta)$

```python
def gaussian(parameters):
    """a simple Gaussian function"""
    return -parameters[0] * \
        np.exp(-0.5*(z - parameters[1])**2/
                parameters[2]**2)
```



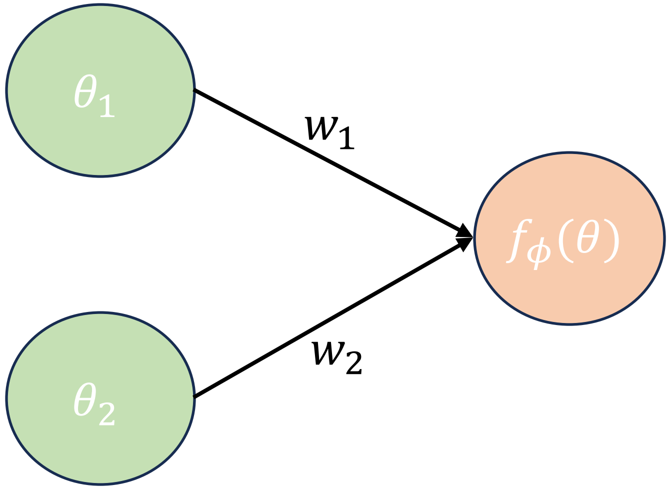
Designing the emulator

- Need to consider
  - Loss function and early stopping
  - Architecture and dimensionality reduction
  - Normalization
  - Activation functions
  - Measures of accuracy



Emulators as an approximation

- In $\mathrm{T}_{21}(\theta) \approx f_{\phi}(\theta)$, ϕ are the parameters of our neural network
- ϕ has to be optimized so that the approximation is accurate as possible
- Can say $\mathrm{T}_{21}(\theta) = f_{\phi}(\theta) + \epsilon_{\phi}(\theta)$ where we are attempting to minimize the error $\epsilon_{\phi}(\theta)$



$$f_\phi(\theta) = \sigma(w_1\theta_1 + w_2\theta_2)$$
where
$$\phi = \{w_1, w_2\}$$
and σ is an activation function

Loss Function

- In order to minimize the error $\epsilon_\phi(\theta)$ we have to provide the network with training data $\{T_{21}, \theta\}$

- For each example we evaluate some measure of error e.g.

$$L = \frac{1}{N}\sum_i \left|\epsilon_\phi(\theta)\right| \text{ or } L = \frac{1}{N}\sum_i \left(\epsilon_\phi\right)^2$$

- And adjust ϕ using Stochastic Gradient descent to minimize L

UNIVERSITY OF CAMBRIDGE

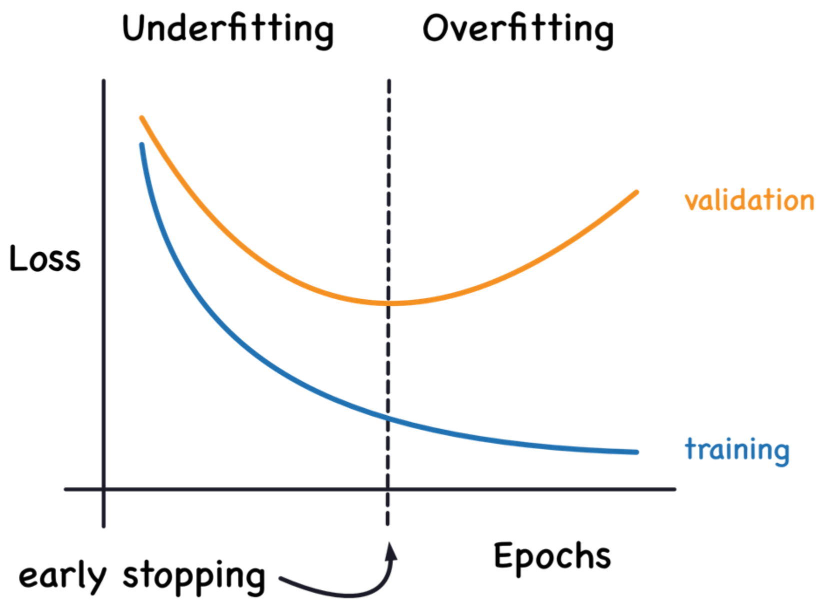
Training and test data

- We typically generate a few thousand simulations for training

- We reserve a proportion for testing emulator accuracy during training (early stopping)

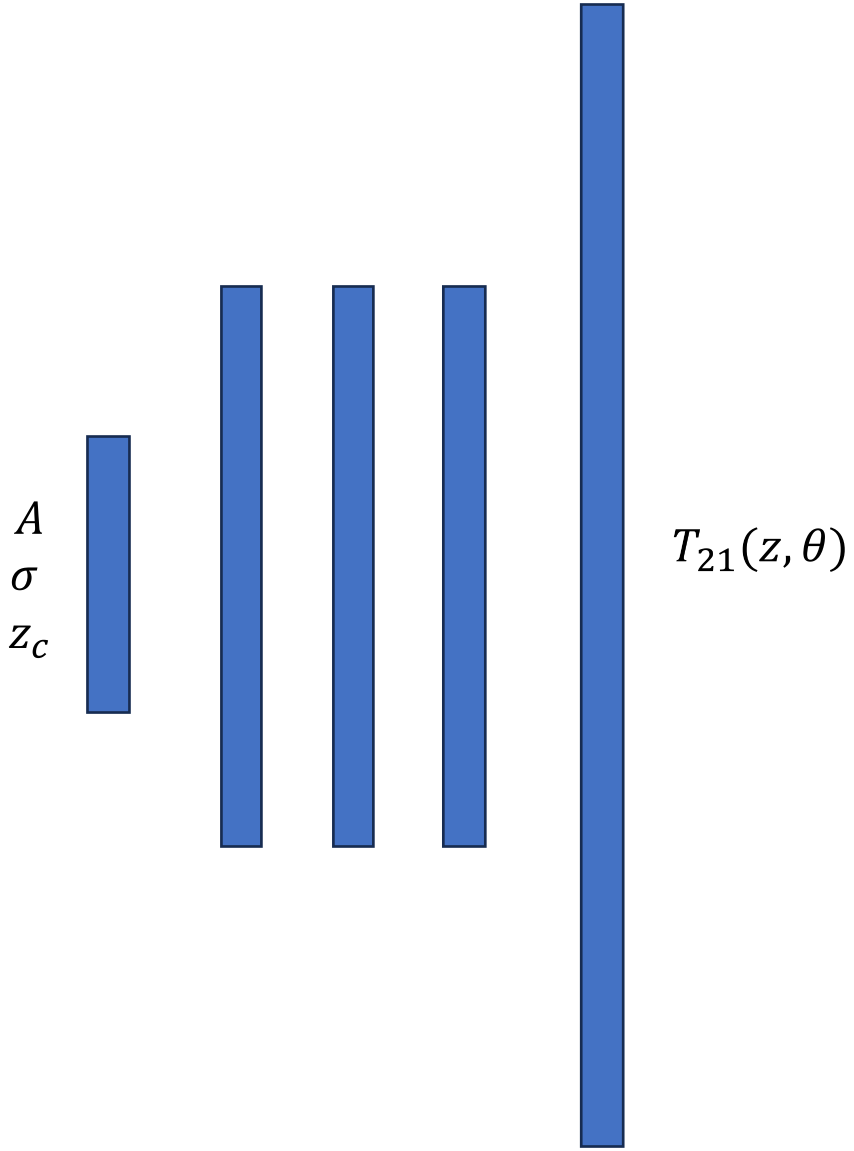
- Prevents overfitting of the training data

```
# split the data
idx = random.sample(range(n), int(n*0.8))
train_params_pretile = parameters[idx]
train_signals_pretile = signals[idx]
test_params_pretile = np.delete(parameters, idx, axis=0)
test_signals_pretile = np.delete(signals, idx, axis=0)
```



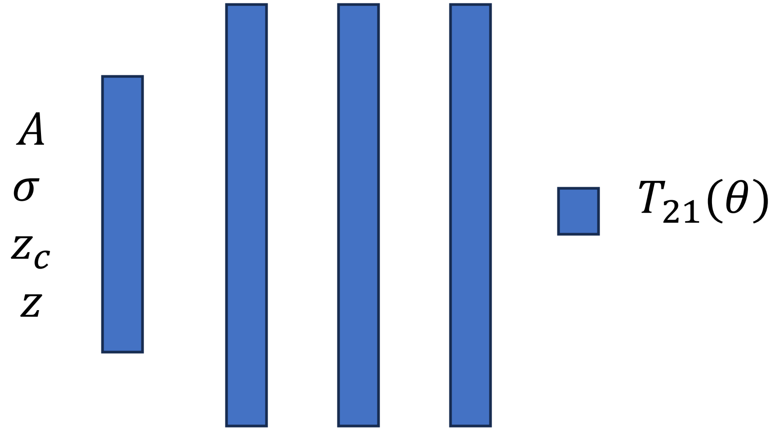
Architecture choices

- We could attempt to train a network to go directly from θ to $T_{21}(z)$
- However, we might want to evaluate the 21-cm signal at a range of different redshifts
- Leads to a big network and lots of parameters to optimize



Architecture choices

- We could perform some form of dimensionality reduction such as PCA (see later)
- But for 21-cm Cosmology we choice to make redshift (the independent variable) an input to our network
- Loop over the network to predict $T_{21}(z, \theta)$



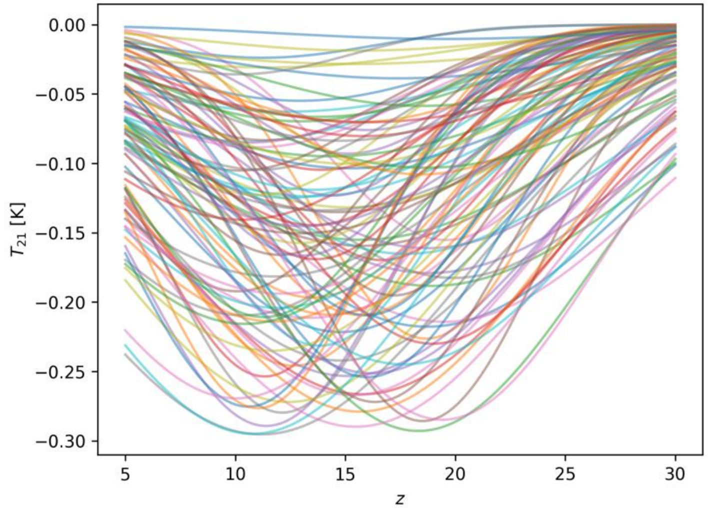
UNIVERSITY OF CAMBRIDGE

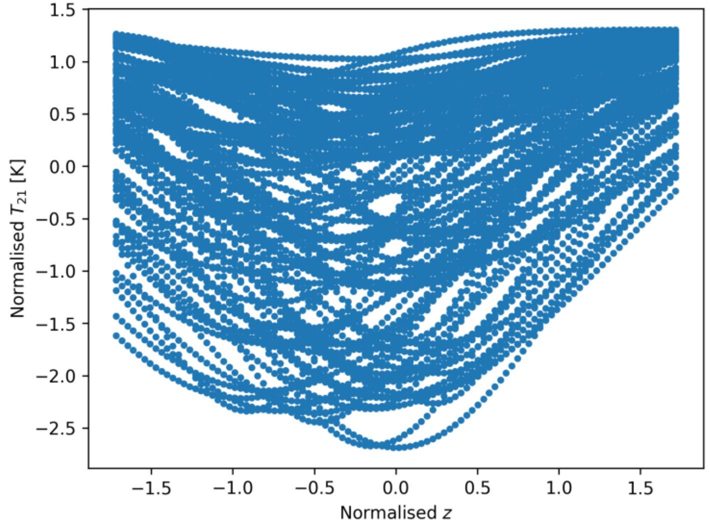
Normalisation and data processing

- We have to carefully format our data so that we input $\{A, \sigma, z_c, z\}$ and output $T_{21}(z, \theta)$

- Perform standardization on training and test data sets

$$\tilde{A} = \frac{A - \bar{A}}{\sigma_A}$$

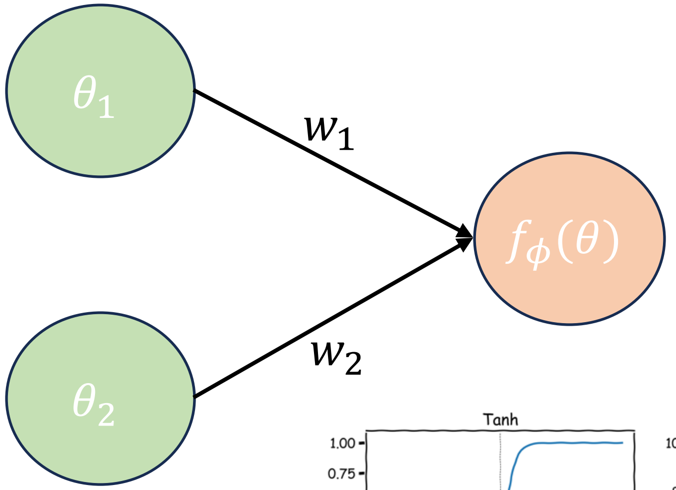




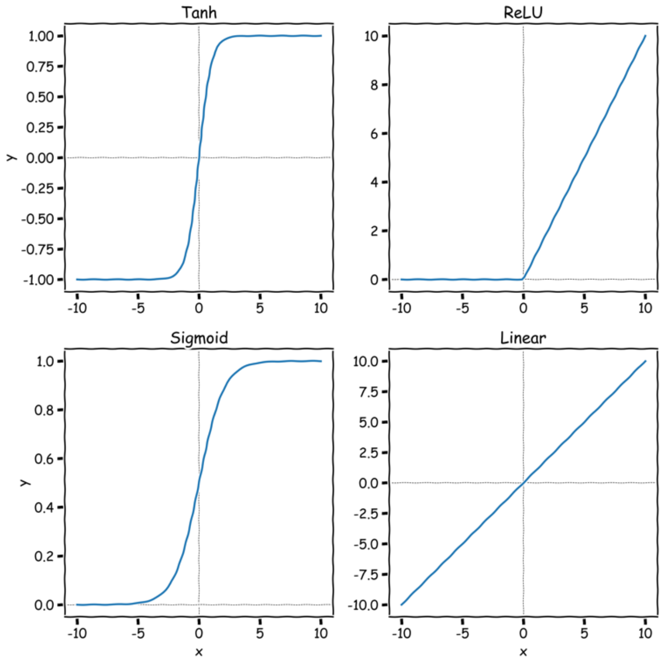
UNIVERSITY OF CAMBRIDGE

Activation Functions

- Activation functions add non-linearity into our modelling
- We often want to make careful choices for our activation functions
- E.g. if we know our output (after normalization) should be between 0-1 we might choose a sigmoid activation



$$f_\phi(\theta) = \sigma(w_1\theta_1 + w_2\theta_2)$$
where
$$\phi = \{w_1, w_2\}$$
and σ is an activation function



Building the neural network

```
# callback for early stopping
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)

# neural network architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(4, activation='sigmoid'),
    tf.keras.layers.Dense(8, activation='sigmoid'),
    tf.keras.layers.Dense(8, activation='sigmoid'),
    tf.keras.layers.Dense(8, activation='sigmoid'),
    tf.keras.layers.Dense(8, activation='sigmoid'),
    tf.keras.layers.Dense(1, activation='linear'),
])

# building the model with the adam optimizer and mean squared error loss function
model.compile(optimizer='adam',
              loss='mse',)
```

```
# training the model
model.fit(train_params, train_signals, epochs=200, batch_size=250,
          callbacks=[callback], validation_data=(test_params, test_signals))
```

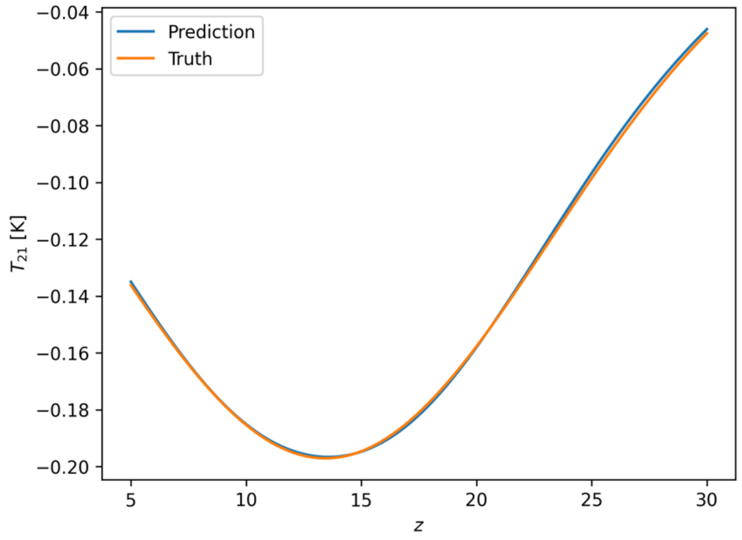
Assessing the accuracy

- Once trained we want to assess whether the network is doing a good job before we use it in our Bayesian analysis

- The network will predict signals in the normalized space

```
def prediction(params):
    """
    This function takes in a set of unormalized parameters (A, zc, sigma)
    and returns a predicted signal in the unormalized space.
    """
    params = np.tile(params, len(z)).reshape(len(z), 3)
    params = np.hstack((params, z.reshape(-1, 1)))
    params = (params - norm_param_means) / norm_param_stds
    pred = model.predict(params, verbose=0)
    return pred*norm_signal_stds + norm_signal_means

pred = prediction(test_params_pretile[100])
plt.plot(z, pred)
plt.plot(z, test_signals_pretile[100])
plt.xlabel('z')
plt.ylabel('signal')
plt.show()
```



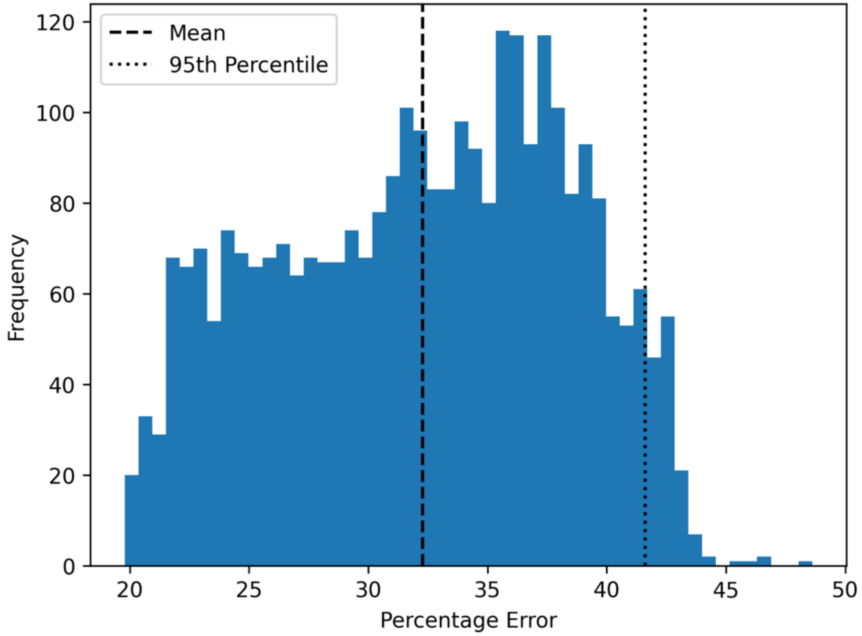
Assessing the accuracy

- Often we want a more general measure of accuracy

- We assess on the whole test data set and take the average and 95th percentile values

- This network isn't very accurate!

```
error = []
for i in tqdm(range(len(test_params_pretile))):
    pred = prediction(test_params_pretile[i])
    error.append(100*np.mean(np.abs(pred - test_signals_pretile[i]))
                /np.max(np.abs(test_signals_pretile[i])))
error = np.array(error)
```



UNIVERSITY OF CAMBRIDGE

Current state of the art

- There are many approaches for emulating the sky-averaged 21-cm signal

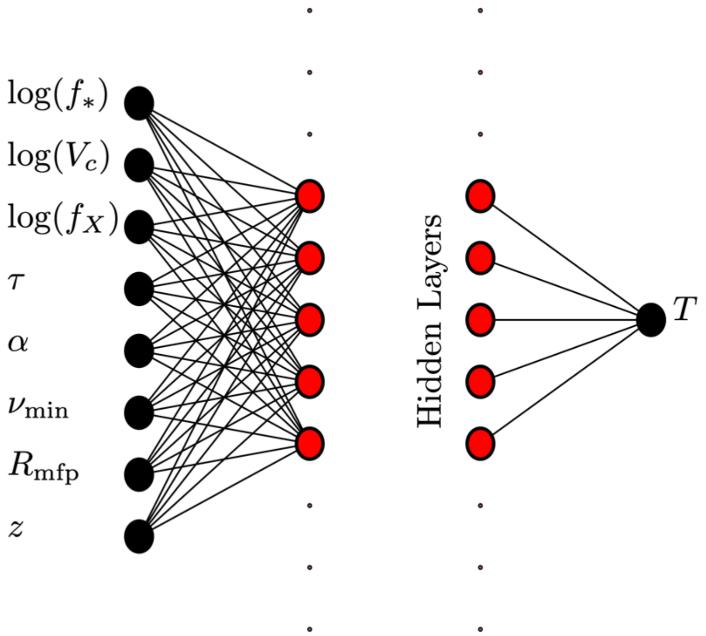
- globalemu [2104.04336] uses some of the ideas discussed here

- It is light weight, easy to retrain and very fast while maintaining a good level of accuracy

globalemu: Robust and Fast Global 21-cm Signal Emulation

Introduction

globalemu:	Robust Global 21-cm Signal Emulation
Author:	Harry Thomas Jones Bevins
Version:	1.8.2
Homepage:	https://github.com/htjb/globalemu
Documentation:	https://globalemu.readthedocs.io/



UNIVERSITY OF CAMBRIDGE

Current state of the art

- 21cmVAE [2107.05581] uses Variational Autoencoders
- More complicated architecture, harder to retrain and an order of magnitude slower than globalemu
- But more accurate than globalemu
- Also see 21cmLSTM [2410.07619]

21cmVAE: A Very Accurate Emulator of the 21-cm Global Signal

Christian H. Bye,[1,2] Stephen K. N. Portillo,[3] and Anastasia Fialkov[4,5]

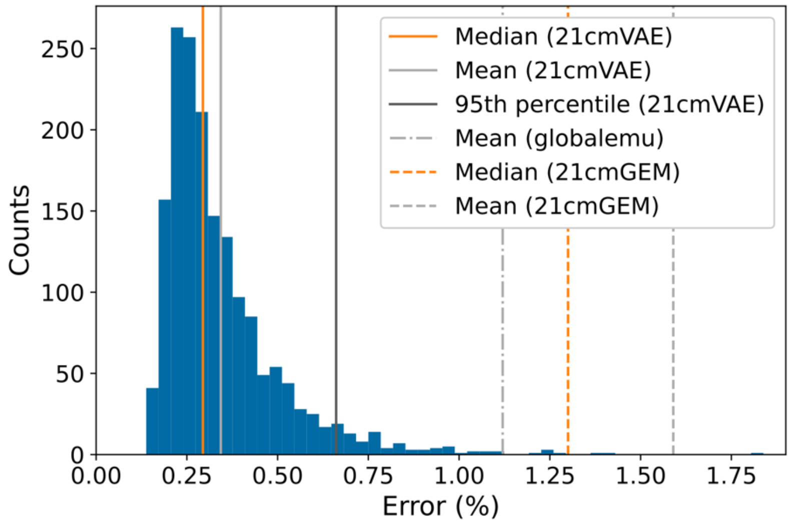
[1] *Department of Astronomy, University of California, Berkeley, CA 94720, USA*
[2] *Department of Physics, McGill University, Montréal, QC H3A 2T8, Canada*
[3] *DIRAC Institute, Department of Astronomy, University of Washington, 3910 15th Ave. NE, Seattle, WA 98195, USA*
[4] *Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge CB3 0HA, United Kingdom*
[5] *Kavli Institute for Cosmology, Madingley Road, Cambridge CB3 0HA, UK*

(Received Month Date, 2021; Revised Month Date, 2022; Accepted Month Date, 2022)

Submitted to ApJ

ABSTRACT

Considerable observational efforts are being dedicated to measuring the sky-averaged (global) 21-cm signal of neutral hydrogen from Cosmic Dawn and the Epoch of Reionization. Deriving observational constraints on the astrophysics of this era requires modeling tools that can quickly and accurately



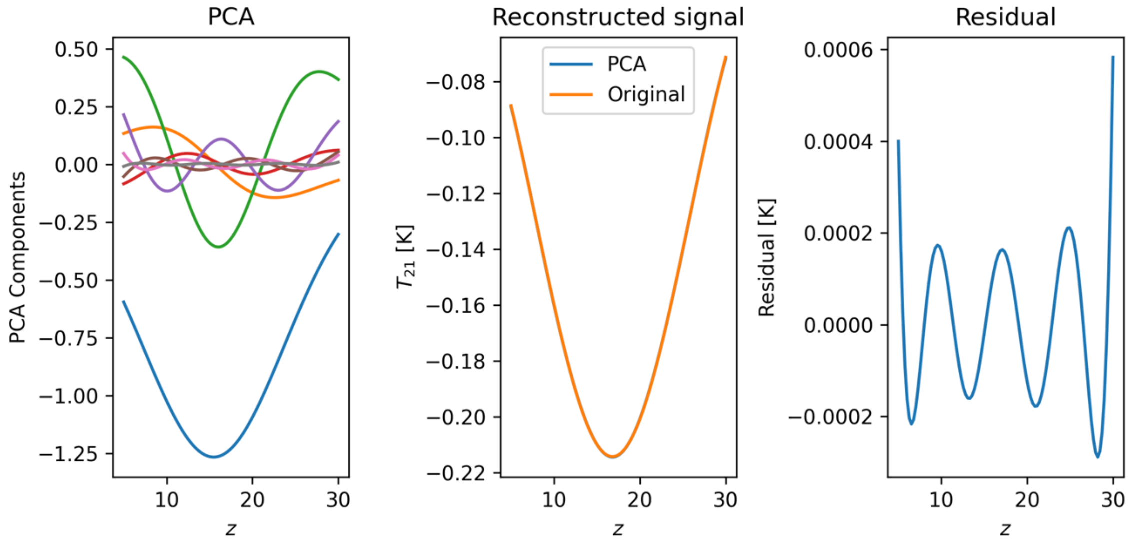
Dimensionality Reduction

Dimensionality Reduction

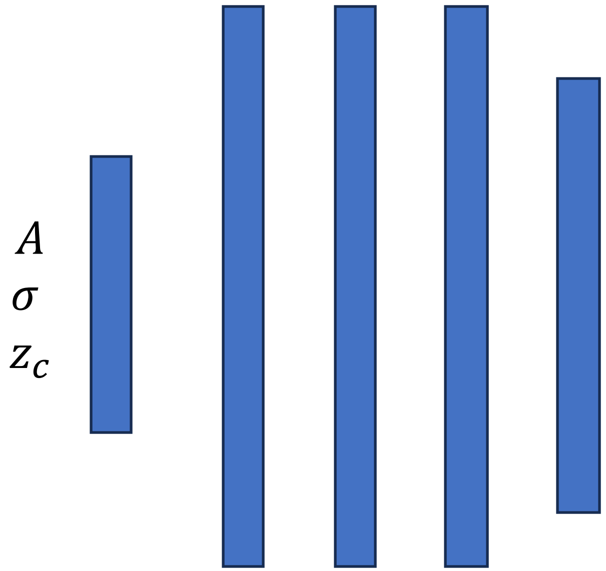
- The idea is to compress the data so that we can predict a few parameters
- And then reconstruct the signal from those parameters
- The issue is that these techniques are often result in information loss

Principle Component Analysis

- Decompose training data into eigenvectors and order based on eigenvalues
- Eigenvectors with largest eigenvalues describe the directions in the training data with the biggest variance
- Discard eigenvectors that correspond to low variance directions
- Reconstruct signals with weighted sum of most important eigenvectors where weights vary for each signal



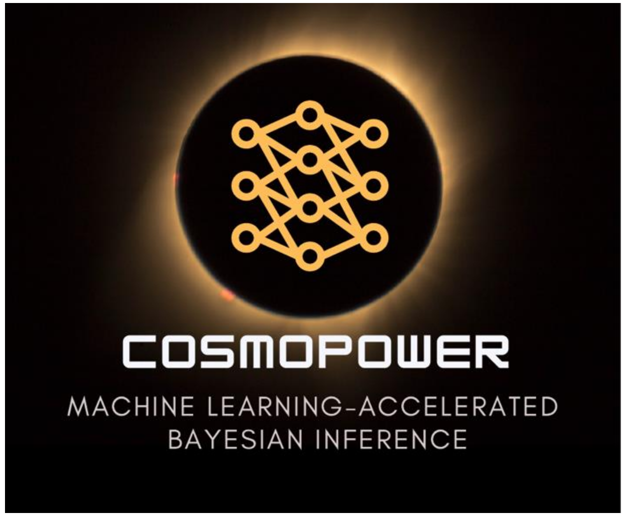
Principle Component Analysis

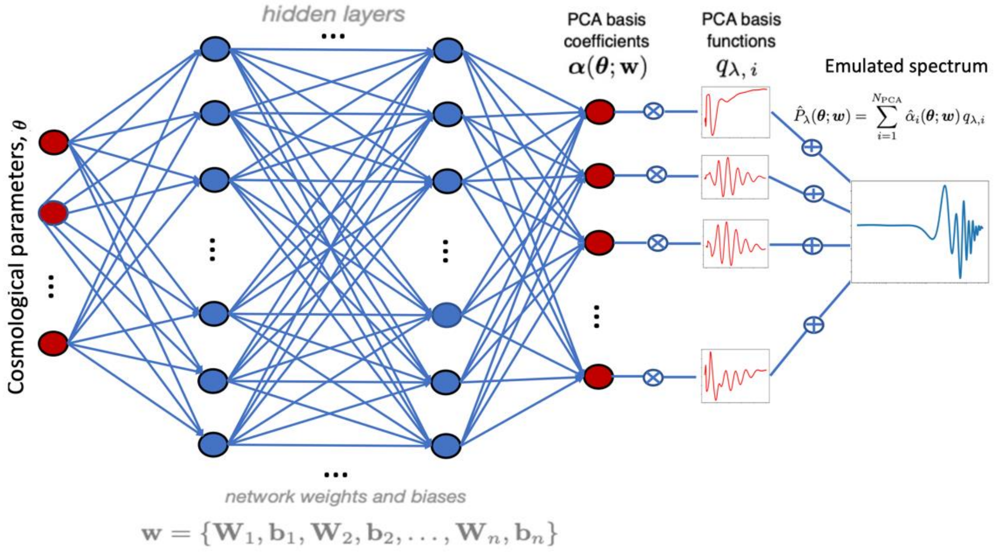


Signal Specific PCA coefficients $\times$ Important eigenvectors $\longrightarrow$ Sum terms $= T_{21}\,(z, \theta)$

Cosmopower

- Emulating the CMB power spectrum [2106.03846] as a function of cosmological parameters
- Very accurate and easily retrainable
- Uses Principle Component Analysis





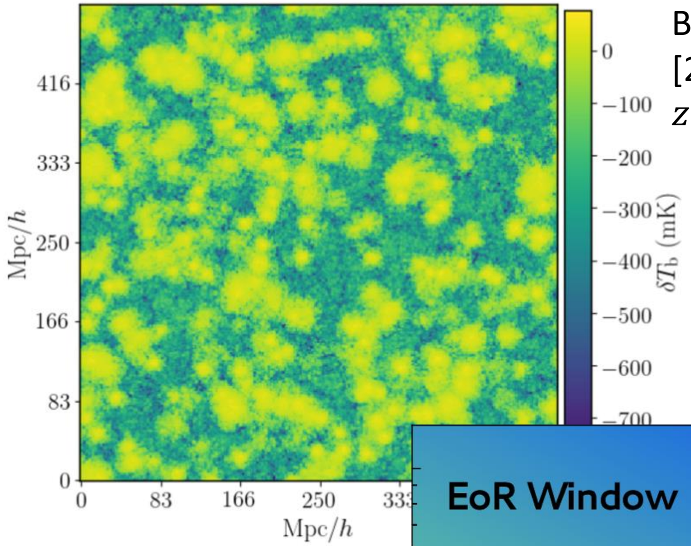
21-cm Power Spectrum with the SKA

The SKA and 21-cm Cosmology

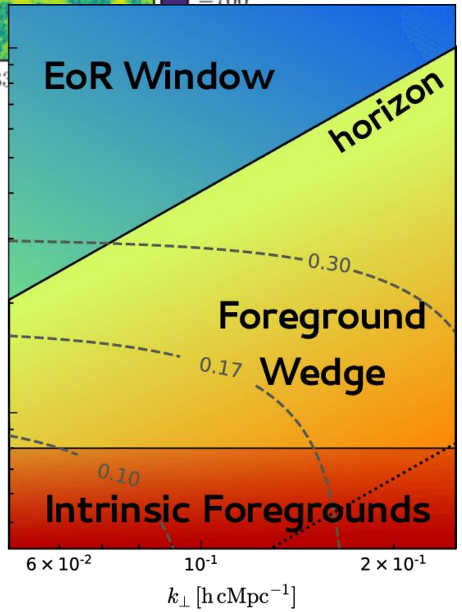
- The SKA will attempt to measure how the 21-cm signal varies spatially and temporally on the sky

- This is known as tomography

- Can extract science from tomography and summary statistics like the power spectrum

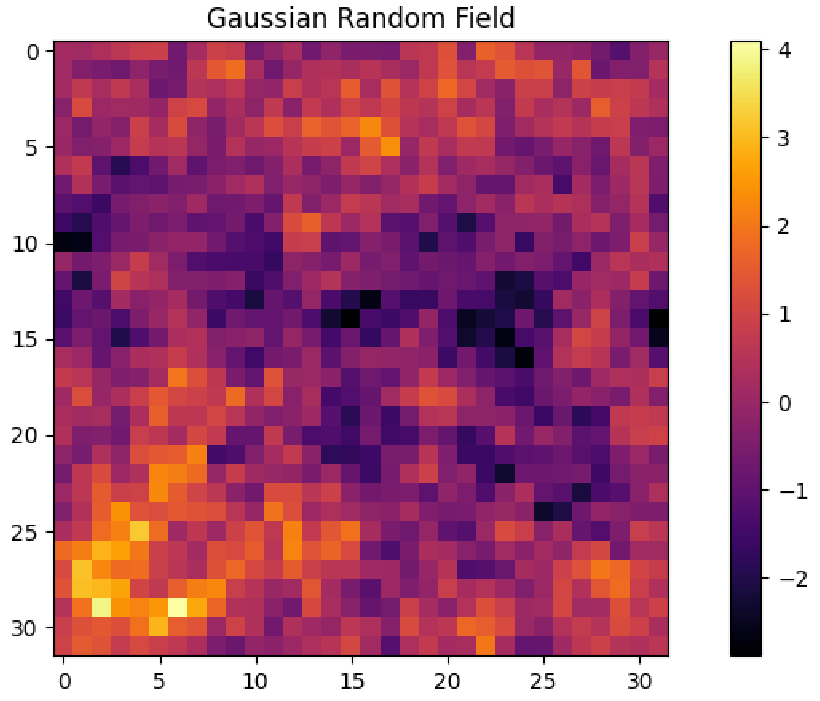


Bera et al. 2022 [2210.12164]
$z \approx 9$



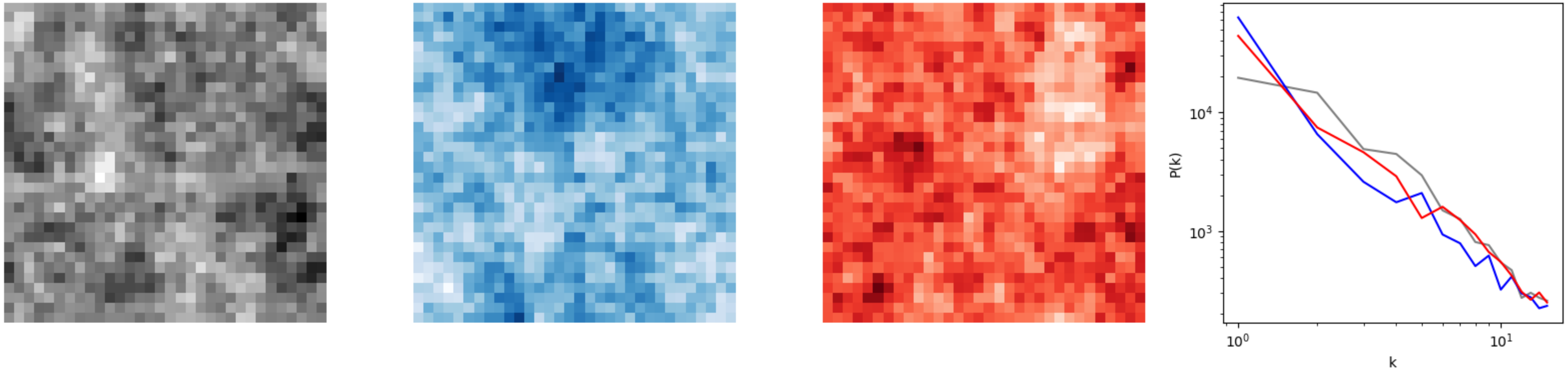
Random Fields and the 21cm signal

- The 21cm signal during the cosmic dawn follows the distribution of galaxies
- Distribution of galaxies is set by the initial density fluctuations in the early universe
- This is a gaussian random field with some correlated structure



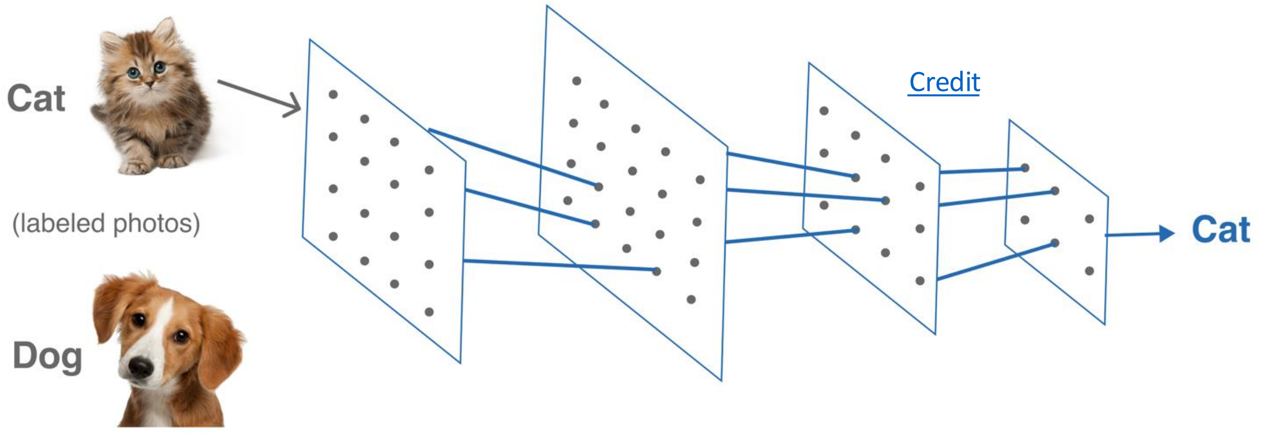
Random Fields and the 21cm signal

- The trick here is that you can have many different fields with the same properties
- They share a power spectrum that encodes the correlations
- That power spectrum and therefore field for 21-cm is dependent on astrophysics



How do we handle images?

- Typically use Convolutional Neural Networks
- Traditionally used for pattern recognition
- And subsequent classification of images
- Can also use normalising flows (next lecture) and diffusion models

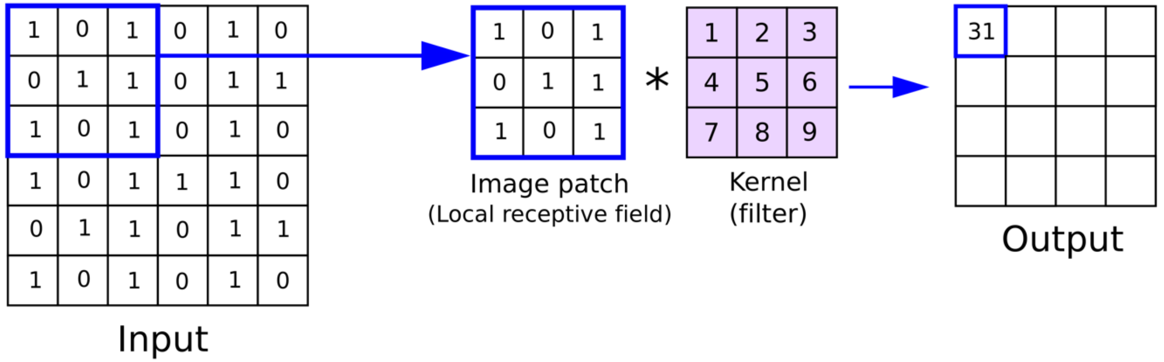


Convolutional Layers

- Convolutional layers take an image and slide a filter across the image performing a dot product as they go

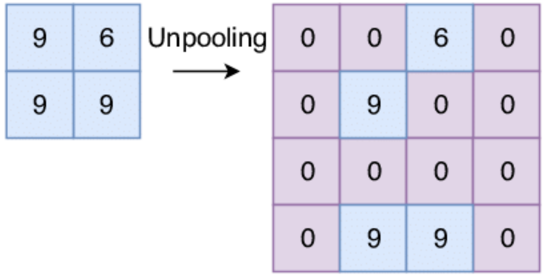
- Many filters are used to pick out key features gives you a stack of filtered images or a volume

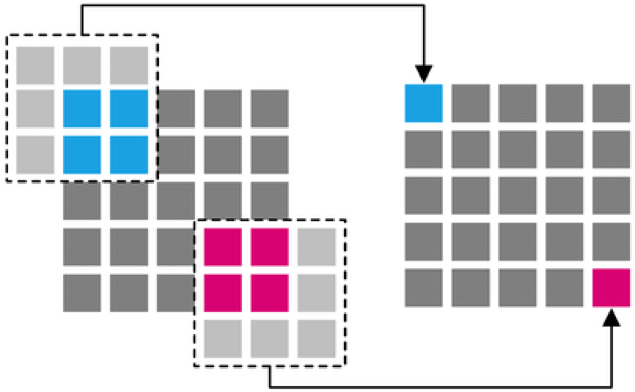
- Filters can be 2D or 3D in nature to increase or decrease the volume of the feature space at each layer



De-convolutional Layers

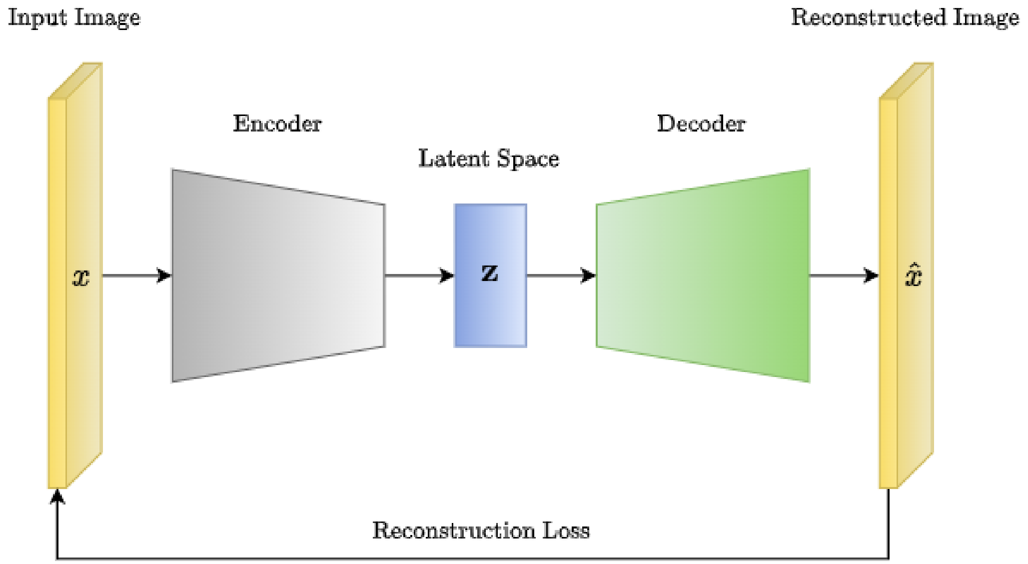
- The idea here is to up sample an image
- This is done by padding the image with zeros
- Then running a convolution over it with a kernel that returns the same shape as the padded image





Variational Autoencoders

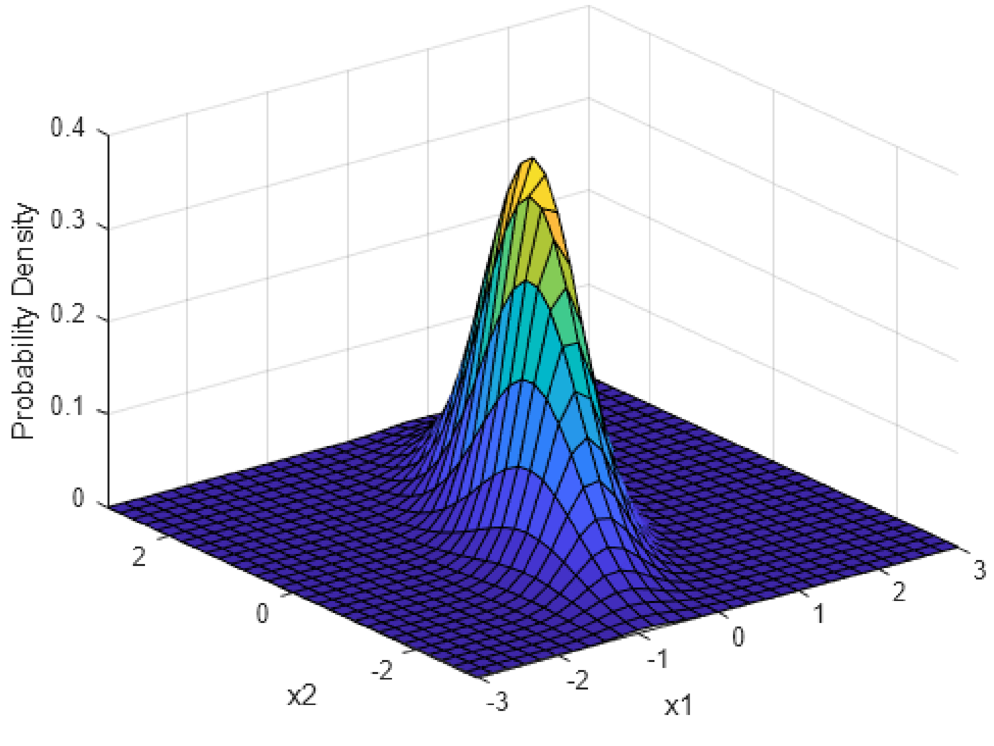
- We want a way to emulate the tomographic images with certain characteristics that captures the stochastic nature of the fields
- We can do this with variational autoencoders
- They are composed of two components an encoder and a decoder



UNIVERSITY OF CAMBRIDGE

The latent space

- The goal of the encoder is to compress the image into a latent space that closely resembles a known analytic distribution
- Typically use a standard normal distribution
- Learn a mean and variance conditioned on the parameters of the power spectrum



The encoder

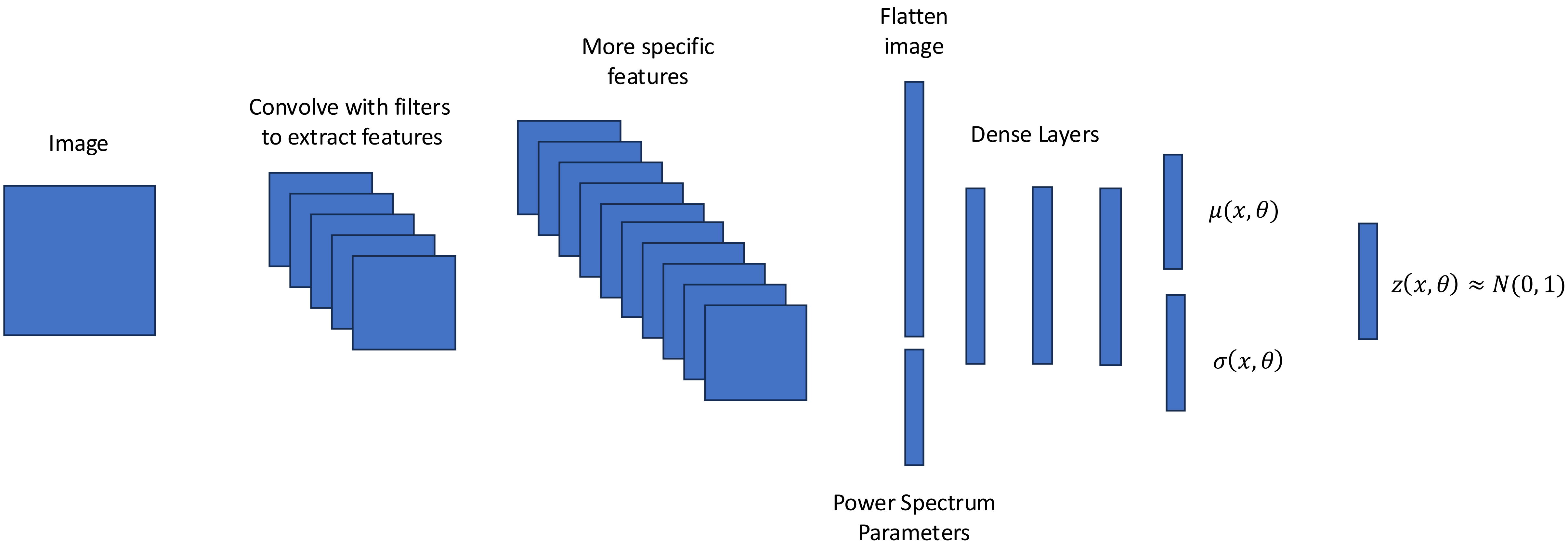
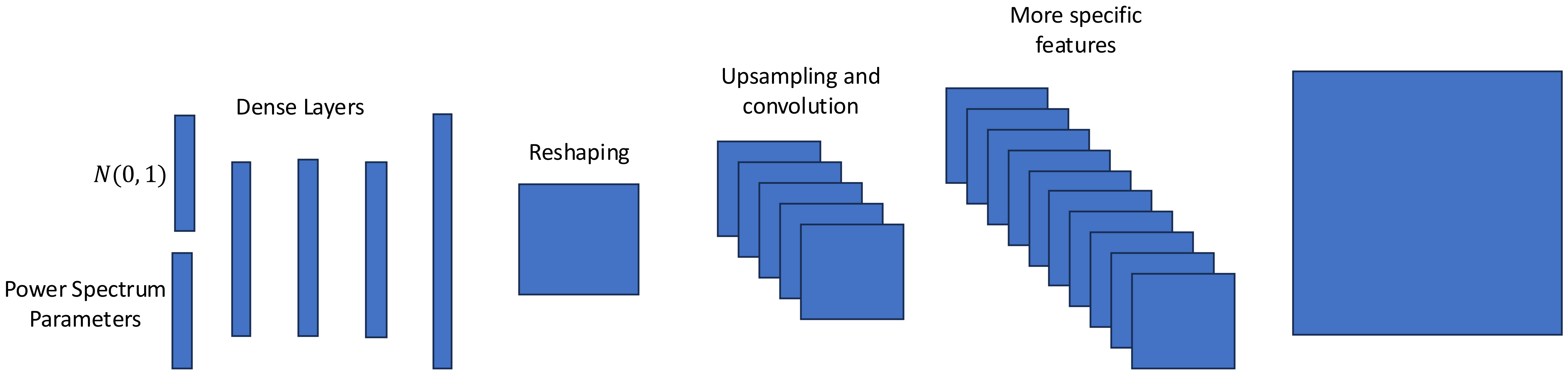


Image
Convolve with filters to extract features
More specific features
Flatten image
Dense Layers
$\mu(x,\theta)$
$\sigma(x,\theta)$
$z(x,\theta) \approx N(0,1)$
Power Spectrum Parameters

The decoder



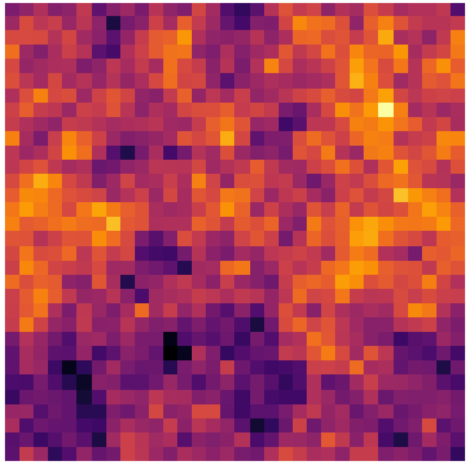
$N(0, 1)$
Power Spectrum Parameters
Dense Layers
Reshaping
Upsampling and convolution
More specific features

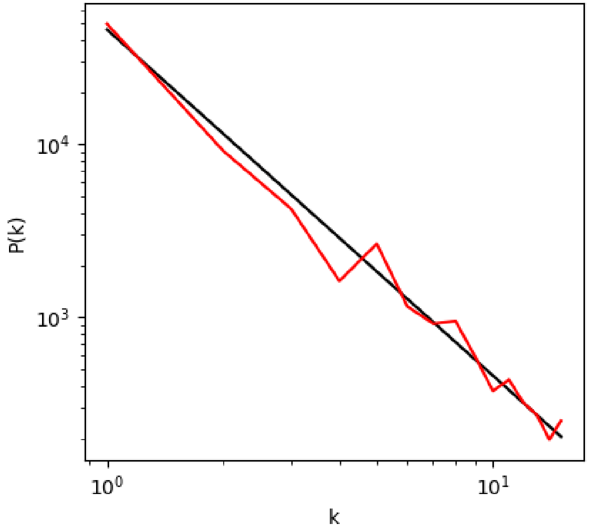
Toy example

- We can generate gaussian random fields with a given power spectrum equal to

$$P(k) = k^{-\beta}$$

- where β is a free parameter in the model



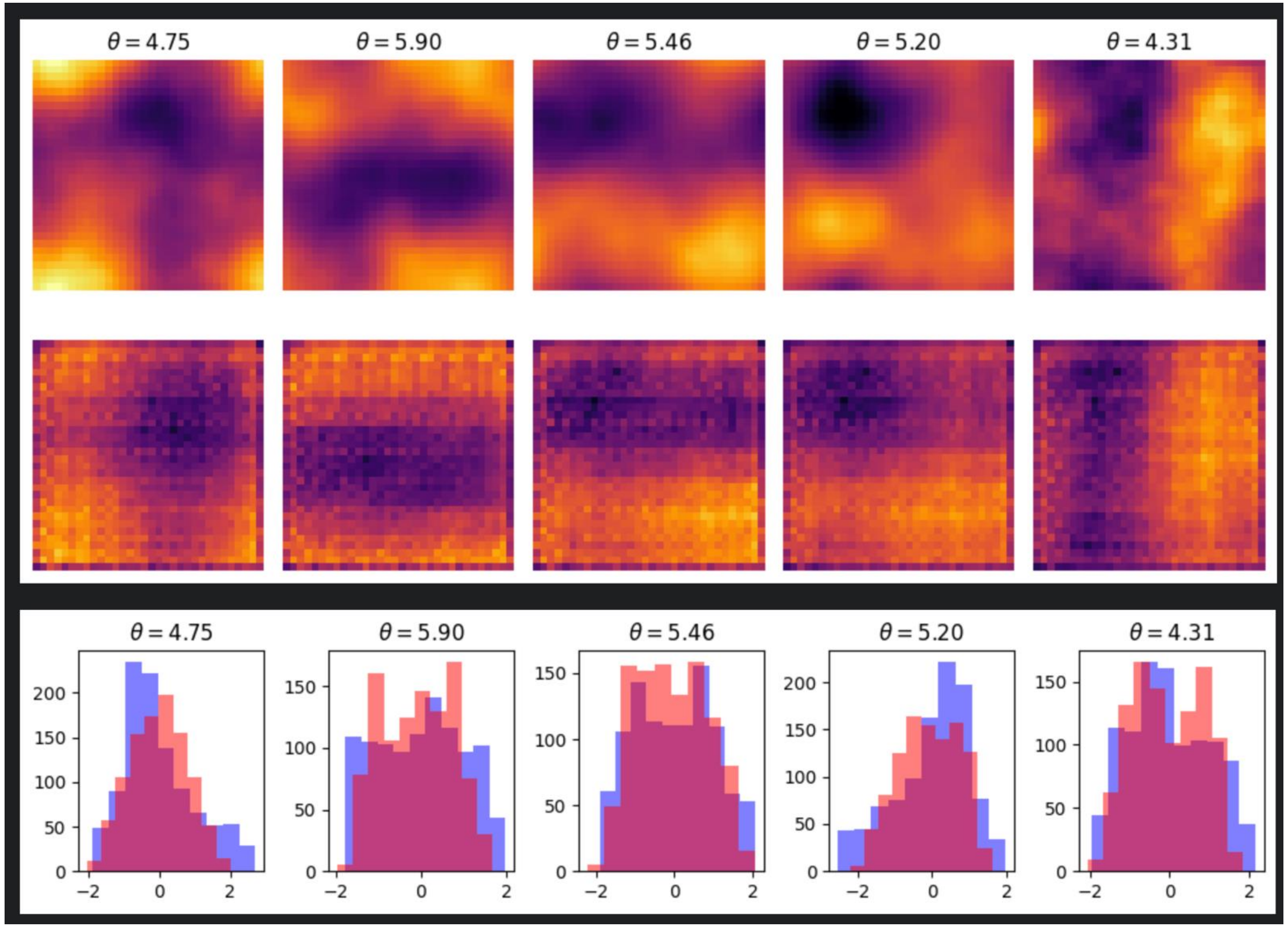


Training

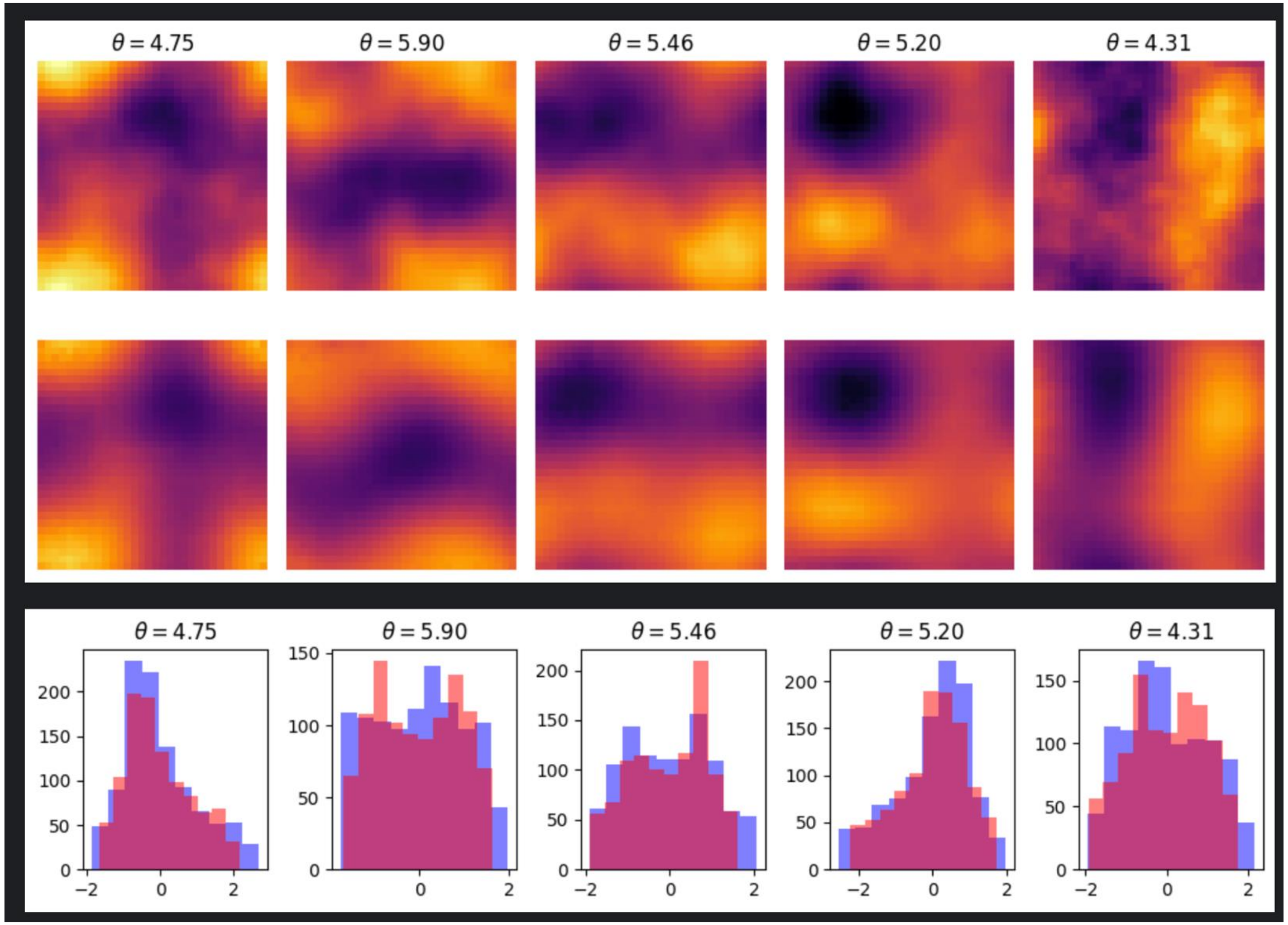
- The goal is to train the encoder-decoder pair so that
  - the latent space looks a lot like the standard normal
  - And the images are reconstructed well

$$L = \frac{1}{N} \sum_i (x - x_{Pred})^2 + KL(z' || z)$$

Example of training output

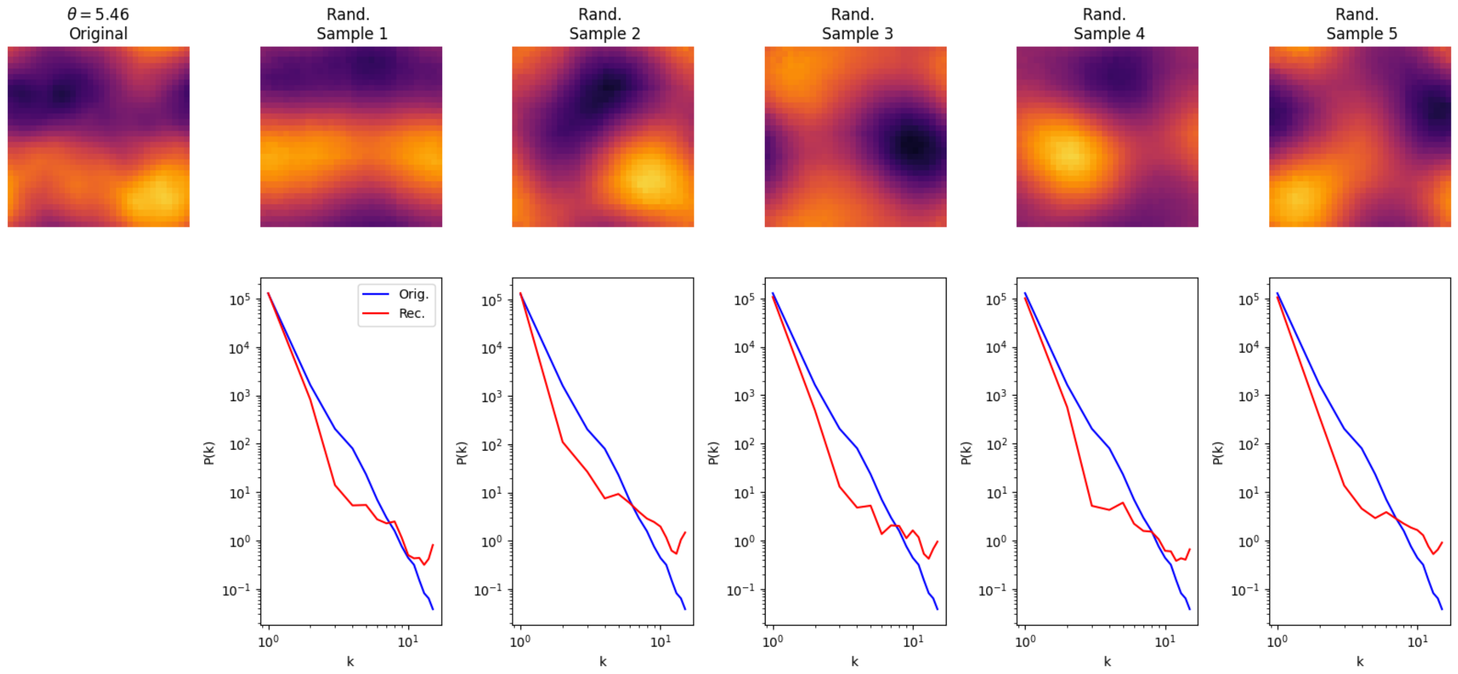


Example of training output



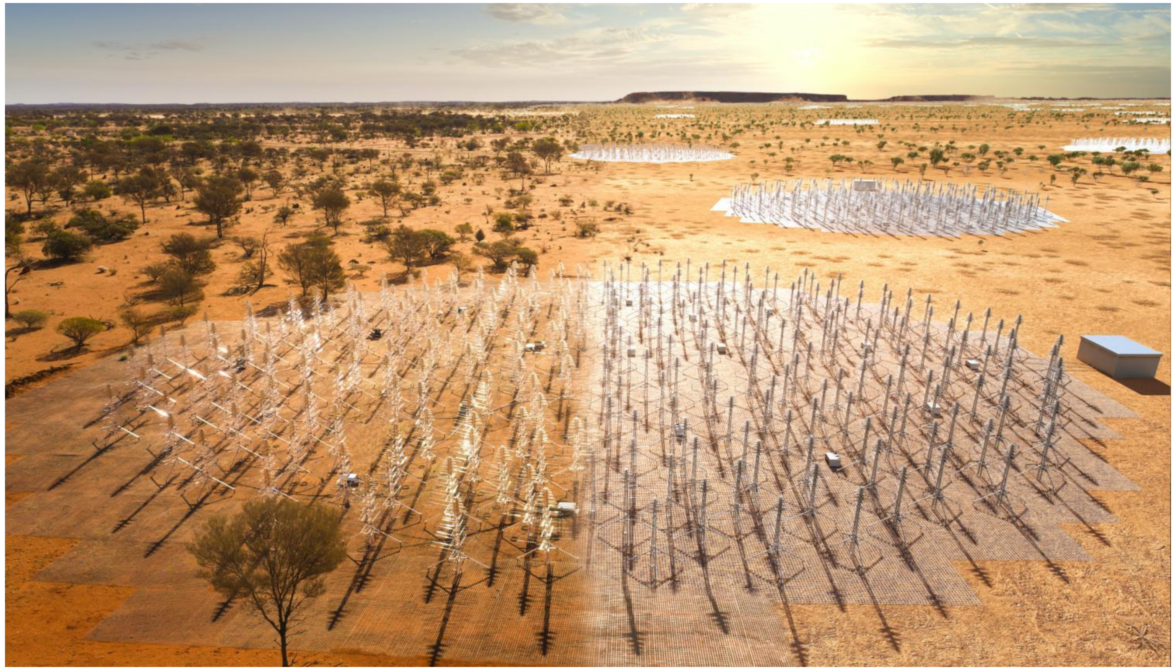
Generating new samples

- Once trained we can draw samples from the normal distribution
- Put them into the decoder with a value for the slope of the power spectrum and generate new realisaitons of the density field



For the SKA 21-cm Observations

- The likelihood is going to be on the power spectrum
- In practice we would likely just emulate the power spectrum directly
- But it's a nice example of how to work with images
- An example application using CNNs to directly emulate the 2D power spectrum can be found in 21cmEMU [2309.05697]



Summary

Emulators allow us to do inference

- Emulators are an efficient way to approximate complex semi-numerical simulations

- They take of order milliseconds to evaluate compared to hours per realization making inference possible

- Sensible choices about architectures, use of dimensionality reduction techniques, activation functions, loss funcitons etc can make a big difference to the run time and accuracy of emualtors

Many different ways to do this

- Host of different tools that can be used to build emulators
  - Here we have looked at Dense Neural Networks and Convolutional Neural Networks
  - But we can also do this with Normalizing Flows and Gaussian Processes for example

Slides available at https://github.com/htjb/Talks (and Moodle!)
Example codes on github too!

Next Lecture: Simulation Based Inference

- Why not just emulate the whole likelihood?