GENERATIVE ML

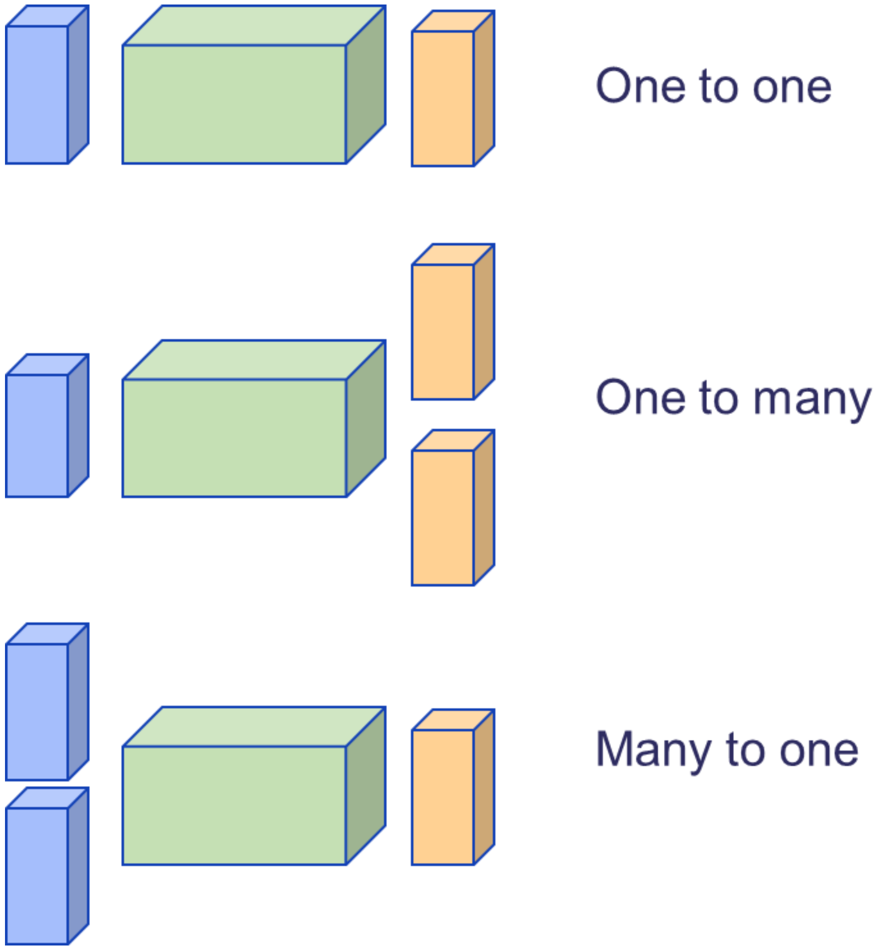
Keith Butler

GENERATIVE MODELS

Image-based models
Variational Autoencoder
Generative Adversarial Network
Diffusion models

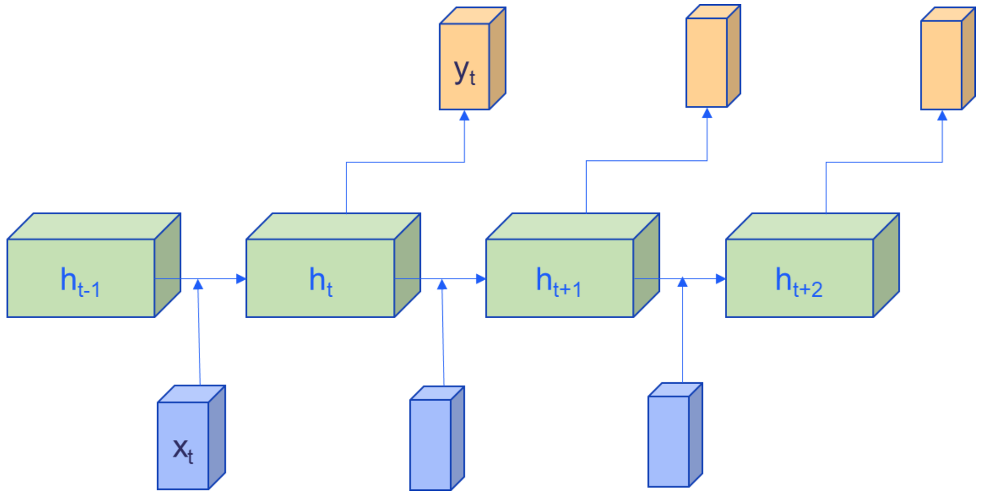
Language based models
Recurrent neural networks
Long-short term memory models
Transformers/LLMs

SEQUENCE DATA



MLPs and CNNs struggle with sequence data. There is no fixed length of input, the data can be many to one.

RECURRENT NEURAL NETWORKS (RNNS)



$$h_t = f_W(W_{hh}h_{t-1}, W_{xh}x_t)$$

ISSUES WITH RNNS

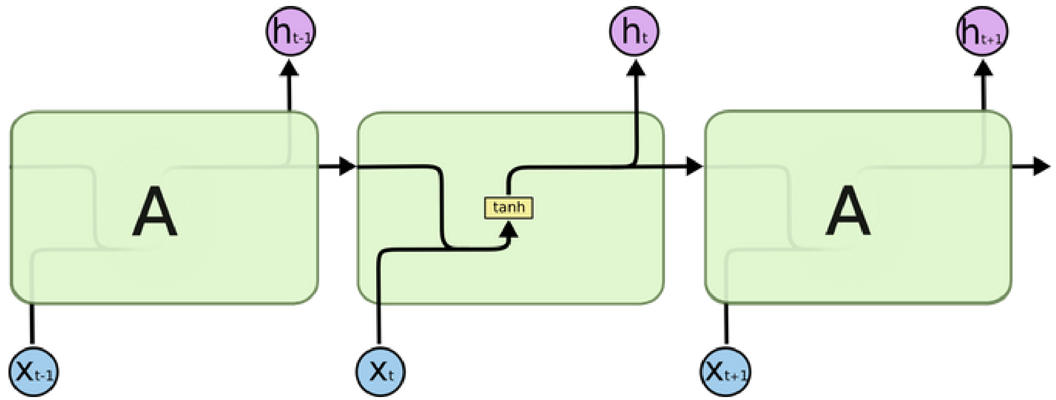
RNNs have very short term memory – they lose context quickly

```
The clouds are in the _________. ✓
```

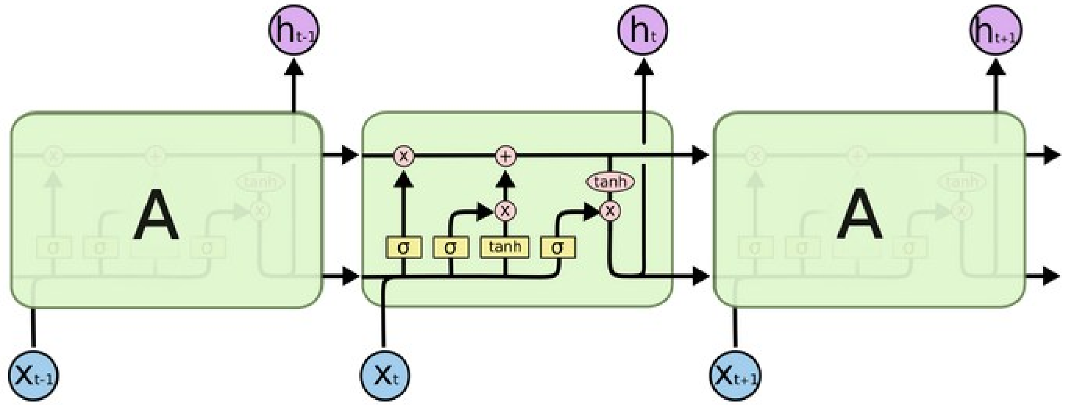
```
I was born in France. At the
age of 16 I moved country. I
have lived here since I was
21. Nonetheless, I still speak
fluent ________. ✗
```

INTRODUCING MEMORY

RNN



LSTM

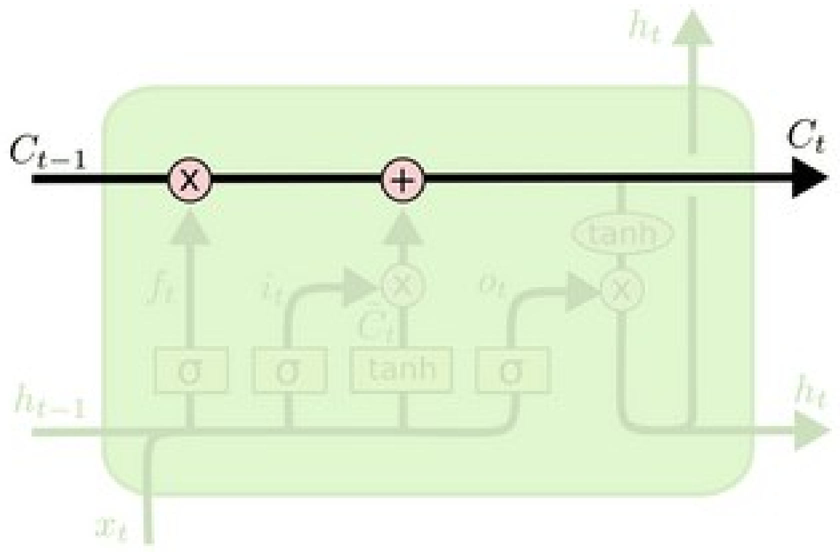


Long short term memory (LSTM) networks introduce extra memory features compared to a standard RNN

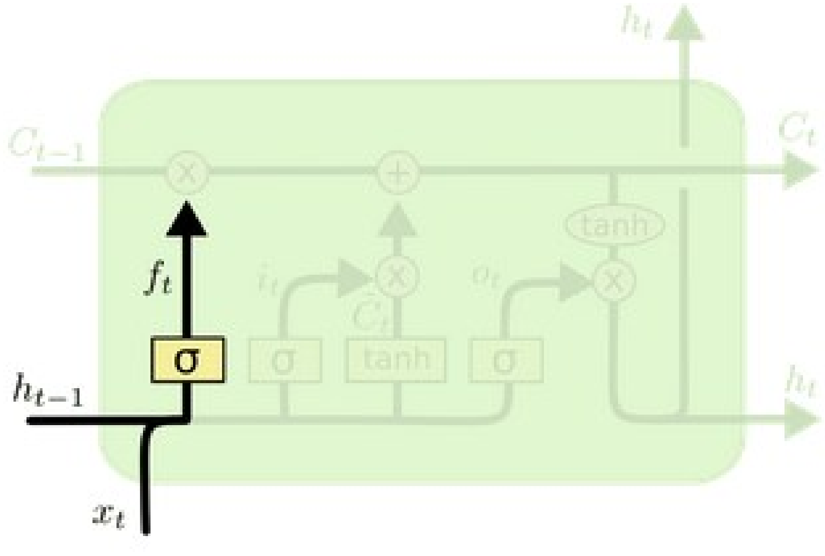
LSTM – THE MEMORY STATE

A single channel that runs all the way along the sequence structure

Only has some minor interactions with the rest of the network

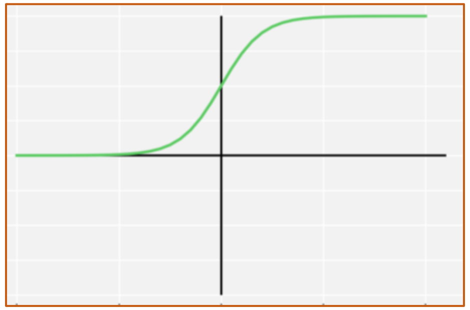


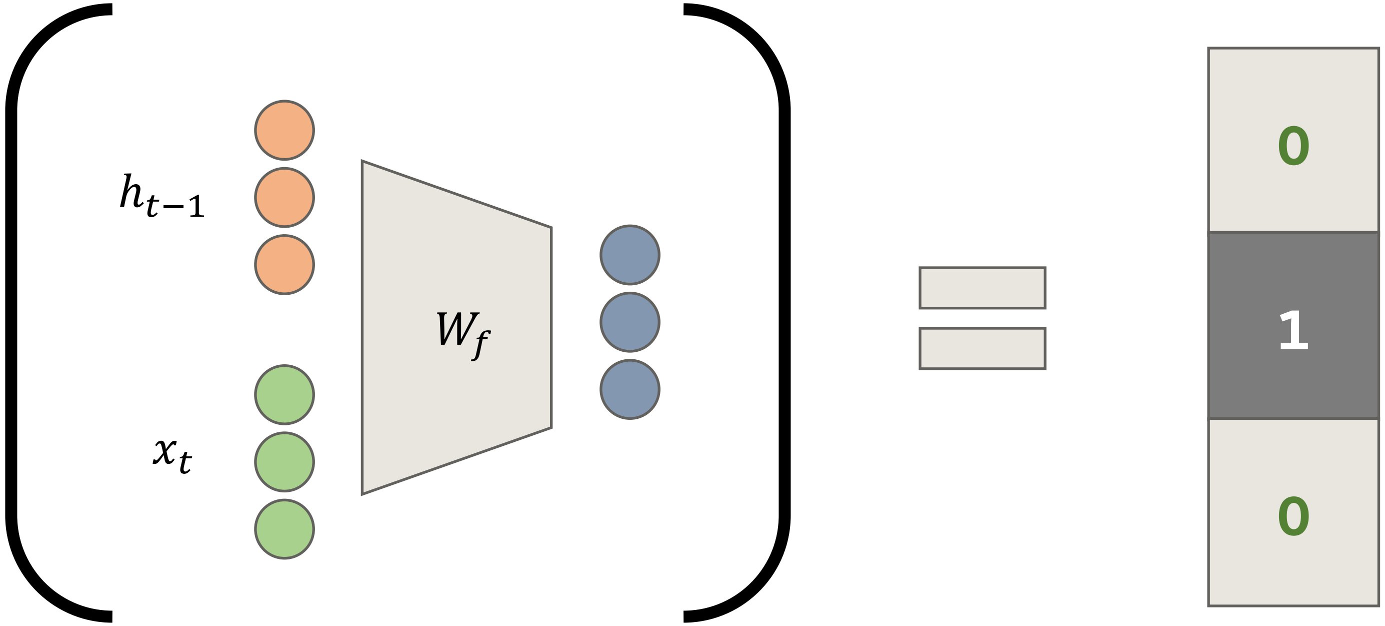
LSTM – FORGETTING



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

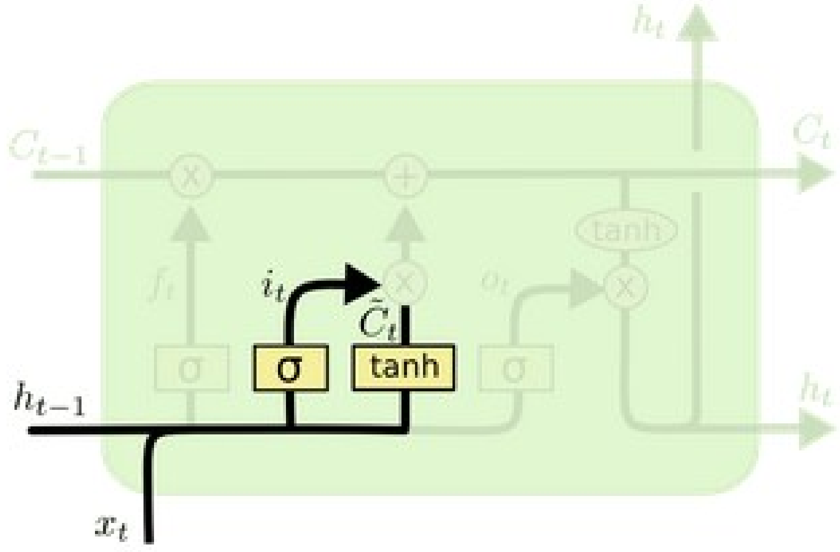
FORGETTING





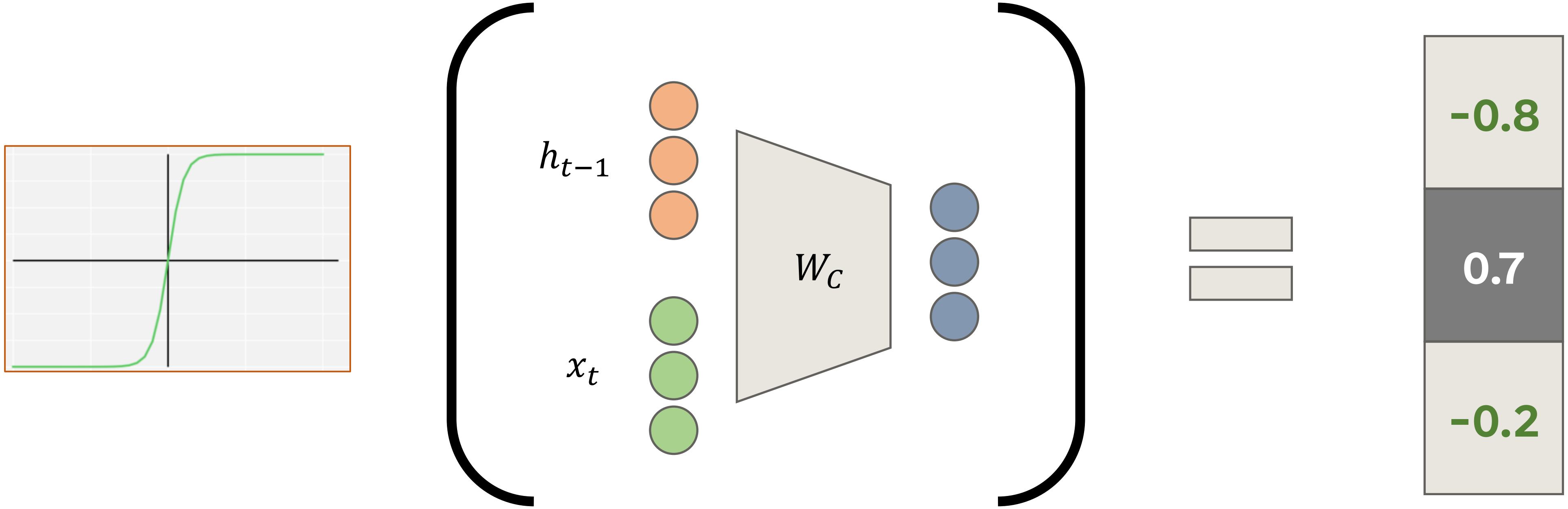
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

LSTMS - ADDING



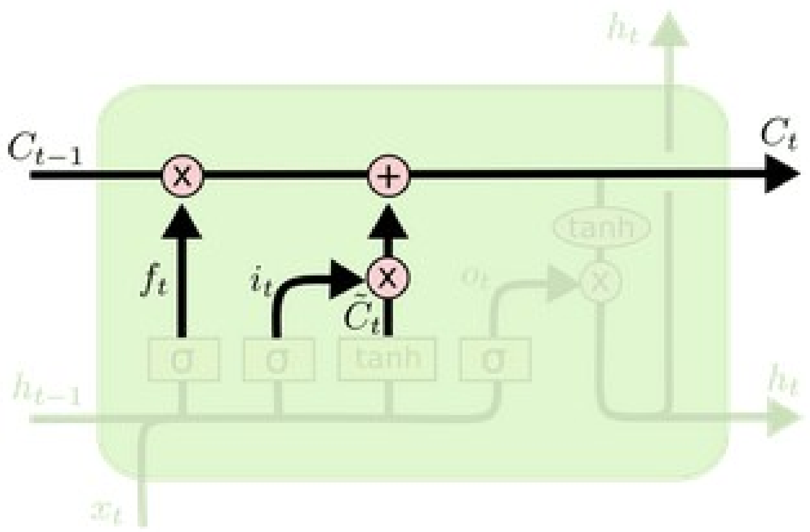
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

ADDING



$$\tilde{C}_t = tanh\,(W_C \cdot [h_{t-1}, x_t] + b_C)$$

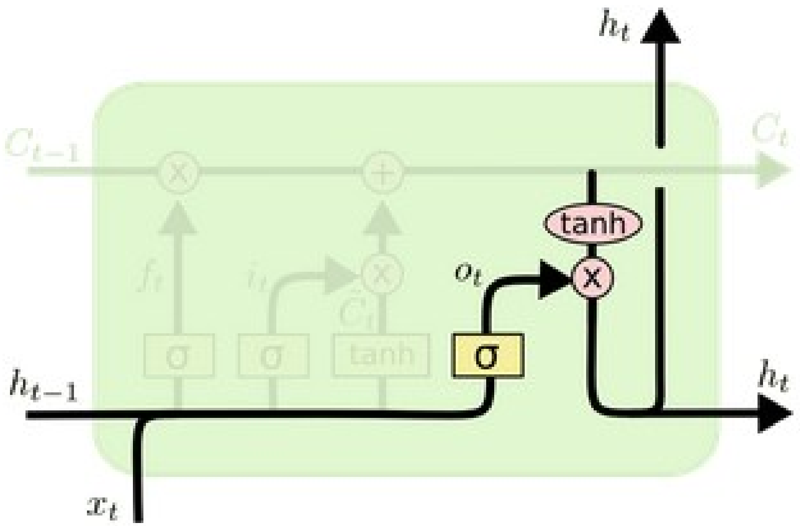
LSTM – UPDATING THE MEMORY



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM – GENERATE OUTPUT

student



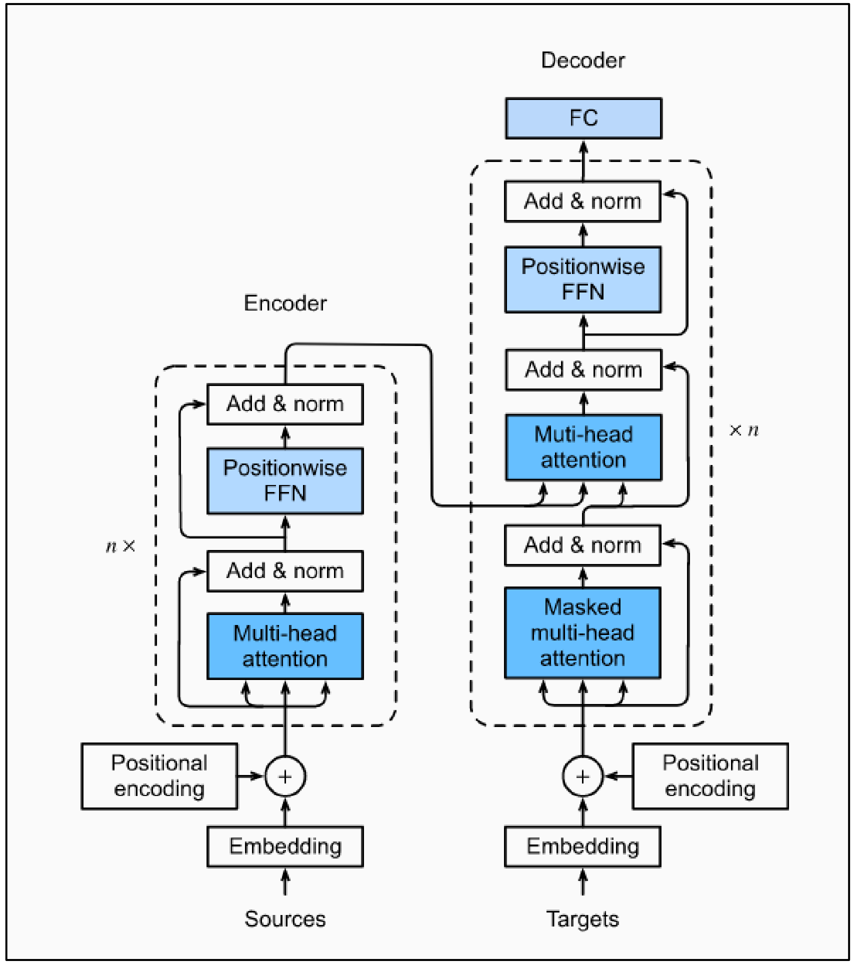
Je suis etudiant

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

TRANSFORMERS

student



Je suis etudiant I am a

THE ATTENTION MECHANISM

High attention

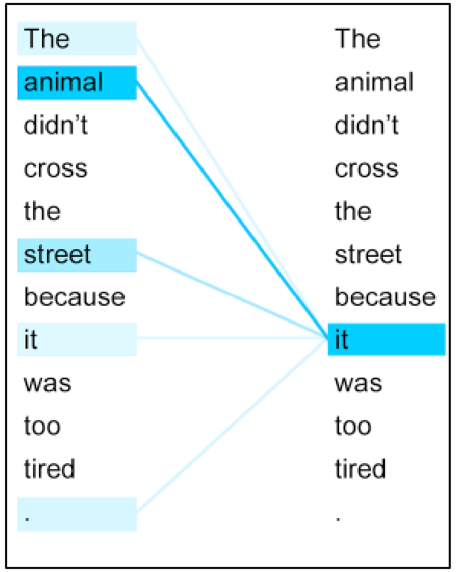
Low attention

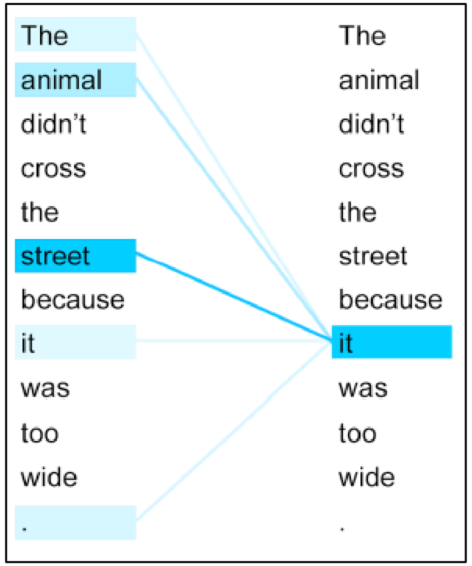
She is eating a green apple.

We use attention to "focus" on some part of interest in an input

SELF-ATTENTION

With self-attention, each token $\boldsymbol{t}_n$ can “attend to” all other tokens of the same sequence when computing this token’s embedding $\boldsymbol{x}_n$





HOW SELF-ATTENTION WORKS

Attention Is All You Need

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Based of the concept of query, key and value vectors

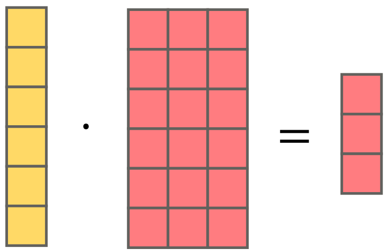
arXiv:1706.03762 (2017)

ATTENTION – THE INPUTS

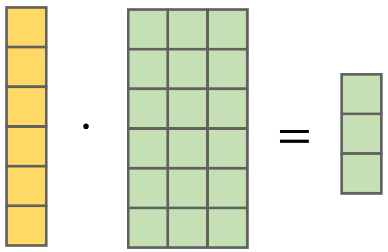
Each token has a d-dimensional representation

Each token also has a query and key vector q-dimensional; q << d

W_K is a matrix of learnable weights



$$\overrightarrow{x_i} \cdot W_Q = \overrightarrow{Q_i}$$



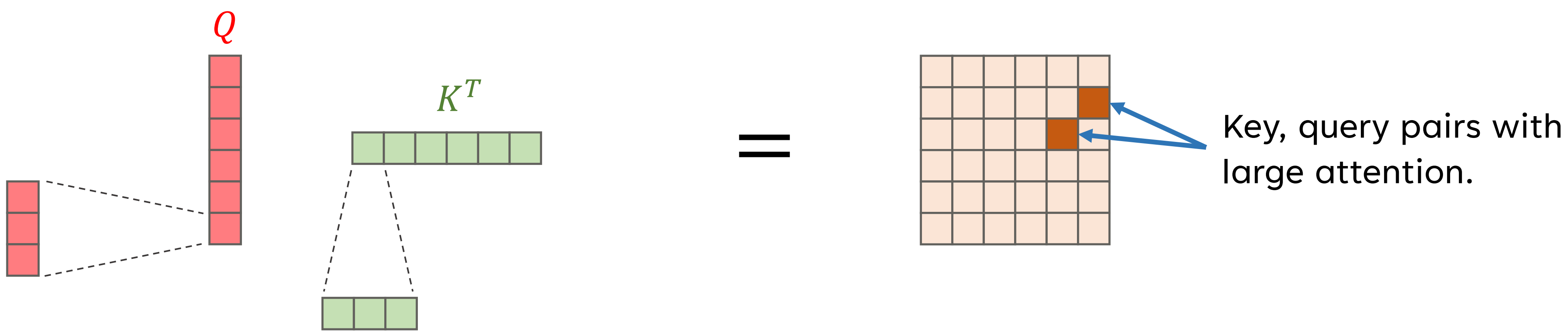
$$\overrightarrow{x_i} \cdot W_K = \vec{K}_i$$

In bed an old woman lies.

QUERY KEY MULTIPLICATION

$$softmax\left(\frac{QK^T}{\sqrt{d}}\right)$$

In bed an old woman lies.



Softmax normalises the columns; root of d makes it numerically stable

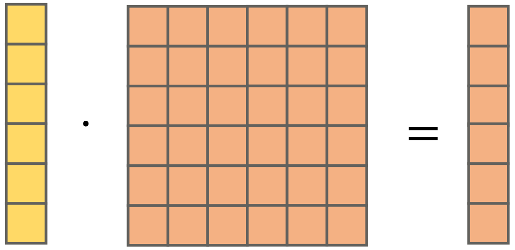
NOTE –in this case each cube in Q and K has 3 dimensions

THE VALUE MATRIX

Tells you how a given token modifies another token

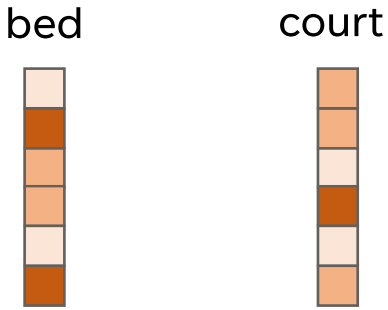
The resultant $\vec{V}$ gets added to the other vector

The extent of the addition is scaled by the QK^T product

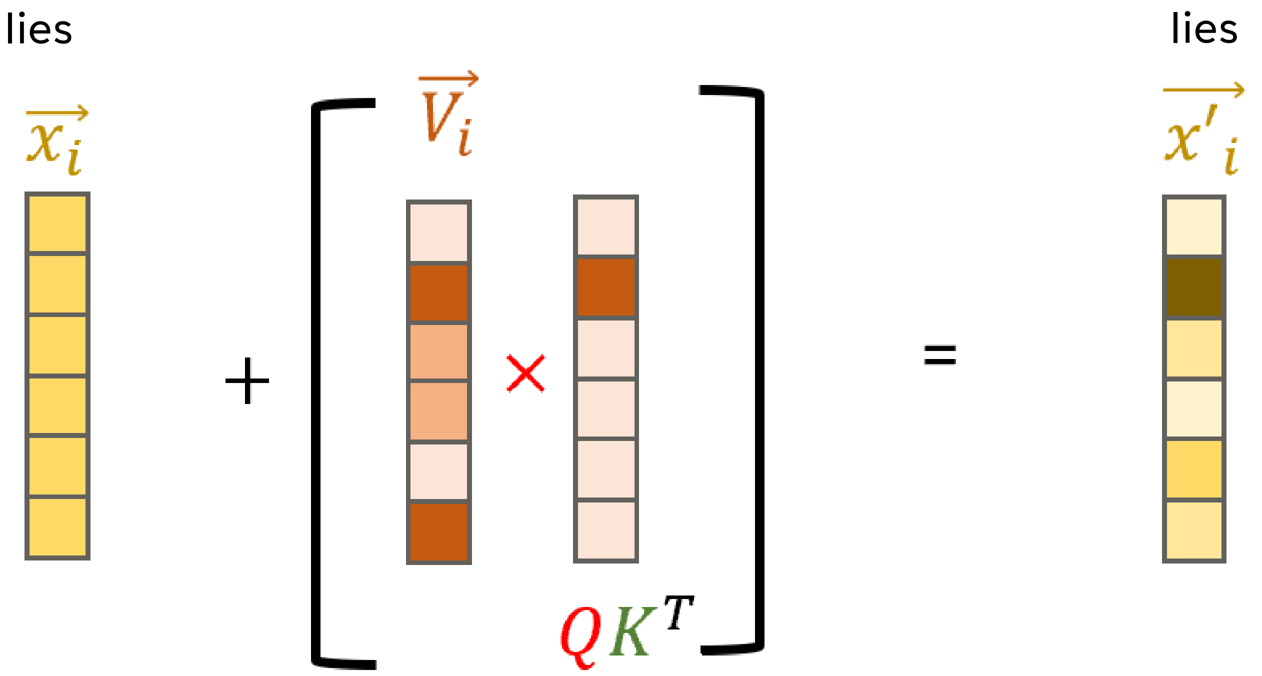


$$\vec{x_i} \cdot W_V = \vec{V_i}$$

In ---- an old woman lies.

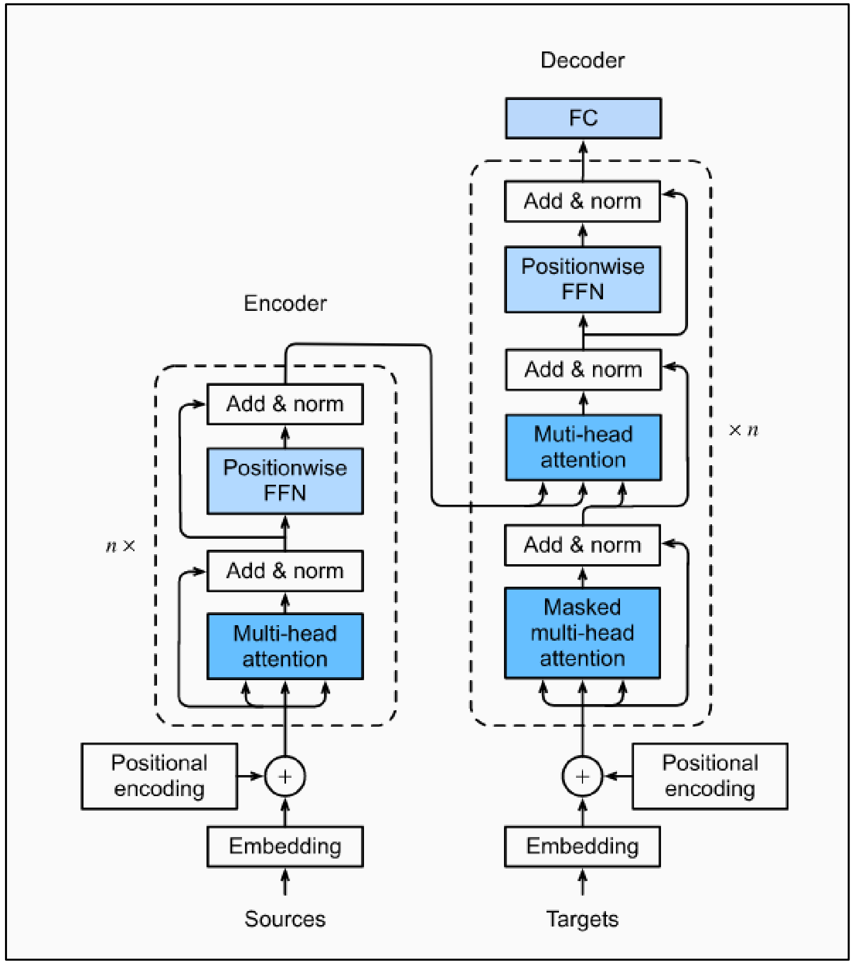


ADDING THE VALUE TO THE EMBEDDING



The value is modified by the attention from the QK pair and added to the initial embedding

UPDATING THE EMBEDDINGS

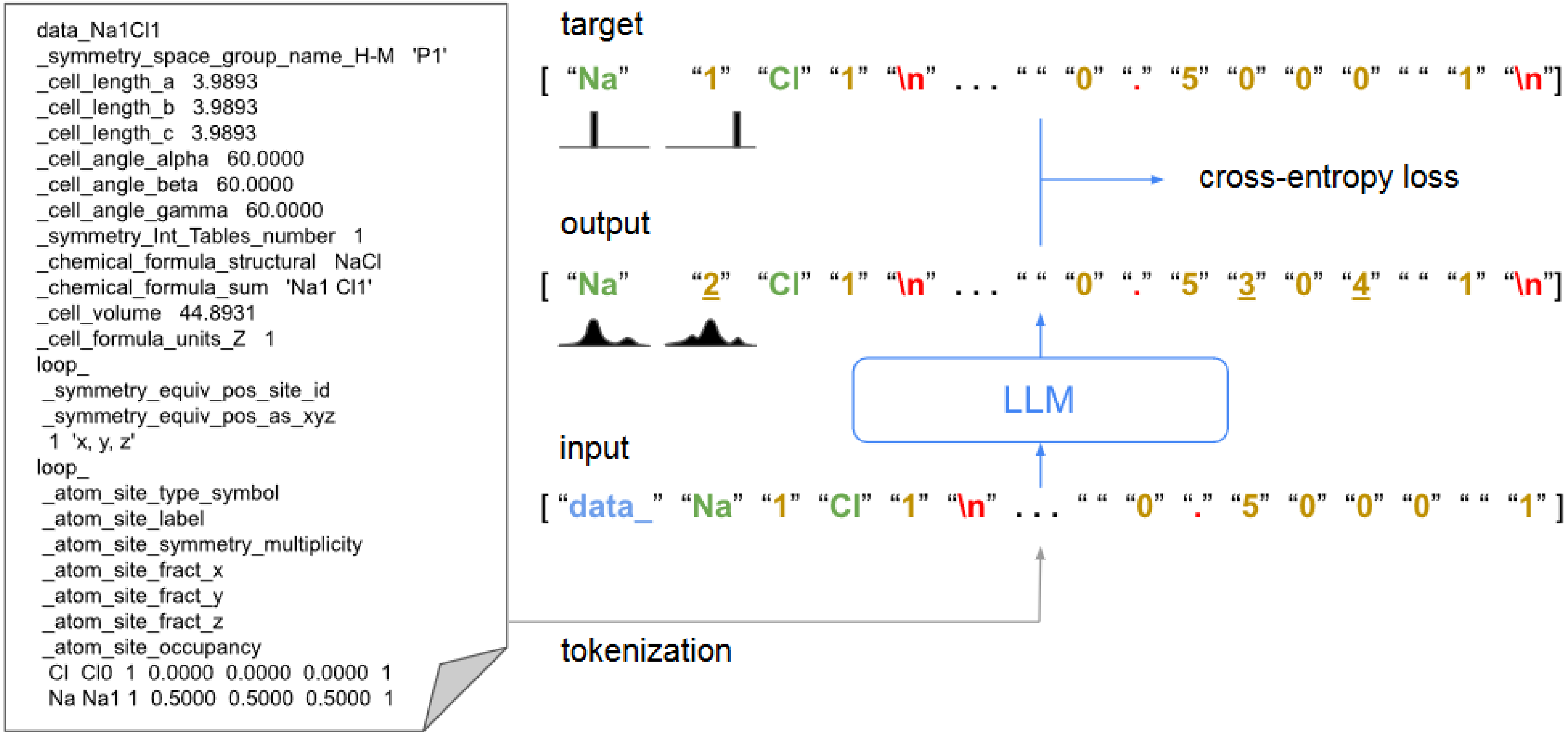


$$E \longrightarrow \boxed{softmax\left(\frac{QK^T}{\sqrt{d}}\right)V} \longrightarrow E'$$

Each of these softmax matrix multiplications is a ‘head’

GO TO NOTEBOOK

CRYSTALLM

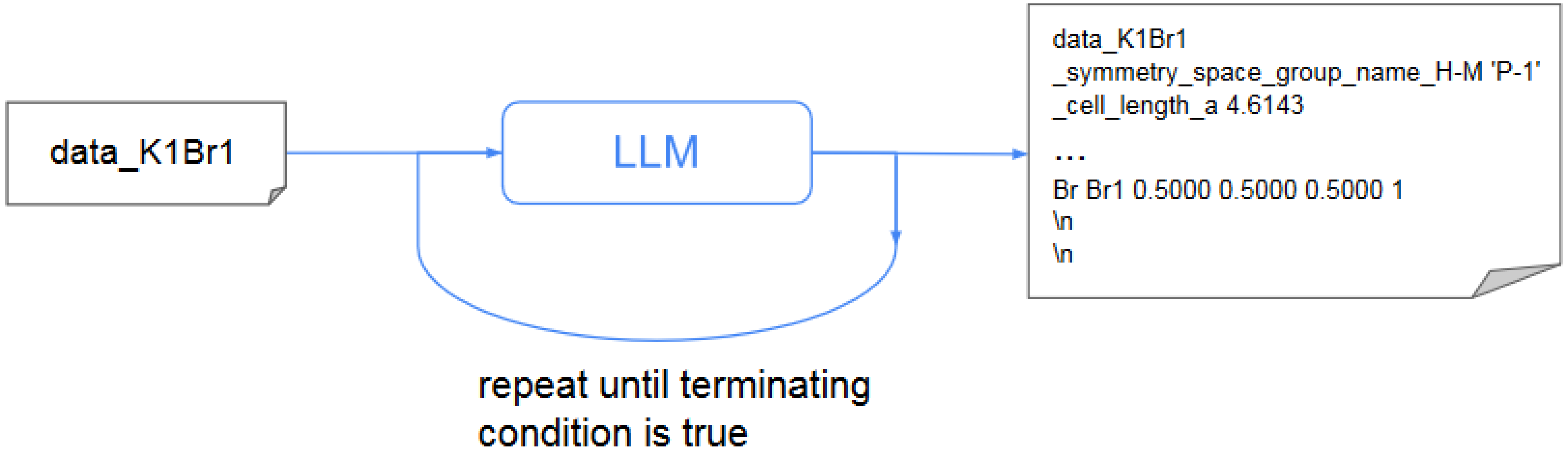


A decoder only transformer trained on cif files for materials structure generation

Nature Communications 15, 1 (2024)

AUTOREGRESSIVE GENERATION

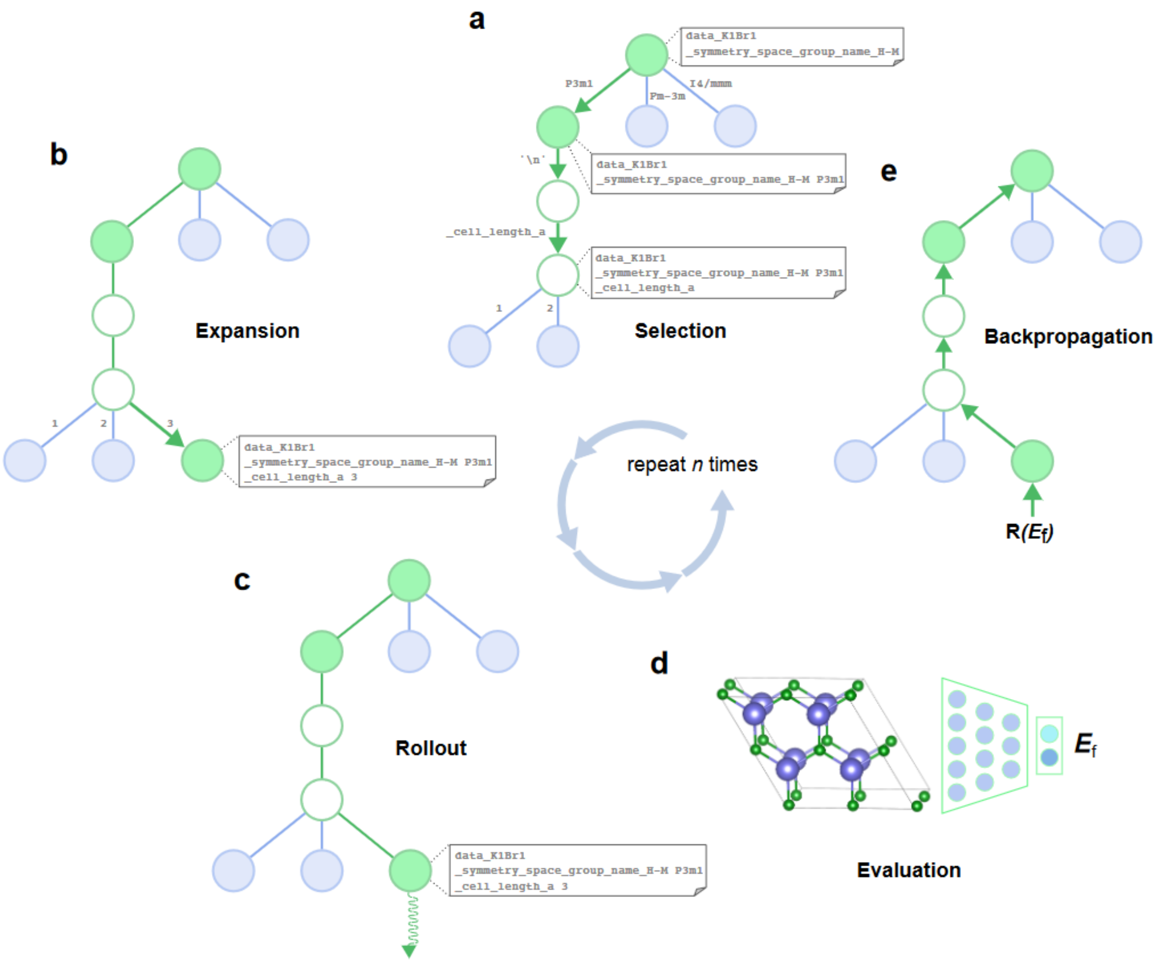
After training, CrystaLLM can be prompted with new text and can produce a predicted complete cif



Prompting is flexible so we can provide as little or as much information as we like

Nature Communications 15, 1 (2024)

MONTE CARLO TREE SEARCH FOR CONSISTENCY



Autoregressive generation is stochastic and can lead to non-ideal structures

MCTS is more expensive but uses an energy estimator to drive to low energy solutions

Nature Communications 15, 1 (2024)

CONCEPT CHECKLIST

Sequential data benefits from contextual awareness and memory

Recurrent networks are an early answer

Memory was improved with LSTMs

Transformers use attention to map across sequences

Autoregression can be applied to generate crystal structures

DISTRIBUTION BASED GENERATIVE MODELS

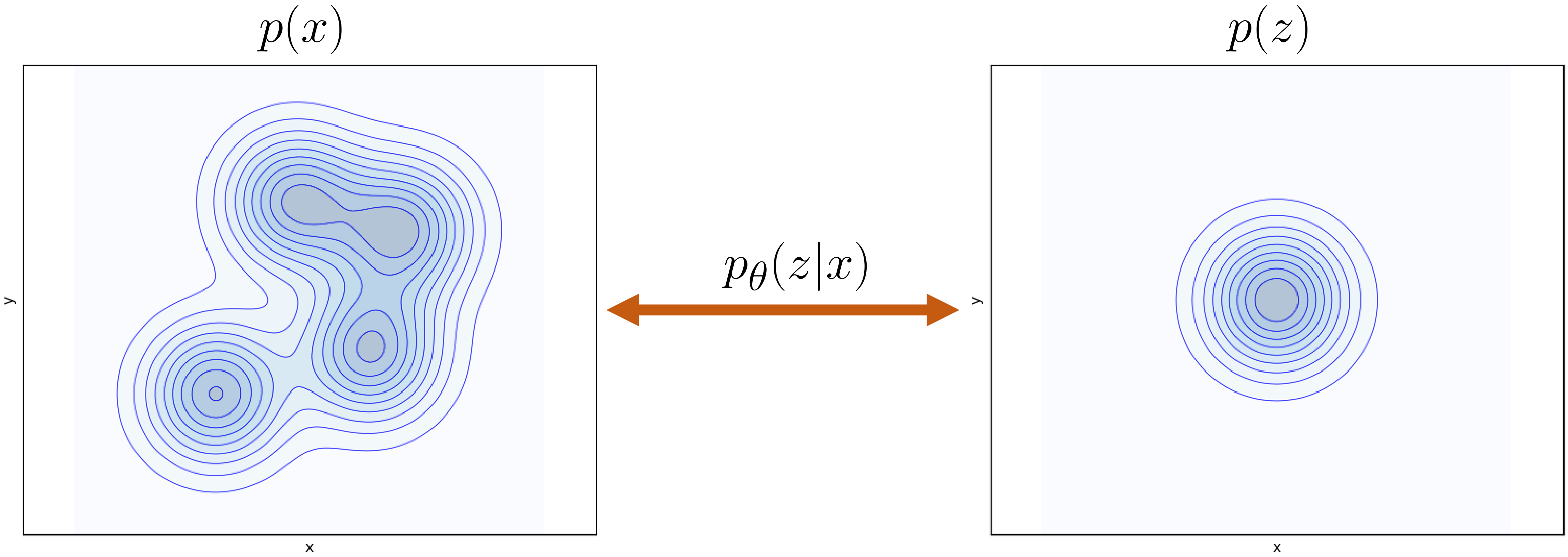
Image-based models

Variational Autoencoder

Generative Adversarial Network

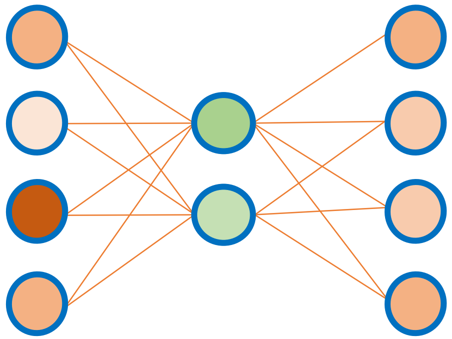
Diffusion models

MAPPING BETWEEN DISTRIBUTIONS



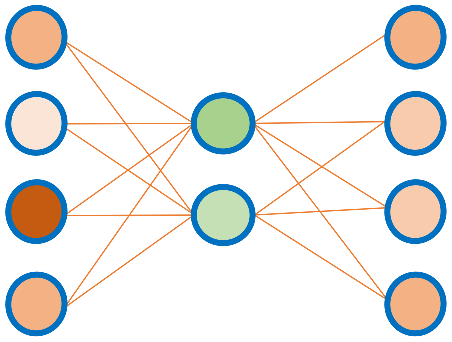
AUTOENCODERS

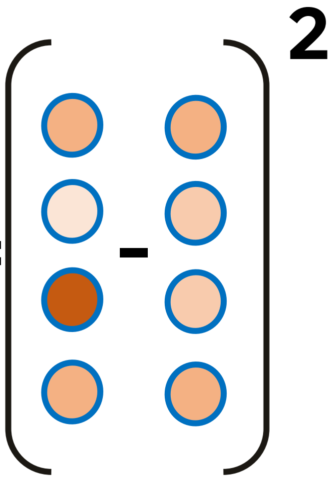
Unsupervised/self-supervised learning: The model does not require labels to learn about the data distribution



AUTOENCODERS

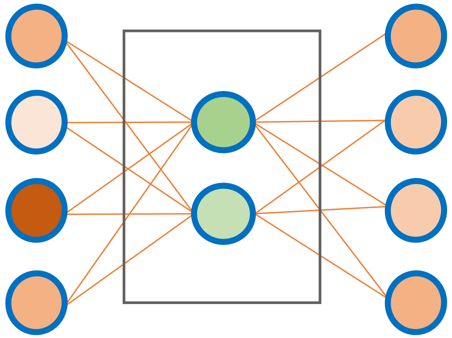
Training objective: To minimise the difference between the input and the output



Loss(L) = 

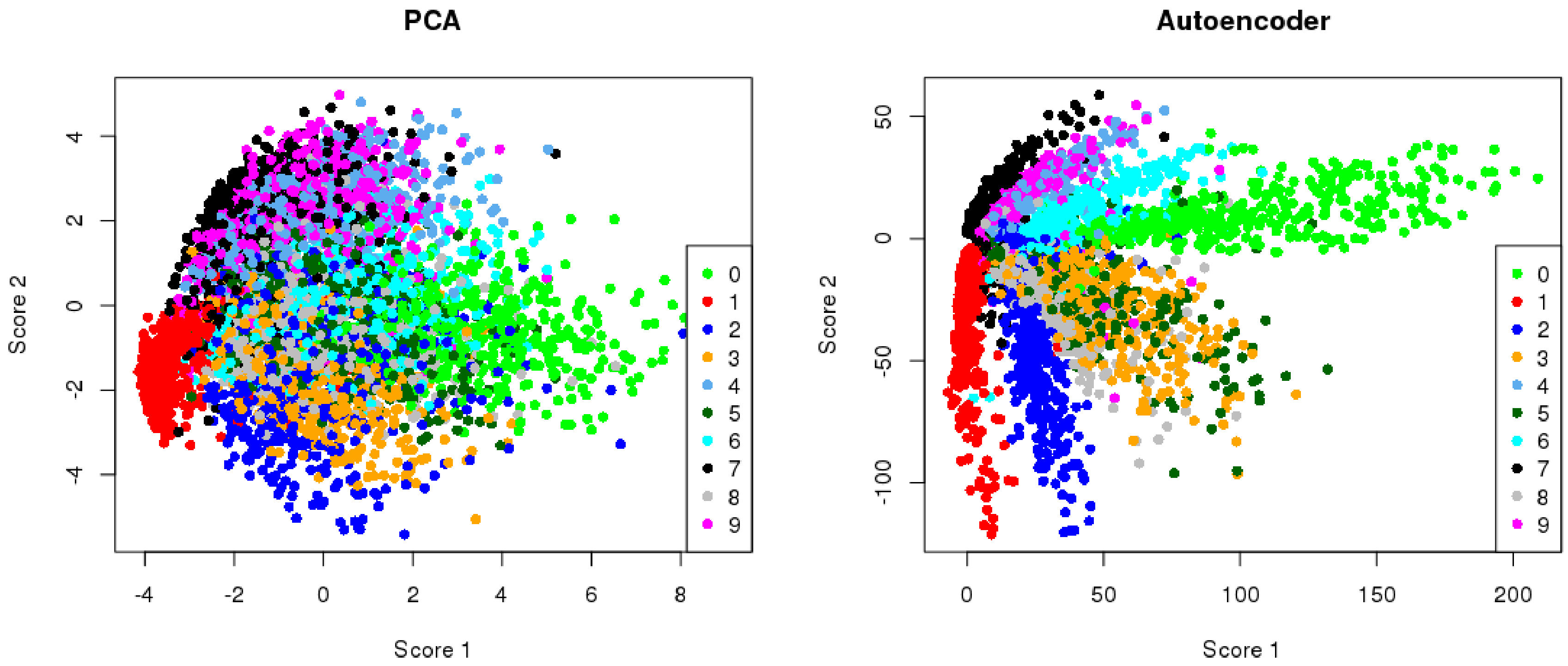
AUTOENCODERS

Outcome: A model that can compress data into lower dimensions with minimal information loss – the latent space.



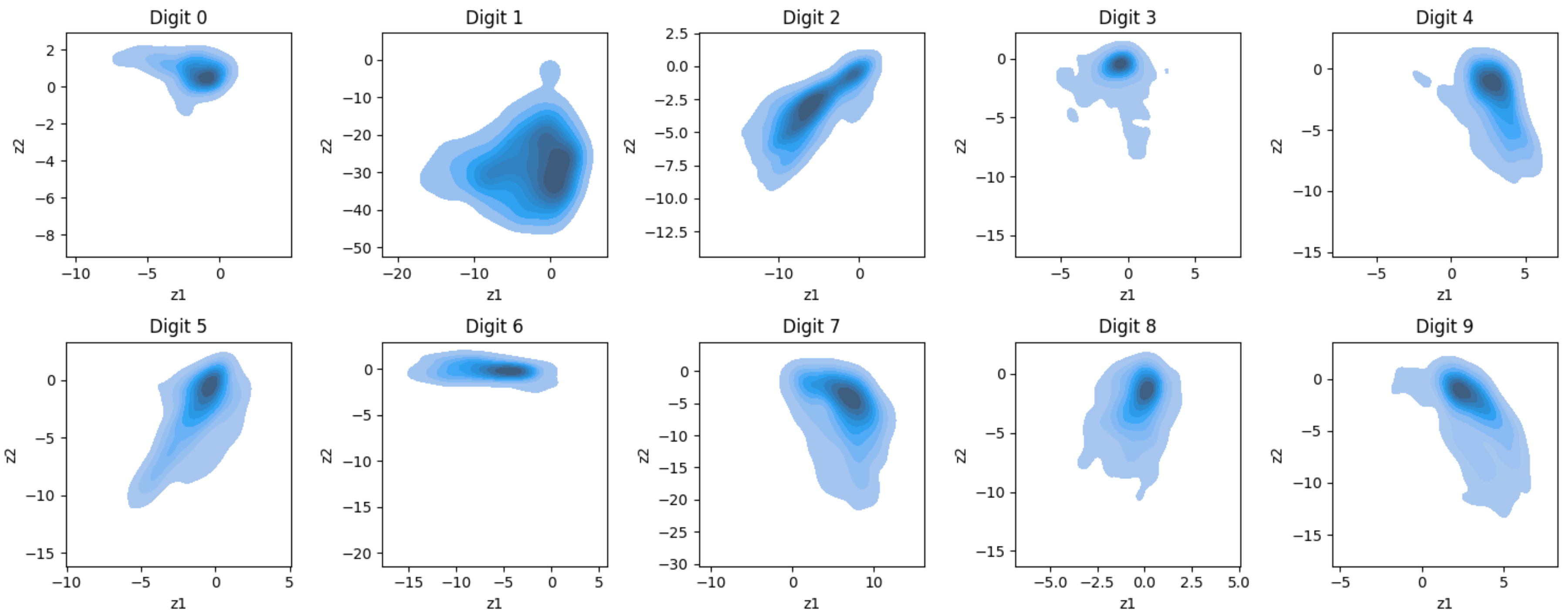
https://medium.com/@Ravitejakandimalla/variational-autoencoder-vs-pca-pytorch-3469e7ad12a1

AUTOENCODER VS PCA



The autoencoder does a much better job of separating classes, it can learn non-linear compression.

SHORTCOMINGS OF THE AUTOENCODER

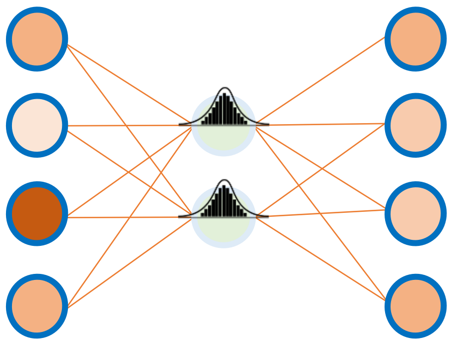


As a generative model it is not great, because the latent space is not very regular

GO TO NOTEBOOKS

VARIATIONAL AUTOENCODERS

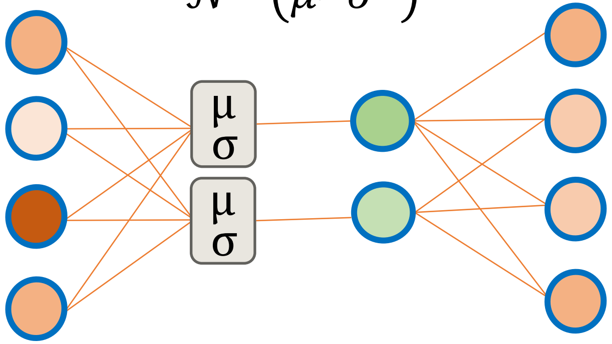
Instead of learning a number for the latent variable we learn a distribution of numbers.



VARIATIONAL AUTOENCODERS

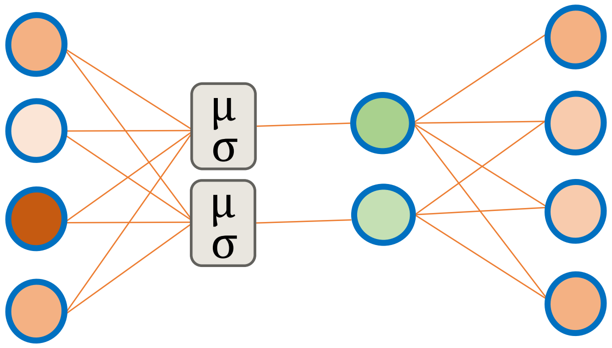
In practice we learn a mean and a (log-)variance value and then sample from the normal distribution defined by these values.

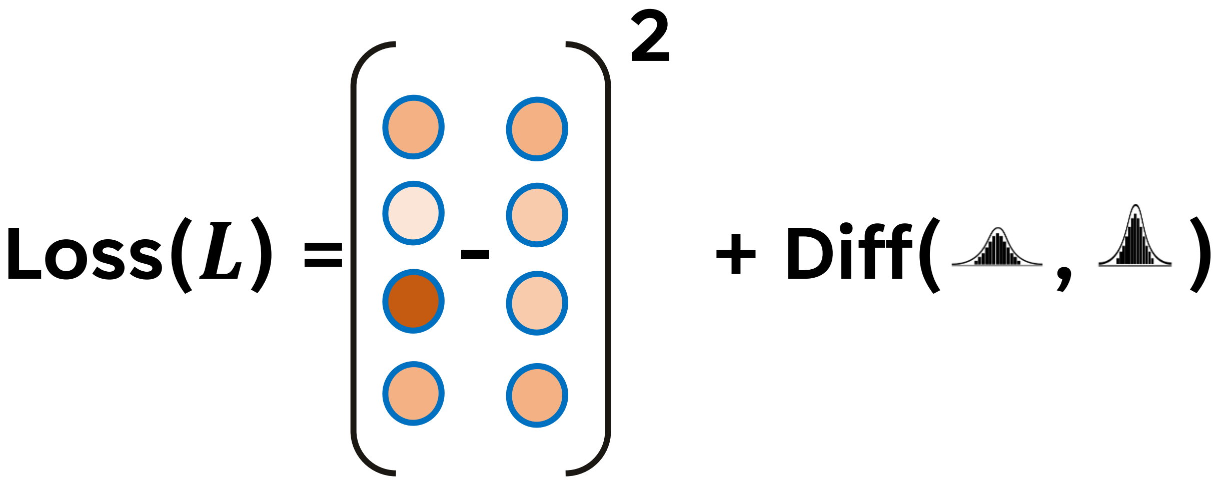
$\mathcal{N} \sim (\mu, \sigma^2)$



VARIATIONAL AUTOENCODERS

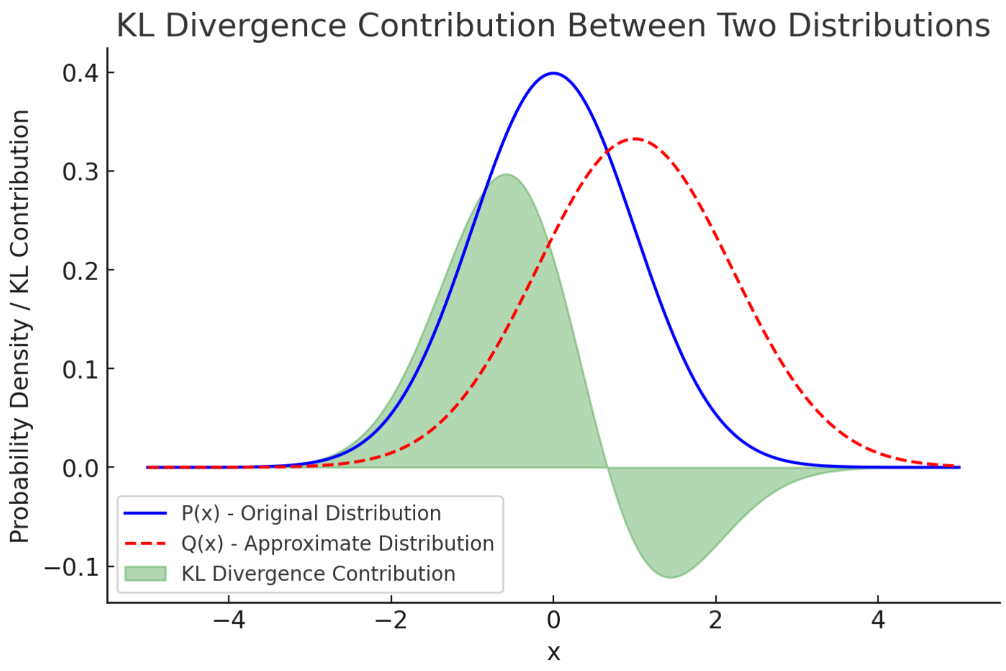
We add a new term to the loss function, which penalises the latent distribution for deviating from the normal distribution



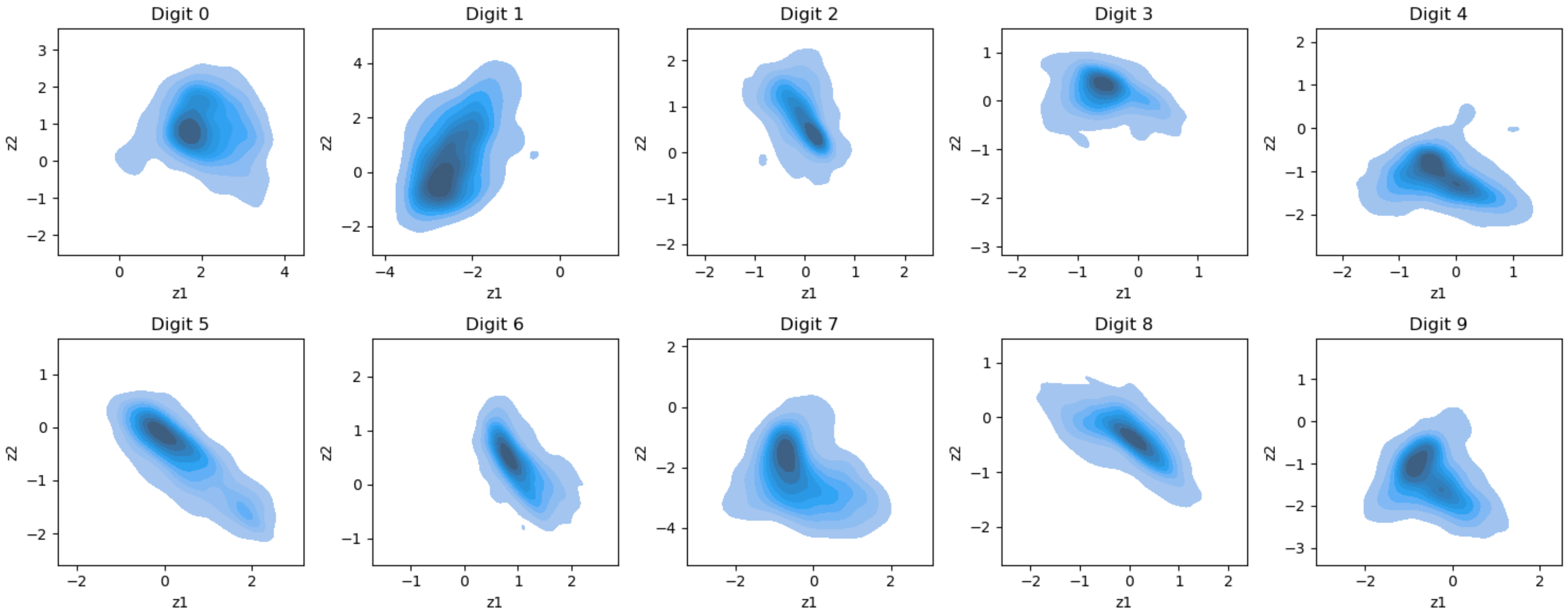


KL DIVERGENCE

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) log(\frac{p(x_i)}{q(x_i)})$$



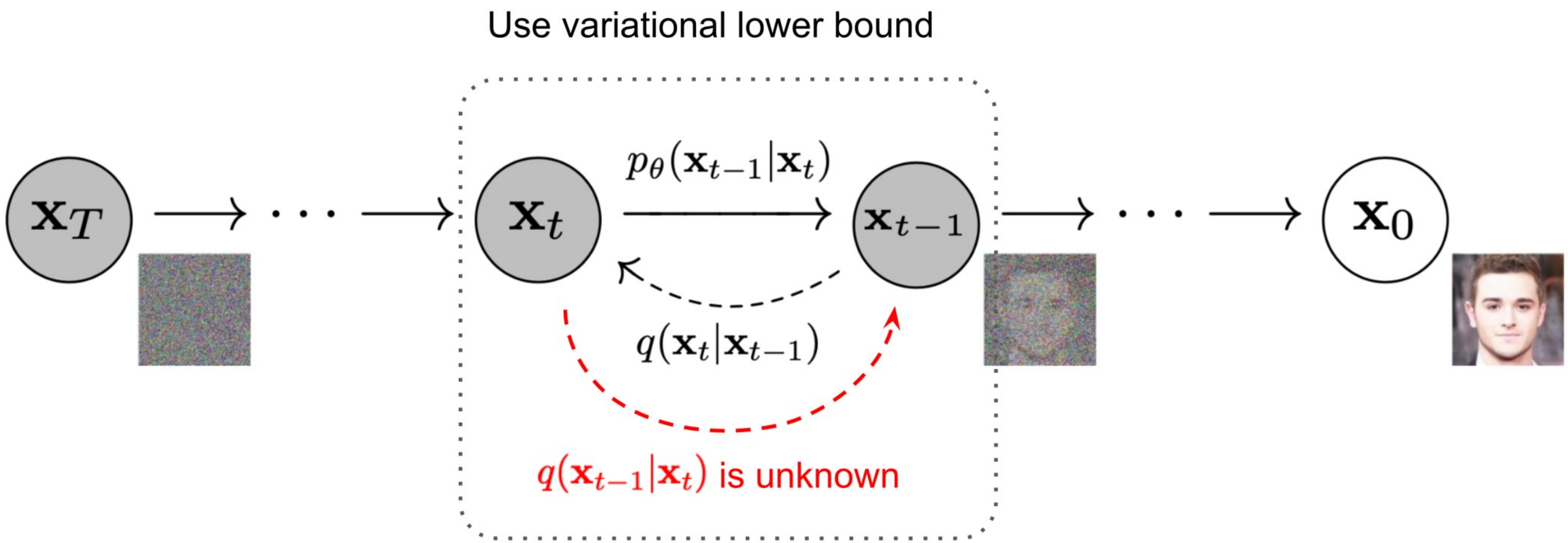
VAES: REGULARISED LATENT SPACE



A much more regular latent space – samples close together will look similar

GO TO NOTEBOOKS

DIFFUSION MODELS



DIFFUSION MODELS

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta}\mathbf{x}_{t-1}; \beta\mathbf{I})$$

Simple forward model. Just add a random sample from a normal distribution to the previous step.

DIFFUSION MODELS

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t); \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$

Reverse process, adding noise to the distribution, but this time the multimodal mean and covariance are complicated. Here we learn a model of this process.

DIFFUSION MODELS

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

until converged

Choose an initial sample
Choose a time
Generate a noised sample based on the time
Calculate the difference between the actual noise and the noise predicted by the noise model
Update params of the noise model to minimise this difference

DIFFUSION MODEL INFERENCE

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Start from a sample from the normal distribution
Going backwards in time
Draw some random noise
Calculate the distribution at the ‘previous’ timestep:
Use current distribution and subtract the predicted noise (add some random noise to this)
Continue back to t zero

CONCEPT CHECKLIST

Generative models involve mapping between distributions

Autoencoders can reduce dimensionality

Variational autoencoders learn reduced dimensional distributions

Diffusion models work by learning a reverse process from normal noise to true distribution