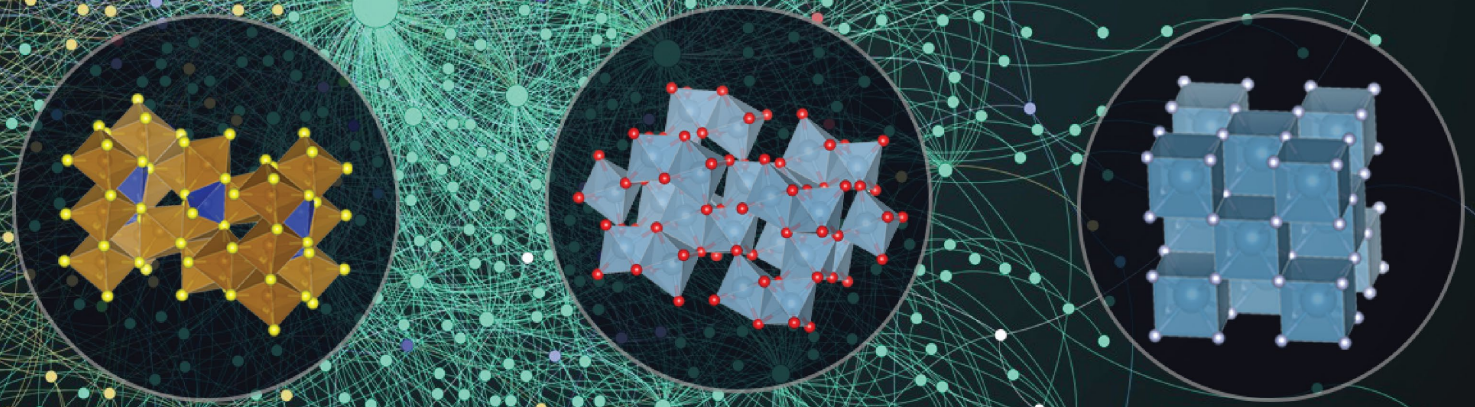
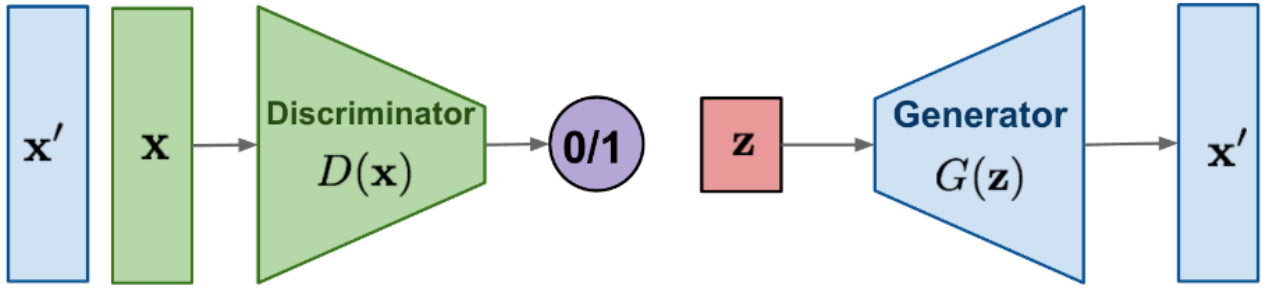
Generative Adversarial Networks

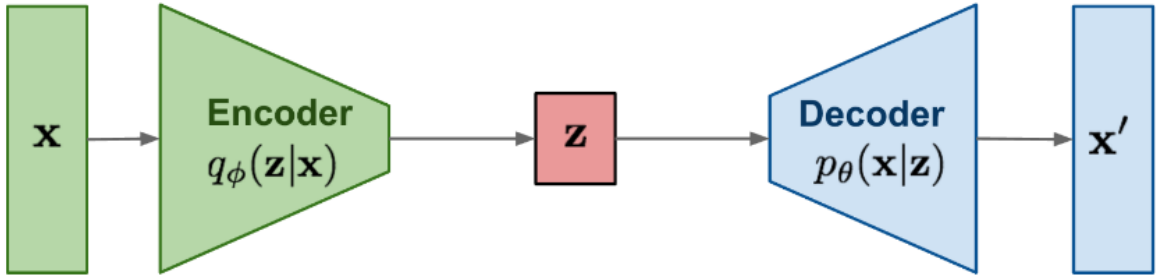


There are many generative models to choose from these days!

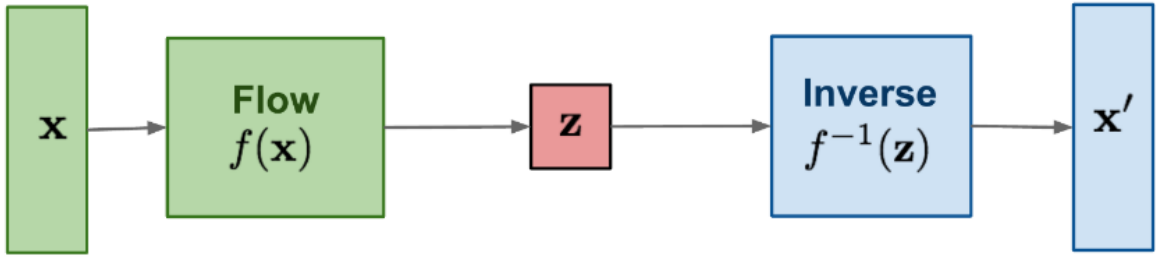
GAN: Adversarial training



VAE: maximize variational lower bound



Flow-based models: Invertible transform of distributions



Diffusion models: Gradually add Gaussian noise and then reverse

Generative models are increasing at an incredible rate!

2014

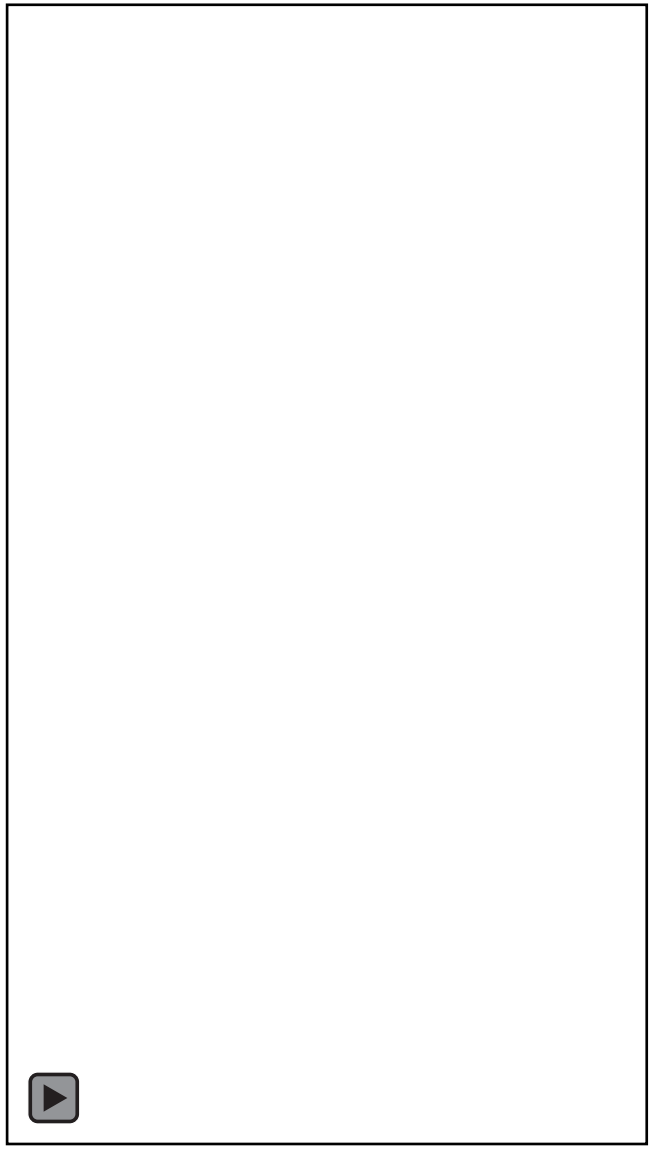
2015

2016

2017

2018

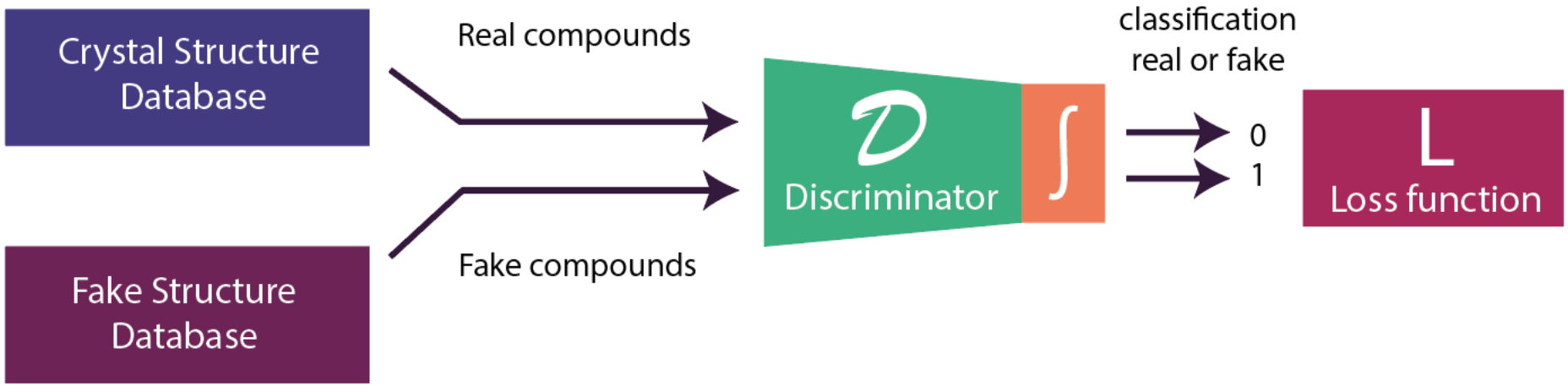
2021



2023

https://twitter.com/goodfellow_ian/status/1084973596236144640

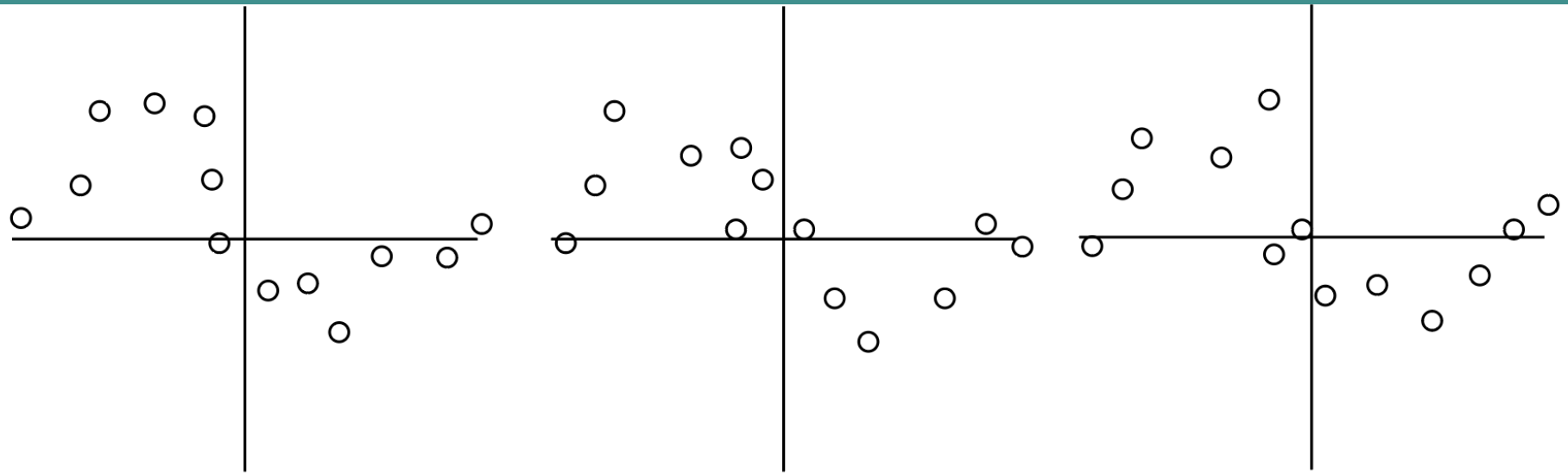
GANs start with a discriminator



This alone can’t generate something new…

Let alone something that matches a specific distribution

The discriminator's job is to learn the underlying data distribution



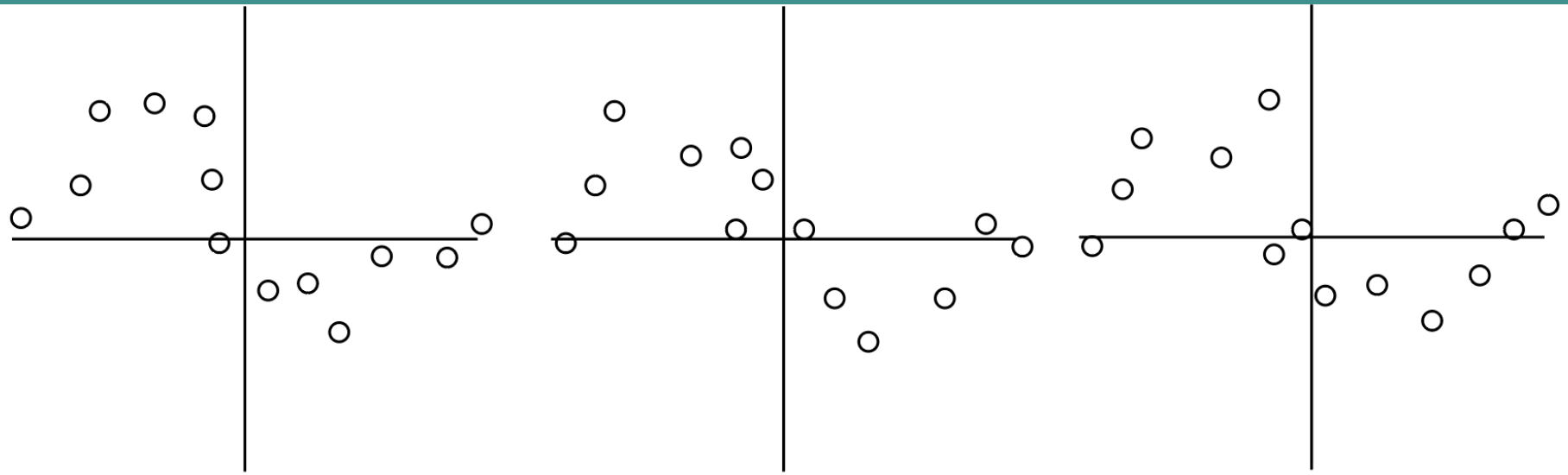
What would the underlying distribution look like?

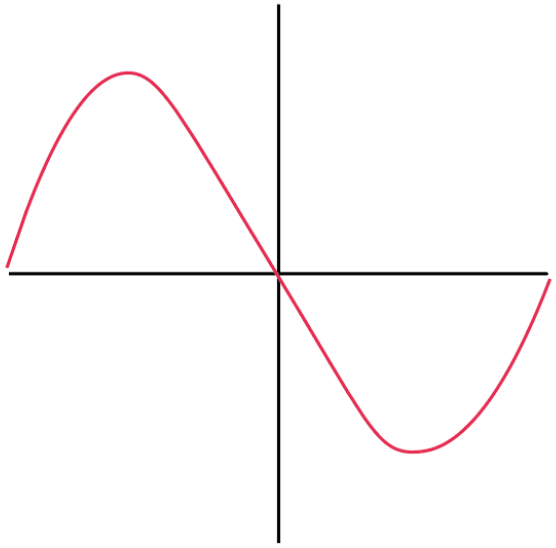
If we used the model to generate data, how would it look?

What would we want a generative model to output?

Example from Generative Adversarial Networks (GANs), Computerphile YouTube

The discriminator's job is to learn the underlying data distribution



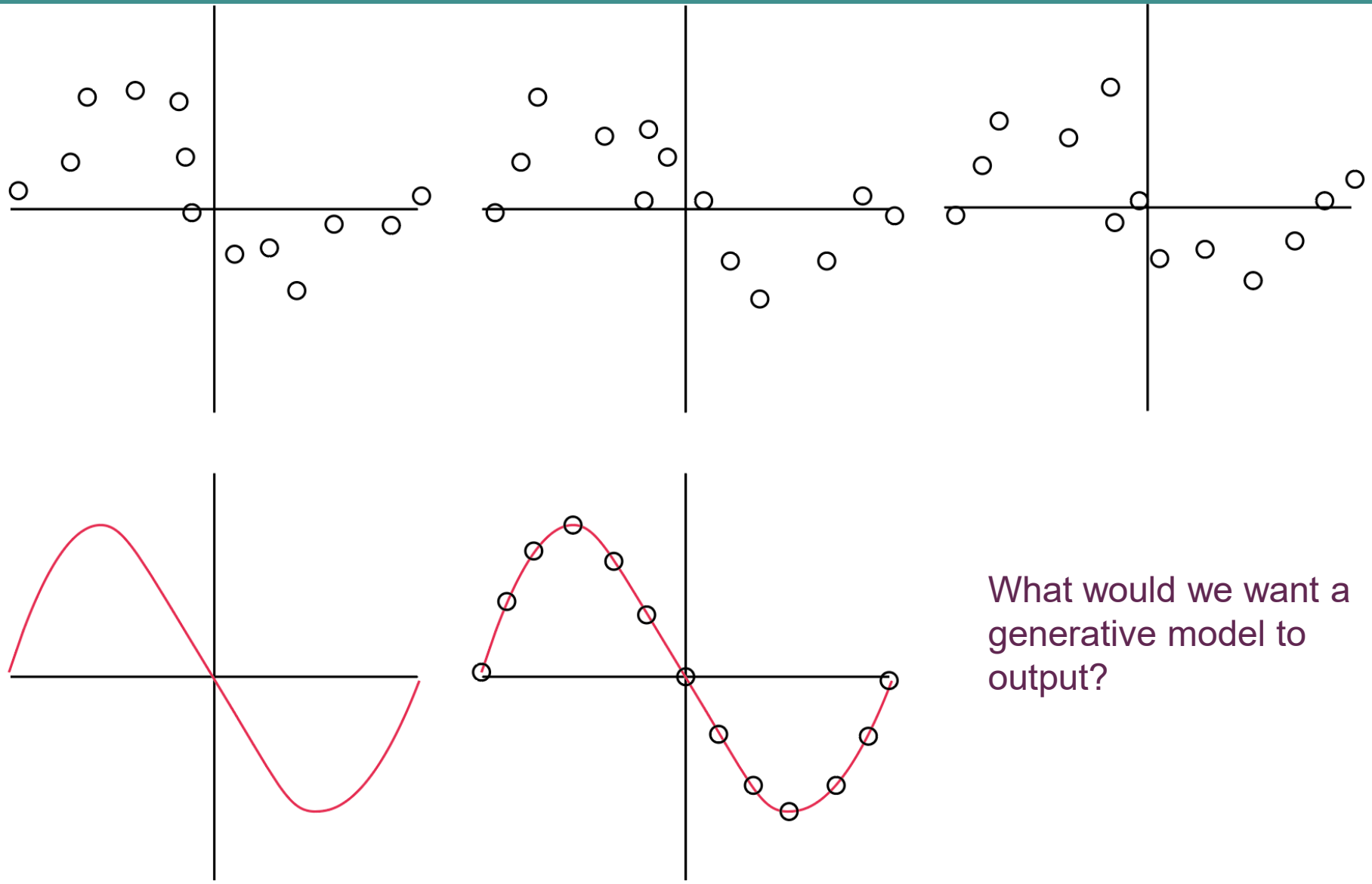


If we used the model to generate data, how would it look?

What would we want a generative model to output?

Example from Generative Adversarial Networks (GANs), Computerphile YouTube

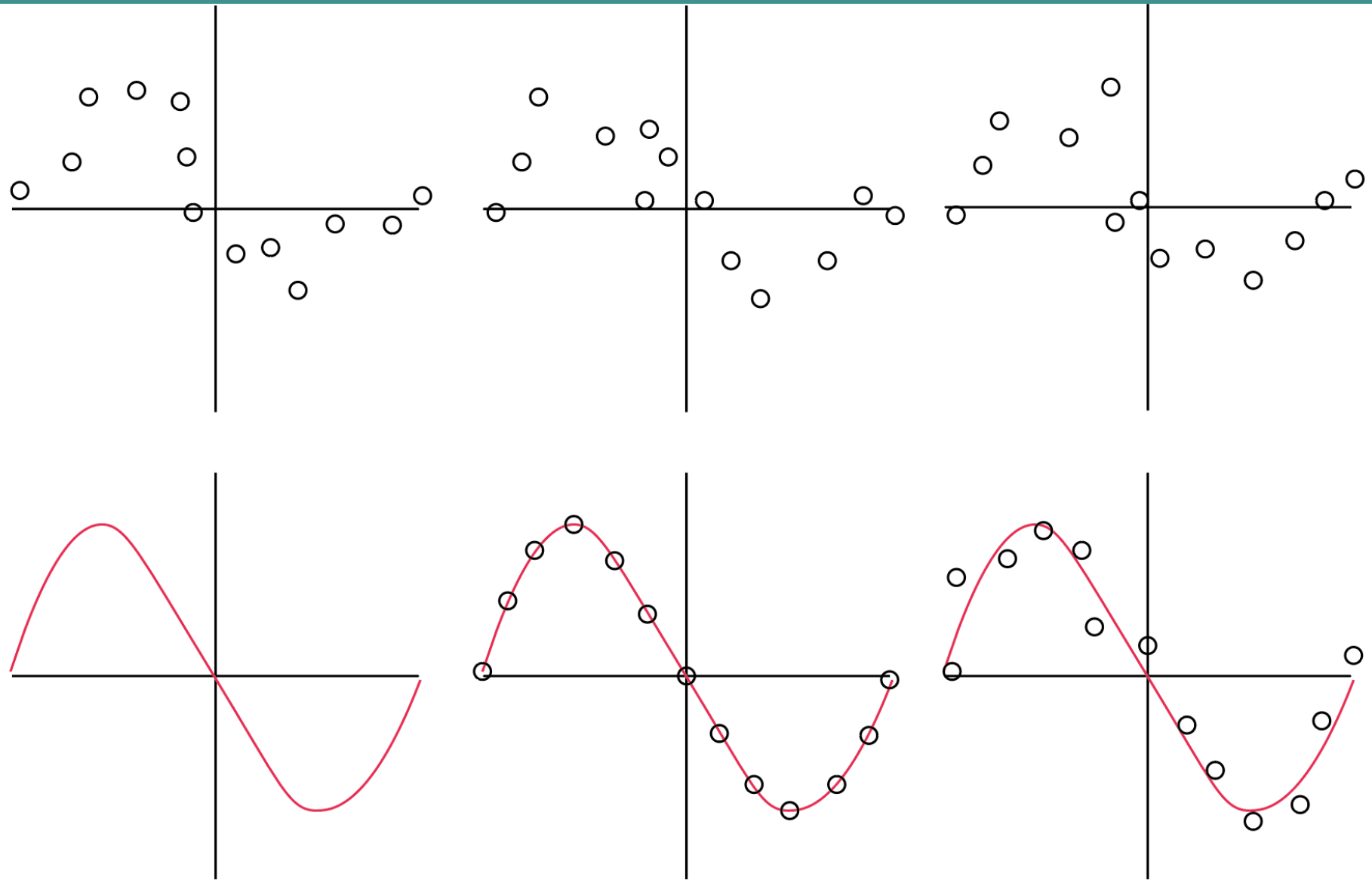
The discriminator's job is to learn the underlying data distribution



What would we want a generative model to output?

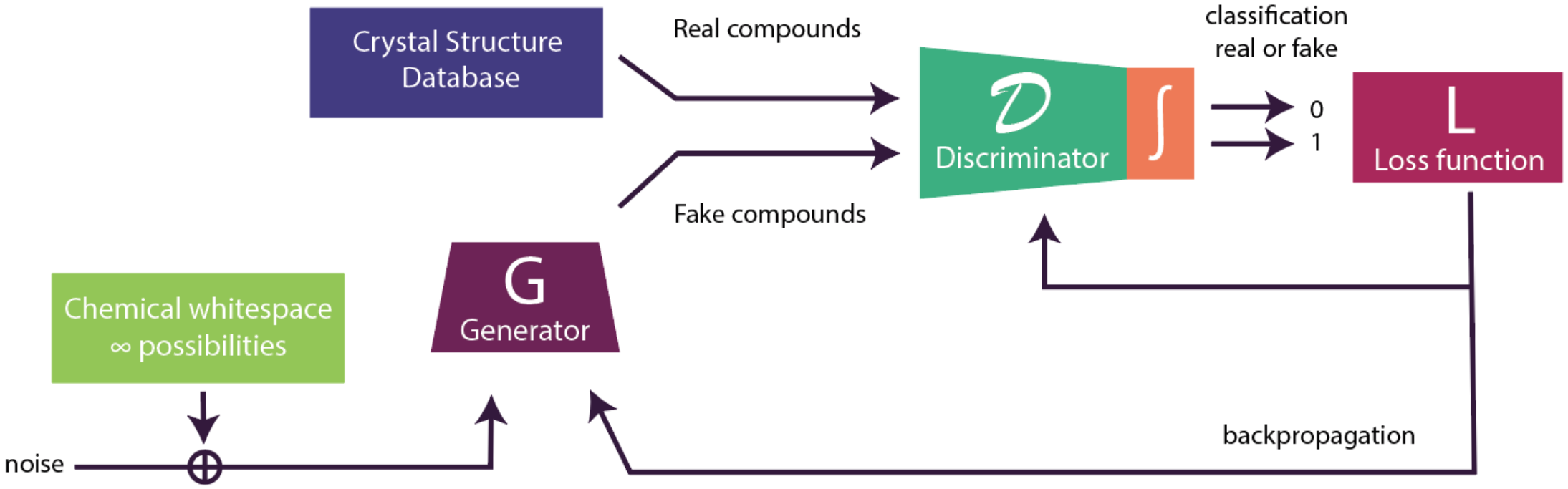
Example from Generative Adversarial Networks (GANs), Computerphile YouTube

The discriminator's job is to learn the underlying data distribution



Example from Generative Adversarial Networks (GANs), Computerphile YouTube

To help us generate new samples, we can train adversarially



Two networks!

Both compete in a min/max game

We do a min max game on the cost function

$$V(D,G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_{data}(z)}[(1 - \log D(G(z)))]$$

Discriminator's prediction on real data

Discriminator's prediction on fake data

We do a min max game on the cost function

$$V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_{data}(z)}[(1 - \log D(G(z)))]$$

Discriminator's prediction on real data

Discriminator's prediction on fake data

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_{data}(z)}[(1 - \log D(G(z)))]$$

During training we update both networks iteratively

For each training do:

for k steps do:

sample m noise samples $\{z_1, z_2, \ldots z_m\}$ and transform with Generator

sample m real samples $\{x_1, x_2, \ldots x_m\}$ from real data

update the Discriminator by ascending the gradient

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [log\ D(x_i) + \log\left(1 - D\left(G(z_i)\right)\right)]$$

During training we update both networks iteratively

For each training do:

for k steps do:

sample m noise samples $\{z_1, z_2, \ldots z_m\}$ and transform with Generator

sample m real samples $\{x_1, x_2, \ldots x_m\}$ from real data

update the Discriminator by ascending the gradient

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [log\ D(x_i) + \log\left(1 - D\left(G(z_i)\right)\right)]$$

end for

sample m noise samples $\{z_1, z_2, \ldots z_m\}$ and transform with Generator

update the Generator by descending the gradient

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [log\ D(x_i) + \log\left(1 - G\left(D(z_i)\right)\right)]$$

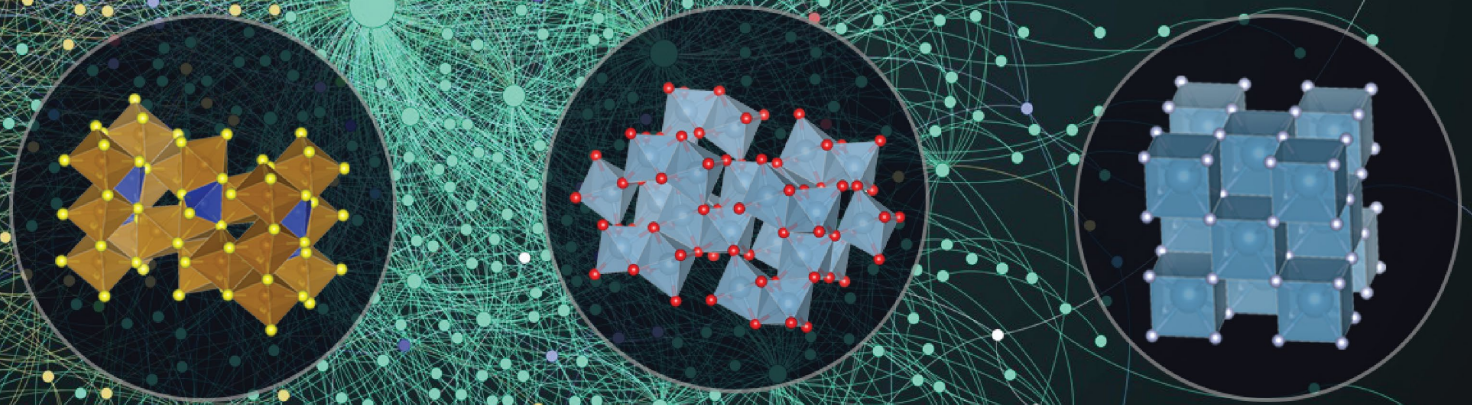
Training GANs is notoriously difficult

- Non-convex game: we have to find a Nash equilibrium between a non-convex game
- Mode collapse: a certain solution keeps fooling discriminator, so it never changes
- Vanishing gradient: if D outperforms then G's gradient vanishes so G can't learn which makes D even better -> feedback loop
- Exploding gradient: if G outperforms then D's loss escalates and gradient explodes
- Balancing act: D & G must find a balance to adversarially train
- Sensitivity to hyperparameters: learning rate, batch size, architectures

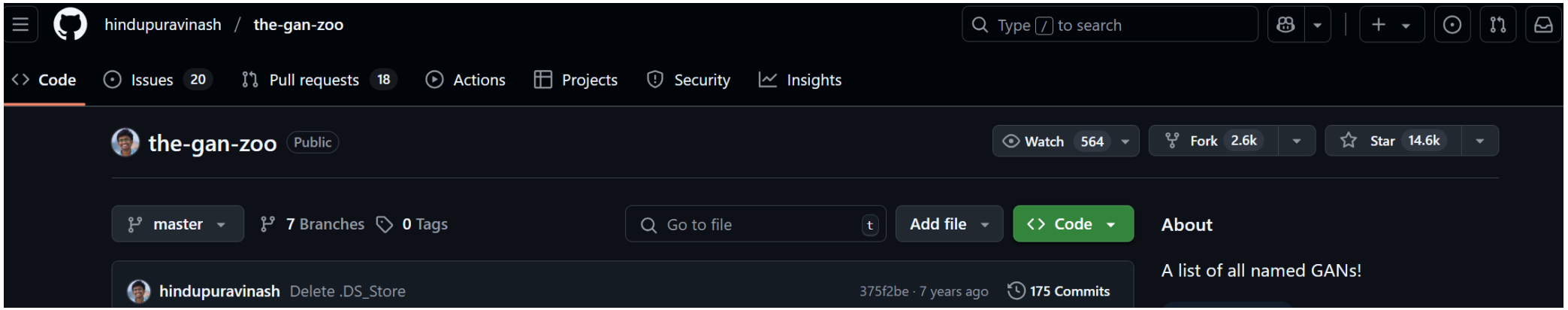
Things we can do to help:

- Alternative loss functions
- Regularization techniques
- Architectures (Wasserstein or Conditional GANs) each give additional info to G, D

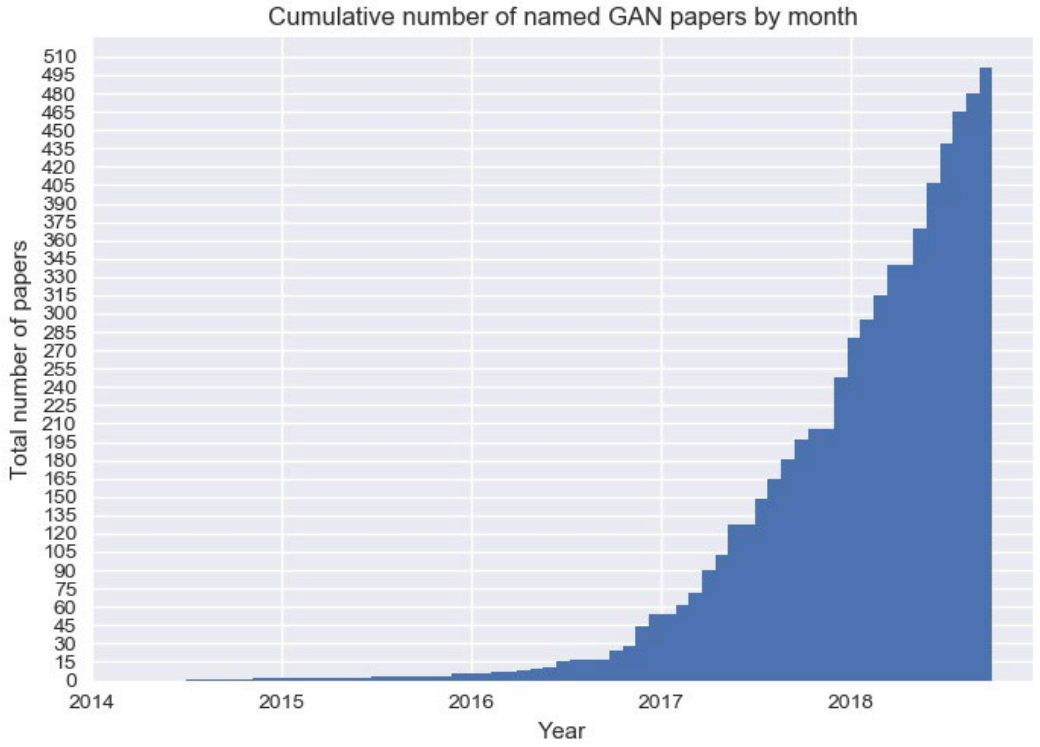
The GAN Zoo: endless variations



There are so ~~hundreds~~ thousands of distinct GAN architectures



- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
- 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
- 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACGAN - Coverless Information Hiding Based on Generative adversarial networks
- acGAN - On-line Adaptative Curriculum Learning for GANs
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
- AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples



1. Core GAN architectures

Vanilla GAN

- Generator & Descriminator with minimax loss
- Key features: Fully connected layers
- Challenges: Mode collapse, training instability
- Use Case: Basic image generation, proof of concept models

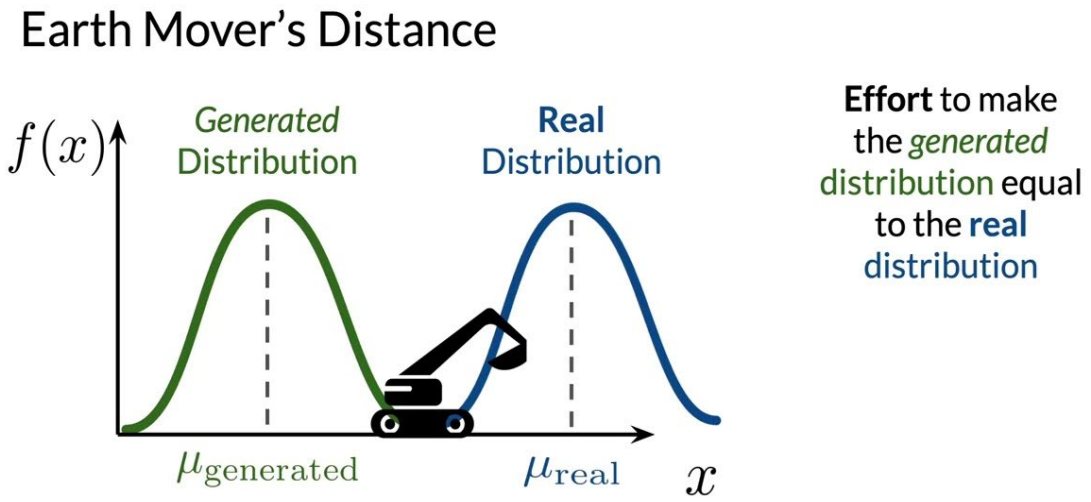
Deep convolutional GAN (DCGAN)

- Includes CNNs for better generation
- Key features: replace fully connected layers with CNNs, batch normalization, ReLU
- Improvements: stabilize training, higher quality images
- Use Case: Basic image generation

1. Core GAN architectures

Wasserstein GAN

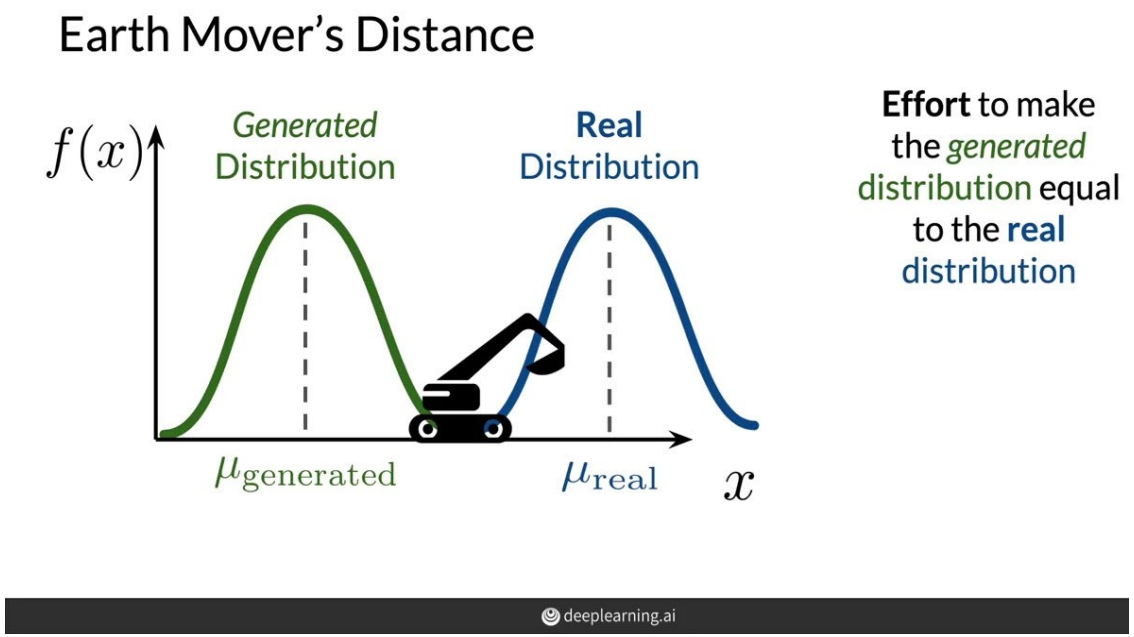
- Uses Wasserstein (Earth Mover's) distance instead of Kullback-Leibler or Jensen-Shannon divergence
- Key features: Weight clipping or gradient penalty (WGAN-GP) to enforce Lipschitz continuity, improved loss function.
- Use Case: high quality image generation, robust training



Earth mover's distance

The critic outputs a single scalar score (not a probability between 0 and 1). This score represents how "real" the input is, but it's unbounded (not constrained to [0, 1]). For a given batch, these scores show a distribution

Wasserstein Distance: The difference between the **average** critic scores for real samples and fake samples approximates the Wasserstein distance.



What is weight clipping?

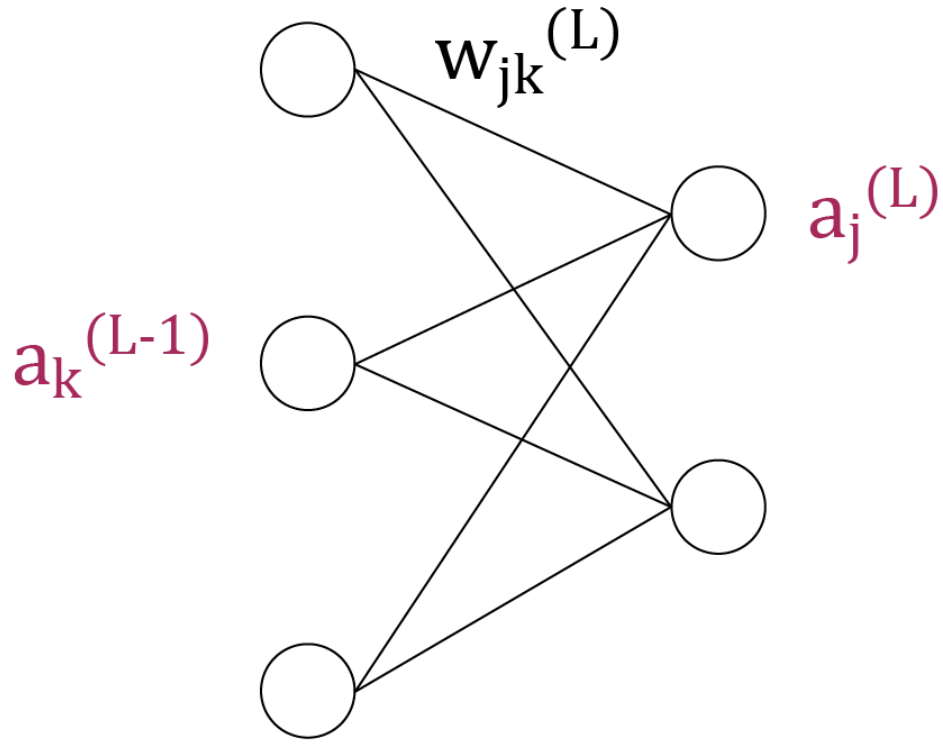
$a_0^{(1)}$ means 0-th node of first layer

$$a_0^{(1)} = \sigma\left(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + \cdots + w_{0,n}a_n^{(0)} + b_0\right)$$

Tunable parameters are weights and biases!

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix}\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

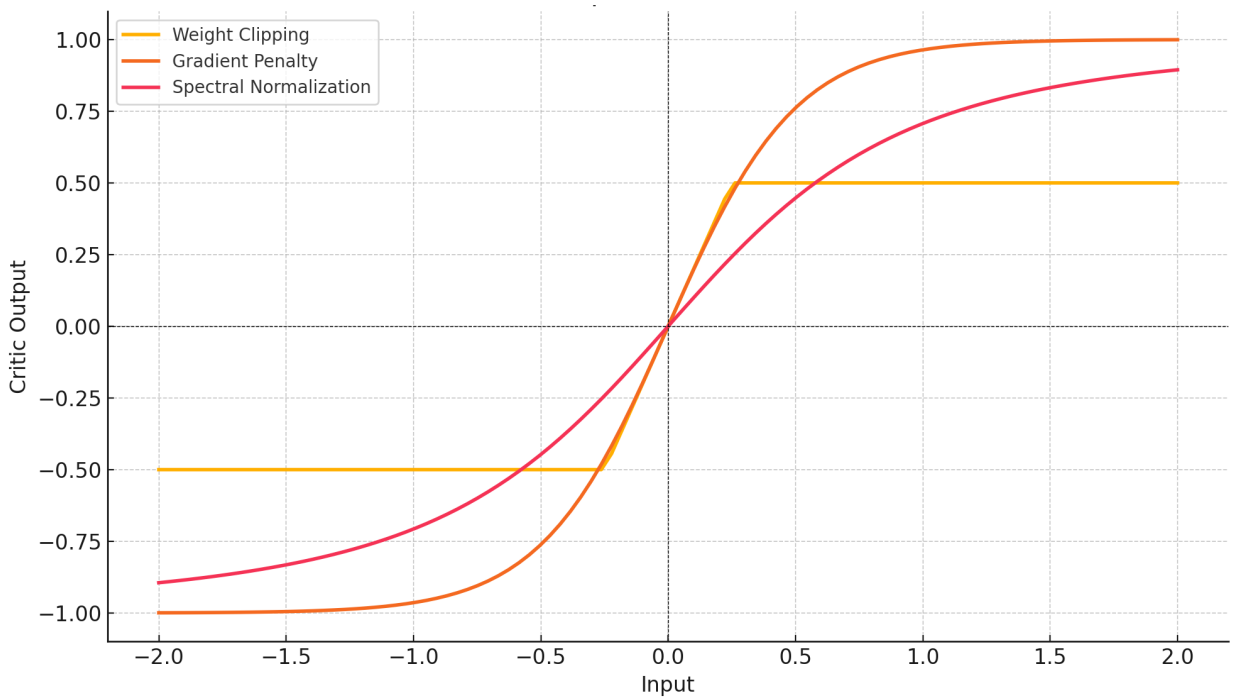
$$a^{(1)} = \sigma(Wa^{(0)} + b)$$



1. Core GAN architectures

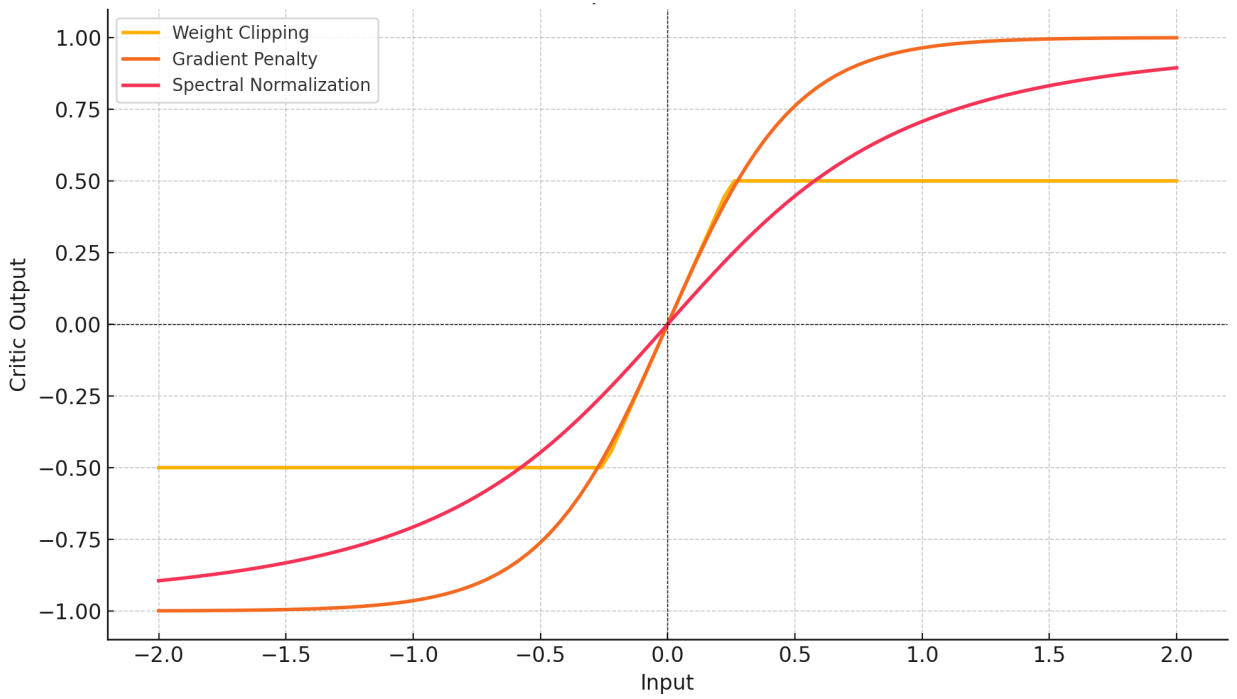
Weight Clipping: Limits the weights of the discriminator to a fixed range, like [-0.01, 0.01]. This enforces the Lipschitz constraint (smoothness) required for WGANs, preventing the discriminator from becoming too powerful. However, it can overly restrict the model, reducing its capacity to learn complex patterns.

$$|f(x_1) - f(x_2)| \leq K \cdot |x_1 - x_2|$$



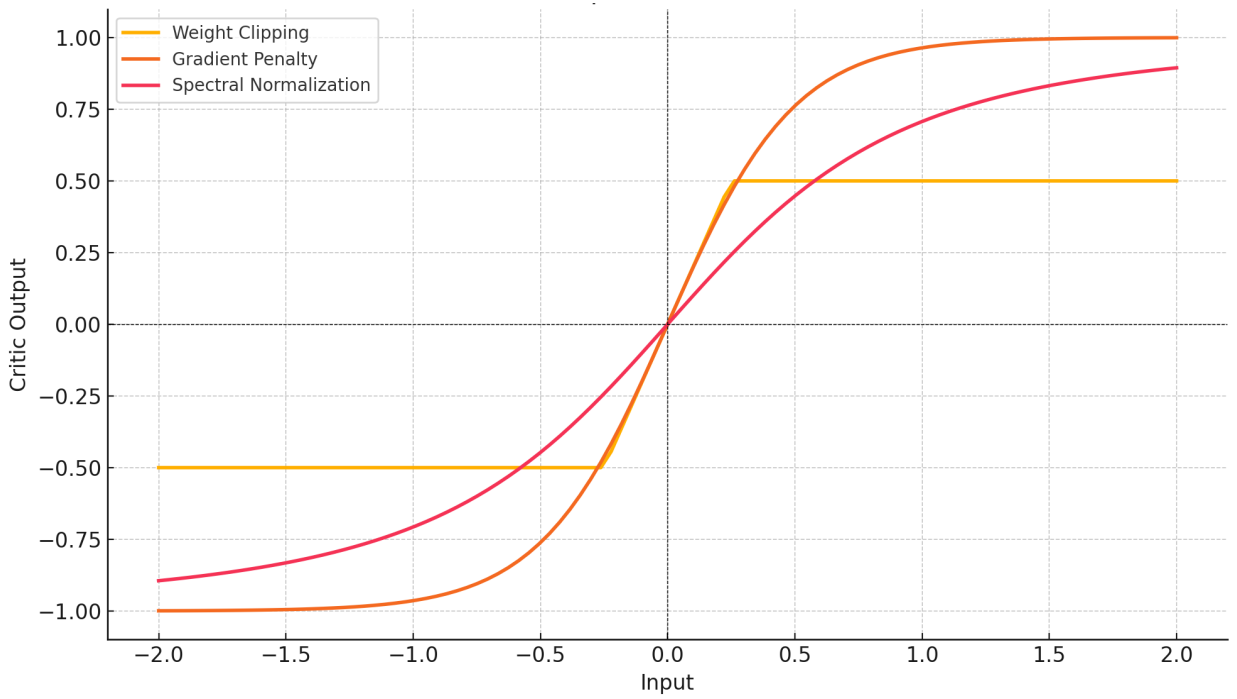
1. Core GAN architectures

Gradient Penalty: Instead of clipping weights, this adds a penalty term to the loss function to enforce the Lipschitz constraint. It penalizes the discriminator if the gradient of its output with respect to the input is too large (norm > 1). This is more flexible than weight clipping and often leads to better performance, as used in WGAN-GP (Gradient Penalty).



1. Core GAN architectures

Hyperspectral Normalization: Less common, this refers to normalizing the discriminator's weights or activations across feature dimensions (like spectral normalization). It stabilizes training by controlling the scale of the discriminator's output, ensuring it doesn't grow too large or unstable, while still allowing the model to learn complex patterns.



2. Conditional GANs

Conditional GAN (CGAN)

- Conditions generator and discriminator on auxiliary info (e.g. class label)
- Key features: Adds labels or data embeddings to input to guide generation
- Use Case: class specific image generation (MNIST digits)

Auxiliary Classifier GAN (ACGAN)

- Discriminator outputs both real/fake and class probabilities, improving conditional generation.
- Key features: Introduces an auxiliary classification loss in addition to the adversarial loss, allowing the generator to receive direct guidance on class identity.
- Use Case: Multi-class image synthesis with clear class separation.

2. Conditional GANs

Image-to-Image GANs (Pix2Pix)

- Maps input images to output images using paired data (e.g., sketches to photos).
- Key features: Conditional GAN with U-Net-based generator, L1 loss for pixel-level precision.
- Use Case: Image translation (e.g., day-to-night)

Cycle Consistent GAN (CycleGAN)

- Performs unpaired image-to-image translation using cycle-consistency loss.
- Key features: Two generators and discriminators ensure bidirectional mapping (e.g., A to B and back to A).
- Use Case: Style transfer (e.g., horse to zebra, summer to winter).

2. Conditional GANs

Text-to-Image GANs (Pix2Pix)

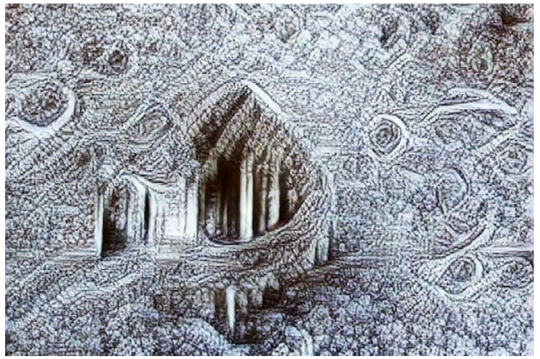
- Generates images from text descriptions using conditional embeddings.
- Key features: Incorporates NLP embeddings (e.g., word2vec, BERT) into the GAN framework.
- Use Case: Generating images from captions (e.g., “a red bird with a blue beak”).

digitaltrends

Best Products | Product Reviews | News | Deals | Original Series | Buying Guides | 5G | Downloads | More

The BigSleep A.I. is like Google Image Search for pictures that don’t exist yet

By Luke Dormehl
February 12, 2021



BigSleep

In case you’re wondering, the picture above is “an intricate drawing of eternity.” But it’s not the work of a human artist; it’s the creation of BigSleep, the latest amazing example of generative artificial intelligence (A.I.) in action.

3. Progressive and Scalable GANs

Progressive GAN (ProGAN)

- Trains GANs incrementally, starting with low-resolution images and adding layers for higher resolutions.
- Key features: Progressive growing of generator and discriminator networks, stabilizes high-resolution training.
- Use Case: High resolution image generation (CelebA-HQ)

StyleGAN and variants

- Introduces style-based generation with a mapping network for controlling features at different scales.
- Key features: Adaptive instance normalization (AdaIN), mapping network for disentangled latent space.
- Use Case: Photorealistic faces, style mixing, video generation.

3. Progressive and Scalable GANs

BigGAN

- Scales GANs with large models and datasets for high-fidelity, diverse outputs.
- Key features: Class-conditional generation, large batch sizes, spectral normalization.
- Use Case: High-quality, diverse image generation (trained on ImageNet).

4. Regularization and stabilization GANs

Spectral Normalization GAN (SN-GAN)

- Normalizes weights in the discriminator to stabilize training.
- Key features: Enforces Lipschitz continuity via spectral norm, computationally efficient.
- Use Case: Stable training for diverse datasets

Least Squares GAN (LSGAN)

- Replaces cross-entropy loss with least squares loss for smoother gradients.
- Key features: Reduces vanishing gradients, improves quality.
- Use Case: General image training, improved quality

4. Regularization and stabilization GANs

Energy-Based GAN (EBGAN)

- Treats discriminator as an energy function, assigning low energy to real data
- Key features: Margin-based loss, autoencoder-like discriminator.
- Use Case: Robust training for diverse data

5. GANs for specialized data types

3D-GAN

- Generates 3D voxel grids or point clouds.
- Key features: 3D convolutional layers, volumetric discriminators.
- Use Case: 3D object synthesis (e.g., chairs, cars).

Video GAN (VGAN)

- Generates video sequences by modeling temporal dynamics.
- Key features: Spatiotemporal convolutions, dual discriminators for frames and sequences.
- Use Case: Video production/generation

5. GANs for specialized data types

Audio GAN

- Generates audio waveforms or spectrograms.
- Key features: 1D convolutions or recurrent networks, time-frequency representations.
- Use Case: music generation, speech synthesis

6. Hybrid GANs with other models

GAN+VAE (VAE-GAN)

- Combines GAN's realism with VAE's latent space structure.
- Key features: VAE encodes data into a latent space, GAN refines outputs.
- Use Case: Image generation with structured latent spaces.

GAN + Diffusion models

- Integrates GANs with diffusion processes for high-quality generation.
- Key features: GANs accelerate diffusion's sampling, diffusion stabilizes GAN training.
- Use Case: High-fidelity images with faster sampling.

6. Hybrid GANs with other models

GAN + Reinforcement Learning (RL-GAN)

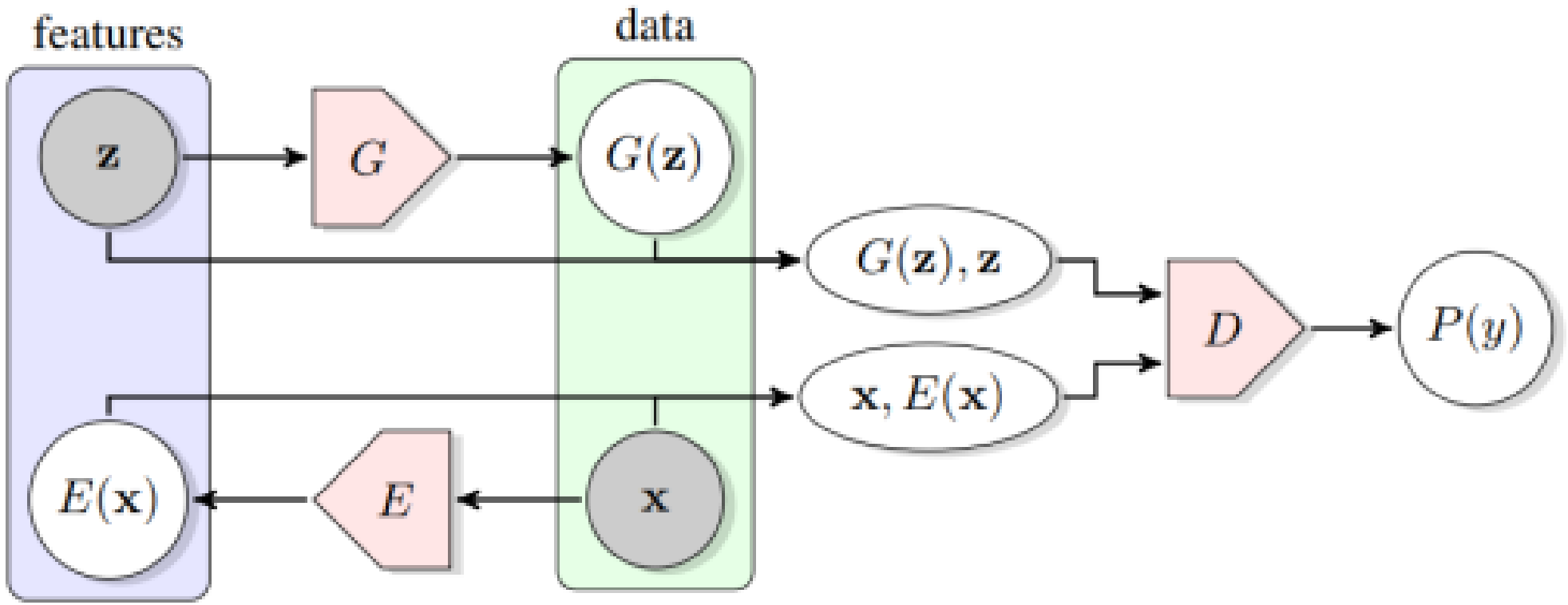
- Uses RL to optimize generator or discriminator policies.
- Key features: Treats generation as a sequential decision process, reward-based training.
- Use Case: Text generation, sequential data

GAN + Self Supervised Learning

- Incorporates self-supervised tasks (e.g., contrastive learning) to improve representation learning.
- Key features: Pretrained discriminators, contrastive losses.
- Use Case: Data efficient generationtasks

Example of unusual GAN architecture

x is real data, z is sampled from “noise”, P(y) output of real or fake



7. Domain-specific and Task-oriented GANs

Adversarial Autoencoder (AAE)

- Uses GANs to match latent distributions in autoencoders.
- Key features: Adversarial training on latent space, no explicit KL-divergence.
- Use Case: Disentangled representations, semi-supervised learning.

Domain adaptation GANs

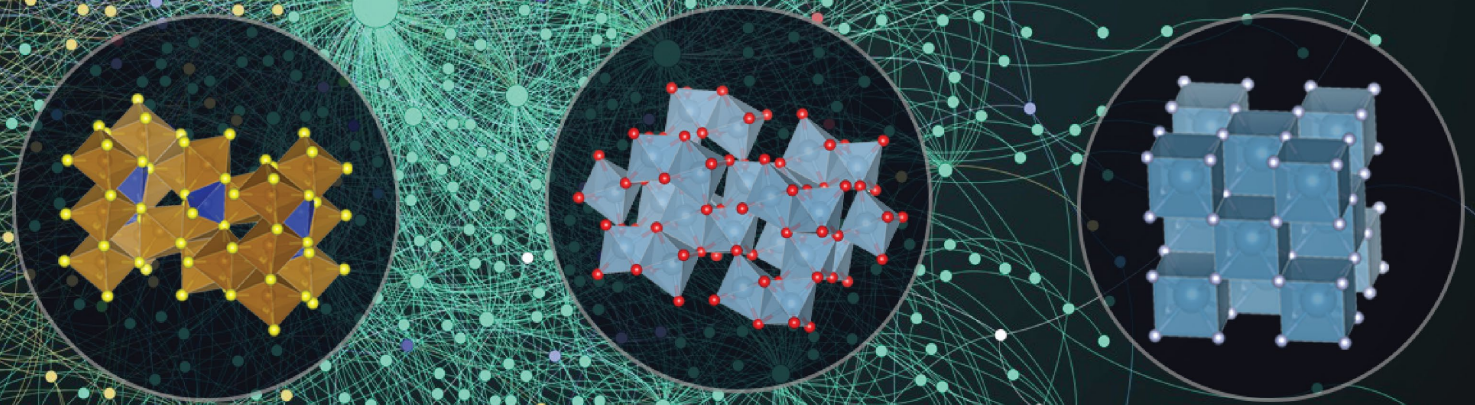
- Aligns source and target domains for cross-domain tasks.
- Key features: Domain-invariant features, cycle-consistency or shared discriminators.
- Use Case: Transfer learning (e.g. synthetic-to-real)

7. Domain-specific and Task-oriented GANs

Anomaly Detection GANs

- Uses GANs to model normal data and detect outliers.
- Key features: Reconstruction-based or distribution-based anomaly scoring.
- Use Case: defect detection, medical imaging etc

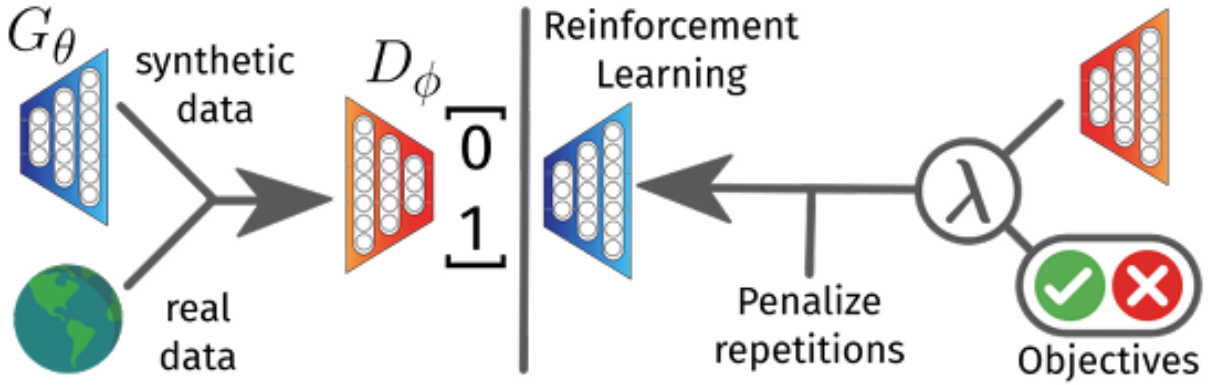
GAN use in Materials Science



ORGAN combined GAN with RL for guided discovery

Guimaraes et al. (2017) "ORGAN" for new molecules. GAN + RL

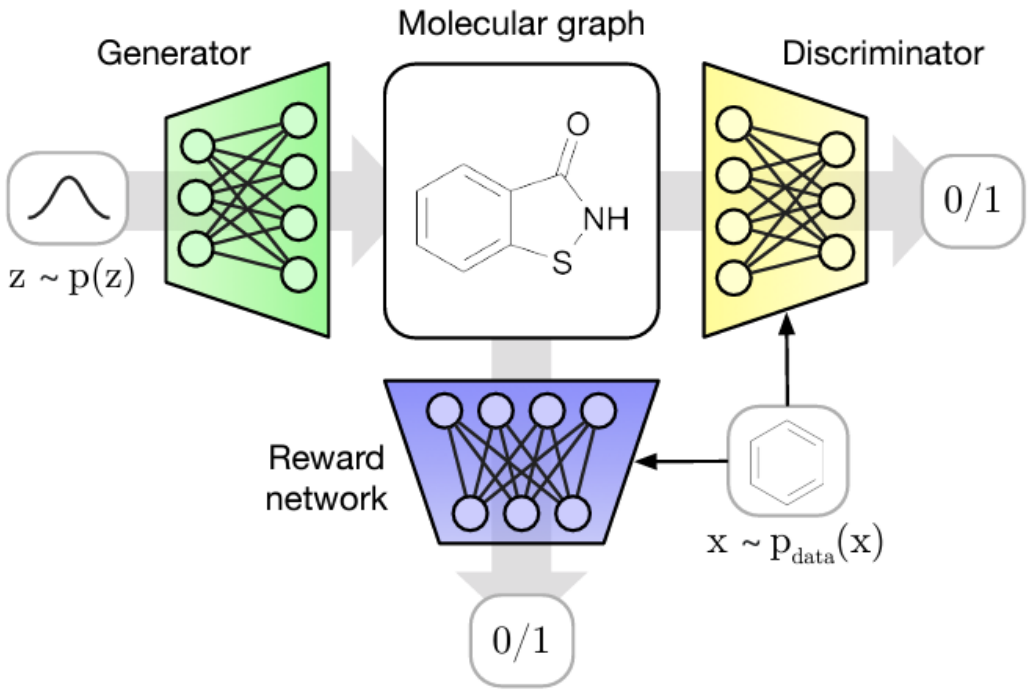
- Reinforcement learning following GAN
- Penalties for introducing nonunique generated sequences
- Generator had RNN architecture
- Discriminator had CNN architecture
- Data was 5000 molecules form ZINC database (drug and nondrugs)
- Objectives: solubility, synthesizability, drug-likeliness



MolGAN improves this by working with graphs

Cao et al. (2018) "MolGAN" for new molecules.

- Generated molecule graphs directly
- By avoiding SMILES, it had nearly 100% valid molecule output
- GAN+RL to steer generation towards desired molecules
- Trained on QM9 dataset



Mol-CycleGAN lets us generate based on templates

Maziarka et al. (2020) "Mol-CycleGAN" based on CycleGAN (Zhu 2017, 20k citations)

- Generated molecule based on input molecule (image to image)
- Structural similarity -> property similarity
- High Tanimoto similarity

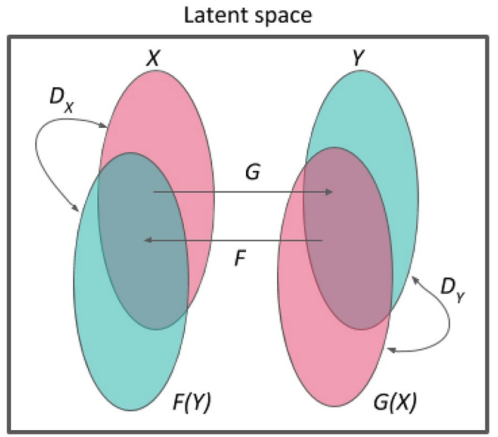
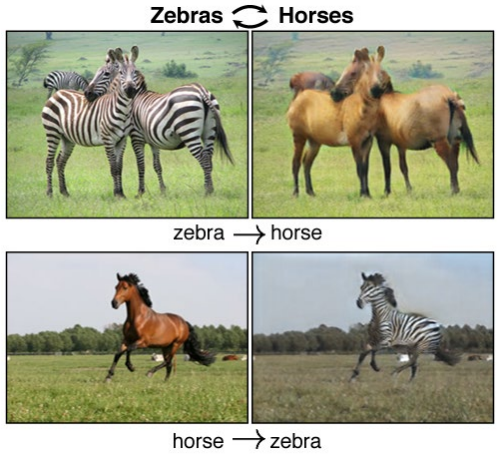


Fig. 1 Schematic diagram of our Mol-CycleGAN. *X* and *Y* are the sets of molecules with selected values of the molecular property (e.g. active/inactive or with high/low values of logP). *G* and *F* are the generators. D_X and D_Y are the discriminators



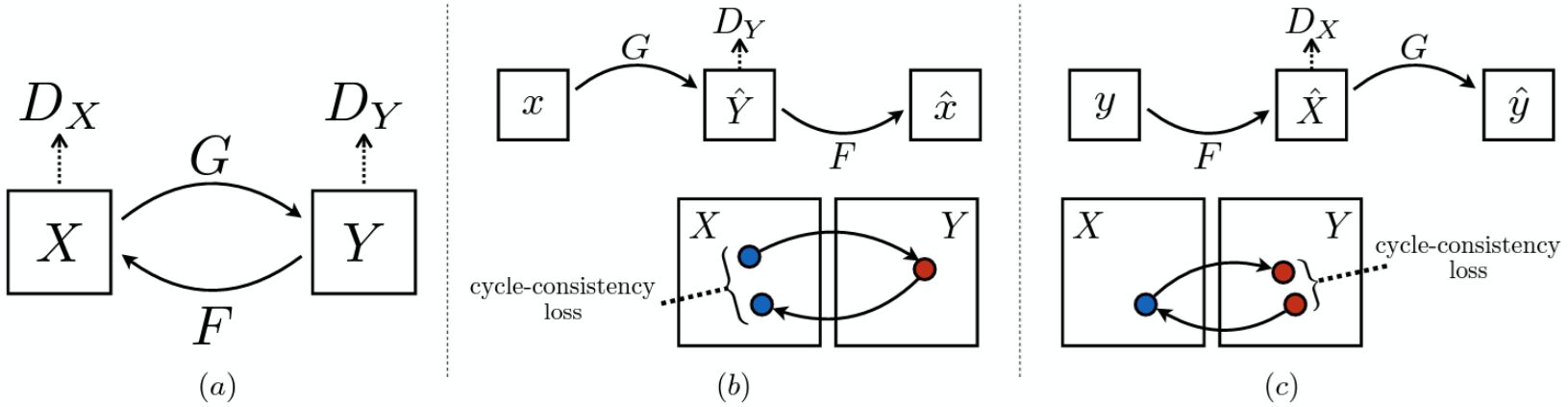


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

MATGANIP learns structure-property relationships in perovskites

Gao et al. (2019) “MATGANIP”

- Uses GAN to learn real vs fake property data to eventually “generate” labels that could be close to real

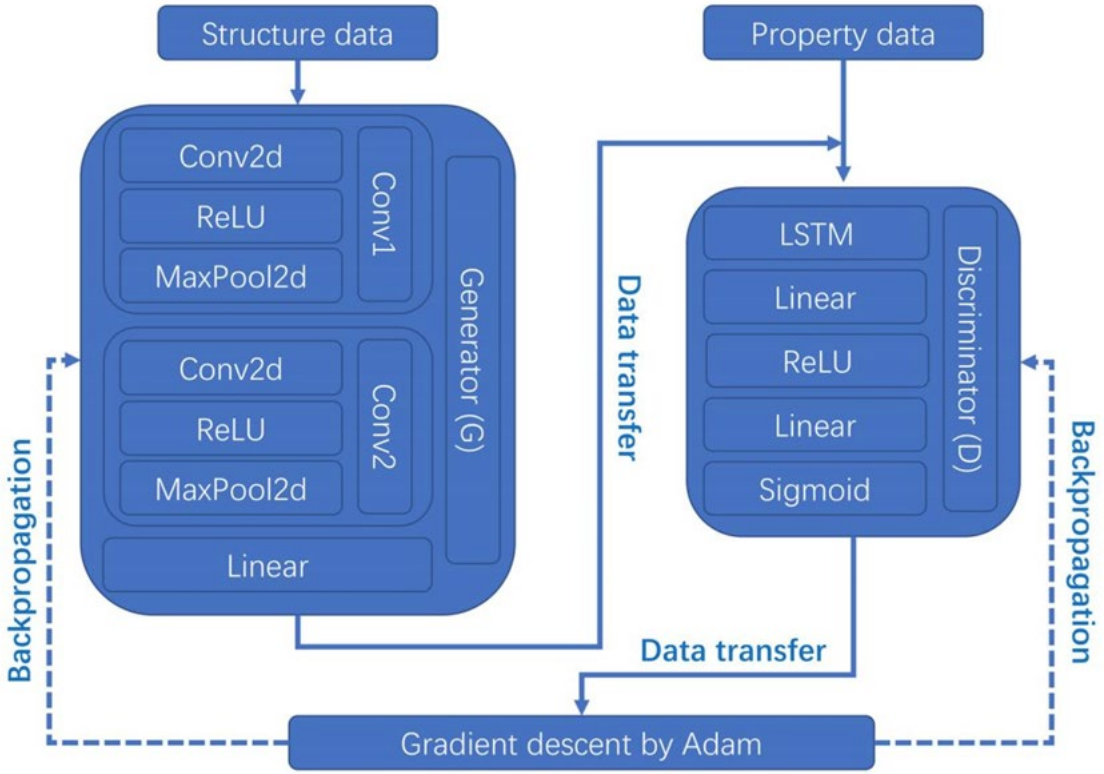


Figure 1: The schema diagram of MATGANIP. The structure data and property data are the input of the generator (G) and the discriminator (D) of the MATGANIP, respectively. In the inner of the G, there are the Conv1, Conv2, and Linear. Both of the Conv1 and Conv2 are the CNN layers, which contains the Conv2d, ReLU, and MaxPool2d layers. Linear is fully connected layers and provides a data-transfer from the G to the LSTM of the D. In the inner of the D. There are LSTM, Linear, ReLU, Linear, and Sigmoid. LSTM represents the long short term memory and enhances the ability to identify of the distribution of the input data. The sigmoid layer provides a Sigmoid function for the D and makes the output of the D to be a probability value. The other three layers, two Linear and ReLU, transform the data of LSTM into the Sigmoid. ReLU provides an active function. Based on the Adam, gradient descent calculation receives the data of the Sigmoid of the D and generates the backpropagation for the G and D of the MATGANIP.

CubicGAN generates cubic materials

Zhao et al. (2021) “CubicGAN”

- Uses GAN to generate candidate cubic materials
- Trained on DFT data (OQMD)
- Rediscovered most of training data plus many new compounds
- 506 validated with phonon dispersion calcs

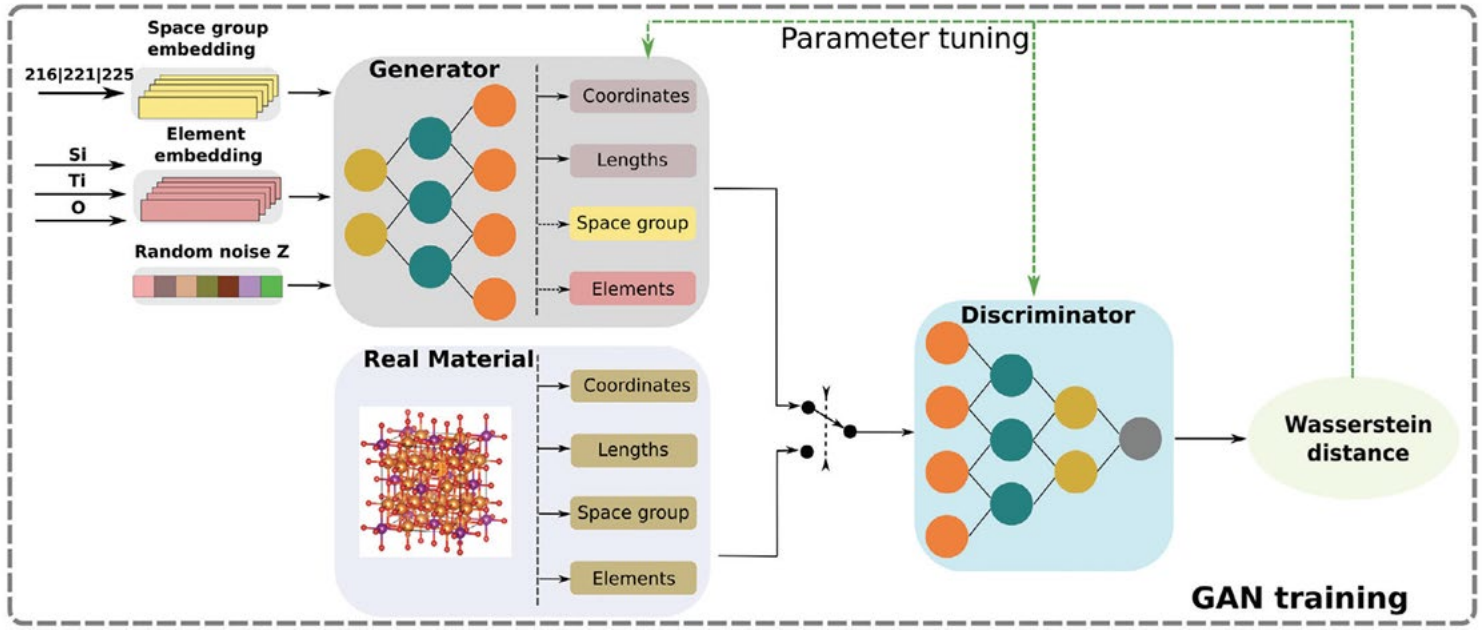


Figure 1. The workflow of our CubicGAN framework.

MatGAN generates compositions

Dan et al. (2020) “MatGAN”

- Uses GAN to generate compositions
- Unusual representation
- Inorganics trained on ICSD, MP, OQMD
- Show that compositional space could be big
- Screened output with SMACT and showed the generated compounds matched

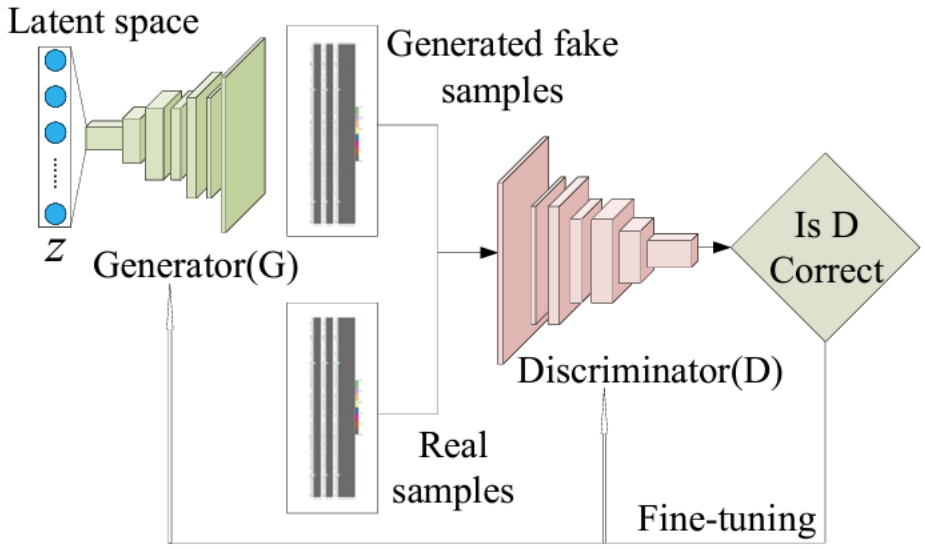
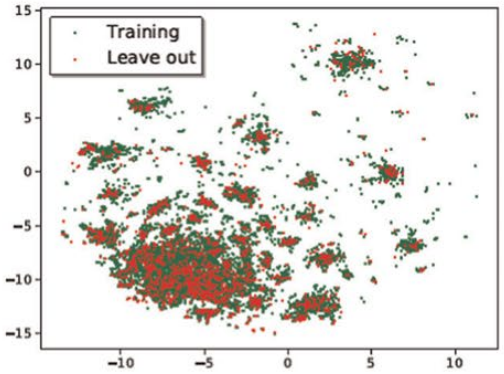
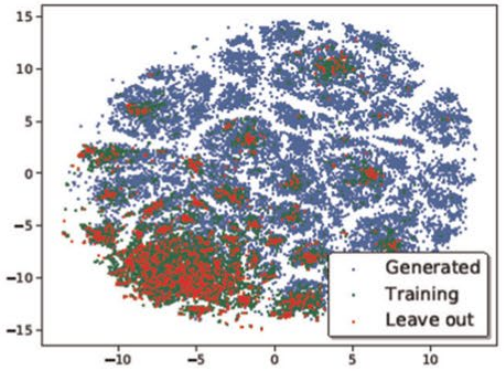
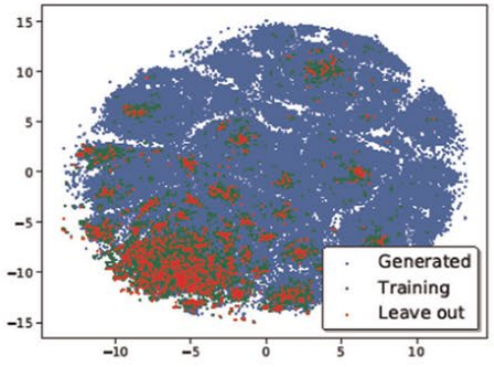


Fig. 1 Architecture of MatGAN for inorganic materials. It is composed of a generator, which maps random vectors into generated samples and a discriminator, which tries to differentiate real materials and generated ones. Detailed configuration parameters are listed in Supplementary Table 1 and Supplementary Fig. 1.



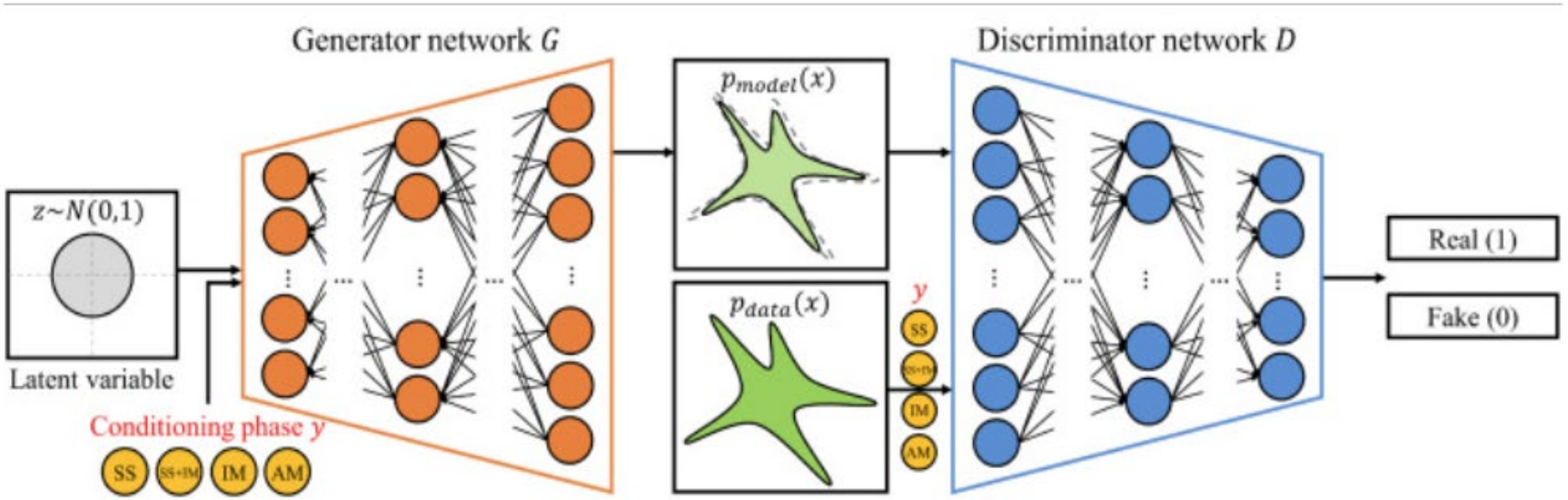




Conditional GANs used to augment data and predict properties

Lee et al. (2021)

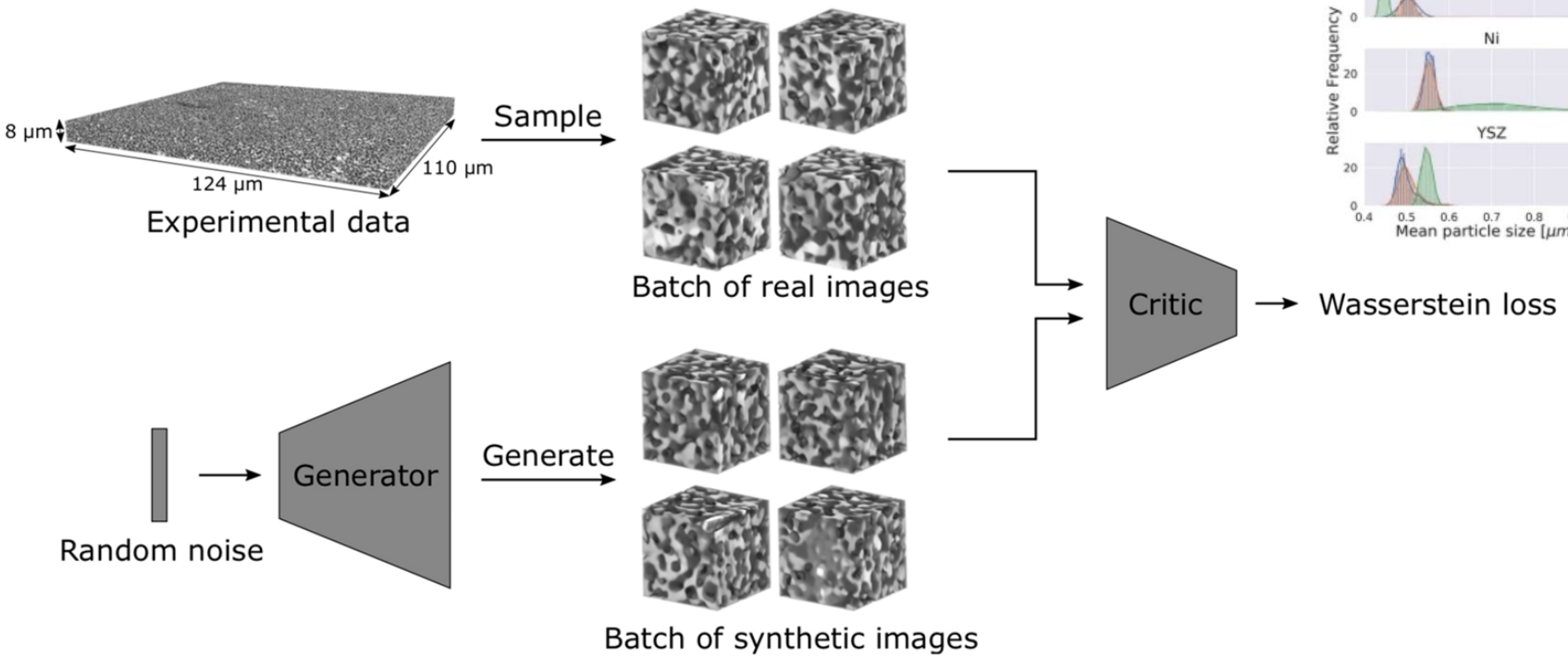
- Uses GAN to predict HEA phases compositions
- Phases get encoded alongside representation
- Used GAN for data augmentation resulting in improved test set performance

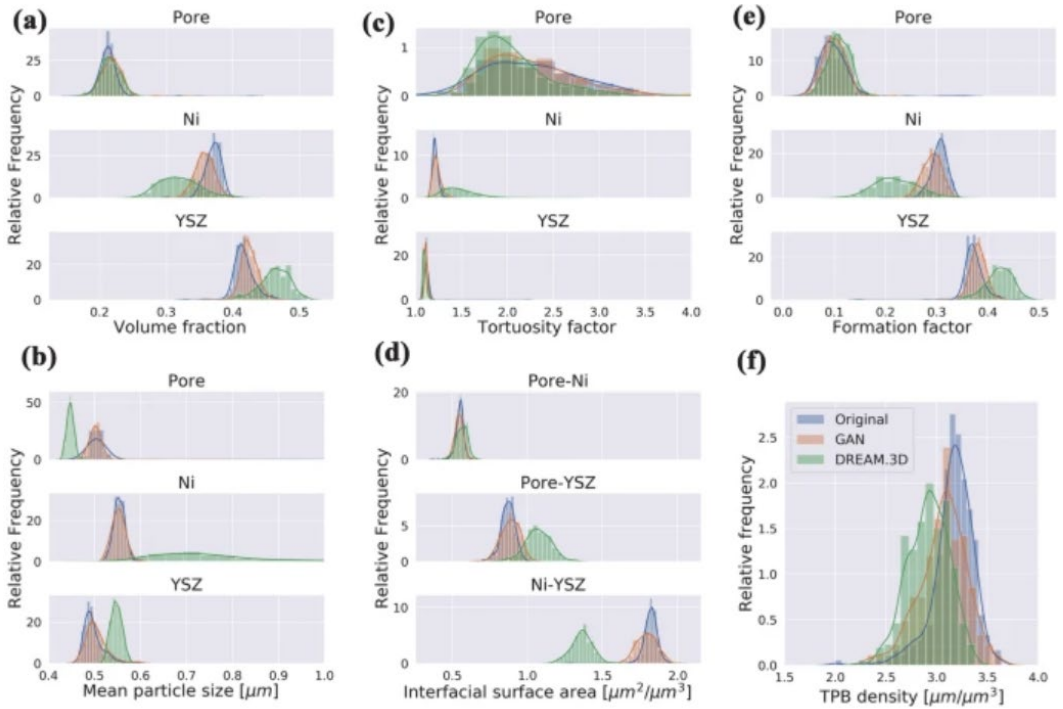


Wasserstein GANs used to create synthetic microstructures

Hsu et al. (2021)

- Train on 3D microstructure data
- WGANs do a great job of matching real data
- DREAM.3D uses grain-based genetic algorithm does worse!

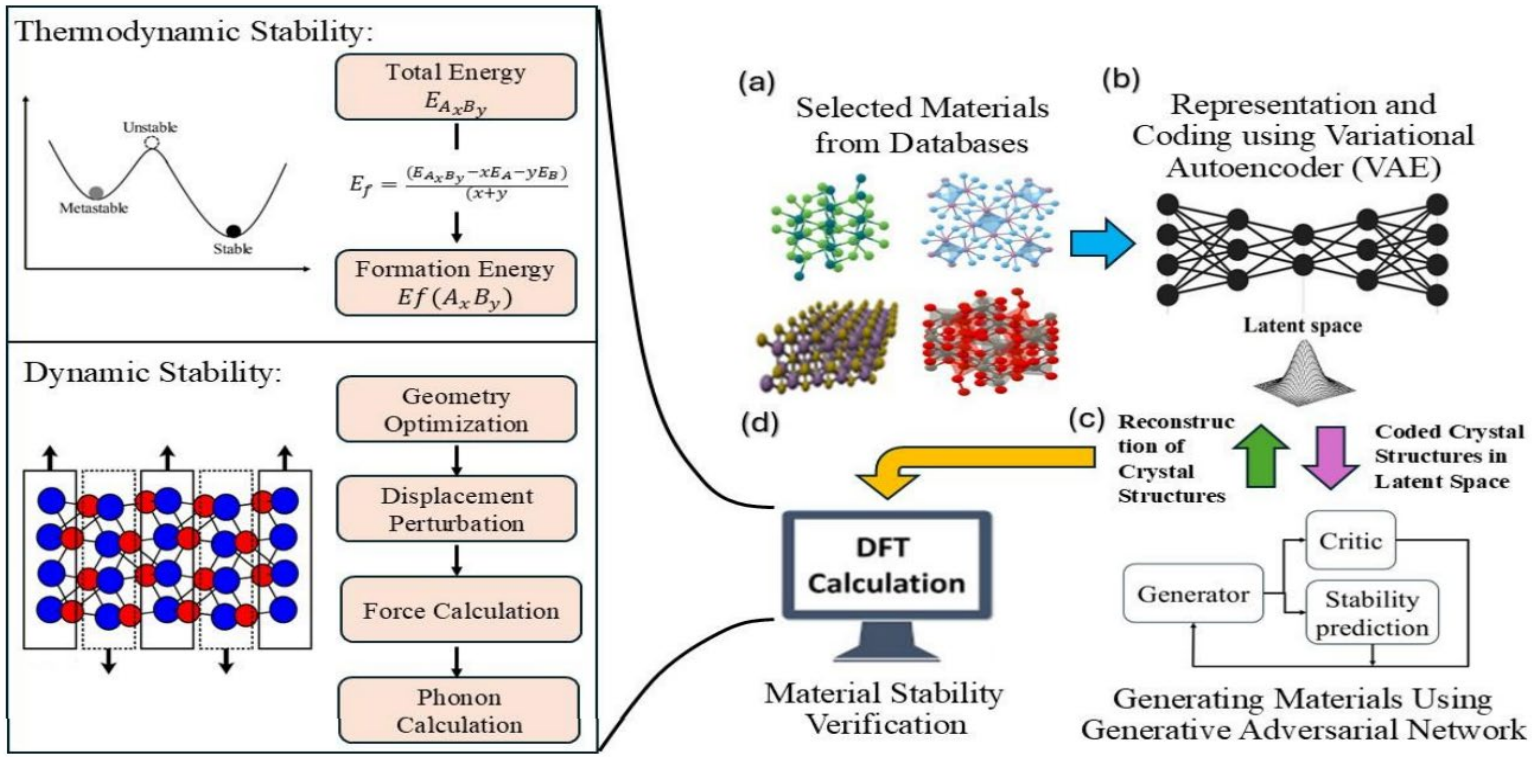




Generation of new crystals with WGAN-VAE

Ebrahimzadeh et al. (2025)

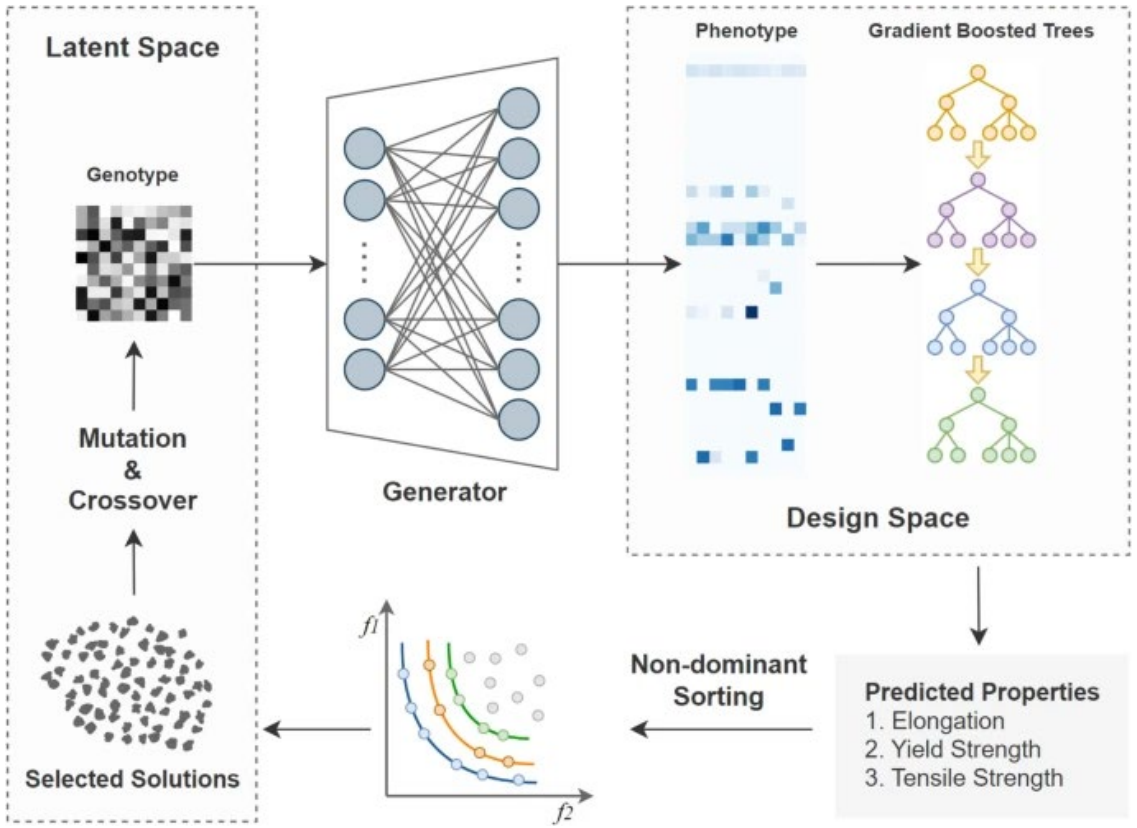
- WGAN generator, VAE latent space
- Stability prediction
- Reconstruction of crystal structure



NSGAN (GAN + genetic algorithm)

Li & Birbilis (2024)

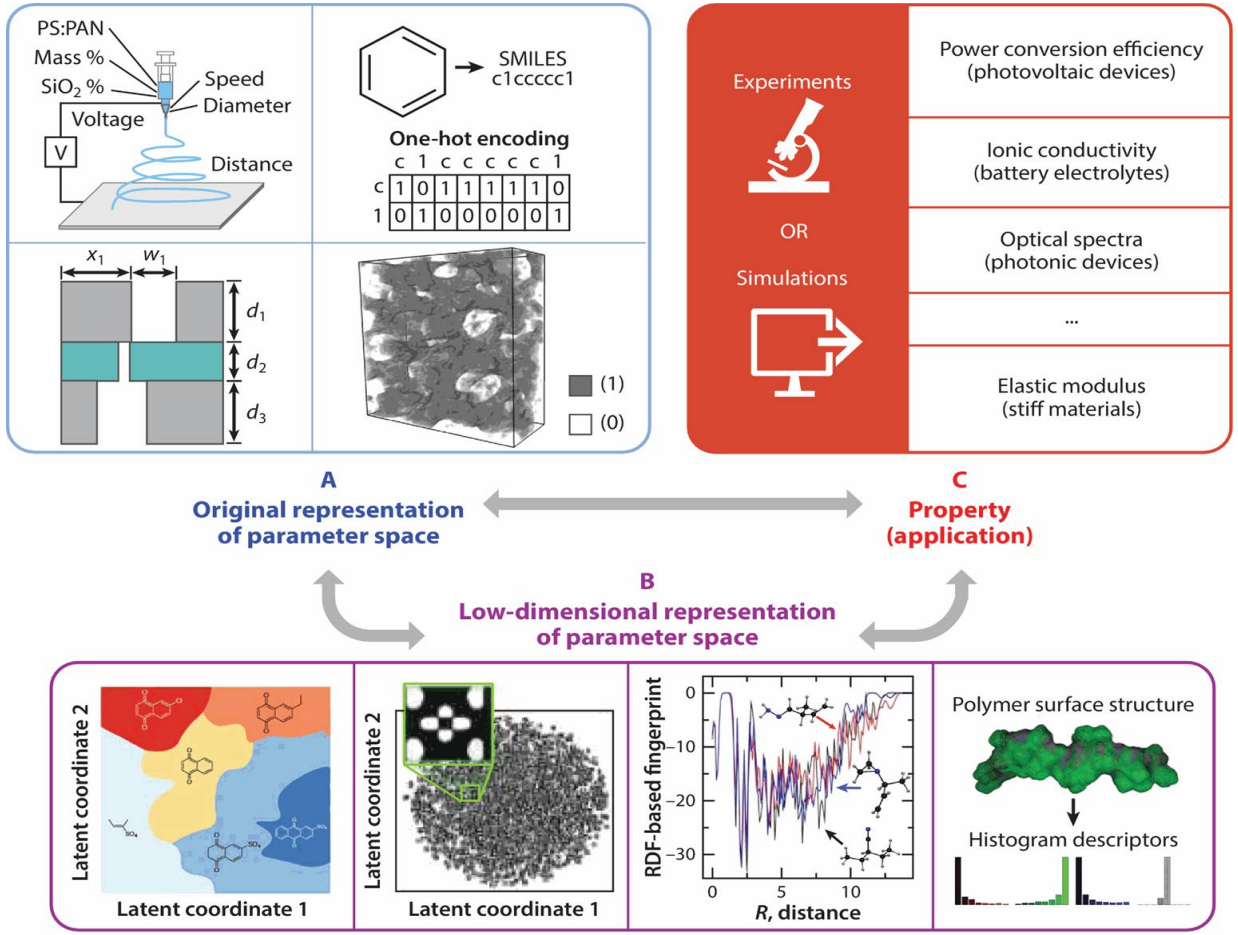
- GAN maps sparse high-dimensional data to simpler distribution.
- GA then searches space without “curse of dimensionality”



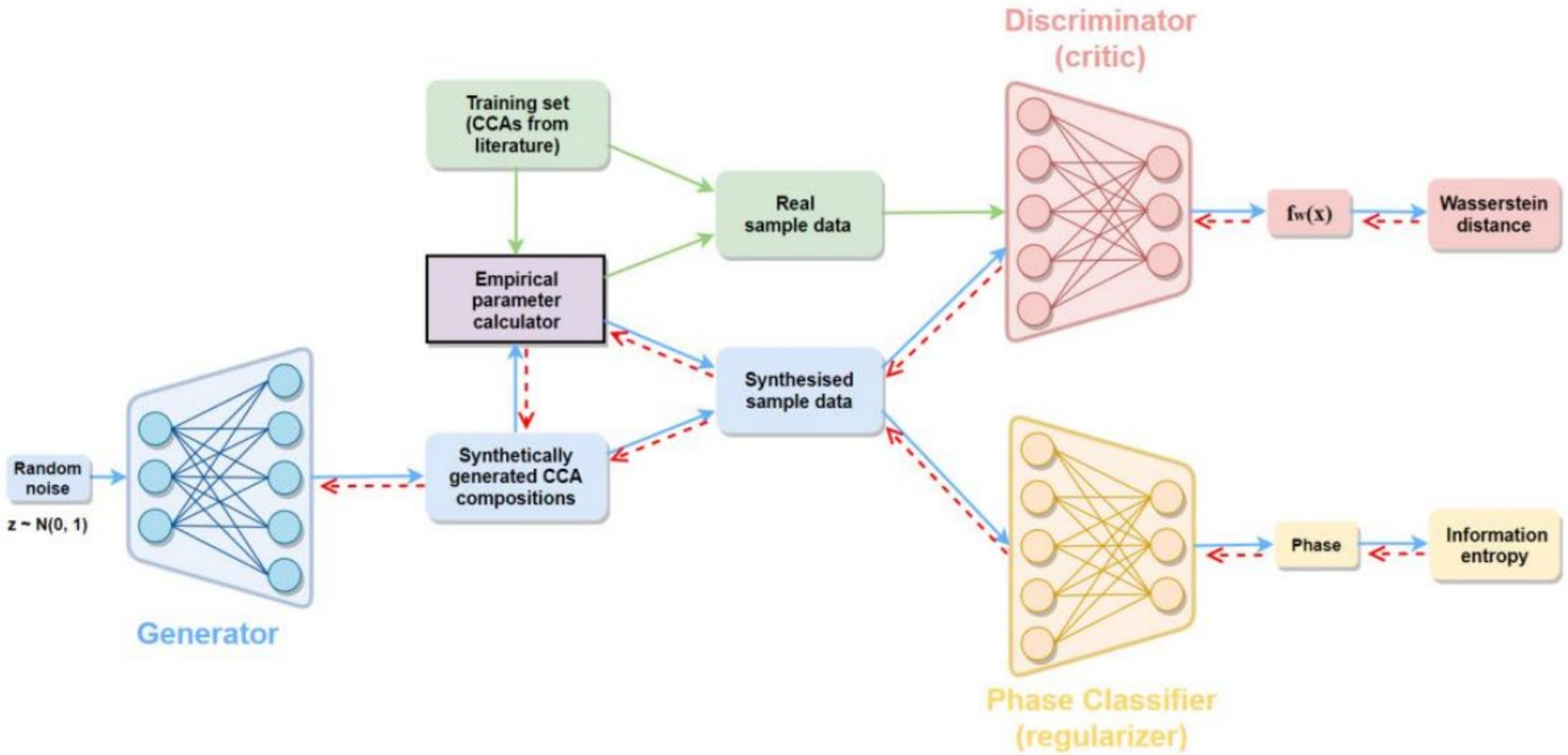
CGAN in high dimensional material design

Kadulkar et al (2022)

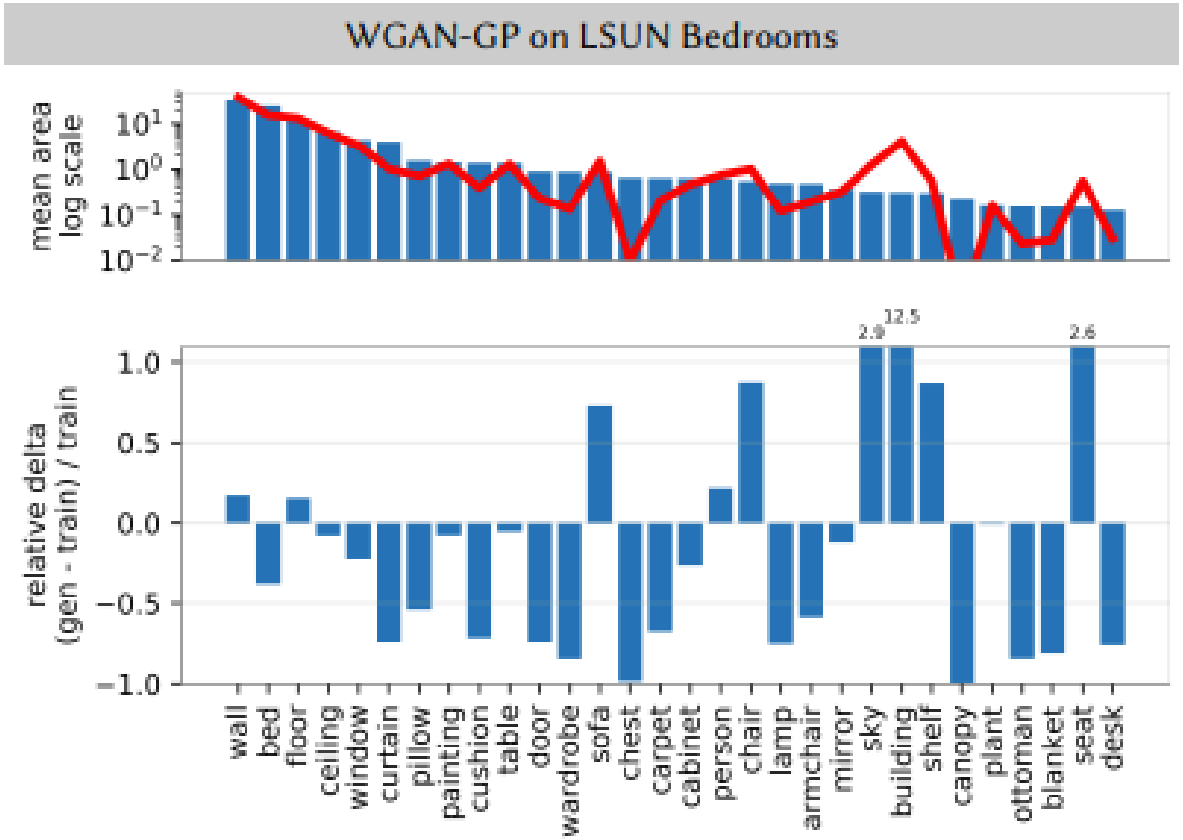
- CGAN steered generation to specific spectra and nanostructure geometries
- Realistic nanostructured trained in low-dimensional space

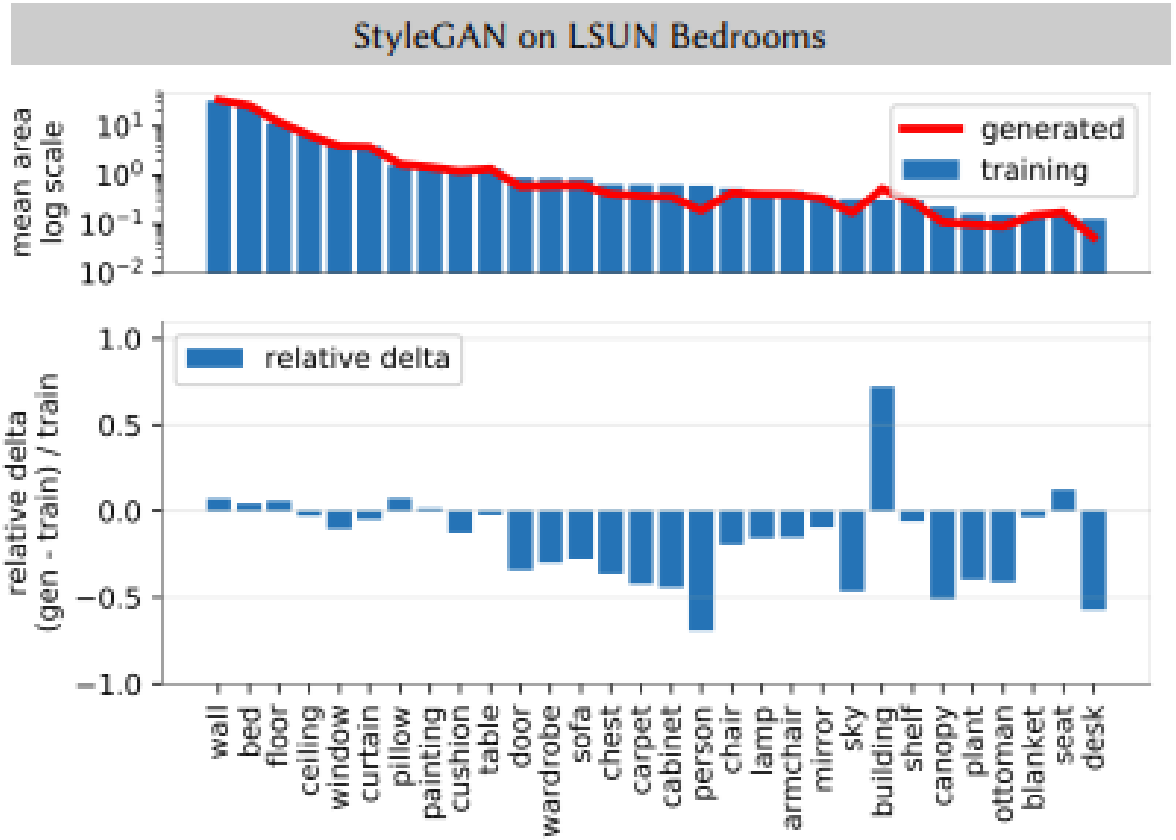


CardiGAN: Compositionally complex alloy design via GAN



Even SOTA GANs can systematically omit entire object classes



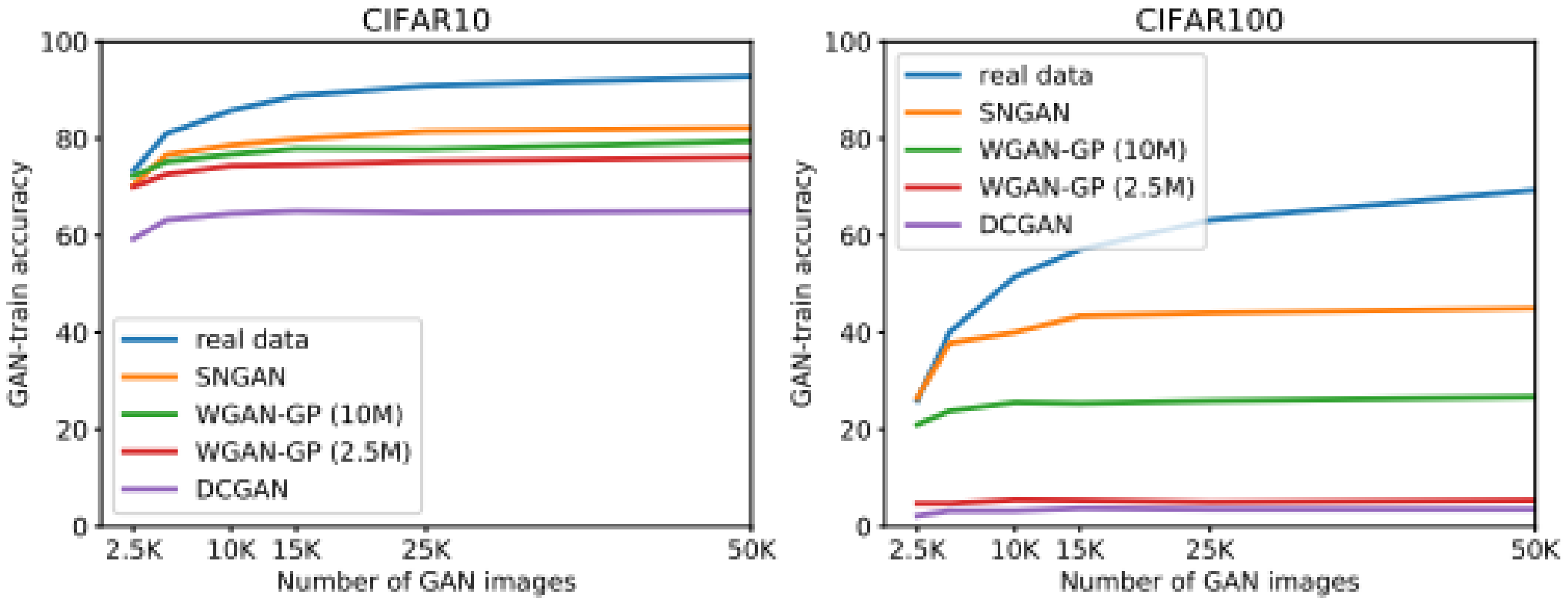


Bau et al 2019

Generate images, then segment and compare distributions of objects

Will microstructures be missing key objects??

Computational cost & accuracy: a tricky trade-off

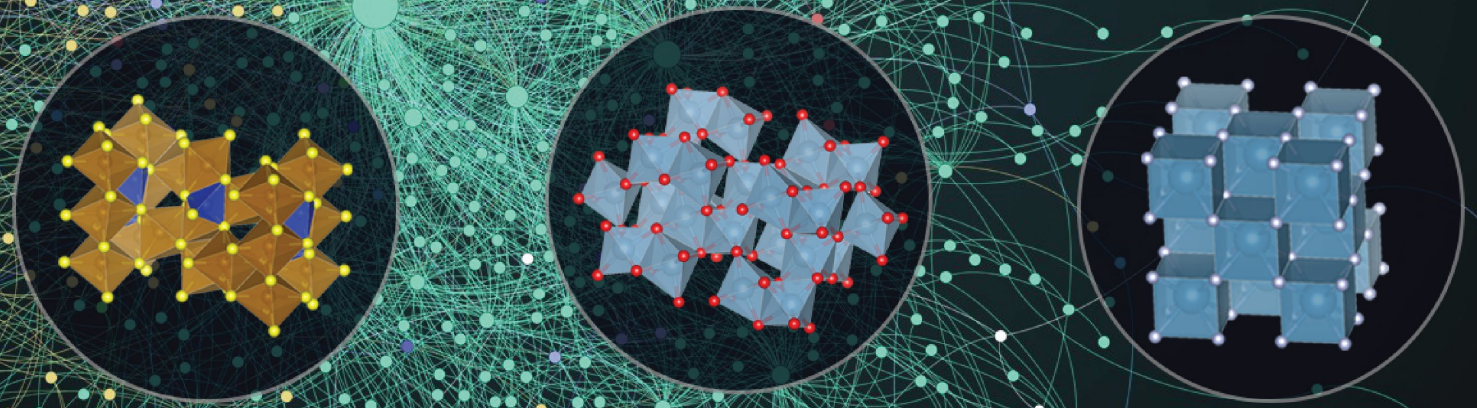


Shmelkov et al 2018

Generate images, then segment and compare distributions of objects

Will microstructures be missing key objects??

Can GANs generate new crystal structures?



Can we use generative models to create full crystal structures?

Issue 1, 2024

Previous Article | Next Article

From the journal:
Digital Discovery

Generative adversarial networks and diffusion models in material discovery

Michael Alverson, *[ac] Sterling G. Baird,[a] Ryan Murdock,[a] (Enoch) Sin-Hang Ho,[b] Jeremy Johnson [b] and Taylor D. Sparks [a]

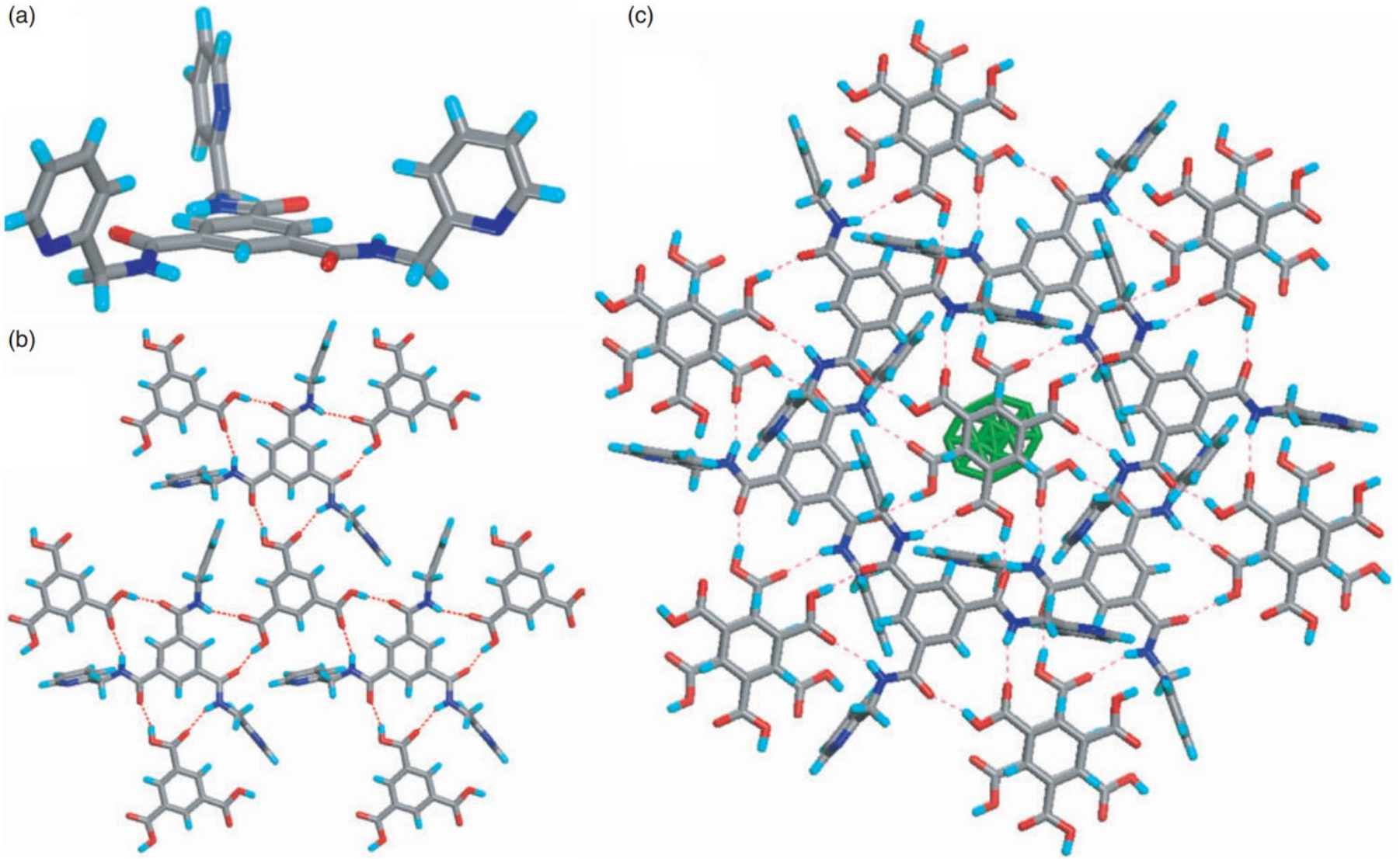
Author affiliations

Abstract

The idea of materials discovery has excited and perplexed research scientists for centuries. Several different methods have been employed to find new types of materials, ranging from the arbitrary replacement of atoms in a crystal structure to advanced machine learning methods for predicting entirely new crystal structures. In this work, we pursue three primary objectives. (I) Introduce CrysTens, a crystal encoding that can be used in a wide variety of deep learning generative models. (II) Investigate and analyze the relative performance of Generative Adversarial Networks (GANs) and Diffusion Models to find an innovative and effective way of generating theoretical crystal structures that are synthesizable and stable. (III) Show that the models that have a better "understanding" of the structure of CrysTens produce more symmetrical and realistic crystals and exhibit a better apprehension of the dataset as a whole. We accomplish these objectives using over fifty thousand Crystallographic Information Files (CIFs) from Pearson's Crystal Database.

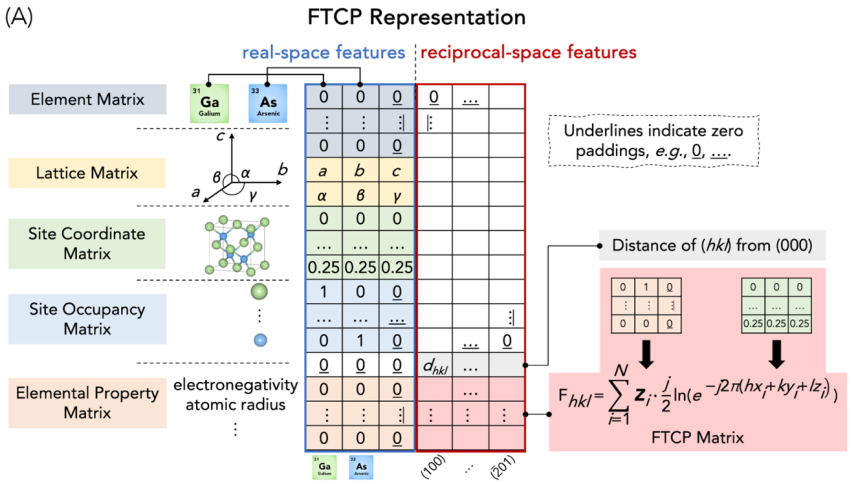
Alverson et al. "Generative adversarial networks and diffusion models in material discovery" Digital Discover (2024)

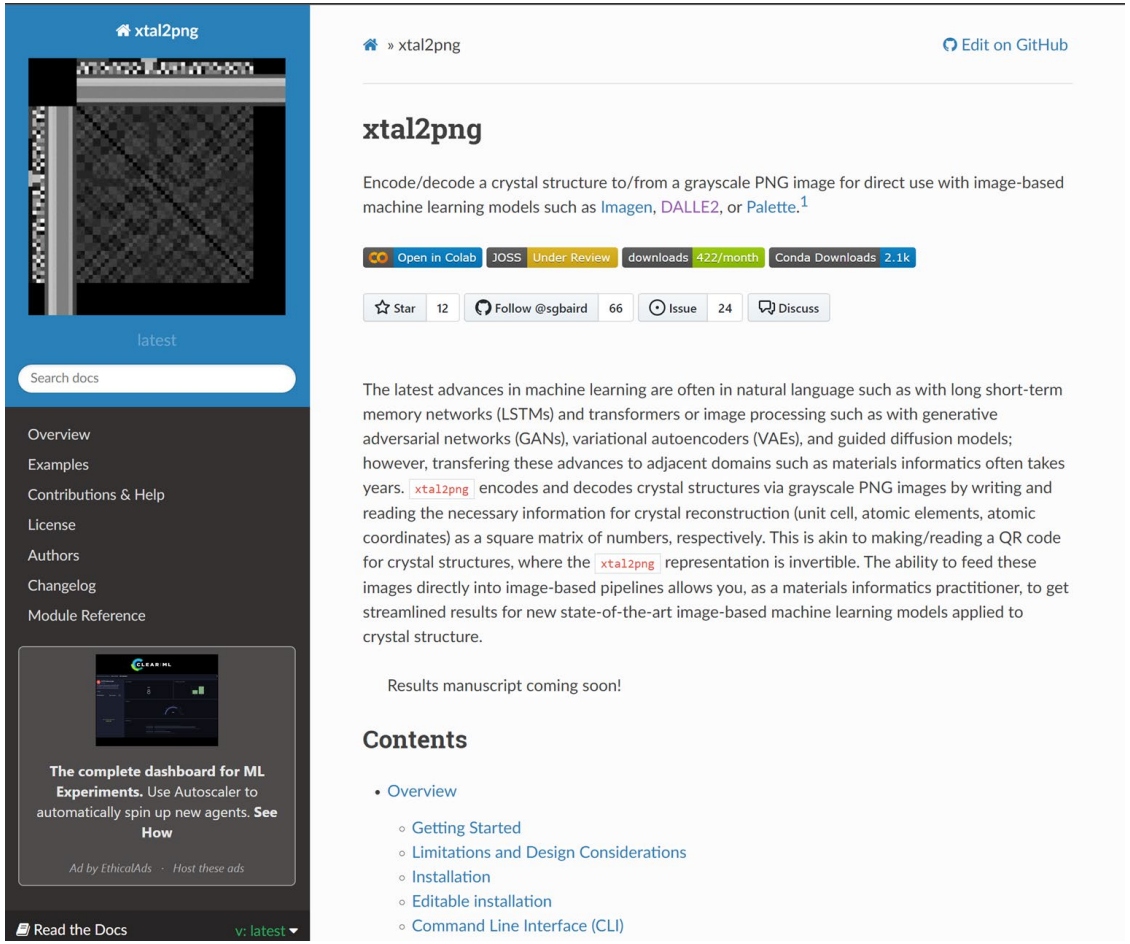
Solid-state chem has lagged organic chemistry here for years



Rajput et al, Crystal Engineering Studies on Ionic Crystals of Pyridine and Carboxylic Acid Derivatives Containing Amide Functional Groups, Aust. J. Chem (2010)

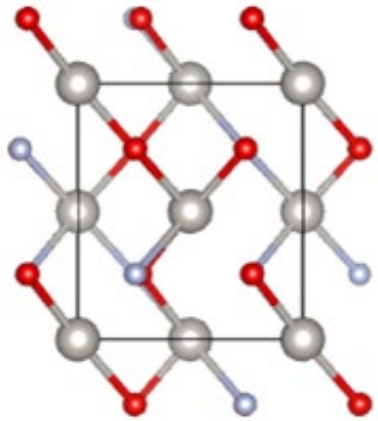
Encoding cif info into representations has been key for crystalline materials

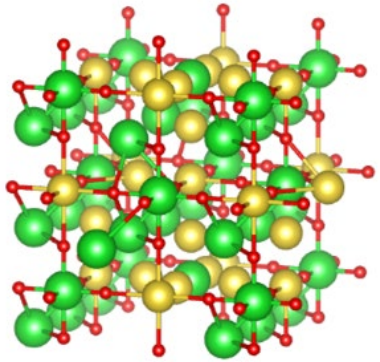


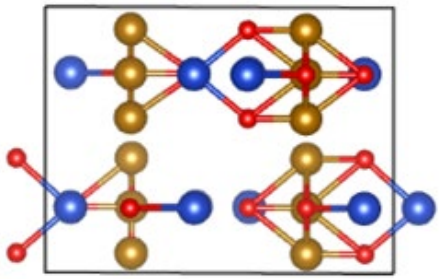


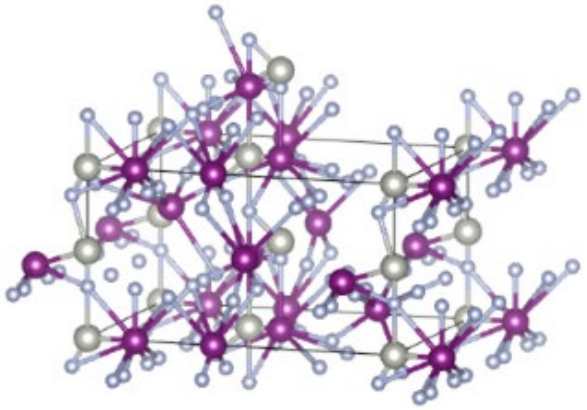
Ren et al, An invertible crystallographic representation for general inverse design of inorganic crystals with targeted properties, Matter 2022,
Baird et al, xtal2png: A Python package for representing crystal structure as PNG files, JOSS 2022

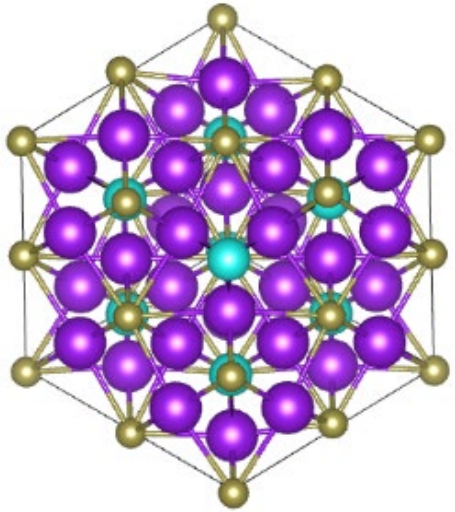
Generative models can now create increasingly convincing new structures

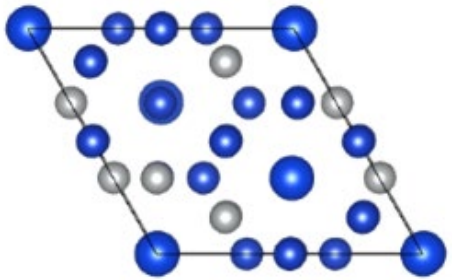






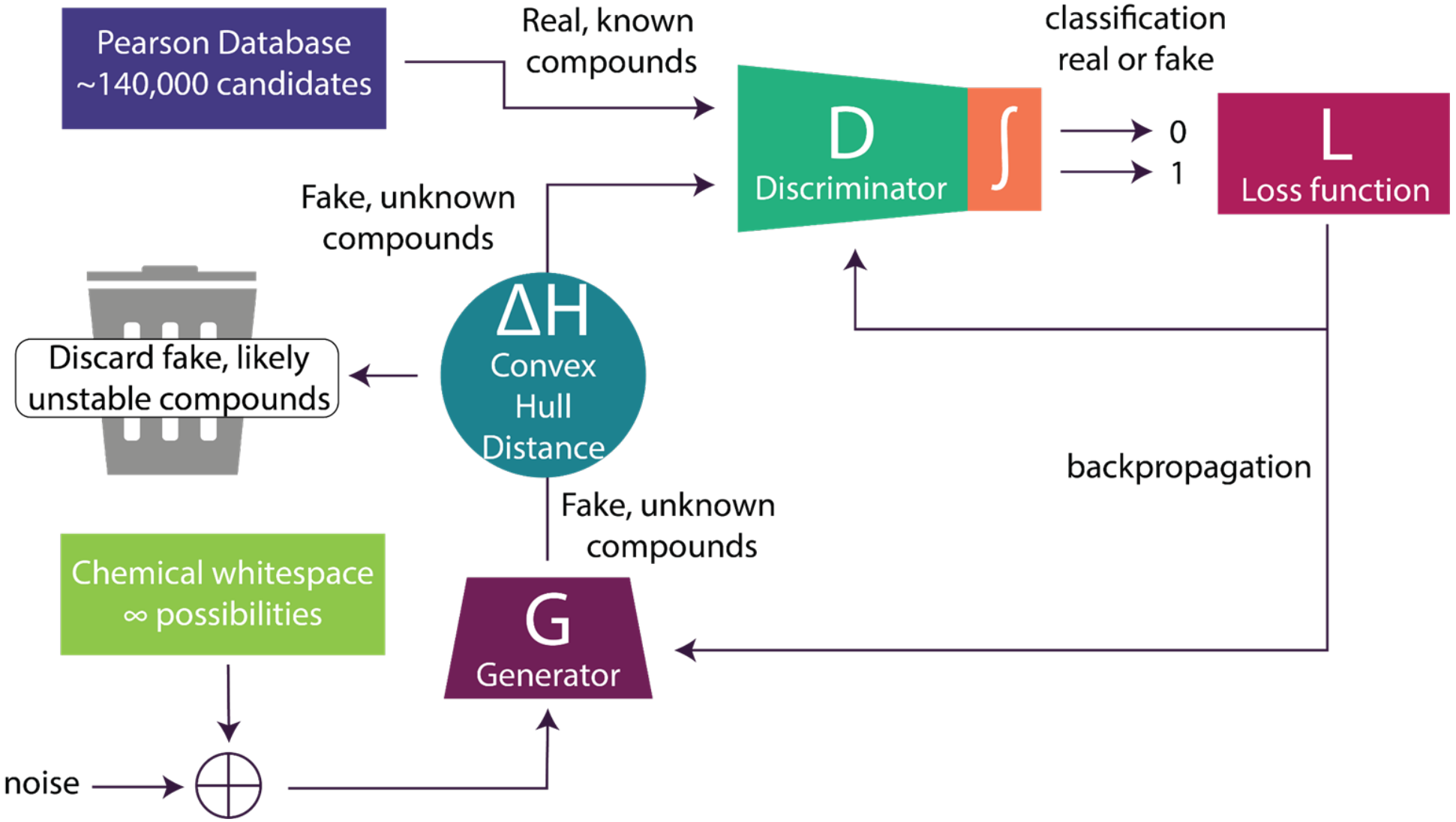






Alverson et al. Generative adversarial networks and diffusion models in material discovery, ChemRxiv 2022

We initially used a modified GAN architecture

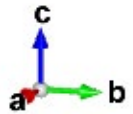


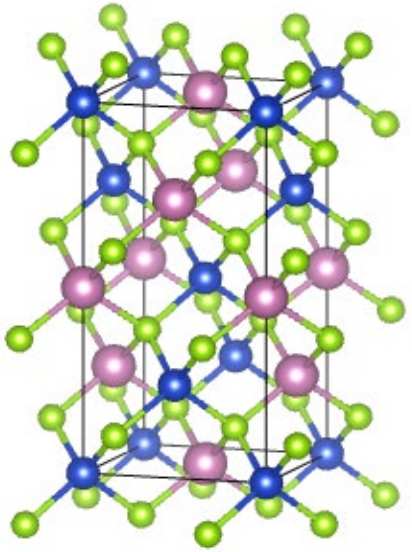
The necessary cif information is pretty simple

```
_cell_length_a                  5.7828
_cell_length_b                  5.7828
_cell_length_c                  11.6207
_cell_angle_alpha               90
_cell_angle_beta                90
_cell_angle_gamma               90
_cell_volume                    388.6
_cell_formula_units_Z           4
 space_group_IT_number          122
_space_group_name_H-M_alt       'I -4 2 d'
```

```
Se Se 8 d 0.2344 0.25 0.125 1
In In 4 b 0 0 0.5 1
Cu Cu 4 a 0 0 0 1
```

VESTA





Tools like pymatgen make it easy to extract

```
Structure Summary
Lattice
    abc : 5.7828 5.7828 11.6207
 angles : 90.0 90.0 90.0
 volume : 388.605223803888
      A : 5.7828 0.0 3.540943755054657e-16
      B : 9.299451658550054e-16 5.7828 3.540943755054657e-16
      C : 0.0 0.0 11.6207
PeriodicSite: In (0.0000, 0.0000, 5.8103) [0.0000, 0.0000, 0.5000]
PeriodicSite: In (0.0000, 2.8914, 8.7155) [0.0000, 0.5000, 0.7500]
PeriodicSite: In (2.8914, 2.8914, 0.0000) [0.5000, 0.5000, 0.0000]
PeriodicSite: In (2.8914, 0.0000, 2.9052) [0.5000, 0.0000, 0.2500]
PeriodicSite: Cu (0.0000, 0.0000, 0.0000) [0.0000, 0.0000, 0.0000]
PeriodicSite: Cu (0.0000, 2.8914, 2.9052) [0.0000, 0.5000, 0.2500]
PeriodicSite: Cu (2.8914, 2.8914, 5.8103) [0.5000, 0.5000, 0.5000]
PeriodicSite: Cu (2.8914, 0.0000, 8.7155) [0.5000, 0.0000, 0.7500]
PeriodicSite: Se (1.3555, 1.4457, 1.4526) [0.2344, 0.2500, 0.1250]
PeriodicSite: Se (4.4273, 4.3371, 1.4526) [0.7656, 0.7500, 0.1250]
PeriodicSite: Se (4.3371, 1.5359, 4.3578) [0.7500, 0.2656, 0.3750]
PeriodicSite: Se (4.3371, 1.3555, 10.1681) [0.7500, 0.2344, 0.8750]
PeriodicSite: Se (1.4457, 4.4273, 10.1681) [0.2500, 0.7656, 0.8750]
PeriodicSite: Se (1.4457, 4.2469, 4.3578) [0.2500, 0.7344, 0.3750]
PeriodicSite: Se (4.2469, 4.3371, 7.2629) [0.7344, 0.7500, 0.6250]
PeriodicSite: Se (1.5359, 1.4457, 7.2629) [0.2656, 0.2500, 0.6250]
```

Some data cleaning was necessary

CRYSTAL IMPACT

Software for Scientists

Register Contact Search

About us | Diamond | Endeavour | Match! | Pearson's CD

About Pearson's CD

Data Information...
Software Functions...
Features...
Brochure (PDF)...
References...

Get Pearson's CD

Order Now

Demo Version

A **free-of-charge** demo version with a few thousand datasets can be downloaded. More...

Quickstart

A quickstart manual is also available in Spanish. More...

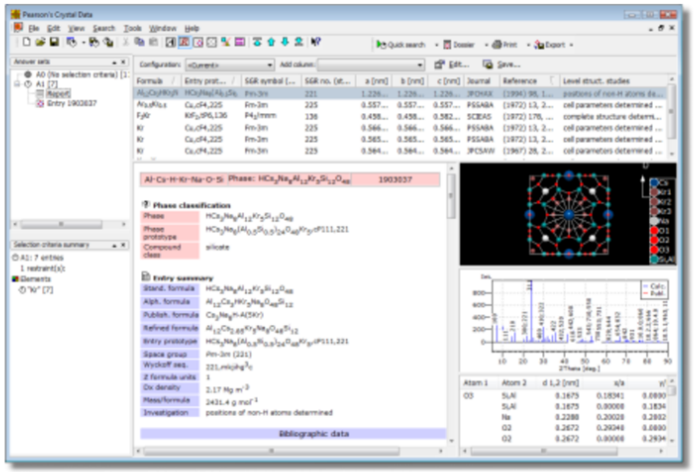
Support

Updates...
Known Bugs...
HowTo...
Frequently Asked Questions...
User Group...

Pearson's Crystal Data

Crystal Structure Database for Inorganic Compounds

Pearson's Crystal Data is a crystallographic database published by ASM International (Materials Park, Ohio, USA), edited by Pierre Villars and Karin Cenzual. It has its roots in the well-known PAULING FILE project and contains crystal structures of a large variety of inorganic materials and compounds. The "PCD" (as it is typically abbreviated) is a collaboration between ASM International and Material Phases Data System, Vitznau, Switzerland (MPDS), aiming to create and maintain the world's largest critically evaluated "Non-organic database".



The current release 2020/21 contains about 350,000 structural data sets (including atom coordinates and displacement parameters, when determined) for about 195,000 different chemical formulas, roughly 20,500 experimental powder diffraction patterns and about 297,000 calculated patterns (interplanar spacings, intensities, Miller indices). In addition over 53,000 figure descriptions for such as cell parameters as a function of temperature, pressure or concentration are given. To reach this result, scientific editors have critically analyzed and processed over 112,500 original publications.

The database comes with an innovative retrieval software for Windows PCs developed by Crystal Impact. It offers a large variety of new elaborate new features which make retrieval of the desired information extremely easy and comfortable.

Pearson's CD News

October 15, 2020
The new **Release 2020/21** of **Pearson's Crystal Data** has just become available, with a total entry count of about **350,000**. More...

September 30, 2010
Release 2019/20 of **Pearson's Crystal Data** has just become available, with a total entry count of about **335,000**. More...

July 31, 2019
Another software **update** is available for both **release 2017/18** and **release 2018/19** of **Pearson's Crystal Data**. More...

April 1, 2019
Another software **update** is available for both **release 2017/18** and **release 2018/19** of **Pearson's Crystal Data**. More...

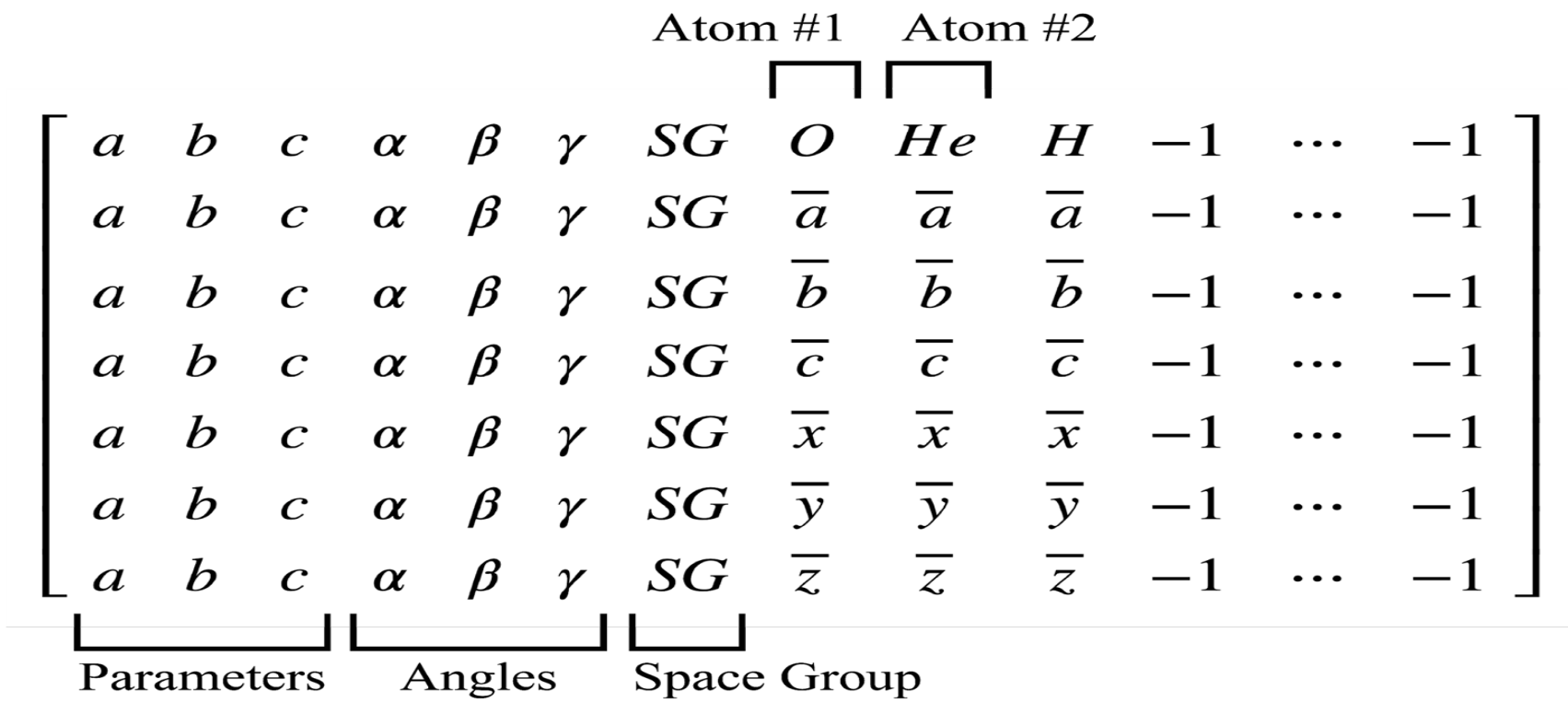
February 7, 2019
A software **update** is available for both **release 2017/18** and **release 2018/19** of **Pearson's Crystal Data**. More...

October 5, 2018
Release 2018/19 of **Pearson's Crystal Data** has just become available, with a total entry count of about **319,000**. More...

~300,000 CIFs in PCD
After cleaning there was 143892 entries retained

Our original data representation was very simple

Structural information was averaged over 7 generated entries



Some constraints were necessary to obey symmetry rules

Obeying Pauling's 5th rule, parsimony, required clustering

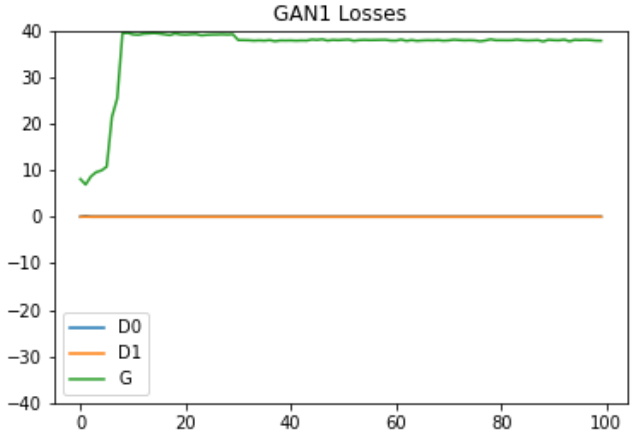
13.4425125
12.922896
11.164495
15.148039
22.145111
20.458208
20.795254
19.797886
14.355895
12.193807
14.700731
13.9915695
11.117646
8.640331
11.288832

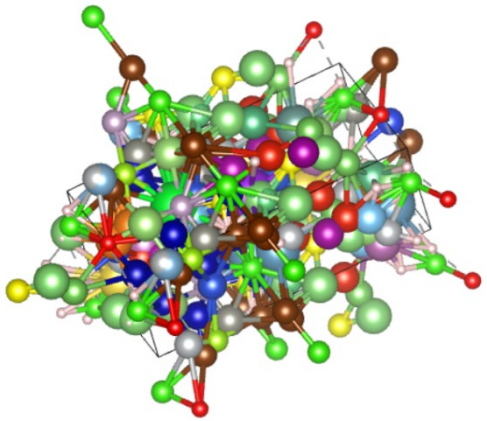
Obeying Pauling's 5th rule, parsimony, required clustering

13.4425125	12.633341	Aluminium	Aluminium
12.922896	12.633341	Aluminium	Aluminium
11.164495	12.633341	Sodium	Aluminium
15.148039	12.633341	Phosphorus	Aluminium
22.145111	20.799114	Titanium	Scandium
20.458208	20.799114	Calcium	Scandium
20.795254 →	20.799114	Scandium →	Scandium
19.797886	20.799114	Calcium	Scandium
14.355895	12.633341	Silicon	Aluminium
12.193807	12.633341	Magnesium	Aluminium
14.700731	12.633341	Phosphorus	Aluminium
13.9915695	12.633341	Silicon	Aluminium
11.117646	12.633341	Sodium	Aluminium
8.640331	12.633341	Fluorine	Aluminium
11.288832	12.633341	Sodium	Aluminium

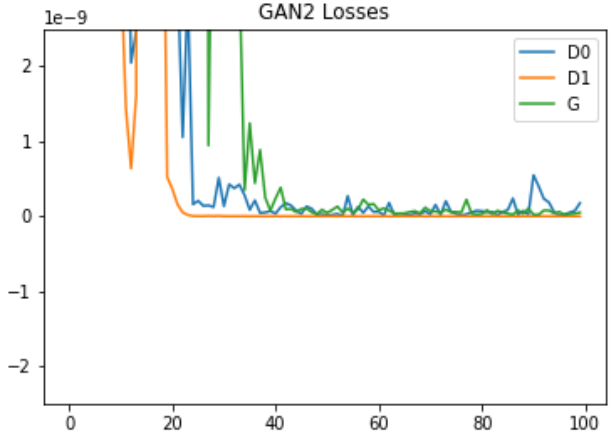
Traditional GANS did not perform well

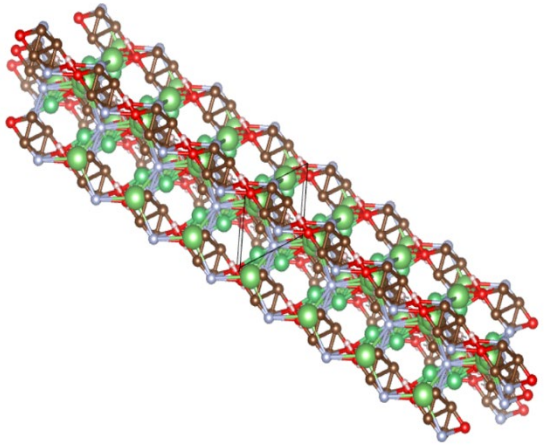
GAN1: Baseline Performance, only convolutions, no normalizations



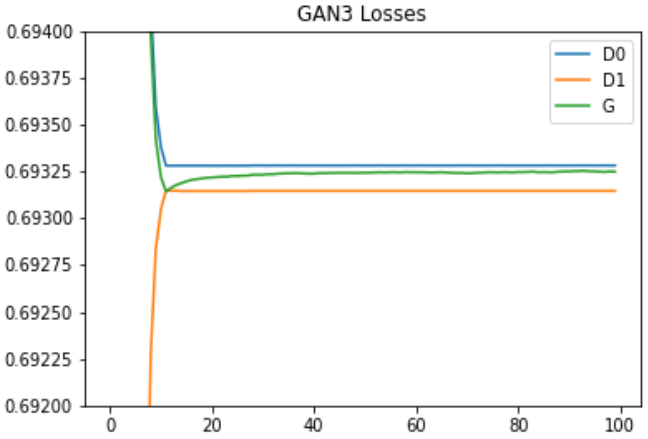


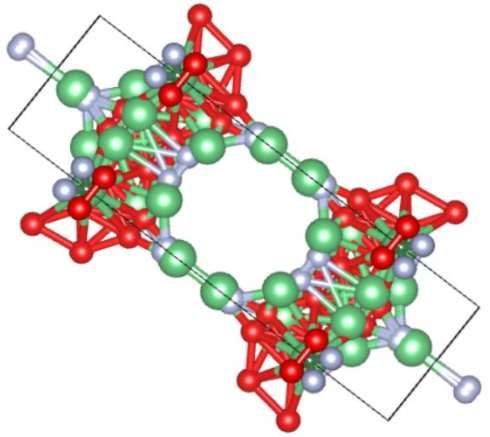
GAN2: Batch normalization, only convolutions





GAN3: Added LSTM layers





Symmetry only representations aren't going to cut it

2014

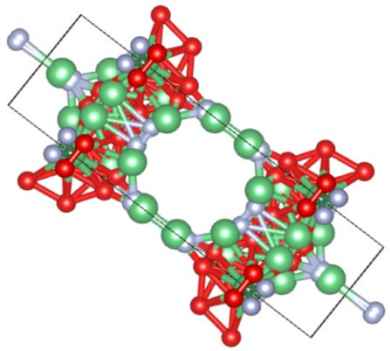
2015

2016

2017

2018

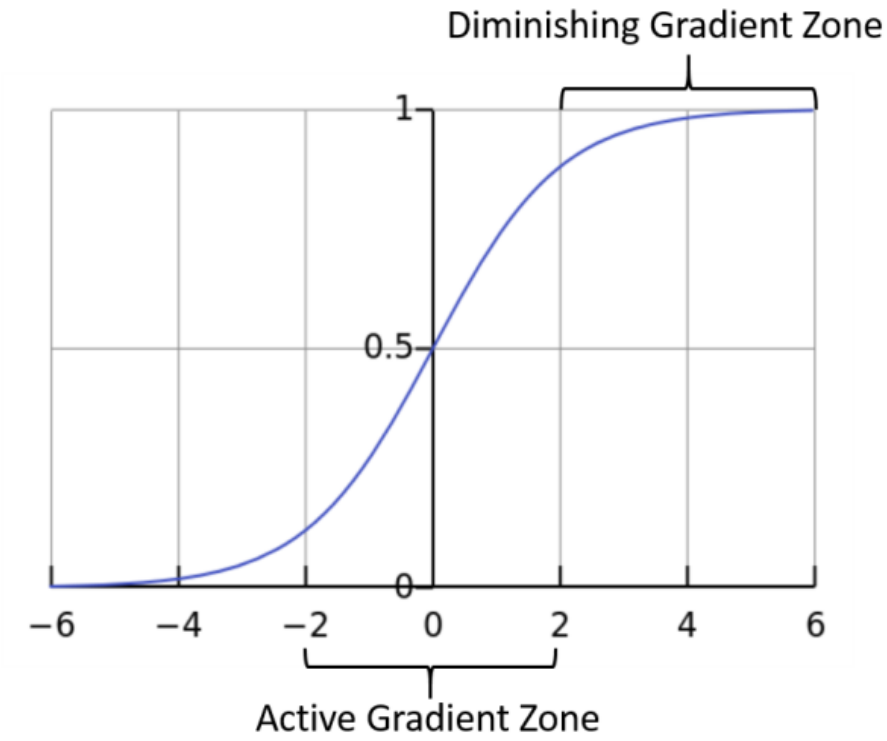
2021



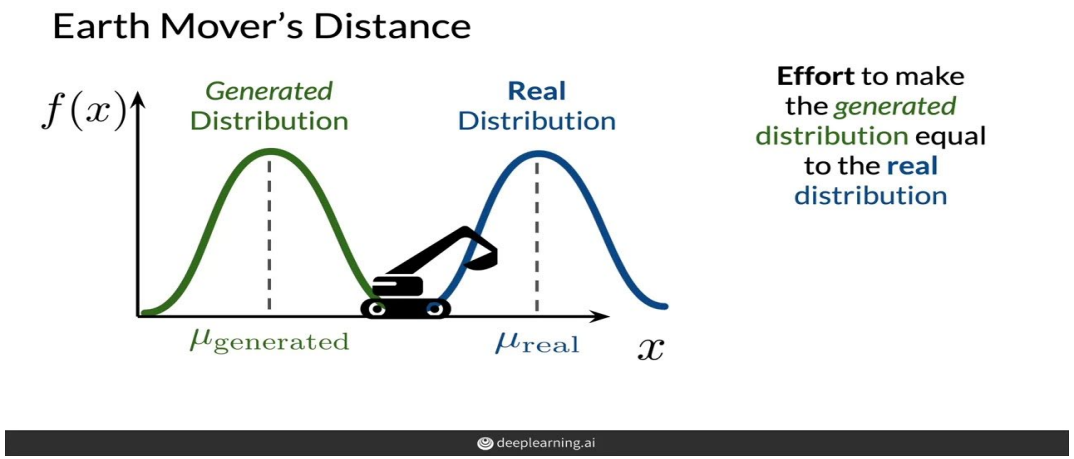
2023

Traditional GANs have some training limitations

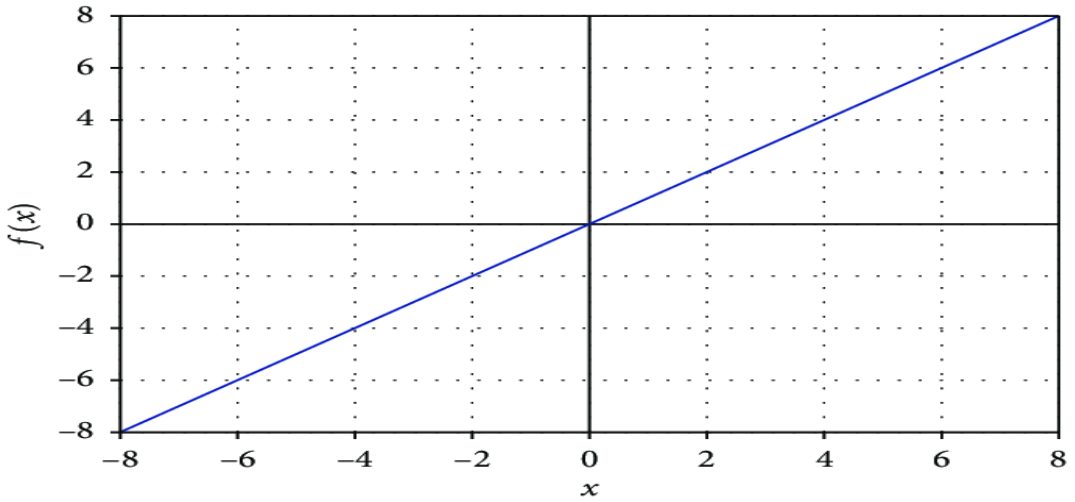
$$A = \frac{1}{1+e^{-x}}$$



Wasserstein GANs offer some solutions



Effort to make the *generated* distribution equal to the **real** distribution

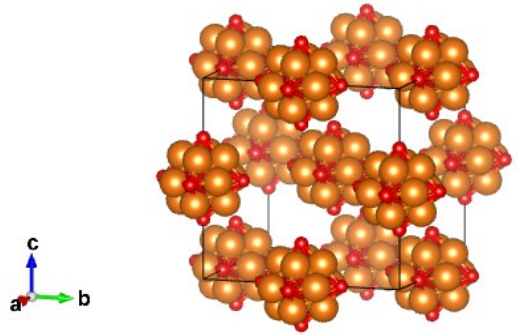


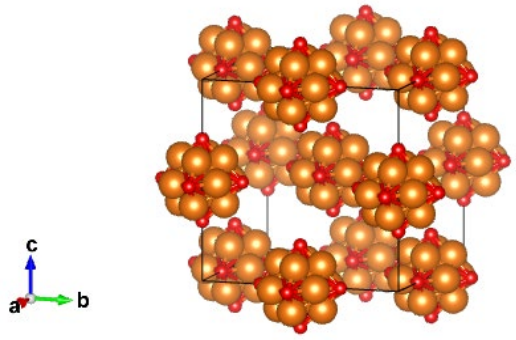
Lipschitz continuity: weight clipping vs gradient penalty

Normalization: batch normalization vs

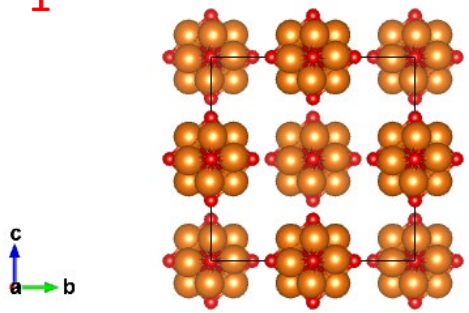
layer normalization vs

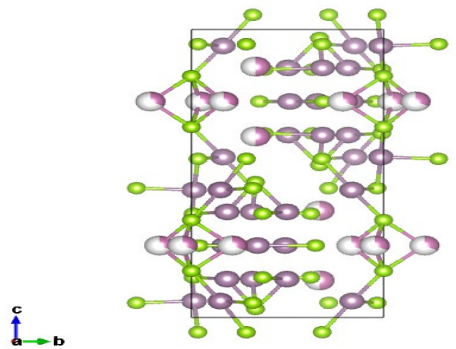
spectral normalization



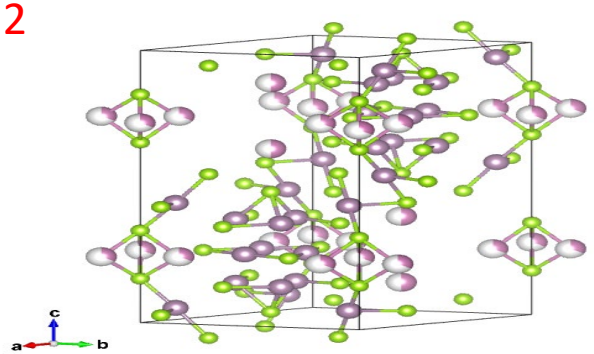


1

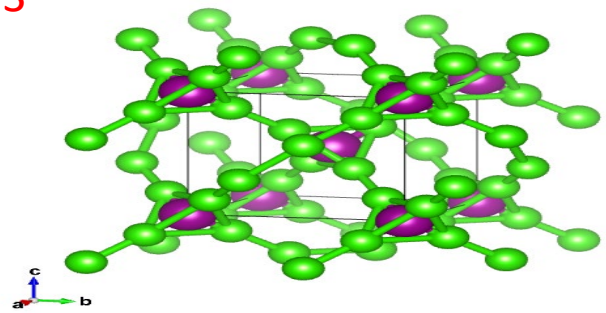


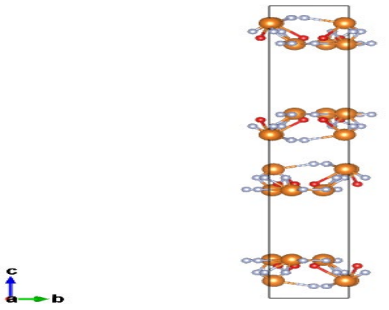


2

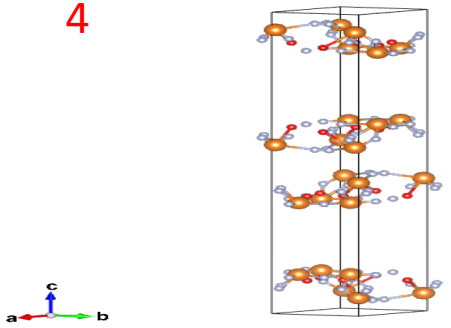


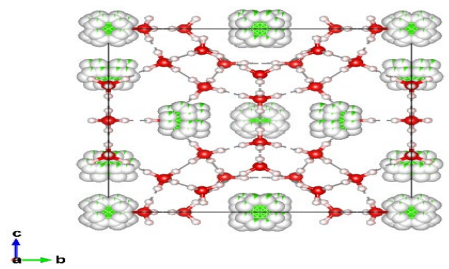
3



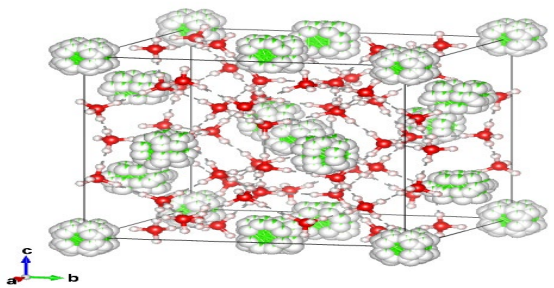


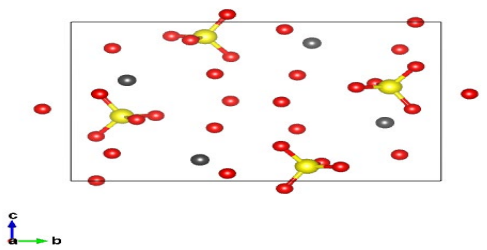
4



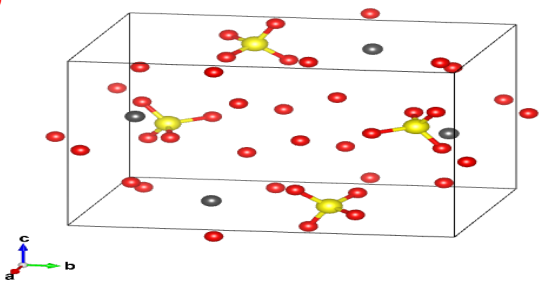


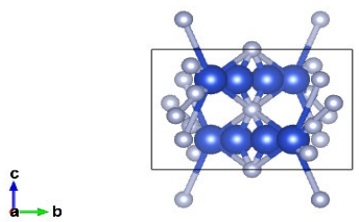
5



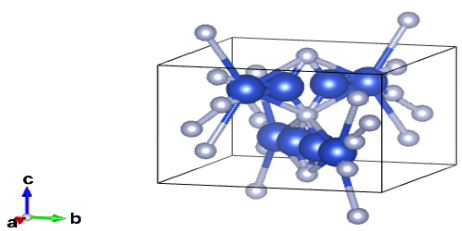


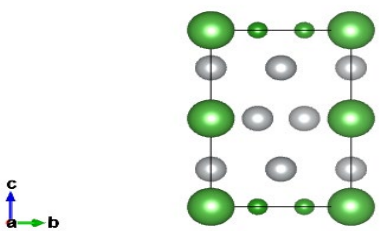
6



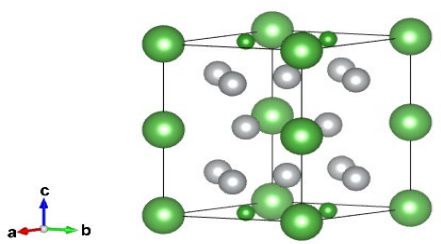


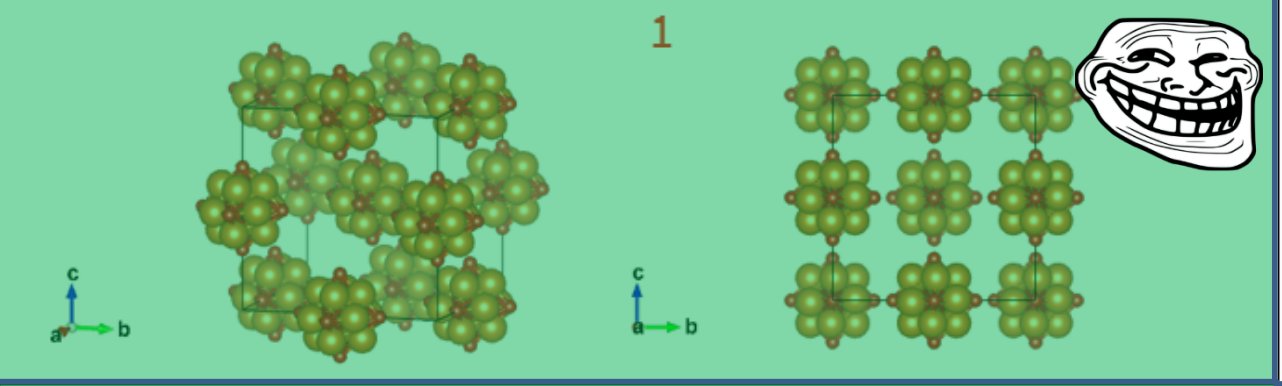
7

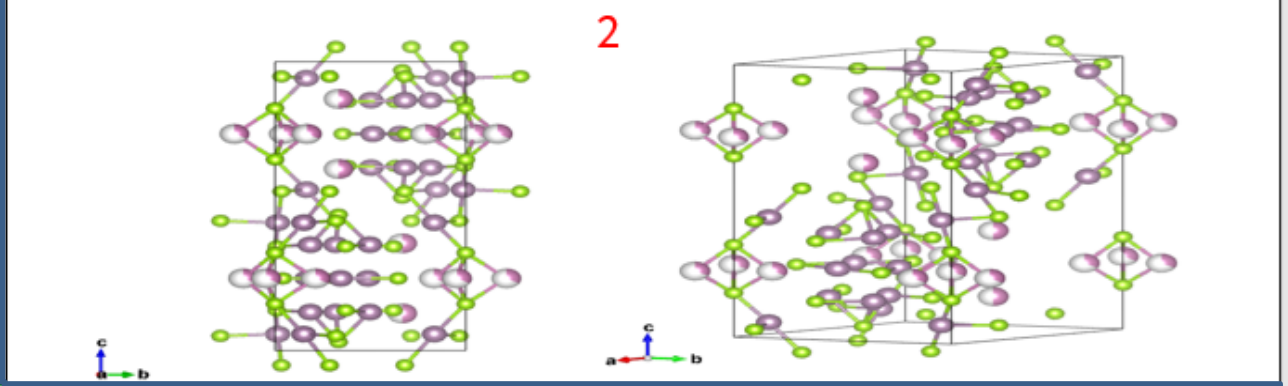


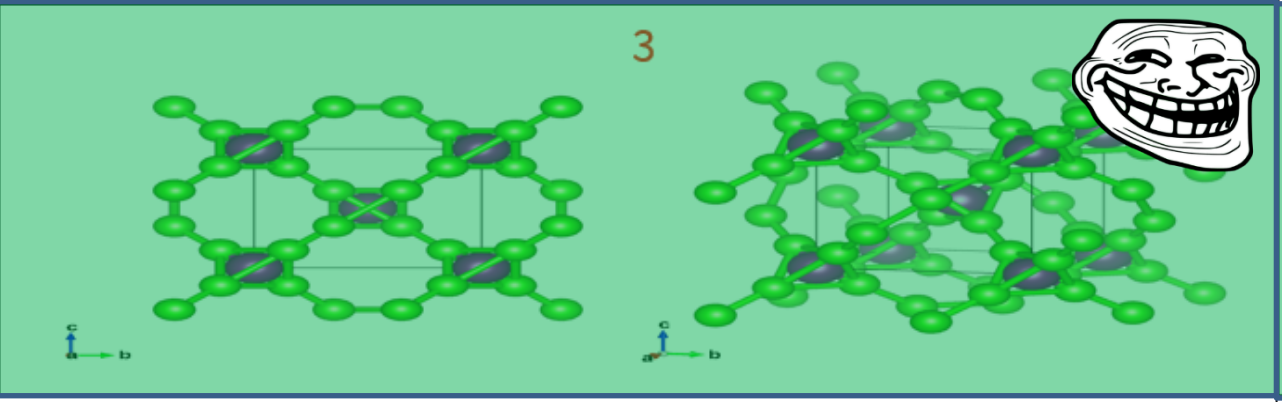


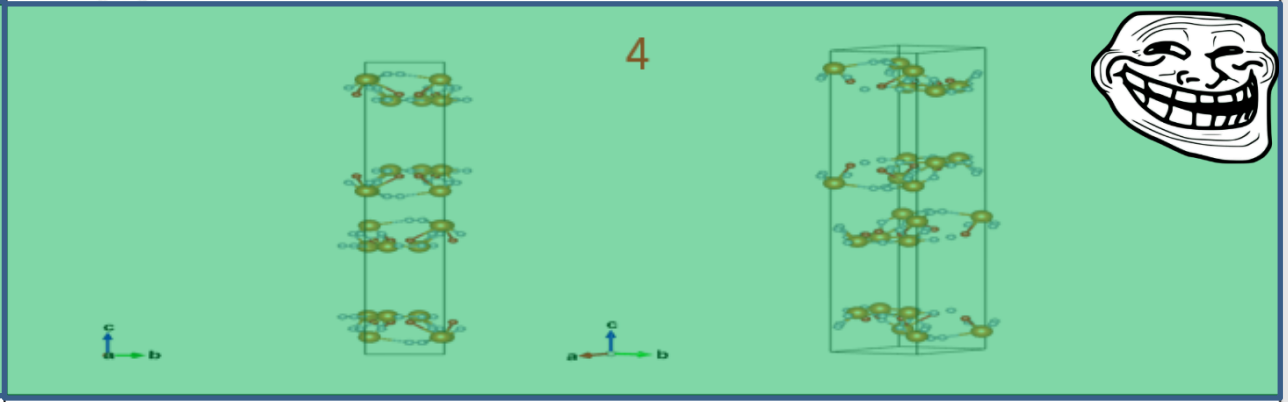
8

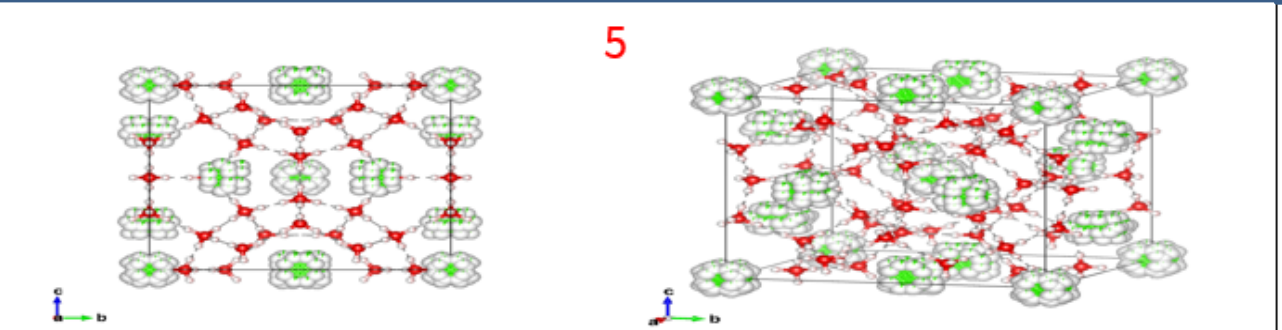


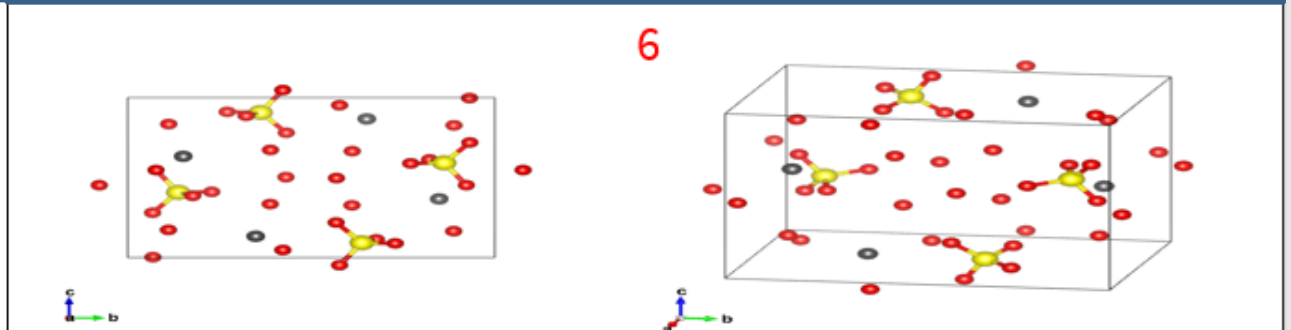

1

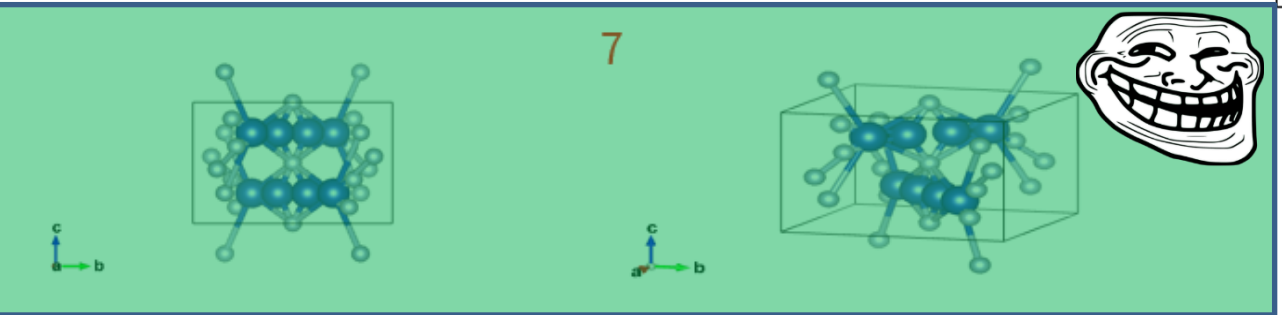

2

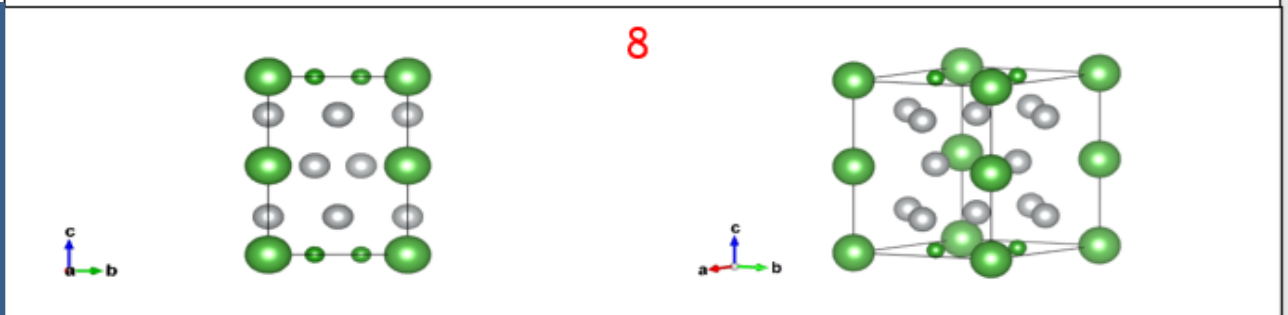

3


4

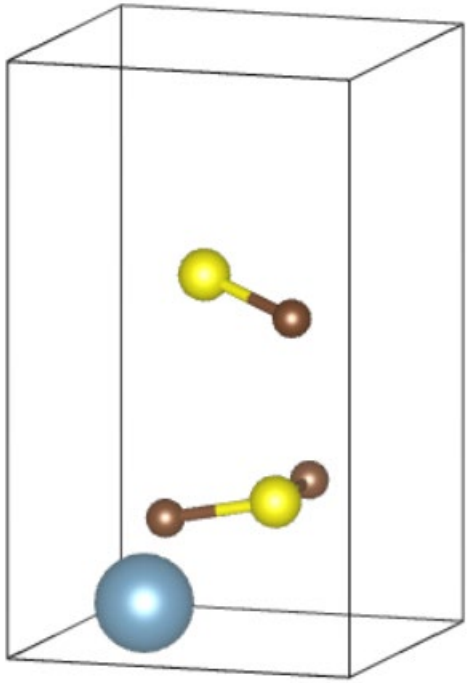

5


6

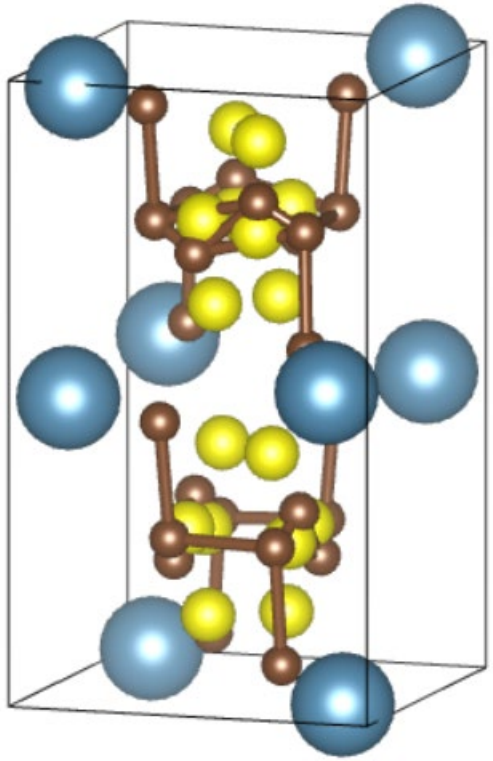

7


8

We are still not learning basics of chemistry like bond distances

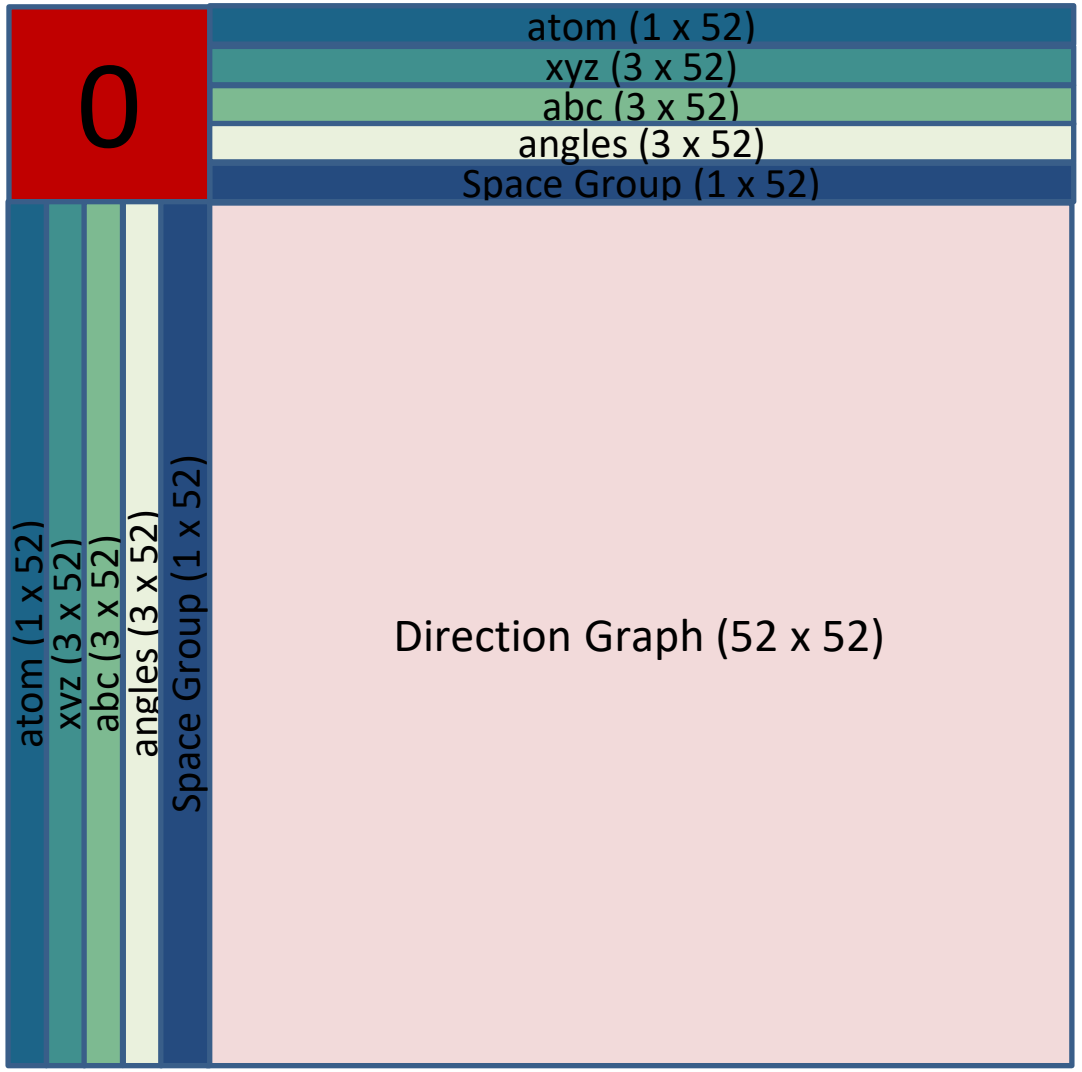


Pymatgen symmetry

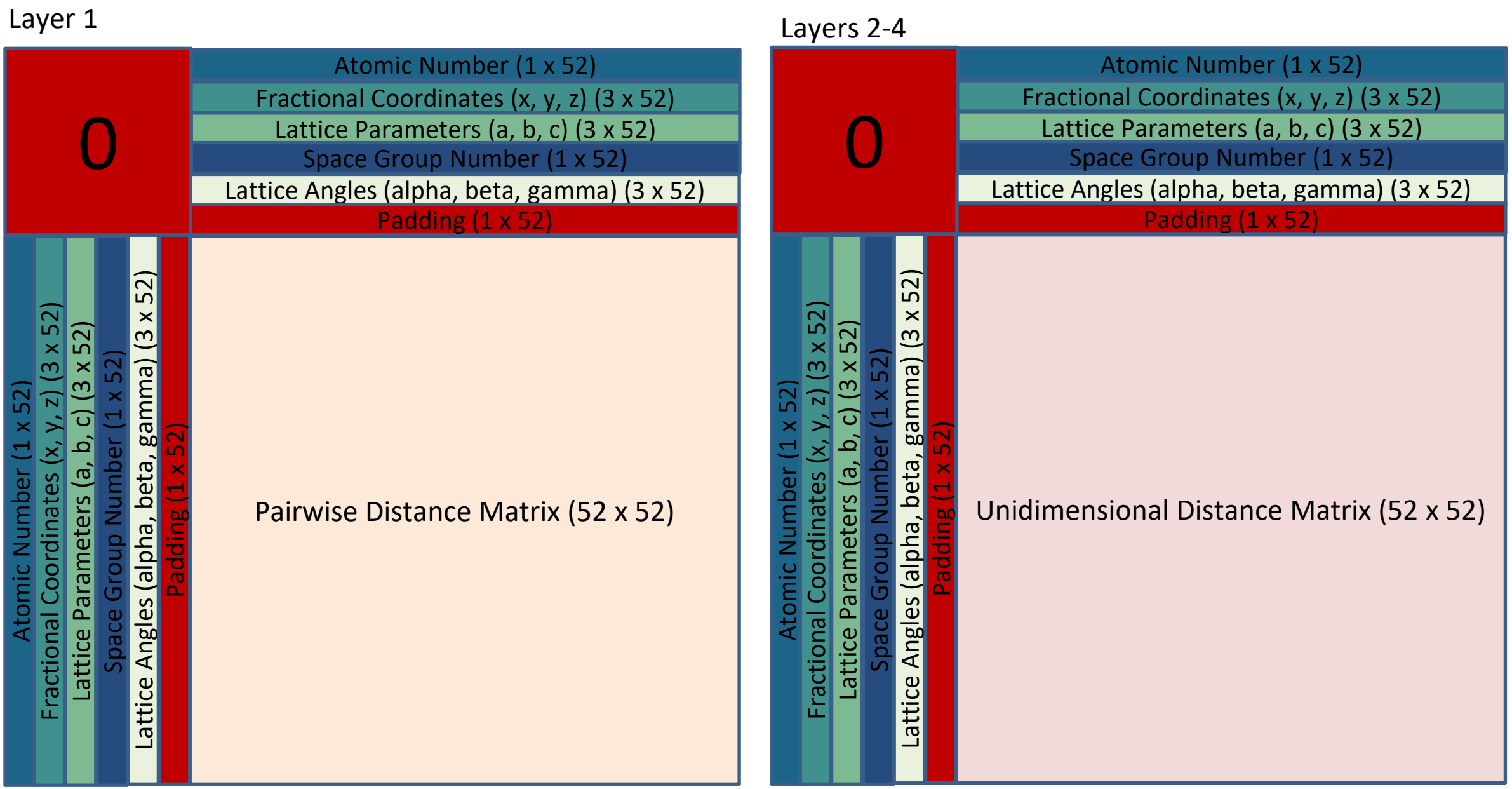


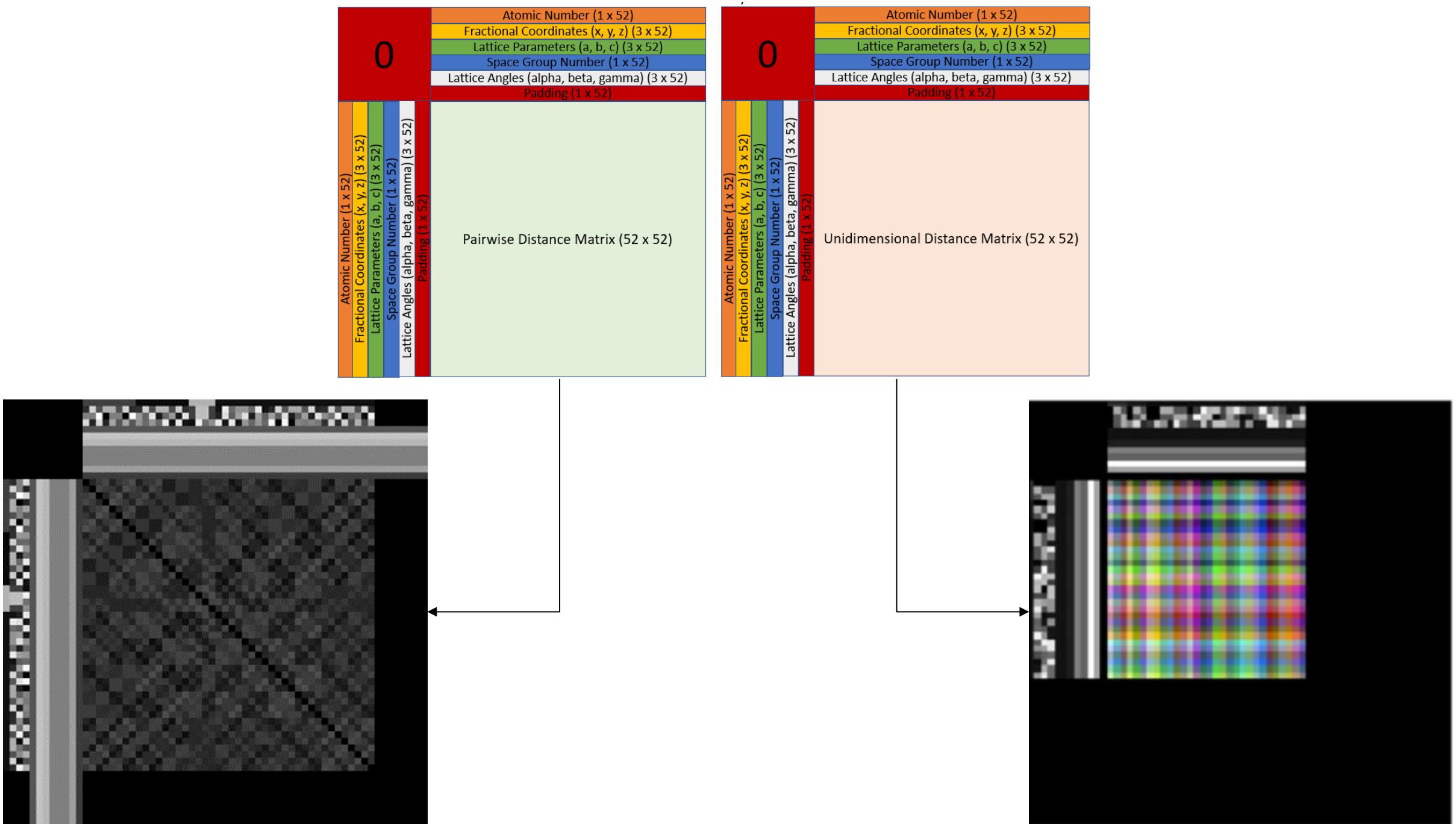
We need to encode chemical information into our representation

Graph tensor representation is needed



We need to encode chemical information into our representation




0
Atomic Number (1 x 52)
Fractional Coordinates (x, y, z) (3 x 52)
Lattice Parameters (a, b, c) (3 x 52)
Space Group Number (1 x 52)
Lattice Angles (alpha, beta, gamma) (3 x 52)
Padding (1 x 52)
Pairwise Distance Matrix (52 x 52)
0
Atomic Number (1 x 52)
Fractional Coordinates (x, y, z) (3 x 52)
Lattice Parameters (a, b, c) (3 x 52)
Space Group Number (1 x 52)
Lattice Angles (alpha, beta, gamma) (3 x 52)
Padding (1 x 52)
Unidimensional Distance Matrix (52 x 52)

xtal2png

Encode/decode a crystal structure to/from a grayscale PNG image for direct use with image-based machine learning models such as Imagen, DALLE2, or Palette.[1]

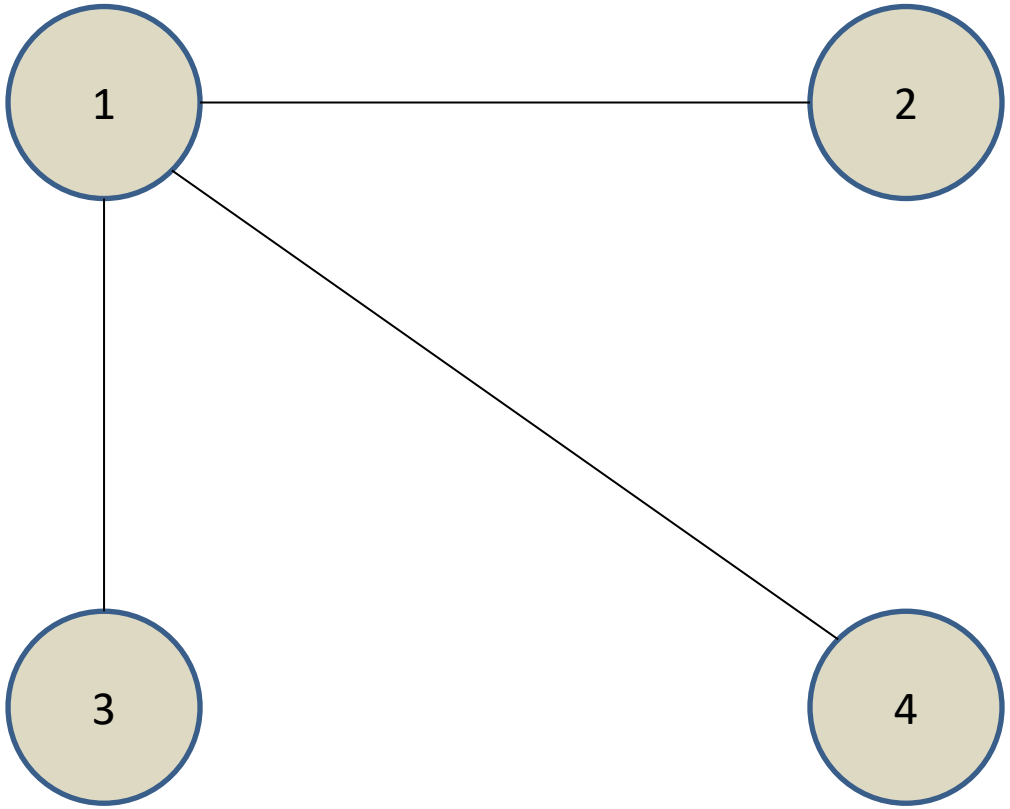
The latest advances in machine learning are often in natural language such as with long short-term memory networks (LSTMs) and transformers or image processing such as with generative adversarial networks (GANs), variational autoencoders (VAEs), and guided diffusion models; however, transfering these advances to adjacent domains such as materials informatics often takes years. `xtal2png` encodes and decodes crystal structures via grayscale PNG images by writing and reading the necessary information for crystal reconstruction (unit cell, atomic elements, atomic coordinates) as a square matrix of numbers, respectively. This is akin to making/reading a QR code for crystal structures, where the `xtal2png` representation is invertible. The ability to feed these images directly into image-based pipelines allows you, as a materials informatics practitioner, to get streamlined results for new state-of-the-art image-based machine learning models applied to crystal structure.

> Results manuscript coming soon!

Contents

- Overview
  - Getting Started
  - Limitations and Design Considerations
  - Installation
  - Editable installation
  - Command Line Interface (CLI)

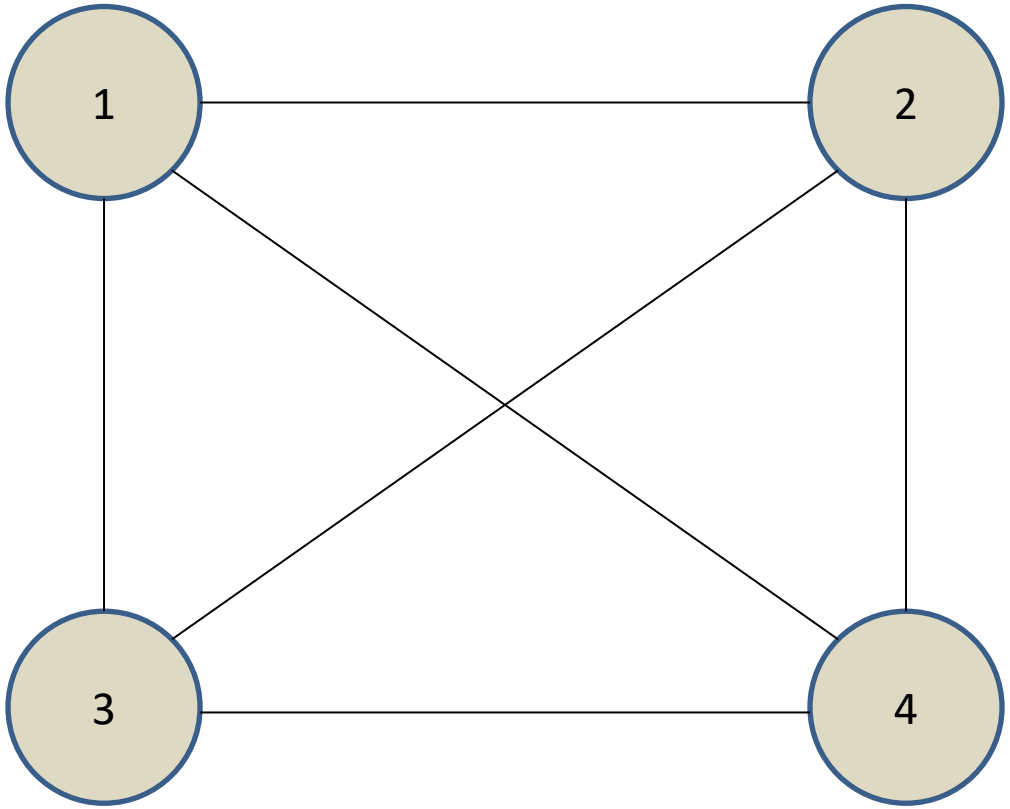
Direction graphs can be constructed in x, y, z directions



	1	2	3	4
1	0	s	0	s
2	-s	0		
3	0		0	
4	-s			0

X direction matrix

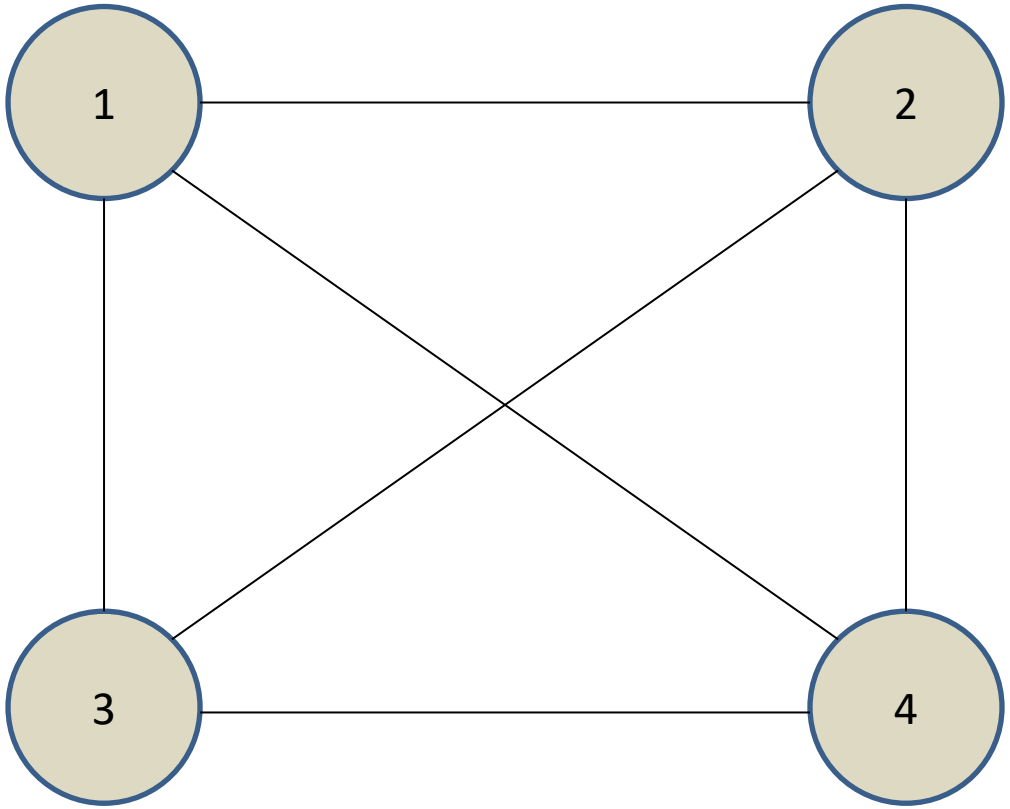
Direction graphs can be constructed in x, y, z directions



	1	2	3	4
1	0	s	0	s
2	-s	0	-s	0
3	0	s	0	s
4	-s	0	-s	0

X direction matrix

Direction graphs can be constructed in x, y, z directions

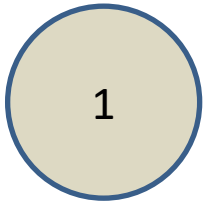


	1	2	3	4
1	0	0	s	s
2	0	0	s	s
3	-s	-s	0	0
4	-s	-s	0	0

Y direction matrix

Each atom has a map to where every other atom is

Atom 1
X = 0.0
Y = 0.0

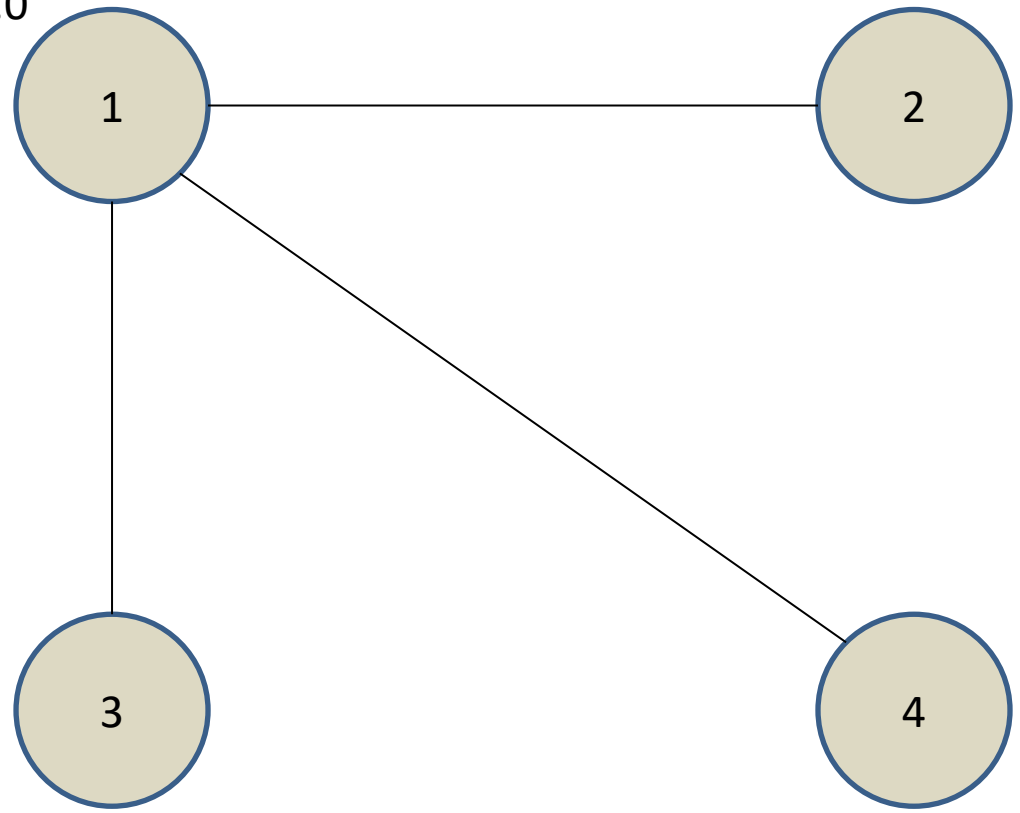


	1	2	3	4
1	0	s	0	s
2	-s	0	-s	0
3	0	s	0	s
4	-s	0	-s	0

	1	2	3	4
1	0	0	s	s
2	0	0	s	s
3	-s	-s	0	0
4	-s	-s	0	0

Each atom has a map to where every other atom is

Atom 1
X = 0.0
Y = 0.0

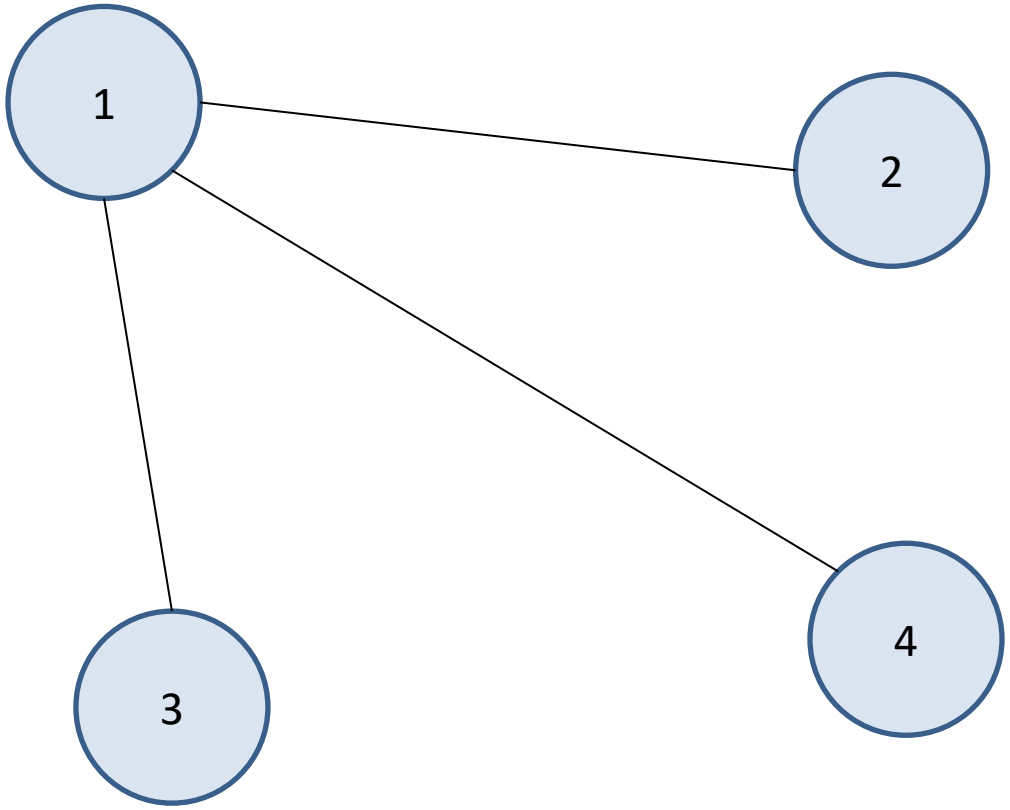


	1	2	3	4
1	0	s	0	s
2	-s	0	-s	0
3	0	s	0	s
4	-s	0	-s	0

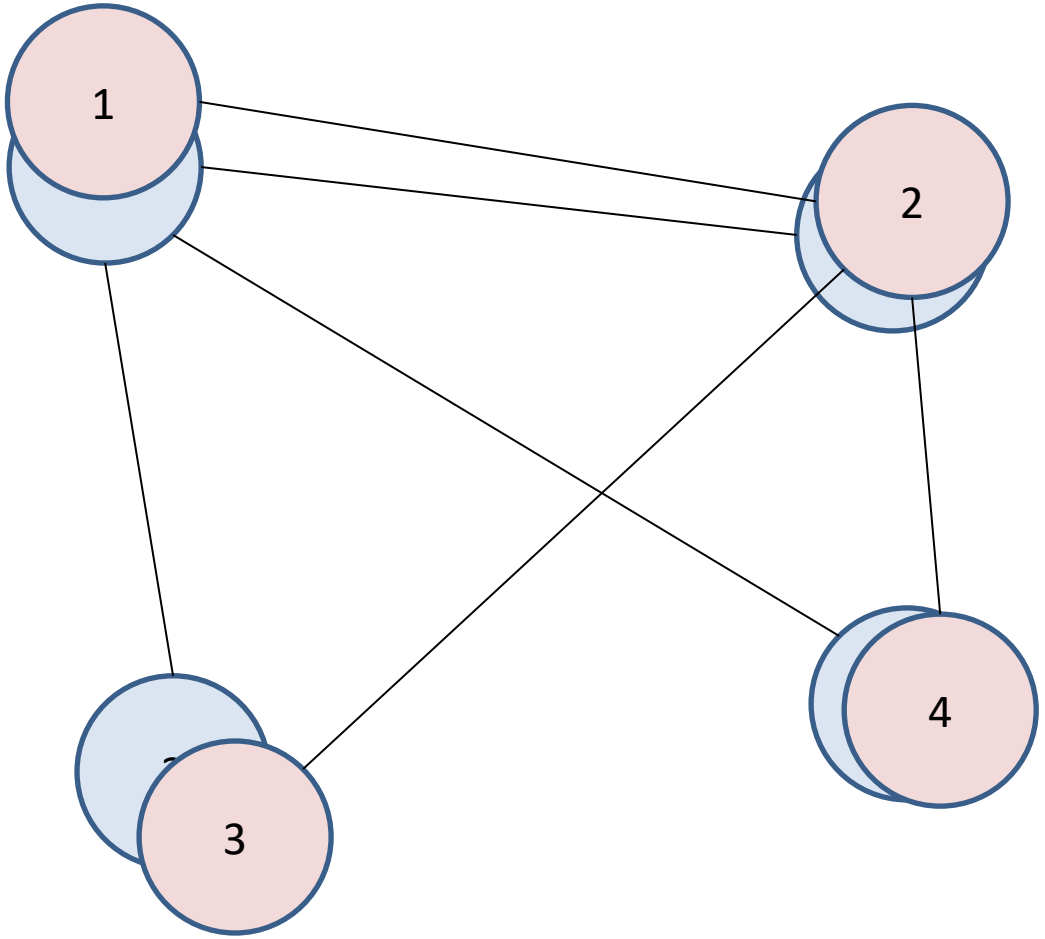
	1	2	3	4
1	0	0	s	s
2	0	0	s	s
3	-s	-s	0	0
4	-s	-s	0	0

Every atom must learn where every other atom is

Iteration 1
Atom 1 guesses where 2, 3, and 4 are relative to itself

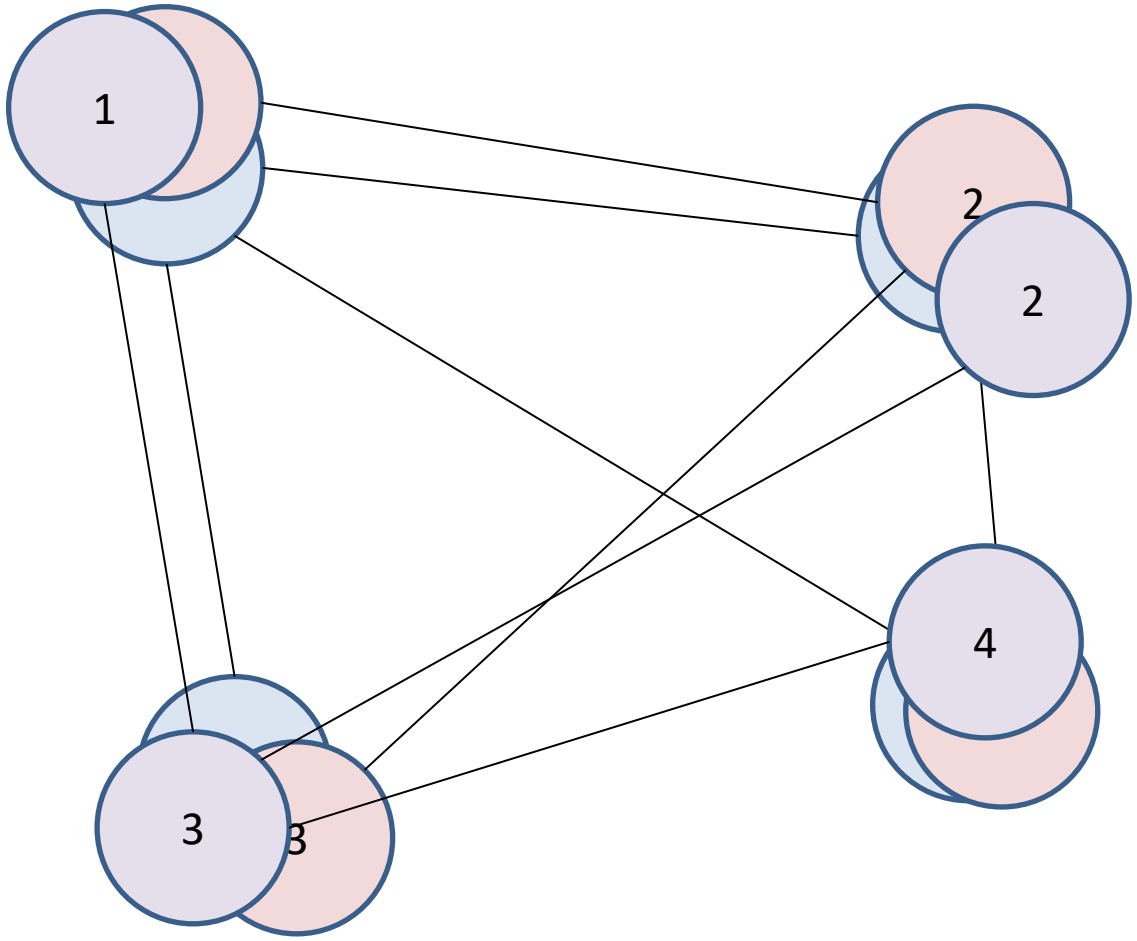


Iteration 2
Atom 2 guesses where 1, 3, and 4 are relative to itself

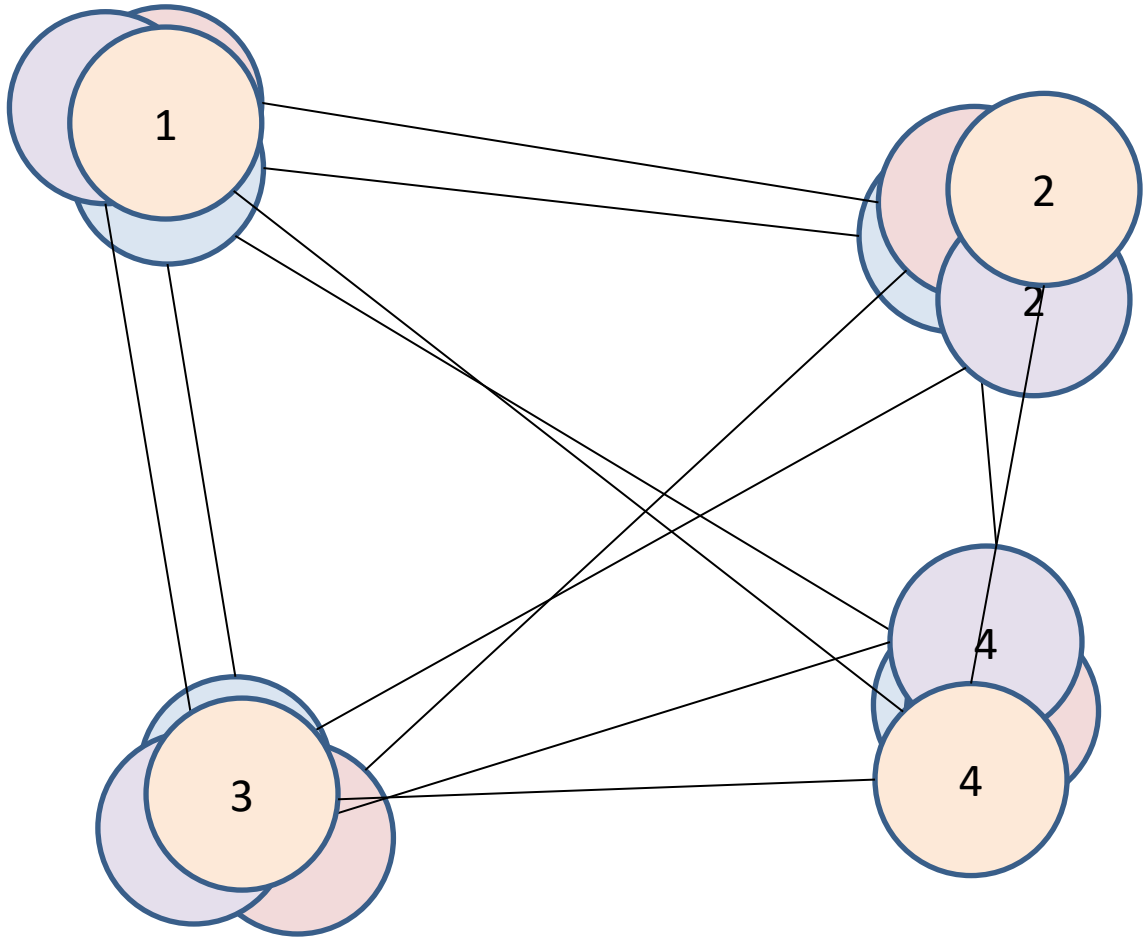


Every atom must learn where every other atom is

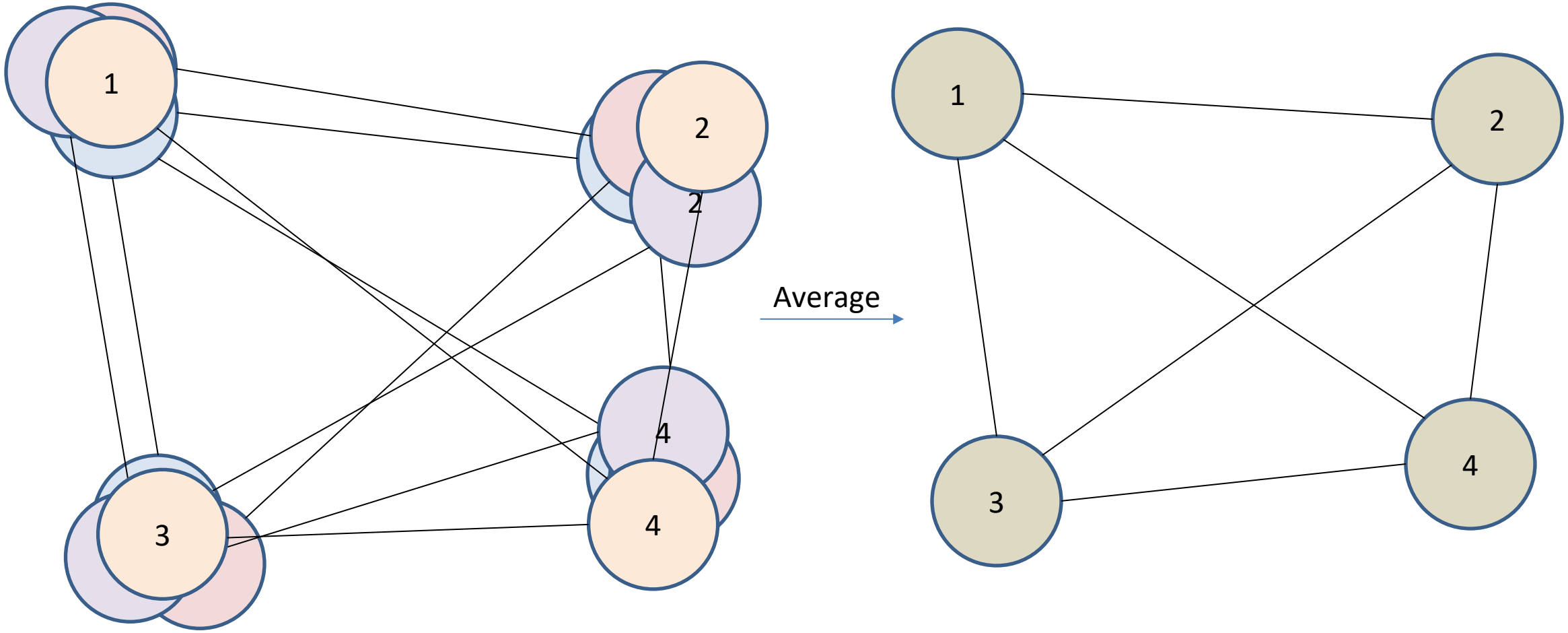
Iteration 3
Atom 3 guesses where 1, 2, and 4 are relative to itself



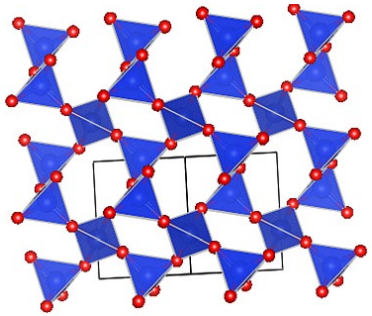
Iteration 4
Atom 4 guesses where 1, 2, and 3 are relative to itself



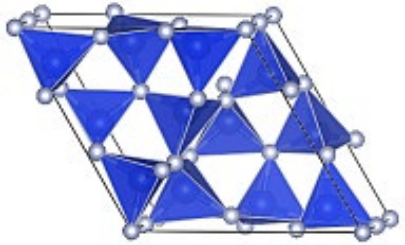
Positions can be averaged across all learned positions

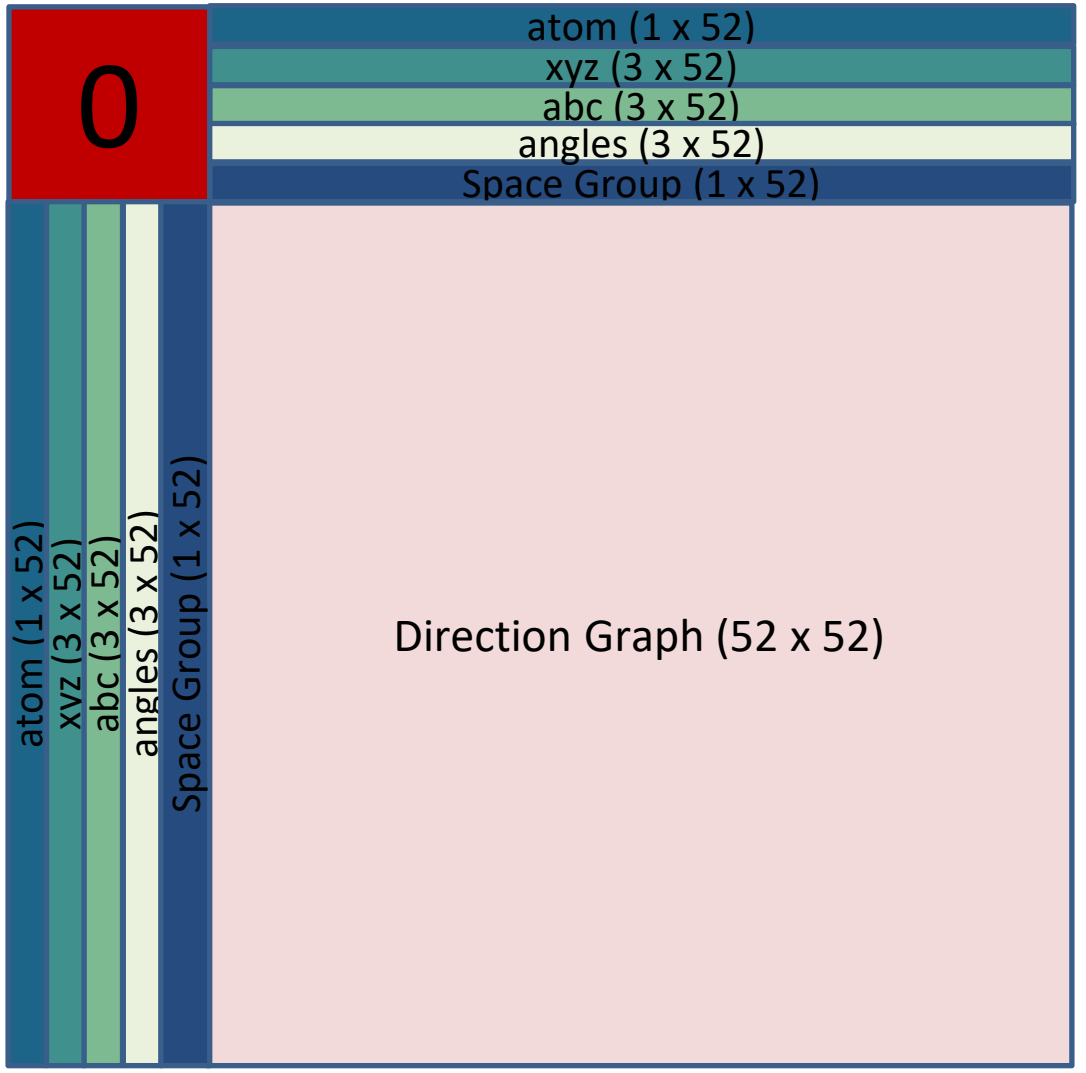


Ultimately... How do we know how good these models are?



vs





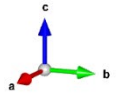
Diffusion models are the top performers by most metrics

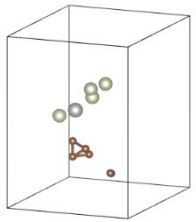
Model					
GAN	8.34	41.90	2497	2.63e-2	4.77e-2
WGAN	**4.56e-1**	5.03e-1	37	2.25e-3	1.35e-2
Diffusion	5.33e-1	**3.42e-1**	**6.1**	**3.72e-4**	**9.46e-4**

Vanilla GANs do awful

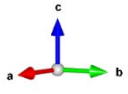
Obvious mode collapse + zero understanding of what crystals look like

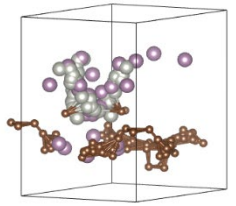
$InRh_2C_3$



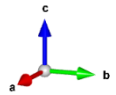


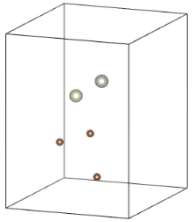
$Mo_5Pd_7C_{12}$



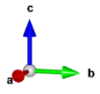


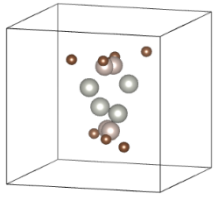
$PdRhC$



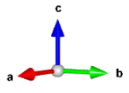


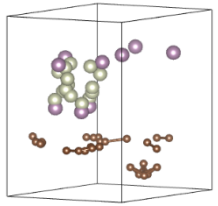
$RuPdC_2$



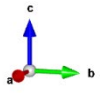


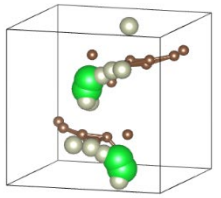
$Mo_5Pd_7C_{12}$





$Sr_2Rh_5C_8$

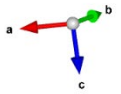


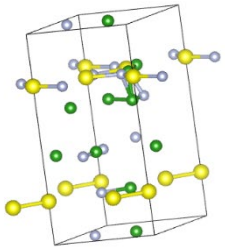


Wasserstein GANs do better

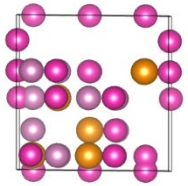
More variability of shapes and atoms, overall structures still not great

$B_{11}S_5F_8$

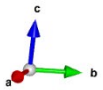


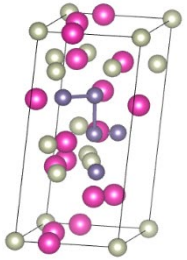


$NdInNe_6$

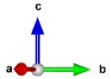


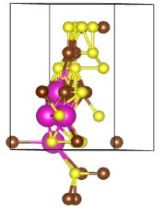
$Ge_5Rh_7Ne_{15}$



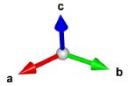


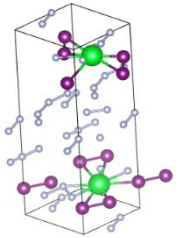
$Eu_4S_{11}B_9$



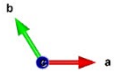


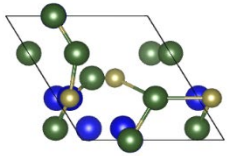
SrI_2N_3





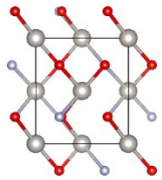
$Pm_5Er_8Te_3$



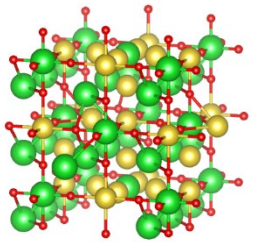


Diffusion models produce the most realistic crystals

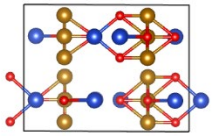
Diverse symmetrical crystal structures



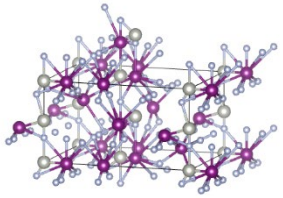
$Pt_4O_5F_3$



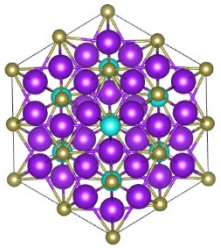
$BaNaO_4$



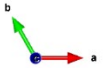
$Fe_4Cu_2O_4$

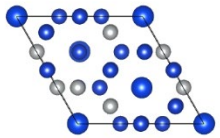


PdI_2F_9



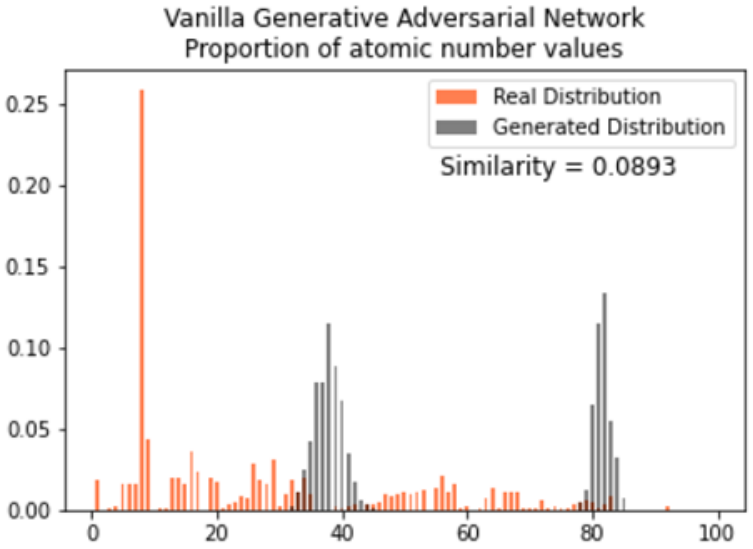
$PmPTe_3$

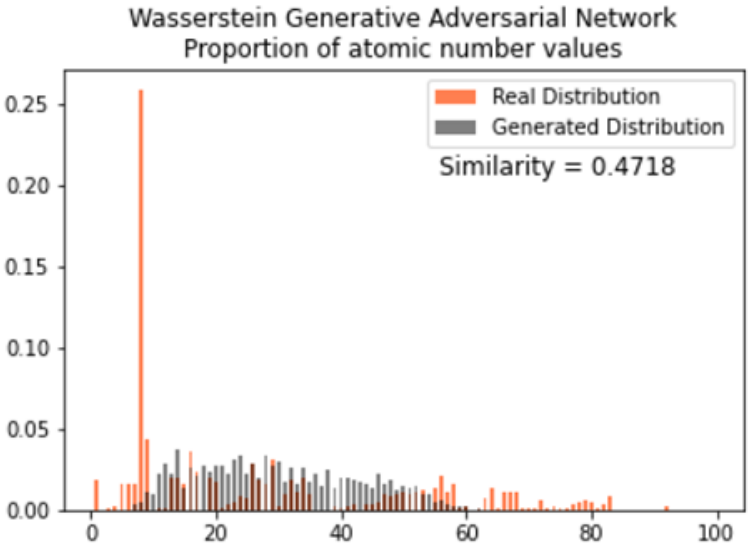


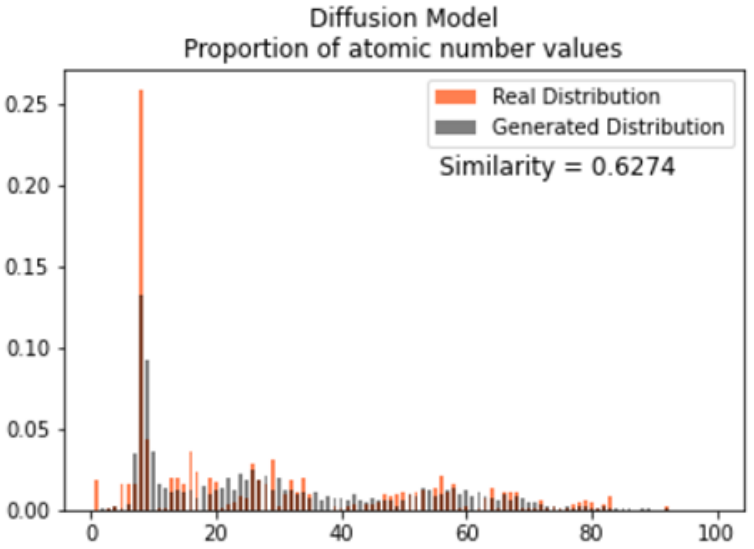


$HoNi_4Cu_5$

Advanced models seem to be capable of drawing from the breadth of training data

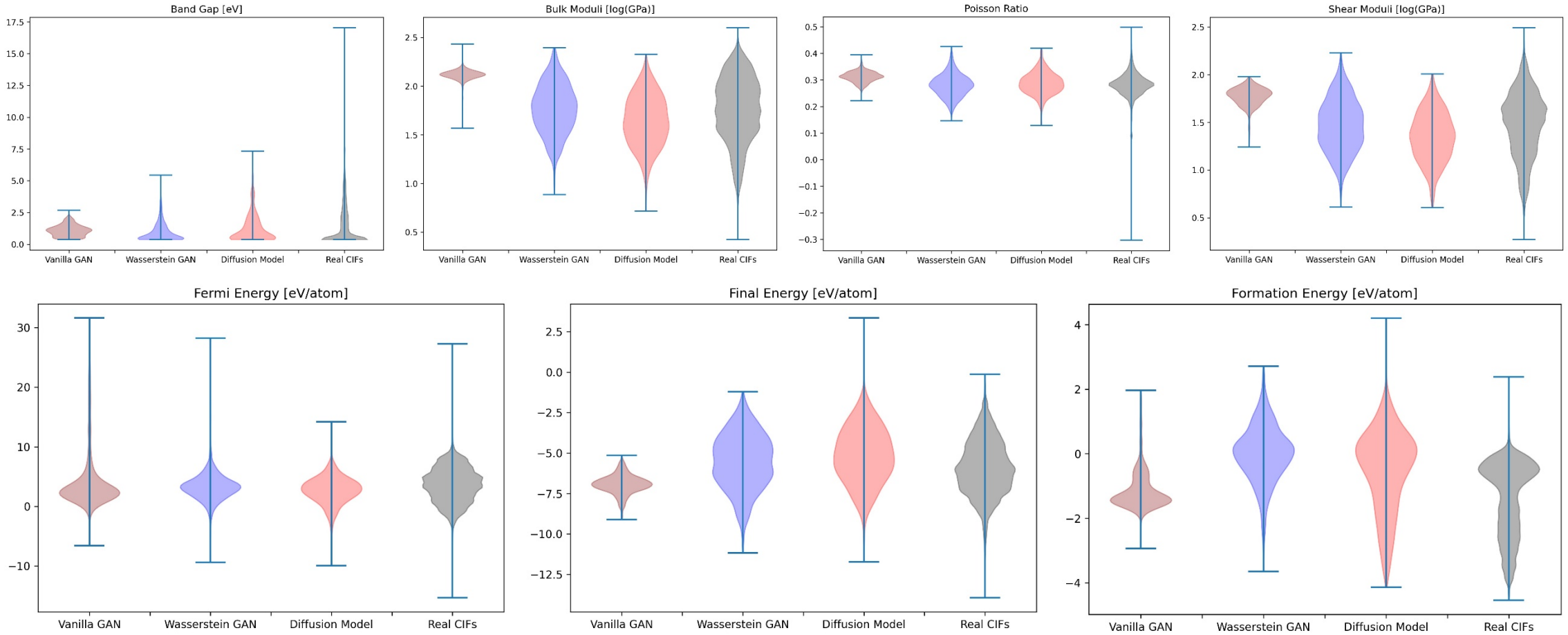






Alverson et al. Generative adversarial networks and diffusion models in material discovery, ChemRxiv 2022

Comparison to real properties also suggests good learning



Models are approaching utility!

2014

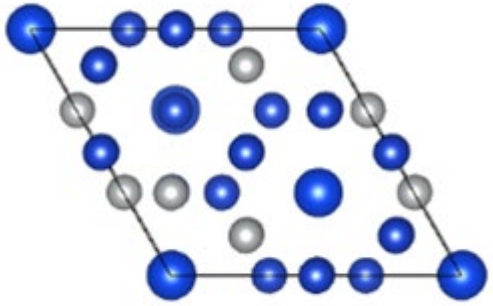
2015

2016

2017

2018

2021



2023

Let’s see a GAN tutorial on Google Colab

I hope you'll consider IMMI for publishing your best digital materials work!

- Community journal
- Free access all TMS members
- 5-year Impact factor 3.3
- Hybrid publishing options
- We publish..
  - Data
  - Tools
  - Research
  - Reviews

Editor Summary videos!

Many more available on YouTube and at the Materialism Podcast