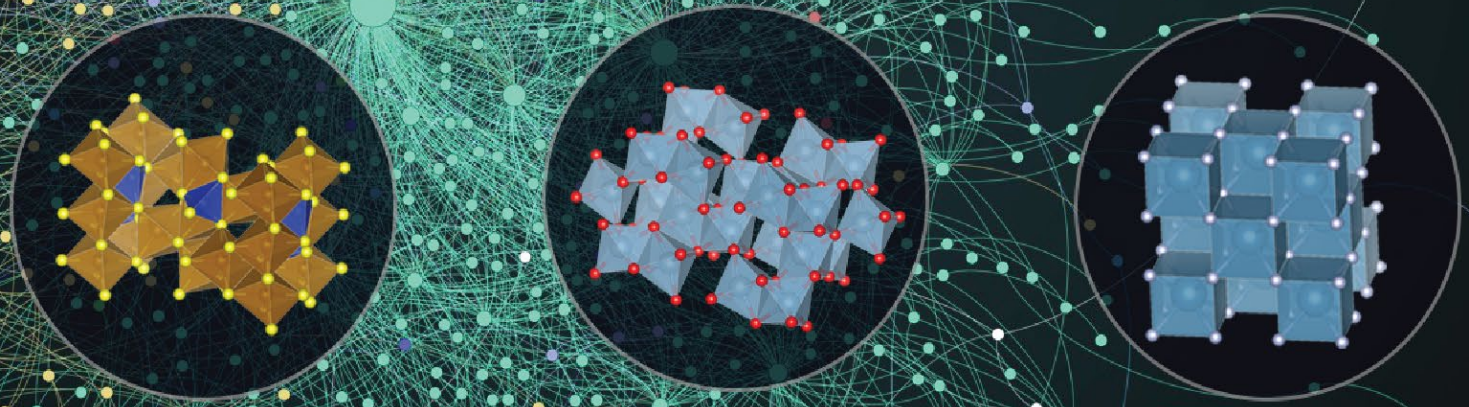
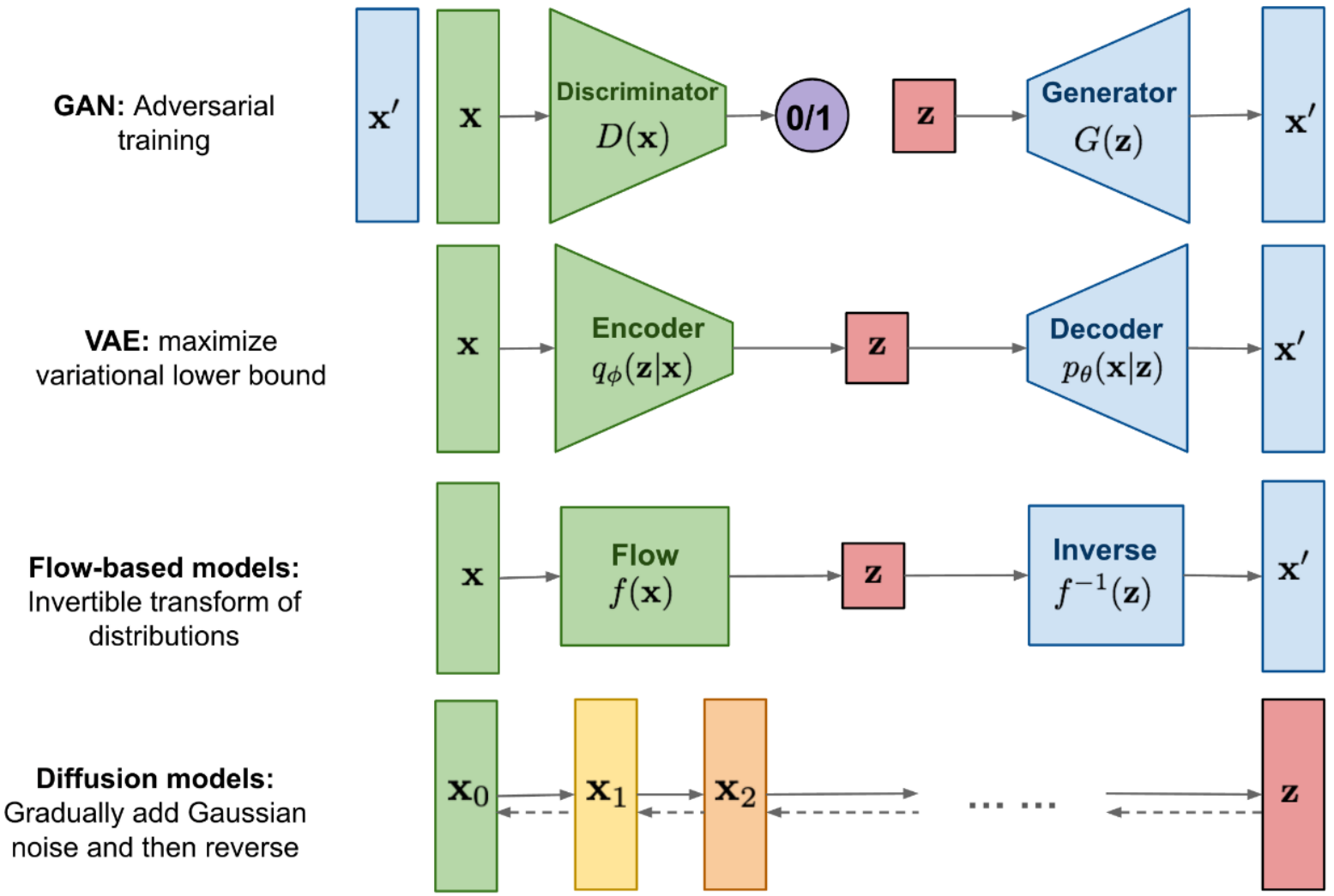
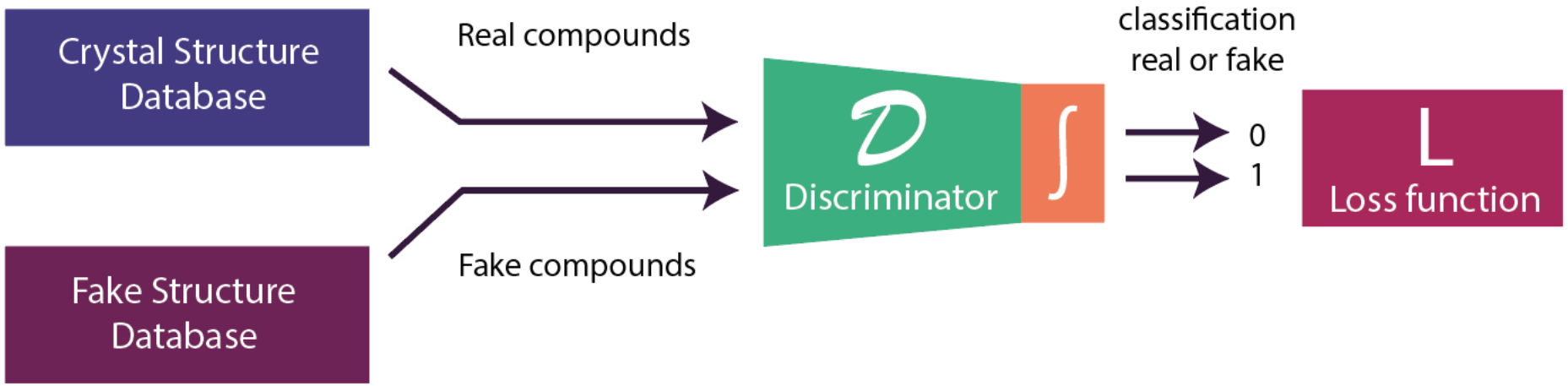
generative machine learning models



There are many generative models to choose from these days!



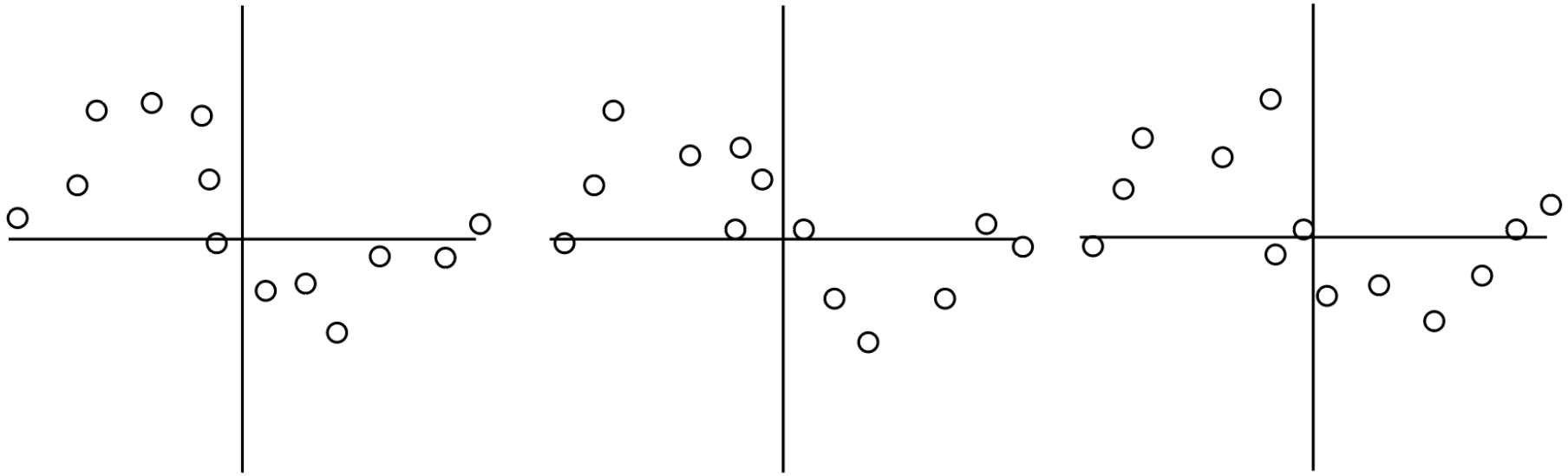
GANs stands for generative adversarial network. They start with a discriminator



This alone can't generate something new…

Let alone something that matches a specific distribution

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



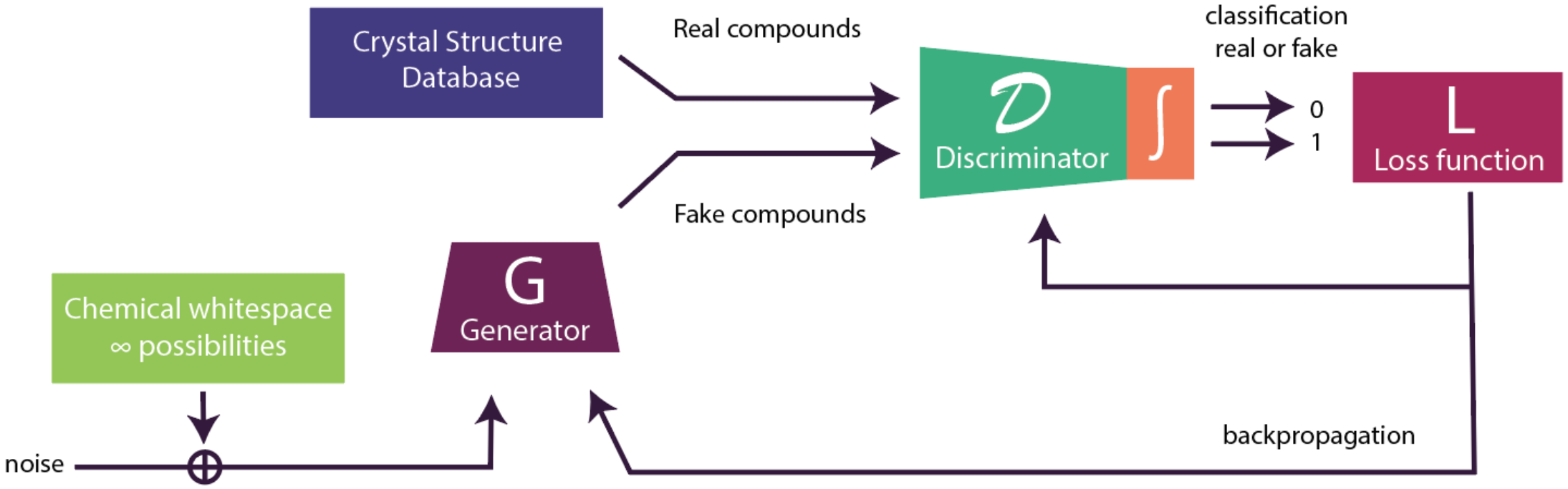
What would the underlying distribution look like?

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

To help us generate new samples, we can use adversarial training



Two networks!

Both compete in a min/max game

We do 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

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

Training iterations involve updating both networks sequentially

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

Training GANs can be kind of a pain!

GANs are powerful generative models that create high quality, realistic data

However… they suffer from training instability. What's going on?

- 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

variational autoencoders