Machine Learning – Generative Models

Generative Adversarial Network (GANs)

- Form of unsupervised learning made up of two different models – generator and discriminator
- Generator creates fake samples and feeds it into a discriminator alongside real samples from the domain
- The discriminator's job is to determine which is real and which is fake
- Idea is to create samples which the discriminator cannot distinguish between
- Often used by image generation model – generator tries to create images based on real samples from the domain
- On each run through, either the generator successfully 'tricks' the discriminator or the generator fails to do so – in either case, the 'loser' must update their model
- Often the generator and discriminator are both CNN
- The adaptations to the model are done with the minimax objective function:
$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$
- Where the *G* refers to the generator, *D* is the discriminator, p_{data} is real data drawn from the domain, and p_z is a noise vector
- Nash equilibrium – want to get to the point where $D(x)$=0.5 for real and fake data.

Diffusion Models

- Inspired from non-equilibrium thermodynamics – like undoing the process of diffusion in a glass of water for example
- Do this by adding noise to the images and then later learning later how to remove this noise
- Gaussian noise is added from a Markov Chain model, which is easily reversible so we can then take images given to us and then remove the noise
- We use CNN to revert the noise, usually through a UNet
- Usually want to minimise the Variational Lower Bound:
$$L = \mathbb{E}_{q(x_{1:T}|x_0)}\left[\sum_{t=1}^{T} D_{\text{KL}}\left(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)\right)\right]$$
- Where the D_{KL} is the Kullback-Leibler divergence
- Very common for image generation and for image enhancement
- Advantages over GANs: no mode collapse, more stable training

Normalizing Flows

- Start with a simple distribution to start with, e.g. multivariate normal, and then apply a series of bijective transformations to get a new probability distribution – change of variable formula
- Used particularly in generative models, including inferring latent variables – if we write the composition of all the functions as *f* we get:
$$\log p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^{K} \log \left|\det\left(\frac{\partial f_i^{-1}}{\partial z_i}\right)\right|$$
- And then run MLE on this function
- Exact log-likelihood evaluation – c.f. VAEs with lower bound on log-likelihood and GANs where we have no log-likelihood evaluation
- Exact posterior inference, from bijectivity – c.f. VAEs with approximate posterior and GANs with no latent variable inference

- Trick – ensure that all the Jacobians are triangular, so their determinants are easy to calculate – use coupling layers, or autoregressive/masked autoregressive flows where the Jacobian is easily tractable

Transformer

- GPT – generative pre-trained transformer
- Each word is broken up into tokens, encoded into vectors
- Attention block – creates dot products of the token vectors with each other, it helps you understand how much attention each token should take relative to each other.
- Then the vectors are going to an MLP, where certain weightings and activation function (softmax) are applied.
- Repeat the alternating process until you get a final vector, which will then generate a probability distribution for the next word
- Tokenization involves an embedding matrix, which takes the tokens and projecting them as vectors – 'similar' tokens are given similar-ish tokens, i.e. the difference between 'boy' and 'girl' is like the difference between 'king' and 'queen'
- At the last step, we have an unembedding matrix to take all the final stage vectors to put all the information into the very last vector, from which we use the softmax function
- Softmax takes the exponential of all the vector data points at the end and then normalize it to give the output probability distribution – you can refer to Temperature, c.f. stat mech

Variational Autoencoders (VAEs)

- Unsupervised NN, made up of encoder taking in some input and compresses it into some code, or bottleneck, which then goes through a decoder and give an output
- The encoder compresses the information into a latent space representation
- Variational autoencoders learn to generate new data, and aim to minimise reconstruction loss and latent loss
- The objective is the Evidence Lower Bound (ELBO) – want to maximise the marginal likelihood after integrating out latent variables – integral is intractable so we maximise the ELBO, so we only have an approximate posterior:

$$\log p\,(x) \geq \mathrm{ELBO} = \mathbb{E}_{q_\phi(\mathrm{z}|x)}[\log p_\theta\,(x|z)] - D_{\mathrm{KL}}\left(q_\phi(z|x) \parallel p(z)\right)$$

- Closed form solution of 'end of training'
- Image output is very good but very blurry
- Advantages of VAE: anomaly detection, data compression and semi-supervised learning