Artificial Models for Music Creativity

Lesson 4 - Introduction to Audio Deep Learning 2/2

Alessandro Anatrini - 4.1.2025

TAXONOMY OF GENERATIVE MODEL

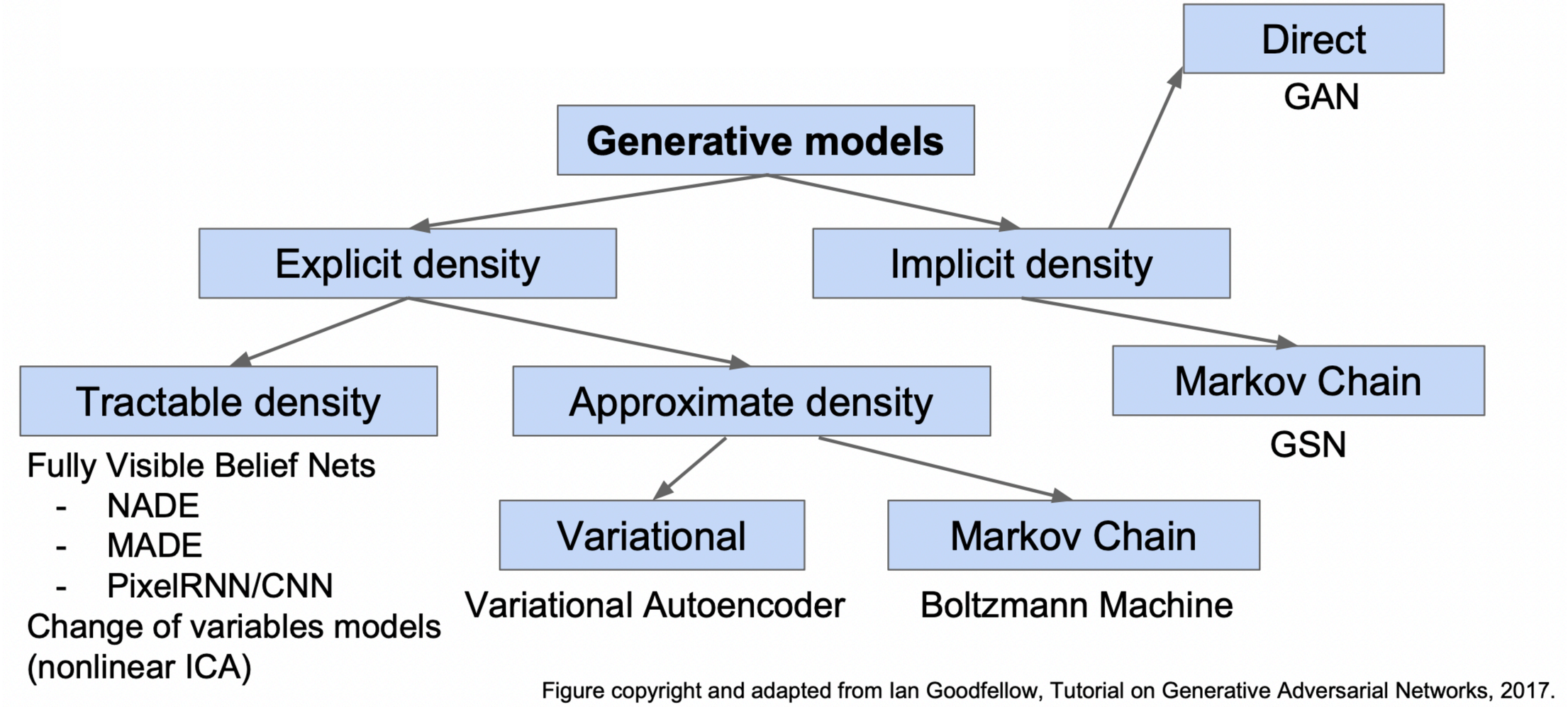


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

TAXONOMY OF GENERATIVE MODEL

Explicit vs Implicit Density: In generative models, explicit density models, such as Gaussian Mixture Models, define a clear probability distribution and can directly compute probabilities for given instances, making them suitable for data like images where we might want to compute the likelihood of a pixel's intensity; on the other hand, implicit density models, like Generative Adversarial Networks (GANs), do not define an explicit probability distribution but can generate new samples, which can be useful for generating sequences of audio where the goal is to produce new, realistic sounds

TAXONOMY OF GENERATIVE MODEL

Tractable vs Approximate Density tractable density models, like Naive Bayes, allow for exact computation of probabilities and likelihoods, which can be beneficial when dealing with text data where we want to compute the exact probability of a word given its context; in contrast, approximate density models, such as VAEs, use approximations for inference or learning when exact computation is infeasible due to complexity, making them suitable for high-dimensional data like images where exact computation of probabilities is computationally expensive

What is a latent space?

Latent space or latent variable is a projection or compression of a data distribution

A latent space provides a compression or high-level concepts of the observed raw data such as the input data distribution

Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GANs) are a class of machine learning models introduced by Ian Goodfellow and his colleagues in 2014. They are widely used for generating new data samples that resemble a given dataset

Generative modeling is an unsupervised learning problem, a clever property of the GAN architecture is that the training of the generative model is framed as a supervised learning problem

Generative Adversarial Networks (GAN)

Machine-learning models can learn the statistical latent space of images, music, and stories, and they can then sample from this space, creating new artworks with characteristics similar to those the model has seen in its training data

GANs provide a way to sample from the latent distribution when this cannot be achieved directly. GANs samples from a more simple distribution, e.g. random noise. and learn transformation to training distribution

Generative Adversarial Networks (GAN)

Generator: The generator takes random noise as input and tries to generate synthetic data samples that resemble the real data. Its goal is to generate samples that are indistinguishable from the real data by the discriminator

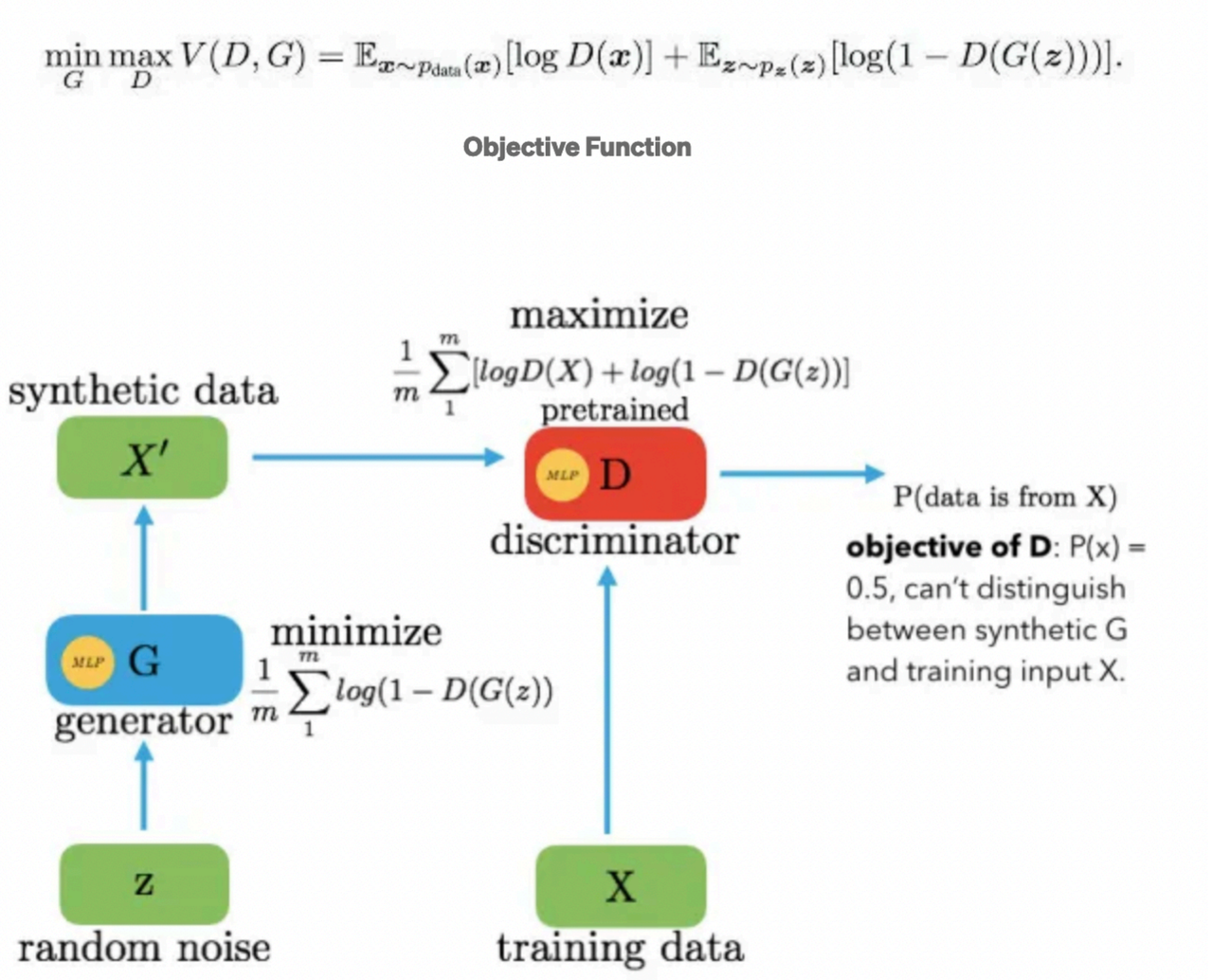
Discriminator: The discriminator is like a binary classifier that distinguishes between real data samples from the training set and fake data samples generated by the generator. It is trained to assign high probabilities to real samples and low probabilities to fake samples

Generative Adversarial Networks (GAN)

The generator and discriminator, are trained together: the generator generates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake

The discriminator is then updated to get better at discriminating real and fake samples in the next round, and importantly, the generator is updated based on how well, or not, the generated samples fooled the discriminator

Generative Adversarial Networks (GAN)



The Discriminator and Generator play a two-player minimax game with the value function V(G, D)

D() gives us the probability that the given sample is from training data X. For the Generator, we want to minimize log(1-D(G(z)) i.e. when the value of D(G(z)) is high then D will assume that G(z) is nothing but X and this makes 1-D(G(z)) very low and we want to minimize it which this even lower. For the Discriminator, we want to maximize D(X) and (1-D(G(z))). So the optimal state of D will be P(x)=0.5. However, we want to train the generator G such that it will produce the results for the discriminator D so that D won't be able to distinguish between z and X

Generative Adversarial Networks (GAN)

GANs come in different flavours: VanillaGAN, CGAN, DCGAN, LAPGAN, SRGAN

Image synthesis and generation, image2image translation, text2image synthesis, data augmentation, data generation for training

Generative Adversarial Networks (GAN)

Synthetic Data Generation: GANs can generate new, synthetic data that resembles some known data distribution, which can be useful for data augmentation, anomaly detection, or creative applications;

High Quality Results: can produce high-quality, photorealistic results in image synthesis, video synthesis, music synthesis, and other tasks;

Unsupervised learning: GANs can be trained without labeled data, making them suitable for unsupervised learning tasks, where labeled data is scarce or difficult to obtain;

Generative Adversarial Networks (GAN)

Training Instability: GANs can be difficult to train, with the risk of instability, mode collapse, or failure to converge;
Computational Cost: GANs can require a lot of computational resources and can be slow to train, especially for high-resolution images or large datasets;
Overfitting: GANs can overfit the training data, producing synthetic data that is too similar to the training data and lacking diversity;
Accountability, Bias and Fairness

TRANSFORMERS

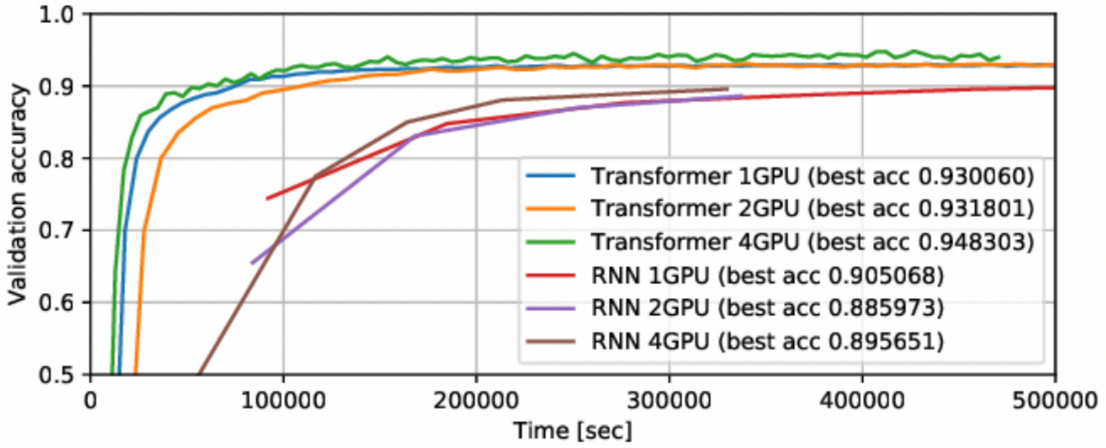
A transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. It is used primarily in the fields of natural language processing (NLP) and computer vision (CV)

Similar to recurrent neural networks (RNN) , transformers are designed to process sequential input data, such as natural language, with applications towards tasks such as translation and text summarisation

TRANSFORMERS

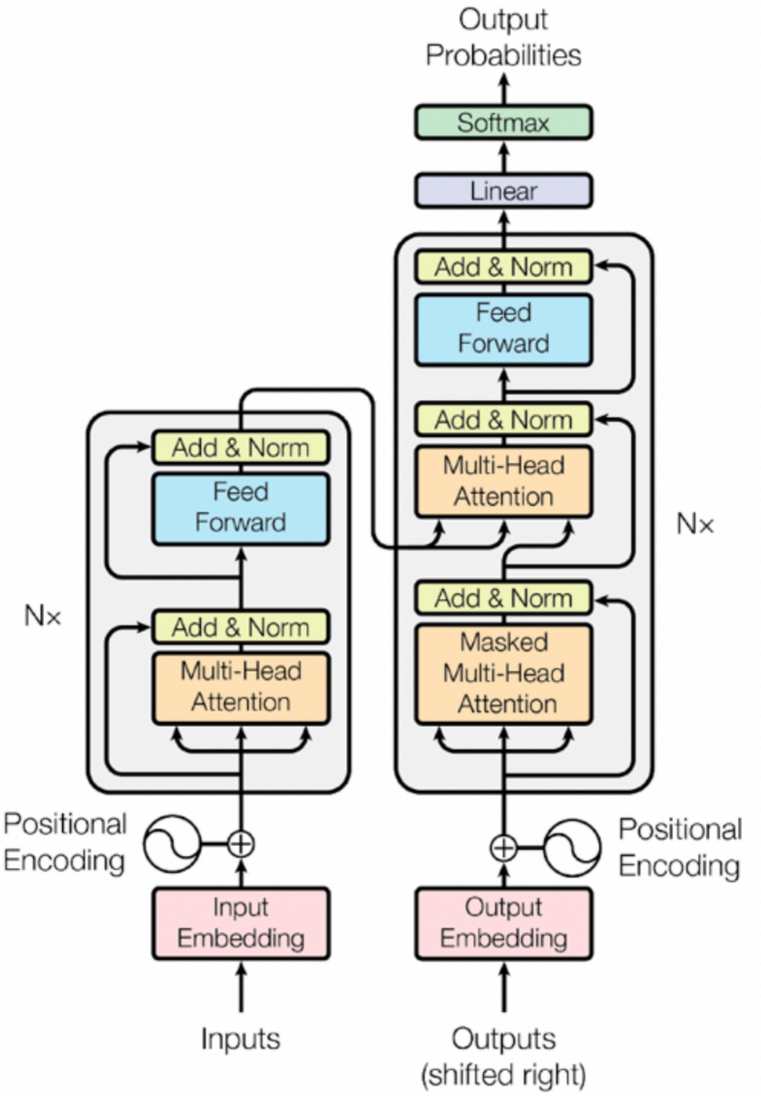
Unlike RNNs, transformers process the entire input all at once; this provides context for any position in the input sequence. For example, if the input data is a natural language sentence, the transformer does not have to process one word at a time. This allows for more parallelisation than RNNs and therefore reduces training times

Transformers outperforming RNNs



TRANSFORMERS

The Transformer model, used in machine learning and particularly in natural language processing, consists of two main components: an Encoder and a Decoder. Each of these components is made up of multiple layers, as indicated by “Nx” in the diagram, suggesting that these operations are repeated a number of times



A transformer architecture

TRANSFORMERS

Encoder: the process begins with an Input block connected to Positional Encoding; This means that the input data is combined with positional information

Encoder: this then feeds into a series of Add & Norm and Feed Forward blocks stacked vertically. These represent the encoding process where the input data is processed through multiple layers

TRANSFORMERS

Decoder: on the output side, there is an Output Embedding block connected to Positional Encoding, similar to the encoder side but for outputs

Decoder: A top both columns, there are connections leading out to an Output Probabilities box via a Softmax function which indicates that after processing through encoders and decoders, output probabilities for predicted elements are generated using softmax

TRANSFORMERS

Natural Language Processing

Computer Vision

Audio and Speech Processing

Signal processing e.g. in biology for studying protein folding

Multimodal AI systems

Customisation and Transfer Learning

TRANSFORMERS

Parallel computations: process input sequences as a whole

Long range dependencies

Contextual representation: they can understand the meaning of an element based on its context in the sequence

Flexibility: they make few assumptions about the structural bias of input data

TRANSFORMERS

Complexity: complex architecture limits interpretability > Black box

Computational overhead due to extensive attention mechanism

Resource demanding systems prevent a wider adoption