

Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Advanced Deep Learning

Dr. Rastgoo

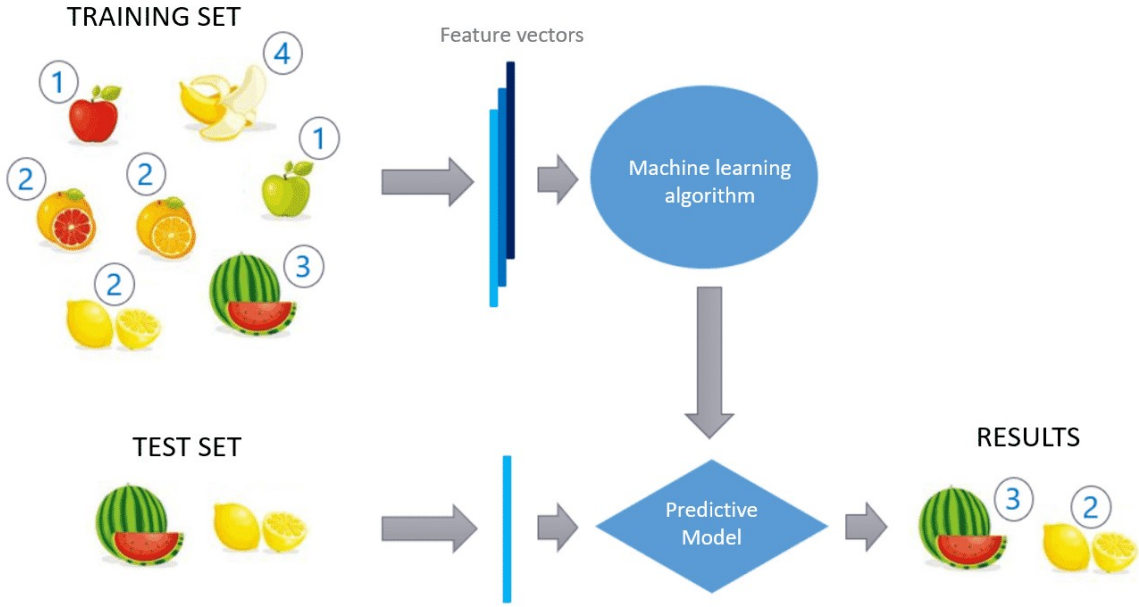






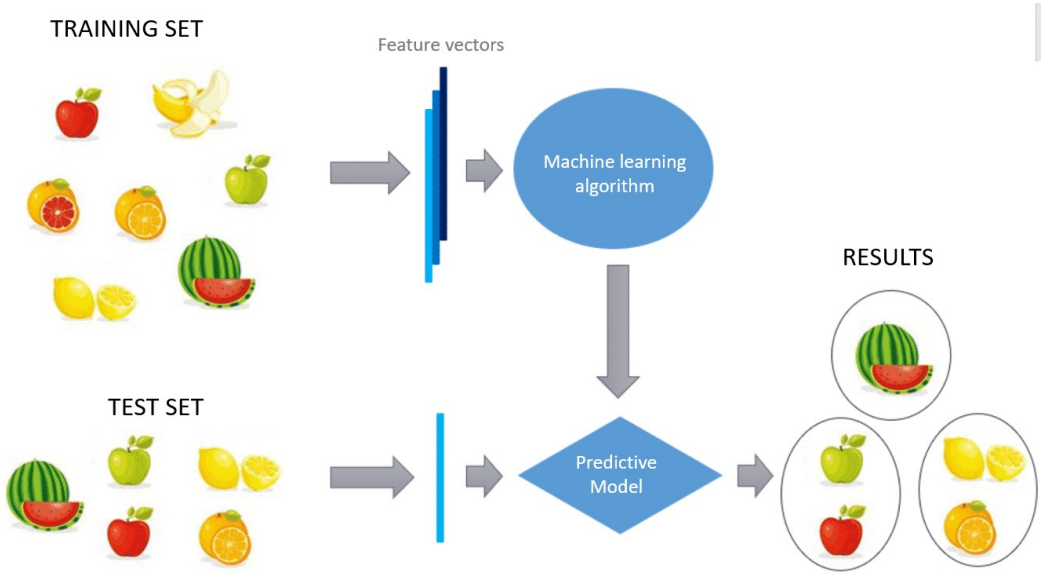
Machine learning methods

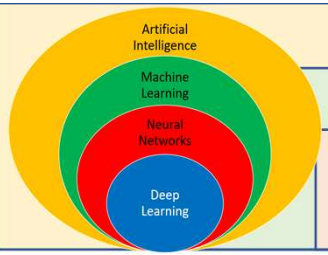
- **Supervised machine learning algorithms.** The system is able to provide targets for any new input after sufficient training.



Machine learning methods

- **Unsupervised machine learning algorithms.** The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.





Artificial Intelligence (AI)

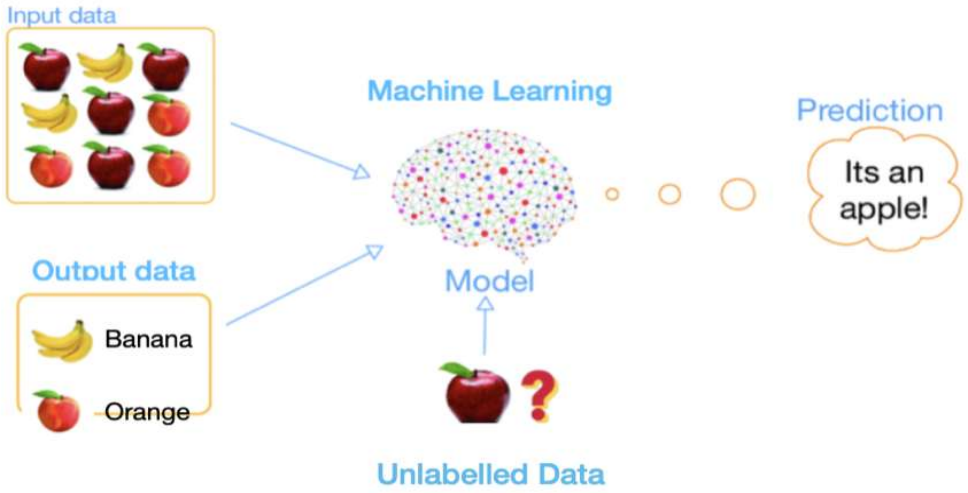
Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

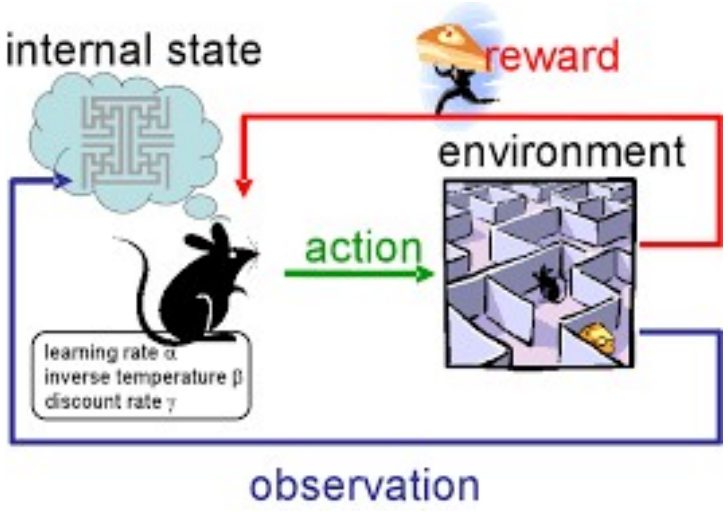
Machine learning methods

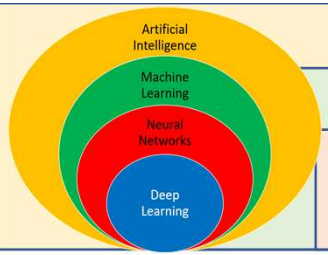
- **Semi-supervised machine learning algorithms.** Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it.



Machine learning methods

- **Reinforcement machine learning algorithms.** It is a learning method that interacts with its environment by producing actions and discovers errors or rewards (Trial and error search).





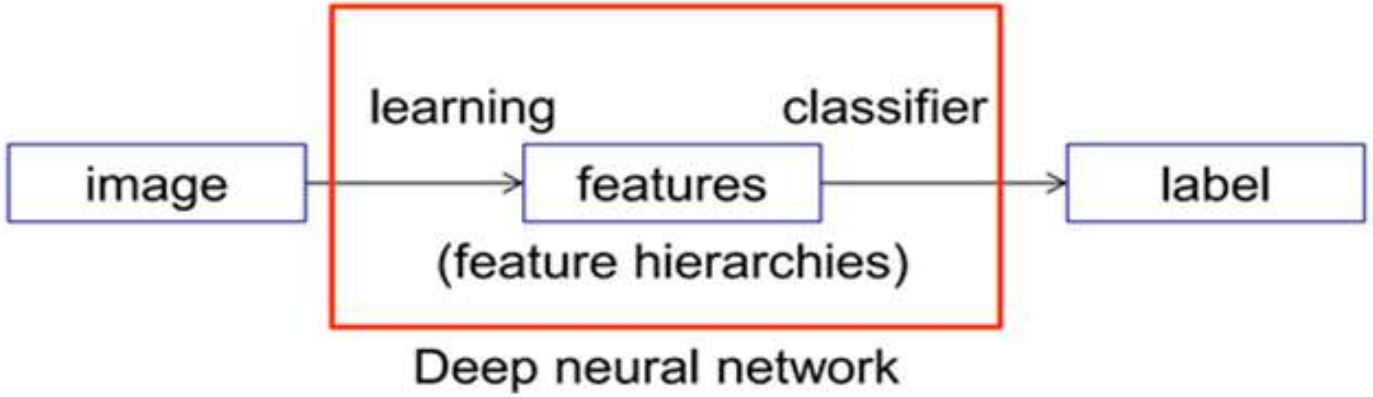
Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

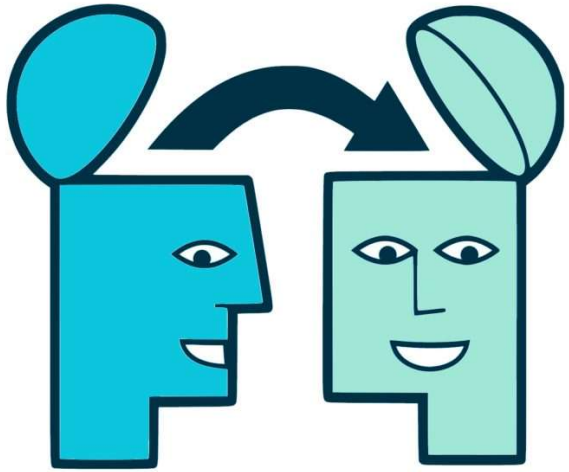
Deep Learning (DL)

What is the "Deep Learning" ?



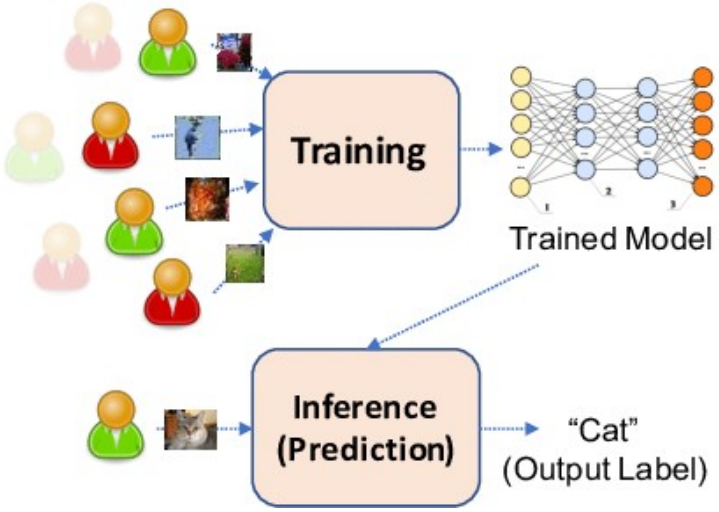
Learning methods in DL

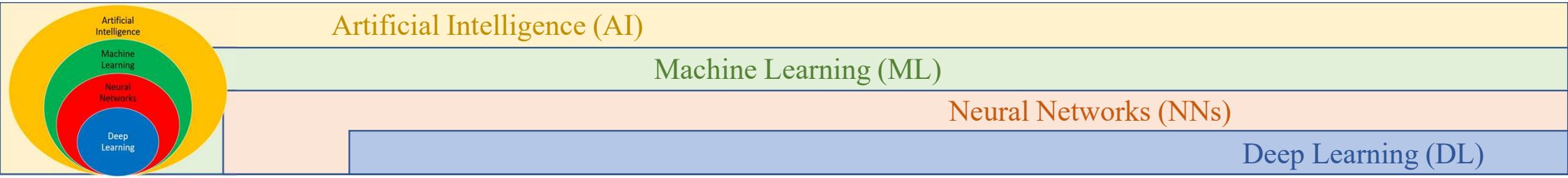
- **Transfer learning**. This process involves perfecting a previously trained model.



Learning methods in DL

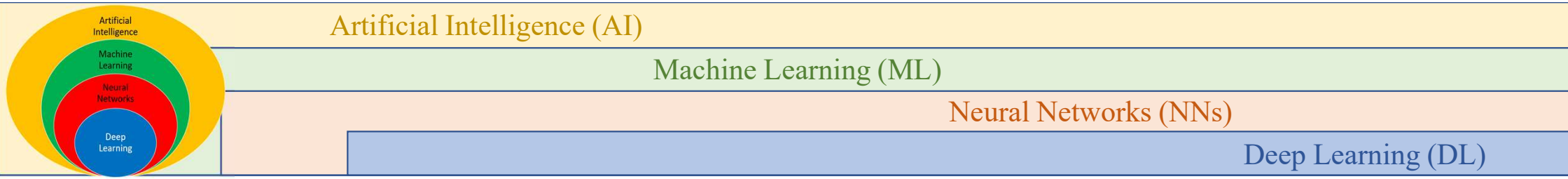
- **Training from scratch**. This method requires a developer to collect a large labeled data set and configure a network architecture that can learn the features and model.





Advanced Deep Learning Models

- Diffusion Models
- Explainable AI
- Quantum AI

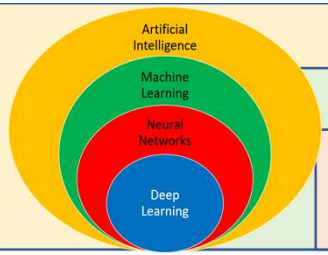


Diffusion Models – Introduction

- Diffusion Models are **generative models**, meaning that they are used to generate data similar to the data on which they are trained.

- Fundamentally, Diffusion Models work by **destroying training data** through the successive addition of Gaussian noise, and then learning to recover the data by reversing this noising process.

- After training, the trained Diffusion Models can be used to generate data by simply passing randomly sampled noise through the learned denoising process.



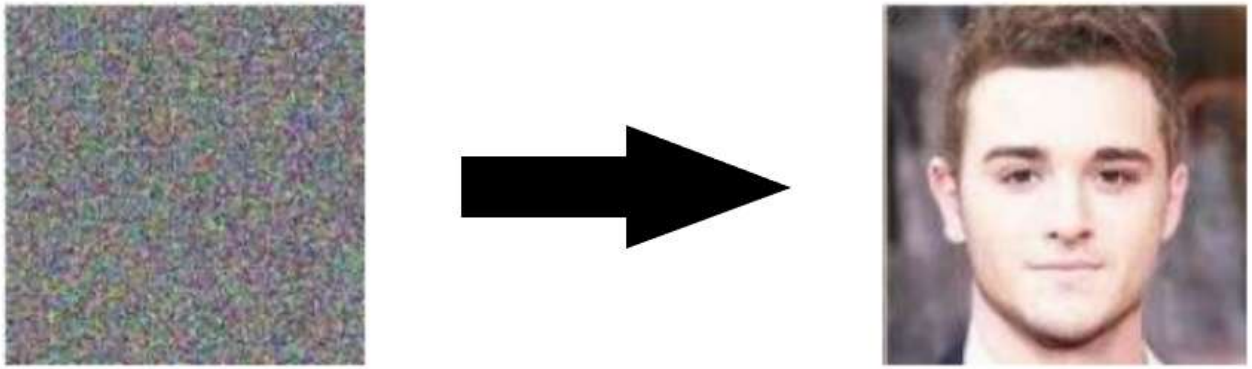
Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Diffusion Models – Introduction

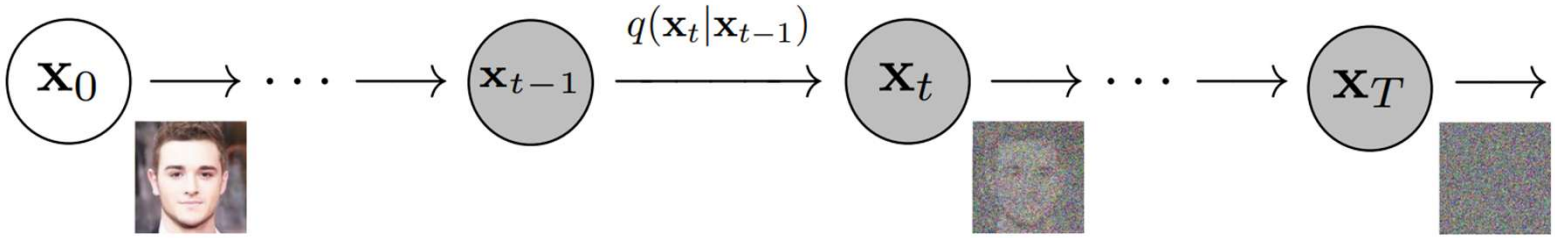


Diffusion Models can be used to generate images from noise

Diffusion Models – Introduction

- More specifically, a Diffusion Model is a **latent variable model** which maps to the latent space using a fixed **Markov chain**.

- This chain gradually adds noise to the data in order to obtain the approximate posterior $q(x_{1:T}|x_0)$, where $x_0, \cdots, x_T$ are the latent variables with the same dimensionality as x_0.

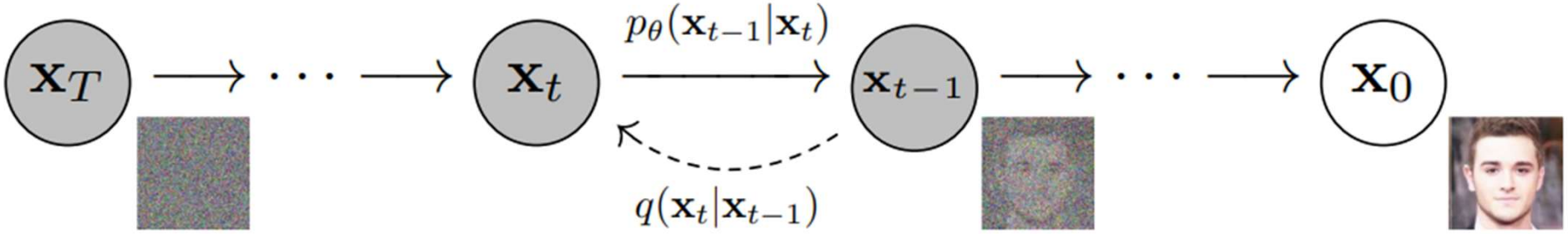


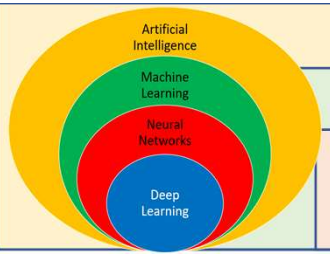
A Markov chain manifested for image data

Diffusion Models – Introduction

- Ultimately, the image is transformed to pure Gaussian noise. The goal of training a diffusion model is to learn the reverse process - i.e. training $p_\theta(x_{t-1}|x_t)$.

- By traversing backwards along this chain, we can generate new data.





Artificial Intelligence (AI)

Machine Learning (ML)

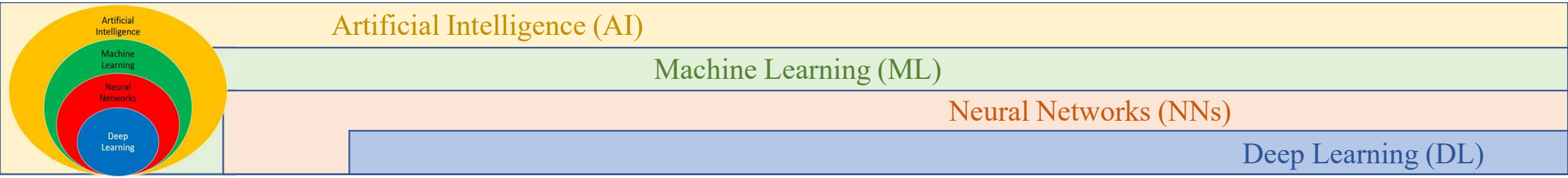
Neural Networks (NNs)

Deep Learning (DL)

Benefits of Diffusion Models

- Research into Diffusion Models has exploded in recent years.
- Diffusion Models currently produce State-of-the-Art image quality, examples of which can be seen below:



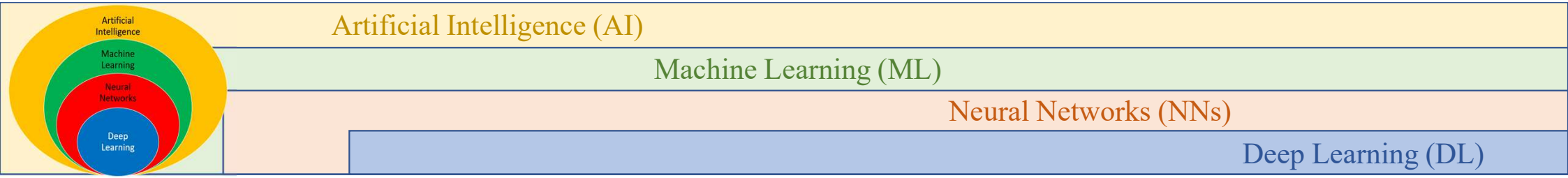


Benefits of Diffusion Models

- Beyond cutting-edge image quality, Diffusion models come with a host of other benefits, including **not requiring adversarial training**.

- The difficulties of adversarial training are well-documented; and, in cases where non-adversarial alternatives exist with comparable performance and training efficiency, it is usually best to utilize them.

- On the topic of training efficiency, Diffusion models also have the added benefits of scalability and parallelizability.



Challenges of adversarial training: Training Instability

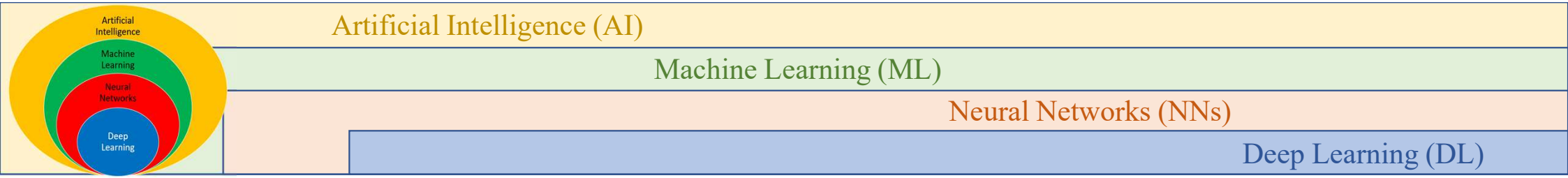
- let's focus purely on the challenges of adversarial training (the difficulties that GANs face because of their adversarial setup). These are the key issues that diffusion models largely avoid:

- GAN training is a minimax game:

$$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

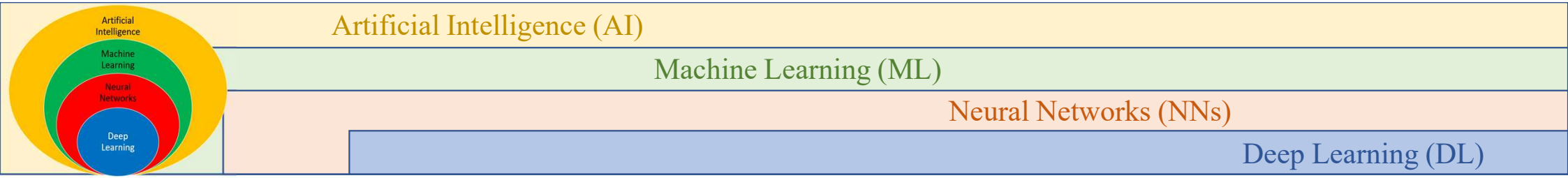
- The generator G and discriminator D are updating in opposition, which makes optimization unstable.

- Instead of converging to equilibrium, training often oscillates or collapses.



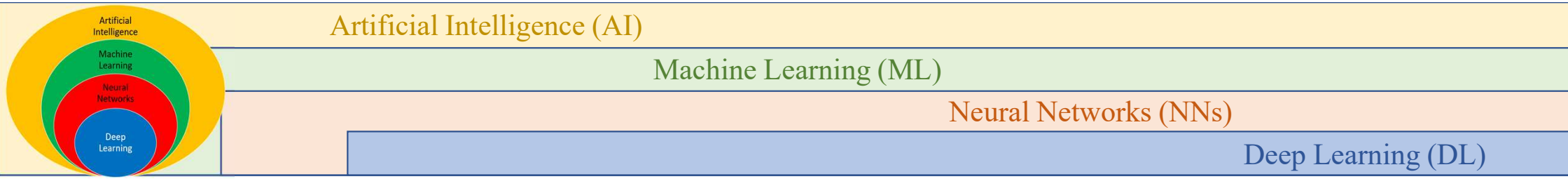
Challenges of adversarial training: Mode Collapse

- The generator sometimes learns to produce only a few kinds of outputs that fool the discriminator, ignoring the rest of the data distribution.
- For example, a GAN trained on digits might generate only “3” and “7,” while ignoring other numbers.



Challenges of adversarial training: Vanishing Gradients

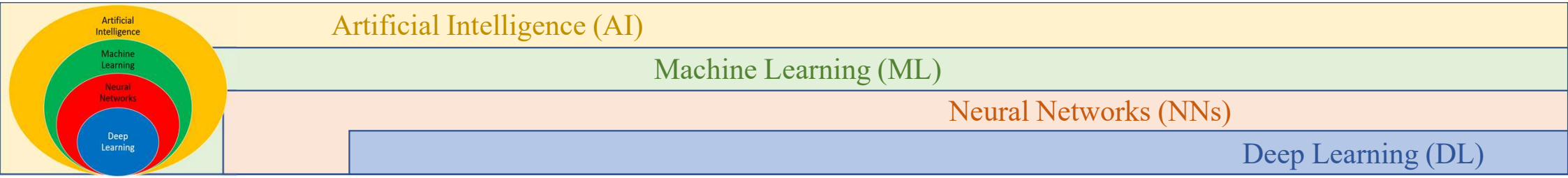
- If the discriminator becomes too strong, the generator receives almost no gradient signal to improve, stalling training.
- Conversely, if the discriminator is too weak, the generator "wins" too easily, producing low-quality samples.



Challenges of adversarial training: Hyperparameter Sensitivity

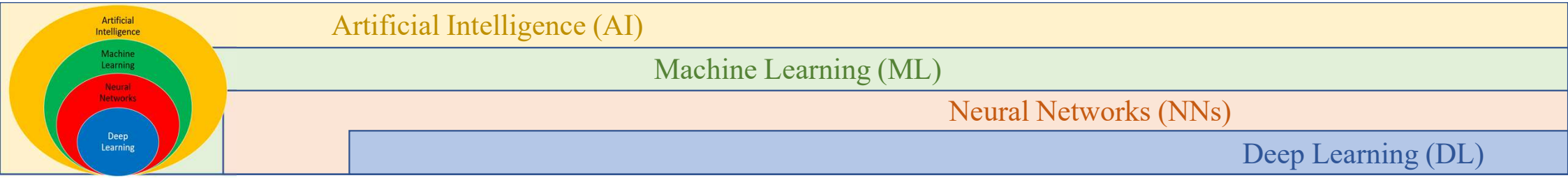
- GANs are notoriously sensitive to Learning rates (even slight misalignment can destabilize training), network architectures (small changes affect convergence), and batch size and normalization schemes.

- This makes them difficult to reproduce and tune.



Challenges of adversarial training: Evaluation Challenges

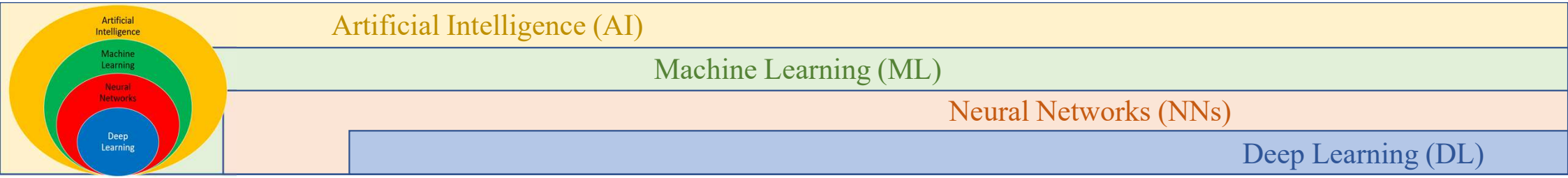
- GANs don't model an explicit probability distribution, so evaluating sample quality and diversity is tricky.
- Metrics like FID (Fréchet Inception Distance) or IS (Inception Score) are only proxies, not true likelihood-based measures.



Challenges of adversarial training: Resource Intensiveness

- Because of instability and trial-and-error tuning, training GANs often requires large compute budgets and long experimentation cycles.

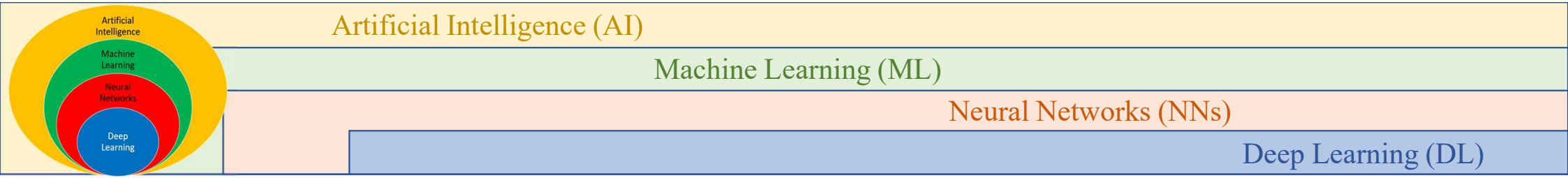
- Diffusion models, while slower in sampling, are generally more predictable during training.



Challenges of adversarial training: Discriminator Forgetting

- The discriminator can "forget" previously learned distinctions as the generator evolves, leading to instability or regressions in sample quality.

- In short: adversarial training struggles with instability, mode collapse, vanishing gradients, extreme sensitivity to tuning, and lack of reliable evaluation tools. These challenges make GANs hard to scale and reproduce, which is one of the main reasons diffusion models gained momentum.



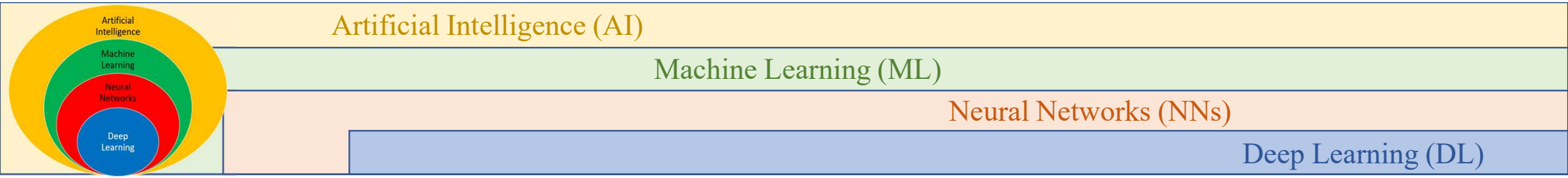
Benefits of Diffusion Models: No Adversarial Training Instability

❖ GANs pit a generator against a discriminator in a minimax game. This setup is notoriously unstable:

✓ Mode collapse (generator produces limited diversity).
✓ Non-convergence (training oscillates instead of stabilizing).
✓ Sensitivity to architecture and hyperparameters.

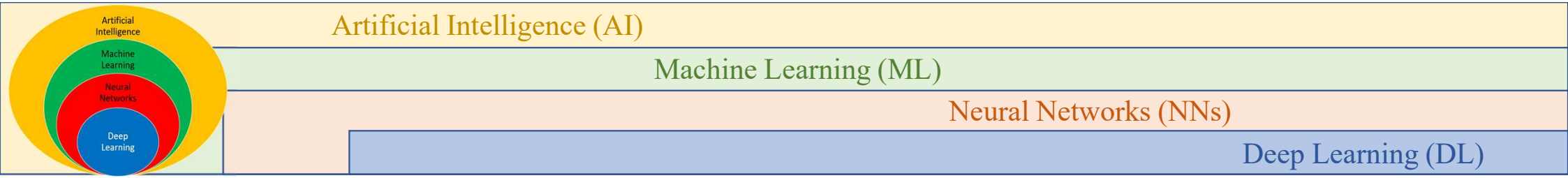
❖ Diffusion models, on the other hand, use a likelihood-based training objective (denoising score matching or variational lower bound).

❖ They optimize a well-defined log-likelihood, which makes training much more stable and predictable, without the tug-of-war dynamics.



Benefits of Diffusion Models: Likelihood-Based Training

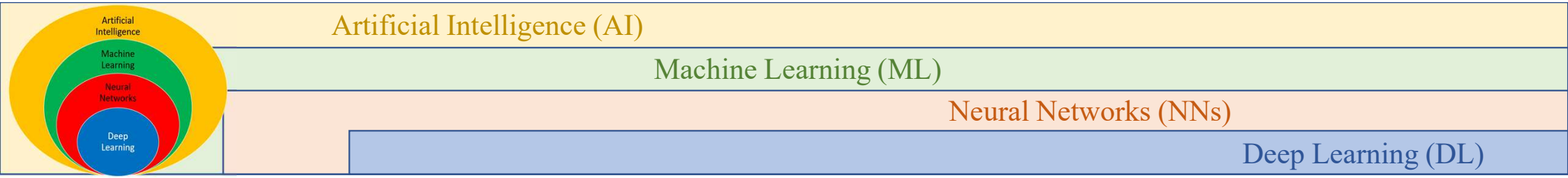
- Diffusion models explicitly approximate the data distribution via a sequence of denoising steps.
- This means they can provide tractable likelihood estimates, unlike GANs which only generate samples without an explicit probability model.
- This property also helps with evaluation and integration with downstream probabilistic models.



Benefits of Diffusion Models: Mode Coverage and Diversity

- GANs often drop modes (ignore parts of the data distribution). For example, a GAN trained on faces might generate only certain types of faces.

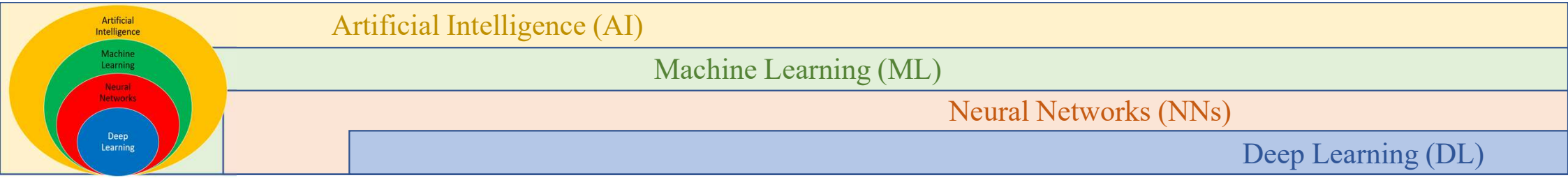
- Diffusion models, thanks to their probabilistic denoising process, cover the full distribution better, producing more diverse and representative samples.



Benefits of Diffusion Models: Flexibility in Conditioning

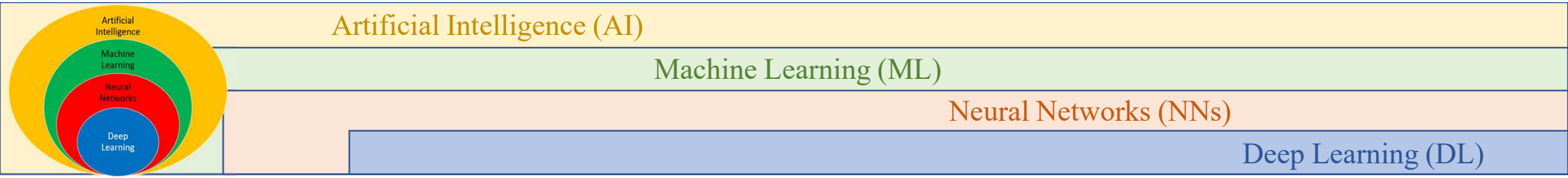
- Diffusion models are inherently modular. By injecting conditioning signals (like text, class labels, segmentation maps, audio), one can easily adapt them to text-to-image (e.g., Stable Diffusion), inpainting, super-resolution, etc.

- GANs can do conditioning too, but it often requires retraining or architectural tweaks. Diffusion handles it more gracefully via guidance mechanisms.



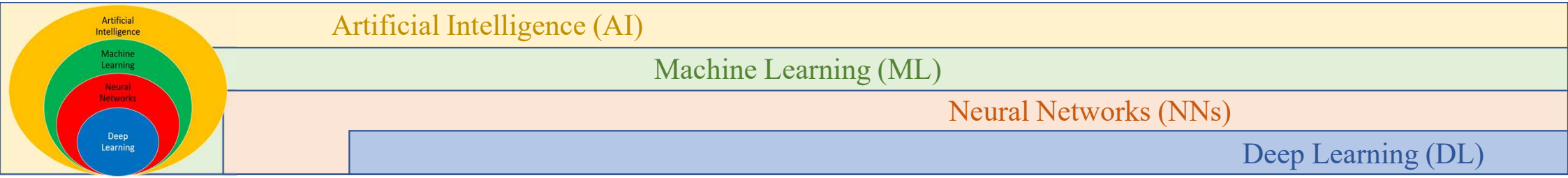
Benefits of Diffusion Models: High Fidelity + Coherence

- Diffusion models achieve fine-grained detail because they iteratively refine noise into structure.
- Each denoising step can correct small mistakes, leading to highly coherent global structure and crisp local details.
- GANs, generating in one shot, can struggle with balancing global structure and local detail.



Benefits of Diffusion Models: Better Trade-off Between Speed and Quality

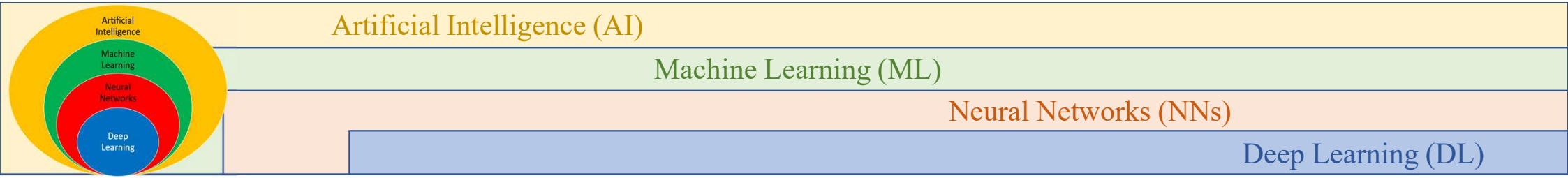
- Diffusion sampling is traditionally slow (hundreds of denoising steps).
- But recent advances (DDIM, DPM-Solver, consistency models, rectified flow, etc.) allow fast generation without sacrificing much quality.
- Unlike GANs, where speeding up sampling usually means training a new model, diffusion can flexibly trade off speed vs. quality during inference.



Benefits of Diffusion Models: Robustness to Training Data Size and Quality

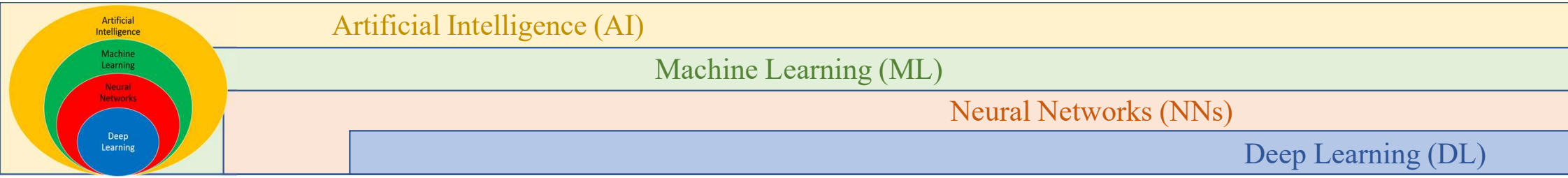
- Diffusion models often generalize well even with noisy or imperfect data, since the denoising task inherently teaches robustness.

- GANs usually need carefully curated, large datasets to avoid artifacts and instability.



To sum up ....

In short, Diffusion models go beyond just better images. They're easier to train, more stable, probabilistically grounded, mode-covering, and highly flexible for conditioning and control—all while avoiding the fragile adversarial training loop of GANs.



Diffusion Models - A Deep Dive

- As mentioned above, a Diffusion Model consists of a forward process (or diffusion process), in which a datum (generally an image) is progressively noised, and a reverse process (or reverse diffusion process), in which noise is transformed back into a sample from the target distribution.

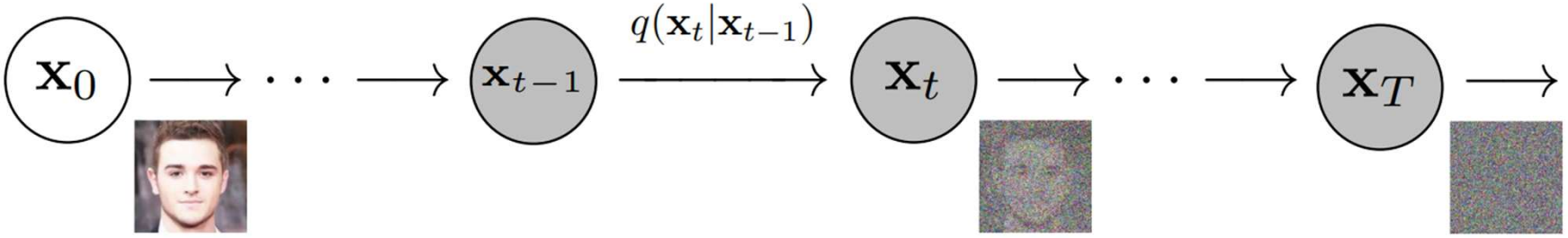
- The sampling chain transitions in the forward process can be set to conditional Gaussians when the noise level is sufficiently low. Combining this fact with the Markov assumption leads to a simple parameterization of the forward process:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

where $\beta_1, \cdots, \beta_T$ is a variance schedule (either learned or fixed) which, if well-behaved, ensures that x_T is nearly an isotropic Gaussian for sufficiently large T.

Diffusion Models - A Deep Dive

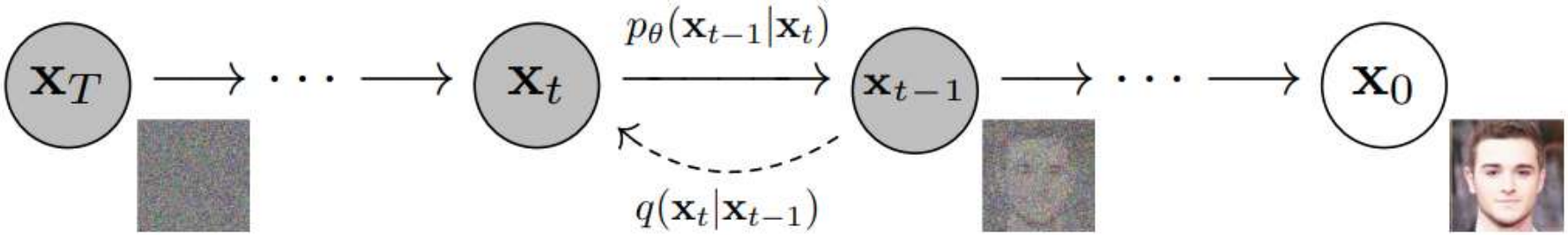
- As mentioned previously, the "magic" of diffusion models comes in the reverse process.
- During training, the model learns to reverse this diffusion process in order to generate new data.
- Starting with the pure Gaussian noise $p(x_T) = \mathcal{N}(x_T, 0, I)$, the model learns the joint distribution $p_\theta(x_{0:T})$ as:



Diffusion Models - A Deep Dive

- where the time-dependent parameters of the Gaussian transitions are learned. Note in particular that the Markov formulation asserts that a given reverse diffusion transition distribution depends only on the previous timestep (or following timestep, depending on how you look at it):

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



Diffusion Models - Training

- A Diffusion Model is trained by finding the reverse Markov transitions that maximize the likelihood of the training data.

- In practice, training equivalently consists of minimizing the variational upper bound on the negative log likelihood.

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L_{vlb}$$

Note that L_{vlb} is technically an upper bound (the negative of the ELBO) which we are trying to minimize, but we refer to it as L_{vlb} for consistency with the literature.

What is the KL Divergence?

- The mathematical form of the KL divergence for continuous distributions is:

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$

- Below you can see the KL divergence of a varying distribution P from a reference distribution Q.

Casting Lvlb in Terms of KL Divergences

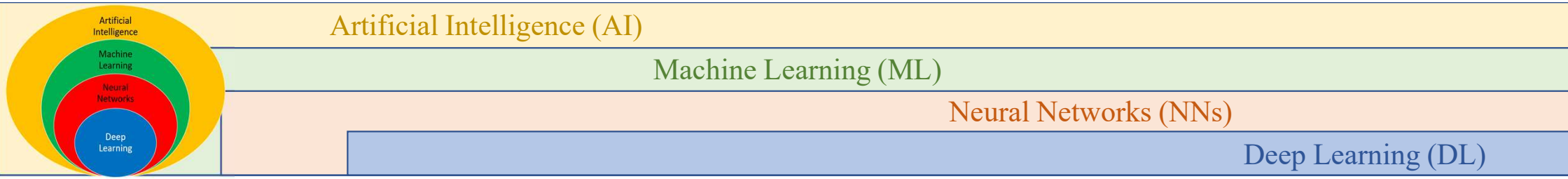
❖ As mentioned previously, it is possible to rewrite Lvlb almost completely in terms of KL divergences:

$$L_{vlb} = L_0 + L_1 + ... + L_{T-1} + L_T$$

$$L_0 = -\log p_\theta(x_0|x_1)$$

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$

$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T))$$



Model Choices

- With the mathematical foundation for our objective function established, we now need to make several choices regarding how our Diffusion Model will be implemented.

- For the forward process, the only choice required is defining the variance schedule, the values of which are generally increasing during the forward process.

- For the reverse process, we much choose the Gaussian distribution parameterization / model architecture(s). Note the high degree of flexibility that Diffusion Models afford - the only requirement on our architecture is that its input and output have the same dimensionality.

Forward Process and LT

- As noted above, regarding the forward process, we must define the variance schedule. In particular, we set them to be time-dependent constants, ignoring the fact that they can be learned.

- For example, a linear schedule from β1=10−4 to βT=0.2 might be used, or perhaps a geometric series.

- Regardless of the particular values chosen, the fact that the variance schedule is fixed results in L_T becoming a constant with respect to our set of learnable parameters, allowing us to ignore it as far as training is concerned.

$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T))$$

Reverse Process and L_1:T−1

- Now we discuss the choices required in defining the reverse process. Recall from above we defined the reverse Markov transitions as a Gaussian:

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

- We must now define the functional forms of μ_θ or Σ_θ. While there are more complicated ways to parameterize Σ_θ, we simply set:

$$\boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2 \mathbb{I}$$
$$\sigma_t^2 = \beta_t$$

Reverse Process and $L_{1:T-1}$

- That is, we assume that the multivariate Gaussian is a product of independent gaussians with identical variance, a variance value which can change with time. We set these variances to be equivalent to our forward process variance schedule.

- Given this new formulation of Σ_θ, we have

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

- which allows us to transform:

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$

- to

$$L_{t-1} \propto ||\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)||^2$$

Reverse Process and $L_{1:T-1}$

- Where the first term in the difference is a linear combination of xt and x0 that depends on the variance schedule βt.

- The significance of the above proportion is that the most straightforward parameterization of μ_θ simply predicts the diffusion posterior mean. Importantly, the researchers actually found that training μ_θ to predict the noise component at any given timestep yields better results. In particular, let:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

$$\alpha_t := 1 - \beta_t \quad \text{and} \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

Reverse Process and $L_{1:T-1}$

- This leads to the following alternative loss function, which the researchers found to lead to more stable training and better results:

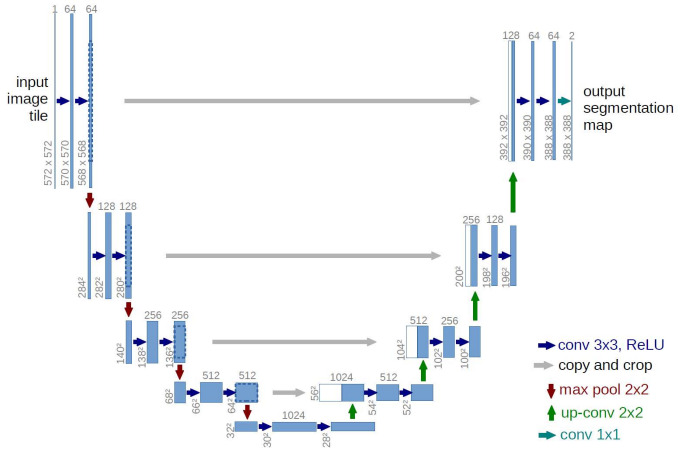
$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

- The researchers also note connections of this formulation of Diffusion models to score-matching generative models based on Langevin dynamics. Indeed, it appears that Diffusion Models and Score-Based models may be two sides of the same coin, akin to the independent and concurrent development of wave-based quantum mechanics and matrix-based quantum mechanics revealing two equivalent formulations of the same phenomena.

Network Architecture

- While our simplified loss function seeks to train a model ϵ_θ, we have still not yet defined the architecture of this model. Note that **the only requirement for the model is that its input and output dimensionality are identical.**

- Given this restriction, it is perhaps unsurprising that image Diffusion models are commonly implemented with U-Net-like architectures.



Reverse Process Decoder and L0

- The path along the reverse process consists of many transformations under continuous conditional Gaussian distributions. At the end of the reverse process, recall that we are trying to produce an image, which is composed of integer pixel values. Therefore, we must devise a way to obtain discrete (log) likelihoods for each possible pixel value across all pixels.

- The way that this is done is by setting the last transition in the reverse diffusion chain to an independent discrete decoder. To determine the likelihood of a given image $\boldsymbol{x_0}$ given $\boldsymbol{x_1}$, we first impose independence between the data dimensions:

$$p_\theta(x_0|x_1) = \prod_{i=1}^{D} p_\theta(x_0^i|x_1^i)$$

where D is the dimensionality of the data and the superscript i indicates the extraction of one coordinate.

Reverse Process Decoder and L_0

- The goal now is to determine how likely each integer value is for a given pixel given the distribution across possible values for the corresponding pixel in the slightly noised image at time t=1:

$$\mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2)$$

- where the pixel distributions for t=1 are derived from the below multivariate Gaussian whose diagonal covariance matrix allows us to split the distribution into a product of univariate Gaussians, one for each dimension of the data:

$$\mathcal{N}(x; \mu_\theta(x_1, 1), \sigma_1^2 \mathbb{I}) = \prod_{i=1}^{D} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2)$$

Reverse Process Decoder and L_0

- We assume that the images consist of integers in 0,1,...,255 (as standard RGB images do) which have been scaled linearly to [−1,1].
- We then break down the real line into small "buckets", where, for a given scaled pixel value x, the bucket for that range is [x−1/255,x+1/255].
- The probability of a pixel value x, given the univariate Gaussian distribution of the corresponding pixel in x1, is the area under that univariate Gaussian distribution within the bucket centered at x.

Reverse Process Decoder and L_0

- Given a t=0 pixel value for each pixel, the value of $p_\theta(x_0|x_1)$ is simply their product. This process is succinctly encapsulated by the following equation:

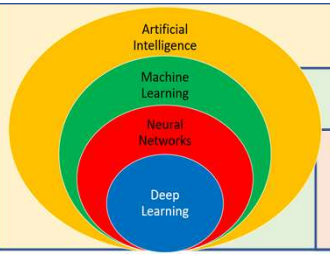
$$p_\theta(x_0|x_1) = \prod_{i=1}^{D} p_\theta(x_0^i|x_1^i) = \prod_{i=1}^{D} \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(x_1, 1), \sigma_1^2) dx$$

$$\delta_-(x) = \begin{cases} -\infty & x = -1 \\ x - \frac{1}{255} & x > -1 \end{cases}$$

$$\delta_+(x) = \begin{cases} \infty & x = 1 \\ x + \frac{1}{255} & x < 1 \end{cases}$$

- Given this equation for $p_\theta(x_0|x_1)$, we can calculate the final term of Lvlb which is not formulated as a KL Divergence:

$$L_0 = -\log p_\theta(x_0|x_1)$$



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

Final Objective

- As mentioned above, the researchers found that predicting the noise component of an image at a given timestep produced the best results. Ultimately, the objective is defined as follows:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\, \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2 \,\right]$$

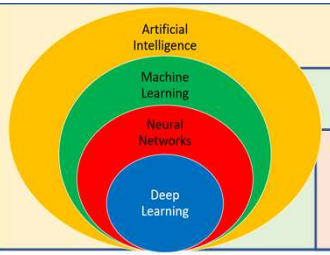
- The training and sampling algorithms for our Diffusion model therefore can be succinctly captured in the below figure:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2$
6: **until** converged

Algorithm 2 Sampling

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



Artificial Intelligence (AI)

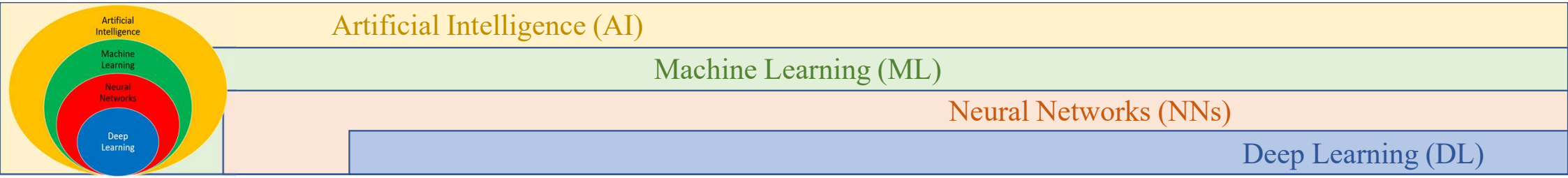
Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

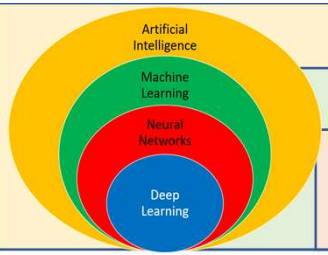
Diffusion Model Theory Summary

- In this section we took a detailed dive into the theory of Diffusion models. It can be easy to get caught up in mathematical details, so we note the most important points within this section below in order to keep ourselves oriented from a birds-eye perspective:
- Diffusion model is parameterized as a Markov chain, meaning that our latent variables $x_1, \cdots, x_T$ depend only on the previous (or following) timestep.
- The transition distributions in the Markov chain are Gaussian, where the forward process requires a variance schedule, and the reverse process parameters are learned.
- The diffusion process ensures that x_T is asymptotically distributed as a Gaussian for sufficiently large T.
- In our case, the variance schedule was fixed, but it can be learned as well. The variances are generally increasing with time in the series (i.e. $\beta i < \beta j$ for $i < j$).



Diffusion Model Theory Summary

- Diffusion models are highly flexible and allow for any architecture whose input and output dimensionality are the same to be used. Many implementations use U-Net-like architectures.
- The training objective is to maximize the likelihood of the training data. This is manifested as tuning the model parameters to minimize the variational upper bound of the negative log likelihood of the data.
- Almost all terms in the objective function can be cast as KL Divergences as a result of our Markov assumption. These values become tenable to calculate given that we are using Gaussians, therefore omitting the need to perform Monte Carlo approximation.
- Ultimately, using a simplified training objective to train a function which predicts the noise component of a given latent variable yields the best and most stable results.
- A discrete decoder is used to obtain log likelihoods across pixel values as the last step in the reverse diffusion process.



Artificial Intelligence (AI)

Machine Learning (ML)

Neural Networks (NNs)

Deep Learning (DL)

