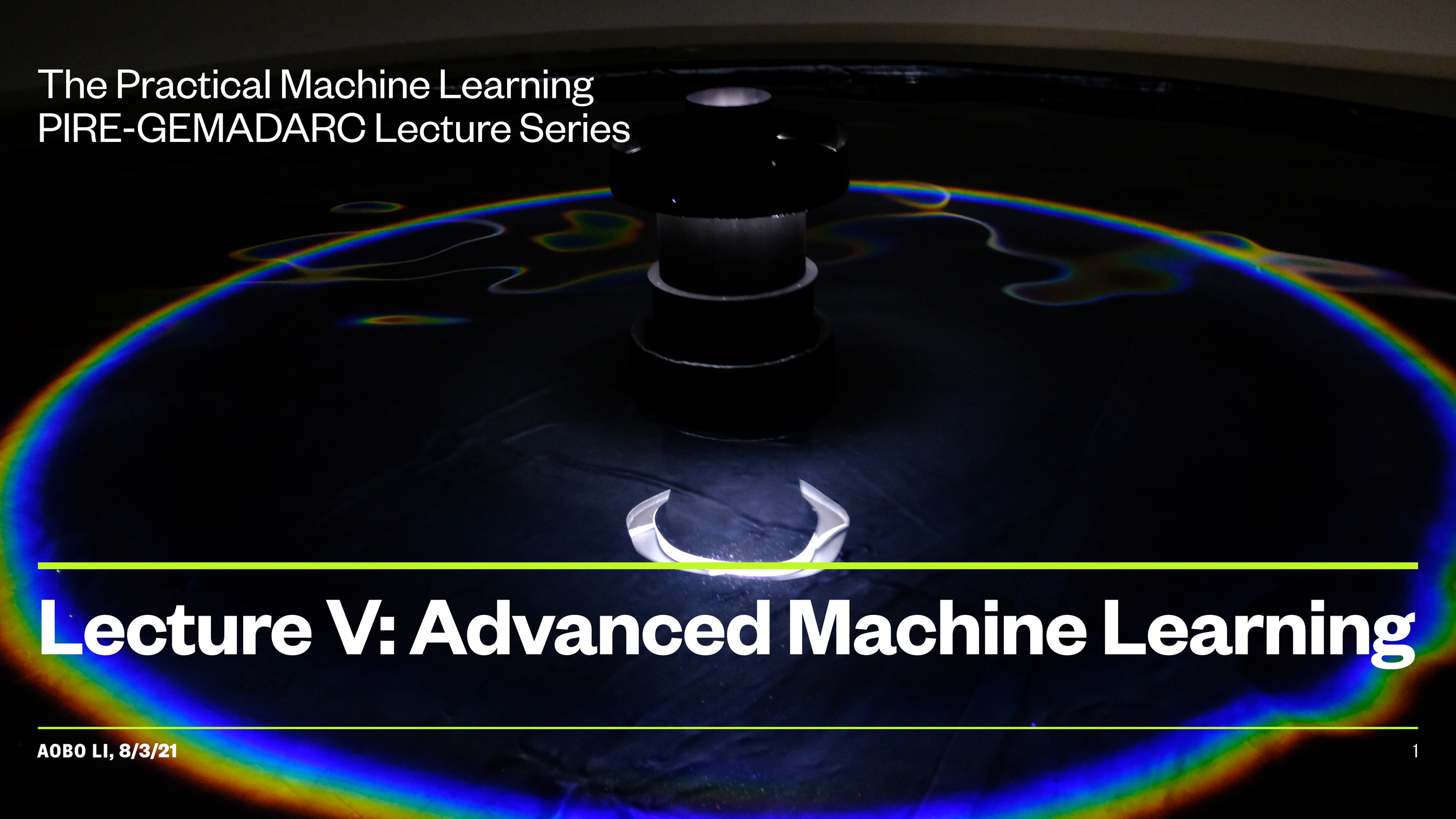


The Practical Machine Learning
PIRE-GEMADARC Lecture Series

# Lecture V: Advanced Machine Learning

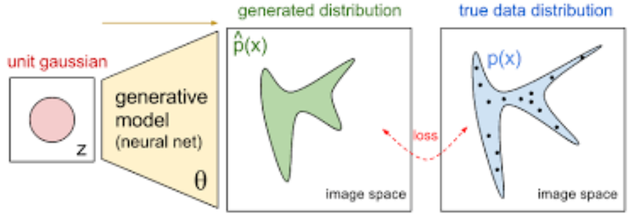
AOBO LI, 8/3/21

# Outline

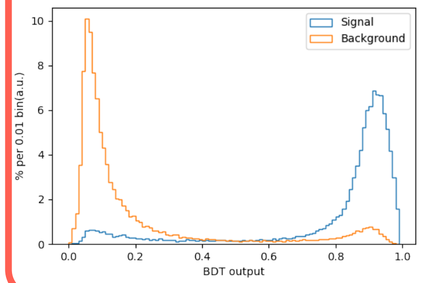
- Overview
- Generative Model
- Representation Learning
  - Transfer Learning
  - Self-supervised Learning
- ML Model Optimization
- Course Summary & Thanks


# Advanced Machine Learning

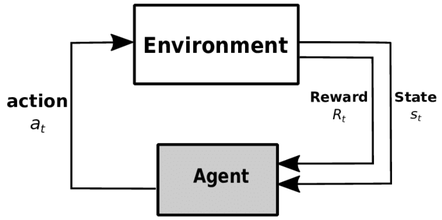
## GENERATIVE MODEL



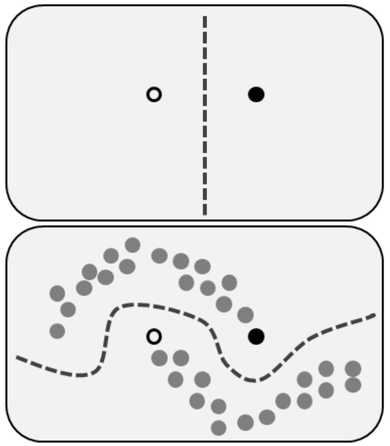

## CLASSIFICATION & REGRESSION




## REINFORCEMENT LEARNING



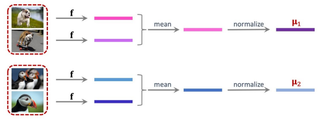

## SEMI-SUPERVISED LEARNING



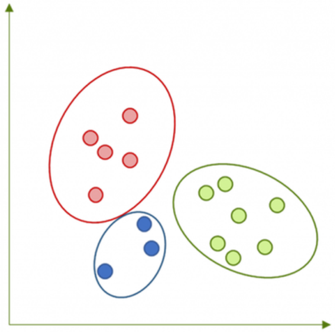
## PATTERN RECOGNITION, META LEARNING .....

## FEW SHOTS LEARNING

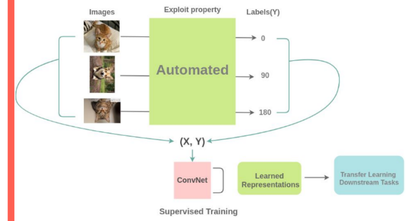
Few-Shot Learning:
Pretraining + Fine Tuning




## CLUSTERING



## CONTRASTIVE REPRESENTATION LEARNING




Labeled

Delayed label

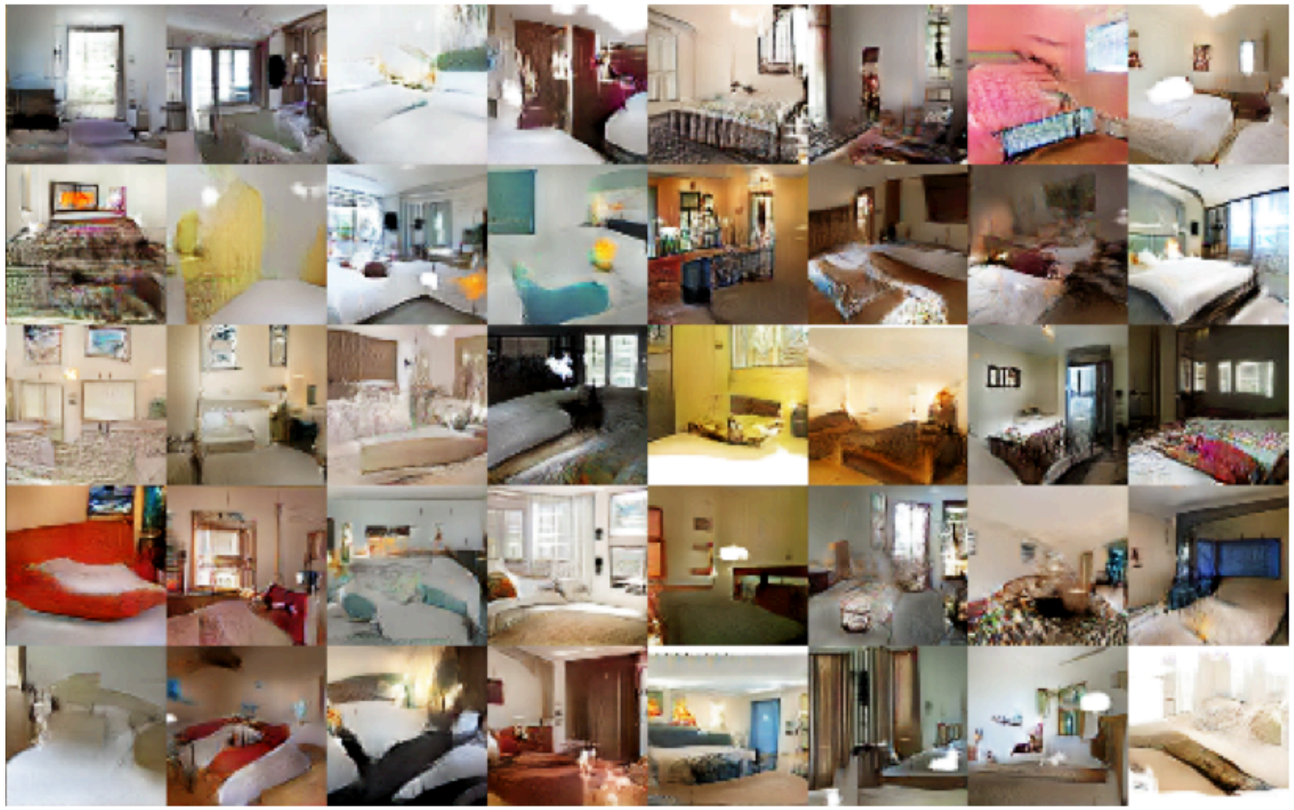
Both labeled and unlabeled

Very little labeled

No Label

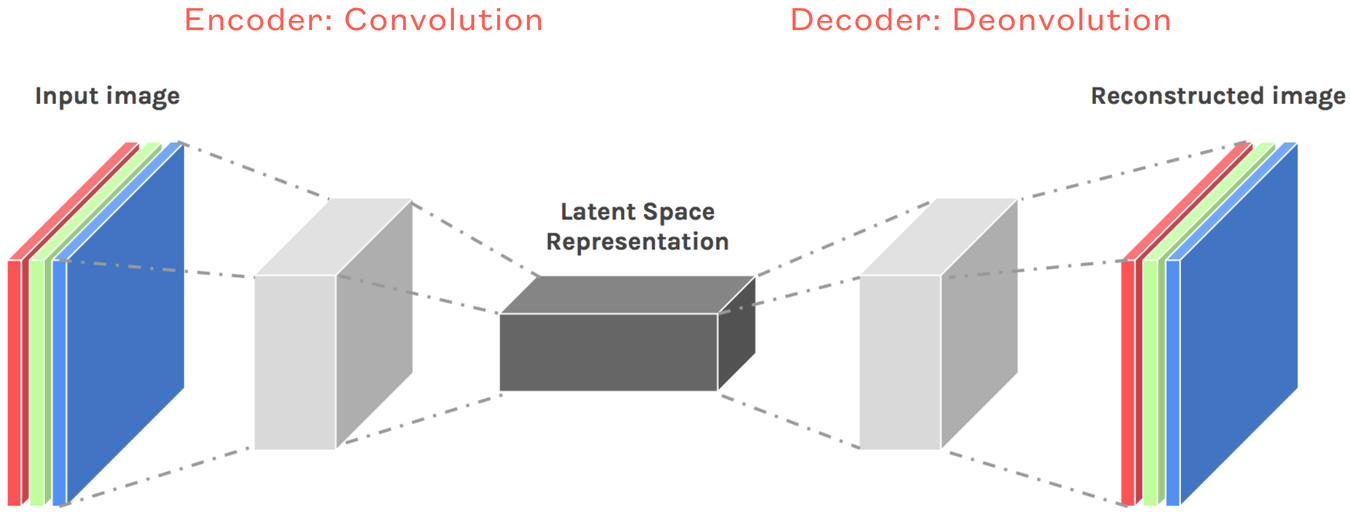
# Generative Model

- Unsupervised learning, does not require label
- Generate fake images indistinguishable to real images
- Formulate generative model using Bayes Rule:
  - $p(X|Z) = \frac{p(Z)p(Z|X)}{p(X)}$
  - Z: Randomly sampled generating seed
  - X: Image we'd like to generate
  - NN models try to approximate the likelihood function $p(Z|X)$ in order to produce images from arbitrary input Z



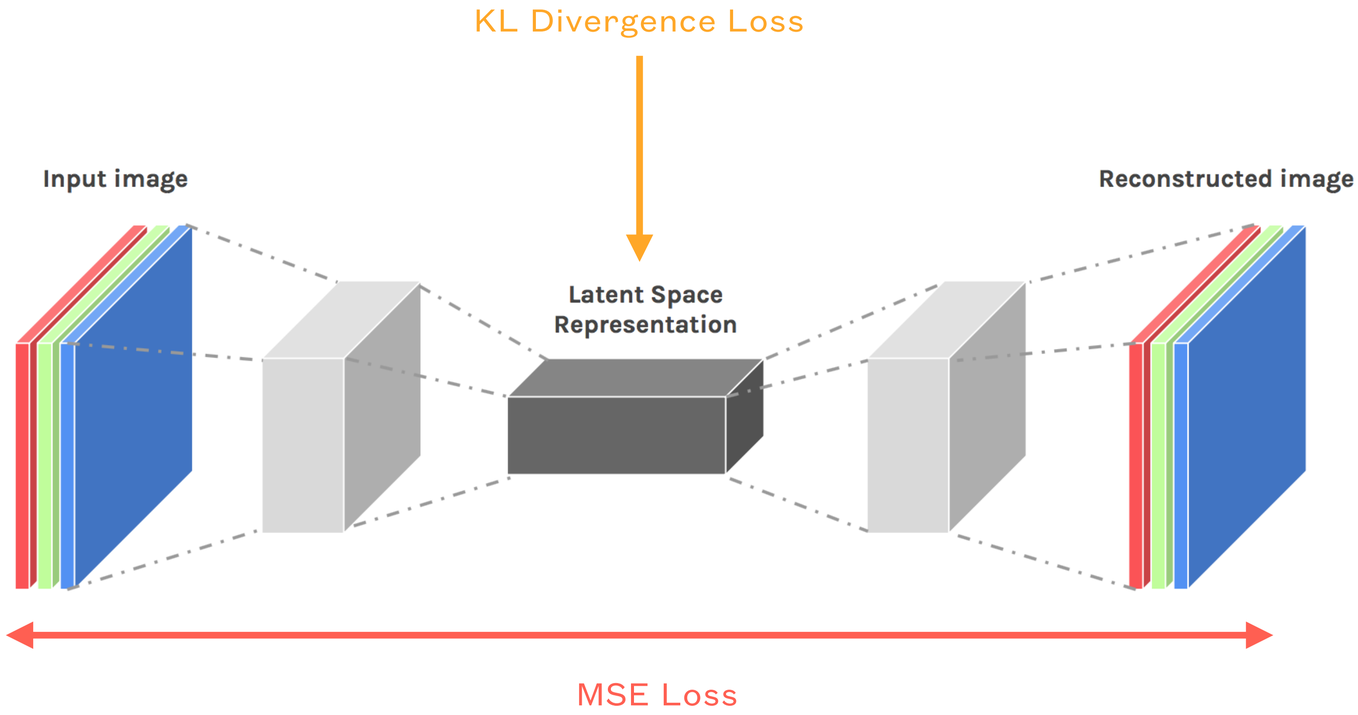
# Convolutional Autoencoder

- Concatenating two CNNs back-to-back
  - Encoder: stacked convolutional layers
  - Decoder: stacked deconvolutional layers
- Train to reconstruct the input image
  - Minimize the MSE loss between input images and reconstructed images
- Each image is encoded into a latent space vector by the encoder
- The image can be reconstructed from the latent space vector via the decoder



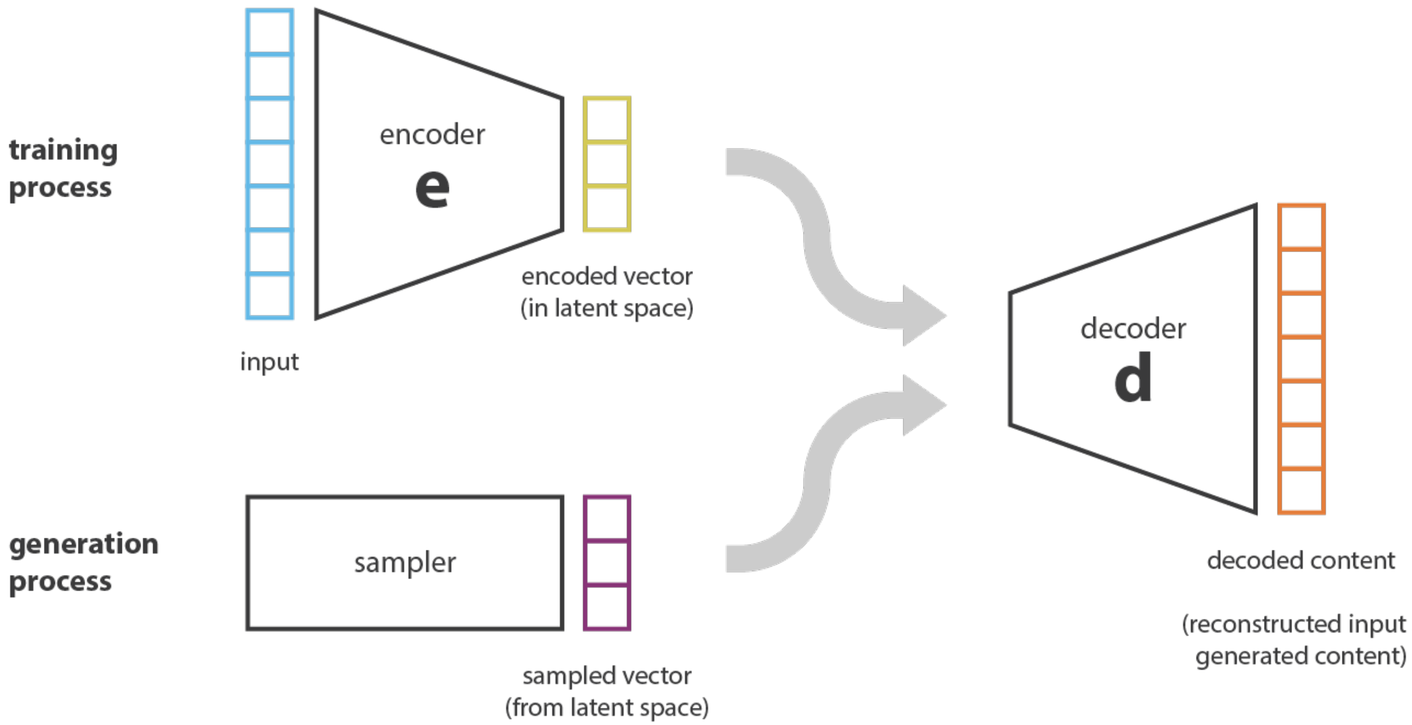
# Variational Autoencoder

- We can modify Convolutional Autoencoder to generate fake images
- In CAE, the latent space vector contains all information needed to reconstruct an image
  - But it does not follow any particular distribution
  - Therefore, a valid value could fall in any range
- Variational Autoencoder add another loss to regulate the latent space vector
  - KL Divergence: measuring the distance between two probability distributions
  - Train to regulate the latent space vector to follow a multi-dimensional gaussian distribution



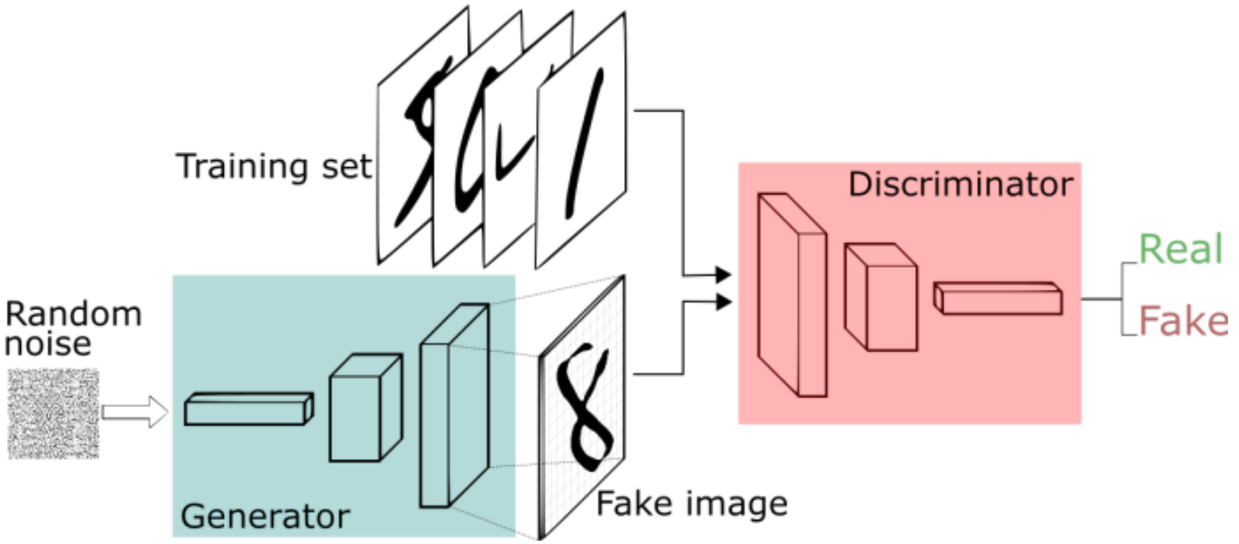
# Variational Autoencoder

- MSE Loss makes sure that generated images and input images are similar
- KL Divergence Loss makes sure that the latent space would follow a multi-dimensional gaussian distribution
- To generate fake images, we can sample a new value from the latent space, then feeding it to the decoder



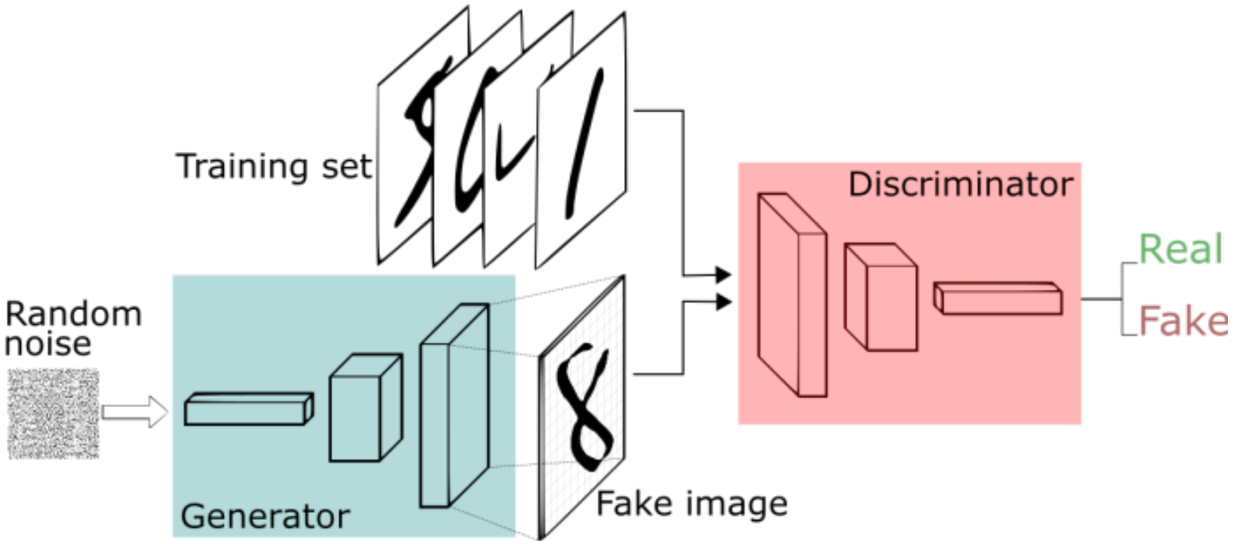
# Generative Adversarial Network

- Generative Adversarian Network (GAN) generate images using a different approach.
- Generator $G(x)$ exhibits the same structure as the decoder of CAE and VAE
  - It produces fake images from random noise
- Discriminator $D(x)$:
  - A typical binary NN classifier
  - Its goal is to classify whether the image is real or fake



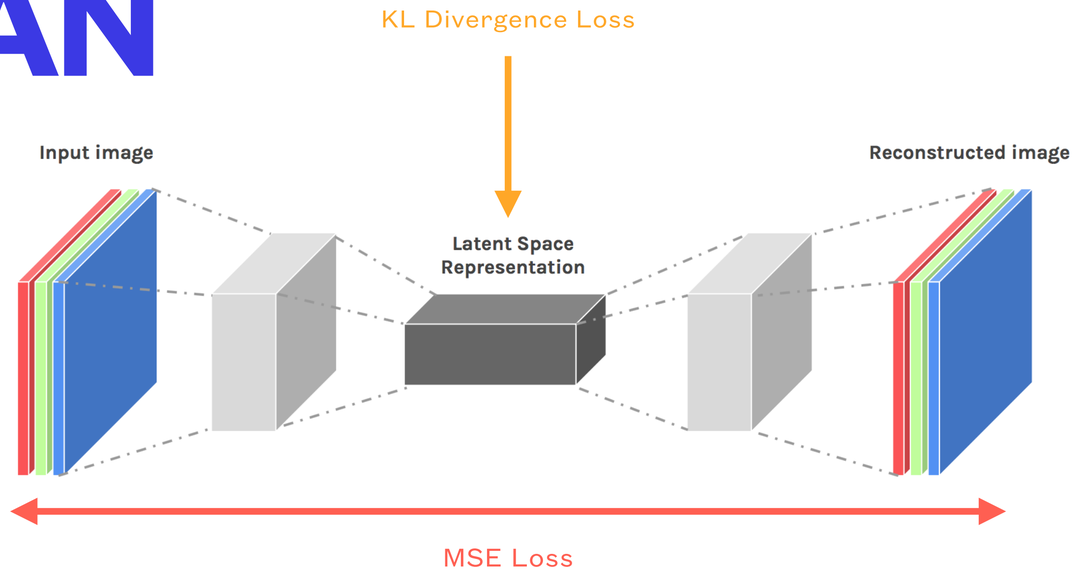
# Generative Adversarial Network

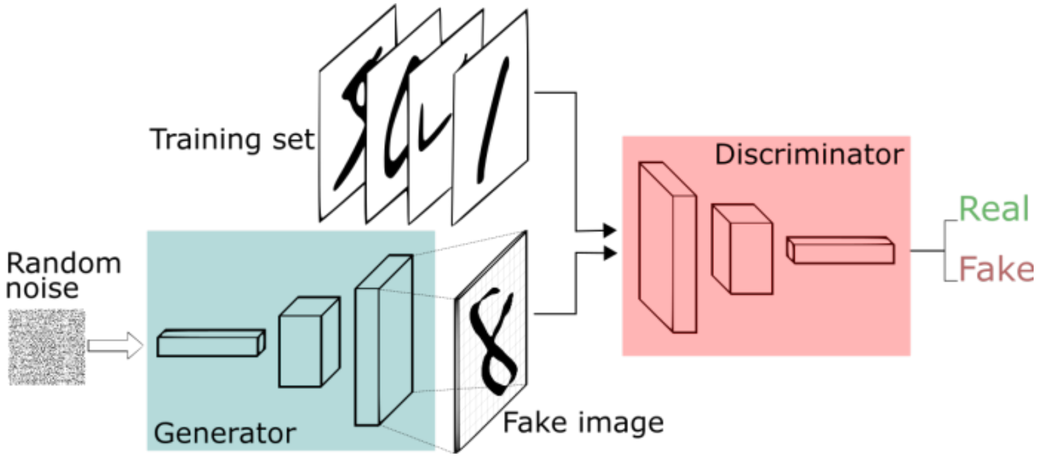
- When training the GAN:
  - Feed in each unlabeled real image to the network
  - For each real image, the generator will generate a fake image from random noise
  - The generator tries to fool the discriminator with the generated fake image
  - The discriminator tries to classify real images from the fake images
  - Adversarial loss: $E_x(log(D(x)) + E_z(1 - D(G(z)))$
  - Training goal: the generated fake image can no longer be distinguished from real images



# Compare VAE and GAN

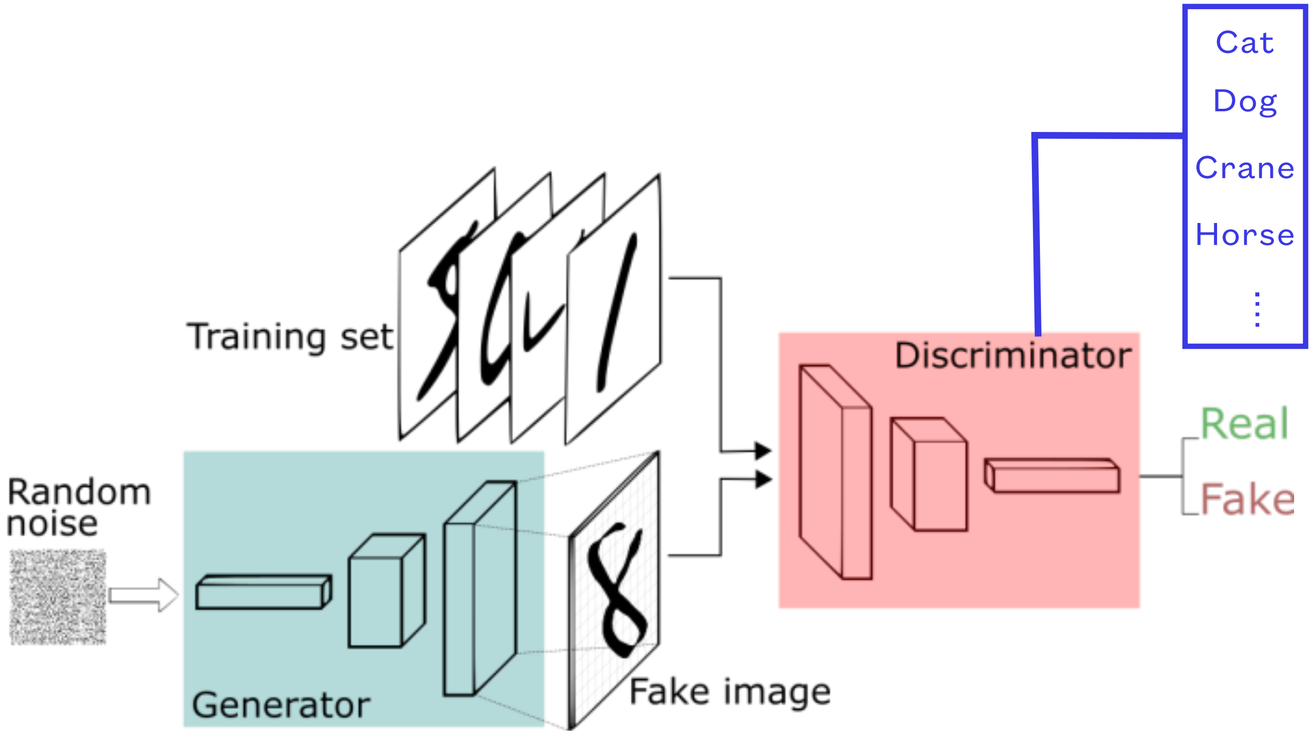
- VAE explicitly regulate the likelihood function $p(Z|X)$ through KL Divergence Loss
  - This isn't a very good approach, because $p(Z|X)$ does not have to follow a gaussian distribution
- On the other hand, GAN does not assume $p(Z|X)$ to take any specific form
  - $p(Z|X)$ is produced through the generator, then compared to the real images by discriminator
  - This gives the model more freedom to fit a proper likelihood function
- In general, GAN is a better generative model compared to vanilla VAE



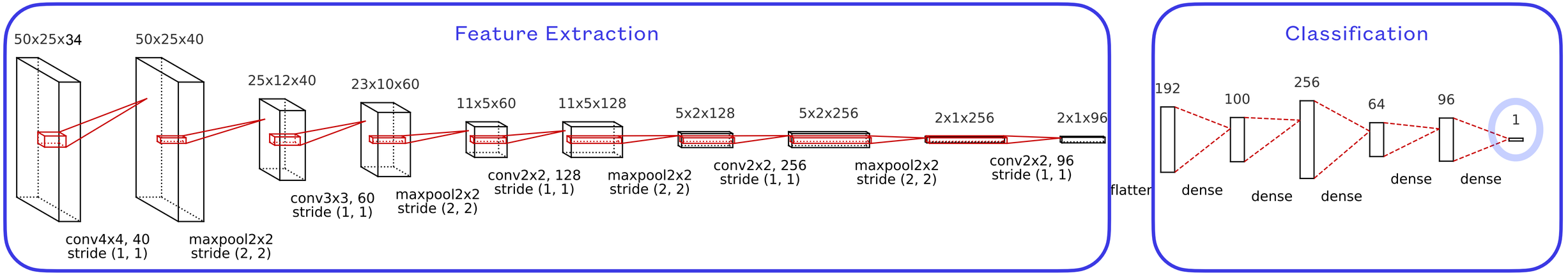



# Semi-Supervised Learning with GAN

- Suppose we have a dataset with both labeled and unlabeled data
  - We want labeled and unlabeled data to jointly contribute
- We can utilize GAN to perform Semi-Supervised Learning:
  - If unlabeled, ask the discriminator to classify for real/fake only
  - If labeled, ask discriminator to classify for real/fake and the class of animals
- Unlabeled data helps discrimination, while labeled data helps generation



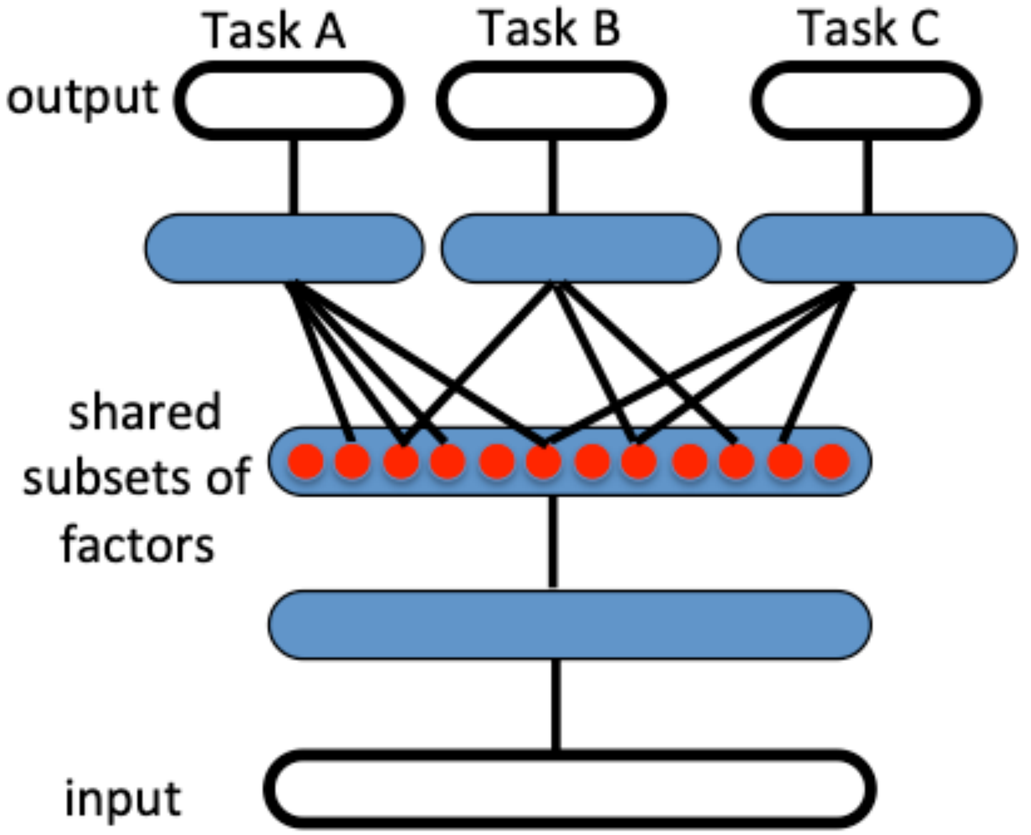
# Representation Learning




- An NN classifier contains two parts, a feature extractor and a classifier
  - Feature extractor: ConvLayers, RNN layers or MHSA layers
  - Classifier: stacks of fully connected NN
- Feature extractor encode each image into a specific representation
  - This representation is then fed to the classifier for classification decision

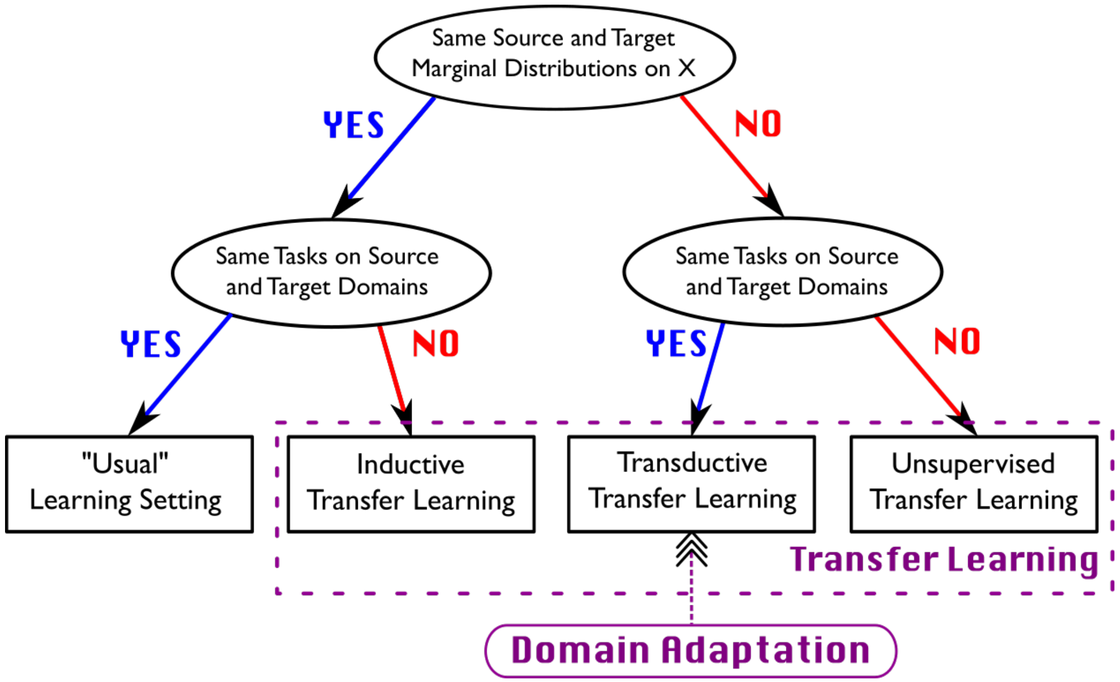
# Representation Learning

- Learning a representation instead of the classification decision is extremely useful
- The learned representation contains important features of the data
- This representation can then be transferred to different tasks.
- Even better, representation learning oftentimes does not require label:
  - This is a big advantage, because labeling data is an expensive process
  - Can be conducted in contrastive, self-supervised way



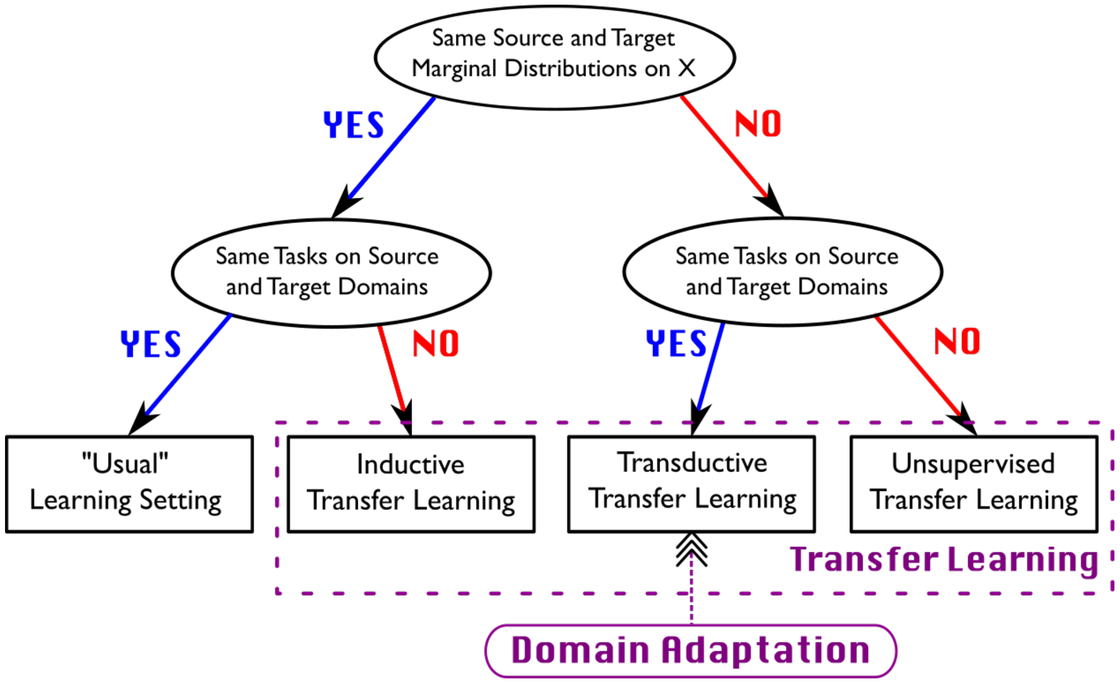
# Transfer Learning

- Transfer learning:
  - What has been learned in one setting is exploited to improve generalization in another setting
- Source Domain: Where we have a lot of data and would like to train our machine on
- Target Domain: Where we have limited data, but we'd like our machine to perform well on
- Source and target domain are different but correlated
- In this case, the representation is more useful than the classification decision



# Transfer Learning

- Transducive Transfer Learning:
  - Sometimes called domain adaptation
  - Same task, different data in the same feature space
  - To train a classifier for Pokemon, I can pull out the feature extractor of ImageNet, and hook a new classifier on top
  - Although ImageNet may not be trained on Pokemon, its representation is still useful for the extraction of lo
- Inductive Transfer Learning:
  - Different task, same data
  - To train a GAN to generate cat/dog, I can pre-train a classifier of cat/dog, then use it as the discriminator of GAN

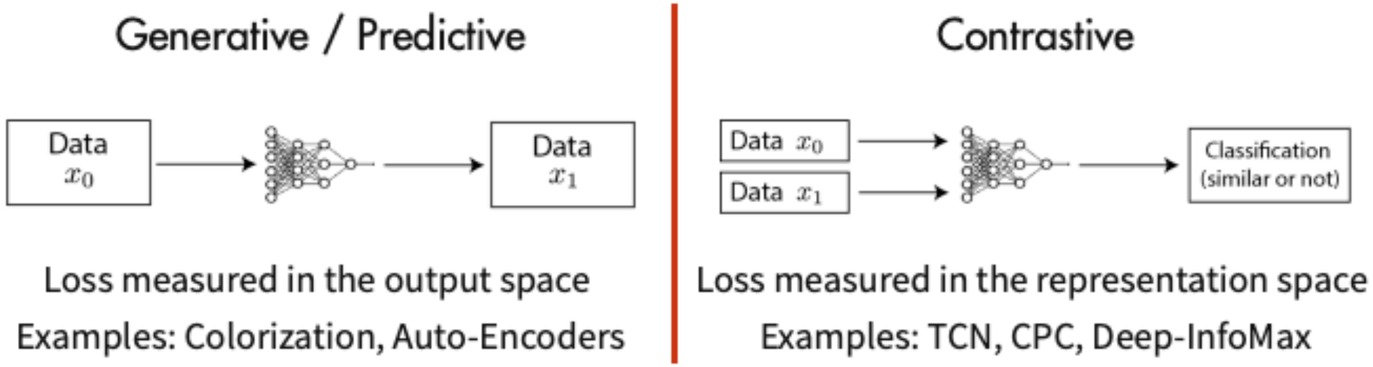


# Contrastive Self-Supervised Learning

Drawing with the dollar bill present


Drawing w/o the dollar bill present


When we learn, we do not learn all the gory details, but only the features that makes the object outstanding/ contrastive with similar objects.

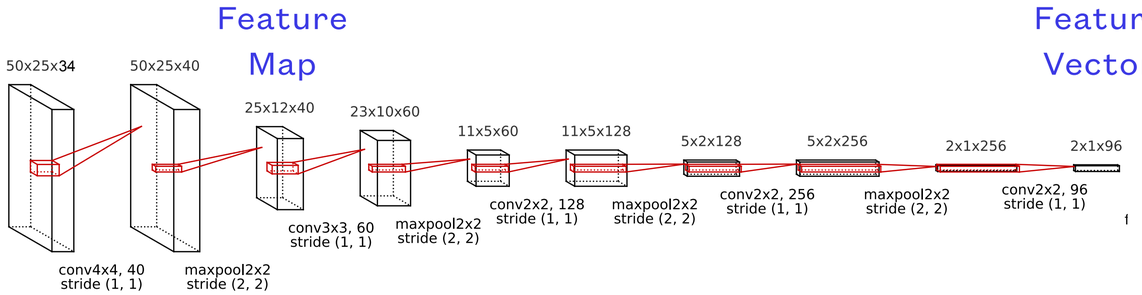


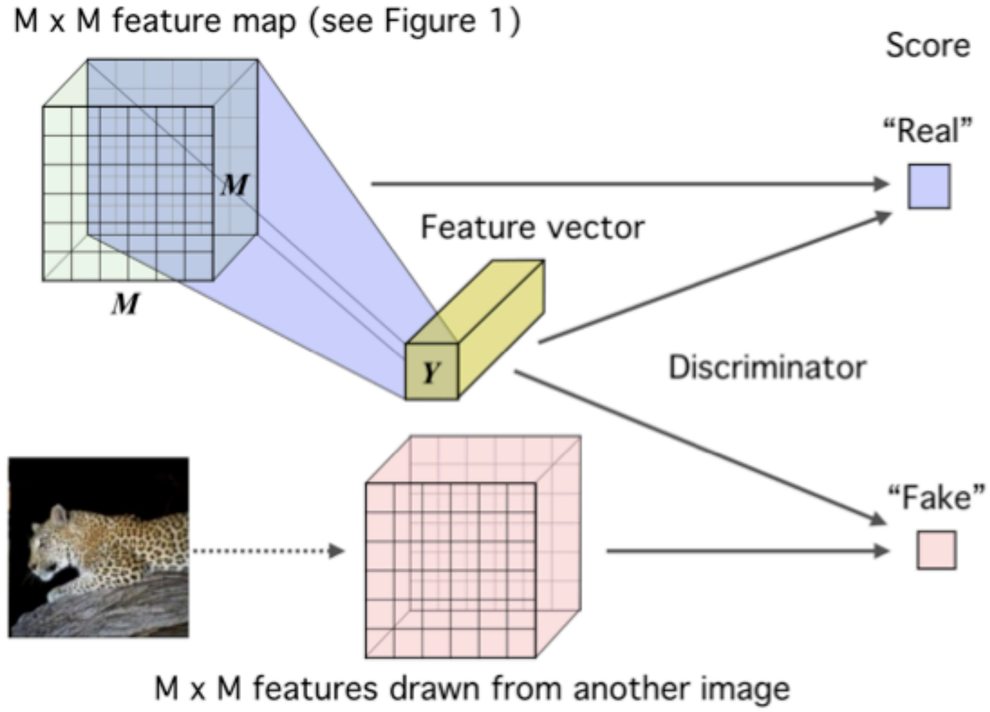

**Learning the Representation:**

- Learning an embedding/representation of each data so that similar data points are closer, and dissimilar datapoint are further away
- **DOES NOT** require any labeled data
- Performance has surpassed traditional supervised learning approach(AlexNet)
- Using a few anchors to regulate the representation space

# Deep InfoMax

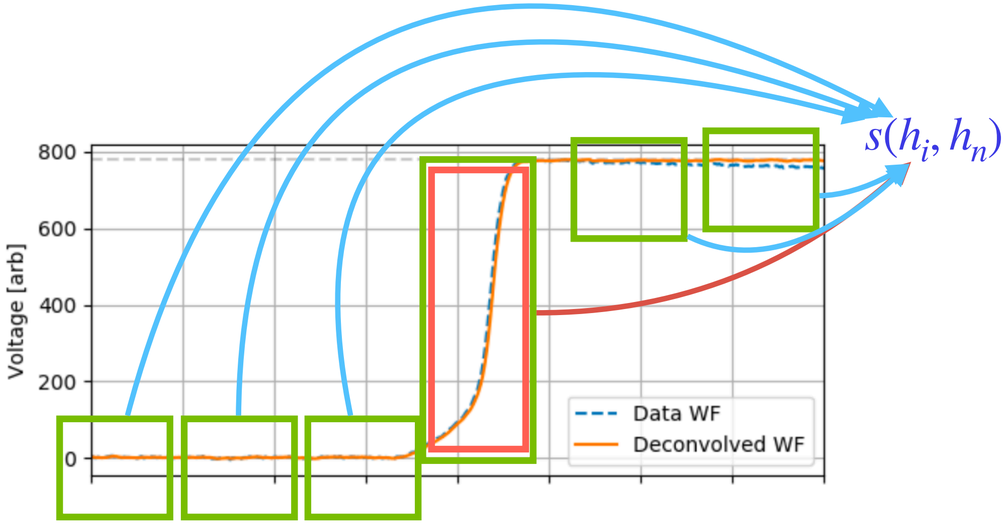
- Hjelm et al: https://arxiv.org/pdf/1808.06670.pdf
- Pass both high level feature vector and low level feature map to a discriminator to get a score:
  - “Real” if feature vector and feature map comes from the same image
  - “Fake” if feature vector and feature map comes from different image
- The discriminator is a way of quoting mutual information (MI):
  - MI measures the mutual dependence of two variables

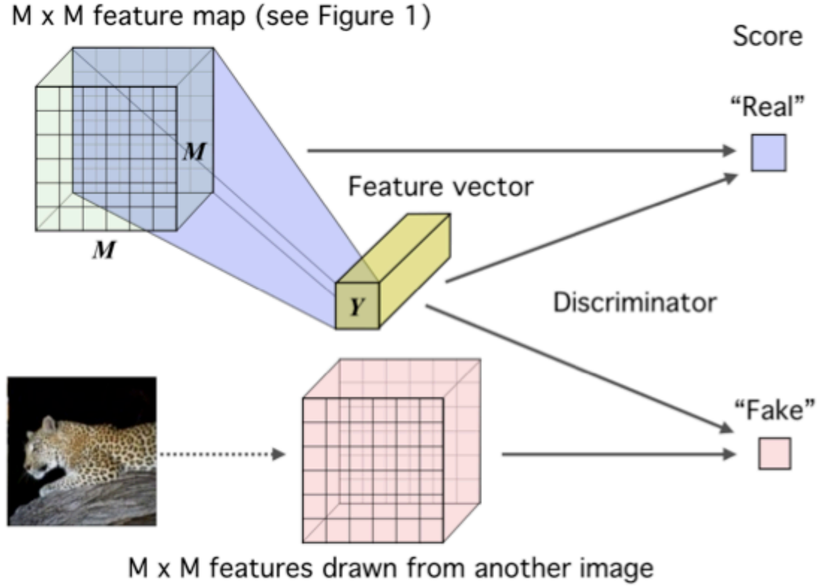


# Deep InfoMax

- Analogy with attention mechanism:
  - If a hidden state $h_3$ is important, then attention score $s(h_3, h_n)$ should be large, and $s(h_{i\neq 3}, h_n)$ should be low
  - If the feature vector accurately represents the image:
    - Its MI with real low level feature map should be high
    - Its MI with fake low level feature map should be low
- Deep InfoMax does not directly produce classification, but it gives us a representation (feature vector) to facilitate other tasks



# Likelihood fit vs. model tuning

## Likelihood Fit

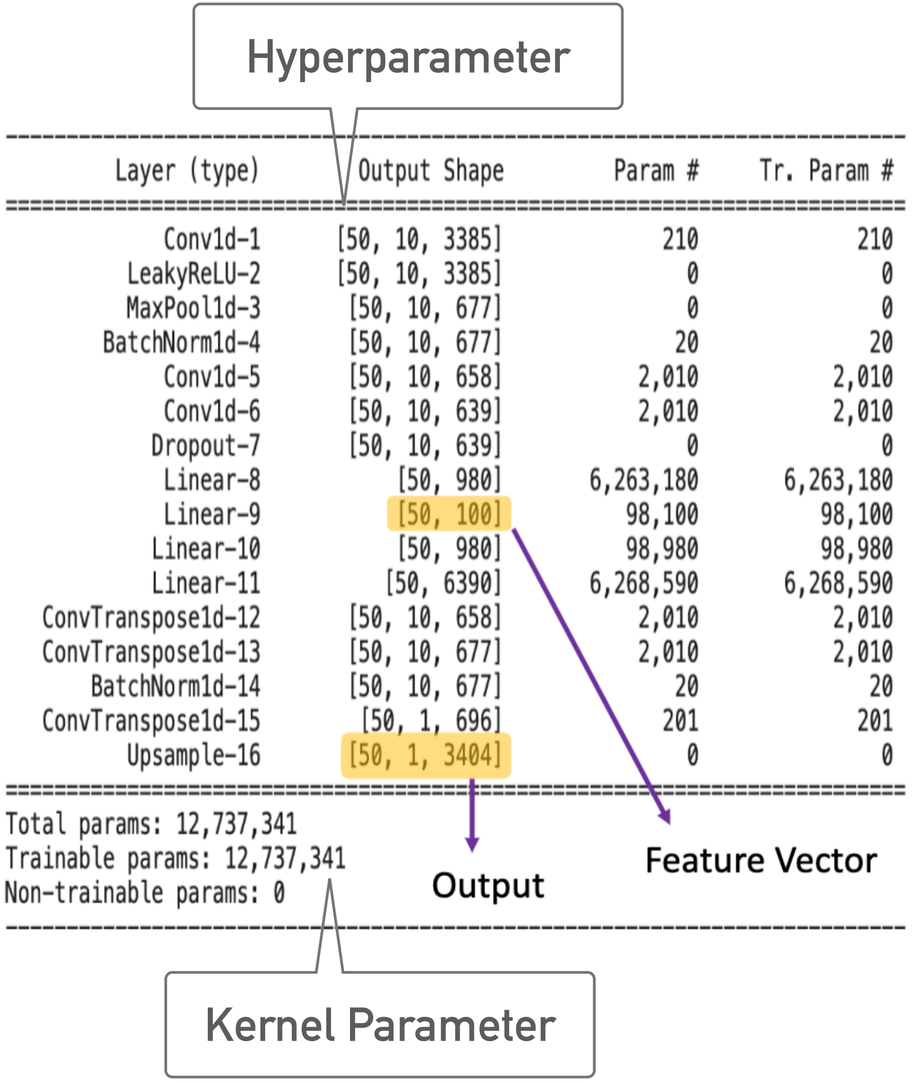
- Fitting MC energy spectrum to data
- Fit is guided by a well-defined likelihood function
- Evaluation of each step is cheap
- Require a lot of evaluations, but usually quite accurate:
  - Gradient Descent (Frequentist)
  - MCMC (Bayesian)

## Model Tuning

- Changing model hyper-parameter to achieve best performance
- No well-defined likelihood function
  - Black box optimization problem
- Evaluation of each step is usually very expensive
- Usually a “search” instead of “fit”:
  - In finite steps, we would like to explore more parameter space while exploit the best model hyper-parameters

# Re-define model tuning

- What are we searching:
  - Kernel Parameters: parameters that change during training
  - Hyperparameters: parameters that define the structure of network, thus remains unchanged during training
    - Batch size, # of layers, # of channels, # of neurons per channel, etc.
- Some hyper parameters are correlated or anti-correlated
  - Learning rate and regularization strength parameters usually have anti-correlation
- Each time we change hyper-parameter,  a new model is created.
- The new model is trained/evaluated on the same dataset.
- Define one metric to compare performance.
  - Final loss, accuracy, AUC ROC curve, etc

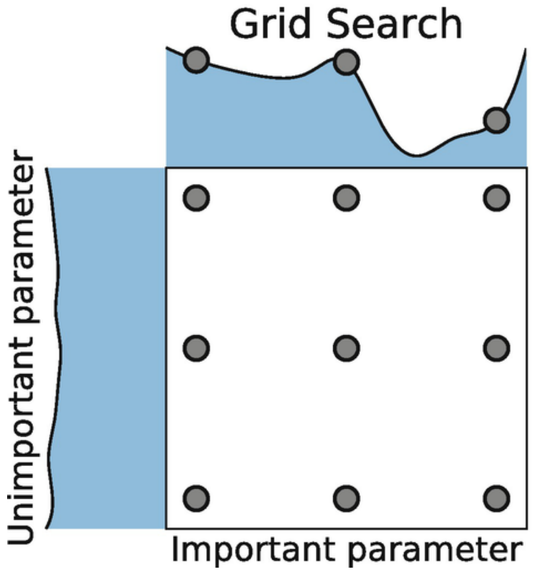


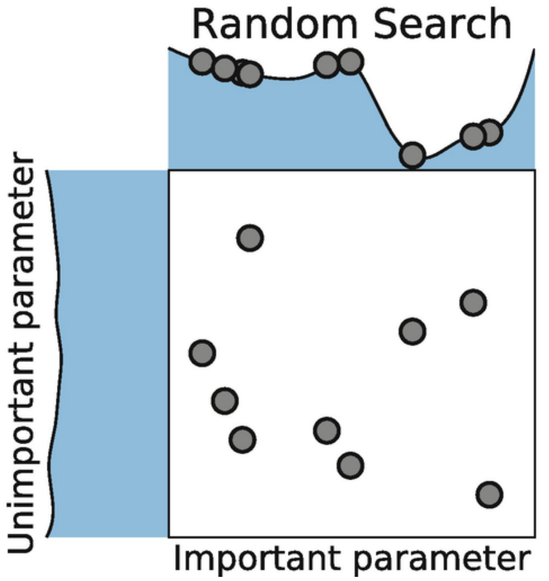
# How to tune our model?

- An ancient art — graduate student descent
  - Build model, then ask your graduate student to figure out the best parameters
  - Pros:
    - Human intelligence based
  - Cons:
    - In first place, you need a graduate student

# How to tune our model?

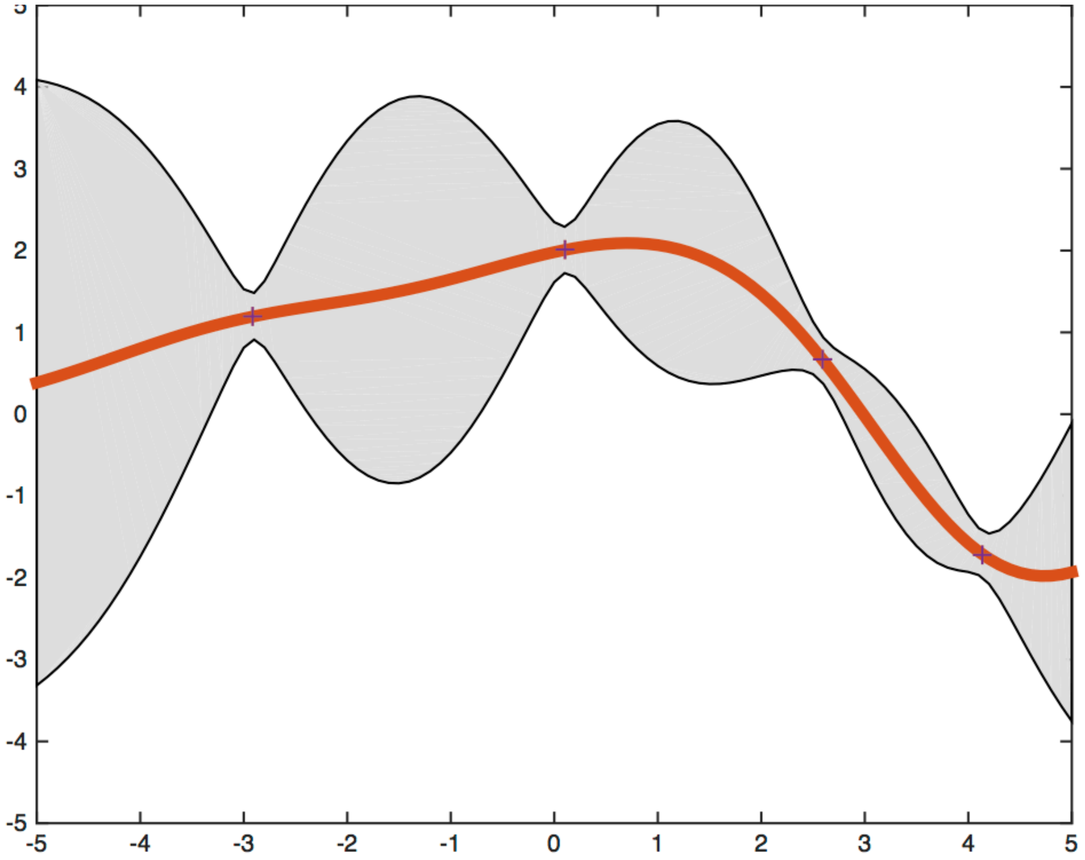
- Grid search:
  - Pros:
    - Always give the best performance, if your grid is fine enough
  - Cons:
    - Most expensive method
    - Suffer heavily from curse of dimensionality
- Random search:
  - Pros:
    - Less expensive than grid search
    - Suit better for machine learning model, since some parameters are unimportant (JMLR13, 281–305 (2012))
  - Cons:
    - Does not utilize the correlation between parameters
    - Suffer less from curse of dimensionality,  but for large parameter space, still need to sample a lot of points
- With the same number of evaluations, random search is better than grid search for ML model.



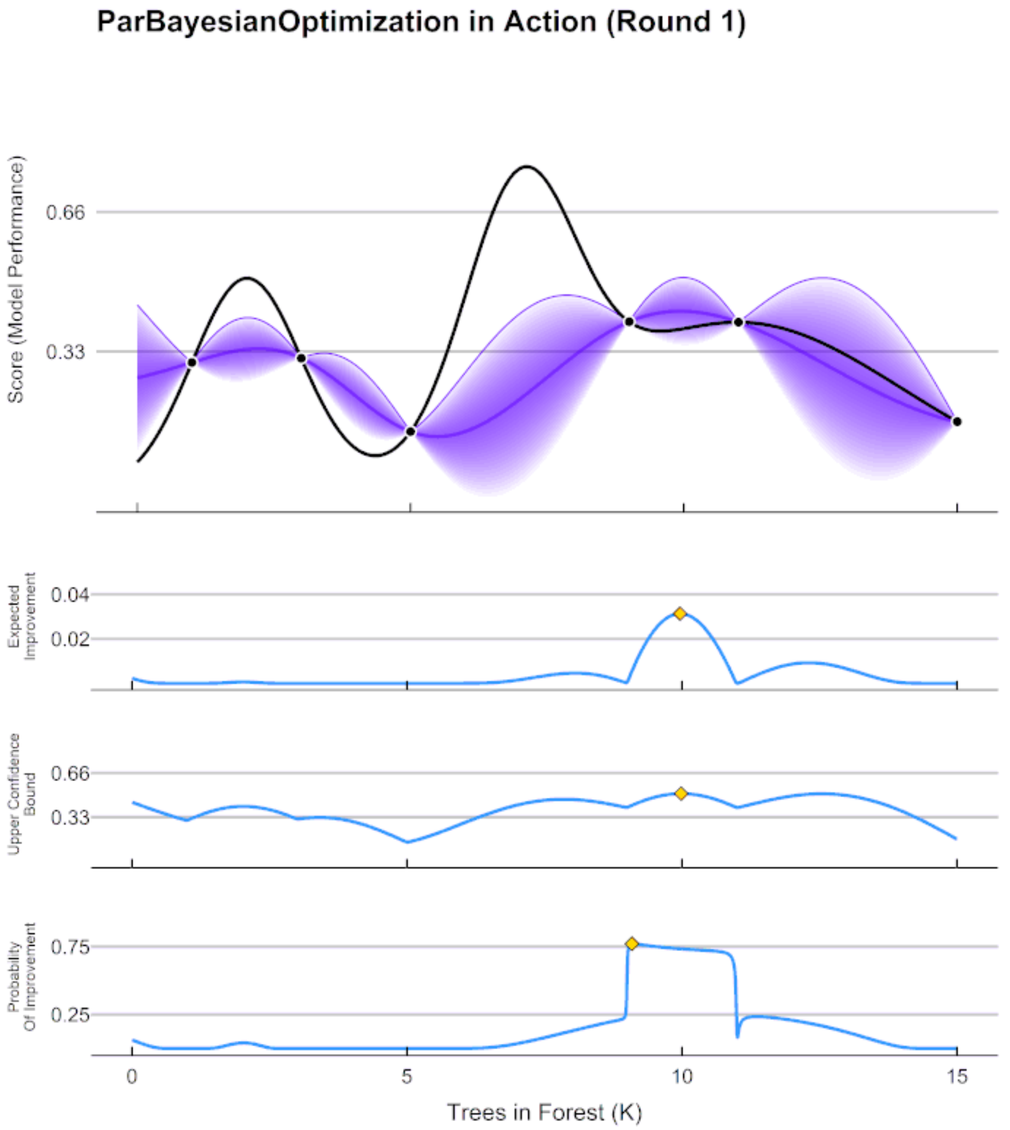
# Gaussian Process

- Bayesian Optimization performs better than simple random search by incorporating parameter correlation
  - It assumes the input is proper
- To introduce BO, we have to start from a statistical concept called Gaussian Process
  - A a Gaussian process is a Gaussian where the mean and variance are both functions
  - 1D Gaussian process is an infinite-dimensional multivariate gaussian



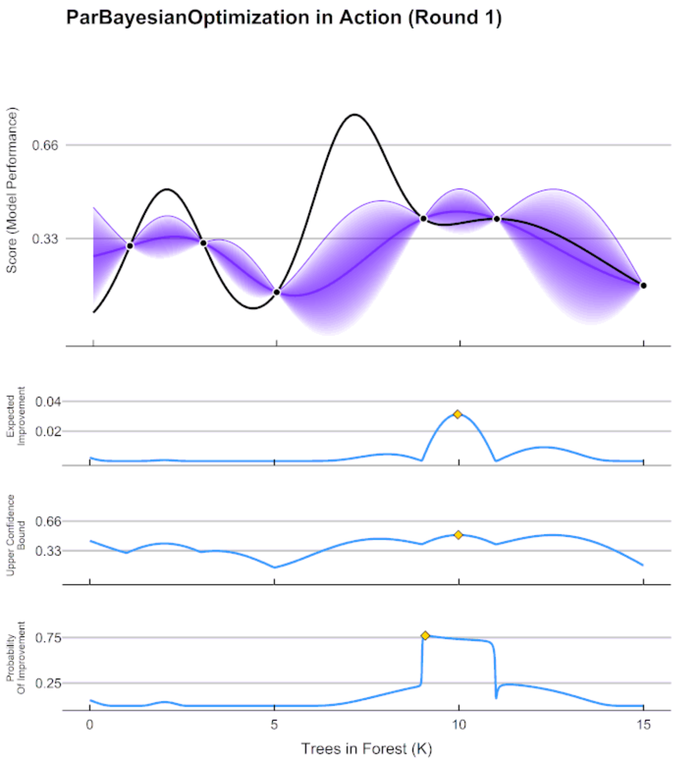
# Bayesian Optimization

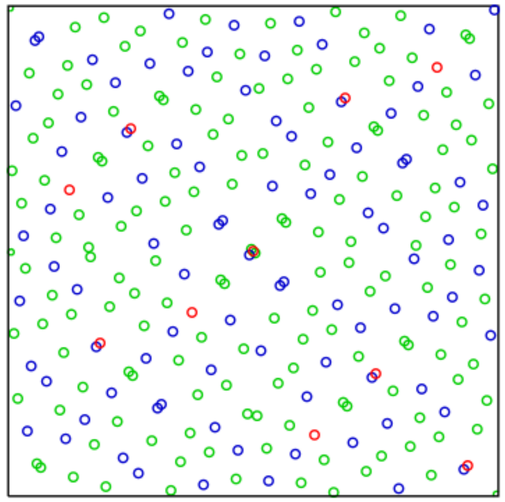
- Start with Gaussian Process prior:
  - The input to the Gaussian Process is the possible values of all hyper parameters
- Each time we make an evaluation, it reduces the uncertainty at this point to 0
- Construct an acquisition function after each step, then pick the highest value in acquisition function as our next step of improvement
  - Each step is based on all previous steps
- Explore vs. Exploit
  - Explore: evaluate at the region with maximum uncertainty
  - Exploit: evaluate at the region that has the highest probability to make improvement



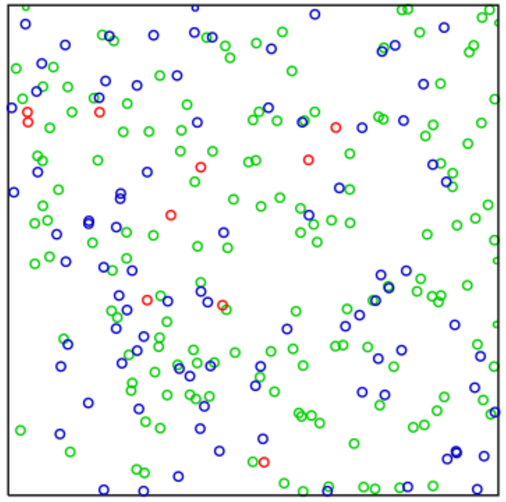
# Initializing Bayesian Optimization

- In BO, we usually perform a pre-run with a few randomly sampled trials, then run BO to explore-exploit
- Since we do not have any information about the model in the rerun, we'd like to sample as uniformly as possible
- Sobol sequence:
  - A sequential sampling technique to sample from high-dimensional space
  - The sampled points are more uniformly distributed compared to vanilla random sampling
  - Explore parameter space in Pseudo-Grid Search manner
- Then BO will be conducted according to the acquisition function



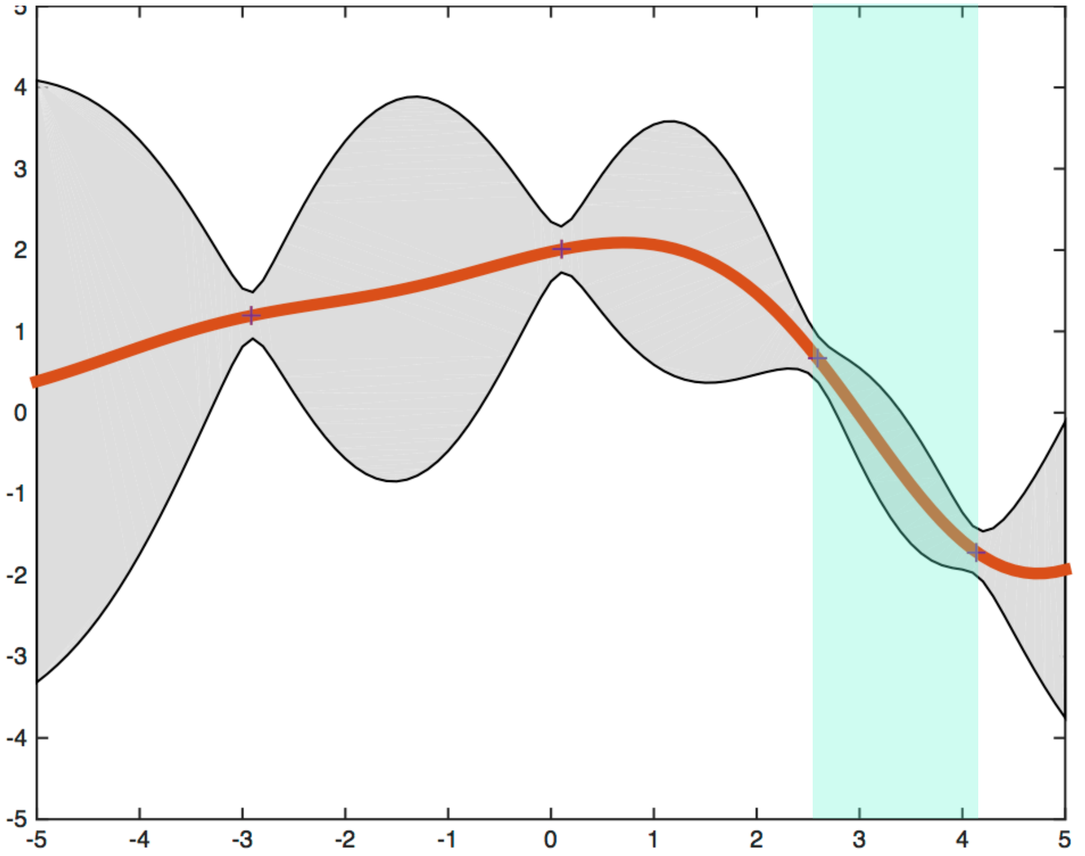
Sobol Sequence

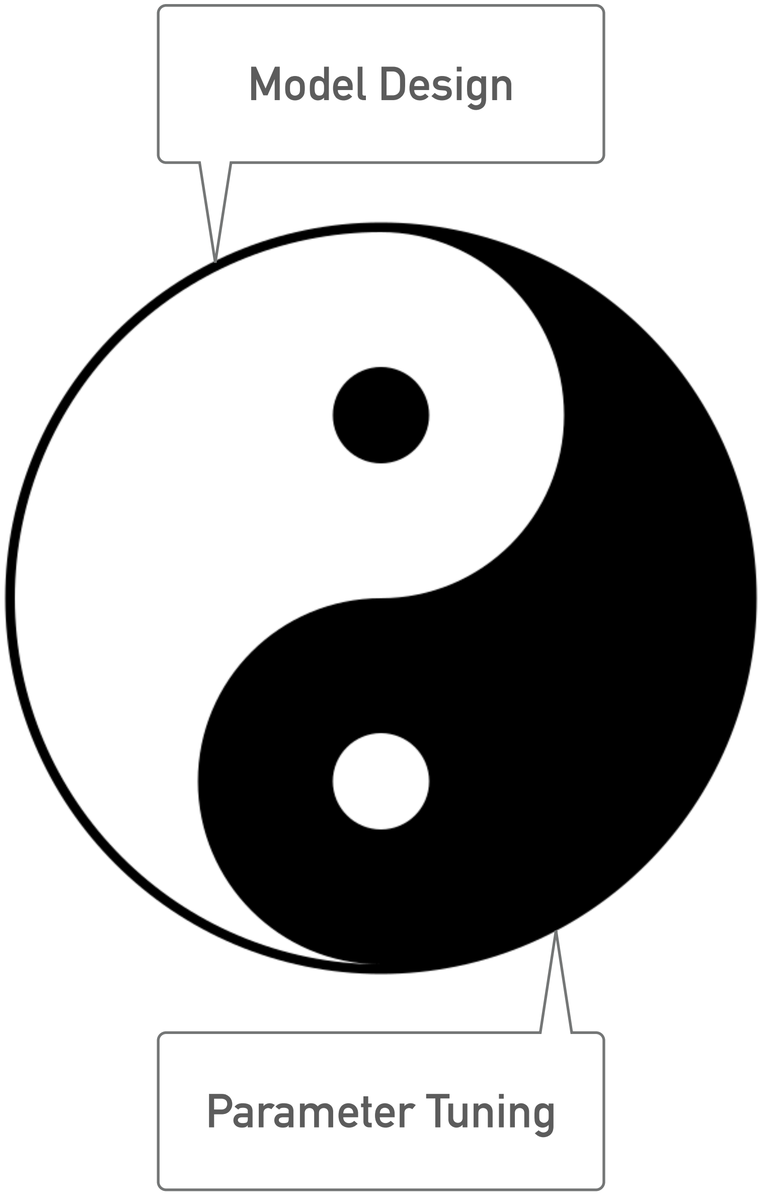


Random Sampling

# Why is BO better?

- BO uses the parameter correlation to avoid redundant evaluations
  - Only search in region that will most likely lead to a performance improvement
  - Much less expensive than grid search and random search
- BO performs better as you introduce more evaluation steps
  - Ideal for a model tuning problem
- BO assumes gaussian nature of input parameter, which is not too far away from reality






# A final word about model tuning

- Bayesian Optimization is also suitable for other expensive tuning tasks:
  - Tuning MC simulation parameters
- Tuning is a dynamic process:
  - BO helps to fine-tune parameters, but fine tuning doesn't solve all problems
  - Changing the network design is sometimes more efficient
  - Design & tuning is a reciprocal process
- After all, graduate student descent is needed

# Summary

- Generative Models:
  - GAN and VAE
- Semi-supervised learning with GAN
- Representation Learning:
  - Transfer Learning: Use the representation
  - Constrastive, Self-Supervised Learning: produce representation
- Tuning hyperparameter for a machine learning Model:
  - Random Search vs. Grid Search
  - Bayesian Optimization
  - Tuning and Design is a reciprocal process