Image from NASA: https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

# Geospatial Deep Learning

## GANs and Autoencoders

WV VIEW

AmericaView
Empowering Earth Observation Education
AmericaView.org - Est. 2003

This class primarily focuses on using deep learning for making spatial predictions. We have briefly discussed using deep learning regression to predict a continuous variable; however, our primary focus has been using deep learning for classification or to predict a categorical variable.

There are other uses of deep learning in the geospatial sciences. In this short module, we will introduce generative adversarial networks (GANs) as a means to generate data.

We will then discuss autoencoders as a means to compress data into a latent space, denoise data, and detect anomalies.

We will not discuss these topics in great detail. The goal here is to simply introduce them.

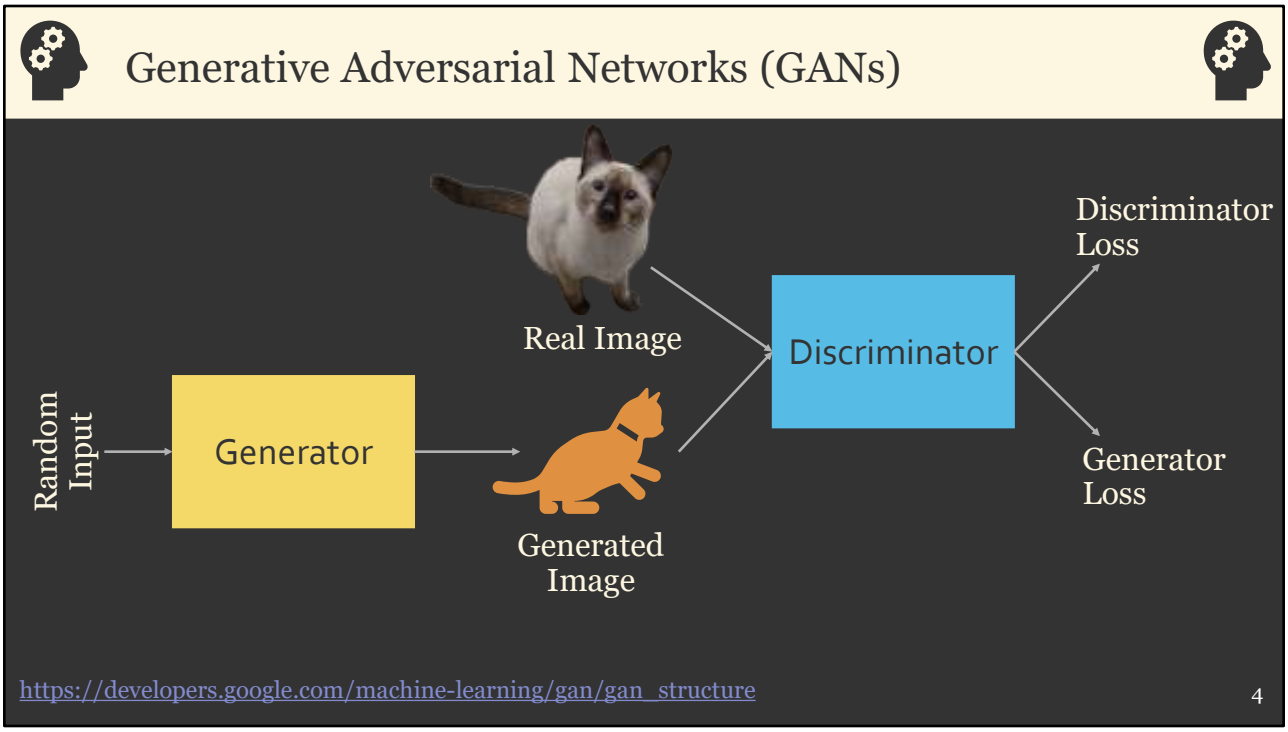
## Module Topics

1. Generative modeling using GANs
2. Autoencoders, denoising autoencoders, variational autoencoders



These are the topics covered in this module.

# Generative Models

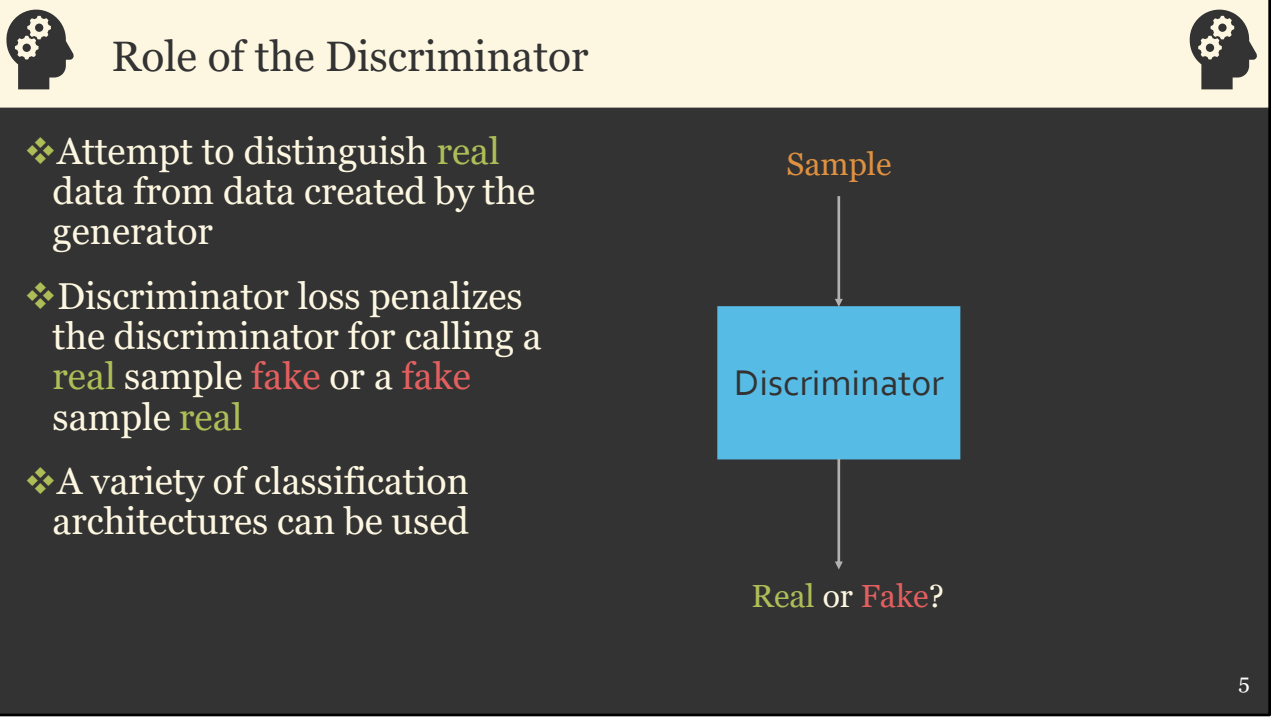



The goal of generative AI is to generate realistic or believable fake or synthetic data. This could include text, images, or videos. This technology has allowed for the development of "deep fakes", which raises serious ethical concerns. However, in the geospatial sciences synthetic data generation methods have several positive applications. One example is developing synthetic training data to train supervised deep learning algorithms, such as those designed for semantic or instance segmentation. Another use is filling data gaps, such as replacing clouds in imagery with predicted pixel values.

Generative adversarial networks (GANs) offer one means of performing generative AI. These architectures have two components: the generator and the discriminator. The goal of the generator is to convert a random input to a realistic or fake sample while the goal of the discriminator is to differentiate between real data and those generated by the discriminator. During the training process the parameters associated with the generator are updated in an attempt to trick the discriminator while the parameters associated with the discriminator are updated to try and detect fake data generated by the generator. This training process progresses in an attempt to update the trainable parameters associated with the generator such that it can successfully trick the discriminator.

The example associated with the link provided on this slide was created by

Google and does a good job of explaining GANs.




The goal of the discriminator is to learn to tell the difference between fake data created by the generator and real examples. During the training processes, it is necessary to train the discriminator since its predictions must improve and force the generator to continue to improve to "outsmart" it.

The loss function associated with training the discriminator penalizes it for incorrectly labeling a real sample as fake or a fake sample as real.

Since the goal of this component of the GAN is to return a binary prediction for each sample (i.e., label it as real or fake), then the associated architecture can be any scene labeling architecture, such as a convolutional neural network or a fully connected neural network. CNNs are commonly used.

## Role of the Generator

- Generate fake data
- Fake data generated from random noise
- Goal is to generate fake data to trick the discriminator

https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

6

The goal of the generator is to produce believable fake data that can trick the discriminator. The example images on this slide represent fake images generated by a generator that are supposed to represent celebrities. A GAN architecture was trained using real celebrity faces and those generated with a generator with the goal of tricking the discriminator. This was taken from the link provided on this page, which steps through a PyTorch GAN implementation. This example is worth exploring if you are interested in generative AI.

The generator is provided with random noise and attempts to modify this random input to represent a sample that can trick the discriminator. The goal is to generate data that match the distribution of the real images provided in the training set.

The generator component of the model is an architecture that can take input data and transform it to a new distribution. This could be accomplished using a modified semantic segmentation or decoder architecture that accepts random data and transform them to an output.

## Training a GAN

- Alternating process of:
  1. Training discriminator for one or multiple epochs
  2. Training the generator for one or multiple epochs

- Monitor two losses
  - min-max loss
    - Generator attempts to minimize
    - Discriminator attempts to maximize
  - Alternatives
    - non-saturating GAN loss, Wasserstein GAN loss
- Assessing or detecting convergence can be difficult/challenging

7

Training a GAN is a bit different from training a classification deep learning architecture. Generally, learning will progress by alternating between training the discriminator and then the generator. When training the discriminator, real samples will be fed in along with samples generated by the generator. However, only parameters associated with the discriminator will be updated. The goal is to update the weights to improve the correct prediction of real and fake samples.

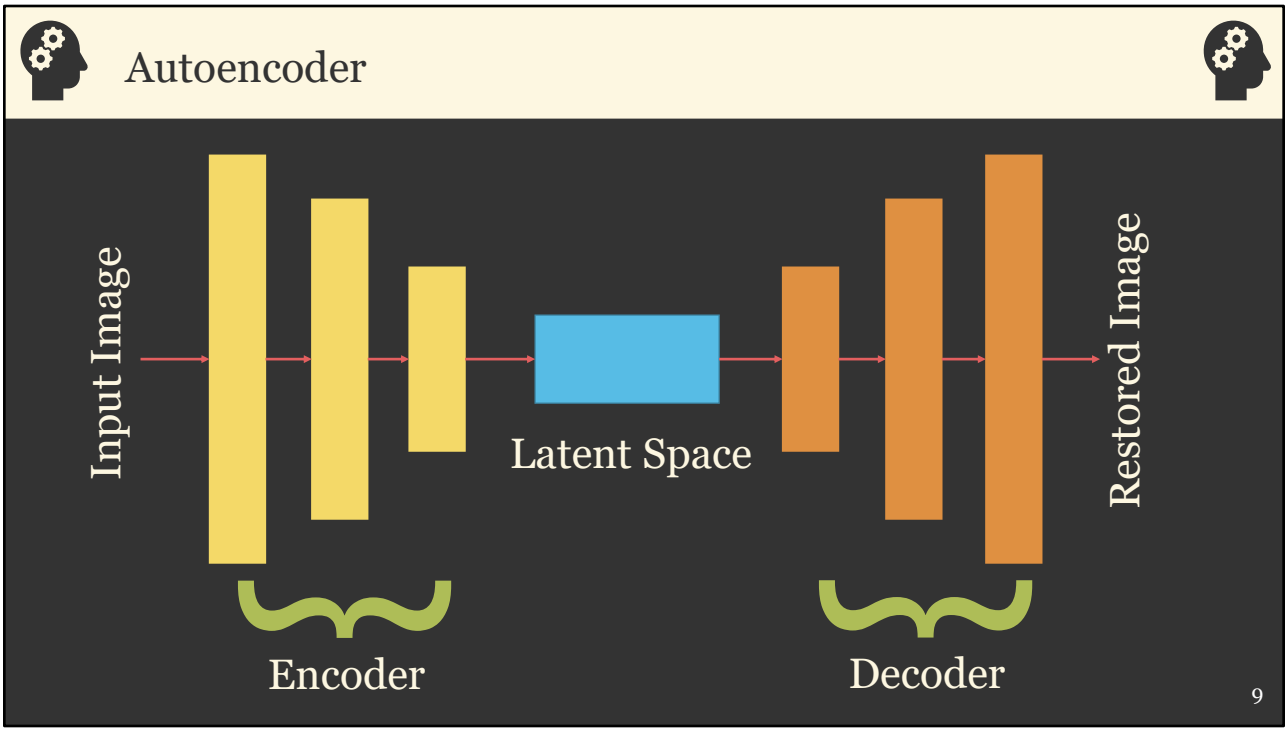
When updating the generator, only parameters associated with the generator will be updated; however, the predictions made by the discriminator will be used to guide these updates.

Different loss metrics are available for training GANs, and two losses will need to be monitored: a loss associated with the generator and a loss associated with the discriminator. One option is the min-max loss. The specifics of this loss metric are outside the scope of this class. When training the generator, the goal is to minimize this loss. When training the discriminator, the goal is to maximize this loss (i.e., minimize 1 – this loss). Optimal performance occurs when the generator is able to fool the discriminator. Note that there are alternative loss metrics for GANs, some of which are noted on this slide.

It can be challenging to train GANs or know when they are converging. This is

still an active area of research. Also, there are other uses of GANs other than just creating new samples that are similar to the provided training samples. One example is style GANs. The goal of these architectures is to apply styles learned from one dataset to samples from another dataset. For example, satellite image could be altered to take on the style of a van Gogh painting. We will not discuss other GAN applications in this course.

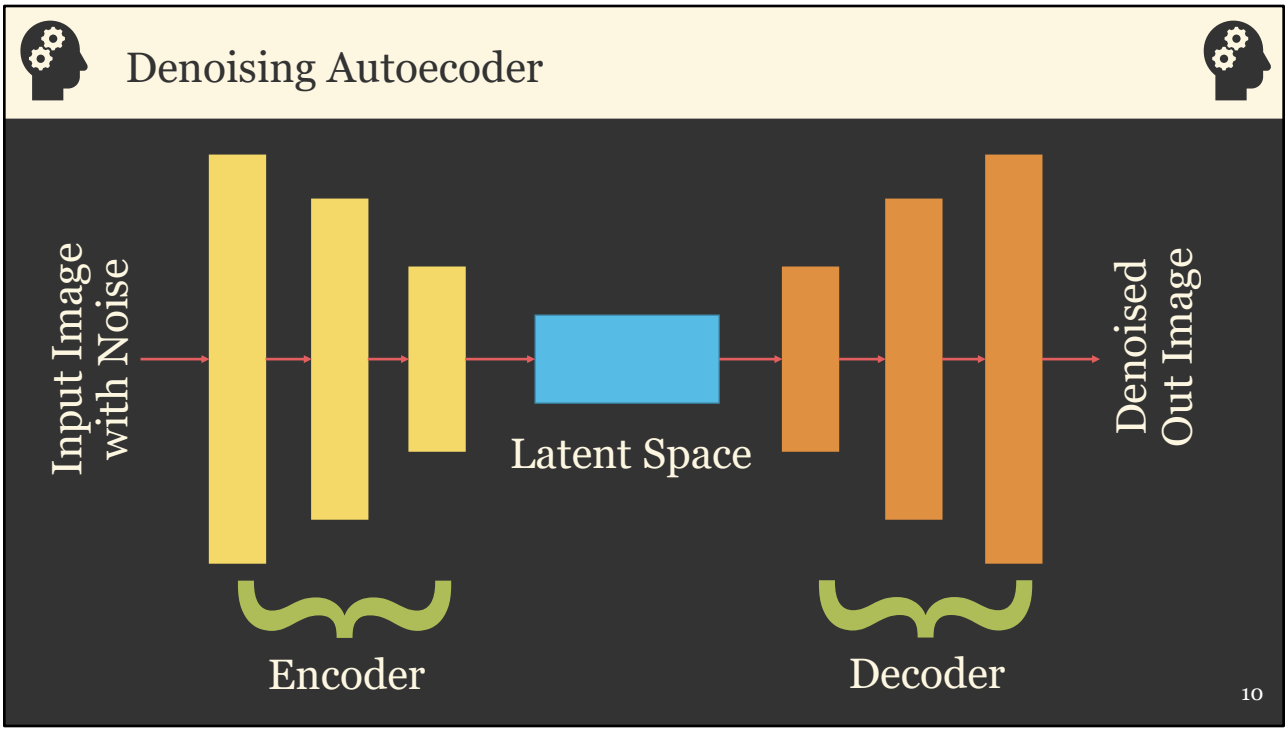
# Autoencoders




The goal of an autoencoder is to be able to accept input data, transform it to a lower dimensional space, then rebuild or restore the data with limited errors. The lower dimensional space is generally termed the latent space. The encoder component converts the input data to the lower dimensional space while the decoder attempts to restore the data. The encoder and decoder components can be created using CNN or fully connected architectures. For example, A UNet-like architecture could be used. However, instead of attempting to predict pixel-level labels, the goal would be to reconstruct the original image. Also, skip connections would not be included since the goal of the decoder is to be able to reconstruct the input using only the latent space (i.e., the output from the bottleneck layer).

One common and very practical use of autoencoders is to compress files. The latent space would take up less space or memory on a system. It would also be easier to transfer these data over a network.

An autoencoder could be used to convert an image or audio file to a latent space for storage and/or transfer then subsequently restore the data with limited loss of data.




A denoising autoencoder is similar to a traditional autoencoder. However, the goal is to learn to denoise data. Random noise is added to the input data, and the goal is to predict the original image.

By using a loss function to compare the denoised output to the original image without any added noise, the architecture is tasked with learning to remove noise.

## Reconstruction Loss

$$= \frac{1}{N}\sum_{i=1}^{N} |x_i - \hat{x}_i|^2$$

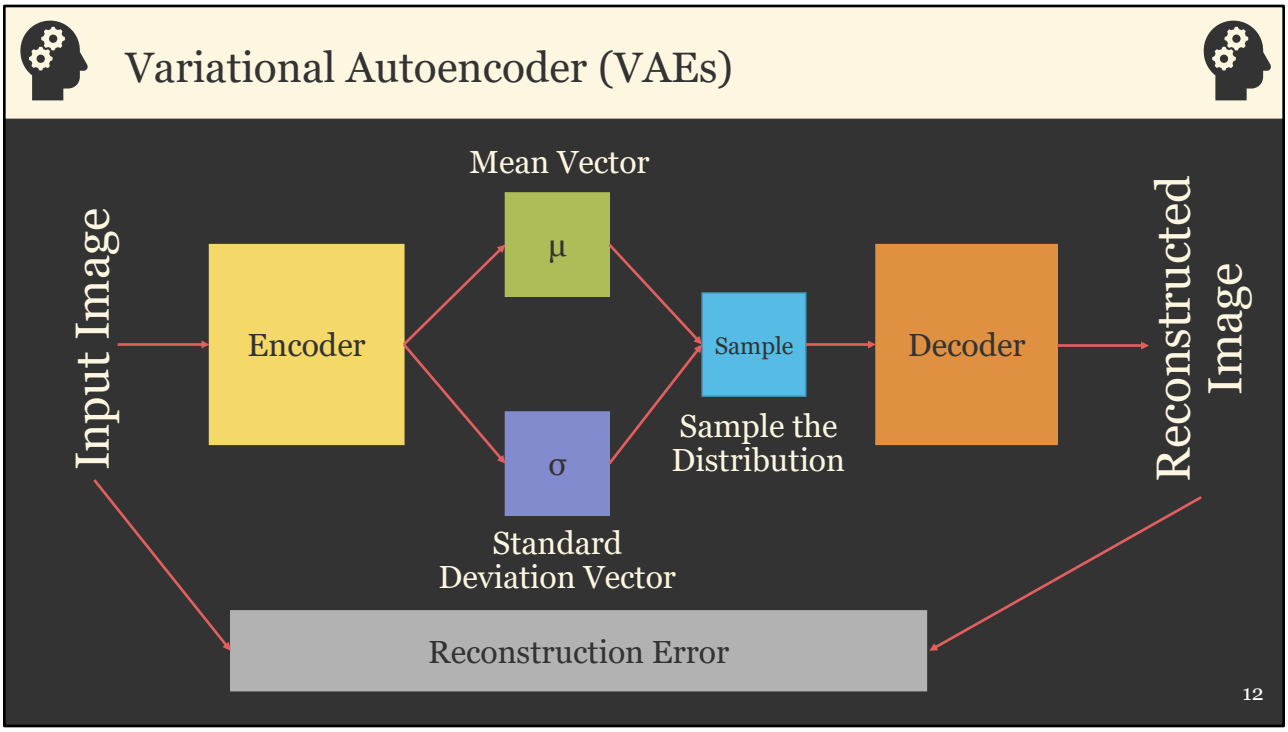
$$= -\frac{1}{N}\sum_{i=1}^{N} (x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i))$$

11

There are different options for the loss function for autoencoders.

Generally, the goal is to compare the reference data to the output of the decoder. For example, if the goal is to reconstruct an image, the actual pixel values will be compared to the predicted pixel values. A lower average or total difference in pixel values would indicate that the reconstructed image is similar to the actual image.

Reconstruction loss can be configured using the average of the absolute value of the squared residuals (the top equation) or using binary cross entropy (the bottom equation).




One augmentation of autoencoders that have been shown to be especially useful are variational autoencoders, or VAEs. VAEs are different from autoencoders because they do not generate a single latent space. Instead, the goal is to predict mean and standard deviation vectors that represent the latent space as a distribution.

A sample can be drawn from the learned distribution and be fed through the decoder to generate a reconstruction.

## Autoencoder Loss

- Reconstruction Loss = measure of quality of reconstruction

$$= -\frac{1}{N}\sum_{i=1}^{N}(x_i \log \hat{x}_i + (1 - x_i)\log(1 - \hat{x}_i))$$

- KL Divergence = how normal is learned distribution

$$= \mathrm{KL}\big(q(z|x)|p(z)\big) = \frac{1}{2}\sum_{j=1}^{d}\big(1 + \log\big(\sigma_j^2\big) - \mu_j^2 - \sigma_j^2\big)$$

13

The loss for VAEs has two components.

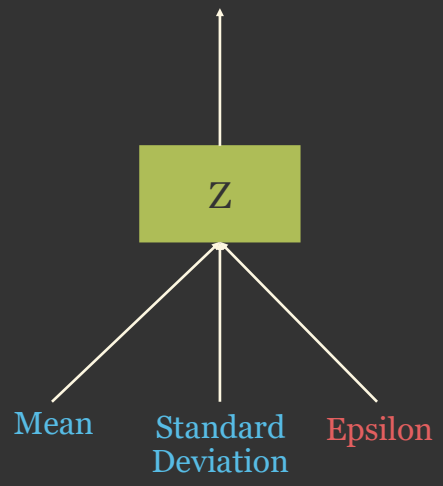
The first component is the reconstruction loss, similar to traditional autoencoders. This loss is minimized as the difference between the actual data and the reconstructed data is minimized. For example, this loss would be reduced if the predicted pixel values become more similar to the actual pixel values if the goal is to reconstruct an image.

The KL divergence loss is used to measure how well the learned distribution fits a normal distribution. The goal is for the distribution to be close to normal; as a result, this loss is minimized as the learned distribution, which represent the latent space, approaches a normal distribution. Practically, this loss is included to force the learning process to work towards a latent space representation that is close to a normal distribution.

In short, the goal is to generate a model that can reconstruct data using a latent space distribution that is close to normal.

## Reparameterization Trick

- Can track gradients through a sampling process
- Breaks backpropagation
- Epsilon serves a fixed stochastic node
- Split to deterministic and stochastic components
- Deterministic = can use backpropagation to update




14

There is one additional issue that needs to be overcome. The process of randomly sampling from the distribution is stochastic. As a result, backpropagation cannot be performed, and the model cannot be trained end-to-end.

This issue is solved using the reparameterization tick that allows for the mean and standard deviation vectors to be treated as deterministic while an added variable, epsilon, is treated as stochastic. Since epsilon does not need to be updated, this allows for backpropagation to be performed and the model to be trained.

## Is a sample an anomaly?

- Population distribution learned from training samples
- Does new sample fit learned distribution?
- High reconstruction error = more likely to be an anomaly

15

One use of a VAE is to determine whether a new sample is an anomaly, or different from the samples that were used to train it. If the model is not able to reconstruct an input accurately, this could indicate that it is outside of the learned distribution.

For example, if an architecture is trained using only images of forests without any animals in the scene, it would be expected that reconstructing an image that does have an animal in a scene could not be done accurately. As a result, this model could be used to find images that contain animals.

One complexity is that the user must determine what level of reconstruction error should be flagged as indicating an anomaly. This generally is done by looking at the range of reconstruction errors for anomalous and non-anomalous samples.

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

Thanks! Hope you found this useful.