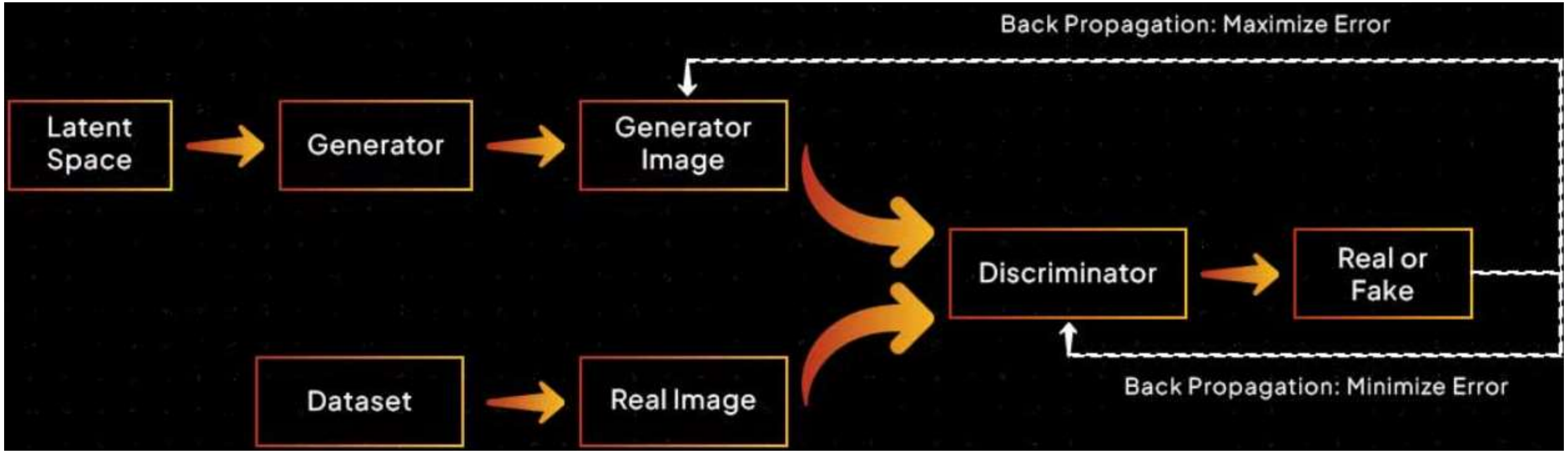
CIFAR10 USING GAN

Presented By Akshit Jain

Implement generator and the discriminator architectures by introducing convolution and up/transposed convolution layers. Use one of the classes from the CIFAR-10 dataset to generate images. Show the effect of controlling the noise vector.

GAN Architecture



CIFAR-10 Classification

- **Dataset:** CIFAR-10 has 60,000 32x32 RGB images
- **Output Class:** 10 Classes
- **Frequency**: Each class has exactly 5,000 rows.

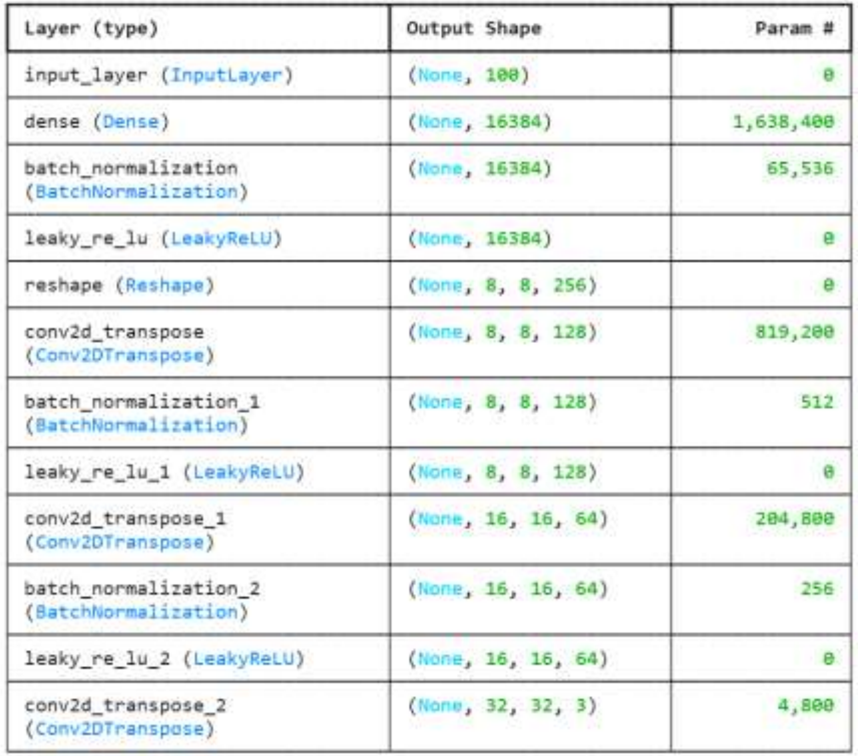
Training Setup

- **Optimizer:** Adam
- **Epochs:** 5000
- **Batch Size**: 64
- **Loss**: BinaryCrossentropy
- **Target Class**: 7 (Horse) – Only used 1 class

Generator

- **Input :** Random noise vector (latent_dim = 100)
- **Dense Expansion :** Transforms noise into 8 x 8 x 256 feature.
- **Reshape & Activation:** Reshapes to 8 x 8 x 256. Used LeakyReLU
- **BatchNormalization**: Yes
- **UpSampling (8 x 8 → 8 x 8) :**Uses Conv2DTranspose to refine features
- **Resolution Increase (8 x 8 → 16 x 16)**: Another Conv2DTranspose doubles the size.
- **Final Output (16 x 16 → 32 x 32 x 3):** Last Conv2DTranspose creates 32 x 32 RGB image with Tanh activation

Generator

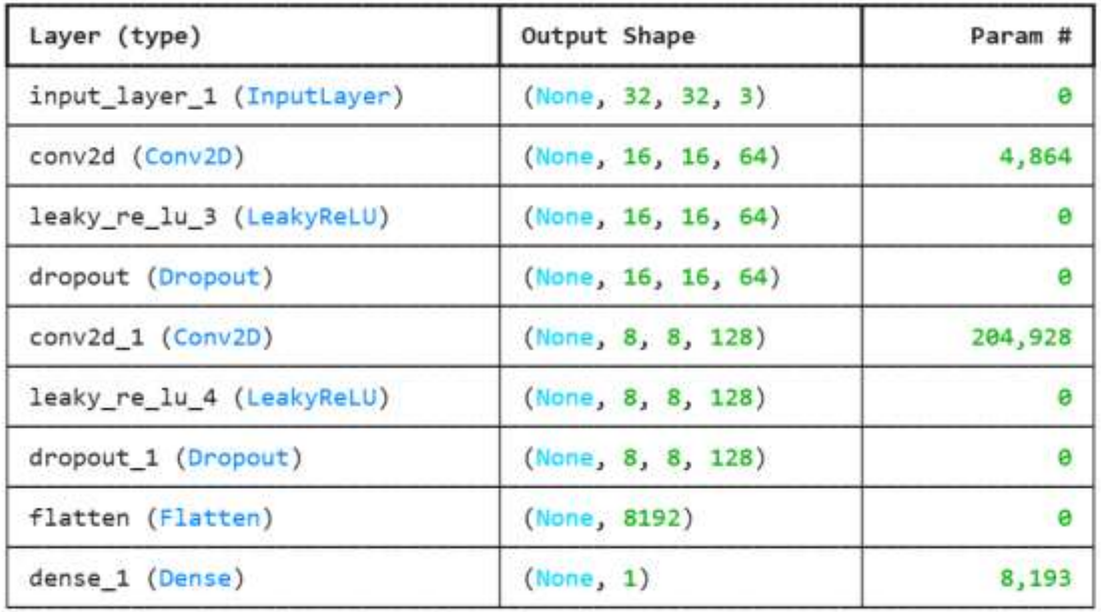


Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 100)	0
dense (Dense)	(None, 16384)	1,638,400
batch_normalization (BatchNormalization)	(None, 16384)	65,536
leaky_re_lu (LeakyReLU)	(None, 16384)	0
reshape (Reshape)	(None, 8, 8, 256)	0
conv2d_transpose (Conv2DTranspose)	(None, 8, 8, 128)	819,200
batch_normalization_1 (BatchNormalization)	(None, 8, 8, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 8, 8, 128)	0
conv2d_transpose_1 (Conv2DTranspose)	(None, 16, 16, 64)	204,800
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 16, 16, 64)	0
conv2d_transpose_2 (Conv2DTranspose)	(None, 32, 32, 3)	4,800

Discriminator

- **Input:** Takes a 32×32×3 image as input.
- **First Convolution (32 x 32 → 16 x 16)**: Extracts features with a 5 x 5 Conv2D layer, using stride 2 for downsampling.
- **Activation & Regularization:** LeakyReLU introduces non-linearity, followed by Dropout (0.3) to prevent overfitting.
- **Second Convolution (16 x 16 → 8 x 8):** Another 5×5 Conv2D further downsamples and extracts deeper features.
- **Flatten & Dense Layer:** Flattens the feature maps into a 1D vector and passes it to a single Dense layer.
- **Output:** Outputs a single value (logit) indicating whether the image is real or fake.

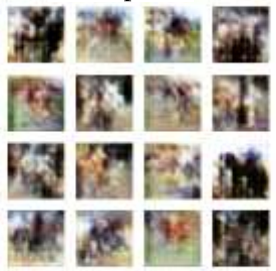
Discriminator



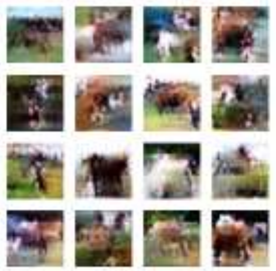
Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d (Conv2D)	(None, 16, 16, 64)	4,864
leaky_re_lu_3 (LeakyReLU)	(None, 16, 16, 64)	0
dropout (Dropout)	(None, 16, 16, 64)	0
conv2d_1 (Conv2D)	(None, 8, 8, 128)	204,928
leaky_re_lu_4 (LeakyReLU)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
flatten (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 1)	8,193

Generated images from class 7 (Horse)

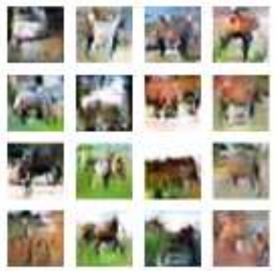
200 Epoch



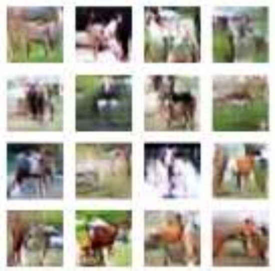
400 Epoch



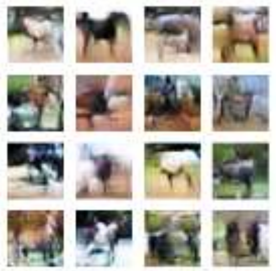
600 Epoch



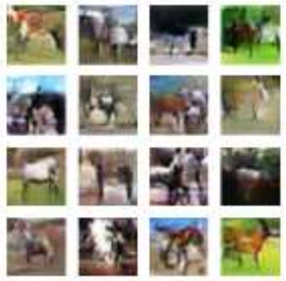
800 Epoch



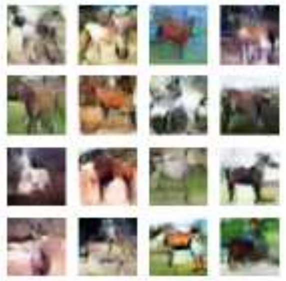
1000 Epoch



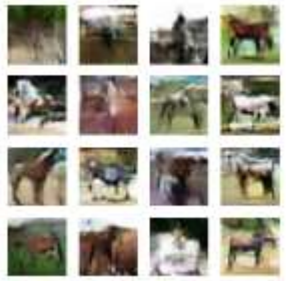
1200 Epoch



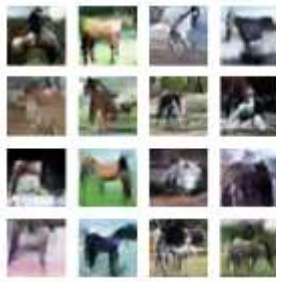
1400 Epoch



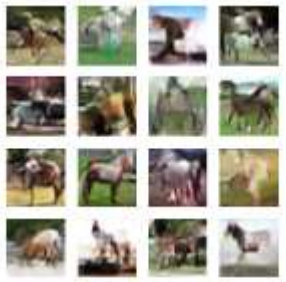
1600 Epoch



1800 Epoch

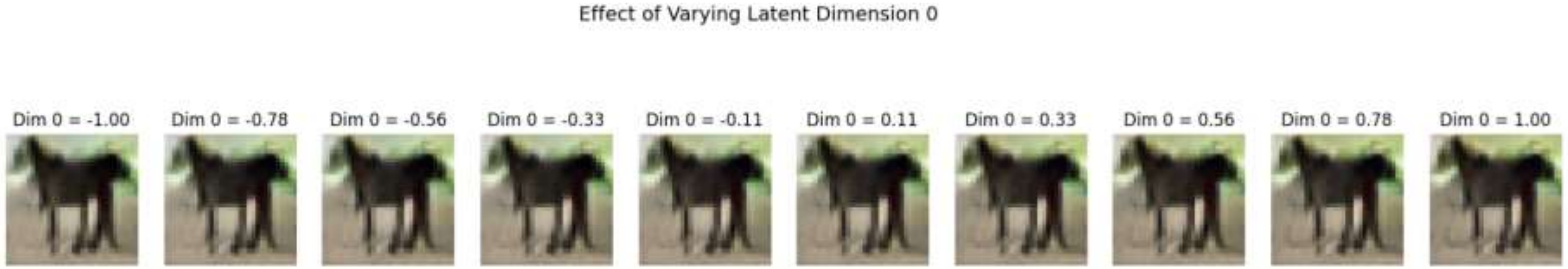


2000 Epoch



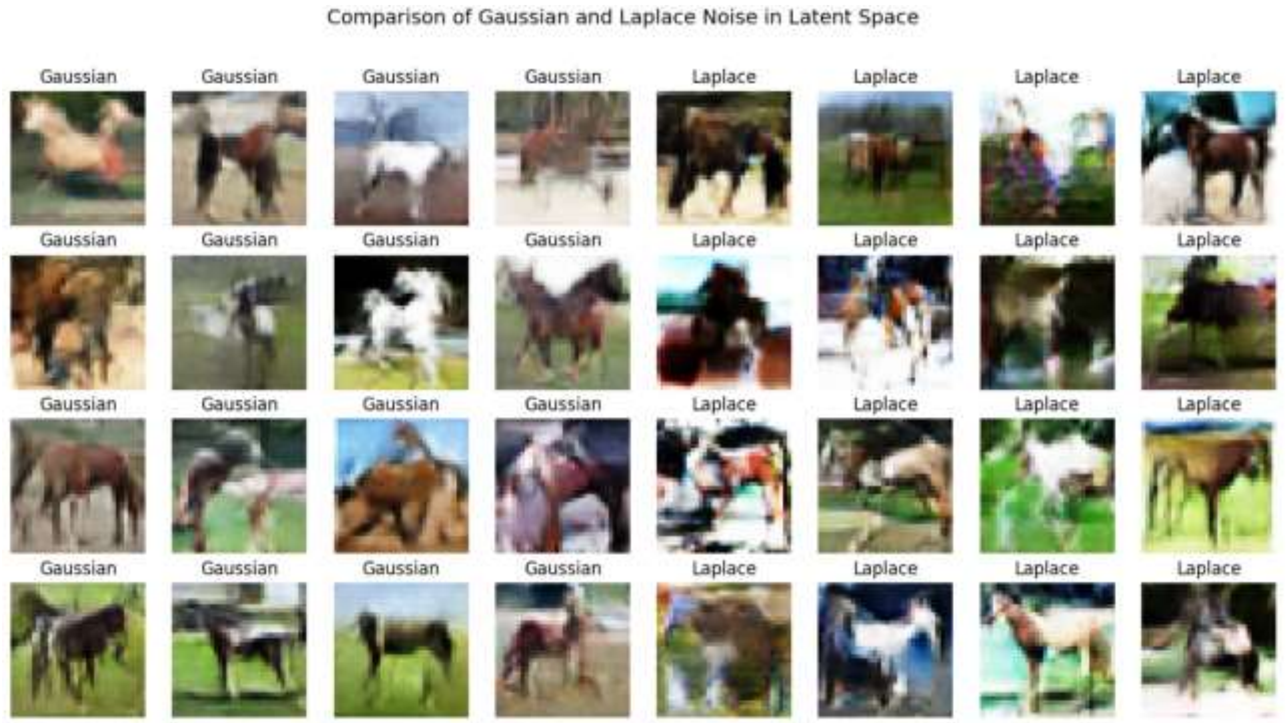
Results

Effect of Noise vector



Results

Comparison of Gaussian and Laplace Noise in Latent Space



Thank You