Introduction
In this assignment, we modify the GAN(Generative Adversarial Network), VAE(Variational AutoEncoder), and Conditional GAN models to generate and display 25 different digit images in a 5x5 grid. The dataset used is the MNIST dataset which consists of 28x28 pixel images of handwritten digits from 0 to 9. Generative models are used to create new data that resembles the training data. GANS create realistic data by putting two models against each other. VAEs encode and decode data in a probabilistic way. Conditional GANs generate data based on specific input conditions like the label of the digit in this case. These models are used to generate new, realistic images of handwritten digits.

Differences
Between Model Architectures, Inputs/Outputs, Loss(es) and Training
Explain the difference between how we generate samples in the 3 models (e.g. what are the differences in the inputs of each model, how they're trained, architectures, the types of layers, activations, sizes, the way they generate samples...etc).

VAE
The Variational Autoencoder model consists of 2 components: the encoder and decoder. The encoder takes inputs images of shape (28,28,1) and maps them to a latent space defined by the parameters mean and log variance of the latent distribution. These are computed through a series of convolutional and dense layers. The sampling layer uses these parameters to reparameterize the latent space with a random noise vector, generating the latent space.

The decoder takes the latent vector as input and reconstructs the original image. This is achieved by reshaping the latent vector into a tensor and using a series of convolutional layers to umsample to the original dimensions. The output of the decoder is a reconstructed image with pixel values ranging from 0 to 1 (sigmoid activation).

The training process optimizes two losses: the reconstruction loss and the KL divergence loss. The reconstruction loss measures how well the reconstructed image matches the original image using binary cross entropy. The KL divergence loss regularizes the latent space by encouraging the learned latent distribution to match a standard normal distribution. They’re combined for the total loss which is minimized during training.

The training uses the train_step method to calculate gradients of the total loss with respect to the model weights. These gradients are applied using the Adam Optimizer to update model parameters. The training data is MNIST images normalized form 0 to 1. The model is trained for 30 epochs with a batch size of 128.

GAN
The Generative Adversarial Network uses a generator and discriminator. The generator learns to create realistic images, while the discriminator distinguishes between real and fake images.

These networks are trained together in a zero-sum game, where the generator aims to fool the discriminator, and the discriminator strives to correctly classify the images as real or fake.

The generator is a neural network that takes a random noise vector of size 100 as input and transforms it into an image of shape (28,28,1) through dense and transposed convolutional layers. The generator uses batch normalization and Leaky ReLU activations to improve stability when training. A final tanh activation function makes the final pixel values in range of -1 to 1.

The discriminator is a convolutional neural network that processes input images and predicts a single scalar value, representing the likelihood of that image being real. It uses convolutional layers with strides, leaky ReLU activation, and dropout layers for the model architecture. It's trained using binary cross-entropy loss, where real images are labeled 1 and fake are 0.

The training alternates between the generator and discriminator. During each step, the generator creates a batch of fakes images from random noise, which are then passes to the discriminator along with a batch of real images from the dataset. The discriminator's loss is calculated based on how well it can detect if an image is real or fake. The generator's loss is how well it fools the discriminator into classifying as real.

The optimizer used for both networks is Adam with a learning rate of 0.0001. The model is trained for 50 epochs with a batch size of 256. Periodically, the generator's outputs are visualized using a fixed seed to measure the progress over time. Checkpoint saves model every 15 epochs.

CGAN

The conditional GAN also uses the generator and discriminators. It starts by combining the train and test set into a single dataset. The dataset is normalized from 0 to 1 and reshapes to include a channel dimension. The labels are one-hot encoded, and the data is put into a tensorflow dataset. It is shuffled and batches for training. The dimensions of the generator and discriminator are adjusted for label embeddings.

The discriminator is a CNN that takes an input pair of an image and its label. The first layer of the discriminator applies a convolutional operation with 64 filters, a kernel size of (3,3) and strides of (2,2) to extract features and downsample the pair. The output is passed through a leaky ReLU activation function with a small negative slope. A 2nd layer with 182 filters and the same activation is added. A global max pooling layer is used to create a fixed length feature vector. The final output is produced by a dense layer with a single number representing the probability that the input pair is real or fake.

The generator is a transposed convolutional network designed to creat realistic images conditioned on a specific label. The input of the generator is a vector of random noise (latent space) with a one-hot encoded label. The vector is passed through a fully connected layer that outputs a flattened tensor reshape into a 3D array of size (7,7, 129). To upsample the feature map, the generator uses a transposed convolutional layer with 128 filters which increases the

spatial dimensions to (14,14) with a leaky ReLU activation function. Another transposed convolutional layer with sigmoid activation outputs the grayscale image with pixel values normalized between 0 and 1.

The model is trained for 20 epochs with a batch size of 64. During each epoch, the generator and discriminators losses are tracked through 2 custom metrics. The generator learns to create realistic images for each digit, while the discriminator tries to detect real or fake. Same as above.

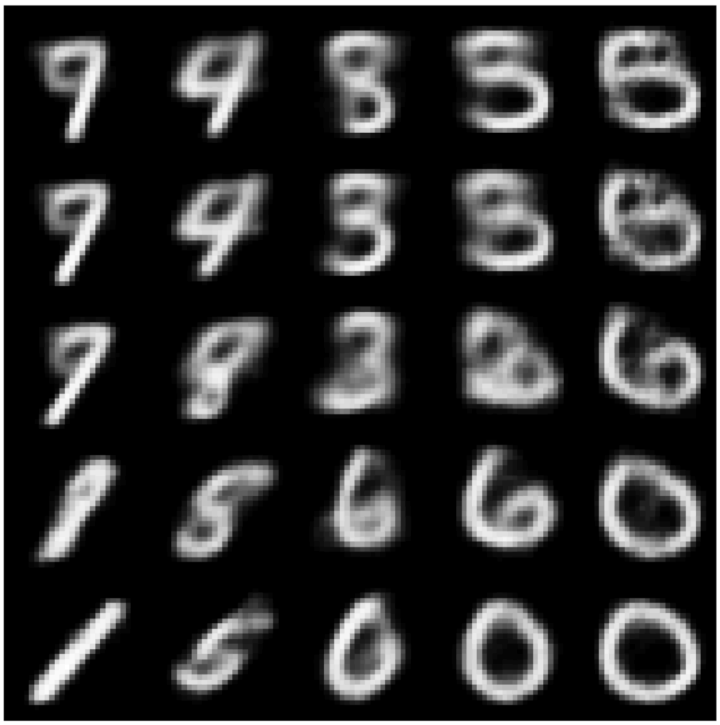
Differences

The main differences between the three models are in their approach to data generation and training. VAEs generate data by encoding into a probabilistic latent space then decode it. This results in diverse outputs but often lacks sharpness, making the images blurrier which can be seen in the 25 image output below. In contrast, GANs use a generator and discriminator in an adversarial setup where the generator tries to produce data the can fool the discriminator. This process usually results in sharper, more realistic images but can suffer from issues like mode collapse or instability. CGANs extend GANs by conditioning the generation process on additional information, or in this case, labels. The allows more control over generated data, enabling the model to produce specific types of images based on those class labels. This makes it more useful for tasks where you want to generate data from specific categories, unlike standard GANs, which generate more general outputs.

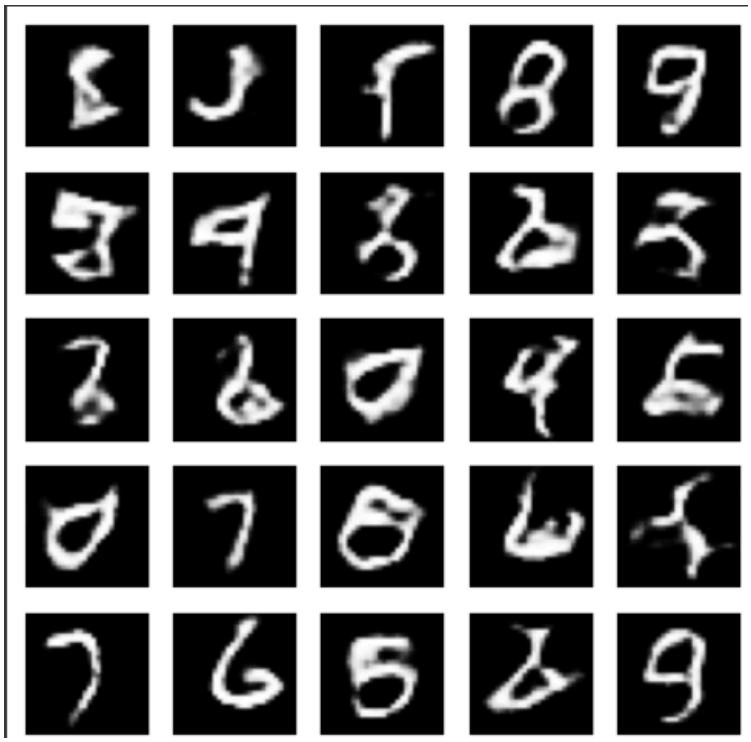
Images

The VAE images are pretty blurry throughout, but have some numbers that are represented decently well. All the images represent some type of digit 1 through 9 aside from the one on the bottom left, which does represent a 1 but is slanted.. However a good portion of the images are mixes of 2 or more digits. This makes it hard to tell what the actual digit is. On the other hand, the GAN images are pretty sharp throughout all 25 images. It makes it a lot clearer which images are what digits for the ones that are readable. The main problem is that it generated images that weren't really any of the digits 1-9. It generated random symbols in some of them. CGAN images fixes this problem as it was able to generate all 25 images according to the wanted output. Aside from a little missing segment in 2 of the images, it did really well in generating the images. They're all very sharp, readable, and accurate. Overall the CGANs model was the best and generated the best images. Reasons for these differences of outputs were given in the previous section.

VAE Digits Image Output



GAN Digits Image Output



CGAN Digits Image Output