NM PROJECT

IMAGE-TO-IMAGE TRANSLATION WITH GAN

KRUPA JANANI G

Table of Contents :

Problem Statement

The project aims to build and train a conditional generative adversarial network (cGAN) called pix2pix that learns a mapping from input images to output images. The cGAN network will generate synthetic images resembling BUILDING FACADES(***exterior faces or fronts of buildings***) based on the ***CMP Facade Database***, provided by the Center for Machine Perception at the Czech Technical University in Prague.

PROPOSED SOLUTION

- In the pix2pix cGAN, you condition on input images and generate corresponding output images. cGANs were first proposed in Conditional Generative Adversarial Nets.

- The architecture of your network will contain:

    1. A generator with a U-Net-based architecture.
    2. A discriminator represented by a convolutional PatchGAN classifier

PROPOSED SOLUTION (CONT.)

Build the generator :

The generator of your pix2pix cGAN is a modified U-Net. A U-Net consists of an encoder (downsampler) and decoder (upsampler). (You can find out more about it in the Image segmentation tutorial and on the U-Net project website.)

- Each block in the encoder is: Convolution -> Batch normalization -> Leaky ReLU
- Each block in the decoder is: Transposed convolution -> Batch normalization -> Dropout (applied to the first 3 blocks) -> ReLU
- There are skip connections between the encoder and decoder (as in the U-Net).

PROPOSED SOLUTION (CONT.)

Build the discriminator :

The discriminator in the pix2pix cGAN is a convolutional PatchGAN classifier—it tries to classify if each image patch is real or not real, as described in the pix2pix paper.

- Each block in the discriminator is: Convolution -> Batch normalization -> Leaky ReLU.
- The shape of the output after the last layer is (batch_size, 30, 30, 1).
- Each 30 x 30 image patch of the output classifies a 70 x 70 portion of the input image.
- The discriminator receives 2 inputs:
  - The input image and the target image, which it should classify as real.
  - The input image and the generated image (the output of the generator), which it should classify as fake.
  - Use tf.concat([inp, tar], axis=-1) to concatenate these 2 inputs together.

SYSTEM APPROACH

System Requirements :

1. Hardware :

- GPU: GPU with at least 12GB VRAM for faster training.
- RAM: At least 16GB RAM for handling large datasets efficiently.

2. Software :

Python: Version 3.x.

TensorFlow: Deep learning framework for building and training the cGAN.

Keras: High-level neural networks API (usually comes with TensorFlow) for easy model building.

System Requirements (cont.) :

Google Colab: Cloud-based Jupyter notebook environment with GPU support.

Matplotlib, NumPy, OpenCV: Commonly used libraries for data visualization and manipulation.

CMP Facade Database: Dataset for training the pix2pix model on building facades.

ALGORITHM & DEPLOYMENT:

Data Preparation :

Need to apply random jittering and mirroring to preprocess the training set.

Define several functions that:

- Resize each 256 x 256 image to a larger height and width—286 x 286.
- Randomly crop it back to 256 x 256.
- Randomly flip the image horizontally i.e., left to right (random mirroring).
- Normalize the images to the [-1, 1] range.

Build an input pipeline :

Create TensorFlow data pipelines for training and testing datasets using the CMP Facade Database. It loads image files, applies preprocessing functions, shuffles the training data, and batches both datasets to facilitate efficient model training and evaluation.

Build the generator and discriminator :

Generator Structure: The generator of the pix2pix cGAN employs a modified U-Net architecture, featuring an encoder (downsampler) and decoder (upsampler) with skip connections between them.

Encoder Blocks: Each block in the encoder consists of Convolution, Batch Normalization, and Leaky ReLU activation functions.

Decoder Blocks: Each block in the decoder comprises Transposed Convolution, Batch Normalization, Dropout (applied to the first 3 blocks), and ReLU activation functions.

Discriminator Design: The discriminator is a convolutional PatchGAN classifier, evaluating whether each image patch is real or fake. It receives two inputs: the input image and the target image (real), and the input image and the generated image (fake). The output shape after the last layer is (batch_size, 30, 30, 1), where each 30x30 image patch classifies a 70x70 portion of the input image.

Generate Images :

Write a function to plot some images during training.

- Pass images from the test set to the generator.
- The generator will then translate the input image into the output.
- The last step is to plot the predictions

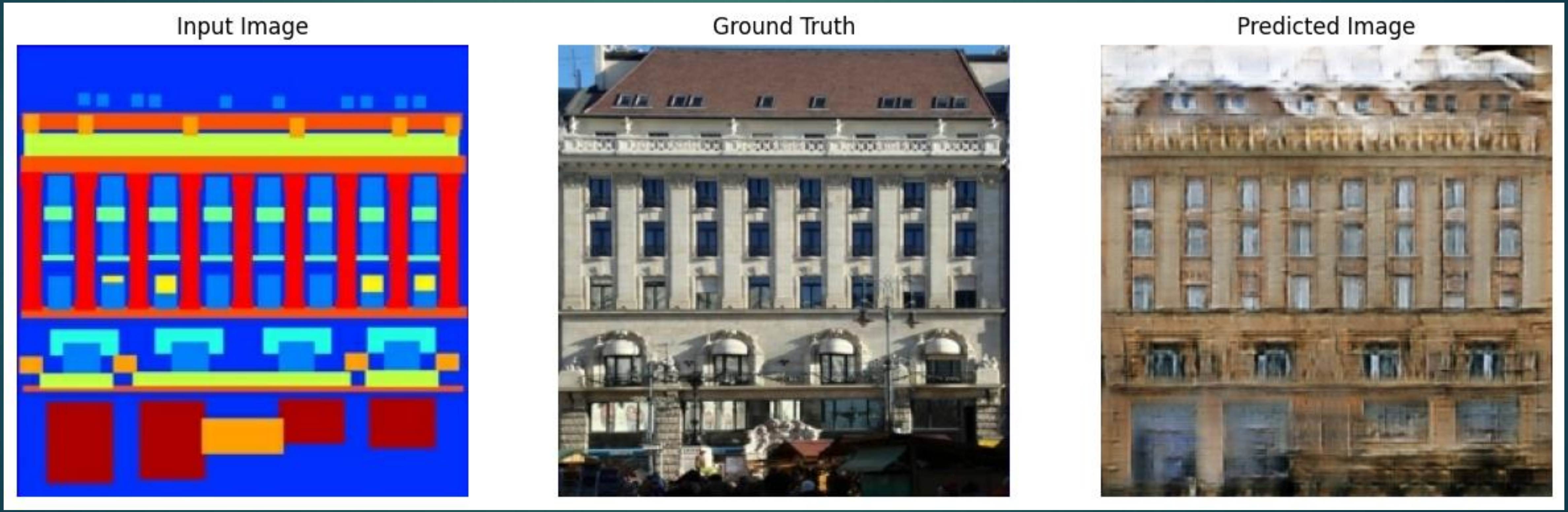
Training :

- For each example input generates an output.
- The discriminator receives the input_image and the generated image as the first input.
- The second input is the input_image and the target_image.
- Next, calculate the generator and the discriminator loss.
- Then, calculate the gradients of loss with respect to both the generator and the discriminator variables(inputs) and apply those to the optimizer.

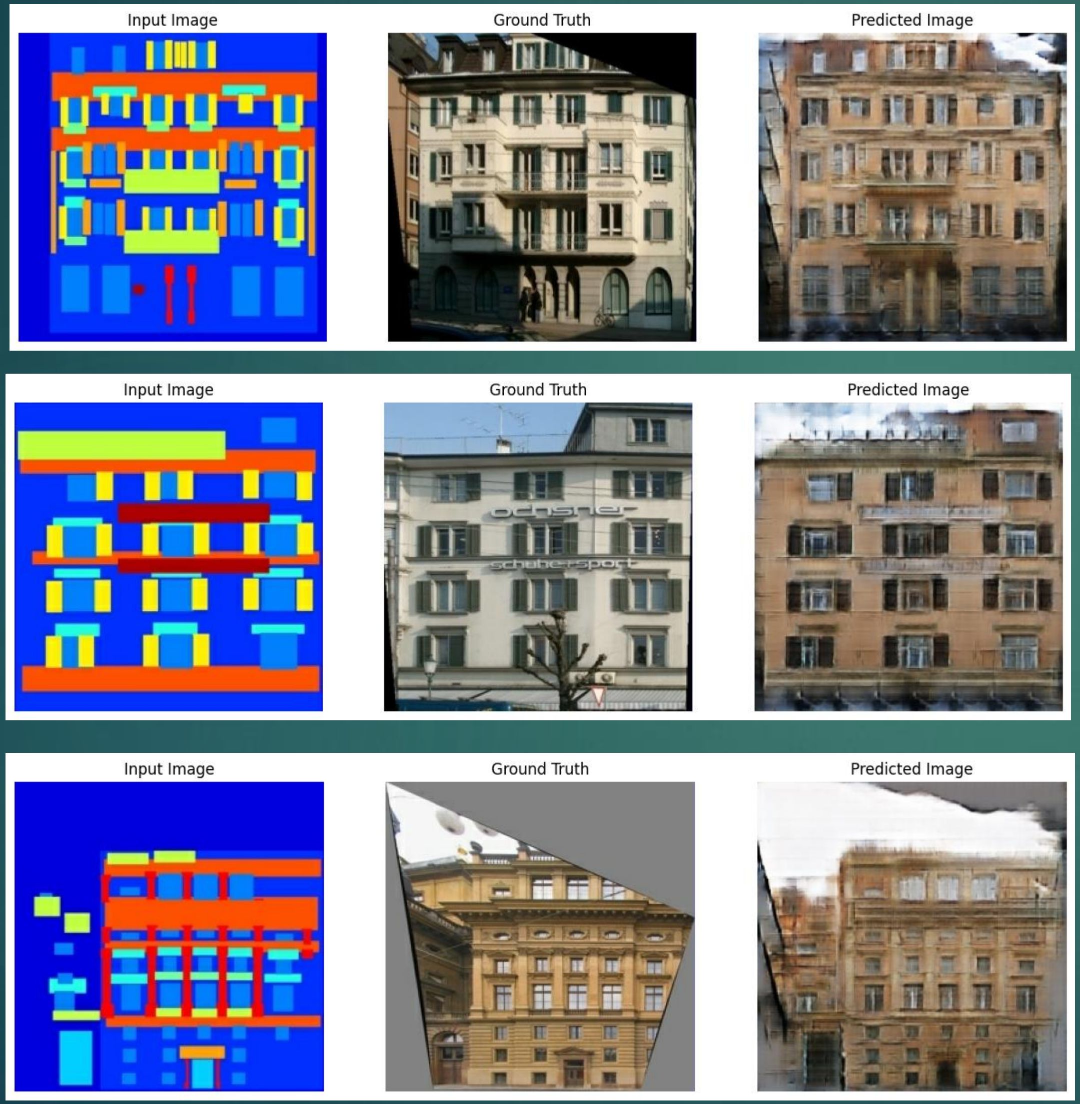
Finally, run the training loop:

```
fit(train_dataset, test_dataset, steps=40000)
```

(Time taken for 1000 steps: 115.74 sec)



Generate some images using the test set :



RESULTS

The project has successfully developed and trained a conditional generative adversarial network (cGAN), known as pix2pix, which excels in transforming input images into detailed output images resembling building facades. Utilizing the CMP Facade Database, the cGAN has been able to produce synthetic images that capture the intricate architectural elements of real building facades, such as windows, doors, and ornamental details.

CONCLUSION

In conclusion, the project's successful deployment of the pix2pix conditional generative adversarial network has effectively demonstrated the capability of this model to generate realistic images of building facades from the CMP Facade Database. This achievement not only underscores the model's proficiency in complex image-to-image translation tasks but also highlights its potential applications in architectural visualization and urban planning. The project sets a promising foundation for future advancements in synthetic image generation using advanced machine learning techniques.

REFERENCES

GenAI.ipynb - Colaboratory (google.com)