Configuration Manual

MSc Research Project
Data Analytics

Zuu Zuu Kyaw Shwe
Student ID: x24106585

School of Computing
National College of Ireland

Supervisor: Jaswinder Singh

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Zuu Zuu Kyaw Shwe
Student ID:	x24106585
Programme:	Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Jaswinder Singh
Submission Due Date:	15/09/2025
Project Title:	Configuration Manual
Word Count:	787
Page Count:	5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	*Zuu*
Date:	15th of September

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Zuu Zuu Kyaw Shwe
x24106585

1 Project Overview

This manual documents the configuration and execution process for the DCGAN and CGAN training and evaluation on CIFAR-10.

2 Environment Setup

1. Install Python 3.9 or newer.
2. Install the required packages:

```
pip install torch torchvision torchmetrics --quiet
pip uninstall -y torch-fidelity -q
pip install torch-fidelity==0.3.0 --quiet
pip install matplotlib tqdm --quiet
```

3. Ensure CUDA is installed and configured if GPU acceleration is desired.

3 Reproducibility

The following fixed seeds are used to ensure experiment reproducibility:

```
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed_all(SEED)
cudnn.deterministic = True
cudnn.benchmark = False
```

4 Google Drive Mounting (Colab Only)

If running in Google Colab, mount Google Drive to store checkpoints, generated images, and metrics:

```
from google.colab import drive
drive.mount('/content/drive')
```

5 Experiment Configuration

Each experiment is defined by a configuration dictionary `config`. Common keys:

```
config = {
    "experiment_id": "E1_Baseline",
    "model": "DCGAN", # or "CGAN"
    "regularization_type": None, # "L1", "L2", or None
    "regularization_lambda_L1": 0.0,
    "regularization_lambda_L2": 0.0,
    "regularization_placement": None, # e.g. "generator_output",
                                      # "discriminator_input",
                                      # "generator_weights",
                                      # "discriminator_weights_early",
                                      # "discriminator_weights_late"
    "dataset": "CIFAR-10",
    "image_size": 32,
    "latent_dim": 100,
    "num_classes": 10, # used in CGAN
    "num_epochs": 100,
    "batch_size": 128,
    "optimizer": "Adam",
    "learning_rate": 0.0002,
    "beta1": 0.5,
    "fid_samples": 10000,
    "save_images_every": 10,
    "timestamp": time.strftime("%Y-%m-%d %H:%M:%S"),
    "pytorch_version": torch.__version__,
    "cuda_version": torch.version.cuda if torch.cuda.is_available() else "None",
    "device": torch.device("cuda" if torch.cuda.is_available() else "cpu").type
}
```

6 Dataset Setup

This project uses the CIFAR-10 dataset, which contains 60,000 colour images of size 32×32 across 10 object classes. In line with common GAN practice, only the 50,000 training images are used for model training to ensure maximum diversity for the generator and discriminator.

For DCGAN experiments, all training images are normalised to the range $[-1, 1]$ to match the tanh activation output of the generator, which improves numerical stability during adversarial training.

For CGAN experiments, in addition to the same normalisation, class conditioning is introduced by mapping each class label to a learnable embedding using nn.Embedding. These embeddings are concatenated with:

- The noise vector in the generator before passing through transposed convolutions.
- The image tensor in the discriminator along the channel dimension (after spatial replication).

The preprocessing for training is implemented as follows:

```
transform_train = transforms.Compose([
    transforms.Resize(config["image_size"]),
    transforms.ToTensor(),
    transforms.Normalize((0.5,) * 3, (0.5,) * 3)
])
```

Evaluation Data Preparation (FID/IS): For calculating FID and IS, a separate copy of the training data is preprocessed with the same normalisation and stored as “real” reference images. This ensures that FID/IS compare the generated samples against the same distribution used for training rather than a separate test set.

```
transform_eval = transforms.Compose([
    transforms.Resize(config["image_size"]),
    transforms.CenterCrop(config["image_size"]),
    transforms.ToTensor(),
    transforms.Normalize((0.5,) * 3, (0.5,) * 3)
])
```

7 Model Architectures

- DCGAN Generator & Discriminator: Standard transposed convolution generator and convolutional discriminator, with batch normalization and ReLU/LeakyReLU activations
- CGAN Generator & Discriminator: Same as DCGAN but with label embeddings concatenated to the noise (generator) or image (discriminator) channels

8 Regularization Options

Applied according to regularization_placement:

- ”generator_output” - L1/L2 penalty on the generator’s output pixels
- ”discriminator_input” - L1/L2 penalty on the discriminator’s input (fake images)
- ”generator_weights” - L1/L2 penalty on all generator transposed convolution weights
- ”discriminator_weights_early” - penalty on the first 2 convolution layers of the discriminator
- ”discriminator_weights_late” - penalty on the last 2 convolution layers of the discriminator

9 Loss Functions

- Generator Loss: Binary cross-entropy with labels as real
- Discriminator Loss: Binary cross-entropy over real and fake images

10 Optimizer Settings

Default optimizer: Adam

```
optimizer_G = torch.optim.Adam(
    G.parameters(),
    lr=config["learning_rate"],
    betas=(config["beta1"], 0.999)
)
optimizer_D = torch.optim.Adam(
    D.parameters(),
    lr=config["learning_rate"],
    betas=(config["beta1"], 0.999)
)
```

11 Training Loop

For each epoch:

- Train the discriminator with real and fake batches
- Apply regularization if enabled
- Train the generator to fool the discriminator
- Log average losses for D and G
- Save sample images every 10 epochs
- Save checkpoints every 10 epochs or at the end

12 Checkpointing

Each experiment saves:

- checkpoint.pt - generator, discriminator, optimizers, and last epoch
- losses.csv - epoch wise discriminator and generator losses
- training_time.txt - total runtime

13 Evaluation Metrics

Two metrics are computed after training:

- FID (Frechet Inception Distance) - via torch_fidelity.calculate_metrics
- Inception Score - computed using a pretrained classifier to assess image quality and diversity

Real CIFAR-10 images are saved in /eval_data/real and generated images in /eval_data/fake before metric calculation.

14 Outputs

Each experiment directory contains:

- images/ - generated samples saved at specified epochs
- metrics/losses.csv - training losses
- metrics/loss_curve.png - plotted loss curve
- metrics/eval_metrics.txt - calculated FID and Inception Score
- eval_data/real/ - real CIFAR-10 images used for metric calculation
- eval_data/fake/ - generated images used for metric calculation
- checkpoint.pt - model state and optimizer state
- fixed_noise.pt - fixed latent vectors for consistent sampling
- config_manual.txt - experimental configuration details
- training_time.txt - total training duration