National College of Ireland

Configuration Manual

MSc Research Project
Masters in Data Analytics

Simi Correia
Student ID: x23320818

School of Computing
National College of Ireland

Supervisor: Teerath Kumar Menghwar

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Simi Correia
Student ID:	x23320818
Programme:	Masters in Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Teerath Kumar Menghwar
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	377
Page Count:	3

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:	Simi Silvester Correia
Date:	15th September 2025

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

Simi Correia
x23320818

1 System Requirements

To execute the code successfully and efficiently, the following hardware and software requirements were taken into consideration.

1.1 Hardware Requirements

- Google account with sufficient Google Drive storage for datasets and generated images (approximately 1 GB).
- Stable internet connection.
- GPU access in Google Colab, such as T4, P100, or A100, for faster training (He et al.; 2016). In this project, a Colab T4 GPU was used.

1.2 Software Requirements

- Google Colab with GPU runtime.
- Python 3.10 (default in Colab; can be upgraded to the latest version if required).
- Installed libraries:
  - **Core Python Libraries:** `os`, `shutil`, `glob`.
  - **Data Processing:** `pandas`, `numpy`.
  - **Image Handling & Processing:** `opencv-python`, `Pillow`, `matplotlib`, `seaborn`.
  - **Google Drive Integration:** `google.colab`.
  - **Deep Learning Framework:** `torch`, `torchvision`.
  - **Model-Specific / Pretrained Networks:** `efficientnet_pytorch` (Tan and Le; 2019), `timm`.
  - **Stable Diffusion v5:** `diffusers`, `transformers`, `accelerate`, `safetensors`, `xformers` (Rombach et al.; 2022).
  - **Other Utilities:** `scikit-learn`, `tqdm`, `requests`.

2 Analysis Phases

The analysis was split into two phases:

Phase 1

The author first mounted Google Drive in Colab to access the dataset. After uploading and processing the images, the author preprocessed them by resizing and formatting for consistency. The author then trained the **ResNet-18** model (He et al.; 2016) and evaluated its performance in terms of accuracy, precision, recall, and ROC-AUC Curve. Grad CAM visualization was used to check how the model was being trained.

Phase 2

In the second phase, synthetic images were generated using the Stable Diffusion method (Rombach et al.; 2022), enriching the dataset. The training process was repeated using both **ResNet-18** (He et al.; 2016) and **EfficientNet-B3** (Tan and Le; 2019). The author then collected and saved results from both models to understand their performance with a larger and more varied dataset.

3 Setup for PreTrained Models

Phase 1 – ResNet-18 Training

Figure 1 shows the ResNet-18 configuration and training setup used in Phase 1 (He et al.; 2016). This setup ensures that the model is initialised with pre-trained weights, the computation device (GPU-T4) is detected, and parameters are ready for training. Having this configuration is essential for achieving consistent and reproducible results.

Pre Trained Resnet 18 Model

```
# Check device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

# Load pre-trained ResNet18
model = models.resnet18(pretrained=True)

# Replace the final layer for 2 classes
num_features = model.fc.in_features
model.fc = nn.Linear(num_features, 2)

# Move to device
model = model.to(device)
print(model)
```

Figure 1: Pre-trained ResNet-18 model setup.

Phase 2 – Stable Diffusion and EfficientNet-B3

Figures 2 and 3 illustrate the setup used in Phase 2, which combines synthetic data generation using Stable Diffusion (Rombach et al.; 2022) and advanced model training with EfficientNet-B3 (Tan and Le; 2019). The Stable Diffusion configuration is required to create realistic, high-quality synthetic images that expand the training dataset, while the EfficientNet-B3 setup ensures the model is optimised for performance. Both configurations are critical for reproducibility of the project.

```
model_id = "runwayml/stable-diffusion-v1-5"  # SD v1.5 for stable results
pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16)
pipe = pipe.to("cuda")  # Use GPU for faster generation
```

Figure 2: Stable Diffusion model setup for synthetic image generation.

EfficientNet-B3

```
# Device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)

# Load EfficientNet-B3 pretrained
model = models.efficientnet_b3(weights=models.EfficientNet_B3_Weights.IMAGENET1K_V1)

# Replace classifier head for 2 classes
model.classifier[1] = nn.Linear(model.classifier[1].in_features, 2)

model = model.to(device)

# Loss + optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
```

Figure 3: EfficientNet-B3 configuration and optimizer setup.

References

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep residual learning for image recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 770–778. Accessed: 09 August 2025.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 10684–10695. Accessed: 09 August 2025.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks, *Proceedings of the 36th International Conference on Machine Learning (ICML)* pp. 6105–6114. Accessed: 09 August 2025.