National College of Ireland

# Configuration Manual

MSc Research Project
Master's in data Analytics

Vishnu Vardhan Reddy Rikkula
Student ID: X23409452

School of Computing
National College of Ireland

Supervisor: Mr. Jaswinder Singh

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Vishnu Vardhan Reddy Rikkula |
| **Student ID:** | 23409452 |
| **Programme:** | Master's in data Analytics **Year:** 2025-2026 |
| **Module:** | Research Practicum |
| **Lecturer:** | Mr. Jaswinder Singh |
| **Submission Due Date:** | 28-01-2026 |
| **Project Title:** | Self-Supervised Learning for Class-Imbalanced Guava Disease Detection: A Contrastive Reconstruction Enhancement Approach |
| **Word Count:** | **466** **Page Count: 6** |

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:** Vishnu Vardhan Reddy Rikkula

**Date:** 28-01-2026

**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): | |

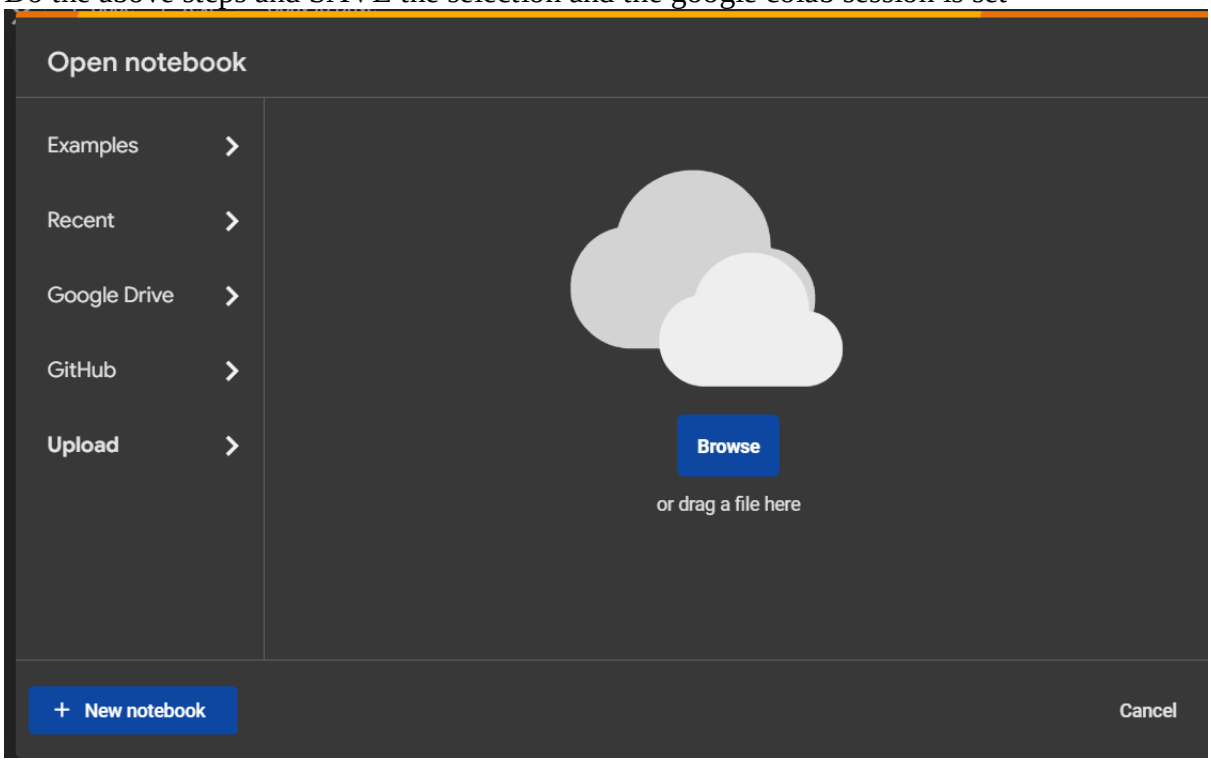
# Configuration Manual

Vishnu Vardhan Reddy Rikkula
X23409452

## 1. Connecting a Colab session to Google Drive for file access

Step 1 : Launching the colab environment (navigating to google colab website)
Step 2: Sign in into the google account
Step 3:uploading the ipynb file /notebook
Step 4: Enabling the GPU Acceleration ( in this change the runtime)
Step 5 : change the hardware accelerator to T4 GPU

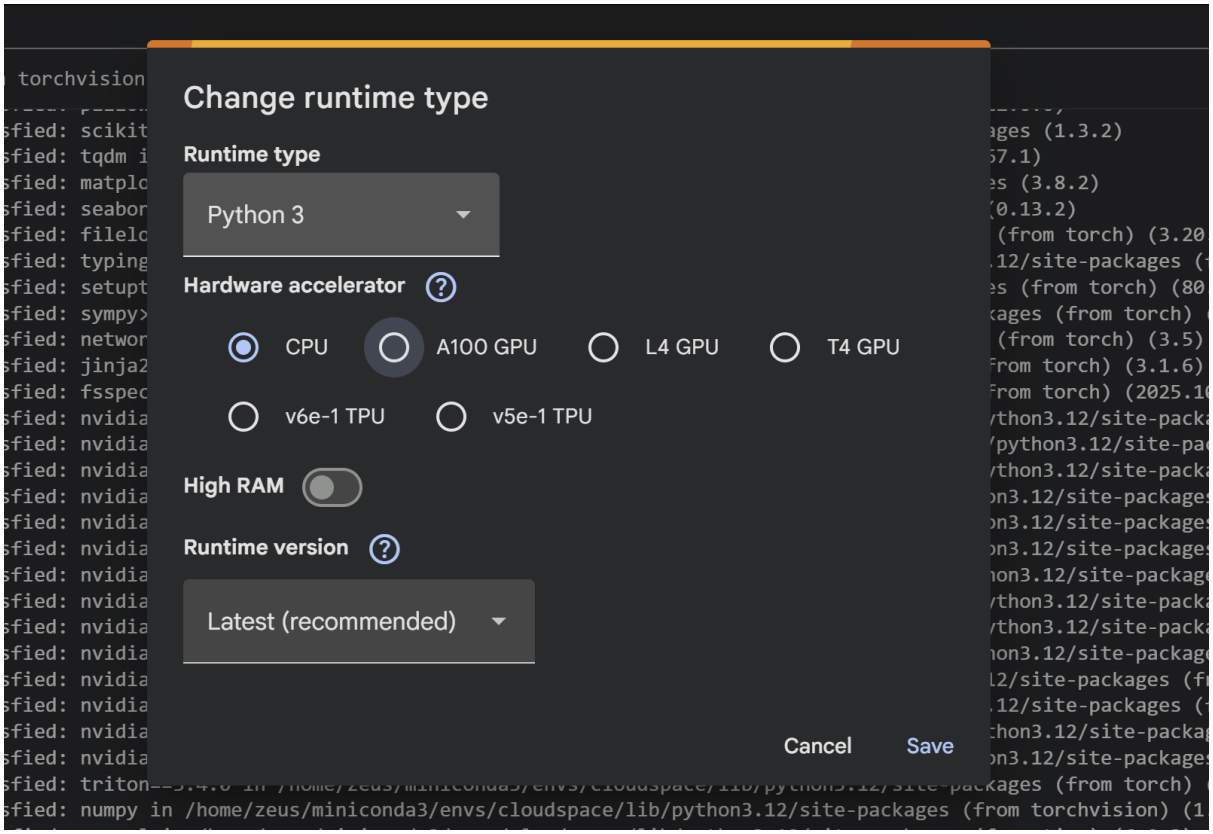
Do the above steps and SAVE the selection and the google colab session is set




## 2. Preparing the session environment

Step 1: open the google colab session
Step 2:Sign in to your Gmail

Step 3:start a new note book
Step 4 : Activate the GPU acceleration
Step 5:Create a folder on the website
1.Upload the notebook




# 3.Data import

Step 1:Setting up the configuration and data import

1. Here we are importing 3 data sets

```
    confusion_matrix, classification_report
)
from sklearn.model_selection import train_test_split

# Configuration
CONFIG = {
    # Paths
    "base_path": "vishnu_Thesis/",
    "plant_leaf_path": "vishnu_Thesis/plant_leaf/",
    "guava_1_path": "vishnu_Thesis/Guava-1/",
    "guava_2_path": "vishnu_Thesis/Guava-2/",
    "guava_final_path": "vishnu_Thesis/GuavaFinal/",
    "checkpoint_path": "vishnu_Thesis/checkpoints/",

    # Dataset URLs
    "plantvillage_url": "https://data.mendeley.com/public-files/datasets/tywbtsjrjv/files/b4e3a32f-c0bd-4060-81e9-6144231f2520/file_downloaded",
    "guava_original_url": "https://data.mendeley.com/public-files/datasets/2ksdzxdvbm/files/e4c12f85-7f56-41f6-8bb9-a55dc1073b43/file_downloaded",
    "guava_augmented_url": "https://data.mendeley.com/public-files/datasets/2ksdzxdvbm/files/b7a8aadd-1502-4459-848b-74b46d99aed6/file_downloaded",
```

Step 2:Downloading the data import to the notebook

```
    class_dirs = [d for d in directory.iterdir() if d.is_dir()]
    total = 0
    for cls_dir in class_dirs:
        for p in cls_dir.rglob("*"):
            if p.is_file() and is_image(p):
                total += 1
    return len(class_dirs), total

def download_plantvillage():
    base_path = Path(CONFIG["base_path"])
    base_path.mkdir(parents=True, exist_ok=True)

    zip_path = base_path / "plant_leaf.zip"
    extract_root = Path(CONFIG["plant_leaf_path"])

    # Check if already extracted
    if extract_root.exists():
        num_classes, num_images = count_images_in_dir(extract_root)
        if num_images > 60000:
            print("PlantVillage already downloaded and extracted")
            print("Path:", extract_root)
            print("Classes:", num_classes)
            print("Images:", num_images)
            return extract_root

    extract_root.mkdir(parents=True, exist_ok=True)

    # Download
    print("Downloading PlantVillage dataset...")
    url = CONFIG["plantvillage_url"]
    exit_code = subprocess.call(["curl", "-L", url, "-o", str(zip_path)])
```

```
# ============================================================================
# CELL 3: Download and Merge Guava Dataset (Fine-tuning)
# ============================================================================

def get_file_hash(filepath):
    hash_md5 = hashlib.md5()
    with open(filepath, "rb") as f:
        for chunk in iter(lambda: f.read(4096), b""):
            hash_md5.update(chunk)
    return hash_md5.hexdigest()

def normalize_class_name(name):
    name = name.strip().rstrip("/")
    name = name.replace("Healthly", "Healthy")
    name = name.replace("Insects eatten class", "Insects eaten")
    return name

def download_guava_datasets():
    base_path = Path(CONFIG["base_path"])
    base_path.mkdir(parents=True, exist_ok=True)

    # Download Guava Original
    zip1_path = base_path / "guava_original.zip"
    extract1_root = Path(CONFIG["guava_1_path"])

    if not extract1_root.exists() or count_images_in_dir(extract1_root)[1] < 500:
        extract1_root.mkdir(parents=True, exist_ok=True)
        print("Downloading Guava Original dataset...")
        url = CONFIG["guava_original_url"]
        exit_code = subprocess.call(["curl", "-L", url, "-o", str(zip1_path)])
```

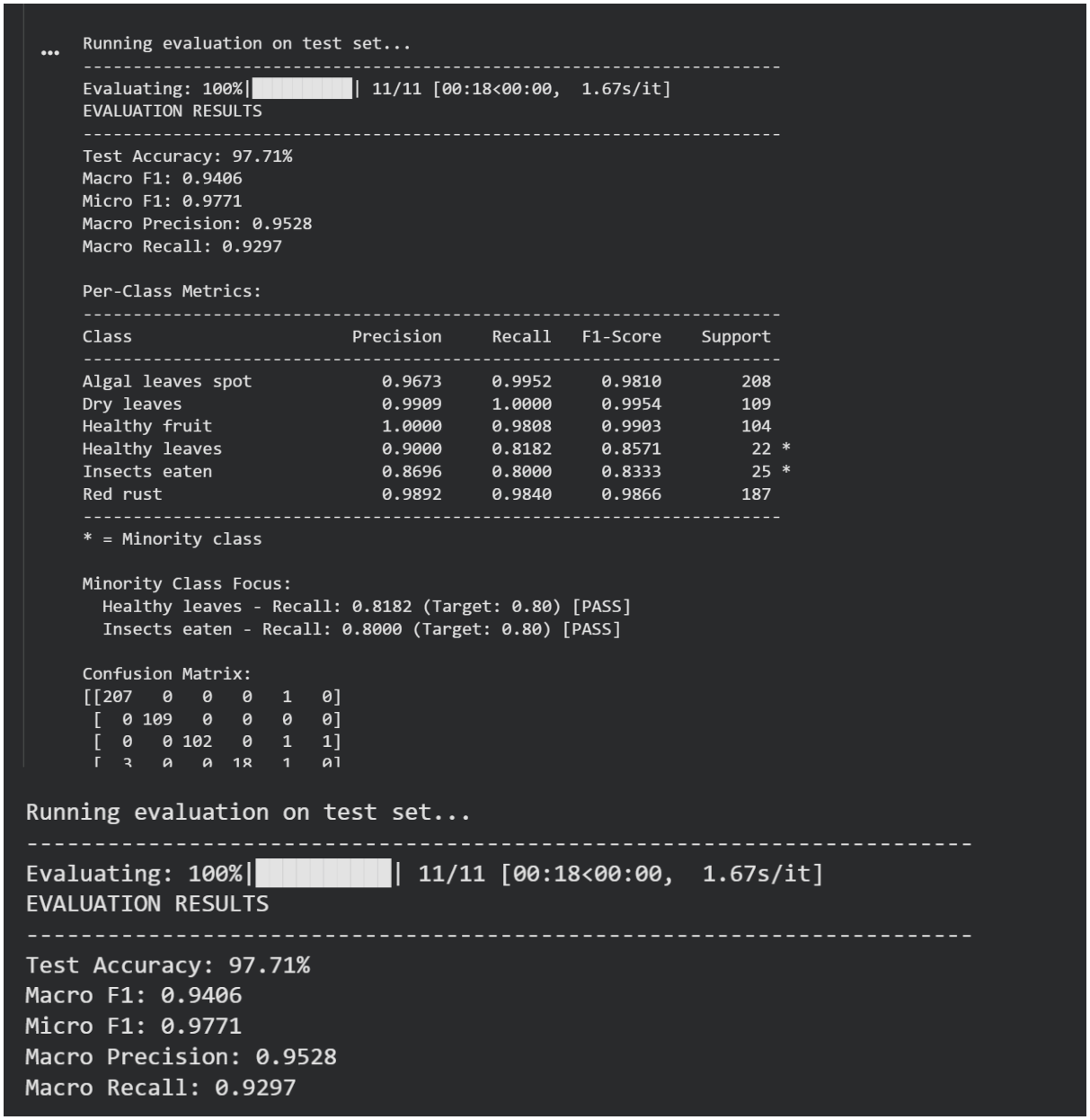
# 4.Results

```python

# ================================================================================
# CELL 9: Pre-Training Loop (PlantVillage)
# ================================================================================

def pretrain_cre(model, train_loader, config, device):
    """
    Pre-train CRE model on PlantVillage dataset.
    """
    model = model.to(device)
    model.train()

    # Optimizer
    optimizer = optim.AdamW(
        model.parameters(),
        lr=config["pretrain_lr"],
        weight_decay=config["pretrain_weight_decay"]
    )

    # Scheduler with warmup
    total_steps = config["pretrain_epochs"] * len(train_loader)
    warmup_steps = config["warmup_epochs"] * len(train_loader)

    def lr_lambda(step):
        if step < warmup_steps:
            return step / warmup_steps
        else:
            progress = (step - warmup_steps) / (total_steps - warmup_steps)
            return 0.5 * (1 + np.cos(np.pi * progress))

    scheduler = optim.lr_scheduler.LambdaLR(optimizer, lr_lambda)

    # Mixed precision
    scaler = torch.amp.GradScaler("cuda")
```

```
    Decoder depth: 2
    Patch size: 16
  Projection Head initialized
    Input dim: 768
    Hidden dim: 2048
    Output dim: 256

  CRE Model initialized
    Num patches: 196
    Mask ratio: 0.5
    Lambda contrastive: 0.5

  Starting CRE Pre-training
  ----------------------------------------------------------------------
  Epochs: 50
  Batch size: 256
  Learning rate: 0.00015
  Total batches per epoch: 240
  ----------------------------------------------------------------------
  Epoch 1/50: 100%|██████████| 240/240 [07:38<00:00,  1.91s/it, recon=0.8754, contrast=4.2374, total=2.9941]
  Epoch 1/50 | Recon Loss: 1.0437 | Contrastive Loss: 5.2291 | Total Loss: 3.6583 | LR: 0.000015 | Time: 458.3s
  Epoch 2/50: 100%|██████████| 240/240 [07:33<00:00,  1.89s/it, recon=0.7493, contrast=2.8540, total=2.1763]
  Epoch 2/50 | Recon Loss: 0.7858 | Contrastive Loss: 3.3971 | Total Loss: 2.4843 | LR: 0.000030 | Time: 453.2s
  Epoch 3/50: 100%|██████████| 240/240 [07:36<00:00,  1.90s/it, recon=0.6871, contrast=2.0158, total=1.6951]
  Epoch 3/50 | Recon Loss: 0.7083 | Contrastive Loss: 2.3186 | Total Loss: 1.8676 | LR: 0.000045 | Time: 456.3s
  Epoch 4/50: 100%|██████████| 240/240 [07:35<00:00,  1.90s/it, recon=0.6725, contrast=1.8392, total=1.5921]
  Epoch 4/50 | Recon Loss: 0.6732 | Contrastive Loss: 1.7922 | Total Loss: 1.5693 | LR: 0.000060 | Time: 455.5s
  Epoch 5/50: 100%|██████████| 240/240 [07:32<00:00,  1.89s/it, recon=0.6497, contrast=1.2109, total=1.2552]
  Epoch 5/50 | Recon Loss: 0.6673 | Contrastive Loss: 1.4776 | Total Loss: 1.4061 | LR: 0.000075 | Time: 452.7s
  Epoch 6/50: 100%|██████████| 240/240 [07:28<00:00,  1.87s/it, recon=0.6568, contrast=1.1781, total=1.2458]
  Epoch 6/50 | Recon Loss: 0.6637 | Contrastive Loss: 1.2771 | Total Loss: 1.3023 | LR: 0.000090 | Time: 448.5s
  Epoch 7/50: 100%|██████████| 240/240 [07:36<00:00,  1.90s/it, recon=0.6607, contrast=1.0349, total=1.1781]
  Epoch 7/50 | Recon Loss: 0.6599 | Contrastive Loss: 1.1333 | Total Loss: 1.2265 | LR: 0.000105 | Time: 456.1s
  Epoch 8/50: 100%|██████████| 240/240 [07:34<00:00,  1.89s/it, recon=0.6583, contrast=0.9167, total=1.1167]
  Epoch 8/50 | Recon Loss: 0.6586 | Contrastive Loss: 1.0134 | Total Loss: 1.1653 | LR: 0.000120 | Time: 454.6s
```

```python

# ============================================================================
# CELL 11: Fine-Tuning Loop (Guava) - With Focal Loss for Hard Samples
# ============================================================================

class FocalLoss(nn.Module):
    """Focal Loss to focus on hard-to-classify samples."""
    def __init__(self, alpha=None, gamma=2.0, reduction='mean'):
        super().__init__()
        self.alpha = alpha  # Class weights
        self.gamma = gamma  # Focusing parameter
        self.reduction = reduction

    def forward(self, inputs, targets):
        ce_loss = F.cross_entropy(inputs, targets, weight=self.alpha, reduction='none')
        pt = torch.exp(-ce_loss)
        focal_loss = ((1 - pt) ** self.gamma) * ce_loss

        if self.reduction == 'mean':
            return focal_loss.mean()
        elif self.reduction == 'sum':
            return focal_loss.sum()
        return focal_loss


def finetune_classifier(model, train_loader, val_loader, config, device):
    """Fine-tune classifier on Guava dataset with Focal Loss."""
    model = model.to(device)

    # Optimizer
    optimizer = optim.AdamW(
        filter(lambda p: p.requires_grad, model.parameters()),
        lr=config["finetune_lr"],
        weight_decay=0.02
    )

    # Class weights - balanced for both minority classes
    class_weights = torch.tensor([
        1.0,  # Algal leaves spot (1385)
```

```python

# ============================================================================
# CELL 12: Evaluation Functions
# ============================================================================

def compute_metrics(y_true, y_pred, class_names):
    """Compute comprehensive metrics."""
    metrics = {}

    # Overall metrics
    metrics["accuracy"] = accuracy_score(y_true, y_pred) * 100
    metrics["macro_f1"] = f1_score(y_true, y_pred, average="macro")
    metrics["micro_f1"] = f1_score(y_true, y_pred, average="micro")
    metrics["macro_precision"] = precision_score(y_true, y_pred, average="macro")
    metrics["macro_recall"] = recall_score(y_true, y_pred, average="macro")

    # Per-class metrics
    per_class_precision = precision_score(y_true, y_pred, average=None, zero_division=0)
    per_class_recall = recall_score(y_true, y_pred, average=None, zero_division=0)
    per_class_f1 = f1_score(y_true, y_pred, average=None, zero_division=0)

    metrics["per_class"] = {}
    for i, class_name in enumerate(class_names):
        metrics["per_class"][class_name] = {
            "precision": per_class_precision[i],
            "recall": per_class_recall[i],
            "f1": per_class_f1[i],
            "support": int(np.sum(np.array(y_true) == i))
        }

    return metrics


def generate_confusion_matrix(y_true, y_pred, class_names):
    """Generate confusion matrix."""
    cm = confusion_matrix(y_true, y_pred)
    return cm

```




# 5. References

**Google Colab (Reference):** https://colab.research.google.com/

Bangladesh Guava Disease Dataset. (2024). Guava leaf disease images from Bangladesh. Mendeley Data, v1. DOI: 10.17632/2ksdzxdvbm.1.

PlantVillage Dataset. (2019). Plant leaf diseases dataset with augmentation. Mendeley Data, v1. DOI: 10.17632/tywbtsjrjv.1.