Supplementary materials

Title: MT-TransUNet: Efficient Multi-Task Vision Transformer for Agricultural Cellular Phenotyping in Microscopy Images

Shitephen Wang[1,*]

[1]School of Forestry and Resource Conservation, National Taiwan University, Taipei 10617, Taiwan

* Corresponding author: shitephenwang@ntu.edu.tw

Material S1 - User Interface Documentation

Multi-Task TransUNet Software User Guide

Table of Contents

S1.1. Introduction

This supplementary material provides comprehensive documentation for the graphical user interfaces developed for the Multi-Task TransUNet biological image segmentation system. The software comprises two distinct applications:

(a) **Prediction Interface** (app_gui.py): A desktop application built with Tkinter for model loading, single image prediction, and batch processing.
(b) **Training Interface** (app_train.py): A web-based application built with Gradio for model training and monitoring.

Both interfaces are designed to be accessible to researchers without extensive programming experience, whilst providing sufficient flexibility for advanced users.

S1.2. System Requirements

S1.2.1 Minimum Requirements

Component	Specification
Operating System	Windows 10
RAM	16 GB
CPU	Intel Core i5 or equivalent
Storage	50 GB available space
Python	3.8 or higher

S1.2.2 Recommended Requirements

Component	Specification
Operating System	Windows 11
RAM	32 GB or higher
CPU	Intel Core i7 or equivalent
GPU	NVIDIA RTX 5080 (16 GB VRAM)
Storage	100 GB SSD
Python	3.10 or higher

S1.2.3 GPU Compatibility

The software supports all NVIDIA GPUs with CUDA capability, including:

- GeForce RTX 50 series (5080, 5090)[1]

[1] GeForce RTX 40 series (4060) and GeForce RTX 50 series (5060) were developed in an early single-task version on Github-- https://github.com/gn03138868/multiTUNAparty-model/tree/main/Old_version_party-model/party-model

S1.3. Installation

S1.3.1 Environment Setup

```
# Create a new conda environment
conda create -n multitask python=3.11.14
conda activate multitask

# Install PyTorch with CUDA 12.8
pip install torch==2.10.0.dev20251122+cu128 torchvision==0.25.0.dev20251122+cu128
torchaudio==2.10.0.dev20251122+cu128 --index-url
https://download.pytorch.org/whl/nightly/cu128

# Install other dependencies
pip install albumentations==2.0.8 opencv-python==4.11.0.86 pillow==12.0.0
pip install timm==1.0.22 transformers==4.54.1
pip install numpy==1.26.4 scipy==1.16.3 pandas==2.3.3 scikit-learn==1.6.1 scikit-
image==0.26.0
pip install matplotlib==3.9.4 gradio==6.1.0 tqdm==4.67.1 pyyaml==6.0.3
```

S1.3.2 Verify Installation

```
# Verify GPU availability
python -c "import torch; print(f'CUDA available: {torch.cuda.is_available()}')"
python -c "import torch; print(f'GPU: {torch.cuda.get_device_name(0)}')"
```

S1.3.3 Directory Structure

Ensure your project directory follows this structure:

```
multiparty-model/
├── app_gui.py              # Prediction interface
├── app_train.py            # Training interface
├── model_multitask.py      # Model architecture
├── dataset_multitask.py    # Dataset loader
├── losses_multitask.py     # Loss functions
├── train_multitask.py      # Training script
├── data/                   # Dataset directory
│   ├── train/
│   │   ├── cell/images/ + masks/
│   │   ├── blood/images/ + masks/
│   │   └── Other/images/ + masks/
│   └── val/ (same structure)
└── outputs/                # Output directory
```

S1.4. Software Architecture

S1.4.1 Design Rationale

The separation of training and prediction interfaces addresses several practical considerations:

- **Stability:** Tkinter provides a stable, dependency-free desktop experience for routine prediction tasks.
- **Flexibility:** Gradio offers a modern web interface ideal for parameter adjustment during training.
- **Compatibility:** Both interfaces maintain compatibility across different operating systems and hardware configurations.

S1.5. Prediction Interface (app_gui.py)

S1.5.1 Launching the Application

```
python app_gui_en.py
```

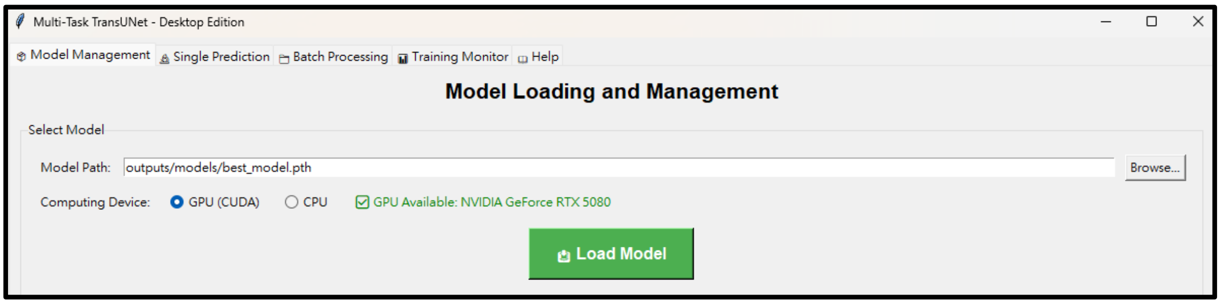
The application window will appear with five tabs: Model Management, Single Prediction, Batch Processing, Training Monitor.

S1.5.2 Model Management Tab

This tab handles model loading and configuration.

Loading a Model

(a) Click **Browse...** to select a model file (.pth format)
(b) Select the computing device (GPU recommended for faster inference)
(c) Click **Load Model**



Model Path: outputs/models/best_model.pth

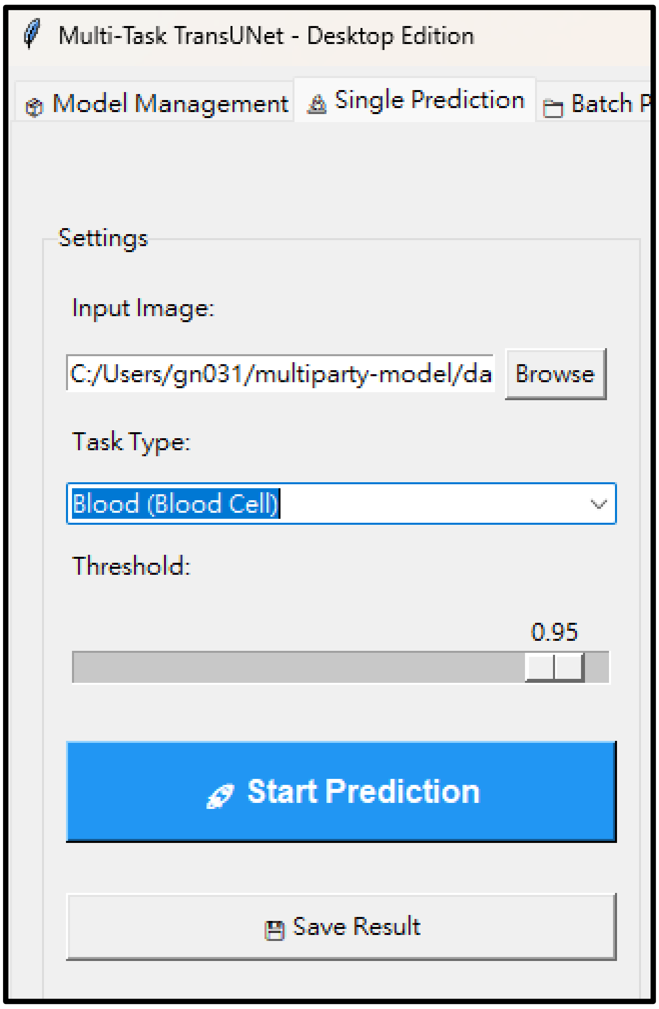
Computing Device: ● GPU (CUDA) ○ CPU ☑ GPU Available: NVIDIA GeForce RTX 5080

S1.5.3 Single Prediction Tab

This tab enables prediction on individual images.

Workflow

(a) Click **Select Image File** to choose an input image
(b) Select the appropriate **Task Type** (Cell, Blood, or Root)
(c) Adjust the **Segmentation Threshold** (0.0-1.0)
(d) Click **Start Prediction**



Input Image: C:/Users/gn031/multiparty-model/da

Task Type: Blood (Blood Cell)

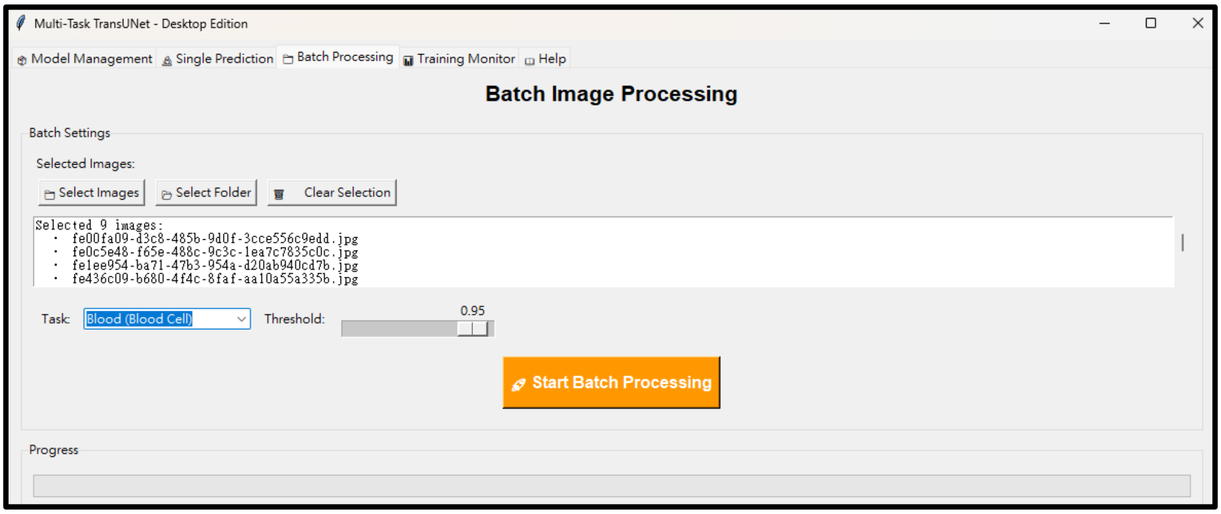
Threshold: 0.95

Recommended Threshold Values

Task	Recommended Range	Notes
Cell	0.5 - 0.7	Higher values for cleaner boundaries
Blood	0.4 - 0.6	Medium values for circular structures
Others	Test by yourself	

S1.5.4 Batch Processing Tab

This tab enables the processing of multiple images simultaneously.



```
Selected 9 images:
  ・ fe00fa09-d3c8-485b-9d0f-3cce556c9edd.jpg
  ・ fe0c5e48-f65e-488c-9c3c-1ea7c7835c0c.jpg
  ・ fe1ee954-ba71-47b3-954a-d20ab940cd7b.jpg
  ・ fe436c09-b680-4f4c-8faf-aa10a55a335b.jpg
```

Task: Blood (Blood Cell)

Threshold: 0.95

Output Structure

Results are saved to a timestamped directory:

```
outputs/batch_predictions/YYYYMMDD_HHMMSS_TaskName/
├── image1_original.png    # Original image
├── image1_heatmap.png     # Probability heatmap
├── image1_binary.png      # Binary mask
└── image1_overlay.png     # Overlay visualisation
```

S1.6. Training Interface (app_train.py)

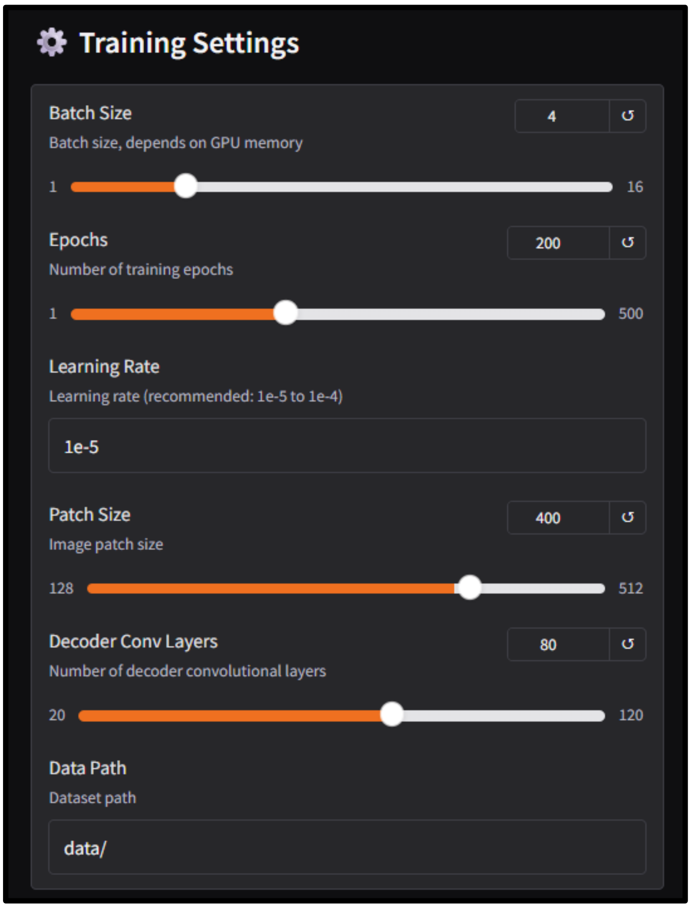
S1.6.1 Launching the Application

```
python app_train_en.py
```

A web browser will automatically open at http://localhost:7860. If not, navigate to this address manually.

S1.6.2 Training Settings

Basic Parameters



Training Settings

Batch Size: 4
Batch size, depends on GPU memory

Epochs: 200
Number of training epochs

Learning Rate
Learning rate (recommended: 1e-5 to 1e-4)
1e-5

Patch Size: 400
Image patch size

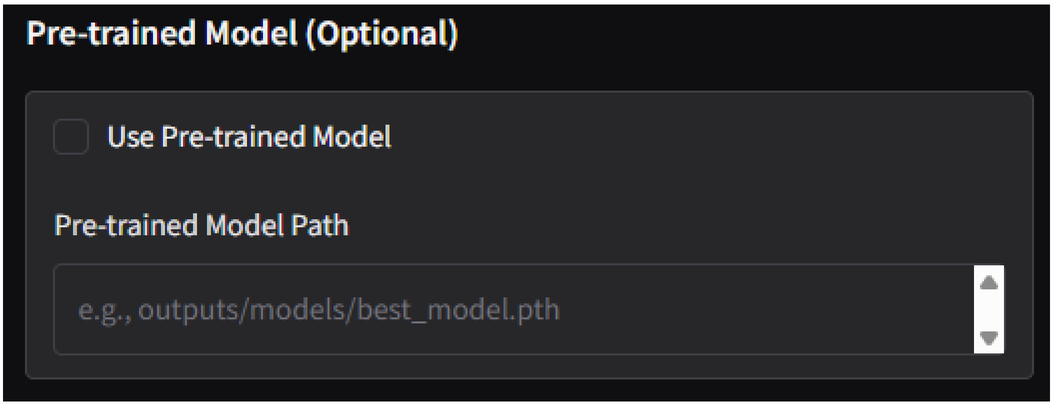
Decoder Conv Layers: 80
Number of decoder convolutional layers

Data Path
Dataset path
data/

Parameter	Default	Range	Description
Batch Size	4	1-16	Samples per iteration
Epochs	200	1-500	Number of training cycles
Learning Rate	1e-5	1e-6 to 1e-3	Optimiser step size
Patch Size	400	128-512	Input image dimensions
Decoder Layers	80	20-120	Convolutional layers

Pre-trained Model (Optional)

Enable **Use Pre-trained Model** to continue training from an existing checkpoint.



Pre-trained Model (Optional)

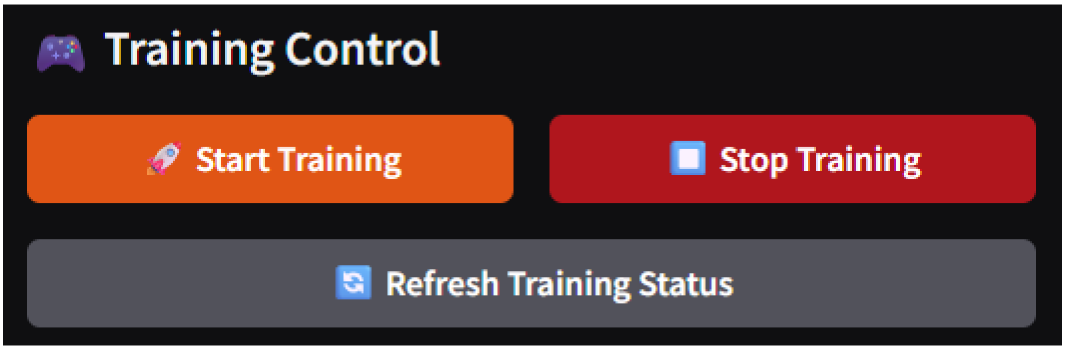
□ Use Pre-trained Model

Pre-trained Model Path

e.g., outputs/models/best_model.pth

Training control

Press “Start Training" to start training for your images. It is also possible to stop it if you like.



6.3 Output Files

File	Location	Description
Best Model	outputs/models/best_model.pth	Highest validation IoU
Final Model	outputs/models/final_model.pth	Last epoch weights
Checkpoints	outputs/models/checkpoint_epoch*.pth	Periodic saves
Training History	outputs/training_history.json	Metrics per epoch

S1.7. Troubleshooting

7.1 Common Issues and Solutions

Issue: CUDA out of memory

Solution: Reduce batch size or use CPU mode.

Issue: Module not found: model_multitask

Solution: Ensure all Python files are in the same directory.

Issue: Prediction results are entirely black

Solution:

(a) Lower the threshold value (try 0.3 or lower)
(b) Verify the model loaded correctly
(c) Check that the task type matches the input image

Issue: GPU not detected

Solution: Reinstall PyTorch with CUDA support:

```
pip uninstall torch torchvision
pip install torch torchvision --index-url https://download.pytorch.org/whl/cu118
```

S1.8. Frequently Asked Questions

Q: Can I use the tool without a GPU?

A: Yes. Both interfaces support CPU-only operation, though inference will be slower (approximately 10-50x compared to GPU).

Q: What image formats are supported?

A: The software accepts JPEG (.jpg, .jpeg) for original training and PNG (.png) for mask image.

Q: How do I resume training from a checkpoint?

A: In the Training Interface, enable "Use Pre-trained Model" and specify the checkpoint path.

Q: Where are my batch prediction results saved?

A: Results are saved to outputs/batch_predictions/YYYYMMDD_HHMMSS_TaskName/.

Q: Can I train on my own dataset?

A: Yes. Organise your data following the directory structure specified in Section S1.3.3, ensuring images and corresponding masks share the same filename.

Q: What is the recommended threshold for my images?

A: Start with the recommended values in Section S1.5.3 and adjust based on visual inspection. Lower thresholds capture more detail but may include noise; higher thresholds produce cleaner results but may miss fine structures.

Q: How long does training take?

A: Training duration depends on dataset size, hardware, and parameters. As a reference, 200 epochs with 1,000 training images on an RTX 5080 requires approximately 4-6 hours.

Version Information

- Last Updated: January 2026
- If you have any further questions, please feel free to contact Shitephen WANG (email: gn03138868@gmail.com; shitephenwang@ntu.edu.tw)