Configuration Manual

MSc Research Project
Data Analytics

Siddhant Bahadkar
Student ID: x23270926

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Siddhant Bahadkar
Student ID:	x23270926
Programme:	Data Analytics
Year:	2025
Module:	MSc Research Project
Supervisor:	Vladimir Milosavljevic
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	1154
Page Count:	9

Signature:	Siddhant Bahadkar
Date:	10th August 2025

Configuration Manual

Siddhant Bahadkar
x23270926

1 Hardware Specifications

This project was carried out on a personal laptop and the specifications are as follows:

- **Processor:** AMD Ryzen 5 5600H (3.30GHz)
- **Graphics:** NVIDIA GeForce RTX 3050 (4GB VRAM)
- **Memory:** 16GB DDR4 RAM
- **Storage:** Enough capacity for 83,000+ training images, saved models, histories, and generated results.
- **Operating System:** Windows 10 (64-bit).

Note: TensorFlow with GPU acceleration is officially supported only up to TensorFlow 2.10, if you wish to use it on Windows. Go through this link [1] *for more detailed explanation and steps on the official TensorFlow installation guide.*

2 GPU and Software Requirements

2.1 GPU and CUDA Requirements

The system uses TensorFlow-GPU version 2.10, which requires:

- **Compatible GPU architectures:** NVIDIA GPUs with CUDA Compute Capability ≥ 3.5 (including 5.0, 6.0, 7.0, 7.5, and 8.0+).
- **CUDA Toolkit:** 11.2
- **cuDNN:** 8.1.0
- **NVIDIA Driver:** Must also be compatible with CUDA 11.2 (typically driver version ≥ 460).

The packages provided by TensorFlow do not include PTX for older GPUs. For older or unsupported GPUs, TensorFlow may fail to load unless `CUDA_FORCE_PTX_JIT=1` is disabled.

[1] `https://www.tensorflow.org/install/pip`

2.2 Environment Variables

After installing CUDA and cuDNN, you need to add the following paths to your system's **PATH**:

```
NVIDIA GPU Computing Toolkit\CUDA\v11.2\bin
NVIDIA GPU Computing Toolkit\CUDA\v11.2\libnvvp
```

2.3 Verifying GPU Setup

Run the following command to verify if TensorFlow detects the GPU correctly:

```
python -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

If you see a GPU device listed, then the installation is correct.

3 Python Virtual Environment

- **Python Version:** 3.10 (fully supported by TensorFlow 2.10).
- **Environment Manager:** `venv` (created using `python -m venv .venv`).
- **Code IDE:** Visual Studio Code (VS Code).

To activate the environment before running:

```
.\.venv\Scripts\activate
```

All libraries within this environment remain isolated, guaranteeing compatibility and reproducibility across different setups.

4 Installed Libraries

The project employs a fixed set of libraries with exact version numbers, ensuring compatibility with TensorFlow 2.10 and support for GPU acceleration.

4.1 Key Dependencies

Key dependencies include:

- `tensorflow` version 2.10.0, `tensorflow-gpu` version 2.10.0
- `opencv-python` version 4.11.0.86, `opencv-contrib-python` version 4.11.0.86
- `mediapipe` version 0.10.9, `imutils` version 0.5.4
- `numpy` version 1.26.4, `pandas` version 2.3.0, `matplotlib` version 3.10.3, `seaborn` version 0.13.2, `plotly` version 6.2.0
- `scikit-learn` version 1.7.0, `scipy` version 1.15.3
- `keras` version 2.10.0, `Keras-Preprocessing` version 1.1.2

4.2 Complete Package List

The complete package list, with versions, was generated using `pip freeze` within the virtual environment as shown in Table 1:

Table 1: Full Package List

Package	**Version**
absl-py	2.3.0
asttokens	3.0.0
astunparse	1.6.3
attrs	25.3.0
cachetools	5.5.2
certifi	2025.6.15
cffi	1.17.1
charset-normalizer	3.4.2
colorama	0.4.6
comm	0.2.2
contourpy	1.3.2
cycler	0.12.1
debugpy	1.8.14
decorator	5.2.1
exceptiongroup	1.3.0
executing	2.2.0
fastjsonschema	2.21.1
flatbuffers	25.2.10
fonttools	4.58.4
gast	0.4.0
google-auth	2.40.3
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
graphviz	0.21
grpcio	1.73.0
h5py	3.14.0
idna	3.10
imutils	0.5.4
ipykernel	6.29.5
ipython	8.37.0
jax	0.6.2
jaxlib	0.6.2
jedi	0.19.2
joblib	1.5.1
jsonschema	4.24.0
jsonschema-specifications	2025.4.1
jupyter_client	8.6.3
jupyter_core	5.8.1
keras	2.10.0

Continued on next page

Table 1 – continued from previous page

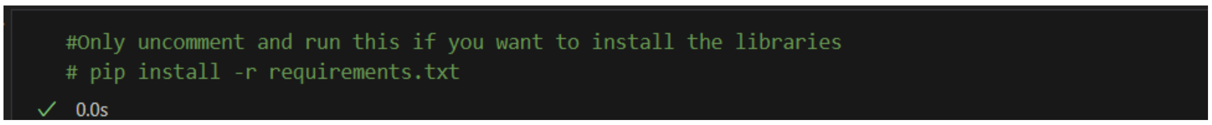
Package	Version
Keras-Preprocessing	1.1.2
kiwisolver	1.4.8
libclang	18.1.1
Markdown	3.8.2
MarkupSafe	3.0.2
matplotlib	3.10.3
matplotlib-inline	0.1.7
mediapipe	0.10.9
ml_dtypes	0.5.1
narwhals	1.44.0
nbformat	5.10.4
nest-asyncio	1.6.0
numpy	1.26.4
oauthlib	3.3.1
opencv-contrib-python	4.11.0.86
opencv-python	4.11.0.86
opt_einsum	3.4.0
packaging	25.0
pandas	2.3.0
pandoc	2.4
parso	0.8.4
pillow	11.2.1
platformdirs	4.3.8
plotly	6.2.0
plumbum	1.9.0
ply	3.11
prompt_toolkit	3.0.51
protobuf	3.20.3
psutil	7.0.0
pure_eval	0.2.3
pyasn1	0.6.1
pyasn1_modules	0.4.2
pycparser	2.22
pydot	4.0.1
Pygments	2.19.2
pyparsing	3.2.3
python-dateutil	2.9.0.post0
pytz	2025.2
pywin32	310
pyzmq	27.0.0
referencing	0.36.2
requests	2.32.4
requests-oauthlib	2.0.0

Continued on next page

Table 1 – continued from previous page

Package	Version
rpds-py	0.25.1
rsa	4.9.1
scikit-learn	1.7.0
scipy	1.15.3
seaborn	0.13.2
sentencepiece	0.2.0
six	1.17.0
sounddevice	0.5.2
split-folders	0.5.1
stack-data	0.6.3
tensorboard	2.10.1
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorflow	2.10.0
tensorflow-estimator	2.10.0
tensorflow-gpu	2.10.0
tensorflow-io-gcs-filesystem	0.31.0
termcolor	3.1.0
threadpoolctl	3.6.0
tornado	6.5.1
tqdm	4.67.1
traitlets	5.14.3
typing_extensions	4.14.0
tzdata	2025.2
urllib3	2.5.0
wcwidth	0.2.13
Werkzeug	3.1.3
wrapt	1.17.2

Only these versions should be installed, as other versions may break compatibility. You can uncomment and run the command as shown in the Figure 1 which is at the start of the code with the *requirements.txt* file to install the specific versions of the libraries.



```
#Only uncomment and run this if you want to install the libraries
# pip install -r requirements.txt
```

Figure 1: Command to install the required libraries

5 Dataset Configuration

The dataset is arranged as seen in Figure 2.

- Training Set: approximately 83,000 images (about 3,100 per class).

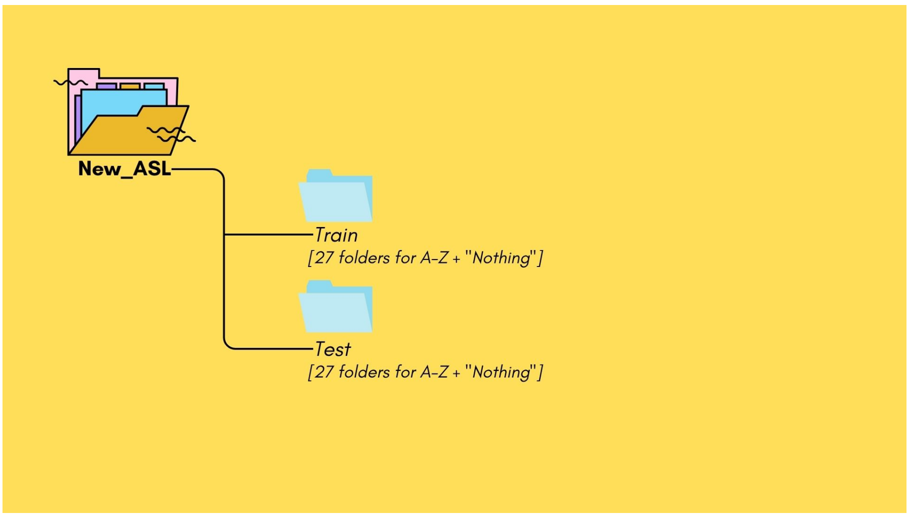

New_ASL
Train
[27 folders for A-Z + "Nothing"]
Test
[27 folders for A-Z + "Nothing"]

Figure 2: Dataset Folders

- Test Set: 162 images (6 per class).
- Images are in colour and originally sized at 200×200 pixels.

Class labels are generated automatically from folder names and are stored in `class_indices.json` to maintain consistent mapping during prediction.

6 Saved Models and Files

Each model trained or fine-tuned is stored as follows:

- Model weights: `.h5`
- Training history (loss/accuracy): `.pkl` (used for plotting curves).
- Class indices: `.json` (ensures that the order of prediction labels matches what was used during training).

Using these files allows you to:

- Skip training the models again by loading the `.h5` model.
- Use the same class indices file to keep labels consistent and avoid mislabelling.
- Visualize training metrics without rerunning training by loading `.pkl` histories.

7 Steps for Running the Code

Important Note:
Do not train the models again unless necessary. For testing or real-time usage, just load the saved models and histories to avoid any kind of errors or hassle during execution.

Step-by-Step Process

1. Import needed Libraries (TensorFlow, Keras, OpenCV, NumPy, etc.).
2. Define the Input Dimensions for each model (like e.g., 224×224 for VGG16, ResNet50, DenseNet201, InceptionV3; 160×160 for MobileNetV2, Custom CNN).
3. Set the file paths for `train_dir` and `test_dir`.
4. Initialize Generators using `ImageDataGenerator` for both train and test data.
   - Augmentations included are rotation, zoom, translation, shear, and horizontal flips.
   - These simulate real-world variations helping the model learn to generalize better.
5. Save Class Indices as `class_indices.json` (preventing mismatched labels later).
6. Create the Test Generator (resized images, no augmentations).
7. Run EDA:
   - Plotted the class distribution using `countplot`.
   - Created a montage to show one sample per class.
8. Load Pretrained Model (VGG16, ResNet50, DenseNet201, InceptionV3, MobileNetV2, or Custom CNN).
   - All convolutional base layers were kept frozen during initial training.
   - Added a custom classification head (Dense + Dropout + Softmax).
9. Set up Callbacks:
   - `EarlyStopping` monitors validation loss and prevents overfitting.
   - `ModelCheckpoint` (update save path as required).
10. Train the Model (set epochs as required, batch size = 16 to prevent OOM errors).
11. Save Training History as `.pkl` file.
12. Reload Saved Model for evaluation or fine-tuning.
13. Fine-Tune Model:
    - Gradually unfreeze the top layers of convolutional blocks.
    - Lower the learning rate (1e-5 or 1e-6).
    - Save the newly trained `.h5` file and the updated `.pkl` history.
14. Plot Accuracy/Loss Curves from the saved histories of the models.
15. Load the Final Model (which is usually fine-tuned version).

16. Generate a Confusion Matrix (use the saved `class_indices.json` to maintain correct label ordering).

17. Evaluate Test Accuracy and Loss.

18. Generate a classification report listing precision, recall, and F1-score.

19. Display Correctly Predicted Samples for inspection.

20. Repeat Steps 8–19 for Each Model (skipping fine-tuning for architectures that do not require it).

21. At the end, conduct a Real-Time Test:

    - Load the selected `.h5` model.
    - Specify the input dimensions.
    - Load `class_indices.json`.
    - Start the webcam feed with a defined ROI for live predictions.

8 Real-Time Testing Instructions

The real-time test used the best model (typically VGG16) to classify signs using a webcam.

Steps to Run:

1. Load the trained model (`.h5`) along with `class_indices.json`.

2. Set the input size for the model (e.g., 224×224 for VGG16).

3. Launch the webcam:

    - A window displays a rectangular Region of Interest (ROI) where signs will be detected as shown in Figure 3.
    - On-screen instructions:
        - P – Confirm recognized sign.
        - B – Backspace (delete last character).
        - Q – Quit the session.

4. Optionally, there is a text history display to keep a record of approved signs if required.

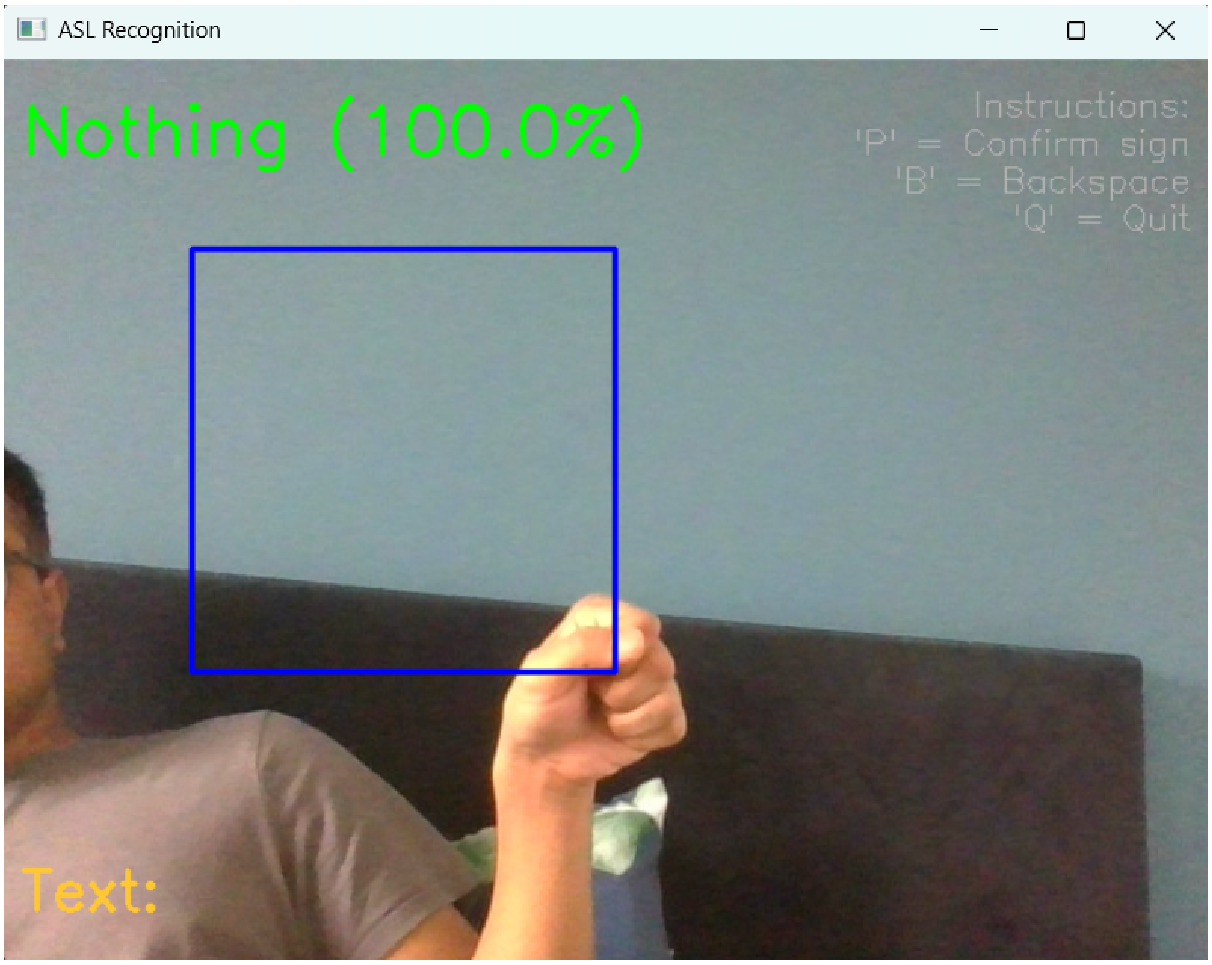


Figure 3: Sample Window of Real-Time Test

9 Additional Notes

- All models and histories can be reused again; no retraining is needed unless you want to change or experiment with the augmentations or architecture are required.
- Keep all libraries at the version listed to prevent TensorFlow from having compatibility and GPU issues.
- Only GPUs with the supported CUDA architecture will run; otherwise, TensorFlow will default to the CPU.
- The batch size is fixed at 16 since it caused memory constraints on the used GPU card RTX 3050.