National College of Ireland

Configuration Manual

MSc Research Project
Artificial Intelligence

Natalia Ellen Nicholas
Student ID: 23351560

School of Computing
National College of Ireland

Supervisor: Ade Fajemisin

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Natalia Ellen Nicholas
Student ID:	23351560
Programme:	Artificial Intelligence
Year:	2025
Module:	MSc Research Project
Supervisor:	Ade Fajemisin
Submission Due Date:	11/08/2025
Project Title:	Configuration Manual
Word Count:	2042
Page Count:	12

I hereby certify that the information contained in this (my submission) is information related 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 back of the project.

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

Signature:	Natalia Ellen Nicholas
Date:	14th 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 in the assignment box located outside the office.

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

Configuration Manual

Natalia Ellen Nicholas
23351560

1 Introduction

This configuration manual outlines the system environment and infrastructure used during the implementation and evaluation phase of the multimodal biometric authentication project. It includes the specifications of the hardware used locally, the cloud environment used for high-computation tasks, and the relevant software and library versions. This manual is intended to support reproducibility and transparency in the setup used for training, validating, and testing multiple Convolutional Neural Network (CNN) models across various biometric traits.

1.1 Hardware

The experimental setup used a local laptop for development/light tests and Kaggle for GPU training:

- **CPU (Local)**: Intel(R) Core(TM) i7-8650U @ 1.90GHz (4 Cores, 8 Threads)
- **GPU (Local)**: Intel(R) UHD Graphics 620 (integrated; no CUDA)
- **Memory (Local)**: 16 GB DDR4 RAM
- **Storage (Local)**: 238 GB SSD (peak usage $\approx$ 228 GB)
- **OS (Local)**: Windows 11 Pro (Version 24H2)
- **External GPU Environment (Kaggle)**:
  - GPU: NVIDIA Tesla T4, 16 GB GDDR6 (sessions may expose 1–2 GPUs)
  - Driver: 560.35.03; CUDA runtime (`nvidia-smi`): 12.6
  - TensorFlow build: CUDA 12.5.1, cuDNN 9
  - OS: Ubuntu 22.04.4 LTS (Jammy)
  - Used for all model training and GPU-accelerated evaluation

1.2 Software

- **Programming Language**: Python 3.11.11 Python Software Foundation (2025)
- **Deep Learning**: TensorFlow 2.18.0 with `tensorflow.keras` (tf.keras 2.18.0) Abadi et al. (2015)

- **Core Libraries**: NumPy 1.26.4 Harris et al. (2020); pandas 2.2.3; scikit-learn 1.2.2 Pedregosa et al. (2011); SciPy 1.15.2; OpenCV 4.11.0.86 Bradski (2000); Matplotlib 3.7.2; librosa 0.11.0; SoundFile 0.13.1; scikit-image 0.25.2; Pillow 11.1.0; tqdm 4.67.1; XGBoost 2.0.3
- **Sustainability**: CodeCarbon 2.3.6 AI (2021)
- **Development Environment**: Jupyter Notebooks on Kaggle (Tesla T4)

Reproducible Environment

To install the curated environment, use the command:

```
pip install -r requirements.txt
```

2 Dataset

This project uses the multimodal biometric LUTBIO dataset Yang (2025), obtained from its Mendeley Data record (Version 6, DOI: 10.17632/jszw485f8j.6). LUTBIO contains biometric samples from 306 participants (ages 8–90; 164 male, 142 female) with nine modalities: voice, face, fingerprint, contact-based palmprint, contactless palmprint, electrocardiogram (ECG), ear, periocular, and opisthenar (back of hand). The collection includes both contact and contactless scenarios and was designed for multimodal and missing-modality research.

Access and licensing. The dataset is available under CC BY 4.0. Access requires completing the provider request form and sending it by email as instructed on the dataset page. We used Version 6 and accessed it on *23 June 2025*. The record indicates more than 11,000 images and ˜1,000 lossless .wav voice files collected in two different phases.

Use in this project. The authors' modality definitions were followed and used the provider's folder structure when preparing train/validation/test splits. Trait-specific preprocessing (e.g., crop/align/resize/CLAHE; ECG 1D and 2D variants) is detailed in *Project Report — Methodology* (Section 3).

3 Models

This section outlines the technical configuration used for trait-wise implementation and training of CNN-based biometric verification models. Each biometric trait is processed using a CNN-based Siamese Hadsell et al. (2006) verification pipeline implemented in TensorFlow. Pre-trained CNN models (e.g., Xception, EfficientNetB3, DenseNet121) are used as base encoders. MobileNetV2 is used for hard negative mining. All models are trained and evaluated on Kaggle using 3-fold cross-validation.

3.1 Dataset Access and Folder Structure

The LUTBIO folder structure was kept exactly as provided (including original folder names).The LUTBIO dataset is structured trait-wise for 306 subjects as shown below.
Note: the dataset uses `plam_touchless` *(sic). It is preserved for reproducibility.*

```
/kaggle/input/dataset-lutbio/LUTBIODataset/LUTBIO/data/
```

```
001/
    ear/
        001_male_56_ear_01.jpg
    ecg/
    face/
    finger/
    opisthenar/
    palm_touch/
    periocular/
    plam_touchless/
    voice/
002/
003/
...
306/
```

- **Data access logic.** During pre-processing, each subject/trait directory for (i) trait availability, (ii) minimum sample count, and (iii) file integrity (e.g., unreadable/corrupt images are skipped) is validated. Paths are built dynamically:

```
DATA_PATH = "/kaggle/input/dataset-lutbio/LUTBIODataset/LUTBIO/data/"
subject_path = os.path.join(DATA_PATH, subject_id, trait_name)
```

- **Naming note.** The dataset's original folder names are kept for reproducibility. The dataset directory is `plam_touchless` (sic). The code accepts `plam_touchless` and maps them to the on-disk folder.

- **Missing-modality policy.** If a subject lacks a certain trait (e.g., ECG missing for $n = 153$ subjects in the dataset), that subject is excluded from training for that specific trait. The trait remains in the system; all other subjects with valid data for that trait are used to train the encoder and generate embeddings/scores. This maximizes usable data without discarding complete subjects from other traits.

- **Subject-disjoint splits and outputs.** 3-fold cross-validation with folds split *by subject* (no subject appears in more than one fold) is performed. During training/evaluation, per-fold artifacts (scores, ROC points, and summary metrics) are written on the current working directory (Kaggle's `/kaggle/working/`). The counts of valid files per trait and the list of missing subjects are *printed to the console* during preprocessing/training; they are *not saved as CSV* in the current version.

3.2 Trait-wise Model Training Details

Note. Traits are implemented across `Notebook1.ipynb`, `Notebook2.ipynb`, and `Notebook3.ipynb`. The pretrained model files may be referenced from `/kaggle/input/pre-trained-cnns/` or `/kaggle/input/pretrained-cnns/`; the code resolves both roots.

3.2.1 Ear Trait

Preprocessing: images resized to (224 × 224); CLAHE on grayscale.
Models trained: EfficientNetB3, DenseNet121, VGG16, ResNet101.
Selected model: EfficientNetB3 Tan and Le (2019).
Model weights:
`/kaggle/input/pre-trained-cnns/models/efficientnetb3_notop.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.2 Face Trait

Preprocessing: images resized to (224 × 224); normalised; no CLAHE.
Models trained: NASNetMobile, InceptionResNetV2, VGG19.
Selected model: NASNetMobile Zoph et al. (2018).
Model weights:
`/kaggle/input/pre-trained-cnns/models/nasnet_mobile_notop.weights.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.3 Periocular Trait

Preprocessing: CLAHE; resize to (224 × 224).
Models trained: VGG16, InceptionV3.
Selected model: InceptionV3 Szegedy, Vanhoucke, Ioffe, Shlens and Wojna (2016).
Model weights:
`/kaggle/input/pre-trained-cnns/models/inception_v3_weights_tf_dim_ordering_tf_kernels.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation. *Pipeline note: For periocular, the feature-level pipeline uses InceptionV3, whereas the score-level pipeline (Section 4.2) uses VGG16 due to better calibrated score behavior.*(Szegedy, Vanhoucke, Ioffe, Shlens and Wojna (2016); Simonyan and Zisserman (2014))

3.2.4 Fingerprint Trait

Preprocessing: grayscale + CLAHE; resize to (224 × 224).
Models trained: EfficientNetB0, NASNetMobile, MobileNetV3Large, ResNet50.
Selected model: EfficientNetB0 Tan and Le (2019).
Model weights: `/kaggle/input/pre-trained-cnns/models/efficientnetb0_notop.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.5 Palm Touchless Trait

Preprocessing: grayscale + CLAHE; resize to (224 × 224).
Models trained: EfficientNetB4, VGG16, VGG19, ResNet50.
Selected model: EfficientNetB4 Tan and Le (2019).
Model weights:
`/kaggle/input/pre-trained-cnns/models/efficientnetb4_notop.h5`

Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.6 Voice Trait

Preprocessing:

- validate `.wav` (duration 2–7 s, non-silent), normalise; copy to `/kaggle/working/fixed_voice_data`
- compute MFCC (128 Mel filters); pad/trim; resize to 224×224; normalise; expand to RGB

Models trained: MobileNetV3Small, InceptionResNetV2.
Selected model: InceptionResNetV2 Szegedy, Ioffe, Vanhoucke and Alemi (2016).
Model weights:
`/kaggle/input/pre-trained-cnns/models/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels_notop.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.7 Opisthenar Trait

Preprocessing: resize to (224×224); normalize to [0,1].
Models trained: VGG16, VGG19, ResNet50, ResNet101, MobileNetV3Large.
Selected model: MobileNetV3Large Howard et al. (2019).
Model weights:
`/kaggle/input/pre-trained-cnns/models/weights_mobilenet_v3_large_224_1.0_float_no_top_v2.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.8 ECG Trait (1D-CNN pipeline)

Preprocessing: resize ECG images to (224×224); CLAHE on channel L (Lab); convert to grayscale; *flatten* to a length-50,176 vector and normalise to [0,1].
Architecture: custom 1D-CNN (Conv1D, GlobalMaxPooling1D, Dense).
Augmentation: additive noise, amplitude scaling, circular shift applied along the vectorised sequence.
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.9 ECG Trait (2D-CNN pipeline)

Preprocessing: resize to (224×224); normalise to [0,1].
Models trained: VGG16, MobileNetV3Small, ResNet50.
Selected model: MobileNetV3Small Howard et al. (2019).
Model weights:
`/kaggle/input/pre-trained-cnns/models/weights_mobilenet_v3_small_224_1.0_float_no_top_v2.h5`
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.2.10 Palm Touch Trait

Preprocessing: CLAHE in Lab space; resize to (224 × 224); normalise to [0,1].
Selected model: DenseNet121 Huang et al. (2017).
Model weights:
`/kaggle/input/pre-trained-cnns/models/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5`
Hard negative mining: MobileNetV2 (pretrained).
Training setup: Siamese network with contrastive loss; 3-fold cross-validation.

3.3 Evaluation Protocol and Metrics

Evaluation was conducted trait-wise using 3-fold cross-validation. For each trait and fold, the trained Siamese model generated feature embeddings and similarity scores for genuine and impostor pairs.

3.3.1 Saved Evaluation Artifacts

For each *selected model* of each trait, the following steps are performed:

Per fold ($X \in \{1, 2, 3\}$).

- **Encoder weights:** `<TRAIT>_<MODEL>_encoder_weights_foldX.weights.h5`
- **Embeddings:** `<TRAIT>_<MODEL>_embeddings_foldX.npy`
- **Pairwise scores:** `<TRAIT>_<MODEL>_scores_foldX.csv`
- **ROC points:** `<TRAIT>_<MODEL>_roc_foldX.csv`

Single, consolidated files (all folds).

- **Evaluation metrics (all traits/models/folds):** `metrics_summary.csv`
- **Energy & emissions (per run):** `emissions.csv`

All files are written to the working directory during training/evaluation:

`/kaggle/working/`

Example (Voice, MobileNetV3Small, fold 2):
`voice_mobilenetv3small_encoder_weights_fold2.weights.h5`,
`voice_mobilenetv3small_embeddings_fold2.npy`,
`voice_mobilenetv3small_scores_fold2.csv`,
`voice_mobilenetv3small_roc_fold2.csv`;
and the consolidated `metrics_summary.csv`, `emissions.csv`.

4 Fusion Techniques

4.1 Feature-Level Fusion Evaluation Configuration

This section defines the logic and parameters for offline simulation and evaluation of feature-level fusion using precomputed trait-wise embedding dictionaries.

4.1.1 Parameters

- `DATA_PATH`: `/kaggle/input/feature-fusion-embeddings/fusion`
- `folds`: [1, 2, 3]
- `noise_std`: 0.05 (Gaussian noise added to embeddings)

4.1.2 Input Files

- Files: `<trait>_<model>_embeddings_foldX.npy`
- Content: Python dict `{subject_id: [embedding1, embedding2, ...]}`
- Loading note: `np.load(file, allow_pickle=True).item()`
- Traits used: `ear`, `ecg`, `face`, `finger`, `opisthenar`, `palmtouch`, `periocular`, `plamtouchless`, `voice`.

4.1.3 Trait-Model Mapping

```
trait_models = {
  "ear": ["EfficientNetB3"],
  "ecg": ["1dconv", "MobileNetV3Small"],
  "face": ["NASNetMobile"],
  "finger": ["EfficientNetB0"],
  "opisthenar": ["MobileNetV3Large"],
  "palmtouch": ["DenseNet"],
  "periocular": ["InceptionV3"],
  "plamtouchless": ["EfficientNetB4"],
  "voice": ["InceptionResNetV2"]
}
```

Periocular uses InceptionV3 in feature-level fusion and VGG16 in score-level fusion; selection is pipeline-specific.

4.1.4 Fusion and Evaluation

- For each subject, sample two embeddings per trait (duplicate if only one); add Gaussian noise.
- Concatenate trait embeddings to form fused representations; compute cosine similarity for genuine/impostor pairs.
- Metrics per fold: AUC, EER, Accuracy, Precision, Recall, F1, FAR, FRR; threshold chosen at EER.
- Evaluate all model combinations (`itertools.product`); average metrics over folds.
- Rank by F1 (desc), then AUC (desc), then EER (asc).

4.2 Score-Level Fusion Evaluation Configuration

Individual per-trait similarity scores are used as features for a learned classifier (XGBoost). (A simple mean baseline exists in a separate cell but is not the main path.)

4.2.1 Input

- Scores (CSV per trait/fold):
  `/kaggle/input/score-again-new/score_again_new/<trait>_<model>_scores_fold<fold>.csv`
- Labels (NumPy per fold):
  `/kaggle/input/fused-labels/fused_labels/fused_labels_fold<fold>.npy`
- Trait→model map (best per trait, as used in code):

```
TRAIT_MODEL_MAP = {
  "plamtouchless": "EfficientNetB4",
  "palmtouch":     "DenseNet",
  "face":          "NASNetMobile",
  "ecg":           "1dconv",
  "voice":         "InceptionResNetV2",
  "opisthenar":    "MobileNetV3Large",
  "finger":        "EfficientNetB0",
  "ear":           "EfficientNetB3",
  "periocular":    "VGG16"
}
```

  Periocular differs across pipelines: VGG16 here (score-level) vs InceptionV3 in feature-level fusion.

4.2.2 Normalization, Aggregation, and Fusion

- **Per fold:** each trait's scores are MinMax-scaled (fit on that fold's scores), optionally multiplied by a raw weight, and stacked as columns.

```
X_all.append(np.vstack(fold_scores).T)  # (n_samples, n_traits)
y_all.append(labels)
```

- **Pooling:** folds 1–3 are concatenated into a single dataset (no additional scaling after pooling).
- **Row alignment:** rows are aligned by construction (same pair order per fold); no explicit merge key is used.
- **Weights:** used as feature scaling; not normalized to sum to 1 in this path.

4.2.3 Classification and Evaluation

- **Resampling:** the minority class is up-sampled on the *pooled dataset before* the train/test split.
- **Split:**
  `train_test_split(..., test_size=0.3, random_state=42, stratify=...)`
- **Classifier:**
  `XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)`
- **Metrics:** AUC, Accuracy; EER from ROC on classifier probabilities:

  ```
  eer = brentq(lambda x: 1. - x - interp1d(fpr, tpr)(x), 0., 1.)
  ```

4.2.4 Traits Included

`face`, `palmtouch`, `plamtouchless` (sic), `ecg`, `voice`, `opisthenar`, `finger`, `ear`, `periocular`.

4.2.5 Output

- Prints one results dict after the pooled train/test evaluation:

  ```
  {"AUC": ..., "Accuracy": ..., "EER": ...}
  ```

- No files are written by default.

5 Trait-Wise Testing and Generalizability Evaluation

This section outlines the experimental setup and results of trait-wise generalization testing. The objective was to validate the performance of the selected CNN-based models on unseen data across three biometric traits: fingerprint, ear, and palm touchless. No fusion was applied in this evaluation — each trait was evaluated independently.

5.1 Experimental Configuration

- **Traits Evaluated:** Fingerprint, Ear, Palm Touchless
- **Encoders Used:**
  - Fingerprint: EfficientNetB0
  - Ear: EfficientNetB3
  - Palm Touchless: EfficientNetB4
- **Folds:** 3-fold evaluation for each trait
- **Image Size:** 224x224
- **Embedding Dimension:** 256 (reduced using PCA to 128 or less based on sample size)

- **Evaluation Metrics:**
  - AUC (Area Under Curve)
  - EER (Equal Error Rate)
  - Accuracy
  - F1-Score
  - Confusion Matrix

5.2 Dataset Sources

- **Fingerprint:** `/kaggle/input/fingerprint-test/Finger_Print_test`
- **Ear:** `/kaggle/input/ear-dataset/subset-1`
- **Palm Touchless:** `/kaggle/input/birjand-university-palm/Birjand University Mobile Palmprint Database (BMPD)`

5.3 Evaluation Pipeline

- Images were preprocessed (CLAHE applied where necessary) and normalized.
- Encoders were loaded with pre-trained fold-specific weights.
- Embeddings were extracted and reduced via PCA.
- Cosine similarity was computed between genuine and imposter pairs.
- Balanced pairing ensured equal representation of genuine and imposter samples.
- Threshold was selected at the best F1-Score using precision-recall curve.

5.4 Summary of Results

Trait-wise testing was conducted independently for each of the three traits across 3 folds. The mean metrics obtained across folds are as follows:

- **Fingerprint:** EfficientNetB0
  - AUC: 0.6367
  - EER: 0.4074
  - Accuracy: 0.6055
  - F1-Score: 0.6742
- **Ear:** EfficientNetB3
  - AUC: 0.6934
  - EER: 0.3708
  - Accuracy: 0.6225
  - F1-Score: 0.6950

- **Palm Touchless:** EfficientNetB4
  - AUC: 0.6312
  - EER: 0.4142
  - Accuracy: 0.5199
  - F1-Score: 0.6676

5.5 Confidence Intervals (Palm Touchless only)

For Palm Touchless, 95% CIs are computed across folds as mean ±1.96×SE. Confidence intervals for Fingerprint and Ear are omitted in this version.

5.6 Remarks

This evaluation demonstrates the generalization ability of the selected models on independent unseen datasets. While these metrics are relatively moderate, they reflect real-world performance scenarios without fusion enhancements. They were further considered in multimodal fusion experiments. No output files were stored. All results were printed to the console.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

AI, E. (2021), 'Codecarbon: Track carbon emissions from ai computing', `https://codecarbon.io/`. Accessed: 2025-08-06.

Bradski, G. (2000), 'The opencv library', *Dr. Dobb's Journal of Software Tools* .

Hadsell, R., Chopra, S. and LeCun, Y. (2006), Dimensionality reduction by learning an invariant mapping, *in* 'Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 2, IEEE, pp. 1735–1742.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J. et al. (2020), 'Array programming with numpy', *Nature* **585**, 357–362.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. et al. (2019), 'Searching for mobilenetv3', *arXiv preprint arXiv:1905.02244* .

Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. (2017), Densely connected convolutional networks, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 4700–4708.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, É. (2011), 'Scikit-learn: Machine learning in python', *Journal of Machine Learning Research* **12**, 2825–2830.

Python Software Foundation (2025), 'Python language reference', `https://www.python.org/`. Accessed: 2025-08-06.

Simonyan, K. and Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. (2016), 'Inception-v4, inception-resnet and the impact of residual connections on learning', *arXiv preprint arXiv:1602.07261* .

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016), Rethinking the inception architecture for computer vision, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 2818–2826.

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

Yang, R. (2025), 'Lutbio dataset (version 6)', Mendeley Data. Accessed: 2025-08-10. **URL:** *https://data.mendeley.com/datasets/jszw485f8j/6*

Zoph, B., Vasudevan, V., Shlens, J. and Le, Q. V. (2018), Learning transferable architectures for scalable image recognition, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 8697–8710.