Faculty of Engineering

INFO 7004 ICT Practicum

Industrial Training Report

Academic Year 2025

Student Name: Ka Ki Li

Student ID: 22069245

Degree Pursue: MICT

Department:
School of Computer, Data and Mathematical Sciences

Organisation's Name: Meta 61

Organisation's Address:
17 Hoy Street, Schofields NSW 2762, Australia

Organisation's Stamp:

Student Declaration and Acknowledgement

I, the undersigned, hereby declare that the training report submitted is the result of my own independent work and that all sources of information and data used have been appropriately acknowledged.

I confirm that I have actively participated in the training program and am satisfied with the knowledge and experience I have gained during this period. I understand that this report reflects my own learning and practical involvement, and it has not been copied or submitted previously for academic credit elsewhere.

By signing this document and submitting this report, I automatically certify the above declaration to be true and valid.

Student Name:	Ka Ki Li
Student ID:	22069245
Training Period:	Mar - May 2025
Date of Submission:	28-05-2025
Student Signature:	Ki

Contents

1 Organization, Team & Individual

1.1 Company Background

META61 is a technology firm that specializes in the convergence of Metaverse, virtual reality (VR), artificial intelligence (AI) and blockchain platforms. Its current research and development agenda focuses on creating immersive, integrated systems that increase customer interaction and strengthen brand presence in digital environments. By adopting Web3 protocols at an early stage and embedding decentralized architectures within its solution portfolio, the company aims to provide clients with scalable and resilient pathways for future growth. The META61 offerings include strategic consulting, software engineering, and system integration services, thus helping partner organizations navigate the rapidly evolving digital ecosystem.

1.2 Individual Responsibilities: Summary of Duties

During the internship I assumed responsibility for the end to end development of two convolutional neural network based systems. For the solar panel cleanliness classification project I designed the network architecture using TensorFlow's Keras functional API, constructing sequential convolutional layers with rectified linear unit activation functions and batch normalization, followed by pooling operations and fully connected layers with dropout regularization. I implemented an image augmentation pipeline using the ImageDataGenerator class to perform rescaling rotations flips brightness adjustments and zoom transformations. I managed the training process with early stopping criteria based on validation loss and conducted detailed performance analysis through confusion matrices and classification reports. To benchmark against classical machine learning approaches I extracted bottleneck features from a pre trained backbone and trained a support vector machine with a radial basis function kernel, optimising hyperparameters via grid search. In the leaf classification project I developed TensorFlow tf.data pipelines to read decode resize and normalise approximately four thousand high resolution images while applying on the fly augmentation. I extended a MobileNetV2 backbone with parallel classification heads for species identification and health assessment, configured a composite loss function to balance categorical and binary objectives and conducted systematic experiments across thirty epochs with continuous metric logging. For additional implementation details and reproducible code, interested readers can refer to the project repository available at `https://github.com/kylekkl/meta61_internship`.

1.3 Application of Engineering Knowledge

The projects integrated principles from signal processing, software engineering, and statistical learning theory to produce robust image-classification pipelines. Controlled perturbations of input images were grounded in signal-processing theory to mitigate over-fitting and improve model generalisation on limited datasets (Shorten and Khoshgoftaar, 2019). Efficient data ingestion and preprocessing were achieved by leveraging TensorFlow's `tf.data` APIs to orchestrate shuffling, mapping, batching, and prefetching—thereby maximising hardware utilisation and minimising I/O bottlenecks (Abadi et al., 2016).

Within the convolutional neural network (CNN) architecture, small learnable kernels (filters) slide across spatial dimensions to exploit local connectivity and weight sharing, capturing low-level edges that deeper layers compose into high-level semantic concepts (LeCun et al., 1998). Interleaved pooling operations—most commonly max-pooling—down-sample feature maps, provide translational invariance, and reduce computational complexity while preserving salient activations (Boureau et al., 2010). Transfer-learning methodologies were applied by adopting pre-trained CNN backbones with classification heads removed and fine-tuning selected layers to adapt learned representations to domain-specific tasks (Krizhevsky et al., 2012); this paradigm was later enhanced with residual connections (He et al., 2016), inception modules (Szegedy et al., 2016), very-deep VGG stacks (Simonyan and Zisserman, 2014), and mobile-friendly depthwise-separable convolutions (Howard et al., 2017).

The multi-task learning framework employed parallel output heads under a weighted-loss scheme to jointly optimise species identification and health-detection objectives, demonstrating an understanding of multi-objective optimisation (Caruana, 1997). Uncertainty-based loss balancing further improved convergence and task harmony (Kendall et al., 2018). Finally, deep-CNN feature extraction was combined with a classical support-vector-machine (SVM) classifier to generate complementary insights into model performance, illustrating proficiency in both contemporary deep-learning techniques and established machine-learning methods (Razavian et al., 2014).

2 Internship Work Experience

2.1 Details of Work Experience

During the seven-week placement, I engaged in a sequence of progressively complex tasks that supported both my understanding of convolutional neural networks (CNNs) and their practical application. In Weeks 1–2, I familiarized myself with the fundamentals of CNN architectures, researched key components (e.g. convolutional blocks, pooling layers), and prepared summary reports for my supervisor. In Weeks 3–4, I sourced suitable image datasets, implemented a baseline CNN model in TensorFlow/Keras, and iteratively refined its training pipeline—experimenting with data□augmentation strategies (flips, rotations, brightness/contrast adjustments) and monitoring loss and accuracy metrics. During Weeks 5–6, I conducted exploratory data analysis (including distribution plots and class□balance assessments), selected hyperparameters via systematic experimentation, and extended training epochs to improve convergence. Finally, in Week 7, I evaluated model performance on unseen data, interpreted confusion matrices to identify error patterns, and compiled a comprehensive project report. Throughout, I collaborated closely with my supervisor, incorporated feedback on code structure and presentation, and adhered to professional standards for documentation and version control.

2.2 Application of Engineering Knowledge

Throughout these projects, fundamental engineering principles were employed to develop robust and efficient image classification systems. Signal processing concepts underpinned the design of the data augmentation pipeline comprising image normalization, random rotations, flips, brightness and zoom adjustments, and spatial shifts to mitigate overfitting and enhance generalization across limited datasets . Scalability and throughput were addressed by leveraging TensorFlow's tf.data API to construct pipelines that perform shuffling, mapping, batching, and prefetching, thereby optimizing GPU utilization and reducing I/O bottlenecks . Transfer learning techniques were integrated by adopting pre trained convolutional backbones (Xception for solar panel classification and MobileNetV2 for leaf multi task learning) with include_top=False and ImageNet weights; strategic freezing of early layers combined with dropout regularization facilitated a balance between leveraging generic feature representations and fine tuning for domain specific tasks.

2.3 Achievements and Results

Over the course of my placement, I successfully designed and delivered two end-to-end convolutional neural network systems: a bespoke SolNet for binary classification of solar-panel imagery and a MobileNetV2-based multi-task CNN for simultaneous species identification and health detection in leaf images. I architected a high-throughput tf.data preprocessing pipeline that performed on-the-fly augmentations—such as flips, rotations, and photometric adjustments—enabling efficient batching of over six thousand images. Through systematic experimentation with learning rates, dropout levels, and augmentation intensities, I achieved stable model convergence within 20–30 epochs for both projects. I leveraged TensorBoard to monitor training dynamics in real time and employed scikit-learn to compute precision, recall, F1 scores, and confusion matrices for rigorous evaluation. These experiences not only enhanced my technical proficiency in deep learning workflows and metric analysis but also strengthened my skills in project planning, collaborative development, and professional technical reporting.

3 Implementation

3.1 System Architecture

The implementation comprises three sequential modules. First, the data ingestion and preprocessing component reads raw images from a class-organized directory structure, decodes and resizes them to the network's input dimensions $227 \times 227 \times 3$ for SolNet and $128 \times 128 \times 3$ for the leaf CNN, and scales pixel values to the 0 to 1 range. Optional on-the-fly augmentations are applied during training to increase dataset diversity. Second, the model training module instantiates the chosen convolutional architectures in TensorFlow/Keras, compiles them with appropriate losses, and conducts optimization with real-time monitoring via TensorBoard. Third, the evaluation module computes performance metrics including accuracy, precision, recall, and F1 score on held-out validation and test splits, ensuring no data leakage and facilitating early stopping based on validation loss.

3.2 Model Architectures

3.2.1 SolNet

SolNet accepts $227 \times 227 \times 3$ inputs and comprises five convolutional blocks with increasing filter counts. Each block consists of a 3×3 convolution, ReLU activation, batch normalization, and 2×2 max pooling. The flattened features pass through two fully connected layers of 4096 units with 0.5 dropout before a sigmoid output node produces the probability of dusty. Training uses the Adam optimizer with binary cross-entropy loss.

3.2.2 Multi-Task Leaf CNN

Based on MobileNetV2, this network features a shared convolutional backbone followed by two task-specific heads. The species head applies global average pooling, a 128 unit dense layer with ReLU and 0.5 dropout, then a 10 way softmax. The health head similarly pools and uses a 64 unit dense layer before a single sigmoid unit. The combined loss is the sum of categorical cross-entropy (species) and binary cross-entropy (health).

3.3 Training and Evaluation

All datasets employ an 0.8:0.1:0.1 split for training, validation, and testing. The solar panel 2562 images are divided at the class level: clean vs. dusty, whereas the leaf dataset is stratified by both species and health status to preserve proportional representation. Augmentation strategies differ slightly: solar panel examples undergo horizontal flips and ±10° rotations; leaf images also include 90° rotations and random brightness/contrast adjustments.

The custom SolNet model attained a validation accuracy of 70.65% and a test accuracy of 71% on the held-out solar panel dataset. Class-wise performance revealed that clean panels were identified with a precision of 0.68 and a recall of 0.94, whereas dusty panels achieved a higher precision of 0.81 but a substantially lower recall of 0.39. On the stratified leaf dataset, the MobileNetV2-based multi-task CNN converged by epoch 30 with training accuracies of approximately 97.3% for species and 92.9% for health status, and corresponding validation accuracies near 96% and 95%. Final test□set performance was 92.92% species□classification accuracy and 92.22% health□detection accuracy, with macro-averaged F1 scores of 0.932 (species) and 0.922 (health).

3.4 Challenges

Despite these strong aggregate metrics, both projects exhibited clear challenges related to class imbalance and inter-class similarity. In the solar-panel task, the overrepresentation of clean images contributed directly to the low recall on the dusty class, suggesting that additional oversampling or cost-sensitive loss functions may be required to recover missed detections. In the leaf dataset, certain species pairs most notably Alstonia scholaris and Jamun remained confounded by the network, and approximately 3% of healthy leaves were misclassified as diseased, highlighting the need for more targeted augmentations or attention mechanisms to capture subtle inter-species and disease-related features

4 Exposure Gained

4.1 Exposure to Engineering, and IT Tools

During the internship I developed expertise in Python-based machine learning frameworks and their associated toolchains through two CNN projects. I employed TensorFlow's Keras functional API to architect, train and evaluate deep convolutional networks for both solar panel cleanliness classification and multi task leaf species/health prediction. To ensure efficient data handling and optimal utilization, I built end to end pipelines with the tf.data API performing on the fly image decoding, resizing, normalization, augmentation and prefetching. For classical benchmarking, I used pre-trained Xception and MobileNetV2 backbones, conducting systematic hyperparameter sweeps to balance class performance. Exploratory data analysis, iterative prototyping and comprehensive performance reporting (including confusion matrices and classification reports) were carried out in Jupyter notebooks, which facilitated rapid experimentation, metric visualization and reproducible documentation of results.

4.2 Exposure to Project Management

I was responsible for planning and executing each phase of the CNN development workflow, from dataset curation through to model delivery. This required defining clear milestones, estimating timelines for data preparation and model training, and tracking progress against objectives.

5 Working Environment

5.1 Health & Safety, Corporate Social Responsibility

Although the internship was fully remote, maintaining a safe and healthy work routine remained essential. From a broader standpoint, remote work reduced commuting emissions and supported sustainable practices by minimising resource consumption. Throughout the placement, I also adhered to responsible data handling policies, ensuring that all materials were managed securely and in accordance with privacy guidelines.

5.2 Professional Ethics and Company Code of Conduct

All collaboration took place via online channels, where clear, respectful communication was vital. I complied with established codes of conduct by safeguarding proprietary information, attributing all external resources appropriately, and responding to queries in a timely manner. Upholding principles of integrity, accountability and mutual respect fostered a positive virtual environment and reinforced trust among team members despite the lack of face-to-face interaction.

6 Conclusions

The internship provided an opportunity to translate theoretical concepts into practical solutions for complex image classification challenges. The design and implementation of two convolutional neural network systems for solar panel cleanliness assessment and for simultaneous leaf species and health prediction enabled thorough exploration of network architecture design, data preprocessing techniques, transfer learning strategies and model evaluation methods. Systematic experimentation and metrics□driven refinement led to models whose performance supports deployment in automated monitoring scenarios.

Engagement with industry standard toolchains such as TensorFlow's Keras functional API, the tf data pipeline and scikit□learn's support vector machine implementation deepened my ability to construct scalable and reproducible machine learning workflows. The application of signal processing principles in data augmentation and the use of a multi objective learning framework for joint optimization of species identification and health detection illustrate the integration of engineering theory and empirical practice.

Overall this practicum has strengthened my capability to oversee machine learning projects from start to finish, fostered critical analysis of model behavior and results, and prepared me to contribute effectively to future research and development efforts in intelligent systems.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 111–118.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

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

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 806–813.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60.

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

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)*, pages 2818–2826.