FIT5221 – Assignment 3

This assignment focuses on practicing advanced model architectures. The objective of this assignment is to get a deeper understanding and hands-on practice of two models: Fully Convolutional Network (FCN) for semantic segmentation and Vision Transformer (ViT) in transfer learning for classification tasks.

Available: 13-May-2025
Submission due: 11.55 PM, 04-June-2025

Instructions:

1. All code should be in Python (>=3.7.x). You should write appropriate comments through the code.
2. Late submission penalty: 5% marks per day or part thereof.
3. Submission is to be made only on Moodle.
4. You must maintain academic integrity. Plagiarism cases will be dealt following the Monash policy.
5. AI & Generative AI tools MUST NOT BE USED within this assessment.

Submission:
You need to submit **a single ZIP file** (size of this file should be less than 500MB), named **A3_<YourMonashID>.zip** (e.g. A3_12345678.zip). The ZIP file should contain:

- Two Jupyter notebooks with answers to the tasks of the assignment and any extra files to complete your assignment. You should name them following the format in the provided notebook files.
- A report file, named **Report_A3_<StudentName_MonashID>.pdf**
- The required checkpoint files.

Task 1: Build a baseline Fully Convolutional Network (FCN) model for semantic segmentation (5 marks)

In this task, you are expected to build a semantic segmentation system with FCN architectures. This system will be trained and evaluated on the PASCAL VOC 2012 dataset. This dataset consists of 20 object classes (+1 for background) and is divided into a training set with 1464 images and a validation set with 1449 images.

We provide the code to load and process this dataset in the notebook file named "**A3_FCN_StudentName_ID.ipynb**".

You are required to build a **baseline FCN model** with the following architecture:

Layer	**Hyperparameter**	**Output shape**	**Param #**
Input	/	(224, 224, 3)	0
EfficientNetB0 as a backbone	/	(7, 7, 1280)	~~4,049,571~~ **4,007,548**
Conv2D	No. Kernels: 21 Kernel size: 1x1 Stride: 1 Padding: "valid" Activation: ReLU	(7, 7, 21)	26,901
TransposeConv2D	No. Kernels: 21 Kernel size: 64x64 Stride: 32 Padding: "same"	(224, 224, 21)	1,806,336
Total number of params			~~5,882,808~~ **5,840,785**

You are required to:

- **Print out the summary** of this model
- **Train and evaluate** this model on the PASCAL VOC 2012 dataset.
- **Plot the accuracy, loss value, and MeanIoU score** of both the training set and the validation set across epochs.
- **Write a report**, in which you should:
    - State the optimizer, loss function, and all hyperparameters used for training.
    - Visualize at least **ten samples** from the **validation set**, and display these images of each sample **side-by-side**:
        1. Input image
        2. Ground truth mask
        3. Predicted mask
    - Comment on prediction quality based on the visualized masks.
    - Discuss training vs. validation performance (e.g., underfitting or overfitting) in terms of loss and metrics.

Note:

- In the notebook, we provide you with the EfficientNetB0 pre-trained weights on the ImageNet dataset. If you want to implement the backbone

EfficientNetB0 from scratch instead of using the library, make sure the number of parameters and output shape match the description above.

- We achieved 0.6 MeanIoU on the training set and 0.49 MeanIoU on the validation set using the above **baseline FCN model** with the pre-trained EfficientNetB0 weights.

- We also provide the function "*calculate_segmentation_metrics*" to calculate various segmentation metrics, including: precision, recall, Intersection over Union (IoU), Dice score, and pixel accuracy. This function calculates the average metrics for all classes, excluding the background class. Refer to this blog for a comparison between IoU and Dice score: Understanding Evaluation Metrics in Medical Image Segmentation. We recommend that you use this function to track the performance of the model during training, or you could implement your own function/metrics.

Task 2: Improve the baseline FCN model (8 marks)

In this task, you are expected to **improve the baseline FCN model that uses an EfficientNet-B0 backbone, as provided in Task 1**, **by incorporating multi-scale features.**

You are required to implement **at least one** multi-scale approach, either based on published research papers or your own design. **You must continue to use the EfficientNet-B0 backbone** in all implementations.

Your model must have **fewer than 10 million parameters**. **Submissions with more than 10 million parameters will receive a penalty.**

Your submission must include:

- **Write a report**, in which you should:
  - Describe each implemented approach and cite any references used.
  - Visualize at least **ten samples** from the **validation set**, and display these images of each sample **side-by-side**:
    1. Input image
    2. Ground truth mask
    3. Predicted mask
  - Comment on prediction quality based on the visualized masks.
  - Discuss training vs. validation performance (e.g., underfitting or overfitting) in terms of loss and metrics.
- The **full code** for each implemented model.
- A **trained checkpoint file** for each model variant.

Mark allocation for this task:

- Implementation and report: **4 marks.**
- MeanIoU on the validation set:

-		MeanIoU	≥	0.50	**4 marks**
-	0.5 >	MeanIoU	≥	0.45	**3 marks**
-	0.45 >	MeanIoU	≥	0.40	**1.5 mark**
-	0.40 >	MeanIoU			**0 marks**

References for Task 2

Hint: It is recommended to read relevant papers for ideas about multi-scale features.

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", MICCAI, 2015.

[3] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[4] T. -Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Task 3: Fine-tuning (a part of) the ViT Model (3 marks)

In this task, you will fine-tune **a pre-trained Vision Transformer (ViT-Small) model** on the Oxford-IIIT Pet dataset. A starter notebook *A3_VIT_StudentName_ID.ipynb* is provided, which includes the model architecture and pre-trained weights. **You must use the provided pre-trained weights to train the ViT-Small model.**

3.1 Report and Implementation Requirements (2 marks):

Given a model, using *torchinfo* or *torchsummary* library, you will see these two numbers:

- **Total parameters**: **N**, which refers to the total number of parameters of the model.
- **Trainable parameters**: **M**, which refers to the number of parameters that will be updated during training.

In subtask 3.1, you are required to explore **two fine-tuning strategies**:

1. **Full fine-tuning**: Fine-tune the entire model. In this case, **M must equal N.**
2. **Partial fine-tuning**: Fine-tune only part of the model (e.g., specific layers or modules). In this case, **M must be less than N.**

For each fine-tuning strategy, you must:

- Print the **Total parameters** and **Trainable parameters**.
- Plot the training and testing accuracies and losses across all training epochs.
- Report the final test accuracy after training.
- Save and submit the checkpoints of your models.

Tip: Use the `requires_grad` attribute to control whether parameters are trainable.

3.2 Best Performance (1 mark):

- You will receive an **additional 0.5 mark** if your best fine-tuned model achieves **test accuracy above 77%**
- You will receive another **additional 0.5 mark** if your best fine-tuned model achieves **test accuracy above 85%.**

Note: Please clean up all verbose training logs before submitting, to ensure your notebook is tidy and easy to follow.

Task 4: Explore and Implement a Parameter-Efficient Transfer Learning (PEFT) Technique (4 marks)

In this task, you will apply a **Parameter-Efficient Fine-tuning (PEFT)** technique to fine-tune **ViT-small** model. PETL methods allow you to fine-tune large models by training **only a small number of additional parameters of the compact auxiliary modules** that plug into the backbone, together with the backbone's **classifier head**, while **freezing all other parameters.**

You are required to:

- Implement **at least one PETL technique that introduces new trainable parameters** into the model. You may choose a method from the literature (e.g., LoRA [5], Adapter [6], AdaptFormer [7], Visual Prompt Tuning[8]) or propose your own.
- You must use the **ViT-Small model and its pre-trained weights from Task 3** as the starting point.
- **You must train the classifier and the additional parameters from the PETL technique. Other parameters from the pre-trained weights must remain frozen.**

Important: Not all PETL techniques introduce additional parameters. However, for this task, you must choose a PETL method that **adds new trainable parameters**.

Implementation and Report (3 marks):

- Clearly explain the PETL method you implemented, your choice of hyperparameters, with proper citations if it is from existing literature.
- Apply the PETL method to the provided ViT-Small model (with Task 3 pre-trained weights).
- Print the **total number of trainable parameters**
- Plot the training and testing accuracies and losses across all training epochs.
- Report the final **test accuracy.**
- Save and submit the checkpoints of your models.

Best Performance (1 mark):

- You will receive **1 bonus mark** if your PETL-based ViT-Small model achieves **higher test accuracy** than your best result from **Task 3**.

References for Task 4

You may want to take a look at these references:

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In Proceedings of the International Conference on Machine Learning (ICML), 2019.

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. Proceedings of the International Conference on Learning Representations (ICLR), 2022.

[7] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. Advances in Neural Information Processing Systems (NeurIPS), 2022.

[8] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In Proceedings of the European Conference on Computer Vision (ECCV), 2022.

– END OF ASSIGNMENT –