

Lab3: Image classification

3099704 AI for Digital Health (2025/2)

Objective

- **Create and train** image classification models with Teachable Machine and the PyTorch library.
- **Tune hyperparameters** of image classification models.
- **Inference & evaluate** performance each model



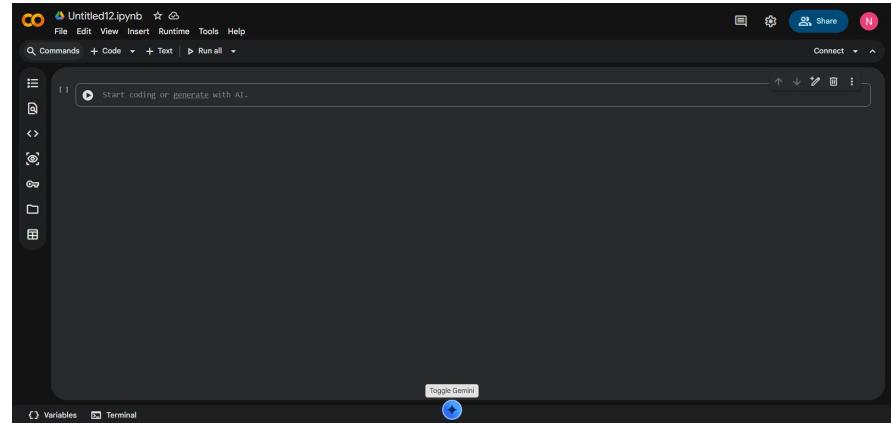
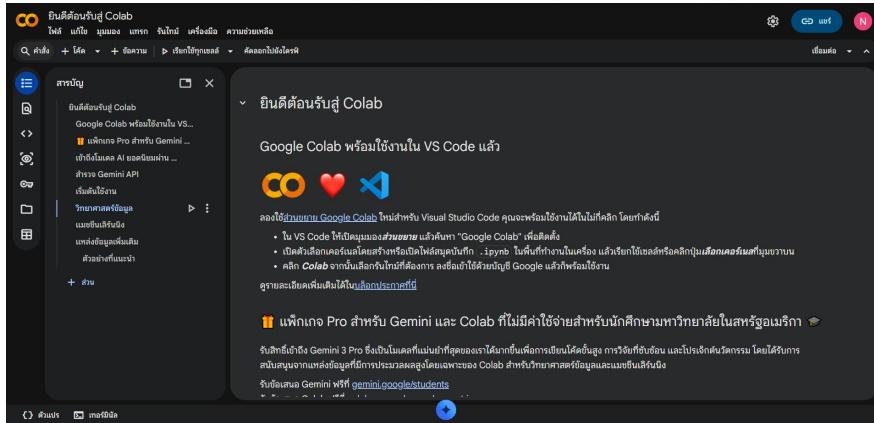
Material

- **Google Colab** is a free cloud-based platform that runs Jupyter notebook (Python code) in a browser, without needing local setup.
- **Teachable Machine** is a web-based tool by Google that trains machine learning models for images, audio, or poses without coding.
- **PyTorch, TensorFlow** is a deep learning library that enables building, training, and deploying neural networks using Python



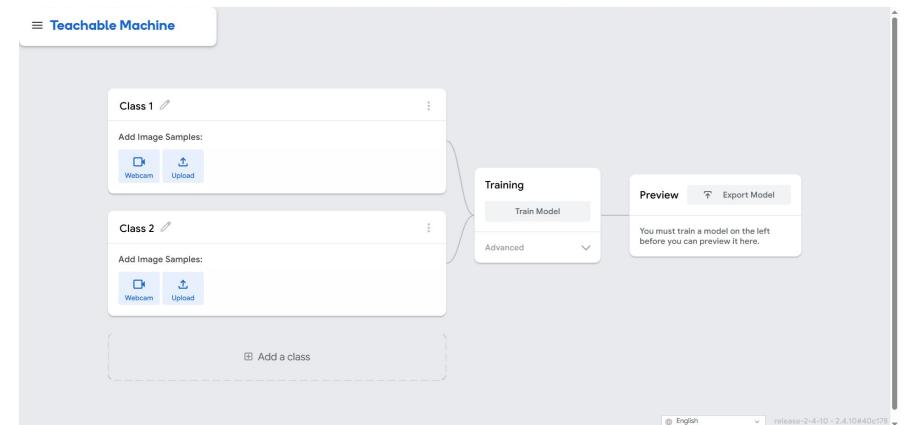
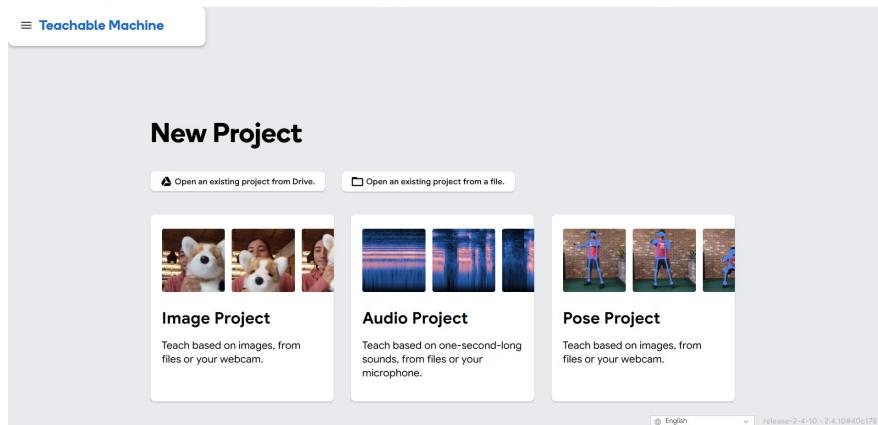
Material

- With **Google Colab**, you don't need to install any software. All you need is a Google account, and you can start using it right away. Simply visit: <https://colab.research.google.com/> or select NEW NOTEBOOK to start a new file.



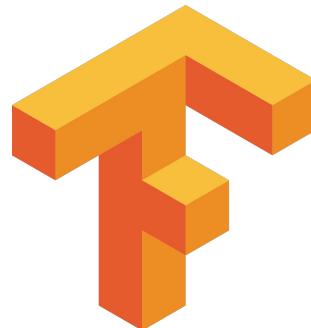
Material

- **Teachable Machine** is a web-based tool from Google that allows you to easily create models without writing code or installing software. Simply access it through your web browser at <https://teachablemachine.withgoogle.com/>



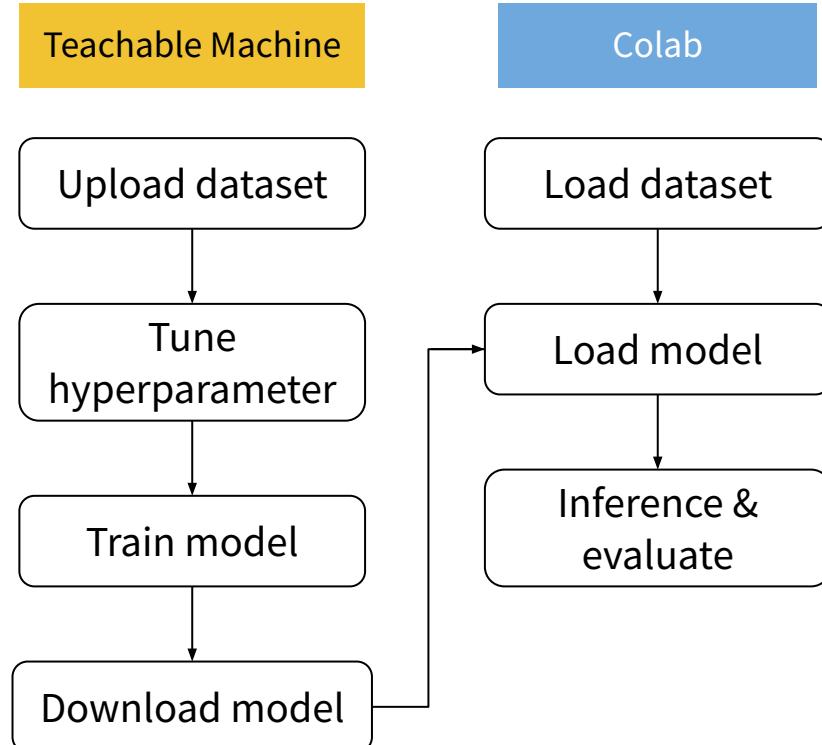
Material

- **TensorFlow and PyTorch** are deep learning libraries used with Python. They can be run either on a local computer or on Google Colab, and can be installed easily using the **pip install** command.



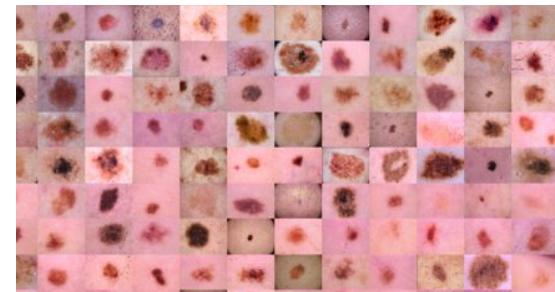
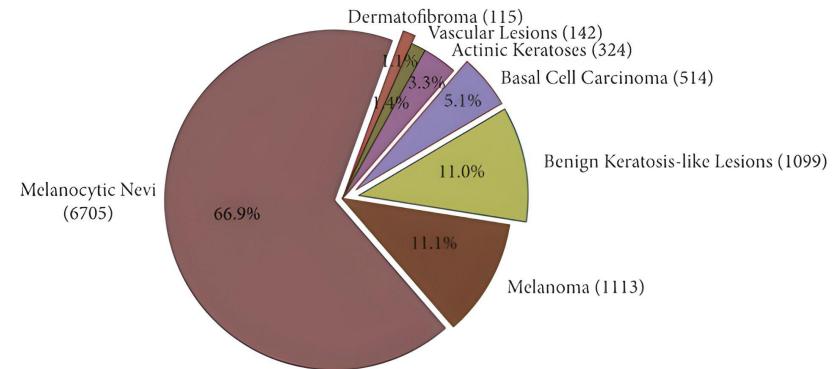
Lab3.1: MobileNet (Teachable Machine)

In this lab, you will create an image classification model using **Teachable Machine** and then evaluate its performance using a **Google Colab notebook**.



Dataset: Skin Cancer MNIST: HAM10000

- The dataset consists of 10015 images with 10013 labeled objects belonging to 7 skin cancer classes.
- The data contains image in JPG format and documents in JSON format
- In the experiment, we reduced the amount of data and formatted it to simplify the experiment.



Lab3.1: MobileNet (Teachable Machine)

- 1) Create image project(standard image model) in Teachable Machine

The image shows two screenshots of the Teachable Machine web application.

Left Screenshot: The main 'New Project' screen. It features three main project categories: 'Image Project', 'Audio Project', and 'Pose Project'. Each category has a thumbnail image, a title, and a brief description. There are also buttons to 'Open an existing project from Drive.' and 'Open an existing project from a file.'

Right Screenshot: A modal window titled 'New Image Project' is displayed over the main screen. This modal contains two options: 'Standard image model' and 'Embedded image model'. The 'Standard image model' box is highlighted with a red border. It includes the following details:

- Best for most uses**
- 224x224px color images
- Export to TensorFlow, TFLite, and TF.js
- Model size: around 5mb

Bottom Navigation: Both screenshots show a navigation bar at the bottom with links for 'Image Project', 'Audio Project', and 'Pose Project', along with language and release version information.

Lab3.1: MobileNet (Teachable Machine)

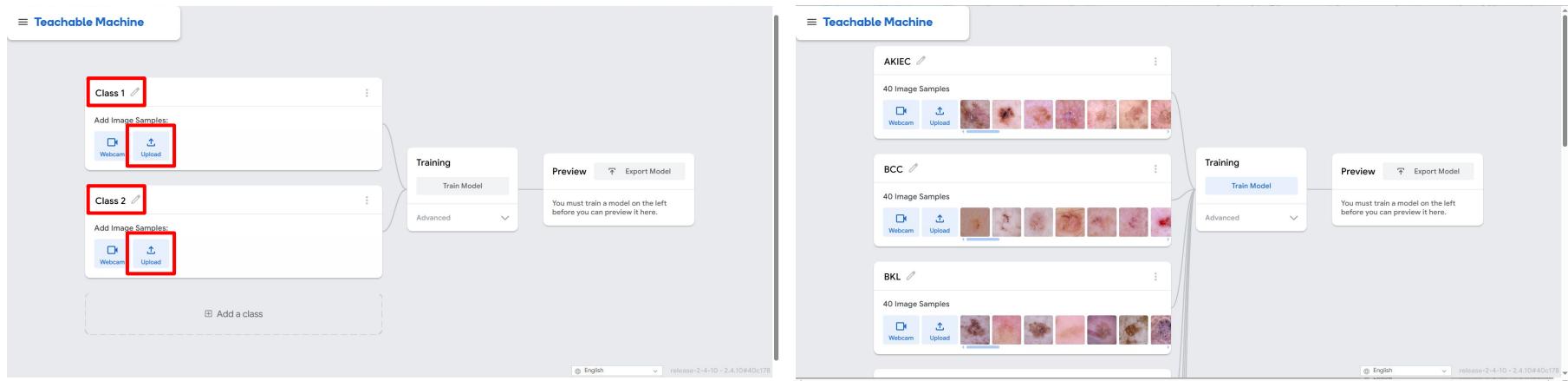
- 2) Download the dataset from [GitHub](#), then upload the image samples and set labels in **Teachable Machine**.

HAM10000_TM directory

```
HAM10000_TM
├── train/
│   ├── AKIEC/
│   │   ├── ISIC_0024329.jpg (original image)
│   │   .
│   └── VASC/
└── ...
```

Lab3.1: MobileNet (Teachable Machine)

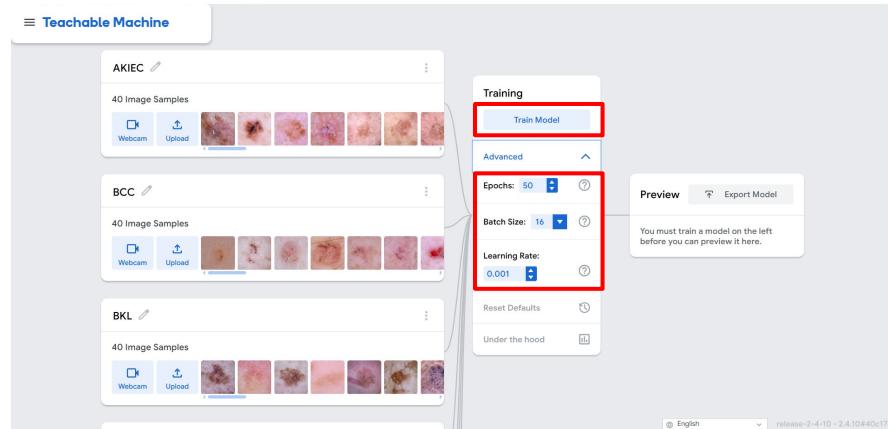
- 2) Download the dataset from [GitHub](#), then upload the image samples and set labels in Teachable Machine.



Lab3.1: MobileNet (Teachable Machine)

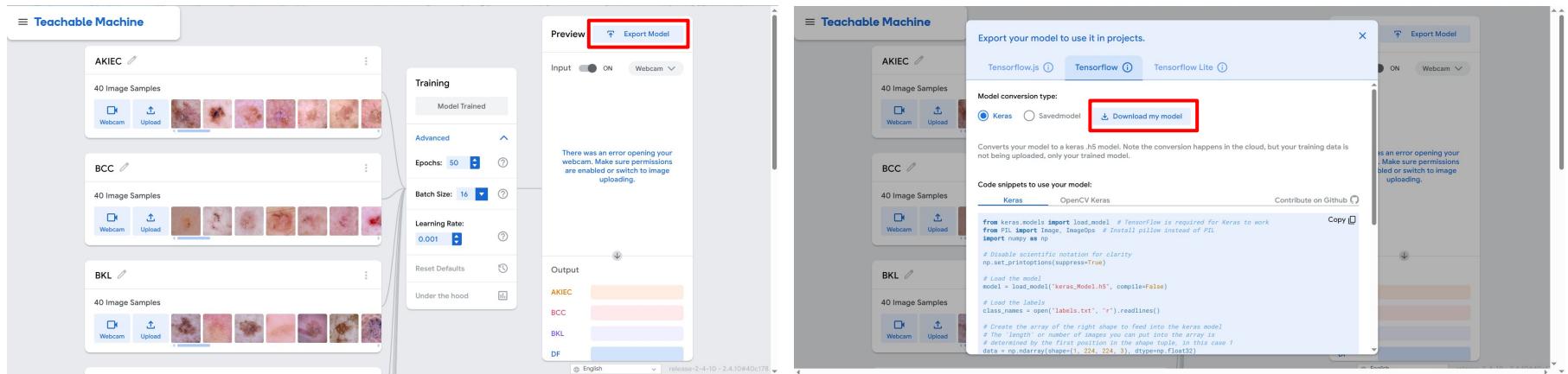
3) Tune learning parameter & Train model

- a) epoch: [5, 20, 50, 100]
- b) learning rate: [0.00001, 0.001, 0.1]



Lab3.1: MobileNet (Teachable Machine)

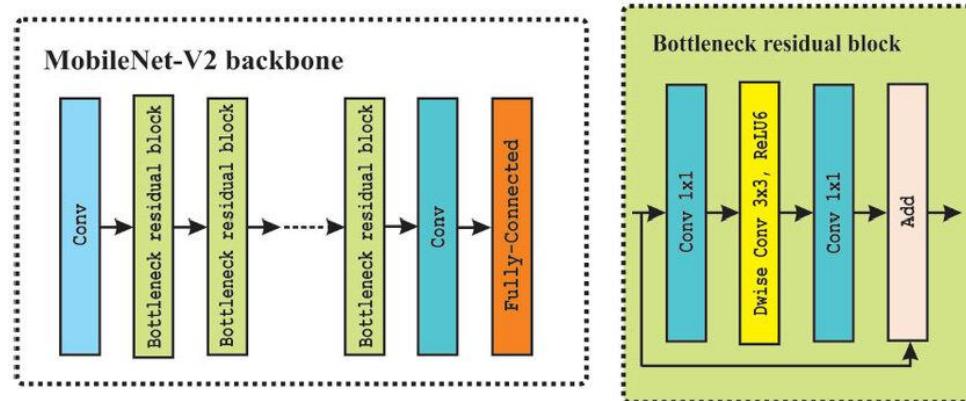
4) Export model (tensorflow version)



Lab3.1: MobileNet (Teachable Machine)

The image model in Teachable Machine is MobileNet V2, with the following details:

- There are 3 layers for bottleneck block.
- This time, the **first layer** is **1×1 convolution with ReLU6**.
- The **second layer** is the **depthwise convolution**.
- The **third layer** is another **1×1 convolution but without any non-linearity**.



Lab3.1: MobileNet (Teachable Machine)

5) Open Lab_3_1_TeachableMachine.ipynb (in **colab**)

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Lab_3_1_TeachableMachine.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help; Share, Connect
- Cell Content:** The first cell contains the following text:

Lab 3.1 – Skin Cancer classification: MobileNet (Teachable Machine)

This notebook is used to evaluate the performance of skin cancer classification models trained on Teachable Machine.

Dataset consisting of 7 classes: Melanoma(MEL), Melanocytic nevi(NV), Basal cell carcinoma(BCC), Actinic keratoses(AKIEC), Benign keratosis lesions(BKL), Dermatofibroma(DF), and Vascular lesions(VASC)

This example code will consist of:

 0. Setup
 1. Load Model
 2. Inference & Evaluate
- Code Cell:** The second cell contains the following Python code:

```
# Download Prepared dataset from github
!wget https://github.com/pvateekul/digitalhealth-ai2025/raw/main/dataset/Lab_3_1_HAM10000.zip
!unzip -q -o "Lab_3_1_HAM10000.zip"

# Download library
!pip install --upgrade tensorflow
```
- Bottom Navigation:** Variables, Terminal

Lab3.1: MobileNet (Teachable Machine)

6) Run [Lab_3_1_TeachableMachine.ipynb](#) for evaluation & inference.

label: MEL
output: BKL
confidence: 0.728



label: NV
output: BKL
confidence: 0.818



label: BCC
output: BCC
confidence: 0.979



label: AKIEC
output: BCC
confidence: 0.798



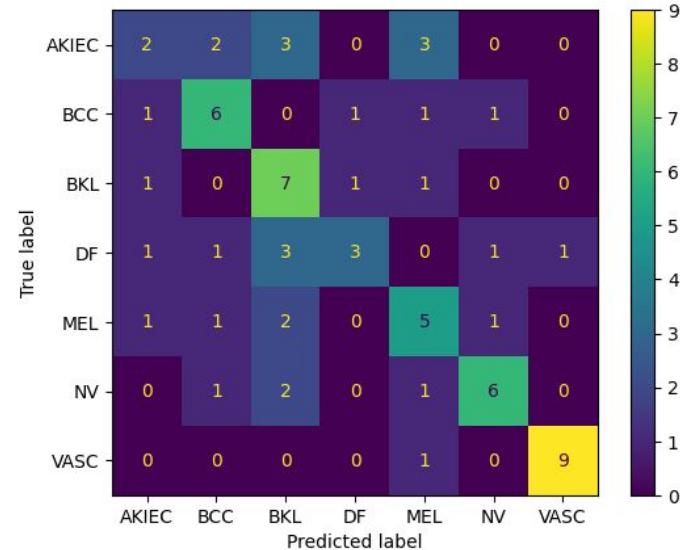
label: BKL
output: DF
confidence: 0.334



label: DF
output: BKL
confidence: 0.487



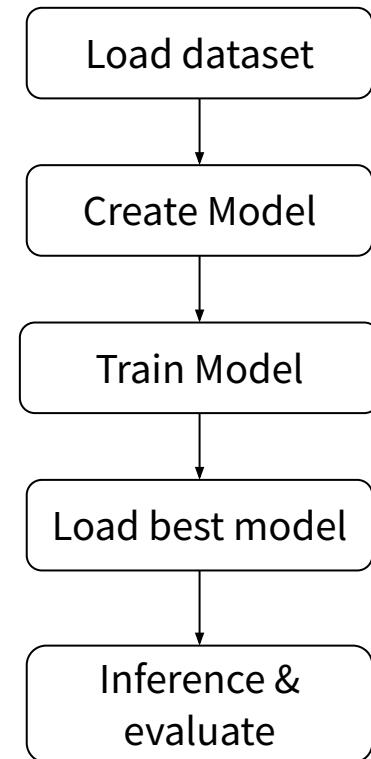
label: VASC
output: VASC
confidence: 1.0



Lab3.2: EfficientNet (Pytorch)

Colab

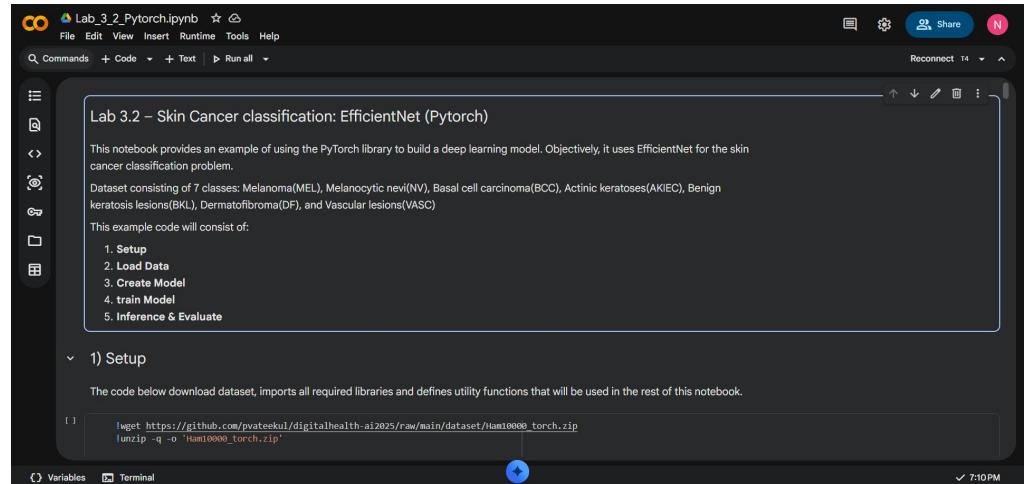
In this lab, you will create an image classification model and then evaluate its performance using **Pytorch library**



Lab3.2: EfficientNet (Pytorch)

Run [Lab_3_2_PyTorch.ipynb](#) (in colab)

- 1) Setup
- 2) Load Data
- 3) Create Model
- 4) train Model
- 5) Inference & Evaluate



Lab 3.2 – Skin Cancer classification: EfficientNet (Pytorch)

This notebook provides an example of using the PyTorch library to build a deep learning model. Objectively, it uses EfficientNet for the skin cancer classification problem.

Dataset consisting of 7 classes: Melanoma(MEL), Melanocytic nevi(NV), Basal cell carcinoma(BCC), Actinic keratoses(AKIEC), Benign keratosis lesions(BKL), Dermatofibroma(DF), and Vascular lesions(VASC)

This example code will consist of:

1. Setup
2. Load Data
3. Create Model
4. train Model
5. Inference & Evaluate

1) Setup

The code below download dataset, imports all required libraries and defines utility functions that will be used in the rest of this notebook.

```
[ ]    !wget https://github.com/pvateekul/digitalhealth-ai2025/raw/main/dataset/Ham10000_torch.zip  
!unzip -q -o 'Ham10000_torch.zip'
```



Lab3.2: EfficientNet (Pytorch)

In this lab, we chose to use EfficientNet V2, which has the following key architectural innovations:

- **Fused-MBConv Blocks:** Replace standard MBConv with a fused 3×3 conv for faster computation without adding many parameters.
- **Smaller Kernels and Expansion Ratios:** The architecture favors smaller 3×3 kernel sizes and smaller expansion ratios within the MBConv blocks, which reduces memory access overhead.
- **Removal of the Final Stage:** The final stride-1 stage of the original EfficientNet is removed to further lower parameter count and memory consumption.

