Chanatip Kowsurat 6530075721

**Notebook #1 Image classification with CNN**

Input: 32x32 RGB images in 10 classes from the CIFAR-10 dataset

Output: Predicted class probabilities or labels for the 10 categories.

Hardware requirement: NVIDIA GPUs

Process

1. Preprocess
   1. Install and import libraries
   2. Load datasets from CIFAR-10
   3. Apply normalization and data augmentation
2. Create CNN model
   1. Pipeline

* First convolutional layer has 3 input and 6 output channels with 5x5 kernel.
* Pooling layer uses a 2x2 kernel with a stride of 2.
* Second convolutional layer has 6 input and 16 output channels with 5x5 kernel.
* First fully connected layer takes 400 input and outputs 120 units.
* Second fully connected layer takes 120 input units and outputs 84 units.
* Third fully connected layer takes 84 input units and outputs 10 units
  1. Summary
* Activation function: ReLU, Softmax
* Loss function: Cross-Entropy Loss
* Optimizer: Adam
* Total params: 62,006

1. Training

* Batch size: 32
* Epoch: 1
* 60,000 images split into: 50,000 training images and 10,000 test images
* Each class contains an equal number of images.

1. Evaluate the model

* Accuracy, Confusion Matrix

Important commands

A screen shot of a computer program

AI-generated content may be incorrect.

Results

A graph with a line

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A line graph with text on it

AI-generated content may be incorrect.

A chart with numbers and labels

AI-generated content may be incorrect.

A blurry image of a boat

AI-generated content may be incorrect.

**Notebook #2 Image classification with EfficientNet**

Input: Image dataset with 10 animal classes  
Output: Predicted class labels

Hardware requirement: NVIDIA GPUs

Process

1. Preprocess
   1. Install and import libraries
   2. Load datasets from GitHub repository
   3. Image augmentation ex. Rotation, Horizontal and Vertical flip
   4. Normalize images
   5. Split data into 1,400 training images, 300 validation images and 300 test images
2. Training Setup

* Model: EficientNetV2
* Loss function: Cross-Entropy function
* Optimizer: Adam
* Learning rate: 0.001
* Batch size: 32
* Epoch: 5
* Trainable parameters: 20.2 M

1. Training
2. Evaluate the model

* Accuracy, Confusion Matrix

Important Commands

A screenshot of a computer program

AI-generated content may be incorrect.

**A graph of loss and loss

AI-generated content may be incorrect.**Results

**A chart of numbers and a number

AI-generated content may be incorrect.A graph with blue and orange lines

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**A blue and black butterfly

AI-generated content may be incorrect.**

**Notebook #3 Object detection with yolov8 (basic)**

Input: Images in 20 classes from the Pascal-VOC dataset.  
Output: Predicted class labels

Hardware requirement: NVIDIA GPUs

Process:

1. Preprocess
   1. Import packages from Ultralytics
   2. Checks software and hardware configuration
2. Train

* Model: YOLOv8
* Images of a size 640x640 with 20 classes
* Epoch: 3
* Learning rate: 0.0004
* Optimizer: Adam
* Parameters: 3 M

1. Evaluate model

Important commands

A screen shot of a computer

AI-generated content may be incorrect.

**A screenshot of a graph

AI-generated content may be incorrect.**Results

**A yellow taxi on a street

AI-generated content may be incorrect.A group of graphs with numbers

AI-generated content may be incorrect.**

**A screenshot of a video

AI-generated content may be incorrect.**

**A collage of images of different types of animals

AI-generated content may be incorrect.**

**Notebook #4 Object detection with yolov8 (advanced)**

Input: Pascal VOC datasets of 20 object classes  
Output: Trained YOLOv8 object detection model with bounding box predictions

Hardware requirement: NVIDIA GPUs

Process:

1. Preprocess
   1. Install and import packages
   2. Download and extract Pascal VOC dataset
   3. Define .yml file structure for YOLO
   4. Convert VOC XML annotations to YOLO format
2. Setup model

* Model: YOLOv8
* Epoch: 3
* Learning rate: 0.0004
* Optimizer: Adam
* Parameters: 3 M
* 10,000 images split into 8,000 training images and 2,000 test images

1. Train
   1. Load YOLOv8 model
   2. Train with dataset and monitor loss
2. Evaluate
   1. Validate performance with mAP
3. Test
   1. Test detection on sample images

Important commands

A screen shot of a computer program

AI-generated content may be incorrect.

Results

A screenshot of a graph

AI-generated content may be incorrect.

A group of graphs with numbers

AI-generated content may be incorrect.**A screenshot of a video

AI-generated content may be incorrect.**

**Notebook #5 Semantic segmentation with DeepLabV3**

Input: Image from CamSeq dataset with a size of 960x720  
Output: Segmentation mask

Hardware requirement: NVIDIA GPUs

Process:

1. Preprocess
   1. Install and import packages
   2. Download 100 image datasets from CamSeq
   3. Load colormap from label\_color.txt
   4. Create Data loader
   5. Apply transformations (resize, normalize).
   6. Split 101 images into 70 training images, 15 validation images and 16 test images
2. Setup model

* Model: DeepLabV3
* Trainable parameters: 42 M
* Epoch: 2
* Training Metrics: Cross-Entropy, accuracy and IoU.

1. Train model
2. Evaluate model

* Compute IoU

1. Visualize results

* Test model with sample image

Important commands

A screenshot of a computer program

AI-generated content may be incorrect.

Results:

A black screen with white text

AI-generated content may be incorrect.A graph of loss and training

AI-generated content may be incorrect.A graph of a line

AI-generated content may be incorrect.A collage of a picture of a street with people walking

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.