Usage Instructions for Code

1. General Overview

This repository implements **MK-CAViT**, a multi-scale Vision Transformer with Fast-HGR correlation attention and gated two-stage fusion.
`MK_CAViT.py` – model definition
(three-path tokenizer, Fast-HGR attention, small↔mid gated fusion + large-context mixing, adaptive head gating, classification head, light segmentation head)
`train_utils.py` – training/evaluation utilities for classification and segmentation
Dataset drivers
`MK_CAViT_ImageNet.py` — ImageNet classification
`MK_CAViT_COCO.py` — COCO multi-label classification (80 classes)
`MK_CAViT_ADE20K.py` — ADE20K semantic segmentation (150 classes)
Each driver trains and evaluates on its dataset.

2. Prerequisites

Python ≥ 3.8
PyTorch ≥ 1.12 (2.x recommended) and torchvision matching your PyTorch build
Pillow (image I/O)
NumPy (ADE20K loader)
pycocotools (COCO only)
CUDA GPU recommended; mixed precision is optional via `--amp`
Install example:
```bash
pip install torch torchvision pillow numpy pycocotools
```

3. Code Map

Fast-HGR logits: `fast_hgr_logits` in `MK_CAViT.py`
(all-pairs cosine + batch trace(cov(Q)·cov(K)) broadcast)
Attention: `FastHGRAttention` (self/cross by passing different `x_q`, `x_kv`)
Tokenizer: `MultiScaleTokenizer` (3×3 / 7×7 stride 2 / 15×15 "same")
Fusion: `MultiScaleFusion` (gated small↔mid cross-mix → large-context mixing with adaptive γ)
Blocks: `Block` (LN → Fast-HGR attn → MLP + DropPath)
Model builders: `mk_cavit_tiny/small/base`
Segmentation path: `forward_seg` (token map → small head → bilinear upsample)

4. Dataset Preparation

ImageNet
```
/path/to/imagenet/
train/class1/...
```

val/class1/...
```

**COCO 2017**

```
/path/to/coco/
train2017/
val2017/
annotations/
instances_train2017.json
instances_val2017.json
```

Use the standard COCO layout above. The script takes `--root` (the folder containing `train2017/`, `val2017/`, `annotations/`) and two JSON paths.

ADE20K

```
/path/to/ADE20K/
images/training/
images/validation/
annotations/training/
annotations/validation/
```

Segmentation head uses 150 classes; `ignore_index=255`.
Input size note: default sizes (224 for classification, 512 for ADE20K) are even to match the small branch's stride-2 grid. If you change `--size`, keep it even.

5. How to Run

ImageNet — single-label classification

```bash
python MK_CAViT_ImageNet.py \
root /path/to/imagenet \
epochs 100 --batch_size 128 --workers 8 \
lr 5e-4 --mu 0.1 --amp
```

`--mu` weights the Fast-HGR term in the combined loss ($L=L_{\text{CE}}-\mu L_{\text{F-HGR}}$).

COCO — multi-label classification (80 classes)

```bash
python MK_CAViT_COCO.py \
root /path/to/coco \
ann_train /path/to/coco/annotations/instances_train2017.json \
ann_val /path/to/coco/annotations/instances_val2017.json \
epochs 50 --batch_size 64 --workers 8 \
```
```

lr 5e-4 --mu 0.1 --amp
```
Targets are 80-D multi-hot vectors; the driver uses `BCEWithLogitsLoss`.

**ADE20K — semantic segmentation (150 classes)**
```bash
python MK_CAViT_ADE20K.py \
root /path/to/ADE20K \
size 512 \
epochs 80 --batch_size 4 --workers 8 \
lr 3e-4 --amp
```
Uses pixel-wise cross-entropy with `ignore_index=255`.
Logs include validation pixel accuracy.

6. Changing Model Size / Classes
```python
from MK_CAViT import mk_cavit_tiny, mk_cavit_small, mk_cavit_base
model = mk_cavit_small(num_classes=K, img_size=224)
```
Pick a builder (`tiny/small/base`) and set `num_classes` to your dataset's class count.
For segmentation, `MK_CAViT_ADE20K.py` builds the model with `num_classes=150` by default.

7. Mixed Precision & Performance Tips
Add `--amp` to enable automatic mixed precision (saves memory, speeds up training).
Increase `--batch_size` until GPU memory is saturated; tune `--workers` for I/O.
COCO: ensure `--root` points to the directory containing `train2017/`, `val2017/`, and both annotation JSONs.
Determinism: set seeds with the built-in utility (drivers already call `set_seed`).

8. Troubleshooting
Shape assertion: "Token count does not match small-branch spatial size" — check that `--size` is even and that your transforms do not alter spatial size unexpectedly.
pycocotools import error: install a wheel matching your Python/CUDA; on Windows use `pip install pycocotools-windows`.
Slow dataloading: increase `--workers`, use SSD storage, and set `pin_memory=True` (already set).
```