## **Sedimentary‐Structures Classification Pipeline**

### **EfficientNet-B2 · ResNet-50 · MobileNet-V3**

### **1. Repository Layout (relative paths)**

text

CopyEdit

train/ # everything produced during training lives here

│

├─ Annotations/ # PASCAL-VOC XML files (raw)

├─ JPEGImages/ # corresponding RGB images (raw)

├─ kfold\_training\_outputs/ # checkpoints + reports + plots

│ ├─ efficientnet\_b2\_f\*.pth

│ ├─ resnet50\_f\*.pth

│ ├─ mobilenet\_v3\_f\*.pth

│ └─ … (CMs, CSV reports, JSON summaries, comparison plots)

└─ regenerated\_cms/ # fresh, hi-res confusion matrices (script #2)

**Tip:** Adjust all absolute base paths once (the BASE\_PATH constant); every script re-uses it.

### **2. Environment**

|  |  |  |
| --- | --- | --- |
| **Dependency** | **Version tested** | **Note** |
| Python | ≥ 3.8 |  |
| PyTorch / torchvision | 2.1 / 0.16 | CUDA automatically detected |
| Albumentations | 2.x | Script warns if 1.x is installed |
| scikit-learn | 1.3 | confusion matrices, metrics |
| pytorch-grad-cam | 1.4 | GradCAM++ during inference |
| Pillow / matplotlib | current | image I/O + plotting |

Install everything at once (CUDA 11.8 build shown here):

bash

CopyEdit

conda create -n sediment\_env python=3.10

conda activate sediment\_env

pip install torch torchvision --index-url https://download.pytorch.org/whl/cu118

pip install albumentations scikit-learn tqdm pillow matplotlib pandas pytorch-grad-cam xmltodict

### **3. Training & Evaluation – multi\_model\_train.py**

#### **What it does**

* Converts VOC XML → image\_labels.csv (cached for future runs)
* **Stratified k = 5** folds with identical random seed (42)
* Class-balance via CrossEntropyLoss(weight= …) + Label smoothing
* Early stopping (PATIENCE = 5) + ReduceLROnPlateau
* Stores **best** weights per fold as <model>\_fold<k>.pth
* Generates per-fold & overall:  
  + confusion matrix image (cm\_…jpg)
  + classification report (report\_…csv)
* Writes a lightweight JSON summary (mean\_ / std\_ for acc & macro-F1)
* Finishes with comparison plots (loss / acc / macro-F1).

#### **How to run**

bash

CopyEdit

python multi\_model\_train.py

# ~60 epochs × 5 folds × 3 CNNs → time depends on GPU

### **4. Confusion-Matrix Rebuild – regenerate\_cm.py**

#### **Why you might need it**

Sometimes you tweak CLASS\_NAMES, improve plotting aesthetics, or simply  
 mis-placed old CM images. This script **re-loads the checkpoints** produced by  
 training and **re-derives the confusion matrices** **without** retraining.

* Uses the **same k-fold split** (StratifiedKFold(seed = 42))
* Saves big (14 × 12 in, 400 DPI) PNG/JPGs into regenerated\_cms/
* Colours: per-fold = *Blues*, overall = *Greens*.

#### **One-liner**

bash

CopyEdit

python regenerate\_cm.py # a few seconds – just forward passes

### **5. Timed Batch Inference – multi\_model\_inference\_timed.py**

#### **Highlights**

* Loads any subset of folds via the FOLD\_CHOICE dict.
* Performs **Grad-CAM++** visualisation for the top-1 class.
* Measures per-image latency **around the forward-pass + CAM** (excludes PIL I/O & plotting).
* Writes:  
  + side-by-side figure: <stem>\_<model>\_fold<k>.jpg
  + CSV: preds\_<model>\_fold<k>.csv with columns  
     filename, prediction, prob, runtime\_ms.
* Prints total & average inference time per model.

#### **Typical run**

bash

CopyEdit

python multi\_model\_inference\_timed.py

Console snippet:

text

CopyEdit

▶ efficientnet\_b2 (fold 5) – 79 images

✓ BSM7.jpg → Lenticular bedding (91.2%) 11.6 ms

...

⏱ efficientnet\_b2: total 0.92 s | avg 11.7 ms / image

### **6. Extending / Customising**

|  |  |
| --- | --- |
| **Task** | **Where to edit** |
| **Filter classes** | EXCLUDED set (all scripts) |
| **Different image size** | Single A.Resize line in each script |
| **Hyper-parameters** (lr, epochs …) | Top of multi\_model\_train.py |
| **Extra backbones** | Add to REG / MODELS & create matching class |
| **CAM layer selection** | cam\_layer() in inference script |

### **7. Troubleshooting Cheatsheet**

|  |  |
| --- | --- |
| **Symptom** | **Likely cause & fix** |
| Missing key(s) / Unexpected key(s) when loading weights | Backbone attribute names differ – ensure the inference / CM-regen builder matches the training builder (features → fc, backbone → classifier …). |
| “element 0 of tensors does not require grad” during CAM | GradCAMPlusPlus **must** receive the original tensor (w/ requires\_grad=True). In the timed script we call logits = model(tensor) **without** disabling grads – that fixes it. |
| Grad-CAM colours washed out | Adjust the gaussian σ or alpha inside show\_cam\_on\_image. |

### **8. Re-use Checklist**

1. **Update BASE\_PATH** – point to *your* data root.
2. Run **training** → verify checkpoints in kfold\_training\_outputs/.
3. Optionally call **regenerate\_cm.py** for high-res matrices.
4. Point **TEST\_DIR** in the timed inference script to unseen images & go!