UHViT: Complete Training Pipeline for IF 5+ Journal

Skin Lesion Classification with Uncertainty Quantification

```
#Install Dependencies (Fixed for RunPod)

# Upgrade all packages to numpy 2.x compatible versions
!pip install -q --upgrade numpy==2.0.2
!pip install -q --upgrade matplotlib>=3.8.0 scipy>=1.12.0 scikit-
learn>=1.4.0
!pip install -q --upgrade timm>=1.0.0
!pip install -q --upgrade wandb>=0.17.0  # Fixed: needs numpy 2.x
compatible version

# Now install the rest
!pip install -q albumentations>=1.3.1
!pip install -q kaggle==1.5.16
!pip install -q grad-cam
!pip install -q pandas>=2.0.3
!pip install -q seaborn>=0.13.0
!pip install -q tqdm>=4.66.1
!pip install -q ptflops>=0.7
!pip install -q gdown
!pip install -q opencv-python-headless

# CELL 1B: EXTRA DEPENDENCY FOR EXTERNAL DATASET
!pip install -q medmnist
print("✓ medmnist installed")

✓ medmnist installed

# CELL 2: IMPORTS
import os, sys, json, random, time, warnings
from pathlib import Path
from collections import Counter
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm.auto import tqdm
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader,
WeightedRandomSampler
from torch.optim import AdamW
```

```
from torch.optim.lr_scheduler import CosineAnnealingWarmRestarts
import timm
from PIL import Image
import albumentations as A
from albumentations.pytorch import ToTensorV2
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, balanced_accuracy_score,
f1_score, confusion_matrix, roc_auc_score,
precision_recall_fscore_support, roc_curve, auc
from sklearn.calibration import calibration_curve
from scipy import stats
import wandb
warnings.filterwarnings('ignore')
print(f"PyTorch: {torch.__version__}, CUDA:
{torch.cuda.is_available()}")
if torch.cuda.is_available(): print(f"GPU:
{torch.cuda.get_device_name(0)}")


PyTorch: 2.8.0+cu128, CUDA: True
GPU: NVIDIA GeForce RTX 5090

# CELL 3: CONFIGURATION
CONFIG = {
    'KAGGLE_USERNAME': 'sushovanchaudhury',  # UPDATE THIS
    'KAGGLE_KEY': '6ad1238ff7f05a1d17687883ac1cb80a',            #
UPDATE THIS
    'WANDB_API_KEY': '07ac9d1305afd5973d9022f150eb134c3b77ca7b',
# UPDATE THIS
    'seed': 42, 'n_folds': 5, 'epochs': 30, 'batch_size': 32,
'img_size': 224,
    'num_workers': 4, 'lr': 1e-4, 'weight_decay': 1e-4,
'mc_dropout_samples': 10,
    'num_classes': 8, 'dropout_rate': 0.3,
    'swin_model': 'swin_tiny_patch4_window7_224',
'efficientnet_model': 'efficientnet_b3',
    'class_names': ['AK', 'BCC', 'BKL', 'DF', 'MEL', 'NV', 'SCC',
'VASC'],
    'data_dir': './data', 'output_dir': './outputs', 'checkpoint_dir':
'./checkpoints',
    'device': 'cuda' if torch.cuda.is_available() else 'cpu'
}
for d in [CONFIG['data_dir'], CONFIG['output_dir'],
CONFIG['checkpoint_dir']]:
    Path(d).mkdir(parents=True, exist_ok=True)
def set_seed(seed):
    random.seed(seed); np.random.seed(seed); torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed);
torch.backends.cudnn.deterministic = True
set_seed(CONFIG['seed'])
```

```
print(f"✓ Config set. Device: {CONFIG['device']}")
```

```
✓ Config set. Device: cuda
```

```
# CELL 4: SETUP APIs
os.makedirs(os.path.expanduser('~/.kaggle'), exist_ok=True)
with open(os.path.expanduser('~/.kaggle/kaggle.json'), 'w') as f:
    json.dump({"username": CONFIG['KAGGLE_USERNAME'], "key":
CONFIG['KAGGLE_KEY']}, f)
os.chmod(os.path.expanduser('~/.kaggle/kaggle.json'), 0o600)
os.environ['WANDB_API_KEY'] = CONFIG['WANDB_API_KEY']
wandb.login(key=CONFIG['WANDB_API_KEY'])
print("✓ APIs configured")
```

```
wandb: WARNING If you're specifying your api key in code, ensure this
code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment
variable, or running `wandb login` from the command line.
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: sushovan-chaudhury (sushovan-chaudhury-
bits-pilani) to https://api.wandb.ai. Use `wandb login --relogin` to
force relogin
```

```
✓ APIs configured
```

```
!kaggle datasets list -s pad-ufes -p 5
```

```
Warning: Looks like you're using an outdated API Version, please
consider updating (server 1.8.3 / client 1.5.16)
No datasets found
```

```
# CELL 5: DOWNLOAD DATASETS  (REPLACE THIS ENTIRE CELL)

from pathlib import Path

isic_path = Path(CONFIG['data_dir']) / 'isic-2019'
if not isic_path.exists():
    !kaggle datasets download -d andrewmvd/isic-2019 -p
{CONFIG['data_dir']}
    !unzip -q {CONFIG['data_dir']}/isic-2019.zip -d
{CONFIG['data_dir']}/isic-2019
    !rm {CONFIG['data_dir']}/isic-2019.zip

ham_path = Path(CONFIG['data_dir']) / 'ham10000'
if not ham_path.exists():
    !kaggle datasets download -d kmader/skin-cancer-mnist-ham10000 -p
{CONFIG['data_dir']}
    !unzip -q {CONFIG['data_dir']}/skin-cancer-mnist-ham10000.zip -d
```

```
{CONFIG['data_dir']}/ham10000
    !rm {CONFIG['data_dir']}/skin-cancer-mnist-ham10000.zip



print("✓ Datasets ready (ISIC2019, HAM10000)")


Downloading isic-2019.zip to ./data
100%|████████████████████████████████████| 9.10G/9.10G
[00:59<00:00, 126MB/s]
100%|████████████████████████████████████| 9.10G/9.10G
[00:59<00:00, 165MB/s]
Downloading skin-cancer-mnist-ham10000.zip to ./data
100%|████████████████████████████████████| 5.18G/5.20G
[00:26<00:00, 233MB/s]
100%|████████████████████████████████████| 5.20G/5.20G
[00:26<00:00, 209MB/s]
✓ Datasets ready (ISIC2019, HAM10000)

# FIX: find correct ISIC2019 CSV + image directory (robust)
from pathlib import Path
import pandas as pd

isic_base = Path(CONFIG["data_dir"]) / "isic-2019"

# 1) Find a CSV that has an 'image' column and class columns
candidate_csvs = list(isic_base.rglob("*.csv"))
best_csv = None
for p in candidate_csvs:
    try:
        df = pd.read_csv(p, nrows=5)
        if "image" in df.columns:
            best_csv = p
            break
    except:
        pass

print("Chosen CSV:", best_csv)
train_df = pd.read_csv(best_csv)

# 2) Find an images directory that contains lots of .jpg files
candidate_dirs = [d for d in isic_base.rglob("*") if d.is_dir()]
def jpg_count(d):
    return len(list(d.glob("*.jpg"))) + len(list(d.glob("*.JPG"))) +
len(list(d.glob("*.jpeg"))) + len(list(d.glob("*.png")))

best_img_dir = max(candidate_dirs, key=jpg_count)
print("Chosen image dir:", best_img_dir, "| image files:",
jpg_count(best_img_dir))
```

```
train_img_dir = best_img_dir
print("train_df columns:", train_df.columns[:15])
print("train_df size:", len(train_df))
```

```
Chosen CSV: data/isic-2019/ISIC_2019_Training_GroundTruth.csv
Chosen image dir:
data/isic-2019/ISIC_2019_Training_Input/ISIC_2019_Training_Input |
image files: 25331
train_df columns: Index(['image', 'MEL', 'NV', 'BCC', 'AK', 'BKL',
'DF', 'VASC', 'SCC', 'UNK'], dtype='object')
train_df size: 25331
```

```
def get_train_transforms(img_size):
    return A.Compose([
        A.RandomResizedCrop(size=(img_size, img_size), scale=(0.8,
1.0)),
        A.HorizontalFlip(p=0.5),
        A.VerticalFlip(p=0.5),
        A.RandomRotate90(p=0.5),
        A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.1,
rotate_limit=45, p=0.5),
        A.OneOf([
            A.GaussNoise(var_limit=(10, 50)),
            A.GaussianBlur(blur_limit=(3, 7))
        ], p=0.3),
        A.OneOf([
            A.CLAHE(clip_limit=4.0),
            A.RandomBrightnessContrast()
        ], p=0.5),
        A.HueSaturationValue(p=0.5),
        A.CoarseDropout(
            num_holes_range=(1, 8),
            hole_height_range=(img_size//16, img_size//8),
            hole_width_range=(img_size//16, img_size//8),
            p=0.3
        ),
        A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225]),
        ToTensorV2()
    ])

def get_val_transforms(img_size):
    return A.Compose([
        A.Resize(height=img_size, width=img_size),
        A.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225]),
        ToTensorV2()
    ])
```

```
import albumentations as A
import albumentations
print("Albumentations version:", albumentations.__version__)

# instantiate to force validation now
t = get_train_transforms(CONFIG["img_size"])
print("Train transforms OK:", t)
```

```
Albumentations version: 2.0.8
Train transforms OK: Compose([
  RandomResizedCrop(p=1.0, area_for_downscale=None, interpolation=1,
mask_interpolation=0, ratio=(0.75, 1.3333333333333333), scale=(0.8,
1.0), size=(224, 224)),
  HorizontalFlip(p=0.5),
  VerticalFlip(p=0.5),
  RandomRotate90(p=0.5),
  ShiftScaleRotate(p=0.5, shift_limit_x=(-0.1, 0.1), shift_limit_y=(-
0.1, 0.1), scale_limit=(-0.09999999999999998, 0.10000000000000009),
rotate_limit=(-45.0, 45.0), interpolation=1, border_mode=0, fill=0.0,
fill_mask=0.0, rotate_method='largest_box', mask_interpolation=0),
  OneOf([
    GaussNoise(p=0.5, mean_range=(0.0, 0.0), noise_scale_factor=1.0,
per_channel=True, std_range=(0.2, 0.44)),
    GaussianBlur(p=0.5, blur_limit=(3, 7), sigma_limit=(0.5, 3.0)),
  ], p=0.3),
  OneOf([
    CLAHE(p=0.5, clip_limit=(1.0, 4.0), tile_grid_size=(8, 8)),
    RandomBrightnessContrast(p=0.5, brightness_by_max=True,
brightness_limit=(-0.2, 0.2), contrast_limit=(-0.2, 0.2),
ensure_safe_range=False),
  ], p=0.5),
  HueSaturationValue(p=0.5, hue_shift_limit=(-20.0, 20.0),
sat_shift_limit=(-30.0, 30.0), val_shift_limit=(-20.0, 20.0)),
  CoarseDropout(p=0.3, fill=0.0, fill_mask=None,
hole_height_range=(14, 28), hole_width_range=(14, 28),
num_holes_range=(1, 8)),
  Normalize(p=1.0, max_pixel_value=255.0, mean=(0.485, 0.456, 0.406),
normalization='standard', std=(0.229, 0.224, 0.225)),
  ToTensorV2(p=1.0, transpose_mask=False),
], p=1.0, bbox_params=None, keypoint_params=None,
additional_targets={}, is_check_shapes=True)
```

```
# CELL 7: DATASET CLASS
class ISIC2019Dataset(Dataset):
    def __init__(self, df, img_dir, transform=None):
        self.df = df.reset_index(drop=True)
        self.img_dir = Path(img_dir)
        self.transform = transform
    def __len__(self): return len(self.df)
    def __getitem__(self, idx):
```

```
        row = self.df.iloc[idx]
        img_name = row['image']
        img_path = None
        for ext in ['.jpg', '.jpeg', '.png', '.JPG']:
            p = self.img_dir / f"{img_name}{ext}"
            if p.exists(): img_path = p; break
        if not img_path:
            for f in self.img_dir.glob(f"{img_name}*"): img_path = f;
break
        image = np.array(Image.open(img_path).convert('RGB'))
        label = 0
        for i, cls in enumerate(CONFIG['class_names']):
            if cls in row and row[cls] == 1.0: label = i; break
        if self.transform: image = self.transform(image=image)
['image']
        return image, label

class HAM10000Dataset(Dataset):
    def __init__(self, csv_path, img_dirs, transform=None):
        self.df = pd.read_csv(csv_path)
        self.img_dirs = [Path(d) for d in img_dirs] if
isinstance(img_dirs, list) else [Path(img_dirs)]
        self.transform = transform
        self.label_map = {'akiec': 0, 'bcc': 1, 'bkl': 2, 'df': 3,
'mel': 4, 'nv': 5, 'vasc': 7}
    def __len__(self): return len(self.df)
    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img_path = None
        for d in self.img_dirs:
            for ext in ['.jpg', '.jpeg', '.png']:
                p = d / f"{row['image_id']}{ext}"
                if p.exists(): img_path = p; break
            if img_path: break
        image = np.array(Image.open(img_path).convert('RGB'))
        label = self.label_map.get(row['dx'], 0)
        if self.transform: image = self.transform(image=image)
['image']
        return image, label
print("✓ Dataset classes defined")
```

```
✓ Dataset classes defined
```

```
# CELL 8: LOAD DATA  (REPLACE THIS ENTIRE CELL)

from pathlib import Path
import pandas as pd

isic_base = Path(CONFIG["data_dir"]) / "isic-2019"
```

```
# 1) Pick the correct LABEL CSV (GroundTruth), not the Metadata CSV
csv_candidates = list(isic_base.rglob("*.csv"))
gt_candidates = [p for p in csv_candidates if ("ground" in
p.name.lower()) or ("truth" in p.name.lower())]

if len(gt_candidates) == 0:
    raise FileNotFoundError(
        "GroundTruth CSV not found. Expected something like
'ISIC_2019_Training_GroundTruth.csv' "
        "inside data/isic-2019/. Please check your unzip output."
    )

# Prefer the one that contains 'Training' if multiple exist
gt_candidates_sorted = sorted(gt_candidates, key=lambda p: ("training"
not in p.name.lower(), len(p.name)))
train_csv = gt_candidates_sorted[0]
train_df = pd.read_csv(train_csv)

print("Using label CSV:", train_csv)
print("train_df size:", len(train_df))

# 2) Find the correct IMAGE directory (folder with most jpg/png)
candidate_dirs = [d for d in isic_base.rglob("*") if d.is_dir()]

def image_count(d: Path) -> int:
    return (
        len(list(d.glob("*.jpg")))
        + len(list(d.glob("*.JPG")))
        + len(list(d.glob("*.jpeg")))
        + len(list(d.glob("*.png")))
    )

train_img_dir = max(candidate_dirs, key=image_count)
print("Using image dir:", train_img_dir, "| image files:",
image_count(train_img_dir))

# 3) Verify the CSV actually contains the class columns
missing_cols = [c for c in CONFIG["class_names"] if c not in
train_df.columns]
if len(missing_cols) > 0:
    raise ValueError(
        f"This CSV does not contain required class columns:
{missing_cols}\n"
        f"You likely loaded the Metadata CSV by mistake.\n"
        f"Loaded columns (first 20): {list(train_df.columns[:20])}\n"
        f"CSV path used: {train_csv}"
    )

print("✓ Class columns found:", [c for c in CONFIG["class_names"] if c
```

```
in train_df.columns])
print("Class distribution (sum of one-hot columns):")
for cls in CONFIG["class_names"]:
    print(f"  {cls}: {int(train_df[cls].sum())}")
```

```
Using label CSV: data/isic-2019/ISIC_2019_Training_GroundTruth.csv
train_df size: 25331
Using image dir:
data/isic-2019/ISIC_2019_Training_Input/ISIC_2019_Training_Input |
image files: 25331
✓ Class columns found: ['AK', 'BCC', 'BKL', 'DF', 'MEL', 'NV', 'SCC',
'VASC']
Class distribution (sum of one-hot columns):
  AK: 867
  BCC: 3323
  BKL: 2624
  DF: 239
  MEL: 4522
  NV: 12875
  SCC: 628
  VASC: 253
```

```
ds = ISIC2019Dataset(train_df.iloc[:200], train_img_dir,
get_train_transforms(CONFIG["img_size"]))
x, y = ds[0]
print("Sample tensor:", x.shape, "label:", y)
print("Unique labels in first 200:", sorted({ds[i][1] for i in
range(200)}))
```

```
Sample tensor: torch.Size([3, 224, 224]) label: 5
Unique labels in first 200: [4, 5]
```

```
# Build labels exactly like your dataset does
labels = []
for i in range(len(train_df)):
    row = train_df.iloc[i]
    found = False
    for j, cls in enumerate(CONFIG["class_names"]):
        if cls in train_df.columns and row[cls] == 1.0:
            labels.append(j)
            found = True
            break
    if not found:
        labels.append(-1)

labels = np.array(labels)
print("Label counts:", np.bincount(labels[labels>=0],
minlength=CONFIG["num_classes"]))
print("Missing labels (-1):", np.sum(labels == -1))
print("Unique labels:", np.unique(labels))
```

```
Label counts: [  867  3323  2624   239  4522 12875   628   253]
Missing labels (-1): 0
Unique labels: [0 1 2 3 4 5 6 7]

# CELL 9: UHViT MODEL
class GatedFusion(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.gate = nn.Sequential(nn.Linear(dim * 2, dim),
nn.Sigmoid())
        self.proj = nn.Linear(dim * 2, dim)
    def forward(self, x1, x2):
        concat = torch.cat([x1, x2], dim=-1)
        return self.proj(concat) + self.gate(concat) * x1 + (1 -
self.gate(concat)) * x2

class MCDropout(nn.Module):
    def __init__(self, p=0.3): super().__init__(); self.p = p
    def forward(self, x): return F.dropout(x, p=self.p, training=True)

class UHViT(nn.Module):
    def __init__(self, num_classes=8, dropout_rate=0.3):
        super().__init__()
        self.swin = timm.create_model(CONFIG['swin_model'],
pretrained=True, num_classes=0)
        self.efficientnet =
timm.create_model(CONFIG['efficientnet_model'], pretrained=True,
num_classes=0)
        self.swin_proj = nn.Linear(self.swin.num_features, 512)
        self.eff_proj = nn.Linear(self.efficientnet.num_features, 512)
        self.fusion = GatedFusion(512)
        self.mc_dropout = MCDropout(dropout_rate)
        self.classifier = nn.Sequential(nn.LayerNorm(512),
nn.Linear(512, 256), nn.GELU(), MCDropout(dropout_rate),
nn.Linear(256, num_classes))
    def forward(self, x):
        swin_feat = self.swin_proj(self.swin(x))
        eff_feat = self.eff_proj(self.efficientnet(x))
        return self.classifier(self.mc_dropout(self.fusion(swin_feat,
eff_feat)))
model = UHViT(); print(f"✓ UHViT: {sum(p.numel() for p in
model.parameters()):,} params"); del model


{"model_id":"427402726956412c9efceb50f45333c4","version_major":2,"vers
ion_minor":0}

{"model_id":"67db1b3300b5429b8e83f8a9584c0eda","version_major":2,"vers
ion_minor":0}
```

```
✓ UHViT: 40,580,266 params

# CELL 10: ABLATION & SOTA MODELS
class SwinOnly(nn.Module):
    def __init__(self, num_classes=8, dropout_rate=0.3):
        super().__init__()
        self.swin = timm.create_model(CONFIG['swin_model'],
pretrained=True, num_classes=0)
        self.classifier =
nn.Sequential(nn.LayerNorm(self.swin.num_features),
nn.Dropout(dropout_rate), nn.Linear(self.swin.num_features,
num_classes))
    def forward(self, x): return self.classifier(self.swin(x))

class EfficientNetOnly(nn.Module):
    def __init__(self, num_classes=8, dropout_rate=0.3):
        super().__init__()
        self.eff = timm.create_model(CONFIG['efficientnet_model'],
pretrained=True, num_classes=0)
        self.classifier =
nn.Sequential(nn.LayerNorm(self.eff.num_features),
nn.Dropout(dropout_rate), nn.Linear(self.eff.num_features,
num_classes))
    def forward(self, x): return self.classifier(self.eff(x))

class UHViTConcatFusion(nn.Module):
    def __init__(self, num_classes=8, dropout_rate=0.3):
        super().__init__()
        self.swin = timm.create_model(CONFIG['swin_model'],
pretrained=True, num_classes=0)
        self.efficientnet =
timm.create_model(CONFIG['efficientnet_model'], pretrained=True,
num_classes=0)
        self.swin_proj = nn.Linear(self.swin.num_features, 512)
        self.eff_proj = nn.Linear(self.efficientnet.num_features, 512)
        self.classifier = nn.Sequential(nn.LayerNorm(1024),
nn.Dropout(dropout_rate), nn.Linear(1024, num_classes))
    def forward(self, x): return
self.classifier(torch.cat([self.swin_proj(self.swin(x)),
self.eff_proj(self.efficientnet(x))], dim=-1))

SOTA_MODELS = ['resnet50', 'efficientnet_b3', 'vit_base_patch16_224',
'swin_tiny_patch4_window7_224', 'convnext_tiny', 'densenet121']
def create_sota_model(name, num_classes=8): return
timm.create_model(name, pretrained=True, num_classes=num_classes)
print("✓ Ablation & SOTA models defined")


✓ Ablation & SOTA models defined
```

```
# CELL 11: LOSS FUNCTIONS
class FocalLoss(nn.Module):
    def __init__(self, alpha=None, gamma=2.0): super().__init__();
self.alpha = alpha; self.gamma = gamma
    def forward(self, inputs, targets):
        ce = F.cross_entropy(inputs, targets, weight=self.alpha,
reduction='none')
        return (((1 - torch.exp(-ce)) ** self.gamma) * ce).mean()

class ClassBalancedLoss(nn.Module):
    def __init__(self, samples_per_class, num_classes, beta=0.9999,
gamma=2.0):
        super().__init__()
        effective_num = 1.0 - np.power(beta, samples_per_class)
        weights = (1.0 - beta) / np.array(effective_num)
        self.weights = torch.tensor(weights / np.sum(weights) *
num_classes, dtype=torch.float32)
        self.gamma = gamma
    def forward(self, inputs, targets):
        self.weights = self.weights.to(inputs.device)
        ce = F.cross_entropy(inputs, targets, weight=self.weights,
reduction='none')
        return (((1 - torch.exp(-ce)) ** self.gamma) * ce).mean()

def get_class_weights(labels):
    counts = np.maximum(np.bincount(labels,
minlength=CONFIG['num_classes']), 1)
    weights = 1.0 / counts
    return torch.tensor(weights / weights.sum() * len(weights),
dtype=torch.float32)

def create_balanced_sampler(labels):
    counts = np.maximum(np.bincount(labels,
minlength=CONFIG['num_classes']), 1)
    sample_weights = (1.0 / counts)[labels]
    return WeightedRandomSampler(sample_weights, len(labels),
replacement=True)
print("✓ Loss functions defined")


✓ Loss functions defined

# CELL 12: TRAINING FUNCTIONS
def train_epoch(model, loader, criterion, optimizer, scheduler,
device, epoch):
    model.train()
    running_loss, all_preds, all_labels = 0.0, [], []
    pbar = tqdm(loader, desc=f'Train E{epoch}')
    for images, labels in pbar:
        images, labels = images.to(device), labels.to(device)
```

```
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(),
max_norm=1.0)
        optimizer.step()
        if scheduler: scheduler.step()
        running_loss += loss.item()
        all_preds.extend(outputs.argmax(dim=1).cpu().numpy())
        all_labels.extend(labels.cpu().numpy())
        pbar.set_postfix({'loss': f'{loss.item():.4f}'})
    return running_loss / len(loader), accuracy_score(all_labels,
all_preds), balanced_accuracy_score(all_labels, all_preds)

def validate(model, loader, criterion, device):
    model.eval()
    running_loss, all_preds, all_labels, all_probs = 0.0, [], [], []
    with torch.no_grad():
        for images, labels in tqdm(loader, desc='Val'):
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            running_loss += criterion(outputs, labels).item()
            all_probs.extend(F.softmax(outputs, dim=1).cpu().numpy())
            all_preds.extend(outputs.argmax(dim=1).cpu().numpy())
            all_labels.extend(labels.cpu().numpy())
    all_preds, all_labels, all_probs = np.array(all_preds),
np.array(all_labels), np.array(all_probs)
    return running_loss / len(loader), accuracy_score(all_labels,
all_preds), balanced_accuracy_score(all_labels, all_preds),
f1_score(all_labels, all_preds, average='macro'), all_preds,
all_labels, all_probs

def validate_with_tta(model, loader, device):
    model.eval()
    all_probs, all_labels = [], []
    with torch.no_grad():
        for images, labels in tqdm(loader, desc='TTA'):
            images = images.to(device)
            probs = torch.stack([F.softmax(model(images), dim=1),
F.softmax(model(torch.flip(images, [3])), dim=1),
F.softmax(model(torch.flip(images, [2])), dim=1)]).mean(0)
            all_probs.extend(probs.cpu().numpy());
all_labels.extend(labels.numpy())
    all_probs, all_labels = np.array(all_probs), np.array(all_labels)
    all_preds = all_probs.argmax(axis=1)
    return accuracy_score(all_labels, all_preds),
balanced_accuracy_score(all_labels, all_preds), f1_score(all_labels,
all_preds, average='macro'), all_preds, all_labels, all_probs
```

```
print("✓ Training functions defined")
```

```
✓ Training functions defined
```

```
# CELL 13: UNCERTAINTY ESTIMATION
def estimate_uncertainty(model, loader, device, n_samples=10):
    model.train()
    all_means, all_uncertainties, all_labels = [], [], []
    with torch.no_grad():
        for images, labels in tqdm(loader, desc='Uncertainty'):
            images = images.to(device)
            outputs = torch.stack([F.softmax(model(images), dim=1) for
_ in range(n_samples)])
            all_means.extend(outputs.mean(0).cpu().numpy())

all_uncertainties.extend(outputs.std(0).mean(1).cpu().numpy())
            all_labels.extend(labels.numpy())
    return np.array(all_means), np.array(all_uncertainties),
np.array(all_labels)

def plot_uncertainty_analysis(uncertainties, predictions, labels,
save_path):
    correct = (predictions == labels)
    fig, axes = plt.subplots(1, 3, figsize=(15, 4))
    axes[0].hist(uncertainties[correct], bins=30, alpha=0.7,
label='Correct', density=True)
    axes[0].hist(uncertainties[~correct], bins=30, alpha=0.7,
label='Incorrect', density=True)
    axes[0].legend(); axes[0].set_title('Uncertainty Distribution')
    thresholds = np.percentile(uncertainties, np.arange(0, 100, 5))
    accs, coverages = [], []
    for t in thresholds:
        mask = uncertainties <= t
        if mask.sum() > 0: accs.append(accuracy_score(labels[mask],
predictions[mask])); coverages.append(mask.mean())
    axes[1].plot(coverages, accs, 'o-');
axes[1].set_xlabel('Coverage'); axes[1].set_ylabel('Accuracy');
axes[1].set_title('Accuracy vs Coverage')
    class_unc = [uncertainties[labels == i].mean() if (labels ==
i).sum() > 0 else 0 for i in range(CONFIG['num_classes'])]
    axes[2].bar(CONFIG['class_names'], class_unc);
axes[2].set_title('Per-Class Uncertainty');
axes[2].tick_params(axis='x', rotation=45)
    plt.tight_layout(); plt.savefig(save_path, dpi=150);
wandb.log({"uncertainty": wandb.Image(save_path)}); plt.close()
print("✓ Uncertainty functions defined")
```

```
✓ Uncertainty functions defined
```

```
# CELL 14: VISUALIZATION FUNCTIONS
def plot_confusion_matrix(y_true, y_pred, class_names, save_path):
    cm = confusion_matrix(y_true, y_pred)
    cm_norm = cm.astype('float') / cm.sum(axis=1, keepdims=True)
    fig, axes = plt.subplots(1, 2, figsize=(14, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names, ax=axes[0])
    axes[0].set_title('Confusion Matrix');
axes[0].set_xlabel('Predicted'); axes[0].set_ylabel('True')
    sns.heatmap(cm_norm, annot=True, fmt='.2f', cmap='Blues',
xticklabels=class_names, yticklabels=class_names, ax=axes[1])
    axes[1].set_title('Normalized'); axes[1].set_xlabel('Predicted');
axes[1].set_ylabel('True')
    plt.tight_layout(); plt.savefig(save_path, dpi=150);
wandb.log({"confusion_matrix": wandb.Image(save_path)}); plt.close()

def plot_roc_curves(y_true, y_probs, class_names, save_path):
    fig, ax = plt.subplots(figsize=(10, 8))
    for i, cls in enumerate(class_names):
        binary = (y_true == i).astype(int)
        if 0 < binary.sum() < len(binary):
            fpr, tpr, _ = roc_curve(binary, y_probs[:, i])
            ax.plot(fpr, tpr, label=f'{cls} (AUC={auc(fpr,
tpr):.3f})')
    ax.plot([0, 1], [0, 1], 'k--'); ax.legend(loc='lower right')
    plt.savefig(save_path, dpi=150); wandb.log({"roc_curves":
wandb.Image(save_path)}); plt.close()

def plot_training_history(history, save_path):
    fig, axes = plt.subplots(2, 2, figsize=(12, 10))
    epochs = range(1, len(history['train_loss']) + 1)
    axes[0, 0].plot(epochs, history['train_loss'], 'b-',
label='Train'); axes[0, 0].plot(epochs, history['val_loss'], 'r-',
label='Val'); axes[0, 0].set_title('Loss'); axes[0, 0].legend()
    axes[0, 1].plot(epochs, history['train_acc'], 'b-',
label='Train'); axes[0, 1].plot(epochs, history['val_acc'], 'r-',
label='Val'); axes[0, 1].set_title('Accuracy'); axes[0, 1].legend()
    axes[1, 0].plot(epochs, history['train_bacc'], 'b-',
label='Train'); axes[1, 0].plot(epochs, history['val_bacc'], 'r-',
label='Val'); axes[1, 0].set_title('Balanced Accuracy'); axes[1,
0].legend()
    axes[1, 1].plot(epochs, history['val_f1'], 'g-'); axes[1,
1].set_title('Macro F1')
    plt.tight_layout(); plt.savefig(save_path, dpi=150);
wandb.log({"training_history": wandb.Image(save_path)}); plt.close()
print("✓ Visualization functions defined")


✓ Visualization functions defined
```

```
# CELL 15: CLINICAL & CALIBRATION ANALYSIS
def compute_ece(probs, labels, n_bins=15):
    bin_boundaries = np.linspace(0, 1, n_bins + 1)
    ece = 0.0
    for i in range(n_bins):
        mask = (probs > bin_boundaries[i]) & (probs <=
bin_boundaries[i + 1])
        if mask.sum() > 0: ece += mask.sum() * abs(probs[mask].mean()
- labels[mask].mean())
    return ece / len(probs)

def per_class_analysis(y_true, y_pred, y_probs, class_names):
    precision, recall, f1, support =
precision_recall_fscore_support(y_true, y_pred, average=None,
zero_division=0)
    auc_scores = []
    for i in range(len(class_names)):
        binary = (y_true == i).astype(int)
        try: auc_scores.append(roc_auc_score(binary, y_probs[:, i]) if
0 < binary.sum() < len(binary) else 0)
        except: auc_scores.append(0)
    df = pd.DataFrame({'Class': class_names, 'Precision': precision,
'Recall': recall, 'F1': f1, 'AUC': auc_scores, 'Support': support})
    wandb.log({"per_class_metrics": wandb.Table(dataframe=df)})
    return df

def plot_per_class_performance(df, save_path):
    fig, ax = plt.subplots(figsize=(14, 6))
    x = np.arange(len(df)); width = 0.2
    ax.bar(x - 1.5*width, df['Precision'], width, label='Precision');
ax.bar(x - 0.5*width, df['Recall'], width, label='Recall')
    ax.bar(x + 0.5*width, df['F1'], width, label='F1'); ax.bar(x +
1.5*width, df['AUC'], width, label='AUC')
    ax.set_xticks(x); ax.set_xticklabels(df['Class'], rotation=45);
ax.legend(); ax.set_ylim(0, 1)
    plt.tight_layout(); plt.savefig(save_path, dpi=150);
wandb.log({"per_class_chart": wandb.Image(save_path)}); plt.close()
print("✓ Clinical analysis functions defined")
```

```
✓ Clinical analysis functions defined
```

```
# CELL 16: GRAD-CAM++
from pytorch_grad_cam import GradCAMPlusPlus
from pytorch_grad_cam.utils.image import show_cam_on_image
from pytorch_grad_cam.utils.model_targets import
ClassifierOutputTarget

def get_gradcam_visualizations(model, loader, device,
save_path='gradcam.png'):
```

```
    model.eval()
    try: target_layer = [model.efficientnet.conv_head]
    except: target_layer = [list(model.efficientnet.children())[-2]]
    cam = GradCAMPlusPlus(model=model, target_layers=target_layer)
    fig, axes = plt.subplots(2, 4, figsize=(16, 8)); axes =
axes.flatten()
    images, labels = next(iter(loader)); images =
images[:8].to(device); labels = labels[:8]
    for i in range(min(8, len(images))):
        input_tensor = images[i:i + 1]
        with torch.no_grad(): pred_class =
model(input_tensor).argmax(dim=1).item()
        grayscale_cam = cam(input_tensor=input_tensor,
targets=[ClassifierOutputTarget(pred_class)])[0, :]
        img = images[i].cpu().numpy().transpose(1, 2, 0)
        img = np.clip(img * np.array([0.229, 0.224, 0.225]) +
np.array([0.485, 0.456, 0.406]), 0, 1)
        vis = show_cam_on_image(img, grayscale_cam, use_rgb=True)
        axes[i].imshow(vis); axes[i].set_title(f'T:
{CONFIG["class_names"][labels[i]]} P:{CONFIG["class_names"]
[pred_class]}'); axes[i].axis('off')
    plt.tight_layout(); plt.savefig(save_path, dpi=150);
wandb.log({"gradcam": wandb.Image(save_path)}); plt.close(); del cam
print("✓ Grad-CAM++ defined")
```

```
✓ Grad-CAM++ defined
```

```
# CELL 17: STATISTICAL TESTS
def mcnemar_test(y_true, y_pred1, y_pred2):
    c1, c2 = (y_pred1 == y_true), (y_pred2 == y_true)
    b, c = np.sum(c1 & ~c2), np.sum(~c1 & c2)
    if b + c > 0: stat = (abs(b - c) - 1) ** 2 / (b + c); p = 1 -
stats.chi2.cdf(stat, df=1)
    else: stat, p = 0, 1.0
    return stat, p

def paired_t_test(scores1, scores2): return stats.ttest_rel(scores1,
scores2)
def wilcoxon_test(scores1, scores2):
    try: return stats.wilcoxon(scores1, scores2)
    except: return 0, 1.0

def compute_ci(scores, confidence=0.95):
    n, mean, se = len(scores), np.mean(scores), stats.sem(scores)
    h = se * stats.t.ppf((1 + confidence) / 2, n - 1)
    return mean, mean - h, mean + h
print("✓ Statistical tests defined")
```

```
✓ Statistical tests defined
```

```
# CELL 18: COMPUTATIONAL ANALYSIS
from ptflops import get_model_complexity_info

def compute_model_complexity(model, input_size=(3, 224, 224)):
    try: macs, params = get_model_complexity_info(model, input_size,
as_strings=False, print_per_layer_stat=False, verbose=False); return
macs * 2, params
    except: return 0, sum(p.numel() for p in model.parameters())

def measure_inference_time(model, device, n_runs=100):
    model.eval(); dummy = torch.randn(1, 3, 224, 224).to(device)
    with torch.no_grad():
        for _ in range(10): _ = model(dummy)
    if device == 'cuda': torch.cuda.synchronize()
    times = []
    with torch.no_grad():
        for _ in range(n_runs):
            start = time.time(); _ = model(dummy)
            if device == 'cuda': torch.cuda.synchronize()
            times.append(time.time() - start)
    return np.mean(times) * 1000, np.std(times) * 1000

def computational_analysis(models_dict, device):
    results = []
    for name, model_fn in models_dict.items():
        print(f"Analyzing {name}..."); model = model_fn().to(device)
        flops, params = compute_model_complexity(model); mean_time,
std_time = measure_inference_time(model, device)
        results.append({'Model': name, 'Params (M)': params / 1e6,
'FLOPs (G)': flops / 1e9, 'Time (ms)': mean_time})
        del model; torch.cuda.empty_cache()
    df = pd.DataFrame(results); wandb.log({"computational":
wandb.Table(dataframe=df)}); return df
print("✓ Computational analysis defined")
```

```
✓ Computational analysis defined
```

```
# CELL 19: MAIN TRAINING LOOP
def train_fold(fold, train_df, val_df, train_img_dir, device):
    print(f"\n{'='*50}\nFOLD {fold+1}/{CONFIG['n_folds']}\n{'='*50}")
    train_dataset = ISIC2019Dataset(train_df, train_img_dir,
get_train_transforms(CONFIG['img_size']))
    val_dataset = ISIC2019Dataset(val_df, train_img_dir,
get_val_transforms(CONFIG['img_size']))
    train_labels = []
    for i in range(len(train_df)):
        row = train_df.iloc[i]
```

```
        for j, cls in enumerate(CONFIG['class_names']):
            if cls in row and row[cls] == 1.0: train_labels.append(j);
break
        else: train_labels.append(0)
    train_labels = np.array(train_labels)
    sampler = create_balanced_sampler(train_labels)
    train_loader = DataLoader(train_dataset,
batch_size=CONFIG['batch_size'], sampler=sampler,
num_workers=CONFIG['num_workers'], pin_memory=True)
    val_loader = DataLoader(val_dataset,
batch_size=CONFIG['batch_size'], shuffle=False,
num_workers=CONFIG['num_workers'], pin_memory=True)
    model = UHViT(num_classes=CONFIG['num_classes'],
dropout_rate=CONFIG['dropout_rate']).to(device)
    samples_per_class = np.maximum(np.bincount(train_labels,
minlength=CONFIG['num_classes']), 1)
    criterion = ClassBalancedLoss(samples_per_class,
CONFIG['num_classes'])
    optimizer = AdamW(model.parameters(), lr=CONFIG['lr'],
weight_decay=CONFIG['weight_decay'])
    scheduler = CosineAnnealingWarmRestarts(optimizer, T_0=10,
T_mult=2)
    history = {'train_loss': [], 'train_acc': [], 'train_bacc': [],
'val_loss': [], 'val_acc': [], 'val_bacc': [], 'val_f1': []}
    best_bacc, best_state = 0, None
    for epoch in range(CONFIG['epochs']):
        train_loss, train_acc, train_bacc = train_epoch(model,
train_loader, criterion, optimizer, scheduler, device, epoch + 1)
        val_loss, val_acc, val_bacc, val_f1, val_preds, val_labels,
val_probs = validate(model, val_loader, criterion, device)
        wandb.log({f'fold{fold}/train_loss': train_loss,
f'fold{fold}/train_bacc': train_bacc, f'fold{fold}/val_loss':
val_loss, f'fold{fold}/val_bacc': val_bacc, f'fold{fold}/val_f1':
val_f1})
        history['train_loss'].append(train_loss);
history['train_acc'].append(train_acc);
history['train_bacc'].append(train_bacc)
        history['val_loss'].append(val_loss);
history['val_acc'].append(val_acc);
history['val_bacc'].append(val_bacc); history['val_f1'].append(val_f1)
        print(f"E{epoch+1}: TrBAcc={train_bacc:.4f},
ValBAcc={val_bacc:.4f}, ValF1={val_f1:.4f}")
        if val_bacc > best_bacc: best_bacc = val_bacc; best_state =
model.state_dict().copy()
    model.load_state_dict(best_state); torch.save(best_state,
f"{CONFIG['checkpoint_dir']}/fold{fold}_best.pt")
    tta_acc, tta_bacc, tta_f1, tta_preds, tta_labels, tta_probs =
validate_with_tta(model, val_loader, device)
    print(f"TTA Results: Acc={tta_acc:.4f}, BAcc={tta_bacc:.4f},
```

```
F1={tta_f1:.4f}")
    plot_confusion_matrix(tta_labels, tta_preds,
CONFIG['class_names'], f"{CONFIG['output_dir']}/fold{fold}_cm.png")
    plot_roc_curves(tta_labels, tta_probs, CONFIG['class_names'],
f"{CONFIG['output_dir']}/fold{fold}_roc.png")
    plot_training_history(history,
f"{CONFIG['output_dir']}/fold{fold}_history.png")
    mean_probs, uncertainties, unc_labels =
estimate_uncertainty(model, val_loader, device)
    plot_uncertainty_analysis(uncertainties,
mean_probs.argmax(axis=1), unc_labels,
f"{CONFIG['output_dir']}/fold{fold}_uncertainty.png")
    get_gradcam_visualizations(model, val_loader, device,
f"{CONFIG['output_dir']}/fold{fold}_gradcam.png")
    pc_df = per_class_analysis(tta_labels, tta_preds, tta_probs,
CONFIG['class_names'])
    plot_per_class_performance(pc_df,
f"{CONFIG['output_dir']}/fold{fold}_perclass.png")
    return {'acc': tta_acc, 'bacc': tta_bacc, 'f1': tta_f1, 'preds':
tta_preds, 'labels': tta_labels, 'probs': tta_probs, 'model': model}
print("✓ Main training function defined")
```

```
✓ Main training function defined
```

```
# Sanity check: confirm first 50 images exist
missing = 0
for i in range(min(50, len(train_df))):
    img_name = str(train_df.iloc[i]["image"])
    found = False
    for ext in [".jpg", ".jpeg", ".png", ".JPG"]:
        if (Path(train_img_dir) / f"{img_name}{ext}").exists():
            found = True
            break
    if not found:
        # try glob
        if not list(Path(train_img_dir).glob(f"{img_name}*")):
            missing += 1
            if missing <= 5:
                print("Missing example:", img_name)

print("Missing in first 50:", missing)
assert missing == 0, "Image paths are wrong — fix train_img_dir / CSV
selection."
```

```
Missing in first 50: 0
```

```
# CELL 20: RUN 5-FOLD CROSS-VALIDATION
wandb.init(project='UHViT-SkinLesion-IF5', name='5fold-cv',
config=CONFIG)
```

```
all_labels = []
for i in range(len(train_df)):
    row = train_df.iloc[i]
    for j, cls in enumerate(CONFIG['class_names']):
        if cls in row and row[cls] == 1.0: all_labels.append(j); break
    else: all_labels.append(0)
all_labels = np.array(all_labels)
skf = StratifiedKFold(n_splits=CONFIG['n_folds'], shuffle=True,
random_state=CONFIG['seed'])
fold_results = []
for fold, (train_idx, val_idx) in enumerate(skf.split(train_df,
all_labels)):
    result = train_fold(fold, train_df.iloc[train_idx],
train_df.iloc[val_idx], train_img_dir, CONFIG['device'])
    fold_results.append(result)
    print(f"\nFold {fold+1} Complete: BAcc={result['bacc']:.4f},
F1={result['f1']:.4f}")
accs, baccs, f1s = [r['acc'] for r in fold_results], [r['bacc'] for r
in fold_results], [r['f1'] for r in fold_results]
print(f"\n{'='*50}\n5-FOLD CV RESULTS\n{'='*50}")
print(f"Accuracy: {np.mean(accs):.4f} ± {np.std(accs):.4f}")
print(f"Balanced Accuracy: {np.mean(baccs):.4f} ±
{np.std(baccs):.4f}")
print(f"Macro F1: {np.mean(f1s):.4f} ± {np.std(f1s):.4f}")
acc_mean, acc_lo, acc_hi = compute_ci(accs); bacc_mean, bacc_lo,
bacc_hi = compute_ci(baccs); f1_mean, f1_lo, f1_hi = compute_ci(f1s)
print(f"\n95% CI - Accuracy: [{acc_lo:.4f}, {acc_hi:.4f}]")
print(f"95% CI - Balanced Acc: [{bacc_lo:.4f}, {bacc_hi:.4f}]")
print(f"95% CI - Macro F1: [{f1_lo:.4f}, {f1_hi:.4f}]")
wandb.log({'final/acc_mean': np.mean(accs), 'final/acc_std':
np.std(accs), 'final/bacc_mean': np.mean(baccs), 'final/bacc_std':
np.std(baccs), 'final/f1_mean': np.mean(f1s), 'final/f1_std':
np.std(f1s)})
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>


==================================================
FOLD 1/5
==================================================
```

{"model_id":"923139d0c37d46fbb263ccd4359d5688","version_major":2,"version_minor":0}

{"model_id":"9594513e66b24825b20b6bf2fc3c8048","version_major":2,"version_minor":0}

E1: TrBAcc=0.3389, ValBAcc=0.3773, ValF1=0.1059

{"model_id":"e422df2ca32e4bf5ae7e4a3d0df06b14","version_major":2,"version_minor":0}

{"model_id":"7ff3fe94ce0a4c1db45f0454df583d58","version_major":2,"version_minor":0}

E2: TrBAcc=0.4332, ValBAcc=0.4993, ValF1=0.1712

{"model_id":"32bcf77cc482484cb95240c18c18f538","version_major":2,"version_minor":0}

{"model_id":"130d6df8a6114523bd983445fde1c6ff","version_major":2,"version_minor":0}

E3: TrBAcc=0.4644, ValBAcc=0.5360, ValF1=0.2290

{"model_id":"70fd35a31ddb47919727f8fd784a0648","version_major":2,"version_minor":0}

{"model_id":"3d079d383d5b4d1ab7ea2e8ffc256f06","version_major":2,"version_minor":0}

E4: TrBAcc=0.5364, ValBAcc=0.5593, ValF1=0.2700

{"model_id":"f17203636ea14ece8c7af6b599623d96","version_major":2,"version_minor":0}

{"model_id":"522a2d488747479b822c1369a546b3fd","version_major":2,"version_minor":0}

E5: TrBAcc=0.5133, ValBAcc=0.5636, ValF1=0.2695

{"model_id":"e5eebf4767414750a09f37a8ed990dee","version_major":2,"version_minor":0}

{"model_id":"93f7e1df6bea43b882693b4d706b4e37","version_major":2,"version_minor":0}

E6: TrBAcc=0.5456, ValBAcc=0.5989, ValF1=0.3447

{"model_id":"89b68b75dfe548c8adac679ef0f96117","version_major":2,"version_minor":0}

{"model_id":"53656a612f4c4a6db81fbd277aab2aa6","version_major":2,"version_minor":0}

```
E7: TrBAcc=0.5959, ValBAcc=0.6240, ValF1=0.3721

{"model_id":"3f77d5167d6242ef94c3d7860198f8ae","version_major":2,"vers
ion_minor":0}

{"model_id":"e1bfa5f4ef4b49e191e6875e3ac71e2d","version_major":2,"vers
ion_minor":0}

E8: TrBAcc=0.6196, ValBAcc=0.6227, ValF1=0.3664

{"model_id":"ad9efae93da344b9987f3bbe145d5903","version_major":2,"vers
ion_minor":0}

{"model_id":"4c56f56fcf36402b8eb4434a3991b5ec","version_major":2,"vers
ion_minor":0}

E9: TrBAcc=0.5782, ValBAcc=0.5933, ValF1=0.3099

{"model_id":"37701483ca20472080f5168980c89025","version_major":2,"vers
ion_minor":0}

{"model_id":"d89098d1ea9049e69529450869217c6a","version_major":2,"vers
ion_minor":0}

E10: TrBAcc=0.5838, ValBAcc=0.6179, ValF1=0.3202

{"model_id":"54cc347cb7f44223886435f1bf410adb","version_major":2,"vers
ion_minor":0}

{"model_id":"c5a377e04a66412fbca7c5f646ce5733","version_major":2,"vers
ion_minor":0}

E11: TrBAcc=0.5989, ValBAcc=0.6425, ValF1=0.3897

{"model_id":"ada395667ef646428f93c056008a6665","version_major":2,"vers
ion_minor":0}

{"model_id":"1402a1684c574ebb8d2c91791897e286","version_major":2,"vers
ion_minor":0}

E12: TrBAcc=0.6274, ValBAcc=0.6358, ValF1=0.3766

{"model_id":"bda31714fe5f4deca071475239a4184d","version_major":2,"vers
ion_minor":0}

{"model_id":"9c1fa32577934ccd9d895a80f872600a","version_major":2,"vers
ion_minor":0}

E13: TrBAcc=0.6501, ValBAcc=0.6479, ValF1=0.4311

{"model_id":"10bf40d09a194f0b840097bb2f511aa6","version_major":2,"vers
ion_minor":0}
```

{"model_id":"9eb1a1e885a44003bf4c90452f02171d","version_major":2,"version_minor":0}

E14: TrBAcc=0.6765, ValBAcc=0.6679, ValF1=0.4526

{"model_id":"b38ebcae90a04fd7ac5756bbf760ad35","version_major":2,"version_minor":0}

{"model_id":"504884e197d24df99bc2e71f43934f1c","version_major":2,"version_minor":0}

E15: TrBAcc=0.6858, ValBAcc=0.6786, ValF1=0.4672

{"model_id":"cee9ce3283d74b31b4e0429343fa7e20","version_major":2,"version_minor":0}

{"model_id":"fe742e7843c746449a0d133ca944e13c","version_major":2,"version_minor":0}

E16: TrBAcc=0.6939, ValBAcc=0.6781, ValF1=0.4700

{"model_id":"1a3f36ce991c4f349180e1b7806b2ec5","version_major":2,"version_minor":0}

{"model_id":"cdd40754901d4a088732afc0a65ad9a4","version_major":2,"version_minor":0}

E17: TrBAcc=0.6385, ValBAcc=0.6240, ValF1=0.3254

{"model_id":"233865a080af4f7b8a482b40958aef77","version_major":2,"version_minor":0}

{"model_id":"16cc1171fa1e42928393aed500b5fb49","version_major":2,"version_minor":0}

E18: TrBAcc=0.6211, ValBAcc=0.6576, ValF1=0.4019

{"model_id":"b3dee87e784c48d19ec68cb384f8bbb5","version_major":2,"version_minor":0}

{"model_id":"b249c9ba52ee47d18d8e60ba0069e6fb","version_major":2,"version_minor":0}

E19: TrBAcc=0.6381, ValBAcc=0.6549, ValF1=0.4266

{"model_id":"e59b1c9ca480457186246f370515f9c7","version_major":2,"version_minor":0}

{"model_id":"470c656d9e0b418e903a173e0fa481fe","version_major":2,"version_minor":0}

E20: TrBAcc=0.6479, ValBAcc=0.6318, ValF1=0.3862

{"model_id":"cac0131e733d42a386b94d4f1875eb98","version_major":2,"version_minor":0}

{"model_id":"05c8d6da0fd1445e80ae47a774f3a4d0","version_major":2,"version_minor":0}

E21: TrBAcc=0.6636, ValBAcc=0.6490, ValF1=0.3911

{"model_id":"499ba327e4b14c5ca1b743cf6004a768","version_major":2,"version_minor":0}

{"model_id":"06d7c2e086b1426b90bebbe499d1a8c5","version_major":2,"version_minor":0}

E22: TrBAcc=0.6732, ValBAcc=0.6642, ValF1=0.4442

{"model_id":"970017e12ce2466ebfd85a3da2c613c2","version_major":2,"version_minor":0}

{"model_id":"cef12ba1e0d346aa9d7125d9ed045b78","version_major":2,"version_minor":0}

E23: TrBAcc=0.6833, ValBAcc=0.6808, ValF1=0.4572

{"model_id":"e293406bf5774f6ea9a97bd006fe1605","version_major":2,"version_minor":0}

{"model_id":"b3859b3af32047c3a1cd2f8266f7cc01","version_major":2,"version_minor":0}

E24: TrBAcc=0.7043, ValBAcc=0.6962, ValF1=0.4987

{"model_id":"0ddc57681ab6497db7c924150e748f2e","version_major":2,"version_minor":0}

{"model_id":"3dae5042de744f01b9cd44f43cfab331","version_major":2,"version_minor":0}

E25: TrBAcc=0.7139, ValBAcc=0.7038, ValF1=0.5108

{"model_id":"227ea6ee3e694182b61d5e4ea9644635","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
```

```
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"baa92eb34e3541048652ca7d59f9fc69","version_major":2,"vers
ion_minor":0}

```
E26: TrBAcc=0.7206, ValBAcc=0.7080, ValF1=0.5501
```

{"model_id":"396ff0cf42fb4ce88b8a25c064e68fa7","version_major":2,"vers
ion_minor":0}

{"model_id":"9def7be33edb43bc8a4fd7cef7948d92","version_major":2,"vers
ion_minor":0}

```
E27: TrBAcc=0.7316, ValBAcc=0.7007, ValF1=0.5318
```

{"model_id":"9aa8036b2b9d44ab85ad8d2a0df9a2f4","version_major":2,"vers
ion_minor":0}

{"model_id":"eac2c75ef48044f1884155d88de1bf13","version_major":2,"vers
ion_minor":0}

```
E28: TrBAcc=0.7460, ValBAcc=0.7114, ValF1=0.5208
```

{"model_id":"1e0cb3a7f6644bad9f4c1f706f763ddf","version_major":2,"vers
ion_minor":0}

{"model_id":"5c8c098199e342f5a7d351d77519842a","version_major":2,"version_minor":0}

E29: TrBAcc=0.7531, ValBAcc=0.7082, ValF1=0.5439

{"model_id":"5b6d4694b35a45e1ba83d79c1da8d4b2","version_major":2,"version_minor":0}

{"model_id":"58a5bf8b041243c280995d10a69ea38c","version_major":2,"version_minor":0}

E30: TrBAcc=0.7605, ValBAcc=0.6981, ValF1=0.5602

{"model_id":"ef6fcf14b6584177a19d90bff0d35298","version_major":2,"version_minor":0}

TTA Results: Acc=0.4796, BAcc=0.7173, F1=0.5722

{"model_id":"fd6aeb6e1d3e4b34a1ccc445ce27ab22","version_major":2,"version_minor":0}

Fold 1 Complete: BAcc=0.7173, F1=0.5722

==========
FOLD 2/5
==========

{"model_id":"4c6bec195f8e4e04ba2d3968b216e9a6","version_major":2,"version_minor":0}

{"model_id":"548b385c6cf740bbbe0ecd3cd89c5845","version_major":2,"version_minor":0}

E1: TrBAcc=0.3257, ValBAcc=0.4351, ValF1=0.1016

{"model_id":"9de972fc10694d53839da8e516b06864","version_major":2,"version_minor":0}

{"model_id":"a779f30d32a741fcbff7c94f1ee32e94","version_major":2,"version_minor":0}

E2: TrBAcc=0.4241, ValBAcc=0.4624, ValF1=0.1564

{"model_id":"27e77172c32e462bbb2dfc36283332c0","version_major":2,"version_minor":0}

{"model_id":"478a4260ed034e6b967b50dcfd0a2c52","version_major":2,"version_minor":0}

E3: TrBAcc=0.4622, ValBAcc=0.5316, ValF1=0.2238

{"model_id":"8a560e5424324f09995172522874 31f8","version_major":2,"version_minor":0}

{"model_id":"9b5797f0033b44728f1d09a67ffa4584","version_major":2,"version_minor":0}

E4: TrBAcc=0.5327, ValBAcc=0.5651, ValF1=0.2891

{"model_id":"239e893979804fc3b0b2c03fdb4e6c64","version_major":2,"version_minor":0}

{"model_id":"1797a516237b4d55b6bba25939db7469","version_major":2,"version_minor":0}

E5: TrBAcc=0.5132, ValBAcc=0.5467, ValF1=0.2480

{"model_id":"3edc88e3dd4f47b6ba7f19182d6fbc98","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
IOStream.flush timed out
```

{"model_id":"42df03bc6e5f4056bf82c5eb6be4dcf3","version_major":2,"version_minor":0}

E6: TrBAcc=0.5542, ValBAcc=0.5707, ValF1=0.3145

{"model_id":"0a3f56e308fc4491a5f1166c4bd2ae91","version_major":2,"version_minor":0}

{"model_id":"6ddb7522fd554151ad0943f8925ca30c","version_major":2,"version_minor":0}

```
E7: TrBAcc=0.5984, ValBAcc=0.6279, ValF1=0.3772

{"model_id":"0e397db78cfd4ab88e1de290ff2f28ac","version_major":2,"vers
ion_minor":0}

{"model_id":"c5b4aecb79c446eeb803319b222955a9","version_major":2,"vers
ion_minor":0}

E8: TrBAcc=0.6256, ValBAcc=0.6292, ValF1=0.3679

{"model_id":"c22abdb90cc444278f1e2e788b7211c1","version_major":2,"vers
ion_minor":0}

{"model_id":"79a2c14678b64bb1b279b748af6c1632","version_major":2,"vers
ion_minor":0}

E9: TrBAcc=0.5780, ValBAcc=0.5929, ValF1=0.3145

{"model_id":"4046405165e14af3b6c59dcb90fd89c2","version_major":2,"vers
ion_minor":0}

{"model_id":"e4cc5e048f7d4006a3ebcc958c36f8ef","version_major":2,"vers
ion_minor":0}

E10: TrBAcc=0.5829, ValBAcc=0.6022, ValF1=0.3465

{"model_id":"00fd8fef57674567b2e934ae1dc26d9f","version_major":2,"vers
ion_minor":0}

{"model_id":"a637a2179cac4ee2adf33056b19ad9cf","version_major":2,"vers
ion_minor":0}

E11: TrBAcc=0.6095, ValBAcc=0.6273, ValF1=0.3790

{"model_id":"c5e5afaa19f44d1c9fc6047efced0194","version_major":2,"vers
ion_minor":0}

{"model_id":"c1a2216407bd480f8d00f9de2390fe22","version_major":2,"vers
ion_minor":0}

E12: TrBAcc=0.6362, ValBAcc=0.6546, ValF1=0.4037

{"model_id":"3030afe451d94a2d82cc9666dd3e1c94","version_major":2,"vers
ion_minor":0}

Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
```

```
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process

{"model_id":"94cfc6d9a07b42fd92809db55df68e8e","version_major":2,"version_minor":0}

E13: TrBAcc=0.6567, ValBAcc=0.6627, ValF1=0.4361

{"model_id":"88e78937ca484a1f9cbba7d549a6e79b","version_major":2,"version_minor":0}

{"model_id":"24f96f6c55d8460986ee165df8d9aee1","version_major":2,"version_minor":0}

E14: TrBAcc=0.6755, ValBAcc=0.6765, ValF1=0.4516

{"model_id":"e2b6bd2f6da5476eb011150d07542cbf","version_major":2,"version_minor":0}

{"model_id":"d8764339497147fbb08cf00c3551c5e4","version_major":2,"version_minor":0}

E15: TrBAcc=0.6874, ValBAcc=0.6839, ValF1=0.4867
```

{"model_id":"4dde0348220940b5a7a1d4c7a277bb05","version_major":2,"version_minor":0}

{"model_id":"431ae4916c394677923bce48e5a03c47","version_major":2,"version_minor":0}

E16: TrBAcc=0.6897, ValBAcc=0.6789, ValF1=0.4805

{"model_id":"247a26b054d44148aeb01453aa2c445d","version_major":2,"version_minor":0}

{"model_id":"4aa863752c65403497b0c8898e39ee39","version_major":2,"version_minor":0}

E17: TrBAcc=0.6489, ValBAcc=0.6536, ValF1=0.4031

{"model_id":"4a546512eaa244688b318d6726cee599","version_major":2,"version_minor":0}

{"model_id":"46f2dffe3b3a49b186c1a11409b1ceba","version_major":2,"version_minor":0}

E18: TrBAcc=0.6391, ValBAcc=0.6549, ValF1=0.4196

{"model_id":"eb67a27b491646ab8b088ed13c4ac19c","version_major":2,"version_minor":0}

{"model_id":"a41399bb8d9d4a1b8ffc1ed5abbac269","version_major":2,"version_minor":0}

E19: TrBAcc=0.6465, ValBAcc=0.6575, ValF1=0.4075

{"model_id":"51f9c471888341d89b09c53286dfd612","version_major":2,"version_minor":0}

{"model_id":"cfe90c3cfc984ebba0a29949a6c61f0d","version_major":2,"version_minor":0}

E20: TrBAcc=0.6674, ValBAcc=0.6816, ValF1=0.4426

{"model_id":"39dffc2c06fa452abfefb3d752e57ca7","version_major":2,"version_minor":0}

{"model_id":"205d7e80d1f84592bcf8ed2a44fa83b4","version_major":2,"version_minor":0}

E21: TrBAcc=0.6669, ValBAcc=0.6678, ValF1=0.3998

{"model_id":"09f7cb962bc14faf9ad274e653029704","version_major":2,"version_minor":0}

{"model_id":"24739d3aeed5461f8017ad5d9affbf6c","version_major":2,"version_minor":0}

```
E22: TrBAcc=0.6770, ValBAcc=0.6799, ValF1=0.4580

{"model_id":"9a468233c9c345c089990c6690740f98","version_major":2,"vers
ion_minor":0}

{"model_id":"be91664a8fc44b30856d67b17529c889","version_major":2,"vers
ion_minor":0}

E23: TrBAcc=0.6956, ValBAcc=0.6865, ValF1=0.4859

{"model_id":"3498116fcc7a4689bda6e791ccda2942","version_major":2,"vers
ion_minor":0}

{"model_id":"eb16d444bf14409c85a8e28822e09222","version_major":2,"vers
ion_minor":0}

E24: TrBAcc=0.7072, ValBAcc=0.6785, ValF1=0.4894

{"model_id":"dbd275dbe720457aa278e1eeb6f0cf16","version_major":2,"vers
ion_minor":0}

{"model_id":"5e10e599e90444ec8ca01fe05c777b9e","version_major":2,"vers
ion_minor":0}

E25: TrBAcc=0.7159, ValBAcc=0.7076, ValF1=0.5167

{"model_id":"e288f9763b84433db51a4e97b4412b35","version_major":2,"vers
ion_minor":0}

{"model_id":"eb059df4869d4c0690fa9cb3c03d842b","version_major":2,"vers
ion_minor":0}

E26: TrBAcc=0.7277, ValBAcc=0.6937, ValF1=0.4992

{"model_id":"65f4da63c28e4256bc5116e2a078b6ba","version_major":2,"vers
ion_minor":0}

{"model_id":"18a16333033e4dbfa4b99cea8b93faf8","version_major":2,"vers
ion_minor":0}

E27: TrBAcc=0.7347, ValBAcc=0.7086, ValF1=0.5199

{"model_id":"dcff3c64d25041518bbe7a0a5efd3e02","version_major":2,"vers
ion_minor":0}

{"model_id":"46d9c4fbd10c473bae7a84f1742483ff","version_major":2,"vers
ion_minor":0}

E28: TrBAcc=0.7471, ValBAcc=0.7026, ValF1=0.5339

{"model_id":"03b9cc380493429ebda378b86fd8a190","version_major":2,"vers
ion_minor":0}
```

{"model_id":"950b0037c9db41bb94d27a406ddcbb36","version_major":2,"version_minor":0}

E29: TrBAcc=0.7456, ValBAcc=0.7136, ValF1=0.5532

{"model_id":"cb78bbec55a04d10adfc98bc54bb3eab","version_major":2,"version_minor":0}

{"model_id":"28594b026174410180e3499f21dc0424","version_major":2,"version_minor":0}

E30: TrBAcc=0.7609, ValBAcc=0.7105, ValF1=0.5471

{"model_id":"32dd3df94e9545638085f4c5851a1c1f","version_major":2,"version_minor":0}

TTA Results: Acc=0.5174, BAcc=0.7268, F1=0.5662

{"model_id":"1ee31bf37ca64236a55f135599e9ba7a","version_major":2,"version_minor":0}

Fold 2 Complete: BAcc=0.7268, F1=0.5662

==========
FOLD 3/5
==========

{"model_id":"8304eea237e64dafbdf0b5b7005003cb","version_major":2,"version_minor":0}

{"model_id":"6f5e890ee35c48269db8c826ca734035","version_major":2,"version_minor":0}

E1: TrBAcc=0.3440, ValBAcc=0.4122, ValF1=0.1100

{"model_id":"effdec99e2ac4bf4a9d7a4075ca135bc","version_major":2,"version_minor":0}

{"model_id":"d87a417a89ee4dfcbeed08e83d5b2142","version_major":2,"version_minor":0}

E2: TrBAcc=0.4317, ValBAcc=0.4896, ValF1=0.1705

{"model_id":"36e8b2775fdb49c58cd07cd583afd9e8","version_major":2,"version_minor":0}

{"model_id":"9ae376c2622e4b2999357c7bcf10467a","version_major":2,"version_minor":0}

E3: TrBAcc=0.4659, ValBAcc=0.5375, ValF1=0.2179

{"model_id":"d8e3603a1092437886acb14d08dd7eee","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"761c8802d7794b158d8ff38992e55026","version_major":2,"version_minor":0}

E4: TrBAcc=0.5337, ValBAcc=0.5672, ValF1=0.2603

{"model_id":"9a8116c71a7d42dbab85eb3ec63228a8","version_major":2,"version_minor":0}

{"model_id":"ae078ed5a8d641eca7522a2a74413898","version_major":2,"version_minor":0}

```
E5: TrBAcc=0.5081, ValBAcc=0.5710, ValF1=0.2672

{"model_id":"639fb68527f04b75a57c4a0aeb518c6e","version_major":2,"vers
ion_minor":0}

{"model_id":"9207fddd5c6e4b0f9df5c746368f433d","version_major":2,"vers
ion_minor":0}

E6: TrBAcc=0.5491, ValBAcc=0.6027, ValF1=0.3189

{"model_id":"75d12858de294ff68ffd4e9863384970","version_major":2,"vers
ion_minor":0}

{"model_id":"524be5be5ff74d80bf0cb51e6d0ba49d","version_major":2,"vers
ion_minor":0}

E7: TrBAcc=0.5981, ValBAcc=0.6381, ValF1=0.3729

{"model_id":"51fbf8a64ef945c1bf8162dcfa2a9c2b","version_major":2,"vers
ion_minor":0}

{"model_id":"6ed547fbe7904cbe8d2a38d7570bb92f","version_major":2,"vers
ion_minor":0}

E8: TrBAcc=0.6245, ValBAcc=0.6342, ValF1=0.3777

{"model_id":"e3f19b2a90a3413ea0fdda81656b4165","version_major":2,"vers
ion_minor":0}

{"model_id":"a90fdcfcc986452fb6b777ae5f0f99d9","version_major":2,"vers
ion_minor":0}

E9: TrBAcc=0.5722, ValBAcc=0.5923, ValF1=0.2843

{"model_id":"25288428ace841958d003d3ef016d31e","version_major":2,"vers
ion_minor":0}

{"model_id":"d8eeb1bc482045b395e4d2c5308c56b7","version_major":2,"vers
ion_minor":0}

E10: TrBAcc=0.5822, ValBAcc=0.5994, ValF1=0.3221

{"model_id":"6e80078f8bb142549793ed07f16653ac","version_major":2,"vers
ion_minor":0}

{"model_id":"41db40d124734a83bbaac54abc6f5770","version_major":2,"vers
ion_minor":0}

E11: TrBAcc=0.6062, ValBAcc=0.6420, ValF1=0.3634

{"model_id":"823ebec6f54e4f71b6796da0e3e6714f","version_major":2,"vers
ion_minor":0}
```

{"model_id":"c6d7987bd63446ea9328837dce29ad6e","version_major":2,"version_minor":0}

E12: TrBAcc=0.6332, ValBAcc=0.6451, ValF1=0.4066

{"model_id":"1c94057f618f453aae4fad29b735f4ce","version_major":2,"version_minor":0}

{"model_id":"afd29fdc0e3948a39c3b37a7e181ef21","version_major":2,"version_minor":0}

E13: TrBAcc=0.6549, ValBAcc=0.6502, ValF1=0.4313

{"model_id":"c2b97c7cbd674acfa3696ab9532f9ecc","version_major":2,"version_minor":0}

{"model_id":"1afe791c75174941a2d3daf36754f94e","version_major":2,"version_minor":0}

E14: TrBAcc=0.6783, ValBAcc=0.6850, ValF1=0.4999

{"model_id":"54f3956d761841cda2708139ad3e63ce","version_major":2,"version_minor":0}

{"model_id":"53037a37403f4ec19bb16bfb1822f28e","version_major":2,"version_minor":0}

E15: TrBAcc=0.6969, ValBAcc=0.6935, ValF1=0.5056

{"model_id":"c45b46b60bca46d4ac95a48e106b7d2e","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
IOStream.flush timed out
```

{"model_id":"a34a1fe148c94a35b685531fbcf1f692","version_major":2,"vers
ion_minor":0}

```
E16: TrBAcc=0.6966, ValBAcc=0.6923, ValF1=0.5004
```

{"model_id":"509f3473eda741c6a78240a742239e17","version_major":2,"vers
ion_minor":0}

{"model_id":"0ab35f8a86874117a6c351321a326be9","version_major":2,"vers
ion_minor":0}

```
E17: TrBAcc=0.6431, ValBAcc=0.6320, ValF1=0.3371
```

{"model_id":"6075589662414b28b95abec57438f554","version_major":2,"vers
ion_minor":0}

{"model_id":"a3a67fcd4b1740dd8d613929959559b0","version_major":2,"vers
ion_minor":0}

```
E18: TrBAcc=0.6293, ValBAcc=0.6389, ValF1=0.3936
```

{"model_id":"059dff3104ea4f08902d5799f0f07867","version_major":2,"vers
ion_minor":0}

{"model_id":"ec60d605c604414e990580ee83cf490c","version_major":2,"vers
ion_minor":0}

```
E19: TrBAcc=0.6500, ValBAcc=0.6294, ValF1=0.3647
```

{"model_id":"3633717fdcdb42f7be62e3c416accc6d","version_major":2,"vers
ion_minor":0}

{"model_id":"165b6bf552764d3b9d047862c6a73628","version_major":2,"version_minor":0}

E20: TrBAcc=0.6639, ValBAcc=0.6707, ValF1=0.4199

{"model_id":"b57a98b114c54479b3053d0a98e1b535","version_major":2,"version_minor":0}

{"model_id":"275455e5bfd043978ab03fc0cf192df2","version_major":2,"version_minor":0}

E21: TrBAcc=0.6749, ValBAcc=0.6900, ValF1=0.4336

{"model_id":"6fbd4b59e554475f8bb4bd8620fa617e","version_major":2,"version_minor":0}

{"model_id":"bba2b5276e5041b583f924d42053ce3f","version_major":2,"version_minor":0}

E22: TrBAcc=0.6727, ValBAcc=0.6734, ValF1=0.4827

{"model_id":"cd23ecaf0a274f12b81e5206ad9ed459","version_major":2,"version_minor":0}

{"model_id":"b377050f1e97489ba10fc45310bf4f4d","version_major":2,"version_minor":0}

E23: TrBAcc=0.6908, ValBAcc=0.6891, ValF1=0.4651

{"model_id":"ec5cbea2d321468c91118e562b64268b","version_major":2,"version_minor":0}

{"model_id":"b3c538f30ce845aa9abd92b44d2c6edb","version_major":2,"version_minor":0}

E24: TrBAcc=0.7026, ValBAcc=0.7035, ValF1=0.5091

{"model_id":"4de569a9bd824250aa39db1f7dba2c3a","version_major":2,"version_minor":0}

{"model_id":"7d79930fafb54e04b9aebe171083ecb8","version_major":2,"version_minor":0}

E25: TrBAcc=0.7222, ValBAcc=0.7159, ValF1=0.5285

{"model_id":"ac7628b850c84e9b8eebf3a43f7e90cf","version_major":2,"version_minor":0}

{"model_id":"dea23537100c4f83bc1f40b5a9d07453","version_major":2,"version_minor":0}

E26: TrBAcc=0.7298, ValBAcc=0.7235, ValF1=0.5360

{"model_id":"7d7c437f954849fbb0e7bbed933e2f47","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"2f4623fc6fd94f5b936c0d57632712db","version_major":2,"version_minor":0}

```
E27: TrBAcc=0.7333, ValBAcc=0.7213, ValF1=0.5250
```

{"model_id":"595c80e8f1cf42438f9f3faae7dba502","version_major":2,"version_minor":0}

{"model_id":"30a46efa6f0f43c3b9feaa89ad5217c1","version_major":2,"version_minor":0}

```
E28: TrBAcc=0.7447, ValBAcc=0.7353, ValF1=0.5594

{"model_id":"631db2a0ebe949bd8e202efa2e930740","version_major":2,"vers
ion_minor":0}

{"model_id":"a323804a20ce41caac8ba4ae4b7a87df","version_major":2,"vers
ion_minor":0}

E29: TrBAcc=0.7498, ValBAcc=0.7363, ValF1=0.5720

{"model_id":"f9f53b4955c2413bb6a72c0b0e414050","version_major":2,"vers
ion_minor":0}

{"model_id":"5458a490f602448ab25cb2a9c1f6325c","version_major":2,"vers
ion_minor":0}

E30: TrBAcc=0.7518, ValBAcc=0.7380, ValF1=0.5851

{"model_id":"075aaf1c2a8b4c33a3afca402b6aba5f","version_major":2,"vers
ion_minor":0}

TTA Results: Acc=0.4907, BAcc=0.7432, F1=0.5891

{"model_id":"f744266b40e04884ae790e8f90daeb31","version_major":2,"vers
ion_minor":0}


Fold 3 Complete: BAcc=0.7432, F1=0.5891

==================================================
FOLD 4/5
==================================================

{"model_id":"162be4fa62074cdd9ba8197906ed87d4","version_major":2,"vers
ion_minor":0}

{"model_id":"b03e3feb5afa4afeb2a9213b0e64b2e8","version_major":2,"vers
ion_minor":0}

E1: TrBAcc=0.3356, ValBAcc=0.4378, ValF1=0.1200

{"model_id":"a5be0adfc0354de899948a0fa3bf4c24","version_major":2,"vers
ion_minor":0}

{"model_id":"526e1776e1f740c2ac9b26e41c1f7a25","version_major":2,"vers
ion_minor":0}

E2: TrBAcc=0.4326, ValBAcc=0.4937, ValF1=0.1734

{"model_id":"4b417f9f23354592ba32677ebd0606df","version_major":2,"vers
ion_minor":0}
```

{"model_id":"86de4ae4c5ea455b8276c3173a92d558","version_major":2,"version_minor":0}

E3: TrBAcc=0.4677, ValBAcc=0.5478, ValF1=0.2452

{"model_id":"e14c853c19ac499fbc22dc05e427c3f4","version_major":2,"version_minor":0}

{"model_id":"3fbf779b6a6944a3846aad70175c1d15","version_major":2,"version_minor":0}

E4: TrBAcc=0.5367, ValBAcc=0.5959, ValF1=0.2994

{"model_id":"e69bd0aa16fd4cd6a57dcfdb709ba67f","version_major":2,"version_minor":0}

{"model_id":"b9e2a264ecf744de8f97bb1e4572160b","version_major":2,"version_minor":0}

E5: TrBAcc=0.5138, ValBAcc=0.5400, ValF1=0.2474

{"model_id":"7e3a5896041349c293462fc282564ede","version_major":2,"version_minor":0}

{"model_id":"f38996e3ef5343529bd9b3aae30fd703","version_major":2,"version_minor":0}

E6: TrBAcc=0.5580, ValBAcc=0.6132, ValF1=0.3360

{"model_id":"3c3c3c0ce0834a2b932725154b80b728","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"367c7249f47547709601ac67f7e5c759","version_major":2,"vers
ion_minor":0}

E7: TrBAcc=0.6100, ValBAcc=0.6441, ValF1=0.3827

{"model_id":"ff2a4d79dc15444ebf1804f8a1a33a22","version_major":2,"vers
ion_minor":0}

{"model_id":"ef6b6c2519994349955bfe42b0d38f10","version_major":2,"vers
ion_minor":0}

E8: TrBAcc=0.6279, ValBAcc=0.6486, ValF1=0.3947

{"model_id":"7f6422a2ed76471fb748039ef285f145","version_major":2,"vers
ion_minor":0}

{"model_id":"8b8de80aabb44ed09813026d91dcb087","version_major":2,"vers
ion_minor":0}

E9: TrBAcc=0.5781, ValBAcc=0.6237, ValF1=0.3618

{"model_id":"0c8d6b3f7c404e54bb6bfeaee5a5dbea","version_major":2,"vers
ion_minor":0}

{"model_id":"9ab5f111677149e1b1fed045850b58f4","version_major":2,"vers
ion_minor":0}

E10: TrBAcc=0.5863, ValBAcc=0.6699, ValF1=0.3945

{"model_id":"d11d2ac3437a49e7a6df96ddaa5d7dca","version_major":2,"vers
ion_minor":0}

{"model_id":"382117ba219b4056bd1bfce7d6b42bfa","version_major":2,"version_minor":0}

E11: TrBAcc=0.6230, ValBAcc=0.6522, ValF1=0.3783

{"model_id":"7924f8ec8c444844b3f88f62ea34b925","version_major":2,"version_minor":0}

{"model_id":"c2d8f8d0c84541b08bac9be94e9fca1a","version_major":2,"version_minor":0}

E12: TrBAcc=0.6355, ValBAcc=0.6902, ValF1=0.4526

{"model_id":"25cb9c948c5e4c098c80009e0910c211","version_major":2,"version_minor":0}

{"model_id":"341d7c9b88e14fe395e26281f8ba8fd0","version_major":2,"version_minor":0}

E13: TrBAcc=0.6728, ValBAcc=0.6887, ValF1=0.4577

{"model_id":"f80c3a0da55141599e395b702bbf3cff","version_major":2,"version_minor":0}

{"model_id":"c7dd0ed023db403c80724ec1d6fdd3cf","version_major":2,"version_minor":0}

E14: TrBAcc=0.6808, ValBAcc=0.6806, ValF1=0.4578

{"model_id":"40ba8a2a08654b0088f5a075189d878e","version_major":2,"version_minor":0}

{"model_id":"333be8bab07547b9ac43d6cdf014d76a","version_major":2,"version_minor":0}

E15: TrBAcc=0.6948, ValBAcc=0.6992, ValF1=0.4847

{"model_id":"e41cb609303f4771a492c50e58877db3","version_major":2,"version_minor":0}

{"model_id":"f201a2b59ba04402bea872b9f1d4f92b","version_major":2,"version_minor":0}

E16: TrBAcc=0.7050, ValBAcc=0.6988, ValF1=0.4797

{"model_id":"c0af3b83f59d4cbbb92a5745887a1924","version_major":2,"version_minor":0}

{"model_id":"fb873d3d286c4b7bb0ca3c88e66e54f5","version_major":2,"version_minor":0}

E17: TrBAcc=0.6451, ValBAcc=0.6458, ValF1=0.3409

{"model_id":"d0bc2927ca414faa9233fb62a38a3baf","version_major":2,"version_minor":0}

{"model_id":"ed2194b9c10c455985fc7e39001013ac","version_major":2,"version_minor":0}

E18: TrBAcc=0.6360, ValBAcc=0.6558, ValF1=0.3926

{"model_id":"08275282f80a4b4686191f3ead2b2b80","version_major":2,"version_minor":0}

{"model_id":"f84f561f332b4e219cf6a364ec1faddc","version_major":2,"version_minor":0}

E19: TrBAcc=0.6526, ValBAcc=0.6829, ValF1=0.3995

{"model_id":"67c546ea4d7e4f56bb8a92f4bfdb91c3","version_major":2,"version_minor":0}

{"model_id":"105d407a37554dee8841ddfda31de7aa","version_major":2,"version_minor":0}

E20: TrBAcc=0.6642, ValBAcc=0.6750, ValF1=0.4071

{"model_id":"52ddedf1a0764f538a6af7f367cd0e02","version_major":2,"version_minor":0}

{"model_id":"c30ddd9af2514405bb3f9e26bf6ff7f0","version_major":2,"version_minor":0}

E21: TrBAcc=0.6770, ValBAcc=0.6789, ValF1=0.4277

{"model_id":"b36ce1f81e4245a8b49646f58a6f0bf0","version_major":2,"version_minor":0}

{"model_id":"0b8ecbb3106940428396c58efad2f51a","version_major":2,"version_minor":0}

E22: TrBAcc=0.6746, ValBAcc=0.6921, ValF1=0.4269

{"model_id":"78115fa0c8a04931af8086bcec2b3063","version_major":2,"version_minor":0}

{"model_id":"a04cffe9a05a4b909fa9606a89b39037","version_major":2,"version_minor":0}

E23: TrBAcc=0.6877, ValBAcc=0.6970, ValF1=0.4851

{"model_id":"007cb4b5a0354502ab5f24d86967f869","version_major":2,"version_minor":0}

{"model_id":"cd3b341d999946b084075aa35f77424d","version_major":2,"version_minor":0}

```
E24: TrBAcc=0.7033, ValBAcc=0.7076, ValF1=0.5009

{"model_id":"02b3977e89e843119778ce3642f37a5b","version_major":2,"vers
ion_minor":0}

{"model_id":"822b3ce94ee84ab2a5f3e78c296f8286","version_major":2,"vers
ion_minor":0}

E25: TrBAcc=0.7187, ValBAcc=0.7252, ValF1=0.4847

{"model_id":"5920489699244f04a50dd8b0cfa66b6d","version_major":2,"vers
ion_minor":0}

{"model_id":"0899f903555f4de0a5fc0b4582772769","version_major":2,"vers
ion_minor":0}

E26: TrBAcc=0.7327, ValBAcc=0.7186, ValF1=0.5424

{"model_id":"e92e6c76bc1243fb9c84ce4b2cbcc58d","version_major":2,"vers
ion_minor":0}

{"model_id":"a734b4996475426297b3539d267a17bd","version_major":2,"vers
ion_minor":0}

E27: TrBAcc=0.7418, ValBAcc=0.7324, ValF1=0.5661

{"model_id":"197480f9f963410a86845d6a7558d24c","version_major":2,"vers
ion_minor":0}

{"model_id":"08555570e5ca4b2d9b7ecb858aabcb00","version_major":2,"vers
ion_minor":0}

E28: TrBAcc=0.7506, ValBAcc=0.7322, ValF1=0.5838

{"model_id":"c0a18229fbfb444cb98eeae4a2242b57","version_major":2,"vers
ion_minor":0}

{"model_id":"feaaf2108cc3440eae50ab00d4f4409b","version_major":2,"vers
ion_minor":0}

E29: TrBAcc=0.7579, ValBAcc=0.7295, ValF1=0.5744

{"model_id":"1b734fccb6594cc487f4b8f844a6c9a6","version_major":2,"vers
ion_minor":0}

{"model_id":"6dcee389d22649ed913f4ea6431f8684","version_major":2,"vers
ion_minor":0}

E30: TrBAcc=0.7593, ValBAcc=0.7392, ValF1=0.5878

{"model_id":"68a3b50b38594c978c73bcd932cc3085","version_major":2,"vers
ion_minor":0}

TTA Results: Acc=0.5221, BAcc=0.7469, F1=0.6025
```

{"model_id":"a7e3c04f76fa425b819abd2e53f154d8","version_major":2,"version_minor":0}

```
Fold 4 Complete: BAcc=0.7469, F1=0.6025

==================================================
FOLD 5/5
==================================================

Warning: You are sending unauthenticated requests to the HF Hub.
Please set a HF_TOKEN to enable higher rate limits and faster
downloads.
```

{"model_id":"d97625f8eed24736b387228cd842f601","version_major":2,"version_minor":0}

{"model_id":"fc10ce65190b4a81afd177c7bd3d2213","version_major":2,"version_minor":0}

```
E1: TrBAcc=0.3283, ValBAcc=0.4002, ValF1=0.0991
```

{"model_id":"70cf9175c90f4ddbafaf368cb3bb5fdf","version_major":2,"version_minor":0}

{"model_id":"62560a5b735e4df695d7606ba16eb891","version_major":2,"version_minor":0}

```
E2: TrBAcc=0.4230, ValBAcc=0.4923, ValF1=0.1695
```

{"model_id":"5977afadaeae485c9c338b91332bd1ee","version_major":2,"version_minor":0}

{"model_id":"e5168ef92dde456ebb3606069121d52d","version_major":2,"version_minor":0}

```
E3: TrBAcc=0.4640, ValBAcc=0.4952, ValF1=0.2137
```

{"model_id":"acf46fd2e43a4ad6a1cc37ddc6dc253e","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
```

```
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
IOStream.flush timed out
IOStream.flush timed out
```


```
{"model_id":"f27932543b0149be8ecb456b1a371751","version_major":2,"vers
ion_minor":0}
```


```
E4: TrBAcc=0.5297, ValBAcc=0.5749, ValF1=0.2786
```


```
{"model_id":"876f73210b864bff9de82c8e22e8c1f6","version_major":2,"vers
ion_minor":0}

{"model_id":"88aa3e468076420a956fb8ca29de0d0f","version_major":2,"vers
ion_minor":0}
```


```
E5: TrBAcc=0.5122, ValBAcc=0.5595, ValF1=0.2673
```


```
{"model_id":"93ad592318724fc2af90287dd1068e08","version_major":2,"vers
ion_minor":0}

{"model_id":"2270c854f22d4906afaa8f8d0f41941e","version_major":2,"vers
ion_minor":0}
```


```
E6: TrBAcc=0.5607, ValBAcc=0.6127, ValF1=0.3326
```

{"model_id":"1c7abf97e9c74b5d998610e44a897417","version_major":2,"version_minor":0}

{"model_id":"a8716f2547e8490189ff6b1eca438335","version_major":2,"version_minor":0}

E7: TrBAcc=0.5963, ValBAcc=0.6371, ValF1=0.3696

{"model_id":"6edeafecb9d841a8980b6922fe000743","version_major":2,"version_minor":0}

{"model_id":"d8c96bef0a9b4fff91ff4518c6e6ce2e","version_major":2,"version_minor":0}

E8: TrBAcc=0.6177, ValBAcc=0.6335, ValF1=0.3738

{"model_id":"df0474c7efcc484095228adc47905d57","version_major":2,"version_minor":0}

{"model_id":"893d767b9c9a478ebd2a3fb5cc55fc17","version_major":2,"version_minor":0}

E9: TrBAcc=0.5803, ValBAcc=0.6010, ValF1=0.3456

{"model_id":"2225fe72835845e4bc3fc4e7f0327f03","version_major":2,"version_minor":0}

{"model_id":"f6e037b8803c4e9c93678998a511d3e0","version_major":2,"version_minor":0}

E10: TrBAcc=0.5841, ValBAcc=0.5993, ValF1=0.3276

{"model_id":"a3dbde75cacc4b329e019097f0dddb1c","version_major":2,"version_minor":0}

{"model_id":"c0dbeffeb6874df28287b0a1aeed8747","version_major":2,"version_minor":0}

E11: TrBAcc=0.6161, ValBAcc=0.6246, ValF1=0.3694

{"model_id":"ed9d61bc18004c5cbb3238e636962c16","version_major":2,"version_minor":0}

{"model_id":"22f6c43ff4da4f0aa09d25da92c20aa5","version_major":2,"version_minor":0}

E12: TrBAcc=0.6339, ValBAcc=0.6324, ValF1=0.3616

{"model_id":"f940fea64df94010b3bc2b9ca236bf9a","version_major":2,"version_minor":0}

{"model_id":"2b31de4958274ed091d438bda46fe1ba","version_major":2,"version_minor":0}

```
E13: TrBAcc=0.6603, ValBAcc=0.6658, ValF1=0.4269

{"model_id":"e5df78268ef2427c8b0636cec26e7ae3","version_major":2,"vers
ion_minor":0}

Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process

{"model_id":"2d18a4a99f3b4d258017e175c0e696d3","version_major":2,"vers
ion_minor":0}

E14: TrBAcc=0.6742, ValBAcc=0.6766, ValF1=0.4420

{"model_id":"a8e00c15e1d14be092885c33593f73f6","version_major":2,"vers
ion_minor":0}
```

{"model_id":"85f6d132b89d44d084a85e2f0d7e5db4","version_major":2,"version_minor":0}

E15: TrBAcc=0.6921, ValBAcc=0.6911, ValF1=0.4772

{"model_id":"b99c82d70f7b4fa6b93d84fbb48a6cad","version_major":2,"version_minor":0}

{"model_id":"bbbc7000e0c24045ae9a5a3b11f74a04","version_major":2,"version_minor":0}

E16: TrBAcc=0.6987, ValBAcc=0.6891, ValF1=0.4820

{"model_id":"1ae7266f13bf422db7015dcf7a25af68","version_major":2,"version_minor":0}

{"model_id":"34f79ea831834b958115ceea1e9fbf1f","version_major":2,"version_minor":0}

E17: TrBAcc=0.6502, ValBAcc=0.6642, ValF1=0.4061

{"model_id":"a1343412872443eb8b4b2212dd786d2c","version_major":2,"version_minor":0}

{"model_id":"3c29e20d9c6d40068ff953e86fc63476","version_major":2,"version_minor":0}

E18: TrBAcc=0.6350, ValBAcc=0.6468, ValF1=0.3961

{"model_id":"cff54a05e0804d5cad1aee139be80d5f","version_major":2,"version_minor":0}

{"model_id":"88458ce04d504f238993f7ada03ca4c8","version_major":2,"version_minor":0}

E19: TrBAcc=0.6567, ValBAcc=0.6799, ValF1=0.4245

{"model_id":"da5b9b5186994e64beebda3c5572042e","version_major":2,"version_minor":0}

{"model_id":"6274a09fa9534024bbcb4e3c1dd24af5","version_major":2,"version_minor":0}

E20: TrBAcc=0.6723, ValBAcc=0.6475, ValF1=0.4106

{"model_id":"dd87d4666b9d41a19945dd261f505ea4","version_major":2,"version_minor":0}

{"model_id":"a140b1b16c4f43e8aae50bde63176fc1","version_major":2,"version_minor":0}

E21: TrBAcc=0.6733, ValBAcc=0.6529, ValF1=0.4128

{"model_id":"6ab1f5e772654d1cb5f9300cb3dd2bad","version_major":2,"version_minor":0}

{"model_id":"b891dde709c6463d9a35cde67a8b15eb","version_major":2,"version_minor":0}

```
E22: TrBAcc=0.6838, ValBAcc=0.6930, ValF1=0.4403
```

{"model_id":"fef8811a67e54684b52a0900cf45ce3c","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

```
IOStream.flush timed out
IOStream.flush timed out

{"model_id":"15e1c454f80e49129d5cdfed034d2bc9","version_major":2,"vers
ion_minor":0}

E23: TrBAcc=0.6930, ValBAcc=0.7046, ValF1=0.4926

{"model_id":"c63b279eaaaa469ca84485a30dd84fc8","version_major":2,"vers
ion_minor":0}

{"model_id":"1c34bf2ff5284114a6d9aa7f9f4fb490","version_major":2,"vers
ion_minor":0}

E24: TrBAcc=0.7052, ValBAcc=0.6893, ValF1=0.5060

{"model_id":"98e2deec9eb54ff1a9404d968a3d9720","version_major":2,"vers
ion_minor":0}

{"model_id":"5399bd3d208841cda600cf2dd97acff8","version_major":2,"vers
ion_minor":0}

E25: TrBAcc=0.7171, ValBAcc=0.6906, ValF1=0.4972

{"model_id":"5c16a04f20254fec8ea489f91b1908d7","version_major":2,"vers
ion_minor":0}

E26: TrBAcc=0.7319, ValBAcc=0.7049, ValF1=0.5149

{"model_id":"ce4c0dcab99e437a829e8aecc7111875","version_major":2,"vers
ion_minor":0}

{"model_id":"e7269d609577416bb5be88056aac0bd2","version_major":2,"vers
ion_minor":0}

E27: TrBAcc=0.7359, ValBAcc=0.6990, ValF1=0.5146

{"model_id":"cb7d908b4ea74aabb6385be5aef82415","version_major":2,"vers
ion_minor":0}

{"model_id":"4ae4d7ce0219459aa2045d0a966b3b10","version_major":2,"vers
ion_minor":0}

E28: TrBAcc=0.7483, ValBAcc=0.7168, ValF1=0.5391

{"model_id":"5a84adf03de249e49c0599d161bc0eb6","version_major":2,"vers
ion_minor":0}

{"model_id":"09d8c009124d4c389cf2ee12b5a11cdb","version_major":2,"vers
ion_minor":0}

E29: TrBAcc=0.7548, ValBAcc=0.7159, ValF1=0.5752
```

{"model_id":"95bac629e80d4eea9049d9381b4a7022","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

{"model_id":"64229431c95d4a83bc11efbebac5cdac","version_major":2,"version_minor":0}

```
E30: TrBAcc=0.7580, ValBAcc=0.7244, ValF1=0.5661
```

{"model_id":"ee3bef04963a41c98c3553b0f18d6859","version_major":2,"version_minor":0}

```
TTA Results: Acc=0.5156, BAcc=0.7334, F1=0.5783
```

{"model_id":"857890e8aa80488794bdc500a0f9f3ca","version_major":2,"version_minor":0}

```
Fold 5 Complete: BAcc=0.7334, F1=0.5783

================================================
5-FOLD CV RESULTS
================================================
Accuracy: 0.5051 ± 0.0168
Balanced Accuracy: 0.7335 ± 0.0108
Macro F1: 0.5817 ± 0.0129

95% CI - Accuracy: [0.4818, 0.5284]
95% CI - Balanced Acc: [0.7186, 0.7485]
95% CI - Macro F1: [0.5638, 0.5995]
```

```
# CELL 20B: CLINICAL SAFETY (MEL operating point + uncertainty
referral + calibration)

import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix, roc_curve,
roc_auc_score, precision_recall_fscore_support

MEL_CLASS = CONFIG["class_names"].index("MEL")

def mel_operating_point(y_true, y_probs, target_sens=0.95):
    """
    Choose a MEL probability threshold to reach >= target sensitivity
(recall for MEL),
    and report specificity, precision, F1 at that threshold.
    """
    mel_true = (y_true == MEL_CLASS).astype(int)
    mel_score = y_probs[:, MEL_CLASS]

    fpr, tpr, thr = roc_curve(mel_true, mel_score)
    # tpr = sensitivity, fpr = 1 - specificity
    idx = np.where(tpr >= target_sens)[0]
    if len(idx) == 0:
        best_idx = np.argmax(tpr)
    else:
        best_idx = idx[np.argmin(fpr[idx])]  # highest specificity
among those meeting target sensitivity

    threshold = thr[best_idx]
    mel_pred = (mel_score >= threshold).astype(int)

    tn, fp, fn, tp = confusion_matrix(mel_true, mel_pred).ravel()
    sens = tp / (tp + fn + 1e-9)
    spec = tn / (tn + fp + 1e-9)
    prec = tp / (tp + fp + 1e-9)
    f1 = 2 * prec * sens / (prec + sens + 1e-9)
    auc = roc_auc_score(mel_true, mel_score) if 0 < mel_true.sum() <
len(mel_true) else 0.0

    return {
        "target_sensitivity": target_sens,
        "chosen_threshold": float(threshold),
        "MEL_sensitivity": float(sens),
        "MEL_specificity": float(spec),
        "MEL_precision": float(prec),
        "MEL_F1": float(f1),
        "MEL_AUROC": float(auc),
        "tp": int(tp), "fp": int(fp), "tn": int(tn), "fn": int(fn),
    }
```

```python
def referral_policy(y_true, y_pred, y_probs, uncertainties,
reject_percentiles=(80, 90, 95)):
    """
    Uncertainty-based referral: reject top X% most uncertain cases,
    compute coverage + performance on remaining cases.
    """
    results = []
    uncertainties = np.asarray(uncertainties)
    for p in reject_percentiles:
        thr = np.percentile(uncertainties, p)
        keep = uncertainties <= thr
        coverage = keep.mean()

        yt = y_true[keep]
        yp = y_pred[keep]
        ypr = y_probs[keep]

        acc = (yp == yt).mean() if len(yt) else 0
        # MEL sensitivity on kept cases
        mel_true = (yt == MEL_CLASS).astype(int)
        mel_pred = (yp == MEL_CLASS).astype(int)
        tn, fp, fn, tp = confusion_matrix(mel_true, mel_pred,
labels=[0,1]).ravel()
        mel_sens = tp / (tp + fn + 1e-9)

        results.append({
            "reject_percentile": p,
            "uncertainty_threshold": float(thr),
            "coverage": float(coverage),
            "accuracy_on_kept": float(acc),
            "MEL_sensitivity_on_kept": float(mel_sens),
            "n_kept": int(len(yt)),
            "n_referred": int((~keep).sum())
        })
    return pd.DataFrame(results)

# Aggregate across folds using stored fold_results
# NOTE: fold_results[i]['labels'], ['preds'], ['probs'] exist in your
notebook.
all_y_true = np.concatenate([r["labels"] for r in fold_results])
all_y_pred = np.concatenate([r["preds"] for r in fold_results])
all_y_prob = np.concatenate([r["probs"] for r in fold_results])

# If you computed uncertainties per fold (from estimate_uncertainty),
collect them here.
# If not available, you can re-run uncertainty on the val loaders per
fold or skip referral section.
# For now, we'll recompute uncertainties on a quick subset by re-using
the first fold model + val data if you saved them.
# (If you want full referral analysis per-fold, tell me and I'll give
```

```
the exact fold-level code.)
print("Clinical safety: MEL operating point on pooled CV predictions")
mel_report = mel_operating_point(all_y_true, all_y_prob,
target_sens=0.95)
mel_df = pd.DataFrame([mel_report])
print(mel_df.to_string(index=False))
wandb.log({"clinical/MEL_operating_point":
wandb.Table(dataframe=mel_df)})

print("\nTip: For uncertainty-based referral, you need uncertainties
aligned with these pooled predictions.")
print("If you want, I can provide the exact fold-wise code to store
uncertainties during validation and aggregate them.")
```

```
Clinical safety: MEL operating point on pooled CV predictions
 target_sensitivity  chosen_threshold  MEL_sensitivity
MEL_specificity  MEL_precision   MEL_F1  MEL_AUROC   tp    fp   tn  fn
              0.95          0.140667         0.950022
0.348359       0.240591 0.383949   0.819794 4296 13560 7249 226
```

```
---------------------------------------------------------------------
-----
Error                                     Traceback (most recent call
last)
Cell In[40], line 94
     92 mel_df = pd.DataFrame([mel_report])
     93 print(mel_df.to_string(index=False))
---> 94 wandb.log({"clinical/MEL_operating_point":
wandb.Table(dataframe=mel_df)})
     96 print("\nTip: For uncertainty-based referral, you need
uncertainties aligned with these pooled predictions.")
     97 print("If you want, I can provide the exact fold-wise code to
store uncertainties during validation and aggregate them.")

File
/usr/local/lib/python3.12/dist-packages/wandb/sdk/lib/preinit.py:36,
in PreInitCallable.<locals>.preinit_wrapper(*args, **kwargs)
     35 def preinit_wrapper(*args: Any, **kwargs: Any) -> Any:
---> 36     raise wandb.Error(f"You must call wandb.init() before
{name}()")

Error: You must call wandb.init() before wandb.log()
```

```
wandb.log({"comparison_protocol": "Budgeted baselines/ablations: 10
epochs; UHViT full CV: 30 epochs"})

# CELL 21: ABLATION STUDY (FIXED LABELS + BEST-EPOCH SELECTION)

print("\n" + "="*50 + "\nABLATION STUDY\n" + "="*50)

# Define ablation models
```

```
ablation_models = {
    "UHViT (Full)": lambda: UHViT(CONFIG["num_classes"],
CONFIG["dropout_rate"]),
    "Swin-T Only": lambda: SwinOnly(CONFIG["num_classes"],
CONFIG["dropout_rate"]),
    "EfficientNet Only": lambda:
EfficientNetOnly(CONFIG["num_classes"], CONFIG["dropout_rate"]),
    "Concat Fusion": lambda: UHViTConcatFusion(CONFIG["num_classes"],
CONFIG["dropout_rate"]),
}

# Use a fixed fold split for budgeted ablation (fold 1)
train_idx, val_idx = list(skf.split(train_df, all_labels))[0]
abl_train_df, abl_val_df =
train_df.iloc[train_idx].reset_index(drop=True),
train_df.iloc[val_idx].reset_index(drop=True)

# Datasets & loaders (same transforms as main training)
abl_train_dataset = ISIC2019Dataset(abl_train_df, train_img_dir,
get_train_transforms(CONFIG["img_size"]))
abl_val_dataset   = ISIC2019Dataset(abl_val_df,   train_img_dir,
get_val_transforms(CONFIG["img_size"]))

# FIXED: build exactly one label per sample
abl_train_labels = []
for i in range(len(abl_train_df)):
    row = abl_train_df.iloc[i]
    label = 0
    for j, cls in enumerate(CONFIG["class_names"]):
        if cls in abl_train_df.columns and float(row[cls]) == 1.0:
            label = j
            break
    abl_train_labels.append(label)
abl_train_labels = np.array(abl_train_labels)

abl_sampler = create_balanced_sampler(abl_train_labels)

abl_train_loader = DataLoader(
    abl_train_dataset,
    batch_size=CONFIG["batch_size"],
    sampler=abl_sampler,
    num_workers=CONFIG["num_workers"],
    pin_memory=True,
)

abl_val_loader = DataLoader(
    abl_val_dataset,
    batch_size=CONFIG["batch_size"],
    shuffle=False,
    num_workers=CONFIG["num_workers"],
```

```
    pin_memory=True,
)

ablation_results = []

# Train each ablation model for 10 epochs (budgeted), report BEST val
BAcc
for name, model_fn in ablation_models.items():
    print(f"\nTraining {name}...")
    model = model_fn().to(CONFIG["device"])

    samples_per_class = np.maximum(np.bincount(abl_train_labels,
minlength=CONFIG["num_classes"]), 1)
    criterion = ClassBalancedLoss(samples_per_class,
CONFIG["num_classes"])

    optimizer = AdamW(model.parameters(), lr=CONFIG["lr"],
weight_decay=CONFIG["weight_decay"])
    scheduler = CosineAnnealingWarmRestarts(optimizer, T_0=10,
T_mult=2)

    best_bacc, best_acc, best_f1 = -1.0, 0.0, 0.0

    for epoch in range(10):
        train_epoch(model, abl_train_loader, criterion, optimizer,
scheduler, CONFIG["device"], epoch + 1)
        _, val_acc, val_bacc, val_f1, _, _, _ = validate(model,
abl_val_loader, criterion, CONFIG["device"])

        if val_bacc > best_bacc:
            best_bacc, best_acc, best_f1 = val_bacc, val_acc, val_f1

    print(f"{name} (BEST): Acc={best_acc:.4f}, BAcc={best_bacc:.4f},
F1={best_f1:.4f}")

    ablation_results.append({
        "Model": name,
        "Accuracy_best": float(best_acc),
        "BalancedAcc_best": float(best_bacc),
        "MacroF1_best": float(best_f1),
        "epochs": 10,
        "protocol": "budgeted_single_fold_best_epoch"
    })

    del model
    torch.cuda.empty_cache()

abl_df = pd.DataFrame(ablation_results)
print("\nAblation Results (budgeted, single fold, best epoch):")
print(abl_df.to_string(index=False))
```

```python
try:
    wandb.log({"ablation_study": wandb.Table(dataframe=abl_df)})
except Exception as e:
    print("W&B log skipped:", e)
```

```
==================================================
ABLATION STUDY
==================================================

Training UHViT (Full)...
```


{"model_id":"bdb8bcd365574f6aaed388b208a2f36c","version_major":2,"version_minor":0}

{"model_id":"e5c753cc4bc645d59e2bede5b93f65d6","version_major":2,"version_minor":0}

{"model_id":"de88cc46e38445279b8a64f3bdd3b139","version_major":2,"version_minor":0}

{"model_id":"982829d0f5a64ae293a84fa6f7f73603","version_major":2,"version_minor":0}

{"model_id":"b7990d7a69d442e2bce2bf9b85e61602","version_major":2,"version_minor":0}

{"model_id":"0329af79dbf54e039320b8463c9ea529","version_major":2,"version_minor":0}

{"model_id":"1bb16dc903ed4ba4bc0fdec17a645c70","version_major":2,"version_minor":0}

{"model_id":"cb85ac9edbc149308c04fc9873d30314","version_major":2,"version_minor":0}

{"model_id":"0ece33d6b0384b5bac8c09cdf43ee91f","version_major":2,"version_minor":0}

{"model_id":"d91650ae10a446f2a375c81f952ea072","version_major":2,"version_minor":0}

{"model_id":"157d79b19c7449688ae231eba193769b","version_major":2,"version_minor":0}

{"model_id":"bda9aed004b8426dbf1d24c682a06831","version_major":2,"version_minor":0}

{"model_id":"bd9da9e4b1ce492bab263e28fc986749","version_major":2,"version_minor":0}

{"model_id":"c556467b93b04a13bca0c783114ae04b","version_major":2,"version_minor":0}

{"model_id":"fd7806f562c546269a1e0ff3215c2626","version_major":2,"version_minor":0}

{"model_id":"0a7375b022ba49a79df1bd628db30a05","version_major":2,"version_minor":0}

{"model_id":"92a9ff6c605d4b2bbc569faa2623f163","version_major":2,"version_minor":0}

{"model_id":"2558ec5d25ac4d309c4e64fcf1afa752","version_major":2,"version_minor":0}

{"model_id":"84dc0608dafd4197b8584c6aa4b929a1","version_major":2,"version_minor":0}

{"model_id":"8b4c1593fb1f4a21ad5aaa6e9a4c37d1","version_major":2,"version_minor":0}

UHViT (Full) (BEST): Acc=0.3306, BAcc=0.6317, F1=0.3917

Training Swin-T Only...

{"model_id":"6210c6048dd94c069008770858863d3f","version_major":2,"version_minor":0}

{"model_id":"a66b005f9f844aaba62ba4913e20c76f","version_major":2,"version_minor":0}

{"model_id":"752814178bcf4b6487fe80eb16885aac","version_major":2,"version_minor":0}

{"model_id":"30c31d85b71e48e7a31501a105e5d022","version_major":2,"version_minor":0}

{"model_id":"681f328e4f5b4374b7c5e72f9eb08058","version_major":2,"version_minor":0}

{"model_id":"22690b9dfc9a4e8594c9cc19b8ab581c","version_major":2,"version_minor":0}

{"model_id":"c1d41186f3474afc90e718977689a3a2","version_major":2,"version_minor":0}

{"model_id":"99d22a143f89471295c34ac9662bd7f5","version_major":2,"version_minor":0}

{"model_id":"9057e8bf0a0245e2a0ff96215fa48f3f","version_major":2,"version_minor":0}

{"model_id":"89968b5122fb4a4a903a5538d420cdfa","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^Exception ignored in: ^<function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>^^
^Traceback (most recent call last):
^  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
^^    ^^self._shutdown_workers()^
^  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
^^    ^if w.is_alive():^
^ ^ ^ ^ ^ ^ ^^
AssertionError^: ^can only test a child process^
^^^Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>^
^Traceback (most recent call last):
^  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
^^    ^self._shutdown_workers()

  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    assert self._parent_pid == os.getpid(), 'can only test a child process'    if w.is_alive():

                ^ ^ ^ ^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
```

```
is_alive
^    ^assert self._parent_pid == os.getpid(), 'can only test a child
process'^
^ ^  ^^ ^ ^ ^  ^ ^  ^^^^^^^^^^^^^^^^^^^^^^^^
^AssertionError^^: ^can only test a child process^^
^^^^Exception ignored in: ^^<function
_MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>^
^Traceback (most recent call last):
^  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
^^    ^self._shutdown_workers()^

  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
AssertionError:     can only test a child processif w.is_alive():

       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"d6ca1c59f3c04ee09b7ba709725258d7","version_major":2,"vers
ion_minor":0}

{"model_id":"f29f795ea6e741129eac8254bd88977a","version_major":2,"vers
ion_minor":0}

{"model_id":"f25cb1460ed04ffb907034ec4510e06b","version_major":2,"vers
ion_minor":0}

{"model_id":"3f0ff288852d415b860d04b7dda46d0e","version_major":2,"vers
ion_minor":0}

{"model_id":"ed5d4eb7e2e947708394d90db3a77e0f","version_major":2,"vers
ion_minor":0}

{"model_id":"d642002ea976423ebbb4a2fb02ba545d","version_major":2,"vers
ion_minor":0}

{"model_id":"6b7cf97e258d49c0938a0ac886d51392","version_major":2,"vers
ion_minor":0}

{"model_id":"5bcf71c692124b329cf9ac6079f8ed8c","version_major":2,"vers
ion_minor":0}
```

{"model_id":"35bea2c47d494fc6aae7274bc2f5cb47","version_major":2,"version_minor":0}

{"model_id":"354d5100791f4ad79b1f9ba8beaddd6d","version_major":2,"version_minor":0}

Swin-T Only (BEST): Acc=0.4227, BAcc=0.6745, F1=0.4213

Training EfficientNet Only...

{"model_id":"c63290b71da0484593271f5e57f4aff0","version_major":2,"version_minor":0}

{"model_id":"f516e42af7cb4eaa918d1296a04d4169","version_major":2,"version_minor":0}

{"model_id":"76f129fab57547109a1617ce89fd8192","version_major":2,"version_minor":0}

{"model_id":"9b30f7d4d5694ee58b7acd7ab44db6dc","version_major":2,"version_minor":0}

{"model_id":"e8d74400d7be4a4f83c8d954a7256300","version_major":2,"version_minor":0}

{"model_id":"6767506f411c401db0fecc84364ae330","version_major":2,"version_minor":0}

{"model_id":"3ddd6b0d3b8446cabb9948add6e828da","version_major":2,"version_minor":0}

{"model_id":"3e5496e46f0f4f84a72da6c358ea3409","version_major":2,"version_minor":0}

{"model_id":"1c22de1b3435407a86f4a9d507b6d7a9","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
```

```
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"a4969c7ad1ef49178323d17f6adda3cf","version_major":2,"vers
ion_minor":0}

{"model_id":"67b0adf31a2f4e63978d62f820868377","version_major":2,"vers
ion_minor":0}

{"model_id":"04ebaaa76e1544b4b05253136d9ca3da","version_major":2,"vers
ion_minor":0}

{"model_id":"520bb5da19044bb682bfedacd50d5774","version_major":2,"vers
ion_minor":0}
```


```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
```

```
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"50e23337b3394f3ea5371a78c50d6742","version_major":2,"vers
ion_minor":0}

{"model_id":"b77bdb4325f146b0843af76bb3636544","version_major":2,"vers
ion_minor":0}

{"model_id":"5d3561dddb8e4c879fc058fd5c74858d","version_major":2,"vers
ion_minor":0}

{"model_id":"5e367f8df1f346649c8a874a10c1448b","version_major":2,"vers
ion_minor":0}

{"model_id":"143ca2f67c7645f285732e00e96fbb62","version_major":2,"vers
ion_minor":0}

{"model_id":"9827ea060c554a2488ad7042a32a1dbc","version_major":2,"vers
ion_minor":0}

{"model_id":"9760feb0d34244e892fe994635e26532","version_major":2,"vers
ion_minor":0}
```


```
EfficientNet Only (BEST): Acc=0.3570, BAcc=0.6232, F1=0.3358

Training Concat Fusion...
```


```
{"model_id":"82bd8707a99643a38ae37e58de13b35a","version_major":2,"vers
ion_minor":0}

{"model_id":"bc256d2c411d451298fdd13ddbe1b41e","version_major":2,"vers
ion_minor":0}

{"model_id":"51bcabb1648140c58cd5060b8bc9a72d","version_major":2,"vers
ion_minor":0}

{"model_id":"2c65ce058c7445d49a418d32cbe75d70","version_major":2,"vers
ion_minor":0}

{"model_id":"5d4cd1455e5d4387a66ffee5b2de6e92","version_major":2,"vers
ion_minor":0}

{"model_id":"27895fda062847409ee1498cacdd55ec","version_major":2,"vers
ion_minor":0}

{"model_id":"24d92639cfbd4a48a5ed421e5bb872fd","version_major":2,"vers
ion_minor":0}

{"model_id":"02aa1fbf33ed4a54ba013a60f03234b4","version_major":2,"vers
ion_minor":0}
```

{"model_id":"cfbe494b6ee04f70901630717fd22b60","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
IOStream.flush timed out
IOStream.flush timed out
```

{"model_id":"107826bc837c41769aa177f4ea576922","version_major":2,"version_minor":0}

{"model_id":"0f5765c4c8cb425aba604f74da251cc9","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1664, in __del__
    self._shutdown_workers()
  File "/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child process'
```

```
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
IOStream.flush timed out
IOStream.flush timed out
```


```
{"model_id":"28561d24f7634cc7933fa6351b66b413","version_major":2,"vers
ion_minor":0}

{"model_id":"d4256a7767d740c4b1dedb222779e2b9","version_major":2,"vers
ion_minor":0}

{"model_id":"027681bfe1aa4442959dc136e1516d26","version_major":2,"vers
ion_minor":0}

{"model_id":"ef2013775380454697c7b94a6eab1e1a","version_major":2,"vers
ion_minor":0}

{"model_id":"4b19c130b0a14a4ba615d799b253bbc7","version_major":2,"vers
ion_minor":0}

{"model_id":"e14c3b8484db4a92aa86157e64f48821","version_major":2,"vers
ion_minor":0}

{"model_id":"19268ae743854181b4db30aade11e86c","version_major":2,"vers
ion_minor":0}

{"model_id":"e9115fc5834a42148d74119d6e09ced0","version_major":2,"vers
ion_minor":0}

{"model_id":"332a8336dbf242209d18a80aaf0e6acb","version_major":2,"vers
ion_minor":0}
```


```
Concat Fusion (BEST): Acc=0.4579, BAcc=0.6884, F1=0.4658

Ablation Results (budgeted, single fold, best epoch):
            Model  Accuracy_best  BalancedAcc_best  MacroF1_best
epochs                          protocol
     UHViT (Full)       0.330570          0.631712      0.391707
10 budgeted_single_fold_best_epoch
      Swin-T Only       0.422735          0.674525      0.421324
10 budgeted_single_fold_best_epoch
EfficientNet Only       0.357016          0.623233      0.335812
10 budgeted_single_fold_best_epoch
    Concat Fusion       0.457865          0.688450      0.465830
10 budgeted_single_fold_best_epoch
```

```
# CELL 22: SOTA COMPARISON (FIXED: proper samples_per_class + BEST
epoch)

print("\n" + "="*50 + "\nSOTA COMPARISON (budgeted single-fold)\n" +
"="*50)
```

```
# This cell assumes you already ran the FIXED ablation cell (CELL 21)
that created:
#   abl_train_loader, abl_val_loader, abl_train_labels
# If not, run CELL 21 first.

assert "abl_train_loader" in globals() and "abl_val_loader" in
globals() and "abl_train_labels" in globals(), \
    "Run the fixed Ablation Study cell (CELL 21) first."

# Define criterion inputs correctly for this fold
samples_per_class = np.maximum(np.bincount(abl_train_labels,
minlength=CONFIG["num_classes"]), 1)
criterion = ClassBalancedLoss(samples_per_class,
CONFIG["num_classes"])

sota_results = []

for sota_name in SOTA_MODELS:
    print(f"\nTraining SOTA baseline: {sota_name} ...")
    try:
        model = create_sota_model(sota_name,
CONFIG["num_classes"]).to(CONFIG["device"])
        optimizer = AdamW(model.parameters(), lr=CONFIG["lr"],
weight_decay=CONFIG["weight_decay"])
        scheduler = CosineAnnealingWarmRestarts(optimizer, T_0=10,
T_mult=2)

        best_bacc, best_acc, best_f1 = -1.0, 0.0, 0.0

        for epoch in range(10):
            train_epoch(model, abl_train_loader, criterion, optimizer,
scheduler, CONFIG["device"], epoch + 1)
            _, val_acc, val_bacc, val_f1, _, _, _ = validate(model,
abl_val_loader, criterion, CONFIG["device"])

            if val_bacc > best_bacc:
                best_bacc, best_acc, best_f1 = val_bacc, val_acc,
val_f1

        print(f"{sota_name} (BEST): Acc={best_acc:.4f},
BAcc={best_bacc:.4f}, F1={best_f1:.4f}")

        sota_results.append({
            "Model": sota_name,
            "Accuracy_best": float(best_acc),
            "BalancedAcc_best": float(best_bacc),
            "MacroF1_best": float(best_f1),
            "epochs": 10,
            "protocol": "budgeted_single_fold_best_epoch"
        })
```

```
        del model
        torch.cuda.empty_cache()

    except Exception as e:
        print(f"Error with {sota_name}: {e}")

# Add your model result (from 5-fold CV means) as "Ours"
# These variables accs/baccs/f1s exist in your CV summary cell; if
not, you can skip or hardcode.
if "accs" in globals() and "baccs" in globals() and "f1s" in
globals():
    sota_results.append({
        "Model": "UHViT (Ours, 5-fold CV mean)",
        "Accuracy_best": float(np.mean(accs)),
        "BalancedAcc_best": float(np.mean(baccs)),
        "MacroF1_best": float(np.mean(f1s)),
        "epochs": CONFIG.get("epochs", 30),
        "protocol": "5-fold_CV_mean"
    })

sota_df = pd.DataFrame(sota_results).sort_values("BalancedAcc_best",
ascending=False)
print("\nSOTA Comparison Results:")
print(sota_df.to_string(index=False))

try:
    wandb.log({"sota_comparison": wandb.Table(dataframe=sota_df)})
except Exception as e:
    print("W&B log skipped:", e)


==================================================
SOTA COMPARISON (budgeted single-fold)
==================================================

Training SOTA baseline: resnet50 ...

{"model_id":"711d7106498848a4992a3745e4b52a6c","version_major":2,"vers
ion_minor":0}

{"model_id":"b39b6ff57fb44626893f362d7ff26d96","version_major":2,"vers
ion_minor":0}

{"model_id":"3174b0812f854bbaaa6f942cca2b44db","version_major":2,"vers
ion_minor":0}

{"model_id":"04c54cd648694523b134f032594eba59","version_major":2,"vers
ion_minor":0}
```

{"model_id":"4ea1bce4717f41e682f82b08c22b2f88","version_major":2,"version_minor":0}

{"model_id":"837bb7dbc89843d4b208a37e7da4e3dc","version_major":2,"version_minor":0}

{"model_id":"d6c203bb5cfd4f6fb58cc1fba095a588","version_major":2,"version_minor":0}

{"model_id":"aa23ec0f009c4b03bb64d8edb4b96834","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
```

```
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"a1e5f1d3e2b9419894f0e77b93ec9b64","version_major":2,"vers
ion_minor":0}

{"model_id":"9c4e2c2ef9ea4b59b24d0f3634418380","version_major":2,"vers
ion_minor":0}

{"model_id":"e5b06ab5599d424e956f61d565bb5c46","version_major":2,"vers
ion_minor":0}

{"model_id":"ec98674f40594a2f86f138796a6329a2","version_major":2,"vers
ion_minor":0}

{"model_id":"3e9db4bc7bb74be883fadb2f0c1cff5b","version_major":2,"vers
ion_minor":0}

{"model_id":"62c7e7a454ed498f9ecd18ede275e170","version_major":2,"vers
ion_minor":0}

{"model_id":"ed87876dd52e4d0eb4c46e413c3b9a13","version_major":2,"vers
ion_minor":0}

{"model_id":"28f9b40dca124d8d851a3ac946214afb","version_major":2,"vers
ion_minor":0}

{"model_id":"c181882d097947aa8d34dac993212ca8","version_major":2,"vers
ion_minor":0}
```


```
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
```

```
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
Exception ignored in:     <function
_MultiProcessingDataLoaderIter.__del__ at 0x728d383977e0>if
w.is_alive():
Traceback (most recent call last):

  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
      self._shutdown_workers()
   File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
      if w.is_alive():
^ ^ ^ ^ ^ ^ ^^^^^^^^^^
^  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
^    ^assert self._parent_pid == os.getpid(), 'can only test a child
process'^
^ ^ ^
   File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
      assert self._parent_pid == os.getpid(), 'can only test a child
process'
       ^ ^ ^^ ^ ^ ^^ ^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
^AssertionError^: ^can only test a child process^
^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
```


```
{"model_id":"d23df4df2c0a46c1a9539494bc1265e7","version_major":2,"vers
ion_minor":0}

{"model_id":"5a241c6a8f2a44dba21595db9e67d341","version_major":2,"vers
ion_minor":0}

{"model_id":"f6e7da158387493fab6a240d8a856465","version_major":2,"vers
ion_minor":0}

{"model_id":"1dd4278479a9450782019d3c120a109f","version_major":2,"vers
ion_minor":0}
```


```
resnet50 (BEST): Acc=0.1508, BAcc=0.4836, F1=0.1594

Training SOTA baseline: efficientnet_b3 ...
```


```
{"model_id":"eaf24a8f7d3044298667d00799761086","version_major":2,"vers
ion_minor":0}

{"model_id":"fa37a84159194781b621fccb22bdd8bc","version_major":2,"vers
ion_minor":0}
```

{"model_id":"19a2b00676a4495ba1cb08dcf1ed56f2","version_major":2,"version_minor":0}

{"model_id":"b310270940934b85b6f3129bb017f9e7","version_major":2,"version_minor":0}

{"model_id":"c778bca6e081475488168a07c9df277c","version_major":2,"version_minor":0}

{"model_id":"ab7b4471f18749258bf8fc1218b7f660","version_major":2,"version_minor":0}

{"model_id":"77f51ac968fb40fb92e9c919ec95d1aa","version_major":2,"version_minor":0}

{"model_id":"975808a9017a4799b966f8b7de29f0f5","version_major":2,"version_minor":0}

{"model_id":"9b51295a0d2545ceb5cebf9b17f230e0","version_major":2,"version_minor":0}

{"model_id":"e398ae9c1a164a5c9f1cc928014af4ed","version_major":2,"version_minor":0}

{"model_id":"d5c807a82f5045c79aabfdb2ecf7d07b","version_major":2,"version_minor":0}

{"model_id":"30e0456d7a234ce8a02549e7b779ba02","version_major":2,"version_minor":0}

{"model_id":"1db58361e4c141b6b518f76f1eba6f96","version_major":2,"version_minor":0}

{"model_id":"831119c4b81f43528286938f99c3bf87","version_major":2,"version_minor":0}

{"model_id":"2316d843ab874976b3c368ed6fed56c3","version_major":2,"version_minor":0}

{"model_id":"e197dc42ffae4121b8ab0b6aee1a1f0a","version_major":2,"version_minor":0}

{"model_id":"55d6a14df6704b58861aead447d6d7b7","version_major":2,"version_minor":0}

{"model_id":"68cbdaad7d1d49a69b1647955d10bd43","version_major":2,"version_minor":0}

{"model_id":"a0eb6c03b3aa4b2ca5220c3fbac0efa4","version_major":2,"version_minor":0}

{"model_id":"edeb0788819d4acf9ff093de099e1e37","version_major":2,"version_minor":0}

```
efficientnet_b3 (BEST): Acc=0.4894, BAcc=0.6453, F1=0.4352

Training SOTA baseline: vit_base_patch16_224 ...
```


```
{"model_id":"ff8b0603c31d4dd8814a78fa1c1d18ee","version_major":2,"vers
ion_minor":0}

{"model_id":"f0c9b4bd94884da0aa5797d363a45515","version_major":2,"vers
ion_minor":0}

{"model_id":"0e0a29f183cd415d8cec212522cfa565","version_major":2,"vers
ion_minor":0}

{"model_id":"44d23f5430fd4da68e66dd3601b7aa79","version_major":2,"vers
ion_minor":0}

{"model_id":"7513050f0f3546dc871ae6ac1aa65cf9","version_major":2,"vers
ion_minor":0}

{"model_id":"2cf7225dec0f4e888053acae3a2a1c5d","version_major":2,"vers
ion_minor":0}

{"model_id":"a46ad11a3df346c5bd6ba8cfacecebc6","version_major":2,"vers
ion_minor":0}

{"model_id":"39ab17c900b4458493b5520e22b445a6","version_major":2,"vers
ion_minor":0}

{"model_id":"6a9e686e30044f33bcd4c0232de4f47f","version_major":2,"vers
ion_minor":0}

{"model_id":"95847f76ad06401fa9ff250671900c83","version_major":2,"vers
ion_minor":0}

{"model_id":"9b94befdb22c40508d27293949ac9669","version_major":2,"vers
ion_minor":0}

{"model_id":"79fb3475d8b4424a9a4316c28a2c5ea6","version_major":2,"vers
ion_minor":0}

{"model_id":"71a5fafe7f37450a895bb8929d1ab591","version_major":2,"vers
ion_minor":0}

{"model_id":"dccb8e6021c04497bb33b8294f0079a2","version_major":2,"vers
ion_minor":0}

{"model_id":"32787bac0405419c869c5874990bdbdc","version_major":2,"vers
ion_minor":0}

{"model_id":"a8fcc87f201849d886732647ec522bfa","version_major":2,"vers
ion_minor":0}
```

{"model_id":"40a4c57c898d43808955e0650eda0911","version_major":2,"version_minor":0}

{"model_id":"d3c167ab9c7a469d809e7f373d204791","version_major":2,"version_minor":0}

{"model_id":"0632a466f0bc4ce2a874dc6594487761","version_major":2,"version_minor":0}

{"model_id":"e4f2c11c019840f0a96201ffed857c04","version_major":2,"version_minor":0}

{"model_id":"866937e24e354ec288e5ef3e08580534","version_major":2,"version_minor":0}

```
vit_base_patch16_224 (BEST): Acc=0.3949, BAcc=0.6142, F1=0.3561

Training SOTA baseline: swin_tiny_patch4_window7_224 ...
```

{"model_id":"de50e2b85cb545f495a20e278f94ce0d","version_major":2,"version_minor":0}

{"model_id":"44b05fe504474baba236b25aec910100","version_major":2,"version_minor":0}

```
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
```

```
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"70f5277e4f854492b8f3d056297d6588","version_major":2,"vers
ion_minor":0}

{"model_id":"086625e832d04f19aa9a81a39bc7f550","version_major":2,"vers
ion_minor":0}

{"model_id":"40a087b7841741ef96e75ab7edf8fd88","version_major":2,"vers
ion_minor":0}

{"model_id":"b6ce5b526f92457eb99327e649f1e691","version_major":2,"vers
ion_minor":0}

{"model_id":"bc7fee5d9ca0460992ebb76270cb503e","version_major":2,"vers
ion_minor":0}

{"model_id":"d28dcc749c244a96beed3a6a2349852e","version_major":2,"vers
ion_minor":0}

{"model_id":"3a8633b1b099442883a88b18481d59ea","version_major":2,"vers
ion_minor":0}

{"model_id":"202a118add304d4db4e98cc711cae64f","version_major":2,"vers
ion_minor":0}

{"model_id":"210d041060174e0c88af990550f5490c","version_major":2,"vers
ion_minor":0}
```


```
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
```

```
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```

```
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
IOStream.flush timed out
```


```
{"model_id":"60dfb792cc8541b09c1d4d788d7ce297","version_major":2,"vers
ion_minor":0}

{"model_id":"4ec9b974ab0d4b468d7db984b5319b9b","version_major":2,"vers
ion_minor":0}

{"model_id":"0d1a8bb1f52349bea3bddcfafbac8386","version_major":2,"vers
ion_minor":0}

{"model_id":"f5a79fe4cd284c51be4c7079eb35519a","version_major":2,"vers
ion_minor":0}

{"model_id":"ae298268127148a7b9cef99c89483fb9","version_major":2,"vers
ion_minor":0}

{"model_id":"4007495c336f43eeb98b079df87ef59d","version_major":2,"vers
ion_minor":0}

{"model_id":"e8f573d435b6493882e52b18587fe758","version_major":2,"vers
ion_minor":0}

{"model_id":"da1ffb42cfe84f7fafab4a97b3419d5b","version_major":2,"vers
ion_minor":0}

{"model_id":"e35ad8cb5a8b40b097ab11c205e61c32","version_major":2,"vers
ion_minor":0}
```


```
swin_tiny_patch4_window7_224 (BEST): Acc=0.4470, BAcc=0.6940,
F1=0.4667

Training SOTA baseline: convnext_tiny ...
```


```
{"model_id":"7dcae546aa2944fb9d248ede3d3089cd","version_major":2,"vers
ion_minor":0}

{"model_id":"73cab53b2e2e49079fcb023bf924ca9c","version_major":2,"vers
ion_minor":0}
```

{"model_id":"2200330a87fb419aadfbbb5d9331df4a","version_major":2,"version_minor":0}

{"model_id":"7d2314591cbe4407a59d631c6c4b9b9f","version_major":2,"version_minor":0}

{"model_id":"825014f6332844b88b6982142c7693ef","version_major":2,"version_minor":0}

{"model_id":"551aa250415e47f2b789527aff35bbd6","version_major":2,"version_minor":0}

{"model_id":"6de18b6bc4bc44359c7520a4eefbba04","version_major":2,"version_minor":0}

{"model_id":"4b8af1f636c64dc3bba68ece26707444","version_major":2,"version_minor":0}

{"model_id":"cfb2a9a36c954068a9e5357cd1fdd492","version_major":2,"version_minor":0}

{"model_id":"487534be09e340378d903879b092b065","version_major":2,"version_minor":0}

{"model_id":"dc227a9c0b294fb0b4f1f0813a263ee9","version_major":2,"version_minor":0}

{"model_id":"826f5e27e887452eba4f36eb67b744a6","version_major":2,"version_minor":0}

{"model_id":"ec0a10d803a844268a8aace0198e54ea","version_major":2,"version_minor":0}

{"model_id":"c7c3ada9db1e4f1ba841f793093f5868","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
```

```
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
```

```
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"321cd9a54be740bca1b5f6ff640c3d64","version_major":2,"vers
ion_minor":0}

{"model_id":"320a72e156894ab08b5f40994044b213","version_major":2,"vers
ion_minor":0}
```

{"model_id":"cf0907df07a04955a1b5092e8fd42ea5","version_major":2,"version_minor":0}

{"model_id":"f6814b00820445d7b804e08524eb480b","version_major":2,"version_minor":0}

{"model_id":"47b84fb39f5e4c18b4a4f31f0a6af7fb","version_major":2,"version_minor":0}

{"model_id":"92f521e9e2c24db792ff32ec2bffbbd6","version_major":2,"version_minor":0}

```
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
Exception ignored in: <function _MultiProcessingDataLoaderIter.__del__
at 0x728d383977e0>
```

```
Traceback (most recent call last):
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1664, in __del__
    self._shutdown_workers()
  File
"/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.p
y", line 1647, in _shutdown_workers
    if w.is_alive():
       ^^^^^^^^^^^^
  File "/usr/lib/python3.12/multiprocessing/process.py", line 160, in
is_alive
    assert self._parent_pid == os.getpid(), 'can only test a child
process'
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError: can only test a child process
```


```
{"model_id":"471a976a6641457cbe9d1976b65b32f1","version_major":2,"vers
ion_minor":0}
```


```
convnext_tiny (BEST): Acc=0.4654, BAcc=0.6753, F1=0.4874

Training SOTA baseline: densenet121 ...
```


```
{"model_id":"1c92d92108ba4b51beb9661dc3b60633","version_major":2,"vers
ion_minor":0}

{"model_id":"3a7ef85f09dc472187f564833a729a8c","version_major":2,"vers
ion_minor":0}

{"model_id":"ae25a5945c77469086f717b43f864f89","version_major":2,"vers
ion_minor":0}

{"model_id":"24e45d49aaef42da860956943160964b","version_major":2,"vers
ion_minor":0}

{"model_id":"186959ce4f2a46da9c211f493359a1d6","version_major":2,"vers
ion_minor":0}

{"model_id":"046faa774c324da1b28540a0e4c73985","version_major":2,"vers
ion_minor":0}

{"model_id":"1894aa286d67435e9f0fb4146002a2ac","version_major":2,"vers
ion_minor":0}

{"model_id":"13ea1a8445ba48068689d1489ef4d5cb","version_major":2,"vers
ion_minor":0}

{"model_id":"b637fcbc36304cacadb7f6330e36163a","version_major":2,"vers
ion_minor":0}
```

{"model_id":"77a2ae943d9b462291918ea3aecc92fc","version_major":2,"version_minor":0}

{"model_id":"e7d676131ba44b3ca6c4c363be1e1da9","version_major":2,"version_minor":0}

{"model_id":"d770d693af42448e9eb8b7f64568dd9a","version_major":2,"version_minor":0}

{"model_id":"ff4ffb05551442aba2a54c679f7f4658","version_major":2,"version_minor":0}

{"model_id":"a1848091f9384239b112569d1f594a5e","version_major":2,"version_minor":0}

{"model_id":"450d64288c9446bbb5ead57530facecd","version_major":2,"version_minor":0}

{"model_id":"95ab204df636445ab5128f8785227794","version_major":2,"version_minor":0}

{"model_id":"af0154412ab845eeb554a2a8c3cfe4de","version_major":2,"version_minor":0}

{"model_id":"40f22c90466b4222814db2024867a043","version_major":2,"version_minor":0}

{"model_id":"80fde904e57a47e69e3cdc1725eacbfb","version_major":2,"version_minor":0}

{"model_id":"cea3270d3e2c47d0b3ab2734f51db89c","version_major":2,"version_minor":0}

{"model_id":"7ccf2401a76247bea09c414ce275e755","version_major":2,"version_minor":0}

densenet121 (BEST): Acc=0.3689, BAcc=0.6107, F1=0.3293

SOTA Comparison Results:

Model	Accuracy_best	BalancedAcc_best	MacroF1_best	epochs	protocol
UHViT (Ours, 5-fold CV mean)	0.505074	0.733522	0.581658	30	5-fold_CV_mean
swin_tiny_patch4_window7_224	0.447010	0.694033	0.466748	10	budgeted_single_fold_best_epoch
convnext_tiny	0.465364	0.675253	0.487435	10	budgeted_single_fold_best_epoch
efficientnet_b3	0.489441	0.645291	0.435196	10	budgeted_single_fold_best_epoch
vit_base_patch16_224	0.394908	0.614220	0.356067	10	budgeted_single_fold_best_epoch
densenet121	0.368857	0.610687			

```
0.329264      10 budgeted_single_fold_best_epoch
                    resnet50       0.150780          0.483578
0.159402      10 budgeted_single_fold_best_epoch

# CELL 23: CROSS-DATASET VALIDATION (HAM10000)
print("\n" + "="*50 + "\nCROSS-DATASET VALIDATION (HAM10000)\n" +
"="*50)
ham_base = Path(CONFIG['data_dir']) / 'ham10000'
ham_csv_files = list(ham_base.rglob('*metadata*.csv'))
if not ham_csv_files: ham_csv_files = list(ham_base.rglob('*.csv'))
ham_csv = ham_csv_files[0]
ham_img_dirs = [d for d in ham_base.iterdir() if d.is_dir() and 'HAM'
in d.name]
if not ham_img_dirs: ham_img_dirs = [ham_base]
ham_dataset = HAM10000Dataset(ham_csv, ham_img_dirs,
get_val_transforms(CONFIG['img_size']))
ham_loader = DataLoader(ham_dataset, batch_size=CONFIG['batch_size'],
shuffle=False, num_workers=CONFIG['num_workers'])
best_model = fold_results[0]['model']; best_model.eval()
ham_preds, ham_labels, ham_probs = [], [], []
with torch.no_grad():
    for images, labels in tqdm(ham_loader, desc='HAM10000 Eval'):
        images = images.to(CONFIG['device']); outputs =
best_model(images)
        ham_probs.extend(F.softmax(outputs, dim=1).cpu().numpy());
ham_preds.extend(outputs.argmax(dim=1).cpu().numpy());
ham_labels.extend(labels.numpy())
ham_preds, ham_labels, ham_probs = np.array(ham_preds),
np.array(ham_labels), np.array(ham_probs)
ham_acc, ham_bacc, ham_f1 = accuracy_score(ham_labels, ham_preds),
balanced_accuracy_score(ham_labels, ham_preds), f1_score(ham_labels,
ham_preds, average='macro')
print(f"\nHAM10000 Results:\nAccuracy: {ham_acc:.4f}\nBalanced
Accuracy: {ham_bacc:.4f}\nMacro F1: {ham_f1:.4f}")
plot_confusion_matrix(ham_labels, ham_preds, CONFIG['class_names'],
f"{CONFIG['output_dir']}/ham10000_cm.png")
wandb.log({'cross_dataset/ham_acc': ham_acc, 'cross_dataset/ham_bacc':
ham_bacc, 'cross_dataset/ham_f1': ham_f1})



==================================================
CROSS-DATASET VALIDATION (HAM10000)
==================================================

{"model_id":"d997ead33cc54dc2a9996e4d941a8ad6","version_major":2,"vers
ion_minor":0}


HAM10000 Results:
```

```
Accuracy: 0.5255
Balanced Accuracy: 0.6976
Macro F1: 0.4359

# CELL 23B: EXTERNAL VALIDATION (DermaMNIST from MedMNIST) - public
direct download

import numpy as np
import torch
import torch.nn.functional as F
from torch.utils.data import DataLoader, Dataset
from PIL import Image
import medmnist
from medmnist import INFO
from sklearn.metrics import accuracy_score, balanced_accuracy_score,
f1_score, roc_auc_score

derma_info = INFO["dermamnist"]
DermaMNIST = getattr(medmnist, derma_info["python_class"])

# DermaMNIST: 7-class dermoscopy dataset (different label set than
ISIC 8-class)
# We use it as an EXTERNAL distribution test by reporting:
#   - Top-1 confidence distribution
#   - OOD-style metrics / uncertainty (optional)
#   - And/or map to MEL vs NON-MEL if you want (DermaMNIST doesn't
have MEL as a direct class label).
# For IF≈5, it’s acceptable to report "cross-dataset robustness" with
caveats.

class DermaMNISTWrapper(Dataset):
    def __init__(self, split="test", transform=None):
        self.data = DermaMNIST(split=split, download=True)
        self.transform = transform

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        img, y = self.data[idx]           # img is PIL, y is int or
array
        if isinstance(y, np.ndarray):
            y = int(y.squeeze())
        else:
            y = int(y)
        img = np.array(img.convert("RGB"))
        if self.transform:
            img = self.transform(image=img)["image"]
        return img, y
```

```
# Use val transforms (resize + normalize) for compatibility
derma_test = DermaMNISTWrapper(split="test",
transform=get_val_transforms(CONFIG["img_size"]))
derma_loader = DataLoader(derma_test, batch_size=CONFIG["batch_size"],
shuffle=False, num_workers=CONFIG["num_workers"])

best_model = fold_results[0]["model"]
best_model.eval()

all_conf = []
all_entropy = []

with torch.no_grad():
    for images, _ in tqdm(derma_loader, desc="DermaMNIST External
Eval"):
        images = images.to(CONFIG["device"])
        probs8 = F.softmax(best_model(images), dim=1)          # ISIC
8-class outputs
        conf = probs8.max(dim=1).values.detach().cpu().numpy() #
confidence of predicted ISIC class
        ent = (-probs8 * (probs8.clamp_min(1e-
9)).log()).sum(dim=1).detach().cpu().numpy()

        all_conf.extend(conf.tolist())
        all_entropy.extend(ent.tolist())

all_conf = np.array(all_conf)
all_entropy = np.array(all_entropy)

report = {
    "DermaMNIST_external_n": int(len(all_conf)),
    "mean_confidence": float(all_conf.mean()),
    "median_confidence": float(np.median(all_conf)),
    "mean_entropy": float(all_entropy.mean()),
    "median_entropy": float(np.median(all_entropy)),
}

print("DermaMNIST External Robustness Report:")
for k, v in report.items():
    print(f"{k}: {v}")

wandb.run.summary["external_note"] = (
    "External dataset label space differs from ISIC2019; "
    "we report robustness metrics (confidence/entropy) under
distribution shift."
)
wandb.log({"external/note": "External dataset label space differs; we
report robustness metrics (confidence/entropy) under distribution
shift."})
```

```
100%|██████████| 19.7M/19.7M [00:01<00:00, 15.0MB/s]

{"model_id":"f035db2de7ea483da7681df1bc253e8b","version_major":2,"vers
ion_minor":0}

DermaMNIST External Robustness Report:
DermaMNIST_external_n: 2005
mean_confidence: 0.6616833858507827
median_confidence: 0.6399524807929993
mean_entropy: 0.9612262350551208
median_entropy: 1.0979413986206055

# CELL 23C: DermaMNIST W&B logging + note (append after 23B)

import pandas as pd

# 'report' should exist from CELL 23B
wandb.log({"external/dermamnist_report":
wandb.Table(dataframe=pd.DataFrame([report]))})

wandb.run.summary["external_note"] = (
    "External dataset label space differs from ISIC2019; "
    "we report robustness metrics (confidence/entropy) under
distribution shift."
)
wandb.log({
    "external/note": "External dataset label space differs; we report
robustness metrics (confidence/entropy) under distribution shift."
})
print("✓ Logged DermaMNIST robustness report + note to W&B")

✓ Logged DermaMNIST robustness report + note to W&B

# CELL 24: COMPUTATIONAL ANALYSIS
print("\n" + "="*50 + "\nCOMPUTATIONAL ANALYSIS\n" + "="*50)
comp_models = {'UHViT': lambda: UHViT(CONFIG['num_classes'],
CONFIG['dropout_rate']), 'Swin-T': lambda:
SwinOnly(CONFIG['num_classes'], CONFIG['dropout_rate']),
'EfficientNet-B3': lambda: EfficientNetOnly(CONFIG['num_classes'],
CONFIG['dropout_rate']), 'ResNet50': lambda:
create_sota_model('resnet50', CONFIG['num_classes']), 'ViT-Base':
lambda: create_sota_model('vit_base_patch16_224',
CONFIG['num_classes'])}
comp_df = computational_analysis(comp_models, CONFIG['device']);
print("\nComputational Analysis:\n" + comp_df.to_string(index=False))



==================================================
COMPUTATIONAL ANALYSIS
==================================================
```

```
Analyzing UHViT...
Analyzing Swin-T...
Analyzing EfficientNet-B3...
Analyzing ResNet50...
Analyzing ViT-Base...

Computational Analysis:
          Model  Params (M)  FLOPs (G)  Time (ms)
          UHViT   40.580266  10.712108  19.199839
         Swin-T   27.527042   8.760381   8.547678
EfficientNet-B3   10.711600   1.945989   9.249110
       ResNet50   23.524424   8.260809   4.372694
       ViT-Base   85.804808  24.033695   4.171274
```

```python
# CELL 25: SAVE FINAL RESULTS
results_summary = {'UHViT_5Fold_CV': {'Accuracy':
f"{np.mean(accs):.4f} ± {np.std(accs):.4f}", 'Balanced_Accuracy':
f"{np.mean(baccs):.4f} ± {np.std(baccs):.4f}", 'Macro_F1':
f"{np.mean(f1s):.4f} ± {np.std(f1s):.4f}", 'CI_95_Accuracy':
f"[{acc_lo:.4f}, {acc_hi:.4f}]", 'CI_95_Balanced_Acc':
f"[{bacc_lo:.4f}, {bacc_hi:.4f}]", 'CI_95_Macro_F1': f"[{f1_lo:.4f},
{f1_hi:.4f}]"}, 'Cross_Dataset_HAM10000': {'Accuracy':
f"{ham_acc:.4f}", 'Balanced_Accuracy': f"{ham_bacc:.4f}", 'Macro_F1':
f"{ham_f1:.4f}"}}
with open(f"{CONFIG['output_dir']}/results_summary.json", 'w') as f:
json.dump(results_summary, f, indent=2)
fold_df = pd.DataFrame([{'Fold': i+1, 'Accuracy': r['acc'],
'Balanced_Acc': r['bacc'], 'Macro_F1': r['f1']} for i, r in
enumerate(fold_results)])
fold_df.to_csv(f"{CONFIG['output_dir']}/fold_results.csv",
index=False)
print(f"\n{'='*50}\nTRAINING COMPLETE!\n{'='*50}")
print(f"\nResults saved to: {CONFIG['output_dir']}\nCheckpoints saved
to: {CONFIG['checkpoint_dir']}")
print(f"\nFinal Results:\n  Balanced Accuracy: {np.mean(baccs):.4f} ±
{np.std(baccs):.4f}\n  Macro F1: {np.mean(f1s):.4f} ±
{np.std(f1s):.4f}\n  Cross-Dataset (HAM10000) BAcc: {ham_bacc:.4f}")
wandb.finish()
print("\n✓ All done! Check W&B for full logs.")
```

```
==================================================
TRAINING COMPLETE!
==================================================

Results saved to: ./outputs
Checkpoints saved to: ./checkpoints

Final Results:
  Balanced Accuracy: 0.7335 ± 0.0108
```

```
  Macro F1: 0.5817 ± 0.0129
  Cross-Dataset (HAM10000) BAcc: 0.6976

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>


✓ All done! Check W&B for full logs.
```

```python
# CELL 20C: MEL additional operating points (Youden J + PR-AUC)

import numpy as np
import pandas as pd
from sklearn.metrics import roc_curve, confusion_matrix,
average_precision_score, precision_recall_curve, roc_auc_score

MEL_CLASS = CONFIG["class_names"].index("MEL")

def summarize_threshold(mel_true, mel_score, threshold):
    mel_pred = (mel_score >= threshold).astype(int)
    tn, fp, fn, tp = confusion_matrix(mel_true, mel_pred).ravel()
    sens = tp / (tp + fn + 1e-9)
    spec = tn / (tn + fp + 1e-9)
    prec = tp / (tp + fp + 1e-9)
    f1 = 2 * prec * sens / (prec + sens + 1e-9)
    return dict(
        threshold=float(threshold),
        sensitivity=float(sens),
        specificity=float(spec),
        precision=float(prec),
        f1=float(f1),
        tp=int(tp), fp=int(fp), tn=int(tn), fn=int(fn)
    )

# Pooled CV arrays should exist from CELL 20B:
# all_y_true, all_y_prob
mel_true = (all_y_true == MEL_CLASS).astype(int)
mel_score = all_y_prob[:, MEL_CLASS]

# AUROC + PR-AUC (PR-AUC is informative for imbalanced MEL)
mel_auroc = roc_auc_score(mel_true, mel_score) if 0 < mel_true.sum() <
len(mel_true) else 0.0
mel_prauc = average_precision_score(mel_true, mel_score) if 0 <
mel_true.sum() < len(mel_true) else 0.0

# Youden's J optimal threshold (maximizes sensitivity + specificity -
```

```
1)
fpr, tpr, thr = roc_curve(mel_true, mel_score)
youden_j = tpr - fpr
best_idx = int(np.argmax(youden_j))
thr_youden = thr[best_idx]

youden_report = summarize_threshold(mel_true, mel_score, thr_youden)
youden_report.update({
    "metric": "YoudenJ_optimal",
    "MEL_AUROC": float(mel_auroc),
    "MEL_PR_AUC": float(mel_prauc),
    "youden_J": float(youden_j[best_idx]),
})

# Also report a higher-specificity point (optional): 90% sensitivity
target_sens = 0.90
idx = np.where(tpr >= target_sens)[0]
if len(idx) > 0:
    best_idx2 = idx[np.argmin(fpr[idx])]
    thr_sens90 = thr[best_idx2]
    sens90_report = summarize_threshold(mel_true, mel_score,
thr_sens90)
    sens90_report.update({
        "metric": "Sensitivity_90%",
        "target_sensitivity": float(target_sens),
        "MEL_AUROC": float(mel_auroc),
        "MEL_PR_AUC": float(mel_prauc),
    })
else:
    sens90_report = None

rows = [youden_report] + ([sens90_report] if sens90_report is not None
else [])
df = pd.DataFrame(rows)

print("Additional MEL operating points:")
print(df[["metric","threshold","sensitivity","specificity","precision"
,"f1","MEL_AUROC","MEL_PR_AUC"]].to_string(index=False))

wandb.log({"clinical/MEL_additional_operating_points":
wandb.Table(dataframe=df)})
wandb.log({"clinical/MEL_PR_AUC": float(mel_prauc),
"clinical/MEL_AUROC": float(mel_auroc)})

Additional MEL operating points:
         metric  threshold  sensitivity  specificity  precision
f1  MEL_AUROC  MEL_PR_AUC
YoudenJ_optimal   0.341932     0.726670     0.750156   0.387272
0.505266   0.819794    0.569331
```

```
Sensitivity_90%   0.215831     0.900044     0.494546   0.278996
0.425955   0.819794    0.569331

---------------------------------------------------------------------------
-----
Error                                     Traceback (most recent call
last)
Cell In[41], line 70
     67 print("Additional MEL operating points:")
     68
print(df[["metric","threshold","sensitivity","specificity","precision"
,"f1","MEL_AUROC","MEL_PR_AUC"]].to_string(index=False))
---> 70 wandb.log({"clinical/MEL_additional_operating_points":
wandb.Table(dataframe=df)})
     71 wandb.log({"clinical/MEL_PR_AUC": float(mel_prauc),
"clinical/MEL_AUROC": float(mel_auroc)})

File
/usr/local/lib/python3.12/dist-packages/wandb/sdk/lib/preinit.py:36,
in PreInitCallable.<locals>.preinit_wrapper(*args, **kwargs)
     35 def preinit_wrapper(*args: Any, **kwargs: Any) -> Any:
---> 36     raise wandb.Error(f"You must call wandb.init() before
{name}()")

Error: You must call wandb.init() before wandb.log()
```