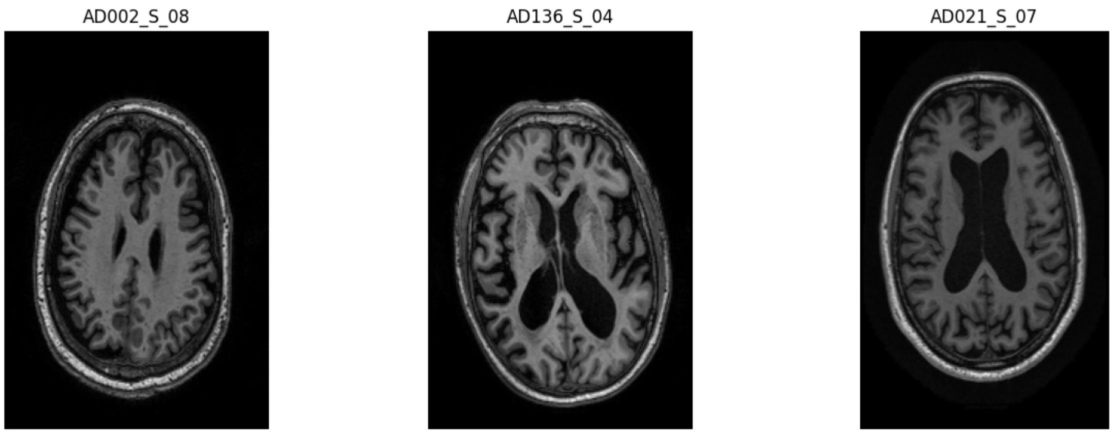
mer-detection-pre-procesing-model2

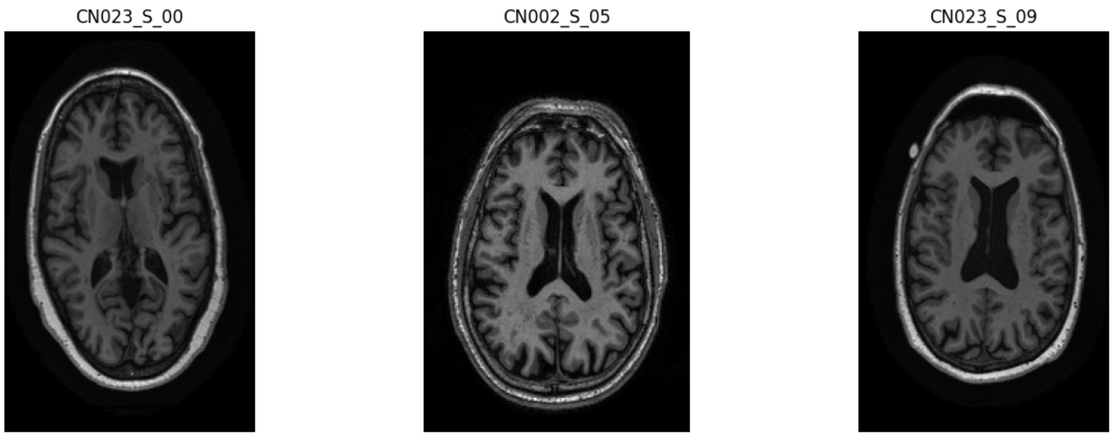
October 28, 2025

[1]:
```
import os
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

# Keep your original directories and subfolders
test_dir = "/kaggle/input/alzheimer-disease/test-20251010T094612Z-1-001/test"
train_dir = "/kaggle/input/alzheimer-disease/train-20251010T183510Z-1-001/train"
subfolders = ["AD", "CN", "MCI"]

# Function to show 3 images from each subfolder (unchanged signature)
def show_images_from_dir(path, n=3):
    # List only image files (same behavior; expanded to be robust if needed)
    files = [f for f in os.listdir(path) if f.lower().endswith(('.png', '.jpg',␣
 ↪'.jpeg', '.bmp', '.tif', '.tiff'))]
    files = files[:n]  # Take only the first n images

    plt.figure(figsize=(15, 5))
    for i, file in enumerate(files):
        img_path = os.path.join(path, file)

        # Open with PIL
        img = Image.open(img_path)

        # --- CRITICAL FIX ---
        # Force grayscale to ensure single-channel display (no colorization)
        # Even if the source is already grayscale, this guarantees mode 'L'
        img = img.convert('L')

        # Convert to numpy for safe imshow with explicit bounds
        arr = np.asarray(img)

        # Display as true grayscale with fixed intensity bounds
        plt.subplot(1, len(files), i + 1)
        plt.imshow(arr, cmap='gray', vmin=0, vmax=255)
        plt.axis("off")
        plt.title(file[:10])
    plt.show()


# Show images for Test set
for subfolder in subfolders:
    print(f"{subfolder} (Test)")
    show_images_from_dir(os.path.join(test_dir, subfolder))

# Show images for Train set
for subfolder in subfolders:
    print(f"{subfolder} (Train)")
    show_images_from_dir(os.path.join(train_dir, subfolder))
```

```
AD (Test)
```

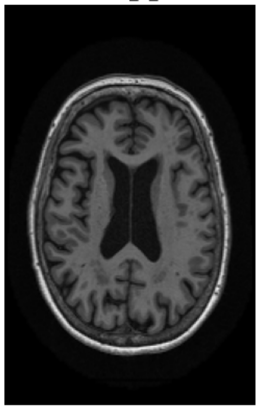


```
CN (Test)
```

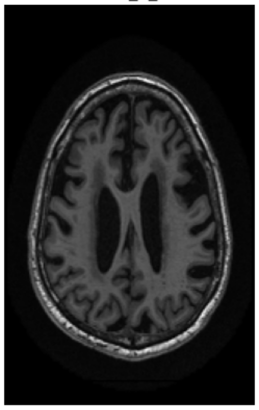


MCI (Test)

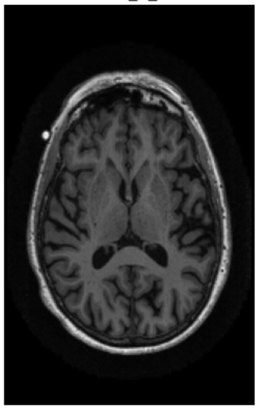
CI128_S_11



CI027_S_12

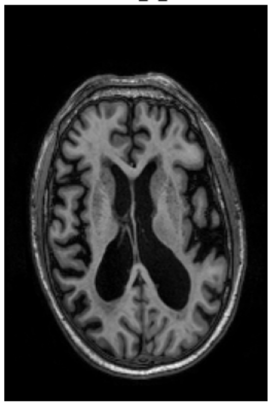


CI067_S_06

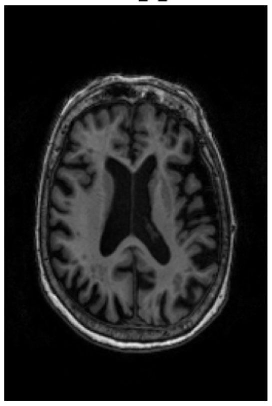


AD (Train)

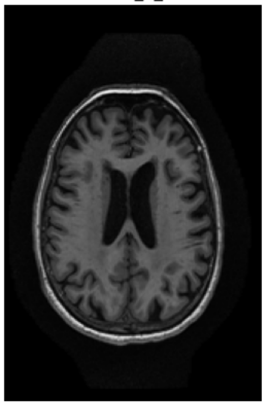
AD136_S_04



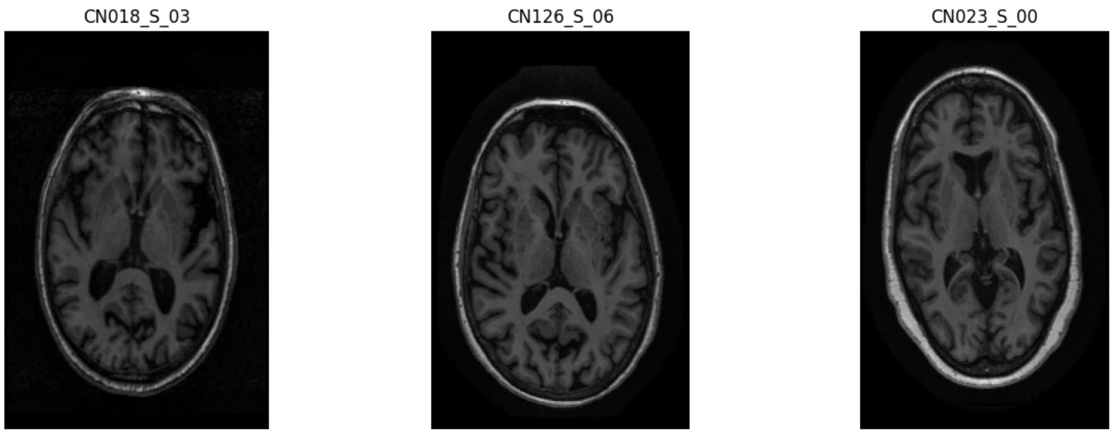
AD136_S_02



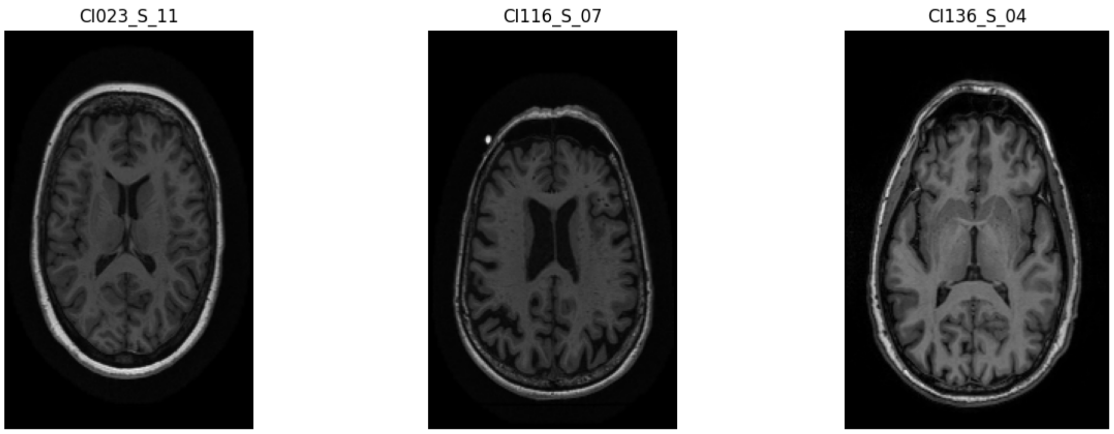
AD007_S_13



CN (Train)



```
MCI (Train)
```



[2]:
```python
import os
import sys
import numpy as np
from PIL import Image, UnidentifiedImageError
import matplotlib.pyplot as plt

# SciPy for morphology / connected components (Kaggle preinstalled)
from scipy.ndimage import (
    gaussian_filter,
    binary_opening,
    binary_closing,
    binary_fill_holes,
    label
)
```

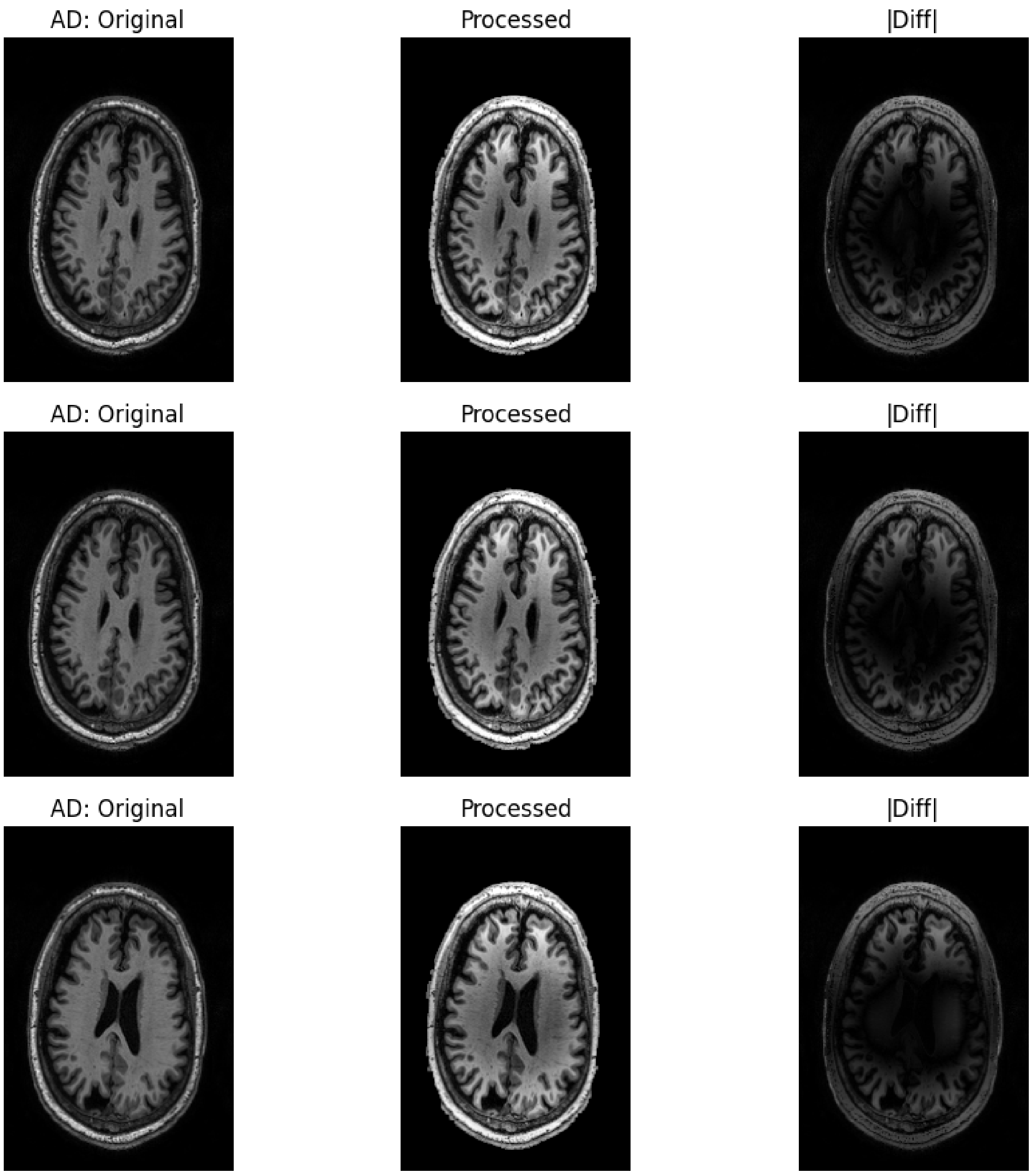
```
# Try skimage for CLAHE; fall back gracefully if missing
try:
    from skimage.exposure import equalize_adapthist
    _HAS_SKIMAGE = True
except Exception:
    _HAS_SKIMAGE = False


# GPU via PyTorch (for homomorphic filtering)

def _ensure(pkg):
    import importlib
    try:
        importlib.import_module(pkg)
    except Exception:
        import subprocess
        subprocess.check_call([sys.executable, "-m", "pip", "install", pkg,␣
 ↪"--quiet"])

_ensure("torch")
import torch
import torch.nn.functional as F

def torch_device():
    return torch.device("cuda" if torch.cuda.is_available() else "cpu")

def gaussian_kernel_2d(sigma: float, radius_factor: float = 3.0, device=None,␣
 ↪dtype=torch.float32):
    """Create a 2D Gaussian kernel tensor for conv2d (normalized)."""
    device = device or torch_device()
    rad = max(1, int(radius_factor * sigma))
    xs = torch.arange(-rad, rad + 1, device=device, dtype=dtype)
    g1 = torch.exp(-0.5 * (xs / sigma) ** 2)
    g1 = g1 / g1.sum()
    g2 = g1[:, None] @ g1[None, :]
    g2 = g2 / g2.sum()
    return g2  # (K, K)

def homomorphic_filter_gpu_u8(arr_u8: np.ndarray, sigma: float = 50.0) -> np.
 ↪ndarray:
    """
    GPU homomorphic filtering (log → low-pass via conv2d → exp → robust␣
 ↪rescale).
    Input uint8 [0..255], output uint8 [0..255].
    """
    dev = torch_device()
    # [B=1,C=1,H,W] float32
```

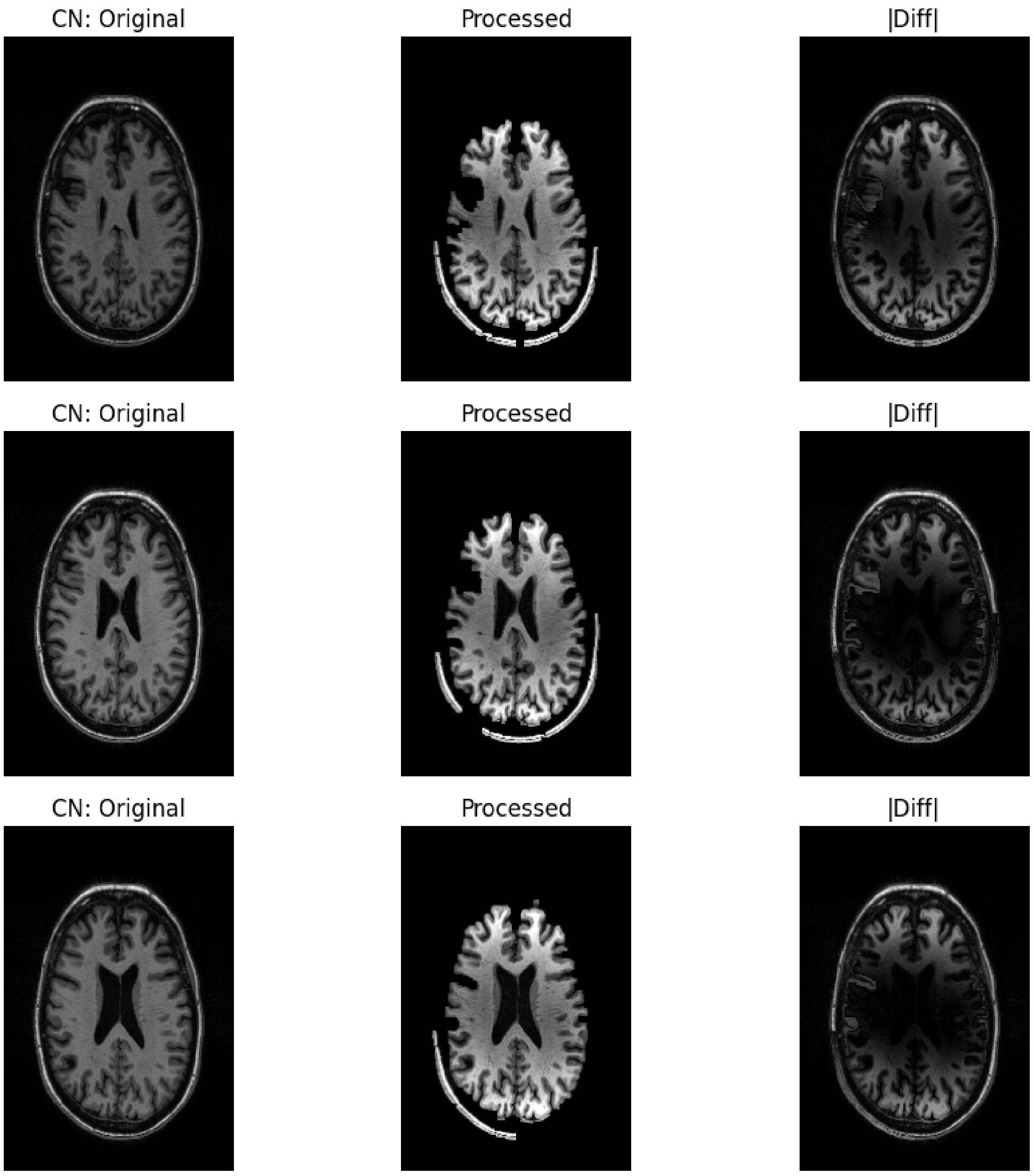
```python
    img = torch.from_numpy(arr_u8.astype(np.float32)).to(dev) + 1.0  # avoid␣
↪log(0)
    img = img[None, None, :, :]  # NCHW
    loga = torch.log(img)
    # 2D Gaussian low-pass via conv2d (reflect padding to avoid border␣
↪artifacts)
    k = gaussian_kernel_2d(sigma=sigma, device=dev)
    k = k[None, None, :, :]  # (out_c,in_c,H,W)
    pad = k.shape[-1] // 2
    low = F.conv2d(F.pad(loga, (pad, pad, pad, pad), mode="reflect"), k)
    high = loga - low
    corr = torch.exp(high) - 1.0  # back to linear domain

    # Robust percentile rescale to uint8
    a = corr.squeeze(0).squeeze(0)  # HxW
    # Compute percentiles on CPU for simplicity
    a_np = a.detach().cpu().numpy().astype(np.float32)
    lo, hi = np.percentile(a_np, [0.5, 99.5])
    if hi - lo < 1e-6:
        a_np = (a_np - a_np.min()) / (a_np.ptp() + 1e-8)
    else:
        a_np = np.clip((a_np - lo) / (hi - lo), 0.0, 1.0)
    out_u8 = (a_np * 255.0 + 0.5).astype(np.uint8)
    return out_u8


test_dir = "/kaggle/input/alzheimer-disease/test-20251010T094612Z-1-001/test"
train_dir = "/kaggle/input/alzheimer-disease/train-20251010T183510Z-1-001/train"
subfolders = ["AD", "CN", "MCI"]


# Output (mirrors the structure)

preprocessed_root = "/kaggle/working/alzheimer-preprocessed"
preprocessed_test = os.path.join(preprocessed_root, "test")
preprocessed_train = os.path.join(preprocessed_root, "train")

IMG_EXT = (".png", ".jpg", ".jpeg", ".bmp", ".tif", ".tiff")

def ensure_dir(p: str):
    os.makedirs(p, exist_ok=True)


# Utilities

def load_gray(path: str) -> np.ndarray:
    """Load as uint8 grayscale [0..255]."""
    return np.asarray(Image.open(path).convert("L"), dtype=np.uint8)

def save_gray(arr_u8: np.ndarray, path: str):
    Image.fromarray(arr_u8, mode="L").save(path)

def percentile_rescale_u8(arr: np.ndarray, p_low=1.0, p_high=99.0) -> np.
 ↪ndarray:
    """Robust contrast stretching to uint8."""
    a = arr.astype(np.float32)
    lo, hi = np.percentile(a, [p_low, p_high])
    if hi - lo < 1e-6:
        a = (a - a.min()) / (a.ptp() + 1e-8)
    else:
        a = np.clip((a - lo) / (hi - lo), 0.0, 1.0)
    return (a * 255.0 + 0.5).astype(np.uint8)

def otsu_threshold_u8(arr_u8: np.ndarray) -> int:
    """Pure NumPy Otsu threshold (returns 0..255)."""
    hist = np.bincount(arr_u8.ravel(), minlength=256).astype(np.float64)
    prob = hist / (arr_u8.size + 1e-12)
    omega = np.cumsum(prob)
    mu = np.cumsum(prob * np.arange(256))
    mu_t = mu[-1]
    sigma_b2 = (mu_t * omega - mu) ** 2 / (omega * (1 - omega) + 1e-12)
    sigma_b2[~np.isfinite(sigma_b2)] = 0.0
    return int(np.argmax(sigma_b2))

def largest_cc(mask: np.ndarray) -> np.ndarray:
    """Keep only largest connected component of a binary mask."""
    lbl, n = label(mask.astype(np.uint8))
    if n <= 1:
        return mask.astype(bool)
    counts = np.bincount(lbl.ravel())
    counts[0] = 0  # background
    keep = counts.argmax()
    return (lbl == keep)

def clahe_u8(arr_u8: np.ndarray) -> np.ndarray:
    """CLAHE if skimage is available; otherwise identity."""
    if _HAS_SKIMAGE:
        arr01 = arr_u8.astype(np.float32) / 255.0
        out01 = equalize_adapthist(arr01, clip_limit=0.01)
        return (np.clip(out01, 0.0, 1.0) * 255.0 + 0.5).astype(np.uint8)
    else:
        return arr_u8
```

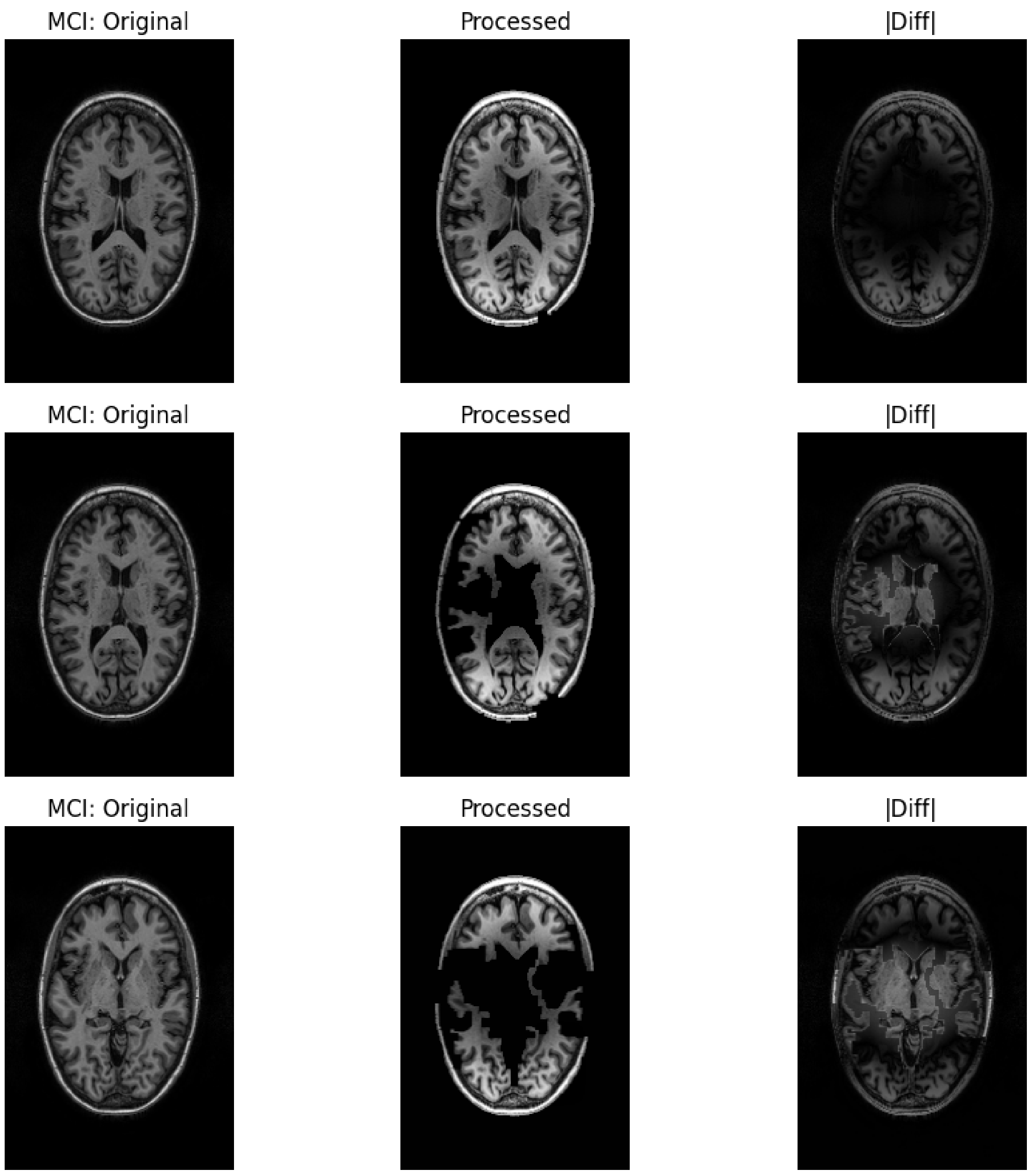
```
# Core: one-image preprocessing

def preprocess_single_image(
    arr_u8: np.ndarray,
    do_skull_strip: bool = True,
    sigma_homomorphic: float = 50.0,
    min_area: int = 500
):
    """
    Steps:
      1) Robust pre-stretch (percentile)
      2) GPU homomorphic filter (PyTorch)
      3) (Optional) 2D skull/background stripping via Otsu + morphology + LCC +␣
↪hole fill
      4) Gentle CLAHE (if available), else percentile rescale
    Returns processed uint8 image, plus an optional mask (uint8).
    """
    # 1) Robust stretch
    a1 = percentile_rescale_u8(arr_u8, 1.0, 99.0)

    # 2) Homomorphic filtering on GPU
    a2 = homomorphic_filter_gpu_u8(a1, sigma=sigma_homomorphic)

    brain_mask = None
    if do_skull_strip:
        # 3) Otsu thresholding (foreground bright)
        th = otsu_threshold_u8(a2)
        mask = (a2 >= th).astype(np.uint8)

        # Morphological clean-up
        mask = binary_opening(mask, structure=np.ones((3,3), dtype=np.uint8))
        mask = binary_closing(mask, structure=np.ones((5,5), dtype=np.uint8))
        mask = binary_fill_holes(mask)
        mask = largest_cc(mask)

        # Remove tiny masks (safety)
        if mask.sum() < min_area:
            mask = np.ones_like(mask, dtype=bool)  # fallback: keep as-is

        a3 = (a2 * mask).astype(np.uint8)
        brain_mask = (mask.astype(np.uint8) * 255)
    else:
        a3 = a2

    # 4) CLAHE or robust stretch
    a4 = clahe_u8(a3)
    a4 = percentile_rescale_u8(a4, 0.5, 99.5)
    return a4, brain_mask


# Dataset-level processing & visualization

def preprocess_dataset(src_root: str, dst_root: str, n_preview: int = 3,␣
↪do_skull_strip=True):
    """
    Applies the pipeline to all images under src_root/{AD|CN|MCI}
    and writes to dst_root/{AD|CN|MCI}. Silent on per-file issues.
    """
    ensure_dir(dst_root)
    summary = {}

    # Report GPU/CPU
    dev = torch_device()
    print(f"Device for homomorphic filtering: {dev}")

    for cls in subfolders:
        src_cls = os.path.join(src_root, cls)
        dst_cls = os.path.join(dst_root, cls)
        ensure_dir(dst_cls)

        processed = skipped = 0
        if not os.path.isdir(src_cls):
            summary[cls] = (0, 0)
            continue

        files = sorted([f for f in os.listdir(src_cls) if f.lower().
↪endswith(IMG_EXT)])

        # Process & save
        for fname in files:
            spath = os.path.join(src_cls, fname)
            dpath = os.path.join(dst_cls, fname)
            try:
                arr = load_gray(spath)
                proc, _ = preprocess_single_image(arr,␣
↪do_skull_strip=do_skull_strip)
                save_gray(proc, dpath)
                processed += 1
            except (UnidentifiedImageError, OSError, RuntimeError, ValueError):
                skipped += 1
                continue

        summary[cls] = (processed, skipped)
        # Preview few examples
        preview = files[:n_preview]
        if preview:
            fig, axs = plt.subplots(len(preview), 3, figsize=(10,␣
 ↪3*len(preview)))
            if len(preview) == 1:
                axs = np.expand_dims(axs, 0)
            for i, fname in enumerate(preview):
                sp = os.path.join(src_cls, fname)
                dp = os.path.join(dst_cls, fname)
                try:
                    orig = load_gray(sp)
                    proc = load_gray(dp)
                    diff = np.abs(proc.astype(np.int16) - orig.astype(np.
 ↪int16)).astype(np.uint8)

                    axs[i, 0].imshow(orig, cmap="gray", vmin=0, vmax=255);␣
 ↪axs[i,0].axis("off"); axs[i,0].set_title(f"{cls}: Original")
                    axs[i, 1].imshow(proc, cmap="gray", vmin=0, vmax=255);␣
 ↪axs[i,1].axis("off"); axs[i,1].set_title("Processed")
                    axs[i, 2].imshow(diff, cmap="gray", vmin=0, vmax=255);␣
 ↪axs[i,2].axis("off"); axs[i,2].set_title("|Diff|")
                except Exception:
                    continue
            plt.tight_layout()
            plt.show()

    return summary


print("=== 2D MRI Preprocessing (GPU homomorphic + 2D skull-strip): TEST ===")
sum_test = preprocess_dataset(test_dir, preprocessed_test, n_preview=3,␣
 ↪do_skull_strip=True)

print("=== 2D MRI Preprocessing (GPU homomorphic + 2D skull-strip): TRAIN ===")
sum_train = preprocess_dataset(train_dir, preprocessed_train, n_preview=3,␣
 ↪do_skull_strip=True)

def _fmt(s): return ", ".join([f"{k}: {v[0]} ok / {v[1]} skipped" for k, v in s.
 ↪items()])
print("Preprocessed images saved to:")
print(f"  • Test : {preprocessed_test}")
print(f"  • Train: {preprocessed_train}")
print("Summary TEST :", _fmt(sum_test))
print("Summary TRAIN:", _fmt(sum_train))
```

```
=== 2D MRI Preprocessing (GPU homomorphic + 2D skull-strip): TEST ===
Device for homomorphic filtering: cuda
```

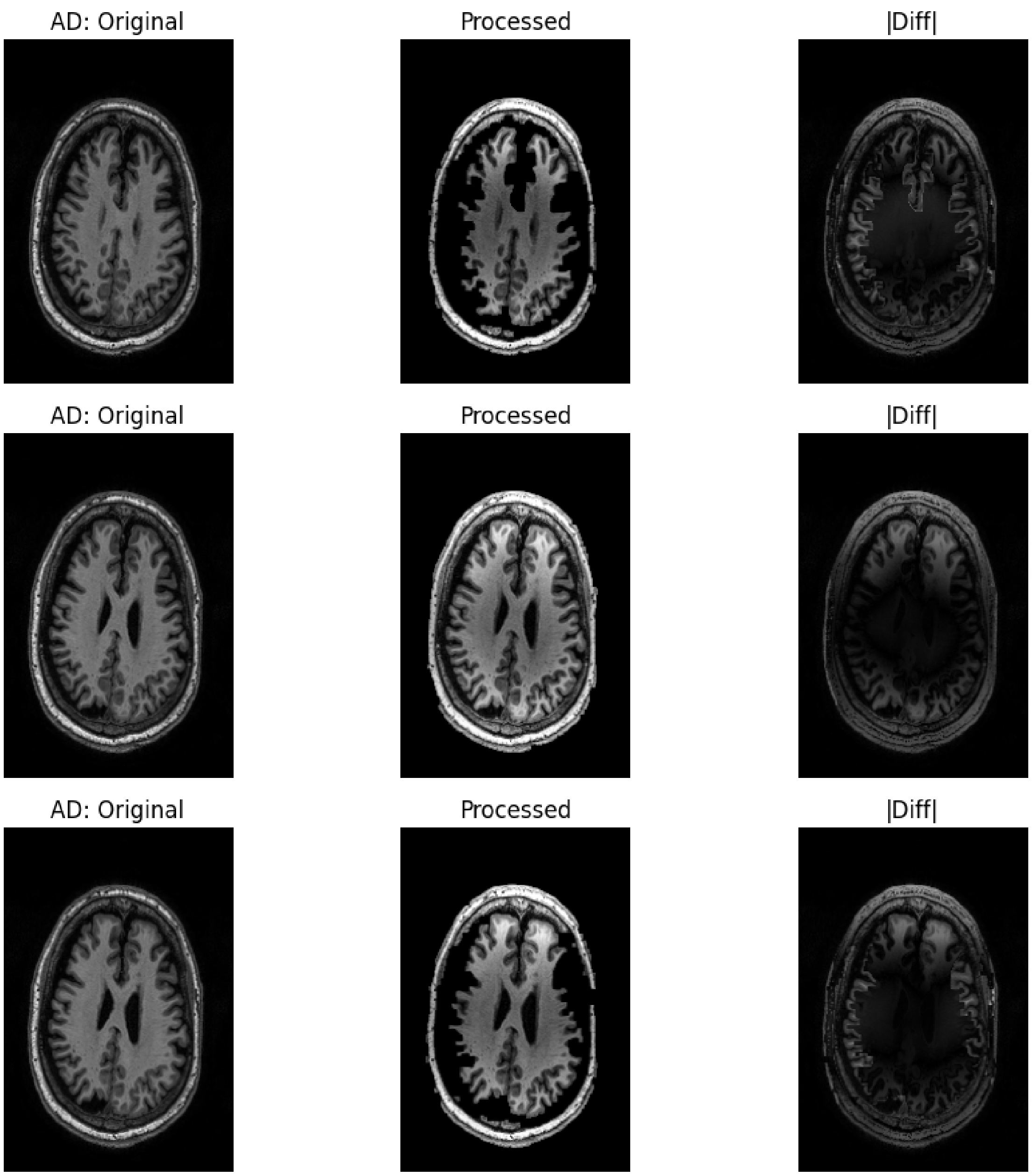
```
/tmp/ipykernel_19/2569239085.py:107: DeprecationWarning: 'mode' parameter is
deprecated and will be removed in Pillow 13 (2026-10-15)
  Image.fromarray(arr_u8, mode="L").save(path)
```

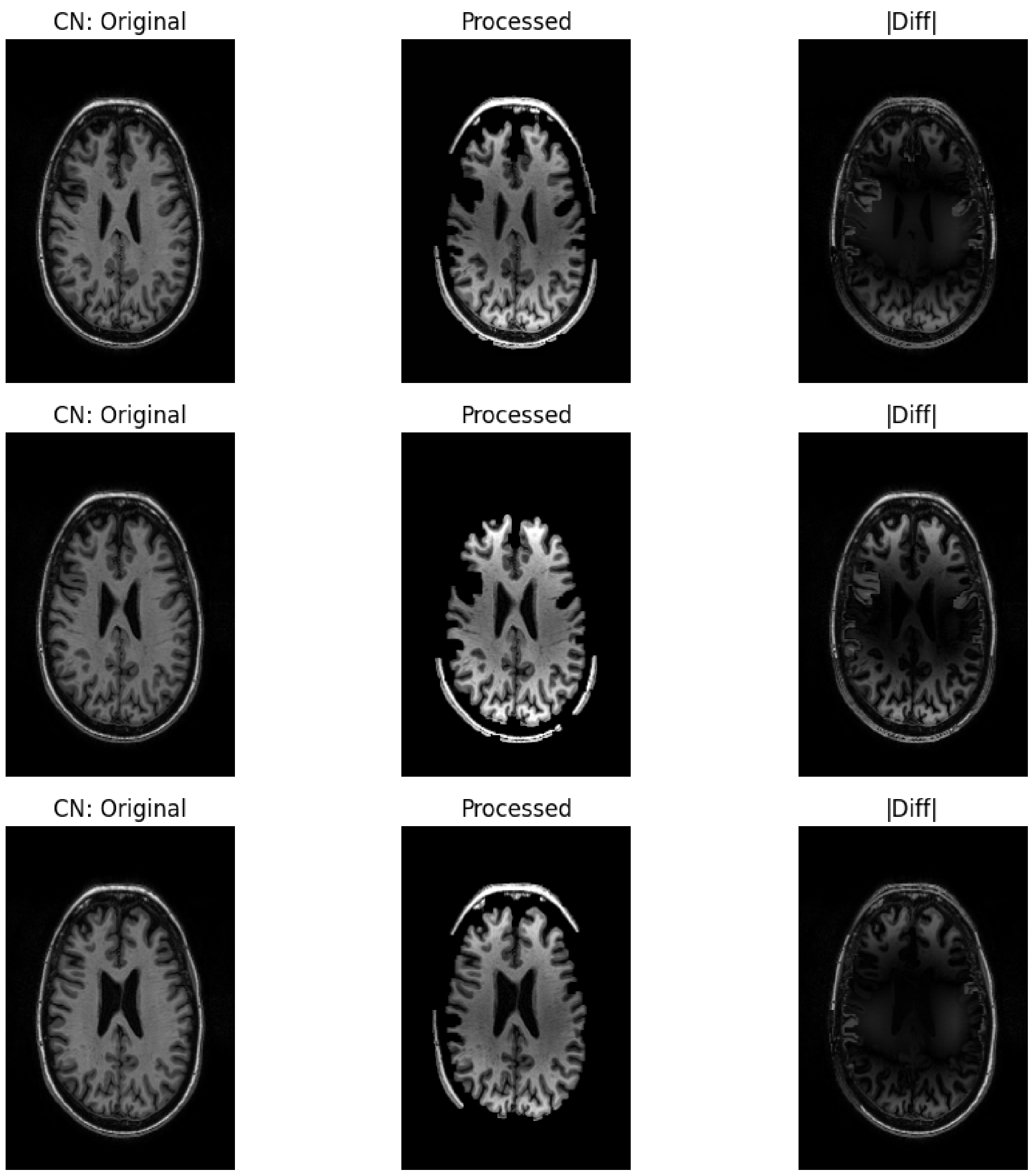


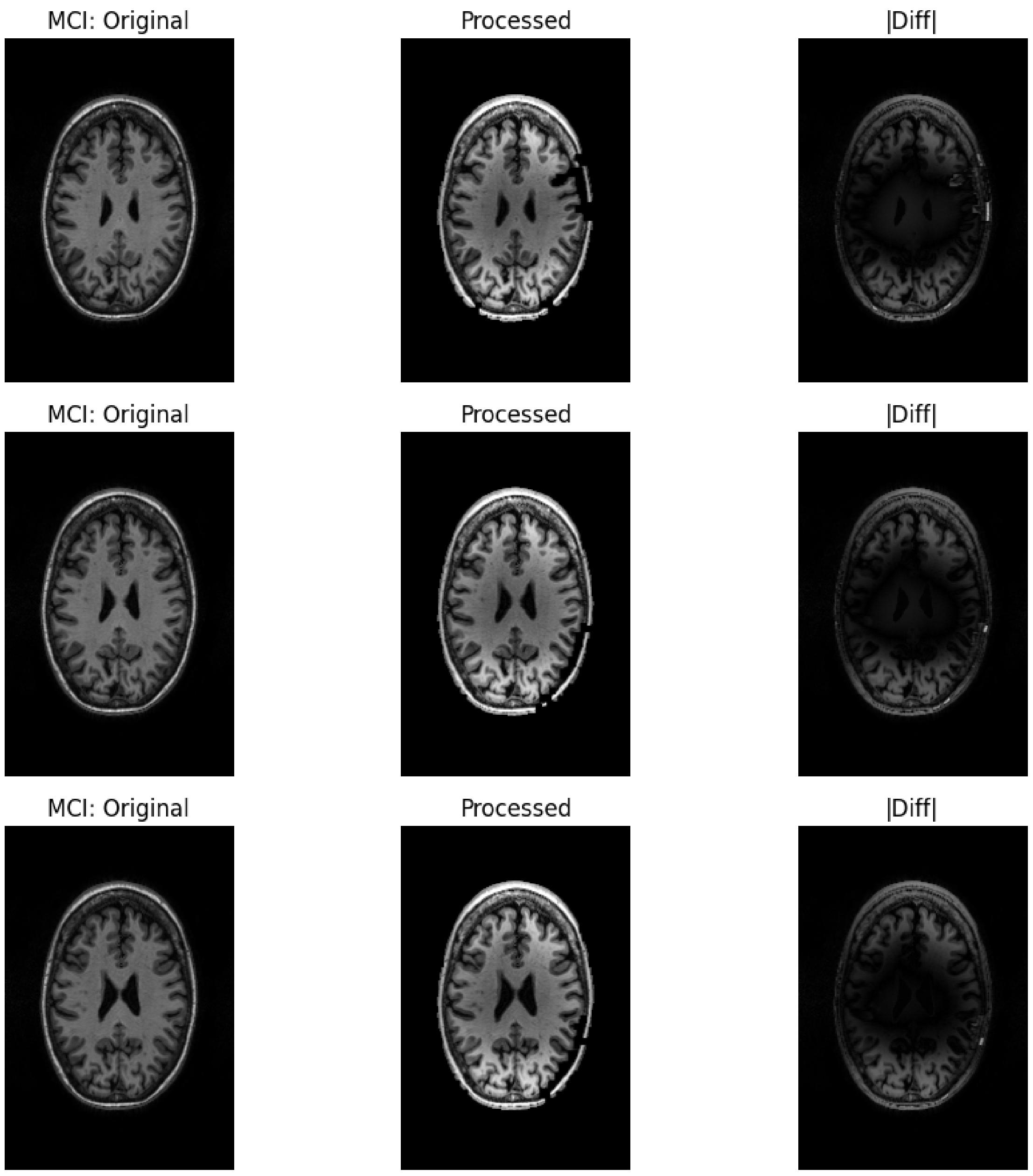

CN: Original
Processed
|Diff|
CN: Original
Processed
|Diff|
CN: Original
Processed
|Diff|



```
=== 2D MRI Preprocessing (GPU homomorphic + 2D skull-strip): TRAIN ===
Device for homomorphic filtering: cuda
```


AD: Original
Processed
|Diff|
AD: Original
Processed
|Diff|
AD: Original
Processed
|Diff|


CN: Original
Processed
|Diff|
CN: Original
Processed
|Diff|
CN: Original
Processed
|Diff|



```
Preprocessed images saved to:
  • Test : /kaggle/working/alzheimer-preprocessed/test
  • Train: /kaggle/working/alzheimer-preprocessed/train
Summary TEST : AD: 225 ok / 0 skipped, CN: 288 ok / 0 skipped, MCI: 518 ok / 0
skipped
Summary TRAIN: AD: 899 ok / 0 skipped, CN: 1152 ok / 0 skipped, MCI: 2072 ok / 0
skipped
```

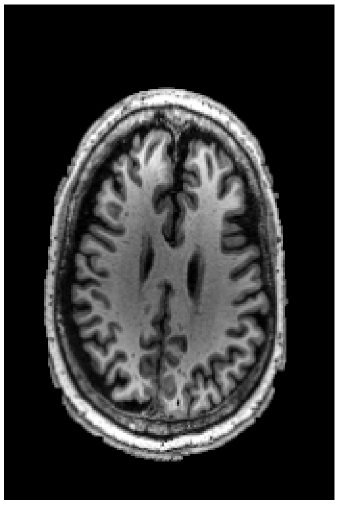
[3]:
```
import os
import numpy as np
from PIL import Image, UnidentifiedImageError
import matplotlib.pyplot as plt

test_dir = "/kaggle/input/alzheimer-disease/test-20251010T094612Z-1-001/test"
train_dir = "/kaggle/input/alzheimer-disease/train-20251010T183510Z-1-001/train"
subfolders = ["AD", "CN", "MCI"]

preprocessed_test = "/kaggle/working/alzheimer-preprocessed/test"
preprocessed_train = "/kaggle/working/alzheimer-preprocessed/train"
resized_root = "/kaggle/working/alzheimer-resized-224"
resized_test = os.path.join(resized_root, "test")
resized_train = os.path.join(resized_root, "train")

IMG_EXT = (".png", ".jpg", ".jpeg", ".bmp", ".tif", ".tiff")
TARGET_SIZE = (224, 224)

def ensure_dir(p: str):
    os.makedirs(p, exist_ok=True)

def load_gray(path: str) -> np.ndarray:
    return np.asarray(Image.open(path).convert("L"), dtype=np.uint8)

def resize_and_save_gray(arr_u8: np.ndarray, path: str):
    img = Image.fromarray(arr_u8, mode="L")
    img_resized = img.resize(TARGET_SIZE, Image.Resampling.LANCZOS)
    img_resized.save(path)

def resize_dataset(src_root: str, dst_root: str, n_preview: int = 3):
    ensure_dir(dst_root)
    summary = {}

    for cls in subfolders:
        src_cls = os.path.join(src_root, cls)
        dst_cls = os.path.join(dst_root, cls)
        ensure_dir(dst_cls)

        processed = skipped = 0
        if not os.path.isdir(src_cls):
            summary[cls] = (0, 0)
            continue

        files = sorted([f for f in os.listdir(src_cls) if f.lower().
 ↪endswith(IMG_EXT)])

        for fname in files:
            spath = os.path.join(src_cls, fname)
            dpath = os.path.join(dst_cls, fname)
            try:
                arr = load_gray(spath)
                resize_and_save_gray(arr, dpath)
                processed += 1
            except (UnidentifiedImageError, OSError, RuntimeError, ValueError):
                skipped += 1
                continue

        summary[cls] = (processed, skipped)

        preview = files[:n_preview]
        if preview:
            fig, axs = plt.subplots(len(preview), 2, figsize=(8,␣
↪3*len(preview)))
            if len(preview) == 1:
                axs = np.expand_dims(axs, 0)
            for i, fname in enumerate(preview):
                sp = os.path.join(src_cls, fname)
                dp = os.path.join(dst_cls, fname)
                try:
                    orig = load_gray(sp)
                    resized = load_gray(dp)

                    axs[i, 0].imshow(orig, cmap="gray", vmin=0, vmax=255)
                    axs[i, 0].axis("off")
                    axs[i, 0].set_title(f"{cls}: Original {orig.shape}")

                    axs[i, 1].imshow(resized, cmap="gray", vmin=0, vmax=255)
                    axs[i, 1].axis("off")
                    axs[i, 1].set_title(f"Resized {resized.shape}")
                except Exception:
                    continue
            plt.tight_layout()
            plt.show()

    return summary

print("=== Resizing Preprocessed Images: TEST ===")
sum_test = resize_dataset(preprocessed_test, resized_test, n_preview=3)

print("=== Resizing Preprocessed Images: TRAIN ===")
sum_train = resize_dataset(preprocessed_train, resized_train, n_preview=3)

def _fmt(s): return ", ".join([f"{k}: {v[0]} ok / {v[1]} skipped" for k, v in s.
↪items()])
```

```python
print("Resized images saved to:")
print(f"  • Resized Test : {resized_test}")
print(f"  • Resized Train: {resized_train}")
print("Summary TEST :", _fmt(sum_test))
print("Summary TRAIN:", _fmt(sum_train))
```

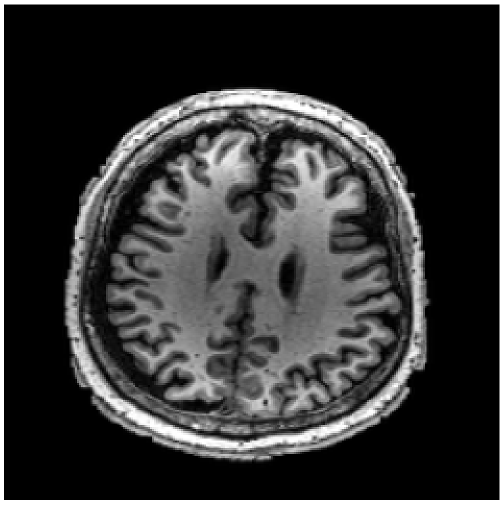
```
=== Resizing Preprocessed Images: TEST ===
```

```
/tmp/ipykernel_19/502029755.py:26: DeprecationWarning: 'mode' parameter is
deprecated and will be removed in Pillow 13 (2026-10-15)
  img = Image.fromarray(arr_u8, mode="L")
```

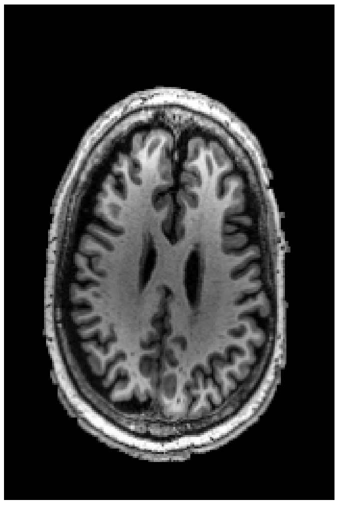
AD: Original (256, 170)



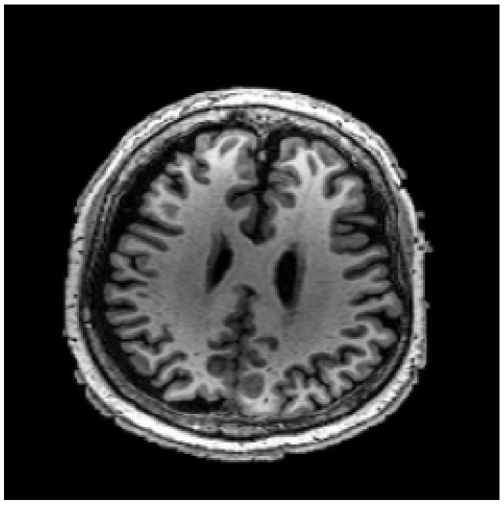
Resized (224, 224)



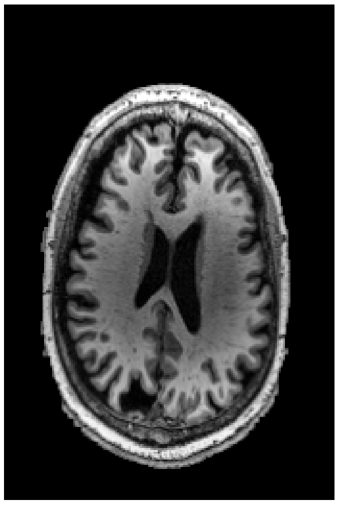
AD: Original (256, 170)



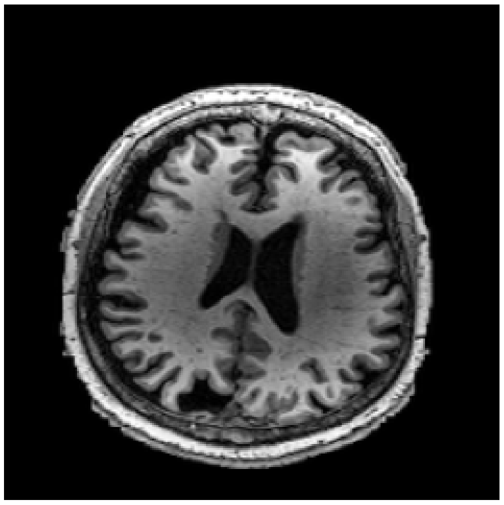
Resized (224, 224)

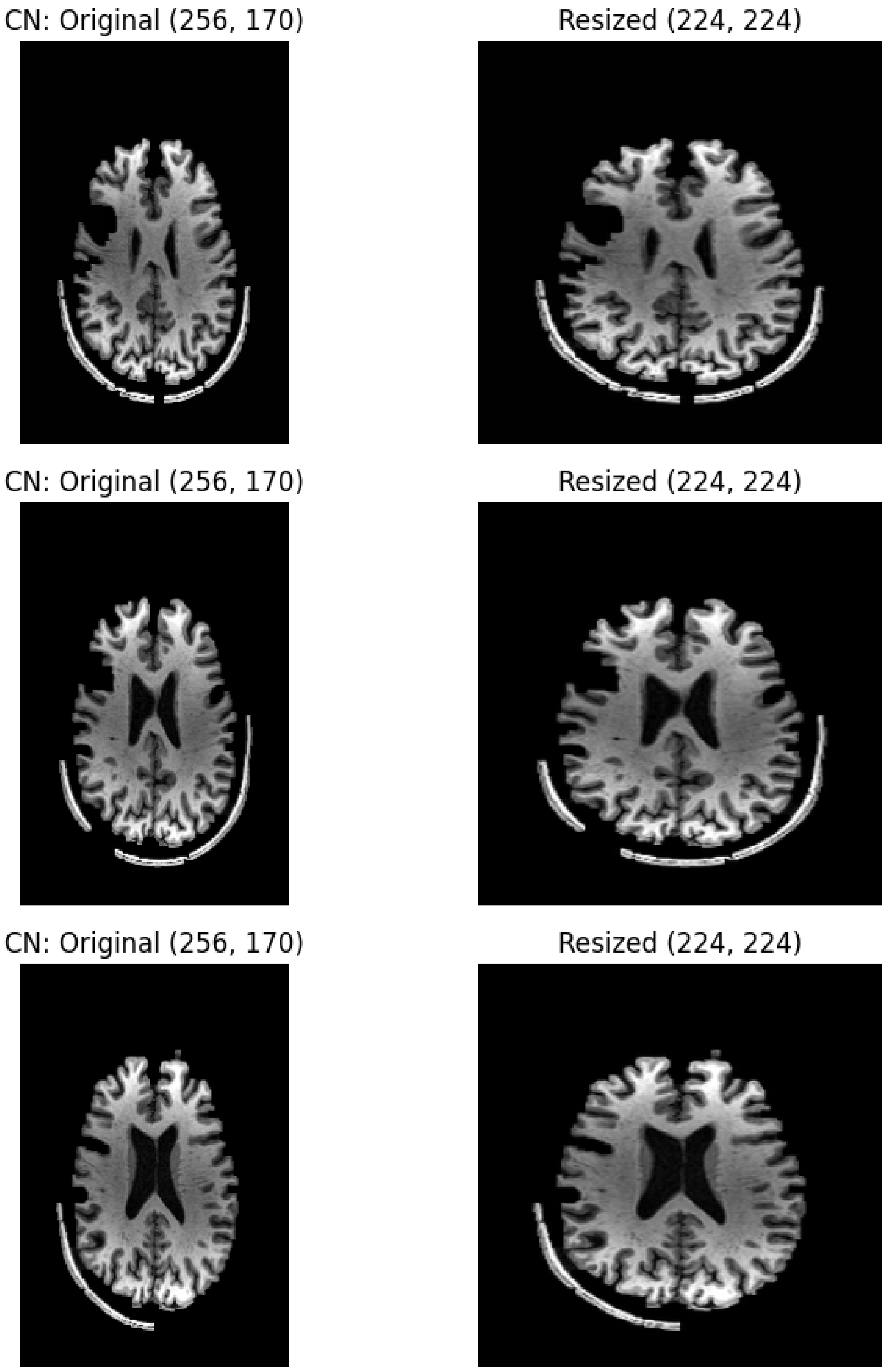


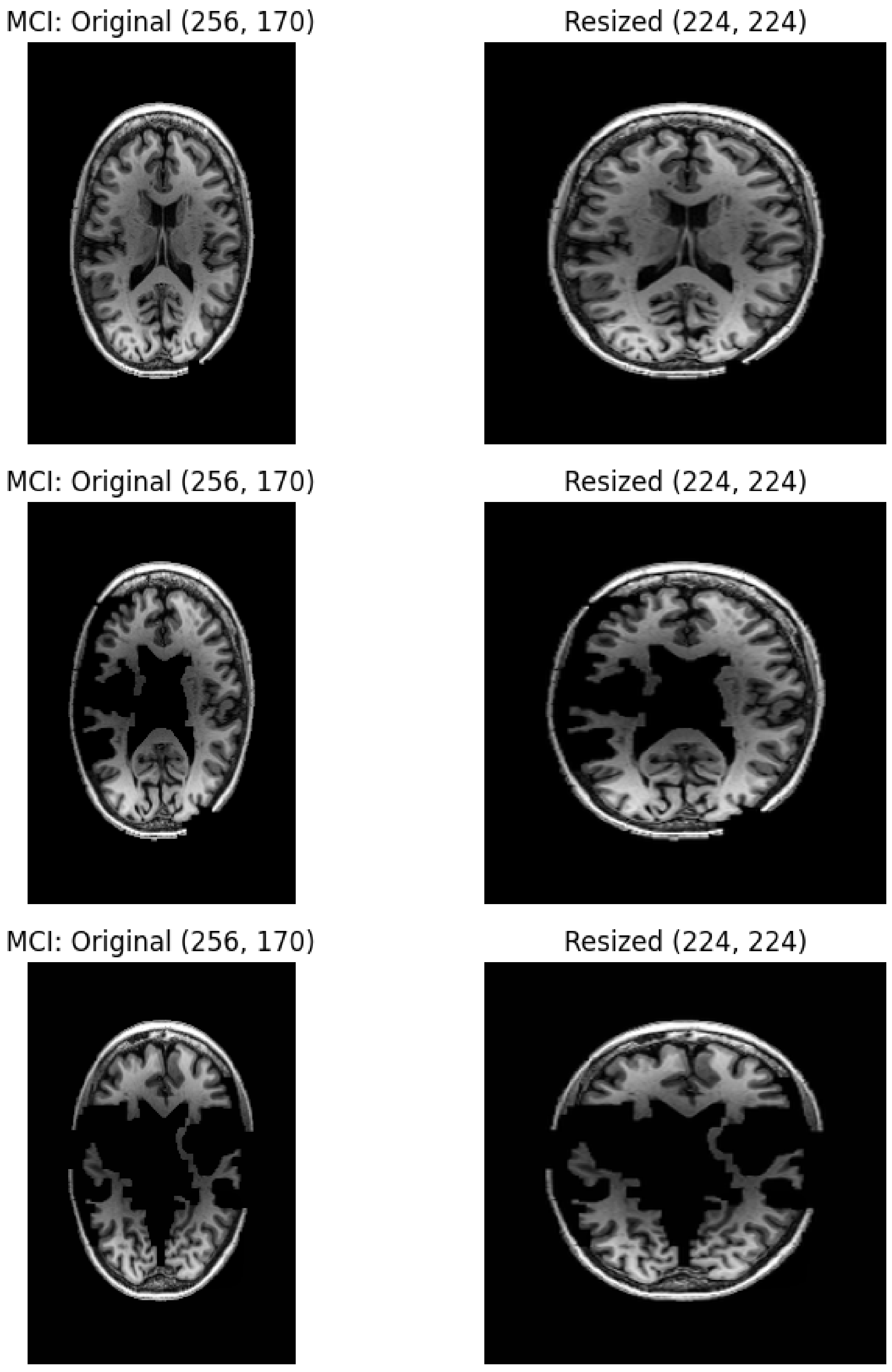
AD: Original (256, 170)



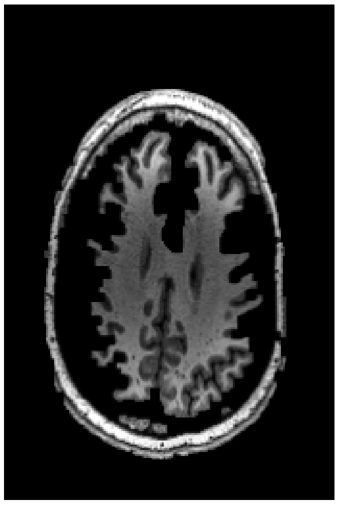
Resized (224, 224)

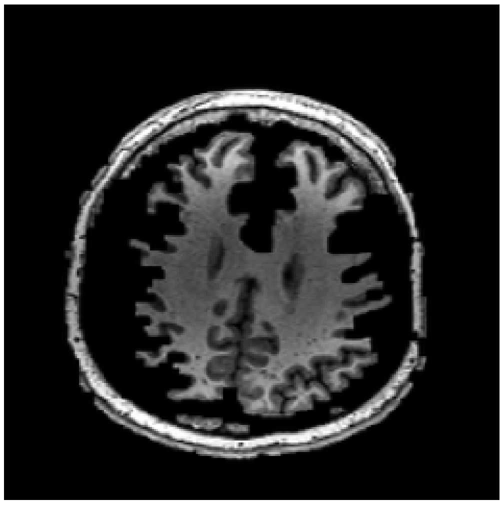


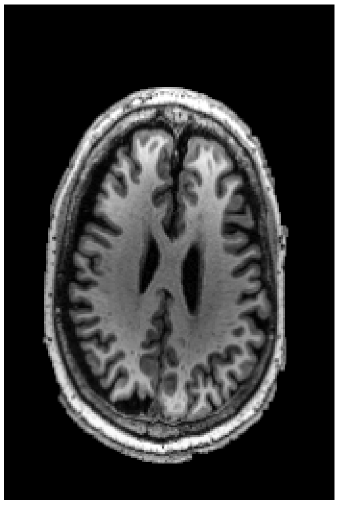

CN: Original (256, 170)
Resized (224, 224)
CN: Original (256, 170)
Resized (224, 224)
CN: Original (256, 170)
Resized (224, 224)

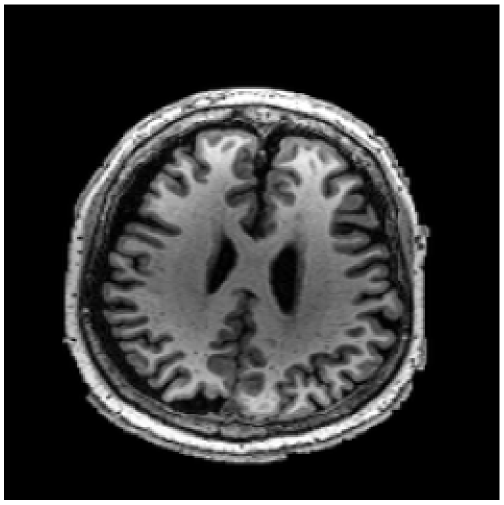

MCI: Original (256, 170)
Resized (224, 224)
MCI: Original (256, 170)
Resized (224, 224)
MCI: Original (256, 170)
Resized (224, 224)

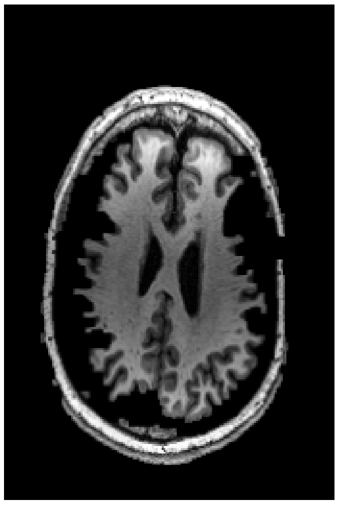
```
=== Resizing Preprocessed Images: TRAIN ===
```

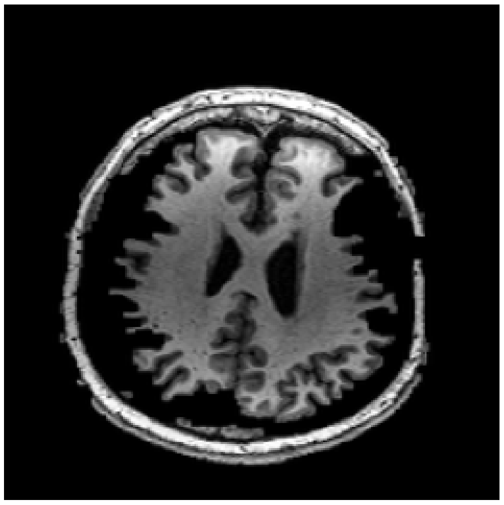

AD: Original (256, 170)

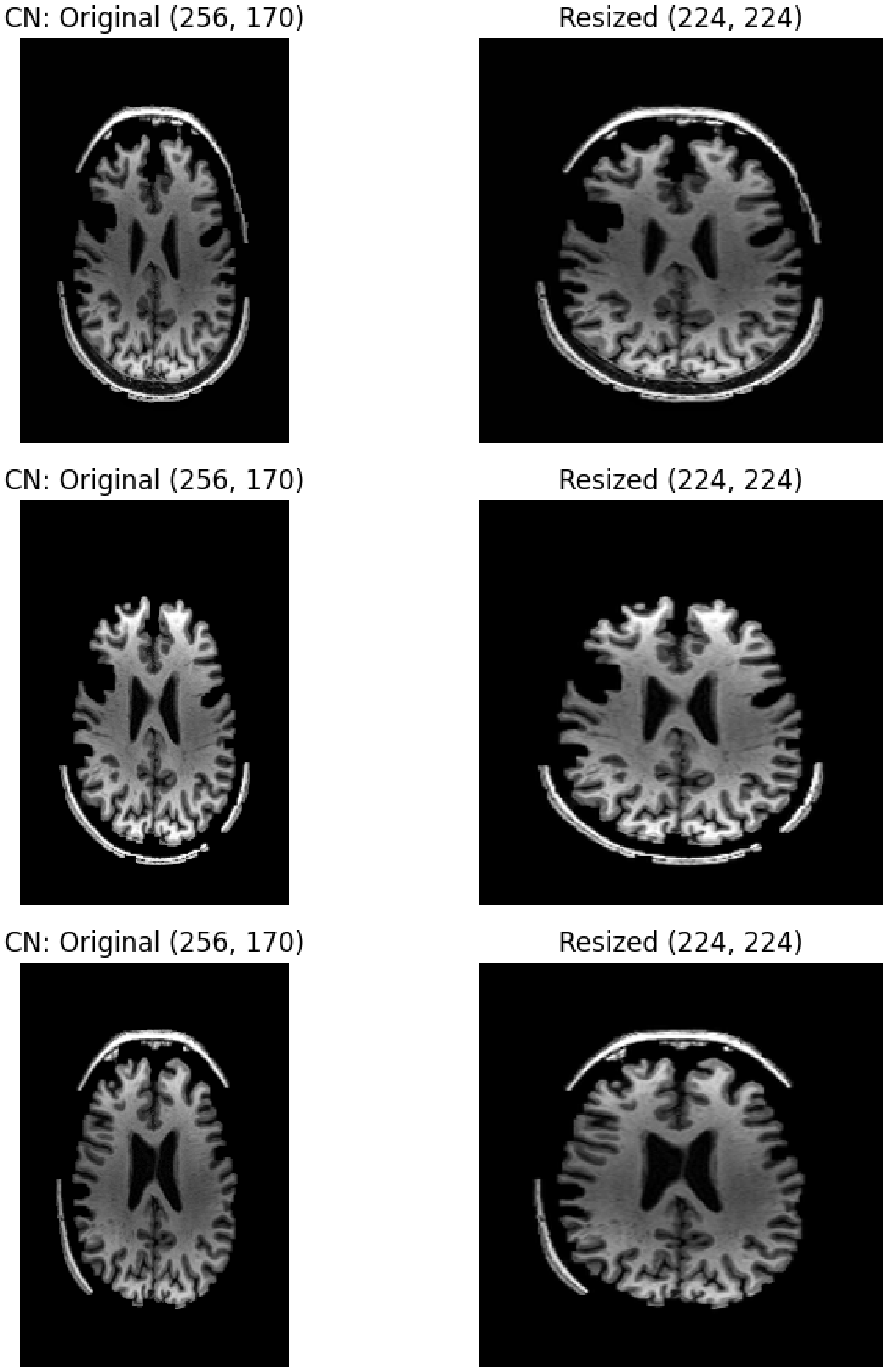

Resized (224, 224)

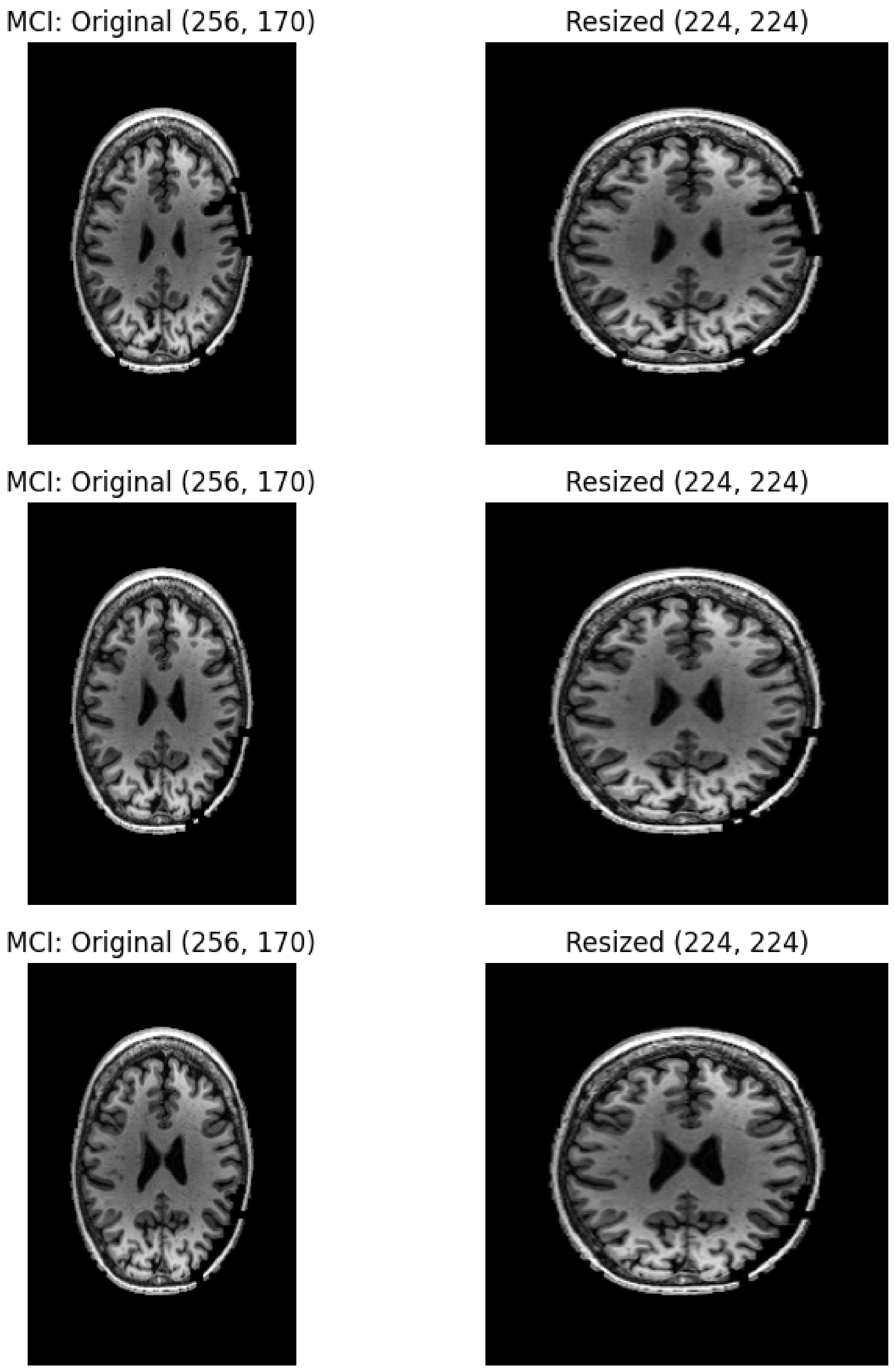

AD: Original (256, 170)


Resized (224, 224)


AD: Original (256, 170)


Resized (224, 224)


CN: Original (256, 170)
Resized (224, 224)
CN: Original (256, 170)
Resized (224, 224)
CN: Original (256, 170)
Resized (224, 224)


MCI: Original (256, 170)
Resized (224, 224)
MCI: Original (256, 170)
Resized (224, 224)
MCI: Original (256, 170)
Resized (224, 224)

```
Resized images saved to:
  • Resized Test : /kaggle/working/alzheimer-resized-224/test
  • Resized Train: /kaggle/working/alzheimer-resized-224/train
Summary TEST : AD: 225 ok / 0 skipped, CN: 288 ok / 0 skipped, MCI: 518 ok / 0
skipped
Summary TRAIN: AD: 899 ok / 0 skipped, CN: 1152 ok / 0 skipped, MCI: 2072 ok / 0
skipped
```

Train And Test Split

[4]:
```python
import os
import sys
import csv
import random
import shutil
from pathlib import Path
from typing import Dict, List, Tuple

# ------------------------------------------
# Configuration (modified for resized images)
# ------------------------------------------
# Source resized dataset (224x224)
RESIZED_ROOT = os.environ.get("RESIZED_ROOT", "/kaggle/working/
 ↪alzheimer-resized-224")
SRC_TRAIN = os.path.join(RESIZED_ROOT, "train")
SRC_TEST  = os.path.join(RESIZED_ROOT, "test")

# Classes (same as before)
CLASSES = ["AD", "CN", "MCI"]

# Where to write splits
SPLITS_ROOT = "/kaggle/working/alzheimer-resized-224_splits"
os.makedirs(SPLITS_ROOT, exist_ok=True)

# The split ratios (train:test) you requested
RATIO_LIST = [
    (0.90, 0.10),
    (0.80, 0.20),
    (0.70, 0.30),
    (0.60, 0.40),
    (0.50, 0.50),
    (0.40, 0.60),
    (0.30, 0.70),
    (0.20, 0.80),
    (0.10, 0.90)
]

# Validation share taken from the training portion
VAL_FRACTION = 0.10

# Base random seed (deterministic builds). Each ratio derives its own seed.
BASE_SEED = 2025

# -----------------------------------------
# Helpers
# -----------------------------------------
IMG_EXTS = (".png", ".jpg", ".jpeg", ".bmp", ".tif", ".tiff")

def list_images_in_class(class_dir: str) -> List[str]:
    """Return absolute paths of all images in a class directory."""
    if not os.path.isdir(class_dir):
        return []
    files = sorted([
        str(Path(class_dir) / f) for f in os.listdir(class_dir)
        if f.lower().endswith(IMG_EXTS)
    ])
    return files

def gather_all_images() -> Dict[str, List[str]]:
    """
    Gather all resized images per class, from both 'train' and 'test'
    to form a single full pool for stratified splitting by ratio.
    """
    all_by_class = {c: [] for c in CLASSES}
    for c in CLASSES:
        # From resized train
        all_by_class[c].extend(list_images_in_class(os.path.join(SRC_TRAIN, c)))
        # From resized test
        all_by_class[c].extend(list_images_in_class(os.path.join(SRC_TEST, c)))
    return all_by_class

def ensure_dirs(*paths: str):
    for p in paths:
        os.makedirs(p, exist_ok=True)

def link_or_copy(src: str, dst: str):
    """Create a symlink; if not permitted, copy the file."""
    try:
        # Remove dst if exists
        if os.path.lexists(dst):
            os.unlink(dst)
        os.symlink(src, dst)
    except OSError:
        shutil.copy2(src, dst)

def write_manifest(split_dir: str, split_name: str, rows: List[Tuple[str, str,␣
↪str, str]]):
    """
    Write both CSV manifest and plain path list.
    rows: list of (split, cls, filename, src_path)
    """
    # CSV
    csv_path = os.path.join(split_dir, f"{split_name}.csv")
    with open(csv_path, "w", newline="") as f:
        w = csv.writer(f)
        w.writerow(["split", "class", "filename", "path"])
        for r in rows:
            w.writerow(r)

    # TXT list
    txt_path = os.path.join(split_dir, f"{split_name}.txt")
    with open(txt_path, "w") as f:
        for _, _, _, p in rows:
            f.write(p + "\n")

def summarize_counts(counts: Dict[str, Dict[str, int]]):
    """
    Print counts per split (train/val/test) and per class for quick sanity␣
↪check.
    """
    for split in ["train", "val", "test"]:
        info = counts.get(split, {})
        total = sum(info.values())
        detail = ", ".join([f"{k}: {v}" for k, v in info.items()])
        print(f"{split.capitalize():5s} => total {total:5d} | {detail}")

# -----------------------------------------
# Split builder
# -----------------------------------------
def build_splits():
    # 1) Pool all images by class (union of resized/train and resized/test)
    all_by_class = gather_all_images()

    # Optional: quick report of total availability
    print("Total images by class (union of resized/train + resized/test):")
    for c in CLASSES:
        print(f"  {c}: {len(all_by_class[c])}")

    # 2) For each (train_ratio, test_ratio), create a split folder and populate
    for tr_ratio, te_ratio in RATIO_LIST:
        # sanity: ratios sum approx 1
        assert abs(tr_ratio + te_ratio - 1.0) < 1e-6, "Train+Test ratio must␣
↪sum to 1"

        # deterministic seed per ratio
        seed = BASE_SEED + int(round(te_ratio * 100))
        rng = random.Random(seed)

        # Split name and dirs
        split_name =␣
↪f"split_{int(round(tr_ratio*100))}_{int(round(te_ratio*100))}"
        split_root = os.path.join(SPLITS_ROOT, split_name)
        train_root = os.path.join(split_root, "train")
        val_root   = os.path.join(split_root, "val")
        test_root  = os.path.join(split_root, "test")
        ensure_dirs(split_root, train_root, val_root, test_root)
        for c in CLASSES:
            ensure_dirs(os.path.join(train_root, c), os.path.join(val_root, c),␣
↪os.path.join(test_root, c))

        rows_train, rows_val, rows_test = [], [], []
        counts = {"train": {}, "val": {}, "test": {}}

        # 3) Per-class stratified splitting
        for c in CLASSES:
            full_list = list(all_by_class[c])  # copy
            rng.shuffle(full_list)             # deterministic shuffle

            n_total = len(full_list)
            n_test  = max(0, int(round(n_total * te_ratio)))
            n_test  = min(n_test, n_total)   # guard

            test_list = full_list[:n_test]
            train_pool = full_list[n_test:]

            # Validation from training portion (10%)
            n_val = max(0, int(round(len(train_pool) * VAL_FRACTION)))
            val_list = train_pool[:n_val]
            train_list = train_pool[n_val:]

            # 4) Materialize (symlink/copy) into folders and write manifests
            # test
            for src_path in test_list:
                fname = os.path.basename(src_path)
                dst_path = os.path.join(test_root, c, fname)
                link_or_copy(src_path, dst_path)
                rows_test.append(("test", c, fname, dst_path))

            # val
            for src_path in val_list:
                fname = os.path.basename(src_path)
                dst_path = os.path.join(val_root, c, fname)
                link_or_copy(src_path, dst_path)
                rows_val.append(("val", c, fname, dst_path))

            # train
            for src_path in train_list:
                fname = os.path.basename(src_path)
                dst_path = os.path.join(train_root, c, fname)
                link_or_copy(src_path, dst_path)
                rows_train.append(("train", c, fname, dst_path))

            # counts
            counts["test"][c]  = len(test_list)
            counts["val"][c]   = len(val_list)
            counts["train"][c] = len(train_list)

        # 5) Write manifest files for this split
        write_manifest(split_root, "train", rows_train)
        write_manifest(split_root, "val",   rows_val)
        write_manifest(split_root, "test",  rows_test)

        # 6) Summary printout
        print(f"\n=== {split_name} ===")
        summarize_counts(counts)
        print(f"Paths:\n  Train: {train_root}\n  Val  : {val_root}\n  Test :␣
 ↪{test_root}\n")

    # 7) Export env var for convenient access in later cells
    os.environ["RESIZED_SPLITS_ROOT"] = SPLITS_ROOT
    print(f"All splits created under: {SPLITS_ROOT}")

# -----------------------------------------
# Execute
# -----------------------------------------
build_splits()
```

```
Total images by class (union of resized/train + resized/test):
  AD: 1124
  CN: 1440
  MCI: 2590

=== split_90_10 ===
Train => total  4175 | AD: 911, CN: 1166, MCI: 2098
Val   => total   464 | AD: 101, CN: 130, MCI: 233
Test  => total   515 | AD: 112, CN: 144, MCI: 259
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_90_10/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_90_10/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_90_10/test


=== split_80_20 ===
Train => total  3711 | AD: 809, CN: 1037, MCI: 1865
Val   => total   412 | AD: 90, CN: 115, MCI: 207
Test  => total  1031 | AD: 225, CN: 288, MCI: 518
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_80_20/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_80_20/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_80_20/test


=== split_70_30 ===
Train => total  3247 | AD: 708, CN: 907, MCI: 1632
Val   => total   361 | AD: 79, CN: 101, MCI: 181
Test  => total  1546 | AD: 337, CN: 432, MCI: 777
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_70_30/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_70_30/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_70_30/test


=== split_60_40 ===
Train => total  2784 | AD: 607, CN: 778, MCI: 1399
Val   => total   308 | AD: 67, CN: 86, MCI: 155
Test  => total  2062 | AD: 450, CN: 576, MCI: 1036
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_60_40/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_60_40/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_60_40/test


=== split_50_50 ===
Train => total  2319 | AD: 506, CN: 648, MCI: 1165
Val   => total   258 | AD: 56, CN: 72, MCI: 130
Test  => total  2577 | AD: 562, CN: 720, MCI: 1295
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_50_50/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_50_50/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_50_50/test
```

```
=== split_40_60 ===
Train => total  1855 | AD: 405, CN: 518, MCI: 932
Val   => total   207 | AD: 45, CN: 58, MCI: 104
Test  => total  3092 | AD: 674, CN: 864, MCI: 1554
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_40_60/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_40_60/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_40_60/test


=== split_30_70 ===
Train => total  1391 | AD: 303, CN: 389, MCI: 699
Val   => total   155 | AD: 34, CN: 43, MCI: 78
Test  => total  3608 | AD: 787, CN: 1008, MCI: 1813
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_30_70/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_30_70/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_30_70/test


=== split_20_80 ===
Train => total   928 | AD: 203, CN: 259, MCI: 466
Val   => total   103 | AD: 22, CN: 29, MCI: 52
Test  => total  4123 | AD: 899, CN: 1152, MCI: 2072
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_20_80/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_20_80/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_20_80/test


=== split_10_90 ===
Train => total   464 | AD: 101, CN: 130, MCI: 233
Val   => total    51 | AD: 11, CN: 14, MCI: 26
Test  => total  4639 | AD: 1012, CN: 1296, MCI: 2331
Paths:
  Train: /kaggle/working/alzheimer-resized-224_splits/split_10_90/train
  Val  : /kaggle/working/alzheimer-resized-224_splits/split_10_90/val
  Test : /kaggle/working/alzheimer-resized-224_splits/split_10_90/test

All splits created under: /kaggle/working/alzheimer-resized-224_splits
```

[5]:
```python
#MobileNetV3-Large
import os
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
import pandas as pd
import numpy as np
from PIL import Image
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪accuracy_score, classification_report
import time
from torch.optim.lr_scheduler import ReduceLROnPlateau

class AlzheimerDataset(Dataset):
    def __init__(self, split_dir, split_type, transform=None):
        self.split_dir = split_dir
        self.split_type = split_type
        self.transform = transform

        csv_path = os.path.join(split_dir, f"{split_type}.csv")
        self.df = pd.read_csv(csv_path)

        self.class_to_idx = {"AD": 0, "CN": 1, "MCI": 2}
        self.idx_to_class = {v: k for k, v in self.class_to_idx.items()}

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img_path = row['path']
        label = self.class_to_idx[row['class']]

        image = Image.open(img_path).convert('RGB')

        if self.transform:
            image = self.transform(image)

        return image, label

def get_data_transforms():
    train_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomRotation(10),
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,␣
 ↪hue=0.1),
        transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225])
```

```
    ])

    val_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
↪225])
    ])

    return train_transform, val_transform

def train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs, device):
    best_val_acc = 0
    patience = 5
    patience_counter = 0

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        for batch_idx, (images, labels) in enumerate(train_loader):
            images, labels = images.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(train_loader)
        epoch_acc = 100 * correct / total

        val_acc = evaluate_model(model, val_loader, device)

        if scheduler:
            scheduler.step(val_acc)

        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Train␣
↪Acc: {epoch_acc:.2f}%, Val Acc: {val_acc:.2f}%')
```

```
        if val_acc > best_val_acc:
            best_val_acc = val_acc
            patience_counter = 0
        else:
            patience_counter += 1

        if patience_counter >= patience:
            print(f"Early stopping at epoch {epoch+1}")
            break

def evaluate_model(model, data_loader, device):
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in data_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    return accuracy

def test_model(model, test_loader, device):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)

            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    return all_preds, all_labels

def calculate_metrics(y_true, y_pred, split_name):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted',␣
 ↪zero_division=0)
```

```
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    print(f"\n=== {split_name} Results ===")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred, target_names=['AD', 'CN',␣
 ↪'MCI'], zero_division=0))

    return {
        'split': split_name,
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1_score': f1
    }

def run_mobilenetv3_large_on_splits():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    print(f"Using device: {device}")

    splits_root = "/kaggle/working/alzheimer-resized-224_splits"
    split_folders = [f for f in os.listdir(splits_root) if f.
 ↪startswith('split_')]
    split_folders.sort()

    results = []

    train_transform, val_transform = get_data_transforms()

    for split_folder in split_folders:
        print(f"\n{'='*60}")
        print(f"Processing: {split_folder}")
        print(f"{'='*60}")

        split_path = os.path.join(splits_root, split_folder)

        train_dataset = AlzheimerDataset(split_path, 'train', train_transform)
        val_dataset = AlzheimerDataset(split_path, 'val', val_transform)
        test_dataset = AlzheimerDataset(split_path, 'test', val_transform)

        train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True,␣
 ↪num_workers=2)
```

```
        val_loader = DataLoader(val_dataset, batch_size=16, shuffle=False,␣
↪num_workers=2)
        test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False,␣
↪num_workers=2)

        print(f"Train samples: {len(train_dataset)}")
        print(f"Val samples: {len(val_dataset)}")
        print(f"Test samples: {len(test_dataset)}")

        model = models.mobilenet_v3_large(weights=models.
↪MobileNet_V3_Large_Weights.IMAGENET1K_V2)
        num_ftrs = model.classifier[3].in_features
        model.classifier[3] = nn.Linear(num_ftrs, 3)
        model = model.to(device)

        criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
        optimizer = torch.optim.AdamW(model.parameters(), lr=0.0001,␣
↪weight_decay=1e-4)
        scheduler = ReduceLROnPlateau(optimizer, mode='max', factor=0.5,␣
↪patience=3, verbose=True)

        print("Starting training...")
        start_time = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs=30, device=device)
        training_time = time.time() - start_time

        print("Testing model...")
        test_preds, test_labels = test_model(model, test_loader, device)

        split_results = calculate_metrics(test_labels, test_preds, split_folder)
        split_results['training_time'] = training_time
        results.append(split_results)

        torch.cuda.empty_cache()

    results_df = pd.DataFrame(results)
    print(f"\n{'='*80}")
    print("MobileNetV3-Large - SUMMARY OF ALL SPLITS")
    print(f"{'='*80}")
    print(results_df.to_string(index=False))

    results_csv_path = "/kaggle/working/mobilenetv3_large_results.csv"
    results_df.to_csv(results_csv_path, index=False)
    print(f"\nDetailed results saved to: {results_csv_path}")
    return results_df

if __name__ == "__main__":
    results = run_mobilenetv3_large_on_splits()
```

```
Using device: cuda

============================================================
Processing: split_10_90
============================================================
Train samples: 464
Val samples: 51
Test samples: 4639

Downloading:
"https://download.pytorch.org/models/mobilenet_v3_large-5c1a4163.pth" to
/root/.cache/torch/hub/checkpoints/mobilenet_v3_large-5c1a4163.pth
100%|      | 21.1M/21.1M [00:00<00:00, 232MB/s]

Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0622, Train Acc: 48.49%, Val Acc: 52.94%
Epoch [2/30], Loss: 1.0318, Train Acc: 50.43%, Val Acc: 54.90%
Epoch [3/30], Loss: 1.0103, Train Acc: 51.94%, Val Acc: 60.78%
Epoch [4/30], Loss: 0.9779, Train Acc: 54.09%, Val Acc: 58.82%
Epoch [5/30], Loss: 0.9336, Train Acc: 58.84%, Val Acc: 60.78%
Epoch [6/30], Loss: 0.9025, Train Acc: 62.72%, Val Acc: 60.78%
Epoch [7/30], Loss: 0.8577, Train Acc: 66.59%, Val Acc: 56.86%
Epoch [8/30], Loss: 0.7784, Train Acc: 71.98%, Val Acc: 52.94%
Early stopping at epoch 8
Testing model…

=== split_10_90 Results ===
Accuracy: 0.5115
Precision: 0.4727
Recall: 0.5115
F1-Score: 0.4502

Classification Report:
              precision    recall  f1-score   support

          AD       0.37      0.08      0.14      1012
          CN       0.43      0.26      0.33      1296
```

```
         MCI       0.54      0.83      0.65      2331

    accuracy                           0.51      4639
   macro avg       0.45      0.39      0.37      4639
weighted avg       0.47      0.51      0.45      4639


============================================================
Processing: split_20_80
============================================================
Train samples: 928
Val samples: 103
Test samples: 4123
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0566, Train Acc: 48.92%, Val Acc: 50.49%
Epoch [2/30], Loss: 1.0221, Train Acc: 51.51%, Val Acc: 52.43%
Epoch [3/30], Loss: 0.9887, Train Acc: 52.16%, Val Acc: 55.34%
Epoch [4/30], Loss: 0.9414, Train Acc: 57.22%, Val Acc: 50.49%
Epoch [5/30], Loss: 0.9113, Train Acc: 59.91%, Val Acc: 52.43%
Epoch [6/30], Loss: 0.8799, Train Acc: 62.18%, Val Acc: 48.54%
Epoch [7/30], Loss: 0.8156, Train Acc: 68.00%, Val Acc: 60.19%
Epoch [8/30], Loss: 0.7624, Train Acc: 72.09%, Val Acc: 57.28%
Epoch [9/30], Loss: 0.7450, Train Acc: 73.17%, Val Acc: 56.31%
Epoch [10/30], Loss: 0.7020, Train Acc: 76.62%, Val Acc: 68.93%
Epoch [11/30], Loss: 0.7098, Train Acc: 74.35%, Val Acc: 69.90%
Epoch [12/30], Loss: 0.6342, Train Acc: 79.53%, Val Acc: 66.99%
Epoch [13/30], Loss: 0.6226, Train Acc: 80.82%, Val Acc: 55.34%
Epoch [14/30], Loss: 0.6073, Train Acc: 81.57%, Val Acc: 69.90%
Epoch [15/30], Loss: 0.5641, Train Acc: 84.81%, Val Acc: 77.67%
Epoch [16/30], Loss: 0.5719, Train Acc: 84.38%, Val Acc: 77.67%
Epoch [17/30], Loss: 0.5659, Train Acc: 84.48%, Val Acc: 64.08%
Epoch [18/30], Loss: 0.5448, Train Acc: 86.85%, Val Acc: 70.87%
Epoch [19/30], Loss: 0.5255, Train Acc: 87.72%, Val Acc: 79.61%
Epoch [20/30], Loss: 0.4976, Train Acc: 90.19%, Val Acc: 79.61%
Epoch [21/30], Loss: 0.5258, Train Acc: 87.18%, Val Acc: 78.64%
Epoch [22/30], Loss: 0.5110, Train Acc: 89.44%, Val Acc: 75.73%
Epoch [23/30], Loss: 0.4918, Train Acc: 90.41%, Val Acc: 80.58%
Epoch [24/30], Loss: 0.4656, Train Acc: 91.92%, Val Acc: 79.61%
Epoch [25/30], Loss: 0.4562, Train Acc: 91.81%, Val Acc: 67.96%
Epoch [26/30], Loss: 0.4681, Train Acc: 90.19%, Val Acc: 83.50%
Epoch [27/30], Loss: 0.4442, Train Acc: 93.21%, Val Acc: 86.41%
Epoch [28/30], Loss: 0.4237, Train Acc: 93.75%, Val Acc: 83.50%
Epoch [29/30], Loss: 0.4395, Train Acc: 93.32%, Val Acc: 84.47%
Epoch [30/30], Loss: 0.4294, Train Acc: 94.29%, Val Acc: 84.47%
Testing model…

=== split_20_80 Results ===
Accuracy: 0.7924
Precision: 0.8025
Recall: 0.7924
F1-Score: 0.7944

Classification Report:
              precision    recall  f1-score   support

          AD       0.64      0.78      0.71       899
          CN       0.86      0.73      0.79      1152
         MCI       0.84      0.83      0.83      2072

    accuracy                           0.79      4123
   macro avg       0.78      0.78      0.78      4123
weighted avg       0.80      0.79      0.79      4123


============================================================
Processing: split_30_70
============================================================
Train samples: 1391
Val samples: 155
Test samples: 3608
Starting training…
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
Epoch [1/30], Loss: 1.0405, Train Acc: 50.04%, Val Acc: 52.90%
Epoch [2/30], Loss: 0.9956, Train Acc: 54.13%, Val Acc: 51.61%
Epoch [3/30], Loss: 0.9541, Train Acc: 57.15%, Val Acc: 50.32%
Epoch [4/30], Loss: 0.8840, Train Acc: 62.83%, Val Acc: 54.19%
Epoch [5/30], Loss: 0.8246, Train Acc: 66.28%, Val Acc: 58.71%
Epoch [6/30], Loss: 0.7949, Train Acc: 70.24%, Val Acc: 61.94%
Epoch [7/30], Loss: 0.7261, Train Acc: 75.84%, Val Acc: 62.58%
Epoch [8/30], Loss: 0.7022, Train Acc: 75.20%, Val Acc: 65.16%
Epoch [9/30], Loss: 0.6920, Train Acc: 75.70%, Val Acc: 74.19%
Epoch [10/30], Loss: 0.6347, Train Acc: 79.87%, Val Acc: 72.90%
Epoch [11/30], Loss: 0.6058, Train Acc: 83.32%, Val Acc: 67.74%
Epoch [12/30], Loss: 0.5822, Train Acc: 83.82%, Val Acc: 69.03%
Epoch [13/30], Loss: 0.5747, Train Acc: 84.90%, Val Acc: 76.77%
Epoch [14/30], Loss: 0.5610, Train Acc: 83.97%, Val Acc: 78.71%
Epoch [15/30], Loss: 0.5419, Train Acc: 87.71%, Val Acc: 78.71%
Epoch [16/30], Loss: 0.5313, Train Acc: 86.99%, Val Acc: 83.87%
Epoch [17/30], Loss: 0.5131, Train Acc: 88.71%, Val Acc: 80.65%
Epoch [18/30], Loss: 0.4967, Train Acc: 90.22%, Val Acc: 81.94%
Epoch [19/30], Loss: 0.4819, Train Acc: 90.73%, Val Acc: 83.23%
Epoch [20/30], Loss: 0.4839, Train Acc: 90.73%, Val Acc: 78.71%
Epoch [21/30], Loss: 0.4401, Train Acc: 93.89%, Val Acc: 83.87%
Early stopping at epoch 21
Testing model…

=== split_30_70 Results ===
Accuracy: 0.8428
Precision: 0.8439
Recall: 0.8428
F1-Score: 0.8400

Classification Report:
              precision    recall  f1-score   support

          AD       0.81      0.70      0.75       787
          CN       0.88      0.78      0.83      1008
         MCI       0.83      0.94      0.88      1813

    accuracy                           0.84      3608
   macro avg       0.84      0.81      0.82      3608
weighted avg       0.84      0.84      0.84      3608


============================================================
Processing: split_40_60
============================================================
Train samples: 1855
Val samples: 207
Test samples: 3092
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0254, Train Acc: 51.05%, Val Acc: 47.34%
Epoch [2/30], Loss: 0.9638, Train Acc: 57.68%, Val Acc: 49.76%
Epoch [3/30], Loss: 0.8876, Train Acc: 62.80%, Val Acc: 51.21%
Epoch [4/30], Loss: 0.8347, Train Acc: 67.12%, Val Acc: 61.35%
Epoch [5/30], Loss: 0.7704, Train Acc: 71.86%, Val Acc: 67.63%
Epoch [6/30], Loss: 0.7101, Train Acc: 74.72%, Val Acc: 72.95%
Epoch [7/30], Loss: 0.6867, Train Acc: 77.79%, Val Acc: 80.19%
Epoch [8/30], Loss: 0.6256, Train Acc: 81.99%, Val Acc: 74.88%
Epoch [9/30], Loss: 0.6156, Train Acc: 82.16%, Val Acc: 77.29%
Epoch [10/30], Loss: 0.5918, Train Acc: 83.18%, Val Acc: 75.85%
Epoch [11/30], Loss: 0.5921, Train Acc: 83.02%, Val Acc: 75.85%
Epoch [12/30], Loss: 0.5159, Train Acc: 87.76%, Val Acc: 83.09%
Epoch [13/30], Loss: 0.5231, Train Acc: 88.63%, Val Acc: 83.09%
Epoch [14/30], Loss: 0.5018, Train Acc: 88.73%, Val Acc: 82.61%
Epoch [15/30], Loss: 0.4832, Train Acc: 90.24%, Val Acc: 81.64%
Epoch [16/30], Loss: 0.4802, Train Acc: 90.03%, Val Acc: 83.09%
Epoch [17/30], Loss: 0.4637, Train Acc: 92.40%, Val Acc: 85.02%
Epoch [18/30], Loss: 0.4555, Train Acc: 92.78%, Val Acc: 86.47%
Epoch [19/30], Loss: 0.4557, Train Acc: 92.29%, Val Acc: 85.99%
Epoch [20/30], Loss: 0.4453, Train Acc: 92.72%, Val Acc: 85.51%
Epoch [21/30], Loss: 0.4485, Train Acc: 92.35%, Val Acc: 85.51%
Epoch [22/30], Loss: 0.4460, Train Acc: 93.10%, Val Acc: 85.99%
Epoch [23/30], Loss: 0.4323, Train Acc: 93.48%, Val Acc: 83.57%
Early stopping at epoch 23
Testing model…

=== split_40_60 Results ===
Accuracy: 0.8852
Precision: 0.8877
Recall: 0.8852
F1-Score: 0.8854

Classification Report:
              precision    recall  f1-score   support

          AD       0.80      0.86      0.83       674
          CN       0.93      0.84      0.88       864
         MCI       0.90      0.92      0.91      1554

    accuracy                           0.89      3092
   macro avg       0.88      0.87      0.87      3092
weighted avg       0.89      0.89      0.89      3092


============================================================
Processing: split_50_50
============================================================
Train samples: 2319
Val samples: 258
Test samples: 2577
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0314, Train Acc: 50.37%, Val Acc: 46.12%
Epoch [2/30], Loss: 0.9639, Train Acc: 56.53%, Val Acc: 54.26%
Epoch [3/30], Loss: 0.9005, Train Acc: 61.23%, Val Acc: 52.33%
Epoch [4/30], Loss: 0.8187, Train Acc: 67.31%, Val Acc: 65.12%
Epoch [5/30], Loss: 0.7909, Train Acc: 69.64%, Val Acc: 71.71%
Epoch [6/30], Loss: 0.7151, Train Acc: 75.33%, Val Acc: 70.54%
Epoch [7/30], Loss: 0.6820, Train Acc: 77.19%, Val Acc: 70.93%
Epoch [8/30], Loss: 0.6321, Train Acc: 80.85%, Val Acc: 79.84%
Epoch [9/30], Loss: 0.6110, Train Acc: 82.92%, Val Acc: 79.07%
Epoch [10/30], Loss: 0.5798, Train Acc: 84.26%, Val Acc: 81.40%
Epoch [11/30], Loss: 0.5572, Train Acc: 85.68%, Val Acc: 76.36%
Epoch [12/30], Loss: 0.5387, Train Acc: 87.67%, Val Acc: 84.50%
Epoch [13/30], Loss: 0.5120, Train Acc: 88.27%, Val Acc: 90.70%
Epoch [14/30], Loss: 0.5145, Train Acc: 88.79%, Val Acc: 87.21%
Epoch [15/30], Loss: 0.4898, Train Acc: 90.21%, Val Acc: 86.82%
Epoch [16/30], Loss: 0.4773, Train Acc: 90.90%, Val Acc: 89.15%
Epoch [17/30], Loss: 0.4600, Train Acc: 91.89%, Val Acc: 87.21%
Epoch [18/30], Loss: 0.4137, Train Acc: 95.08%, Val Acc: 90.70%
Early stopping at epoch 18
Testing model…

=== split_50_50 Results ===
Accuracy: 0.9166
Precision: 0.9174
Recall: 0.9166
F1-Score: 0.9169

Classification Report:
              precision    recall  f1-score   support

          AD       0.86      0.89      0.87       562
          CN       0.91      0.91      0.91       720
         MCI       0.95      0.93      0.94      1295

    accuracy                           0.92      2577
   macro avg       0.91      0.91      0.91      2577
weighted avg       0.92      0.92      0.92      2577


============================================================
Processing: split_60_40
============================================================
Train samples: 2784
Val samples: 308
Test samples: 2062
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
```

```
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0283, Train Acc: 51.54%, Val Acc: 52.27%
Epoch [2/30], Loss: 0.9560, Train Acc: 56.97%, Val Acc: 50.97%
Epoch [3/30], Loss: 0.8718, Train Acc: 64.91%, Val Acc: 53.57%
Epoch [4/30], Loss: 0.7890, Train Acc: 69.47%, Val Acc: 62.66%
Epoch [5/30], Loss: 0.7321, Train Acc: 75.04%, Val Acc: 70.13%
Epoch [6/30], Loss: 0.6875, Train Acc: 77.33%, Val Acc: 66.56%
Epoch [7/30], Loss: 0.6512, Train Acc: 79.06%, Val Acc: 69.81%
Epoch [8/30], Loss: 0.6078, Train Acc: 83.08%, Val Acc: 76.95%
Epoch [9/30], Loss: 0.5888, Train Acc: 83.26%, Val Acc: 82.14%
Epoch [10/30], Loss: 0.5626, Train Acc: 84.95%, Val Acc: 80.52%
Epoch [11/30], Loss: 0.5253, Train Acc: 87.57%, Val Acc: 83.12%
Epoch [12/30], Loss: 0.5163, Train Acc: 88.07%, Val Acc: 86.69%
Epoch [13/30], Loss: 0.4899, Train Acc: 89.51%, Val Acc: 86.04%
Epoch [14/30], Loss: 0.4775, Train Acc: 90.62%, Val Acc: 89.29%
Epoch [15/30], Loss: 0.4583, Train Acc: 91.56%, Val Acc: 89.29%
Epoch [16/30], Loss: 0.4610, Train Acc: 92.06%, Val Acc: 85.71%
Epoch [17/30], Loss: 0.4245, Train Acc: 93.43%, Val Acc: 87.66%
Epoch [18/30], Loss: 0.4257, Train Acc: 94.00%, Val Acc: 88.31%
Epoch [19/30], Loss: 0.4078, Train Acc: 95.47%, Val Acc: 91.23%
Epoch [20/30], Loss: 0.3900, Train Acc: 95.65%, Val Acc: 90.58%
Epoch [21/30], Loss: 0.3851, Train Acc: 96.48%, Val Acc: 90.91%
Epoch [22/30], Loss: 0.3765, Train Acc: 96.62%, Val Acc: 91.88%
Epoch [23/30], Loss: 0.3724, Train Acc: 96.80%, Val Acc: 92.21%
Epoch [24/30], Loss: 0.3626, Train Acc: 97.52%, Val Acc: 93.51%
Epoch [25/30], Loss: 0.3707, Train Acc: 96.98%, Val Acc: 93.18%
Epoch [26/30], Loss: 0.3571, Train Acc: 97.49%, Val Acc: 93.18%
Epoch [27/30], Loss: 0.3680, Train Acc: 96.88%, Val Acc: 93.51%
Epoch [28/30], Loss: 0.3582, Train Acc: 97.56%, Val Acc: 93.51%
Epoch [29/30], Loss: 0.3488, Train Acc: 97.81%, Val Acc: 94.48%
Epoch [30/30], Loss: 0.3432, Train Acc: 98.38%, Val Acc: 95.13%
Testing model…

=== split_60_40 Results ===
Accuracy: 0.9685
Precision: 0.9685
Recall: 0.9685
F1-Score: 0.9685

Classification Report:
              precision    recall  f1-score   support

          AD       0.96      0.96      0.96       450
          CN       0.96      0.96      0.96       576
         MCI       0.98      0.98      0.98      1036
```

```
    accuracy                           0.97      2062
   macro avg       0.97      0.97      0.97      2062
weighted avg       0.97      0.97      0.97      2062


============================================================
Processing: split_70_30
============================================================
Train samples: 3247
Val samples: 361
Test samples: 1546
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0272, Train Acc: 51.09%, Val Acc: 46.54%
Epoch [2/30], Loss: 0.9280, Train Acc: 60.18%, Val Acc: 55.68%
Epoch [3/30], Loss: 0.8179, Train Acc: 68.37%, Val Acc: 55.68%
Epoch [4/30], Loss: 0.7353, Train Acc: 73.51%, Val Acc: 70.08%
Epoch [5/30], Loss: 0.6736, Train Acc: 78.10%, Val Acc: 79.78%
Epoch [6/30], Loss: 0.6368, Train Acc: 80.66%, Val Acc: 69.53%
Epoch [7/30], Loss: 0.6031, Train Acc: 82.41%, Val Acc: 86.15%
Epoch [8/30], Loss: 0.5682, Train Acc: 84.94%, Val Acc: 87.81%
Epoch [9/30], Loss: 0.5265, Train Acc: 88.30%, Val Acc: 90.86%
Epoch [10/30], Loss: 0.5201, Train Acc: 88.14%, Val Acc: 87.81%
Epoch [11/30], Loss: 0.4897, Train Acc: 90.27%, Val Acc: 89.47%
Epoch [12/30], Loss: 0.4798, Train Acc: 90.08%, Val Acc: 92.52%
Epoch [13/30], Loss: 0.4575, Train Acc: 91.56%, Val Acc: 86.43%
Epoch [14/30], Loss: 0.4449, Train Acc: 92.67%, Val Acc: 90.86%
Epoch [15/30], Loss: 0.4288, Train Acc: 93.69%, Val Acc: 94.18%
Epoch [16/30], Loss: 0.4170, Train Acc: 94.18%, Val Acc: 93.91%
Epoch [17/30], Loss: 0.4178, Train Acc: 93.93%, Val Acc: 94.18%
Epoch [18/30], Loss: 0.3965, Train Acc: 95.60%, Val Acc: 93.35%
Epoch [19/30], Loss: 0.3900, Train Acc: 95.78%, Val Acc: 93.63%
Epoch [20/30], Loss: 0.3663, Train Acc: 97.14%, Val Acc: 94.18%
Early stopping at epoch 20
Testing model…

=== split_70_30 Results ===
Accuracy: 0.9605
Precision: 0.9605
Recall: 0.9605
F1-Score: 0.9605

Classification Report:
              precision    recall  f1-score   support
```

```
          AD       0.95      0.94      0.95       337
          CN       0.96      0.95      0.95       432
         MCI       0.96      0.98      0.97       777

    accuracy                           0.96      1546
   macro avg       0.96      0.95      0.96      1546
weighted avg       0.96      0.96      0.96      1546


============================================================
Processing: split_80_20
============================================================
Train samples: 3711
Val samples: 412
Test samples: 1031
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0171, Train Acc: 51.31%, Val Acc: 49.51%
Epoch [2/30], Loss: 0.9045, Train Acc: 61.79%, Val Acc: 55.58%
Epoch [3/30], Loss: 0.8193, Train Acc: 67.64%, Val Acc: 67.96%
Epoch [4/30], Loss: 0.7369, Train Acc: 73.30%, Val Acc: 75.97%
Epoch [5/30], Loss: 0.6688, Train Acc: 77.90%, Val Acc: 82.28%
Epoch [6/30], Loss: 0.6242, Train Acc: 81.60%, Val Acc: 81.31%
Epoch [7/30], Loss: 0.5848, Train Acc: 83.35%, Val Acc: 81.55%
Epoch [8/30], Loss: 0.5497, Train Acc: 86.15%, Val Acc: 89.32%
Epoch [9/30], Loss: 0.5305, Train Acc: 86.82%, Val Acc: 84.47%
Epoch [10/30], Loss: 0.4962, Train Acc: 89.60%, Val Acc: 93.20%
Epoch [11/30], Loss: 0.4708, Train Acc: 91.22%, Val Acc: 95.15%
Epoch [12/30], Loss: 0.4611, Train Acc: 91.84%, Val Acc: 91.99%
Epoch [13/30], Loss: 0.4465, Train Acc: 92.45%, Val Acc: 94.17%
Epoch [14/30], Loss: 0.4262, Train Acc: 93.83%, Val Acc: 96.12%
Epoch [15/30], Loss: 0.4195, Train Acc: 93.88%, Val Acc: 96.36%
Epoch [16/30], Loss: 0.4064, Train Acc: 94.53%, Val Acc: 93.69%
Epoch [17/30], Loss: 0.3923, Train Acc: 95.39%, Val Acc: 95.39%
Epoch [18/30], Loss: 0.3887, Train Acc: 96.09%, Val Acc: 93.69%
Epoch [19/30], Loss: 0.3842, Train Acc: 95.88%, Val Acc: 96.60%
Epoch [20/30], Loss: 0.3712, Train Acc: 96.82%, Val Acc: 96.36%
Epoch [21/30], Loss: 0.3680, Train Acc: 96.71%, Val Acc: 96.84%
Epoch [22/30], Loss: 0.3608, Train Acc: 97.36%, Val Acc: 96.12%
Epoch [23/30], Loss: 0.3659, Train Acc: 96.85%, Val Acc: 97.57%
Epoch [24/30], Loss: 0.3537, Train Acc: 97.36%, Val Acc: 95.87%
Epoch [25/30], Loss: 0.3450, Train Acc: 97.93%, Val Acc: 96.84%
Epoch [26/30], Loss: 0.3464, Train Acc: 97.98%, Val Acc: 96.84%
Epoch [27/30], Loss: 0.3397, Train Acc: 98.09%, Val Acc: 96.36%
Epoch [28/30], Loss: 0.3310, Train Acc: 98.46%, Val Acc: 98.54%
Epoch [29/30], Loss: 0.3241, Train Acc: 98.81%, Val Acc: 98.79%
Epoch [30/30], Loss: 0.3236, Train Acc: 98.90%, Val Acc: 98.06%
Testing model…

=== split_80_20 Results ===
Accuracy: 0.9748
Precision: 0.9749
Recall: 0.9748
F1-Score: 0.9748

Classification Report:
              precision    recall  f1-score   support

          AD       0.96      0.97      0.97       225
          CN       0.97      0.99      0.98       288
         MCI       0.98      0.97      0.98       518

    accuracy                           0.97      1031
   macro avg       0.97      0.98      0.97      1031
weighted avg       0.97      0.97      0.97      1031


============================================================
Processing: split_90_10
============================================================
Train samples: 4175
Val samples: 464
Test samples: 515
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/30], Loss: 1.0039, Train Acc: 53.41%, Val Acc: 52.16%
Epoch [2/30], Loss: 0.8793, Train Acc: 63.23%, Val Acc: 65.95%
Epoch [3/30], Loss: 0.7851, Train Acc: 69.37%, Val Acc: 75.86%
Epoch [4/30], Loss: 0.7055, Train Acc: 75.47%, Val Acc: 76.94%
Epoch [5/30], Loss: 0.6317, Train Acc: 80.81%, Val Acc: 83.62%
Epoch [6/30], Loss: 0.5990, Train Acc: 83.07%, Val Acc: 84.70%
Epoch [7/30], Loss: 0.5555, Train Acc: 86.18%, Val Acc: 84.70%
Epoch [8/30], Loss: 0.5252, Train Acc: 87.74%, Val Acc: 86.21%
Epoch [9/30], Loss: 0.4964, Train Acc: 89.70%, Val Acc: 90.09%
Epoch [10/30], Loss: 0.4775, Train Acc: 91.11%, Val Acc: 92.24%
Epoch [11/30], Loss: 0.4542, Train Acc: 92.26%, Val Acc: 93.53%
Epoch [12/30], Loss: 0.4328, Train Acc: 93.46%, Val Acc: 90.95%
Epoch [13/30], Loss: 0.4222, Train Acc: 93.94%, Val Acc: 93.32%
Epoch [14/30], Loss: 0.4200, Train Acc: 93.70%, Val Acc: 93.53%
Epoch [15/30], Loss: 0.3965, Train Acc: 95.23%, Val Acc: 96.34%
Epoch [16/30], Loss: 0.3815, Train Acc: 96.10%, Val Acc: 93.75%
Epoch [17/30], Loss: 0.3806, Train Acc: 96.14%, Val Acc: 96.12%
Epoch [18/30], Loss: 0.3681, Train Acc: 96.81%, Val Acc: 95.26%
Epoch [19/30], Loss: 0.3640, Train Acc: 97.05%, Val Acc: 95.47%
Epoch [20/30], Loss: 0.3469, Train Acc: 97.89%, Val Acc: 96.98%
Epoch [21/30], Loss: 0.3412, Train Acc: 98.01%, Val Acc: 96.55%
Epoch [22/30], Loss: 0.3375, Train Acc: 98.20%, Val Acc: 96.55%
Epoch [23/30], Loss: 0.3342, Train Acc: 98.44%, Val Acc: 96.77%
Epoch [24/30], Loss: 0.3276, Train Acc: 99.02%, Val Acc: 95.69%
Epoch [25/30], Loss: 0.3257, Train Acc: 98.92%, Val Acc: 97.41%
Epoch [26/30], Loss: 0.3205, Train Acc: 99.33%, Val Acc: 96.34%
Epoch [27/30], Loss: 0.3215, Train Acc: 99.04%, Val Acc: 97.41%
Epoch [28/30], Loss: 0.3206, Train Acc: 99.07%, Val Acc: 97.41%
Epoch [29/30], Loss: 0.3194, Train Acc: 99.02%, Val Acc: 97.63%
Epoch [30/30], Loss: 0.3171, Train Acc: 99.28%, Val Acc: 97.84%
Testing model…

=== split_90_10 Results ===
Accuracy: 0.9825
Precision: 0.9825
Recall: 0.9825
F1-Score: 0.9825

Classification Report:
              precision    recall  f1-score   support

          AD       0.97      0.96      0.97       112
          CN       0.98      0.99      0.98       144
         MCI       0.99      0.99      0.99       259

    accuracy                           0.98       515
   macro avg       0.98      0.98      0.98       515
weighted avg       0.98      0.98      0.98       515


================================================================================
MobileNetV3-Large - SUMMARY OF ALL SPLITS
================================================================================
      split  accuracy  precision   recall  f1_score  training_time
split_10_90  0.511533   0.472666 0.511533  0.450184      15.468586
split_20_80  0.792384   0.802541 0.792384  0.794430     101.326068
split_30_70  0.842849   0.843855 0.842849  0.839961     104.212670
split_40_60  0.885188   0.887740 0.885188  0.885444     150.704233
split_50_50  0.916570   0.917441 0.916570  0.916873     144.666528
split_60_40  0.968477   0.968474 0.968477  0.968474     289.130308
```

```
split_70_30  0.960543   0.960498 0.960543  0.960462     223.756592
split_80_20  0.974782   0.974926 0.974782  0.974791     382.935752
split_90_10  0.982524   0.982512 0.982524  0.982511     429.871090

Detailed results saved to: /kaggle/working/mobilenetv3_large_results.csv
```

[6]:
```
#ConvNeXt-Small
import os
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
import pandas as pd
import numpy as np
from PIL import Image
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪accuracy_score, classification_report
import time
from torch.optim.lr_scheduler import ReduceLROnPlateau

class AlzheimerDataset(Dataset):
    def __init__(self, split_dir, split_type, transform=None):
        self.split_dir = split_dir
        self.split_type = split_type
        self.transform = transform

        csv_path = os.path.join(split_dir, f"{split_type}.csv")
        self.df = pd.read_csv(csv_path)

        self.class_to_idx = {"AD": 0, "CN": 1, "MCI": 2}
        self.idx_to_class = {v: k for k, v in self.class_to_idx.items()}

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img_path = row['path']
        label = self.class_to_idx[row['class']]

        image = Image.open(img_path).convert('RGB')

        if self.transform:
            image = self.transform(image)

        return image, label
```

```
def get_data_transforms():
    train_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomRotation(10),
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,␣
 ↪hue=0.1),
        transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
        transforms.RandomGrayscale(p=0.1),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225])
    ])

    val_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225])
    ])

    return train_transform, val_transform

def train_model(model, train_loader, val_loader, criterion, optimizer,␣
 ↪scheduler, num_epochs, device):
    best_val_acc = 0
    patience = 7
    patience_counter = 0

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        for batch_idx, (images, labels) in enumerate(train_loader):
            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()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(train_loader)
        epoch_acc = 100 * correct / total

        val_acc = evaluate_model(model, val_loader, device)

        if scheduler:
            scheduler.step(val_acc)

        current_lr = optimizer.param_groups[0]['lr']
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Train␣
↪Acc: {epoch_acc:.2f}%, Val Acc: {val_acc:.2f}%, LR: {current_lr:.6f}')

        if val_acc > best_val_acc:
            best_val_acc = val_acc
            patience_counter = 0
        else:
            patience_counter += 1

        if patience_counter >= patience:
            print(f"Early stopping at epoch {epoch+1}")
            break

def evaluate_model(model, data_loader, device):
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in data_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    return accuracy

def test_model(model, test_loader, device):
    model.eval()
    all_preds = []
    all_labels = []
    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)

            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    return all_preds, all_labels

def calculate_metrics(y_true, y_pred, split_name):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted',␣
↪zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    print(f"\n=== {split_name} Results ===")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred, target_names=['AD', 'CN',␣
↪'MCI'], zero_division=0))

    return {
        'split': split_name,
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1_score': f1
    }

def run_convnext_small_on_splits():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    print(f"Using device: {device}")

    splits_root = "/kaggle/working/alzheimer-resized-224_splits"
    split_folders = [f for f in os.listdir(splits_root) if f.
↪startswith('split_')]
    split_folders.sort()

    results = []

    train_transform, val_transform = get_data_transforms()
    for split_folder in split_folders:
        print(f"\n{'='*60}")
        print(f"Processing: {split_folder}")
        print(f"{'='*60}")

        split_path = os.path.join(splits_root, split_folder)

        train_dataset = AlzheimerDataset(split_path, 'train', train_transform)
        val_dataset = AlzheimerDataset(split_path, 'val', val_transform)
        test_dataset = AlzheimerDataset(split_path, 'test', val_transform)

        train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True,␣
↪num_workers=2, pin_memory=True)
        val_loader = DataLoader(val_dataset, batch_size=16, shuffle=False,␣
↪num_workers=2, pin_memory=True)
        test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False,␣
↪num_workers=2, pin_memory=True)

        print(f"Train samples: {len(train_dataset)}")
        print(f"Val samples: {len(val_dataset)}")
        print(f"Test samples: {len(test_dataset)}")

        model = models.convnext_small(weights=models.ConvNeXt_Small_Weights.
↪IMAGENET1K_V1)
        num_ftrs = model.classifier[2].in_features
        model.classifier[2] = nn.Linear(num_ftrs, 3)

        for param in model.parameters():
            param.requires_grad = False

        for param in model.classifier.parameters():
            param.requires_grad = True

        for param in model.features[-2:].parameters():
            param.requires_grad = True

        model = model.to(device)

        criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
        optimizer = torch.optim.AdamW([
            {'params': model.classifier.parameters(), 'lr': 0.0001},
            {'params': model.features[-2:].parameters(), 'lr': 0.00005}
        ], weight_decay=1e-4)

        scheduler = ReduceLROnPlateau(optimizer, mode='max', factor=0.5,␣
↪patience=3, verbose=True)
```

```
        print("Starting training...")
        start_time = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs=25, device=device)
        training_time = time.time() - start_time

        print("Fine-tuning all layers...")
        for param in model.parameters():
            param.requires_grad = True

        optimizer = torch.optim.AdamW(model.parameters(), lr=0.00001,␣
↪weight_decay=1e-5)
        scheduler = ReduceLROnPlateau(optimizer, mode='max', factor=0.5,␣
↪patience=2, verbose=True)

        start_time_finetune = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs=10, device=device)
        training_time += time.time() - start_time_finetune

        print("Testing model...")
        test_preds, test_labels = test_model(model, test_loader, device)

        split_results = calculate_metrics(test_labels, test_preds, split_folder)
        split_results['training_time'] = training_time
        results.append(split_results)

        torch.cuda.empty_cache()

    results_df = pd.DataFrame(results)
    print(f"\n{'='*80}")
    print("ConvNeXt-Small - SUMMARY OF ALL SPLITS")
    print(f"{'='*80}")
    print(results_df.to_string(index=False))

    results_csv_path = "/kaggle/working/convnext_small_results.csv"
    results_df.to_csv(results_csv_path, index=False)
    print(f"\nDetailed results saved to: {results_csv_path}")

    return results_df

if __name__ == "__main__":
    results = run_convnext_small_on_splits()
```

```
Using device: cuda
```

```
============================================================
Processing: split_10_90
============================================================
Train samples: 464
Val samples: 51
Test samples: 4639

Downloading: "https://download.pytorch.org/models/convnext_small-0c510722.pth"
to /root/.cache/torch/hub/checkpoints/convnext_small-0c510722.pth
100%|      | 192M/192M [00:00<00:00, 236MB/s]

Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0681, Train Acc: 48.71%, Val Acc: 49.02%, LR: 0.000100
Epoch [2/25], Loss: 1.0323, Train Acc: 51.72%, Val Acc: 49.02%, LR: 0.000100
Epoch [3/25], Loss: 1.0150, Train Acc: 50.86%, Val Acc: 52.94%, LR: 0.000100
Epoch [4/25], Loss: 1.0005, Train Acc: 55.82%, Val Acc: 50.98%, LR: 0.000100
Epoch [5/25], Loss: 0.9822, Train Acc: 56.68%, Val Acc: 47.06%, LR: 0.000100
Epoch [6/25], Loss: 0.9772, Train Acc: 56.47%, Val Acc: 52.94%, LR: 0.000100
Epoch [7/25], Loss: 0.9536, Train Acc: 57.33%, Val Acc: 60.78%, LR: 0.000100
Epoch [8/25], Loss: 0.9344, Train Acc: 57.97%, Val Acc: 56.86%, LR: 0.000100
Epoch [9/25], Loss: 0.9275, Train Acc: 60.13%, Val Acc: 60.78%, LR: 0.000100
Epoch [10/25], Loss: 0.8961, Train Acc: 60.78%, Val Acc: 60.78%, LR: 0.000100
Epoch [11/25], Loss: 0.8971, Train Acc: 62.72%, Val Acc: 64.71%, LR: 0.000100
Epoch [12/25], Loss: 0.8812, Train Acc: 62.28%, Val Acc: 64.71%, LR: 0.000100
Epoch [13/25], Loss: 0.8660, Train Acc: 63.79%, Val Acc: 60.78%, LR: 0.000100
Epoch [14/25], Loss: 0.8758, Train Acc: 67.46%, Val Acc: 66.67%, LR: 0.000100
Epoch [15/25], Loss: 0.8813, Train Acc: 63.36%, Val Acc: 64.71%, LR: 0.000100
Epoch [16/25], Loss: 0.8532, Train Acc: 65.09%, Val Acc: 60.78%, LR: 0.000100
Epoch [17/25], Loss: 0.8254, Train Acc: 67.24%, Val Acc: 60.78%, LR: 0.000100
Epoch [18/25], Loss: 0.8297, Train Acc: 67.24%, Val Acc: 49.02%, LR: 0.000050
Epoch [19/25], Loss: 0.8382, Train Acc: 67.03%, Val Acc: 66.67%, LR: 0.000050
Epoch [20/25], Loss: 0.7728, Train Acc: 71.98%, Val Acc: 66.67%, LR: 0.000050
Epoch [21/25], Loss: 0.7844, Train Acc: 71.77%, Val Acc: 68.63%, LR: 0.000050
Epoch [22/25], Loss: 0.7747, Train Acc: 71.34%, Val Acc: 66.67%, LR: 0.000050
Epoch [23/25], Loss: 0.7463, Train Acc: 75.65%, Val Acc: 64.71%, LR: 0.000050
Epoch [24/25], Loss: 0.7365, Train Acc: 73.92%, Val Acc: 66.67%, LR: 0.000050
Epoch [25/25], Loss: 0.7622, Train Acc: 71.98%, Val Acc: 68.63%, LR: 0.000025
Fine-tuning all layers…
Epoch [1/10], Loss: 0.7570, Train Acc: 73.49%, Val Acc: 62.75%, LR: 0.000010
Epoch [2/10], Loss: 0.7495, Train Acc: 73.49%, Val Acc: 64.71%, LR: 0.000010
Epoch [3/10], Loss: 0.7223, Train Acc: 75.00%, Val Acc: 66.67%, LR: 0.000010
Epoch [4/10], Loss: 0.6997, Train Acc: 78.23%, Val Acc: 68.63%, LR: 0.000010
Epoch [5/10], Loss: 0.6812, Train Acc: 78.66%, Val Acc: 68.63%, LR: 0.000010
Epoch [6/10], Loss: 0.6611, Train Acc: 79.53%, Val Acc: 64.71%, LR: 0.000010
Epoch [7/10], Loss: 0.6459, Train Acc: 80.39%, Val Acc: 72.55%, LR: 0.000010
Epoch [8/10], Loss: 0.6435, Train Acc: 81.68%, Val Acc: 66.67%, LR: 0.000010
Epoch [9/10], Loss: 0.6452, Train Acc: 78.88%, Val Acc: 66.67%, LR: 0.000010
Epoch [10/10], Loss: 0.6031, Train Acc: 83.84%, Val Acc: 66.67%, LR: 0.000005
Testing model…

=== split_10_90 Results ===
Accuracy: 0.6368
Precision: 0.6345
Recall: 0.6368
F1-Score: 0.6353

Classification Report:
              precision    recall  f1-score   support

          AD       0.56      0.50      0.53      1012
          CN       0.58      0.60      0.59      1296
         MCI       0.70      0.72      0.71      2331

    accuracy                           0.64      4639
   macro avg       0.61      0.61      0.61      4639
weighted avg       0.63      0.64      0.64      4639


============================================================
Processing: split_20_80
============================================================
Train samples: 928
Val samples: 103
Test samples: 4123
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0770, Train Acc: 46.88%, Val Acc: 50.49%, LR: 0.000100
Epoch [2/25], Loss: 1.0466, Train Acc: 48.60%, Val Acc: 53.40%, LR: 0.000100
Epoch [3/25], Loss: 1.0394, Train Acc: 51.72%, Val Acc: 56.31%, LR: 0.000100
Epoch [4/25], Loss: 1.0163, Train Acc: 51.51%, Val Acc: 55.34%, LR: 0.000100
Epoch [5/25], Loss: 0.9938, Train Acc: 55.06%, Val Acc: 52.43%, LR: 0.000100
Epoch [6/25], Loss: 0.9932, Train Acc: 55.71%, Val Acc: 57.28%, LR: 0.000100
Epoch [7/25], Loss: 0.9716, Train Acc: 54.31%, Val Acc: 58.25%, LR: 0.000100
Epoch [8/25], Loss: 0.9499, Train Acc: 57.65%, Val Acc: 55.34%, LR: 0.000100
Epoch [9/25], Loss: 0.9364, Train Acc: 58.19%, Val Acc: 53.40%, LR: 0.000100
Epoch [10/25], Loss: 0.9243, Train Acc: 59.27%, Val Acc: 53.40%, LR: 0.000100
Epoch [11/25], Loss: 0.9085, Train Acc: 59.70%, Val Acc: 58.25%, LR: 0.000050
Epoch [12/25], Loss: 0.9009, Train Acc: 61.31%, Val Acc: 58.25%, LR: 0.000050
Epoch [13/25], Loss: 0.8596, Train Acc: 63.90%, Val Acc: 58.25%, LR: 0.000050
Epoch [14/25], Loss: 0.8654, Train Acc: 64.55%, Val Acc: 60.19%, LR: 0.000050
Epoch [15/25], Loss: 0.8488, Train Acc: 65.30%, Val Acc: 61.17%, LR: 0.000050
Epoch [16/25], Loss: 0.8434, Train Acc: 65.73%, Val Acc: 57.28%, LR: 0.000050
Epoch [17/25], Loss: 0.8389, Train Acc: 68.10%, Val Acc: 58.25%, LR: 0.000050
Epoch [18/25], Loss: 0.8310, Train Acc: 66.49%, Val Acc: 60.19%, LR: 0.000050
Epoch [19/25], Loss: 0.8286, Train Acc: 67.24%, Val Acc: 60.19%, LR: 0.000025
Epoch [20/25], Loss: 0.7880, Train Acc: 70.91%, Val Acc: 60.19%, LR: 0.000025
Epoch [21/25], Loss: 0.8068, Train Acc: 67.89%, Val Acc: 58.25%, LR: 0.000025
Epoch [22/25], Loss: 0.8067, Train Acc: 69.07%, Val Acc: 58.25%, LR: 0.000025
Early stopping at epoch 22
Fine-tuning all layers…
Epoch [1/10], Loss: 0.8197, Train Acc: 67.46%, Val Acc: 64.08%, LR: 0.000010
Epoch [2/10], Loss: 0.7970, Train Acc: 68.53%, Val Acc: 65.05%, LR: 0.000010
Epoch [3/10], Loss: 0.7780, Train Acc: 70.80%, Val Acc: 70.87%, LR: 0.000010
Epoch [4/10], Loss: 0.7244, Train Acc: 76.19%, Val Acc: 68.93%, LR: 0.000010
Epoch [5/10], Loss: 0.7381, Train Acc: 73.38%, Val Acc: 72.82%, LR: 0.000010
Epoch [6/10], Loss: 0.7076, Train Acc: 76.40%, Val Acc: 71.84%, LR: 0.000010
Epoch [7/10], Loss: 0.7036, Train Acc: 76.94%, Val Acc: 71.84%, LR: 0.000010
Epoch [8/10], Loss: 0.6710, Train Acc: 79.74%, Val Acc: 69.90%, LR: 0.000005
Epoch [9/10], Loss: 0.6562, Train Acc: 79.53%, Val Acc: 73.79%, LR: 0.000005
Epoch [10/10], Loss: 0.6352, Train Acc: 82.65%, Val Acc: 75.73%, LR: 0.000005
Testing model…

=== split_20_80 Results ===
Accuracy: 0.7735
Precision: 0.7702
Recall: 0.7735
F1-Score: 0.7678

Classification Report:
              precision    recall  f1-score   support

          AD       0.73      0.56      0.63       899
          CN       0.77      0.74      0.75      1152
         MCI       0.79      0.89      0.83      2072

    accuracy                           0.77      4123
   macro avg       0.76      0.73      0.74      4123
weighted avg       0.77      0.77      0.77      4123


============================================================
Processing: split_30_70
============================================================
Train samples: 1391
Val samples: 155
Test samples: 3608
```

```
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0607, Train Acc: 47.81%, Val Acc: 50.32%, LR: 0.000100
Epoch [2/25], Loss: 1.0246, Train Acc: 51.69%, Val Acc: 52.26%, LR: 0.000100
Epoch [3/25], Loss: 1.0146, Train Acc: 51.55%, Val Acc: 54.19%, LR: 0.000100
Epoch [4/25], Loss: 0.9884, Train Acc: 54.42%, Val Acc: 59.35%, LR: 0.000100
Epoch [5/25], Loss: 0.9673, Train Acc: 56.51%, Val Acc: 54.84%, LR: 0.000100
Epoch [6/25], Loss: 0.9459, Train Acc: 57.58%, Val Acc: 56.13%, LR: 0.000100
Epoch [7/25], Loss: 0.9335, Train Acc: 59.74%, Val Acc: 66.45%, LR: 0.000100
Epoch [8/25], Loss: 0.9010, Train Acc: 62.62%, Val Acc: 67.10%, LR: 0.000100
Epoch [9/25], Loss: 0.8810, Train Acc: 64.13%, Val Acc: 66.45%, LR: 0.000100
Epoch [10/25], Loss: 0.8694, Train Acc: 64.41%, Val Acc: 63.23%, LR: 0.000100
Epoch [11/25], Loss: 0.8496, Train Acc: 64.92%, Val Acc: 64.52%, LR: 0.000100
Epoch [12/25], Loss: 0.8387, Train Acc: 67.51%, Val Acc: 69.68%, LR: 0.000100
Epoch [13/25], Loss: 0.8230, Train Acc: 67.22%, Val Acc: 69.03%, LR: 0.000100
Epoch [14/25], Loss: 0.8019, Train Acc: 69.16%, Val Acc: 76.77%, LR: 0.000100
Epoch [15/25], Loss: 0.8014, Train Acc: 68.66%, Val Acc: 74.19%, LR: 0.000100
Epoch [16/25], Loss: 0.7953, Train Acc: 67.72%, Val Acc: 73.55%, LR: 0.000100
Epoch [17/25], Loss: 0.7672, Train Acc: 71.39%, Val Acc: 76.13%, LR: 0.000100
Epoch [18/25], Loss: 0.7480, Train Acc: 73.18%, Val Acc: 72.26%, LR: 0.000050
Epoch [19/25], Loss: 0.7298, Train Acc: 74.48%, Val Acc: 73.55%, LR: 0.000050
Epoch [20/25], Loss: 0.7019, Train Acc: 75.92%, Val Acc: 76.77%, LR: 0.000050
Epoch [21/25], Loss: 0.7222, Train Acc: 74.69%, Val Acc: 77.42%, LR: 0.000050
Epoch [22/25], Loss: 0.7059, Train Acc: 74.91%, Val Acc: 77.42%, LR: 0.000050
Epoch [23/25], Loss: 0.6917, Train Acc: 77.64%, Val Acc: 80.65%, LR: 0.000050
Epoch [24/25], Loss: 0.7001, Train Acc: 76.78%, Val Acc: 79.35%, LR: 0.000050
Epoch [25/25], Loss: 0.6771, Train Acc: 78.00%, Val Acc: 77.42%, LR: 0.000050
Fine-tuning all layers…
Epoch [1/10], Loss: 0.7022, Train Acc: 74.84%, Val Acc: 81.29%, LR: 0.000010
Epoch [2/10], Loss: 0.6616, Train Acc: 78.79%, Val Acc: 76.13%, LR: 0.000010
Epoch [3/10], Loss: 0.6489, Train Acc: 79.51%, Val Acc: 78.71%, LR: 0.000010
Epoch [4/10], Loss: 0.6318, Train Acc: 79.87%, Val Acc: 80.65%, LR: 0.000005
Epoch [5/10], Loss: 0.5935, Train Acc: 83.54%, Val Acc: 83.87%, LR: 0.000005
Epoch [6/10], Loss: 0.5697, Train Acc: 85.26%, Val Acc: 81.29%, LR: 0.000005
Epoch [7/10], Loss: 0.5803, Train Acc: 83.61%, Val Acc: 83.87%, LR: 0.000005
Epoch [8/10], Loss: 0.5600, Train Acc: 85.84%, Val Acc: 83.87%, LR: 0.000003
Epoch [9/10], Loss: 0.5484, Train Acc: 87.63%, Val Acc: 82.58%, LR: 0.000003
Epoch [10/10], Loss: 0.5473, Train Acc: 86.77%, Val Acc: 80.65%, LR: 0.000003
Testing model…

=== split_30_70 Results ===
Accuracy: 0.8196
Precision: 0.8254
Recall: 0.8196
```

```
F1-Score: 0.8156

Classification Report:
              precision    recall  f1-score   support

          AD       0.83      0.66      0.73       787
          CN       0.89      0.74      0.80      1008
         MCI       0.79      0.94      0.86      1813

    accuracy                           0.82      3608
   macro avg       0.83      0.78      0.80      3608
weighted avg       0.83      0.82      0.82      3608


============================================================
Processing: split_40_60
============================================================
Train samples: 1855
Val samples: 207
Test samples: 3092
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0505, Train Acc: 50.35%, Val Acc: 51.21%, LR: 0.000100
Epoch [2/25], Loss: 1.0175, Train Acc: 52.45%, Val Acc: 55.07%, LR: 0.000100
Epoch [3/25], Loss: 0.9818, Train Acc: 55.09%, Val Acc: 56.04%, LR: 0.000100
Epoch [4/25], Loss: 0.9570, Train Acc: 56.55%, Val Acc: 57.97%, LR: 0.000100
Epoch [5/25], Loss: 0.9302, Train Acc: 60.32%, Val Acc: 63.29%, LR: 0.000100
Epoch [6/25], Loss: 0.8922, Train Acc: 62.96%, Val Acc: 59.90%, LR: 0.000100
Epoch [7/25], Loss: 0.8802, Train Acc: 64.37%, Val Acc: 62.80%, LR: 0.000100
Epoch [8/25], Loss: 0.8641, Train Acc: 63.50%, Val Acc: 64.25%, LR: 0.000100
Epoch [9/25], Loss: 0.8422, Train Acc: 67.01%, Val Acc: 64.73%, LR: 0.000100
Epoch [10/25], Loss: 0.8115, Train Acc: 68.03%, Val Acc: 66.18%, LR: 0.000100
Epoch [11/25], Loss: 0.8005, Train Acc: 68.84%, Val Acc: 74.40%, LR: 0.000100
Epoch [12/25], Loss: 0.7898, Train Acc: 70.51%, Val Acc: 72.46%, LR: 0.000100
Epoch [13/25], Loss: 0.7693, Train Acc: 70.57%, Val Acc: 72.95%, LR: 0.000100
Epoch [14/25], Loss: 0.7662, Train Acc: 72.18%, Val Acc: 75.36%, LR: 0.000100
Epoch [15/25], Loss: 0.7373, Train Acc: 73.37%, Val Acc: 76.81%, LR: 0.000100
Epoch [16/25], Loss: 0.7318, Train Acc: 73.53%, Val Acc: 73.43%, LR: 0.000100
Epoch [17/25], Loss: 0.7064, Train Acc: 75.74%, Val Acc: 73.91%, LR: 0.000100
Epoch [18/25], Loss: 0.7050, Train Acc: 75.63%, Val Acc: 75.36%, LR: 0.000100
Epoch [19/25], Loss: 0.7208, Train Acc: 74.61%, Val Acc: 73.43%, LR: 0.000050
Epoch [20/25], Loss: 0.6739, Train Acc: 78.44%, Val Acc: 78.26%, LR: 0.000050
Epoch [21/25], Loss: 0.6735, Train Acc: 78.11%, Val Acc: 78.74%, LR: 0.000050
Epoch [22/25], Loss: 0.6679, Train Acc: 78.27%, Val Acc: 77.29%, LR: 0.000050
Epoch [23/25], Loss: 0.6469, Train Acc: 79.62%, Val Acc: 79.71%, LR: 0.000050
Epoch [24/25], Loss: 0.6552, Train Acc: 79.08%, Val Acc: 78.26%, LR: 0.000050
Epoch [25/25], Loss: 0.6372, Train Acc: 80.75%, Val Acc: 80.19%, LR: 0.000050
Fine-tuning all layers…
Epoch [1/10], Loss: 0.6354, Train Acc: 80.65%, Val Acc: 79.71%, LR: 0.000010
Epoch [2/10], Loss: 0.6161, Train Acc: 82.59%, Val Acc: 80.68%, LR: 0.000010
Epoch [3/10], Loss: 0.5946, Train Acc: 83.18%, Val Acc: 80.68%, LR: 0.000010
Epoch [4/10], Loss: 0.5739, Train Acc: 85.66%, Val Acc: 84.06%, LR: 0.000010
Epoch [5/10], Loss: 0.5558, Train Acc: 86.47%, Val Acc: 81.64%, LR: 0.000010
Epoch [6/10], Loss: 0.5600, Train Acc: 86.15%, Val Acc: 85.99%, LR: 0.000010
Epoch [7/10], Loss: 0.5234, Train Acc: 88.52%, Val Acc: 84.06%, LR: 0.000010
Epoch [8/10], Loss: 0.5259, Train Acc: 87.92%, Val Acc: 84.54%, LR: 0.000010
Epoch [9/10], Loss: 0.5114, Train Acc: 88.89%, Val Acc: 87.44%, LR: 0.000010
Epoch [10/10], Loss: 0.4995, Train Acc: 90.08%, Val Acc: 86.47%, LR: 0.000010
Testing model…

=== split_40_60 Results ===
Accuracy: 0.8765
Precision: 0.8775
Recall: 0.8765
F1-Score: 0.8750

Classification Report:
              precision    recall  f1-score   support

          AD       0.90      0.77      0.83       674
          CN       0.88      0.84      0.86       864
         MCI       0.87      0.94      0.90      1554

    accuracy                           0.88      3092
   macro avg       0.88      0.85      0.86      3092
weighted avg       0.88      0.88      0.87      3092


============================================================
Processing: split_50_50
============================================================
Train samples: 2319
Val samples: 258
Test samples: 2577
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0428, Train Acc: 49.42%, Val Acc: 53.88%, LR: 0.000100
Epoch [2/25], Loss: 1.0036, Train Acc: 53.34%, Val Acc: 60.08%, LR: 0.000100
Epoch [3/25], Loss: 0.9718, Train Acc: 55.63%, Val Acc: 55.04%, LR: 0.000100
Epoch [4/25], Loss: 0.9515, Train Acc: 56.75%, Val Acc: 63.95%, LR: 0.000100
Epoch [5/25], Loss: 0.9153, Train Acc: 59.21%, Val Acc: 66.67%, LR: 0.000100
Epoch [6/25], Loss: 0.8877, Train Acc: 62.35%, Val Acc: 62.02%, LR: 0.000100
Epoch [7/25], Loss: 0.8653, Train Acc: 64.47%, Val Acc: 68.22%, LR: 0.000100
Epoch [8/25], Loss: 0.8541, Train Acc: 66.02%, Val Acc: 74.03%, LR: 0.000100
Epoch [9/25], Loss: 0.8346, Train Acc: 67.01%, Val Acc: 74.42%, LR: 0.000100
Epoch [10/25], Loss: 0.8078, Train Acc: 67.79%, Val Acc: 74.81%, LR: 0.000100
Epoch [11/25], Loss: 0.7812, Train Acc: 70.85%, Val Acc: 76.36%, LR: 0.000100
Epoch [12/25], Loss: 0.7759, Train Acc: 70.94%, Val Acc: 74.03%, LR: 0.000100
Epoch [13/25], Loss: 0.7548, Train Acc: 72.62%, Val Acc: 75.97%, LR: 0.000100
Epoch [14/25], Loss: 0.7495, Train Acc: 72.53%, Val Acc: 79.07%, LR: 0.000100
Epoch [15/25], Loss: 0.7411, Train Acc: 74.00%, Val Acc: 76.74%, LR: 0.000100
Epoch [16/25], Loss: 0.7266, Train Acc: 73.74%, Val Acc: 83.33%, LR: 0.000100
Epoch [17/25], Loss: 0.7325, Train Acc: 73.82%, Val Acc: 82.95%, LR: 0.000100
Epoch [18/25], Loss: 0.7079, Train Acc: 76.50%, Val Acc: 79.84%, LR: 0.000100
Epoch [19/25], Loss: 0.6982, Train Acc: 77.62%, Val Acc: 84.11%, LR: 0.000100
Epoch [20/25], Loss: 0.6697, Train Acc: 79.04%, Val Acc: 81.78%, LR: 0.000100
Epoch [21/25], Loss: 0.6700, Train Acc: 78.22%, Val Acc: 82.56%, LR: 0.000100
Epoch [22/25], Loss: 0.6530, Train Acc: 79.60%, Val Acc: 84.11%, LR: 0.000100
Epoch [23/25], Loss: 0.6657, Train Acc: 78.53%, Val Acc: 84.88%, LR: 0.000100
Epoch [24/25], Loss: 0.6519, Train Acc: 80.38%, Val Acc: 84.88%, LR: 0.000100
Epoch [25/25], Loss: 0.6325, Train Acc: 80.77%, Val Acc: 84.88%, LR: 0.000100
Fine-tuning all layers…
Epoch [1/10], Loss: 0.6238, Train Acc: 81.85%, Val Acc: 87.21%, LR: 0.000010
Epoch [2/10], Loss: 0.5907, Train Acc: 83.31%, Val Acc: 88.76%, LR: 0.000010
Epoch [3/10], Loss: 0.5659, Train Acc: 85.99%, Val Acc: 87.21%, LR: 0.000010
Epoch [4/10], Loss: 0.5519, Train Acc: 86.42%, Val Acc: 88.37%, LR: 0.000010
Epoch [5/10], Loss: 0.5234, Train Acc: 88.53%, Val Acc: 88.76%, LR: 0.000005
Epoch [6/10], Loss: 0.4994, Train Acc: 89.56%, Val Acc: 90.31%, LR: 0.000005
Epoch [7/10], Loss: 0.5071, Train Acc: 89.95%, Val Acc: 91.47%, LR: 0.000005
Epoch [8/10], Loss: 0.4901, Train Acc: 89.87%, Val Acc: 91.47%, LR: 0.000005
Epoch [9/10], Loss: 0.4800, Train Acc: 91.12%, Val Acc: 91.09%, LR: 0.000005
Epoch [10/10], Loss: 0.4814, Train Acc: 91.03%, Val Acc: 91.09%, LR: 0.000003
Testing model…

=== split_50_50 Results ===
Accuracy: 0.9065
Precision: 0.9065
Recall: 0.9065
F1-Score: 0.9060

Classification Report:
              precision    recall  f1-score   support

          AD       0.91      0.83      0.87       562
          CN       0.89      0.90      0.90       720
         MCI       0.91      0.94      0.93      1295
```

```
    accuracy                           0.91      2577
   macro avg       0.91      0.89      0.90      2577
weighted avg       0.91      0.91      0.91      2577


============================================================
Processing: split_60_40
============================================================
Train samples: 2784
Val samples: 308
Test samples: 2062
Starting training…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/25], Loss: 1.0498, Train Acc: 48.64%, Val Acc: 50.32%, LR: 0.000100
Epoch [2/25], Loss: 0.9921, Train Acc: 54.13%, Val Acc: 54.55%, LR: 0.000100
Epoch [3/25], Loss: 0.9572, Train Acc: 57.00%, Val Acc: 62.99%, LR: 0.000100
Epoch [4/25], Loss: 0.9264, Train Acc: 60.20%, Val Acc: 61.36%, LR: 0.000100
Epoch [5/25], Loss: 0.9095, Train Acc: 60.92%, Val Acc: 66.88%, LR: 0.000100
Epoch [6/25], Loss: 0.8699, Train Acc: 63.47%, Val Acc: 69.48%, LR: 0.000100
Epoch [7/25], Loss: 0.8511, Train Acc: 66.06%, Val Acc: 73.05%, LR: 0.000100
Epoch [8/25], Loss: 0.8261, Train Acc: 67.42%, Val Acc: 73.05%, LR: 0.000100
Epoch [9/25], Loss: 0.8098, Train Acc: 68.71%, Val Acc: 74.35%, LR: 0.000100
Epoch [10/25], Loss: 0.7839, Train Acc: 69.11%, Val Acc: 76.95%, LR: 0.000100
Epoch [11/25], Loss: 0.7573, Train Acc: 71.66%, Val Acc: 77.27%, LR: 0.000100
Epoch [12/25], Loss: 0.7377, Train Acc: 72.67%, Val Acc: 78.57%, LR: 0.000100
Epoch [13/25], Loss: 0.7467, Train Acc: 72.77%, Val Acc: 77.60%, LR: 0.000100
Epoch [14/25], Loss: 0.7132, Train Acc: 75.32%, Val Acc: 80.52%, LR: 0.000100
Epoch [15/25], Loss: 0.7070, Train Acc: 74.96%, Val Acc: 81.82%, LR: 0.000100
Epoch [16/25], Loss: 0.6920, Train Acc: 76.83%, Val Acc: 83.44%, LR: 0.000100
Epoch [17/25], Loss: 0.6819, Train Acc: 77.05%, Val Acc: 82.14%, LR: 0.000100
Epoch [18/25], Loss: 0.6757, Train Acc: 77.77%, Val Acc: 82.14%, LR: 0.000100
Epoch [19/25], Loss: 0.6577, Train Acc: 78.88%, Val Acc: 84.09%, LR: 0.000100
Epoch [20/25], Loss: 0.6474, Train Acc: 79.42%, Val Acc: 85.06%, LR: 0.000100
Epoch [21/25], Loss: 0.6443, Train Acc: 80.14%, Val Acc: 85.71%, LR: 0.000100
Epoch [22/25], Loss: 0.6295, Train Acc: 81.29%, Val Acc: 85.06%, LR: 0.000100
Epoch [23/25], Loss: 0.6290, Train Acc: 80.60%, Val Acc: 85.39%, LR: 0.000100
Epoch [24/25], Loss: 0.6095, Train Acc: 81.72%, Val Acc: 85.71%, LR: 0.000100
Epoch [25/25], Loss: 0.6192, Train Acc: 81.54%, Val Acc: 85.39%, LR: 0.000050
Fine-tuning all layers…
Epoch [1/10], Loss: 0.5952, Train Acc: 82.79%, Val Acc: 86.36%, LR: 0.000010
Epoch [2/10], Loss: 0.5624, Train Acc: 85.74%, Val Acc: 88.64%, LR: 0.000010
Epoch [3/10], Loss: 0.5432, Train Acc: 85.96%, Val Acc: 89.29%, LR: 0.000010
Epoch [4/10], Loss: 0.5354, Train Acc: 87.00%, Val Acc: 89.29%, LR: 0.000010
Epoch [5/10], Loss: 0.5081, Train Acc: 89.55%, Val Acc: 93.18%, LR: 0.000010
Epoch [6/10], Loss: 0.4906, Train Acc: 90.59%, Val Acc: 88.64%, LR: 0.000010
Epoch [7/10], Loss: 0.4918, Train Acc: 90.19%, Val Acc: 93.18%, LR: 0.000010
Epoch [8/10], Loss: 0.4859, Train Acc: 90.84%, Val Acc: 92.53%, LR: 0.000005
Epoch [9/10], Loss: 0.4595, Train Acc: 92.39%, Val Acc: 92.86%, LR: 0.000005
Epoch [10/10], Loss: 0.4545, Train Acc: 92.42%, Val Acc: 93.51%, LR: 0.000005
Testing model…

=== split_60_40 Results ===
Accuracy: 0.9365
Precision: 0.9365
Recall: 0.9365
F1-Score: 0.9362

Classification Report:
              precision    recall  f1-score   support

          AD       0.93      0.89      0.91       450
          CN       0.94      0.93      0.93       576
         MCI       0.94      0.96      0.95      1036

    accuracy                           0.94      2062
   macro avg       0.94      0.93      0.93      2062
weighted avg       0.94      0.94      0.94      2062


============================================================
Processing: split_70_30
============================================================
Train samples: 3247
Val samples: 361
Test samples: 1546
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0570, Train Acc: 48.35%, Val Acc: 56.79%, LR: 0.000100
Epoch [2/25], Loss: 1.0049, Train Acc: 52.63%, Val Acc: 60.39%, LR: 0.000100
Epoch [3/25], Loss: 0.9595, Train Acc: 57.35%, Val Acc: 64.82%, LR: 0.000100
Epoch [4/25], Loss: 0.9230, Train Acc: 61.01%, Val Acc: 60.66%, LR: 0.000100
Epoch [5/25], Loss: 0.8942, Train Acc: 61.47%, Val Acc: 69.81%, LR: 0.000100
Epoch [6/25], Loss: 0.8650, Train Acc: 64.95%, Val Acc: 68.98%, LR: 0.000100
Epoch [7/25], Loss: 0.8447, Train Acc: 66.55%, Val Acc: 70.91%, LR: 0.000100
Epoch [8/25], Loss: 0.8156, Train Acc: 68.31%, Val Acc: 74.79%, LR: 0.000100
Epoch [9/25], Loss: 0.7937, Train Acc: 69.94%, Val Acc: 76.18%, LR: 0.000100
Epoch [10/25], Loss: 0.7766, Train Acc: 71.02%, Val Acc: 75.90%, LR: 0.000100
Epoch [11/25], Loss: 0.7537, Train Acc: 72.07%, Val Acc: 77.01%, LR: 0.000100
Epoch [12/25], Loss: 0.7343, Train Acc: 73.54%, Val Acc: 76.45%, LR: 0.000100
Epoch [13/25], Loss: 0.7277, Train Acc: 74.04%, Val Acc: 77.01%, LR: 0.000100
Epoch [14/25], Loss: 0.7078, Train Acc: 76.41%, Val Acc: 82.55%, LR: 0.000100
Epoch [15/25], Loss: 0.6866, Train Acc: 76.87%, Val Acc: 82.55%, LR: 0.000100
Epoch [16/25], Loss: 0.6785, Train Acc: 77.92%, Val Acc: 82.55%, LR: 0.000100
Epoch [17/25], Loss: 0.6727, Train Acc: 78.35%, Val Acc: 83.10%, LR: 0.000100
Epoch [18/25], Loss: 0.6500, Train Acc: 79.33%, Val Acc: 83.10%, LR: 0.000100
Epoch [19/25], Loss: 0.6372, Train Acc: 80.14%, Val Acc: 83.66%, LR: 0.000100
Epoch [20/25], Loss: 0.6436, Train Acc: 79.64%, Val Acc: 84.76%, LR: 0.000100
Epoch [21/25], Loss: 0.6291, Train Acc: 80.54%, Val Acc: 85.32%, LR: 0.000100
Epoch [22/25], Loss: 0.6251, Train Acc: 81.71%, Val Acc: 86.15%, LR: 0.000100
Epoch [23/25], Loss: 0.6071, Train Acc: 82.57%, Val Acc: 87.81%, LR: 0.000100
Epoch [24/25], Loss: 0.6087, Train Acc: 82.17%, Val Acc: 85.87%, LR: 0.000100
Epoch [25/25], Loss: 0.5872, Train Acc: 83.71%, Val Acc: 89.75%, LR: 0.000100
Fine-tuning all layers…
Epoch [1/10], Loss: 0.5874, Train Acc: 83.43%, Val Acc: 88.92%, LR: 0.000010
Epoch [2/10], Loss: 0.5522, Train Acc: 86.20%, Val Acc: 88.09%, LR: 0.000010
Epoch [3/10], Loss: 0.5325, Train Acc: 87.71%, Val Acc: 90.58%, LR: 0.000010
Epoch [4/10], Loss: 0.5191, Train Acc: 88.64%, Val Acc: 91.97%, LR: 0.000010
Epoch [5/10], Loss: 0.5009, Train Acc: 88.97%, Val Acc: 92.80%, LR: 0.000010
Epoch [6/10], Loss: 0.4853, Train Acc: 89.74%, Val Acc: 92.80%, LR: 0.000010
Epoch [7/10], Loss: 0.4729, Train Acc: 91.65%, Val Acc: 91.41%, LR: 0.000010
Epoch [8/10], Loss: 0.4675, Train Acc: 91.84%, Val Acc: 93.91%, LR: 0.000010
Epoch [9/10], Loss: 0.4593, Train Acc: 91.87%, Val Acc: 94.74%, LR: 0.000010
Epoch [10/10], Loss: 0.4460, Train Acc: 93.04%, Val Acc: 93.91%, LR: 0.000010
Testing model…

=== split_70_30 Results ===
Accuracy: 0.9483
Precision: 0.9484
Recall: 0.9483
F1-Score: 0.9483

Classification Report:
              precision    recall  f1-score   support

          AD       0.94      0.92      0.93       337
          CN       0.93      0.94      0.93       432
         MCI       0.97      0.96      0.96       777

    accuracy                           0.95      1546
   macro avg       0.94      0.94      0.94      1546
weighted avg       0.95      0.95      0.95      1546


============================================================
Processing: split_80_20
```

```
============================================================
Train samples: 3711
Val samples: 412
Test samples: 1031
Starting training…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/25], Loss: 1.0393, Train Acc: 49.80%, Val Acc: 54.13%, LR: 0.000100
Epoch [2/25], Loss: 0.9873, Train Acc: 55.03%, Val Acc: 56.31%, LR: 0.000100
Epoch [3/25], Loss: 0.9428, Train Acc: 58.18%, Val Acc: 61.89%, LR: 0.000100
Epoch [4/25], Loss: 0.9112, Train Acc: 59.55%, Val Acc: 63.35%, LR: 0.000100
Epoch [5/25], Loss: 0.8735, Train Acc: 63.78%, Val Acc: 66.75%, LR: 0.000100
Epoch [6/25], Loss: 0.8496, Train Acc: 64.62%, Val Acc: 71.12%, LR: 0.000100
Epoch [7/25], Loss: 0.8290, Train Acc: 66.29%, Val Acc: 73.79%, LR: 0.000100
Epoch [8/25], Loss: 0.7940, Train Acc: 69.42%, Val Acc: 73.30%, LR: 0.000100
Epoch [9/25], Loss: 0.7774, Train Acc: 70.57%, Val Acc: 78.16%, LR: 0.000100
Epoch [10/25], Loss: 0.7604, Train Acc: 71.76%, Val Acc: 78.88%, LR: 0.000100
Epoch [11/25], Loss: 0.7295, Train Acc: 73.92%, Val Acc: 79.85%, LR: 0.000100
Epoch [12/25], Loss: 0.7266, Train Acc: 74.89%, Val Acc: 81.80%, LR: 0.000100
Epoch [13/25], Loss: 0.7113, Train Acc: 75.88%, Val Acc: 82.77%, LR: 0.000100
Epoch [14/25], Loss: 0.6897, Train Acc: 76.80%, Val Acc: 85.19%, LR: 0.000100
Epoch [15/25], Loss: 0.6781, Train Acc: 77.66%, Val Acc: 85.68%, LR: 0.000100
Epoch [16/25], Loss: 0.6697, Train Acc: 78.42%, Val Acc: 86.17%, LR: 0.000100
Epoch [17/25], Loss: 0.6507, Train Acc: 79.33%, Val Acc: 86.17%, LR: 0.000100
Epoch [18/25], Loss: 0.6455, Train Acc: 79.90%, Val Acc: 86.65%, LR: 0.000100
Epoch [19/25], Loss: 0.6302, Train Acc: 81.24%, Val Acc: 86.89%, LR: 0.000100
Epoch [20/25], Loss: 0.6214, Train Acc: 81.78%, Val Acc: 89.32%, LR: 0.000100
Epoch [21/25], Loss: 0.6153, Train Acc: 82.27%, Val Acc: 89.08%, LR: 0.000100
Epoch [22/25], Loss: 0.6012, Train Acc: 82.54%, Val Acc: 89.81%, LR: 0.000100
Epoch [23/25], Loss: 0.5919, Train Acc: 83.45%, Val Acc: 88.83%, LR: 0.000100
Epoch [24/25], Loss: 0.5903, Train Acc: 83.56%, Val Acc: 89.08%, LR: 0.000100
Epoch [25/25], Loss: 0.5817, Train Acc: 84.51%, Val Acc: 90.53%, LR: 0.000100
Fine-tuning all layers…
Epoch [1/10], Loss: 0.5686, Train Acc: 84.99%, Val Acc: 89.08%, LR: 0.000010
Epoch [2/10], Loss: 0.5354, Train Acc: 87.42%, Val Acc: 91.26%, LR: 0.000010
Epoch [3/10], Loss: 0.5145, Train Acc: 88.52%, Val Acc: 92.72%, LR: 0.000010
Epoch [4/10], Loss: 0.4966, Train Acc: 89.30%, Val Acc: 93.93%, LR: 0.000010
Epoch [5/10], Loss: 0.4826, Train Acc: 90.54%, Val Acc: 93.93%, LR: 0.000010
Epoch [6/10], Loss: 0.4702, Train Acc: 91.27%, Val Acc: 94.66%, LR: 0.000010
Epoch [7/10], Loss: 0.4603, Train Acc: 91.48%, Val Acc: 95.87%, LR: 0.000010
Epoch [8/10], Loss: 0.4528, Train Acc: 92.51%, Val Acc: 96.36%, LR: 0.000010
Epoch [9/10], Loss: 0.4438, Train Acc: 92.97%, Val Acc: 96.12%, LR: 0.000010
Epoch [10/10], Loss: 0.4247, Train Acc: 94.04%, Val Acc: 97.33%, LR: 0.000010
Testing model…
```

```
=== split_80_20 Results ===
Accuracy: 0.9525
Precision: 0.9525
Recall: 0.9525
F1-Score: 0.9524

Classification Report:
              precision    recall  f1-score   support

          AD       0.93      0.92      0.93       225
          CN       0.96      0.93      0.95       288
         MCI       0.96      0.97      0.97       518

    accuracy                           0.95      1031
   macro avg       0.95      0.94      0.95      1031
weighted avg       0.95      0.95      0.95      1031


============================================================
Processing: split_90_10
============================================================
Train samples: 4175
Val samples: 464
Test samples: 515
Starting training…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/25], Loss: 1.0334, Train Acc: 50.97%, Val Acc: 53.23%, LR: 0.000100
Epoch [2/25], Loss: 0.9822, Train Acc: 54.42%, Val Acc: 58.41%, LR: 0.000100
Epoch [3/25], Loss: 0.9334, Train Acc: 58.68%, Val Acc: 63.58%, LR: 0.000100
Epoch [4/25], Loss: 0.8858, Train Acc: 62.42%, Val Acc: 68.10%, LR: 0.000100
Epoch [5/25], Loss: 0.8598, Train Acc: 65.15%, Val Acc: 71.12%, LR: 0.000100
Epoch [6/25], Loss: 0.8349, Train Acc: 66.30%, Val Acc: 73.28%, LR: 0.000100
Epoch [7/25], Loss: 0.8039, Train Acc: 69.27%, Val Acc: 76.08%, LR: 0.000100
Epoch [8/25], Loss: 0.7747, Train Acc: 71.43%, Val Acc: 80.82%, LR: 0.000100
Epoch [9/25], Loss: 0.7514, Train Acc: 72.74%, Val Acc: 79.53%, LR: 0.000100
Epoch [10/25], Loss: 0.7246, Train Acc: 74.92%, Val Acc: 80.60%, LR: 0.000100
Epoch [11/25], Loss: 0.7185, Train Acc: 75.09%, Val Acc: 83.41%, LR: 0.000100
Epoch [12/25], Loss: 0.7001, Train Acc: 75.93%, Val Acc: 83.19%, LR: 0.000100
Epoch [13/25], Loss: 0.6847, Train Acc: 77.44%, Val Acc: 86.42%, LR: 0.000100
Epoch [14/25], Loss: 0.6748, Train Acc: 77.96%, Val Acc: 84.91%, LR: 0.000100
Epoch [15/25], Loss: 0.6594, Train Acc: 78.68%, Val Acc: 85.56%, LR: 0.000100
Epoch [16/25], Loss: 0.6405, Train Acc: 80.12%, Val Acc: 86.21%, LR: 0.000100
Epoch [17/25], Loss: 0.6244, Train Acc: 80.86%, Val Acc: 87.28%, LR: 0.000100
Epoch [18/25], Loss: 0.6163, Train Acc: 81.89%, Val Acc: 90.09%, LR: 0.000100
Epoch [19/25], Loss: 0.6046, Train Acc: 83.66%, Val Acc: 89.22%, LR: 0.000100
Epoch [20/25], Loss: 0.6099, Train Acc: 82.28%, Val Acc: 89.66%, LR: 0.000100
Epoch [21/25], Loss: 0.5941, Train Acc: 83.07%, Val Acc: 88.79%, LR: 0.000100
Epoch [22/25], Loss: 0.5851, Train Acc: 83.69%, Val Acc: 88.36%, LR: 0.000050
Epoch [23/25], Loss: 0.5540, Train Acc: 85.92%, Val Acc: 90.73%, LR: 0.000050
Epoch [24/25], Loss: 0.5536, Train Acc: 85.89%, Val Acc: 90.73%, LR: 0.000050
Epoch [25/25], Loss: 0.5393, Train Acc: 86.80%, Val Acc: 92.24%, LR: 0.000050
Fine-tuning all layers…
Epoch [1/10], Loss: 0.5384, Train Acc: 87.09%, Val Acc: 91.59%, LR: 0.000010
Epoch [2/10], Loss: 0.5223, Train Acc: 88.07%, Val Acc: 92.89%, LR: 0.000010
Epoch [3/10], Loss: 0.4962, Train Acc: 90.08%, Val Acc: 93.32%, LR: 0.000010
Epoch [4/10], Loss: 0.4780, Train Acc: 91.07%, Val Acc: 94.40%, LR: 0.000010
Epoch [5/10], Loss: 0.4673, Train Acc: 92.00%, Val Acc: 93.75%, LR: 0.000010
Epoch [6/10], Loss: 0.4593, Train Acc: 92.00%, Val Acc: 94.40%, LR: 0.000010
Epoch [7/10], Loss: 0.4448, Train Acc: 93.03%, Val Acc: 96.55%, LR: 0.000010
Epoch [8/10], Loss: 0.4292, Train Acc: 93.84%, Val Acc: 95.69%, LR: 0.000010
Epoch [9/10], Loss: 0.4239, Train Acc: 94.42%, Val Acc: 96.12%, LR: 0.000010
Epoch [10/10], Loss: 0.4195, Train Acc: 94.23%, Val Acc: 95.91%, LR: 0.000005
Testing model…

=== split_90_10 Results ===
Accuracy: 0.9417
Precision: 0.9423
Recall: 0.9417
F1-Score: 0.9413

Classification Report:
              precision    recall  f1-score   support

          AD       0.95      0.87      0.91       112
          CN       0.96      0.94      0.95       144
         MCI       0.93      0.97      0.95       259

    accuracy                           0.94       515
   macro avg       0.95      0.93      0.94       515
weighted avg       0.94      0.94      0.94       515


================================================================================
ConvNeXt-Small - SUMMARY OF ALL SPLITS
================================================================================
      split  accuracy  precision   recall  f1_score  training_time
split_10_90  0.636775   0.634546 0.636775  0.635251     290.025709
split_20_80  0.773466   0.770169 0.773466  0.767802     542.540642
split_30_70  0.819568   0.825429 0.819568  0.815577     844.342106
split_40_60  0.876455   0.877528 0.876455  0.874956    1121.625662
split_50_50  0.906480   0.906529 0.906480  0.905963    1395.717910
split_60_40  0.936469   0.936461 0.936469  0.936248    1671.911056
```

```
split_70_30  0.948254   0.948387 0.948254  0.948274    1949.841236
split_80_20  0.952473   0.952461 0.952473  0.952366    2227.222875
split_90_10  0.941748   0.942265 0.941748  0.941317    2503.637483

Detailed results saved to: /kaggle/working/convnext_small_results.csv
```

[7]:
```python
import os
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
import pandas as pd
import numpy as np
from PIL import Image
from sklearn.metrics import precision_score, recall_score, f1_score,␣
↪accuracy_score, classification_report
import time
from torch.optim.lr_scheduler import CosineAnnealingLR, ReduceLROnPlateau

class AlzheimerDataset(Dataset):
    def __init__(self, split_dir, split_type, transform=None):
        self.split_dir = split_dir
        self.split_type = split_type
        self.transform = transform

        csv_path = os.path.join(split_dir, f"{split_type}.csv")
        self.df = pd.read_csv(csv_path)

        self.class_to_idx = {"AD": 0, "CN": 1, "MCI": 2}
        self.idx_to_class = {v: k for k, v in self.class_to_idx.items()}

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img_path = row['path']
        label = self.class_to_idx[row['class']]

        try:
            image = Image.open(img_path).convert('RGB')

            if self.transform:
                image = self.transform(image)

            return image, label
        except Exception as e:
            print(f"Error loading image {img_path}: {e}")
            # Return a dummy image and label
            dummy_image = torch.zeros(3, 224, 224)
            return dummy_image, label

def get_data_transforms():
    train_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomRotation(10),
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,␣
↪hue=0.1),
        transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
        transforms.RandomGrayscale(p=0.1),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
↪225])
    ])

    val_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
↪225])
    ])

    return train_transform, val_transform

def train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs, device, warmup_epochs=3):
    best_val_acc = 0
    patience = 8
    patience_counter = 0

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        # Learning rate warmup
        if epoch < warmup_epochs:
            lr_scale = min(1.0, float(epoch + 1) / warmup_epochs)
            for param_group in optimizer.param_groups:
                param_group['lr'] = param_group['initial_lr'] * lr_scale

        for batch_idx, (images, labels) in enumerate(train_loader):
            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()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(train_loader)
        epoch_acc = 100 * correct / total

        val_acc = evaluate_model(model, val_loader, device)

        if scheduler and epoch >= warmup_epochs:
            if isinstance(scheduler, CosineAnnealingLR):
                scheduler.step()
            else:
                scheduler.step(val_acc)

        current_lr = optimizer.param_groups[0]['lr']
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Train␣
↪Acc: {epoch_acc:.2f}%, Val Acc: {val_acc:.2f}%, LR: {current_lr:.6f}')

        if val_acc > best_val_acc:
            best_val_acc = val_acc
            patience_counter = 0
        else:
            patience_counter += 1

        if patience_counter >= patience:
            print(f"Early stopping at epoch {epoch+1}")
            break

def evaluate_model(model, data_loader, device):
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in data_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    return accuracy

def test_model(model, test_loader, device):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)

            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    return all_preds, all_labels

def calculate_metrics(y_true, y_pred, split_name):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted',␣
 ↪zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    print(f"\n=== {split_name} Results ===")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred, target_names=['AD', 'CN',␣
 ↪'MCI'], zero_division=0))

    return {
        'split': split_name,
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1_score': f1
    }

def run_densenet121_on_splits():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    print(f"Using device: {device}")

    splits_root = "/kaggle/working/alzheimer-resized-224_splits"
    split_folders = [f for f in os.listdir(splits_root) if f.
 ↪startswith('split_')]
    split_folders.sort()

    results = []

    train_transform, val_transform = get_data_transforms()

    for split_folder in split_folders:
        print(f"\n{'='*60}")
        print(f"Processing: {split_folder}")
        print(f"{'='*60}")

        split_path = os.path.join(splits_root, split_folder)

        train_dataset = AlzheimerDataset(split_path, 'train', train_transform)
        val_dataset = AlzheimerDataset(split_path, 'val', val_transform)
        test_dataset = AlzheimerDataset(split_path, 'test', val_transform)

        train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True,␣
 ↪num_workers=2, pin_memory=True)
        val_loader = DataLoader(val_dataset, batch_size=16, shuffle=False,␣
 ↪num_workers=2, pin_memory=True)
        test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False,␣
 ↪num_workers=2, pin_memory=True)

        print(f"Train samples: {len(train_dataset)}")
        print(f"Val samples: {len(val_dataset)}")
        print(f"Test samples: {len(test_dataset)}")

        model = models.densenet121(weights=models.DenseNet121_Weights.
 ↪IMAGENET1K_V1)

        # Freeze all layers initially
        for param in model.parameters():
            param.requires_grad = False

        # Unfreeze classifier
        num_ftrs = model.classifier.in_features
        model.classifier = nn.Linear(num_ftrs, 3)
        # Unfreeze last dense block and transition layer
        for param in model.features.denseblock4.parameters():
            param.requires_grad = True
        for param in model.features.norm5.parameters():
            param.requires_grad = True

        model = model.to(device)

        # Layer-wise learning rates for DenseNet
        optimizer = torch.optim.AdamW([
            {'params': model.classifier.parameters(), 'lr': 0.0001,␣
↪'initial_lr': 0.0001},
            {'params': model.features.denseblock4.parameters(), 'lr': 0.00005,␣
↪'initial_lr': 0.00005},
            {'params': model.features.norm5.parameters(), 'lr': 0.00005,␣
↪'initial_lr': 0.00005},
            {'params': model.features.denseblock3.parameters(), 'lr': 0.00001,␣
↪'initial_lr': 0.00001}
        ], weight_decay=1e-4)

        criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
        scheduler = CosineAnnealingLR(optimizer, T_max=20, eta_min=1e-6)

        print("Starting phase 1 training (partial unfreeze)...")
        start_time = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs=20, device=device)
        phase1_time = time.time() - start_time

        print("Starting phase 2 training (full fine-tuning)...")
        # Unfreeze all parameters for final fine-tuning
        for param in model.parameters():
            param.requires_grad = True

        optimizer = torch.optim.AdamW(model.parameters(), lr=0.00001,␣
↪weight_decay=1e-5)
        scheduler = ReduceLROnPlateau(optimizer, mode='max', factor=0.5,␣
↪patience=3, verbose=True)

        start_time_phase2 = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs=15, device=device, warmup_epochs=0)
        total_time = phase1_time + (time.time() - start_time_phase2)

        print("Testing model...")
        test_preds, test_labels = test_model(model, test_loader, device)

        split_results = calculate_metrics(test_labels, test_preds, split_folder)
        split_results['training_time'] = total_time
        results.append(split_results)

        torch.cuda.empty_cache()

    results_df = pd.DataFrame(results)
    print(f"\n{'='*80}")
    print("DenseNet-121 - SUMMARY OF ALL SPLITS")
    print(f"{'='*80}")
    print(results_df.to_string(index=False))

    results_csv_path = "/kaggle/working/densenet121_results.csv"
    results_df.to_csv(results_csv_path, index=False)
    print(f"\nDetailed results saved to: {results_csv_path}")

    return results_df

if __name__ == "__main__":
    results = run_densenet121_on_splits()
```

```
Downloading: "https://download.pytorch.org/models/densenet121-a639ec97.pth" to
/root/.cache/torch/hub/checkpoints/densenet121-a639ec97.pth

Using device: cuda

============================================================
Processing: split_10_90
============================================================
Train samples: 464
Val samples: 51
Test samples: 4639

100%|      | 30.8M/30.8M [00:00<00:00, 226MB/s]

Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.1429, Train Acc: 32.11%, Val Acc: 35.29%, LR: 0.000033
Epoch [2/20], Loss: 1.0483, Train Acc: 49.35%, Val Acc: 50.98%, LR: 0.000067
Epoch [3/20], Loss: 1.0144, Train Acc: 53.23%, Val Acc: 49.02%, LR: 0.000100
Epoch [4/20], Loss: 0.9889, Train Acc: 53.66%, Val Acc: 49.02%, LR: 0.000099
Epoch [5/20], Loss: 0.9693, Train Acc: 54.31%, Val Acc: 54.90%, LR: 0.000098
Epoch [6/20], Loss: 0.9488, Train Acc: 58.84%, Val Acc: 54.90%, LR: 0.000095
Epoch [7/20], Loss: 0.9013, Train Acc: 61.42%, Val Acc: 52.94%, LR: 0.000091
Epoch [8/20], Loss: 0.8932, Train Acc: 63.79%, Val Acc: 52.94%, LR: 0.000086
Epoch [9/20], Loss: 0.8880, Train Acc: 62.07%, Val Acc: 52.94%, LR: 0.000080
Epoch [10/20], Loss: 0.8617, Train Acc: 63.79%, Val Acc: 56.86%, LR: 0.000073
Epoch [11/20], Loss: 0.8390, Train Acc: 66.16%, Val Acc: 58.82%, LR: 0.000066
Epoch [12/20], Loss: 0.8354, Train Acc: 68.32%, Val Acc: 58.82%, LR: 0.000058
Epoch [13/20], Loss: 0.8368, Train Acc: 66.38%, Val Acc: 56.86%, LR: 0.000051
Epoch [14/20], Loss: 0.8008, Train Acc: 70.91%, Val Acc: 64.71%, LR: 0.000043
Epoch [15/20], Loss: 0.8073, Train Acc: 71.34%, Val Acc: 64.71%, LR: 0.000035
Epoch [16/20], Loss: 0.7787, Train Acc: 71.55%, Val Acc: 58.82%, LR: 0.000028
Epoch [17/20], Loss: 0.7723, Train Acc: 74.35%, Val Acc: 60.78%, LR: 0.000021
Epoch [18/20], Loss: 0.7786, Train Acc: 73.28%, Val Acc: 64.71%, LR: 0.000015
Epoch [19/20], Loss: 0.7631, Train Acc: 73.49%, Val Acc: 62.75%, LR: 0.000010
Epoch [20/20], Loss: 0.7657, Train Acc: 74.57%, Val Acc: 60.78%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/15], Loss: 0.7700, Train Acc: 73.06%, Val Acc: 68.63%, LR: 0.000010
Epoch [2/15], Loss: 0.7560, Train Acc: 74.57%, Val Acc: 64.71%, LR: 0.000010
Epoch [3/15], Loss: 0.7474, Train Acc: 74.57%, Val Acc: 64.71%, LR: 0.000010
Epoch [4/15], Loss: 0.7282, Train Acc: 76.08%, Val Acc: 70.59%, LR: 0.000010
Epoch [5/15], Loss: 0.7140, Train Acc: 75.86%, Val Acc: 70.59%, LR: 0.000010
Epoch [6/15], Loss: 0.6882, Train Acc: 81.03%, Val Acc: 72.55%, LR: 0.000010
Epoch [7/15], Loss: 0.6854, Train Acc: 79.96%, Val Acc: 70.59%, LR: 0.000010
Epoch [8/15], Loss: 0.6737, Train Acc: 80.82%, Val Acc: 66.67%, LR: 0.000010
Epoch [9/15], Loss: 0.6551, Train Acc: 82.11%, Val Acc: 68.63%, LR: 0.000010
Epoch [10/15], Loss: 0.6580, Train Acc: 82.33%, Val Acc: 68.63%, LR: 0.000005
Epoch [11/15], Loss: 0.6253, Train Acc: 83.41%, Val Acc: 68.63%, LR: 0.000005
Epoch [12/15], Loss: 0.6566, Train Acc: 81.68%, Val Acc: 70.59%, LR: 0.000005
Epoch [13/15], Loss: 0.6378, Train Acc: 83.62%, Val Acc: 70.59%, LR: 0.000005
Epoch [14/15], Loss: 0.6239, Train Acc: 82.54%, Val Acc: 70.59%, LR: 0.000003
Early stopping at epoch 14
Testing model…

=== split_10_90 Results ===
Accuracy: 0.6264
Precision: 0.6384
Recall: 0.6264
F1-Score: 0.6157

Classification Report:
              precision    recall  f1-score   support

          AD       0.49      0.56      0.52      1012
          CN       0.71      0.39      0.50      1296
         MCI       0.66      0.79      0.72      2331

    accuracy                           0.63      4639
   macro avg       0.62      0.58      0.58      4639
weighted avg       0.64      0.63      0.62      4639
```

```
============================================================
Processing: split_20_80
============================================================
Train samples: 928
Val samples: 103
Test samples: 4123
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.1140, Train Acc: 38.47%, Val Acc: 48.54%, LR: 0.000033
Epoch [2/20], Loss: 1.0469, Train Acc: 49.03%, Val Acc: 49.51%, LR: 0.000067
Epoch [3/20], Loss: 1.0261, Train Acc: 52.26%, Val Acc: 51.46%, LR: 0.000100
Epoch [4/20], Loss: 0.9732, Train Acc: 55.60%, Val Acc: 52.43%, LR: 0.000099
Epoch [5/20], Loss: 0.9655, Train Acc: 56.57%, Val Acc: 54.37%, LR: 0.000098
Epoch [6/20], Loss: 0.9349, Train Acc: 58.94%, Val Acc: 53.40%, LR: 0.000095
Epoch [7/20], Loss: 0.9143, Train Acc: 59.59%, Val Acc: 57.28%, LR: 0.000091
Epoch [8/20], Loss: 0.8819, Train Acc: 62.93%, Val Acc: 60.19%, LR: 0.000086
Epoch [9/20], Loss: 0.8796, Train Acc: 63.36%, Val Acc: 60.19%, LR: 0.000080
Epoch [10/20], Loss: 0.8598, Train Acc: 64.01%, Val Acc: 62.14%, LR: 0.000073
Epoch [11/20], Loss: 0.8257, Train Acc: 67.89%, Val Acc: 62.14%, LR: 0.000066
Epoch [12/20], Loss: 0.8253, Train Acc: 66.38%, Val Acc: 63.11%, LR: 0.000058
Epoch [13/20], Loss: 0.8184, Train Acc: 68.75%, Val Acc: 66.99%, LR: 0.000051
Epoch [14/20], Loss: 0.8072, Train Acc: 68.64%, Val Acc: 63.11%, LR: 0.000043
Epoch [15/20], Loss: 0.7943, Train Acc: 69.40%, Val Acc: 68.93%, LR: 0.000035
Epoch [16/20], Loss: 0.7800, Train Acc: 71.23%, Val Acc: 66.99%, LR: 0.000028
Epoch [17/20], Loss: 0.7642, Train Acc: 71.66%, Val Acc: 66.02%, LR: 0.000021
Epoch [18/20], Loss: 0.7651, Train Acc: 72.74%, Val Acc: 68.93%, LR: 0.000015
Epoch [19/20], Loss: 0.7674, Train Acc: 71.12%, Val Acc: 65.05%, LR: 0.000010
Epoch [20/20], Loss: 0.7620, Train Acc: 73.71%, Val Acc: 66.99%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.7667, Train Acc: 72.31%, Val Acc: 65.05%, LR: 0.000010
Epoch [2/15], Loss: 0.7454, Train Acc: 72.09%, Val Acc: 69.90%, LR: 0.000010
Epoch [3/15], Loss: 0.7389, Train Acc: 74.35%, Val Acc: 69.90%, LR: 0.000010
Epoch [4/15], Loss: 0.7178, Train Acc: 75.43%, Val Acc: 73.79%, LR: 0.000010
Epoch [5/15], Loss: 0.7081, Train Acc: 77.26%, Val Acc: 72.82%, LR: 0.000010
Epoch [6/15], Loss: 0.6806, Train Acc: 77.48%, Val Acc: 71.84%, LR: 0.000010
Epoch [7/15], Loss: 0.6982, Train Acc: 76.51%, Val Acc: 73.79%, LR: 0.000010
Epoch [8/15], Loss: 0.6496, Train Acc: 78.77%, Val Acc: 73.79%, LR: 0.000005
Epoch [9/15], Loss: 0.6598, Train Acc: 79.53%, Val Acc: 76.70%, LR: 0.000005
Epoch [10/15], Loss: 0.6292, Train Acc: 84.70%, Val Acc: 75.73%, LR: 0.000005
Epoch [11/15], Loss: 0.6509, Train Acc: 78.99%, Val Acc: 76.70%, LR: 0.000005
Epoch [12/15], Loss: 0.6334, Train Acc: 82.11%, Val Acc: 73.79%, LR: 0.000005
Epoch [13/15], Loss: 0.6226, Train Acc: 81.79%, Val Acc: 77.67%, LR: 0.000005
Epoch [14/15], Loss: 0.6041, Train Acc: 83.94%, Val Acc: 78.64%, LR: 0.000005
Epoch [15/15], Loss: 0.5866, Train Acc: 85.78%, Val Acc: 80.58%, LR: 0.000005
Testing model…

=== split_20_80 Results ===
Accuracy: 0.7317
Precision: 0.7282
Recall: 0.7317
F1-Score: 0.7261

Classification Report:
              precision    recall  f1-score   support

          AD       0.65      0.54      0.59       899
          CN       0.75      0.66      0.70      1152
         MCI       0.75      0.86      0.80      2072

    accuracy                           0.73      4123
   macro avg       0.72      0.68      0.70      4123
weighted avg       0.73      0.73      0.73      4123


============================================================
Processing: split_30_70
============================================================
Train samples: 1391
Val samples: 155
Test samples: 3608
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0540, Train Acc: 47.52%, Val Acc: 53.55%, LR: 0.000033
Epoch [2/20], Loss: 1.0246, Train Acc: 51.11%, Val Acc: 56.13%, LR: 0.000067
Epoch [3/20], Loss: 0.9905, Train Acc: 53.70%, Val Acc: 60.65%, LR: 0.000100
Epoch [4/20], Loss: 0.9559, Train Acc: 56.36%, Val Acc: 61.94%, LR: 0.000099
Epoch [5/20], Loss: 0.9310, Train Acc: 58.88%, Val Acc: 63.87%, LR: 0.000098
Epoch [6/20], Loss: 0.8967, Train Acc: 62.11%, Val Acc: 63.87%, LR: 0.000095
Epoch [7/20], Loss: 0.8896, Train Acc: 61.61%, Val Acc: 65.16%, LR: 0.000091
Epoch [8/20], Loss: 0.8423, Train Acc: 66.57%, Val Acc: 65.81%, LR: 0.000086
Epoch [9/20], Loss: 0.8457, Train Acc: 65.78%, Val Acc: 67.74%, LR: 0.000080
Epoch [10/20], Loss: 0.8197, Train Acc: 66.50%, Val Acc: 64.52%, LR: 0.000073
Epoch [11/20], Loss: 0.8035, Train Acc: 69.23%, Val Acc: 68.39%, LR: 0.000066
Epoch [12/20], Loss: 0.7875, Train Acc: 70.02%, Val Acc: 67.10%, LR: 0.000058
Epoch [13/20], Loss: 0.7737, Train Acc: 71.89%, Val Acc: 69.03%, LR: 0.000051
Epoch [14/20], Loss: 0.7664, Train Acc: 71.96%, Val Acc: 67.10%, LR: 0.000043
Epoch [15/20], Loss: 0.7706, Train Acc: 70.81%, Val Acc: 68.39%, LR: 0.000035
Epoch [16/20], Loss: 0.7323, Train Acc: 73.18%, Val Acc: 66.45%, LR: 0.000028
Epoch [17/20], Loss: 0.7143, Train Acc: 75.56%, Val Acc: 67.74%, LR: 0.000021
Epoch [18/20], Loss: 0.7145, Train Acc: 75.49%, Val Acc: 69.03%, LR: 0.000015
Epoch [19/20], Loss: 0.7188, Train Acc: 76.28%, Val Acc: 69.68%, LR: 0.000010
Epoch [20/20], Loss: 0.7126, Train Acc: 75.77%, Val Acc: 70.97%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.7177, Train Acc: 74.91%, Val Acc: 69.03%, LR: 0.000010
Epoch [2/15], Loss: 0.7031, Train Acc: 76.71%, Val Acc: 69.68%, LR: 0.000010
Epoch [3/15], Loss: 0.6930, Train Acc: 75.92%, Val Acc: 72.90%, LR: 0.000010
Epoch [4/15], Loss: 0.6771, Train Acc: 78.00%, Val Acc: 69.68%, LR: 0.000010
Epoch [5/15], Loss: 0.6608, Train Acc: 78.94%, Val Acc: 70.97%, LR: 0.000010
Epoch [6/15], Loss: 0.6213, Train Acc: 82.31%, Val Acc: 74.84%, LR: 0.000010
Epoch [7/15], Loss: 0.6317, Train Acc: 80.95%, Val Acc: 69.03%, LR: 0.000010
Epoch [8/15], Loss: 0.6125, Train Acc: 83.54%, Val Acc: 70.32%, LR: 0.000010
Epoch [9/15], Loss: 0.6018, Train Acc: 82.75%, Val Acc: 75.48%, LR: 0.000010
Epoch [10/15], Loss: 0.5915, Train Acc: 84.47%, Val Acc: 72.90%, LR: 0.000010
Epoch [11/15], Loss: 0.5831, Train Acc: 85.12%, Val Acc: 72.90%, LR: 0.000010
Epoch [12/15], Loss: 0.5761, Train Acc: 84.97%, Val Acc: 71.61%, LR: 0.000010
Epoch [13/15], Loss: 0.5423, Train Acc: 87.28%, Val Acc: 74.84%, LR: 0.000005
Epoch [14/15], Loss: 0.5420, Train Acc: 87.42%, Val Acc: 74.19%, LR: 0.000005
Epoch [15/15], Loss: 0.5399, Train Acc: 87.56%, Val Acc: 74.19%, LR: 0.000005
Testing model…

=== split_30_70 Results ===
Accuracy: 0.8071
Precision: 0.8058
Recall: 0.8071
F1-Score: 0.8056

Classification Report:
              precision    recall  f1-score   support

          AD       0.76      0.70      0.72       787
          CN       0.81      0.77      0.79      1008
         MCI       0.82      0.88      0.85      1813

    accuracy                           0.81      3608
   macro avg       0.80      0.78      0.79      3608
weighted avg       0.81      0.81      0.81      3608


============================================================
Processing: split_40_60
============================================================
Train samples: 1855
Val samples: 207
Test samples: 3092
```

```
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0519, Train Acc: 49.65%, Val Acc: 53.14%, LR: 0.000033
Epoch [2/20], Loss: 1.0122, Train Acc: 52.29%, Val Acc: 54.11%, LR: 0.000067
Epoch [3/20], Loss: 0.9748, Train Acc: 55.09%, Val Acc: 56.04%, LR: 0.000100
Epoch [4/20], Loss: 0.9387, Train Acc: 58.22%, Val Acc: 55.56%, LR: 0.000099
Epoch [5/20], Loss: 0.8999, Train Acc: 61.62%, Val Acc: 62.80%, LR: 0.000098
Epoch [6/20], Loss: 0.8769, Train Acc: 64.74%, Val Acc: 61.84%, LR: 0.000095
Epoch [7/20], Loss: 0.8388, Train Acc: 66.25%, Val Acc: 57.97%, LR: 0.000091
Epoch [8/20], Loss: 0.8208, Train Acc: 67.39%, Val Acc: 57.00%, LR: 0.000086
Epoch [9/20], Loss: 0.7928, Train Acc: 70.35%, Val Acc: 64.25%, LR: 0.000080
Epoch [10/20], Loss: 0.7676, Train Acc: 71.86%, Val Acc: 61.84%, LR: 0.000073
Epoch [11/20], Loss: 0.7444, Train Acc: 74.02%, Val Acc: 66.67%, LR: 0.000066
Epoch [12/20], Loss: 0.7452, Train Acc: 73.48%, Val Acc: 65.70%, LR: 0.000058
Epoch [13/20], Loss: 0.7327, Train Acc: 74.77%, Val Acc: 68.60%, LR: 0.000051
Epoch [14/20], Loss: 0.7052, Train Acc: 76.17%, Val Acc: 61.84%, LR: 0.000043
Epoch [15/20], Loss: 0.7107, Train Acc: 75.09%, Val Acc: 67.15%, LR: 0.000035
Epoch [16/20], Loss: 0.6822, Train Acc: 77.30%, Val Acc: 68.12%, LR: 0.000028
Epoch [17/20], Loss: 0.6831, Train Acc: 77.04%, Val Acc: 67.63%, LR: 0.000021
Epoch [18/20], Loss: 0.6669, Train Acc: 79.14%, Val Acc: 70.05%, LR: 0.000015
Epoch [19/20], Loss: 0.6658, Train Acc: 78.76%, Val Acc: 68.60%, LR: 0.000010
Epoch [20/20], Loss: 0.6826, Train Acc: 77.68%, Val Acc: 69.57%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6779, Train Acc: 78.49%, Val Acc: 68.60%, LR: 0.000010
Epoch [2/15], Loss: 0.6554, Train Acc: 79.68%, Val Acc: 73.43%, LR: 0.000010
Epoch [3/15], Loss: 0.6293, Train Acc: 81.78%, Val Acc: 71.50%, LR: 0.000010
Epoch [4/15], Loss: 0.6201, Train Acc: 81.67%, Val Acc: 72.46%, LR: 0.000010
Epoch [5/15], Loss: 0.6024, Train Acc: 83.07%, Val Acc: 76.33%, LR: 0.000010
Epoch [6/15], Loss: 0.5865, Train Acc: 84.10%, Val Acc: 78.74%, LR: 0.000010
Epoch [7/15], Loss: 0.5897, Train Acc: 84.74%, Val Acc: 78.74%, LR: 0.000010
Epoch [8/15], Loss: 0.5706, Train Acc: 85.44%, Val Acc: 82.13%, LR: 0.000010
Epoch [9/15], Loss: 0.5521, Train Acc: 87.22%, Val Acc: 79.23%, LR: 0.000010
Epoch [10/15], Loss: 0.5446, Train Acc: 86.85%, Val Acc: 78.26%, LR: 0.000010
Epoch [11/15], Loss: 0.5328, Train Acc: 88.09%, Val Acc: 82.61%, LR: 0.000010
Epoch [12/15], Loss: 0.5221, Train Acc: 88.36%, Val Acc: 79.23%, LR: 0.000010
Epoch [13/15], Loss: 0.5229, Train Acc: 88.19%, Val Acc: 82.61%, LR: 0.000010
Epoch [14/15], Loss: 0.5103, Train Acc: 89.81%, Val Acc: 81.16%, LR: 0.000010
Epoch [15/15], Loss: 0.4857, Train Acc: 91.37%, Val Acc: 80.68%, LR: 0.000005
Testing model…

=== split_40_60 Results ===
Accuracy: 0.8195
Precision: 0.8268
Recall: 0.8195
```

```
F1-Score: 0.8207

Classification Report:
              precision    recall  f1-score   support

          AD       0.69      0.81      0.74       674
          CN       0.88      0.75      0.81       864
         MCI       0.86      0.87      0.86      1554

    accuracy                           0.82      3092
   macro avg       0.81      0.81      0.80      3092
weighted avg       0.83      0.82      0.82      3092


============================================================
Processing: split_50_50
============================================================
Train samples: 2319
Val samples: 258
Test samples: 2577
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0487, Train Acc: 47.99%, Val Acc: 51.94%, LR: 0.000033
Epoch [2/20], Loss: 1.0032, Train Acc: 53.39%, Val Acc: 58.91%, LR: 0.000067
Epoch [3/20], Loss: 0.9598, Train Acc: 56.71%, Val Acc: 55.04%, LR: 0.000100
Epoch [4/20], Loss: 0.9081, Train Acc: 60.54%, Val Acc: 56.98%, LR: 0.000099
Epoch [5/20], Loss: 0.8767, Train Acc: 63.13%, Val Acc: 58.91%, LR: 0.000098
Epoch [6/20], Loss: 0.8543, Train Acc: 65.93%, Val Acc: 62.02%, LR: 0.000095
Epoch [7/20], Loss: 0.8176, Train Acc: 68.39%, Val Acc: 65.50%, LR: 0.000091
Epoch [8/20], Loss: 0.7861, Train Acc: 70.25%, Val Acc: 63.57%, LR: 0.000086
Epoch [9/20], Loss: 0.7786, Train Acc: 71.45%, Val Acc: 63.57%, LR: 0.000080
Epoch [10/20], Loss: 0.7593, Train Acc: 72.75%, Val Acc: 67.05%, LR: 0.000073
Epoch [11/20], Loss: 0.7521, Train Acc: 73.01%, Val Acc: 70.16%, LR: 0.000066
Epoch [12/20], Loss: 0.7236, Train Acc: 74.51%, Val Acc: 69.38%, LR: 0.000058
Epoch [13/20], Loss: 0.7104, Train Acc: 74.60%, Val Acc: 69.77%, LR: 0.000051
Epoch [14/20], Loss: 0.6980, Train Acc: 76.89%, Val Acc: 69.38%, LR: 0.000043
Epoch [15/20], Loss: 0.6845, Train Acc: 78.01%, Val Acc: 70.16%, LR: 0.000035
Epoch [16/20], Loss: 0.6713, Train Acc: 78.70%, Val Acc: 70.93%, LR: 0.000028
Epoch [17/20], Loss: 0.6743, Train Acc: 78.83%, Val Acc: 75.58%, LR: 0.000021
Epoch [18/20], Loss: 0.6533, Train Acc: 79.95%, Val Acc: 72.48%, LR: 0.000015
Epoch [19/20], Loss: 0.6567, Train Acc: 79.17%, Val Acc: 74.03%, LR: 0.000010
Epoch [20/20], Loss: 0.6605, Train Acc: 78.53%, Val Acc: 74.03%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6585, Train Acc: 79.26%, Val Acc: 76.74%, LR: 0.000010
Epoch [2/15], Loss: 0.6389, Train Acc: 81.46%, Val Acc: 74.81%, LR: 0.000010
Epoch [3/15], Loss: 0.6233, Train Acc: 81.72%, Val Acc: 75.58%, LR: 0.000010
Epoch [4/15], Loss: 0.6130, Train Acc: 82.66%, Val Acc: 75.19%, LR: 0.000010
Epoch [5/15], Loss: 0.6014, Train Acc: 83.35%, Val Acc: 82.17%, LR: 0.000010
Epoch [6/15], Loss: 0.5791, Train Acc: 84.56%, Val Acc: 81.01%, LR: 0.000010
Epoch [7/15], Loss: 0.5678, Train Acc: 85.12%, Val Acc: 81.78%, LR: 0.000010
Epoch [8/15], Loss: 0.5487, Train Acc: 87.02%, Val Acc: 81.40%, LR: 0.000010
Epoch [9/15], Loss: 0.5316, Train Acc: 87.54%, Val Acc: 82.56%, LR: 0.000010
Epoch [10/15], Loss: 0.5181, Train Acc: 88.70%, Val Acc: 86.05%, LR: 0.000010
Epoch [11/15], Loss: 0.5296, Train Acc: 88.18%, Val Acc: 85.66%, LR: 0.000010
Epoch [12/15], Loss: 0.5142, Train Acc: 89.18%, Val Acc: 85.27%, LR: 0.000010
Epoch [13/15], Loss: 0.5033, Train Acc: 90.38%, Val Acc: 83.72%, LR: 0.000010
Epoch [14/15], Loss: 0.5059, Train Acc: 89.00%, Val Acc: 86.43%, LR: 0.000010
Epoch [15/15], Loss: 0.4979, Train Acc: 89.87%, Val Acc: 86.82%, LR: 0.000010
Testing model…

=== split_50_50 Results ===
Accuracy: 0.8716
Precision: 0.8751
Recall: 0.8716
F1-Score: 0.8725

Classification Report:
              precision    recall  f1-score   support

          AD       0.77      0.87      0.82       562
          CN       0.88      0.84      0.86       720
         MCI       0.92      0.89      0.90      1295

    accuracy                           0.87      2577
   macro avg       0.86      0.87      0.86      2577
weighted avg       0.88      0.87      0.87      2577


============================================================
Processing: split_60_40
============================================================
Train samples: 2784
Val samples: 308
Test samples: 2062
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0475, Train Acc: 48.92%, Val Acc: 48.70%, LR: 0.000033
Epoch [2/20], Loss: 0.9968, Train Acc: 53.92%, Val Acc: 50.97%, LR: 0.000067
Epoch [3/20], Loss: 0.9575, Train Acc: 56.86%, Val Acc: 59.74%, LR: 0.000100
Epoch [4/20], Loss: 0.9086, Train Acc: 61.89%, Val Acc: 61.36%, LR: 0.000099
Epoch [5/20], Loss: 0.8655, Train Acc: 64.51%, Val Acc: 62.66%, LR: 0.000098
Epoch [6/20], Loss: 0.8505, Train Acc: 66.06%, Val Acc: 64.29%, LR: 0.000095
Epoch [7/20], Loss: 0.8170, Train Acc: 68.35%, Val Acc: 65.26%, LR: 0.000091
Epoch [8/20], Loss: 0.7786, Train Acc: 70.58%, Val Acc: 69.81%, LR: 0.000086
Epoch [9/20], Loss: 0.7693, Train Acc: 71.19%, Val Acc: 69.48%, LR: 0.000080
Epoch [10/20], Loss: 0.7483, Train Acc: 73.64%, Val Acc: 69.48%, LR: 0.000073
Epoch [11/20], Loss: 0.7227, Train Acc: 75.04%, Val Acc: 72.08%, LR: 0.000066
Epoch [12/20], Loss: 0.7084, Train Acc: 75.11%, Val Acc: 74.03%, LR: 0.000058
Epoch [13/20], Loss: 0.6960, Train Acc: 76.08%, Val Acc: 72.73%, LR: 0.000051
Epoch [14/20], Loss: 0.6846, Train Acc: 77.19%, Val Acc: 75.65%, LR: 0.000043
Epoch [15/20], Loss: 0.6648, Train Acc: 78.81%, Val Acc: 75.00%, LR: 0.000035
Epoch [16/20], Loss: 0.6668, Train Acc: 78.59%, Val Acc: 74.03%, LR: 0.000028
Epoch [17/20], Loss: 0.6599, Train Acc: 78.34%, Val Acc: 74.03%, LR: 0.000021
Epoch [18/20], Loss: 0.6539, Train Acc: 80.14%, Val Acc: 74.03%, LR: 0.000015
Epoch [19/20], Loss: 0.6418, Train Acc: 80.24%, Val Acc: 72.40%, LR: 0.000010
Epoch [20/20], Loss: 0.6403, Train Acc: 80.10%, Val Acc: 74.68%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/15], Loss: 0.6539, Train Acc: 79.71%, Val Acc: 74.35%, LR: 0.000010
Epoch [2/15], Loss: 0.6256, Train Acc: 81.50%, Val Acc: 76.62%, LR: 0.000010
Epoch [3/15], Loss: 0.6100, Train Acc: 82.58%, Val Acc: 77.92%, LR: 0.000010
Epoch [4/15], Loss: 0.5918, Train Acc: 83.66%, Val Acc: 77.27%, LR: 0.000010
Epoch [5/15], Loss: 0.5785, Train Acc: 84.30%, Val Acc: 80.19%, LR: 0.000010
Epoch [6/15], Loss: 0.5573, Train Acc: 86.35%, Val Acc: 82.79%, LR: 0.000010
Epoch [7/15], Loss: 0.5599, Train Acc: 86.03%, Val Acc: 82.79%, LR: 0.000010
Epoch [8/15], Loss: 0.5346, Train Acc: 87.57%, Val Acc: 87.01%, LR: 0.000010
Epoch [9/15], Loss: 0.5199, Train Acc: 88.36%, Val Acc: 86.36%, LR: 0.000010
Epoch [10/15], Loss: 0.5160, Train Acc: 88.76%, Val Acc: 87.66%, LR: 0.000010
Epoch [11/15], Loss: 0.5110, Train Acc: 88.58%, Val Acc: 86.69%, LR: 0.000010
Epoch [12/15], Loss: 0.5020, Train Acc: 89.80%, Val Acc: 86.36%, LR: 0.000010
Epoch [13/15], Loss: 0.4948, Train Acc: 90.55%, Val Acc: 89.29%, LR: 0.000010
Epoch [14/15], Loss: 0.4858, Train Acc: 90.70%, Val Acc: 88.96%, LR: 0.000010
Epoch [15/15], Loss: 0.4771, Train Acc: 91.16%, Val Acc: 88.64%, LR: 0.000010
Testing model…

=== split_60_40 Results ===
Accuracy: 0.9035
Precision: 0.9042
Recall: 0.9035
F1-Score: 0.9024

Classification Report:
              precision    recall  f1-score   support

          AD       0.91      0.79      0.85       450
          CN       0.92      0.89      0.91       576
         MCI       0.89      0.96      0.92      1036
```

```
    accuracy                           0.90      2062
   macro avg       0.91      0.88      0.89      2062
weighted avg       0.90      0.90      0.90      2062


============================================================
Processing: split_70_30
============================================================
Train samples: 3247
Val samples: 361
Test samples: 1546
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0488, Train Acc: 48.41%, Val Acc: 52.08%, LR: 0.000033
Epoch [2/20], Loss: 0.9982, Train Acc: 53.83%, Val Acc: 52.63%, LR: 0.000067
Epoch [3/20], Loss: 0.9473, Train Acc: 57.99%, Val Acc: 56.51%, LR: 0.000100
Epoch [4/20], Loss: 0.9046, Train Acc: 61.38%, Val Acc: 63.43%, LR: 0.000099
Epoch [5/20], Loss: 0.8608, Train Acc: 65.85%, Val Acc: 63.71%, LR: 0.000098
Epoch [6/20], Loss: 0.8240, Train Acc: 67.48%, Val Acc: 67.87%, LR: 0.000095
Epoch [7/20], Loss: 0.7917, Train Acc: 69.73%, Val Acc: 63.71%, LR: 0.000091
Epoch [8/20], Loss: 0.7693, Train Acc: 72.16%, Val Acc: 71.75%, LR: 0.000086
Epoch [9/20], Loss: 0.7383, Train Acc: 72.96%, Val Acc: 73.41%, LR: 0.000080
Epoch [10/20], Loss: 0.7190, Train Acc: 75.08%, Val Acc: 76.73%, LR: 0.000073
Epoch [11/20], Loss: 0.7013, Train Acc: 76.47%, Val Acc: 75.07%, LR: 0.000066
Epoch [12/20], Loss: 0.6813, Train Acc: 77.24%, Val Acc: 77.01%, LR: 0.000058
Epoch [13/20], Loss: 0.6808, Train Acc: 78.72%, Val Acc: 76.18%, LR: 0.000051
Epoch [14/20], Loss: 0.6672, Train Acc: 78.44%, Val Acc: 76.45%, LR: 0.000043
Epoch [15/20], Loss: 0.6609, Train Acc: 79.12%, Val Acc: 76.18%, LR: 0.000035
Epoch [16/20], Loss: 0.6491, Train Acc: 79.67%, Val Acc: 75.90%, LR: 0.000028
Epoch [17/20], Loss: 0.6297, Train Acc: 81.06%, Val Acc: 78.12%, LR: 0.000021
Epoch [18/20], Loss: 0.6318, Train Acc: 81.71%, Val Acc: 76.73%, LR: 0.000015
Epoch [19/20], Loss: 0.6314, Train Acc: 81.52%, Val Acc: 78.67%, LR: 0.000010
Epoch [20/20], Loss: 0.6229, Train Acc: 81.52%, Val Acc: 76.73%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/15], Loss: 0.6357, Train Acc: 80.04%, Val Acc: 78.39%, LR: 0.000010
Epoch [2/15], Loss: 0.6110, Train Acc: 82.23%, Val Acc: 80.89%, LR: 0.000010
Epoch [3/15], Loss: 0.5947, Train Acc: 83.22%, Val Acc: 82.55%, LR: 0.000010
Epoch [4/15], Loss: 0.5676, Train Acc: 85.34%, Val Acc: 83.93%, LR: 0.000010
Epoch [5/15], Loss: 0.5550, Train Acc: 86.60%, Val Acc: 84.49%, LR: 0.000010
Epoch [6/15], Loss: 0.5521, Train Acc: 86.02%, Val Acc: 87.81%, LR: 0.000010
Epoch [7/15], Loss: 0.5252, Train Acc: 87.99%, Val Acc: 84.21%, LR: 0.000010
Epoch [8/15], Loss: 0.5191, Train Acc: 88.51%, Val Acc: 88.37%, LR: 0.000010
Epoch [9/15], Loss: 0.5121, Train Acc: 88.45%, Val Acc: 86.98%, LR: 0.000010
Epoch [10/15], Loss: 0.5021, Train Acc: 89.59%, Val Acc: 88.37%, LR: 0.000010
Epoch [11/15], Loss: 0.4959, Train Acc: 89.93%, Val Acc: 89.20%, LR: 0.000010
Epoch [12/15], Loss: 0.4786, Train Acc: 90.82%, Val Acc: 90.03%, LR: 0.000010
Epoch [13/15], Loss: 0.4793, Train Acc: 91.22%, Val Acc: 90.86%, LR: 0.000010
Epoch [14/15], Loss: 0.4633, Train Acc: 91.87%, Val Acc: 89.75%, LR: 0.000010
Epoch [15/15], Loss: 0.4578, Train Acc: 92.52%, Val Acc: 89.47%, LR: 0.000010
Testing model…

=== split_70_30 Results ===
Accuracy: 0.9069
Precision: 0.9075
Recall: 0.9069
F1-Score: 0.9068

Classification Report:
              precision    recall  f1-score   support

          AD       0.87      0.83      0.85       337
          CN       0.87      0.93      0.90       432
         MCI       0.94      0.93      0.94       777

    accuracy                           0.91      1546
   macro avg       0.90      0.90      0.90      1546
weighted avg       0.91      0.91      0.91      1546


============================================================
Processing: split_80_20
============================================================
Train samples: 3711
Val samples: 412
Test samples: 1031
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0421, Train Acc: 49.02%, Val Acc: 53.64%, LR: 0.000033
Epoch [2/20], Loss: 0.9811, Train Acc: 54.43%, Val Acc: 57.52%, LR: 0.000067
Epoch [3/20], Loss: 0.9220, Train Acc: 60.36%, Val Acc: 59.47%, LR: 0.000100
Epoch [4/20], Loss: 0.8775, Train Acc: 64.21%, Val Acc: 67.48%, LR: 0.000099
Epoch [5/20], Loss: 0.8368, Train Acc: 67.21%, Val Acc: 68.45%, LR: 0.000098
Epoch [6/20], Loss: 0.8026, Train Acc: 69.23%, Val Acc: 71.60%, LR: 0.000095
Epoch [7/20], Loss: 0.7642, Train Acc: 72.00%, Val Acc: 68.93%, LR: 0.000091
Epoch [8/20], Loss: 0.7429, Train Acc: 73.43%, Val Acc: 73.30%, LR: 0.000086
Epoch [9/20], Loss: 0.7226, Train Acc: 74.75%, Val Acc: 74.51%, LR: 0.000080
Epoch [10/20], Loss: 0.6962, Train Acc: 76.26%, Val Acc: 74.03%, LR: 0.000073
Epoch [11/20], Loss: 0.7031, Train Acc: 75.69%, Val Acc: 75.97%, LR: 0.000066
Epoch [12/20], Loss: 0.6900, Train Acc: 76.21%, Val Acc: 76.70%, LR: 0.000058
Epoch [13/20], Loss: 0.6599, Train Acc: 78.63%, Val Acc: 75.49%, LR: 0.000051
Epoch [14/20], Loss: 0.6466, Train Acc: 80.54%, Val Acc: 77.91%, LR: 0.000043
Epoch [15/20], Loss: 0.6390, Train Acc: 80.19%, Val Acc: 78.88%, LR: 0.000035
Epoch [16/20], Loss: 0.6314, Train Acc: 81.03%, Val Acc: 79.85%, LR: 0.000028
Epoch [17/20], Loss: 0.6286, Train Acc: 81.16%, Val Acc: 79.37%, LR: 0.000021
Epoch [18/20], Loss: 0.6317, Train Acc: 80.01%, Val Acc: 81.07%, LR: 0.000015
Epoch [19/20], Loss: 0.6182, Train Acc: 82.27%, Val Acc: 79.85%, LR: 0.000010
Epoch [20/20], Loss: 0.6115, Train Acc: 82.54%, Val Acc: 81.31%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6239, Train Acc: 81.81%, Val Acc: 80.58%, LR: 0.000010
Epoch [2/15], Loss: 0.5982, Train Acc: 82.86%, Val Acc: 82.77%, LR: 0.000010
Epoch [3/15], Loss: 0.5829, Train Acc: 84.10%, Val Acc: 84.22%, LR: 0.000010
Epoch [4/15], Loss: 0.5781, Train Acc: 84.29%, Val Acc: 80.58%, LR: 0.000010
Epoch [5/15], Loss: 0.5615, Train Acc: 85.93%, Val Acc: 85.92%, LR: 0.000010
Epoch [6/15], Loss: 0.5474, Train Acc: 86.58%, Val Acc: 88.59%, LR: 0.000010
Epoch [7/15], Loss: 0.5226, Train Acc: 87.98%, Val Acc: 87.86%, LR: 0.000010
Epoch [8/15], Loss: 0.5064, Train Acc: 89.46%, Val Acc: 87.62%, LR: 0.000010
Epoch [9/15], Loss: 0.5001, Train Acc: 90.14%, Val Acc: 88.59%, LR: 0.000010
Epoch [10/15], Loss: 0.4914, Train Acc: 90.62%, Val Acc: 87.86%, LR: 0.000005
Epoch [11/15], Loss: 0.4749, Train Acc: 91.70%, Val Acc: 90.05%, LR: 0.000005
Epoch [12/15], Loss: 0.4557, Train Acc: 92.54%, Val Acc: 90.05%, LR: 0.000005
Epoch [13/15], Loss: 0.4616, Train Acc: 92.24%, Val Acc: 92.23%, LR: 0.000005
Epoch [14/15], Loss: 0.4552, Train Acc: 93.05%, Val Acc: 91.50%, LR: 0.000005
Epoch [15/15], Loss: 0.4602, Train Acc: 92.29%, Val Acc: 91.99%, LR: 0.000005
Testing model…

=== split_80_20 Results ===
Accuracy: 0.9166
Precision: 0.9166
Recall: 0.9166
F1-Score: 0.9165

Classification Report:
              precision    recall  f1-score   support

          AD       0.87      0.87      0.87       225
          CN       0.94      0.92      0.93       288
         MCI       0.92      0.94      0.93       518

    accuracy                           0.92      1031
   macro avg       0.91      0.91      0.91      1031
weighted avg       0.92      0.92      0.92      1031


============================================================
Processing: split_90_10
```

```
============================================================
Train samples: 4175
Val samples: 464
Test samples: 515
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0574, Train Acc: 46.35%, Val Acc: 51.29%, LR: 0.000033
Epoch [2/20], Loss: 0.9790, Train Acc: 55.47%, Val Acc: 56.68%, LR: 0.000067
Epoch [3/20], Loss: 0.9216, Train Acc: 60.74%, Val Acc: 62.07%, LR: 0.000100
Epoch [4/20], Loss: 0.8730, Train Acc: 63.95%, Val Acc: 66.81%, LR: 0.000099
Epoch [5/20], Loss: 0.8264, Train Acc: 67.50%, Val Acc: 68.10%, LR: 0.000098
Epoch [6/20], Loss: 0.7824, Train Acc: 70.44%, Val Acc: 68.10%, LR: 0.000095
Epoch [7/20], Loss: 0.7588, Train Acc: 71.90%, Val Acc: 73.92%, LR: 0.000091
Epoch [8/20], Loss: 0.7268, Train Acc: 74.44%, Val Acc: 72.84%, LR: 0.000086
Epoch [9/20], Loss: 0.7088, Train Acc: 75.69%, Val Acc: 75.86%, LR: 0.000080
Epoch [10/20], Loss: 0.6920, Train Acc: 76.72%, Val Acc: 76.94%, LR: 0.000073
Epoch [11/20], Loss: 0.6703, Train Acc: 78.08%, Val Acc: 75.86%, LR: 0.000066
Epoch [12/20], Loss: 0.6656, Train Acc: 78.80%, Val Acc: 79.96%, LR: 0.000058
Epoch [13/20], Loss: 0.6386, Train Acc: 80.53%, Val Acc: 77.16%, LR: 0.000051
Epoch [14/20], Loss: 0.6336, Train Acc: 80.38%, Val Acc: 81.25%, LR: 0.000043
Epoch [15/20], Loss: 0.6372, Train Acc: 80.41%, Val Acc: 80.39%, LR: 0.000035
Epoch [16/20], Loss: 0.6110, Train Acc: 82.44%, Val Acc: 82.11%, LR: 0.000028
Epoch [17/20], Loss: 0.6039, Train Acc: 82.30%, Val Acc: 82.33%, LR: 0.000021
Epoch [18/20], Loss: 0.5964, Train Acc: 83.57%, Val Acc: 82.11%, LR: 0.000015
Epoch [19/20], Loss: 0.5953, Train Acc: 83.07%, Val Acc: 84.05%, LR: 0.000010
Epoch [20/20], Loss: 0.5896, Train Acc: 83.43%, Val Acc: 85.34%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6086, Train Acc: 82.63%, Val Acc: 79.96%, LR: 0.000010
Epoch [2/15], Loss: 0.5839, Train Acc: 83.98%, Val Acc: 84.27%, LR: 0.000010
Epoch [3/15], Loss: 0.5504, Train Acc: 86.66%, Val Acc: 84.91%, LR: 0.000010
Epoch [4/15], Loss: 0.5460, Train Acc: 86.66%, Val Acc: 86.42%, LR: 0.000010
Epoch [5/15], Loss: 0.5340, Train Acc: 87.33%, Val Acc: 86.85%, LR: 0.000010
Epoch [6/15], Loss: 0.5237, Train Acc: 87.86%, Val Acc: 88.15%, LR: 0.000010
Epoch [7/15], Loss: 0.5029, Train Acc: 89.53%, Val Acc: 89.66%, LR: 0.000010
Epoch [8/15], Loss: 0.4896, Train Acc: 90.97%, Val Acc: 90.73%, LR: 0.000010
Epoch [9/15], Loss: 0.4873, Train Acc: 90.99%, Val Acc: 87.28%, LR: 0.000010
Epoch [10/15], Loss: 0.4752, Train Acc: 91.62%, Val Acc: 89.87%, LR: 0.000010
Epoch [11/15], Loss: 0.4688, Train Acc: 91.88%, Val Acc: 91.59%, LR: 0.000010
Epoch [12/15], Loss: 0.4547, Train Acc: 92.41%, Val Acc: 92.89%, LR: 0.000010
Epoch [13/15], Loss: 0.4492, Train Acc: 93.20%, Val Acc: 92.24%, LR: 0.000010
Epoch [14/15], Loss: 0.4454, Train Acc: 93.51%, Val Acc: 93.32%, LR: 0.000010
Epoch [15/15], Loss: 0.4379, Train Acc: 93.75%, Val Acc: 92.89%, LR: 0.000010
Testing model…
```

```
=== split_90_10 Results ===
Accuracy: 0.9107
Precision: 0.9128
Recall: 0.9107
F1-Score: 0.9097

Classification Report:
              precision    recall  f1-score   support

          AD       0.89      0.85      0.87       112
          CN       0.96      0.84      0.90       144
         MCI       0.90      0.98      0.94       259

    accuracy                           0.91       515
   macro avg       0.92      0.89      0.90       515
weighted avg       0.91      0.91      0.91       515


================================================================================
DenseNet-121 - SUMMARY OF ALL SPLITS
================================================================================
      split  accuracy  precision   recall  f1_score  training_time
split_10_90  0.626428   0.638367 0.626428  0.615734      88.385802
split_20_80  0.731749   0.728188 0.731749  0.726099     169.700626
split_30_70  0.807095   0.805800 0.807095  0.805617     248.054758
split_40_60  0.819534   0.826809 0.819534  0.820689     333.181419
split_50_50  0.871556   0.875118 0.871556  0.872503     414.224900
split_60_40  0.903492   0.904215 0.903492  0.902374     491.754236
split_70_30  0.906856   0.907481 0.906856  0.906768     562.409564
split_80_20  0.916586   0.916611 0.916586  0.916521     641.465663
split_90_10  0.910680   0.912797 0.910680  0.909669     720.117219

Detailed results saved to: /kaggle/working/densenet121_results.csv
```

[8]:
```python
#DenseNet-121
import os
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
import pandas as pd
import numpy as np
from PIL import Image
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪accuracy_score, classification_report
import time
from torch.optim.lr_scheduler import CosineAnnealingLR, ReduceLROnPlateau
```

```
class AlzheimerDataset(Dataset):
    def __init__(self, split_dir, split_type, transform=None):
        self.split_dir = split_dir
        self.split_type = split_type
        self.transform = transform

        csv_path = os.path.join(split_dir, f"{split_type}.csv")
        self.df = pd.read_csv(csv_path)

        self.class_to_idx = {"AD": 0, "CN": 1, "MCI": 2}
        self.idx_to_class = {v: k for k, v in self.class_to_idx.items()}

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img_path = row['path']
        label = self.class_to_idx[row['class']]

        image = Image.open(img_path).convert('RGB')

        if self.transform:
            image = self.transform(image)

        return image, label

def get_data_transforms():
    train_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomRotation(10),
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, ␣
 ↪hue=0.1),
        transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
        transforms.RandomGrayscale(p=0.1),
        transforms.GaussianBlur(kernel_size=3, sigma=(0.1, 2.0)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
 ↪225])
    ])

    val_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
↪225])
    ])

    return train_transform, val_transform

def train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs, device, warmup_epochs=3):
    best_val_acc = 0
    patience = 8
    patience_counter = 0

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        # Learning rate warmup
        if epoch < warmup_epochs:
            lr_scale = min(1.0, float(epoch + 1) / warmup_epochs)
            for param_group in optimizer.param_groups:
                param_group['lr'] = param_group['initial_lr'] * lr_scale

        for batch_idx, (images, labels) in enumerate(train_loader):
            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()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(train_loader)
        epoch_acc = 100 * correct / total

        val_acc = evaluate_model(model, val_loader, device)

        if scheduler and epoch >= warmup_epochs:
            if isinstance(scheduler, CosineAnnealingLR):
                scheduler.step()
            else:
                scheduler.step(val_acc)

        current_lr = optimizer.param_groups[0]['lr']
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Train␣
↪Acc: {epoch_acc:.2f}%, Val Acc: {val_acc:.2f}%, LR: {current_lr:.6f}')

        if val_acc > best_val_acc:
            best_val_acc = val_acc
            patience_counter = 0
        else:
            patience_counter += 1

        if patience_counter >= patience:
            print(f"Early stopping at epoch {epoch+1}")
            break

def evaluate_model(model, data_loader, device):
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad():
        for images, labels in data_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    return accuracy

def test_model(model, test_loader, device):
    model.eval()
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)

            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())
    return all_preds, all_labels

def calculate_metrics(y_true, y_pred, split_name):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted',␣
 ↪zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)

    print(f"\n=== {split_name} Results ===")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred, target_names=['AD', 'CN',␣
 ↪'MCI'], zero_division=0))

    return {
        'split': split_name,
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1_score': f1
    }

def run_densenet121_on_splits():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    print(f"Using device: {device}")

    splits_root = "/kaggle/working/alzheimer-resized-224_splits"
    split_folders = [f for f in os.listdir(splits_root) if f.
 ↪startswith('split_')]
    split_folders.sort()

    results = []

    train_transform, val_transform = get_data_transforms()

    for split_folder in split_folders:
        print(f"\n{'='*60}")
        print(f"Processing: {split_folder}")
        print(f"{'='*60}")

        split_path = os.path.join(splits_root, split_folder)
        train_dataset = AlzheimerDataset(split_path, 'train', train_transform)
        val_dataset = AlzheimerDataset(split_path, 'val', val_transform)
        test_dataset = AlzheimerDataset(split_path, 'test', val_transform)

        train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True,␣
↪num_workers=2, pin_memory=True)
        val_loader = DataLoader(val_dataset, batch_size=16, shuffle=False,␣
↪num_workers=2, pin_memory=True)
        test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False,␣
↪num_workers=2, pin_memory=True)

        print(f"Train samples: {len(train_dataset)}")
        print(f"Val samples: {len(val_dataset)}")
        print(f"Test samples: {len(test_dataset)}")

        model = models.densenet121(weights=models.DenseNet121_Weights.
↪IMAGENET1K_V1)

        # Freeze all layers initially
        for param in model.parameters():
            param.requires_grad = False

        # Unfreeze classifier
        num_ftrs = model.classifier.in_features
        model.classifier = nn.Linear(num_ftrs, 3)

        # Unfreeze last dense block and transition layer
        for param in model.features.denseblock4.parameters():
            param.requires_grad = True
        for param in model.features.norm5.parameters():
            param.requires_grad = True

        model = model.to(device)

        # Layer-wise learning rates for DenseNet
        optimizer = torch.optim.AdamW([
            {'params': model.classifier.parameters(), 'lr': 0.0001,␣
↪'initial_lr': 0.0001},
            {'params': model.features.denseblock4.parameters(), 'lr': 0.00005,␣
↪'initial_lr': 0.00005},
            {'params': model.features.norm5.parameters(), 'lr': 0.00005,␣
↪'initial_lr': 0.00005},
            {'params': model.features.denseblock3.parameters(), 'lr': 0.00001,␣
↪'initial_lr': 0.00001}
        ], weight_decay=1e-4)
        criterion = nn.CrossEntropyLoss(label_smoothing=0.1)
        scheduler = CosineAnnealingLR(optimizer, T_max=20, eta_min=1e-6)

        print("Starting phase 1 training (partial unfreeze)...")
        start_time = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
 ↪scheduler, num_epochs=20, device=device)
        phase1_time = time.time() - start_time

        print("Starting phase 2 training (full fine-tuning)...")
        # Unfreeze all parameters for final fine-tuning
        for param in model.parameters():
            param.requires_grad = True

        optimizer = torch.optim.AdamW(model.parameters(), lr=0.00001,␣
 ↪weight_decay=1e-5)
        scheduler = ReduceLROnPlateau(optimizer, mode='max', factor=0.5,␣
 ↪patience=3, verbose=True)

        start_time_phase2 = time.time()
        train_model(model, train_loader, val_loader, criterion, optimizer,␣
 ↪scheduler, num_epochs=15, device=device, warmup_epochs=0)
        total_time = phase1_time + (time.time() - start_time_phase2)

        print("Testing model...")
        test_preds, test_labels = test_model(model, test_loader, device)

        split_results = calculate_metrics(test_labels, test_preds, split_folder)
        split_results['training_time'] = total_time
        results.append(split_results)

        torch.cuda.empty_cache()

    results_df = pd.DataFrame(results)
    print(f"\n{'='*80}")
    print("DenseNet-121 - SUMMARY OF ALL SPLITS")
    print(f"{'='*80}")
    print(results_df.to_string(index=False))

    results_csv_path = "/kaggle/working/densenet121_results.csv"
    results_df.to_csv(results_csv_path, index=False)
    print(f"\nDetailed results saved to: {results_csv_path}")

    return results_df

if __name__ == "__main__":
    results = run_densenet121_on_splits()
```

```
Using device: cuda

============================================================
Processing: split_10_90
============================================================
Train samples: 464
Val samples: 51
Test samples: 4639
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0743, Train Acc: 43.97%, Val Acc: 45.10%, LR: 0.000033
Epoch [2/20], Loss: 1.0538, Train Acc: 48.71%, Val Acc: 49.02%, LR: 0.000067
Epoch [3/20], Loss: 1.0170, Train Acc: 49.57%, Val Acc: 54.90%, LR: 0.000100
Epoch [4/20], Loss: 0.9897, Train Acc: 53.45%, Val Acc: 58.82%, LR: 0.000099
Epoch [5/20], Loss: 0.9825, Train Acc: 54.74%, Val Acc: 60.78%, LR: 0.000098
Epoch [6/20], Loss: 0.9547, Train Acc: 57.33%, Val Acc: 56.86%, LR: 0.000095
Epoch [7/20], Loss: 0.9430, Train Acc: 57.54%, Val Acc: 60.78%, LR: 0.000091
Epoch [8/20], Loss: 0.9166, Train Acc: 62.72%, Val Acc: 62.75%, LR: 0.000086
Epoch [9/20], Loss: 0.8985, Train Acc: 63.58%, Val Acc: 58.82%, LR: 0.000080
Epoch [10/20], Loss: 0.8836, Train Acc: 64.44%, Val Acc: 60.78%, LR: 0.000073
Epoch [11/20], Loss: 0.8537, Train Acc: 66.38%, Val Acc: 60.78%, LR: 0.000066
Epoch [12/20], Loss: 0.8429, Train Acc: 67.24%, Val Acc: 60.78%, LR: 0.000058
Epoch [13/20], Loss: 0.8517, Train Acc: 64.87%, Val Acc: 60.78%, LR: 0.000051
Epoch [14/20], Loss: 0.8261, Train Acc: 69.61%, Val Acc: 60.78%, LR: 0.000043
Epoch [15/20], Loss: 0.8274, Train Acc: 69.18%, Val Acc: 64.71%, LR: 0.000035
Epoch [16/20], Loss: 0.8149, Train Acc: 69.83%, Val Acc: 62.75%, LR: 0.000028
Epoch [17/20], Loss: 0.8072, Train Acc: 70.69%, Val Acc: 66.67%, LR: 0.000021
Epoch [18/20], Loss: 0.8150, Train Acc: 68.75%, Val Acc: 62.75%, LR: 0.000015
Epoch [19/20], Loss: 0.7878, Train Acc: 73.71%, Val Acc: 64.71%, LR: 0.000010
Epoch [20/20], Loss: 0.7995, Train Acc: 71.34%, Val Acc: 66.67%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.7989, Train Acc: 70.26%, Val Acc: 64.71%, LR: 0.000010
Epoch [2/15], Loss: 0.7878, Train Acc: 72.20%, Val Acc: 66.67%, LR: 0.000010
Epoch [3/15], Loss: 0.7830, Train Acc: 72.20%, Val Acc: 64.71%, LR: 0.000010
Epoch [4/15], Loss: 0.7694, Train Acc: 74.14%, Val Acc: 66.67%, LR: 0.000010
Epoch [5/15], Loss: 0.7552, Train Acc: 75.86%, Val Acc: 64.71%, LR: 0.000010
Epoch [6/15], Loss: 0.7179, Train Acc: 78.02%, Val Acc: 66.67%, LR: 0.000005
Epoch [7/15], Loss: 0.7186, Train Acc: 78.45%, Val Acc: 64.71%, LR: 0.000005
Epoch [8/15], Loss: 0.7142, Train Acc: 76.51%, Val Acc: 64.71%, LR: 0.000005
Epoch [9/15], Loss: 0.7006, Train Acc: 79.74%, Val Acc: 62.75%, LR: 0.000005
Epoch [10/15], Loss: 0.6941, Train Acc: 77.80%, Val Acc: 62.75%, LR: 0.000003
Early stopping at epoch 10
Testing model…
```

```
=== split_10_90 Results ===
Accuracy: 0.5820
Precision: 0.5999
Recall: 0.5820
F1-Score: 0.5414

Classification Report:
              precision    recall  f1-score   support

          AD       0.50      0.28      0.36      1012
          CN       0.72      0.28      0.40      1296
         MCI       0.58      0.88      0.70      2331

    accuracy                           0.58      4639
   macro avg       0.60      0.48      0.49      4639
weighted avg       0.60      0.58      0.54      4639


============================================================
Processing: split_20_80
============================================================
Train samples: 928
Val samples: 103
Test samples: 4123
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.1661, Train Acc: 29.31%, Val Acc: 46.60%, LR: 0.000033
Epoch [2/20], Loss: 1.0541, Train Acc: 48.81%, Val Acc: 49.51%, LR: 0.000067
Epoch [3/20], Loss: 1.0195, Train Acc: 52.16%, Val Acc: 55.34%, LR: 0.000100
Epoch [4/20], Loss: 0.9949, Train Acc: 53.34%, Val Acc: 54.37%, LR: 0.000099
Epoch [5/20], Loss: 0.9750, Train Acc: 54.20%, Val Acc: 60.19%, LR: 0.000098
Epoch [6/20], Loss: 0.9554, Train Acc: 55.71%, Val Acc: 65.05%, LR: 0.000095
Epoch [7/20], Loss: 0.9225, Train Acc: 60.13%, Val Acc: 57.28%, LR: 0.000091
Epoch [8/20], Loss: 0.9095, Train Acc: 61.10%, Val Acc: 58.25%, LR: 0.000086
Epoch [9/20], Loss: 0.8998, Train Acc: 60.13%, Val Acc: 65.05%, LR: 0.000080
Epoch [10/20], Loss: 0.8857, Train Acc: 62.93%, Val Acc: 65.05%, LR: 0.000073
Epoch [11/20], Loss: 0.8636, Train Acc: 63.36%, Val Acc: 64.08%, LR: 0.000066
Epoch [12/20], Loss: 0.8444, Train Acc: 65.84%, Val Acc: 68.93%, LR: 0.000058
Epoch [13/20], Loss: 0.8476, Train Acc: 66.27%, Val Acc: 66.99%, LR: 0.000051
Epoch [14/20], Loss: 0.8270, Train Acc: 68.53%, Val Acc: 60.19%, LR: 0.000043
Epoch [15/20], Loss: 0.8186, Train Acc: 66.70%, Val Acc: 67.96%, LR: 0.000035
Epoch [16/20], Loss: 0.8111, Train Acc: 68.75%, Val Acc: 64.08%, LR: 0.000028
Epoch [17/20], Loss: 0.7918, Train Acc: 71.44%, Val Acc: 66.99%, LR: 0.000021
Epoch [18/20], Loss: 0.7931, Train Acc: 70.47%, Val Acc: 66.02%, LR: 0.000015
Epoch [19/20], Loss: 0.7957, Train Acc: 69.29%, Val Acc: 64.08%, LR: 0.000010
Epoch [20/20], Loss: 0.7771, Train Acc: 72.74%, Val Acc: 68.93%, LR: 0.000006
Early stopping at epoch 20
Starting phase 2 training (full fine-tuning)…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.7744, Train Acc: 72.84%, Val Acc: 66.99%, LR: 0.000010
Epoch [2/15], Loss: 0.7830, Train Acc: 71.98%, Val Acc: 72.82%, LR: 0.000010
Epoch [3/15], Loss: 0.7452, Train Acc: 73.92%, Val Acc: 68.93%, LR: 0.000010
Epoch [4/15], Loss: 0.7505, Train Acc: 72.95%, Val Acc: 74.76%, LR: 0.000010
Epoch [5/15], Loss: 0.7365, Train Acc: 73.71%, Val Acc: 75.73%, LR: 0.000010
Epoch [6/15], Loss: 0.7232, Train Acc: 74.89%, Val Acc: 73.79%, LR: 0.000010
Epoch [7/15], Loss: 0.7134, Train Acc: 77.05%, Val Acc: 72.82%, LR: 0.000010
Epoch [8/15], Loss: 0.6932, Train Acc: 76.94%, Val Acc: 76.70%, LR: 0.000010
Epoch [9/15], Loss: 0.6786, Train Acc: 78.45%, Val Acc: 75.73%, LR: 0.000010
Epoch [10/15], Loss: 0.6602, Train Acc: 81.47%, Val Acc: 76.70%, LR: 0.000010
Epoch [11/15], Loss: 0.6616, Train Acc: 81.14%, Val Acc: 79.61%, LR: 0.000010
Epoch [12/15], Loss: 0.6180, Train Acc: 83.30%, Val Acc: 76.70%, LR: 0.000010
Epoch [13/15], Loss: 0.6222, Train Acc: 83.08%, Val Acc: 78.64%, LR: 0.000010
Epoch [14/15], Loss: 0.5980, Train Acc: 85.13%, Val Acc: 72.82%, LR: 0.000010
Epoch [15/15], Loss: 0.6094, Train Acc: 84.16%, Val Acc: 72.82%, LR: 0.000005
Testing model…

=== split_20_80 Results ===
Accuracy: 0.6949
Precision: 0.6943
Recall: 0.6949
F1-Score: 0.6910

Classification Report:
              precision    recall  f1-score   support

          AD       0.57      0.55      0.56       899
          CN       0.74      0.59      0.66      1152
         MCI       0.72      0.82      0.77      2072

    accuracy                           0.69      4123
   macro avg       0.68      0.65      0.66      4123
weighted avg       0.69      0.69      0.69      4123


============================================================
Processing: split_30_70
============================================================
Train samples: 1391
Val samples: 155
Test samples: 3608
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0815, Train Acc: 43.49%, Val Acc: 49.03%, LR: 0.000033
Epoch [2/20], Loss: 1.0262, Train Acc: 53.13%, Val Acc: 53.55%, LR: 0.000067
Epoch [3/20], Loss: 0.9904, Train Acc: 54.06%, Val Acc: 54.84%, LR: 0.000100
Epoch [4/20], Loss: 0.9739, Train Acc: 55.00%, Val Acc: 58.71%, LR: 0.000099
Epoch [5/20], Loss: 0.9406, Train Acc: 57.66%, Val Acc: 58.06%, LR: 0.000098
Epoch [6/20], Loss: 0.9186, Train Acc: 61.61%, Val Acc: 60.00%, LR: 0.000095
Epoch [7/20], Loss: 0.8901, Train Acc: 61.75%, Val Acc: 62.58%, LR: 0.000091
Epoch [8/20], Loss: 0.8802, Train Acc: 63.26%, Val Acc: 63.87%, LR: 0.000086
Epoch [9/20], Loss: 0.8452, Train Acc: 66.00%, Val Acc: 64.52%, LR: 0.000080
Epoch [10/20], Loss: 0.8278, Train Acc: 67.58%, Val Acc: 69.03%, LR: 0.000073
Epoch [11/20], Loss: 0.8268, Train Acc: 67.36%, Val Acc: 70.97%, LR: 0.000066
Epoch [12/20], Loss: 0.8130, Train Acc: 68.73%, Val Acc: 65.81%, LR: 0.000058
Epoch [13/20], Loss: 0.8013, Train Acc: 68.51%, Val Acc: 67.10%, LR: 0.000051
Epoch [14/20], Loss: 0.7772, Train Acc: 70.17%, Val Acc: 67.10%, LR: 0.000043
Epoch [15/20], Loss: 0.7629, Train Acc: 71.89%, Val Acc: 63.87%, LR: 0.000035
Epoch [16/20], Loss: 0.7539, Train Acc: 73.11%, Val Acc: 65.81%, LR: 0.000028
Epoch [17/20], Loss: 0.7453, Train Acc: 73.40%, Val Acc: 63.23%, LR: 0.000021
Epoch [18/20], Loss: 0.7427, Train Acc: 73.62%, Val Acc: 67.74%, LR: 0.000015
Epoch [19/20], Loss: 0.7339, Train Acc: 73.69%, Val Acc: 66.45%, LR: 0.000010
Early stopping at epoch 19
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.7599, Train Acc: 73.11%, Val Acc: 67.10%, LR: 0.000010
Epoch [2/15], Loss: 0.7198, Train Acc: 75.13%, Val Acc: 68.39%, LR: 0.000010
Epoch [3/15], Loss: 0.7052, Train Acc: 75.70%, Val Acc: 68.39%, LR: 0.000010
Epoch [4/15], Loss: 0.6923, Train Acc: 77.35%, Val Acc: 69.68%, LR: 0.000010
Epoch [5/15], Loss: 0.6868, Train Acc: 77.86%, Val Acc: 72.26%, LR: 0.000010
Epoch [6/15], Loss: 0.6711, Train Acc: 79.51%, Val Acc: 67.74%, LR: 0.000010
Epoch [7/15], Loss: 0.6517, Train Acc: 81.24%, Val Acc: 67.10%, LR: 0.000010
Epoch [8/15], Loss: 0.6533, Train Acc: 79.65%, Val Acc: 73.55%, LR: 0.000010
Epoch [9/15], Loss: 0.6434, Train Acc: 80.52%, Val Acc: 68.39%, LR: 0.000010
Epoch [10/15], Loss: 0.6178, Train Acc: 82.60%, Val Acc: 71.61%, LR: 0.000010
Epoch [11/15], Loss: 0.6067, Train Acc: 83.11%, Val Acc: 74.19%, LR: 0.000010
Epoch [12/15], Loss: 0.5929, Train Acc: 84.47%, Val Acc: 73.55%, LR: 0.000010
Epoch [13/15], Loss: 0.5971, Train Acc: 82.96%, Val Acc: 75.48%, LR: 0.000010
Epoch [14/15], Loss: 0.5995, Train Acc: 82.60%, Val Acc: 74.19%, LR: 0.000010
Epoch [15/15], Loss: 0.5749, Train Acc: 85.12%, Val Acc: 74.84%, LR: 0.000010
Testing model…

=== split_30_70 Results ===
Accuracy: 0.7589
Precision: 0.7567
Recall: 0.7589
F1-Score: 0.7572
```

```
Classification Report:
              precision    recall  f1-score   support

          AD       0.69      0.62      0.66       787
          CN       0.73      0.74      0.74      1008
         MCI       0.80      0.83      0.81      1813

    accuracy                           0.76      3608
   macro avg       0.74      0.73      0.74      3608
weighted avg       0.76      0.76      0.76      3608


============================================================
Processing: split_40_60
============================================================
Train samples: 1855
Val samples: 207
Test samples: 3092
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0465, Train Acc: 49.06%, Val Acc: 50.72%, LR: 0.000033
Epoch [2/20], Loss: 1.0159, Train Acc: 52.40%, Val Acc: 54.11%, LR: 0.000067
Epoch [3/20], Loss: 0.9818, Train Acc: 55.26%, Val Acc: 55.56%, LR: 0.000100
Epoch [4/20], Loss: 0.9465, Train Acc: 58.27%, Val Acc: 58.45%, LR: 0.000099
Epoch [5/20], Loss: 0.9204, Train Acc: 61.67%, Val Acc: 57.97%, LR: 0.000098
Epoch [6/20], Loss: 0.8865, Train Acc: 63.72%, Val Acc: 63.77%, LR: 0.000095
Epoch [7/20], Loss: 0.8647, Train Acc: 64.91%, Val Acc: 64.73%, LR: 0.000091
Epoch [8/20], Loss: 0.8320, Train Acc: 67.33%, Val Acc: 62.32%, LR: 0.000086
Epoch [9/20], Loss: 0.8188, Train Acc: 67.17%, Val Acc: 64.25%, LR: 0.000080
Epoch [10/20], Loss: 0.8056, Train Acc: 70.51%, Val Acc: 62.80%, LR: 0.000073
Epoch [11/20], Loss: 0.7829, Train Acc: 71.48%, Val Acc: 60.87%, LR: 0.000066
Epoch [12/20], Loss: 0.7635, Train Acc: 72.35%, Val Acc: 68.12%, LR: 0.000058
Epoch [13/20], Loss: 0.7400, Train Acc: 74.82%, Val Acc: 65.70%, LR: 0.000051
Epoch [14/20], Loss: 0.7355, Train Acc: 74.23%, Val Acc: 67.63%, LR: 0.000043
Epoch [15/20], Loss: 0.7291, Train Acc: 74.88%, Val Acc: 65.22%, LR: 0.000035
Epoch [16/20], Loss: 0.7127, Train Acc: 75.90%, Val Acc: 67.15%, LR: 0.000028
Epoch [17/20], Loss: 0.7118, Train Acc: 75.26%, Val Acc: 71.01%, LR: 0.000021
Epoch [18/20], Loss: 0.7052, Train Acc: 76.98%, Val Acc: 68.60%, LR: 0.000015
Epoch [19/20], Loss: 0.7066, Train Acc: 75.31%, Val Acc: 67.63%, LR: 0.000010
Epoch [20/20], Loss: 0.6898, Train Acc: 76.66%, Val Acc: 70.05%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6922, Train Acc: 77.47%, Val Acc: 68.60%, LR: 0.000010
Epoch [2/15], Loss: 0.6847, Train Acc: 77.14%, Val Acc: 71.01%, LR: 0.000010
Epoch [3/15], Loss: 0.6681, Train Acc: 78.44%, Val Acc: 69.57%, LR: 0.000010
Epoch [4/15], Loss: 0.6678, Train Acc: 78.11%, Val Acc: 69.08%, LR: 0.000010
Epoch [5/15], Loss: 0.6351, Train Acc: 81.67%, Val Acc: 74.88%, LR: 0.000010
Epoch [6/15], Loss: 0.6343, Train Acc: 81.35%, Val Acc: 72.95%, LR: 0.000010
Epoch [7/15], Loss: 0.6118, Train Acc: 82.86%, Val Acc: 72.95%, LR: 0.000010
Epoch [8/15], Loss: 0.6041, Train Acc: 83.50%, Val Acc: 75.85%, LR: 0.000010
Epoch [9/15], Loss: 0.5956, Train Acc: 83.29%, Val Acc: 75.36%, LR: 0.000010
Epoch [10/15], Loss: 0.5676, Train Acc: 85.07%, Val Acc: 76.33%, LR: 0.000010
Epoch [11/15], Loss: 0.5663, Train Acc: 85.23%, Val Acc: 76.33%, LR: 0.000010
Epoch [12/15], Loss: 0.5450, Train Acc: 87.01%, Val Acc: 75.85%, LR: 0.000010
Epoch [13/15], Loss: 0.5445, Train Acc: 86.85%, Val Acc: 78.74%, LR: 0.000010
Epoch [14/15], Loss: 0.5397, Train Acc: 87.60%, Val Acc: 77.78%, LR: 0.000010
Epoch [15/15], Loss: 0.5275, Train Acc: 87.71%, Val Acc: 78.74%, LR: 0.000010
Testing model…

=== split_40_60 Results ===
Accuracy: 0.7791
Precision: 0.7799
Recall: 0.7791
F1-Score: 0.7740

Classification Report:
              precision    recall  f1-score   support

          AD       0.80      0.61      0.69       674
          CN       0.77      0.69      0.73       864
         MCI       0.78      0.90      0.83      1554

    accuracy                           0.78      3092
   macro avg       0.78      0.73      0.75      3092
weighted avg       0.78      0.78      0.77      3092


============================================================
Processing: split_50_50
============================================================
Train samples: 2319
Val samples: 258
Test samples: 2577
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0504, Train Acc: 46.53%, Val Acc: 48.45%, LR: 0.000033
Epoch [2/20], Loss: 1.0066, Train Acc: 52.82%, Val Acc: 55.04%, LR: 0.000067
Epoch [3/20], Loss: 0.9709, Train Acc: 56.19%, Val Acc: 52.33%, LR: 0.000100
Epoch [4/20], Loss: 0.9278, Train Acc: 59.68%, Val Acc: 57.36%, LR: 0.000099
Epoch [5/20], Loss: 0.8943, Train Acc: 62.10%, Val Acc: 54.26%, LR: 0.000098
Epoch [6/20], Loss: 0.8647, Train Acc: 64.64%, Val Acc: 60.47%, LR: 0.000095
Epoch [7/20], Loss: 0.8509, Train Acc: 65.07%, Val Acc: 59.30%, LR: 0.000091
Epoch [8/20], Loss: 0.8309, Train Acc: 66.62%, Val Acc: 62.40%, LR: 0.000086
Epoch [9/20], Loss: 0.8144, Train Acc: 68.05%, Val Acc: 64.34%, LR: 0.000080
Epoch [10/20], Loss: 0.8034, Train Acc: 68.56%, Val Acc: 66.67%, LR: 0.000073
Epoch [11/20], Loss: 0.7776, Train Acc: 71.24%, Val Acc: 63.57%, LR: 0.000066
Epoch [12/20], Loss: 0.7515, Train Acc: 73.39%, Val Acc: 69.77%, LR: 0.000058
Epoch [13/20], Loss: 0.7579, Train Acc: 71.54%, Val Acc: 67.44%, LR: 0.000051
Epoch [14/20], Loss: 0.7325, Train Acc: 74.43%, Val Acc: 67.83%, LR: 0.000043
Epoch [15/20], Loss: 0.7173, Train Acc: 75.33%, Val Acc: 69.77%, LR: 0.000035
Epoch [16/20], Loss: 0.7226, Train Acc: 74.82%, Val Acc: 67.05%, LR: 0.000028
Epoch [17/20], Loss: 0.7209, Train Acc: 74.64%, Val Acc: 68.99%, LR: 0.000021
Epoch [18/20], Loss: 0.7087, Train Acc: 75.55%, Val Acc: 71.32%, LR: 0.000015
Epoch [19/20], Loss: 0.6890, Train Acc: 77.10%, Val Acc: 71.32%, LR: 0.000010
Epoch [20/20], Loss: 0.6951, Train Acc: 75.55%, Val Acc: 71.71%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/15], Loss: 0.6993, Train Acc: 75.64%, Val Acc: 70.16%, LR: 0.000010
Epoch [2/15], Loss: 0.6647, Train Acc: 78.65%, Val Acc: 71.32%, LR: 0.000010
Epoch [3/15], Loss: 0.6515, Train Acc: 79.56%, Val Acc: 73.64%, LR: 0.000010
Epoch [4/15], Loss: 0.6437, Train Acc: 79.78%, Val Acc: 71.32%, LR: 0.000010
Epoch [5/15], Loss: 0.6396, Train Acc: 81.29%, Val Acc: 74.81%, LR: 0.000010
Epoch [6/15], Loss: 0.6082, Train Acc: 83.57%, Val Acc: 75.58%, LR: 0.000010
Epoch [7/15], Loss: 0.6049, Train Acc: 82.23%, Val Acc: 78.29%, LR: 0.000010
Epoch [8/15], Loss: 0.5906, Train Acc: 84.48%, Val Acc: 79.84%, LR: 0.000010
Epoch [9/15], Loss: 0.5713, Train Acc: 84.82%, Val Acc: 78.68%, LR: 0.000010
Epoch [10/15], Loss: 0.5581, Train Acc: 85.42%, Val Acc: 80.23%, LR: 0.000010
Epoch [11/15], Loss: 0.5435, Train Acc: 87.15%, Val Acc: 82.95%, LR: 0.000010
Epoch [12/15], Loss: 0.5404, Train Acc: 87.41%, Val Acc: 81.78%, LR: 0.000010
Epoch [13/15], Loss: 0.5387, Train Acc: 87.32%, Val Acc: 82.95%, LR: 0.000010
Epoch [14/15], Loss: 0.5334, Train Acc: 87.45%, Val Acc: 82.17%, LR: 0.000010
Epoch [15/15], Loss: 0.5237, Train Acc: 88.27%, Val Acc: 84.11%, LR: 0.000010
Testing model…

=== split_50_50 Results ===
Accuracy: 0.8184
Precision: 0.8176
Recall: 0.8184
F1-Score: 0.8168

Classification Report:
              precision    recall  f1-score   support

          AD       0.80      0.69      0.74       562
          CN       0.79      0.81      0.80       720
         MCI       0.84      0.88      0.86      1295

    accuracy                           0.82      2577
```

```
   macro avg       0.81      0.79      0.80      2577
weighted avg       0.82      0.82      0.82      2577


============================================================
Processing: split_60_40
============================================================
Train samples: 2784
Val samples: 308
Test samples: 2062
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0588, Train Acc: 47.13%, Val Acc: 51.62%, LR: 0.000033
Epoch [2/20], Loss: 1.0067, Train Acc: 52.44%, Val Acc: 52.60%, LR: 0.000067
Epoch [3/20], Loss: 0.9702, Train Acc: 56.43%, Val Acc: 52.92%, LR: 0.000100
Epoch [4/20], Loss: 0.9269, Train Acc: 59.66%, Val Acc: 59.42%, LR: 0.000099
Epoch [5/20], Loss: 0.8930, Train Acc: 61.85%, Val Acc: 58.12%, LR: 0.000098
Epoch [6/20], Loss: 0.8664, Train Acc: 63.90%, Val Acc: 62.99%, LR: 0.000095
Epoch [7/20], Loss: 0.8351, Train Acc: 66.81%, Val Acc: 61.04%, LR: 0.000091
Epoch [8/20], Loss: 0.8097, Train Acc: 68.32%, Val Acc: 63.96%, LR: 0.000086
Epoch [9/20], Loss: 0.7928, Train Acc: 69.97%, Val Acc: 64.61%, LR: 0.000080
Epoch [10/20], Loss: 0.7596, Train Acc: 70.87%, Val Acc: 63.96%, LR: 0.000073
Epoch [11/20], Loss: 0.7512, Train Acc: 72.49%, Val Acc: 67.21%, LR: 0.000066
Epoch [12/20], Loss: 0.7371, Train Acc: 73.85%, Val Acc: 67.21%, LR: 0.000058
Epoch [13/20], Loss: 0.7215, Train Acc: 75.14%, Val Acc: 66.56%, LR: 0.000051
Epoch [14/20], Loss: 0.7046, Train Acc: 75.90%, Val Acc: 67.86%, LR: 0.000043
Epoch [15/20], Loss: 0.6991, Train Acc: 76.29%, Val Acc: 69.81%, LR: 0.000035
Epoch [16/20], Loss: 0.6740, Train Acc: 78.45%, Val Acc: 69.81%, LR: 0.000028
Epoch [17/20], Loss: 0.6825, Train Acc: 76.69%, Val Acc: 71.10%, LR: 0.000021
Epoch [18/20], Loss: 0.6808, Train Acc: 77.12%, Val Acc: 70.13%, LR: 0.000015
Epoch [19/20], Loss: 0.6715, Train Acc: 77.77%, Val Acc: 71.43%, LR: 0.000010
Epoch [20/20], Loss: 0.6608, Train Acc: 79.06%, Val Acc: 72.73%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…
```

```
/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(
```

```
Epoch [1/15], Loss: 0.6749, Train Acc: 78.56%, Val Acc: 69.81%, LR: 0.000010
Epoch [2/15], Loss: 0.6590, Train Acc: 78.27%, Val Acc: 71.43%, LR: 0.000010
Epoch [3/15], Loss: 0.6376, Train Acc: 79.92%, Val Acc: 70.78%, LR: 0.000010
Epoch [4/15], Loss: 0.6136, Train Acc: 82.51%, Val Acc: 73.38%, LR: 0.000010
Epoch [5/15], Loss: 0.6050, Train Acc: 82.76%, Val Acc: 73.38%, LR: 0.000010
Epoch [6/15], Loss: 0.5847, Train Acc: 84.81%, Val Acc: 76.30%, LR: 0.000010
Epoch [7/15], Loss: 0.5904, Train Acc: 83.66%, Val Acc: 79.22%, LR: 0.000010
Epoch [8/15], Loss: 0.5762, Train Acc: 84.73%, Val Acc: 76.62%, LR: 0.000010
Epoch [9/15], Loss: 0.5534, Train Acc: 85.85%, Val Acc: 81.17%, LR: 0.000010
Epoch [10/15], Loss: 0.5486, Train Acc: 86.31%, Val Acc: 80.19%, LR: 0.000010
Epoch [11/15], Loss: 0.5342, Train Acc: 87.90%, Val Acc: 79.22%, LR: 0.000010
Epoch [12/15], Loss: 0.5138, Train Acc: 89.76%, Val Acc: 80.19%, LR: 0.000010
Epoch [13/15], Loss: 0.5190, Train Acc: 88.97%, Val Acc: 84.09%, LR: 0.000010
Epoch [14/15], Loss: 0.5004, Train Acc: 90.52%, Val Acc: 84.74%, LR: 0.000010
Epoch [15/15], Loss: 0.5046, Train Acc: 89.69%, Val Acc: 87.34%, LR: 0.000010
Testing model…

=== split_60_40 Results ===
Accuracy: 0.8657
Precision: 0.8658
Recall: 0.8657
F1-Score: 0.8646

Classification Report:
              precision    recall  f1-score   support

          AD       0.85      0.79      0.82       450
          CN       0.88      0.82      0.85       576
         MCI       0.86      0.93      0.89      1036

    accuracy                           0.87      2062
   macro avg       0.86      0.84      0.85      2062
weighted avg       0.87      0.87      0.86      2062


============================================================
Processing: split_70_30
============================================================
Train samples: 3247
Val samples: 361
Test samples: 1546
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0714, Train Acc: 46.04%, Val Acc: 50.42%, LR: 0.000033
Epoch [2/20], Loss: 1.0015, Train Acc: 53.06%, Val Acc: 54.29%, LR: 0.000067
Epoch [3/20], Loss: 0.9632, Train Acc: 56.36%, Val Acc: 61.50%, LR: 0.000100
Epoch [4/20], Loss: 0.9169, Train Acc: 60.92%, Val Acc: 64.82%, LR: 0.000099
Epoch [5/20], Loss: 0.8870, Train Acc: 62.77%, Val Acc: 67.31%, LR: 0.000098
Epoch [6/20], Loss: 0.8434, Train Acc: 67.35%, Val Acc: 70.36%, LR: 0.000095
Epoch [7/20], Loss: 0.8163, Train Acc: 67.48%, Val Acc: 65.37%, LR: 0.000091
Epoch [8/20], Loss: 0.7938, Train Acc: 69.82%, Val Acc: 69.25%, LR: 0.000086
Epoch [9/20], Loss: 0.7573, Train Acc: 72.13%, Val Acc: 68.42%, LR: 0.000080
Epoch [10/20], Loss: 0.7399, Train Acc: 72.77%, Val Acc: 68.42%, LR: 0.000073
Epoch [11/20], Loss: 0.7327, Train Acc: 74.35%, Val Acc: 72.02%, LR: 0.000066
Epoch [12/20], Loss: 0.7078, Train Acc: 76.29%, Val Acc: 71.75%, LR: 0.000058
Epoch [13/20], Loss: 0.6998, Train Acc: 76.56%, Val Acc: 68.42%, LR: 0.000051
Epoch [14/20], Loss: 0.6928, Train Acc: 76.44%, Val Acc: 74.52%, LR: 0.000043
Epoch [15/20], Loss: 0.6935, Train Acc: 76.66%, Val Acc: 74.52%, LR: 0.000035
Epoch [16/20], Loss: 0.6674, Train Acc: 79.12%, Val Acc: 75.90%, LR: 0.000028
Epoch [17/20], Loss: 0.6784, Train Acc: 77.09%, Val Acc: 74.79%, LR: 0.000021
Epoch [18/20], Loss: 0.6527, Train Acc: 79.74%, Val Acc: 73.41%, LR: 0.000015
Epoch [19/20], Loss: 0.6602, Train Acc: 79.49%, Val Acc: 73.13%, LR: 0.000010
Epoch [20/20], Loss: 0.6565, Train Acc: 79.40%, Val Acc: 73.96%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6754, Train Acc: 77.98%, Val Acc: 74.79%, LR: 0.000010
Epoch [2/15], Loss: 0.6487, Train Acc: 79.15%, Val Acc: 76.18%, LR: 0.000010
Epoch [3/15], Loss: 0.6260, Train Acc: 81.24%, Val Acc: 78.12%, LR: 0.000010
Epoch [4/15], Loss: 0.6180, Train Acc: 81.92%, Val Acc: 79.50%, LR: 0.000010
Epoch [5/15], Loss: 0.5850, Train Acc: 83.95%, Val Acc: 81.44%, LR: 0.000010
Epoch [6/15], Loss: 0.5738, Train Acc: 85.06%, Val Acc: 79.78%, LR: 0.000010
Epoch [7/15], Loss: 0.5625, Train Acc: 85.59%, Val Acc: 82.55%, LR: 0.000010
Epoch [8/15], Loss: 0.5529, Train Acc: 86.08%, Val Acc: 83.38%, LR: 0.000010
Epoch [9/15], Loss: 0.5483, Train Acc: 86.70%, Val Acc: 82.83%, LR: 0.000010
Epoch [10/15], Loss: 0.5306, Train Acc: 88.08%, Val Acc: 82.83%, LR: 0.000010
Epoch [11/15], Loss: 0.5122, Train Acc: 89.28%, Val Acc: 86.15%, LR: 0.000010
Epoch [12/15], Loss: 0.5138, Train Acc: 88.82%, Val Acc: 86.43%, LR: 0.000010
Epoch [13/15], Loss: 0.4961, Train Acc: 89.87%, Val Acc: 87.26%, LR: 0.000010
Epoch [14/15], Loss: 0.5001, Train Acc: 89.31%, Val Acc: 87.53%, LR: 0.000010
Epoch [15/15], Loss: 0.4870, Train Acc: 91.01%, Val Acc: 90.58%, LR: 0.000010
Testing model…

=== split_70_30 Results ===
Accuracy: 0.8706
Precision: 0.8721
Recall: 0.8706
F1-Score: 0.8712

Classification Report:
              precision    recall  f1-score   support

          AD       0.78      0.82      0.80       337
          CN       0.86      0.87      0.86       432
         MCI       0.92      0.89      0.91       777

    accuracy                           0.87      1546
   macro avg       0.85      0.86      0.86      1546
weighted avg       0.87      0.87      0.87      1546


============================================================
Processing: split_80_20
============================================================
Train samples: 3711
```

```
Val samples: 412
Test samples: 1031
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0402, Train Acc: 49.02%, Val Acc: 51.46%, LR: 0.000033
Epoch [2/20], Loss: 0.9893, Train Acc: 53.84%, Val Acc: 55.10%, LR: 0.000067
Epoch [3/20], Loss: 0.9490, Train Acc: 57.83%, Val Acc: 59.22%, LR: 0.000100
Epoch [4/20], Loss: 0.9028, Train Acc: 62.68%, Val Acc: 63.35%, LR: 0.000099
Epoch [5/20], Loss: 0.8606, Train Acc: 64.38%, Val Acc: 66.02%, LR: 0.000098
Epoch [6/20], Loss: 0.8253, Train Acc: 67.12%, Val Acc: 64.32%, LR: 0.000095
Epoch [7/20], Loss: 0.7969, Train Acc: 69.74%, Val Acc: 63.83%, LR: 0.000091
Epoch [8/20], Loss: 0.7780, Train Acc: 70.52%, Val Acc: 66.99%, LR: 0.000086
Epoch [9/20], Loss: 0.7658, Train Acc: 71.71%, Val Acc: 71.36%, LR: 0.000080
Epoch [10/20], Loss: 0.7432, Train Acc: 73.86%, Val Acc: 69.66%, LR: 0.000073
Epoch [11/20], Loss: 0.7200, Train Acc: 74.67%, Val Acc: 73.30%, LR: 0.000066
Epoch [12/20], Loss: 0.7037, Train Acc: 75.75%, Val Acc: 72.33%, LR: 0.000058
Epoch [13/20], Loss: 0.6967, Train Acc: 76.23%, Val Acc: 75.24%, LR: 0.000051
Epoch [14/20], Loss: 0.6806, Train Acc: 78.01%, Val Acc: 76.21%, LR: 0.000043
Epoch [15/20], Loss: 0.6654, Train Acc: 79.12%, Val Acc: 74.76%, LR: 0.000035
Epoch [16/20], Loss: 0.6700, Train Acc: 79.09%, Val Acc: 75.97%, LR: 0.000028
Epoch [17/20], Loss: 0.6528, Train Acc: 79.66%, Val Acc: 77.18%, LR: 0.000021
Epoch [18/20], Loss: 0.6507, Train Acc: 79.98%, Val Acc: 76.70%, LR: 0.000015
Epoch [19/20], Loss: 0.6329, Train Acc: 81.38%, Val Acc: 77.67%, LR: 0.000010
Epoch [20/20], Loss: 0.6392, Train Acc: 80.44%, Val Acc: 77.18%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn(

Epoch [1/15], Loss: 0.6487, Train Acc: 79.76%, Val Acc: 71.84%, LR: 0.000010
Epoch [2/15], Loss: 0.6375, Train Acc: 80.68%, Val Acc: 77.91%, LR: 0.000010
Epoch [3/15], Loss: 0.6113, Train Acc: 82.81%, Val Acc: 77.67%, LR: 0.000010
Epoch [4/15], Loss: 0.5991, Train Acc: 83.37%, Val Acc: 79.61%, LR: 0.000010
Epoch [5/15], Loss: 0.5686, Train Acc: 85.53%, Val Acc: 79.85%, LR: 0.000010
Epoch [6/15], Loss: 0.5631, Train Acc: 85.61%, Val Acc: 81.31%, LR: 0.000010
Epoch [7/15], Loss: 0.5366, Train Acc: 88.06%, Val Acc: 80.58%, LR: 0.000010
Epoch [8/15], Loss: 0.5396, Train Acc: 86.93%, Val Acc: 85.44%, LR: 0.000010
Epoch [9/15], Loss: 0.5231, Train Acc: 88.04%, Val Acc: 84.47%, LR: 0.000010
Epoch [10/15], Loss: 0.5043, Train Acc: 89.63%, Val Acc: 83.25%, LR: 0.000010
Epoch [11/15], Loss: 0.4964, Train Acc: 89.98%, Val Acc: 86.17%, LR: 0.000010
Epoch [12/15], Loss: 0.4958, Train Acc: 90.06%, Val Acc: 85.68%, LR: 0.000010
Epoch [13/15], Loss: 0.4798, Train Acc: 91.32%, Val Acc: 87.38%, LR: 0.000010
Epoch [14/15], Loss: 0.4739, Train Acc: 91.75%, Val Acc: 88.35%, LR: 0.000010
Epoch [15/15], Loss: 0.4748, Train Acc: 91.46%, Val Acc: 88.35%, LR: 0.000010
Testing model…

=== split_80_20 Results ===
Accuracy: 0.9108
```

```
Precision: 0.9122
Recall: 0.9108
F1-Score: 0.9106

Classification Report:
              precision    recall  f1-score   support

          AD       0.85      0.86      0.86       225
          CN       0.96      0.87      0.91       288
         MCI       0.91      0.96      0.93       518

    accuracy                           0.91      1031
   macro avg       0.91      0.89      0.90      1031
weighted avg       0.91      0.91      0.91      1031


============================================================
Processing: split_90_10
============================================================
Train samples: 4175
Val samples: 464
Test samples: 515
Starting phase 1 training (partial unfreeze)…
Epoch [1/20], Loss: 1.0352, Train Acc: 50.97%, Val Acc: 53.88%, LR: 0.000033
Epoch [2/20], Loss: 0.9782, Train Acc: 54.23%, Val Acc: 59.27%, LR: 0.000067
Epoch [3/20], Loss: 0.9329, Train Acc: 59.31%, Val Acc: 61.42%, LR: 0.000100
Epoch [4/20], Loss: 0.8842, Train Acc: 62.80%, Val Acc: 61.64%, LR: 0.000099
Epoch [5/20], Loss: 0.8483, Train Acc: 65.87%, Val Acc: 63.58%, LR: 0.000098
Epoch [6/20], Loss: 0.8134, Train Acc: 68.50%, Val Acc: 65.73%, LR: 0.000095
Epoch [7/20], Loss: 0.7847, Train Acc: 70.04%, Val Acc: 70.04%, LR: 0.000091
Epoch [8/20], Loss: 0.7594, Train Acc: 72.22%, Val Acc: 71.98%, LR: 0.000086
Epoch [9/20], Loss: 0.7370, Train Acc: 74.04%, Val Acc: 69.83%, LR: 0.000080
Epoch [10/20], Loss: 0.7236, Train Acc: 74.32%, Val Acc: 72.20%, LR: 0.000073
Epoch [11/20], Loss: 0.7016, Train Acc: 76.38%, Val Acc: 76.08%, LR: 0.000066
Epoch [12/20], Loss: 0.6880, Train Acc: 76.69%, Val Acc: 72.41%, LR: 0.000058
Epoch [13/20], Loss: 0.6760, Train Acc: 77.99%, Val Acc: 75.43%, LR: 0.000051
Epoch [14/20], Loss: 0.6679, Train Acc: 78.73%, Val Acc: 74.57%, LR: 0.000043
Epoch [15/20], Loss: 0.6626, Train Acc: 78.78%, Val Acc: 77.37%, LR: 0.000035
Epoch [16/20], Loss: 0.6443, Train Acc: 79.64%, Val Acc: 76.08%, LR: 0.000028
Epoch [17/20], Loss: 0.6409, Train Acc: 80.14%, Val Acc: 77.37%, LR: 0.000021
Epoch [18/20], Loss: 0.6218, Train Acc: 82.42%, Val Acc: 80.17%, LR: 0.000015
Epoch [19/20], Loss: 0.6308, Train Acc: 81.51%, Val Acc: 80.17%, LR: 0.000010
Epoch [20/20], Loss: 0.6310, Train Acc: 81.05%, Val Acc: 78.66%, LR: 0.000006
Starting phase 2 training (full fine-tuning)…

/usr/local/lib/python3.11/dist-packages/torch/optim/lr_scheduler.py:62:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
```

```
  warnings.warn(
Epoch [1/15], Loss: 0.6338, Train Acc: 80.86%, Val Acc: 79.96%, LR: 0.000010
Epoch [2/15], Loss: 0.6126, Train Acc: 82.18%, Val Acc: 77.59%, LR: 0.000010
Epoch [3/15], Loss: 0.5917, Train Acc: 83.81%, Val Acc: 82.33%, LR: 0.000010
Epoch [4/15], Loss: 0.5748, Train Acc: 85.05%, Val Acc: 83.62%, LR: 0.000010
Epoch [5/15], Loss: 0.5602, Train Acc: 85.60%, Val Acc: 82.11%, LR: 0.000010
Epoch [6/15], Loss: 0.5351, Train Acc: 87.88%, Val Acc: 83.62%, LR: 0.000010
Epoch [7/15], Loss: 0.5376, Train Acc: 87.59%, Val Acc: 85.34%, LR: 0.000010
Epoch [8/15], Loss: 0.5165, Train Acc: 88.74%, Val Acc: 87.07%, LR: 0.000010
Epoch [9/15], Loss: 0.5101, Train Acc: 89.22%, Val Acc: 89.22%, LR: 0.000010
Epoch [10/15], Loss: 0.5036, Train Acc: 89.41%, Val Acc: 87.93%, LR: 0.000010
Epoch [11/15], Loss: 0.4931, Train Acc: 90.08%, Val Acc: 89.22%, LR: 0.000010
Epoch [12/15], Loss: 0.4762, Train Acc: 91.43%, Val Acc: 89.22%, LR: 0.000010
Epoch [13/15], Loss: 0.4777, Train Acc: 91.64%, Val Acc: 90.95%, LR: 0.000010
Epoch [14/15], Loss: 0.4621, Train Acc: 92.29%, Val Acc: 91.16%, LR: 0.000010
Epoch [15/15], Loss: 0.4552, Train Acc: 92.50%, Val Acc: 88.36%, LR: 0.000010
Testing model…

=== split_90_10 Results ===
Accuracy: 0.8854
Precision: 0.8857
Recall: 0.8854
F1-Score: 0.8849

Classification Report:
              precision    recall  f1-score   support

          AD       0.82      0.83      0.82       112
          CN       0.90      0.83      0.86       144
         MCI       0.91      0.94      0.92       259

    accuracy                           0.89       515
   macro avg       0.87      0.87      0.87       515
weighted avg       0.89      0.89      0.88       515


================================================================================
DenseNet-121 - SUMMARY OF ALL SPLITS
================================================================================
      split  accuracy  precision   recall  f1_score  training_time
split_10_90  0.582022   0.599896 0.582022  0.541413      90.557603
split_20_80  0.694882   0.694307 0.694882  0.690960     204.277958
split_30_70  0.758869   0.756701 0.758869  0.757176     293.513171
split_40_60  0.779107   0.779911 0.779107  0.774039     398.096600
split_50_50  0.818393   0.817609 0.818393  0.816841     497.732667
split_60_40  0.865664   0.865824 0.865664  0.864625     585.891587
split_70_30  0.870634   0.872084 0.870634  0.871198     688.507176
```

```
split_80_20  0.910766   0.912176 0.910766  0.910570     782.058805
split_90_10  0.885437   0.885662 0.885437  0.884912     878.856004

Detailed results saved to: /kaggle/working/densenet121_results.csv
```

[9]:
```python
import os
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
import pandas as pd
import numpy as np
from PIL import Image
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪accuracy_score, classification_report
import time
from torch.optim.lr_scheduler import CosineAnnealingLR, ReduceLROnPlateau

class AlzheimerDataset(Dataset):
    def __init__(self, split_dir, split_type, transform=None):
        self.split_dir = split_dir
        self.split_type = split_type
        self.transform = transform

        csv_path = os.path.join(split_dir, f"{split_type}.csv")
        self.df = pd.read_csv(csv_path)

        self.class_to_idx = {"AD": 0, "CN": 1, "MCI": 2}
        self.idx_to_class = {v: k for k, v in self.class_to_idx.items()}

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        row = self.df.iloc[idx]
        img_path = row['path']
        label = self.class_to_idx[row['class']]

        image = Image.open(img_path).convert('RGB')

        if self.transform:
            image = self.transform(image)

        return image, label

def get_data_transforms():
    train_transform = transforms.Compose([
        transforms.Resize((380, 380)),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomRotation(10),
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,␣
↪hue=0.1),
        transforms.RandomAffine(degrees=0, translate=(0.1, 0.1)),
        transforms.RandomGrayscale(p=0.1),
        transforms.GaussianBlur(kernel_size=3, sigma=(0.1, 2.0)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
↪225]),
        transforms.RandomErasing(p=0.2, scale=(0.02, 0.2), ratio=(0.3, 3.3))
    ])

    val_transform = transforms.Compose([
        transforms.Resize((380, 380)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.
↪225])
    ])

    return train_transform, val_transform

def train_model(model, train_loader, val_loader, criterion, optimizer,␣
↪scheduler, num_epochs, device, warmup_epochs=3):
    best_val_acc = 0
    patience = 8
    patience_counter = 0

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        correct = 0
        total = 0

        # Learning rate warmup
        if epoch < warmup_epochs:
            lr_scale = min(1.0, float(epoch + 1) / warmup_epochs)
            for param_group in optimizer.param_groups:
                param_group['lr'] = param_group['initial_lr'] * lr_scale

        for batch_idx, (images, labels) in enumerate(train_loader):
            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()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(train_loader)
        epoch_acc = 100 * correct / total

        val_acc = evaluate_model(model, val_loader, device)

        if scheduler and epoch >= warmup_epochs:
            if isinstance(scheduler, CosineAnnealingLR):
                scheduler.step()
            else:
                scheduler.step(val_acc)

        current_lr = optimizer.param_groups[0]['lr']
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Train␣
↪Acc: {
```

```
  File "/tmp/ipykernel_19/419255855.py", line 107
    print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}, Train Acc:
          ^
SyntaxError: unterminated string literal (detected at line 107)
```