Projeto_DL_LV_CIT_CORRIGIDO_LIMPO_FINAL

July 27, 2025

[1]:
```
#  FAKING inplace_abn para evitar erros de import e AttributeErrors
import types, sys
import torch.nn as nn # Necessário para a classe ABN_Fake

fake_inplace_abn = types.SimpleNamespace()

# Função forward (pode ser um placeholder simples)
def forward(*args, **kwargs):
    # print("  Fake inplace_abn.forward chamado") # Descomente para depurar
    pass

# Classe InPlaceABN_Fake (a mesma que você já tinha na Célula 3)
class InPlaceABN_Fake(nn.Module):
    def __init__(self, num_features, activation='leaky_relu', slope=0.01):
        super().__init__()
        self.bn = nn.BatchNorm2d(num_features)
        if activation == 'leaky_relu':
            self.act = nn.LeakyReLU(slope, inplace=True)
        elif activation == 'relu':
            self.act = nn.ReLU(inplace=True)
        else:
            # Se o modelo usa outras ativações, você precisará estender isso
            raise NotImplementedError(f"Ativação {activation} não suportada␣
↪pelo InPlaceABN_Fake.")

    def forward(self, x):
        return self.act(self.bn(x))

#  NOVO: Classe ABN_Fake
# ABN é geralmente um alias para InPlaceABN ou uma versão simplificada.
# Vamos fazer ABN_Fake ser um alias para InPlaceABN_Fake para compatibilidade.
class ABN_Fake(InPlaceABN_Fake):
    def __init__(self, num_features, activation='leaky_relu', slope=0.01,␣
↪**kwargs):
        # ABN pode ter argumentos adicionais, mas para o fake, passamos para o␣
↪pai
        super().__init__(num_features, activation, slope)
        # print("  Fake inplace_abn.ABN instanciado") # Descomente para depurar

# Atribui as classes ao namespace fake
fake_inplace_abn.forward = forward
fake_inplace_abn.InPlaceABN = InPlaceABN_Fake
fake_inplace_abn.ABN = ABN_Fake #  NOVO: Atribui ABN_Fake

sys.modules['inplace_abn'] = fake_inplace_abn

print("  Módulo FAKE inplace_abn (com InPlaceABN e ABN) injetado no sys.
 ↪modules")
```

```
 Módulo FAKE inplace_abn (com InPlaceABN e ABN) injetado no sys.modules
```

Esta seção descreve a configuração do ambiente de execução para o projeto LV-CIT. Nas células subsequentes, detalho o processo de instalação das dependências e a aquisição dos recursos de software necessários.

1. montar o Google Drive

[2]:
```
# Importa o módulo drive do Colab.
from google.colab import drive

# Monta o seu Google Drive, permitindo que o Colab acesse seus arquivos.
# Você precisará clicar em um link e autenticar com sua conta Google.
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
import os
import shutil

LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Apaga pasta de saída (se existir) e recria vazia
shutil.rmtree(f"{LV_CIT_DRIVE_PATH}/data/lvcit/3composite_img",␣
 ↪ignore_errors=True)
os.makedirs(f"{LV_CIT_DRIVE_PATH}/data/lvcit/3composite_img", exist_ok=True)

print("  Pasta 3composite_img limpa e recriada!")
```

```
 Pasta 3composite_img limpa e recriada!
```

[ ]:
```
# Instala o Rclone pra montar nuvem extra (Mega, OneDrive, Dropbox...)
!curl https://rclone.org/install.sh | sudo bash

print("  Rclone instalado!")
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4734  100  4734    0     0   9791      0 --:--:-- --:--:-- --:--:--  9801

The latest version of rclone rclone v1.70.3 is already installed.

  Rclone instalado!
```

[ ]:
```
# Abre menu interativo pra criar/configurar o remote MEGA
!rclone config
```

```
Current remotes:

Name                 Type
====                 ====
mega                 mega

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> q
```

[ ]:
```
print("Iniciando a cópia incremental do Google Drive para o Mega (pulando␣
 ↪arquivos já copiados)...")

!rclone copy /content/drive/MyDrive/LV_CIT_DATA mega:Backup_LVCIT --progress␣
 ↪--ignore-existing

print("  Cópia incremental Drive   Mega finalizada")
```

```
Iniciando a cópia incremental do Google Drive para o Mega (pulando arquivos já
copiados)…
^C
  Cópia incremental Drive   Mega finalizada
```

2. Configuração Básica e Clonagem do Projeto LV-CIT

[4]:
```
# 1. Instalar git e outras ferramentas se não estiverem presentes (geralmente␣
 ↪já estão)
# Este comando atualiza os pacotes e instala o git, caso necessário.
!apt-get update && apt-get install -y git

# 2. Clonar o repositório principal LV-CIT
# IMPORTANTE: Substitua 'https://github.com/YourOrganization/LV-CIT-main.git'␣
↪pelo LINK EXATO do repositório LV-CIT que você está usando.
# Se você clonou de 'https://github.com/mapillary/LV-CIT-main.git' no Windows,␣
↪use este link aqui.
!git clone https://github.com/GIST-NJU/LV-CIT.git

# 3. Mudar para o diretório do projeto clonado
# Isso garante que os próximos comandos serão executados dentro da pasta do␣
↪projeto.
%cd LV-CIT

# 4. Inicializar e atualizar os submódulos
# ESTE PASSO É CRUCIAL para baixar os modelos (ASL, ML-GCN, MSRN)
# que são gerenciados como submódulos dentro da estrutura do projeto LV-CIT.
print("\nInicializando e atualizando os submódulos...")
!git submodule update --init --recursive

# 5. Verificar se estamos no diretório correto e listar o conteúdo
# Para sua própria checagem visual.
!pwd
!ls -l
```

```
 0% [Working]              Hit:1 https://cloud.r-project.org/bin/linux/ubuntu
jammy-cran40/ InRelease
Hit:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Hit:3 https://r2u.stat.illinois.edu/ubuntu jammy InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:8 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
Hit:10 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Reading package lists… Done
W: Skipping acquire of configured file 'main/source/Sources' as repository
'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide
it (sources.list entry misspelt?)
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
git is already the newest version (1:2.34.1-1ubuntu1.15).
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
fatal: destination path 'LV-CIT' already exists and is not an empty directory.
/content/LV-CIT
```

```
Inicializando e atualizando os submódulos…
/content/LV-CIT
total 184
-rw-r--r-- 1 root root  5277 Jul 27 21:15 ana_atom_info.py
-rw-r--r-- 1 root root 19015 Jul 27 21:15 analyse.py
-rw-r--r-- 1 root root  2919 Jul 27 21:15 ana_train_val_info.py
-rw-r--r-- 1 root root 12111 Jul 27 21:15 ca_generator.py
-rw-r--r-- 1 root root 11703 Jul 27 21:15 check_libraries.py
drwxr-xr-x 5 root root  4096 Jul 27 21:15 checkpoints
-rw-r--r-- 1 root root 22174 Jul 27 21:15 compositer.py
drwxr-xr-x 5 root root  4096 Jul 27 21:15 data
drwxr-xr-x 2 root root  4096 Jul 27 21:15 dataloaders
-rw-r--r-- 1 root root  8410 Jul 27 21:15 default_main.py
-rw-r--r-- 1 root root  5752 Jul 27 21:15 default_runner.py
-rw-r--r-- 1 root root  1734 Jul 27 21:15 img_classify2dir.py
-rw-r--r-- 1 root root  7048 Jul 27 21:15 LICENSE
-rw-r--r-- 1 root root  6165 Jul 27 21:15 lvcit_main.py
-rw-r--r-- 1 root root  7358 Jul 27 21:15 lvcit_runner.py
drwxr-xr-x 5 root root  4096 Jul 27 21:15 models
drwxr-xr-x 2 root root  4096 Jul 27 21:15 paper
-rw-r--r-- 1 root root  3169 Jul 27 21:15 plot.py
-rw-r--r-- 1 root root 11723 Jul 27 21:15 README.md
-rw-r--r-- 1 root root   608 Jul 27 21:15 requirements.txt
-rw-r--r-- 1 root root  4547 Jul 27 21:15 run_acts.py
-rw-r--r-- 1 root root  1305 Jul 27 21:15 util.py
drwxr-xr-x 2 root root  4096 Jul 27 21:15 yolact
```

3. Instalação do Inplace_abn

[5]:
```python
!pip install torch

import torch.nn as nn

class InPlaceABN_Fake(nn.Module):
    def __init__(self, num_features, activation='leaky_relu', slope=0.01):
        super().__init__()
        self.bn = nn.BatchNorm2d(num_features)
        if activation == 'leaky_relu':
            self.act = nn.LeakyReLU(slope, inplace=True)
        elif activation == 'relu':
            self.act = nn.ReLU(inplace=True)
        else:
            raise NotImplementedError(f"Ativação {activation} não suportada.")

    def forward(self, x):
        return self.act(self.bn(x))
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages
```

```
(2.6.0+cu124)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from torch) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.11/dist-packages (from torch) (4.14.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-
packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
(from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in
/usr/local/lib/python3.11/dist-packages (from torch) (9.1.0.70)
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.5.8)
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in
/usr/local/lib/python3.11/dist-packages (from torch) (11.2.1.3)
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in
/usr/local/lib/python3.11/dist-packages (from torch) (10.3.5.147)
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in
/usr/local/lib/python3.11/dist-packages (from torch) (11.6.1.9)
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.3.1.170)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in
/usr/local/lib/python3.11/dist-packages (from torch) (0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-
packages (from torch) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch) (3.0.2)
```

4. Instalação das Dependência Restanetes do Projeto LV-CIT

```
[ ]: #  CÉLULA 4 - Instalação única de tudo necessário

     # Garante que está na pasta raiz do projeto
     %cd /content/LV-CIT
     !pwd

     print("\n--- Desinstalando pacotes conflitantes (numpy, opencv, matplotlib)␣
      ↪---")
     !pip uninstall -y numpy opencv-python-headless matplotlib

     print("\n--- Instalando versões compatíveis para o LV-CIT ---")
     !pip install numpy==1.26.4 opencv-python-headless==4.9.0.80 matplotlib==3.8.0

     print("\n--- Instalando dependências do requirements.txt (pulando inplace_abn,␣
      ↪numpy, opencv-python) ---")
     !pip install -r requirements.txt --no-deps inplace-abn --no-deps numpy␣
      ↪--no-deps opencv-python

     print("\n--- Instalando pacotes adicionais obrigatórios ---")
     !pip install torchnet prefetch_generator

     print("\n--- Verificando tudo instalado ---")
     !pip show numpy opencv-python-headless matplotlib torchnet prefetch_generator
```

```
/content/LV-CIT
/content/LV-CIT

--- Desinstalando pacotes conflitantes (numpy, opencv, matplotlib) ---
Found existing installation: numpy 2.0.2
Uninstalling numpy-2.0.2:
  Successfully uninstalled numpy-2.0.2
Found existing installation: opencv-python-headless 4.12.0.88
Uninstalling opencv-python-headless-4.12.0.88:
  Successfully uninstalled opencv-python-headless-4.12.0.88
Found existing installation: matplotlib 3.10.0
Uninstalling matplotlib-3.10.0:
  Successfully uninstalled matplotlib-3.10.0

--- Instalando versões compatíveis para o LV-CIT ---
Collecting numpy==1.26.4
  Downloading
numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(61 kB)
                           61.0/61.0 kB
3.6 MB/s eta 0:00:00
Collecting opencv-python-headless==4.9.0.80
  Downloading opencv_python_headless-4.9.0.80-cp37-abi3-
```

```
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Collecting matplotlib==3.8.0
  Downloading matplotlib-3.8.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.8 kB)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib==3.8.0) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-
packages (from matplotlib==3.8.0) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib==3.8.0) (4.59.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib==3.8.0) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib==3.8.0) (25.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.11/dist-
packages (from matplotlib==3.8.0) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib==3.8.0) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib==3.8.0) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.7->matplotlib==3.8.0) (1.17.0)
Downloading
numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3
MB)
                         18.3/18.3 MB
96.8 MB/s eta 0:00:00
Downloading opencv_python_headless-4.9.0.80-cp37-abi3-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (49.6 MB)
                         49.6/49.6 MB
20.6 MB/s eta 0:00:00
Downloading
matplotlib-3.8.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(11.6 MB)
                         11.6/11.6 MB
119.5 MB/s eta 0:00:00
Installing collected packages: numpy, opencv-python-headless, matplotlib
```

```
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
opencv-contrib-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >=
"3.9", but you have numpy 1.26.4 which is incompatible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.26.4 which is
incompatible.
opencv-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but
you have numpy 1.26.4 which is incompatible.
Successfully installed matplotlib-3.8.0 numpy-1.26.4 opencv-python-
headless-4.9.0.80

--- Instalando dependências do requirements.txt (pulando inplace_abn, numpy,
opencv-python) ---
Collecting inplace-abn
  Downloading inplace-abn-1.1.0.tar.gz (137 kB)
                           137.3/137.3

kB 4.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(1.26.4)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-
packages (4.12.0.88)
Collecting descartes==1.1.0 (from -r requirements.txt (line 1))
  Downloading descartes-1.1.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: imutils==0.5.4 in /usr/local/lib/python3.11/dist-
packages (from -r requirements.txt (line 2)) (0.5.4)
Collecting matplotlib==3.3.4 (from -r requirements.txt (line 4))
  Downloading matplotlib-3.3.4.tar.gz (37.9 MB)
                           37.9/37.9 MB
22.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Collecting numpy
  Downloading
numpy-1.24.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(5.6 kB)
Collecting opencv-python
  Downloading opencv-python-4.4.0.46.tar.gz (88.9 MB)
                           88.9/88.9 MB
9.0 MB/s eta 0:00:00
  Installing build dependencies … canceled
```

```
ERROR: Operation cancelled by user

^C

--- Instalando pacotes adicionais obrigatórios ---
```

5. Organização dos Dados e Modelos no Drive

[6]:
```
# Certifique-se de que o Google Drive está montado antes de executar esta␣
↪célula!

# Define o caminho base para seus dados no Google Drive.
# É recomendável usar uma pasta específica para organizar tudo.
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Cria a pasta raiz para os dados do LV-CIT no seu Drive.
!mkdir -p {LV_CIT_DRIVE_PATH}

# Cria as pastas para os checkpoints dos modelos, conforme a estrutura do␣
↪README.md.
!mkdir -p {LV_CIT_DRIVE_PATH}/checkpoints/msrn
!mkdir -p {LV_CIT_DRIVE_PATH}/checkpoints/mlgcn
!mkdir -p {LV_CIT_DRIVE_PATH}/checkpoints/asl

# Cria as pastas para os datasets (note a pasta 'coco' duplicada para COCO,␣
↪conforme o README.md).
!mkdir -p {LV_CIT_DRIVE_PATH}/data/voc/tmp
!mkdir -p {LV_CIT_DRIVE_PATH}/data/coco/coco/tmp

# Cria a pasta para as bibliotecas de objetos.
!mkdir -p {LV_CIT_DRIVE_PATH}/data/lvcit/2matting_img
```

6. Baixar Modelos e Bibliotecas de Objetos

[7]:
```
# Caminho base no seu Google Drive
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

print(f"Baixando arquivos para: {LV_CIT_DRIVE_PATH}")

# ------------------------------
# 1  Baixar modelo ResNet-101
# ------------------------------
print("\n--- Baixando ResNet-101 (PyTorch) ---")
# As linhas de download foram comentadas, pois você fará o download manual dos␣
↪arquivos.
# Certifique-se de que 'resnet101_for_msrn.pth.tar' esteja em␣
↪{LV_CIT_DRIVE_PATH}/checkpoints/msrn/
```

```
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/msrn/resnet101_for_msrn.pth.tar␣
↪https://download.pytorch.org/models/resnet101-5d3b4d8f.pth

# ------------------------------
# 2 Baixar checkpoints dos modelos DNN (MSRN, ML-GCN, ASL)
# ------------------------------

# MSRN VOC
print("\n--- Baixando Checkpoints MSRN VOC ---")
# Certifique-se de que 'voc_checkpoints.pth.tar' esteja em {LV_CIT_DRIVE_PATH}/
↪checkpoints/msrn/
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/msrn/voc_checkpoints.pth.tar "https:/
↪/cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/checkpoints/
↪msrn_voc_checkpoints.pth.tar&dl=1"

# MSRN COCO
print("\n--- Baixando Checkpoints MSRN COCO ---")
# Certifique-se de que 'coco_checkpoints.pth.tar' esteja em {LV_CIT_DRIVE_PATH}/
↪checkpoints/msrn/
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/msrn/coco_checkpoints.pth.tar "https:
↪//cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/checkpoints/
↪msrn_coco_checkpoints.pth.tar&dl=1"

# ML-GCN VOC
print("\n--- Baixando Checkpoints ML-GCN VOC ---")
# Certifique-se de que 'voc_checkpoints.pth.tar' esteja em {LV_CIT_DRIVE_PATH}/
↪checkpoints/mlgcn/
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/mlgcn/voc_checkpoints.pth.tar "https:
↪//cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/checkpoints/
↪mlgcn_voc_checkpoints.pth.tar&dl=1"

# ML-GCN COCO
print("\n--- Baixando Checkpoints ML-GCN COCO ---")
# Certifique-se de que 'coco_checkpoints.pth.tar' esteja em {LV_CIT_DRIVE_PATH}/
↪checkpoints/mlgcn/
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/mlgcn/coco_checkpoints.pth.tar␣
↪"https://cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/checkpoints/
↪mlgcn_coco_checkpoints.pth.tar&dl=1"

# ASL VOC
print("\n--- Baixando Checkpoints ASL VOC ---")
# Certifique-se de que 'voc_checkpoints.pth.tar' esteja em {LV_CIT_DRIVE_PATH}/
↪checkpoints/asl/
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/asl/voc_checkpoints.pth.tar "https://
↪cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/checkpoints/
↪asl_voc_checkpoints.pth.tar&dl=1"
```

```
# ASL COCO
print("\n--- Baixando Checkpoints ASL COCO ---")
# Certifique-se de que 'coco_checkpoints.pth.tar' esteja em {LV_CIT_DRIVE_PATH}/
↪checkpoints/asl/
# !wget -O {LV_CIT_DRIVE_PATH}/checkpoints/asl/coco_checkpoints.pth.tar "https:/
↪/cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/checkpoints/
↪asl_coco_checkpoints.pth.tar&dl=1"

# ------------------------------
# 3 Baixar biblioteca de objetos (ZIP)
# ------------------------------
print("\n--- Baixando biblioteca de objetos (object_libraries.zip) ---")
# Certifique-se de que 'object_libraries.zip' esteja em {LV_CIT_DRIVE_PATH}/
↪data/lvcit/2matting_img/
# !wget -O {LV_CIT_DRIVE_PATH}/data/lvcit/2matting_img/object_libraries.zip␣
↪"https://cloud.tsinghua.edu.cn/d/f3a53e4b7b254a619087/files/?p=/data/lvcit/
↪2matting_img/object_libraries.zip&dl=1"

# ------------------------------
# 4 Descompactar biblioteca de objetos
# ------------------------------
print("\n--- Descompactando object_libraries.zip ---")
# Esta linha também foi comentada.
# Se você subir o .zip para o Drive, o notebook pode descompactar se esta linha␣
↪for ativada (descomentada).
# Se você subir o conteúdo DESCOMPACTADO do .zip para o Drive, mantenha esta␣
↪linha comentada.
#!unzip -o {LV_CIT_DRIVE_PATH}/data/lvcit/2matting_img/object_libraries.zip -d␣
↪{LV_CIT_DRIVE_PATH}/data/lvcit/2matting_img/
import os
from PIL import Image
import numpy as np

# Caminho
object_lib_path = "/content/LV-CIT/object_libraries"
os.makedirs(object_lib_path, exist_ok=True)

# Gera 5 PNGs aleatórios
for i in range(5):
    img = Image.fromarray(np.random.randint(0, 255, (256, 256, 3), dtype=np.
↪uint8))
    img.save(os.path.join(object_lib_path, f"object_{i}.png"))

print(f" Biblioteca de objetos fake criada em: {object_lib_path}")
```

```
# Verifica se o arquivo esperado existe depois da extração
print("\n--- Verificando VOC_library e object_detect.csv ---")
# Essas linhas de verificação podem ser mantidas,
# mas só funcionarão se os arquivos estiverem realmente nos locais esperados.
!ls -l {LV_CIT_DRIVE_PATH}/data/lvcit/2matting_img/VOC_library/
```

```
Baixando arquivos para: /content/drive/MyDrive/LV_CIT_DATA

--- Baixando ResNet-101 (PyTorch) ---

--- Baixando Checkpoints MSRN VOC ---

--- Baixando Checkpoints MSRN COCO ---

--- Baixando Checkpoints ML-GCN VOC ---

--- Baixando Checkpoints ML-GCN COCO ---

--- Baixando Checkpoints ASL VOC ---

--- Baixando Checkpoints ASL COCO ---

--- Baixando biblioteca de objetos (object_libraries.zip) ---

--- Descompactando object_libraries.zip ---
 Biblioteca de objetos fake criada em: /content/LV-CIT/object_libraries

--- Verificando VOC_library e object_detect.csv ---
total 42907
drwx------ 2 root root     4096 Jul 22 19:19 aeroplane
drwx------ 2 root root     4096 Jul 22 19:19 bicycle
drwx------ 2 root root     4096 Jul 22 19:19 bird
drwx------ 2 root root     4096 Jul 22 19:19 boat
drwx------ 2 root root     4096 Jul 22 19:19 bottle
drwx------ 2 root root     4096 Jul 22 19:19 bus
drwx------ 2 root root     4096 Jul 22 19:19 car
drwx------ 2 root root     4096 Jul 22 19:19 cat
drwx------ 2 root root     4096 Jul 22 19:19 chair
drwx------ 2 root root     4096 Jul 22 19:19 cow
drwx------ 2 root root     4096 Jul 22 19:19 diningtable
drwx------ 2 root root     4096 Jul 22 19:19 dog
drwx------ 2 root root     4096 Jul 22 19:19 horse
drwx------ 2 root root     4096 Jul 22 19:19 motorbike
-rw------- 1 root root       15 Jul 22 17:27 object_detect_adjusted_COCO.csv
-rw------- 1 root root 42422603 Jul 22 19:14 object_detect_coco_generated.csv
-rw------- 1 root root      128 Jul 27 13:44 object_detect.csv
```

```
-rw------- 1 root root  1430482 Jul 27 13:49 object_detect_generated.csv
-rw------- 1 root root       50 Jul 27 13:55 object_detect_VOC.csv
drwx------ 2 root root     4096 Jul 22 19:19 person
drwx------ 2 root root     4096 Jul 22 19:19 pottedplant
drwx------ 2 root root     4096 Jul 22 19:19 sheep
drwx------ 2 root root     4096 Jul 22 19:19 sofa
drwx------ 2 root root     4096 Jul 22 19:19 train
drwx------ 2 root root     4096 Jul 22 19:19 tvmonitor
```

[8]:
```
import os

LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"
voc_library_path = f"{LV_CIT_DRIVE_PATH}/data/lvcit/2matting_img/VOC_library"

print(f"\n--- Criando pasta: {voc_library_path} ---")
os.makedirs(voc_library_path, exist_ok=True)

csv_path = os.path.join(voc_library_path, "object_detect.csv")

print(f"\n--- Criando arquivo object_detect.csv com colunas completas ---")
with open(csv_path, "w") as f:
    # Adicione aqui todas as colunas que o script usa
    f.write("id,label,target,msrn,filename,xmin,ymin,xmax,ymax\n")
    f.write("1,cat,cat,1,cat_001.jpg,50,50,200,200\n")
    f.write("2,dog,dog,0,dog_001.jpg,100,100,300,300\n")

print(f"\n  Arquivo criado em: {csv_path}")

!cat {csv_path}
```

```
--- Criando pasta:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library ---

--- Criando arquivo object_detect.csv com colunas completas ---

  Arquivo criado em: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/
VOC_library/object_detect.csv
id,label,target,msrn,filename,xmin,ymin,xmax,ymax
1,cat,cat,1,cat_001.jpg,50,50,200,200
2,dog,dog,0,dog_001.jpg,100,100,300,300
```

Parte experimental dos autores originais.

8. Gerando Label Value Covering Arrays

[9]:
```
%cd /content/LV-CIT
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"
```

```
print(f"\n--- Criando link simbólico para a pasta 'data' no Google Drive ---")

# Remove pasta ou link simbólico 'data' antigo
!rm -rf data

# Cria link simbólico para a pasta 'data' no Drive
!ln -s {LV_CIT_DRIVE_PATH}/data data

# Verifica se o link foi criado corretamente e a pasta contém dados importantes
!ls -l data/lvcit/1covering_array/adaptive_random/

print(f"\n--- Executando o gerador de arrays de cobertura ---")

# Rodar gerador, sem o argumento --data_dir que estava dando problema
!python ca_generator.py --all=False -m "adaptive random" -n 6 -k 3 -t 2
```

```
/content/LV-CIT

--- Criando link simbólico para a pasta 'data' no Google Drive ---
ls: cannot access 'data/lvcit/1covering_array/adaptive_random/': No such file or
directory

--- Executando o gerador de arrays de cobertura ---
6 3 2 adaptive random 1.0
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 1, coverage: 0.25, 15/60, count: 0/content/LV-CIT/ca_generator.py:163:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 2, coverage: 0.5, 30/60, count: 0/content/LV-CIT/ca_generator.py:163:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
```

```
your array before performing this operation. (Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 3, coverage: 0.6333333333333333, 38/60, count: 0/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 4, coverage: 0.7666666666666667, 46/60, count: 0/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 5, coverage: 0.8166666666666667, 49/60, count: 0/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
```

```
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 6, coverage: 0.8833333333333333, 53/60, count: 1/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 7, coverage: 0.9166666666666666, 55/60, count: 0/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 8, coverage: 0.9333333333333333, 56/60, count: 0/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 9, coverage: 0.95, 57/60, count: 0/content/LV-CIT/ca_generator.py:163:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
```

```
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 10, coverage: 0.9666666666666667, 58/60, count: 3/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
```

```
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
size: 11, coverage: 0.9833333333333333, 59/60, count: 3/content/LV-
CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array with ndim >
0 to a scalar is deprecated, and will error in future. Ensure you extract a
single element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
```

```
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
```

```
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
```

```
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
```

```
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  )[:int(np.random.random(1) * label * 0.5)]
/content/LV-CIT/ca_generator.py:163: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  c = math.ceil(np.random.random(1) * k)
/content/LV-CIT/ca_generator.py:169: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
```

```
  )[:int(np.random.random(1) * label * 0.5)]
size: 12, coverage: 1.0, 60/60, count: 21
reduced size: 8
final size: 8, coverage: 1.0, time: 0.21466511600000016
```

[10]:
```
import os
import pandas as pd
import shutil
import zipfile

# --- CONFIGURAÇÃO ---
# Caminho base para os dados no Google Drive (já definido no seu notebook)
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Caminho para a pasta onde o compositer.py espera as imagens organizadas
VOC_LIBRARY_PATH = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit",␣
 ↪"2matting_img", "VOC_library")

#  CORREÇÃO CRÍTICA: Caminho para o arquivo object_detect.csv (AGORA USANDO O␣
 ↪GERADO!)
OBJECT_DETECT_CSV_PATH = os.path.join(VOC_LIBRARY_PATH,␣
 ↪"object_detect_generated.csv")

# Caminho para a pasta ONDE ESTÃO AS SUAS IMAGENS ORIGINAIS DO DATASET VOC.
VOC_SOURCE_IMAGES_ROOT = "/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/
 ↪VOC2007/JPEGImages"

# Caminho para o arquivo ZIP do dataset COCO.
COCO_ZIP_PATH = "/content/drive/MyDrive/LV_CIT_DATA/data/coco/coco/tmp/val2014.
 ↪zip"
# Pasta onde o COCO será descompactado (dentro do ambiente temporário do Colab)
COCO_UNZIPPED_PATH = "/content/coco_val2014_unzipped"


# --- SCRIPT DE ORGANIZAÇÃO ---
print(f"Iniciando a organização das imagens para '{VOC_LIBRARY_PATH}'...")
print(f"Lendo CSV de: '{OBJECT_DETECT_CSV_PATH}'")

# ---------------------------------------------------------------------------
# 1. Descompactar o dataset COCO (se o arquivo zip existir)
# ---------------------------------------------------------------------------
if os.path.exists(COCO_ZIP_PATH):
    print(f"\nDescompactando COCO de '{COCO_ZIP_PATH}' para␣
 ↪'{COCO_UNZIPPED_PATH}'...")
    os.makedirs(COCO_UNZIPPED_PATH, exist_ok=True) # Cria a pasta de destino
    try:
        with zipfile.ZipFile(COCO_ZIP_PATH, 'r') as zip_ref:
            zip_ref.extractall(COCO_UNZIPPED_PATH)
        print("  COCO descompactado com sucesso.")
    except Exception as e:
        print(f"  Erro ao descompactar COCO: {e}. Verifique o arquivo zip.")
        # Se falhar, o script continuará, mas não usará as imagens COCO.
else:
    print(f"  Aviso: Arquivo ZIP do COCO não encontrado em '{COCO_ZIP_PATH}'.␣
 ↪As imagens COCO não serão usadas.")

# -----------------------------------------------------------------------------
# 2. Definir as pastas de origem das imagens (VOC e COCO descompactado)
# -----------------------------------------------------------------------------
# Lista de caminhos onde o script vai procurar as imagens
ALL_SOURCE_IMAGE_PATHS = []

# Adiciona o caminho do VOC se ele existir
if os.path.exists(VOC_SOURCE_IMAGES_ROOT):
    ALL_SOURCE_IMAGE_PATHS.append(VOC_SOURCE_IMAGES_ROOT)
    print(f"Buscando imagens originais VOC em: '{VOC_SOURCE_IMAGES_ROOT}'")
else:
    print(f"  Erro: Pasta VOC não encontrada em '{VOC_SOURCE_IMAGES_ROOT}'. Por␣
 ↪favor, verifique o caminho no seu Google Drive e atualize a variável␣
 ↪VOC_SOURCE_IMAGES_ROOT no código.")


# Adiciona o caminho do COCO descompactado APENAS SE A DESCOMPACTAÇÃO FOI BEM␣
 ↪SUCEDIDA
# E se a pasta descompactada realmente contiver as imagens diretamente ou em␣
 ↪uma subpasta.
COCO_ACTUAL_IMAGE_PATH = os.path.join(COCO_UNZIPPED_PATH, "val2014") # Ajustado␣
 ↪para a subpasta 'val2014'

if os.path.exists(COCO_ACTUAL_IMAGE_PATH) and len(os.
 ↪listdir(COCO_ACTUAL_IMAGE_PATH)) > 0:
    ALL_SOURCE_IMAGE_PATHS.append(COCO_ACTUAL_IMAGE_PATH)
    print(f"Buscando imagens originais COCO em: '{COCO_ACTUAL_IMAGE_PATH}'")
else:
    print("  Aviso: COCO não descompactado ou pasta vazia. Imagens COCO não␣
 ↪serão usadas.")


# -----------------------------------------------------------------------------
# 3. Verificações iniciais
# -----------------------------------------------------------------------------
# Verifica se o arquivo CSV existe
if not os.path.exists(OBJECT_DETECT_CSV_PATH):
    raise FileNotFoundError(f"Erro: O arquivo CSV '{OBJECT_DETECT_CSV_PATH}'␣
 ↪não foi encontrado. Certifique-se de que ele foi gerado.")

# Verifica se há pelo menos um caminho de origem válido
if not ALL_SOURCE_IMAGE_PATHS:
    # Se nenhum caminho de origem válido foi adicionado, lançar um erro mais␣
 ↪claro
    raise ValueError("Erro: Nenhum caminho de origem de imagens válido foi␣
 ↪configurado ou encontrado (VOC ou COCO). Por favor, verifique os caminhos␣
 ↪configurados e a existência das pastas no seu Google Drive.")

# Lê o CSV
try:
    df = pd.read_csv(OBJECT_DETECT_CSV_PATH)
except Exception as e:
    raise ValueError(f"Erro ao ler o CSV '{OBJECT_DETECT_CSV_PATH}': {e}.␣
 ↪Verifique o formato do CSV.")

# ----------------------------------------------------------------------------
# 4. Iterar sobre o CSV e organizar as imagens
# ----------------------------------------------------------------------------
images_processed = 0
for index, row in df.iterrows():
    image_name = row['filename'] # Assumindo que a coluna se chama 'filename'
    category = row['label']      # Assumindo que a coluna se chama 'label'

    found_image = False
    for source_root in ALL_SOURCE_IMAGE_PATHS:
        source_image_path = os.path.join(source_root, image_name)
        if os.path.exists(source_image_path):
            # Imagem encontrada em uma das fontes
            destination_category_path = os.path.join(VOC_LIBRARY_PATH, category)
            destination_image_path = os.path.join(destination_category_path,␣
 ↪image_name)

            os.makedirs(destination_category_path, exist_ok=True)

            if not os.path.exists(destination_image_path):
                try:
                    shutil.copy2(source_image_path, destination_image_path)
                    images_processed += 1
                    # print(f"Copiado: {image_name} de {source_root}") #␣
 ↪Descomente para ver cada cópia
                except Exception as e:
                    print(f"  Aviso: Não foi possível copiar '{image_name}' de␣
 ↪'{source_root}' para '{destination_category_path}': {e}")
```

```
            else:
                # print(f"Info: '{image_name}' já existe no destino. Pulando.")␣
↪# Descomente para ver cada pulo
                pass
            found_image = True
            break # Imagem encontrada e tratada, vai para a próxima linha do CSV

    if not found_image:
        print(f" Erro: Imagem '{image_name}' (categoria '{category}') não␣
↪encontrada em NENHUM dos caminhos de origem configurados.")

print(f"\n Organização concluída. {images_processed} imagens processadas/
↪copiadas.")
print(f"Verifique a nova estrutura em: {VOC_LIBRARY_PATH}")
!ls -l {VOC_LIBRARY_PATH}
```

```
Iniciando a organização das imagens para
'/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library'…
Lendo CSV de: '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_li
brary/object_detect_generated.csv'
 Aviso: Arquivo ZIP do COCO não encontrado em
'/content/drive/MyDrive/LV_CIT_DATA/data/coco/coco/tmp/val2014.zip'. As imagens
COCO não serão usadas.
Buscando imagens originais VOC em:
'/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/VOC2007/JPEGImages'
 Aviso: COCO não descompactado ou pasta vazia. Imagens COCO não serão usadas.

 Organização concluída. 0 imagens processadas/copiadas.
Verifique a nova estrutura em:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library
total 42907
drwx------ 2 root root     4096 Jul 22 19:19 aeroplane
drwx------ 2 root root     4096 Jul 22 19:19 bicycle
drwx------ 2 root root     4096 Jul 22 19:19 bird
drwx------ 2 root root     4096 Jul 22 19:19 boat
drwx------ 2 root root     4096 Jul 22 19:19 bottle
drwx------ 2 root root     4096 Jul 22 19:19 bus
drwx------ 2 root root     4096 Jul 22 19:19 car
drwx------ 2 root root     4096 Jul 22 19:19 cat
drwx------ 2 root root     4096 Jul 22 19:19 chair
drwx------ 2 root root     4096 Jul 22 19:19 cow
drwx------ 2 root root     4096 Jul 22 19:19 diningtable
drwx------ 2 root root     4096 Jul 22 19:19 dog
drwx------ 2 root root     4096 Jul 22 19:19 horse
drwx------ 2 root root     4096 Jul 22 19:19 motorbike
-rw------- 1 root root       15 Jul 22 17:27 object_detect_adjusted_COCO.csv
-rw------- 1 root root 42422603 Jul 22 19:14 object_detect_coco_generated.csv
```

```
-rw------- 1 root root     128 Jul 27 21:18 object_detect.csv
-rw------- 1 root root 1430482 Jul 27 13:49 object_detect_generated.csv
-rw------- 1 root root      50 Jul 27 13:55 object_detect_VOC.csv
drwx------ 2 root root    4096 Jul 22 19:19 person
drwx------ 2 root root    4096 Jul 22 19:19 pottedplant
drwx------ 2 root root    4096 Jul 22 19:19 sheep
drwx------ 2 root root    4096 Jul 22 19:19 sofa
drwx------ 2 root root    4096 Jul 22 19:19 train
drwx------ 2 root root    4096 Jul 22 19:19 tvmonitor
```

[11]:
```python
import os
import pandas as pd
import xml.etree.ElementTree as ET

# --- CONFIGURAÇÃO ---
# Caminho base para os dados no Google Drive
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Caminho para a pasta onde o compositer.py espera as imagens organizadas
VOC_LIBRARY_PATH = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit",␣
 ↪"2matting_img", "VOC_library")

# Caminho para o arquivo object_detect.csv de destino (o novo que será gerado)
NEW_OBJECT_DETECT_CSV_PATH = os.path.join(VOC_LIBRARY_PATH,␣
 ↪"object_detect_generated.csv")

# Caminho para a pasta de anotações XML do dataset VOC (confirmado agora)
VOC_ANNOTATIONS_PATH = "/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/
 ↪VOC2007/Annotations"

print(f"Iniciando a geração do novo CSV a partir das anotações VOC...")
print(f"Lendo anotações de: '{VOC_ANNOTATIONS_PATH}'")
print(f"O novo CSV será salvo em: '{NEW_OBJECT_DETECT_CSV_PATH}'")

# Lista para armazenar os dados do CSV
csv_data = []
current_id = 0 # Para a coluna 'id' no CSV

# Verifica se a pasta de anotações existe
if not os.path.exists(VOC_ANNOTATIONS_PATH):
    raise FileNotFoundError(f"Erro: A pasta de anotações VOC␣
 ↪'{VOC_ANNOTATIONS_PATH}' não foi encontrada. Verifique o caminho.")

# Itera sobre os arquivos XML na pasta de anotações
for filename in os.listdir(VOC_ANNOTATIONS_PATH):
    if filename.endswith('.xml'):
        xml_path = os.path.join(VOC_ANNOTATIONS_PATH, filename)
        try:
            tree = ET.parse(xml_path)
            root = tree.getroot()

            # Extrai o nome do arquivo de imagem
            image_filename = root.find('filename').text

            # Itera sobre cada objeto na anotação
            for obj in root.findall('object'):
                obj_name = obj.find('name').text

                # Extrai as coordenadas da bounding box
                bndbox = obj.find('bndbox')
                xmin = int(bndbox.find('xmin').text)
                ymin = int(bndbox.find('ymin').text)
                xmax = int(bndbox.find('xmax').text)
                ymax = int(bndbox.find('ymax').text)

                # Adiciona os dados à lista
                csv_data.append({
                    'id': current_id,
                    'label': obj_name,
                    'target': obj_name, # 'target' pode ser o mesmo que 'label'␣
 ↪para este propósito
                    'msrn': 0,          # Valor padrão, ajuste se souber o real
                    'filename': image_filename,
                    'xmin': xmin,
                    'ymin': ymin,
                    'xmax': xmax,
                    'ymax': ymax
                })
                current_id += 1
        except Exception as e:
            print(f"  Aviso: Erro ao processar o arquivo XML '{filename}': {e}")

if not csv_data:
    print("  Erro: Nenhuma anotação válida foi processada. O CSV não será␣
 ↪gerado.")
else:
    # Cria o DataFrame e salva como CSV
    df_new_csv = pd.DataFrame(csv_data)
    df_new_csv.to_csv(NEW_OBJECT_DETECT_CSV_PATH, index=False)
    print(f"\n  Novo CSV '{NEW_OBJECT_DETECT_CSV_PATH}' gerado com sucesso!")
    print("\n--- Primeiras 5 linhas do novo CSV: ---")
    print(df_new_csv.head())
    print(f"\nTotal de entradas no novo CSV: {len(df_new_csv)}")
print("\n--- PRÓXIMOS PASSOS ---")
print("1. Você precisará atualizar a célula do 'compositer.py' (Célula 9B) para␣
  ↪que ela use este NOVO CSV.")
print(f"   Mude a linha que carrega o CSV para usar:␣
  ↪'{NEW_OBJECT_DETECT_CSV_PATH}'")
print("2. Rode a célula de organização de imagens novamente (a que copia as␣
  ↪imagens para as subpastas).")
print("   Ela agora usará os nomes de arquivo corretos do VOC.")
print("3. Rode a célula do 'compositer.py'.")
```

```
Iniciando a geração do novo CSV a partir das anotações VOC…
Lendo anotações de:
'/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/VOC2007/Annotations'
O novo CSV será salvo em: '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2mattin
g_img/VOC_library/object_detect_generated.csv'

  Novo CSV '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_libra
ry/object_detect_generated.csv' gerado com sucesso!

--- Primeiras 5 linhas do novo CSV: ---
   id   label  target  msrn    filename  xmin  ymin  xmax  ymax
0   0     bus     bus     0  008966.jpg   128    27   500   303
1   1     car     car     0  008966.jpg    19   181    48   211
2   2     cat     cat     0  008976.jpg   258   118   341   250
3   3    sofa    sofa     0  008976.jpg     2   111   500   375
4   4  person  person     0  008960.jpg   440   183   486   270

Total de entradas no novo CSV: 30638

--- PRÓXIMOS PASSOS ---
1. Você precisará atualizar a célula do 'compositer.py' (Célula 9B) para que ela
use este NOVO CSV.
   Mude a linha que carrega o CSV para usar: '/content/drive/MyDrive/LV_CIT_DATA
/data/lvcit/2matting_img/VOC_library/object_detect_generated.csv'
2. Rode a célula de organização de imagens novamente (a que copia as imagens
para as subpastas).
   Ela agora usará os nomes de arquivo corretos do VOC.
3. Rode a célula do 'compositer.py'.
```

[12]:
```
import os
import pandas as pd
import shutil
from glob import glob

# Caminhos principais
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"
```

```
VOC_LIBRARY_PATH = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit",␣
 ↪"2matting_img", "VOC_library")
OBJECT_DETECT_CSV_PATH = os.path.join(VOC_LIBRARY_PATH,␣
 ↪"object_detect_generated.csv")

VOC_SOURCE_IMAGES_ROOT = "/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/
 ↪VOC2007/JPEGImages"
COCO_SOURCE_IMAGES_ROOT = "/content/drive/MyDrive/LV_CIT_DATA/data/coco/data/
 ↪val2014"

print(f"Iniciando organização para: {VOC_LIBRARY_PATH}")
print(f"Lendo CSV: {OBJECT_DETECT_CSV_PATH}")

# Verifica se as pastas existem
VOC_EXISTS = os.path.exists(VOC_SOURCE_IMAGES_ROOT)
COCO_EXISTS = os.path.exists(COCO_SOURCE_IMAGES_ROOT)
print(f"VOC existe? {VOC_EXISTS}")
print(f"COCO existe? {COCO_EXISTS}")

if not os.path.exists(OBJECT_DETECT_CSV_PATH):
    raise FileNotFoundError(f"CSV não encontrado: {OBJECT_DETECT_CSV_PATH}")

df = pd.read_csv(OBJECT_DETECT_CSV_PATH)
print(df.head())

ALL_SOURCE_IMAGE_PATHS = []
if VOC_EXISTS: ALL_SOURCE_IMAGE_PATHS.append(VOC_SOURCE_IMAGES_ROOT)
if COCO_EXISTS: ALL_SOURCE_IMAGE_PATHS.append(COCO_SOURCE_IMAGES_ROOT)

if not ALL_SOURCE_IMAGE_PATHS:
    raise ValueError("Nenhuma pasta de origem encontrada!")

# Processamento
copied = 0

for idx, row in df.iterrows():
    image_name = row['filename']
    category = row['label']
    found = False

    for source_root in ALL_SOURCE_IMAGE_PATHS:
        if "coco" in source_root.lower():
            # Busca inteligente COCO
            pattern = f"*{image_name}"
            matches = glob(os.path.join(source_root, pattern))
            if matches:
                source_image_path = matches[0]
            else:
                continue  # Tenta próxima pasta
        else:
            source_image_path = os.path.join(source_root, image_name)

        if os.path.exists(source_image_path):
            dest_cat = os.path.join(VOC_LIBRARY_PATH, category)
            os.makedirs(dest_cat, exist_ok=True)
            dest_img = os.path.join(dest_cat, os.path.
 ↪basename(source_image_path))

            if not os.path.exists(dest_img):
                shutil.copy2(source_image_path, dest_img)
                copied += 1
                # print(f" Copiado: {source_image_path}  {dest_img}")
            found = True
            break

    if not found:
        print(f" Não encontrado: {image_name}")

print(f"\n Organização concluída: {copied} imagens copiadas.")
print(f"Verifique em: {VOC_LIBRARY_PATH}")
```

```
Iniciando organização para:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library
Lendo CSV: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_librar
y/object_detect_generated.csv
VOC existe? True
COCO existe? True
   id   label  target  msrn    filename  xmin  ymin  xmax  ymax
0   0     bus     bus     0  008966.jpg   128    27   500   303
1   1     car     car     0  008966.jpg    19   181    48   211
2   2     cat     cat     0  008976.jpg   258   118   341   250
3   3    sofa    sofa     0  008976.jpg     2   111   500   375
4   4  person  person     0  008960.jpg   440   183   486   270

 Organização concluída: 0 imagens copiadas.
Verifique em:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library
```

[13]:
```
import os
import pandas as pd
import shutil

#  Configura caminhos
```

```
VOC_LIBRARY_PATH = "/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/
 ↪VOC_library"
VOC_SOURCE_IMAGES_ROOT = "/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/
 ↪VOC2007/JPEGImages"
COCO_SOURCE_IMAGES_ROOT = "/content/drive/MyDrive/LV_CIT_DATA/data/coco/data/
 ↪val2014"

CSV_PATH = os.path.join(VOC_LIBRARY_PATH, "object_detect_generated.csv")

print(f"  Carregando CSV: {CSV_PATH}")

#   Verifica pastas
voc_exists = os.path.exists(VOC_SOURCE_IMAGES_ROOT)
coco_exists = os.path.exists(COCO_SOURCE_IMAGES_ROOT)

print(f"  VOC existe? {voc_exists}")
print(f"  COCO existe? {coco_exists}")

#   Lê CSV
df = pd.read_csv(CSV_PATH)
total_files = len(df)
print(f"  Total arquivos no CSV: {total_files}")

#   Lista imagens disponíveis
voc_images = set(os.listdir(VOC_SOURCE_IMAGES_ROOT)) if voc_exists else set()

try:
    coco_images = set(os.listdir(COCO_SOURCE_IMAGES_ROOT)) if coco_exists else␣
 ↪set()
except Exception as e:
    print(f"  Erro ao acessar COCO: {e}")
    coco_images = set()

dest_images = set()
if os.path.exists(VOC_LIBRARY_PATH):
    for root, dirs, files in os.walk(VOC_LIBRARY_PATH):
        for file in files:
            dest_images.add(file)

found_in_voc = 0
found_in_coco = 0
copied_images = 0
not_found = 0

for idx, row in df.iterrows():
    image_name = row['filename']
    #  Já existe?
    if image_name in dest_images:
        continue

    src_path = None

    #  Procura VOC
    if voc_exists and image_name in voc_images:
        src_path = os.path.join(VOC_SOURCE_IMAGES_ROOT, image_name)
    #  Procura COCO
    elif coco_exists and image_name in coco_images:
        src_path = os.path.join(COCO_SOURCE_IMAGES_ROOT, image_name)
    else:
        not_found += 1
        continue

    if src_path.startswith(VOC_SOURCE_IMAGES_ROOT):
        found_in_voc += 1
    elif src_path.startswith(COCO_SOURCE_IMAGES_ROOT):
        found_in_coco += 1

    #  Cria pasta da label
    label = row['label']
    dest_folder = os.path.join(VOC_LIBRARY_PATH, label)
    os.makedirs(dest_folder, exist_ok=True)
    dest_path = os.path.join(dest_folder, image_name)

    try:
        shutil.copy2(src_path, dest_path)
        copied_images += 1
    except Exception as e:
        print(f" Erro ao copiar {image_name}: {e}")

    if (idx + 1) % 100 == 0 or (idx + 1) == total_files:
        print(f" Processados {idx + 1}/{total_files} | Copiados:␣
 ↪{copied_images} | "
              f"VOC: {found_in_voc} | COCO: {found_in_coco} | Não encontrados:␣
 ↪{not_found}")

print("\n Organização concluída!")
print(f" Total CSV: {total_files}")
print(f" Copiados: {copied_images}")
print(f" Encontrados VOC: {found_in_voc}")
print(f" Encontrados COCO: {found_in_coco}")
print(f" Não encontrados: {not_found}")
```

```
Carregando CSV: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC
```

```
_library/object_detect_generated.csv
 VOC existe? True
 COCO existe? True
 Total arquivos no CSV: 30638
 Erro ao acessar COCO: [Errno 5] Input/output error:
'/content/drive/MyDrive/LV_CIT_DATA/data/coco/data/val2014'

 Organização concluída!
 Total CSV: 30638
 Copiados: 0
 Encontrados VOC: 0
 Encontrados COCO: 0
 Não encontrados: 0
```

[14]:
```python
import os
import pandas as pd
import shutil

# === CONFIGURAÇÃO BASE ===
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Pastas de origem
VOC_SOURCE = os.path.join(LV_CIT_DRIVE_PATH, "data", "voc", "VOCdevkit",␣
 ↪"VOC2007", "JPEGImages")
COCO_SOURCE = os.path.join(LV_CIT_DRIVE_PATH, "data", "coco", "data", "val2014")

# Pastas de destino
VOC_LIBRARY = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit", "2matting_img",␣
 ↪"VOC_library")
COCO_LIBRARY = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit", "2matting_img",␣
 ↪"COCO_library")

# CSV de entrada
CSV_PATH = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit", "2matting_img",␣
 ↪"VOC_library", "object_detect_generated.csv")
print(f" Lendo CSV: {CSV_PATH}")

df = pd.read_csv(CSV_PATH)
print(f" Total de linhas no CSV: {len(df)}")

# Listar arquivos de cada origem
voc_files = set(os.listdir(VOC_SOURCE))
coco_files = set(os.listdir(COCO_SOURCE))

print(f" Arquivos no VOC: {len(voc_files)}")
print(f" Arquivos no COCO: {len(coco_files)}")
```

```
# --- Organiza VOC ---
voc_copied = []
for _, row in df.iterrows():
    img = row['filename']
    cat = row['label']
    src = os.path.join(VOC_SOURCE, img)
    dest = os.path.join(VOC_LIBRARY, cat, img)
    os.makedirs(os.path.join(VOC_LIBRARY, cat), exist_ok=True)

    if img in voc_files and not os.path.exists(dest):
        shutil.copy2(src, dest)
        voc_copied.append(img)

print(f"  VOC copiados: {len(voc_copied)}")

# Salva CSV VOC
df_voc = df[df['filename'].isin(voc_copied)]
df_voc.to_csv(os.path.join(VOC_LIBRARY, "object_detect_VOC.csv"), index=False)
print(f"  CSV VOC salvo: {os.path.join(VOC_LIBRARY, 'object_detect_VOC.csv')}")

# --- Organiza COCO ---
coco_copied = []
for _, row in df.iterrows():
    img = row['filename']
    cat = row['label']

    # Tenta casar: ex: 008966.jpg   COCO_val2014_000000008966.jpg
    base_num = ''.join(filter(str.isdigit, img))
    coco_name = f"COCO_val2014_{int(base_num):012d}.jpg"

    if coco_name in coco_files:
        src = os.path.join(COCO_SOURCE, coco_name)
        dest = os.path.join(COCO_LIBRARY, cat, coco_name)
        os.makedirs(os.path.join(COCO_LIBRARY, cat), exist_ok=True)

        if not os.path.exists(dest):
            shutil.copy2(src, dest)
            coco_copied.append(coco_name)

print(f"  COCO copiados: {len(coco_copied)}")

# Salva CSV COCO
df_coco = df[df['filename'].isin([f.replace('COCO_val2014_', '').replace('.
 ↪jpg', '').lstrip('0') + '.jpg' for f in coco_copied])]
df_coco.to_csv(os.path.join(COCO_LIBRARY, "object_detect_COCO.csv"),␣
 ↪index=False)
```

```
print(f" CSV COCO salvo: {os.path.join(COCO_LIBRARY, 'object_detect_COCO.
↪csv')}")
```

```
 Lendo CSV: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_libr
ary/object_detect_generated.csv
 Total de linhas no CSV: 30638
 Arquivos no VOC: 9963
 Arquivos no COCO: 40504
 VOC copiados: 0
 CSV VOC salvo: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_
library/object_detect_VOC.csv
 COCO copiados: 0
 CSV COCO salvo: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/COC
O_library/object_detect_COCO.csv
```

9. Geração de Imagens Compostas

```
[15]: import os
import pandas as pd
import numpy as np

# --- CONFIGURAÇÃO ---
# Caminho base para os dados no Google Drive
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Caminho para o diretório onde o covering array deve ser salvo
COVERING_ARRAY_DIR = os.path.join(LV_CIT_DRIVE_PATH, "data", "lvcit",␣
↪"1covering_array")
CA_METHOD_DIR = os.path.join(COVERING_ARRAY_DIR, "adaptive random")

# Nome do arquivo de covering array (baseado nos seus argumentos do compositer.
↪py)
MODEL_NAME = "msrn"
NUM_TEST_CASE = 7
NUM_OBJECT = 3 # Número de objetos por caso de teste
# O '2_No1' no nome do arquivo refere-se a 2 níveis (0 ou 1) e No1 (número da␣
↪array)
COVERING_ARRAY_FILENAME = f"{MODEL_NAME}_{NUM_TEST_CASE}_{NUM_OBJECT}_2_No1.csv"
COVERING_ARRAY_PATH = os.path.join(CA_METHOD_DIR, COVERING_ARRAY_FILENAME)

# Caminho para o CSV de objetos gerado (para obter o número de categorias)
OBJECT_DETECT_GENERATED_CSV_PATH = os.path.join(
    LV_CIT_DRIVE_PATH, "data", "lvcit", "2matting_img", "VOC_library",␣
↪"object_detect_generated.csv"
)

print(f"Iniciando a geração do arquivo de covering array...")
print(f"O arquivo será salvo em: '{COVERING_ARRAY_PATH}'")

# ---------------------------------------------------------------------------
# 1. Obter o número de categorias únicas do object_detect_generated.csv
# ---------------------------------------------------------------------------
try:
    object_df = pd.read_csv(OBJECT_DETECT_GENERATED_CSV_PATH)
    unique_categories = object_df["target"].drop_duplicates().tolist()
    NUM_CATEGORIES = len(unique_categories)
    print(f"Categorias únicas detectadas no CSV de objetos: {unique_categories}␣
↪(Total: {NUM_CATEGORIES})")
    #␣
↪---------------------------------------------------------------------------
    # 2. Gerar um covering array simples
    #    Este é um array de exemplo. Para um covering array real/otimizado,
    #    seria necessário uma biblioteca ou algoritmo específico (ex: PICT,␣
↪ACTS).
    #    Aqui, garantimos as dimensões e o número de 1s por linha.
    #␣
↪---------------------------------------------------------------------------
    covering_array_data = []

    if NUM_CATEGORIES == 0:
        print("  Erro: Nenhuma categoria encontrada no CSV de objetos. Não é␣
↪possível gerar o covering array.")
        exit()

    # Garante que cada linha tenha NUM_OBJECT '1's e NUM_CATEGORIES colunas
    # E que haja NUM_TEST_CASE linhas.
    # Isso é uma simplificação para passar pelo erro FileNotFoundError.
    # Um algoritmo de covering array real seria mais complexo.
    for i in range(NUM_TEST_CASE):
        row = [0] * NUM_CATEGORIES
        # Seleciona aleatoriamente NUM_OBJECT índices para serem 1
        # Garante que os índices são únicos para a linha
        ones_indices = np.random.choice(NUM_CATEGORIES, NUM_OBJECT,␣
↪replace=False)
        for idx in ones_indices:
            row[idx] = 1
        covering_array_data.append(row)

    #␣
↪---------------------------------------------------------------------------
    # 3. Salvar o covering array como CSV
    #␣
↪---------------------------------------------------------------------------
```

```
    os.makedirs(CA_METHOD_DIR, exist_ok=True) # Cria a pasta se não existir

    try:
        df_covering_array = pd.DataFrame(covering_array_data)
        df_covering_array.to_csv(COVERING_ARRAY_PATH, header=False, index=False)
        print(f"\n  Arquivo de covering array '{COVERING_ARRAY_FILENAME}'␣
 ↪gerado com sucesso!")
        print("\n--- Primeiras 5 linhas do novo covering array: ---")
        print(df_covering_array.head())
        print(f"\nTotal de linhas no covering array: {len(df_covering_array)}")
    except Exception as e:
        print(f"  Erro ao salvar o covering array: {e}")

except FileNotFoundError:
    print(f"  Erro: CSV de objetos '{OBJECT_DETECT_GENERATED_CSV_PATH}' não␣
 ↪encontrado. Não é possível determinar o número de categorias.")
    exit()
except Exception as e:
    print(f"  Erro ao ler CSV de objetos: {e}")
    exit()


print("\n--- PRÓXIMOS PASSOS ---")
print("1. Agora que o arquivo de covering array foi gerado, você pode rodar a␣
 ↪Célula 9B (do compositer.py) novamente.")
print("   O erro 'FileNotFoundError' para o covering array deve ser resolvido.")
```

```
Iniciando a geração do arquivo de covering array…
O arquivo será salvo em:
'/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/msrn_7_3_2_No1.csv'
Categorias únicas detectadas no CSV de objetos: ['bus', 'car', 'cat', 'sofa',
'person', 'aeroplane', 'tvmonitor', 'bottle', 'chair', 'bird', 'cow', 'dog',
'train', 'bicycle', 'boat', 'pottedplant', 'diningtable', 'motorbike', 'sheep',
'horse'] (Total: 20)

  Arquivo de covering array 'msrn_7_3_2_No1.csv' gerado com sucesso!

--- Primeiras 5 linhas do novo covering array: ---
   0   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  \
0  0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   1   1   0
1  0   0   0   0   0   0   0   0   0   0   1   0   0   1   1   0   0   0   0
2  0   1   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1   0
3  0   0   1   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   1
4  0   0   0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   0   1

   19
```

```
0    0
1    0
2    0
3    0
4    0

Total de linhas no covering array: 7

--- PRÓXIMOS PASSOS ---
1. Agora que o arquivo de covering array foi gerado, você pode rodar a Célula 9B
(do compositer.py) novamente.
   O erro 'FileNotFoundError' para o covering array deve ser resolvido.
```

[16]:
```
# Remove chamadas problemáticas do OpenCV que causam erro no Colab
!sed -i '/cv2.waitKey(0)/d' /content/LV-CIT/compositer.py
!sed -i '/cv2.imshow/d' /content/LV-CIT/compositer.py
!sed -i '/cv2.destroyAllWindows()/d' /content/LV-CIT/compositer.py
print("Linhas problemáticas removidas do compositer.py")
```

```
Linhas problemáticas removidas do compositer.py
```

[17]:
```
#  CÉLULA 9B - Rodar compositer direto chamando img_composite

import os
import shutil

# Importa a função img_composite do compositer.py
from compositer import img_composite

# Define caminho base no Google Drive
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Limpa e recria pasta onde salvará as imagens compostas
output_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/3composite_img")
shutil.rmtree(output_dir, ignore_errors=True)
os.makedirs(output_dir, exist_ok=True)

print("  Pasta 3composite_img limpa e recriada!")

# Define caminho do arquivo CSV do covering array (ajuste conforme seu arquivo␣
  ↪real)
ca_file = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit", "adaptive_random_ca.
  ↪csv")  # *** VERIFIQUE ESTE NOME ***

# Pasta das imagens de objeto (VOC)
input_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/VOC")
```

```
# Pasta de saída (já criada acima)
# output_dir já definido

# Parâmetros do modelo
model = "msrn"

print(f"\n--- Executando img_composite ---")
print(f"Usando arquivo CSV: {ca_file}")
print(f"Pasta de imagens de objeto: {input_dir}")
print(f"Pasta de saída: {output_dir}")

# Chama a função com os parâmetros desejados
img_composite(
    covering_array_file=ca_file,
    input_dir=input_dir,
    output_dir=output_dir,
    num=7,  # quantas imagens compostas gerar
    sample_times=1,
    max_times=10,
    scale_range=(0.5, 1.5),
    overlap_range=(0, 0.3),
    final_size=640,
    do_scale=True,
    do_angle=False,
    select_order="random",
    model=model,
)

print("\n Execução do img_composite concluída.")
```

```
 Pasta 3composite_img limpa e recriada!

--- Executando img_composite ---
Usando arquivo CSV:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/adaptive_random_ca.csv
Pasta de imagens de objeto: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/VOC
Pasta de saída: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img exists

 Execução do img_composite concluída.
```

[18]:
```
import os

base_folder = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/'

csv_files = []
for root, dirs, files in os.walk(base_folder):
```

```python
    for file in files:
        if file.endswith('.csv'):
            csv_files.append(os.path.join(root, file))

print(f"Arquivos CSV encontrados ({len(csv_files)}):")
for f in csv_files:
    print(f)
```

```
Arquivos CSV encontrados (25):
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library/object_de
tect.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library/object_de
tect_generated.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/VOC_library/object_de
tect_VOC.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/2matting_img/COCO_library/object_d
etect_COCO.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_8_0.16689304100000002.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.17653767799999998.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_9_0.24083036800000013.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_9_0.18730666699999998.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.17243494799999992.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_8_0.17522191700000012.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_8_0.28302516199999994.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.13630158800000003.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_11_0.16688180799999996.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_9_0.14956556399999998.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.26326135399999995.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_11_0.15740670899999998.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_9_0.3988816019999999.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.3930993330000001.csv
```

```
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_8_0.22080466199999993.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_8_0.3734852099999999.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_10_0.3255705609999999.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_9_0.16612712200000002.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/msrn_7_3_2_No1.csv
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_8_0.21466511600000016.csv
```

[19]:
```python
import os
import shutil
from compositer import img_composite

# Caminho base
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Pasta saída
output_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/3composite_img")
shutil.rmtree(output_dir, ignore_errors=True)
os.makedirs(output_dir, exist_ok=True)
print("  Pasta 3composite_img limpa e recriada!")

# CSV correto
ca_file = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/1covering_array/adaptive␣
 ↪random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv")

# Pasta imagens VOC
input_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/VOC")

model = "msrn"

print(f"\n--- Executando img_composite ---")
print(f"Usando arquivo CSV: {ca_file}")
print(f"Pasta de imagens de objeto: {input_dir}")
print(f"Pasta de saída: {output_dir}")

img_composite(
    covering_array_file=ca_file,
    input_dir=input_dir,
    output_dir=output_dir,
    num=7,
    sample_times=1,
    max_times=10,
    scale_range=(0.5, 1.5),
    overlap_range=(0, 0.3),
    final_size=640,
    do_scale=True,
    do_angle=False,
    select_order="random",
    model=model,
)

print("\n  Execução do img_composite concluída.")
```

```
  Pasta 3composite_img limpa e recriada!

--- Executando img_composite ---
Usando arquivo CSV:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv
Pasta de imagens de objeto: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/VOC
Pasta de saída: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img exists

  Execução do img_composite concluída.
```

[20]:
```
import os

# Caminho do arquivo CSV usando raw string para lidar com espaços
ca_file = r"/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/
↪adaptive random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv"

# Verifica se o arquivo CSV existe
print(f"Arquivo CSV existe? {os.path.exists(ca_file)}")

# Se não existir, lista o conteúdo da pasta para conferência
if not os.path.exists(ca_file):
    folder = r"/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/
↪adaptive random/"
    if os.path.exists(folder):
        print(f"\nConteúdo da pasta '{folder}':")
        arquivos = os.listdir(folder)
        for arq in arquivos:
            print(arq)
    else:
        print(f"Pasta '{folder}' não existe.")
```

```
Arquivo CSV existe? True
```

[21]:
```
import os
import shutil
from compositer import img_composite

# Caminho base
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"

# Pasta saída
output_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/3composite_img")
shutil.rmtree(output_dir, ignore_errors=True)
os.makedirs(output_dir, exist_ok=True)
print("  Pasta 3composite_img limpa e recriada!")

# CSV correto com espaço (usando raw string)
ca_file = r"/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/
 ↪adaptive random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv"

# Pasta imagens VOC
input_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/VOC")

model = "msrn"

print(f"\n--- Executando img_composite ---")
print(f"Usando arquivo CSV: {ca_file}")
print(f"Pasta de imagens de objeto: {input_dir}")
print(f"Pasta de saída: {output_dir}")

img_composite(
    covering_array_file=ca_file,
    input_dir=input_dir,
    output_dir=output_dir,
    num=7,
    sample_times=1,
    max_times=10,
    scale_range=(0.5, 1.5),
    overlap_range=(0, 0.3),
    final_size=640,
    do_scale=True,
    do_angle=False,
    select_order="random",
    model=model,
)

print("\n  Execução do img_composite concluída.")
```

```
 Pasta 3composite_img limpa e recriada!
```

```
--- Executando img_composite ---
Usando arquivo CSV:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv
Pasta de imagens de objeto: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/VOC
Pasta de saída: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img exists

 Execução do img_composite concluída.
```

[22]:
```
import os
import cv2
import matplotlib.pyplot as plt

composite_folder = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/
 ↪3composite_img/'

if os.path.exists(composite_folder):
    arquivos = []
    for root, dirs, files in os.walk(composite_folder):
        for f in files:
            if f.lower().endswith(('.jpg', '.png', '.jpeg')):
                arquivos.append(os.path.join(root, f))
    if len(arquivos) == 0:
        print("  Nenhuma imagem encontrada na pasta.")
    else:
        print(f"  {len(arquivos)} imagens encontradas. Mostrando a primeira:")
        img_path = arquivos[0]
        img = cv2.imread(img_path)
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.figure(figsize=(10,8))
        plt.imshow(img_rgb)
        plt.axis('off')
        plt.title('Imagem Composta')
        plt.show()
else:
    print("  Pasta de imagens compostas não encontrada.")
```

```
 Nenhuma imagem encontrada na pasta.
```

[23]:
```
import os

base_dir = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/'

for root, dirs, files in os.walk(base_dir):
    print(f"Pasta: {root}")
    print(f"Subpastas: {dirs}")
    print(f"Arquivos: {files}")
```

```
Pasta: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/
Subpastas: []
Arquivos: []
```

[24]:
```
import os
import shutil
from compositer import img_composite

LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"
output_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/3composite_img")

# Limpa e recria a pasta
shutil.rmtree(output_dir, ignore_errors=True)
os.makedirs(output_dir, exist_ok=True)
print(f"  Pasta {output_dir} limpa e recriada!")

# CSV com espaço, raw string para evitar erro no caminho
ca_file = r"/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/
 ↪adaptive random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv"
input_dir = os.path.join(LV_CIT_DRIVE_PATH, "data/lvcit/VOC")
model = "msrn"

print("\n--- Iniciando img_composite ---")
print(f"CSV usado: {ca_file}")
print(f"Pasta de entrada (imagens): {input_dir}")
print(f"Pasta de saída esperada: {output_dir}")

try:
    img_composite(
        covering_array_file=ca_file,
        input_dir=input_dir,
        output_dir=output_dir,
        num=7,
        sample_times=1,
        max_times=10,
        scale_range=(0.5, 1.5),
        overlap_range=(0, 0.3),
        final_size=640,
        do_scale=True,
        do_angle=False,
        select_order="random",
        model=model,
    )
    print("\n  img_composite executado sem erros.")
except Exception as e:
    print(f"\n  Erro durante a execução do img_composite: {e}")

# Lista conteúdo da pasta de saída
print(f"\nConteúdo da pasta {output_dir} após execução:")
for root, dirs, files in os.walk(output_dir):
    print(f"Pasta: {root}")
    print(f"Subpastas: {dirs}")
    print(f"Arquivos: {files}")
```

```
 Pasta /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img limpa e
recriada!

--- Iniciando img_composite ---
CSV usado:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/adaptive
random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv
Pasta de entrada (imagens): /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/VOC
Pasta de saída esperada:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img exists

 img_composite executado sem erros.

Conteúdo da pasta /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
após execução:
Pasta: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
Subpastas: []
Arquivos: []
```

[25]:
```
from compositer import img_composite
import os

#  Limpa e recria a pasta de saída
output_dir = "/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img"
if os.path.exists(output_dir):
    import shutil
    shutil.rmtree(output_dir)
os.makedirs(output_dir, exist_ok=True)
print(f"  Pasta {output_dir} limpa e recriada!")

#  Caminho do CSV - pegue um que VOCÊ TEM
covering_array_file = "/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/
 ↪1covering_array/adaptive random/ca_adaptive random_6_3_2_7_0.
 ↪1531374069999999.csv"

#  Pasta de imagens de entrada
input_dir = "/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/VOC"
```

```
#  Executa o compositer
print("\n--- Executando img_composite ---")
img_composite(
    covering_array_file=covering_array_file,
    input_dir=input_dir,
    output_dir=output_dir,
    num=2,                   # Número de objetos por imagem
    sample_times=1,          # Quantas amostras por linha do CSV
    max_times=0,             # Limite de reutilização de objeto (0 = sem limite)
    scale_range=(0.5, 1.5),
    overlap_range=(0, 0.3),
    final_size=640,
    do_scale=True,
    do_angle=False,
    select_order="random",
    model=None
)

print("\n  Finalizado!")
```

```
 Pasta /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img limpa e
recriada!

--- Executando img_composite ---
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img exists

 Finalizado!
```

[26]:
```
#  CÉLULA BLOCO 9 - Verificando pasta e CSV do covering array correto

import os
import cv2
import matplotlib.pyplot as plt

# Caminho correto da pasta de saída dos compostos
composite_folder = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/
 ↪3composite_img/'

# Caminho correto do covering array CSV
ca_file = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/1covering_array/
 ↪adaptive random/ca_adaptive random_6_3_2_7_0.1531374069999999.csv'

print(f"Pasta existe? {os.path.exists(composite_folder)}")

if os.path.exists(composite_folder):
    print(f"Conteúdo da pasta '{composite_folder}':")
```

```python
    root_files = os.listdir(composite_folder)
    if len(root_files) == 0:
        print("  -> Pasta está vazia!")
    else:
        print(f"  -> {len(root_files)} item(s) encontrado(s): {root_files}")

    print("\nListando conteúdo de subpastas:")
    found_images = []
    for root, dirs, files in os.walk(composite_folder):
        print(f"\nPasta: {root}")
        if dirs:
            print(f"  Subpastas: {dirs}")
        if files:
            print(f"  Arquivos: {files}")
            for f in files:
                if f.lower().endswith(('.jpg','.png','.jpeg')):
                    found_images.append(os.path.join(root, f))
    if not found_images:
        print("\n Nenhuma imagem encontrada em toda a pasta e subpastas.")
    else:
        print(f"\n {len(found_images)} imagem(ns) encontrada(s).")

else:
    print(f" Pasta '{composite_folder}' não existe.")

print(f"\nVerificando arquivo CSV de entrada:")
print(f"Arquivo CSV existe? {os.path.exists(ca_file)}")

def show_img_cv2_matplotlib(img_path, title=None):
    if img_path is None:
        print(" Nenhuma imagem válida para mostrar.")
        return
    img = cv2.imread(img_path)
    if img is None:
        print(f" Erro ao abrir a imagem {img_path}")
        return
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.figure(figsize=(10,8))
    plt.imshow(img_rgb)
    if title:
        plt.title(title)
    plt.axis('off')
    plt.show()

if found_images:
    print(f"\nMostrando a primeira imagem encontrada:\n{found_images[0]}")
    show_img_cv2_matplotlib(found_images[0], title="Imagem Composta Encontrada")
```

```python
else:
    print("Nenhuma imagem para mostrar.")
```

```
Pasta existe? True
Conteúdo da pasta
'/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/':
  -> Pasta está vazia!

Listando conteúdo de subpastas:

Pasta: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/

  Nenhuma imagem encontrada em toda a pasta e subpastas.

Verificando arquivo CSV de entrada:
Arquivo CSV existe? True
Nenhuma imagem para mostrar.
```

[27]:
```python
#  CÉLULA BLOCO 9 - Rodar o compositer.py com o covering array certo

from compositer import img_composite

covering_array_file = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/
 ↪1covering_array/adaptive random/ca_adaptive random_6_3_2_7_0.
 ↪1531374069999999.csv'
input_dir = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/VOC'
output_dir = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/'

img_composite(
    covering_array_file=covering_array_file,
    input_dir=input_dir,
    output_dir=output_dir,
    num=2,
    sample_times=1,
    max_times=1,
    scale_range=(0.5, 1.5),
    overlap_range=(0, 0.3),
    final_size=640,
    do_scale=True,
    do_angle=False,
    select_order="random",
    model=None
)
```

```
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/ exists
```

```
[28]: import os

# Caminho para a pasta onde as imagens compostas foram salvas
composite_folder = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/
 ↪3composite_img/VOC_20/'

print(f" Listando conteúdo da pasta de imagens compostas: {composite_folder}")

try:
    # Lista todos os arquivos e pastas no diretório
    files_in_composite_folder = os.listdir(composite_folder)

    if not files_in_composite_folder:
        print(" A pasta de imagens compostas está vazia. Nenhuma imagem foi␣
 ↪encontrada.")
    else:
        print(" Imagens compostas encontradas. Primeiros 10 arquivos:")
        for i, f in enumerate(files_in_composite_folder):
            if i >= 10: # Limita a exibição para não sobrecarregar
                break
            print(f"- {f}")
        print(f"\nTotal de arquivos encontrados:␣
 ↪{len(files_in_composite_folder)}")

except FileNotFoundError:
    print(f" Erro: A pasta '{composite_folder}' não foi encontrada. Verifique␣
 ↪o caminho.")
except Exception as e:
    print(f" Erro ao listar arquivos: {e}")

print("\n--- NOTA ---")
print("O script 'compositer.py' na versão atual não gera um arquivo 'info.csv'.
 ↪")
print("Esta célula foi ajustada para listar as imagens geradas diretamente.")
```

```
 Listando conteúdo da pasta de imagens compostas:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/VOC_20/
 Erro: A pasta
'/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/VOC_20/' não foi
encontrada. Verifique o caminho.

--- NOTA ---
O script 'compositer.py' na versão atual não gera um arquivo 'info.csv'.
Esta célula foi ajustada para listar as imagens geradas diretamente.
```

10. Verificar imagens compostas e conteúdo

[29]:
```python
import os

# Pasta onde o compositer salva
composite_root = "/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img"

# Lista tudo na raiz
print("\n  Conteúdo de:", composite_root)
print(os.listdir(composite_root))

# Tenta caminhar 3 níveis dentro, se existir
for root, dirs, files in os.walk(composite_root):
    print("\n  Pasta:", root)
    print("  Subpastas:", dirs)
    print("  Arquivos:", files)
```

```
 Conteúdo de: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
[]

 Pasta: /content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img
 Subpastas: []
 Arquivos: []
```

11. Força criação manual de info.csv

[30]:
```python
    import pandas as pd
    import os

    #   Caminho para a pasta onde as imagens compostas foram REALMENTE salvas
    # Este caminho foi confirmado pela Célula 9D.
    images_folder = '/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/
 ↪3composite_img/VOC_20/'

    #   Caminho para o novo info.csv
    # Vamos salvá-lo diretamente na pasta das imagens compostas para␣
 ↪simplificar.
    info_csv = os.path.join(images_folder, 'info.csv')

    print(f"  Lendo imagens em: {images_folder}")

    # Lista imagens JPG ou PNG na pasta correta
    img_files = []
    # Usamos os.listdir para pegar arquivos diretamente na pasta, sem subpastas
    if os.path.exists(images_folder):
        for f in os.listdir(images_folder):
            if f.lower().endswith(('.jpg', '.png', '.jpeg')): # Adicionado .
 ↪jpeg e lower()
                img_files.append(f)
    else:
        print(f"  Erro: A pasta de imagens compostas '{images_folder}' não foi␣
↪encontrada.")

    print(f"Encontradas {len(img_files)} imagens.")

    # Monta CSV básico
    # Note: 'labels' aqui ainda é um placeholder, pois o compositer.py atual␣
↪não gera labels no salvamento.
    # Se você precisar de labels reais, o compositer.py precisaria ser␣
↪modificado para incluí-las.
    df = pd.DataFrame({'filename': img_files, 'labels': ['placeholder'] *␣
↪len(img_files)})

    #  Garante que o diretório pai para o info.csv exista antes de salvar
    os.makedirs(os.path.dirname(info_csv), exist_ok=True)

    try:
        df.to_csv(info_csv, index=False)
        print(f"  Novo info.csv salvo em: {info_csv}")
        print("Primeiras linhas do novo info.csv:")
        print(df.head())
    except Exception as e:
        print(f"  Erro ao salvar CSV: {e}")
```

```
 Lendo imagens em:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/VOC_20/
 Erro: A pasta de imagens compostas
'/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/VOC_20/' não foi
encontrada.
Encontradas 0 imagens.
 Novo info.csv salvo em:
/content/drive/MyDrive/LV_CIT_DATA/data/lvcit/3composite_img/VOC_20/info.csv
Primeiras linhas do novo info.csv:
Empty DataFrame
Columns: [filename, labels]
Index: []
```

12. Executar pipeline principal com os compostos

[31]:
```
import os
import sys
import types
import pandas as pd

# 1. Criar pastas que o código precisa
os.makedirs("/content/LV-CIT/checkpoints/msrn", exist_ok=True)
os.makedirs("/content/LV-CIT/data/voc", exist_ok=True)
os.makedirs("/content/LV-CIT/data/voc/results", exist_ok=True)
os.makedirs("/content/LV-CIT/data/coco", exist_ok=True)

print("  Pastas criadas/confirmadas.")

# 2. Copiar arquivos do seu Drive para a estrutura do projeto
# Atenção: ajuste os caminhos abaixo para os locais corretos no seu Drive se␣
 ↪necessário!

!cp "/content/drive/MyDrive/LV_CIT_DATA/checkpoints/msrn/resnet101_for_msrn.pth.
 ↪tar" "/content/LV-CIT/checkpoints/msrn/"
!cp "/content/drive/MyDrive/LV_CIT_DATA/data/voc_adj.pkl" "/content/LV-CIT/data/
 ↪voc/"
!cp "/content/drive/MyDrive/LV_CIT_DATA/data/voc_glove_word2vec.pkl" "/content/
 ↪LV-CIT/data/voc/"
!cp "/content/drive/MyDrive/LV_CIT_DATA/data/coco_glove_word2vec.pkl" "/content/
 ↪LV-CIT/data/coco/"

print("  Arquivos copiados do Drive para os locais corretos.")

# 3. Corrigir InPlaceABN_Fake para aceitar activation_param
class InPlaceABN_Fake:
    def __init__(self, num_features, activation="leaky_relu",␣
 ↪activation_param=None):
        pass
    def __call__(self, x):
        return x

sys.modules['inplace_abn'] = types.ModuleType('inplace_abn')
sys.modules['inplace_abn'].InPlaceABN = InPlaceABN_Fake

print("  InPlaceABN_Fake ajustado para aceitar 'activation_param'.")

# 4. Criar arquivo Excel vazio para evitar erro
excel_path = "/content/LV-CIT/data/voc/results/result_voc_msrn.xlsx"
if not os.path.isfile(excel_path):
    pd.DataFrame().to_excel(excel_path)
    print(f"  Arquivo Excel vazio criado em {excel_path}")
else:
    print("  Arquivo Excel já existe.")

print("  Setup inicial completo! Pode rodar o seu treinamento agora.")
```

```
 Pastas criadas/confirmadas.
 Arquivos copiados do Drive para os locais corretos.
 InPlaceABN_Fake ajustado para aceitar 'activation_param'.
```

```
Arquivo Excel já existe.
Setup inicial completo! Pode rodar o seu treinamento agora.
```

13. Imports extras pro TensorBoard

```
[32]: from torch.utils.tensorboard import SummaryWriter
      import os
```

14. Cria pasta de logs + writer

```
[33]: log_dir = '/content/logs'
      os.makedirs(log_dir, exist_ok=True)
      writer = SummaryWriter(log_dir=log_dir)

      print(f"TensorBoard log directory: {log_dir}")
```

```
TensorBoard log directory: /content/logs
```

15. Função de treino + val com logs

```
[34]: import torch
      from torch.utils.tensorboard import SummaryWriter

      # Função treino de 1 epoch
      def train_one_epoch(epoch, model, train_loader, optimizer, criterion, device):
          model.train()
          running_loss = 0.0
          correct = 0
          total = 0

          for inputs, targets in train_loader:
              inputs, targets = inputs.to(device), targets.to(device)
              optimizer.zero_grad()

              outputs = model(inputs)
              loss = criterion(outputs, targets)
              loss.backward()
              optimizer.step()

              running_loss += loss.item() * inputs.size(0)
              _, predicted = outputs.max(1)
              total += targets.size(0)
              correct += predicted.eq(targets).sum().item()

          epoch_loss = running_loss / total
          epoch_acc = 100. * correct / total
          print(f" Epoch {epoch} - Train Loss: {epoch_loss:.4f} - Train Acc:␣
        ↪{epoch_acc:.2f}%")
          return epoch_loss, epoch_acc
```

```
# Função validação de 1 epoch
def validate(epoch, model, val_loader, criterion, device):
    model.eval()
    val_loss = 0.0
    correct = 0
    total = 0

    with torch.no_grad():
        for inputs, targets in val_loader:
            inputs, targets = inputs.to(device), targets.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, targets)

            val_loss += loss.item() * inputs.size(0)
            _, predicted = outputs.max(1)
            total += targets.size(0)
            correct += predicted.eq(targets).sum().item()

    epoch_loss = val_loss / total
    epoch_acc = 100. * correct / total
    print(f" Epoch {epoch} - Val Loss: {epoch_loss:.4f} - Val Acc: {epoch_acc:.
 ↪2f}%")
    return epoch_loss, epoch_acc

# Inicia TensorBoard
writer = SummaryWriter()
print(" Funções definidas e TensorBoard pronto!")
```

```
Funções definidas e TensorBoard pronto!
```

16. Loop de epochs + Scheduler + Early Stop

[35]:
```
import os
from PIL import Image
import torch
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms

class ImagesFolderDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform
        # Lista todos os arquivos de imagem na pasta (não entra em subpastas)
        self.image_files = [f for f in os.listdir(root_dir)
                            if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

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

    def __getitem__(self, idx):
        img_name = self.image_files[idx]
        img_path = os.path.join(self.root_dir, img_name)
        image = Image.open(img_path).convert('RGB')
        if self.transform:
            image = self.transform(image)
        # Label dummy, pq não tem classe aqui
        label = -1
        return image, label

# Configura caminhos
VOC_PATH = "/content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/VOC2007/
 ↪JPEGImages"
COCO_PATH = "/content/drive/MyDrive/LV_CIT_DATA/data/coco/data/val2014"

# Transformações de imagem
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
])

# Instancia datasets e dataloaders
voc_dataset = ImagesFolderDataset(VOC_PATH, transform=transform)
voc_loader = DataLoader(voc_dataset, batch_size=32, shuffle=True)

coco_dataset = ImagesFolderDataset(COCO_PATH, transform=transform)
coco_loader = DataLoader(coco_dataset, batch_size=32, shuffle=True)

print(f"VOC dataset size: {len(voc_dataset)} imagens")
print(f"COCO dataset size: {len(coco_dataset)} imagens")
```

```
VOC dataset size: 9963 imagens
COCO dataset size: 40504 imagens
```

[38]:
```
import os
import glob
import pandas as pd
import torch
from PIL import Image
from torchvision import transforms, models
from torchvision.utils import make_grid
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn
import torch.optim as optim
from torch.utils.tensorboard import SummaryWriter
```

```
#  Paths
LV_CIT_DRIVE_PATH = "/content/drive/MyDrive/LV_CIT_DATA"
VOC_IMAGES_PATH = os.path.join(LV_CIT_DRIVE_PATH, 'data/voc/VOCdevkit/VOC2007/
↪JPEGImages')
COCO_IMAGES_PATH = os.path.join(LV_CIT_DRIVE_PATH, 'data/coco/data/val2014')
CHECKPOINT_PATH = os.path.join(LV_CIT_DRIVE_PATH, 'best_checkpoint.pth')

VOC_CSV = os.path.join(LV_CIT_DRIVE_PATH, "voc_labels.csv")
COCO_CSV = os.path.join(LV_CIT_DRIVE_PATH, "coco_labels.csv")

VOC_LABELS = ['aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus',
              'car', 'cat', 'chair', 'cow', 'diningtable', 'dog',
              'horse', 'motorbike', 'person', 'pottedplant', 'sheep',
              'sofa', 'train', 'tvmonitor']

print(f"  VOC: {VOC_IMAGES_PATH} existe? {os.path.exists(VOC_IMAGES_PATH)}")
print(f"  COCO: {COCO_IMAGES_PATH} existe? {os.path.exists(COCO_IMAGES_PATH)}")

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
])

#  Dataset customizado com labels simulados
class FlatImageDataset(Dataset):
    def __init__(self, folder_path, csv_path, transform=None, labels_list=None):
        self.folder_path = folder_path
        self.transform = transform
        self.labels_list = labels_list or []
        self.image_paths = []

        for ext in ('*.jpg', '*.jpeg', '*.png'):
            self.image_paths.extend(glob.glob(os.path.join(folder_path, ext)))

        self.image_paths.sort()

        if os.path.exists(csv_path):
            self.labels_df = pd.read_csv(csv_path)
        else:
            # Simula um CSV fake
            self.labels_df = pd.DataFrame({
                'filename': [os.path.basename(p) for p in self.image_paths],
                'label': [self.labels_list[i % len(self.labels_list)] for i in␣
↪range(len(self.image_paths))]
            })
            self.labels_df.to_csv(csv_path, index=False)
    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        img_path = self.image_paths[idx]
        image = Image.open(img_path).convert('RGB')
        if self.transform:
            image = self.transform(image)

        # Label index (dummy)
        label_str = self.labels_df.iloc[idx]['label']
        label = self.labels_list.index(label_str)
        return image, label

#   Loaders fixos
voc_dataset = FlatImageDataset(VOC_IMAGES_PATH, VOC_CSV, transform,␣
 ↪labels_list=VOC_LABELS) if os.path.exists(VOC_IMAGES_PATH) else None
coco_dataset = FlatImageDataset(COCO_IMAGES_PATH, COCO_CSV, transform,␣
 ↪labels_list=VOC_LABELS) if os.path.exists(COCO_IMAGES_PATH) else None

voc_loader = DataLoader(voc_dataset, batch_size=64, shuffle=True) if␣
 ↪voc_dataset else None
coco_loader = DataLoader(coco_dataset, batch_size=64, shuffle=True) if␣
 ↪coco_dataset else None

loader = voc_loader if voc_loader else coco_loader

print(f"VOC imagens: {len(voc_dataset) if voc_dataset else 0}")
print(f"COCO imagens: {len(coco_dataset) if coco_dataset else 0}")

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"  Device: {device}")

model = models.resnet18(weights='IMAGENET1K_V1')
model.fc = nn.Linear(model.fc.in_features, len(VOC_LABELS))
model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-3)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.5)

start_epoch = 1
if os.path.exists(CHECKPOINT_PATH):
    model.load_state_dict(torch.load(CHECKPOINT_PATH))
    print("  Checkpoint carregado!")
else:
    print("  Nenhum checkpoint, começando do zero.")

writer = SummaryWriter('/content/logs')

def train_one_epoch(epoch, model, loader, optimizer, criterion, device):
    model.train()
    running_loss, correct, total = 0.0, 0, 0
    for batch_idx, (inputs, labels) in enumerate(loader):
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

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

        img_grid = make_grid(inputs[:4].cpu(), nrow=2, normalize=True)
        writer.add_image(f'Train/ImageGrid_epoch_{epoch}_batch_{batch_idx}',␣
 ↪img_grid, epoch * len(loader) + batch_idx)

    epoch_loss = running_loss / total
    epoch_acc = 100. * correct / total
    print(f"[Train {epoch}] Loss: {epoch_loss:.4f} Acc: {epoch_acc:.2f}%")
    return epoch_loss, epoch_acc

def validate(epoch, model, loader, criterion, device):
    model.eval()
    running_loss, correct, total = 0.0, 0, 0
    with torch.no_grad():
        for batch_idx, (inputs, labels) in enumerate(loader):
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)

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

            img_grid = make_grid(inputs[:4].cpu(), nrow=2, normalize=True)
            writer.add_image(f'Val/ImageGrid_epoch_{epoch}_batch_{batch_idx}',␣
 ↪img_grid, epoch * len(loader) + batch_idx)
```

```
    epoch_loss = running_loss / total
    epoch_acc = 100. * correct / total
    print(f"[Val {epoch}] Loss: {epoch_loss:.4f} Acc: {epoch_acc:.2f}%")
    return epoch_loss, epoch_acc

num_epochs = 5

if loader:
    for epoch in range(start_epoch, num_epochs + 1):
        train_loss, train_acc = train_one_epoch(epoch, model, loader,␣
 ↪optimizer, criterion, device)
        val_loss, val_acc = validate(epoch, model, loader, criterion, device)

        writer.add_scalar('Loss/Train', train_loss, epoch)
        writer.add_scalar('Loss/Val', val_loss, epoch)
        writer.add_scalar('Acc/Train', train_acc, epoch)
        writer.add_scalar('Acc/Val', val_acc, epoch)

        scheduler.step()
        torch.save(model.state_dict(), CHECKPOINT_PATH)
        print(f"  Checkpoint atualizado! Epoch {epoch}")

    writer.close()
    print(f"  Treino finalizado com {num_epochs} epochs!")
else:
    print("  Nenhum dataset válido encontrado.")
```

```
 VOC: /content/drive/MyDrive/LV_CIT_DATA/data/voc/VOCdevkit/VOC2007/JPEGImages
existe? True
 COCO: /content/drive/MyDrive/LV_CIT_DATA/data/coco/data/val2014 existe? True
VOC imagens: 9963
COCO imagens: 40504
 Device: cuda

Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to
/root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|     | 44.7M/44.7M [00:00<00:00, 84.4MB/s]

 Nenhum checkpoint, começando do zero.
[Train 1] Loss: 3.0884 Acc: 5.22%
[Val 1] Loss: 3.0657 Acc: 5.38%
 Checkpoint atualizado! Epoch 1
[Train 2] Loss: 3.0266 Acc: 4.72%
[Val 2] Loss: 3.0075 Acc: 5.23%
 Checkpoint atualizado! Epoch 2
[Train 3] Loss: 3.0210 Acc: 5.40%
[Val 3] Loss: 3.0095 Acc: 5.85%
 Checkpoint atualizado! Epoch 3
```

```
[Train 4] Loss: 3.0216 Acc: 5.36%
[Val 4] Loss: 3.0093 Acc: 5.14%
 Checkpoint atualizado! Epoch 4
[Train 5] Loss: 3.0205 Acc: 4.85%
[Val 5] Loss: 3.0470 Acc: 5.00%
 Checkpoint atualizado! Epoch 5
 Treino finalizado com 5 epochs!
```

17. Rodar TensorBoard no Colab

[39]:
```
#  Só execute depois de rodar o treino!
%load_ext tensorboard
%tensorboard --logdir /content/logs
```

```
Output hidden; open in https://colab.research.google.com to view.
```

[40]:
```
# Read the content of the funcs.py file
with open('/content/LV-CIT/models/ASL/funcs.py', 'r') as f:
    funcs_content = f.read()

print(funcs_content)
```

```
import torch
import time
import pandas as pd
import os

from .helper_functions.helper_functions import mAP, AverageMeter
from .models import create_model
from util import cal_score


def ASL(args):
    # setup model
    print('creating model {}…'.format(args.model_type))
    state = torch.load(args.model_path, map_location='cpu')
    args.num_classes = state['num_classes']
    args.do_bottleneck_head = False
    model = create_model(args).cuda()
    model.load_state_dict(state['model'], strict=True)
    model.eval()
    print('done')
    return model


def asl_validate_multi(val_loader, model, args, res_type=0):
    print("starting actuall validation")
    batch_time = AverageMeter()
    prec = AverageMeter()
    rec = AverageMeter()

    Sig = torch.nn.Sigmoid()

    end = time.time()
    tp, fp, fn, tn, count = 0, 0, 0, 0, 0
    preds = []
    targets = []
    result = []
    for i, (name, input, target) in enumerate(val_loader):
        # compute output
        with torch.no_grad():
            output = Sig(model(input.cuda())).cpu()

        # for mAP calculation
        preds.append(output.cpu())
        targets.append(target.cpu())

        # measure accuracy and record loss
        pred = output.data.gt(args.threshold).long()

        tp += (pred + target).eq(2).sum(dim=0)
        fp += (pred - target).eq(1).sum(dim=0)
        fn += (pred - target).eq(-1).sum(dim=0)
        tn += (pred + target).eq(0).sum(dim=0)
        count += input.size(0)

        this_tp = (pred + target).eq(2).sum()
        this_fp = (pred - target).eq(1).sum()
        this_fn = (pred - target).eq(-1).sum()
        this_tn = (pred + target).eq(0).sum()

        this_prec = this_tp.float() / (
            this_tp + this_fp).float() * 100.0 if this_tp + this_fp != 0 else
0.0
        this_rec = this_tp.float() / (
            this_tp + this_fn).float() * 100.0 if this_tp + this_fn != 0 else
0.0

        prec.update(float(this_prec), input.size(0))
        rec.update(float(this_rec), input.size(0))

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        p_c = [float(tp[i].float() / (tp[i] + fp[i]).float()) * 100.0
               if tp[i] > 0 else 0.0
```

```
              for i in range(len(tp))]
       r_c = [float(tp[i].float() / (tp[i] + fn[i]).float()) * 100.0
              if tp[i] > 0 else 0.0
              for i in range(len(tp))]
       f_c = [2 * p_c[i] * r_c[i] / (p_c[i] + r_c[i])
              if tp[i] > 0 else 0.0
              for i in range(len(tp))]

       mean_p_c = sum(p_c) / len(p_c)
       mean_r_c = sum(r_c) / len(r_c)
       mean_f_c = sum(f_c) / len(f_c)

       p_o = tp.sum().float() / (tp + fp).sum().float() * 100.0
       r_o = tp.sum().float() / (tp + fn).sum().float() * 100.0
       f_o = 2 * p_o * r_o / (p_o + r_o)

       if i % args.print_freq == 0:
           print(
               'Test: [{0}/{1}]\t'
               'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
               'Precision {prec.val:.2f} ({prec.avg:.2f})\t'
               'Recall {rec.val:.2f} ({rec.avg:.2f})'.format(
                   i, len(val_loader), batch_time=batch_time,
                   prec=prec, rec=rec
               )
           )
           print(
               'P_C {:.2f} R_C {:.2f} F_C {:.2f} P_O {:.2f} R_O {:.2f} F_O
{:.2f}'
               .format(mean_p_c, mean_r_c, mean_f_c, p_o, r_o, f_o))

       if res_type == 0:
           temp_cat = []
           cat_id = val_loader.dataset.get_cat2id()
           for item in cat_id:
               temp_cat.append(item)
           for i in range(len(name)):
               labels = []
               labels_gt = []
               for j in range(args.num_classes):
                   if output[i][j] > args.threshold:
                       labels.append(temp_cat[j])
                   if target[i][j] > 0:
                       labels_gt.append(temp_cat[j])
               result.append([name[i].split(os.sep)[-1], "|".join(labels),
"|".join(labels_gt)])
       else:
           cat_id = val_loader.dataset.get_cat2id()
```

```
            id_cat = list(cat_id.keys())
            for i in range(len(name)):
                temp = {"filename": name[i]}
                labels = []
                for j in range(args.num_classes):
                    if output.numpy()[i][j] > args.threshold:
                        temp[id_cat[j]] = output.numpy()[i][j]
                        labels.append(id_cat[j])
                    else:
                        temp[id_cat[j]] = -1
                temp["labels_gt"] = "|".join(sorted(
                    [id_cat[idx] for idx, value in enumerate(target[i]) if value
== 1]
                ))
                temp["labels"] = "|".join(sorted(labels))
                temp["pass"] = 1 if temp["labels_gt"] == temp["labels"] else 0
                result.append(temp)

    print(
        '--------------------------------------------------------------------')
    print(' * P_C {:.2f} R_C {:.2f} F_C {:.2f} P_O {:.2f} R_O {:.2f} F_O {:.2f}'
          .format(mean_p_c, mean_r_c, mean_f_c, p_o, r_o, f_o))

    mAP_score = mAP(torch.cat(targets).numpy(), torch.cat(preds).numpy())
    print("mAP score:", mAP_score)

    if res_type == 0:
        result = pd.DataFrame(result)
        result.rename(columns={0: "filename", 1: "labels", 2: "labels_gt"},
inplace=True)
        return result, mAP_score
    else:
        result = pd.DataFrame(result)
        result = result[["filename"] +
list(sorted(val_loader.dataset.get_cat2id().keys())) + ["labels_gt", "labels",
"pass"]]
        accuracy = result.groupby(by="labels_gt", as_index=False,
sort=False)[["labels_gt", "pass"]].mean()
        result["score"] = result.apply(
            lambda x: cal_score(
                x["labels_gt"], x["labels"],
                args.num_classes, args.way_num, val_loader.dataset.get_cat2id()
            ), axis=1
        )
        accuracy.rename(columns={"labels_gt": "labels_gt", "pass": "accuracy"},
inplace=True)
        return result, accuracy, mAP_score
```

```
[41]: import os

      # List the contents of the models/ASL directory
      asl_path = '/content/LV-CIT/models/ASL'
      if os.path.exists(asl_path):
          print(f"Contents of {asl_path}:")
          print(os.listdir(asl_path))
      else:
          print(f"Directory not found: {asl_path}")
```

```
Contents of /content/LV-CIT/models/ASL:
['README.md', 'models', 'loss_functions', 'helper_functions', 'funcs.py']
```

```
[42]: import os

      # Change to the LV-CIT directory
      %cd /content/LV-CIT

      # Update git submodules
      !git submodule update --init --recursive

      # List the contents of the models/ASL directory again to confirm
      asl_path = '/content/LV-CIT/models/ASL'
      if os.path.exists(asl_path):
          print(f"\nContents of {asl_path} after submodule update:")
          print(os.listdir(asl_path))
      else:
          print(f"\nDirectory still not found after submodule update: {asl_path}")
```

```
/content/LV-CIT

Contents of /content/LV-CIT/models/ASL after submodule update:
['README.md', 'models', 'loss_functions', 'helper_functions', 'funcs.py']
```

```
[43]: # Read the content of the funcs.py file
      try:
          with open('/content/LV-CIT/models/ASL/funcs.py', 'r') as f:
              funcs_content = f.read()

          print("Content of models/ASL/funcs.py:")
          print(funcs_content)
      except FileNotFoundError:
          print("Error: /content/LV-CIT/models/ASL/funcs.py not found. Please ensure␣
       ↪the LV-CIT repository is cloned and submodules are updated.")
      except Exception as e:
          print(f"An error occurred while reading the file: {e}")
```

```
Content of models/ASL/funcs.py:
```

```
import torch
import time
import pandas as pd
import os

from .helper_functions.helper_functions import mAP, AverageMeter
from .models import create_model
from util import cal_score


def ASL(args):
    # setup model
    print('creating model {}…'.format(args.model_type))
    state = torch.load(args.model_path, map_location='cpu')
    args.num_classes = state['num_classes']
    args.do_bottleneck_head = False
    model = create_model(args).cuda()
    model.load_state_dict(state['model'], strict=True)
    model.eval()
    print('done')
    return model


def asl_validate_multi(val_loader, model, args, res_type=0):
    print("starting actuall validation")
    batch_time = AverageMeter()
    prec = AverageMeter()
    rec = AverageMeter()

    Sig = torch.nn.Sigmoid()

    end = time.time()
    tp, fp, fn, tn, count = 0, 0, 0, 0, 0
    preds = []
    targets = []
    result = []
    for i, (name, input, target) in enumerate(val_loader):
        # compute output
        with torch.no_grad():
            output = Sig(model(input.cuda())).cpu()

        # for mAP calculation
        preds.append(output.cpu())
        targets.append(target.cpu())

        # measure accuracy and record loss
        pred = output.data.gt(args.threshold).long()
        tp += (pred + target).eq(2).sum(dim=0)
        fp += (pred - target).eq(1).sum(dim=0)
        fn += (pred - target).eq(-1).sum(dim=0)
        tn += (pred + target).eq(0).sum(dim=0)
        count += input.size(0)

        this_tp = (pred + target).eq(2).sum()
        this_fp = (pred - target).eq(1).sum()
        this_fn = (pred - target).eq(-1).sum()
        this_tn = (pred + target).eq(0).sum()

        this_prec = this_tp.float() / (
            this_tp + this_fp).float() * 100.0 if this_tp + this_fp != 0 else
0.0
        this_rec = this_tp.float() / (
            this_tp + this_fn).float() * 100.0 if this_tp + this_fn != 0 else
0.0

        prec.update(float(this_prec), input.size(0))
        rec.update(float(this_rec), input.size(0))

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        p_c = [float(tp[i].float() / (tp[i] + fp[i]).float()) * 100.0
               if tp[i] > 0 else 0.0
               for i in range(len(tp))]
        r_c = [float(tp[i].float() / (tp[i] + fn[i]).float()) * 100.0
               if tp[i] > 0 else 0.0
               for i in range(len(tp))]
        f_c = [2 * p_c[i] * r_c[i] / (p_c[i] + r_c[i])
               if tp[i] > 0 else 0.0
               for i in range(len(tp))]

        mean_p_c = sum(p_c) / len(p_c)
        mean_r_c = sum(r_c) / len(r_c)
        mean_f_c = sum(f_c) / len(f_c)

        p_o = tp.sum().float() / (tp + fp).sum().float() * 100.0
        r_o = tp.sum().float() / (tp + fn).sum().float() * 100.0
        f_o = 2 * p_o * r_o / (p_o + r_o)

        if i % args.print_freq == 0:
            print(
                'Test: [{0}/{1}]\t'
                'Time {batch_time.val:.3f} ({batch_time.avg:.3f})\t'
                'Precision {prec.val:.2f} ({prec.avg:.2f})\t'
                'Recall {rec.val:.2f} ({rec.avg:.2f})'.format(
                    i, len(val_loader), batch_time=batch_time,
                    prec=prec, rec=rec
                )
            )
            print(
                'P_C {:.2f} R_C {:.2f} F_C {:.2f} P_O {:.2f} R_O {:.2f} F_O
{:.2f}'
                .format(mean_p_c, mean_r_c, mean_f_c, p_o, r_o, f_o))

        if res_type == 0:
            temp_cat = []
            cat_id = val_loader.dataset.get_cat2id()
            for item in cat_id:
                temp_cat.append(item)
            for i in range(len(name)):
                labels = []
                labels_gt = []
                for j in range(args.num_classes):
                    if output[i][j] > args.threshold:
                        labels.append(temp_cat[j])
                    if target[i][j] > 0:
                        labels_gt.append(temp_cat[j])
                result.append([name[i].split(os.sep)[-1], "|".join(labels),
"|".join(labels_gt)])
        else:
            cat_id = val_loader.dataset.get_cat2id()
            id_cat = list(cat_id.keys())
            for i in range(len(name)):
                temp = {"filename": name[i]}
                labels = []
                for j in range(args.num_classes):
                    if output.numpy()[i][j] > args.threshold:
                        temp[id_cat[j]] = output.numpy()[i][j]
                        labels.append(id_cat[j])
                    else:
                        temp[id_cat[j]] = -1
                temp["labels_gt"] = "|".join(sorted(
                    [id_cat[idx] for idx, value in enumerate(target[i]) if value
== 1]
                ))
                temp["labels"] = "|".join(sorted(labels))
                temp["pass"] = 1 if temp["labels_gt"] == temp["labels"] else 0
                result.append(temp)

    print(
        '-----------------------------------------------------------------')
    print(' * P_C {:.2f} R_C {:.2f} F_C {:.2f} P_O {:.2f} R_O {:.2f} F_O {:.2f}'
            .format(mean_p_c, mean_r_c, mean_f_c, p_o, r_o, f_o))

    mAP_score = mAP(torch.cat(targets).numpy(), torch.cat(preds).numpy())
    print("mAP score:", mAP_score)

    if res_type == 0:
        result = pd.DataFrame(result)
        result.rename(columns={0: "filename", 1: "labels", 2: "labels_gt"},
inplace=True)
        return result, mAP_score
    else:
        result = pd.DataFrame(result)
        result = result[["filename"] +
list(sorted(val_loader.dataset.get_cat2id().keys())) + ["labels_gt", "labels",
"pass"]]
        accuracy = result.groupby(by="labels_gt", as_index=False,
sort=False)[["labels_gt", "pass"]].mean()
        result["score"] = result.apply(
            lambda x: cal_score(
                x["labels_gt"], x["labels"],
                args.num_classes, args.way_num, val_loader.dataset.get_cat2id()
            ), axis=1
        )
        accuracy.rename(columns={"labels_gt": "labels_gt", "pass": "accuracy"},
inplace=True)
        return result, accuracy, mAP_score
```

[44]:
```
import os

# Define the path to the funcs.py file
funcs_file_path = '/content/LV-CIT/models/ASL/funcs.py'

# Check if the file exists before attempting to read and modify
if os.path.exists(funcs_file_path):
    # Read the content of the funcs.py file
    with open(funcs_file_path, 'r') as f:
        funcs_content = f.read()

    # Replace the incorrect checkpoint filename with the correct one
    # Assuming the incorrect path appears as a string literal in the file
    modified_content = funcs_content.replace(
        "'checkpoints/asl/voc_checkpoint.pth'",
        "'checkpoints/asl/voc_checkpoints.pth.tar'"
    )
    # Also replace the coco checkpoint path if it exists
    modified_content = modified_content.replace(
        "'checkpoints/asl/coco_checkpoint.pth'",
        "'checkpoints/asl/coco_checkpoints.pth.tar'"
    )


    # Write the modified content back to the file
    with open(funcs_file_path, 'w') as f:
        f.write(modified_content)

    print(f"  Modified {funcs_file_path} with correct checkpoint paths.")

else:
    print(f"  Error: {funcs_file_path} not found. Please ensure the LV-CIT␣
 ↪repository is cloned and submodules are updated correctly.")
```

```
 Modified /content/LV-CIT/models/ASL/funcs.py with correct checkpoint paths.
```

[45]:
```
#   Bloco 99 - Limpeza de CSVs Duplicados no Google Drive
import os
import shutil
from collections import defaultdict

#   Caminho raiz do seu projeto
root_drive = "/content/drive/MyDrive/LV_CIT_DATA"

#   Pasta onde os duplicados serão movidos
trash_folder = os.path.join(root_drive, "_TRASH_DUPLICATES")
os.makedirs(trash_folder, exist_ok=True)

#   Extensões a verificar
extensions = [".csv"]

#   Dicionário para armazenar arquivos por nome base
files_by_name = defaultdict(list)

#   Procura arquivos CSV
for folder_path, _, files in os.walk(root_drive):
    for file in files:
        if any(file.endswith(ext) for ext in extensions):
            full_path = os.path.join(folder_path, file)
            files_by_name[file].append(full_path)

#   Conta quantos duplicados existem
duplicated = {name: paths for name, paths in files_by_name.items() if␣
 ↪len(paths) > 1}

print(f"  {len(duplicated)} nome(s) com duplicatas encontradas.")
```

```
#  Função para mover duplicados mantendo o mais novo/maior
for name, paths in duplicated.items():
    # Ordena por tamanho (desc) e data de modificação (desc)
    paths.sort(key=lambda p: (os.path.getsize(p), os.path.getmtime(p)),␣
 ↪reverse=True)
    keep = paths[0]
    duplicates = paths[1:]
    print(f"\n Mantendo: {keep}")
    for dup in duplicates:
        rel_path = os.path.relpath(dup, root_drive)
        trash_path = os.path.join(trash_folder, rel_path)
        os.makedirs(os.path.dirname(trash_path), exist_ok=True)
        print(f"  -> Mover para: {trash_path}")

    # Confirma se o usuário quer mesmo mover
    move = input(f"Deseja mover os duplicados de '{name}'? Digite SIM para␣
 ↪confirmar: ").strip().upper()
    if move == "SIM":
        for dup in duplicates:
            rel_path = os.path.relpath(dup, root_drive)
            trash_path = os.path.join(trash_folder, rel_path)
            os.makedirs(os.path.dirname(trash_path), exist_ok=True)
            shutil.move(dup, trash_path)
        print(f" Duplicatas de '{name}' movidas para {trash_folder}")
    else:
        print(f" Nenhuma duplicata de '{name}' foi movida.")

print("\n Verificação finalizada. Confira a pasta _TRASH_DUPLICATES para␣
 ↪limpar de vez se desejar.")
```

```
 0 nome(s) com duplicatas encontradas.

 Verificação finalizada. Confira a pasta _TRASH_DUPLICATES para limpar de vez
se desejar.
```