

My steps to modify YOLO to YOLO with 4 channels

author: Anna Gelencsér-Horváth
Pázmány Péter Catholic University
Budapest, Hungary

[CC BY-NC](#)

If you create a virtual env and would like to replace ultralytics with the modified directory:
`apptainer exec --nv instance://yolo4dinst python3.10 -m virtualenv -p python3.10 $ENVV`

cfg/models/v8

yolov8.yaml and yolov8-seg.yaml:

```
# Ultralytics YOLO 🚀, AGPL-3.0 license
# YOLOv8-seg instance segmentation model. For Usage examples

# Parameters
nc: 80 # number of classes
scales: # model compound scaling constants, i.e. 'model=yolov8n'
  # [depth, width, max_channels]
  n: [0.33, 0.25, 1024]
  s: [0.33, 0.50, 1024]
  m: [0.67, 0.75, 768]
  l: [1.00, 1.00, 512]
  x: [1.00, 1.25, 512]
  ch: 4

# Ultralytics YOLO 🚀, AGPL-3.0 license
# YOLOv8 object detection model with P3-P5 outputs. For Usage examples

# Parameters
nc: 80 # number of classes
scales: # model compound scaling constants, i.e. 'model=yolov8n'
  # [depth, width, max_channels]
  n: [0.33, 0.25, 1024] # YOLOv8n summary: 225 layers, 3157
  s: [0.33, 0.50, 1024] # YOLOv8s summary: 225 layers, 11166
  m: [0.67, 0.75, 768] # YOLOv8m summary: 295 layers, 259026
  l: [1.00, 1.00, 512] # YOLOv8l summary: 365 layers, 436915
  x: [1.00, 1.25, 512] # YOLOv8x summary: 365 layers, 682296
  ch: 4
```

data

```
class Albumentations:
    """
    Albumentations transformations.

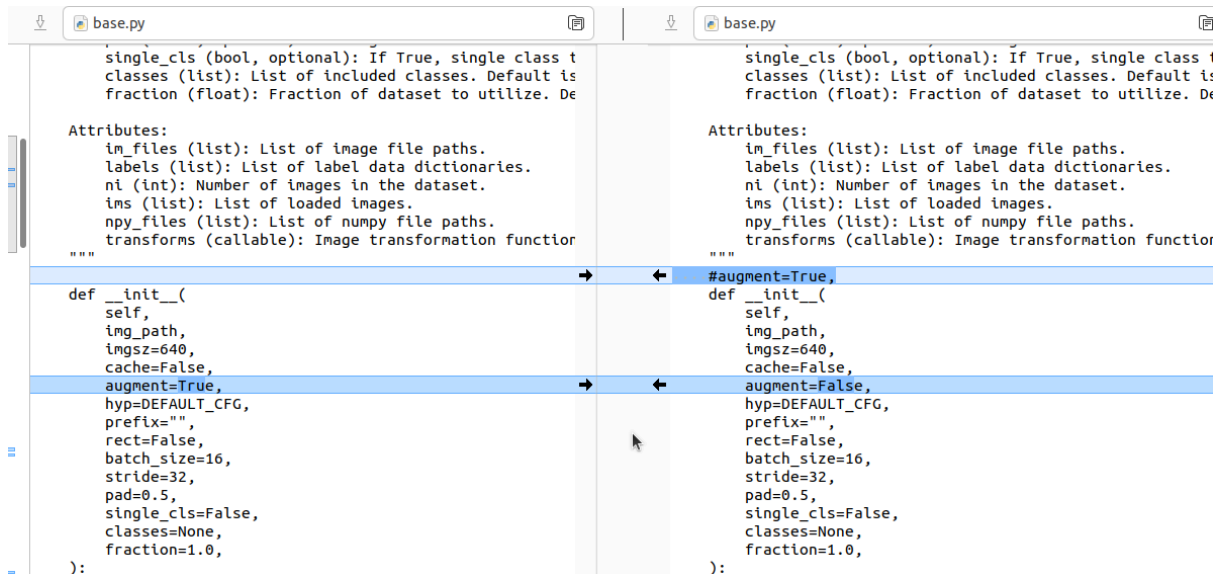
    Optional, uninstall package to disable. Applies Blur, MedianBlur, Histogram Equalization, random change of brightness and compression.
    """

    def __init__(self, p=1.0):
        """Initialize the transform object for YOLO bbox for self.p = p
        self.p = p
        self.transform = None
        prefix = colorstr("albumentations: ")
        try:
            import albumentations as A

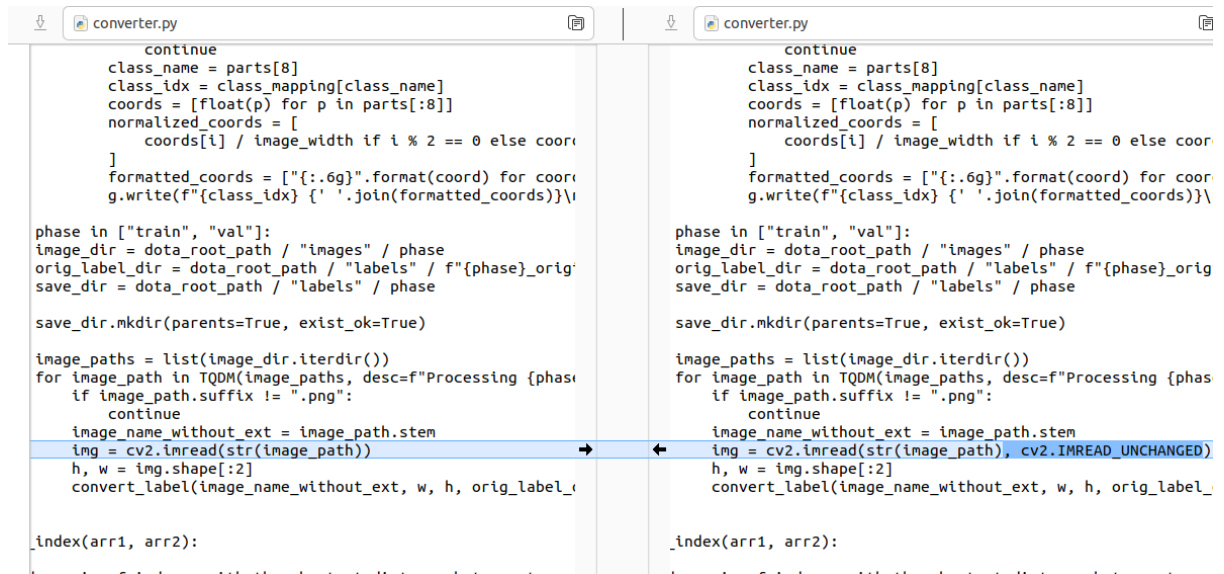
            check_version(A.__version__, "1.0.3", hard=True)

            # Transforms
            T = [
                A.Blur(p=0.01),
                A.MedianBlur(p=0.01),
                A.ToGray(p=0.01),
                A.CLAHE(p=0.01),
                A.RandomBrightnessContrast(p=0.0),
                A.RandomGamma(p=0.0),
                A.ImageCompression(quality_lower=75, p=0.0),
            ]

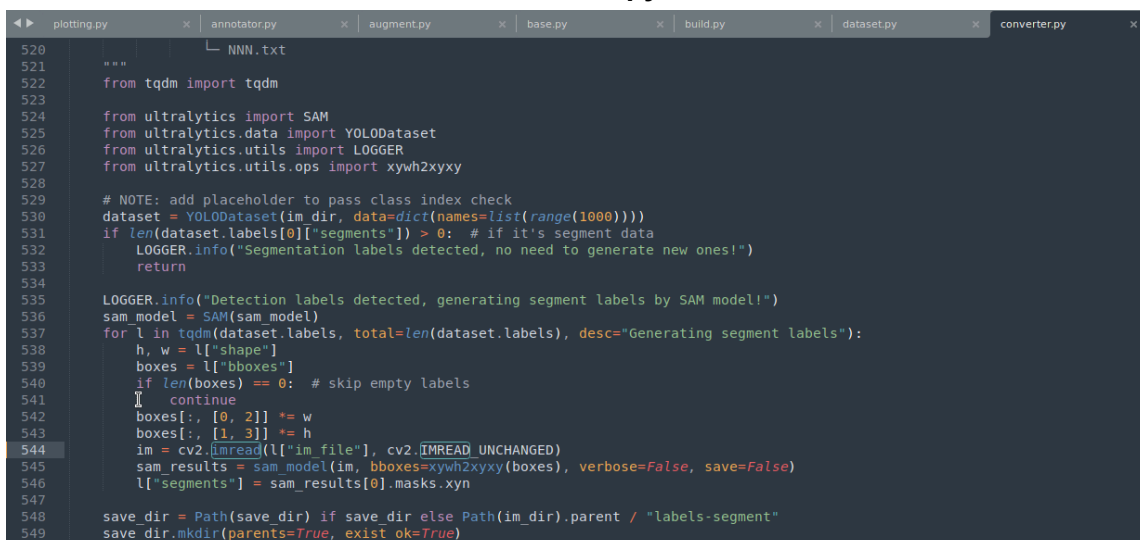
        except ImportError:
            self.transform = None
```



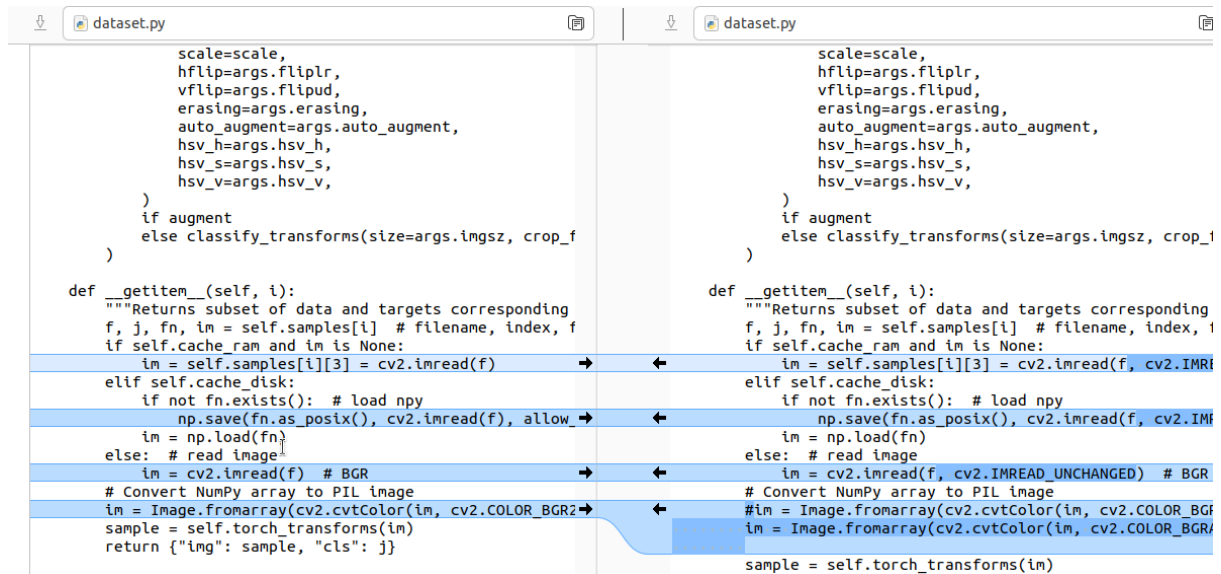
N.B. in base.py:
there are a LOT OF cv2.imread-s.
Search for ALL and add: cv2.IMREAD_UNCHANGED
e.g.:



converter.py:



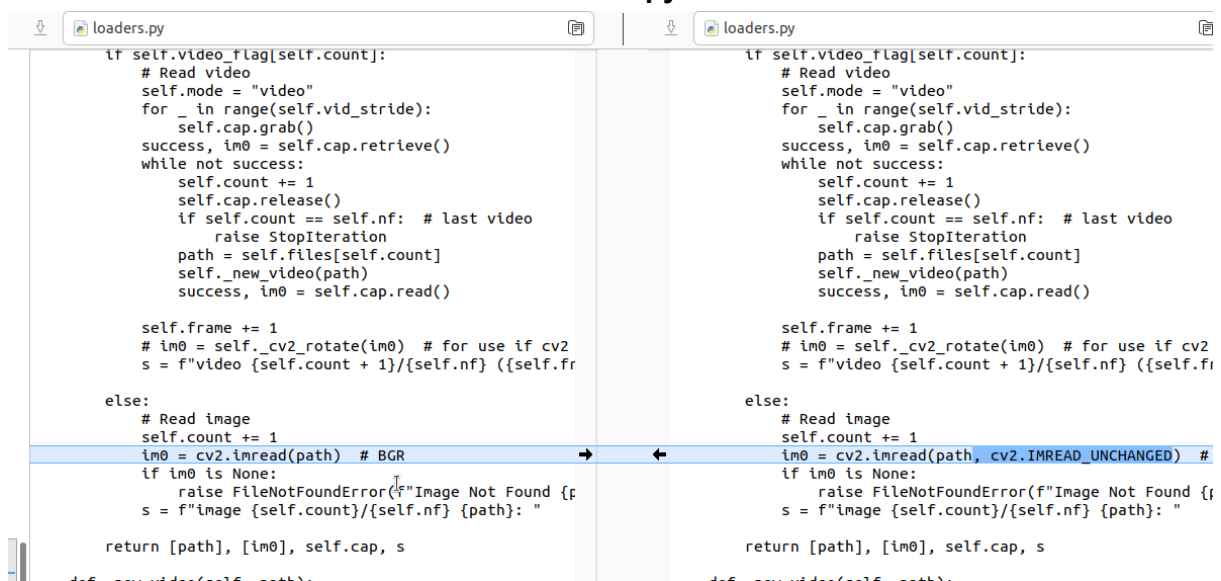
dataset.py:



crtl-c,ctrl-v new function:

```
def __getitem__(self, i):
    """Returns subset of data and targets corresponding
    f, j, fn, im = self.samples[i] # filename, index, filename.with_suffix('.npy'), image
    if self.cache_ram and im is None:
        im = self.samples[i][3] = cv2.imread(f, cv2.IMREAD_UNCHANGED)
    elif self.cache_disk:
        if not fn.exists(): # load npy
            np.save(fn.as_posix(), cv2.imread(f, cv2.IMREAD_UNCHANGED), allow_pickle=False)
            im = np.load(fn)
        else: # read image
            im = cv2.imread(f, cv2.IMREAD_UNCHANGED) # BGR
    # Convert NumPy array to PIL image
    im = Image.fromarray(cv2.cvtColor(im, cv2.COLOR_BGRA2RGBA))
    sample = self.torch_transforms(im)
    return {"img": sample, "cls": j}
```

loaders.py



split_dota.py

The image displays two side-by-side code editors, each showing a Python script named `split_dota.py`. The left editor shows the original code, while the right editor shows a modified version. The code defines a function `crop_and_save` that processes DOTA dataset annotations and images.

Left Editor (Original Code):

```
def crop_and_save(anno, windows, window_objs, im_dir, lb_dir):
    """
    Crop images and save new labels.

    Args:
        anno (dict): Annotation dict, including 'filepath',
        windows (list): A list of windows coordinates.
        window_objs (list): A list of labels inside each wr
        im_dir (str): The output directory path of images.
        lb_dir (str): The output directory path of labels.

    Notes:
        The directory structure assumed for the DOTA dataset
        - data_root
          - images
            - train
            - val
          - labels
            - train
            - val
    """
    im = cv2.imread(anno["filepath"])
    name = Path(anno["filepath"]).stem
    for i, window in enumerate(windows):
        x_start, y_start, x_stop, y_stop = window.tolist()
        new_name = f"{name}_{x_stop - x_start}_{x_start}_{y_stop - y_start}_{y_start}"
        patch_im = im[y_start:y_stop, x_start:x_stop]
        ph, pw = patch_im.shape[:2]
```

Right Editor (Modified Code):

```
def crop_and_save(anno, windows, window_objs, im_dir, lb_dir):
    """
    Crop images and save new labels.

    Args:
        anno (dict): Annotation dict, including 'filepath',
        windows (list): A list of windows coordinates.
        window_objs (list): A list of labels inside each wr
        im_dir (str): The output directory path of images.
        lb_dir (str): The output directory path of labels.

    Notes:
        The directory structure assumed for the DOTA dataset
        - data_root
          - images
            - train
            - val
          - labels
            - train
            - val
    """
    im = cv2.imread(anno["filepath"], cv2.IMREAD_UNCHANGED)
    name = Path(anno["filepath"]).stem
    for i, window in enumerate(windows):
        x_start, y_start, x_stop, y_stop = window.tolist()
        new_name = f"{name}_{x_stop - x_start}_{x_start}_{y_stop - y_start}_{y_start}"
        patch_im = im[y_start:y_stop, x_start:x_stop]
        ph, pw = patch_im.shape[:2]
```

The primary difference between the two versions is the addition of the `cv2.IMREAD_UNCHANGED` flag in the `cv2.imread` function call on the right, which ensures that the image is loaded in grayscale even if it is in color. The rest of the code, including the argument parsing, directory structure notes, and the patching logic, remains identical.

utils.py

utils.py

```
def new(str, optional): The path to the output image
max_dim (int, optional): The maximum dimension (width
quality (int, optional): The image compression quality

Example:
'''python
from pathlib import Path
from ultralytics.data.utils import compress_one_image

for f in Path('path/to/dataset').rglob('*.jpg'):
    compress_one_image(f)

'''

try: # use PIL
    im = Image.open(f)
    r = max_dim / max(im.height, im.width) # ratio
    if r < 1.0: # image too large
        im = im.resize((int(im.width * r), int(im.height * r)))
    im.save(f_new or f, "JPEG", quality=quality, optimize=True)
except Exception as e: # use OpenCV
    LOGGER.info(f"WARNING ⚠️ HUB ops PIL failure {f}: {e}")
    im = cv2.imread(f)
    im_height, im_width = im.shape[:2]
    r = max_dim / max(im_height, im_width) # ratio
    if r < 1.0: # image too large
        im = cv2.resize(im, (int(im_width * r), int(im_height * r)))
    cv2.imwrite(str(f_new or f), im)
```

utils.py

```
def new(str, optional): The path to the output image
max_dim (int, optional): The maximum dimension (width
quality (int, optional): The image compression quality

Example:
'''python
from pathlib import Path
from ultralytics.data.utils import compress_one_image

for f in Path('path/to/dataset').rglob('*.jpg'):
    compress_one_image(f)

'''

try: # use PIL
    im = Image.open(f)
    r = max_dim / max(im.height, im.width) # ratio
    if r < 1.0: # image too large
        im = im.resize((int(im.width * r), int(im.height * r)))
    im.save(f_new or f, "JPEG", quality=quality, optimize=True)
except Exception as e: # use OpenCV
    LOGGER.info(f"WARNING ⚠️ HUB ops PIL failure {f}: {e}")
    im = cv2.imread(f, cv2.IMREAD_UNCHANGED)
    im_height, im_width = im.shape[:2]
    r = max_dim / max(im_height, im_width) # ratio
    if r < 1.0: # image too large
        im = cv2.resize(im, (int(im_width * r), int(im_height * r)))
    cv2.imwrite(str(f_new or f), im)
```

engine

validator.py

```
validator.py
```

```
ing_size = check_ing_size(self.args.ing_size, stride=stride)
if engine:
    self.args.batch = model.batch_size
elif not pt and not jit:
    self.args.batch = 1 # export.py models default to batch=1
    LOGGER.info(f"Forcing batch=1 square inference {1,3},{1,3}")

if str(self.args.data).split(".")[-1] in ("yaml", "yml"):
    self.data = check_det_dataset(self.args.data)
elif self.args.task == "classify":
    self.data = check_cls_dataset(self.args.data, split=self.args.split)
else:
    raise FileNotFoundError(errno.ENOENT, f'Dataset "{self.args.data}" does not exist')

if self.device.type in ("cpu", "mps"):
    self.args.workers = 0 # faster CPU val as time dominated by IO
if not pt:
    self.args.rect = False
self.stride = model.stride # used in get_dataloader()
self.dataloader = self.dataloader or self.get_dataloader(self.data)

model.eval()
model.warmup(ing_size=(1 if pt else self.args.batch, 3, ing_size), stride=self.stride)

f.run_callbacks("on_val_start")
= (
    Profile(device=self.device),
    Profile(device=self.device),
    Profile(device=self.device),
    Profile(device=self.device),
)
```

```
model.warmup(imgsz=(1 if pt else self.args.batch, 4, imgsz, imgsz)) # warmup for 4 channels
```

predictor.py

```
anna on cl ~/.envs/venv_yolo4d_v3/lib/python3.10/site-packages/ultralytics/engine predictor.py
```

```
th self._lock: # for thread-safe inference
# Setup source every time predict is called
self.setup_source(source if source is not None else self.args.source)

# Check if save_dir/ label file exists
if self.args.save or self.args.save_txt:
    (self.save_dir / "labels" if self.args.save_txt else self.save_dir).mkdir(parents=True, exist_ok=True)

# Warmup model
if not self.done_warmup:
    self.model.warmup(imgsz=(1 if self.model.pt or self.model.triton else self.dataset.bs, 3, *self.imgsz))
    self.model.warmup(imgsz=(1 if self.model.pt or self.model.triton else self.dataset.bs, 4, *self.imgsz))
    self.done_warmup = True
```

nn

autobackend.py

```
autobackend.py
y[1] = np.transpose(y[1], (0, 3, 1, 2)) # s
y = [x if isinstance(x, np.ndarray) else x.numpy()

# for x in y:
#     print(type(x), len(x)) if isinstance(x, (list,
if isinstance(y, (list, tuple)):
    return self.from_numpy(y[0]) if len(y) == 1 else
else:
    return self.from_numpy(y)

def from_numpy(self, x):
    """
    Convert a numpy array to a tensor.

    Args:
        x (np.ndarray): The array to be converted.

    Returns:
        (torch.Tensor): The converted tensor
    """
    return torch.tensor(x).to(self.device) if isinstance

def warmup(self, imgsz=(1, 3, 640, 640)):
    """
    Warm up the model by running one forward pass with a

    Args:
        imgsz (tuple): The shape of the dummy input tens
    """
    warmup_images = self.at(self.it, self.conv, self.as

autobackend.py
y = [x if isinstance(x, np.ndarray) else x.numpy()

# for x in y:
#     print(type(x), len(x)) if isinstance(x, (list,
if isinstance(y, (list, tuple)):
    return self.from_numpy(y[0]) if len(y) == 1 else
else:
    return self.from_numpy(y)

def from_numpy(self, x):
    """
    Convert a numpy array to a tensor.

    Args:
        x (np.ndarray): The array to be converted.

    Returns:
        (torch.Tensor): The converted tensor
    """
    return torch.tensor(x).to(self.device) if isinstance

#def warmup(self, imgsz=(1, 3, 640, 640)):
def warmup(self, imgsz=(1, 4, 640, 640)):
    """
    Warm up the model by running one forward pass with a

    Args:
        imgsz (tuple): The shape of the dummy input tens
    """
    warmup_images = self.at(self.it, self.conv, self.as
```

```
def warmup(self, imgsz=(1, 4, 640, 640)):
```

tasks.py

```
def loss(self, batch, preds=None):
    """
    Compute loss.

    Args:
        batch (dict): Batch to compute loss on
        preds (torch.Tensor | List[torch.Tensor]): Predictions
    """
    if not hasattr(self, "criterion"):
        self.criterion = self.init_criterion()

    preds = self.forward(batch["img"]) if preds is None else preds
    return self.criterion(preds, batch)

def init_criterion(self):
    """Initialize the loss criterion for the BaseModel.
    raise NotImplementedError("compute_loss() needs to be implemented")"""

class DetectionModel(BaseModel):
    """YOLOv8 detection model."""

    def __init__(self, cfg="yolov8n.yaml", ch=3, nc=None, verbose=True):
        """Initialize the YOLOv8 detection model with the given config file.
        super().__init__()
        self.yaml = cfg if isinstance(cfg, dict) else yaml_load_file(cfg)

        # Define model
        ch = self.yaml.get("ch", ch) # number of channels
        if not isinstance(ch, int):
            raise ValueError(f"ch must be an integer, got {ch}")

        # Define model
        ch = 4# self.yaml.get("ch", ch) # number of channels
```

utils

plotting.py:

<pre># Build Image mosaic = np.full((int(ns * h), int(ns * w), 3), 255, dtype= for i in range(bs): x, y = int(w * (i // ns)), int(h * (i % ns)) # bloc mosaic[y : y + h, x : x + w, :] = images[i].transpose(1, 2, 0) # Resize (optional) scale = max_size / ns / max(h, w) if scale < 1: images = cv2.resize(images, None, fx=scale, fy=scale, interpolation=cv2.INTER_LINEAR)</pre>	<pre># Build Image mosaic = np.full((int(ns * h), int(ns * w), 3), 255, dtype= for i in range(bs): x, y = int(w * (i // ns)), int(h * (i % ns)) # bloc #mosaic[y : y + h, x : x + w, :] = images[i].transpose(1, 2, 0) (G, B, D, R) = cv2.split(images[i].astype(np.uint8)) merged = cv2.merge([R, G, B]) mosaic[y:y + h, x:x + w, :] = merged</pre>
---	--

change line:

`mosaic[y : y + h, x : x + w, :] = images[i].transpose(1, 2, 0)`

new:

```
#mosaic[y : y + h, x : x + w, :] = images[i].transpose(1, 2, 0)
(G, B, D, R) = cv2.split(images[i])
merged = cv2.merge([R, G, B])
mosaic[y:y + h, x:x + w, :] = merged
```

and:

```
/run/user/1000/gvfs:ftp:host=cl,user=anna/home/anna/.envs/venv_yolo4dv4/lib/python3.10/site-packages/ultralytics/utils/plotting.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
patches.py x plotting.py x utils.py x torch_utils.py x
805 if isinstance(masks, torch.Tensor):
806     masks = masks.cpu().numpy().astype(int)
807 if isinstance(kpts, torch.Tensor):
808     kpts = kpts.cpu().numpy()
809 if isinstance(batch_idx, torch.Tensor):
810     batch_idx = batch_idx.cpu().numpy()
811
812 max_size = 1920 # max image size
813 bs, _, h, w = images.shape # batch size, _, height, width
814 bs = min(bs, max_subplots) # limit plot images
815 ns = np.ceil(bs**0.5) # number of subplots (square)
816 if np.max(images[0]) <= 1:
817     images *= 255 # de-normalise (optional)
818
819 # Build Image
820 print('BATCH SIZE IS: ', bs)
821 mosaic = np.full((int(ns * h), int(ns * w), 3), 255, dtype=np.uint8) # init
822 #mosaic = np.full((int(ns * h), int(ns * w), 4), 255, dtype=np.uint8) # init
823 for i in range(bs):
824     images[i] = images[i].astype(np.uint8)
825
826     print('shape of image: ', images[i].shape)
827     print('type of image: ', images[i].dtype)
828     x, y = int(w * (i // ns)), int(h * (i % ns)) # block origin
829     #mosaic[y : y + h, x : x + w, :] = images[i].transpose(1, 2, 0)
830     #(G, B, D, R) = cv2.split(images[i].astype(np.uint8))
831     #print('shape of channels: ', G.shape, ' ', B.shape, ' ', D.shape, ' ', R.shape)
832     channels = cv2.split(images[i].astype(np.uint8))
833     print('channels: ', channels)
834     print('len channels: ', len(channels))
835     (G,B,R,D) = channels[0]
836     merged = cv2.merge([R, G, B])
837     mosaic[y:y + h, x:x + w, :] = merged
838
839 # Resize (optional)
```

IN THE YAML FILE I GIVE TO THE MODEL FOR TRAINING INFO:

```
lidar_forest_feb9_rgbd.yaml
path: /home/anna/vis_challenge_2024/data/YOLOtraindir_forest_LIDAR_feb9 # dataset root dir

train: rgbd_images/train/images # train images (relative to 'path') 1742289 images
val: rgbd_images/val/images # val images (relative to 'path') 80000 images
test: rgbd_images/val/images # test images (optional)

# Classes
names:
  0: box
  1: tree

ch: 4
```

IN CONCLUSION:

check all the *imread*-s, and *ch=3* and in *plotting.py* and *tensorboard.py* adjust the settings.

VSCoDe can list all *imreads* if you search in the side panel:

```
42 results - 13 files

data/base.py:
154         Path(fn).unlink(missing_ok=True)
155:         im = cv2.imread(f, cv2.IMREAD_UNCHANGED) # BGR
156     else: # read image
157:         im = cv2.imread(f, cv2.IMREAD_UNCHANGED) # BGR
158     if im is None:

202     if not f.exists():
203:         np.save(f.as_posix(), cv2.imread(self.im_files[i], cv2.IMREAD_UNCHANGED), allow_pickle=False)
204

209     for _ in range(n):
210:         im = cv2.imread(random.choice(self.im_files), cv2.IMREAD_UNCHANGED) # sample image
211         ratio = self.imgsz / max(im.shape[0], im.shape[1]) # max(h, w) # ratio

data/converter.py:
428         image_name_without_ext = image_path.stem
429:         img = cv2.imread(str(image_path), cv2.IMREAD_UNCHANGED)
430         h, w = img.shape[:2]

543         boxes[:, [1, 3]] *= h
544:         im = cv2.imread(l["im_file"], cv2.IMREAD_UNCHANGED)
545         sam_results = sam_model(im, bboxes=xywh2xyxy(boxes), verbose=False, save=False)

data/dataset.py:
452     if im is None: # Warning: two separate if statements required here, do not combine this with previous line
453:         im = self.samples[i][3] = cv2.imread(f, cv2.IMREAD_UNCHANGED)
454     elif self.cache_disk:
455:         if not fn.exists(): # load npy
456:             np.save(fn.as_posix(), cv2.imread(f, cv2.IMREAD_UNCHANGED), allow_pickle=False)
457:             im = np.load(fn)
458     else: # read image
459:         im = cv2.imread(f, cv2.IMREAD_UNCHANGED) # BGR
460         # Convert NumPy array to PIL image

data/loaders.py:
362         self.mode = "image"
363:         im0 = cv2.imread(path, cv2.IMREAD_UNCHANGED) # BGR
364         if im0 is None:

data/split_dota.py:
166     """
167:     im = cv2.imread(anno["filepath"], cv2.IMREAD_UNCHANGED)
168     name = Path(anno["filepath"]).stem

276     windows = get_windows((h, w), crop_sizes=crop_sizes, gaps=gaps)
277:     im = cv2.imread(im_file, cv2.IMREAD_UNCHANGED)
278     name = Path(im_file).stem

data/utils.py:
612     LOGGER.info(f"WARNING ⚠️ HUB ops PIL failure {f}: {e}")
613:     im = cv2.imread(f, cv2.IMREAD_UNCHANGED)
614     im_height, im_width = im.shape[:2]

data/explorer/explorer.py:
33         else: # read image
34:             im = cv2.imread(f, cv2.IMREAD_UNCHANGED) # BGR
35             if im is None:

data/explorer/utils.py:
82     for i, imf in enumerate(images):
83:         im = cv2.imread(imf, cv2.IMREAD_UNCHANGED)
84         print('HERE AT READING in data/explorer/utils.py', im.shape)

utils/_init_.py:
1089 # Apply monkey patches
1090: from .patches import imread, imshow, imwrite, torch_save
1091

1094     # Apply cv2 patches for non-ASCII and non-UTF characters in image paths
1095:     cv2.imread, cv2.imwrite, cv2.imshow = imread, imwrite, imshow

utils/patches.py:
14
15: #def imread(filename: str, flags: int = cv2.IMREAD_COLOR):
16: def imread(filename: str, flags: int = cv2.IMREAD_UNCHANGED):
17     """

21     filename (str): Path to the file to read.
22:     flags (int, optional): Flag that can take values of cv2.IMREAD_*. Defaults to cv2.IMREAD_COLOR.
```

```

23

utils/plotting.py:
758     xyxy = [50, 50, 150, 150]
759     im = cv2.imread('image.jpg')
760     cropped_im = save_one_box(xyxy, im, file='cropped.jpg', square=True)

utils/callbacks/clearml.py:
47
48:     img = mpimg.imread(plot_path)
49     fig = plt.figure()

utils/callbacks/neptune.py:
43
44:     img = mpimg.imread(plot_path)
45     fig = plt.figure()

```

In tasks.py-ban there are ch=3 defaults:

```

tasks.py - ultralytics [SSH: neumann] - Visual Studio Code

File Edit Selection View Go Run Terminal Help

SEARCH
OPEN EDITORS
  Search: imread
  Search: channel
  tasks.py nn
  plotting.py utils
  clearml.py utils/callbacks
ULTRALYTICS [SSH: NEUMANN]
  _pycache_
  assets
  cfg
  data
  engine
  hub
  models
  nn
  _pycache_
  modules
  _init_.py
  autobackend.py
  tasks.py
  solutions
  trackers
  utils
  _pycache_
  callbacks
  _pycache_
  _init_.py
  base.py
  clearml.py
  comet.py
  dvc.py
  hub.py
  milflow.py
  neptune.py
OUTLINE
TIMELINE
SEARCH
channel
Replace
160 results in 48 files - Open in editor

nn > tasks.py > BaseModel
74 class BaseModel(nn.Module):
240 def load(self, weights, verbose=True):
241     LOGGER.info(f"Transferred {len(csd)}/{len(self.model.state_dict())} items from pretrained weights")
242
243 def loss(self, batch, preds=None):
244     """
245     Compute loss.
246
247     Args:
248         batch (dict): Batch to compute loss on
249         preds (torch.Tensor | List[torch.Tensor]): Predictions.
250     """
251     if not hasattr(self, "criterion"):
252         self.criterion = self.init_criterion()
253
254     preds = self.forward(batch["img"]) if preds is None else preds
255     return self.criterion(preds, batch)
256
257 def init_criterion(self):
258     """Initialize the loss criterion for the BaseModel."""
259     raise NotImplementedError("compute_loss() needs to be implemented by task heads")
260
261 class DetectionModel(BaseModel):
262     """YOLOv8 detection model."""
263
264     def __init__(self, cfg="yolov8n.yaml", ch=3, nc=None, verbose=True): # model, input channels, number of classes
265         """Initialize the YOLOv8 detection model with the given config and parameters."""
266         super().__init__()
267         self.yaml = cfg if isinstance(cfg, dict) else yaml_model_load(cfg) # cfg dict
268
269         # Define model
270         ch = 4 # self.yaml.get("ch", ch) # input channels
271         if nc and nc != self.yaml["nc"]:
272             LOGGER.info(f"Overriding model.yaml nc={self.yaml['nc']} with nc={nc}")
273             self.yaml["nc"] = nc # override yaml value
274         self.model, self.save = parse_model(deepcopy(self.yaml), ch=ch, verbose=verbose) # model, savelist
275         self.names = {i: f"{i}" for i in range(self.yaml["nc"])} # default names dict
276         self.inplace = self.yaml.get("inplace", True)
277
278         # Build strides
279         m = self.model[-1] # Detect()
280         if isinstance(m, Detect): # includes all Detect subclasses like Segment, Pose, OBB, WorldDetect
281             s = 256 # 2x min stride
282             m.inplace = self.inplace
283             forward = lambda x: self.forward(x)[0] if isinstance(m, (Segment, Pose, OBB)) else self.forward(x)

```

Change in tensorboard.py:

```

tensorboard.py - ultralytics [SSH: neumann] - Visual Studio Code

File Edit Selection View Go Run Terminal Help

SEARCH
OPEN EDITORS
  Search: imread
  Search: channel
  tasks.py nn
  Search: tensorboard
  tensorboard.py utils/callbacks
  clearml.py utils/callbacks
ULTRALYTICS [SSH: NEUMANN]
  _pycache_
  callbacks
  _pycache_
  _init_.py
  base.py
  clearml.py
  comet.py
  dvc.py
  hub.py
  milflow.py
  neptune.py
  raytune.py
  tensorboard.py
  wb.py
  _init_.py

utils > callbacks > tensorboard.py > log_tensorboard_graph
21 except (ImportError, AssertionError, TypeError, AttributeError):
22     # TypeError for handling 'Descriptors cannot not be created directly.' protobuf errors in Windows
23     # AttributeError: module 'tensorflow' has no attribute 'io' if 'tensorflow' not installed
24     SummaryWriter = None
25
26 def log_scalars(scalars, step=0):
27     """Logs scalar values to TensorBoard."""
28     if WRITER:
29         for k, v in scalars.items():
30             WRITER.add_scalar(k, v, step)
31
32 def log_tensorboard_graph(trainer):
33     """Log model graph to TensorBoard."""
34
35     # Input image
36     imgsz = trainer.args.imgsz
37     imgsz = (imgsz, imgsz) if isinstance(imgsz, int) else imgsz
38     p = next(trainer.model.parameters()) # for device, type
39     im = torch.zeros((1, 3, *imgsz), device=p.device, dtype=p.dtype) # input image (must be zeros, not empty)
40     im = torch.zeros((1, 4, *imgsz), device=p.device, dtype=p.dtype) # input image (must be zeros, not empty)
41
42     with warnings.catch_warnings():
43         warnings.simplefilter("ignore", category=UserWarning) # suppress jit trace warning
44         warnings.simplefilter("ignore", category=torch.jit.TracerWarning) # suppress jit trace warning
45
46     # Try simple method first (YOLO)

```