My steps to modify YOLO to YOLO with 4 channels

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

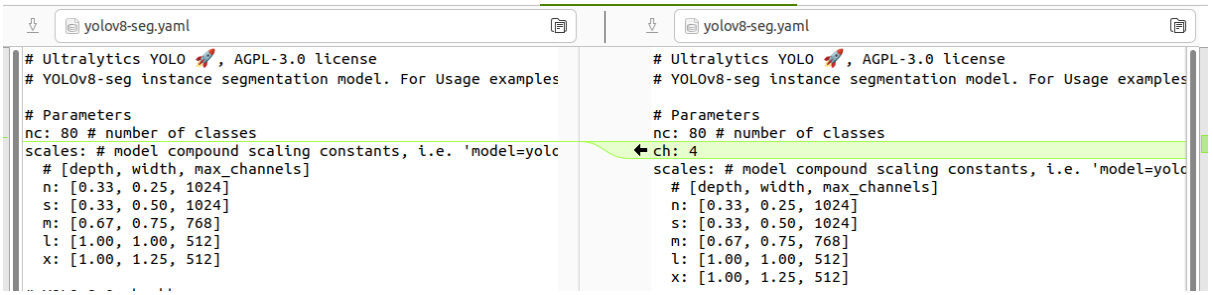
If you create a virtual env and would like to replace ultralytics with the modified directory:
apptainer exec --nv instance://yolo4dinst python3.10 -m virtualenv -p python3.10 $VENV

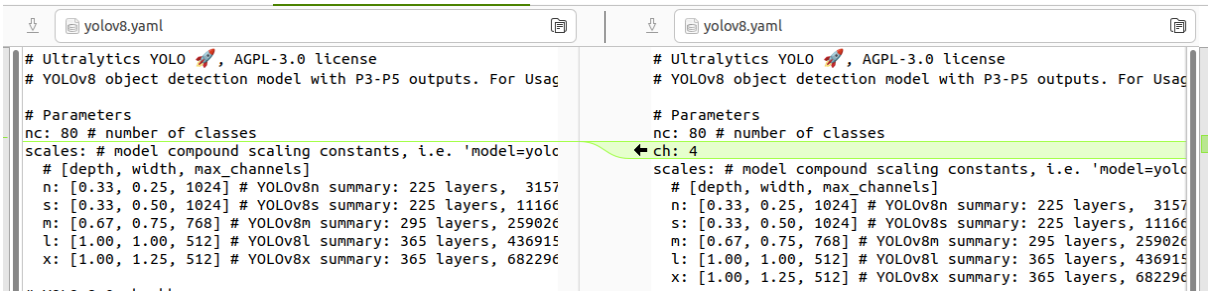
---

cfg/models/v8

---

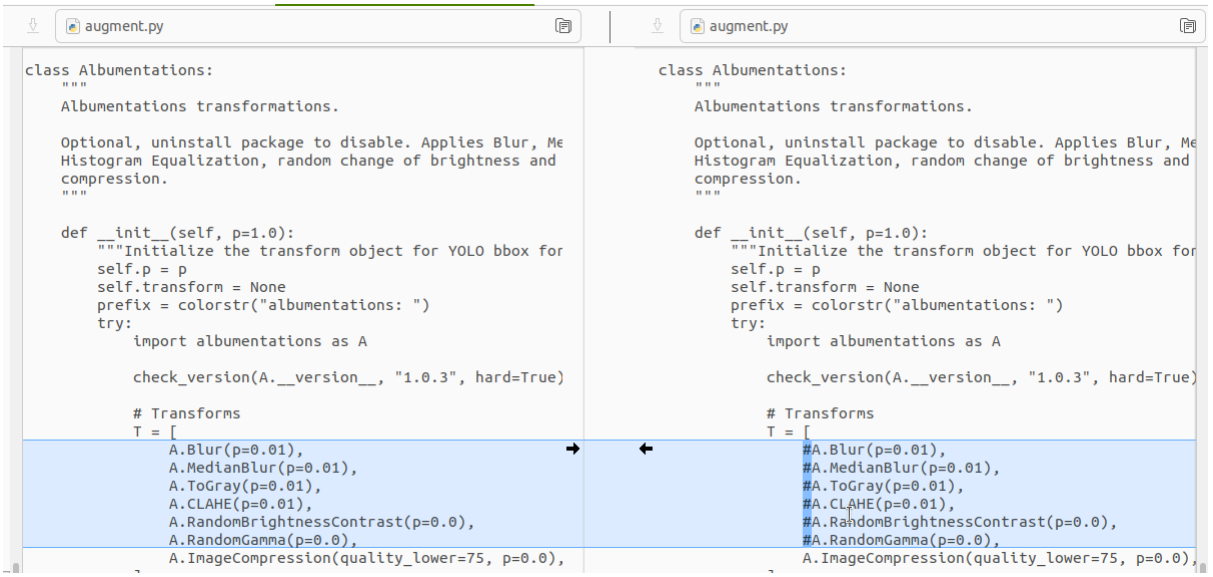
yolov8.yaml and yolov8-seg.yaml:





---

data



base.py

```
        single_cls (bool, optional): If True, single class t
        classes (list): List of included classes. Default is
        fraction (float): Fraction of dataset to utilize. De

    Attributes:
        im_files (list): List of image file paths.
        labels (list): List of label data dictionaries.
        ni (int): Number of images in the dataset.
        ims (list): List of loaded images.
        npy_files (list): List of numpy file paths.
        transforms (callable): Image transformation functior
    """
    def __init__(
        self,
        img_path,
        imgsz=640,
        cache=False,
        augment=True,
        hyp=DEFAULT_CFG,
        prefix="",
        rect=False,
        batch_size=16,
        stride=32,
        pad=0.5,
        single_cls=False,
        classes=None,
        fraction=1.0,
    ):
```

base.py

```
        single_cls (bool, optional): If True, single class t
        classes (list): List of included classes. Default is
        fraction (float): Fraction of dataset to utilize. De

    Attributes:
        im_files (list): List of image file paths.
        labels (list): List of label data dictionaries.
        ni (int): Number of images in the dataset.
        ims (list): List of loaded images.
        npy_files (list): List of numpy file paths.
        transforms (callable): Image transformation functior
    """
    #augment=True,
    def __init__(
        self,
        img_path,
        imgsz=640,
        cache=False,
        augment=False,
        hyp=DEFAULT_CFG,
        prefix="",
        rect=False,
        batch_size=16,
        stride=32,
        pad=0.5,
        single_cls=False,
        classes=None,
        fraction=1.0,
    ):
```

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

```
            continue
        class_name = parts[8]
        class_idx = class_mapping[class_name]
        coords = [float(p) for p in parts[:8]]
        normalized_coords = [
            coords[i] / image_width if i % 2 == 0 else coor
        ]
        formatted_coords = ["{:.6g}".format(coord) for coor
        g.write(f"{class_idx} {' '.join(formatted_coords)}\

phase in ["train", "val"]:
image_dir = dota_root_path / "images" / phase
orig_label_dir = dota_root_path / "labels" / f"{phase}_orig
save_dir = dota_root_path / "labels" / phase

save_dir.mkdir(parents=True, exist_ok=True)

image_paths = list(image_dir.iterdir())
for image_path in TQDM(image_paths, desc=f"Processing {phas
    if image_path.suffix != ".png":
        continue
    image_name_without_ext = image_path.stem
    img = cv2.imread(str(image_path))
    h, w = img.shape[:2]
    convert_label(image_name_without_ext, w, h, orig_label_

_index(arr1, arr2):
```

converter.py

```
            continue
        class_name = parts[8]
        class_idx = class_mapping[class_name]
        coords = [float(p) for p in parts[:8]]
        normalized_coords = [
            coords[i] / image_width if i % 2 == 0 else coor
        ]
        formatted_coords = ["{:.6g}".format(coord) for coor
        g.write(f"{class_idx} {' '.join(formatted_coords)}\

phase in ["train", "val"]:
image_dir = dota_root_path / "images" / phase
orig_label_dir = dota_root_path / "labels" / f"{phase}_orig
save_dir = dota_root_path / "labels" / phase

save_dir.mkdir(parents=True, exist_ok=True)

image_paths = list(image_dir.iterdir())
for image_path in TQDM(image_paths, desc=f"Processing {phas
    if image_path.suffix != ".png":
        continue
    image_name_without_ext = image_path.stem
    img = cv2.imread(str(image_path), cv2.IMREAD_UNCHANGED)
    h, w = img.shape[:2]
    convert_label(image_name_without_ext, w, h, orig_label_

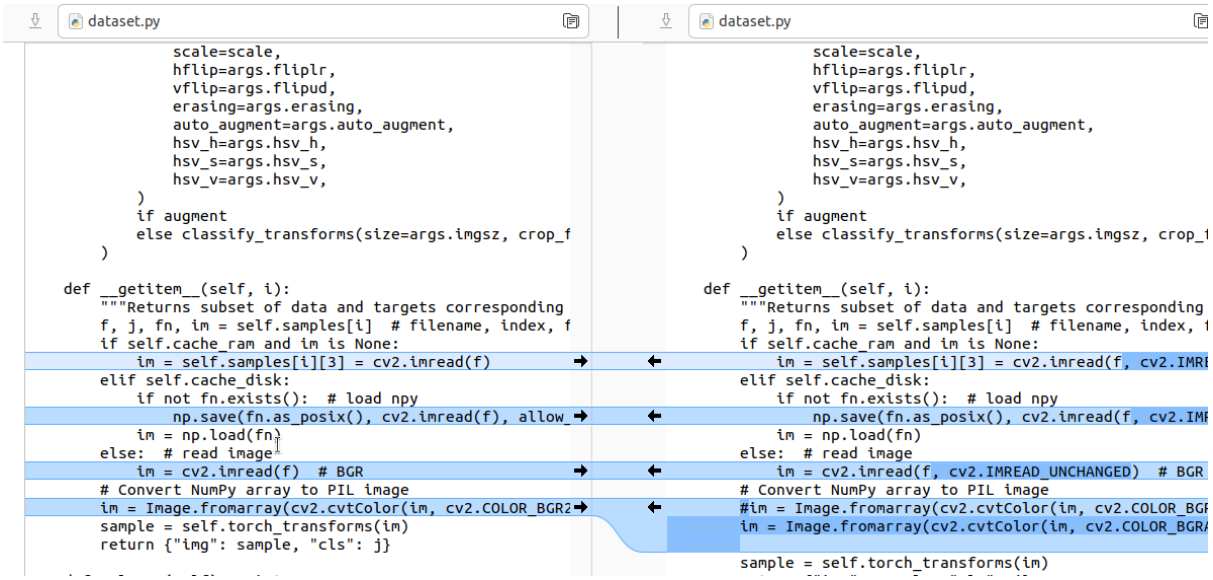
_index(arr1, arr2):
```

converter.py:

plotting.py | annotator.py | augment.py | base.py | build.py | dataset.py | converter.py

```
520                 └─ NNN.txt
521         """
522         from tqdm import tqdm
523
524         from ultralytics import SAM
525         from ultralytics.data import YOLODataset
526         from ultralytics.utils import LOGGER
527         from ultralytics.utils.ops import xywh2xyxy
528
529         # NOTE: add placeholder to pass class index check
530         dataset = YOLODataset(im_dir, data=dict(names=list(range(1000))))
531         if len(dataset.labels[0]["segments"]) > 0:  # if it's segment data
532             LOGGER.info("Segmentation labels detected, no need to generate new ones!")
533             return
534
535         LOGGER.info("Detection labels detected, generating segment labels by SAM model!")
536         sam_model = SAM(sam_model)
537         for l in tqdm(dataset.labels, total=len(dataset.labels), desc="Generating segment labels"):
538             h, w = l["shape"]
539             boxes = l["bboxes"]
540             if len(boxes) == 0:  # skip empty labels
541                 continue
542             boxes[:, [0, 2]] *= w
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)
546             l["segments"] = sam_results[0].masks.xyn
547
548         save_dir = Path(save_dir) if save_dir else Path(im_dir).parent / "labels-segment"
549         save_dir.mkdir(parents=True, exist_ok=True)
```

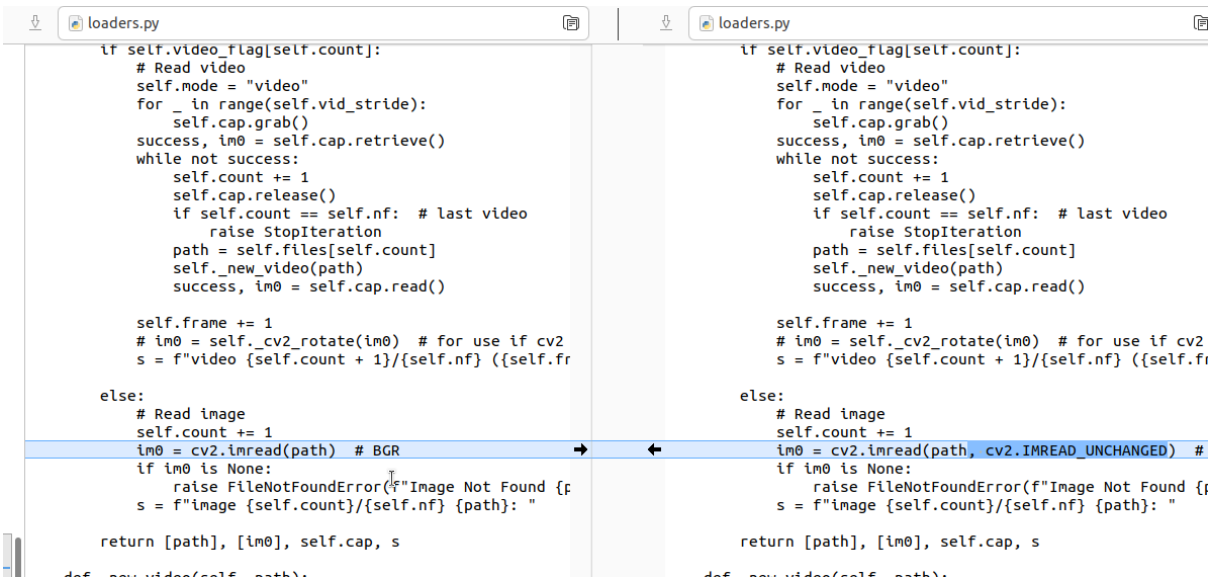
dataset.py:



crtl-c,crtl-v new function:

```
def __getitem__(self, i):
        """Returns subset of data and targets corresponding to given indices."""
        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

split_dota.py

```
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 wir
        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}__
        patch_im = im[y_start:y_stop, x_start:x_stop]
        ph, pw = patch_im.shape[:2]
```

split_dota.py

```
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 wir
        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}__
        patch_im = im[y_start:y_stop, x_start:x_stop]
        ph, pw = patch_im.shape[:2]
```

utils.py

utils.py

```
        max_dim (int, optional): The maximum dimension (widt
        quality (int, optional): The image compression quali

    Example:
        ```python
        from pathlib import Path
        from ultralytics.data.utils import compress_one_imag

        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
        im.save(f_new or f, "JPEG", quality=quality, optimiz
    except Exception as e: # use OpenCV
        LOGGER.info(f"WARNING ⚠️ HUB ops PIL failure {f}: {
        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_h
        cv2.imwrite(str(f_new or f), im)
```

utils.py

```
        max_dim (int, optional): The maximum dimension (widt
        quality (int, optional): The image compression quali

    Example:
        ```python
        from pathlib import Path
        from ultralytics.data.utils import compress_one_imag

        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
        im.save(f_new or f, "JPEG", quality=quality, optimiz
    except Exception as e: # use OpenCV
        LOGGER.info(f"WARNING ⚠️ HUB ops PIL failure {f}: {
        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_h
        cv2.imwrite(str(f_new or f), im)
```

---

engine

---

validator.py

validator.py

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

    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=sel
    else:
        raise FileNotFoundError(emojis(f"Dataset '{self.args.da

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

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

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

validator.py

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

    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=se
    else:
        raise FileNotFoundError(emojis(f"Dataset '{self.args.da

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

    model.eval()
    #model.warmup(imgsz=(1 if pt else self.args.batch, 3, imgs
    model.warmup(imgsz=(1 if pt else self.args.batch, 4, imgsz

    f.run_callbacks("on_val_start")
    = (
        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

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

Save

```
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
        """
```

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
        """
```

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

tasks.py

tasks.py

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

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

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

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


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

    def __init__(self, cfg="yolov8n.yaml", ch=3, nc=None, ve
        """Initialize the YOLOv8 detection model with the gi
        super().__init__()
        self.yaml = cfg if isinstance(cfg, dict) else yaml_r

        # Define model
        ch = self.yaml["ch"] = self.yaml.get("ch", ch)  # ir
```

tasks.py

```
        """
        Compute loss.

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

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

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

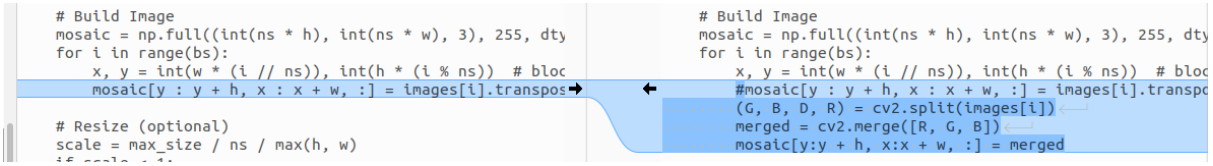

class DetectionModel(BaseModel):
    """YOLOv8 detection model."""

    #def __init__(self, cfg="yolov8n.yaml", ch=3, nc=None, v
    def __init__(self, cfg="yolov8n.yaml", ch=4, nc=None, ve
        """Initialize the YOLOv8 detection model with the gi
        super().__init__()
        self.yaml = cfg if isinstance(cfg, dict) else yaml_r

        # Define model
        ch = 4# self.yaml["ch"] = self.yaml.get("ch", ch)  #
```

utils

plotting.py:



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/sftp: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 | plotting.py | utils.py | torch_utils.py

```
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
```

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.
```

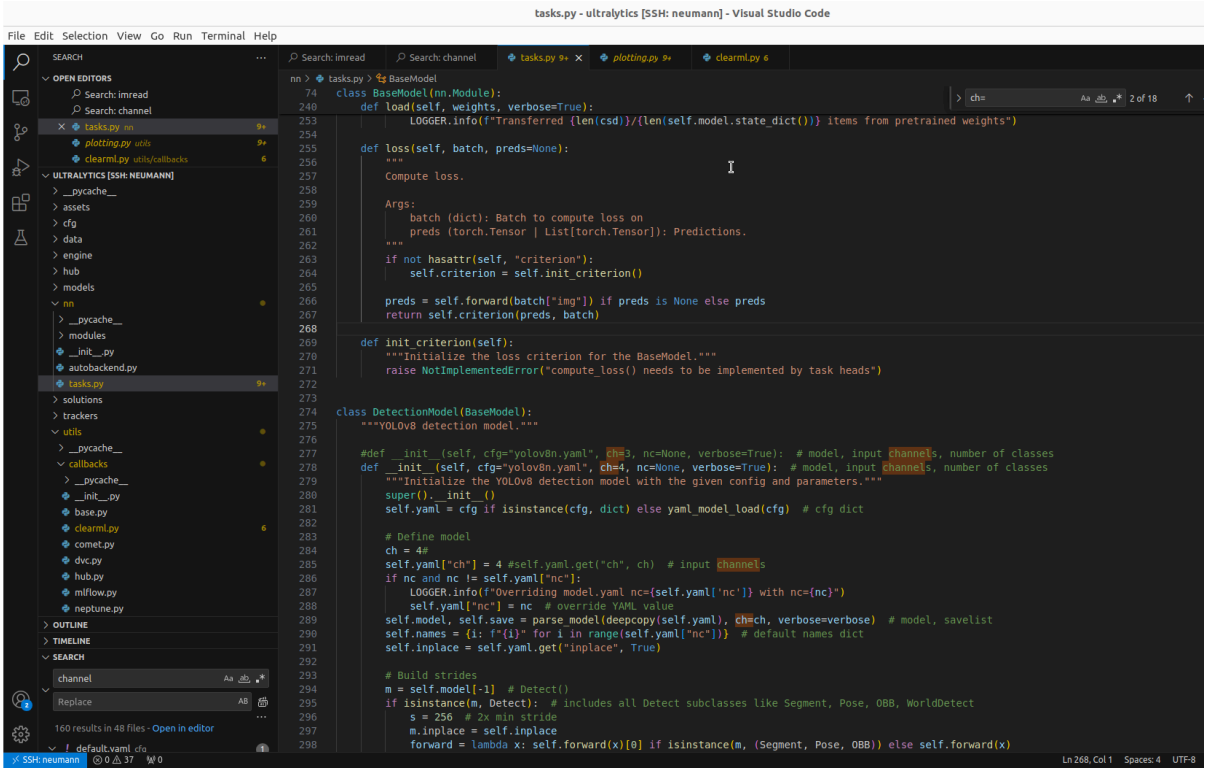
```
23

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:



Change in tensorboard.py: