yolo框架的修改说明

框架改进

yolo11与改进backbone区别：

```yaml
scales: # model compound scaling constants
  # [depth, width, max_channels]
  n: [0.50, 0.25, 1024] # summary: 319 lay
  s: [0.50, 0.50, 1024] # summary: 319 lay
  m: [0.50, 1.00, 512] # summary: 409 laye
  l: [1.00, 1.00, 512] # summary: 631 laye
  x: [1.00, 1.50, 512] # summary: 631 laye

# YOLO11n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 2, C3k2, [256, False, 0.25]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 2, C3k2, [512, False, 0.25]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 2, C3k2, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 2, C3k2, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9
  - [-1, 2, C2PSA, [1024]] # 10
```

```yaml
scales: # [depth, width, max_channels]
  n: [0.50, 0.25, 512]
  s: [0.50, 0.50, 512]
  m: [0.50, 0.75, 384]
  l: [1.00, 0.75, 384]
  x: [1.00, 1.00, 384]

# YOLO11n backbone (P2-focused)
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [48, 3, 2]]   # 0-P1/2
  - [-1, 1, Conv, [96, 3, 2]]   # 1-P2/4

  # P2: 核心高频层，保持WDC能力
  - [-1, 1, C3k2_WDC, [192, False, 0.25]]  # 2

  - [-1, 1, Conv, [192, 3, 2]]  # 3-P3/8
  - [-1, 1, C3k2_WDC, [320, False, 0.25]]  # 4

  - [-1, 1, Conv, [320, 3, 2]]  # 5-P4/16
  - [-1, 1, C3k2_Faster, [320, True, 0.25]]  # 6

  - [-1, 1, Conv, [512, 3, 2]]  # 7-P5/32
  - [-1, 1, C3k2_Faster, [512, True, 0.25]]  # 8

  - [-1, 1, SPPF, [512, 5]]   # 9
  - [-1, 1, C2PSA, [512]]     # 10
```

yolo11与改进neck+head区别：

```yaml
# YOLO11n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 2, C3k2, [512, False]] # 13

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 2, C3k2, [256, False]] # 16 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 13], 1, Concat, [1]] # cat head P4
  - [-1, 2, C3k2, [512, False]] # 19 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 10], 1, Concat, [1]] # cat head P5
  - [-1, 2, C3k2, [1024, True]] # 22 (P5/32-large)

  - [[16, 19, 22], 1, Detect, [nc]] # Detect(P3, P4, P5)
```

```yaml
# YOLO11n head (P2-focused FPN)
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]          # 11
  - [-1, 1, LGTConvNeckLite, [192, 192, 3, 1, 1]]        # 12
  - [[-1, 6], 1, Concat, [1]]                            # 13
  - [-1, 1, C3k2_Faster, [320, False, 0.25]]             # 14

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]          # 15
  - [-1, 1, LGTConvNeckLite, [128, 128, 3, 1, 1]]        # 16
  - [[-1, 4], 1, Concat, [1]]                            # 17
  - [-1, 1, C3k2_Faster, [192, False, 0.25]]             # 18

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]          # 19
  - [-1, 1, LGTConvNeckLite_WT, [96, 96, 3, 1, 1]]       # 20
  - [[-1, 2], 1, Concat, [1]]                            # 21
  - [-1, 1, C3k2_WDC, [160, False, 0.25]]                # 22

  - [-1, 1, Conv, [160, 3, 2]]                           # 23
  - [[-1, 18], 1, Concat, [1]]                           # 24
  - [-1, 1, C3k2_Faster, [192, False, 0.25]]             # 25

  # Detect(P2,P3): Haar-DWT HF Gate on P2 regression + LSDECD-lite shared conv
  - [[22, 25], 1, Detect_LSDECD_DWTP2GateLite, [nc, 192]]
```

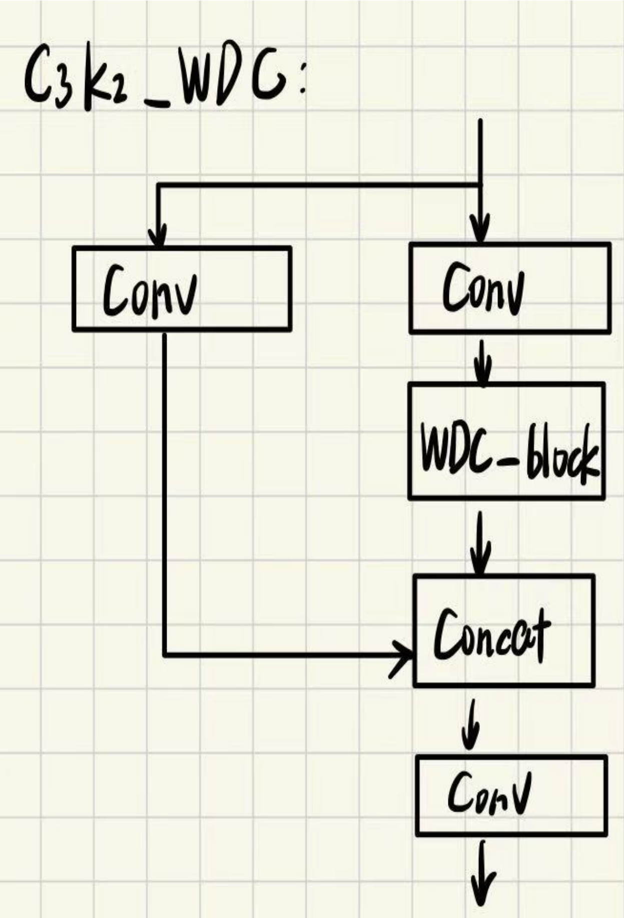
总体改进说明：

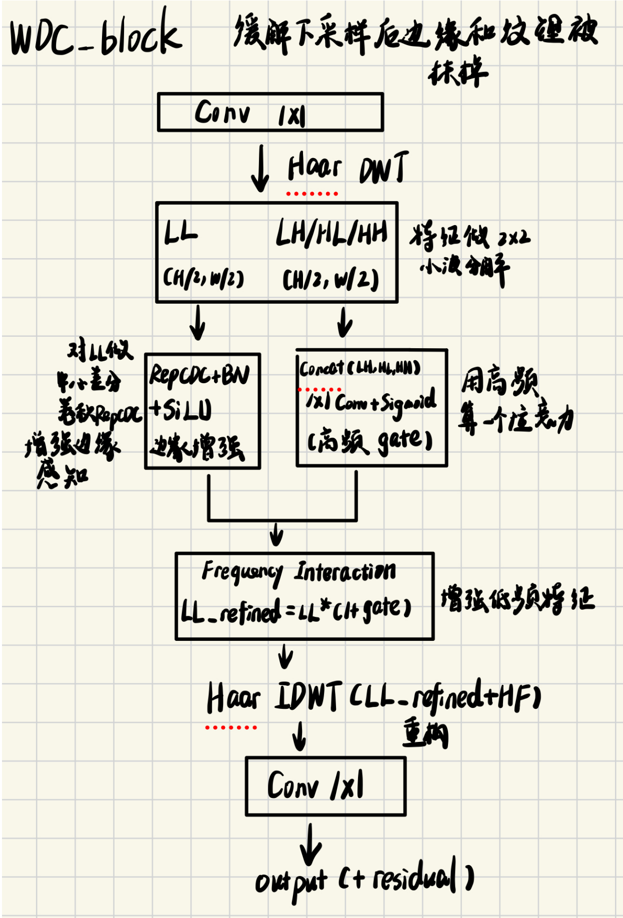
- Backbone：与P2，P3检测头相关的C3k2改成C3k2_WDC,其他改成C3k2_Faster
- Neck:在每次upsample之后，添加LTConvNeckLite/LTConvNeckLite_WT做高频特征增强

- head：从P3、P4、P5修改成P2、P3的改进；使用Detect_LSDECD_DWTP2GateLite
- 整体的通道数都做了缩减

每个模块的具体结构

C3k2_WDC





- WDC-Block 首先通过一个 1×1 卷积对输入特征进行通道投影，并利用 Haar 小波变换将特征显式分解为低频与高频分量
- 低频分支采用重参数化中心差分卷积（RepCDC）以增强边缘感知能力；高频分量则被融合以生成门控权重，用于刻画纹理与边缘区域的重要性
- 随后，通过频率交互机制，高频门控自适应地调制低频特征，从而突出关键结构信息
- 经逆 Haar 小波变换重构特征，并通过 1×1 卷积与可选残差连接生成输出特征。

> **The WDC-Block** first projects the input feature using a 1×1 convolution and explicitly decomposes it into low- and high-frequency components via the Haar wavelet transform.
> The low-frequency branch is enhanced by a re-parameterized central difference convolution (RepCDC) to improve edge sensitivity, while the high-frequency components are fused to generate a gating map that captures texture and edge importance.
> Through a frequency interaction mechanism, the high-frequency gate adaptively modulates the low-frequency features to emphasize critical structural information.
> Finally, the refined features are reconstructed by the inverse Haar wavelet transform, followed by a 1×1 convolution and an optional residual connection to produce the output feature.

C3k2_Faster（非原创）

C3k2，但把bottleneck换成更轻量的 C3k_Faster

Partial Conv / Partial Spatial Mixing：只对**部分通道**做 3×3 空间卷积，其余通道走更轻路径

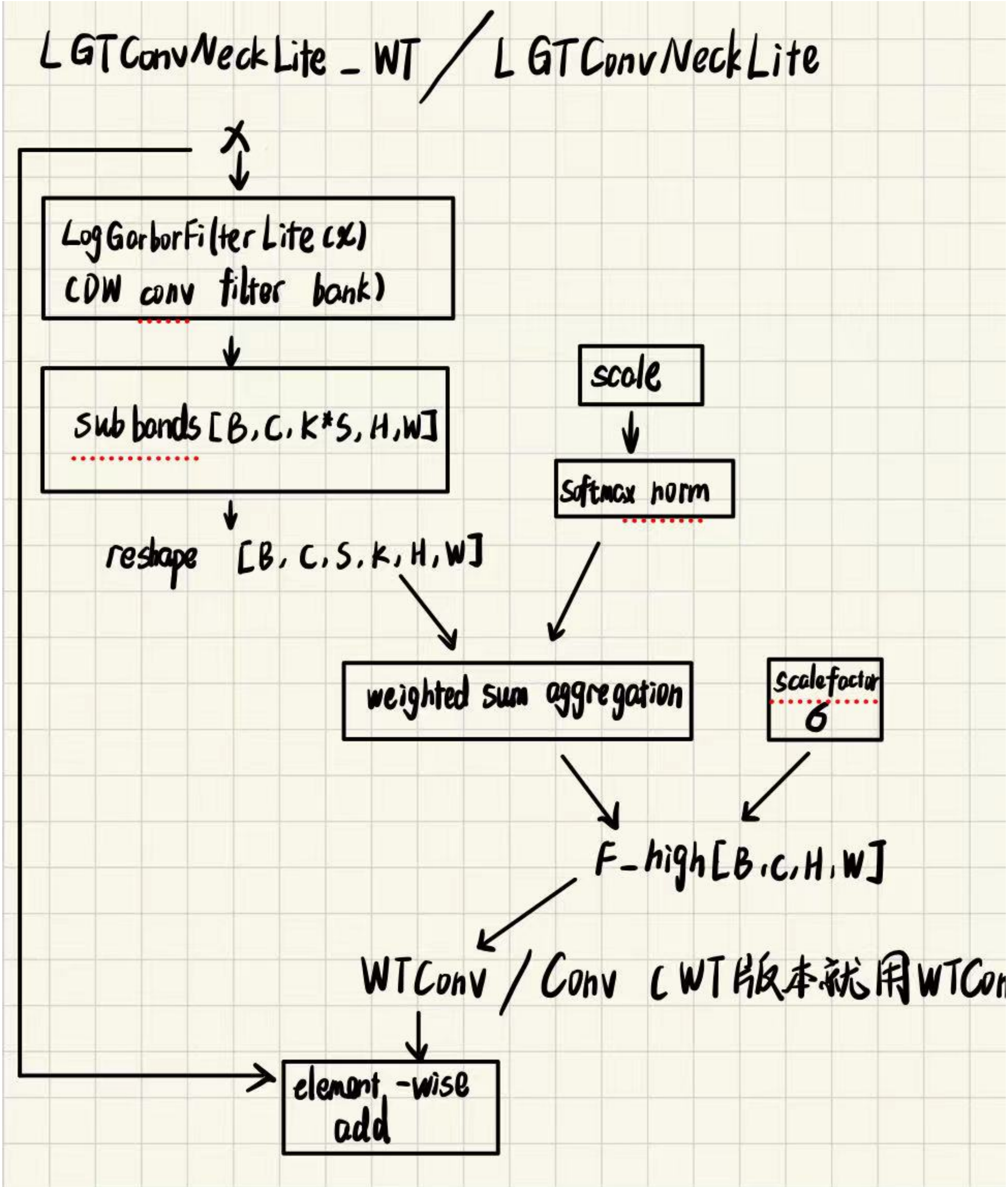
MLP 用 1×1 卷积做通道混合：1×1 卷积很便宜，再配合残差连接实现高效表达。

比较：

C3k2_Faster：工程效率导向——“更快、更省算力”。

C3k2_WDC：表征/任务导向——“显式频域（小波）+ 边缘增强，更偏小目标细节”。

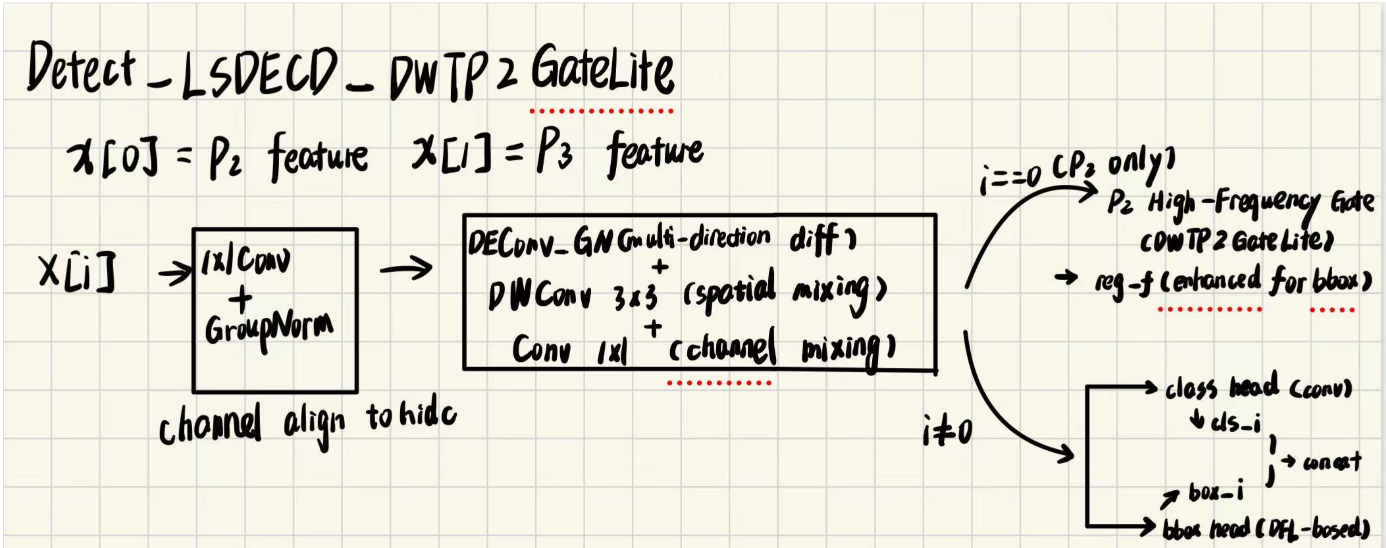
LGTConvNeckLite_WT/LGTConvNeckLite

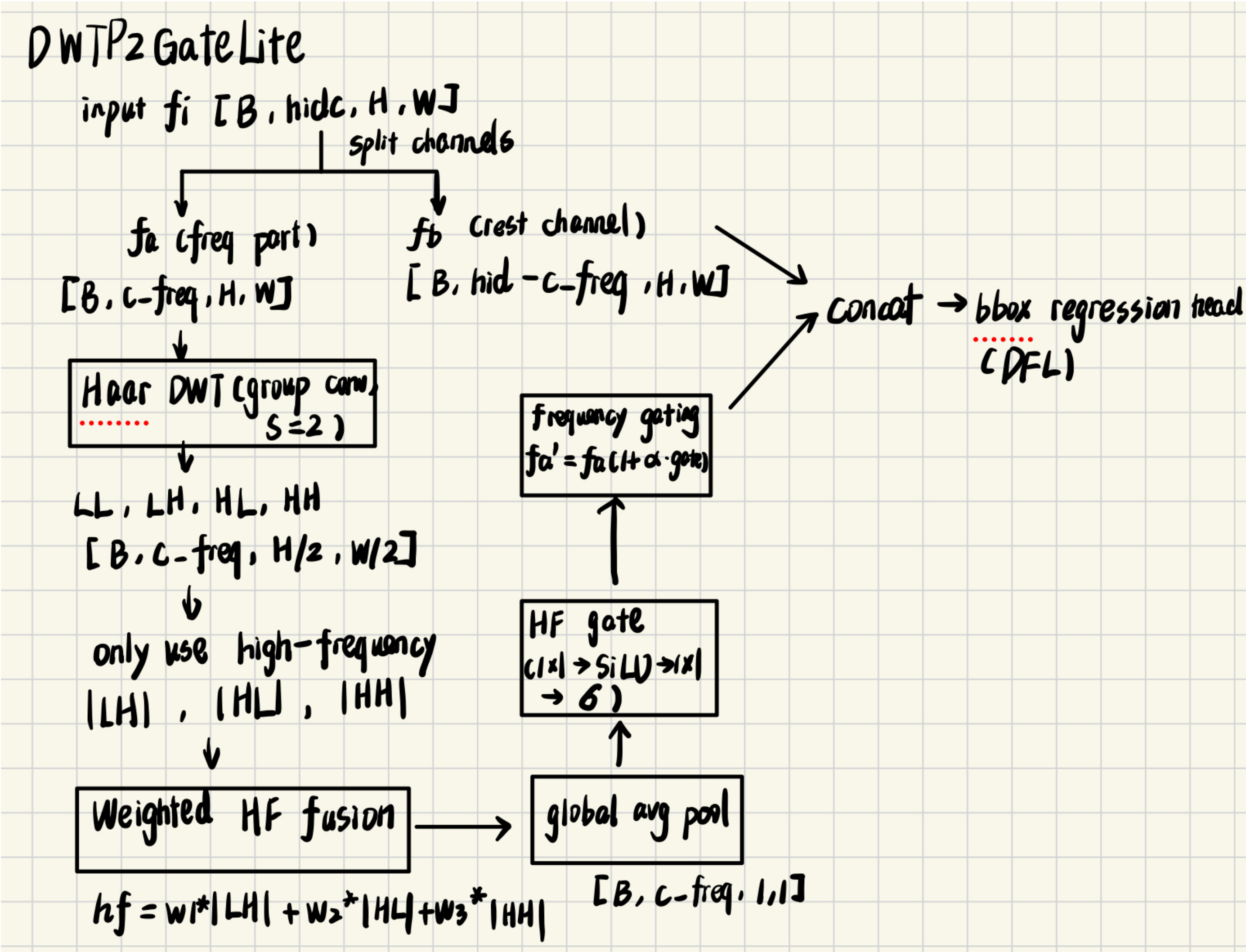


- LGTConvNeckLite(-WT) 用于 YOLO Neck 中的特征增强
- 利用 Log-Gabor 滤波器组对输入特征进行方向与尺度敏感的高频分解，生成多个子带响应
- 通过可学习的方向权重和尺度权重，对不同子带进行加权聚合，形成统一的高频特征表示，并使用一个标量缩放因子对高频增强强度进行调制
- 聚合后的高频特征经 WTConv（或普通卷积)进一步处理，并以残差形式与输入特征相加，保持原始结构信息且增强高频细节

> **LGTConvNeckLite(-WT)** is a lightweight high-frequency enhancement module designed for feature refinement in the YOLO neck.
> It first applies a Log-Gabor filter bank to decompose the input feature into orientation- and scale-sensitive high-frequency subbands.
> Learnable orientation and scale weights are then used to aggregate these subbands via weighted summation, followed by a scalar gating factor to control the enhancement strength.
> The aggregated high-frequency feature is further processed by a WT-based convolution (or a standard convolution when channel alignment is required) and added back to the input through a residual connection.
> This design efficiently enhances texture and edge information while maintaining low computational overhead, making it suitable for lightweight multi-scale feature fusion.

Detect_LSDECD_DWTP2GateLite





- **LSDECD-lite shared conv**：每个尺度先经过 1×1 对齐通道，再走一套共享的增强卷积（包含 DEConv + DWConv + 1×1）
- **仅对 P2 的 bbox 回归做 Haar-DWT 高频门控（DWTP2GateLite）**：用高频响应产生 gate，去增强回归特征的部分通道

> **Detect_LSDECD_DWTP2GateLite** employs shared feature enhancement and introduces a Haar-DWT based high-frequency gating mechanism exclusively on the P2 regression branch, adaptively strengthening edge-aware localization for small objects while preserving semantic stability for classification.