Residual Local Feature Network for Efficient Super-Resolution

Fangyuan Kong*, Mingxi Li*, Songwei Liu*, Ding Liu, Jingwen He,
Yang Bai, Fangmin Chen, Lean Fu

ByteDance Inc

Motivation: What is Efficient

Trade-off: model complexity vs restoration quality

Runtime $\neq$ Parameters / FLOPs / Activations

Motivation: What is Efficient

Runtime $\neq$ Parameters
FLOPs
Activations

Focus of Our Work: Runtime Optimization

RLFN: 1st place winner in the main track of NTIRE 2022 efficient super-resolution challenge.

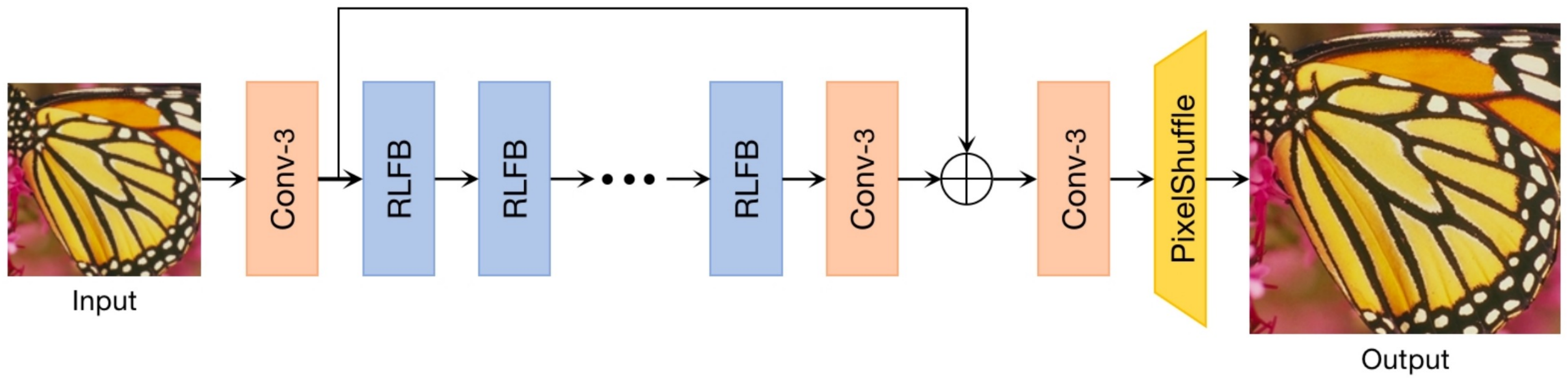
- Residual local feature block —— to speed up runtime
- Novel feature extractor of contrastive loss
- Warm-start training strategy

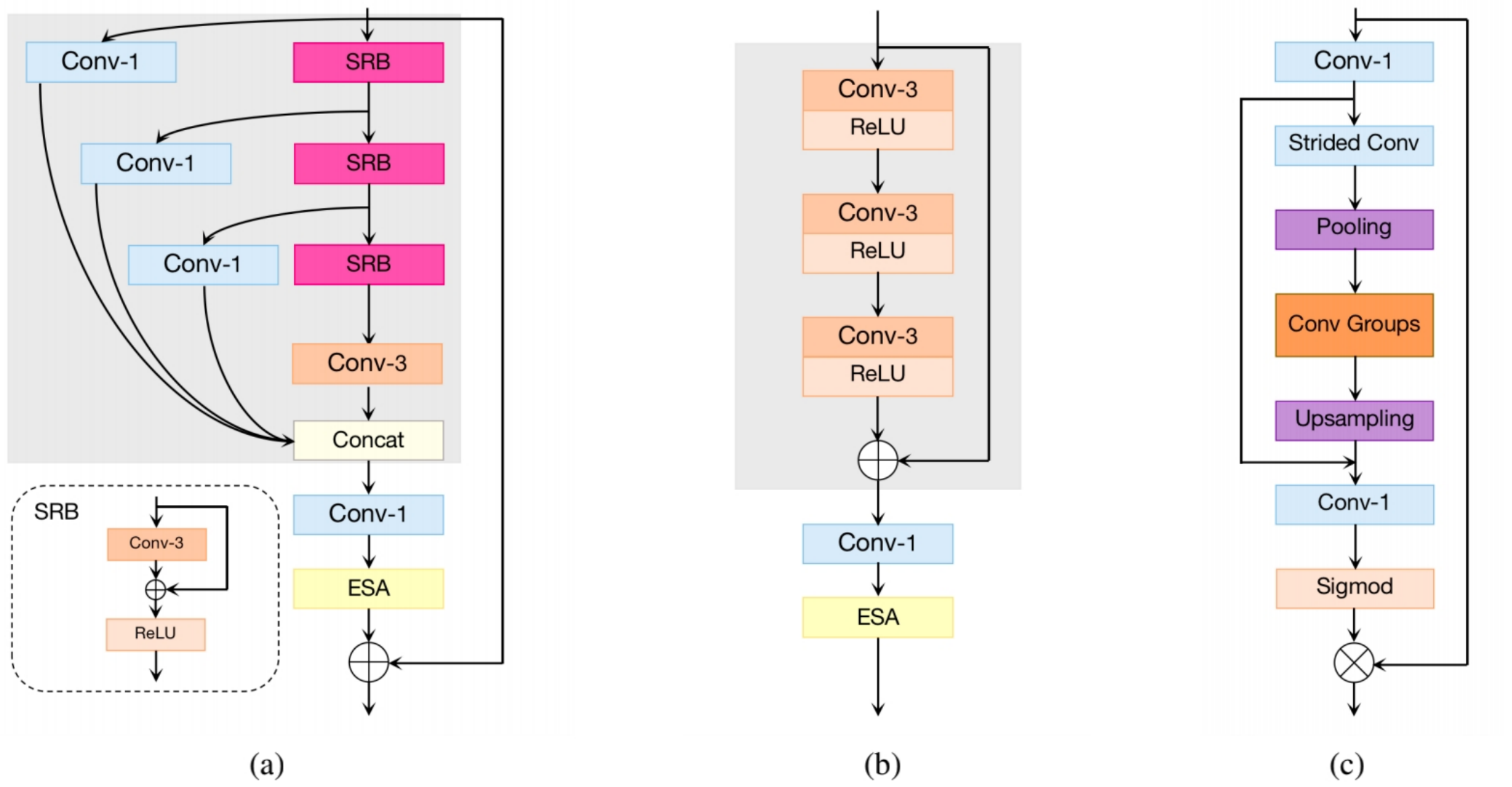
(Novel feature extractor of contrastive loss; Warm-start training strategy) to boost the SR performance

ByteDance字节跳动

Network Architecture

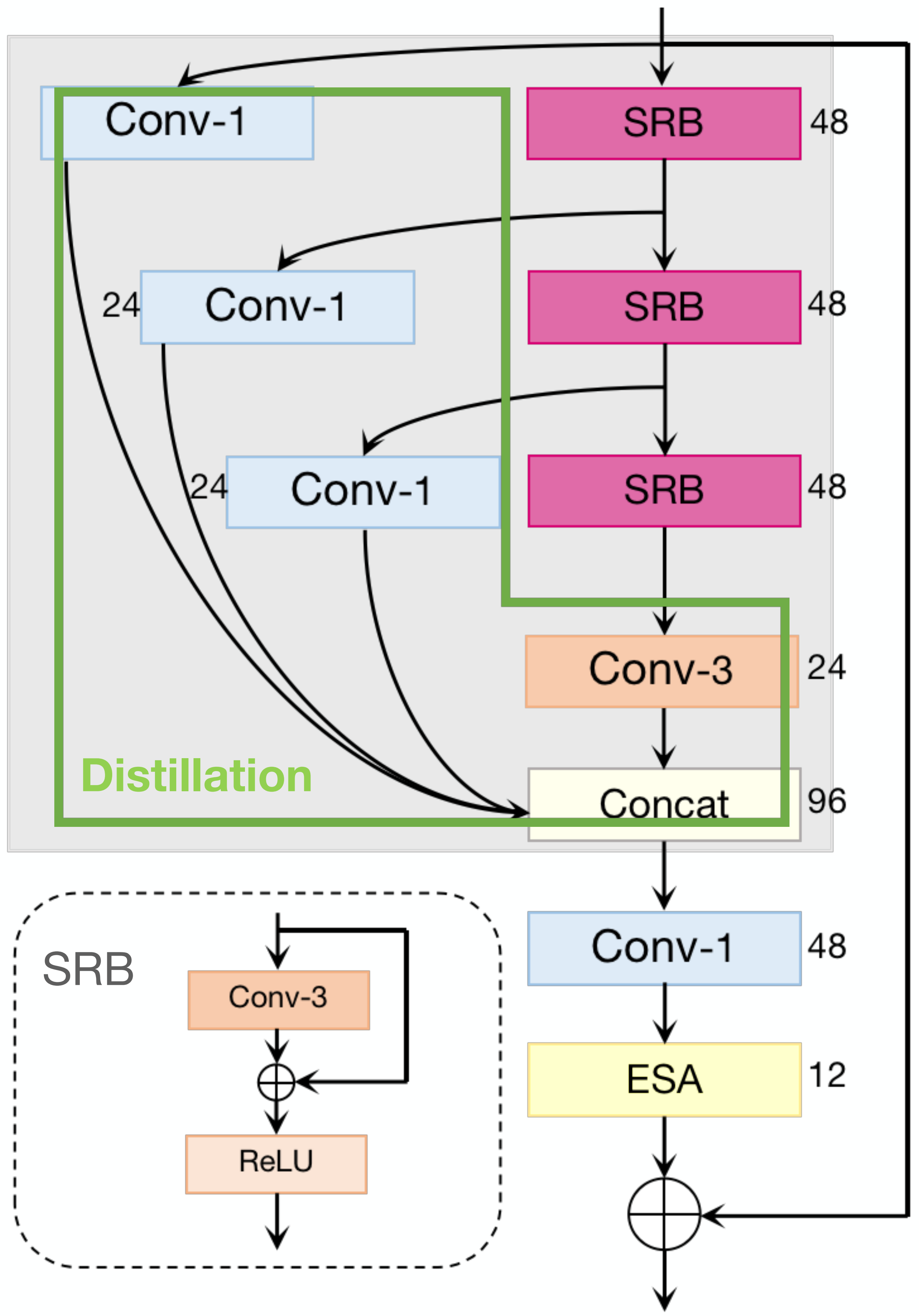
Network Architecture





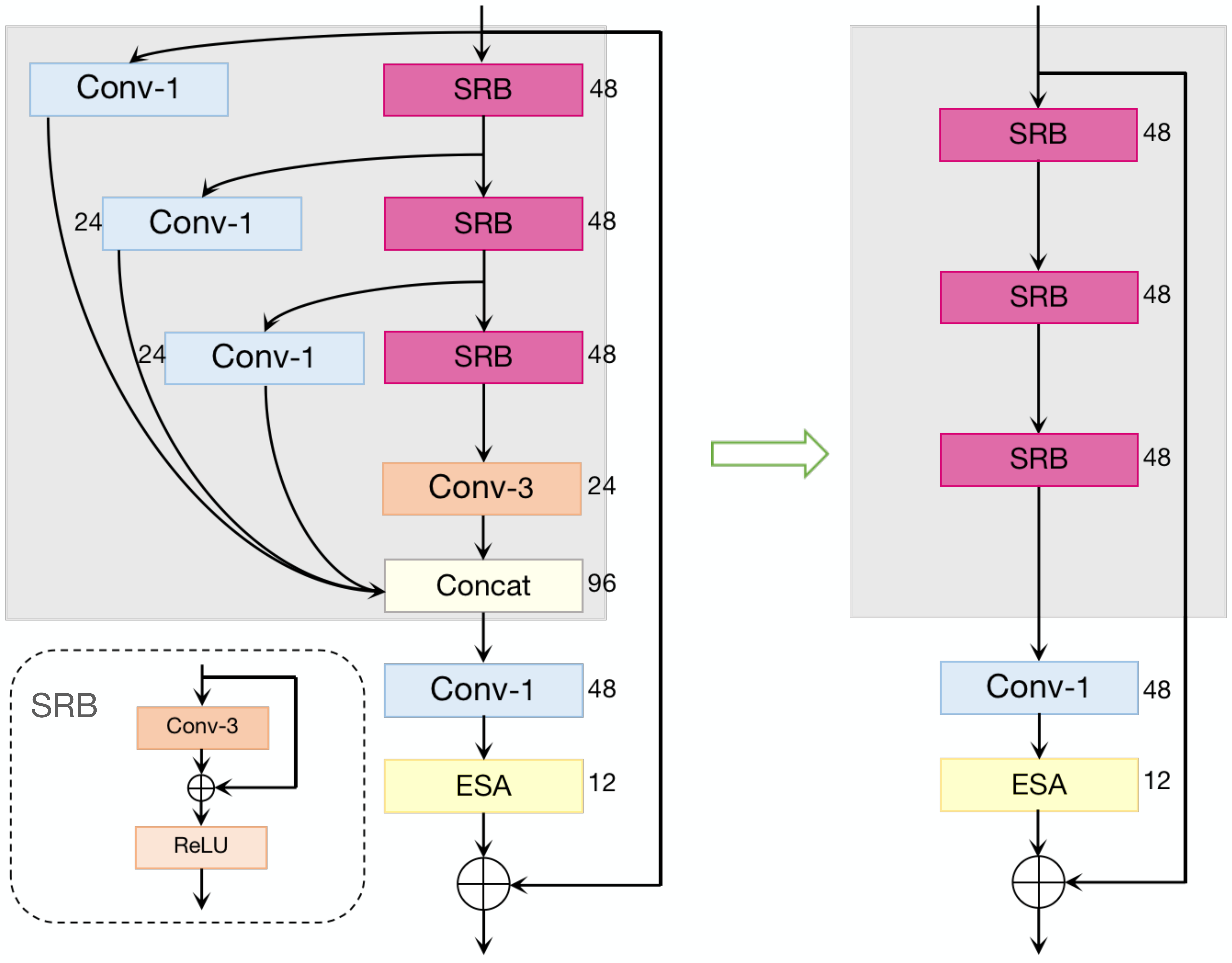
(a) RFDB: residual feature distillation block. (b) RLFB: residual local feature block. (c) ESA: Enhanced Spatial Attention.

Residual Local Feature Block



RFDB
82.3ms

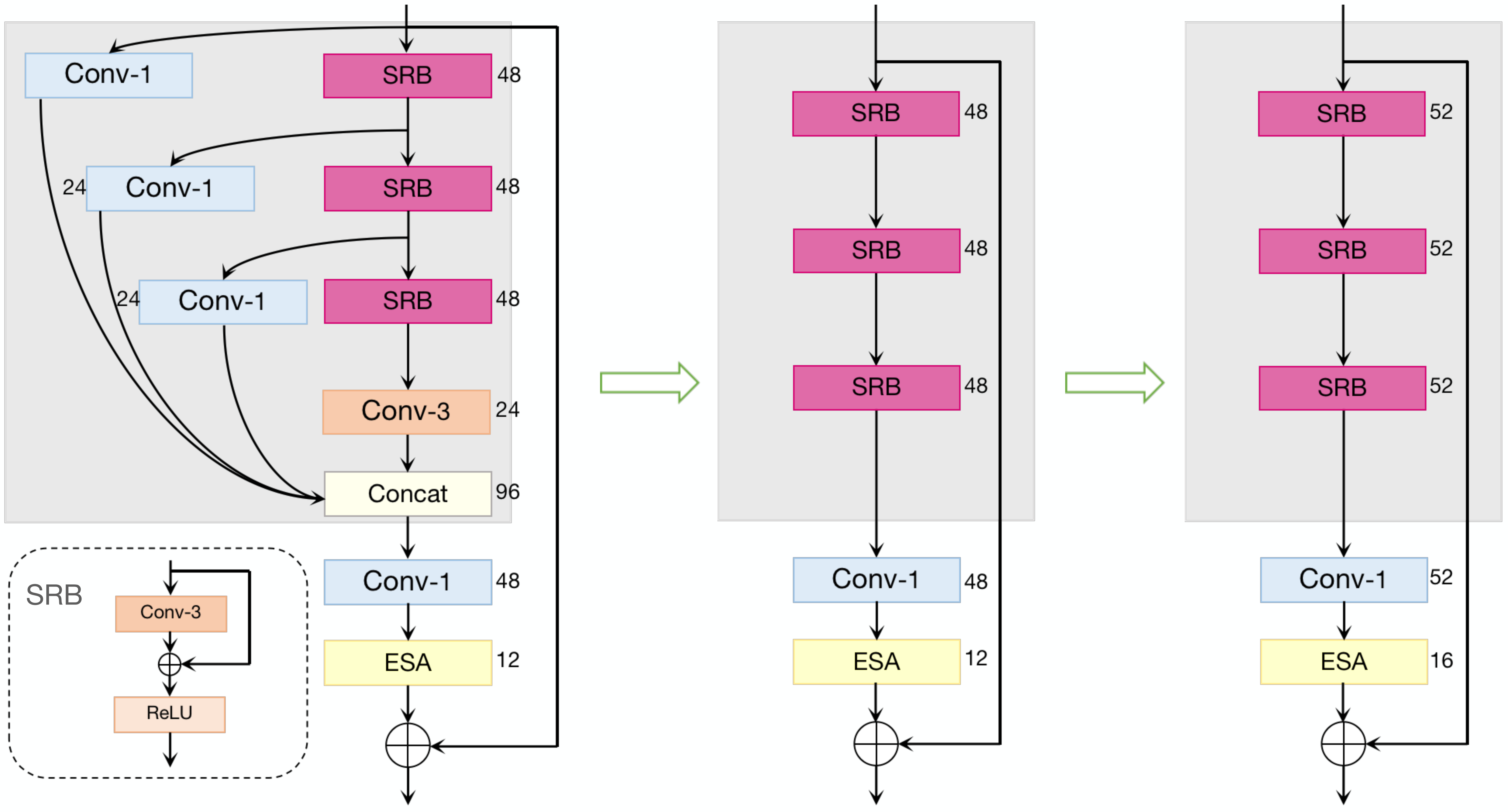
Residual Local Feature Block



RFDB
82.3ms

RFDB_R_48
61.7ms

Residual Local Feature Block

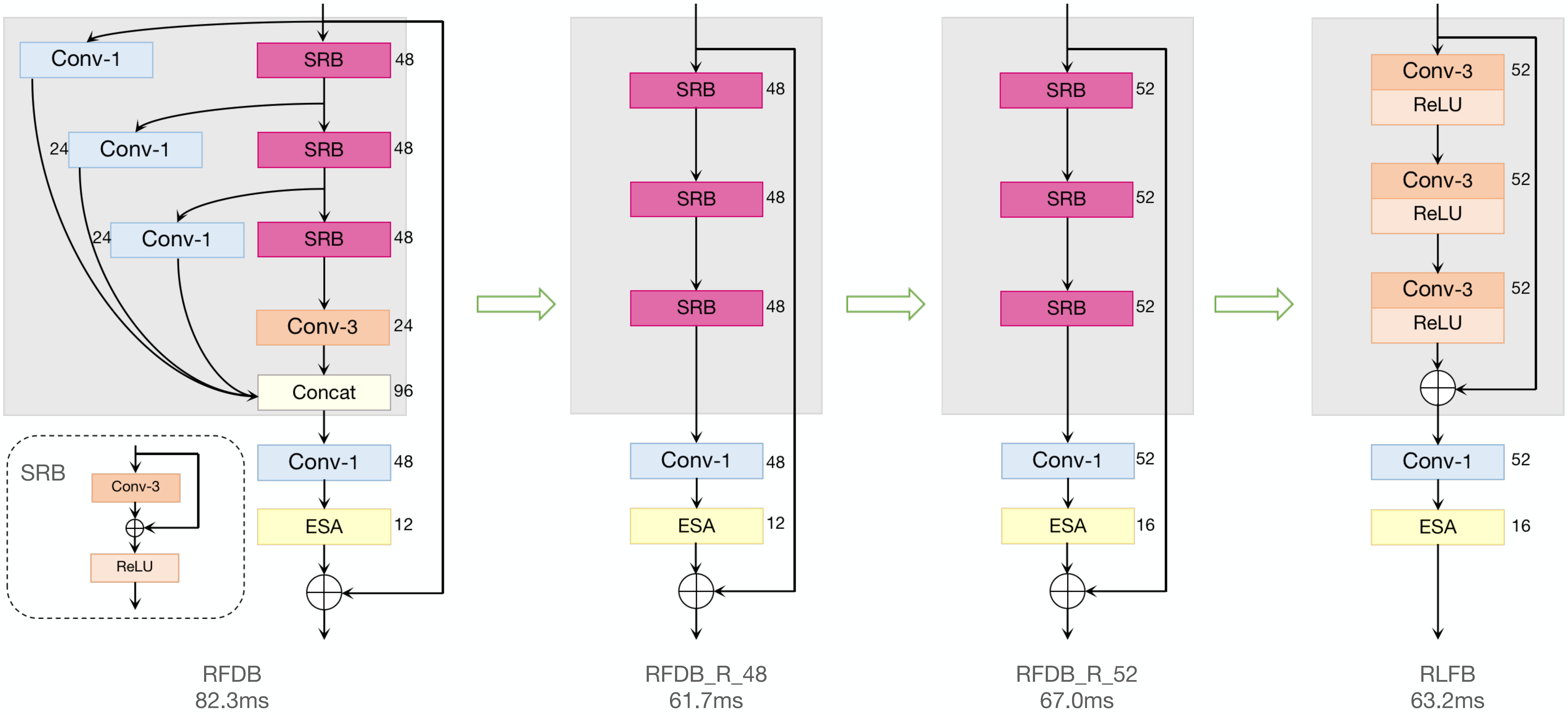


RFDB
82.3ms

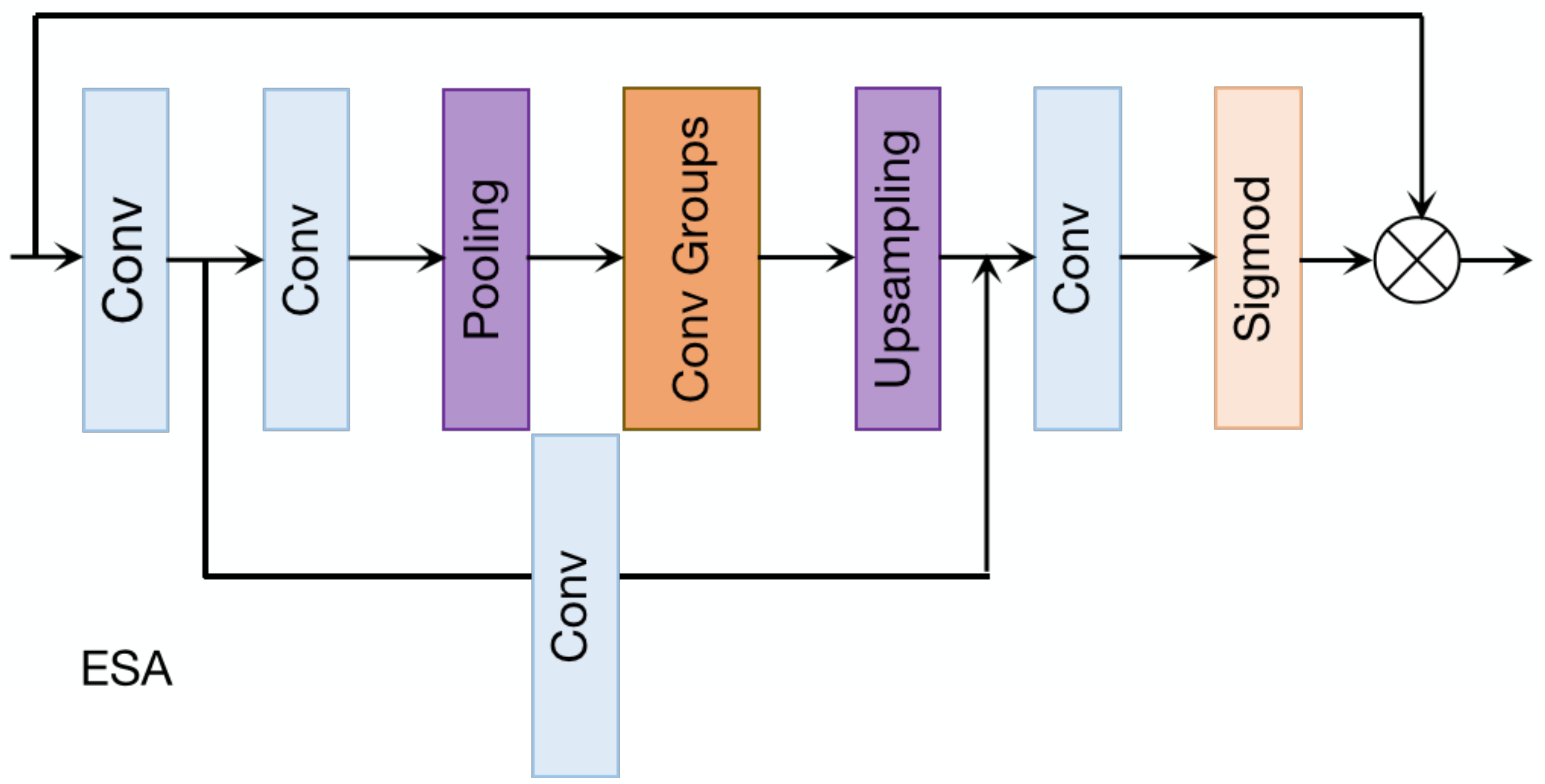
RFDB_R_48
61.7ms

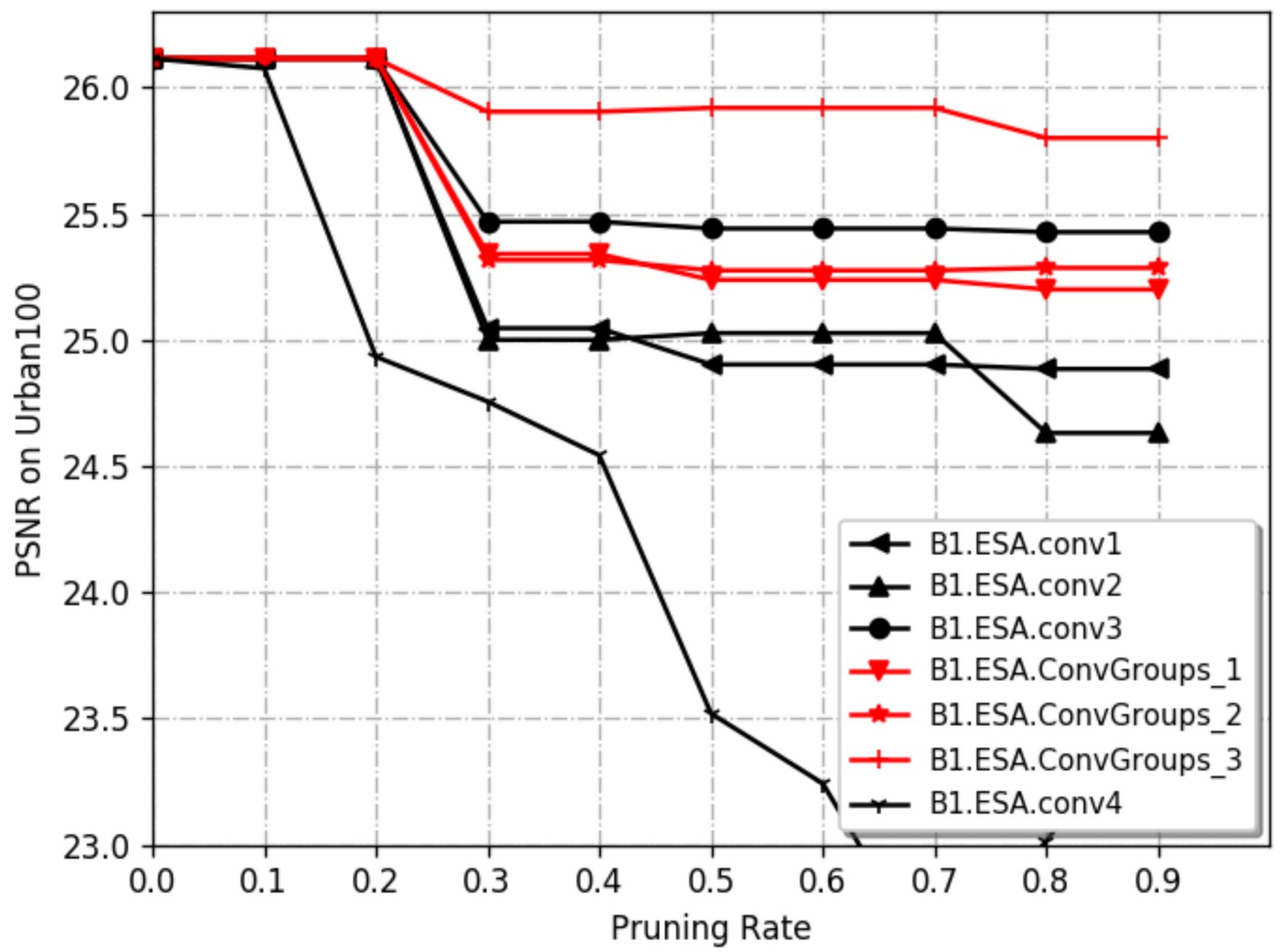
RFDB_R_52
67.0ms

Residual Local Feature Block



Simplify ESA





There are three Conv layer in the Conv Groups of ESA

Pruning sensitivity analysis shows high redundancy in Conv Groups, so we set only one Conv layer in Conv Groups

Revisiting the Contrastive Loss

ByteDance字节跳动

Contrastive Loss

The basic idea of contrastive loss is to push positives closer to anchors, and push negatives away from anchors in the latent space

$$\text{CL}= \sum_{i=1}^{n} \lambda_i \frac{d(\phi_i(Y_{anchor}),\phi_i(Y_{pos}))}{d(\phi_i(Y_{anchor}),\phi_i(Y_{neg}))}$$

anchor=output of network，pos=HR，neg=bicubic LR

Feature Extractor of Contrastive Loss

Our feature extractor：

- Structure: Conv_k3s1 + Tanh + Conv_k3s1
- Replace Relu with Tanh

Warm-Start Training Strategy

Warm-Start Strategy

- In the first stage, the model is trained from scratch.
- In the next stage, load the weights from previous stage and train model with the same settings.
- Train a model in multiple stages to get better results.

Model	Set5 PSNR / SSIM	Set14 PSNR / SSIM	BSD100 PSNR / SSIM	Urban100 PSNR / SSIM
RLFN-S_e2000	32.17 / 0.8953	28.58 / 0.7815	27.57 / 0.7354	26.08 / 0.7849
RFLN-S_clr	32.20 / 0.8959	28.59 / 0.7818	27.56 / 0.7359	26.12 / 0.7865
RLFN-S_ws_1	32.21 / 0.8959	28.60 / 0.7818	27.57 / 0.7360	26.12 / 0.7864

Table 5. Effect of learning rate strategy for 4x SR. RLFN-S_e2000 and RLFN-S_clr set the total epochs to 2000 to be compared with our proposed strategy RLFN-S_ws_1. RLFN-S_e2000 halves the learning rate every 4×10^5 iterations. RLFN-S_clr applies a cyclical learning rate policy. The best and second-best results are marked in red and blue colors, respectively.

RLFN for NTIRE 2022 efficient super-resolution challenge

RLFN for NTIRE 2022 Challenge

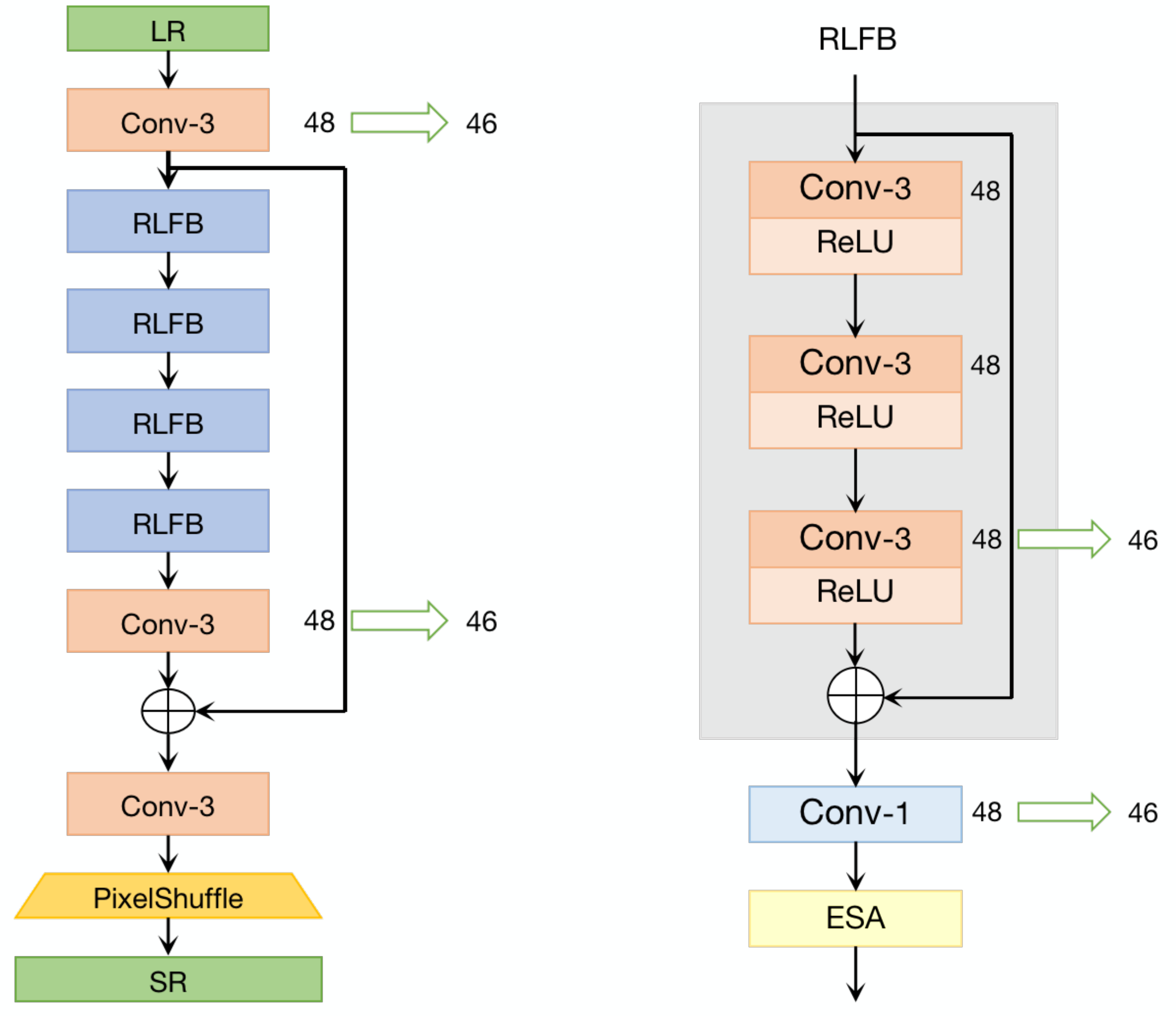
Architecture：4 RLFBs with 48 channels

training steps：

- train model from scratch with L1 loss
- employ warm-start policy and train model twice
- change loss to L1 loss + 255*Contrastive loss
- prune the model with L1 loss
- finetune with MSE loss

1st place winner in the main track (runtime track)

Pruning:



Team	Main Track	Sub-Track 1	Sub-Track 2	PSNR [Val.]	PSNR [Test]	Ave. Time [ms]	#Params [M]	FLOPs [G]	#Acts [M]	GPU Mem. [M]	#Conv
ByteESR	1	22$_{(11)}$	33$_{(2)}$	29.00	28.72	27.11$_{(1)}$	0.317$_{(11)}$	19.70$_{(11)}$	80.05$_{(6)}$	377.91$_{(4)}$	39
NJU_Jet	2	37$_{(18)}$	44$_{(6)}$	29.00	28.69	28.07$_{(2)}$	0.341$_{(18)}$	22.28$_{(19)}$	72.09$_{(4)}$	204.60$_{(1)}$	34
NEESR	3	10$_{(4)}$	27$_{(1)}$	29.01	28.71	29.97$_{(3)}$	0.272$_{(4)}$	16.86$_{(6)}$	79.59$_{(5)}$	575.99$_{(9)}$	59
Super	4	26$_{(12)}$	55$_{(10)}$	29.00	28.71	32.09$_{(4)}$	0.326$_{(14)}$	20.06$_{(12)}$	93.82$_{(10)}$	663.07$_{(15)}$	59
MegSR	5	18$_{(9)}$	43$_{(5)}$	29.00	28.68	32.59$_{(5)}$	0.290$_{(9)}$	17.70$_{(9)}$	91.72$_{(8)}$	640.63$_{(12)}$	64
RFDN AIM2020 Winner				29.04	28.75	41.97	0.433	27.10	112.03	788.13	64
IMDN_baseline				29.13	28.78	50.86	0.894	58.53	154.14	471.76	43

Scan to get our code and model!

ByteDance字节跳动

THANKS

ByteDance 字节跳动

Scan to get our code and model!