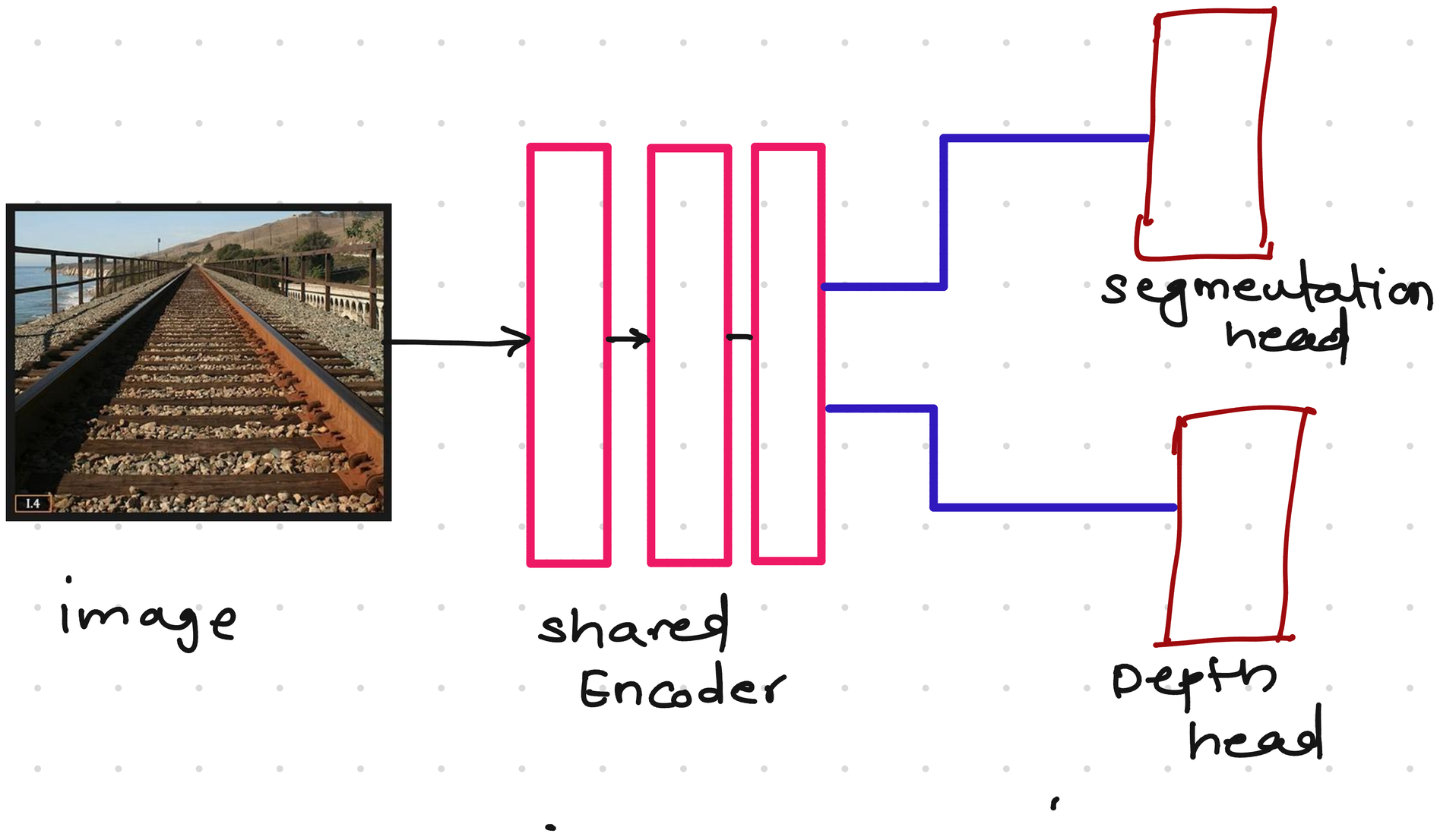
HydraNets :-

Its nothing but multitasking for self driving cars.



- Both training & inference is faster
- light in memory

issues :-

Unbalenced datasets :- data augmentation.
Learning rate :- meet in midway
loss functions :-

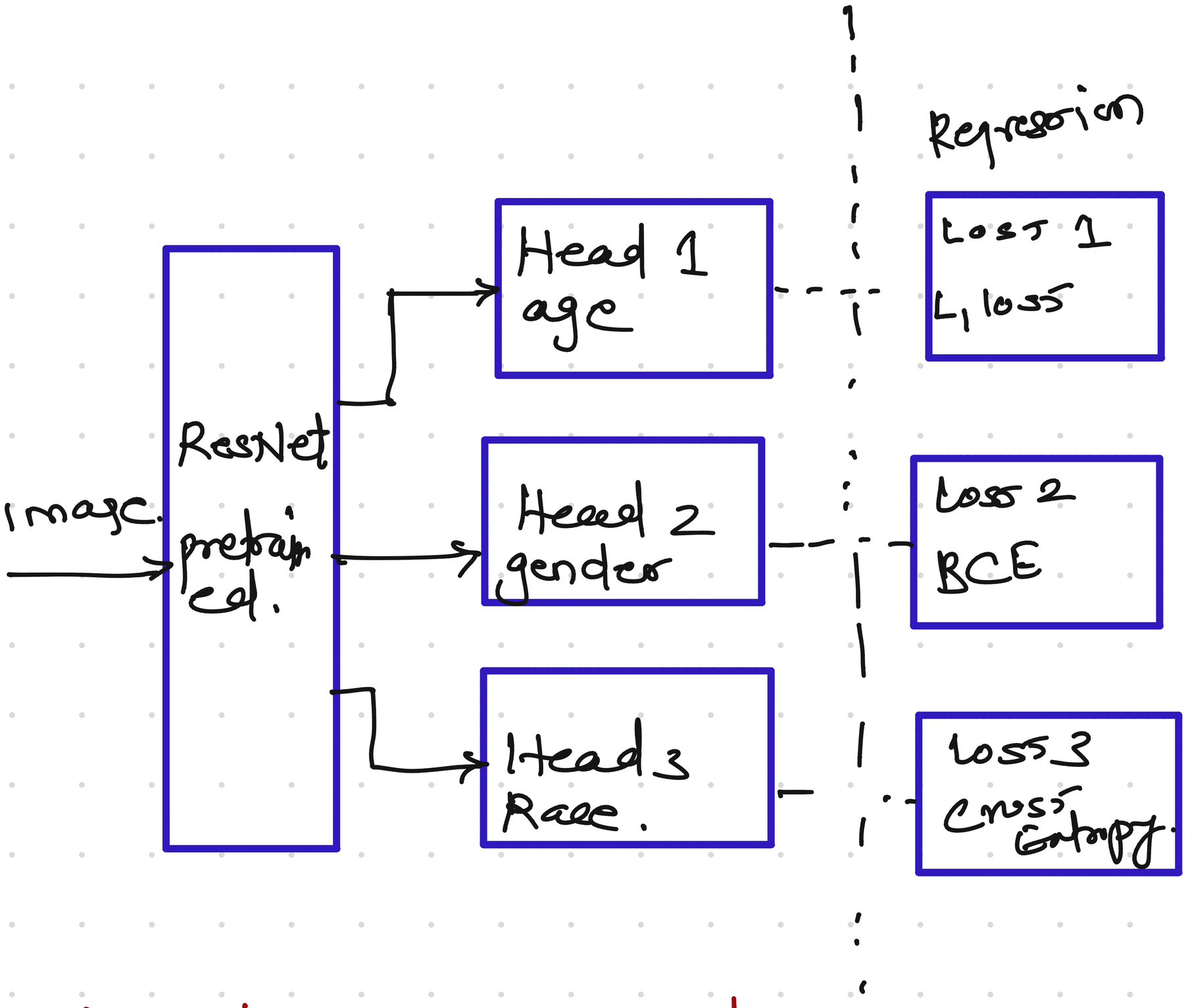
$$L = \alpha L_1 + \beta L_2$$
Reg. Classi.

Project - 01 :-

Multitasking for classification :-

Dataset – Utk Face data.

image → age → number ← head 1
image → gender → male or female ← head 2
image → Race → indian, black, asian, white, other ← head 3



→ Most commonly used multitasking architecture.

Fully adapted feature sharing :- grouping similar tasks

Cross Stitch Network :- to neuron communicate w. each other.

Cross correlated Network :- same as CSN but with Conv.

→ Very imp question :-

Which Task should be learned together.

model - medium Dataset - 4M images

		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	-5.41%	-11.29%	-4.32%	-34.64%	-13.92%
	Depth	4.17%	–	-3.55%	3.49%	3.76%	1.97%
	Normals	8.50%	2.48%	–	1.37%	12.33%	6.17%
	Keypoints	4.82%	1.38%	-0.02%	–	-5.26%	0.23%
	Edges	3.07%	-0.92%	-4.42%	1.37%	–	-0.23%
	Average	5.14%	-0.62%	-4.82%	0.48%	-5.95%	-1.15%

model - large Dataset - 4M images

		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	3.00%	-2.79%	-5.20%	27.80%	5.70%
	Depth	1.72%	–	1.18%	-3.52%	25.73%	6.28%
	Normals	10.81%	7.12%	–	88.98%	71.59%	44.62%
	Keypoints	3.12%	-0.41%	-10.12%	–	61.07%	13.42%
	Edges	0.03%	-1.40%	-4.78%	-3.05%	–	-2.30%
		3.92%	2.08%	-4.13%	19.30%	46.54%	13.54%

model - large Dataset - 200k images.

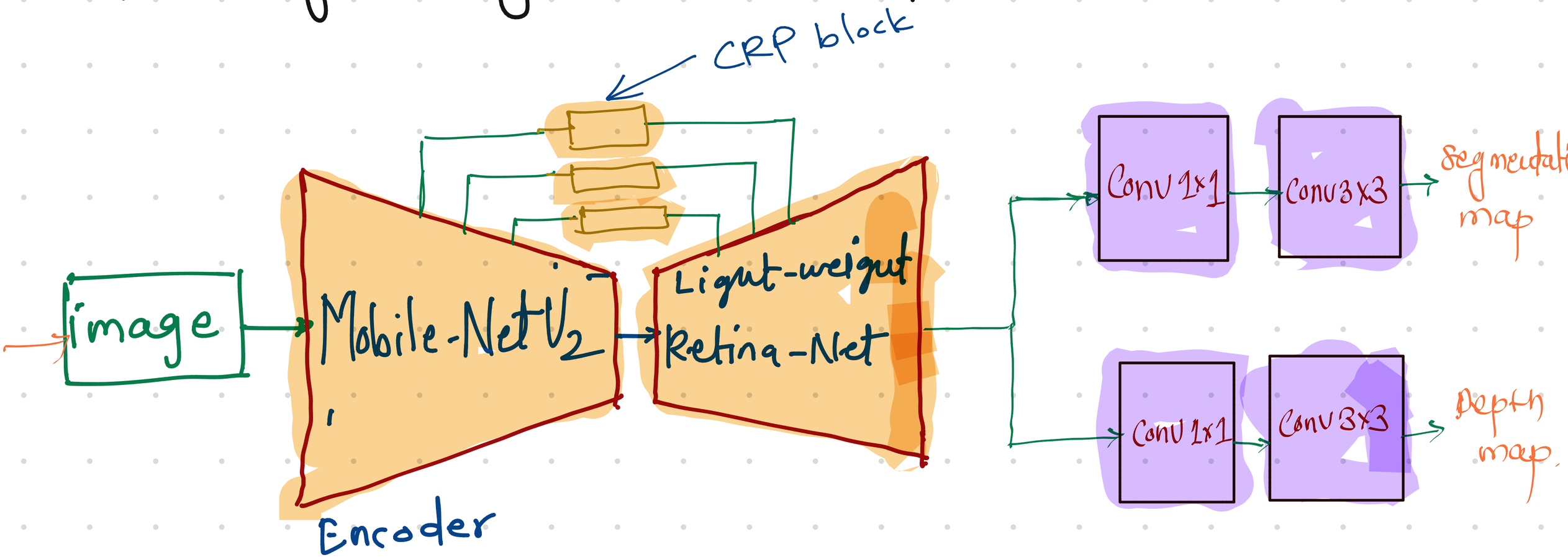
		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	1.91%	-6.00%	-9.91%	-21.93%	-8.98%
	Depth	-12.63%	–	2.95%	1.44%	-9.70%	-4.48%
	Normals	8.32%	15.38%	–	-1.35%	52.08%	18.61%
	Keypoints	-5.84%	-7.21%	-2.26%	–	55.63%	10.08%
	Edges	-5.62%	6.02%	-4.16%	-5.02%	–	-2.20%
		-3.95%	4.03%	-2.37%	-3.71%	19.02%	2.6%

Outcom :- adding Normals to any task improves

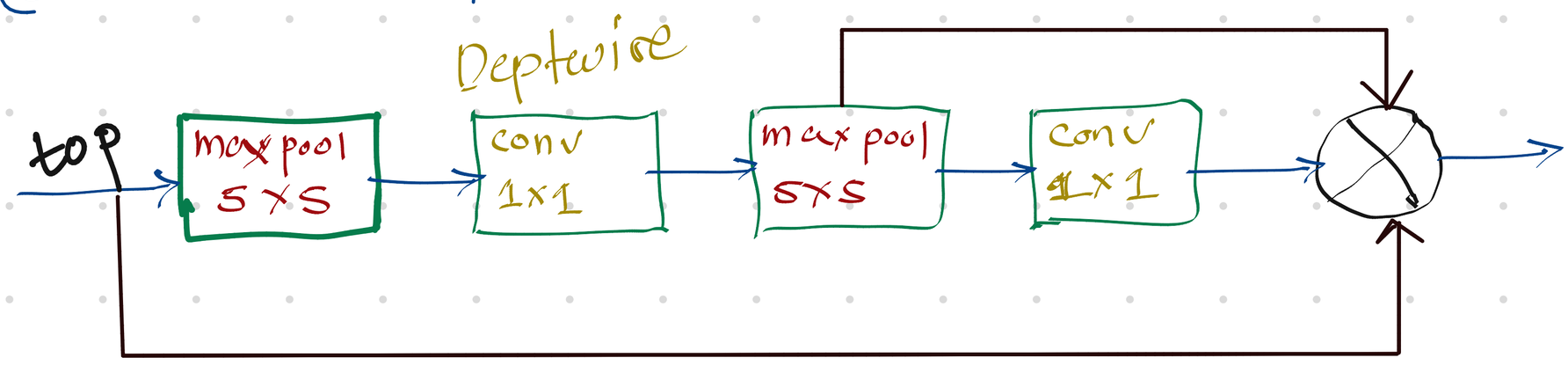
size matters of encoder

→ HydraNets Essential :-

Real-Time joint segmentation & Depth Estimation :-



CRP (chained-Residual-pooling) :-



Mobile-Net :-

mobile-Net is light-weight conv network for faster inference.
It uses deptwise & pointwise conv. rather than normal convolution.

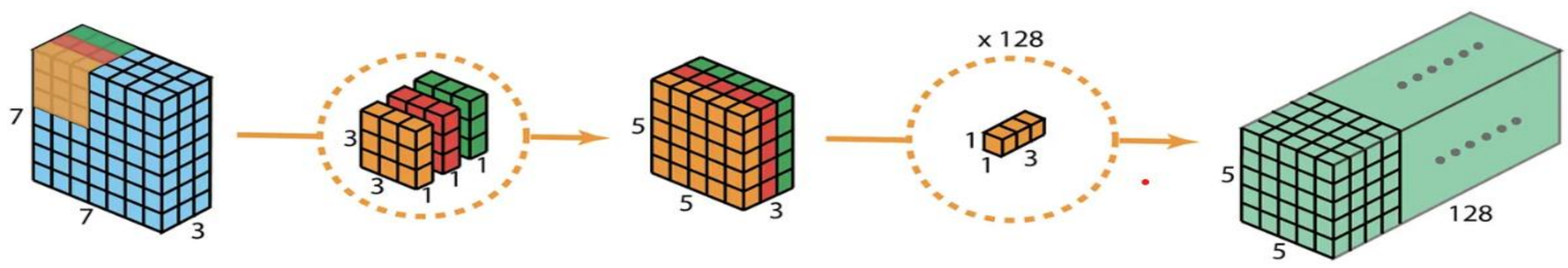


Fig. 5: Concrete example of depth-wise separable convolutions (source)

deptwise convolution keeps the channels same. ← groups = no_channels

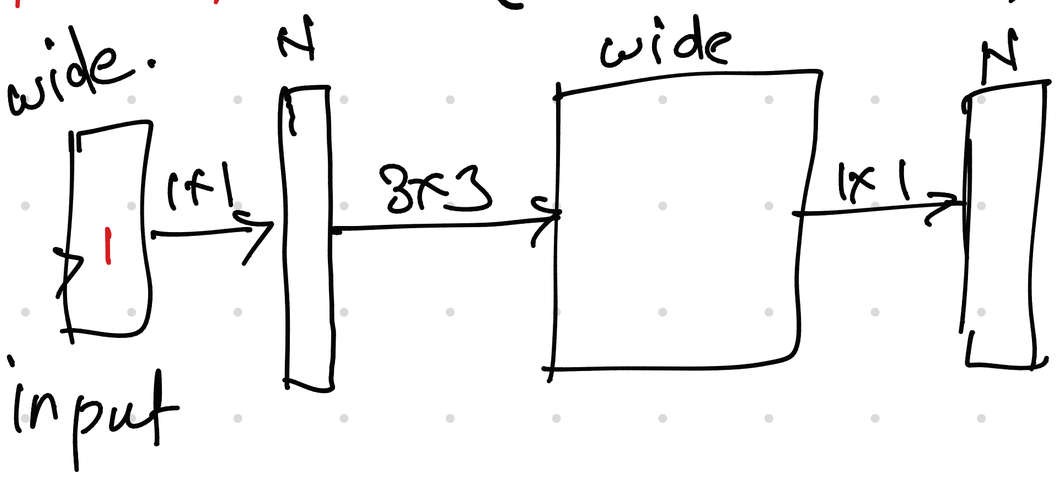
nn.Conv2d (32, 32, stride = 1, padding = 1, bias = false)

pointwise convolution has only one output channel.

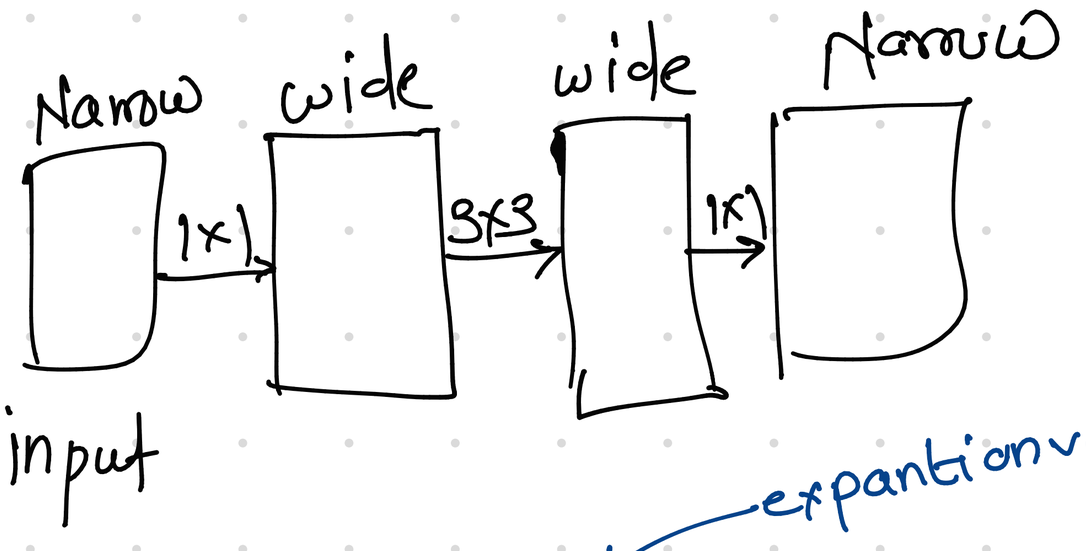
nn.Conv2d (32, 1, stride = 1, padding = 1, bias = false)

Computational parameters (depth & point) = $\frac{1}{4}$ × Normal conv. parameter.

Residual block :- (mobileNet V1)



Inverted Residual block :-



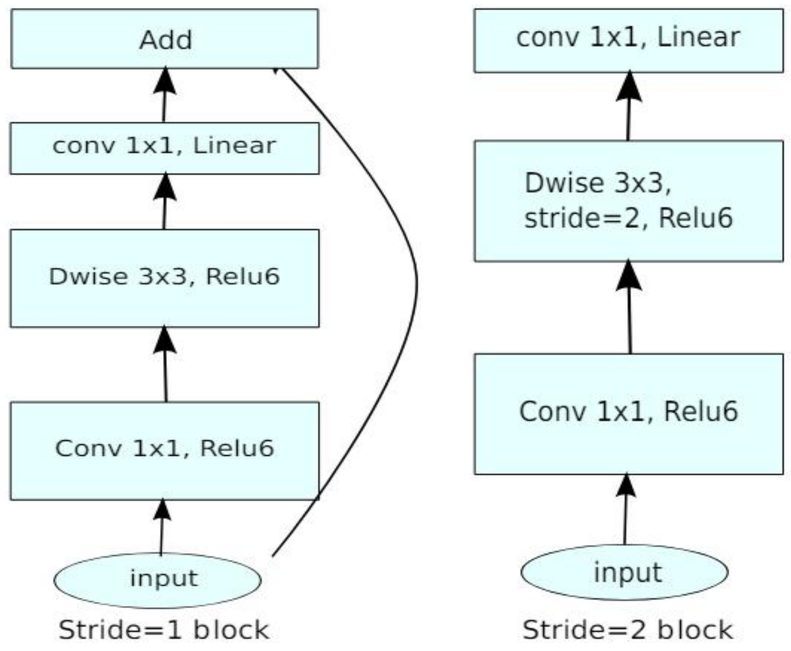
expantionvalue.

(c x h x w) --> (c·e·x h x w) -->

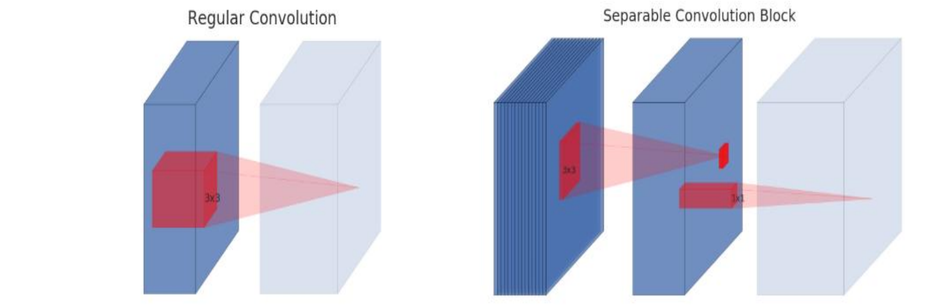
sum.

BottleNeck

When in_features = out_features

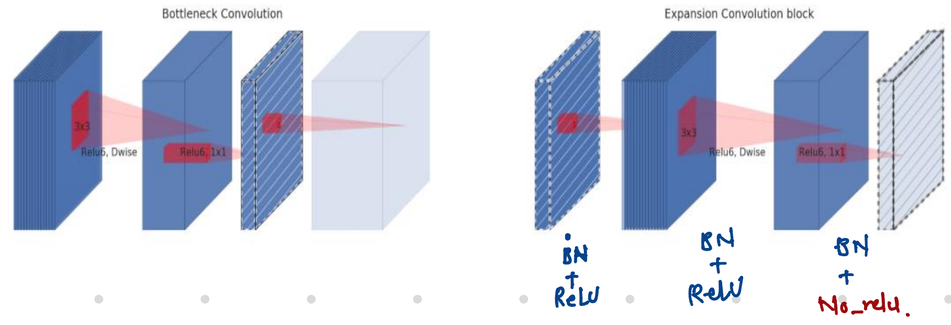


(d) Mobilenet V2



(c) Separable with linear bottleneck

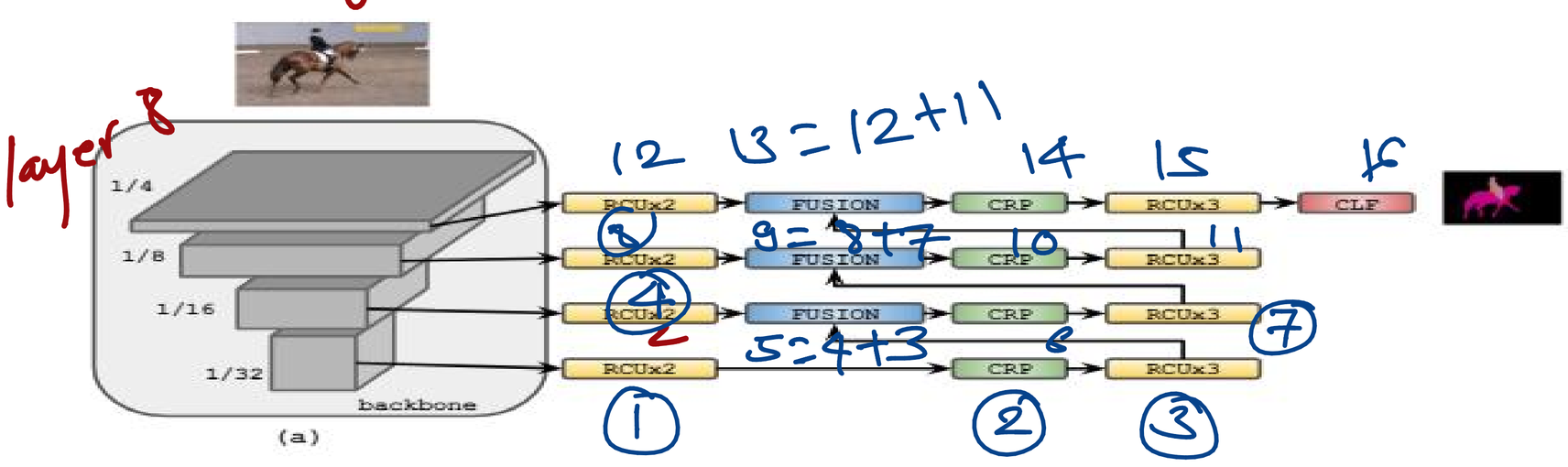
(d) Bottleneck with expansion layer

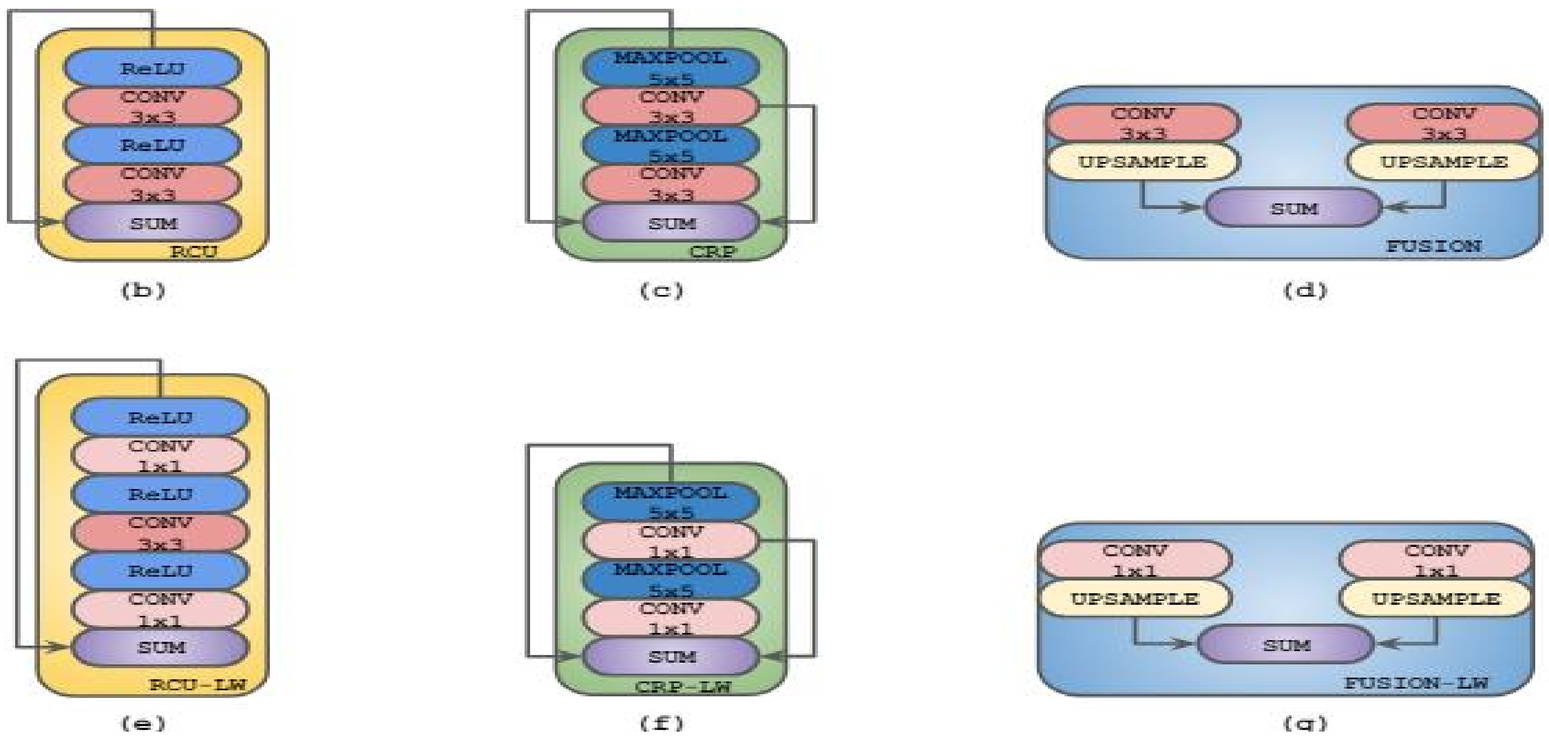


groups - divides the input_channels into different groups & then filter is applied.

To avoid No_linearity.

Decoder :- Light Weight Refine-Net





MobileNet - 8 layers