FCRN

用于深度估计的深度网络，作者提出了一个深度估计模型，基本结构是encoder-decoder，然后decoder用了project模块，其实就是加了一个residual模块。然后上采样用的是unPooling 2x2 + conv 5x5 ，这个叫up-conv。
如果是结合了residual的upconv，就叫做up-proj，up-proj的两个分支都是接了5x5卷积保证resulotion相同，主分支还接了3x3卷积，然后相加。

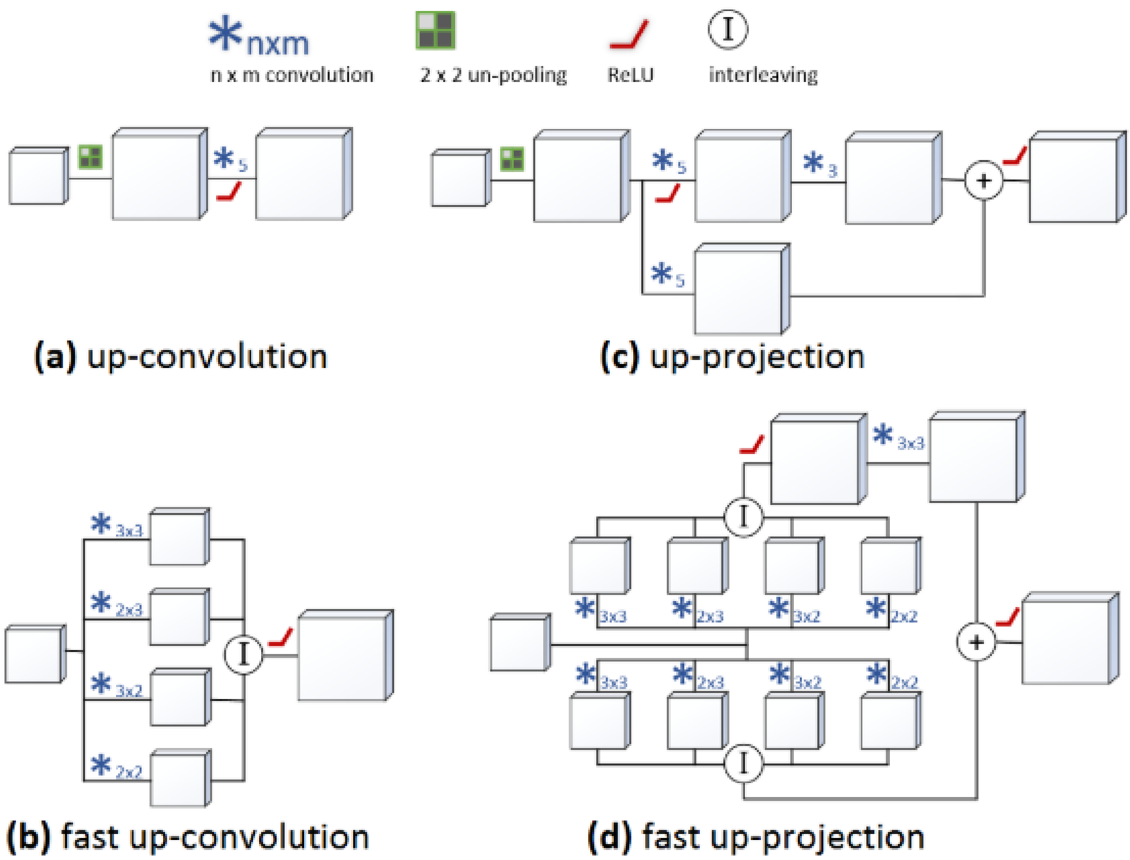

Figure 2. **From up-convolutions to up-projections. (a)** Standard up-convolution. **(b)** The equivalent but faster up-convolution. **(c)** Our novel up-projection block, following residual logic. **(d)** The faster equivalent version of (c)

最后作者为了加速训练，设计了一种等效的fast up-conv/up-proj模块，速度更快，借鉴了fracted conv思想，把5x5卷积拆成小卷积核。

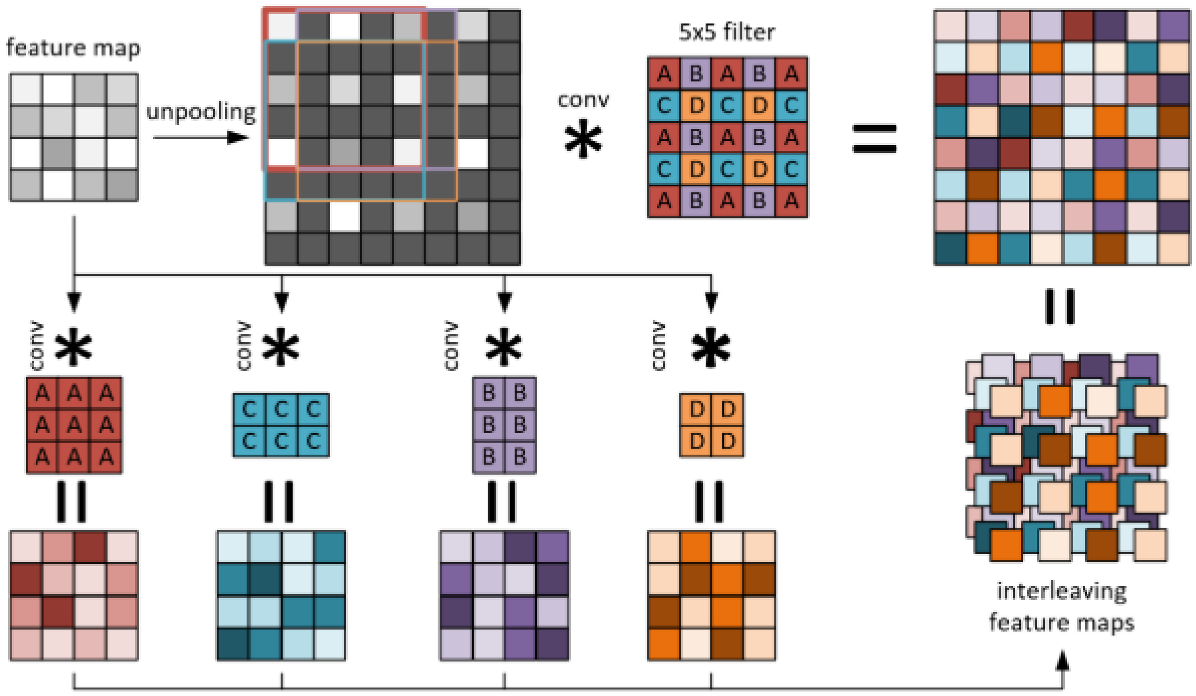

Figure 3. **Faster up-convolutions.** Top row: the common up-convolutional steps: unpooling doubles a feature map's size, filling the holes with zeros, and a 5×5 convolution filters this map. Depending on the position of the filter, only certain parts of it (A,B,C,D) are multiplied with non-zero values. This motivates convolving the original feature map with the 4 differently composed filters (bottom part) and interleaving them to obtain the same output, while avoiding zero multiplications. A,B,C,D only mark locations and the actual weight values will differ

最后的网络是resnet + 4个up模块的堆叠，相当于上采样16x。

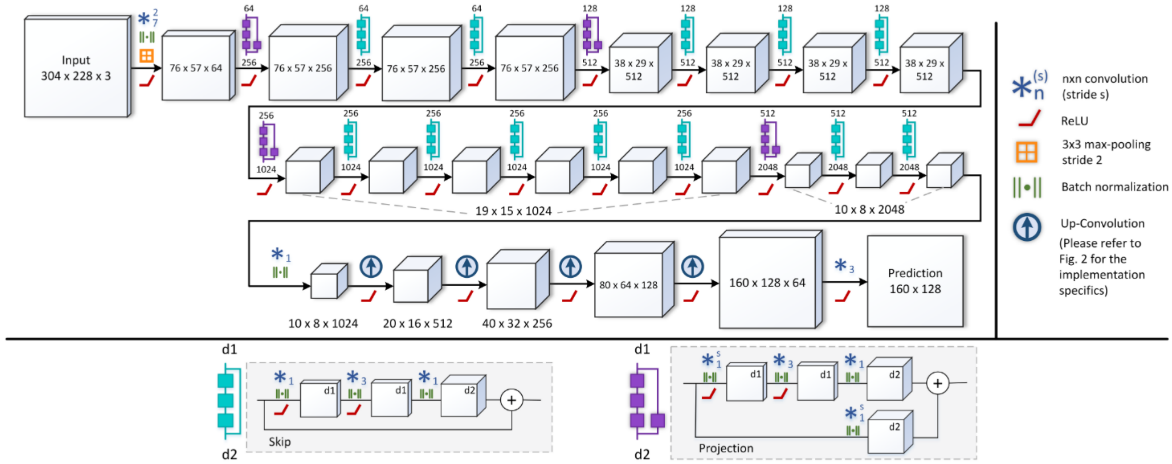

Figure 1. **Network architecture.** The proposed architecture builds upon ResNet-50. We replace the fully-connected layer, which was part of the original architecture, with our novel up-sampling blocks, yielding an output of roughly half the input resolution

然后是loss设计用了berHu loss，在我看来和smoothL1 loss没有太大区别。

$$\mathcal{B}(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2+c^2}{2c} & |x| > c. \end{cases} \qquad (1)$$

最后的结果对比，在不同的backbone上，显然resnet效果比较好，而且作者提出的up-conv模块优于de-conv，这个是我比较疑惑的。但是差的不太多

Architecture		Loss	#params	rel	rms	$\log_{10}$	δ_1	δ_2	δ_3
AlexNet	FC	$\mathcal{L}_2$	104.4×10^6	0.209	0.845	0.090	0.586	0.869	0.967
		berHu		0.207	0.842	0.091	0.581	0.872	0.969
	UpConv	$\mathcal{L}_2$	6.3×10^6	0.218	0.853	0.094	0.576	0.855	0.957
		berHu		0.215	0.855	0.094	0.574	0.855	0.958
VGG	UpConv	$\mathcal{L}_2$	18.5×10^6	0.194	0.746	0.083	0.626	0.894	0.974
		berHu		0.194	0.790	0.083	0.629	0.889	0.971
ResNet	FC-160x128	berHu	359.1×10^6	0.181	0.784	0.080	0.649	0.894	0.971
	FC-64x48	berHu	73.9×10^6	0.154	0.679	0.066	0.754	0.938	0.984
	DeConv	$\mathcal{L}_2$	28.5×10^6	0.152	0.621	0.065	0.749	0.934	0.985
	UpConv	$\mathcal{L}_2$	43.1×10^6	0.139	0.606	0.061	0.778	0.944	0.985
		berHu		0.132	0.604	0.058	0.789	0.946	0.986
	UpProj	$\mathcal{L}_2$	63.6×10^6	0.138	0.592	0.060	0.785	0.952	0.987
		berHu		**0.127**	**0.573**	**0.055**	**0.811**	**0.953**	**0.988**

Table 1. Comparison of the proposed approach against different variants on the NYU Depth v2 dataset. For the reported errors rel, rms, $\log_{10}$ lower is better, whereas for the accuracies $\delta_i < 1.25^i$ higher is better