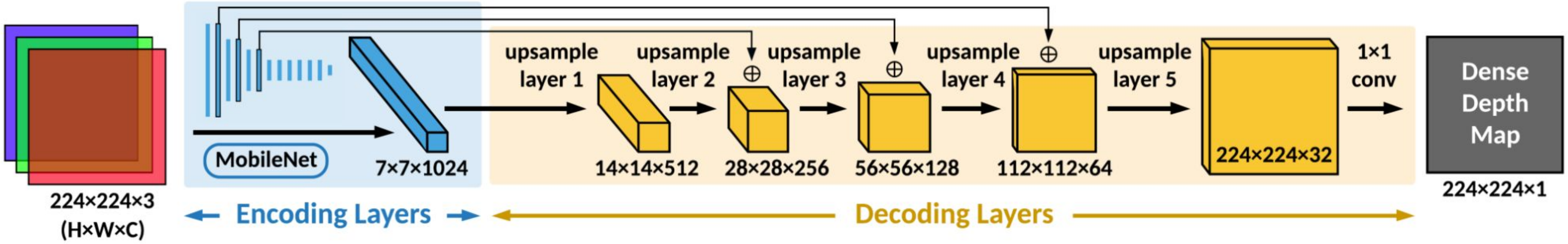
FastDepth: Fast Monocular Depth Estimation on Embedded Systems

Diana Wofk∗ , Fangchang Ma∗ , Tien-Ju Yang, Sertac Karaman, Vivienne Sze

http://fastdepth.mit.edu/

Proposed architecture and features



Proposed decoders:

- ResNet50
- ResNet18
- MobileNet (used if final solution)

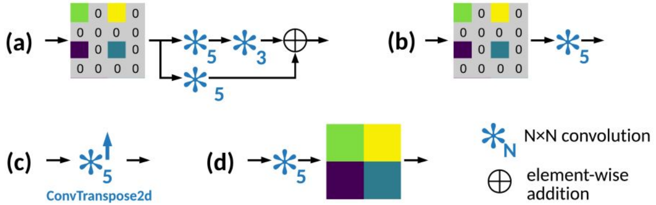
Proposed decoder upsampling layers:

a. UpProj:
   - 2x2 unpooling
   - convs 5x5 -> 3x3 + 5x5

b. UpConv:
   - 2x2 unpooling
   - conv 5x5

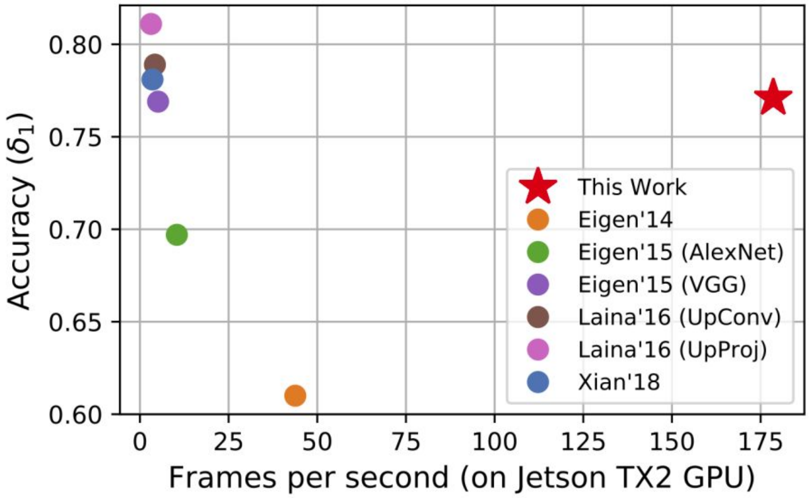
c. DeConv5:
   - transpose conv 5x5

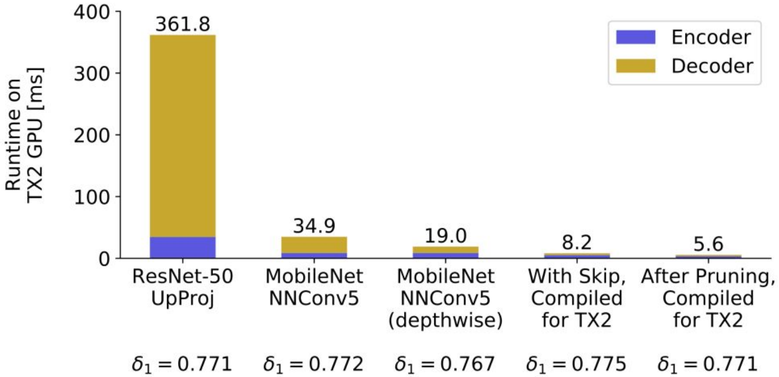
d. NNConv5 (used in final solution):
   - conv 5x5
   - nn interpolation, scale 2



Increasing performance

- NNConv5 as decoder layer
- Depthwise separable convolutions in decoder
- Additive skip-connections
- Hardware-specific optimization
- Network pruning





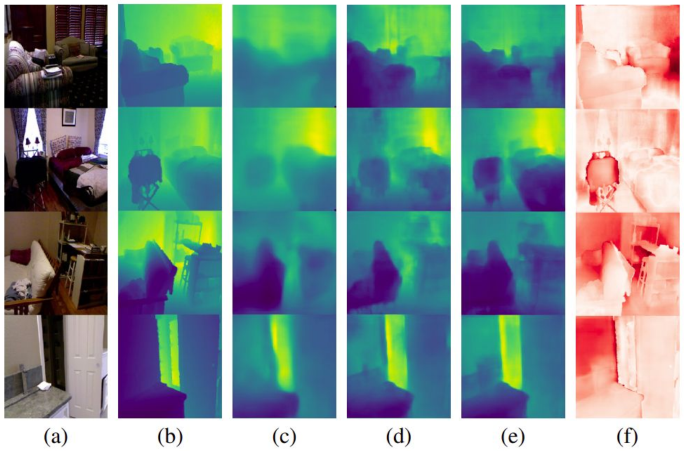


Fig. 4: Visualized results of depth estimation on the NYU Depth v2 dataset. (a) input RGB image; (b) ground truth; (c) our model, without skip connections, unpruned; (d) our model, with skip connections, unpruned; (e) our model, with skip connections, pruned; (f) error map between the output of our final pruned model and ground truth, where redder regions indicate higher error.

Model

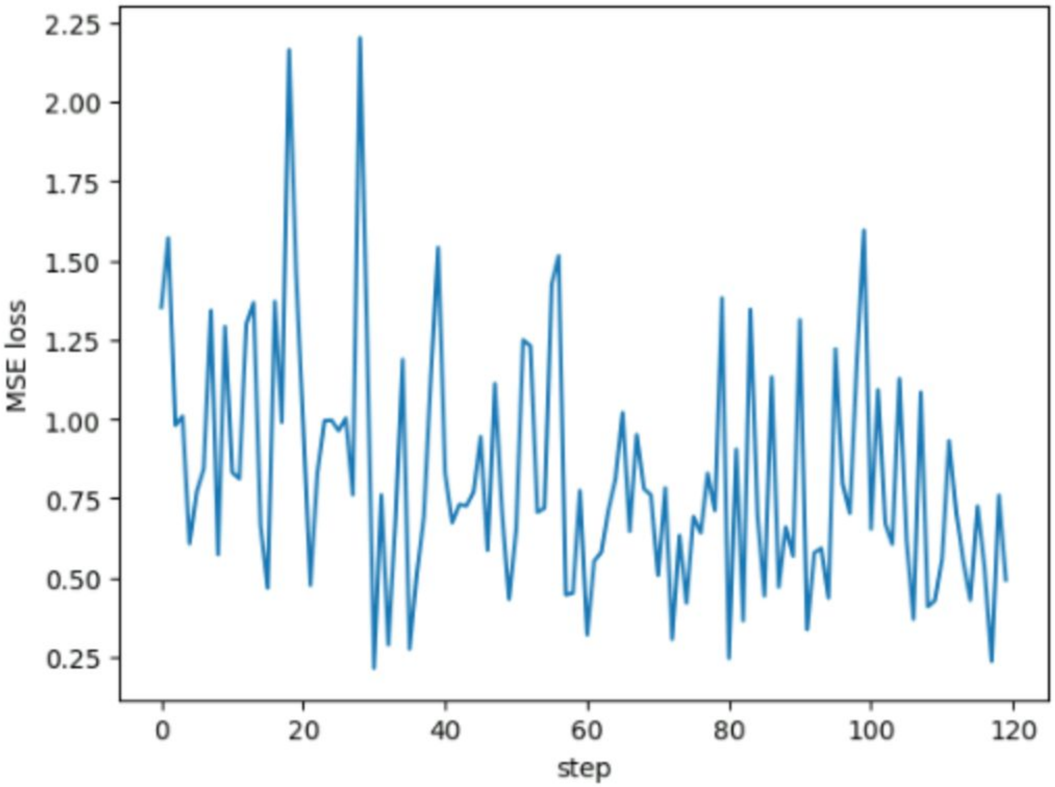
Encoder: MobileNetV2 – first 14 layers, pretrained on CIFAR100 (Imagenet is private now, pretrained model from torchvision has complex naming structure as it is hard to make skip connections)

Decoder: 5 layers of NNConv5 followed by pointwise convolution

Dataset: NYUv2 – train: 47584 images/depthmap pairs, validation: 654 images/depthmap pairs

Learning: SGD – momentum=0.9, weight_decay=0.0001, lr=0.01, batch_size=8, epochs=12

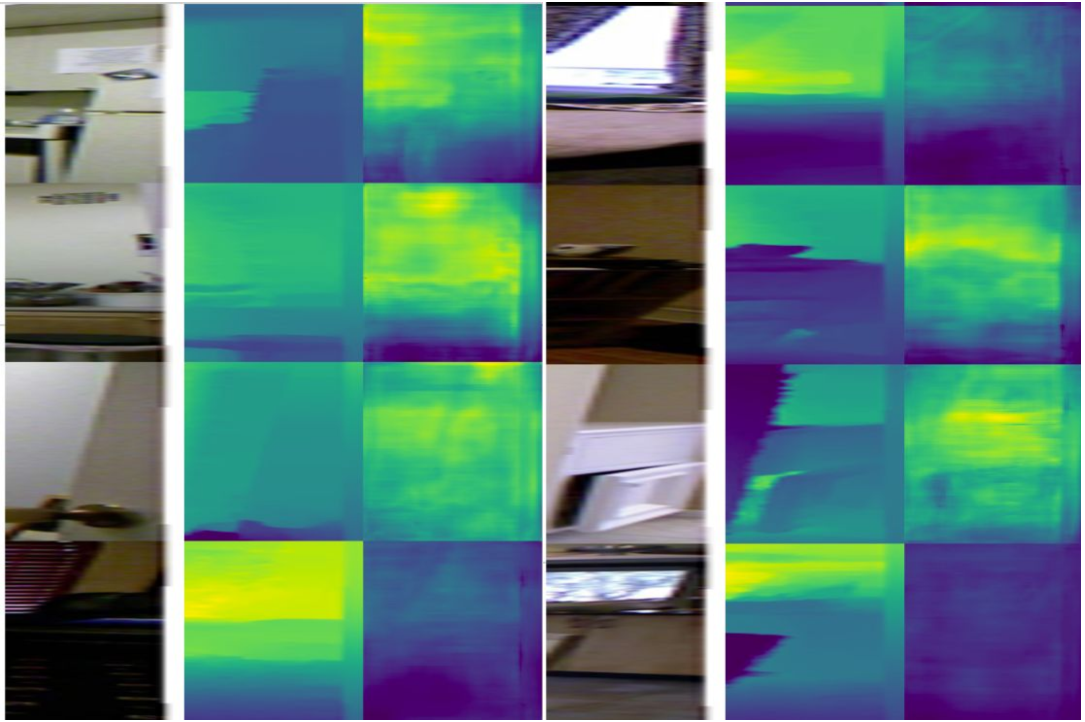
Learning curve



Comparing with original paper’s results

Metric/Instance	Reference	My
RMSE	0.604	0.827
δ1	0.771	0.480
MAE	—	0.702
GPU runtime [ms]	5.6	11
CPU runtime [ms]	37	66

Output samples



Performance for different architectures and frameworks

Architecture	Format	Time [ms]
Personal PC (x86 cpu, NVIDIA gpu)	pytorch/cpu	158.7
	pytorch/gpu	22.0
	onnx/cpu	94.53
	onnx/gpu	20.93
	TVM/unopt/cpu	115.35
	TVM/opt/cpu	120.39
RPi	pytorch/cpu	kernel died
	onnx/cpu	2516

Results

Make model – done

Train model on NYUv2 dataset – done

Converting model to onnx format – done

Optimizing model with TVM – not yet (it takes too many cpu resources and can be run only on my PC)

Pruning model with NetAdapt – not yet

Making inference on different architectures – done

Todo

Train network with more epochs count

Use TVM for decomposed convolutions

Use RPC for optimizing for RaspberryPi

Use NetAdapt algorithm