The Devil is in the Decoder

Zbigniew Wojna[1]
zbigniewwojna@gmail.com
Vittorio Ferrari[2]
vittoferrari@google.com
Sergio Guadarrama[2]
sguada@google.com
Nathan Silberman[2]
nsilberman@google.com
Liang-Chieh Chen[2]
lcchen@google.com
Alireza Fathi[2]
alirezafathi@google.com
Jasper Uijlings[2]
jrru@google.com

[1] University College London
[2] Google, Inc.

Abstract

Many machine vision applications require predictions for every pixel of the input image (for example semantic segmentation, boundary detection). Models for such problems usually consist of encoders which decreases spatial resolution while learning a high-dimensional representation, followed by decoders who recover the original input resolution and result in low-dimensional predictions. While encoders have been studied rigorously, relatively few studies address the decoder side. Therefore this paper presents an extensive comparison of a variety of decoders for a variety of pixel-wise prediction tasks. Our contributions are: (1) Decoders matter: we observe significant variance in results between different types of decoders on various problems. (2) We introduce a novel decoder: bilinear additive upsampling. (3) We introduce new residual-like connections for decoders. (4) We identify two decoder types which give a consistently high performance.

1 Introduction

Many important machine vision applications require predictions for every pixel of the input image. Examples include but are not limited to: semantic segmentation [20], boundary detection [32], super-resolution [16], colorization [10], depth estimation [22], normal surface estimation [6], saliency prediction [26], image generation networks (GANs) [24], and optical flow [11]. Models for such applications are usually composed of a feature extractor that decreases spatial resolution while learning high-dimensional representation and a decoder that recovers the original input resolution. While feature extractors were rigorously studied

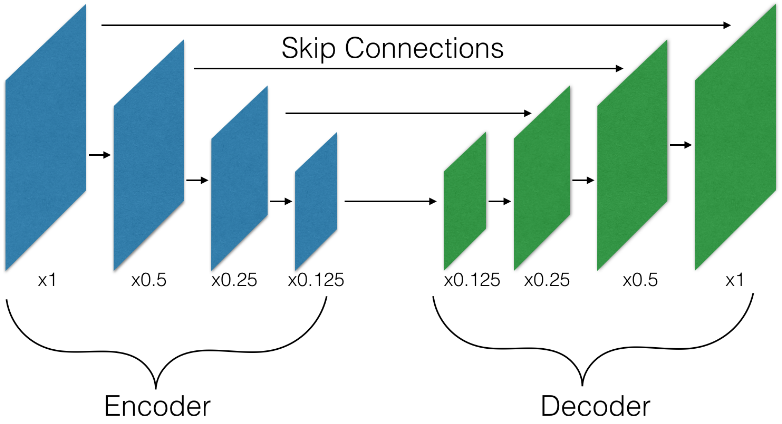


Figure 1: General schematic architecture used for dense prediction problems.

(for example in the context of image classification), relatively few studies have been done on the decoder side.

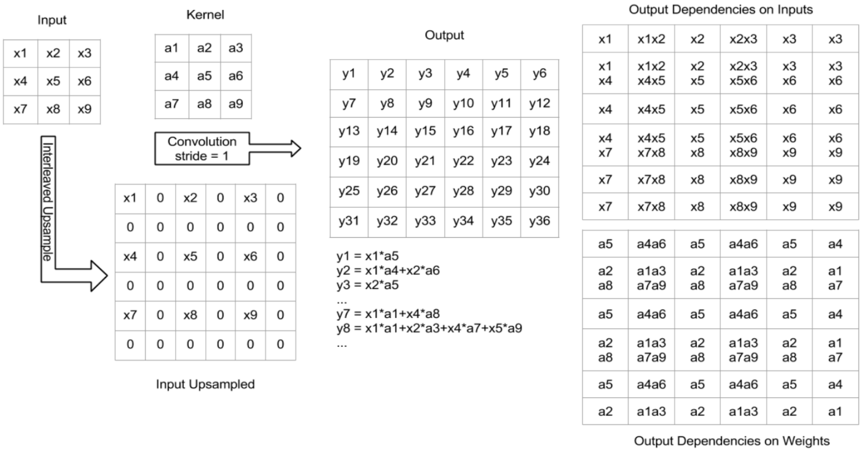
This work presents an extensive analysis of a variety of decoding methods on a broad range of machine vision tasks: semantic segmentation, depth prediction, colorization, super-resolution, and instance edge detection. We make the following contributions: (1) Decoders matter: we observe significant variance in results between different types of decoders on various problems. (2) We introduce a new bilinear additive upsampling layer, which results in significant improvements over normal bilinear upsampling. (3) We introduce residual-like connections for decoders. While the differences in spatial resolution and number of feature channels of the input and output of the decoder make it impossible to use residual connections directly, we show how to create residual-like connections. (4) We identify two decoder types which give a consistently high performance on all tested machine vision tasks.

2 Decoder Architecture

Dense problems which require per pixel predictions are typically addressed with an encoder-decoder architecture (see Figure 1). First, a feature extractor downsamples the spatial resolution (usually by a factor 8-32) while increasing the number of channels. Afterward, a 'decoder' upsamples the representation back to the original input size. Conceptually, such decoder can be seen as a reversed operation to what encoders are doing. One decoder module consists of at least one layer that increases spatial resolution, which we call an upsampling layer, and possibly layers that preserve spatial resolution (e.g. standard convolution, a residual block, an inception block).

Decoder architectures were previously studied only in the context of the single problem: [15] analyzes 5×5 transposed convolution and proposes equivalent convolutional operation but faster. [30] improved on super-resolution through depth to space transformation. [25] examined artifacts which transposed convolution causes in generator network in Generative Adversarial Network.

Layers that preserve spatial resolution were well studied in the literature in the context of neural architectures for image classification [1, 2, 9, 31]. Therefore we only analyze the layers that increase spatial resolution - the upsampling layers. Unlike other studies on decoder



Input

x1	x2	x3
x4	x5	x6
x7	x8	x9

Kernel

a1	a2	a3
a4	a5	a6
a7	a8	a9

Input Upsampled

x1	0	x2	0	x3	0
0	0	0	0	0	0
x4	0	x5	0	x6	0
0	0	0	0	0	0
x7	0	x8	0	x9	0
0	0	0	0	0	0

Output

y1	y2	y3	y4	y5	y6
y7	y8	y9	y10	y11	y12
y13	y14	y15	y16	y17	y18
y19	y20	y21	y22	y23	y24
y25	y26	y27	y28	y29	y30
y31	y32	y33	y34	y35	y36

y1 = x1*a5
y2 = x1*a4+x2*a6
y3 = x2*a5
...
y7 = x1*a1+x4*a8
y8 = x1*a1+x2*a3+x4*a7+x5*a9
...

Output Dependencies on Inputs

x1	x1x2	x2	x2x3	x3	x3
x1 x4	x1x2 x4x5	x2 x5	x2x3 x5x6	x3 x6	x3 x6
x4	x4x5	x5	x5x6	x6	x6
x4 x7	x4x5 x7x8	x5 x8	x5x6 x8x9	x6 x9	x6 x9
x7	x7x8	x8	x8x9	x9	x9
x7	x7x8	x8	x8x9	x9	x9

Output Dependencies on Weights

a5	a4a6	a5	a4a6	a5	a4
a2 a8	a1a3 a7a9	a2 a8	a1a3 a7a9	a2 a8	a1 a7
a5	a4a6	a5	a4a6	a5	a4
a2 a8	a1a3 a7a9	a2 a8	a1a3 a7a9	a2 a8	a1 a7
a5	a4a6	a5	a4a6	a5	a4
a2	a1a3	a2	a1a3	a2	a1

Figure 2: Transposed convolution with kernel size 3 and stride 2.

architectures, we do this on five different machine vision tasks. Recently, stacked hourglass networks were proposed [23], which are multiple encoder-decoder networks stacked in sequence. Our study is valid for a single encoder-decoder network, while intuitively the choice of decoder becomes more important as the number of decoders increase.

2.1 Upsampling layers

Below we present and compare several ways of upsampling the spatial resolution in convolution neural networks, a crucial part of any decoder. We limit our study to upsampling the spatial resolution by a factor of two which is the most common setup in the literature. Figures exemplifying the upsampling operations assume kernel of size $3x3$.

2.1.1 Existing upsampling layers

Transposed Convolution. Transposed Convolutions are the most commonly used upsampling layers and are also sometimes referred to as 'deconvolution' or 'upconvolution' [5, 20, 34] in multiple applications [3, 4, 15, 28]. A Transposed Convolution can be seen as a reversed convolution in the sense of how the input and output are related to each other. However, it is not an inverse operation, since calculating the exact inverse is an under-constrained problem and therefore ill-posed. Transposed convolution is equivalent to interleaving the input features with 0's and applying a standard convolutional operation. The calculations of a transposed convolution are illustrated in Figure 2.

Decomposed Transposed Convolution. Decomposed Transposed Convolution is similar to the transposed convolution, but conceptually it splits the main convolution operation up into multiple low-rank convolutions. For images, it simulates a 2D transposed convolution using two 1D convolutions (Figure 3). Regarding possible feature transformations, Decomposed Transposed Convolution is strictly a subset of regular Transposed Convolution. As an advantage, the number of trainable parameters is reduced (Table 1).

Decomposed Transposed Convolution was successfully applied in the inception architecture [31] where it achieved state of the art results on ILSVRC2012 [29]. It was also used to reduce the number of parameters of the network in [1].

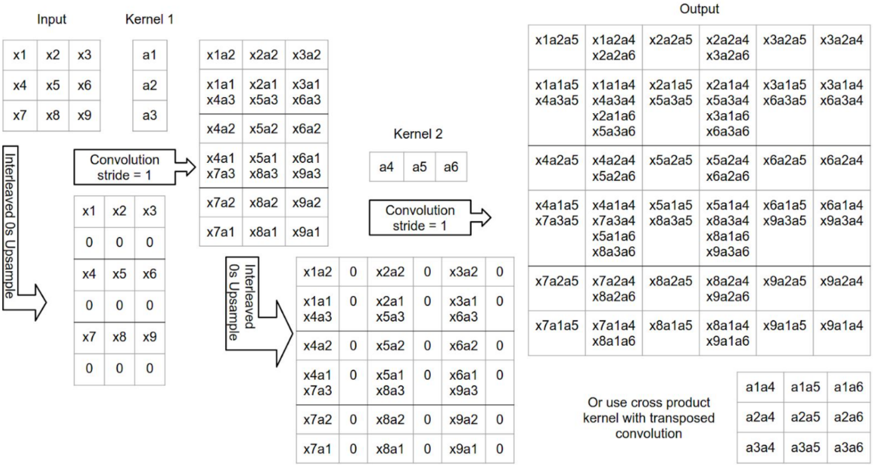

Input
x1 x2 x3
x4 x5 x6
x7 x8 x9
Kernel 1
a1
a2
a3
Interleaved 0s Upsample
Convolution stride = 1
x1 x2 x3
0 0 0
x4 x5 x6
0 0 0
x7 x8 x9
0 0 0
x1a2 x2a2 x3a2
x1a1 x4a3 x2a1 x5a3 x3a1 x6a3
x4a2 x5a2 x6a2
x4a1 x7a3 x5a1 x8a3 x6a1 x9a3
x7a2 x8a2 x9a2
x7a1 x8a1 x9a1
Interleaved 0s Upsample
Kernel 2
a4 a5 a6
Convolution stride = 1
x1a2 0 x2a2 0 x3a2 0
x1a1 x4a3 0 x2a1 x5a3 0 x3a1 x6a3 0
x4a2 0 x5a2 0 x6a2 0
x4a1 x7a3 0 x5a1 x8a3 0 x6a1 x9a3 0
x7a2 0 x8a2 0 x9a2 0
x7a1 0 x8a1 0 x9a1 0
Output
x1a2a5 x1a2a4 x2a2a6 x2a2a5 x2a2a4 x3a2a6 x3a2a5 x3a2a4
x1a1a5 x4a3a5 x1a1a4 x4a3a4 x2a1a6 x5a3a6 x2a1a5 x5a3a5 x2a1a4 x5a3a4 x3a1a6 x6a3a6 x3a1a5 x6a3a5 x3a1a4 x6a3a4
x4a2a5 x4a2a4 x5a2a6 x5a2a5 x5a2a4 x6a2a6 x6a2a5 x6a2a4
x4a1a5 x7a3a5 x4a1a4 x7a3a4 x5a1a6 x8a3a6 x5a1a5 x8a3a5 x5a1a4 x8a3a4 x8a1a6 x9a3a6 x6a1a5 x9a3a5 x6a1a4 x9a3a4
x7a2a5 x7a2a4 x8a2a6 x8a2a5 x8a2a4 x9a2a6 x9a2a5 x9a2a4
x7a1a5 x7a1a4 x8a1a6 x8a1a5 x8a1a4 x9a1a6 x9a1a5 x9a1a4
Or use cross product kernel with transposed convolution
a1a4 a1a5 a1a6
a2a4 a2a5 a2a6
a3a4 a3a5 a3a6

Figure 3: Decomposed transposed convolution.

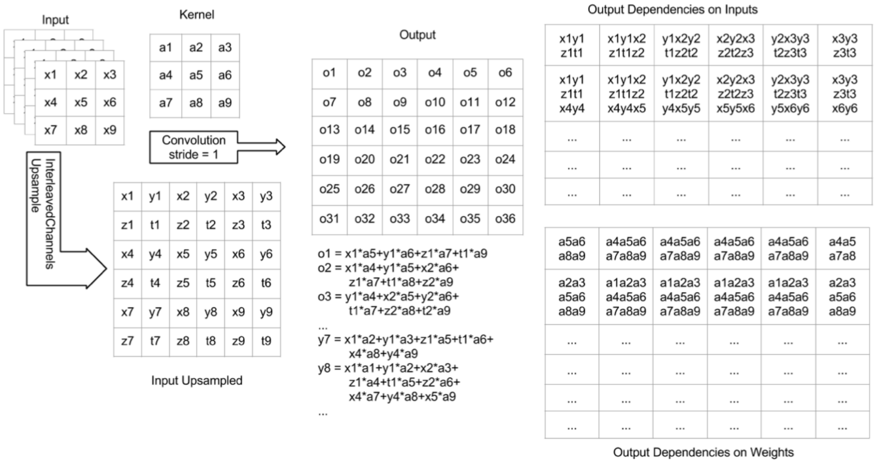
Separable Transposed Convolution. Separable Convolution were used to build a simple and homogenous network architecture [2] which achieved superior results to inception-v3 [31]. A Separable Convolution consists of two operations, a per channel convolution and a pointwise convolution with 1×1 kernel which mixes the channels. Separable transposed convolution is defined in the same way through applying the transposed convolution (Figure 2) however, now on every single channel separately. Afterward, a pointwise 1×1 convolutional kernel is applied. Again, regarding feature transformations, Separable Transposed Convolutions are a strict subset of Transposed Convolutions. In most cases, it has significantly fewer parameters than even Decomposed Transposed Convolutions (Table 1).

Depth To Space. Depth to Space operation [30] (also called subpixel convolution) shifts the feature channels into the spatial domain as illustrated in Figure 4. Depth To Space preserves perfectly all floats inside the high dimensional representation of the image, as it only changes their placement. The drawback of this approach is that it introduces alignment artifacts. To be comparable with other upsampling layers which have learnable parameters, before depth to space transformation we are applying a convolution with four times more output channels than for other upsampling layers.

Bilinear Upsampling. Bilinear Interpolation is another common approach for upsampling the spatial resolution. To be comparable with other methods we assume there is additional convolutional operation applied after the upsampling. The drawback of this strategy is that it is both memory and computationally intensive: bilinear interpolation increases the feature size quadratically while keeping the same amount of 'information' counted in the number of floats. Because the bilinear upsampling is followed by a convolution, the resulting upsampling method is four times more expensive than a transposed convolution.

2.1.2 Bilinear additive upsampling

To overcome the memory and computational problems of bilinear upsampling, we introduce a new upsampling layer: bilinear additive upsampling. In this layer, we propose to do bilinear upsampling as before, but we also add every N consecutive channels together, effectively reducing the output by a factor N. This process is illustrated in Figure 5. Please note that this process is deterministic and has zero tunable parameters (similarly to Depth To Space



Input

x1	x2	x3
x4	x5	x6
x7	x8	x9

Kernel

a1	a2	a3
a4	a5	a6
a7	a8	a9

Input Upsampled

x1	y1	x2	y2	x3	y3
z1	t1	z2	t2	z3	t3
x4	y4	x5	y5	x6	y6
z4	t4	z5	t5	z6	t6
x7	y7	x8	y8	x9	y9
z7	t7	z8	t8	z9	t9

Output

o1	o2	o3	o4	o5	o6
o7	o8	o9	o10	o11	o12
o13	o14	o15	o16	o17	o18
o19	o20	o21	o22	o23	o24
o25	o26	o27	o28	o29	o30
o31	o32	o33	o34	o35	o36

o1 = x1*a5+y1*a6+z1*a7+t1*a9
o2 = x1*a4+y1*a5+x2*a6+
z1*a7+t1*a8+z2*a9
o3 = y1*a4+x2*a5+y2*a6+
t1*a7+z2*a8+t2*a9
...
y7 = x1*a2+y1*a3+z1*a5+t1*a6+
x4*a8+y4*a9
y8 = x1*a1+y1*a2+x2*a3+
z1*a4+t1*a5+z2*a6+
x4*a7+y4*a8+x5*a9
...

Output Dependencies on Inputs

x1y1 z1t1	x1y1x2 z1t1z2	y1x2y2 t1z2t2	x2y2x3 z2t2z3	y2x3y3 t2z3t3	x3y3 z3t3
x1y1 z1t1 x4y4	x1y1x2 z1t1z2 x4y4x5	y1x2y2 t1z2t2 y4x5y5	x2y2x3 z2t2z3 x5y5x6	y2x3y3 t2z3t3 y5x6y6	x3y3 z3t3 x6y6
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

a5a6 a8a9	a4a5a6 a7a8a9	a4a5a6 a7a8a9	a4a5a6 a7a8a9	a4a5a6 a7a8a9	a4a5 a7a8
a2a3 a5a6 a8a9	a1a2a3 a4a5a6 a7a8a9	a1a2a3 a4a5a6 a7a8a9	a1a2a3 a4a5a6 a7a8a9	a1a2a3 a4a5a6 a7a8a9	a2a3 a5a6 a8a9
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...

Output Dependencies on Weights

Figure 4: Depth To Space.

upsampling, but doesn't cause alignment artifacts). Therefore, to be comparable with other upsampling methods we apply a convolution after this upsampling method. In this paper, we choose N in such a way that the final number of floats before and after bilinear additive upsampling is equivalent (we upsample by a factor 2 and choose $N = 4$), which makes the costs of this upsampling method similar to a transposed convolution.

2.2 Skip Connections and Residual Connections

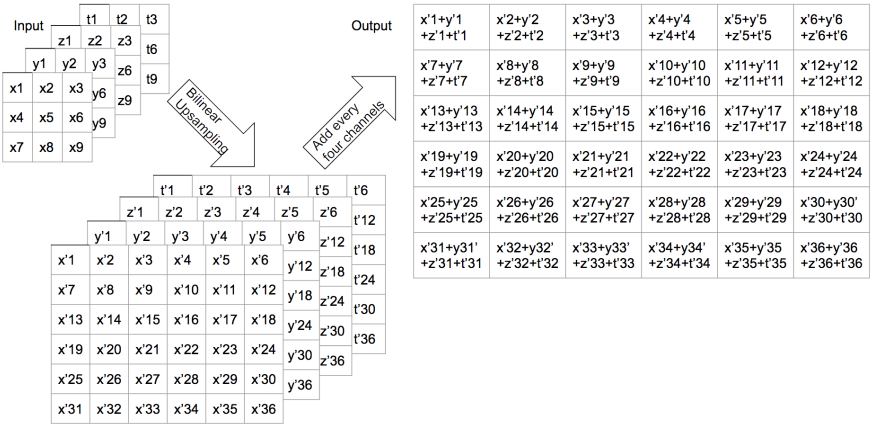
2.2.1 Skip Connections

Skip connections have been successfully used in many decoder architectures [12, 17, 18, 27, 28]. It uses features from the encoder in the decoder part of the same spatial resolution, as illustrated in Figure 1. For our implementation of skip connections, we apply the convolution on the last layer of encoded features for given spatial resolution and concatenate them with the first layer of decoded features for given spatial resolution as illustrated in Figure 1.

2.2.2 Residual Connections for decoders

Residual connections[9] have been shown to be beneficial for a variety of tasks. However, residual connections cannot be directly applied to upsampling methods since the output layer has a higher spatial resolution than the input layer and a lower number of feature channels. In this paper, we introduce a transformation which solves both problems.

In particular, the bilinear additive upsampling method which we introduced above (Figure 5) transforms the input layer into the desired spatial resolution and number of channels without using any parameters. The resulting features contain much of the information of the original features. Therefore we can apply this transformation (this time without doing any convolution) and add its result to the output of any upsampling layer, resulting in a residual-like connection. We demonstrate the effectiveness of our residual connection in Section 4.



x'1+y'1 +z'1+t'1	x'2+y'2 +z'2+t'2	x'3+y'3 +z'3+t'3	x'4+y'4 +z'4+t'4	x'5+y'5 +z'5+t'5	x'6+y'6 +z'6+t'6
x'7+y'7 +z'7+t'7	x'8+y'8 +z'8+t'8	x'9+y'9 +z'9+t'9	x'10+y'10 +z'10+t'10	x'11+y'11 +z'11+t'11	x'12+y'12 +z'12+t'12
x'13+y'13 +z'13+t'13	x'14+y'14 +z'14+t'14	x'15+y'15 +z'15+t'15	x'16+y'16 +z'16+t'16	x'17+y'17 +z'17+t'17	x'18+y'18 +z'18+t'18
x'19+y'19 +z'19+t'19	x'20+y'20 +z'20+t'20	x'21+y'21 +z'21+t'21	x'22+y'22 +z'22+t'22	x'23+y'23 +z'23+t'23	x'24+y'24 +z'24+t'24
x'25+y'25 +z'25+t'25	x'26+y'26 +z'26+t'26	x'27+y'27 +z'27+t'27	x'28+y'28 +z'28+t'28	x'29+y'29 +z'29+t'29	x'30+y30' +z'30+t'30
x'31+y31' +z'31+t'31	x'32+y32' +z'32+t'32	x'33+y33' +z'33+t'33	x'34+y34' +z'34+t'34	x'35+y'35 +z'35+t'35	x'36+y'36 +z'36+t'36

Figure 5: Bilinear additive upsampling, example for an input image with 4 channels.

Upsampling method	# of parameters	# of operations	Comments
Transposed	$whIO$	$whWHIO$	
Dec. Transposed	$(w+h)IO$	$(w+h)WHIO$	Subset of Transposed
Sep. Transposed	$whI+IO$	$whWHI+WHIO$	Subset of Transposed
Conv and Depth To Space	$whI(4O)$	$whWHI(4O)$	
Bilinear with Conv	$whIO$	$wh(2W)(2H)IO$	
Bilinear additive with Conv	$whIO$	$wh(2W)(2H)(I/4)O$	

Table 1: Comparison of different upsampling methods. W,H - feature width and height, w,h - kernel width and height, I,O - number of channels for input and output features.

3 Tasks and Experimental Setup

Instance boundaries detection. For instance-wise boundaries, we use PASCAL VOC 2012 segmentation [7]. This dataset contains 1,464 training and 1,449 validation images, annotated with contours for 20 object classes for all instances. The dataset was originally designed for semantic segmentation. Therefore only interior object pixels are marked, and the boundary location is recovered from the segmentation mask. Similar to [13, 32], we consider only object boundaries without distinguishing semantics, treating all 20 classes as one.

As encoder or feature extractor we use ResNet-50 with stride 8, atrous convolution, initialized with the pre-trained weights. The input to the network is of size 321 × 321. The spatial resolution is reduced to 41 × 41, after which we use 3 upsampling layers with additional convolution layer between to make predictions in the original resolution.

During training, we augment the dataset through rescaling the images by a random factor between 0.5 and 2.0 and random cropping. We train the network with asynchronous stochastic gradient descent for 40,000 iterations using a momentum of 0.9. We use a learning rate of 0.0003 with a polynomial decay of power 0.99. We apply L2 regularization with weight decay 0.0002. We use a batch size 5. We use sigmoid cross entropy loss per pixel (averaged across all pixels), where 1 represents an edge, and 0 represents a non-edge pixel.

Edge detection is evaluated using two measures: f-measure for the best-fixed contour threshold across the entire dataset and average precision (AP). During the evaluation, predicted contour pixels within three from ground truth pixels are assumed to be correct [21].

Super resolution. For super-resolution, we test our approach on the CelebA dataset, which consists of $167,483$ training images and $29,249$ validation images [19]. We follow the setup from [33]: the input images of the network are 16×16 images, which are created by resizing the original images. The goal is to reconstruct the original images which have a resolution of 128×128.

The network architecture used for super-resolution is similar to the one from [14]. We use six resnet-v1 blocks with 32 channels after which we upsample by a factor of 2. We repeat this three times to get to a target upsampling factor of 8. On top of this, we add 2 pointwise convolutional layers with 682 channels with batch normalization in the last layer. Note that in this problem there are only operations which keep the current spatial resolution or which upsample the representation. We train the network on a single machine with 1 GPU, batch size 32, using RMSProp optimizer with a momentum of 0.9, a decay of 0.95 and a batch size of 16. We fix the learning rate at 0.001 for 30000 iterations. We apply L2 regularization with weight decay 0.0005. The network is trained from scratch.

As loss we use the averaged L2 loss between the predicted residuals $\hat{y}$ and actual residuals y. The ground truth residual y in the loss function is the difference between original 128×128 target image and the predicted upsampled image. All target values are scaled to [-1,1]. We evaluate performance using standard metrics for super-resolution: PSNR and SSIM.

Colorization. We train and test our models on the ImageNet dataset [29], which consists of $1,000,000$ training images and $50,000$ validation images.

For the network architecture, we follow [10], where we swap their original bilinear upsampling method with the methods described in Secion 2 in particular, these are three upsampling steps of factor 2.

This model combines joint training of image classification and colorization, where we are mainly interested in the colorization part. The input image is resized to 224×224 pixels. We train the network for $30,000$ iterations using the Adam optimizer with a batch size of 32, fixed the learning rate to 1.0. We apply L2 regularization with weight decay 0.0001. During training, we randomly crop input image and perform random flipping.

As loss function we use the averaged L1 loss for pixel-wise color differences for the colorization part, and a softmax cross entropy loss for the classification part.

$$Loss(y, \hat{y}, y_{cl}, \hat{y_{cl}}) = 10|y - \hat{y}| - y_{cl} \log \hat{y_{cl}} \tag{1}$$

Color predictions are made in the YPbPr color space (luminance, blue - luminance, red - luminance), the luminance is ignored in the loss function during both training and evaluation as is it provided by the input greyscale image. The output pixel value targets are scaled to the range [0,1]. $y_c l$ is a one hot encoding of the predicted class label and $\hat{y_{cl}}$ are the predicted classification logits.

To evaluate colorization we follow [35]. We compute the average of root mean squared error between the color channels in the predicted and ground truth pixels. Then for different thresholds for root mean squared errors we calculate the accuracy of correctly predicted colored pixels within given range. Based on these we compute Area Under the Curve [35]. Additionally, we calculate the top-1 and top-5 Inception-v3 [31] classification accuracy for the colorized images on ImageNet dataset motivated by the assumption that better recognition corresponds to more realistic images.

Depth. We apply our method to depth prediction on NYUDepth v2 dataset [22]. We train using the entire NYUDepth v2 raw data distribution, using the official split. There are $209,822$ train and test $187,825$ images.

As encoder network, we are using ResNet-50 with stride 8 and atrous convolution, initialized with the pre-trained weights. We use input size 304×228 $(width \times height)$. Then we upsample three times with convolutional layers in between to get back to original resolution.

We train the network with asynchronous stochastic gradient descent on 20 machines with a momentum of 0.9 and batch size 16, using a fixed learning rate 0.001. We train for $30,000$ iterations. We apply L2 regularization with weight decay 0.0005. We augment the dataset through random changes in brightness, hue, saturation, random color removal and mirroring.

For depth prediction, we use the reverse Huber following [15].

$$Loss(y,\hat{y}) = \begin{cases} |y-\hat{y}| & for \quad |y-\hat{y}| <= c \\ |y-\hat{y}|^2 & for \quad |y-\hat{y}| > c \end{cases} \tag{2}$$

$$c = \frac{1}{5} \max_{(b,h,w)\in[1...Batch\ Size][1...Height][1...Width]} |y_{b,h,w} - y_{b,\hat{h},w}| \tag{3}$$

The reverse Huber loss is equal to the $L1(x) = |x|$ norm for $x \in [-c,c]$ and equal to L2 norm outside this range. In every gradient descent step c is set to 20% of the maximal pixel error in the batch.

For evaluation we use the metrics from [6] i.e. mean relative error, root mean squared error, root mean squared log error, the percentage of correct prediction within three relative thresholds: $1.25, 1.25^2, 1.25^3$.

Semantic segmentation. We evaluate our approach on the standard PASCAL VOC-2012 dataset [7]. We use both the training dataset and augmented dataset [8] which together consists of 10,582 images. We test on the VOC Pascal 2012 validation dataset of 1,449 images.

As the encoder network, we are using ResNet-50 with stride 8 and atrous convolution, initialized with pre-trained weights on ImageNet dataset with an input size of 321×321. Decoder upsamples three times with factor 2 with the convolutional layer between them.

We train the network with asynchronous stochastic gradient descent on ten machines with a momentum of 0.9 and batch size 12, starting from learning rate 0.001 with polynomial decay with the power 0.9 for 100,000 iterations. We apply L2 regularization with weight decay of 0.0001. We randomly augment the dataset through rescaling the images by a random factor between 0.5 and 2.0 and random cropping of size 321×321.

We train and evaluate following the setup from [20]. We train the model using maximum likelihood estimation per pixel (softmax cross entropy) and use mIOU (mean Intersection Over Union) to benchmark the models.

4 Results

Our results are presented in Table 2. For the sake of discussion, since all evaluation metrics are highly correlated, this table only reports a single metric per problem. A table with all metrics can be found in the supplementary material. We first discuss the upper half of this table, which compares the upsampling types described in Section 2.1 on our target problems, both with and without the skip layer. Afterward, we discuss the benefits of adding our residual connections, resulting in the bottom half of Table 2.

Results without residual-like connections. For semantic segmentation, the use of skip-layers improves performance. Separable transposed convolutions are the best upsampling

Problem	Residual	Skip	Semantic Segmentation	Depth Prediction	Colorization	Super-resolution	Instance Edge Detection
Dataset			VOC Pascal 2012	NYUv2 Depth	ImageNet	CelebA	VOC Pascal 2012
Metric			mIoU	MRE	AUC	SSIM	f-measure
Transposed	N	N	0.651	0.162	0.951	0.674	0.248
Transposed	N	Y	0.659	0.165	0.954		0.639
Decomposed Transposed	N	N	0.642	0.164	0.951	0.68	0.458
Decomposed Transposed	N	Y	0.652	0.166	0.953		0.522
Separable Transposed	N	N	0.659	0.163	0.952	0.676	0.57
Separable Transposed	N	Y	0.671	0.164	0.948		0.57
Upsample Bilinear + Conv	N	N	0.62	0.198	0.949	0.593	0.451
Upsample Bilinear + Conv	N	Y	0.656	0.174	0.949		0.565
Conv + Depth To Space	N	N	0.649	0.162	0.95	0.596	0.5
Conv + Depth To Space	N	Y	0.644	0.174	0.953		0.647
Bilinear Additive Upsampling + Conv	N	N	0.661	0.165	0.949	0.594	0.619
Bilinear Additive Upsampling + Conv	N	Y	0.669	0.169	0.952		0.653
Transposed	Y	N	0.655	0.164	0.951	0.686	0.622
Transposed	Y	Y	0.659	0.164	0.953		0.295
Decomposed Transposed	Y	N	0.645	0.171	0.954	0.682	0.243
Decomposed Transposed	Y	Y	0.637	0.162	0.951		0.531
Separable Transposed	Y	N	0.669	0.166	0.945	0.683	0.61
Separable Transposed	Y	Y	0.652	0.165	0.946		0.517
Upsample Bilinear + Conv	Y	N	0.653	0.171	0.949	0.684	0.53
Upsample Bilinear + Conv	Y	Y	0.651	0.175	0.954		0.537
Conv + Depth To Space	Y	N	0.652	0.174	0.953	0.686	0.624
Conv + Depth To Space	Y	Y	0.653	0.17	0.944		0.62
Bilinear Additive Upsampling + Conv	Y	N	0.658	0.165	0.951	0.683	0.625
Bilinear Additive Upsampling + Conv	Y	Y	0.654	0.167	0.952		0.643

Table 2: Our main results comparing a variety of decoders on five machine vision problems. The upper part shows decoders without residual-like connections; the bottom shows decoders with residual-like connections. The colors represent relative performance: red means top performance, yellow means reasonable performance, blue means poor performance.

Method Measure	edge detection f-measure	super-resolution SSIM	colorization AUC	depth prediction MRE	sem. segmentation mIoU
Our method	0.63	0.68	0.951	0.165	0.658
Recent work	0.62 [13]	0.70 [33]	0.895 [35]	0.158 [6]	0.622 [20]

Table 3: Comparison of our bilinear additive upsampling + conv + res results with other methods from the literature.

method. For depth prediction, all layers except bilinear upsampling have good performance, whereas adding skip layers to these results in equal performance except for depth-to-space, where it slightly lowers performance. For colorization, all upsampling methods perform similarly, and the specific choice matters little. For superresolution, networks with skip-layers are not possible because there are no 'encoder' modules which high-resolution (and relatively low-semantic) features. Therefore this problem has no skip-layer entries. Regarding performance, only all transposed variants perform well on this task; other layers do not. For Instance Edge Detection, the skip-layer is necessary to get good results. The best performance is obtained by Transposed, depth-to-space, and bilinear additive upsampling.

Generalizing over problems, we see that (1) bilinear upsampling plus convolution is always inferior to other methods. (2) Skip layers make a difference: For semantic segmentation and instance edge detection, they give performance improvements. (3) Separable transposed convolutions have the most consistently good performance; only for instance edge detection, it does not reach top performance.

Results with residual-like connections. We now add our residual-like connections to all upsampling methods. Results are presented in the lower half of Table 2. For the majority of combinations, we see that adding residual connections is beneficial. Interestingly, we now can identify two upsampling methods which have consistently good results on *all* problems presented in this paper, both which have residual connections: (1) transposed convolutions + residual connections. (2) bilinear additive upsampling + residual connections (both with and without skip connections).

Finally, we compare our results with recent works in Table 3. This comparison shows that our used architectures are relatively strong and therefore well-suited for our evaluation experiments.

5 Conclusions

This paper provided an extensive evaluation for different decoder types on a broad range of machine vision applications. Our results demonstrate: (1) Decoders matter: there are significant performance differences between different decoders depending on the problem at hand. For example, skip layers were essential for both instance edge detection and semantic segmentation. (2) We introduced the bilinear additive upsampling layer, which considerably improves upon normal bilinear upsampling and which often results in top performance. (3) We introduced residual-like connections, which in most cases yield improvements when added to any upsampling layer. (4) There are two decoder types which give consistently top performance among the problems which we studied: (A) Transposed Convolutions with residual-like connections. (B) bilinear additive upsampling with residual-like connections. We recommend using either of these two decoder types for dense prediction tasks.

References

[1] Jose M. Alvarez and Lars Petersson. Decomposeme: Simplifying convnets for end-to-end learning. *CoRR*, abs/1606.05426, 2016. URL http://arxiv.org/abs/1606.05426.

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL http://arxiv.org/abs/1610.02357.

[3] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[4] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[5] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[6] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

[8] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 991–998. IEEE, 2011.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016.

[11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[12] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015. URL http://arxiv.org/abs/1511.02680.

[13] Anna Khoreva, Rodrigo Benenson, Mohamed Omran, Matthias Hein, and Bernt Schiele. Weakly supervised object boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 183–192, 2016.

[14] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016.

[15] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[16] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[17] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR*, abs/1611.06612, 2016. URL http://arxiv.org/abs/1611.06612.

[18] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. URL http://arxiv.org/abs/1612.03144.

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[21] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE, 2001.

[22] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[23] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[24] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

[25] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016.

[26] Junting Pan, Elisa Sayrol, Xavier Giro-i Nieto, Kevin McGuinness, and Noel E O'Connor. Shallow and deep convolutional networks for saliency prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 598–606, 2016.

[27] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[30] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[32] Jasper RR Uijlings and Vittorio Ferrari. Situational object boundary detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4712–4721, 2015.

[33] Xin Yu and Fatih Porikli. Ultra-resolving face images by discriminative generative networks. In *European Conference on Computer Vision*, pages 318–333. Springer, 2016.

[34] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[35] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

6 Appendix

Problem	Residual	Skip	Semantic Segmentation	Depth prediction						Colorization				Super-resolution		Instance Edge detection	
Dataset			VOC Pascal 2012	NYUv2 Depth						ImageNet				CelebA		VOC Pascal 2012	
Metric			mIoU	MRE	RMSE lin	MRSE log	acc 1.25	acc 1.56	acc 1.95	RMSE	AUC	top1	top5	SSIM	PSNR	f-measure	average precision
Transposed	N	N	0.651	0.162	0.712	0.221	0.773	0.944	0.984	0.053	0.952	0.610	0.840	0.675	23.030	0.248	0.046
Transposed	N	Y	0.659	0.165	0.731	0.224	0.764	0.942	0.984	0.050	0.954	0.611	0.841			0.640	0.599
Decomposed Transposed	N	N	0.642	0.165	0.722	0.225	0.775	0.940	0.982	0.053	0.951	0.605	0.840	0.681	22.980	0.458	0.330
Decomposed Transposed	N	Y	0.652	0.167	0.710	0.235	0.747	0.935	0.981	0.051	0.954	0.608	0.843			0.522	0.399
Separable Transposed	N	N	0.659	0.163	0.706	0.223	0.772	0.942	0.983	0.052	0.952	0.604	0.843	0.677	23.100	0.570	0.507
Separable Transposed	N	Y	0.671	0.165	0.745	0.225	0.768	0.941	0.982	0.056	0.949	0.613	0.843			0.570	0.482
Upsample Bilinear + Conv	N	N	0.620	0.198	0.831	0.264	0.711	0.914	0.971	0.055	0.950	0.598	0.835	0.593	18.700	0.451	0.437
Upsample Bilinear + Conv	N	Y	0.657	0.174	0.754	0.229	0.757	0.939	0.982	0.055	0.950	0.606	0.843			0.566	0.465
Conv + Depth To Space	N	N	0.649	0.163	0.753	0.223	0.766	0.942	0.984	0.054	0.950	0.588	0.830	0.596	20.110	0.501	0.322
Conv + Depth To Space	N	Y	0.645	0.174	0.707	0.230	0.744	0.940	0.985	0.051	0.954	0.609	0.843			0.648	0.601
Bilinear Additive Upsampling + Conv	N	N	0.661	0.165	0.715	0.226	0.767	0.940	0.983	0.055	0.950	0.606	0.840	0.594	18.930	0.620	0.638
Bilinear Additive Upsampling + Conv	N	Y	0.669	0.170	0.710	0.227	0.760	0.940	0.983	0.052	0.953	0.606	0.842			0.654	0.650
Transposed	Y	N	0.655	0.164	0.730	0.236	0.748	0.934	0.980	0.053	0.952	0.602	0.839	0.687	23.120	0.622	0.620
Transposed	Y	Y	0.660	0.165	0.732	0.227	0.767	0.938	0.981	0.051	0.953	0.610	0.844			0.296	0.183
Decomposed Transposed	Y	N	0.645	0.172	0.722	0.226	0.765	0.940	0.983	0.050	0.954	0.609	0.841	0.683	23.030	0.244	0.009
Decomposed Transposed	Y	Y	0.638	0.163	0.718	0.221	0.764	0.945	0.986	0.053	0.952	0.606	0.842			0.531	0.473
Separable Transposed	Y	N	0.670	0.166	0.707	0.222	0.773	0.943	0.984	0.059	0.945	0.587	0.826	0.684	23.020	0.610	0.608
Separable Transposed	Y	Y	0.653	0.166	0.716	0.225	0.768	0.941	0.984	0.058	0.947	0.614	0.844			0.517	0.464
Upsample Bilinear + Conv	Y	N	0.654	0.171	0.703	0.225	0.770	0.941	0.983	0.055	0.950	0.600	0.835	0.684	23.090	0.531	0.222
Upsample Bilinear + Conv	Y	Y	0.652	0.175	0.705	0.220	0.769	0.945	0.985	0.050	0.955	0.608	0.843			0.537	0.469
Conv + Depth To Space	Y	N	0.653	0.175	0.769	0.233	0.751	0.935	0.981	0.051	0.953	0.611	0.844	0.686	23.070	0.625	0.627
Conv + Depth To Space	Y	Y	0.653	0.170	0.788	0.233	0.743	0.937	0.982	0.060	0.945	0.604	0.837			0.629	0.615
Bilinear Additive Upsampling + Conv	Y	N	0.658	0.166	0.683	0.216	0.779	0.948	0.986	0.052	0.952	0.603	0.837	0.684	23.100	0.626	0.627
Bilinear Additive Upsampling + Conv	Y	Y	0.654	0.167	0.709	0.224	0.762	0.942	0.985	0.052	0.952	0.606	0.839			0.644	0.653

Table 4: The full results metrics related to 5 examined problems.