Learning to Generate Chairs with Convolutional Neural Networks

Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox
Department of Computer Science, University of Freiburg

Convolutional neural networks (CNNs) have been shown to be very successful on a variety of computer vision tasks, such as image classification [3, 4], detection and segmentation. All these tasks have in common that they can be posed as discriminative supervised learning problems, and hence can be solved using CNNs which are known to perform well given a large enough labeled dataset. Typically, a task solved by supervised CNNs involves learning mappings from raw sensor inputs to some sort of condensed, abstract output representation, such as object identity, position or scale. In this work, we stick with supervised training, but we turn the standard discriminative CNN upside down and use it to generate images given high-level information.

Given the set of 3D chair models of Aubry et al. [1], we aim to train a neural network capable of generating 2D projections of the models given the chair type, viewpoint, and, optionally, other parameters such as color, brightness, saturation, zoom, etc. Our neural network accepts as input these high-level values and produces an RGB image. We train it using standard backpropagation to minimize the Euclidean reconstruction error of the generated image.

It is not a surprise that a large enough neural network can perfectly approximate any function on the training set. In our case, a network potentially could just learn by heart all examples and provide perfect reconstructions of these, but would behave unpredictably when confronted with inputs it has not seen during training. We show that this is not what is happening, both because the network is too small to just remember all images, and because we observe generalization to previously unseen input data. Namely, we show that the network is capable of: 1) knowledge transfer: given limited number of viewpoints of an object, the network can use the knowledge learned from other similar objects to infer the remaining viewpoints; 2) interpolation between different objects; see Figure 1 for an example.

The structure of the generative network is shown in Figure 2. The network, which we formally refer to as $g(\mathbf{c}, \mathbf{v}, \boldsymbol{\theta})$, looks like a usual CNN turned upside down. It can be thought of as the composition of two processing steps $g = u \circ h$.

Layers FC-1 to FC-4 first build a shared, high dimensional hidden representation $h(\mathbf{c}, \mathbf{v}, \boldsymbol{\theta})$ from the input parameters. Within these layers the three input vectors are first independently fed through two fully connected layers with 512 neurons each, and then the outputs of these three streams are concatenated. This independent processing is followed by two fully connected layers with 1024 neurons each, yielding the response of the fourth fully connected layer (FC-4).

After these fully connected layers the network splits into two streams (layers FC-5 and uconv-1 to uconv-4), which independently generate the image and object mask from the shared hidden representation. We denote these streams $u_{RGB}(\cdot)$ and $u_{segm}(\cdot)$. Each of them consists of a fully connected layer, the output of which is reshaped to a 8×8 multichannel image and fed through 4 'unpooling+convolution' layers with 5×5 filters and 2×2 unpooling. Each layer, except the output layers, is followed by a rectified linear (ReLU) nonlinearity.

In order to map the dense 8×8 representation to a high dimensional image, we need to unpool the feature maps (i.e. increase their spatial span) as opposed to the pooling (shrinking the feature maps) implemented by usual CNNs. We perform unpooling by simply replacing each entry of a feature map by an $s \times s$ block with the entry value in the top left corner and zeros elsewhere (see Figure 3). This increases the width and the height of the feature map s times. We used $s = 2$ in our networks. This is similar to the "deconvolutional" layers used in previous work [2, 4, 5].

In the paper we describe the model in detail, analyze the internal functioning of the network and study generalization of the network to previously unseen input data. As an example of a practical application, we apply point tracking to 'morphings' of different chairs (as in Figure 1) to find correspondences between these chairs. We show that this simple method is more accurate than existing approaches.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

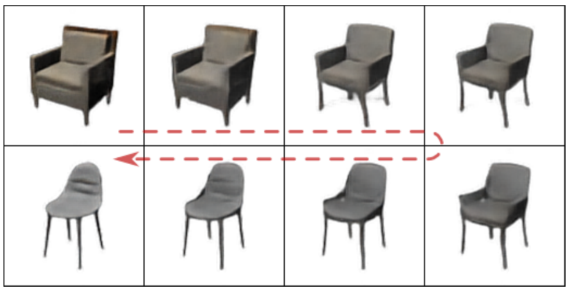

Figure 1: Interpolation between two chair models (original: top left, final: bottom left). The generative convolutional neural network learns the manifold of chairs, allowing it to interpolate between chair styles, producing realistic intermediate styles.

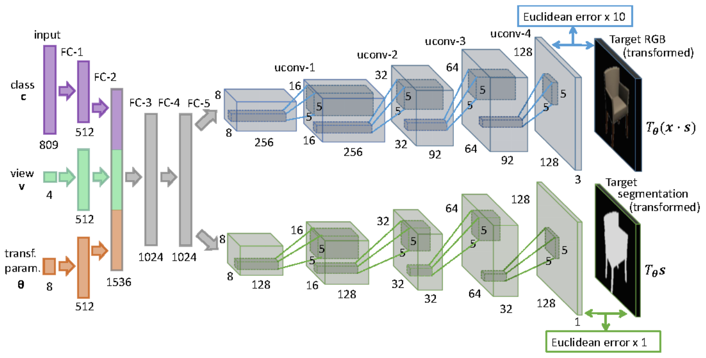

Figure 2: Architecture of the generative network. Layer names are shown above: FC - fully connected, uconv - unpooling+convolution.

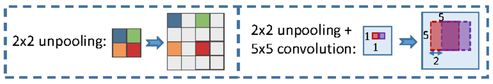

Figure 3: Illustration of unpooling (left) and unpooling+convolution (right) as used in the generative network.

[1] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[4] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[5] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE International Conference on Computer Vision, ICCV*, pages 2018–2025, 2011.