# Learning to Generate Chairs with Convolutional Neural Networks

Alexey Dosovitskiy Jost Tobias Springenberg Thomas Brox
Department of Computer Science, University of Freiburg
{dosovits, springj, brox}@cs.uni-freiburg.de

## Abstract

*We train a generative convolutional neural network which is able to generate images of objects given object type, viewpoint, and color. We train the network in a supervised manner on a dataset of rendered 3D chair models. Our experiments show that the network does not merely learn all images by heart, but rather finds a meaningful representation of a 3D chair model allowing it to assess the similarity of different chairs, interpolate between given viewpoints to generate the missing ones, or invent new chair styles by interpolating between chairs from the training set. We show that the network can be used to find correspondences between different chairs from the dataset, outperforming existing approaches on this task.*

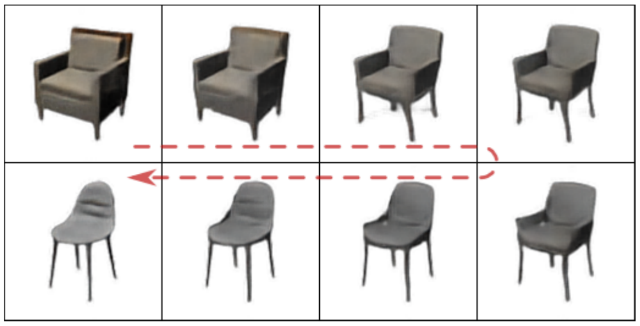
Figure 1. Interpolation between two chair models (original: top left, final: bottom left). The generative convolutional neural network learns the manifold of chairs, allowing it to interpolate between chair styles, producing realistic intermediate styles.

## 1. Introduction

Convolutional neural networks (CNNs) have been shown to be very successful on a variety of computer vision tasks, such as image classification [17, 5, 31], detection [9, 27] and segmentation [9]. 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 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.

In what follows we describe the model in detail, analyze the internal functioning of the network and study generalization of the network to unseen data, as described above. 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 method is more accurate than existing approaches.

## 2. Related work

Work on generative models of images typically addresses the problem of unsupervised learning of a data model which can generate samples from a latent representation. Prominent examples from this line of work are restricted Boltzmann machines (RBMs) [12] and Deep Boltzmann Machines (DBMs) [25], as well as the plethora of models derived from them [11, 21, 19, 28, 22]. RBMs and DBMs are undirected graphical models which aim to build a probabilistic model of the data and treat encoding and generation as an (intractable) joint inference problem.

A different approach is to train directed graphical models of the data distribution. This includes a wide variety of methods ranging from Gaussian mixture models [8, 30] to autoregressive models [18] and stochastic variations of neural networks [2, 10, 24, 16, 29]. Among them Rezende *et al*. [24] developed an approach for training a generative model with variational inference by performing (stochastic) backpropagation through a latent Gaussian representation. Goodfellow *et al*. [10] model natural images using a "deconvolutional" generative network that is similar to our architecture.

Most unsupervised generative models can be extended to incorporate label information, forming semi-supervised and conditional generative models which lie between fully unsupervised approaches and our work. Examples include: gated conditional RBMs [21] for modeling image transformations, training RBMs to disentangle face identity and pose information using conditional RBMs [23], and learning a generative model of digits conditioned on digit class using variational autoencoders [15]. In contrast to our work, these approaches are typically restricted to small models and images, and they often require an expensive inference procedure – both during training and for generating images.

The general difference of our approach to prior work on learning generative models is that we assume a high-level latent representation of the images is given and use supervised training. This allows us 1) to generate relatively large high-quality images of $128 \times 128$ pixels (as compared to maximum of $48 \times 48$ pixels in the aforementioned works) and 2) to completely control which images to generate rather than relying on random sampling. The downside is, of course, the need for a label that fully describes the appearance of each image.

Modeling of viewpoint variation is often considered in the context of pose-invariant face recognition [3, 33]. In a recent work Zhu *et al*. [34] approached this task with a neural network: their network takes a face image as input and generates a random view of this face together with the corresponding viewpoint. The network is fully connected and hence restricted to small images and, similarly to generative models, requires random sampling to generate a desired view. This makes it inapplicable to modeling large and diverse images, such as the chair images we model.

Our work is also loosely related to applications of CNNs to non-discriminative tasks, such as super-resolution [6] or inferring depth from a single image [7].

## 3. Model description

Our goal is to train a neural network to generate accurate images of chairs from a high-level description: class, orientation with respect to the camera, and additional parameters such as color, brightness, etc.

Formally, we assume that we are given a dataset of examples $D = \{(\mathbf{c}^1, \mathbf{v}^1, \theta^1), \dots, (\mathbf{c}^N, \mathbf{v}^N, \theta^N)\}$ with targets $O = \{(\mathbf{x}^1, \mathbf{s}^1), \dots, (\mathbf{x}^N, \mathbf{s}^N)\}$. The input tuples consist of three vectors: $\mathbf{c}$ is the class label in one-hot encoding, $\mathbf{v}$ – azimuth and elevation of the camera position (represented by their sine and cosine [1]) and $\theta$ – the parameters of additional artificial transformations applied to the images. The targets are the RGB output image $\mathbf{x}$ and the segmentation mask $\mathbf{s}$.

We include artificial transformations $T_\theta$ described by the randomly generated parameter vector $\theta$ to increase the amount of variation in the training data and reduce overfitting, analogous to data augmentation in discriminative CNN training [17]. Each $T_\theta$ is a combination of the following transformations: in-plane rotation, translation, zoom, stretching horizontally or vertically, changing hue, changing saturation, changing brightness.

### 3.1. Network architecture

We experimented with networks for generating images of size $64 \times 64$ and $128 \times 128$. The network architectures for both variations are identical except that the smaller network is reduced by one convolutional layer. The structure of the larger $128 \times 128$ generative network is shown in Figure 2.

Conceptually the generative network, which we formally refer to as $g(\mathbf{c}, \mathbf{v}, \theta)$, 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 $h(\mathbf{c}, \mathbf{v}, \theta)$ 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 $8 \times 8$ multichannel image and fed through 4 'unpool-

[1]We do this to deal with periodicity of the angle. If we simply used the number of degrees, the network would have no way to understand that 0 and 359 degrees are in fact very close.

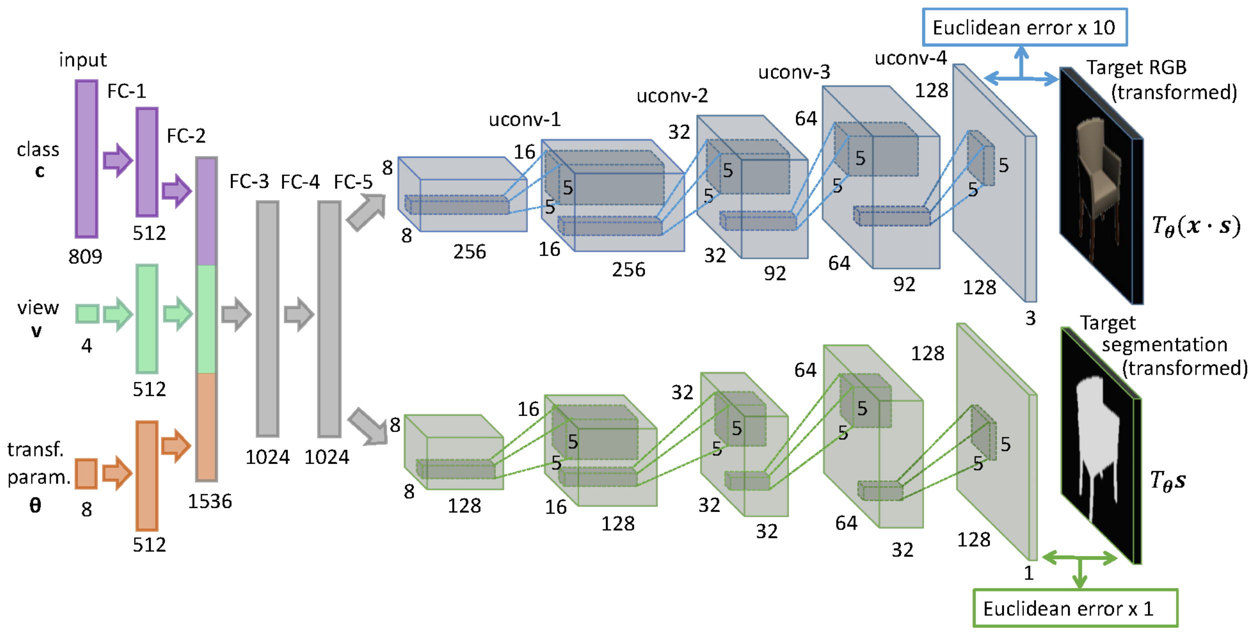

Figure 2. Architecture of the 128 $\times$ 128 network. Layer names are shown above: FC - fully connected, uconv - unpooling+convolution.

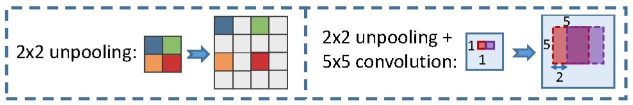

Figure 3. Illustration of unpooling (left) and unpooling+convolution (right) as used in the generative network.

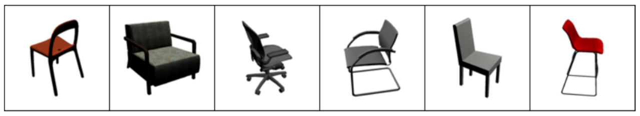

Figure 4. Several representative chair images used for training the network.

ing+convolution' layers with $5 \times 5$ filters and $2 \times 2$ unpooling. Each layer, except the output layers, is followed by a rectified linear (ReLU) nonlinearity.

In order to map the dense $8 \times 8$ 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. As illustrated in Figure 3 (left), 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. This increases the width and the height of the feature map $s$ times. We used $s = 2$ in our networks. When a convolutional layer is preceded by such an unpooling operation we can thus think of unpooling+convolution as the inverse operation of the convolution+pooling steps performed in a standard CNN (see Figure 3 right). This is similar to the "deconvolutional" layers used in previous work [31, 10, 32].

## 3.2. Generative training

The network parameters $\mathbf{W}$, consisting of all layer weights and biases, are then trained by minimizing the Euclidean error of reconstructing the segmented-out chair image and the segmentation mask (the weights $\mathbf{W}$ are omitted from the arguments of $h$ and $u$ for brevity of notation):

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \lambda \| u_{RGB}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i)) - T_{\theta^i}(\mathbf{x}^i \cdot \mathbf{s}^i) \|_2^2 \\ + \| u_{segm}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i)) - T_{\theta^i}\mathbf{s}^i \|_2^2, \tag{1}$$

where $\lambda$ is a weighting term, trading off between accurate reconstruction of the image and its segmentation mask respectively. We set $\lambda = 10$ in all experiments.

Note that although the mask could be inferred indirectly from the RGB image by exploiting monotonous background, we do not rely on this but rather require the network to explicitly output the mask. We never directly show these generated masks in the following, but we use them to add white background to the generated examples in many figures.

## 3.3. Dataset

As training data for the generative networks we used the set of 3D chair models made public by Aubry *et al.* [1].

More specifically, we used the dataset of rendered views they provide. It contains 1393 chair models, each rendered from 62 viewpoints: 31 azimuth angles (with step of 11 degrees) and 2 elevation angles (20 and 30 degrees), with a fixed distance to the chair. We found that the dataset includes many near-duplicate models, models differing only by color, or low-quality models. After removing these we ended up with a reduced dataset of 809 models, which we used in our experiments. We cropped the renders to have a small border around the chair and resized to a common size of $128 \times 128$ pixels, padding with white where necessary to keep the aspect ratio. Example images are shown in Figure 4. For training the network we also used segmentation masks of all training examples, which we produced by subtracting the monotonous white background.

### 3.4. Training details

For training the networks we built on top of the *caffe* CNN implementation [13]. We used stochastic gradient descent with a fixed momentum of 0.9. We first trained with a learning rate of 0.0002 for 500 passes through the whole dataset (epochs), and then performed 300 additional epochs of training, dividing the learning rate by 2 after every 100 epochs. We initialized the weights of the network with orthogonal matrices, as recommended by Saxe *et al.* [26].

When training the $128 \times 128$ network from scratch, we observed that its initial energy value never starts decreasing. Since we expect the high-level representation of the $64 \times 64$ and $128 \times 128$ networks to be very similar, we mitigated this problem by initializing the weights of the $128 \times 128$ network with the weights of the trained $64 \times 64$ network, except for the two last layers.

We used the $128 \times 128$ network in all experiments except for the viewpoint interpolation experiments in section 5.1. In those we used the $64 \times 64$ network to reduce computation time.

## 4. Analysis of the network

Neural networks are known to largely remain 'black boxes' whose function is hard to understand. In this section we provide an analysis of our trained generative network with the aim to obtain some intuition about its internal working. We only present the most interesting results here; more can be found in the supplementary material.

### 4.1. Network capacity

The first observation is that the network successfully models the variation in the data. Figure 5 shows results where the network was forced to generate chairs that are significantly transformed relative to the original images. Each row shows a different type of transformation. Images in the central column are non-transformed. Even in the presence of large transformations, the quality of the generated images

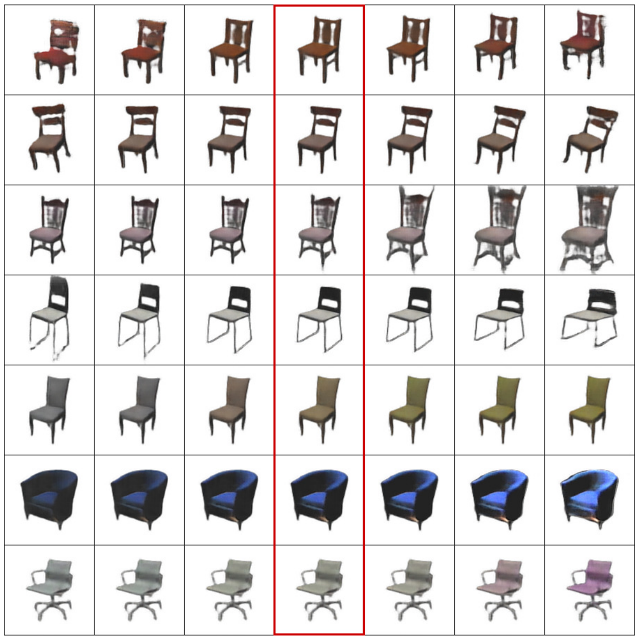

Figure 5. Generation of chair images while activating various transformations. Each row shows one transformation: translation, rotation, zoom, stretch, saturation, brightness, color. The middle column shows the reconstruction without any transformation.

is basically as good as without transformation. The image quality typically degrades a little in case of unusual chair shapes (such as rotating office chairs) and chairs including fine details such as armrests (see *e.g.* one of the armrests in row 7 in Figure 5) or thin elements in the back of the chair (row 3 in Figure 5).

An interesting observation is that the network easily deals with extreme color-related transformations, but has some problems representing large spatial changes, especially translations. Our explanation is that the architecture we use does not have means to efficiently model, say, translations: since transformation parameters only affect fully connected layers, the network needs to learn a separate 'template' for each position. A more complex architecture, which would allow transformation parameters to explicitly affect the feature maps of convolutional layers (by translating, rotating, zooming them) might further improve generation quality.

We did not extensively experiment with different network configurations. However, small variations in the network's depth and width did not seem to have significant effect on the performance. It is still likely that parameters such as the number of layers and their sizes can be further optimized.

The $128 \times 128$ network has approximately 32 million parameters, the majority of which are in the first fully connected layers of RGB and segmentation streams (FC-5): ap-

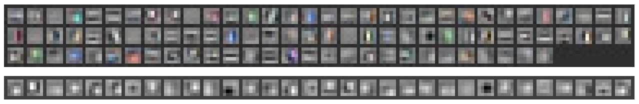
Figure 6. Output layer filters of the $128 \times 128$ network. **Top:** RGB stream. **Bottom:** Segmentation stream.

proximately 16 and 8 million, respectively. This is by far fewer than the approximately 400 million foreground pixels in the training data even when augmentation is not applied. When augmentation is applied, the training data size becomes virtually infinite. These calculations show that learning all samples by heart is not an option.

## 4.2. Activating single units

One way to analyze a neural network (artificial or real) is to visualize the effect of single neuron activations. Although this method does not allow us to judge about the network's actual functioning, which involves a clever combination of many neurons, it still gives a rough idea of what kind of representation is created by the different network layers.

Activating single neurons of uconv-3 feature maps (last feature maps before the output) is equivalent to simply looking at the filters of these layers which are shown in Figure 6. The final output of the network at each position is a linear combination of these filters. As to be expected, they include edges and blobs.

Our model is tailored to generate images from high-level neuron activations, which allows us to activate a single neuron in some of the higher layers and forward-propagate down to the image. The results of this procedure for different layers of the network are shown in Figures 7 and 9. Each row corresponds to a different network layer. The leftmost image in each row is generated by setting all neurons of the layer to zero, and the other images – by activating one randomly selected neuron.

In Figure 7 the first two rows show images produced when activating neurons of FC-1 and FC-2 feature maps of the class stream while keeping viewpoint and transformation inputs fixed. The results clearly look chair-like but do not show much variation (the most visible difference is chair vs armchair), which suggests that larger variations are achievable by activating multiple neurons. The last two rows show results of activating neurons of FC-3 and FC-4 feature maps. These feature maps contain joint class-viewpoint-transformation representations, hence the viewpoint is not fixed anymore. The generated images still resemble chairs but get much less realistic. This is to be expected: the further away from the inputs, the less semantic meaning there is in the activations. One interesting finding is that there is a 'zoom neuron' in layer FC-4 (middle image in the last row of Figure 7). When its value is increased, the

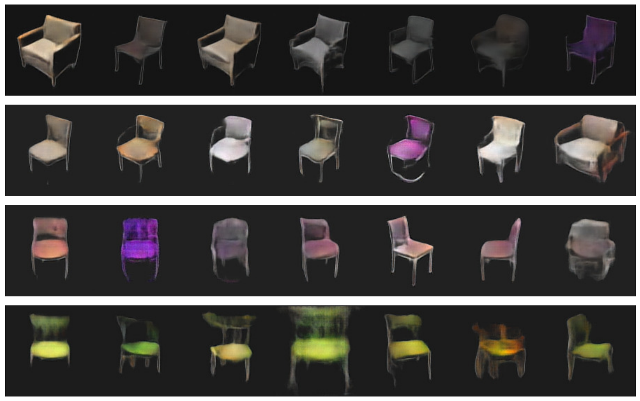
Figure 7. Images generated from single unit activations in feature maps of different fully connected layers of the $128 \times 128$ network. **From top to bottom:** FC-1 and FC-2 of the class stream, FC-3, FC-4.

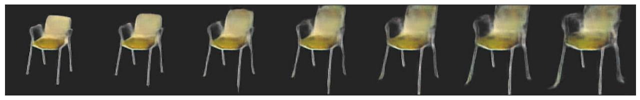
Figure 8. The effect of increasing the activation of the 'zoom neuron' we found in the layer FC-4 feature map.

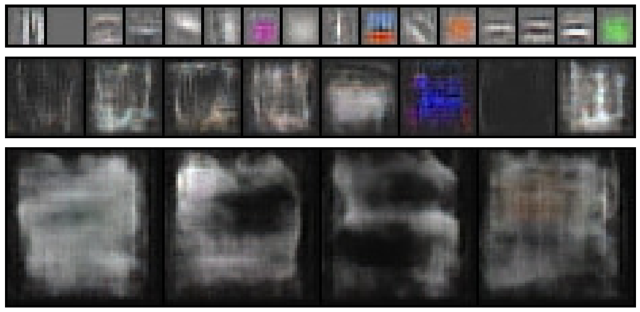
Figure 9. Images generated from single neuron activations in feature maps of some layers of the $128 \times 128$ network. From top to bottom: uconv-2, uconv-1, FC-5 of the RGB stream. Relative scale of the images is correct. Bottom images are $57 \times 57$ pixel, approximately half of the chair size.

output chair image gets zoomed. This holds not only for the case in which all other activations are zero, but also if the hidden representation contains the information for generating an actual chair, see Figure 8 for an example.

Images generated from single neurons of the convolutional layers are shown in Figure 9. A somewhat disappointing observation is that while single neurons in later layers (uconv-2 and uconv-3) produce edge-like images, the neurons of higher deconvolutional layers generate only blurry 'clouds', as opposed to the results of Zeiler and Fergus [31] with a classification network and max-unpooling. Our explanation is that because we use naive regular-grid unpooling, the network cannot slightly shift small parts to precisely arrange them into larger meaningful structures. Hence it must find another way to generate fine details. In the next subsection we show that this is achieved by a combination of spatially neighboring neurons.

## 4.3. Analysis of the hidden layers

Rather than just activating single neurons while keeping all others fixed to zero, we can use the network to normally generate an image and then analyze the hidden layer activations by either looking at them or modifying them and observing the results. An example of this approach was already used above in Figure 8 to understand the effect of the 'zoom neuron'. We present two more results in this direction here, and several more can be found in the supplementary material.

In order to find out how the blurry 'clouds' generated by single high-level deconvolutional neurons (Figure 9) form perfectly sharp chair images, we smoothly interpolate between a single activation and the whole chair. Namely, we start with the FC-5 feature maps of a chair, which have a spatial extent of $8 \times 8$. Next we only keep active neurons in a region around the center of the feature map (setting all other activations to zero), gradually increasing the size of this region from $2 \times 2$ to $8 \times 8$. Hence, we can see the effect of going from almost single-neuron activation level to the whole image level. The outcome is shown in Figure 10. Clearly, the interaction of neighboring neurons is very important: in the central region, where many neurons are active, the image is sharp, while in the periphery it is blurry. One interesting effect that is visible in the images is how sharply the legs of the chair end in the second to last image but appear in the larger image. This suggests highly non-linear suppression effects between activations of neighboring neurons.

Lastly some interesting observations can be made by taking a closer look at the feature maps of the uconv-3 layer (the last pre-output layer). Some of them exhibit regular patterns shown in Figure 11. These feature maps correspond to filters which look near-empty in Figure 6 (such as the 3rd and 10th filters in the first row). Our explanation of these patterns is that they compensate high-frequency artifacts originating from fixed filter sizes and regular-grid unpooling. This is supported by the last row of Figure 11 which shows what happens to the generated image when these feature maps are set to zero.

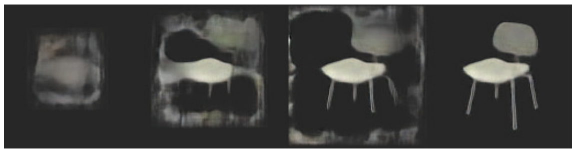

Figure 10. Chairs generated from spatially masked FC-5 feature maps (the feature map size is $8 \times 8$). The size of the non-zero region increases left to right: $2 \times 2$, $4 \times 4$, $6 \times 6$, $8 \times 8$.

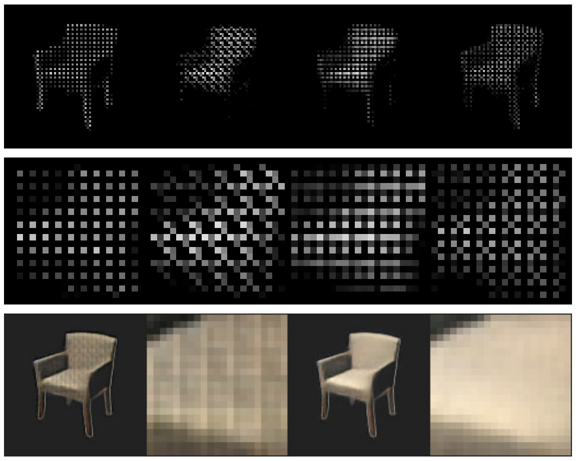

Figure 11. **Top:** Selected feature maps from the pre-output layer (uconv-3) of the RGB stream. These feature maps correspond to the filters which look near-empty in Figure 6. **Middle:** Close-ups of the feature maps. **Bottom:** Generation of a chair with these feature maps set to zero (left image pair) or left unchanged (right). Note the high-frequency artifacts in the left pair of images.

## 5. Experiments

### 5.1. Interpolation between viewpoints

In this section we show that the generative network is able to generate previously unseen views by interpolating between views present in the training data. This demonstrates that the network internally learns a representation of chairs which enables it to judge about chair similarity and use the known examples to generate previously unseen views.

In this experiment we use the $64 \times 64$ network to reduce computational costs. We randomly separate the chair styles into two subsets: the 'source set' with 90 % styles and the 'target set' with the remaining 10 % chairs. We then vary the number of viewpoints per style either in both these datasets together ('no transfer') or just in the target set ('with transfer') and then train the generative network as before. In the second setup the idea is that the network may use the knowledge about chairs learned from the source set (which includes all viewpoints) to generate the missing viewpoints of the chairs from the target set.

Figure 12 shows some representative examples of angle interpolation. For 15 views in the target set (first pair of rows) the effect of the knowledge transfer is already visible: interpolation is smoother and fine details are preserved better, for example a leg in the middle column. Starting from 8 views (second pair of rows and below) the network without knowledge transfer fails to produce satisfactory in-

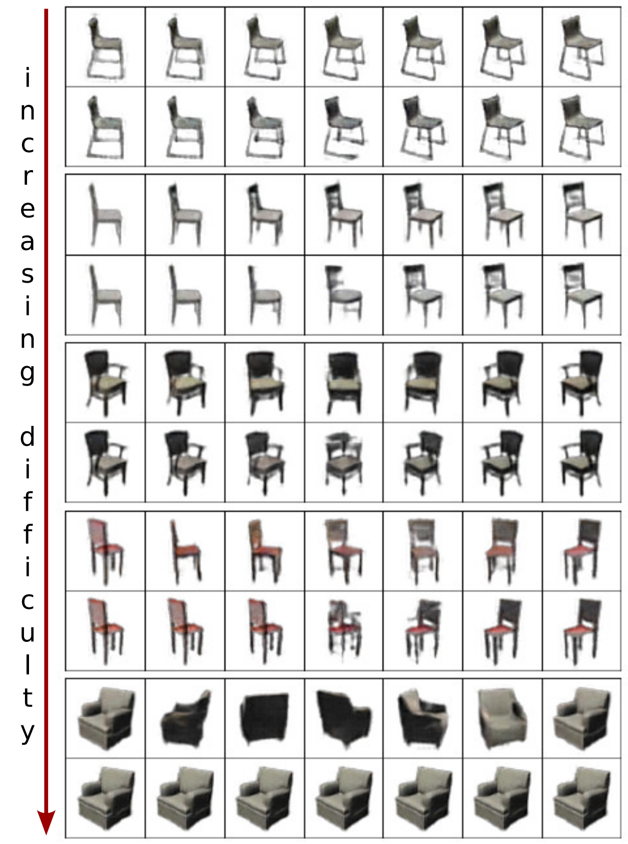

Figure 12. Examples of interpolation between angles. In each pair of rows the top row is with knowledge transfer and the second - without. In each row the leftmost and the rightmost images are the views presented to the network during training, while all intermediate ones are not and hence are results of interpolation. The number of different views per chair available during training is 15, 8, 4, 2, 1 (top-down). Image quality is worse than in other figures because we use the $64 \times 64$ network here.

terpolation, while the one with knowledge transfer works reasonably well even with just one view presented during training (bottom pair of rows). However, in this case some fine details, such as the armrest shape, are lost.

In Figure 13 we plot the average Euclidean error of the generated missing viewpoints from the target set, both with and without transfer (blue and green curves). Clearly, presence of all viewpoints in the source dataset dramatically improves the performance on the target set, especially for small numbers of available viewpoints.

One might suppose (for example looking at the bottom pair of rows of Figure 12) that the network simply learns all the views of the chairs from the source set and then, given a limited number of views of a new chair, finds the most similar one, in some sense, among the known models and simply returns the images of that chair. To check if this is the case,

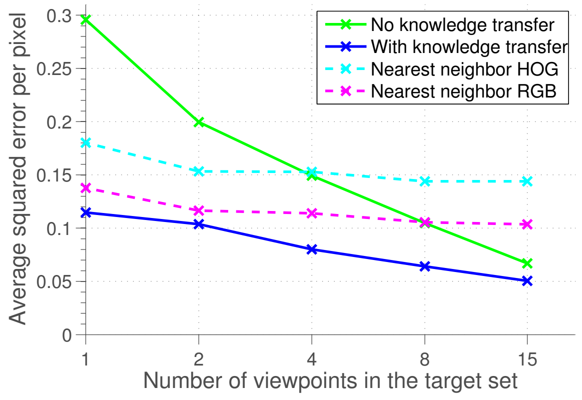

Figure 13. Reconstruction error for unseen views of chairs from the target set depending on the number of viewpoints present during training. Blue: all viewpoints available in the source dataset (knowledge transfer), green: the same number of viewpoints are available in the source and target datasets (no knowledge transfer).

we evaluate the performance of such a naive nearest neighbor approach. For each image in the target set we find the closest match in the source set for each of the given views and interpolate the missing views by linear combinations of the corresponding views of the nearest neighbors. For finding nearest neighbors we try two similarity measures: Euclidean distance between RGB images and between HOG descriptors. The results are shown in Figure 13. Interestingly, although HOG yields semantically much more meaningful nearest neighbors (not shown in figures), RGB similarity performs much better numerically. The performance of this nearest neighbor method is always worse than that of the network, suggesting that the network learns more than just linearly combining the known chairs, especially when many viewpoints are available in the target set.

## 5.2. Interpolation between classes

Remarkably, the generative network can interpolate not only between different viewpoints of the same object, but also between different objects, so that all intermediate images are also meaningful. To obtain such interpolations, we simply linearly change the input label vector from one class to another. Some representative examples of such morphings are shown in Figure 14. The images are sorted by subjective morphing quality (decreasing from top to bottom). The network produces very naturally looking morphings even in challenging cases, such as the first 5 rows. In the last three rows the morphings are qualitatively worse: some of the intermediate samples do not look much like real chairs. However, the result of the last row is quite intriguing as different types of intermediate leg styles are generated. More examples of morphings are shown in the supplementary material.

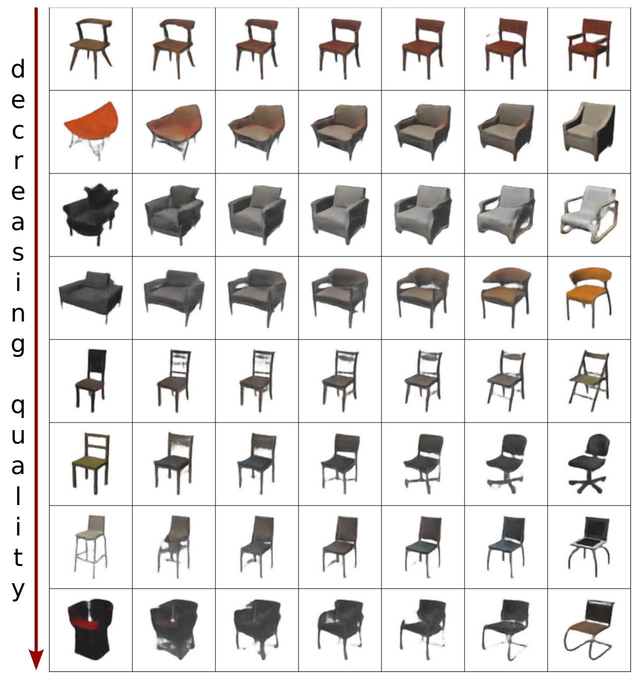

Figure 14. Examples of morphing different chairs, one morphing per row. Leftmost and rightmost chairs in each row are present in the training set, all intermediate ones are generated by the network. Rows are ordered by decreasing subjective quality of the morphing, from top to bottom.

### 5.2.1 Correspondences

The ability of the generative CNN to interpolate between different chairs allows us to find dense correspondences between different object instances, even if their appearance is very dissimilar.

Given two chairs from the training dataset, we use the $128 \times 128$ network to generate a morphing consisting of 64 images (with fixed view). We then compute the optical flow in the resulting image sequence using the code of Brox *et al*. [4]. To compensate for the drift, we refine the computed optical flow by recomputing it with a step of 9 frames, initialized by concatenated per-frame flows. Concatenation of these refined optical flows gives the global vector field that connects corresponding points in the two chair images.

In order to numerically evaluate the quality of the correspondences, we created a small test set of 30 image pairs (examples are shown in the supplementary material). We manually annotated several keypoints in the first image of each pair (in total 295 keypoints in all images) and asked 9 people to manually mark corresponding points in the second image of each pair. We then used mean keypoint positions in the second images as ground truth. At test time we measured the performance of different methods by computing average displacement of predicted keypoints in the second images given keypoints in the first images. We also manually annotated an additional validation set of 20 image pairs to tune the parameters of all methods (however, we were not able to search the parameters exhaustively because some methods have many).

| Method | All | Simple | Difficult |
|---|---|---|---|
| DSP [14] | 5.2 | 3.3 | 6.3 |
| SIFT flow [20] | 4.0 | **2.8** | 4.8 |
| Ours | **3.9** | 3.9 | **3.9** |
| Human | 1.1 | 1.1 | 1.1 |

Table 1. Average displacement (in pixels) of corresponding keypoints found by different methods on the whole test set and on the 'simple' and 'difficult' subsets.

In Table 1 we show the performance of our algorithm compared to human performance and two baselines: SIFT flow [20] and Deformable Spatial Pyramid [14] (DSP). To analyze the performance in more detail, we additionally manually separated the pairs into 10 'simple' ones (two chairs are quite similar in appearance) and 20 'difficult' ones (two chairs differ a lot in appearance). On average the very basic approach we used outperforms both baselines thanks to the intermediate samples produced by the generative neural network. More interestingly, while SIFT flow and DSP have problems with the difficult pairs, our algorithm does not. This suggests that errors of our method are largely due to contrast changes and drift in the optical flow, which does not depend on the difficulty of the image pair. The approaches are hence complementary: while for similar objects direct matching is fairly accurate, for more dissimilar ones intermediate morphings are very helpful.

## 6. Conclusions

We have shown that supervised training of convolutional neural network can be used not only for standard discriminative tasks, but also for generating images given high-level class, viewpoint and lighting information. A network trained for such a generative task does not merely learn to generate the training samples, but also generalizes well, which allows it to smoothly morph different object views or object instances into each other with all intermediate images also being meaningful. It is fascinating that the relatively simple architecture we proposed is already able to learn such complex behavior.

From the technical point of view, it is impressive that the network is able to process very different inputs – class label, viewpoint and the parameters of additional chromatic and spatial transformations – using exactly the same standard layers of ReLU neurons. It demonstrates again the wide applicability of convolutional networks.

## Acknowledgements

AD and TB acknowledge funding by the ERC Starting Grant VideoLearn (279401). JTS is supported by the BrainLinks-BrainTools Cluster of Excellence funded by the German Research Foundation (EXC 1086).

## References

[1] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014.

[2] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *ICML*, 2014.

[3] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1063–1074, Sept. 2003.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[6] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *13th European Conference on Computer Vision (ECCV)*, pages 184–199, 2014.

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[8] R. P. Francos, H. Permuter, and J. Francos. Gaussian mixture models of texture and colour for image database. In *ICASSP*, 2003.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[10] 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.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, pages 504–507, 2006.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[14] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *CVPR*, 2013.

[15] D. Kingma, D. Rezende, S. Mohamed, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[18] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. *JMLR: W&CP*, 15:29–37, 2011.

[19] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009.

[20] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. In *ECCV*, 2008.

[21] R. Memisevic and G. Hinton. Unsupervised learning of image transformations. In *CVPR*, 2007.

[22] M. Ranzato, J. Susskind, V. Mnih, and G. E. Hinton. On deep generative models with applications to recognition. In *CVPR*, pages 2857–2864. IEEE, 2011.

[23] S. Reed, K. Sohn, Y. Zhang, and H. Lee. Learning to disentangle factors of variation with manifold interaction. In *ICML*, 2014.

[24] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[25] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *AISTATS*, 2009.

[26] A. M. Saxe, J. L. McClelland, and S. Ganguli. Learning a nonlinear embedding by preserving class neighbourhood structure. In *ICLR*, 2014.

[27] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. CBLS, April 2014.

[28] Y. Tang and A.-R. Mohamed. Multiresolution deep belief networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS), 2012*, 2012.

[29] Y. Tang and R. Salakhutdinov. Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems 26*, pages 530–538. Curran Associates, Inc., 2013.

[30] L. Theis, R. Hosseini, and M. Bethge. Mixtures of conditional gaussian scale mixtures applied to multiscale image representations. *PLoS ONE*, 2012.

[31] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

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

[33] X. Zhang, Y. Gao, and M. K. H. Leung. Recognizing rotated faces from frontal and side views: An approach toward effective use of mugshot databases. pages 684–697, 2008.

[34] Z. Zhu, P. Luo, X. Wang, and X. Tang. Multi-view perceptron: a deep model for learning face identity and view representations. In *NIPS*, 2014.