Tunnel Effect in CNNs: Image Reconstruction from Max Switch Locations

Matthieu de La Roche Saint Andre*†¶, Laura Rieger*‡¶, Morten Hannemose*§, Junmo Kim¶
*These authors contributed equally to this work
†EFREI, France, ¶KAIST, South Korea, ‡Technische Universität Berlin, §Technical University of Denmark

Abstract—In this paper, we show that reconstruction of an image passed through a neural network is possible, using only the locations of the max pool activations. This was demonstrated with an architecture consisting of an encoder and a decoder. The decoder is a mirrored version of the encoder, where convolutions are replaced with deconvolutions and poolings are replaced with unpooling layers. The locations of the max pool switches are transmitted to the corresponding unpooling layer. The reconstruction is computed only from these switches without the use of feature values. Using only the max switch location information of the pool layers, a surprisingly good image reconstruction can be achieved. We examine this effect in various architectures, as well as how the quality of the reconstruction is affected by the number of features. We also compare the reconstruction with an encoder with randomly initialized weights with an encoder pretrained for classification. Finally, we give recommendations for future architecture decisions.

Index Terms—image reconstruction, convolutional neural networks, pooling, autoencoder, encoding, unpooling, deconvolution

I. Introduction

In the last decade, convolutional neural networks (CNNs) have been used to effectively solve various computer vision problems. However, there is still much to discover about CNNs' inner workings regarding different training techniques and architectures, despite attempts to gain more insight [1]. Springenberg et al. [2] compared several CNN architectures and concluded that pooling and fully connected layers could be replaced with convolution, without incurring any loss of performance.

In this paper, we introduce a surprising phenomenon that occurs while using max unpooling. Pooling layers are used in neural networks to reduce the size of the feature map. They combine the information of the receptive field of input neurons (e.g., a kernel of 2 by 2 pixels) and output a unique value per receptive field. This happens either by using the average of the entire receptive field, known as average pooling, or by outputting only the highest value, known as max pooling.

Unpooling layers [1] approximately reverse the effects of a pooling layer. Max unpooling utilizes max switch locations that are the positions of the input neurons with the highest values in the corresponding max pooling layer and fills the output neurons in these positions with the values. All other neurons are zero, resulting in a sparse feature map.

Copyright (c) 2016 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org. An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site.

Fig. 1. Image reconstruction from the ImageNet test set. Upper row: input images, lower row: output images.

When we refer to deconvolutional layers, it is the type of layer described in [1]. Because we have stride one, we could replace them with conventional convolutional layers. However, we decided to work with deconvolutional layers for easier understanding.

We found that reconstructing an image with deconvolutional and unpooling layers works exceptionally well as shown in Fig. 1, although no feature is used for reconstruction. The reconstruction relies solely on the max switch positions used in the unpooling layers. The information is effectively tunneled through a normally impenetrable wall: All features are discarded; only the information about their position is transmitted through the max switches.

This has potential repercussions for applications that manipulate the image in a higher-level feature space with a CNN. If the image reconstruction depends more on spatial information than on the actual values in the feature map, this information will also have to be manipulated. Likewise, if the max pool tunnel effect is undesirable, then steps will have to be taken to prevent the information from "leaking through" the max switches.

II. Related Work

To our knowledge, there is no literature on the reconstruction of images from only max switch location information. However, multiple papers use max unpooling layers to reconstruct an image from a sparse representation [3]–[7].

Zeiler et al. [6] train a network to learn low-, mid-, and high-level representations of a given image. After each convolutional layer, a deconvolutional layer uses its output along with the max switches to reconstruct the

TABLE I
Architecture of the Network.
Convolution and deconvolution: Kernel size 3, padding 1, stride 1. Pooling and unpooling: Kernel size 2, stride 2

Type	data	conv	conv	pool	conv	conv	pool	conv	conv	pool	bias	unpool	deconv	deconv	unpool	deconv	deconv	unpool	deconv	deconv	L2 Loss
#filters	3	n_1	n_1	n_1	n_2	n_2	n_2	n_3	n_3	n_3	n_3	n_3	n_3	n_2	n_2	n_2	n_1	n_1	n_1	3	1
Feature map size	128	128	128	64	64	64	32	32	32	16	16	32	32	32	64	64	64	128	128	128	128
ReLU	no	yes	yes	no	yes	yes	no	yes	yes	no	no	no	yes	yes	no	yes	yes	no	yes	no	-

image. By encouraging each layer to learn useful features for reconstruction, the network learns good representations on the low, middle, and high levels. Applying a standard classifier on those learned feature representations outperformed SIFT and gave highly competitive error rates for classification. The paper mentions that using max instead of average unpooling was crucial for retaining sharp reconstructions, emphasizing the importance of max unpooling for good spatial reconstruction.

Zhao et al. [5] presented the stacked what-where autoencoders (SWWAEs) that make it possible to train the same network with supervised and unsupervised data. A conventional CNN with convolutional layers followed by max pooling layers is used for supervised learning of classification. For learning with unsupervised data, a decoder network consisting of deconvolutional and max unpooling layers, which trains to reconstruct the original image, is added. The network is trained with a combination of classification loss for supervised data and L2 reconstruction loss for unsupervised data. The L2 reconstruction loss is made up of reconstruction loss at the input level, as well as middle reconstruction loss for the intermediate layers.

Zhang et al. [7] extended the work in [5] and compared several autoencoders regarding image reconstruction as a method for unsupervised learning. They concluded that the feature representation achieved with a deep CNN preserves the input image except for some locational details. Similar to our paper, they used unpooling layers with known max switches but focused on reconstruction with high-level features. They improved upon VGG16 [8] trained with ImageNet ILSVRC2012 as a baseline by using unsupervised training with autoencoders to encourage the preservation of useful high-level features in the encoder.

Noh et al. [3] used a CNN with deconvolutional and unpooling layers to create a semantically segmented version of the image. They reconstructed the input image by transmitting the position of the max switches to the unpooling layer. As in our network, the first half of the network was inspired by VGG16 [8]. The second half of the network mirrored this to enable the reconstruction of the semantically segmented image that goes from a coarse to a fine representation.

III. Experimental Setup

A. Dataset

We used a subset of ImageNet to evaluate the network architectures [9]. We used 40 000 images equally distributed over 100 categories with each image resized to 128×128 pixels.

The dataset was preprocessed by subtracting the mean value.

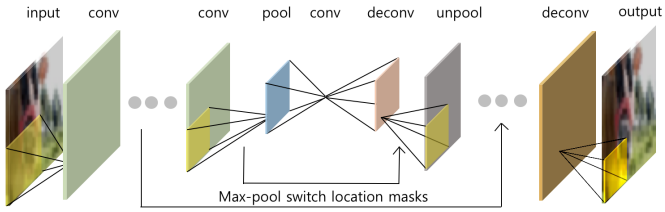

Fig. 2. Schematic overview of the network.

B. Architecture and Training

The network architecture is composed of two parts: the encoder and the decoder. The encoder contains all the convolutional and max pooling layers, and the decoder contains the deconvolutional and max unpooling layers. Fig. 2 visualizes the structure of the network. The encoder consists of three blocks, where each block is composed of two convolutional layers each followed by a rectified linear unit (ReLU), with a max pooling layer in the end of the block. This is in part inspired by VGG [8].

The decoder consists of three blocks as well, where each block contains one max unpooling layer followed by two deconvolutional layers. The middle layer between the encoder and the decoder we call a bias layer, because it has all weights connecting it to the previous layers set to zero, so it can only output constant values. It is implemented as a 1×1 convolution. This causes the only information about the input image that is used for reconstruction to be the max switch locations. No features are given to the decoder.

We will refer a specific architecture by $[n_1, n_2, n_3]$, where n_1 is the number of filters in the outermost pair of blocks and so on. We have experimented with architectures that have the same number of filters in each block and ones that double the number of filters for each successive block. The general architecture is shown in Table I. The results shown in Fig. 1 were produced with the [48,96,192], random encoder.

The network can be considered a degenerate autoencoder. The input data is the image that we are trying to reconstruct, represented as colored pixels. The code is the set of max switch location masks.

Each network architecture was initialized with random weights. If indicated, the network was then pretrained for classification for 50 000 iterations and then trained to reconstruct the original image for 50 000 iterations with L2 reconstruction loss.

If no pretraining occurred, the network was trained for reconstruction directly. Since backpropagation via the max switches is not possible due to the loss function not being differentiable with respect to the max-switch locations and there is no other connection between the encoder

TABLE II
RMSE and Bits used for all Architectures

Architecture	Reconstruction RMSE test [pixels in range [0;1]]		Max-switch information [bits per pixel]
Pretrained	No	Yes	
[8,8,8]	0.1464	0.1545	1.75
[6,12,24]	0.1387	0.1464	1.75
[16,16,16]	0.1210	0.1156	3.5
[12,24,48]	0.1138	0.1206	3.5
[32,32,32]	0.0922	0.0950	7
[24,48,96]	0.0876	0.0915	7
[64,64,64]	0.0744		14
[48,96,192]	0.0718		14

and the decoder, the encoder was not being trained for encoding useful features [5]. The decoder therefore learned to reconstruct the image from the max switch locations of the randomly transformed image if the network was not pretrained. All architectures were trained with Adagrad as the optimizer [10]. Instructions for reproducing our results can be found in Section VIII.

IV. Representation in Another Feature Space

The network maps an image represented as pixels, into a space of max switch locations. Since the size of the max switch representation in some cases is bigger than the size of the input image, this is not a compression but a transformation to another feature space. However, the switch locations do not hold information about the pixel values, only about where different features have their local maximal activations: the maximum switch location.

V. Results of image reconstruction from max pool locations

The reconstruction root mean square error (RMSE) for all architectures can be seen in Table II. For each network architecture, we also included the number of bits used to store the max switch locations per pixel in each of the three color channels. Each pixel in the original image takes up 8 bits per color channel. From this and Table II, it can be seen that only [64,64,64] and [48,96,192] take up more bits per pixel than the uncompressed original image. For most architectures we present the results with a pretrained architecture as described in Section III-B along with the results for randomly initialized weights. In Fig. 3, images from the test set and their reconstructions are shown.

VI. Discussion

In Section V, we showed the reconstruction of 128×128 pixel images from max pooling switches with different network architectures. We varied the number of filters for each (de)convolution unit which led to different information

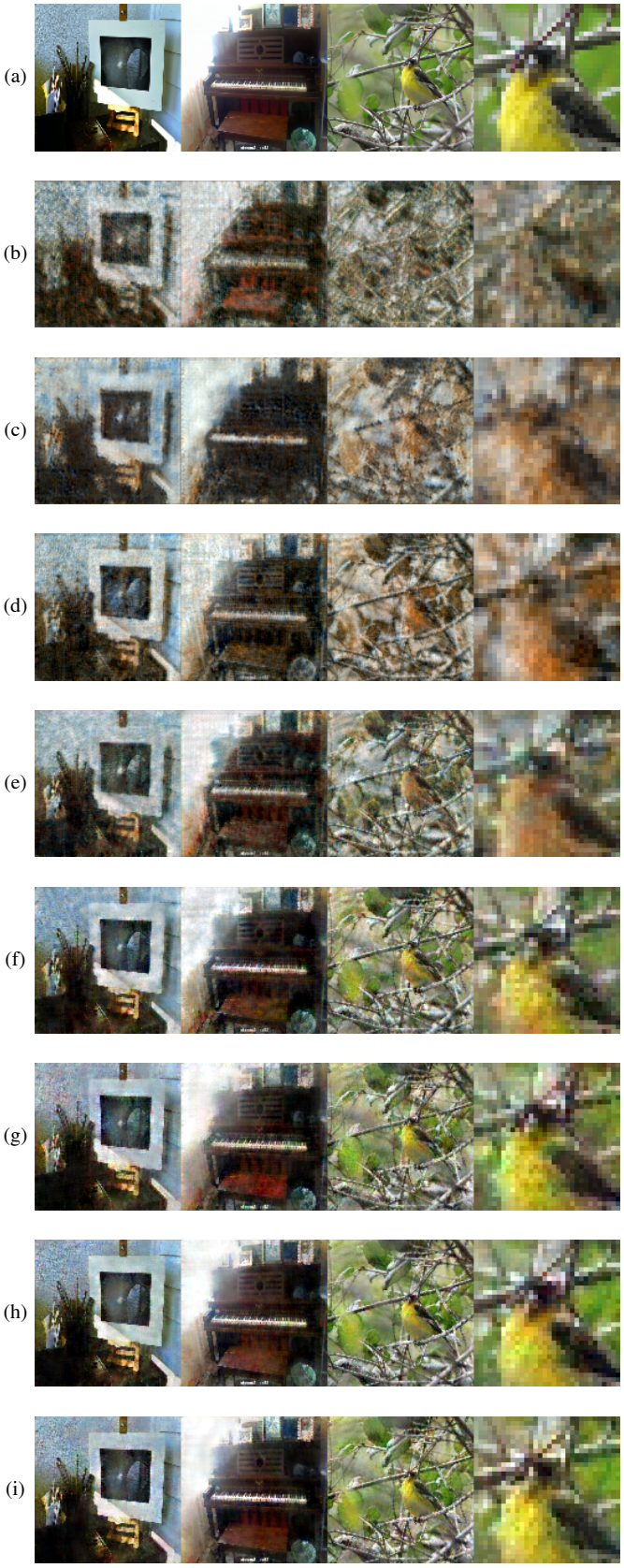

Fig. 3. Image reconstruction from the ImageNet test set. The rightmost image is a close-up of the bird from the third picture. (a) Original images, (b) [8, 8, 8], random encoder, (c) [6, 12, 24], random encoder, (d) [16, 16, 16], random encoder, (e) [16, 16, 16], encoder pretrained for classification, (f) [32, 32, 32], random encoder, (g) [24, 48, 96], random encoder, (h) [64, 64, 64], random encoder, and (i) [48, 96, 192], random encoder.

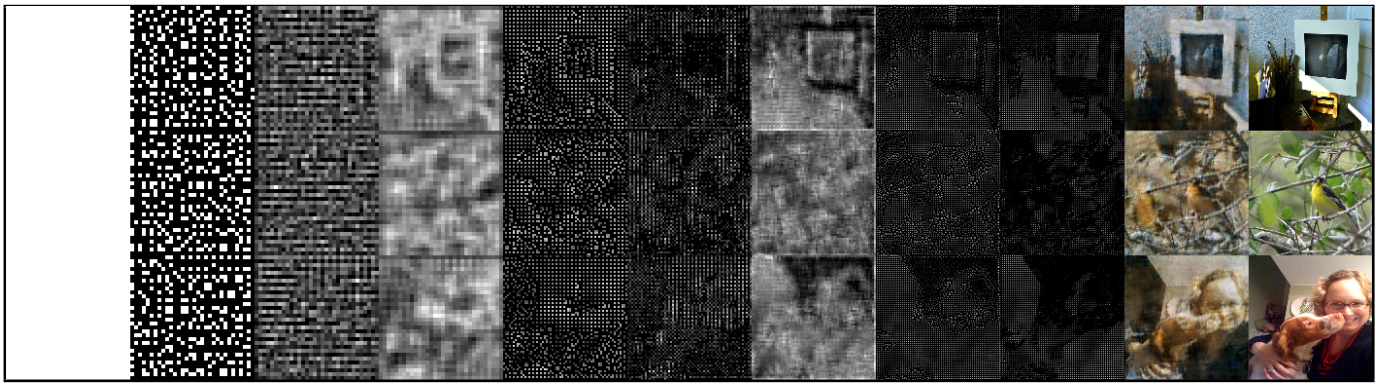

Fig. 4. Activation of a random channel of each layer during reconstruction with [12,24,48]. The first layer is pure white, since it is the bias layer, which has one value per channel. The right-most image is the original image.

amounts being transmitted by the max switches. In Fig. 3 and Table II, we see that a higher number of filters results in better reconstruction and a lower loss. This is to be expected, because the transmitted information available for reconstruction is higher.

By varying the number of filters for each individual unit, we experience a slight improvement in the reconstruction that is difficult to see in the reconstructed images in Fig. 3. In almost all cases, pretraining the network for classification actually increased the loss. We theorize this happens because features that are useful for classification are not necessarily good for reconstruction because the spatial and background information is less useful for classification.

We see that reconstructions from architectures with fewer filters have significantly degraded colors. Although the edges are still reconstructed well, all colors become a lot duller in the process. We do not fully understand why this happens.

The most surprising result is that reconstruction with max switch information works so well in general. As shown in Fig. 4, the reconstructed image goes from very rough to a good reconstruction and becomes more detailed as it is being processed through the layers.

We experimented with different architectures and ImageNet as the dataset to verify that this is not due to one specific feature of a network but is an inherent characteristic of max unpooling.

Our results provide a new perspective on some of the techniques and results shown in Section II. As explained, [5] used autoencoders to make supervised and unsupervised training in one architecture possible. For unsupervised data, the autoencoder part of the network trains for reconstruction of the image with a decoder network consisting of deconvolutional and max unpooling layers. However, training for L2 reconstruction loss will not necessarily yield good features for classification in the encoder network because we have shown that good reconstruction is already possible with max pool switches of randomly initialized weights.

We theorize that the improved error for street view house numbers (SVHN) dataset, for example, that [5] achieved is caused by training w.r.t the intermediate loss for the hidden states of the network. This forces the network to reconstruct at each layer and learn good features for single-layer reconstruction along with the simultaneous supervised training for classification. Further experimentation with the SWWAE architecture with L2 reconstruction loss for only the intermediate terms could clarify this.

Zhang et al. [7] stated that "the intermediate activations of pretrained large-scale classification networks preserve almost all the information of input images except a portion of local spatial details." However, there is already enough information in the max switches to reconstruct this image. Therefore, it is not proven that the activations were responsible for preservation of the input image. The fact that using fixed max switches resulted in significantly worse reconstructions supports this idea.

VII. Conclusion

We have shown that good image reconstruction is possible with the transmission of max pool switch locations without any other information about the actual high-level feature representation in deep layers.

This gives new insights into the abilities of CNNs with deconvolutional and unpooling layers: Although the encoder was not trained to output useful information for the reconstruction of the image, the decoder reconstructs the image from the switch locations. This is remarkable since no information about the pixel values was provided. It explains why reconstruction of the image with max unpooling for semantic segmentation works so well [3]: The spatial information can be extracted completely from the max switch locations.

This is a potential pitfall for future architecture decisions when using unpooling layers as we presented in Section II. A lot of information will flow through any unpooling layers. If this is undesirable, for example, because the image is manipulated in a higher-level feature space and then reconstructed according to the modifications [11], care should be taken when max unpooling is used for this application.

It is also a concern regarding pretraining for classification with unsupervised data, which has recently become popular again [7]. As we demonstrated, it is possible that the network will not learn useful features by training for reconstruction but instead will reconstruct the image solely from the spatial information without learning.

References

[1] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in European Conference on Computer Vision, 2014, pp. 818–833.

[2] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in ICLR (workshop track), 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2015/DB15a

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1520–1528.

[4] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," CVPR, pp. 193–203, 2016.

[5] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun, "Stacked what where auto-encoders," CoRR, vol. abs/1506.02351, 2015.

[6] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in 2011 International Conference on Computer Vision, 2011, pp. 2018–2025.

[7] Y. Zhang, K. Lee, and H. Lee, "Augmenting supervised neural networks with unsupervised objectives for large-scale image classification," ICML, pp. 612—621, 2016.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[10] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Journal of Machine Learning Research, vol. 12, no. Jul, pp. 2121–2159, 2011.

[11] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim, "Rotating your face using multi-task deep neural network," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 676–684.

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

VIII. Supplementary Materials

A. How to Reproduce the Results

The network was trained with Caffe [12]. We modified a Caffe fork to leverage the deconvolution and unpooling layers [3]. It is available on GitHub: github.com/matthieudelaro/caffe.

All files necessary to reproduce our results are available at github.com/laura-rieger/max_switch_reconstruction_tunnel.

Because ImageNet is not publicly available, you must have access to it to reproduce our exact results. To generate the exact dataset used for training, place the ImageNet data in the dataset folder and run the script GenerateLmdb.sh. All models and solvers are in the proto2 folder. The necessary hyperparameters have already been set. Fully trained models are in the snapshots2 folder.

We recommend using the Docker image matthieudelaro/caffe-cifar100, which includes our fork of Caffe.