

Deep Learning for Image Analysis - Visualization of Neural Networks

Thomas Walter, PhD

Centre for Computational Biology (CBIO)
MINES Paris-Tech, PSL Research University
Institut Curie, PSL Research University
INSERM U900

Overview

- 1 Introduction
- 2 Visualization of filters
- 3 Representation visualization
- 4 Visualization of the loss function
- 5 References

Motivation: visualization and understanding

- Convolutional Neural Networks have achieved stunning performance for many tasks, such as image classification, image segmentation and object detection.
- CNN can be difficult to train, and results can be very variable for different settings of hyperparameters.
- It is unclear, how design choices (network architecture, initialization, weight decay, ...) effect trainability and performance.
- It is not very well understood what Neural Networks actually learn.
- One idea to approach some of these issues relies on **visualization**.

Overview

1 Introduction

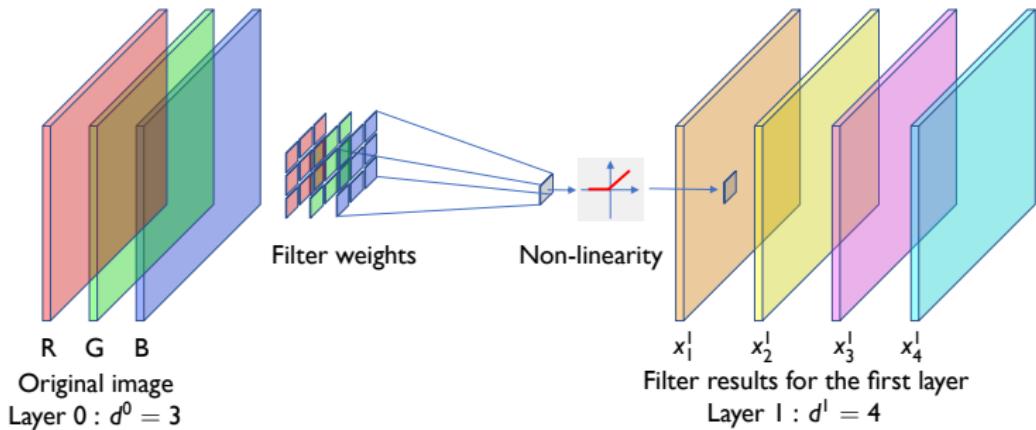
2 Visualization of filters

3 Representation visualization

4 Visualization of the loss function

5 References

Filters in Neural Networks

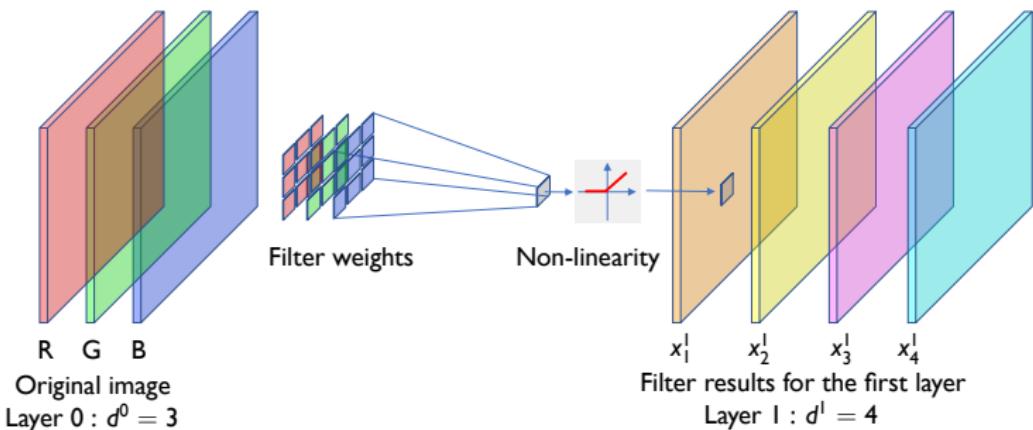


In Convolutional Neural Networks (CNN), a pixel value in activation map r in layer l depends on the pixel values in layer $l - 1$ within the receptive field of the filter W_r^l across all channels. The pixel value $x_r^l(n, m)$ at position n, m in the activation map r in layer l is¹:

$$x_r^l(n, m) = g\left(\sum_k \sum_{i,j} W_r^l(i, j, k) x_k^{l-1}(n + i, m + j) + b_r^l\right) \quad (1)$$

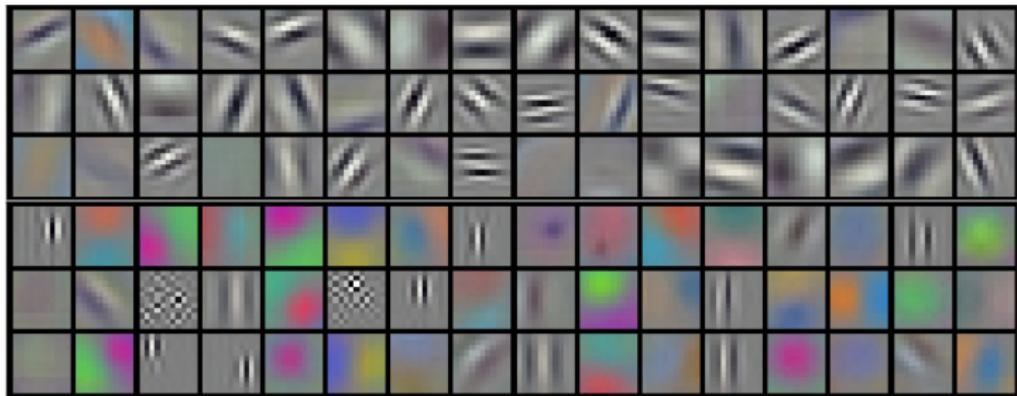
¹For simplicity, we do not consider downsampling operations and to not include bias terms.

Filters in Neural Networks



- Each filter W_r^l (filter r in layer l) is thus a 3D tensor of values with dimension $d^{l-1} \times s^l \times s^l$.
- For the first layer, the number of input channels is typically 3, i.e. for each pixel, there are 3 values available (R , G and B).
- For this reason, all filters of layer 1 have the dimension $3 \times s^1 \times s^1$.
- This means that each filter corresponds itself to an *RGB* image of size $s^1 \times s^1$.

Visualization of the first layer filters



- First layer filters obtained by the Alexnet in the Imagenet competition [Krizhevsky et al., 2012].
- Observation: they contain oriented contours (top row) and color blobs (bottom row).
- How can we interpret this?

Visualization of first layer filters

- The scalar product $w^T x$ is maximal if x points into the same direction as w :

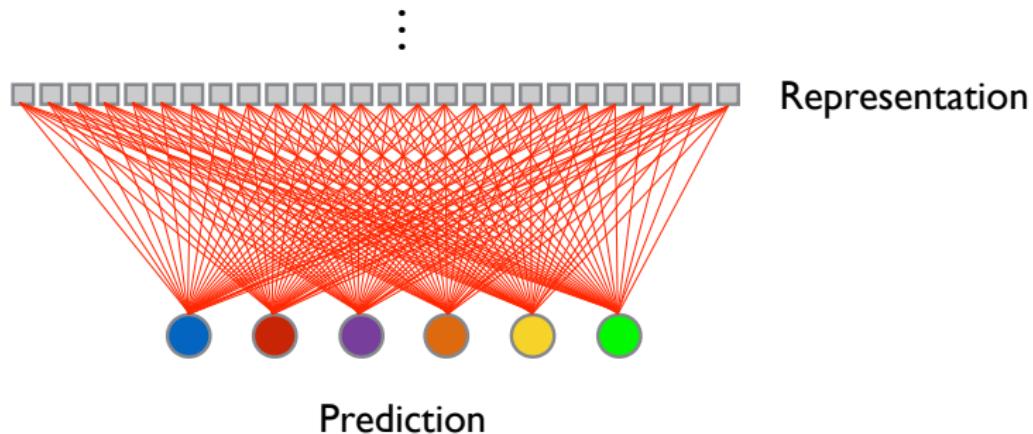
$$w^T x = \|w\| \|x\| \cos \alpha_{x,w}$$

- Consequently the activation maps have high values, if the patterns in the image coincide with the filters.
- We therefore see, that the first layer of a trained convolutional neural network extracts low-level visual information, such as corners, edges, colors.
- The filter visualization strategy is not really applicable to deeper layers: as the filters in deeper layers act on the outputs of the first layer activation maps, it is unclear how to interpret the patterns they might show.

Overview

- 1 Introduction
- 2 Visualization of filters
- 3 Representation visualization
- 4 Visualization of the loss function
- 5 References

Visualization of image representation



- CNNs learn representations of the images. Here, we call the representation the last layer prior to classification.
- We can now collect these representations $x^{l_{max}}$ for all images in the training set.
- In order to visualize these representations, we seek low dimensional representations of these high dimensional vectors.

t-SNE (sketch) 1/2

- Many low dimensional representations: PCA, ICA, ...
- A recent technique that is widely used in this field is called *t-Distributed Stochastic Neighbor Embedding (t-SNE)* [Maaten and Hinton, 2008].
- Let $\{x_i\}_{0 \leq i < N}$ be the set of representations for all images in the training set.
- We define the pairwise probabilities p_{ij} that x_i and x_j are neighbors (which depends on $\|x_i - x_j\|$).
- We now seek a low-dimensional space for which these neighborhood probabilities q_{ij} are conserved.
- This is achieved by minimizing for each point x_i the Kulback-Leibler divergence between the distributions P_i (distribution of p_{ij}) and Q_i (distribution of q_{ij})

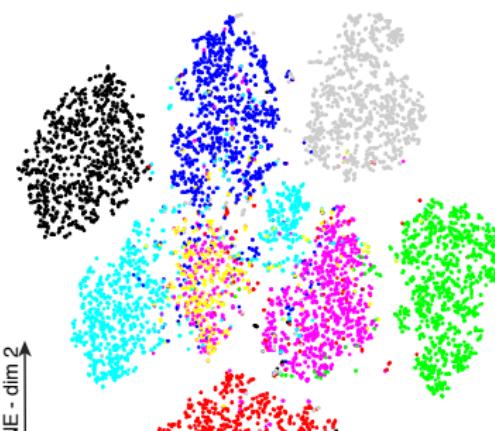
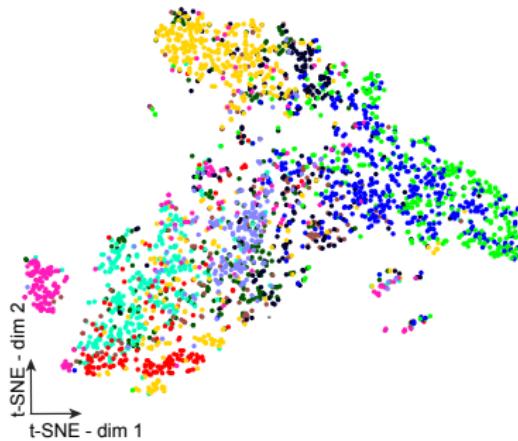
t-SNE (sketch) 2/2

- The cost C that is minimized:

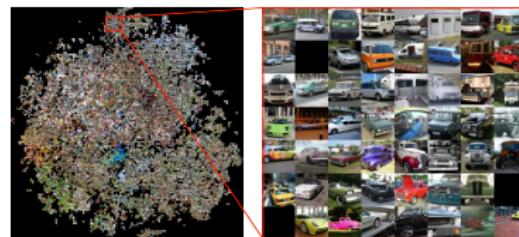
$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j -p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (2)$$

- If x_i and x_j are close in the original space, the algorithm will try to push q_{ij} to become close to x_{ij} .
- If x_i and x_j are far in the original space, only a small contribution is to be expected in any case.
- For this reason, t-SNE is known to respect local structure.

t-SNE map: example (cell images)

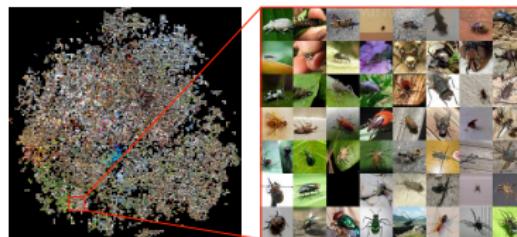


t-SNE map: ImageNet



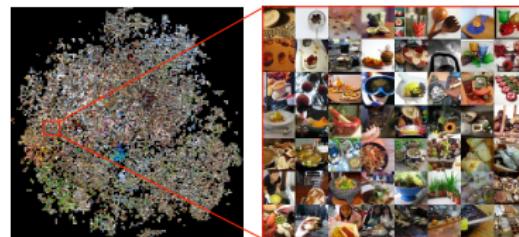
Credits: adapted from
<https://cs.stanford.edu/people/karpathy/cnnembed/>

t-SNE map: ImageNet



Credits: adapted from
<https://cs.stanford.edu/people/karpathy/cnnembed/>

t-SNE map: ImageNet



Credits: adapted from
<https://cs.stanford.edu/people/karpathy/cnnembed/>

Overview

- 1 Introduction
- 2 Visualization of filters
- 3 Representation visualization
- 4 Visualization of the loss function
- 5 References

Motivation: visualization of the loss function

- We have seen that training a Neural Networks corresponds to solving a minimization problem:

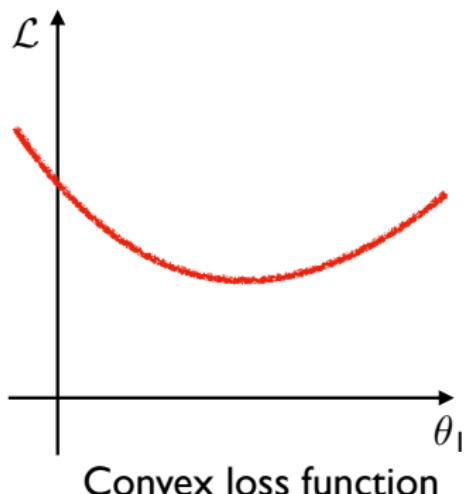
$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \mathcal{R}(\boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is the vector of all parameters, $\mathcal{R}(\boldsymbol{\theta})$ a regularization term and $L(\boldsymbol{\theta})$ the loss term calculated on the training data:

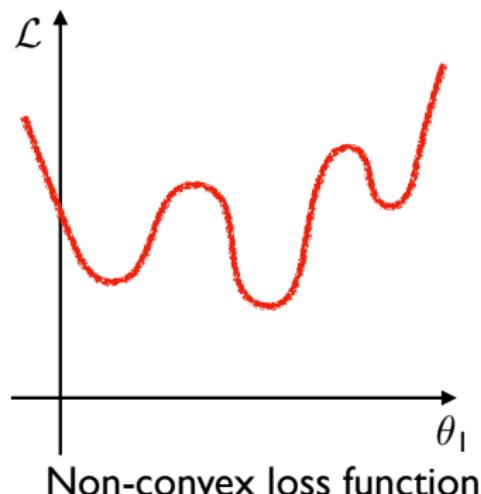
$$L(\boldsymbol{\theta}) = \sum_{i=1}^N L_i(\boldsymbol{\theta})$$

- Visualization of the loss can give us a hint on how complicated this task really is.
- Visualization can also allow us to compare different architectures or initialization schemes.

Convexity



Convex loss function



Non-convex loss function

- Convex optimization functions: only one local minimum (which is also the global minimum).
- In neural networks, the loss is often not convex.
- The shape of the loss is very important for the success of the optimization.
- How can we visualize the loss w.r.t $\sim 10^6$ parameters?

Visualization of the loss function

- Visualizations are limited to 2D or 3D plots.
- We can only plot the loss as a function of one or two parameters.
- Idea: to draw two random directions δ, η in parameter space and plot the loss function in these directions:

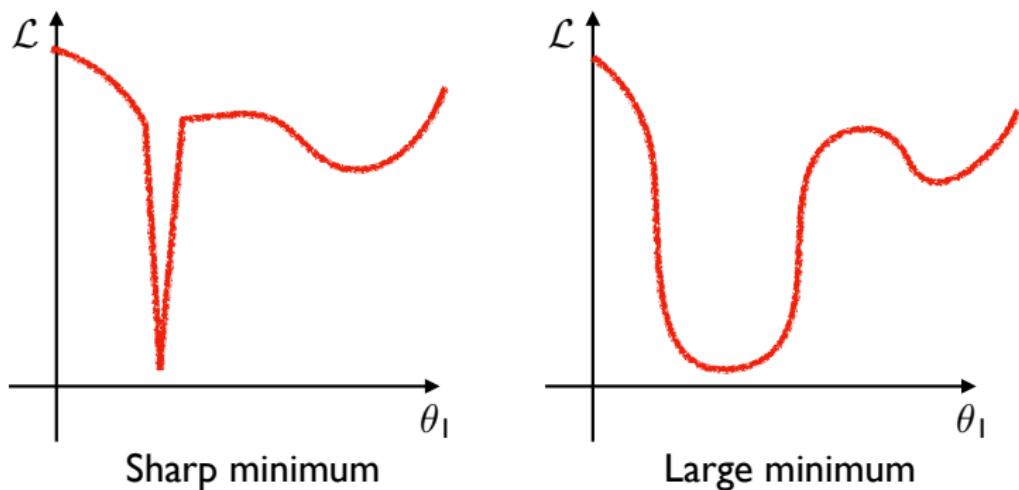
$$L(\alpha, \beta) = L(\theta^* + \alpha\delta + \beta\eta) \quad (3)$$

with L^* our solution which is supposed to be optimal.

- Alternative: if we wish to compare two networks θ^A and θ^B , we can also visualize the loss in the direction of the difference between these two points in parameter space [Goodfellow and Vinyals, 2014]:

$$\theta(\alpha) = (1 - \alpha)\theta^A + \alpha\theta^B \quad (4)$$

The problem of scale invariance



- Sharpness of a minimum: the intuition is that sharp minima are hard to find and less robust.
- Scale invariance: two neural networks with ReLu as non-linearities are equivalent, if we multiply the weights of one layer with a factor and divide the weights of the next layer by the same factor.
- 1D and 2D plots of the loss can therefore be misleading: the

Scale invariance and filter-wise normalization

- One idea to cope with scale invariance is to normalize the directions with respect to the filter weights [Li et al., 2017].
- First we draw a random direction d .
- We then require that the parameters belonging to the same filter r in layer l are normalized such that their norm is that of the filter:

$$d_r^l \leftarrow \frac{d_r^l}{\|d_r^l\|} \theta_r^l \quad (5)$$

where $\|\cdot\|$ is the Frobenius norm.

Example: effect of deeper networks

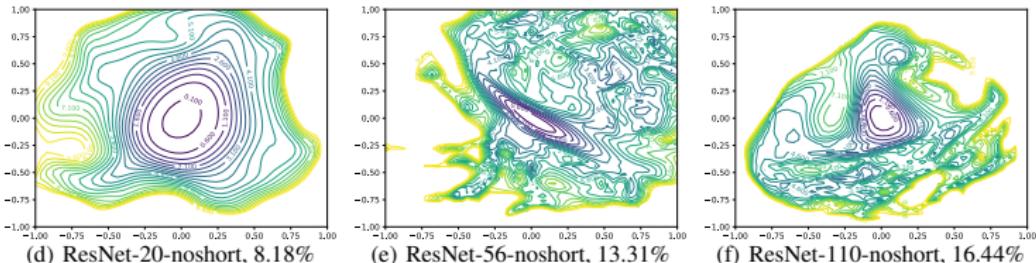


Figure: Resnet architecture without skip connections (varying depth)

- With 20 layers, we observe a fairly convex loss function.
- With more layers, the loss function becomes chaotic and the gradient does not point to the global minimum.
- In addition, the minima are steep and sometimes ill-conditioned (anisotropy).

Example: importance of skip-layers

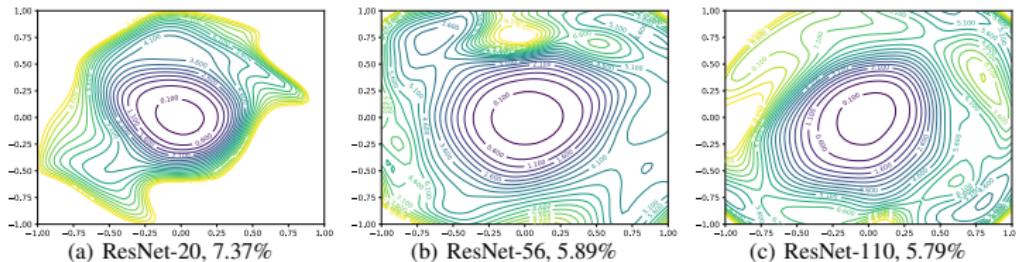
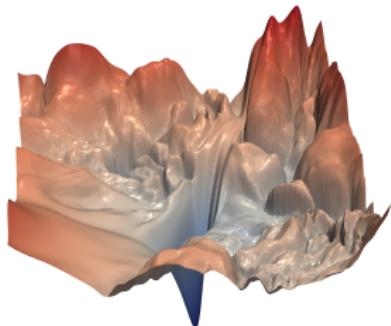


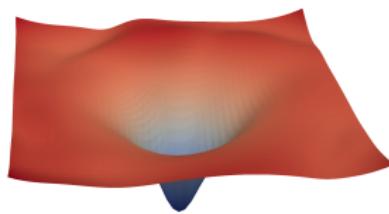
Figure: Resnet architecture with skip connections (varying depth)

- Adding skip connections makes the objective function "more convex".
- This is particularly true for deeper networks (here: 56 and 110 layers).
- We therefore understand the impact of this architectural choice.

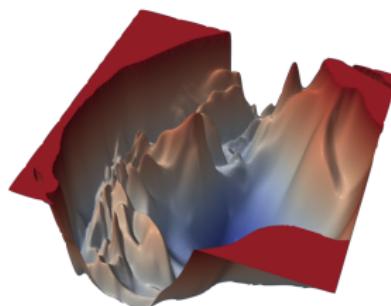
More examples for the importance of skip-layers



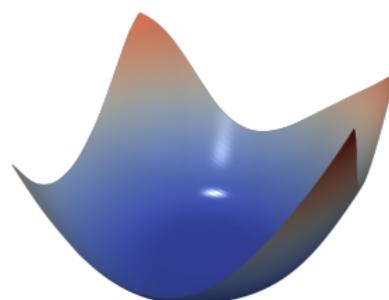
(a) without skip connections



(b) with skip connections



(a) 110 layers, no skip connections



(b) DenseNet, 121 layers

Credits: From [Li et al., 2017]

References I

- [Goodfellow and Vinyals, 2014] Goodfellow, I. J. and Vinyals, O. (2014). Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.
- [Li et al., 2017] Li, H., Xu, Z., Taylor, G., and Goldstein, T. (2017). Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913.
- [Maaten and Hinton, 2008] Maaten, L. V. D. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.