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

Motivation: visualization and understanding

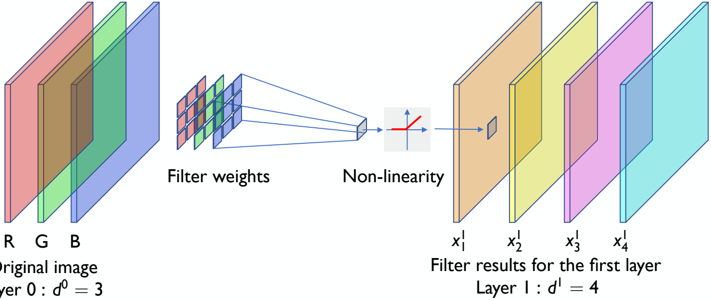
- Convolutional Neural Networks have achieved stunning performance for many tasks, such as image classification, image segmentation and object detection.
- CNN can be difficult to train, and results can be very variable for different settings of hyperparameters.
- It is unclear, how design choices (network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

Motivation: 3 scenarios

1. AI worse than humans: visualization can thus help understanding failure modes to guide the research.
2. AI on par with humans: establish appropriate trust and identify and remove potential biases.
3. AI better than humans: visualization can help in machine teaching.

Overview

Filters in Neural Networks

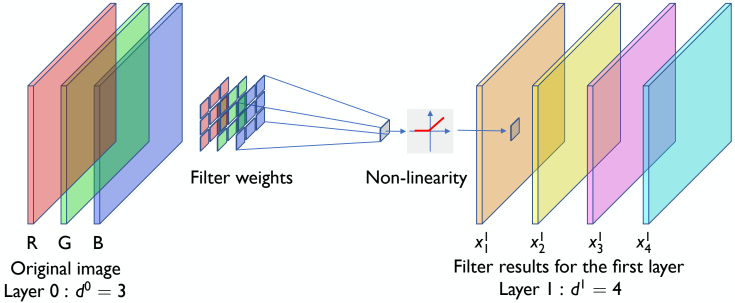


In Convolutional Neural Networks (CNN), a pixel value in activation map r in layer l depends on the pixel values in layer $l-1$ within the receptive field of the filter W_r^l across all channels. The pixel value $x_r^l(n,m)$ at position n, m in the activation map r in layer l is[1]:

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

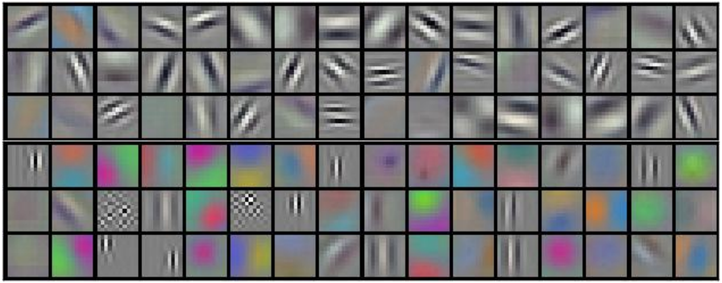
[1] For simplicity, we do not consider downsampling.

Filters in Neural Networks



- Each filter W_r^l (filter r in layer l) is thus a 3D tensor of values with dimension $d^{l-1} \times \nu^l \times \nu^l$ (ν: filter width).
- 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 $3 \times \nu^1 \times \nu^1$.
- This means that each filter corresponds itself to an RGB image of size $\nu^1 \times \nu^1$.

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

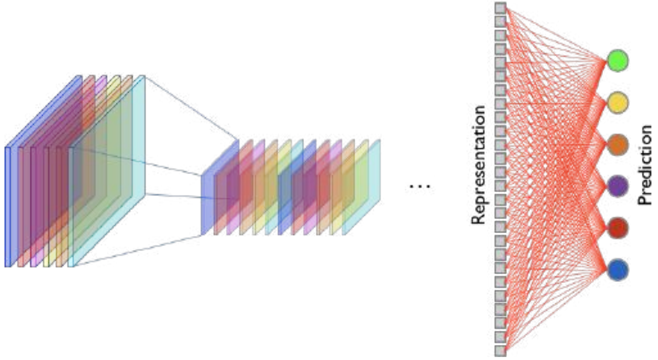
- The scalar product $w^T x$ is maximal if x points into the same direction as w:

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

- 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 activation maps of the previous filters, it is unclear how to interpret the patterns they might show.

Overview

Visualization of image representation



- CNNs learn representations of the images (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) 1/3

- 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 probability that x_j is a neighbor of x_i under the assumption that the neighbor probability is a Gaussian centered in x_i:

$$p_{j|i} = \frac{\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2}\right)} \tag{2}$$

Here, the parameter σ_i varies with the sample x_i.

t-SNE (sketch) 2/3

- We now want to map the data $\mathcal{X} = \{x_i\}$ to low dimensional representations $\mathcal{Y} = \{y_i\}$ for which these neighborhood probabilities in the projected space $q_{j|i}$ are conserved:

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)} \tag{3}$$

- This is achieved by minimizing the Kulback-Leibler divergence between the distributions P_i (distribution of $p_{j|i}$) and Q_i (distribution of $q_{j|i}$):

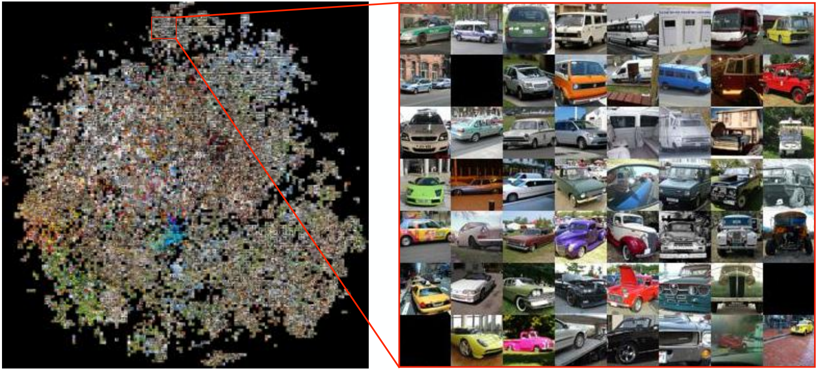
$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{4}$$

- If x_i and x_j are close in the original space, the algorithm will try to push $q_{j|i}$ to become close to $p_{j|i}$.
- If x_i and x_j are far in the original space, they may or may not be far in the projected space.

t-SNE (sketch) 3/3

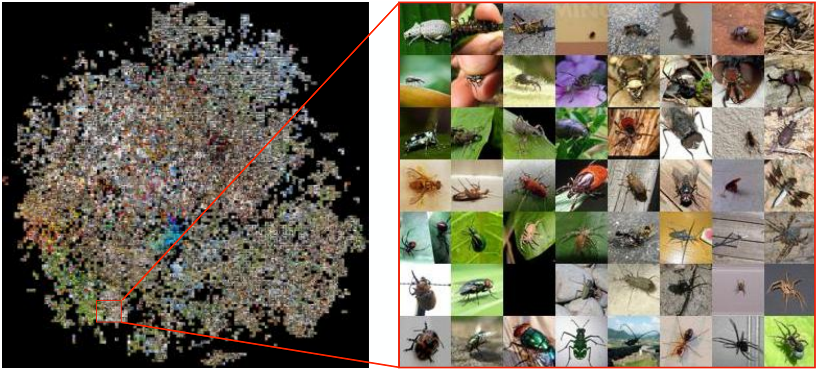
- The parameters σ_i are determined by the algorithm.
- They vary according to the neighbor density, such that for each x_i we have roughly the same number of neighbors.
- The user chooses a value for the *perplexity*, which effectively measures the effective number of neighbors.
- The problem now resumes to solving $\min_{y_i} C$ by gradient descent.
- Importantly, this method does not provide a transformation: the minimization is carried out w.r.t. y_i. If there is a new datapoint, the minimization problem has to be solved again.

t-SNE map: ImageNet



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

t-SNE map: ImageNet



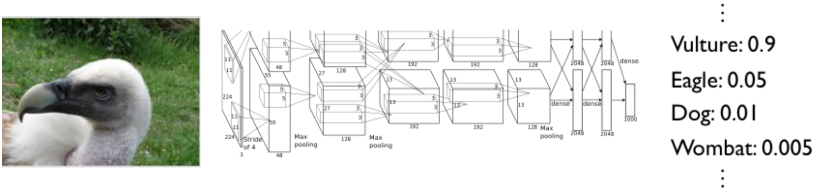
Credits: adapted from https://cs.stanford.edu/people/karpathy/cnnembed/

t-SNE map: ImageNet

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

Overview

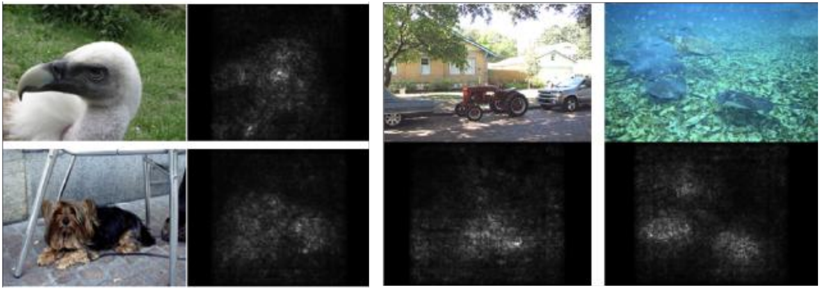
Saliency maps



$$\frac{\partial L}{\partial x_{i,j}}$$

- For each pixel in the input image we can calculate the derivative of the loss with respect to that pixel value [Simonyan et al., 2013].
- This allows us to visualize how much the loss changes with small variations of the pixel values.
- Calculation: simple backgropagation

Saliency maps



Sensitivity analysis: we can see which pixels would have most effect on the classification results.

Credits: adapted from [Simonyan et al., 2013]

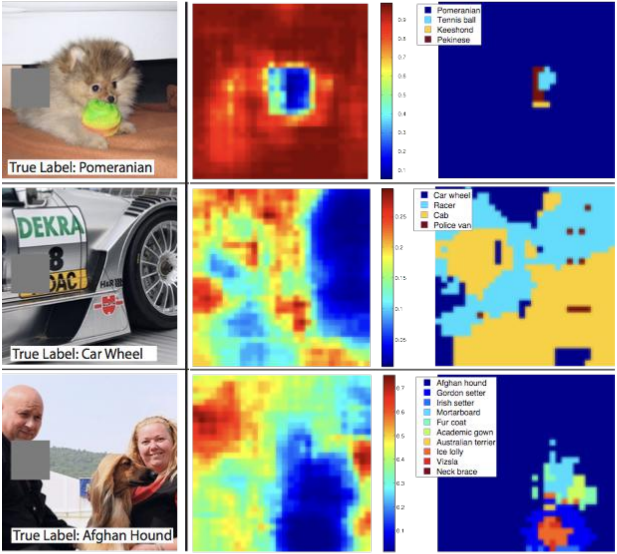
Saliency maps - limitations

- Individual pixels may only marginally contribute to the output.
-

Occlusion of image patches

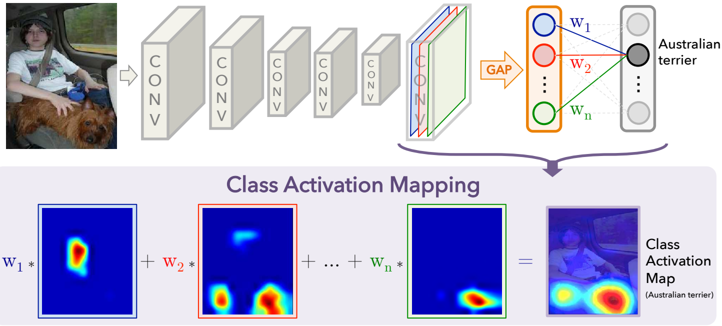
- The output of a classification network is a vector of posterior probabilities $P(y|x)$, where x is the image and y the class label.
- Experiment: we occlude a patch in the image and obtain a changed probability [Zeiler and Fergus, 2013].
- Now we can assign to every position of the patch in the original image the probability of the correct class and the class label with maximal posterior probability.
- While we can therefore measure the importance of parts of the image with respect to the classification result, this method is slow and not very elegant.

Occlusion of image patches



Credits: adapted from [Zeiler and Fergus, 2013]

Class Activation Mapping (CAM)



- In [Zhou et al., 2016], the fully connected layers are replaced by Global Average Pooling (GAP).
- The final prediction is then made from the n-dimensional vector. The training provides us with a n-dimensional weight vector w^c for each class.

Credits: adapted from [Zhou et al., 2016]

Class Activation Mapping (CAM)

- More formally, if the last convolutional layer has n feature maps, GAP maps this layer to a n-dimensional vector:

$$GAP(k) = \frac{1}{N} \sum_{i,j} f_k(i,j) \qquad k = 1, \ldots, n \tag{5}$$

  where N is the number of neurons in each feature map and n the number of feature maps.
- The score S_c for each class c is then the weighted sum of the vector entries:

$$S_c = \sum_k w_k^c GAP(k) \tag{6}$$

- The posterior probabilities are obtained by softmax:

$$P(c \mid x) = \frac{\exp(S_c)}{\sum_c \exp(S_c)} \tag{7}$$

- This modified network is then fine-tuned. Often, this simplification is not affecting classification accuracy very seriously [Zhou et al., 2016].

Class Activation Mapping (CAM)

- The class scores can be written as:

$$S_c = \sum_k w_k^c \frac{1}{N} \sum_{i,j} f_k(i,j) = \frac{1}{N} \sum_{i,j} \sum_k w_k^c f_k(i,j) = \frac{1}{N} \sum_{i,j} M_c(i,j) \tag{8}$$

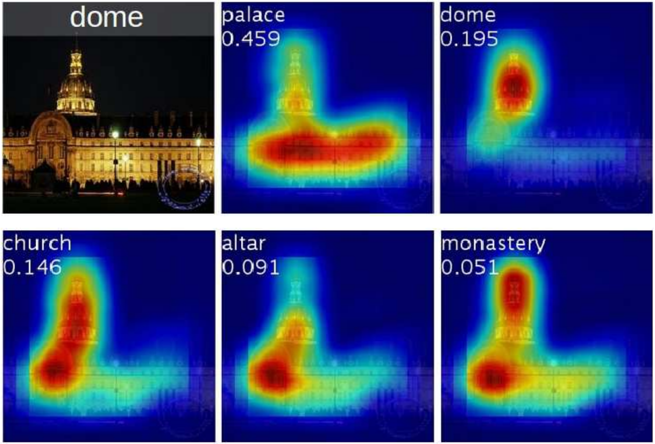
  M_c is a class activation map, which is the weighted average of the last convolutional layer, with the weights optimized for prediction.
- In practice, we might decide not to take the last convolutional layer, but one that is more upstream in order to get better spatial resolution.
- In order to overlay to the original image, we simply upscale (interpolation).

Class Activation Mapping (CAM) - Results

We see that the CAM highlights the regions which are discriminative for the predicted class ("barbell").

Credits: adapted from [Zhou et al., 2016]

Class Activation Mapping (CAM) - Results



We observe the variation of the CAM according to the predicted label. Here the correct label would have been "Dome".

Credits: Image taken from [Zhou et al., 2016]

Generalization of CAM: Grad-CAM

- A drawback of CAM is that we are not really analyzing a given network, but we design a similar network that we can then analyze.
- This can be avoided by grad-CAM [Selvaraju et al., 2020], that is based on the evaluation of gradients:

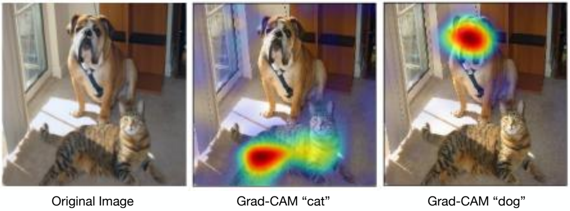
$$\alpha_k^c = \frac{1}{N} \sum_{i,j} \frac{\partial S_c}{\partial f_k(i,j)} \tag{9}$$

- If the fully connected layer is replaced by a global average pooling, this is identical to the w_k^c we have calculated in CAM (up to a constant factor).
- The final map is then calculated by:

$$M^{gradCAM}(c) = ReLu(\sum_k \alpha_k^c f_k) \tag{10}$$

The ReLu is used to only account for positive contributions.

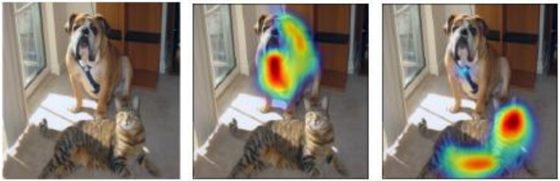
Grad-CAM results



Grad-CAM result for an image with double label ("tiger-cat" and "boxer").

Credits: adapted from [Selvaraju et al., 2020]

Grad-CAM results: counter-factual image regions



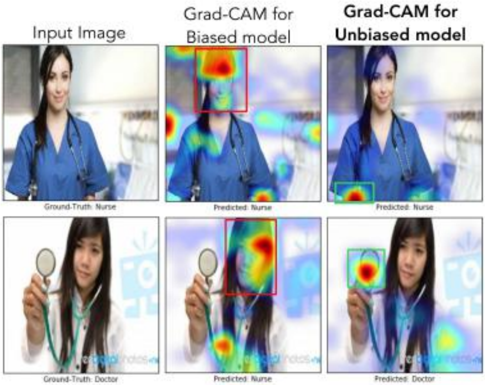
(a) Original Image (b) Cat Counterfactual exp (c) Dog Counterfactual exp

Here, the authors investigated which regions would rather make the network change its prediction. This is simply done by inverting the sign:

$$\alpha_k^c = \frac{1}{N} \sum_{i,j} -\frac{\partial S_c}{\partial f_k(i,j)} \tag{11}$$

Credits: adapted from [Selvaraju et al., 2020]

Grad-CAM results: spotting bias



In [Selvaraju et al., 2020], a neural network was trained to distinguish between two classes "Nurse" and "Doctor". The training set contained a gender bias. We see that the important regions differ when the bias is removed.

Credits: adapted from [Selvaraju et al., 2020]

Maximally activating images

- Another strategy is to generate an image that maximizes a particular neuron, typically a class score:

$$X^* = \arg\max_X S_c(X) - \lambda \|X\|^2 \tag{12}$$

- This can be achieved by a similar technique as training the network, but this time the parameters of the network are fixed and the image is adapted as to maximize the objective function.
- Here, we do not optimize the softmax, but the class score, as the softmax contains contributions from all classes.

Maximally activating images - Results

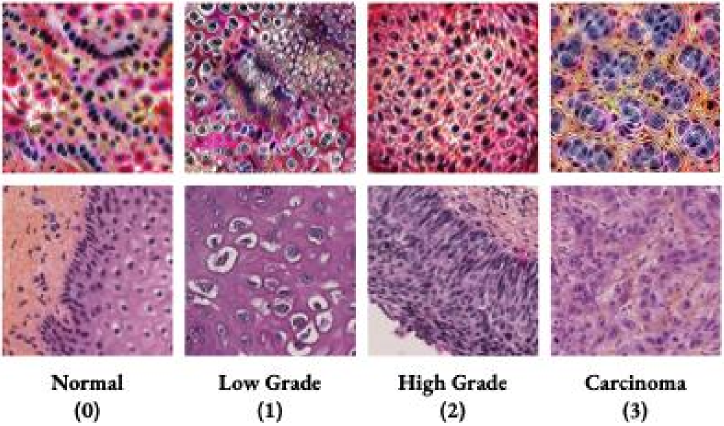
We observe various elements the network is trained to react on.

Credits: adapted from [Simonyan et al., 2014]

Application in digital pathology 1/2

- Task: grading cervix cancer according to severity
- Input data: stained tissue sections from biopsies or surgical specimen.
- Multi-class-classification problem: (1) normal, (2) low grade displasia, (3) high grade displasia, (4) carcinoma
- Challenging question: which morphological patterns trigger the decision?

Application in digital pathology 2/2



- Maximally activating images are realistically looking images
- The patterns correspond to known morphological hallmarks for grading.

Credits: adapted from [Lubrano di Scandalea et al., 2022]

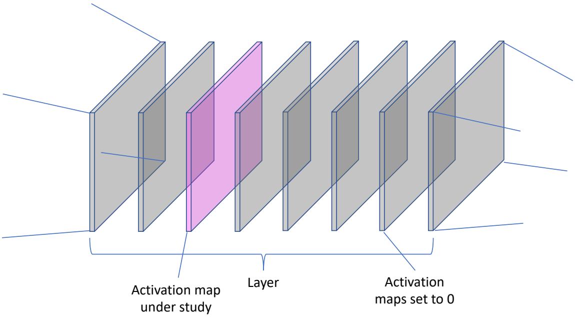
Overview

Looking at activation maps

- Simple visualization of activation maps inside the network.
- Example: channel 151 of conv5 layer.
- Sometimes, the information is very local, i.e. there are strong activations in one map of one layer (specialized map for text, faces, etc.)
- Interestingly, there was no specific face class in the data set.
- There are tools to perform these visualizations efficiently [Yosinski et al., 2015].

Credits: Image taken from [Yosinski et al., 2015]

Mapping activation maps to the original pixel space

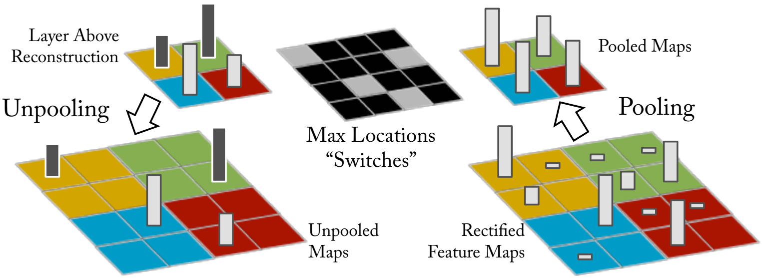


- We would also like to understand which patterns in the original image are important for a particular activation map.
- In this case, we need to map activities back to the input pattern that caused the activity [Zeiler and Fergus, 2013].
  - We set all other maps in the layer to 0.
  - We "revert" the neural network from the feature map by using a deconvnet [Zeiler et al., 2011].

How to map activities back to input pixel space

- There are three operations that need to be "reversed": convolution, ReLu, max-pooling.
- **Deconvolution:** [Zeiler and Fergus, 2013] propose to simply convolve the activation map at the upper level with the transposed learned filter.
- **ReLu:** The Relu cannot be reversed.
  [Zeiler and Fergus, 2013] propose to simply put a ReLu in the deconvolution path as to only keep positive contributions.
- **Unpooling:** Pooling operations cannot be reverted. The deconvnet remembers where the maximum pixels originated from in the corresponding convolutional network.

How to map activities back to input pixel space

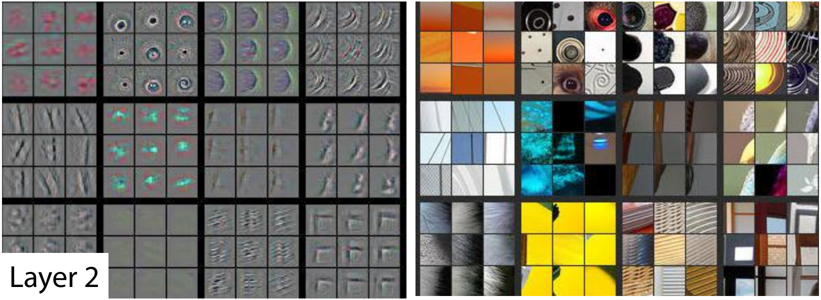


Credits: Image taken from
[Zeiler and Fergus, 2013]

Experiments

- ImageNet Validation Set.
- For a layer, a number of feature maps is (randomly) selected.
- For these feature maps, the 9 maximal activations are then reconstructed at the pixel level.
- Both the reconstructed image and the original crops are visualized.

Visualization of image features for layer 2 activation maps



The top 9 activation maps with the strongest signal across the validation set for layer 2.
Layer 2 features are slightly more complex patterns (corners, bended contours, color combinations) than in layer 1, but still relatively basic.

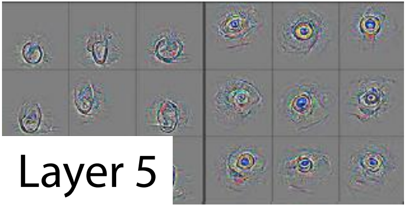
Visualization of image features for layer 3 activation maps

The top 9 activation maps with the strongest signal across the validation set for layer 3.
Higher Level features: faces, text, geometrical features.

Visualization of image features for layer 5 activation maps

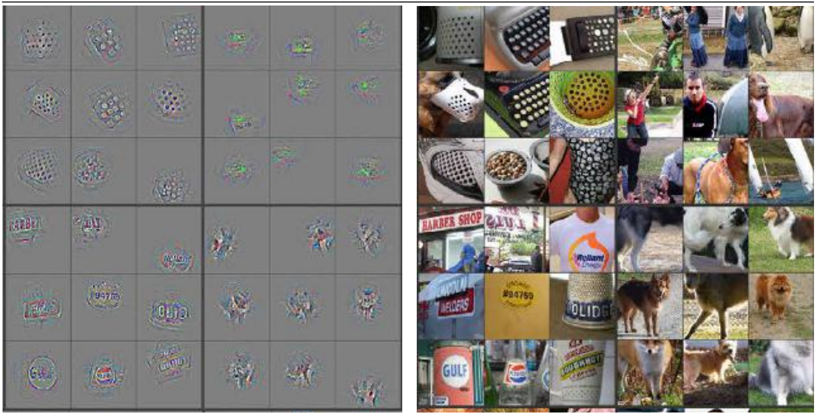
The top 9 projected activation maps for layer 5 across the validation set.
e.g. faces: contours, hair, nose ...

Visualization of image features for layer 5 activation maps



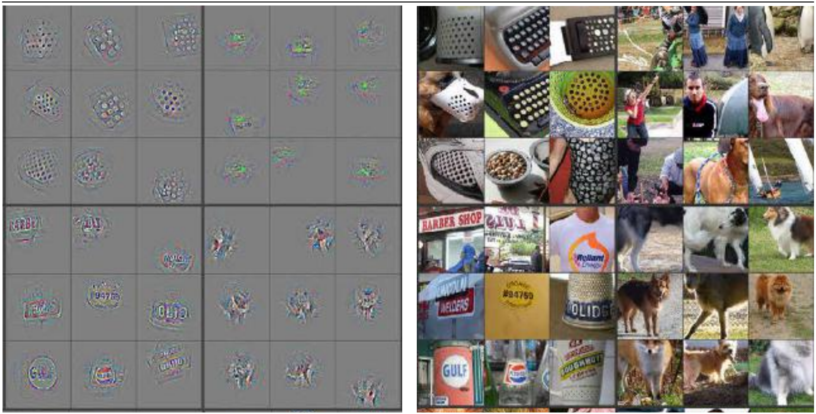
Here, we observe an eye detector and a wheel detector.
The image analysis community has developed many algorithms for eye detection; here the detection is implicit.

Visualization of image features for layer 5 activation maps



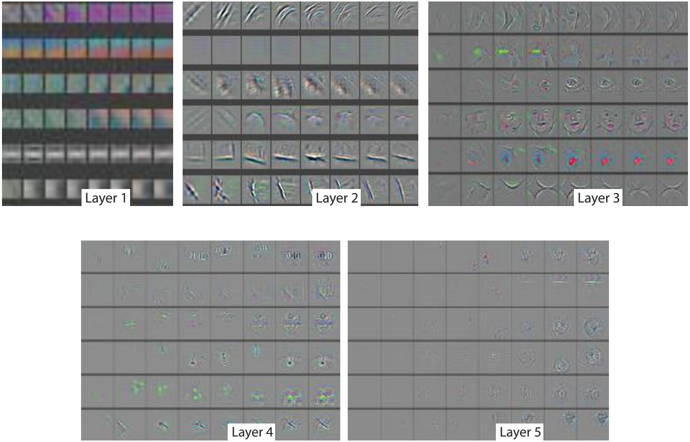
- Sometimes, the results can be surprising ... (top right)

Visualization of image features for layer 5 activation maps



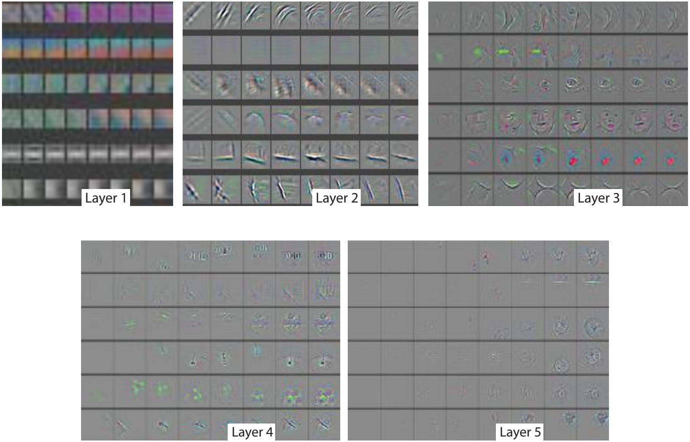
- Sometimes, the results can be surprising ... (top right)
- We have a background grass detector.

Image features during training across different layers



- Image features for activation maps depending on epochs. What do you observe?

Image features during training across different layers



- Image features for activation maps depending on epochs. What do you observe?
- The lower level features are learned first (a few epochs). Higher level features take more time to converge.

Overview

Motivation: visualization of the loss function

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

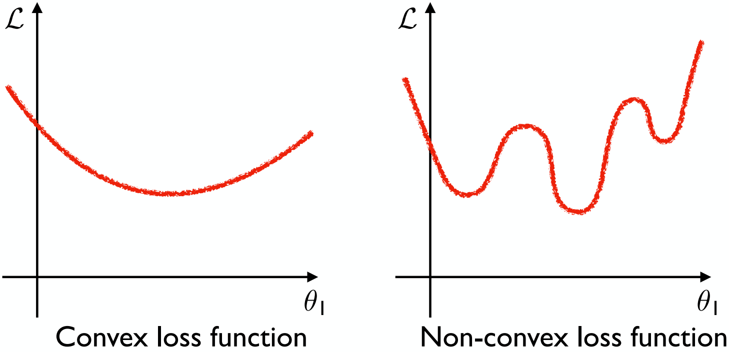
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda \mathcal{R}(\boldsymbol{\theta})$$

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 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

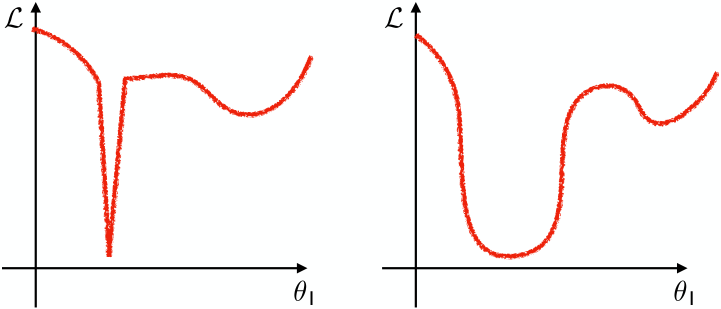
- Visualizations are limited to 2D or 3D plots.
- We can only plot the loss as a function of one or two parameters.
- Idea: to draw two **random directions** δ, η in parameter space and plot the loss function in these directions:

$$L(\alpha, \beta) = L(\boldsymbol{\theta}^* + \alpha\delta + \beta\eta) \tag{13}$$

  with $\boldsymbol{\theta}^*$ our solution which is supposed to be optimal.
- Alternative: if we wish to compare two networks $\boldsymbol{\theta}^A$ and $\boldsymbol{\theta}^B$, we can also visualize the loss in the direction of the difference between these two points in parameter space [Goodfellow and Vinyals, 2014]:

$$\boldsymbol{\theta}(\alpha) = (1 - \alpha)\boldsymbol{\theta}^A + \alpha\boldsymbol{\theta}^B \tag{14}$$

The sharpness of a minimum



Sharp minimum   Large minimum

- Sharpness of a minimum: the intuition is that sharp minima are hard to find and less robust.
- Can we use visualization in order to assess the sharpness of a minimum?

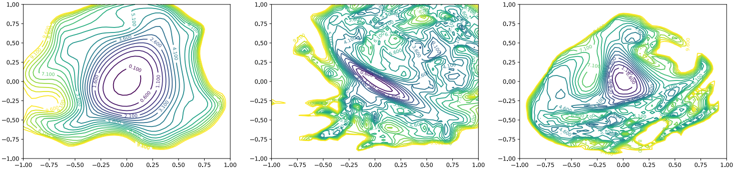
Scale invariance and filter-wise normalization

- 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 sharpness of a minimum might be due to a large extent to the problem of scale invariance.
- 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 (same dimension as $\boldsymbol{\theta}$).
  - For visualization, we require that the values in d that correspond to filter j in layer i have the same norm as the parameter values (i, j):

$$d_{i,j} \leftarrow \frac{d_{i,j}}{\|d_{i,j}\|}\|\boldsymbol{\theta}_{i,j}\| \tag{15}$$

    where $\|\cdot\|$ is the Frobenius norm. This filter-wise normalization ensures that we can interpret the produced maps.

Example: effect of deeper networks



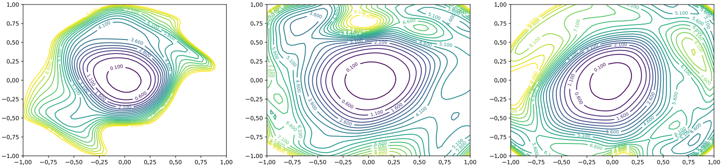
(d) ResNet-20-noshort, 8.18% (e) ResNet-56-noshort, 13.31% (f) ResNet-110-noshort, 16.44%

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).

Credits: From [Li et al., 2017]

Example: importance of skip-layers



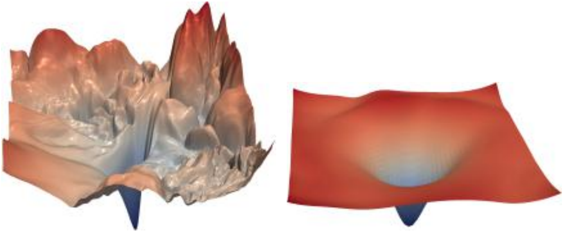
(a) ResNet-20, 7.37% (b) ResNet-56, 5.89% (c) ResNet-110, 5.79%

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.

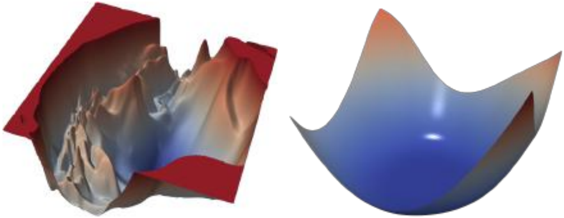
Credits: From [Li et al., 2017]

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]

Overview

What lessons do we learn?

- Features are extracted hierarchically.
- The first layer typically extracts low-level color and contour features.
- Later layers extract more specialized features (combinations of low-level features). This is the reason why you normally extract more feature maps in higher layers.
- We have methods to investigate the image regions that are responsible for classification assignments, and we can therefore understand, on which grounds a decision is made.
- Visualization of the loss function allows to compare architectures and to study architectural choices.
- As an example, skip layers "convexify" the loss function.

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.

References II

[Lubrano di Scandalea et al., 2022] Lubrano di Scandalea, M., Lazard, T., Balezo, G., Bellahsen-Harrar, Y., Badoual, C., Berlemont, S., and Walter, T. (2022). Automatic grading of cervical biopsies by combining full and self-supervision. Preprint, Cancer Biology.

[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.

[Selvaraju et al., 2020] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Journal of Computer Vision*, 128(2):336–359.

References III

[Simonyan et al., 2013] Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

[Simonyan et al., 2014] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv:1312.6034 [cs]*.

[Yosinski et al., 2015] Yosinski, J., Clune, J., Nguyen, A. M., Fuchs, T. J., and Lipson, H. (2015). Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579.

[Zeiler and Fergus, 2013] Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

References IV

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

[Zhou et al., 2016] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning Deep Features for Discriminative Localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, Las Vegas, NV, USA. IEEE.