Interpreting what ConvNets learn

Deep learning - BMDC 2025-2026

Gheorghe Cosmin Silaghi

Universitatea Babeș-Bolyai

October 15, 2025

Overview

Model interpretability

- why did your classifier think that a particular image contains a fridge when all you can see is a truck?
- interpretability is crucial in cases when the DL model complement the human expertise, like in the case of medical imaging
- DL models are black boxes: this is not true for the CNNs
- representations learned by ConvNets are highly amenable to visualization, because they are representations of visual concepts

Visualization of ConvNets representations

- Visualizing intermediate ConvNets activations
- Visualizing ConvNets filters
- Visualizing heatmaps of class activation in an image

Visualizing intermediate ConvNets activations

- the output of a layer is often called *activation*
- *visualizing intermediate ConvNets activations*: to display the values returned by various convolution and pooling layers in a model, given a certain input
- this gives you a view over how an input is decomposed into the different filters learned by the network
- as each channel encodes relatively independent features, we would like to visualize the content of every channel (which is a 2D image)

Things to observe

- the first layer acts as a collection of various edge detectors. At this stage, the activations retain almost all the information present in the initial picture
- as we go higher, activations become increasingly abstract and less visually interpretable. They begin to encode higher-level concepts like cat ear or cat eye.
- higher representations carry increasingly less information about the visual contents of the image and increasingly more information related to class of the image
- sparsity of the activations increases with the depth of the layer: in the first layer all filters are activated by the input image. the following layers more and more filters are blank. This means that the pattern encoded in the filter was not found in this particular image
- the deep network acts as a information distillation pipeline, with the raw data being repeatedly transformed so that the irrelevant information is filtered out and useful information is magnified and refined

Visualizing ConvNets filters

- display the visual pattern each filter is meant to respond.
- this can be done with the **gradient ascend** in the input space: to maximize the response of a specific filter, starting from the blank input image. The resulting image will be the one that the filter is maximally responsive to.
- how to do this: constructs a loss function that maximizes the value of a given filter in a given convolution layer and use stochastic gradient descend to adjust the values of the input image so as to maximize the activation value.

Side takeaway: the difference between model(x) and model.predict(x)

- predict() loops over the data in batches and extract NumPy value for the outputs. Predict can scale to large arrays
- model() happens in memory and do not scale.
- predict() is not differentiable, backend frameworks can not backpropagate through it
- if we want to compute gradients, we should use model()

Filters visualization - takeaways

- each layer in the ConvNet learns a collection of filters such that their inputs can be expressed as a combination of the filters.
- ConvNet filter banks get increasingly complex and refined as you go higher in the model
- filters from the first layers in the model encode simple directional edges and colors (colored edges in some cases)
- filters from layers a bit further up stack (such as block3_sepconv1) encode simple textures made from combinations of edges and colors
- filters in higher layers begin to resemble textures found in natural images: feathers, eyes, leaves and so on.

Visualizing heatmaps of class activation

- useful for understanding which part of a given image led the ConvNet to its final classification decision
- its good for debugging the decision process, in case of a classification mistake
- also allows to locate specific objects in an image
- class activation map (CAM): producing heatmaps of class activation over the input images. It is a 2D grid of scores associated with a specific output class, computed for every location in the input image, indicating how important is that pixel wrt the class under consideration

How to construct a Grad-CAM

- given an input image, take the output feature map of a convolution layer and weight every channel in that feature map by the gradient of the class with respect to the channel
- intuitively, you are weighting a spatial map of *how intense the input image activates different channels* by *how important each channel is with respect to a given class*.
- it results a spatial map of *how intensely the input image activates the class*.

Questions answered by the heatmap:

- why did the network think this image contains an African elephant?
- where is the African elephant located in the picture?