Learning Important Features Through Propagating Activation Differences

Paper | Videos | Notes | Implementation

DeepLIFT

DeepLIFT tries to explain the difference in output from a predefined **reference output** in terms of the difference of the input from a predefined **reference input**. Let's say we have a target neuron *t* with diff-from-ref *dt* and a layer *X* with neurons x1, x2 ... xn. The *dt* has then the following form.

$$\Delta t = t - t^0$$

We want to explain the *dt* in the context of *dx1*, *dx2* and so on. To do so, we are assigning contributions *Cdxi-dt* such that the sum of all *Cdxi-dt* is equal to *dt*.

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t$$

The *reference* values are the values which are being generated by a predefined *reference input*, the choice of which requires a specific domain knowledge. For MNIST, a suitable reference input would be a completely black image.

A **multiplier** *Mdxdt* is defined as the contribution *Cdx-dt* over *dx*. Multipliers satisfy the chain rule and thus, support backpropagation.

$$m_{\Delta x^+ \Delta y^+} = m_{\Delta x^- \Delta y^-} = m_{\Delta x \Delta y} = \frac{\Delta y}{\Delta x}$$

In order to allow for a different handling of positive and negative contributions, the *dy* is being seperated in its positive and negative components.

$$\Delta y = \Delta y^+ + \Delta y^-$$
$$C_{\Delta y \Delta t} = C_{\Delta y^+ \Delta t} + C_{\Delta y^- \Delta t}$$

DeepLIFT defines a few rules for assigning contributions scores. The **Linear rule** applies to DENSE and CONV layers. The **Rescale rule** applies to non-linearities such as ReLU, Tanh, Sigmoid and so on and treats positive and negative contributions equally. The **RevealCancel rule** handles non-linearities, as well, but unlike the Rescale rule gives different treatment to positive and negative contributions.

For the **Rescale rule**, we set *dy(+-)* proportional to *dx(+-)* in the following manner:

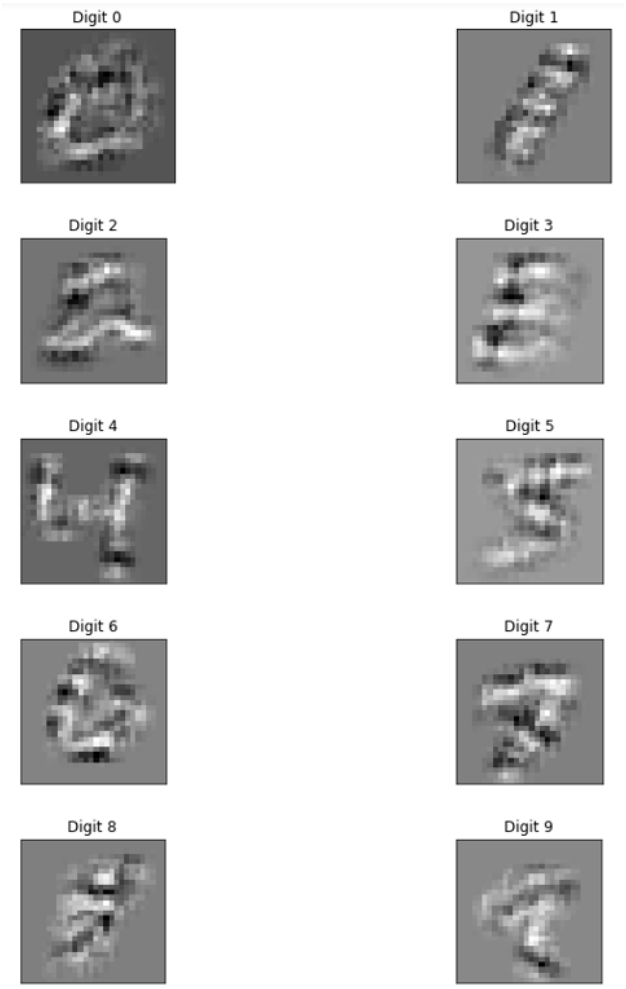
$$\Delta y^+ = \frac{\Delta y}{\Delta x} \Delta x^+ = C_{\Delta x^+ \Delta y^+}$$
$$\Delta y^- = \frac{\Delta y}{\Delta x} \Delta x^- = C_{\Delta x^- \Delta y^-}$$

The multipliers have as a consequence the following form:

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x}$$

In the places, where the values are close to their reference counterparts, numerical instability can be observed due to division by either 0 or extremely small value. In these cases, the gradient instead of the multiplier can be used.

DeepLIFT Rescale Results



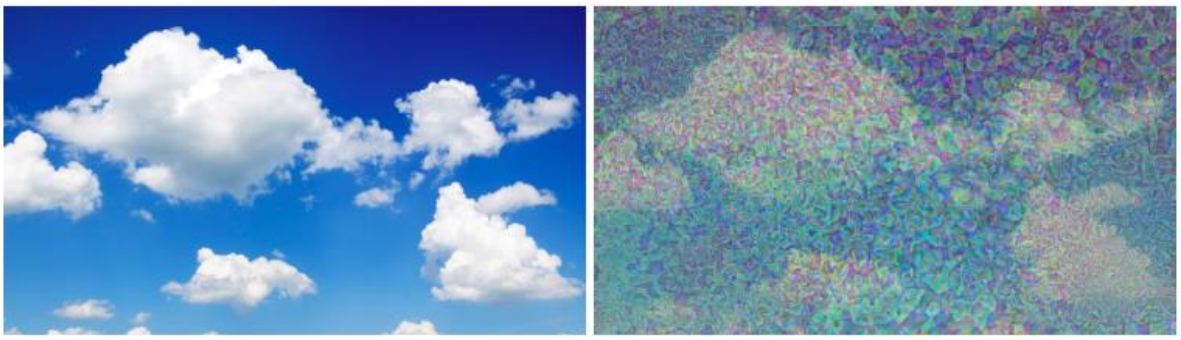
Inceptionism: Going Deeper into Neural Networks | [Paper] | [Notes] | [Implementation]

Deep Dream

Neural networks that we trained to discriminate between different kinds of images have a bit of information needed to generate images. One way is the AM that we start at a random noise and tweak the image until the network considers it as a certain class. Deep dream is a method to **over interpret** features of a image. We feed the network an arbitrary image and instead of prescribing which feature we want to amplify we let the network analyze the image itself. After **choosing which layer and which filter** we ask the network to enhance whatever it detected.

For implementation we downsample the image to different scale and blend the deep dream effects to make the final image looks clearer. Otherwise the deep dream effect will remain in a low resolution and looks like noise.

Deep Dream Results



Original image and image with deep dream effect