Week 11 Report

DeepLIFT Changes

Building up on the work from the last two weeks, we have finished implementing both the Linear and RevealCancel rules for the DeepLIFT method.

The **Linear** rule is used for Linear and CONV layers. The contributions are computed in the following manner

$$C_{\Delta x_i^+ \Delta y^+} = 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^+$$
$$C_{\Delta x_i^- \Delta y^+} = 1\{w_i \Delta x_i > 0\} w_i \Delta x_i^-$$
$$C_{\Delta x_i^+ \Delta y^-} = 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^+$$
$$C_{\Delta x_i^- \Delta y^-} = 1\{w_i \Delta x_i < 0\} w_i \Delta x_i^-$$

.

The **RevealCancel** rule also targets ReLU layers, but unlike the **Rescale** rule it handles the positive and negative contributions in different ways. Here we compute the contributions as follows:

$$\Delta y^+ = \frac{1}{2}\left(f(x^0 + \Delta x^+) - f(x^0)\right) + \frac{1}{2}\left(f(x^0 + \Delta x^- + \Delta x^+) - f(x^0 + \Delta x^-)\right)$$
$$\Delta y^- = \frac{1}{2}\left(f(x^0 + \Delta x^-) - f(x^0)\right) + \frac{1}{2}\left(f(x^0 + \Delta x^+ + \Delta x^-) - f(x^0 + \Delta x^+)\right)$$
$$m_{\Delta x^+ \Delta y^+} = \frac{C_{\Delta x^+ y^+}}{\Delta x^+} = \frac{\Delta y^+}{\Delta x^+}\,;\, m_{\Delta x^- \Delta y^-} = \frac{\Delta y^-}{\Delta x^-}$$

In short, for the positive output contributions we are trying to figure out the average impact of the positive delta input after no terms have been added and after the negative delta input has been added. The same operation is done for the negative output contributions, but in this case, we compute the impact of the negative delta input without any added terms and after the positive delta input have been added.

Furthermore, we have also tried combining different rules. Because we have too many options which would make this report too long, we have included only the results for the Linear and RevelCancel result here. The full results for every possible combination can be found in the repository by going to this **notebook.**

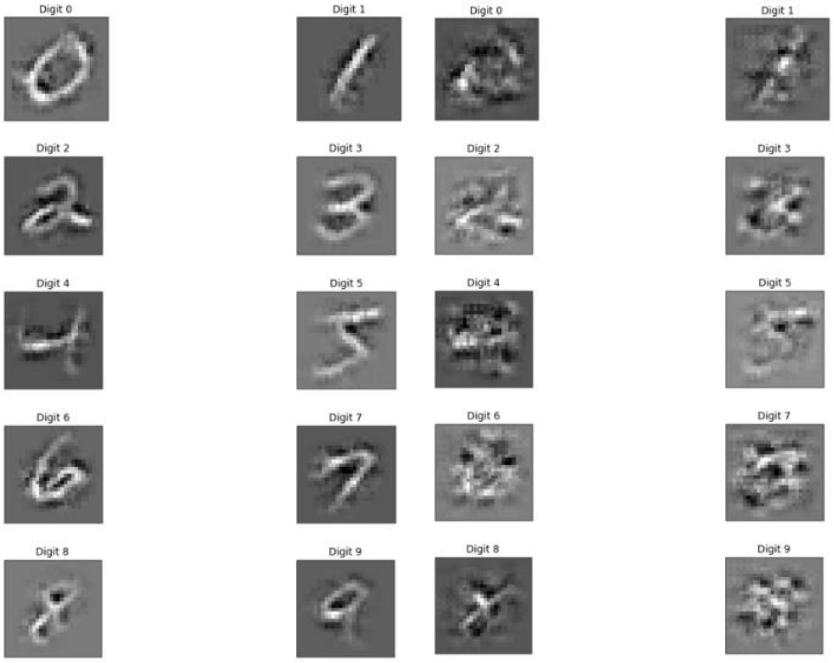


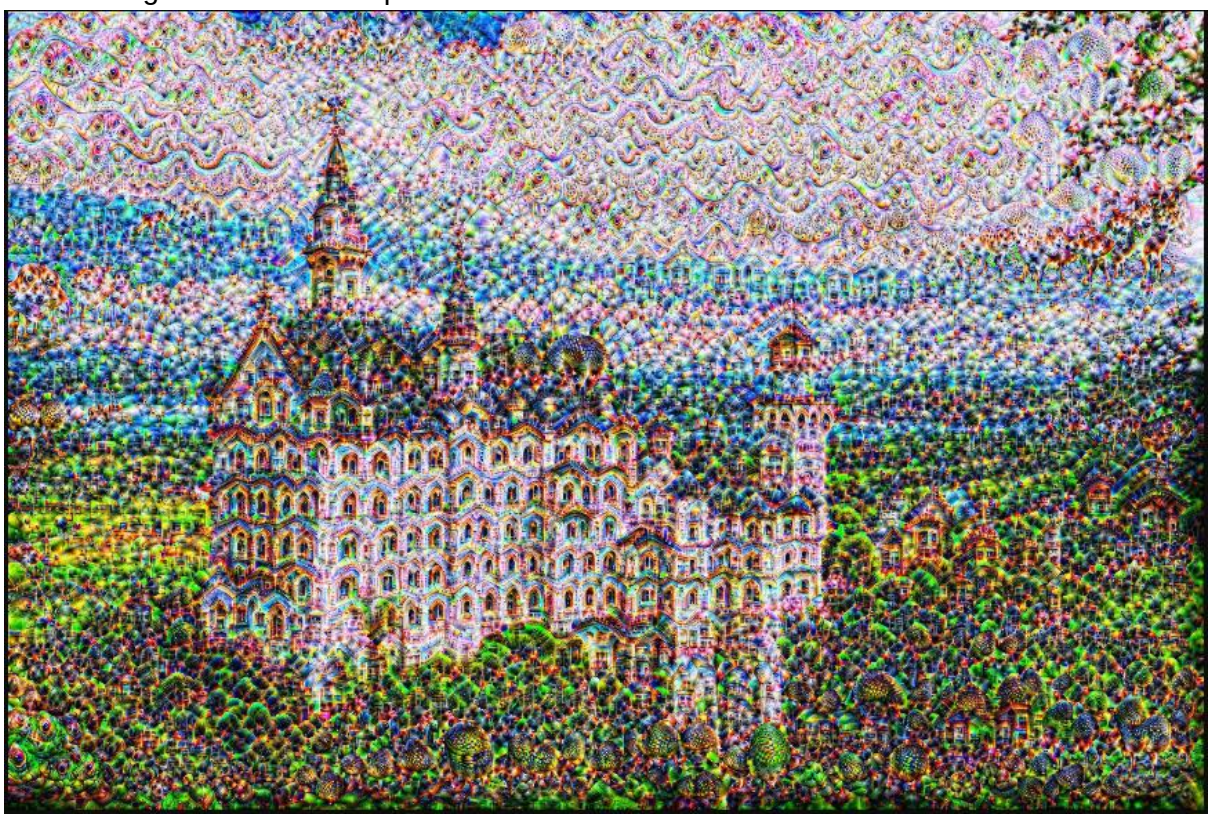
Fig.1. DeepLIFT results: Left is with RevealCancel, right with Linear rule applied

DeepDream Improvements

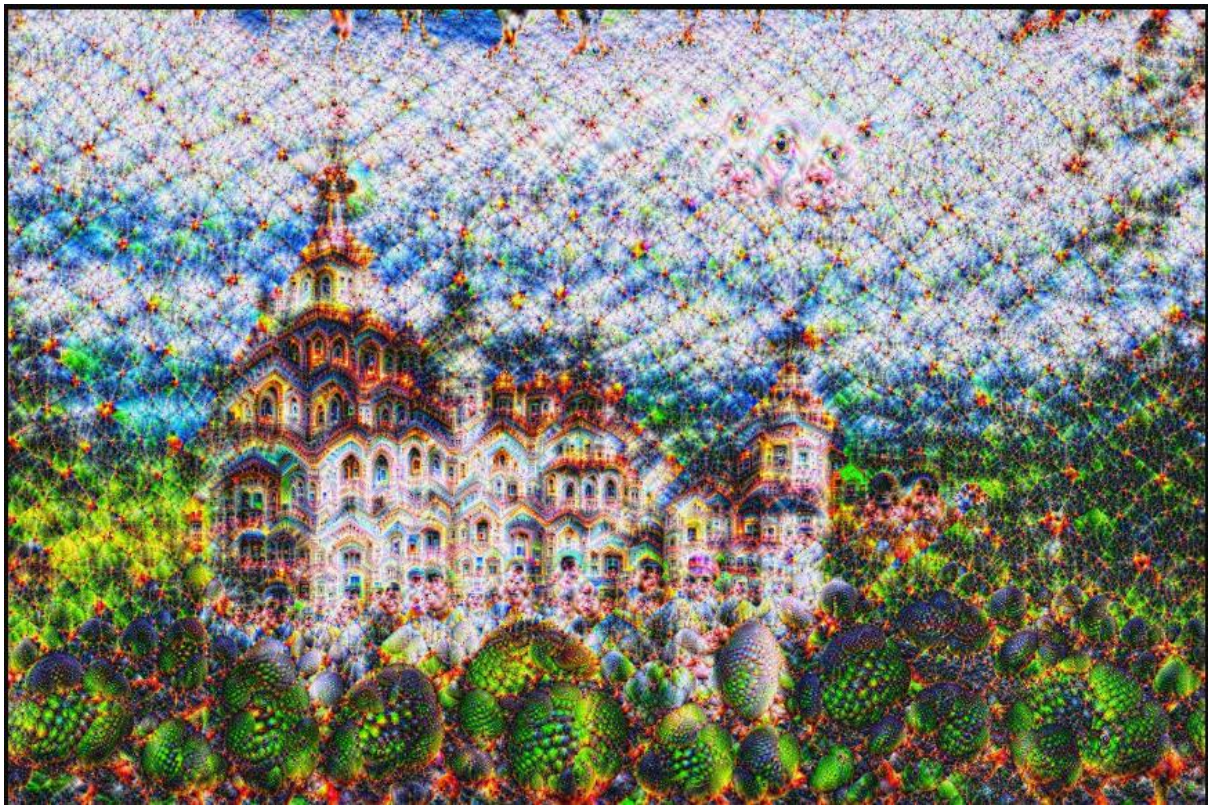
Techniques to improve feature visualization:

1. **L2 Regularization**. Without Regularization we will end up with a low brightness image due to the loss function (mean of pixels)
2. Perform DeepDream on **different scales** (multiple octaves). This technique gives us two benefits 1. Reduce noise 2. Produce patterns in different scales 2. Increase image resolution
3. **Random roll** the image at every epoch. This is a kind of data augmentation techniques that can help us reduce noise
4. **Split** the image into tiles when we compute gradients of the image. This gives us a more scalable DeepDream, where we can upsample the input image to more octaves without GPU memory problem. The current DeepDream only requires 2GB GPU memory to process 8 octaves. Random rolling the image before each tiled computation also prevent tile seams from appearing.
5. Add Gaussian noise after each scaling. This is the idea from SmoothGrad. Currently we only scale the image in 8 octaves so the effect of adding noise to suppress noise isn't that obvious

Result image that use technique 1 and 2



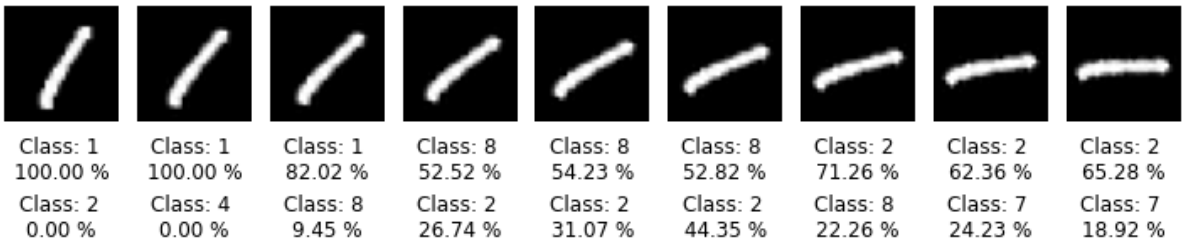
Improvements this time!!! Result image that use all techniques



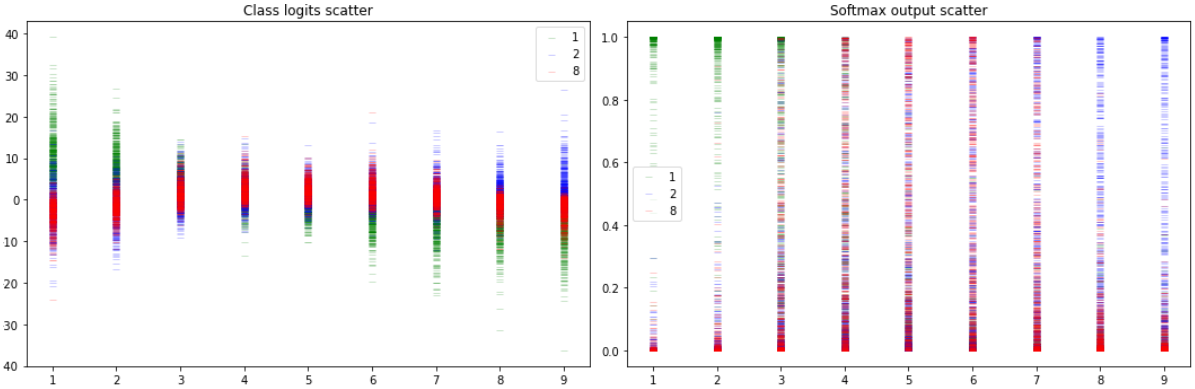
Check our gitlab repository for more high resolution results

Model Uncertainty - MC dropout

While rotating the image we can see the prediction of our model changes. The result below is just the normal prediction without using dropout at test phase. The probability here is the output from softmax.



Perform MC dropout at the test phase for each rotated image. Here the stochastic forward pass T is selected as 500. The two figures below show the distribution of output values from class logits and softmax corresponding to the most predicted class 1(Green), 2(Blue) and 8(Red). The x axis corresponds to the input rotated images.



If the color in the scatter plot are well separated this means our model is more confident with the prediction. For the middle rotated images the colors are much overlapped meaning that the uncertainty of prediction is high. For example though the third rotated image has a very high score from softmax, the prediction preserves high uncertainty.

Interpret Uncertainty with LRP

The LRP rule we used here is a mixture of LRP rules, Composite LRP, which preserves well fidelity and understandability from our previous experiments.

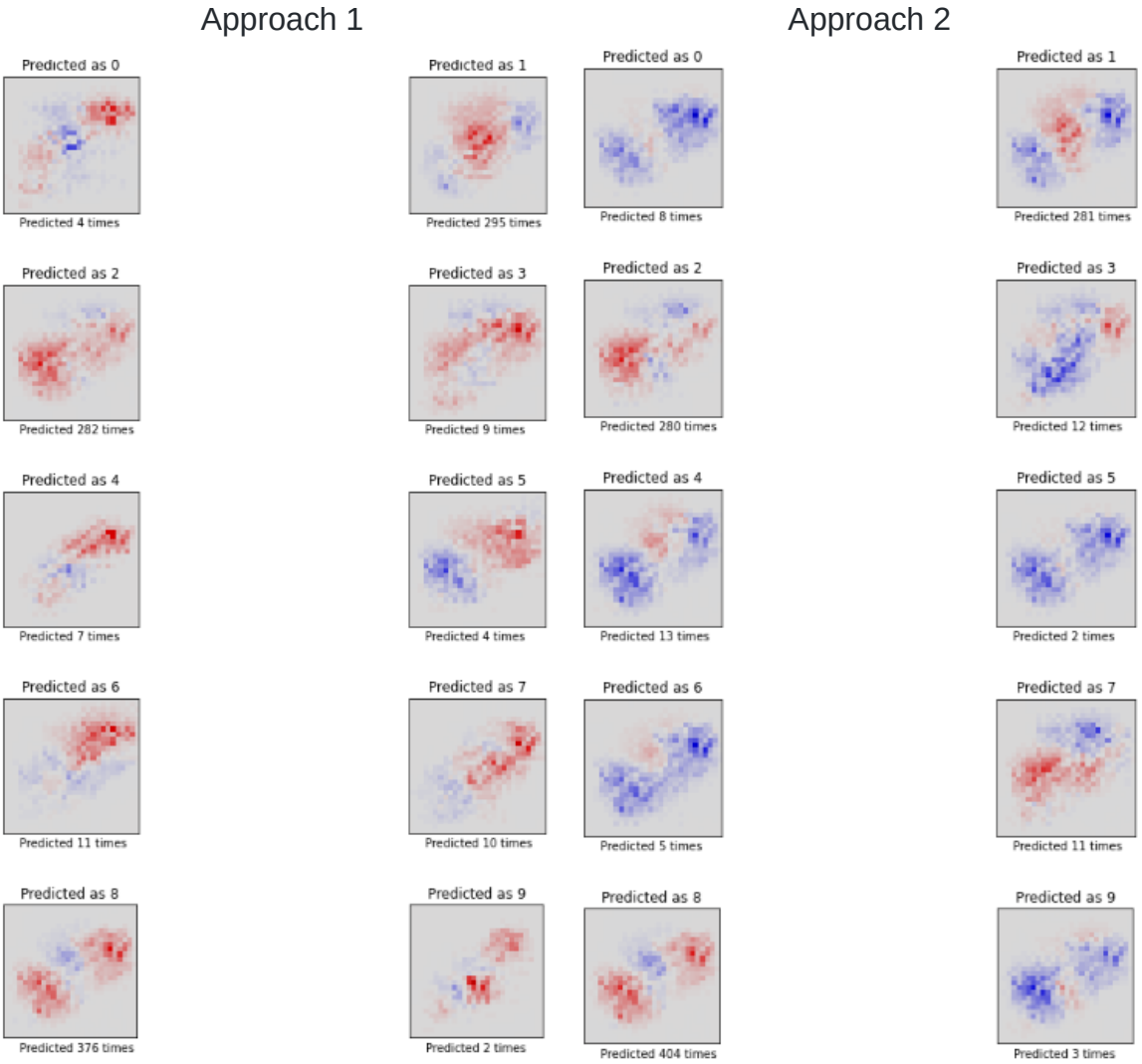
Approach 1 - LRP backward from predicted class

We first observe 1000 predictions. For each prediction we perform LRP backward pass to the bottom layer starting from the **predicted class**. If the predicted classes are the same in our observation, their corresponding heatmaps will sum together. From the following images we see that LRP fails to tell us the uncertainty of the model's predictions.

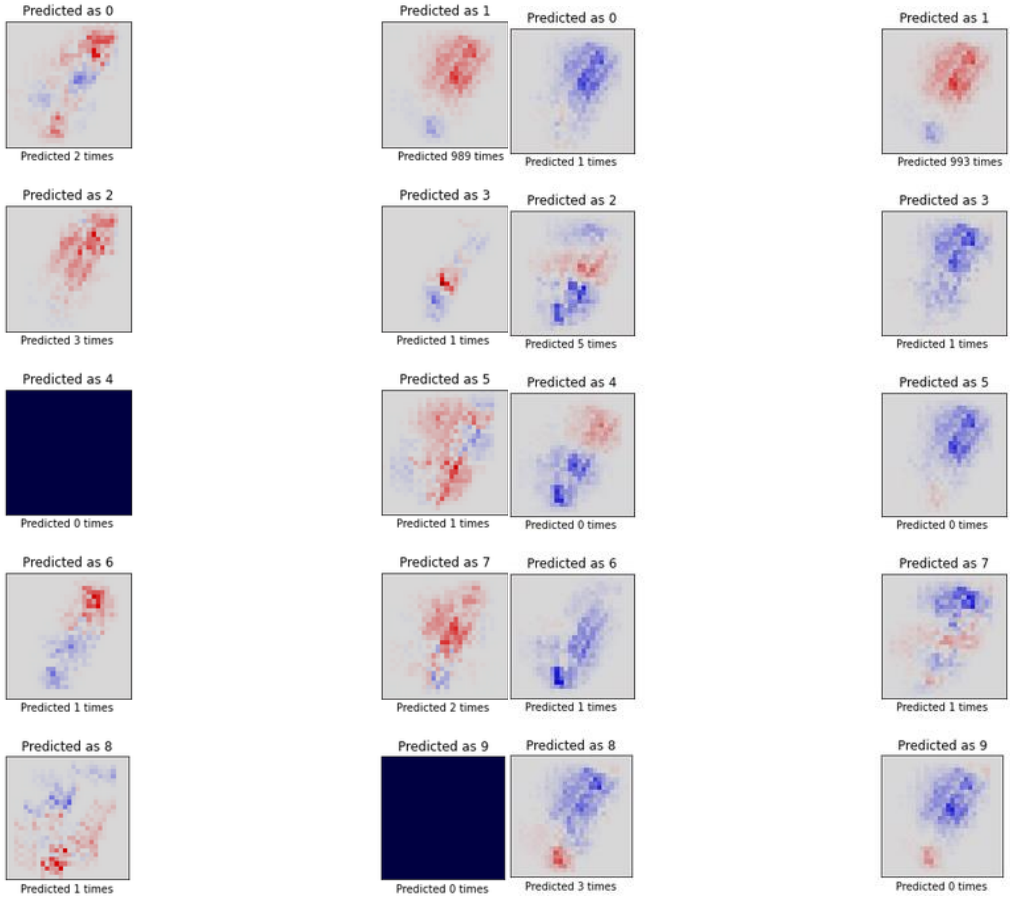
Approach 2 - LRP backward from assigned class

Another approach is to observe 1000 times predictions for each class. Different from the above method, this time we perform LRP backward pass starting from an

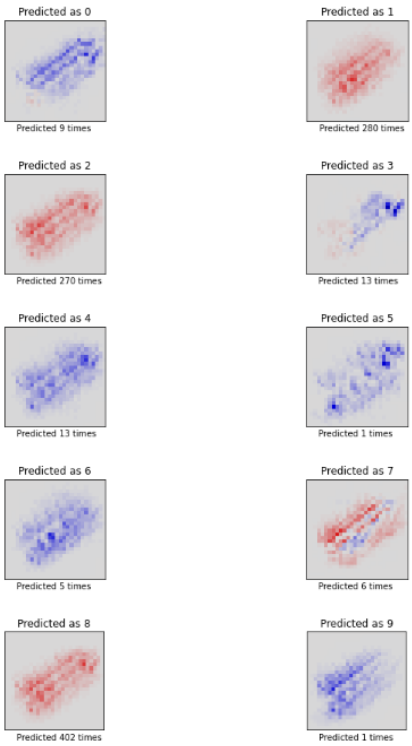
assigned class. We sum the resulting heatmaps from all these 1000 predictions even if the predicted class is not same as the assigned class. This approach shows that for the classes that our model is tend to predict, the resulting heatmap will contain more red pixels. In other words, the more heatmaps consisting mainly of red pixels we have, the more uncertain our prediction is.



Heatmaps from high uncertainty predictions. Approach 2 shows better uncertainty awareness from our predictions.



Heatmaps from low uncertainty predictions. There are fewer heatmaps consisting of red pixels in approach 2. The black figure means that there is no prediction made for that particular class.



We also use **deep taylor decomposition(LRP- alpa1beta0)** to interpret uncertainty following the idea of approach 2 (image above)

Uncertain DeepLIFT

Now we try to introduce uncertainty to the DeepLIFT method and more specifically, the RevealCancel rule. In order to do so, we are going to choose an image and execute DeepLIFT 1000 times for it. In each iteration, we are adding random noise to the image. If the image was classified for the class C, we are adding the result to the collection of results which are classified for the class C. At the end, we are displaying the composite result for each class. Deep blue image refers to classes for which no classifications have occurred.