Visualising and Understanding

Tejas Gaikwad
MT19AI021
Dependable AI

Explanation of model prediction and reasoning for predicting an output by the model is important to understand what model is learning. While designing a model like Convolutional Neural Networks(CNN), we can visualize the learned parameter by a particular layer of the model. Earlier it was like a black box, what's inside was unknown to us. This can be done at different locations of CNN, may it be the first layer, any Intermediate layer or the last layer of the network. The most visually predictable layer is the first and the last layer of the network.
On visualizing the first layer, we understand that the learned weights represent the details related to the colors, oriented edges, bars of light and the dark. For the intermediate layers, the information is not much visually understandable. The last layer tells the learned class representation. With the help of this visualization, using the nearest neighbor approach triplet loss or we can say the last layer loss can be minimized.
The methods for understanding the CNN visualizations/ representations can be segregated into 2 parts

1. Activations
2. Gradients

Activations:
As discussed above for the last layer i.e. the layer before the classifier, we can use the **nearest neighbor approach** between test images and the output obtained and minimize the loss function for obtaining better prediction at the output. For the last layer let say, the size of the layer is 4096, **dimension reductionality** can also be done to visualize the parameters in 2-D. For this t-SNE or PCA can be used for dimension reductionality depends on the requirement. t-SNE is more complex and gives a sharp output as compared to the PCA. PCA gives blurry output.
Visualizing activations can be done by picking a layer and a channel lets say we pick Conv5 layer 126x13x13, and channel 17/128 then run many images through the network, record values of the chosen channel. Then visualize image patches to the maximum activations. (With this method, we get to know that one which features the model is focusing on, for example the particular layer might focus on circular shaped patches or might focus on sharp edges, etc)
Occlusion: Another method to know the critical patches of the image. Occlude some region from image then analyze the prediction. This can be done to identify the most important region of the image that is responsible for prediction. **Heat map generations** also tell us the critical part of the image responsible for the predictions. The most white or yellow part of the image represents the critical part of the image.

Gradient Method:
Visualizations based on compting gradients
Saliency Map:
It can be used to tell which pixel matter for classification. The gradient is computed of the (unnormalized) class score with respect to image pixels, that take absolute value and max over RGB channel because of which we get an image showing white spots on the black background where white spots represent the critical pixels to the image.
Intermediate features via backpropagation:
Guided backpropagation to find part of the image that neuron responds to. Gradient computation of neuron value w.r.t. Image pixels is done and then updated using backpropagation. Due to this, it results in some patterns in patches, better view as compared to forward propagation. Eg. Words, Curves, etc.
Visualizing CNN Features: Gradient Ascent:
Generating a synthetic image that maximally actuates the neuron, and with the help of this neuron information, the weights are updated further. This feature can also be used to visualize intermediate features.
Fooling Images/ Adversarial Examples:
After the computation of the gradient, we get to know about the critical patches of the image. Changing these patches or even a small pixel which is having a very high influence on prediction, the model can be fooled. For example, adding noise to the image which does not change the visual appearance but fools the classifier, let us consider the case of the image of the panda which on perturbing makes the model predict it as a goose.
Feature Inversion:
For a given CNN feature vector for an image finding a new image that matches a given feature vector which after regularisation looks like a natural image. This is obtained by minimizing the loss function between the given feature vector and feature of the new image.

These visualization techniques can also be used to create some interesting artistic images
DeepDream: (By Google)
Amplifying the existing features to create some interesting artistic images. This can be done by selecting an image and a layer in a CNN, computing activation at the chosen layer in forward direction, setting gradient of chosen layer equal to its activation. Then computing gradient in backpropagation as I discussed above and then updating the image vector. Repeating these steps until we get some cool textures over the base image.
Texture Synthesis:
Given a small patch of texture and creating a large continuous image. It is done by generating pixels one at a time in scanline order form the neighborhood of already generated pixels and then copying the nearest neighbor from the input.
Neural Texture synthesis: Gram Matrix
The “Gram matrix” of feature activations is used to represent texture, basically a covariance matrix of features. In this, each layer of the CNN gives CxHxW tensors of feature HxW grid of the C-Dimensional vector. The outer product of this C-dimensional vector gives the CxC matrix measuring co-occurrent. On averaging all the HW pairs of the vector, it gives a gram matrix of

dimensions CxC. Then compute the gram matrix at each layer compute the gram matrix giving an outer product of features. Now, Initialise an image from random noise and pass the generated image and compute the gram matrix for each layer. Compute the loss i.e. computing the weighted sum of L2 distance between the gram matrix. After calculating the loss, perform backpropagation to get the gradient on image. Make a gradient step on the image and repeat the steps until we get minimal loss.

Style Transfer:

Now, in the same manner, the style transfer can also be performed. Style transfer is nothing but forming a new image with the help of parameters learned and the style image where the output image appears similar to the style image perform this, we require 2 images, one is styled image and the other one is the content image, keeping the output image as a noise generated image(for the start. On minimizing the distance between the output image and the style image, the final output image is computed.