CS231n: CNNs for Visual Recognition **Spring 2017**

Lecture 12

Course Coordinator: Prof. Fei-Fei Li *Scribes: Akash Gupta*

Looking inside ConvNets -

- Visualize Filters or raw weights -
  - for first layer looking at oriented edges and opposing colors.
  - Second conv layer is less interpretable on visualizing.
  - Last layer -
    - Nearest Neighbor visualization on feature space.$->$ Pixels are different but semantic meaning is same between neighbors.
    - Dimensionality reduction - PCA, t-SNE (Non-linear)

Q) *Why visualizing filters tells what filter is looking for?*
Ans) Comes from the idea of template matching where we take inner products of template(filter) and arbitrary data to maximally excite the activation. So the visual aspect of the weights will tell us what kind of features the filter is looking for in the image.

- Visualize Activations -
  - E.g. - Visualizing the output of a conv layer in a 2D grid to see which intermediate feature is activating on the portions of the human face.
  - Output of a conv layer is 3D volume and is called an activation volume. Each slice in this volume is called an activation map.
  - Maximally activating patches - Keeping track and visualizing patches that maximize an activation in an intermediate layer.

Occlusion experiments -

- Block out a part of image using some mean value from the dataset and now run that image through that trained network and get a predicted prob.
- Slide the blocking patch through the image and compute the predicted prob. Then compute the heat map showing which parts of the image on blocking out leads to fall in prob. In other words, which are the most important features in the image for classification.

Saliency Maps -

- Compute gradient of class score w.r.t image pixels, take absolute value and max over RGB channels
- Gives an idea of which pixels in the image matter for the predicted class.
- Saliency maps can also be used for semantic segmentation task without having supervised labels by using a GrabCut segmentation algorithm. However the performance is much worse than general supervised segmentation.

Intermediate features via (guided) backprop -

- Compute gradient of intermediate neuron w.r.t image pixels.
- Guided backprop -
  - Slight tweak in normal backprop that ends up giving cleaner image patches.
  - Change the way of backprop thru ReLUs where only +ve gradients thru ReLUs are kept and -ve ReLUs are zeroed.

NOTE - Here we are computing saliency maps/guided backprop w.r.t an input image. Another question is remove this reliance on input image and answer in general. This is answered using the idea of Gradient Ascent.

Gradient Ascent -

- The idea is to generate a synthetic image that maximally activate a neuron.
- We fix the weights of the trained CNN and instead synthesize an image by performing Gradient Ascent on the pixels of the image to maximize score of some intermediate neuron or a class.

$$I^* = argmax_I \, f(I) + R(I)$$

  $f(I)$ is the neuron value, $R(I)$ is the Natural image regularizer.
- Algorithm -
  - Initialize image either 0 or gaussian noise.
  - Forward to compute current scores on pretrained network.
  - Backprop to get the gradient of neuron value w.r.t image pixels.
  - Make a small update to the image.

$->$ Adding regularizer is important to make visualizations look more natural. Else it's just looks like noise but has a high class score (Similar to the idea used in generating Adversarial examples!)

$->$ Other ideas include taking multi-modality into account (other objects also present in the image) and instead of optimizing image pixels optimize latent space (Produces even better images)

Some fun! -

- DeepDream by Google - Instead of synthesizing an image to maximize a specific neuron, instead try to amplify the neuron activations at some layer in the network.
- Feature Inversion - Reconstruct images from features representation. Based on what that image looks like it will give a sense of what type of info was captured in that feature vector.
  - Minimize the distance b/w cached feature vector and feature vector for which we want to generate the image.
  - Variational regularizer - encourages spatial smoothness in generated image.

$$R_{V^\beta}(x) = \sum_{i,j}((x_{i,j+1} - x_{ij}) + (x_{i+1,j} - x_{ij})^2)^{\frac{\beta}{2}}$$

Texture Synthesis - Given a sample patch of some texture, we want to generate a bigger image of the same texture.

- Nearest Neighbor for simple textures
- For complex textures - Gram Matrix - Gives the idea of a second order correlation between different features in an image.
- Now during reconstruction, instead of reconstructing the whole feature map of image, we reconstruct Gram Matrix feature descriptor.

Neural Style Transfer - Combine ideas from Texture synthesis and feature inversion. Gives a lot more control on what to generate.

$->$ Problem very slow. Solution $->$ Train another NN to do style transfer (Fast Style Transfer)

$->$ One network, Many Styles by Google $->$ *Domoulin et. al. A Learned Representation for Artistic Style. ICLR 2017*