Notes on AlexNet, VGG, and Visualizing CNNs

Feb. 28, 2020

AlexNet (2012)

- CNNs had been used for image classification before, but this was the first time the approach was extremely successful.
- Abstract:
  - Architecture overview:
  - "non-saturating neurons" -> ReLU, not sigmoid or tanh
  - regularization via dropout
- Introduction:
  - "Our network takes between five and six days to train on two GTX 580 3GB GPUs. All of our experiments suggest that our results can be improved simply by waiting for faster GPUs and bigger datasets to become available."
- Data Set:
  - ImageNet had 15 million images
  - They cropped to 256 by 256
- Architecture:
  - ReLU was a relative novelty
  - "saturating neuron models"
  - They had to jump through hoops with GPUs that we don't have to worry about so much anymore
  - Local response normalization – compare to input normalization. But next paper does not find this to be helpful.
  - Overlapping pooling. I don't think people worry too much about this
  - Infamous for cutting figure off.
  - 11 by 11 filters! Noone does this anymore (see next paper).
- Strategies to prevent overfitting:
  - Data augmentation
  - Dropout
  - Weight decay (mentioned in next section)
- Learning:
  - SGD, batch size 128, momentum 0.9, "weight decay" = L_2 regularization of 0.0005
  - Initialize using standard ideas
  - Train for "5 to 6 days"
- Results:
  - Ensemble better
  - "Training one CNN, with an extra sixth convolutional layer over the last pooling layer, to classify the entire ImageNet Fall 2011 release (15M images, 22K categories), and then "fine-tuning" it on ILSVRC-2012 gives an error rate of 16.6%."
  - The problem is hard – their error rates are still pretty big.
- Qualitative Evaluations:
  - Ideas for visualization:
    - Show test set images with class probabilities. Trying to understand why model is wrong.
    - Show test set images with "nearest" training set images.

VGG16 (2015)

- Abstract:
  - Contribution is to increase depth while using small (3 by 3) filters
  - Representations generalize well to other data sets
- Architecture:
  - Input is 224 by 224 (relevant later)
  - Operations:
    - 3 by 3 filters
    - 1 by 1 filters
    - pooling with 2 by 2 window, stride 2
  - General Plan: Convolutions (ReLU) -> Max Pooling -> Convolutions (ReLU) -> Max Pooling -> ... -> Fully Connected -> Softmax
    - See table page 3
  - Local response normalization doesn't improve performance and slows things down
  - Insight at bottom of page 2: stacking 3 consecutive layers of 3 by 3 convolutions has:
    - An "effective receptive field" of 7 by 7
    - Fewer parameters than a 7 by 7 filter ($3 * 3 * 3 = 27 < 49 = 7 * 7$)
    - More opportunities for non-linearities via ReLU activations
  - 1 by 1 filter is an opportunity to get more non-linearities
- Training:
  - All settings look familiar by now
  - Ideas about training image size. For us, main thing to note is that you can take estimated weights from one image size and use them for another image size.
  - "On a system equipped with four NVIDIA Titan Black GPUs, training a single net took 2–3 weeks depending on the architecture."
- Results:
  - From Table 2, models C and D both have 16 layers; C has 1 by 1 convolutions while D has 3 by 3 convolutions. E has 19 layers.
  - Models D and E are best
  - Error rates are **much** better than what was reported by AlexNet only 3 years prior.
- Appendix B!!
  - "deep image representations, learnt on ILSVRC, generalise well to other datasets, where they have outperformed hand-crafted representations by a large margin."
  - "To utilise the ConvNets, pre-trained on ILSVRC, for image classification on other datasets, we remove the last fully-connected layer (which performs 1000-way ILSVRC classification), and use 4096-D activations of the penultimate layer as image features, which are aggregated across multiple locations and scales. The resulting image descriptor is L2-normalised and combined with a linear SVM classifier, trained on the target dataset. For simplicity, pre-trained ConvNet weights are kept fixed (no fine-tuning is performed)."
  - See Table 11 for results; this approach does well.

Visualizing CNNs (2013)

- Abstract:
  - Gonna do some visualizations
  - Also exploring transfer learning
- Approach:
  - We know how these networks look and how estimation works now.
  - First 2 sentences of section 2.1: "Understanding the operation of a convnet requires interpreting the feature activity in intermediate layers. We present a novel way to *map these activities back to the input pixel space*, showing what input pattern originally caused a given activation in the feature maps."
  - To do this, we have to try to approximately invert a bunch of non-invertible transformations that happened in forward prop. Figure 1 shows how this works for max pooling.
  - Why is transposing the filters the right thing to do? Not explained here, read Zeiler et al. 2011 to find out.
  - "We stopped training after 70 epochs, which took around 12 days on a single GTX580 GPU"
  - Figure 2:
    - Greyish stuff is the inversion of the activations from the deconvolution model. We also see the corresponding actual segments of images.
    - Layer 1: detecting very simple things: solid patches, diagonal lines in different colors
    - Layer 2: textures, circles, ridges, color-specific
    - Layer 3: some concepts emerge: mesh, "red object on background", text, object in lower right corner, off-white textured thing in lower right corner
    - Layer 4: "dog with white and brown face", "bird with long legs", some of these harder to classify
    - Layer 5: some car wheels look like human faces.
  - Figure 4:
    - Feature activations in later layers do not emerge/stabilize until later in training
  - Figure 5:
    - Feature activatiosn in later layers less sensitive to things like image translation and rotation
  - Figure 7:
    - Blocking important parts of an image affects things like you'd expect
- Experiments:
  - Changes made based on visualizations were helpful
  - Transfer learning works