Fooling Conv Nets

Adding invisible noise

Shah Rukh Qasim
Abdul Muiz Ahmed
Tooba Imtiaz
Muhammad Azeem Sabir
Moeez Aziz
Ahmed Alam

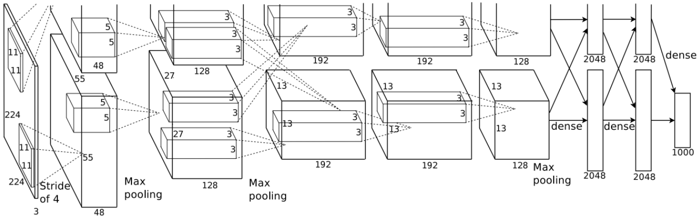
January 09, 2018

Intro to the network

- Krizhevsky et al. (AlexNet) [1]
- Presented in 2012
- Currently has 18,400+ citations

I/O

- Input image is 3 channel 224x224
- There are many conv layers and 2 dense layers
- Trained on 1000 classes of image-net



Input Normalization

- Input image is normalized using z-transform
- z-transform is applied separately on different channels
- For dataset, $\mu = 0, \sigma = 1$ across different channels

Getting probabilities

- Softmax is applied on last layer to get probability score
- $\Sigma x_i = 1$
- We pick the class with maximum probability score
- Softmax cross-entropy with logits is used during training
- We don't need to take softmax to pick maximum

Loss function

- Loss to fool the conv-net:

$$L = 1 - p_f$$

- f is the class to change the prediction to
- p_k for any value of k will not go above 1 guaranteed by softmax

Adding noise

- Add noise to each channel
- The noise is random uniform initialized between 0 and 0.01
- This noise is very small to show any visible output

Optimizing noise

- Apply descent on noise to minimize the loss function
- On each pixel value

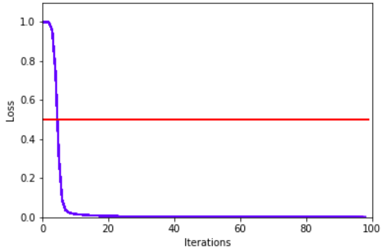
$$\begin{aligned} p'_{ij} &= p_{ij} + n_{ij} \\ \text{where } p_{ij} &= \text{Pixel at location (i,j)} \\ n_{ij} &= \text{Noise at location (i,j)} \\ p'_{ij} &= \text{Modified pixel at location (i,j)} \end{aligned}$$

- Subject to restraints

$$\begin{gathered} -c < n_{ij} < c \\ 0 < p'_{ij} < 255 \\ \text{where } c = \text{Noise value clamp} \end{gathered}$$

Loss reduction

- Adam optimizer [2] is used for optimizing noise



- The loss is easily converged in $<$100 iterations

Technologies used

- Python 3.5 is used
- Pytorch [3] is used as differentiation engine
- Pillow is used for image loading
- Matplotlib is used for plots and image display

Conclusion

- The code is all public https://github.com/shahrukhqasim/FoolingConvNets
- With instructions on how to run
- Demo

References

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.