Training a neural network in practise

Topics

- Activation functions
- Data preprocessing
- **Weight initialization**
- Batch normalization
- Weight update schemes
- Searching for the best parameters

Batch normalization: training

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Batch normalization: test time

- At test time: mean/std is computed for the ENTIRE TRAINING set, not mini batches used during backprop (you should store these).
- Remark: use running average to update

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

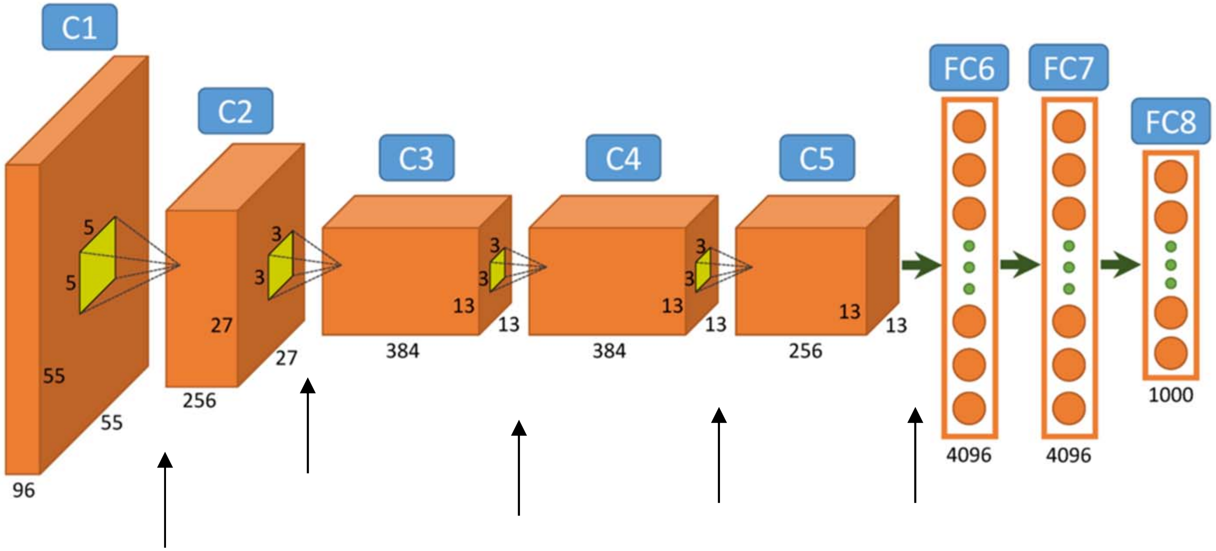
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Visualization

- Visualization filters
- Visualizing activations:
  - Occlusion experiments
- Visualizing class activation maps
  - Guided backprop
  - Gradcam
- Gradient with respect to the image
  - Saliency maps
- Fooling the network (more in a later lecture)
- Feature inversion
- Neural style transfer

What do the layers learn?



What does these intermediate features look like?
Can this help us gain confidence in what the network learns?
How can we fool the network?

Can we visualize the filters themselves?

- Useful for the first couple of layers, then difficult

Visualizing which pixels are most important for a class

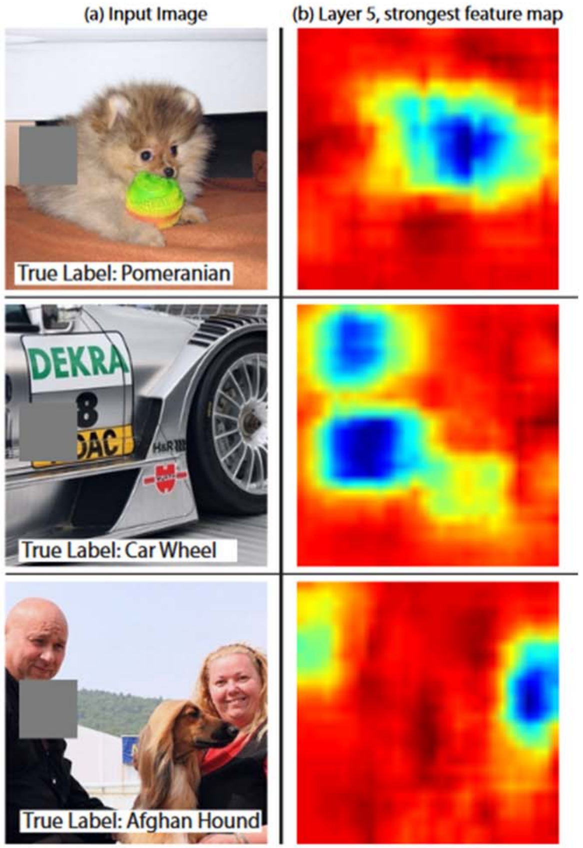
Occlusion experiments

Saliency maps

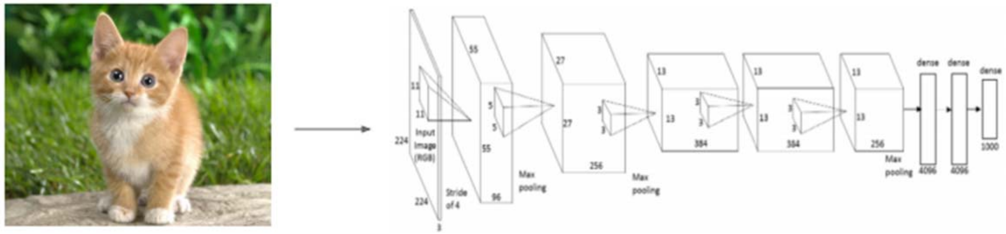
Occlusion experiments

- Create a small patch of zeros.
- Slide this over the image and zero out pixels inside the patch.
- Classify all these images.
- Record how the probability for the given class change over the image as the mask shifts.

Zeiler and Fergus 2013 – occlusion experiments



DECONVNET: Gradient of a neuron with respect to the image



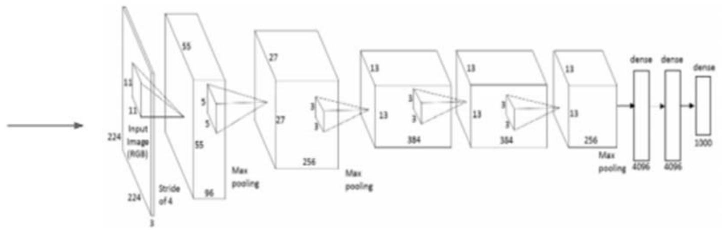
Treat the image as a variable and the network weights as constants

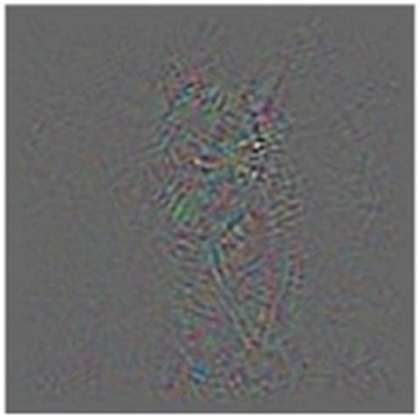
1. Run the image through the network
2. Set the gradients at the layer you want to be zero, except for the neuron of interest
3. Backprop all the way back to the image

Guided backprop

- Springberger et al. (2015) has a couple of interesting points
  – They show that pooling can often be replaced by strided convolution
  – They show that deconv can be improved by only backpropagating positive gradients (called Guided backprop)

Deconvnet vs. Guided backprop





Deconvnet

Guided backprop

More focused

Visualizing class-specific activation maps

GradCam

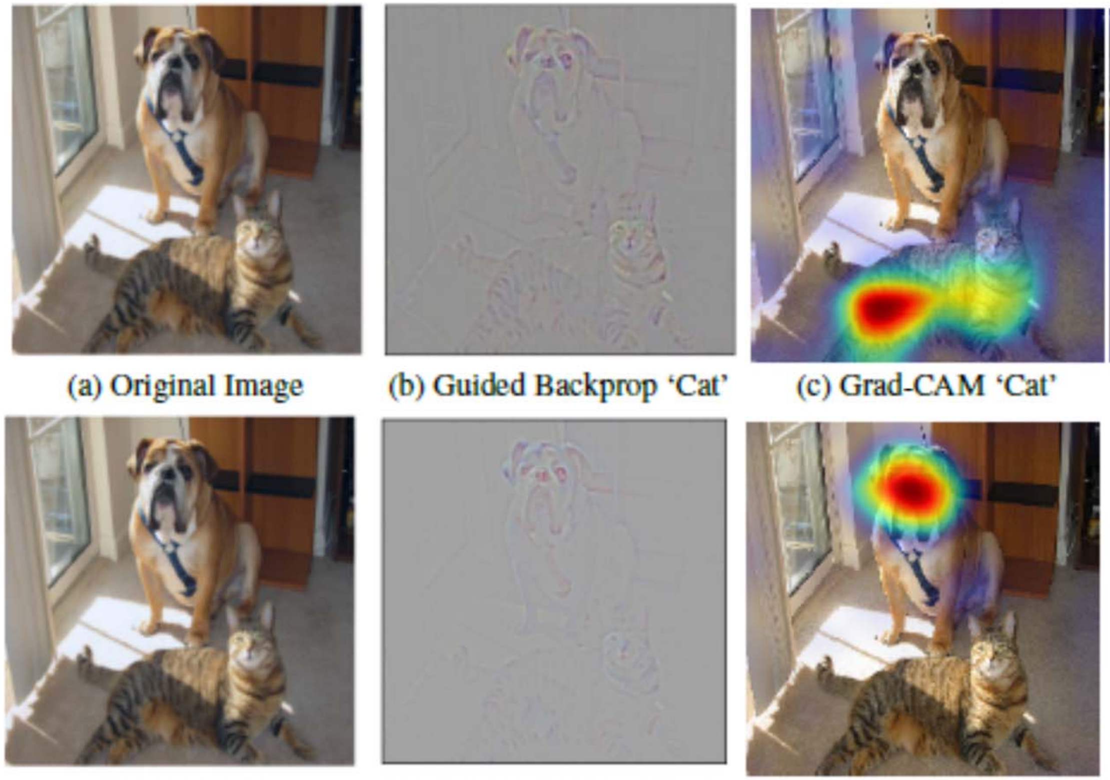
- Visualising explanations from Deep Networks via gradient-based localication

- Drawback of CAM: only works for pure convolutional architectures with general average pooling before softmax.
- GradCAM: use gradients flowing into the last conv-layer.

GradCAM principles

- Start with the score for class c before softmax y_c
- Compute the gradient of this with respect to the feature maps A_k of a conv-layer.
- Then apply GAP of these for all locations to get a weight α_k^c
- Then get the GradCAM localization map as ReLU of a linear combination Ak and α_k^c

GradCAM



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog'

Gradient ascent with respect to the image

- Simonyan, Veldaldi, Zisserman
- Goal: find **an image** such that the score S_c for class c a maximized.

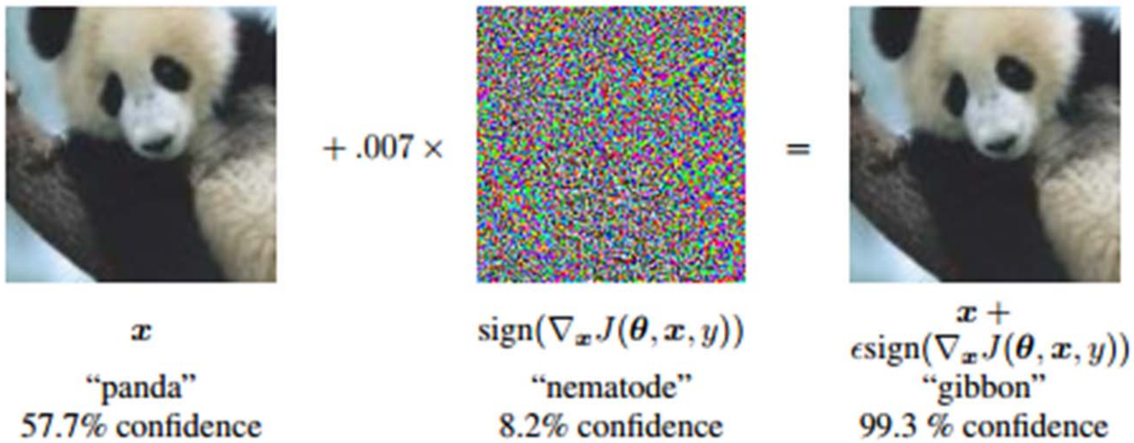
$$\arg\max_{I} S_c(I) - \lambda \|I\|_2^2,$$

Applications for image gradients

Fooling a network

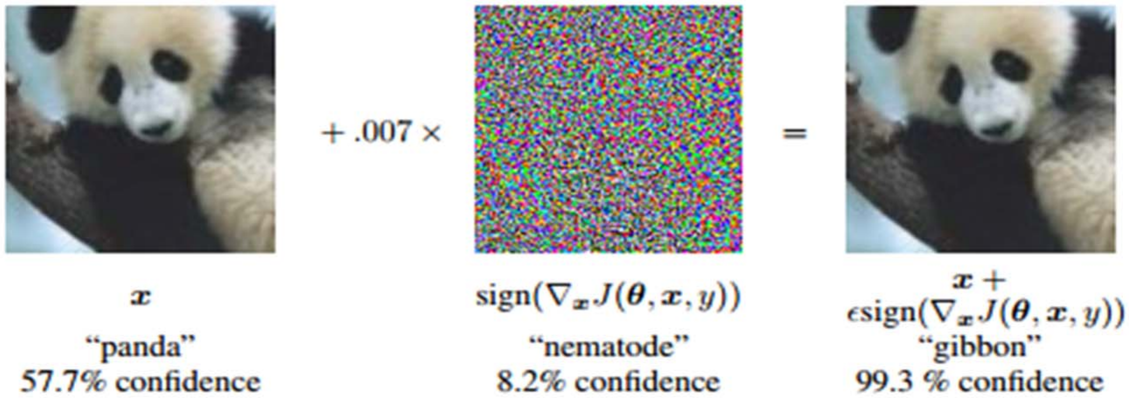
Fooling a neural network

- Adding small values to every pixel gives large change in euclidian distance
- Network representations are different than human representation
- This is not inherent to deep learning or neural networks



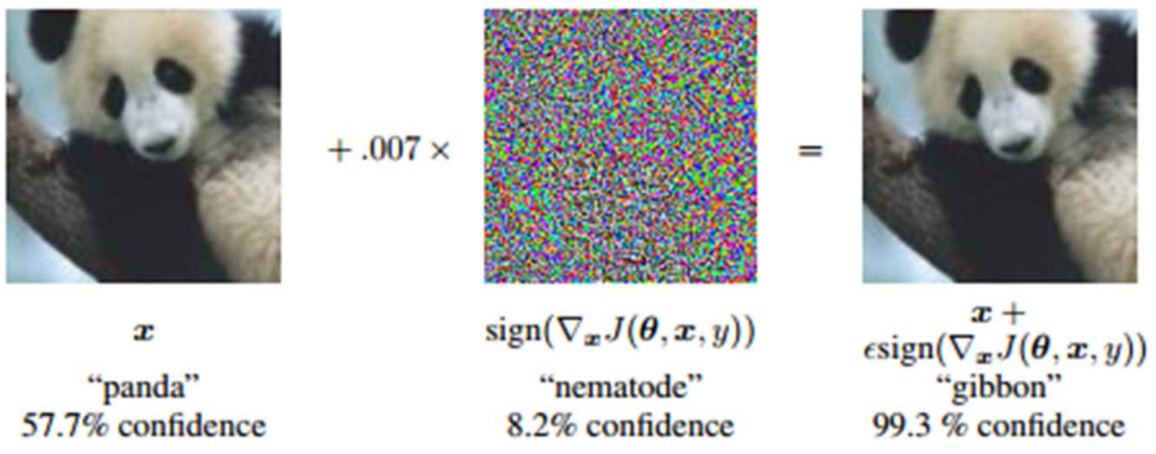
Is there a fix?

- Forcing perturbed images to give similar representations at different levels
- Training on adversarial examples
- Adding noise to training
- Adding noise and smoothing on input images



A final solution is hard

- If you have access to the gradient, there will alway be some “small” direction that can fool the network
- This is not inherent for deep learning, but also exists in other machine learning models




SO, IS THIS GOING TO BE ON THE TEST??
HOW TO ANSWER:
IF I SAY YES
IF I SAY NO
THAT'S TOO EASY.
THEY WON'T LEARN IT.
BUT THEN IF IT'S NOT ON THE TEST
IF I SAY MAYBE OR ANSWER VAGUELY
THEY'LL COMPLAIN
JORGE CHAM © 2016
IT WASN'T GOING TO BE, BUT THANKS FOR THE SUGGESTION...
NOOOO!!
heh heh.
WWW.PHDCOMICS.COM