10417/10617

Neural Network for Vision Part II

Neural Network for Image Classification

- So we learned convolution neural network.
- Given training images $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$, and their corresponding labels $y^{(1)}, y^{(2)}, \ldots, y^{(n)}$.

Neural Network for Image Classification

- We can use a convolution network W to minimize the empirical risk:
- $\min_{W} \frac{1}{N} \sum_{i \in [N]} l\left(h\left(W, x^{(i)}\right), y^{(i)}\right) + R(W)$
- How do we test if the training works well or not?

Test accuracy

- After training, we can test the performance of a neural network using new images $x'^{(1)}, x'^{(2)}, \dots, x'^{(m)}$ and their labels $y'^{(1)}, y'^{(2)}, \dots, y'^{(n)}$, and test to see if the trained neural network can predict the labels of the new ones:
  - $h(W, x'^{(i)}) \approx y'^{(i)}$
- For example, if $h(W, x \quad) \in R^K$, and $y \in [K]$, then we want to test if
  - $argmax_{k\in[K]} h_k(W, x'^{(i)}) \ = \ y'^{(i)}$

Test accuracy

- After we minimize the training loss:
  - $\min_{W} \frac{1}{N} \sum_{i \in [N]} l\left(h\left(W, x^{(i)}\right), y^{(i)}\right) + R(W)$
  - So that $h(W, x^{(i)}) \approx y^{(i)}$
- Does it mean that for new images
  - $h(W, x'^{(i)}) \approx y'^{(i)}$?
- In other words, does the trained neural network generalizes its prediction power from training dataset to test dataset?

Generalization versus Memorization

- Let us consider a simple problem, where the true label is $y = x_1$ (the first coordinate of x)
- The neural network that generalizes:
  - $h(W, x) \approx x_1$ for every x.
- The neural network that minimalizes the training loss but does not generalize:
  - $h(W, x) = \sum_{i \in [n]} x_1^{(i)} 1_{x = x^{(i)}}$
- In fact, its very easy for a neural network to represent this sum of indicator function, using two-layer ReLU network.

Neural Network for Image Classification

How do we make sure that after training a convolution network, it “generalizes” to new images, instead of simply memorizing a bunch of “if” statements for the training images?

Solution 1: Reduce the number of parameters of the neural network

Not a good solution, neural network is performing “feature learning”, we need a large number of neurons to represent certain features.

Solution 2: Increase the number of training examples.

Data augmentation

Solution 2: Increase the number of training examples.

We can use human labelers to label more images …

Can we increase the number of training examples without getting more labeled images?

Solution: Data augmentation.

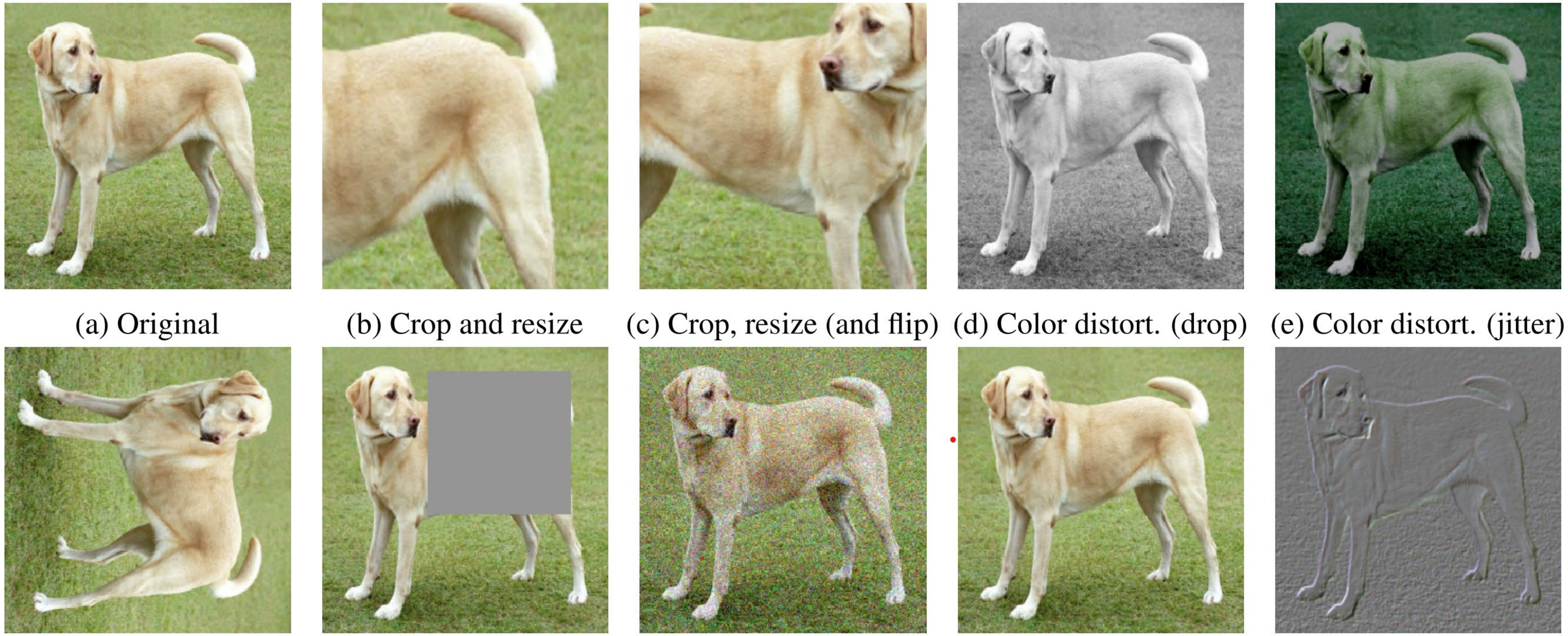
Data augmentation

We can bootstrap a lot more images from the original one while keeping the labels to be the same.

This is called data augmentation for images.

Data augmentation

- We can add the augmented new images (with their labels) to the training dataset.



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter) (f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Contrastive Learning

- What does a neural network that learns $h(W, x) = \sum_{i \in [n]} y^{(i)} 1_{x=x^{(i)}}$ look like?
- Key observation: The hidden embedding of the last layer $h_L(x)$ should have low diversity (because only a linear function is applied on top).
- For example, $h_L(x) = v_i \, 1_{x=x^{(i)}}$ for some vectors v_i .

Feature diversity

- We want to make sure that features in $h_L(x)$ are as diverse as possible.
- Good $h_L(x)$
  - (There is a wheel?, There is a window? , There are furs?, Color = Blue?, There are wings? There are tails? There are horns? , Length of the legs?, ….)
  - A diverse set of features.
- We want the cardinality of $h_L(x)$ to be as large as possible, in other words, $h_L(x)$ should span the entire space, instead of just being a small set of fixed vectors.

Contrastive Learning

- How do we make sure that $h_L(x)$ is diverse?
- Intuition: Given two different images x, x’, $h_L(x)$ should be as different from $h_L(x')$ as possible.
- Contrastive loss: We want to minimize $|< h_L(x) , h_L(x') >|$ for two different images x, x’.
- Key observation: We don’t need the labels of these images x, x’! We can randomly sample them from the internet.

Contrastive Learning

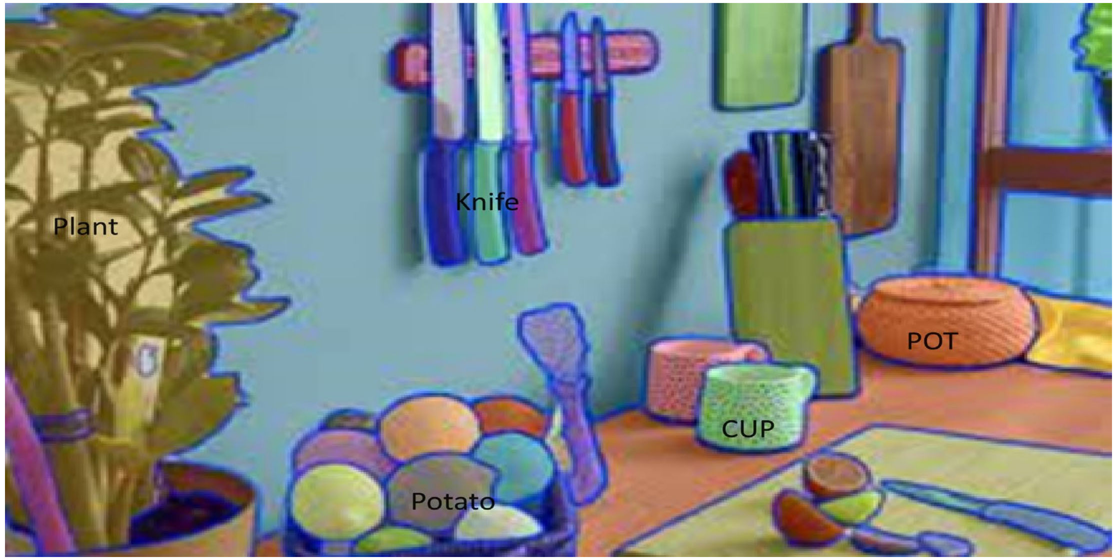
- Contrastive Loss:
- Minimize $E_{x,x'} \exp(< \frac{h_L(x)}{||h_L(x)||_2}, \frac{h_L(x')}{||h_L(x')||_2} > /\tau)$
- Theorem: At the minimizer, $\frac{h_L(x)}{||h_L(x)||_2}$ is a uniform distribution over the sphere.

Neural Network for Image Segmentation

- Another important application is image segmentation.
- We want to locate each object in the image.

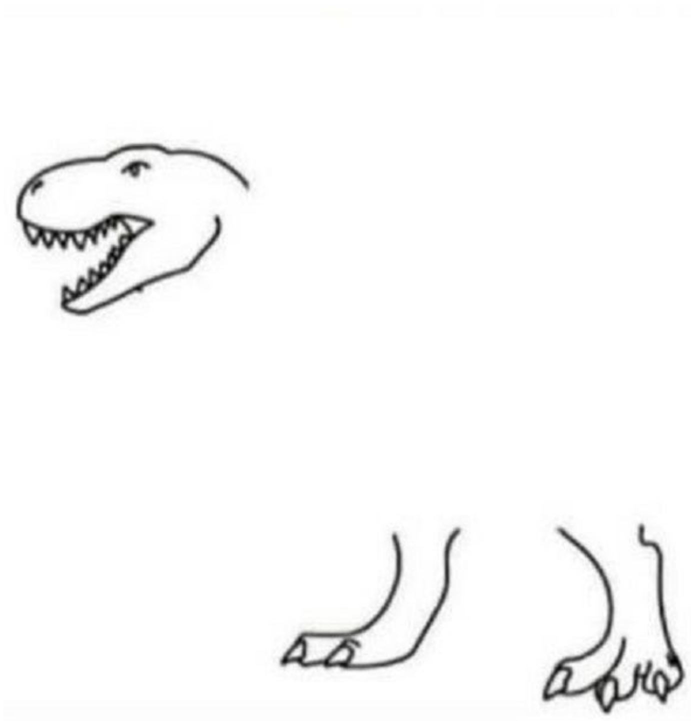
Neural Network for Image Segmentation

- Using a fully convolutional network (FCN).
- Every layer is a convolution, with no MLP/linear layer on top.
- Input is an image, and output is another image.
- Classify each pixel based on its segment.



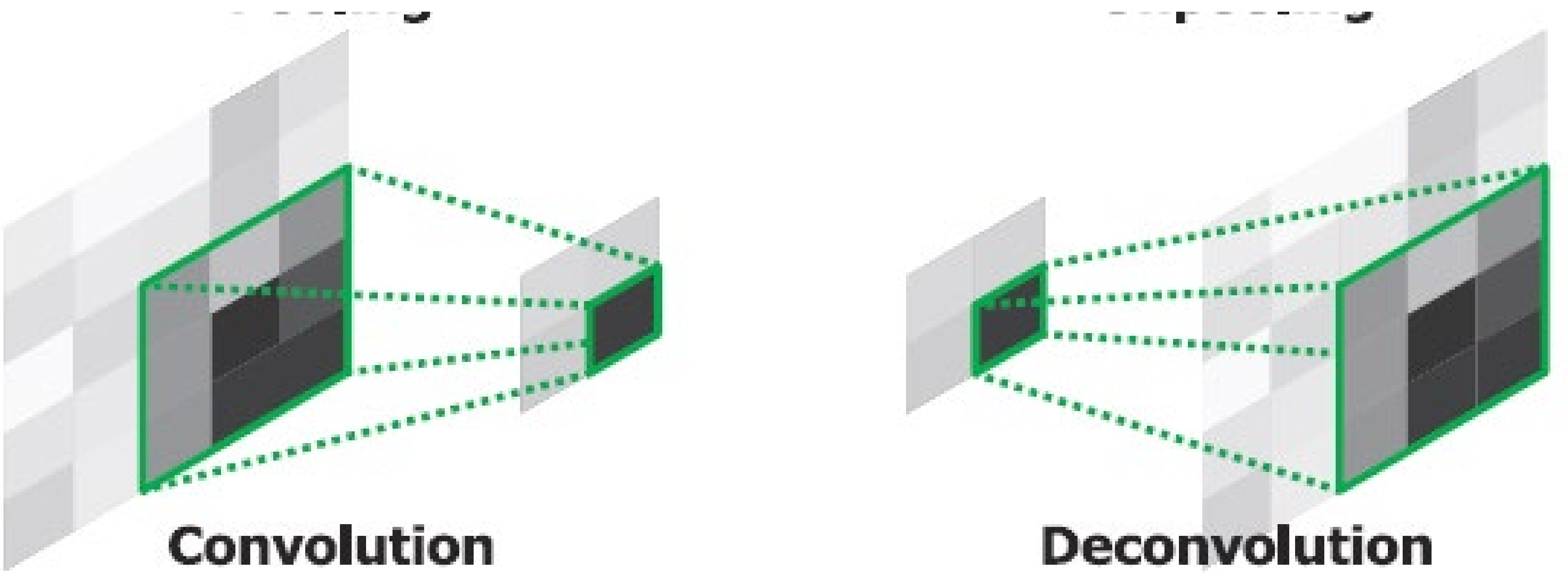
Neural Network for image completion

Given a cropped image, can we use a neural network to complete the cropped part?



Neural Network for image completion

- The most common way to use a neural network for image completion is to use a convolution network + a deconvolution network.
- H(x) = Decovolution ∘ Covolution(x)



Deconvolution operation

- Also known as ConvolutionTranspose layer.
- A Deconvolution operation with stride = s, is defined as:
- On input $x, size = d \times d \times C$
- First, define a new vector x’ of size $\approx ds$
  - $x'[is, js] = x[i, j]$, other entries are zero.
  - Padding: Pad zeros to x’.
- Then apply standard convolution on (padded) x', with stride 1.