Deep Learning based Image Similarity for Flipkart

Krishnendu Chaudhury, Sujay Narumanchi, Ananya H, Devashish Shankar

Flipkart Imaging Sciences
Bangalore

December 9, 2015

Problem Statement

- Input: one or more query images
- Output: top k visually similar catalog images for all input images
- Note:
  - For *v*0, query images are from catalog only
  - Catalog images are easier to deal with (uniform background)
  - ETA for wild images - mid 2016
- Pretty much all approaches *map* the image to one or more Descriptor Vector, aka Feature Vector (FV), aka Embedding
- (Dis)Similarity between a pair of images $\Longrightarrow$ Euclidean Distance or Cosine Similarity between FVs

Approach 1 - Traditional Computer Vision

- **Color histogram**
- **GaborJets** for pattern recognition / description: A gabor filter responds to patterns of alternating light and dark at a specific orientation angle, sharpness and spacing.
  GaborJet $\Longrightarrow$ concatenation of Gabor filter responses over a range of angles, sharpness and spacings
  1. Collect GaborJets from various salient points on the image
  2. Use GJs as index terms in a bag of words model for document search
  3. Combine with color histogram for a composite color + pattern similarity
- Maximally Stable Extremal Regions (**MSER**):
  1. Segment the image into similar colored blobs via MSER
  2. Describe individual blobs with a color histogram and shape descriptor (e.g., moments)
  3. Use blob descriptors as index terms in a bag of words model for document search
- Problem: Hard to filter out *background*, *skin* (face, hands) and *hair* of models / mannequins

Detour - Deep Learning Redux

All decision making	$\vec{x}\,(input) \xrightarrow{f(\vec{x})} y\,(output)$
e.g., face detection	$input\_image \xrightarrow{f(\vec{x})} face/no\_face$

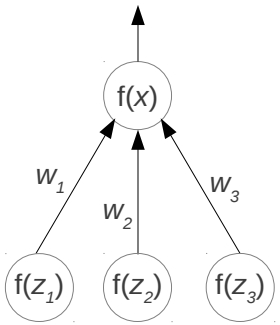
- Problem: we don't know $f(\vec{x})$
- solution 1: Model it as $f(\vec{x}) = \sum_i w_i x_i$. Too simple, not expressive enough.

Deep Learning Redux (contd.)

neuron and multi-layer sigmoid diagram

Neural networks

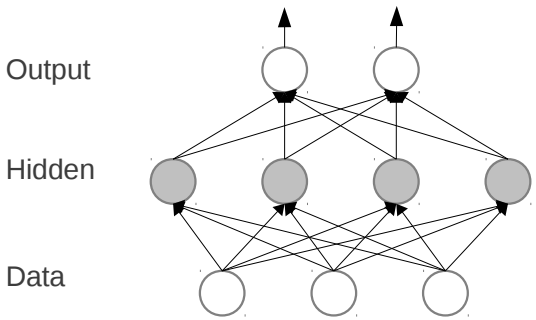
- A neuron



$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

x is called the total input to the neuron, and f(x) is its output

- A neural network



A neural network computes a differentiable function of its input. For example, ours computes: p(label | an input image)

Detour - Deep Learning Redux contd.

All decision making	$\vec{x}\,(input) \xrightarrow{f(\vec{x})} y\,(output)$
e.g., face detection	$input\_image \xrightarrow{f(\vec{x})} face/no\_face$

- Problem: we don't know $f\,(\vec{x})$
- solution 1: Model it as $f\,(\vec{x}) = \sum_i w_i x_i$. Too simple, not expressive enough.
- solution 2 (sigmoid neuron): Add non-linearity. $n\,(\vec{x}) = s\,(\sum_i w_i x_i)$ where $s\,(a) = \frac{1}{1+e^{-a}}$ is sigmoid basis function. Still not expressive enough.

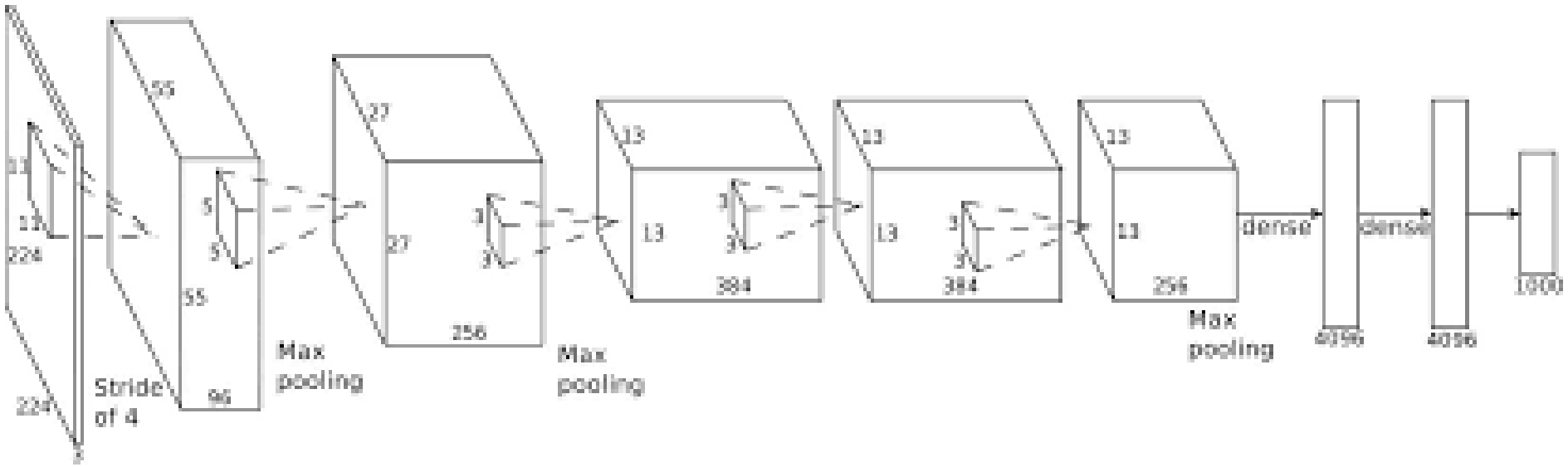
Detour - Deep Learning Redux

All decision making	$\vec{x}\,(input) \xrightarrow{f(\vec{x})} y\,(output)$
e.g., face detection	$input\_image \xrightarrow{f(\vec{x})} face/no\_face$

- Problem: we don't know $f(\vec{x})$
- solution 1: Model it as $f(\vec{x}) = \sum_i w_i x_i$. Too simple, not expressive enough.
- solution 2 (sigmoid neuron): Add non-linearity. $n(\vec{x}) = s\left(\sum_i w_i x_i\right)$ where $s(a) = \frac{1}{1+e^{-a}}$ is sigmoid basis function. Still not expressive enough.
- solution 3: Multi-layered sigmoid neurons. Each neuron emits sigmoid of linear combination of previous layer outputs

Deep Learning Redux (contd.)

Convolutional Neural Networks (CNNs)



Approach 2 - Embeddings from Object Recognition Deep CNNs

- Use final layer of Deep Convolutional Neural Networks for Object Recognition as Feature Vector
- Cons
  - Object recognition networks learn category level similarity - i.e., focus on features common to all t-shirts
  - Object recognition networks basically learn to ignore fine grained details (e.g, spacing and orientation of stripes, color shades etc)
  - Our similarity measure needs category level features as well as finer details
- We tried using both Alexnet and GoogLeNet - works but quality is not great

Approach 3 - Siamese Networks

basics

- Network Input: Image pair. Let corresponding Feature Vectors be $\vec{p}$ and $\vec{q}$.
- Ground Truth: $Y = 1$ (images are not similar) or $Y = 0$ (images are similar)
- Training: Both images of input pair processed by Alexnet with shared weights.
- Contrastive Loss Function:

$$L = (1 - Y)\frac{1}{2}\{D(\vec{p}, \vec{q})\}^2 + (Y)\frac{1}{2}\{max(0, m - D(\vec{p}, \vec{q}))\}^2$$

Approach 3 - Siamese Networks (contd.)

diagram

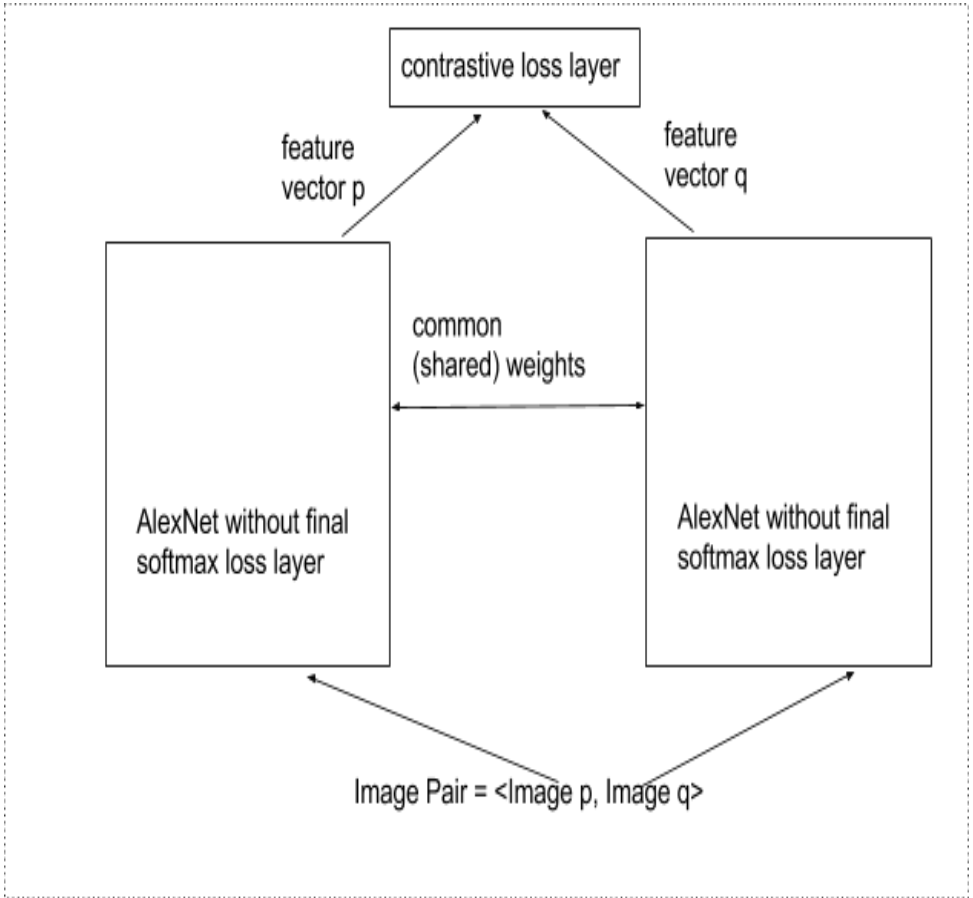


Figure: Siamese Network Basic Architecture

Approach 3 - Siamese Networks (contd.)

Cons

- Binary (yes / no) decision on whether two images are similar is very subjective. Ground Truth will be contradictory and non-uniform
- Final layer of Alexnet is a compact / abstract description of the image
- Loses details $\implies$ loss of quality in similarity

Our Approach - Triplet Networks

Training Data

- Each training data element is a triplet $\langle q, p, n \rangle$ of 3 images
  - $q \implies$ query image
  - $p \implies$ positive image
  - $n \implies$ negative image
- Image p should be *more* visually similar to image q compared to image n
- Trainers only label *relative* similarity
- Much more well defined and less ambiguous than *absolute* similarity required by Siamese networks
- Trainers typically label consistently $\implies$ no need of cross trainer normalization

Our Approach - Triplet Networks

Basic Training Architecture

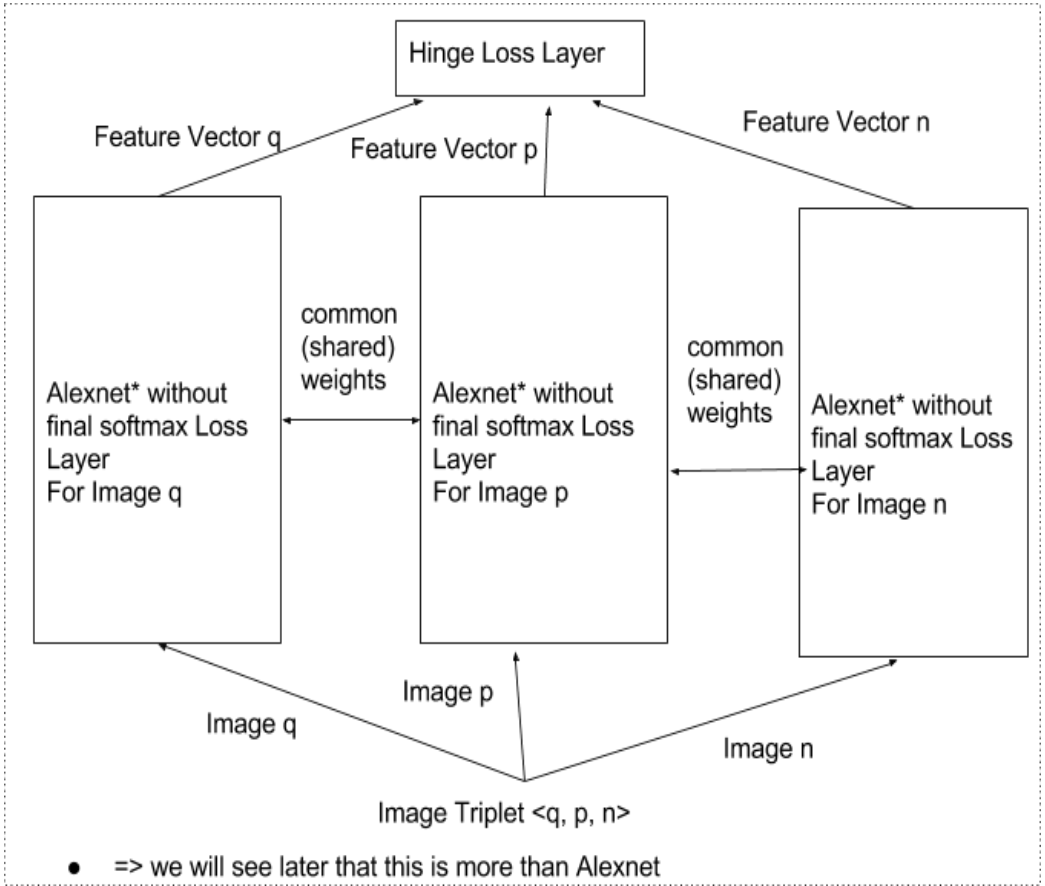


Figure: Our Deep CNN Triplet Network - Basic Training Architecture

Our Approach - Triplet Networks

Hinge Loss

- $L = max\left(0, g + D\left(\vec{q}, \vec{p}\right) - D\left(\vec{q}, \vec{n}\right)\right)$
- $g \implies$ gap parameter. Larger g pushes similar and dissimilar images further apart
- If $g = 0$, given a triplet $\langle q, p, n \rangle$
  - Case 1. Network does the right thing: Then $D\left(\vec{q}, \vec{p}\right) - D\left(\vec{q}, \vec{n}\right) < 0$ $\implies$ *hinge loss* $= 0$
  - Case 2. Network does the wrong thing: Then $D\left(\vec{q}, \vec{p}\right) - D\left(\vec{q}, \vec{n}\right) > 0$ $\implies$ *hinge loss* > 0
  - More the network deviates from correct behavior, higher the hinge loss
- If $g > 0$ $D\left(\vec{q}, \vec{n}\right)$ must exceed $D\left(\vec{q}, \vec{p}\right)$ by g to make loss 0

Our Approach - Triplet Networks

Alexnet* details

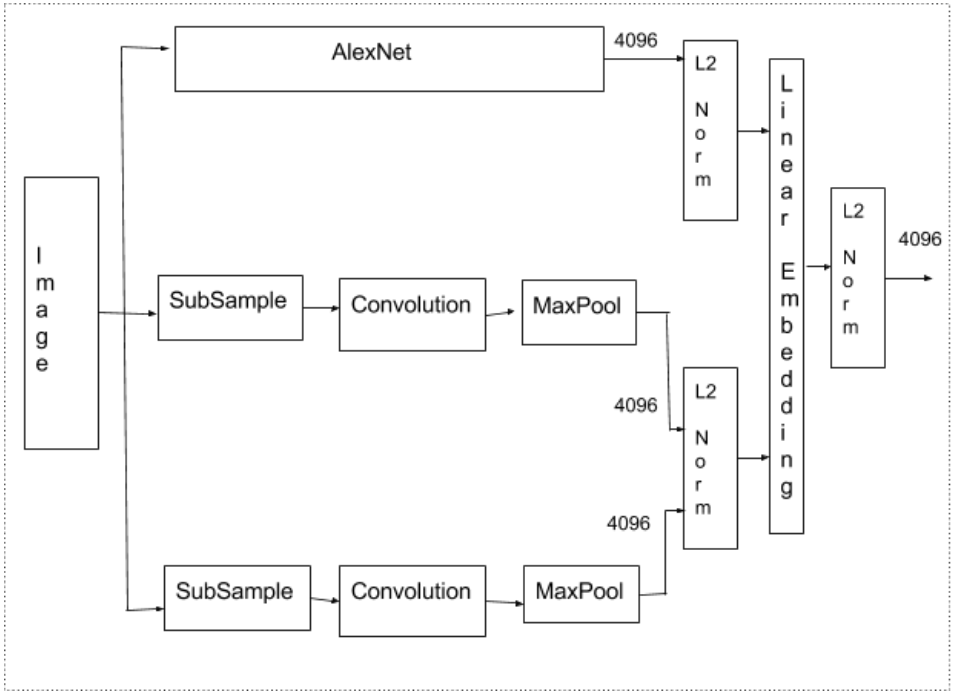


Figure: Combining Alexnet with Shallow Layers to combat loss of detail

Triplet Networks

Results

Query Image

Similar Images


0.0336006


0.0574383


0.0576119


0.0583131


0.0609345


0.0610314


0.0622081


0.0637216


0.0656046


0.0691144


0.0692294


0.0699181

Triplet Networks

Results

Query Image

Similar Images


0.0128639


0.0680414


0.0690463


0.0702025


0.0710221


0.0719871


0.0724716


0.0734705


0.0750466


0.0750608


0.0750711


0.0751758

Triplet Networks

Results

Similar Images


0.0320593


0.05885


0.0603077


0.060968


0.0632308


0.0635759


0.0640452


0.0640756


0.0660681


0.0664078


0.0671022


0.0677973

Triplet Networks

Results

Query Image

Similar Images


0.0209106


0.0560907


0.0571904


0.0611296


0.0615345


0.061628


0.0616518


0.061927


0.0620895


0.0621295


0.0627193


0.0648049

Triplet Networks

Results

Query Image

Similar Images


0.0267305


0.0452929


0.0462054


0.0505968


0.0514264


0.0536007


0.054428


0.054533


0.056055


0.0572254


0.0575072


0.0576526

Triplet Networks

Results

Query Image

Similar Images


0.0115705


0.0553371


0.062125


0.0639291


0.0650089


0.0674306


0.0682997


0.0693976


0.0703669


0.0707189


0.0729333


0.0738593

Triplet Networks

Results

Query Image

Similar Images


0.0163208


0.0667527


0.067453


0.0709344


0.0712831


0.0755576


0.0756447


0.0757563


0.0758546


0.0759709

0.0762427

0.0762884

Conclusion)

Possible Future Work

- Wild images
- Attribute Extraction - automatic product type and detail identification from images