Face blindness: 'I can't recognise my loved-ones'

By Max Evans
BBC News

6 March 2019

A stranger once waved at Boo James on a bus. She did not think any more of it - until it later emerged it was her mother.

She has a relatively rare condition called face blindness, which means she cannot recognise the faces of her family, friends, or even herself.

[...]

But how do people with prosopagnosia perceive faces? Those with the condition say it can be difficult to describe.

"I can see component parts of a face," Boo said. "I can see there's a nose, I can see there are eyes and a mouth and ears."

"But it's very difficult for my brain to hold them all together as the image of a face."
!!! Warning !!!

Learning jargon is always painful...
...even if the concepts behind the jargon are not hard.

So, let’s get used to it.

“In mathematics you don't understand things. You just get used to them.”

von Neumann (a joke)
Gartner Hype Cycle

- **Peak of Inflated Expectations**
- **Slope of Enlightenment**
- **Plateau of Productivity**

- **Innovation Trigger**
- **Trough of Disillusionment**
Launching in 2017, Rocket AI will be the global leader in neurologically-inspired applied machine learning. We build our systems around our patent-pending technology \textit{Temporally Recurrent Optimal Learning}\textsuperscript{™}

\textit{We Are Hiring}

launch@rocketai.org
Rocket AI

- Launch party @ NIPS 2016 [now NeurIPS]

- Neural Information Processing Systems

- Academic conference

Markus Wulfmeier
December 8 at 5:15pm • Barcelona, Spain • 🌐

#rocketai's launch party at #nips2016 clearly the best. Including the police involvement.
Rocket AI

Andréj Karpathy @karpathy · Dec 9
Best party of #nips2016 award goes to #rocketai (rocket.ai.org). Definitely a company to watch closely.

Karl Moritz Hermann @karlmoritz · Dec 9
One day we will look back and realise that the #rocketai launch was the day when things in our field changed forever.

Ian Goodfellow @goodfellow_ian · Dec 11
#rocketai definitely has the most popular Jacobian-Optimized Kernel Expansion of NIPS 2016
Rocket AI

Metrics for the Rocket AI launch party

Email RSVPs to party: 316
People who emailed in their resume: 46
Large name brand funds who contacted us about investing: 5
Media: Twitter, Facebook, HackerNews, Reddit, Quora, Medium etc
Time Planning: < 8 hours
Money Spent: $79 on the domain, $417 on alcohol and snacks + (police fine)
For reference, NIPS sponsorship starts at $10k.

Estimated value of Rocket AI: in the tens of millions.
Launching in 2017, Rocket AI will be the global leader in neurologically-inspired applied machine learning. We build our systems around our patent-pending technology *Temporally Recurrent Optimal Learning™*.

*We Are Hiring*

*launch@rocketai.org*
So far...

PASCAL VOC = ~75% 20-class
ImageNet = ~75% 1000-class, top 5

Human brains used intuition and understanding of how we think vision works to develop computer vision systems, and it’s pretty good.
Image formation (+database+labels)

Filtering (gradients/transforms)

Feature points (saliency+description)

Dictionary building (compression/quantization)

Classifier (decision making)

Captured+manual.

Hand designed.

Hand designed.

Learned.

Recognition: Classification Object Detection Segmentation
Well, what do we have?

Best performing visions systems have commonality:

Hand designed features
  • Gradients + non-linear operations (exponentiation, clamping, binning)
  • Features in combination (parts-based models)
  • Multi-scale representations

Machine learning from databases

Linear classifiers (SVM)
  • Some non-linear kernel tricks
But it’s still not that good...

PASCAL VOC = ~75% 20-class
ImageNet = ~75% 1000-class, top 5
ImageNet (human) = ~95%

Problems:
- Lossy features
- Lossy quantization
- ‘Imperfect’ classifier
But it’s still not that good...

• PASCAL VOC = ~75%
• ImageNet = ~75%; human performance = ~95%

How to solve?

• Features: More principled modeling? We know why the world looks (it’s physics!); Let’s build better physically-meaningful models.
• Quantization: More data and more compute? It’s just an interpolation problem; let’s represent the space with fewer data approximations.
• Classifier: ...
The limits of learning?
Previous claim:

*It is more important to have more or better labeled data than to use a different supervised learning technique.*

“The Unreasonable Effectiveness of Data” - Norvig
No free lunch theorem

Hume (c.1739):
“Even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience.”

-> Learning beyond our experience is impossible.
No free lunch theorem for ML

Wolpert (1996):

‘No free lunch’ for supervised learning:

“In a noise-free scenario where the loss function is the misclassification rate, if one is interested in off-training-set error, then there are no *a priori* distinctions between learning algorithms.”

-> Averaged over all possible datasets, no learning algorithm is better than any other.
OK, well, let’s give up. Class over.

No, no, no!

We can build a classifier which better matches the characteristics of the problem!
But...didn’t we just do that?

- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

We used intuition and understanding of how we think vision works, but it still has limitations.

Why?
Linear spaces - separability

• + kernel trick to transform space.

Linearly separable data + linear classifier = good.
Non-linear spaces - separability

Take XOR – exclusive OR
E.G., human face has two eyes XOR sunglasses

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$Y = X_1 \oplus X_2$
Non-linear spaces - separability

Linear functions are insufficient on their own.

\[
\begin{array}{ccc}
X_1 & X_2 & Y \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

\[Y = X_1 \oplus X_2\]
Curse of Dimensionality

Every feature that we add requires us to learn the useful regions in a much larger volume.

\[ d \text{ binary variables} = O(2^d) \text{ combinations} \]
Curse of Dimensionality

*Not all regions* of this high-dimensional space are meaningful.

```plaintext
>> I = rand(256,256);
>> imshow(I);

@ 8bit = 256 values ^ 65,536
Local constancy / smoothness of feature space

All existing learning algorithms we have seen assume **smoothness** or **local constancy**.

- New example will be near existing examples
- Each region in feature space requires an example
- Cannot generalize beyond examples

Extreme example: k-NN classifier. The number of regions cannot be more than the number of examples.

How to try and represent this high-dimensional space in a way which maximizes generalization?
More specialization?

- PASCAL VOC = ~75%
- ImageNet = ~75%; human performance = ~95%

Is there a way to make our system better suited to the problem?
Image formation (+database+labels)  
Filtering (gradients/transforms)  
Feature points (saliency+description)  
Dictionary building (compression/quantization)  
Classifier (decision making)  

Captured+manual.  
Hand designed.  
Hand designed.  
Learned.  

Recognition:  Classification  Object Detection  Segmentation
 Wouldn’t it be great if we could...

- Image formation (+database+labels)
- Filtering (gradients/transforms)
- Feature points (saliency+description)
- Dictionary building (compression/quantization)
- Classifier (decision making)

Recognition: Classification Object Detection Segmentation

Captured+manual. Learned. Learned. Learned. End to end learning!
Goals

Build a classifier which is more powerful at representing complex functions and more suited to the learning problem.

What does this mean?

1. Assume that the underlying data generating function relies on a composition of factors in a hierarchy.

Dependencies between regions in feature space = factor composition
Example
Example

- Is there an eye in the top left?
- Is there an eyebrow?
- Are there eyelashes?
- Is there an iris?
Non-linear spaces - separability

*Composition* of linear functions can represent more complex functions.

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$Y = X_1 \oplus X_2$

Kawaguchi
Goals

Build a classifier which is more powerful at representing complex functions and more suited to the learning problem.

What does this mean?
1. Assume that the *underlying data generating function* relies on a composition of factors in a hierarchy.
2. Learn a feature representation specific to the dataset.

10k/100k + data points + factor composition = sophisticated representation.
Reminder: Viola Jones Face Detector

Combine *thousands* of ‘weak classifiers’

- Two-rectangle features
- Three-rectangle features
- Etc.

Learn how to combine in cascade with boosting
Viola Jones

Image formation (+database+labels)

Features (saliency+description)

Classifier (decision making)

Recognition: Object Detection

Captured+manual.

Specified space, but selected automatically.

Learned combination.
Neural Networks
Neural Networks

Basic building block for composition is a *perceptron* (Rosenblatt c.1960)
Linear classifier – vector of weights $w$ and a ‘bias’ $b$

$$w = (w_1, w_2, w_3)$$
$$b = 0.3$$

Output (binary)

\[
\text{output} = \begin{cases} 
0 & \text{if } w \cdot x + b \leq 0 \\
1 & \text{if } w \cdot x + b > 0 
\end{cases}
\]

$$w \cdot x = \sum_j w_j x_j.$$
Binary classifying an image

Each pixel of the image would be an input.
So, for a 28 x 28 image, we vectorize:

\[ \mathbf{x} = 1 \times 784 \]

\( \mathbf{w} \) is a vector of weights for each pixel, 784 x 1
\( \mathbf{b} \) is a scalar bias per perceptron

Result = \( \mathbf{xw} + \mathbf{b} \)  \( \rightarrow (1\times784) \times (784\times1) + \mathbf{b} = (1\times1)+\mathbf{b} \)
Neural Networks - multiclass

Add more perceptrons

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
Multi-class classifying an image

Each pixel of the image would be an input. So, for a 28 x 28 image, we vectorize.

\[ \mathbf{x} = 1 \times 784 \]

\( \mathbf{W} \) is a matrix of weights for each pixel/each perceptron

\[ \mathbf{W} = 10 \times 784 \text{ (10-class classification)} \]

\( \mathbf{b} \) is a bias *per perceptron* (vector of biases); (1 x 10)

Result = \( \mathbf{xW} + \mathbf{b} \)  
\[ \rightarrow (1 \times 784) \times (784 \times 10) + \mathbf{b} \]  
\[ \rightarrow (1 \times 10) + (1 \times 10) = \text{output vector} \]
Bias convenience

Let’s turn this operation into a multiplication only:

- Create a ‘fake’ feature with value 1 to represent the bias
- Add an extra weight that can vary

\[ \mathbf{w} = (b, w_1, w_2, w_3) \]

Output (binary)

\[
\begin{align*}
\text{output} &= \begin{cases} 
0 & \text{if } w \cdot x \leq 0 \\
1 & \text{if } w \cdot x > 0
\end{cases} \\
w \cdot x & \equiv \sum_j w_j x_j
\end{align*}
\]
Composition

Attempt to represent complex functions as compositions of smaller functions.

Outputs from one perception are fed into inputs of another perceptron.
Composition

Sets of layers and the connections (weights) between them define the *network architecture*.
Composition

Layers that are in between the input and the output are called *hidden layers*, because we are going to *learn* their weights via an optimization process.
Composition

It's all just matrix multiplication!

GPUs -> special hardware for fast/large matrix multiplication.
Problem 1 with all linear functions

We have formed chains of linear functions.
We know that linear functions can be reduced
• $g = f(h(x))$

Our composition of functions is really just a single function : ( 
Problem 2 with all linear functions

Linear classifiers: small change in input can cause large change in binary output
= problem for composition of functions

Activation function
Problem 2 with all linear functions

Linear classifiers: small change in input can cause large change in binary output.

We want:

\[
\begin{align*}
    w + \Delta w & \quad \text{small change in any weight (or bias)} \\
    & \quad \text{causes a small change in the output} \\
\end{align*}
\]
Let’s introduce non-linearities

We’re going to introduce non-linear functions to transform the features.

\[
\sigma(w \cdot x + b)
\]

\[
\sigma(z) \equiv \frac{1}{1 + e^{-z}}.
\]
Universality

A single-layer of perceptrons can learn any univariate function:
  • So long as it is differentiable
  • To some approximation;
    More perceptrons = a better approximation

Visual proof (Michael Nielson):
Perceptron model

• Use is grounded in theory
  • Universal approximation theorem (Goodfellow 6.4.1)

• Can represent a NAND circuit, from which any binary function can be built by compositions of NANDs

• With enough parameters, it can approximate any function.
If a single-layer network can learn any function... 
...given enough parameters...

...then why do we go deeper?

Intuitively, composition is efficient because it allows *reuse*.

Empirically, deep networks do a better job than shallow networks at learning such hierarchies of knowledge.
Multi-layer perceptron (MLP)

• ...is a ‘fully connected’ neural network with non-linear activation functions.

• ‘Feed-forward’ neural network
Mark 1 Perceptron
c.1960
20x20 pixel camera feed
What is the relationship between SVMs and perceptrons?

SVMs attempt to learn the support vectors which maximize the margin between classes.
What is the relationship between SVMs and perceptrons?

SVMs attempt to learn the support vectors which maximize the margin between classes.

A perceptron does not.

Both of these perceptron classifiers are equivalent.

‘Perceptron of optimal stability’ is used in SVM:

Perceptron
+ optimal stability
+ kernel trick
= foundations of SVM
Does anyone pass along the weight without an activation function?

No – this is linear chaining.
Does anyone pass along the weight without an activation function?

No – this is linear chaining.
Are there other activation functions?

Yes, many.

As long as:
- Activation function $s(z)$ is well-defined as $z \to -\infty$ and $z \to \infty$
- These limits are different

Then *we can make a step!* [Think visual proof]
It can be shown that it is universal for function approximation.
Activation functions:
Rectified Linear Unit

- ReLU \( f(x) = \max(0, x) \).
Rectified Linear Unit

**Question:** What do ReLU layers accomplish?

**Answer:** Piece-wise linear tiling: mapping is locally linear.
Goals

Build a classifier which is more powerful at representing complex functions \textit{and} more suited to the learning problem.

What does this mean?

1. Assume that the \textit{underlying data generating function} relies on a composition of factors.

2. Learn a feature representation that is specific to the dataset.
Supervised Learning

\[ \{(x^i, y^i), i = 1 \ldots P\} \text{ training dataset} \]

\( x^i \) i-th input training example

\( y^i \) i-th target label

\( P \) number of training examples

Goal: predict the target label of unseen inputs.
Supervised Learning: Examples

**Classification**
- Image of a puppy
- Output: "dog"

**Denoising**
- Noisy image of a pepper
- Output: Cleaned image with "2345"

**OCR**
- Image of text "2345"
- Output: "2345"
Supervised Deep Learning

Classification

Denoising

OCR

“dog”

“2345”
Alternative Graphical Representation

\[ h^k \xrightarrow{\text{max}(0, W^{k+1} h^k)} h^{k+1} \]

\[ h^k \xrightarrow{W^{k+1}} h^{k+1} \]
Neural Networks: example

\[ x \xrightarrow{\text{max}(0, W^1 x)} h^1 \xrightarrow{\text{max}(0, W^2 h^1)} h^2 \xrightarrow{W^3 h^2} o \]

\( x \) input

\( h^1 \) 1-st layer hidden units

\( h^2 \) 2-nd layer hidden units

\( o \) output

Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).
Why do we need many layers?

- A hierarchical structure is potentially more efficient because we can reuse intermediate computations.
- Different representations can be distributed across classes.

\[
[0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ \ldots ] \text{ truck feature}
\]

Exponentially more efficient than a 1-of-N representation (a la k-means)
Interpretation

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & \ldots \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & \ldots \\
\end{bmatrix}
\]

motorbike

truck
Interpretation

- prediction of class
- distributed representations
- feature sharing
- compositionality

Input image

Lee et al. “Convolutional DBN’s ...” ICML 2009
Interpretation

**Question:** What does a hidden unit do?

**Answer:** It can be thought of as a classifier or feature detector.

**Question:** How many layers? How many hidden units?

**Answer:** Cross-validation or hyper-parameter search methods are the answer. In general, the wider and the deeper the network the more complicated the mapping.

**Question:** How do I set the weight matrices?

**Answer:** Weight matrices and biases are learned. First, we need to define a measure of quality of the current mapping. Then, we need to define a procedure to adjust the parameters.
Neural Networks: example

\[ x \rightarrow \max(0, W^1 x) \rightarrow \max(0, W^2 h^1) \rightarrow W^3 h^2 \rightarrow o \]

- \( x \): input
- \( h^1 \): 1-st layer hidden units
- \( h^2 \): 2-nd layer hidden units
- \( o \): output

Example of a 2 hidden layer neural network (or 4 layer network, counting also input and output).
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
Images as input to neural networks
Images as input to neural networks

Example: 200x200 image
40K hidden units

~2B parameters!!!
Images as input to neural networks

Example: 200x200 image
40K hidden units

~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..
Motivation

• Sparse interactions – *receptive fields*
  • Assume that in an image, we care about ‘local neighborhoods’ only for a given neural network layer.
  • Composition of layers will expand local -> global.
Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

**Note:** This parameterization is good when input image is registered (e.g., face recognition).
Note: This parameterization is good when input image is registered (e.g., face recognition).

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

STATIONARITY? Statistics is similar at different locations
Motivation

• Sparse interactions – \textit{receptive fields}
  • Assume that in an image, we care about ‘local neighborhoods’ only for a given neural network layer.
  • Composition of layers will expand local -> global.

• Parameter sharing
  • ‘Tied weights’ – use same weights for more than one perceptron in the neural network.
  • Leads to \textit{equivariant representation}
    • If input changes (e.g., translates), then output changes similarly
Share the same parameters across different locations (assuming input is stationary):
Filtering reminder:
Correlation (rotated convolution)

\[ f[\cdot, \cdot] \]

\[ I[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

\[ h[m, n] = \sum_{k,l} f[k, l] I[m+k, n+l] \]
Convolutional Layer

Perceptron:  
\[
\text{output} = \begin{cases} 
0 & \text{if } w \cdot x + b \leq 0 \\
1 & \text{if } w \cdot x + b > 0 
\end{cases}
\]

\[w \cdot x = \sum_j w_j x_j\]

This is convolution!

Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels
Convolution

3x3 kernel

Shared weights

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]
Learn multiple filters.

Filter = ‘local’ perceptron. Also called kernel.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters
Interpretation

- prediction of class
  - distributed representations
  - feature sharing
  - compositionality

Input image

Lee et al. “Convolutional DBN's ...” ICML 2009
\[ h^n_j = \max (0, \sum_{k=1}^{K} h^{n-1}_k * w^n_{kj}) \]

- \( n = \) layer number
- \( K = \) kernel size
- \( j = \) # channels (input) or # filters (depth)

**Convolutional Layer**

[Diagram showing a convolutional layer with input feature map, kernel, and output feature map labeled.]
\[ h^n_j = \max(0, \sum_{k=1}^{K} h^{n-1}_k \ast w^n_{kj}) \]

output feature map

input feature map

kernel
Convolutional Layer

\[ h_j^n = \max(0, \sum_{k=1}^K h_{kj}^{n-1} \ast w_{kj}^n) \]

output feature map

input feature map

kernel

\[ h_1^{n-1} \] \[ h_2^{n-1} \] \[ h_3^{n-1} \] \[ h_1^n \] \[ h_2^n \]
Stride = 1
Stride = 1
Stride = 3
Stride = 3
Stride = 3
Stride = 3
Pooling Layer

Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?
By pooling responses at different locations, we gain robustness to the exact spatial location of image features.
Pooling is similar to downsampling...except sometimes we don’t want to blur, as other functions might be better for classification.
Pooling Layer: Receptive Field Size
Pooling Layer: Examples

Max-pooling:

\[ h^n_j(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y}) \]

Average-pooling:

\[ h^n_j(x, y) = \frac{1}{K} \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h^{n-1}_j(\bar{x}, \bar{y}) \]
Max pooling

Single depth slice

X

Y

1 0 2 3
4 6 6 8
3 1 1 0
1 2 2 4

6
8

3
4
Pooling Layer: Examples

Max-pooling:

\[ h_j^n(x, y) = \max_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y}) \]

Average-pooling:

\[ h_j^n(x, y) = \frac{1}{K} \sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y}) \]

L2-pooling:

\[ h_j^n(x, y) = \sqrt{\sum_{\bar{x} \in N(x), \bar{y} \in N(y)} h_j^{n-1}(\bar{x}, \bar{y})^2} \]

L2-pooling over features:

\[ h_j^n(x, y) = \sqrt{\sum_{k \in N(j)} h_k^{n-1}(x, y)^2} \]
If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$
If convolutional filters have size $K \times K$ and stride 1, and pooling layer has pools of size $P \times P$, then each unit in the pooling layer depends upon a patch (at the input of the preceding conv. layer) of size: $(P+K-1) \times (P+K-1)$
Local Contrast Normalization
Local Contrast Normalization

We want the same response.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

\(N(x,y) = \) model pixel values in window as a normal distribution

\(m = \) mean
\(\sigma = \) variance

**Note:** computational cost is negligible w.r.t. conv. layer.
Local Contrast Normalization

\[ h^{i+1}(x, y) = \frac{h^i(x, y) - m^i(N(x, y))}{\sigma^i(N(x, y))} \]

Performed also across features and in the higher layers.

Effects:
- improves invariance
- improves optimization
- increases sparsity

**Note:** computational cost is negligible w.r.t. conv. layer.
ConvNets: Typical Stage

One stage (zoom)

- Convol.
- LCN
- Pooling

Filter Bank
Rectification + Contrast Normalization
Pooling

courtesy of K. Kavukcuoglu
ConvNets: Typical Architecture

One stage (zoom)

Convol. \rightarrow LCN \rightarrow Pooling

Whole system

Input Image \rightarrow 1^{st} stage \rightarrow 2^{nd} stage \rightarrow 3^{rd} stage \rightarrow Fully Conn. Layers \rightarrow Class Labels
ConvNets: Typical Architecture

Whole system

Input Image → 1st stage → 2nd stage → 3rd stage → Fully Conn. Layers → Class Labels

Conceptually similar to:

SIFT → K-Means → Pyramid Pooling → SVM
Lazebnik et al. “...Spatial Pyramid Matching...” CVPR 2006

SIFT → Fisher Vect. → Pooling → SVM
Yann LeCun’s MNIST CNN architecture
Demo

http://scs.ryerson.ca/~aharley/vis/conv/

Thanks to Adam Harley for making this.

More here: http://scs.ryerson.ca/~aharley/vis
Convolutions: More detail

32x32x3 image

- Height: 32
- Width: 32
- Depth: 3
Convolutions: More detail

32x32x3 image

5x5x3 filter
Convolutions: More detail

Convolution Layer

32x32x3 image
5x5x3 filter

convolve (slide) over all spatial locations

activation map
Convolutions: More detail

For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Convolutions: More detail

- **CONV, ReLU**
  - e.g. 6 5x5x3 filters
  - e.g. 10 5x5x6 filters

Andrej Karpathy
Convolutions: More detail

Output size: \((N - F) / \text{stride} + 1\)
Our connectomics diagram

Auto-generated from network declaration by nolearn (for Lasagne / Theano)

Input
75x75x4
Reading architecture diagrams

Layers
- Kernel sizes
- Strides
- # channels
- # kernels
- Max pooling
AlexNet diagram (simplified)

Input size
227 x 227 x 3

Conv 1
11 x 11 x 3
Stride 4
96 filters

Conv 2
5 x 5 x 96
Stride 1
256 filters

Conv 3
3 x 3 x 256
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
256 filters

[Krizhevsky et al. 2012]
Outline

- Supervised Neural Networks
- Convolutional Neural Networks
- Examples
- Tips
CONV NETS: EXAMPLES

- OCR / House number & Traffic sign classification

Ciresan et al. “MCDNN for image classification” CVPR 2012
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014
CONV NETS: EXAMPLES

- Scene Parsing

Farabet et al. “Learning hierarchical features for scene labeling” PAMI 2013
Pinheiro et al. “Recurrent CNN for scene parsing” arxiv 2013
CONV NETS: EXAMPLES

- Segmentation 3D volumetric images

Ciresan et al. “DNN segment neuronal membranes...” NIPS 2012
Turaga et al. “Maximin learning of image segmentation” NIPS 2009
CONV NETS: EXAMPLES

- Object detection

Szegedy et al. “DNN for object detection” NIPS 2013
CONV NETS: EXAMPLES

- Face Verification & Identification

Dataset: ImageNet 2012

Deng et al. “Imagenet: a large scale hierarchical image database” CVPR 2009
Architecture for Classification

Total nr. params: 60M

4M
LINEAR 4M

16M
FULLY CONNECTED 16M

37M
FULLY CONNECTED 37M

MAX POOLING

442K
CONV 74M

1.3M
CONV 224M

884K
CONV 149M

MAX POOLING

307K
LOCAL CONTRAST NORM 223M

MAX POOLING

35K
LOCAL CONTRAST NORM 105M

CONV

input

Total nr. flops: 832M
Results: ILSVRC 2012

TASK 1 - CLASSIFICATION

Error %

CNN  | SIFT+FV | SVM1 | SVM2 | NCM

35   | 30     | 25   | 20   | 15

TASK 2 - DETECTION

Error %

CNN  | DPM-SVM1 | DPM-SVM2

50   | 45       | 40

Krizhevsky et al. “ImageNet Classification with deep CNNs” NIPS 2012
Wait, why isn’t it called a correlation neural network?

It could be.
Deep learning libraries actually implement correlation.

Correlation relates to convolution via a 180deg rotation of the kernel. When we learn kernels, we could easily learn them flipped.

Associative property of convolution ends up not being important to our application, so we just ignore it.

[p.323, Goodfellow]
Phew!

Monday:
How to Train your Dragon Network

Project 4: Out Friday
- Questions
- Code part 1
Due 15th.
More ConvNet explanations

• https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/