UNIVERSITY OF OXFORD

Mathematical Institute

Visualising the filters and response, memory, in a CNN: deterministic multi-resolution frames and transfer learning

Theories of Deep Learning: C6.5, Video 12

Prof. Jared Tanner
Mathematical Institute
University of Oxford

Oxford
Mathematics

LeNET-5, an early Image processing DNN:

Network architectures often include fully connected and convolutional layers

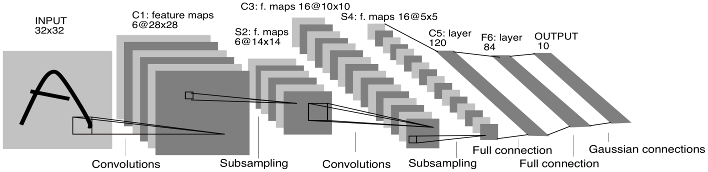


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

C1: conv. layer with 6 feature maps, 5 by 5 support, stride 1.

S2 (and S4): non-overlapping 2 by 2 blocks which equally sum values, mult by weight and add bias.

C3: conv. layer with 16 features, 5 by 5 support, partial connected.

C5: 120 features, 5 by 5 support, no stride; i.e. fully connected.

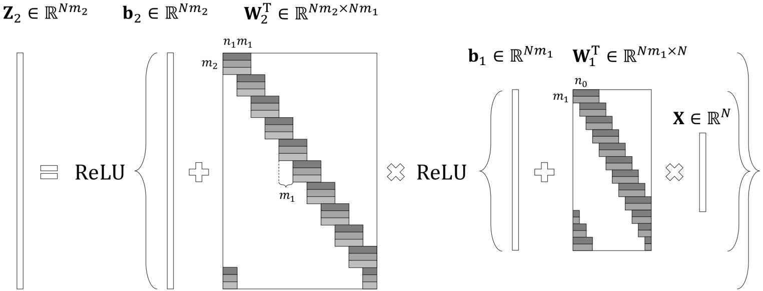
F6: fully connected, $W \in \mathbb{R}^{84\times 120}$.

`http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`

A simple two layer CNN (Papyan et al. 16')

Convolutional structure are the form of multi-resolution analysis

Consider a deep conv. net composed of two convolutional layers:


$\mathbf{Z}_2 \in \mathbb{R}^{Nm_2}$
$\mathbf{b}_2 \in \mathbb{R}^{Nm_2}$
$\mathbf{W}_2^{\mathrm{T}} \in \mathbb{R}^{Nm_2 \times Nm_1}$
$n_1 m_1$
m_2
m_1
$=$ ReLU
$+$
$\times$ ReLU
$\mathbf{b}_1 \in \mathbb{R}^{Nm_1}$
$\mathbf{W}_1^{\mathrm{T}} \in \mathbb{R}^{Nm_1 \times N}$
n_0
m_1
$+$
$\times$
$\mathbf{X} \in \mathbb{R}^{N}$

$$Z_2 = \sigma\left(b^{(2)} + (W^{(2)})^T \sigma\left(b^{(1)} + (W^{(1)})^T x\right)\right)$$

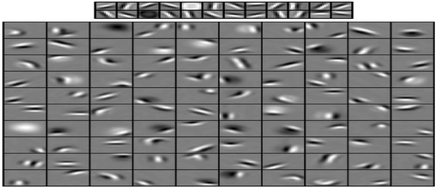
https://arxiv.org/pdf/1607.08194.pdf

Convolutional Deep Belief Networks (H. Lee et al. 11')

Localized Fourier, Wavelet, structure learned

We omit the details of this somewhat different architecture, which is stylistically similar to a deep CNN.

Figure 3. The first layer bases (top) and the second layer bases (bottom) learned from natural images. Each second layer basis (filter) was visualized as a weighted linear combination of the first layer bases.

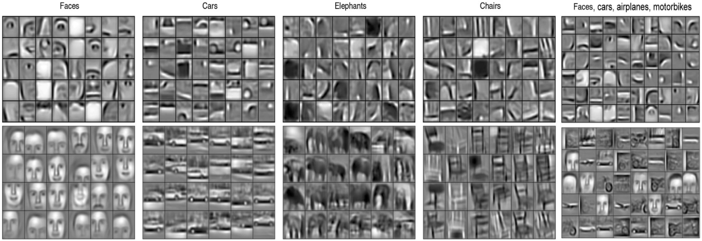


`http://www.cs.utoronto.ca/~rgrosse/cacm2011-cdbn.pdf`
Display of the convolutional masks in layers 1 and 2, trained from Kyoto natural image database.
`http://eizaburo-doi.github.io/kyoto_natim/`

Convolutional Deep Belief Networks (H. Lee et al. 11')

Learned / memorized complex structure from data classes

Figure 4. Columns 1–4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).



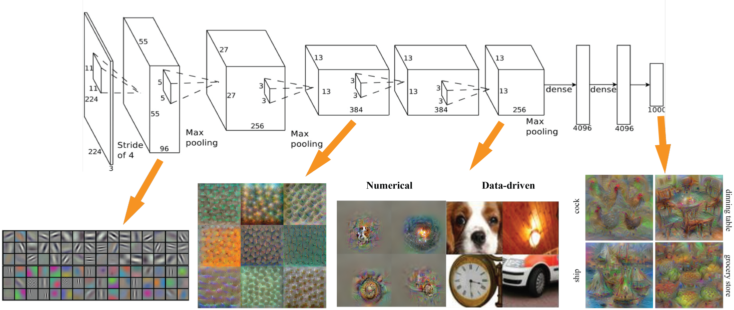
`http://eizaburo-doi.github.io/kyoto_natim/`

The third and fourth layers develop bases which represent features or objects, trained on CalTech 101 dataset.

`http://www.vision.caltech.edu/Image_Datasets/Caltech101/`

Deep CNN, AlexNet (Krizhevsky et al. 12')

Learned / memorized complex structure from data classes



Conv 1: Edge+Blob **Conv 3: Texture** **Conv 5: Object Parts** **Fc8: Object Classes**

Images are those that maximize specific activation responses.
Layer 1 are masks, subsequent layers are their linear combinations.

`http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`

Deep CNN, VGG (Mahendran et al. 16')

Learned / memorized complex structure from data classes

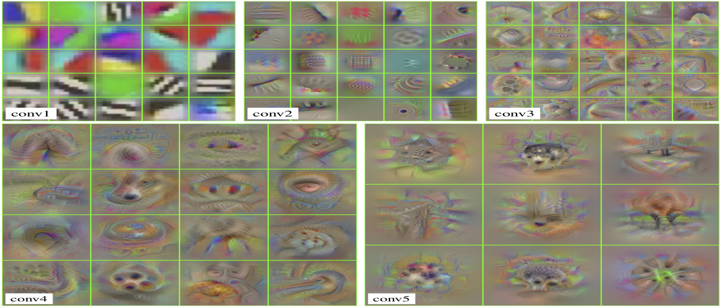


Figure 16: Activation maximization of the first filters of each convolutional layer in VGG-M.

Note, again we observe the same pattern, the initial filters are similar to Gabor/Wavelet filters and later layers are image components.

`https://arxiv.org/abs/1512.02017`

Summary: similarity and importance of initial layers

Importance of training initial layers to develop representation

We observe the initial layer of CNNs to be similar to one another, and to exhibit wavelet like representations. This is to be expected.

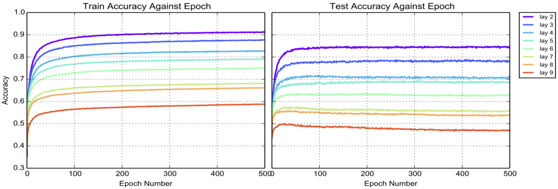


Figure 6: Demonstration of expressive power of remaining depth on MNIST. Here we plot train and test accuracy achieved by training exactly one layer of a fully connected neural net on MNIST. The different lines are generated by varying the hidden layer chosen to train. All other layers are kept frozen after random initialization.

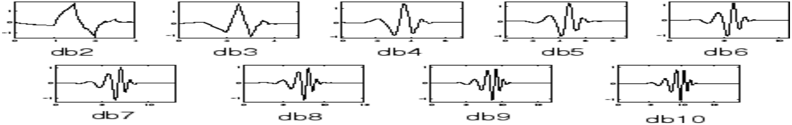
Accuracy of a *random network* is improved most by training earlier layers (Raghu 16').

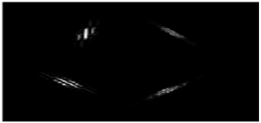
`https://arxiv.org/pdf/1611.08083.pdf`

Wavelet, curvelet, and contourlet: fixed representations

Known optimal representations for natural images

Applied and computational harmonic analysis community developed representations with optimal approximation properties for piecewise smooth functions.

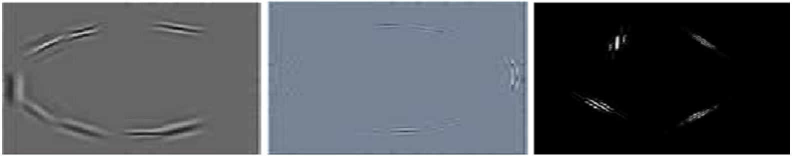




Most notable are the Daubechies wavelets and Curvelets/Contourlets pioneered by Candes and Donoho. While optimal, in a certain sense, for a specific class of functions, they can typically be improved upon for any particular data set.

Optimality of curvelets in 2D

Near optimality suggest a good initial CNN layer.



Theorem (Candes and Donoho 02')

Let f be a two dimensional function that is piecewise C^2 with a boundary that is also C^2. Let f_n^F, f_n^W, and f_n^C be the best approximation of f using n terms of the Fourier, Wavelet and Curvelet representation respectively. Then their approximation error satisfy $\|f - f_n^F\|_{L^2}^2 = \mathcal{O}(n^{-1/2})$, $\|f - f_n^W\|_{L^2}^2 = \mathcal{O}(n^{-1})$, and $\|f - f_n^C\|_{L^2}^2 = \mathcal{O}(n^{-2}\log^3(n))$; moreover, no fixed representation can have a rate exceeding $\mathcal{O}(n^{-2})$.

http://www.curvelet.org/papers/CurveEdges.pdf

Initial layers can start as representations for the data class

Transfer learning: training only the final classification layer

The first layer of a CNN can be initialized from a known representation for the data class. One can perform classification based on two layer net: layer 1: $h_2(x) = \sigma(W^{(1)}x + b^{(1)})$ where $W^{(1)}$ is a fixed transform of x to, say, the wavelet domain and $\sigma(\cdot)$ project to keep just the largest entries with hard or soft thresholding;

$$\sigma_{hard}(x;\tau) = \begin{cases} x & x > \tau \\ 0 & |x| \leq \tau \\ -x & x < -\tau \end{cases}, \quad \sigma_{soft}(x;\tau) = \begin{cases} x - \tau & x > \tau \\ 0 & |x| \leq \tau \\ -x + \tau & x < -\tau \end{cases}$$

layer2: $h_3 = \sigma(W^{(2)}h_2 + b^{(2)})$ with $W^{(2)}$ learned as the classifier based on the sparse codes h_2. However, h_2 does not build in invariance we would desire in classification, such as dilation, rotation, translation, etc... Depth remains important to generate these.