Convolutional Neural Nets

Pascal Fua
IC-CVLab
Fully Connected Layers

- The descriptive power of the net increases with the number of layers.
- In the case of a 1D signal, it is roughly proportional to $\prod_{n} W_n$ where $W_n$ represents the width of a layer.
A MxN image can be represented as an MN vector.
It can therefore be used as an input to an MLP.
However the neighborhood relationships are then lost.

→ This is not the best approach.
Image Specificities

- In a typical image, the values of neighboring pixels tend to be more highly correlated than those of distant ones.
- An image filter should be translation equivariant.

$\Rightarrow$ These two properties can be exploited to drastically reduce the number of weights required by CNNs using so-called convolutional layers.
1D Convolution in the Continuous Domain

\[ g \ast f(t) = \int_{\tau} g(t - \tau) f(\tau) d\tau \]
Example 1: Convolution with a Gaussian

- Each sample is replaced by a weighted average of its neighbors.
- This yields a smoothed version of the original signal.
Example 2: Convolution with the Derivative of a Gaussian

\[ \frac{\partial}{\partial x} (g \ast f) = \frac{\partial g}{\partial x} \ast f \]

- Convolving with the derivative of a gaussian is the same as smoothing first and then differentiating.
Discrete 1D Convolution

Input

1 4 -1 0 2 -2 1 3 3 1

Mask

1 2 0 -1
Discrete 1D Convolution

Input

\[
\begin{array}{cccccccccc}
1 & 4 & -1 & 0 & 2 & -2 & 1 & 3 & 3 & 1 \\
\end{array}
\]

\[ W \]

Output

\[
\begin{array}{c}
9 \\
\end{array}
\]

\[ W - w + 1 \]
Discrete 1D Convolution

Input

\[
\begin{array}{cccccc}
1 & 4 & -1 & 0 & 2 & -2 \\
\end{array}
\]

Output

\[
\begin{array}{cc}
9 & 0 \\
\end{array}
\]

The convolution operation is illustrated by sliding the filter \( w \) across the input array and computing the output as the dot product of the filter with a segment of the input. The output is computed as:

\[
W - w + 1
\]
Discrete 1D Convolution

Input

\[
\begin{array}{cccccccc}
1 & 4 & -1 & 0 & 2 & -2 & 1 & 3 & 3 & 1 \\
\end{array}
\]

\[ W \]

Output

\[
\begin{array}{cccc}
9 & 0 & 1 \\
\end{array}
\]

\[ W - w + 1 \]
Discrete 1D Convolution

Input

\[
\begin{array}{cccccccc}
1 & 4 & -1 & 0 & 2 & -2 & 1 & 3 & 3 & 1
\end{array}
\]

\[
W
\]

Output

\[
\begin{array}{cccc}
9 & 0 & 1 & 3
\end{array}
\]

\[
W - w + 1
\]
1D Convolution

Input

| 1 | 4 | -1 | 0 | 2 | -2 | 1 | 3 | 3 | 1 |

\[ W - w + 1 \]

Output

| 9 | 0 | 1 | 3 | -5 |

\[ W - w + 1 \]

F. Fleuret. EE-559 – Deep learning
Discrete 1D Convolution

Input

\[
\begin{bmatrix}
1 & 4 & -1 & 0 & 2 & -2 & 1 & 3 & 3 & 1
\end{bmatrix}
\]

\[W\]

Output

\[
\begin{bmatrix}
9 & 0 & 1 & 3 & -5 & -3
\end{bmatrix}
\]

\[W - w + 1\]
Discrete 1D Convolution

Input

\[
\begin{array}{cccccc}
1 & 4 & -1 & 0 & 2 & -2 \\
\end{array}
\]

Output

\[
\begin{array}{cccccc}
9 & 0 & 1 & 3 & -5 & -3 & 6 \\
\end{array}
\]
Discrete 1D Convolution

\[ m \ast f(x) = \sum_{i=0}^{w} m(i)f(x - i) \]
Discrete 2D Convolution

Convolution mask \( m \), also known as a *kernel*.

\[
\begin{bmatrix}
m_{11} & \cdots & m_{1w} \\
\vdots & \ddots & \vdots \\
m_{w1} & \cdots & m_{ww}
\end{bmatrix}
\]

\[
m^{* *}f(x, y) = \sum_{i=0}^{w} \sum_{j=0}^{w} m(i, j)f(x - i, y - j)
\]
This approximates an x derivative.
2D Convolutional Layer

- The same weights $w_{x,y}$ are used to compute all the activations.
- There are far fewer weights that in a fully connected layers.
- The neighborhood relationships are explicitly used.
In practice, one uses several filters, that is, sets of weights $w_{x,y}$, to compute several convolved versions of the input. These are known as feature maps.
Derivative Filters

Derivatives

Learned filters
Pooling Layer

- Reduces the number of inputs by replacing all activations in a neighborhood by a single one.
- Can be thought as asking if a particular feature is present in that neighborhood while ignoring the exact location.
Adding the Pooling Layers

The output size is reduced by the pooling layers.
Pooling Example

Max-pooling:
\[
    h_i[u, v] = \max\{ h_{i-1}[2u, 2v], h_{i-1}[2u, 2v + 1], h_{i-1}[2u + 1, 2v], h_{i-1}[2u + 1, 2v + 1] \}
\]
Adding a Fully Connected Layer

- Each neuron in the final fully connected layer is connected to all neurons in the preceding one.
- Deep architecture with many parameters to learn but still far fewer than an equivalent multilayer perceptron.
class ConvNet(nn.Module):

    def __init__(self, nChannel=10, nHidden=50):
        self.cv1 = nn.Conv2d(1, nChannel, kernel_size=5)
        self.cv2 = nn.Conv2d(nChannel, 20, kernel_size=5)
        self.fc1 = nn.Linear(320, nHidden)
        self.fc2 = nn.Linear(nHidden, 10)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.cv1(x), 2))
        x = F.relu(F.max_pool2d(self.cv2(x), 2))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
Without Max Pooling

Springenberg et al., ICLR’15
class ConvNet(nn.Module):
    def __init__(self, nChannel=10, nHidden=50):
        self.cv1 = nn.Conv2d(1, nChannel, kernel_size=5, stride=2)
        self.cv2 = nn.Conv2d(nChannel, 20, kernel_size=5, stride=2)
        self.fc1 = nn.Linear(320, nHidden)
        self.fc2 = nn.Linear(nHidden, 10)

    def forward(self, x):
        x = F.relu(self.cv1(x))
        x = F.relu(self.cv2(x))
        x = x.view(-1, 320)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
The network takes as input 28x28 images represented as 784D vectors.
The output is a 10D vector giving the probability of the image representing any of the 10 digits.
There are 50’000 training pairs of images and the corresponding label, 10’000 validation pairs, and 5’000 testing pairs.
Lenet (1989-1999)
\[ h_1 = [g(f_{1,1} \ast x), \ldots, g(f_{1,m} \ast x)] \]
\[ h_2 = \text{pooling}(h_1) \]
\[ h_3 = [g(f_{3,1} \ast h_2), \ldots, g(f_{3,n} \ast h_2)] \]
\[ h_4 = \text{pooling}(h_3) \]
\[ h'_4 = \text{Vec}(h_4) \]
\[ h_5 = g(W_5 h'_4 + b_5) \]
\[ o = W_6 h_5 + b_6 \]
Given the appropriate architecture, the CNN outperforms the other approaches, whereas the MLP did not.
Lenet5 (1992)

- Worked beautifully on MNIST.
- Very few people believed it would scale up.
Task: Image classification
Training images: Large Scale Visual Recognition Challenge 2010
Training time: 2 weeks on 2 GPUs

Major Breakthrough: Training large networks has now been shown to be practical!!
AlexNet Results

- At the 2012 ImageNet Large Scale Visual Recognition Challenge, AlexNet achieved a top-5 error of 15.3%, more than 10.8% lower than the runner up.
- Since 2015, networks outperform humans on this task.

Krizhevsky, NIPS’12
Feature Maps

• Some of the convolutional masks are very similar to oriented Gaussian or Gabor filters.
• The trained neural nets compute oriented derivatives, which the brain is also **believed** to do.
VGG19, 3 weeks of training.

“It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture.”
Hand Pose Estimation (2015)

Input: Depth image.

Output: 3D pose vector.

Oberweger et al., ICCV’15
In general, the more ResNet layers, the better the results.
Image Classification Taxonomy

- LSTM (Hochreiter and Schmidhuber, 1997)
  - No recurrence

- LeNet5 (LeCun et al., 1989)
  - Deep hierarchical CNN (Ciresan et al., 2012)
  - Bigger + GPU

- Highway Net (Srivastava et al., 2015)
  - Overfeat (Sermanet et al., 2013)
  - Bigger + small filters

- VGG (Simonyan and Zisserman, 2014)
  - Fully convolutional

- AlexNet (Krizhevsky et al., 2012)
  - MLPConv

- ResNet (He et al., 2015)
  - No gating
  - Inception modules
  - Batch Normalization

- Wide ResNet (Zagoruyko and Komodakis, 2016)
  - Dense pass-through
  - Aggregated channels

- DenseNet (Huang et al., 2016)
  - Wider

- ResNeXt (Xie et al., 2016)
  - Inception-ResNet (Szegedy et al., 2016)
  - Bigger + ReLU + dropout

- GoogLeNet (Szegedy et al., 2015)
  - Net in Net (Lin et al., 2013)
  - Inception-ResNet

- BN-Inception (Ioffe and Szegedy, 2015)
  - Batch Normalization

- BN-Inception (Szegedy et al., 2016)
  - ResNeXt (Xie et al., 2016)
  - Wide ResNet (Zagoruyko and Komodakis, 2016)
Recurrent Auto Encoder

\[ q(z_{t-1} | x_{t-1}) \]

\[ \mu_{t-1} \]

\[ \sigma_{t-1} \]

\[ x_{t-1} \]

\[ h_{t-1} \]

\[ p(y_{t-1} | h_{t-1}) \]

\[ q(z_t | x_t) \]

\[ \mu_t \]

\[ \sigma_t \]

\[ x_t \]

\[ h_t \]

\[ p(y_t | h_t) \]

\[ q(z_{t+1} | x_{t+1}) \]

\[ \mu_{t+1} \]

\[ \sigma_{t+1} \]

\[ x_{t+1} \]

\[ h_{t+1} \]

\[ p(y_{t+1} | h_{t+1}) \]

\[ y_{t-1} \]

\[ y_t \]

\[ y_{t+1} \]
• This is considerably more difficult than estimating from range images.
• It requires a large training database.
Connectomics

- Building the wiring diagram of the brain.
- Finding long range connections.
  
  —> One step towards understanding how it works.
Dendrites and Axons

Fluorescent neurons in the adult mouse brain imaged in vivo through a cranial window using a 2-photon microscope.
The road centerlines are used to plot routes.
Before Machine Learning

- Detect road centerlines
- Find generic paths
- Apply semantic filter
Train a classifier to do this.

To train the classifier, we must associate a feature vector to each path and they all must be of the same dimension.
Histogram of Oriented Gradients

- tile window into 8 x 8 pixel cells
- each cell represented by HOG

Feature vector dimension = 16 x 8 (for tiling) x 8 (orientations) = 1024
Histogram of Gradient Deviations

\[ \Psi(x) = \begin{cases} 
\text{angle}(\nabla I(x), N(x)) & , \text{if } \|x - C(s_x)\| > \varepsilon \\
\text{angle}(\nabla I(x), \Pi(x)) & , \text{otherwise},
\end{cases} \]

\( \rightarrow \) One histogram per radius interval plus four geometric features (curvature, tortuosity, ....).
Roads
Brainbow Images
Blood Vessels
Deep Learning Tsunami

An opportunity to revisit and improve the pipeline:

- Reformulate individual components in terms of CNNs.
- Make them consistent with each other.

AlexNet 2012

The end of computer science as we know it

or .....
• Machine learning enables the **same** algorithm to work in many different contexts but requires hand-designed features.

• However, computing the tubularity and classifying the paths are closely related tasks. They should not be treated separately.

—> Can we use Deep Learning to account for this?
ResNet to U-Net

ResNet block

\[ x + l_2(\sigma(l_1(x))) \]

U-Net

Downsampling

Upsampling

Image

Tubularity Map
Reminder: Downsampling by Pooling

- Reduces the number of inputs by replacing all activations in a neighborhood by a single one.
- Can it be reversed?
Upsampling by Duplication

<table>
<thead>
<tr>
<th>i1</th>
<th>i2</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3</td>
<td>i4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i1</th>
<th>i1</th>
</tr>
</thead>
<tbody>
<tr>
<td>i2</td>
<td>i2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i3</th>
<th>i3</th>
</tr>
</thead>
<tbody>
<tr>
<td>i4</td>
<td>i4</td>
</tr>
</tbody>
</table>
Upsampling by Interpolation

\[
\begin{array}{|c|c|c|}
\hline
\text{i1} & \text{i5}=(\text{i1}+\text{i2})/2 & \text{i2} \\
\text{i6}=(\text{i1}+\text{i3})/2 & \text{i9}=(\text{i1}+\text{i2}+\text{i3}+\text{i4})/4 & \text{i7}=(\text{i2}+\text{i4})/2 \\
\text{i3} & \text{i8}=(\text{i3}+\text{i4})/2 & \text{i4} \\
\hline
\end{array}
\]
Upsampling by Bilinear Interpolation

$\text{I10} = (i_1 + i_2 + \ldots) / 4$

$\text{i1}$

$\text{i2}$

$\text{i3}$

$\text{i4}$

$\text{i9} = (i_1 + i_2 + i_3 + i_4) / 4$

$\text{i11} = (\text{I10} + i_1 + i_2 + i_9) / 4$

$\text{i2}$

$\text{i4}$

$\Rightarrow$ Each pixel is the average of 4 neighbors.
Upsampling by Transposed 1D Convolution

\[ W + w - 1 \]

\[ \begin{array}{cccc}
2 & 3 & 0 & -1 \\
1 & 2 & -1 \\
2 & 4 & -2 \\
3 & 6 & -3 \\
\end{array} \]

Output

\[ \begin{array}{cc}
2 & 7 \\
\end{array} \]
Transposed 1D Convolution

\[ W + w - 1 \]

Output

\[ \begin{array}{ccc}
    2 & 7 & 4 \\
\end{array} \]
Transposed 1D Convolution

\[ W + w - 1 \]
Transposed 1D Convolution

Output

2 3 0 -1

2 4 -2
3 6 -3
0 0 0
-1 -2 1

2 7 4 -4 -2 1

W + w − 1
Transposed 1D Convolution

- The summations are performed in the vertical direction instead of the horizontal one.
- If we wrote this in terms of a fully connected layer, this would amount to transposing the weight matrix.
- Can be extended to 2D layers.
Introducing a Stride Parameter

\[ s(W - 1) + w \]

Output
Introducing a Stride Parameter

\[
\begin{align*}
\text{Output} & = s(W - 1) + w \\
& = \begin{bmatrix} 2 & 4 & 1 & 6 \end{bmatrix}
\end{align*}
\]
Introducing a Stride Parameter

\[ s(W - 1) + w \]

Output

\[ 2 \quad 4 \quad 1 \quad 6 \quad -3 \quad 0 \]
Introducing a Stride Parameter

\[ s(W - 1) + w \]

Output

\[
\begin{array}{cccccccc}
2 & 4 & 1 & 6 & -3 & 0 & -1 & -2 & 1 \\
\end{array}
\]
Introducing a Stride Parameter

\[
\begin{array}{cccc}
2 & 3 & 0 & -1 \\
\end{array}
\]

\[
W = (W - 1) + w
\]

\[
\begin{array}{cccc}
2 & 4 & -2 \\
\end{array}
\]

\[
\begin{array}{cccc}
3 & 6 & -3 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 \\
\end{array}
\]

Output

\[
\begin{array}{cccccccc}
2 & 4 & 1 & 6 & -3 & 0 & -1 & -2 & 1 \\
\end{array}
\]

\[
s(W - 1) + w
\]
Introducing a Stride Parameter

\[ s(W - 1) + w \]
Estimating the Tubularity

Train Encoder-decoder U-Net architecture using binary cross-entropy

Minimize

\[ L_{BCE} = \frac{1}{N} \sum_{i=1}^{N} y_n \log(\hat{y}_n) + (1 - y_n) \log(\hat{y}_n) \]

where

- \( \hat{y} = f_w(x) \),
- \( x \) in an input image,
- \( y \) the corresponding ground truth.

Mosinska et al, CVPR’18.
Tubularity Map

Image  BCE Loss  Ground truth
Iterative Refinement

Use the same network to progressively refine the results keeping the number of parameters constant.
Before Deep Learning

U-Net does this better.

Can it also do this?

Turetken et al., PAMI’16.
Dual Use UNet

Image and Binary Mask

Tubularity Map

[0.991]

Path score
1. Compute a probability map.
2. Sample and connect the samples.
3. Assign a weight to the paths.
4. Retain the best paths.

After Deep Learning
Streets of Toronto

- False negatives
- False positives
Dendrites and Axons

- Deep learning allows the same algorithm to work in different contexts.
- The implementation is informed by earlier approaches.
Accounting for Topology

-> Add a term in the loss function that penalizes the existence of a path between A and B.
It is difficult to make predictions, especially about the future. Sometimes attributed to Niels Bohr.
Alpha Go

- Uses Deep Nets to find the most promising locations to focus on.
- Performs Tree based search when possible.
- Relies on reinforcement learning and other ML techniques to train.

→ Beat the world champion in 2017.