Removed sections

Miroslav Kovář

December 21, 2018

Chapter 1

Convolutional Neural Networks

1.1 Mathematical background

TODO: Do we really need this section???

Definition 1. *Let I be an image function, K a kernel. A (discrete)* ***convolution*** *of I and K is a functional defined as*

$$(I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)\,. \tag{1.1}$$

Note that some machine learning libraries (such as Tensorflow) implement **cross-correlation** instead of convolution, but preserving the term convolution for the operation. Cross-correlation corresponds to convolution with kernel rotated by 90 degrees:

$$(I * K)(i, j) = \sum_m \sum_n I(m, n)K(i + m, j + n)\,. \tag{1.2}$$

Unlike convolution, cross-correlation is not commutative, but this property is not required for neural network applications.

Definition 2. *Let f be arbitrary function, and $\mathcal{D}$ its degradation operator. We say f is* ***invariant*** *under $\mathcal{D}$ if*

$$\mathcal{D}(f) \equiv f\,. \tag{1.3}$$

For the following, the reader needs to understand the term **equivariance**.

Definition 3 ([8])**.** *Let G be a group and X, Y its G-sets. Then $F : X \to Y$ is called an* ***equivariant function*** *if*

$$F(g(x)) = g(F(x)) \tag{1.4}$$

for all G actions g and $x \in X$.

For our purposes, we can view G as a group of transformations, and then equivariance as a commutative property of a function with regards to the transformations. In other words, computing the function and then applying the transformation has the same effect as applying the transformation and then computing the function.

Algorithm 1 Gradient descent algorithm.

1: Initialize random $x_0 \in D(f)$
2: $n \leftarrow 0$
3: step_size $\leftarrow 1$
4: **while** step_size $<$ threshold and $n <$ iters_limit **do**
5: $\quad x_{n+1} = x_n - \epsilon \nabla_{x_n} f$
6: $\quad$ step_size $\leftarrow |x_{n+1} - x_n|$
7: $\quad n \leftarrow n + 1$
8: **end while**

TODO: This is probably too basic to be here

Gradient descent is a first order iterative method of finding an extremum a differentiable function $f : \mathbb{R} \to \mathbb{R}^n$, $f \in C^1$, based on continually moving a point in its domain in the direction of negative of its gradient at that point, until the absolute value of the gradient (or the step size) is below a certain threshold. See Algorithm 1.

Stochastic gradient descent...

1.2 History

The classical approach to image pattern recognition consists of the following stages:

preprocessing: supressing unwanted distortions and noise, enhancement beneficient for further processing,

object segmetation: separating disparate objects from the background,

feature extraction: gathering relevant information about the properties of the objects, removing irrelevant variations,

classification: categorizing segmented objects based on obtained features into classes.

The preprocessing step may require additional assumptions about the data or further processing, which are potentially too restrictive or too broad. Getting around this limitation requires dealing with complications such as high dimensionality of the input (number of pixels) and desirability of invariance towards a number of allowable distortions and geometrical transformations.

Artificial neural networks in combination with gradient-based learning are one possible solution to the problem. By gradually optimizing a set of weights based on a training data set using a differentiable error function, they provide a framework for learning a suitable set of assumptions automatically from the data.

One of the oldest neural network architectures, fully connected multi-layer perceptron (FC-MLP), can be used for image pattern recognition. However, it has the following drawbacks:

parameter explosion: the number of parameters of such network is exponential in the number of layers, increasing the capacity of the network and therefore need for more data,

no invariance: no invariance even with respect to common geometrical transformation such as translation, rotation and scaling,

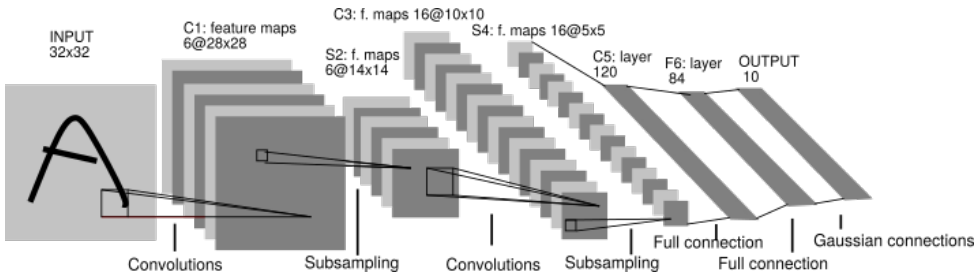

Figure 1.1: LeNet-5 architecture [7].

ignoring input topology: natural ../Images exhibit strong local structure and high correlation between intensities of neighboring pixels, but FC-MLPs are unstructed - inputs can be presented in any order.

Although the main idea dates back to 1980, when K. Fukushima introduced neocognitron [2], the back-propagation algorithm was not known at the time. The first convolutional architecture successfuly applied on an image pattern recognition problem by attempting to solve the aforementioned problems, dubbed LeNet-5, was proposed in 1998 by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner [6].

1.3 Description

Bearing resemblence to visual processing in biological organisms [1], LeNet-5 proposed the following design principles to enforce *shift, scale and distortion invariance*: [7]

local receptive fields: each neuron in a layer receives input from a small neighborhood in the previous layer,

shared weights: each layer is composed of neurons organized in planes within which each neuron have the same weight vector (feature map),

spatial subsampling: adding a subsampling layers, which reduce the resolution of the previous layer by averaging or taking the maximal value of neighboring pixels in the previous layer.

1.3.1 Local receptive fields

Local receptive fields enable the network to synthesize filters that produce strong response to elementary salient features in the early layers (such as lines, edges and corners in a visual input, and their equivalents in other modalities), and then learn to combine them in the subsequent layers to produce higher-order feature detectors.

For a visual explanation of the concept of receptive field, see Figure 1.2. The locality of of those receptive fields implies sparser connectivity, and hence more efficient computations in comparison with fully connected neural networks. A fully connected neural network with no hidden layers with m inputs and n outputs has $m \times n$ weight parameters, and the correspoding feed forward pass (matrix multiplication) is of $O(m \times n)$ time complexity per input. If the number of connections per output unit is limited to $k < m$,

[1] As early as in 1968, D. H. Hubel and T.N. Wiesel discovered that some cells (called simple cells) in cat's primary visual cortex (V1) with small receptive fields (shared by neighboring neurons) are sensitive to straight lines and edges of light of particular orientation, and other cells (called complex cells) with larger receptive fields further in the visual cortex also respond to straight lines and edges, but with invariance to translation [4].

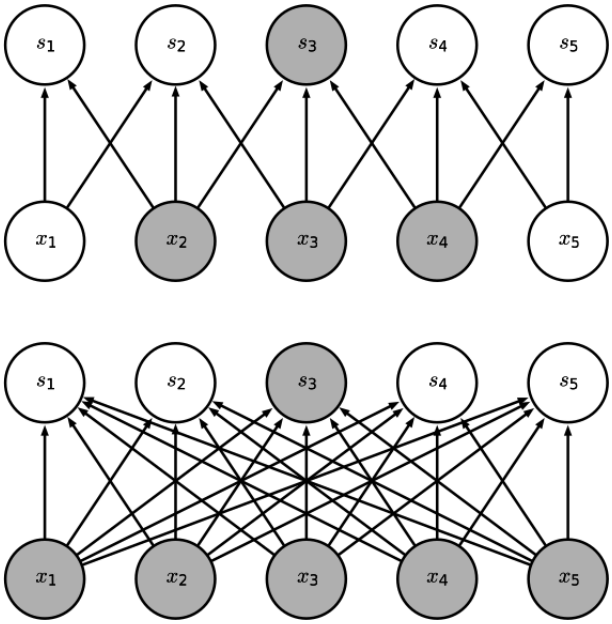

Figure 1.2: Receptive field. [3]

the achieved runtime is $O(k \times n)$, where k is usally in practice several orders of magintude smaller than m. [3]

In shallow neural networks, locality of receptive fields implies locality of "influence" of each input unit on the output. In deep neural networks, on the other hand, units in the deeper layers can be indirectly connected to some or all units of the input, thus enabling them to achieve aforementioned effect of combining more complex features from simpler ones.

1.3.2 Shared weights

With *shared weights*, neural units in a layer with differing receptive fields have the same feature map and the same feature detecting operation (convolution with feature map kernel followed by additive bias and a application of a non-linear function) is performed on differing parts of the image (see Figure 1.3). A single convolutional layer is composed of multiple feature detecting planes.

Shared weights principle exploits the fact that in natural ../Images, a function of small number of neighboring pixels can be useful in multiple parts of the image. For example, an edge detector can be used accross the entire image to detect edges in the first layer, an object detector can then be used to detect presence of edges in particular arrangements in the next layer, etc.

Although it does not reduce the time complexity of the feedforward pass, it does reduce the memory requirements. If the kernel size is k, m the number of inputs, n the number of outputs, the number of parameters per layer is k instead of $m \times n$ (per feature detecting plane) in a fully connected case. Since k is usually in practice several orders of magnitude smaller than m, and usually m and n are comparable in size, the memory savings are highly significant. [3]

One of the drawbacks of classical CNNs is that although convolution in combination with weight sharing causes layer output to be equivariant to translation of the input, this is not the case for scaling and rotation. Moreover, equivariance to input may not be always desirable. Consider a case of face detection, where all training and test ../Images are centered. Then, the relative positions of individual

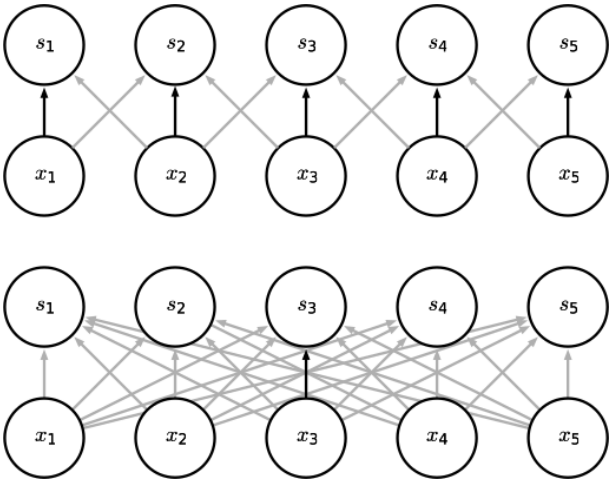

Figure 1.3: Shared weights. [3]

features are important, and it may be favorable to fix feature detectors (and thus weights) to certain locations in the image.

1.3.3 Pooling

The final output activations of a convolutional layer are computed in subsequent stages:

1. linear unit activations are computed via the convolution operation,

2. a non-linear activation function is applied to the activations,

3. a spatial subsampling (pooling) operation is applied.

The rationale behind applying a non-linearity is it makes the network capable of modelling non-linear functions. Common activation functions include rectified linear $\max(0, x)$, sigmoid $\frac{1}{1+\exp(-x)}$, hyperbolic tangent tanh, and many others. They have varying properties making them useful in different situations. We will not explore them further here.

Pooling operation splits the neural units into sets of multiple adjacent activations and computes a summary statistic, such as the maximum element (max pooling) or the average (average pooling), per such set and outputs the result. If the stride between the sets is greater than one, the spatial dimension of output is decreased relative to input (subsampling).

The purpose of spatial subsampling is to ensure scale and distortion invariance[2] by reducing the precision at which a feature is encoded in a feature map by reducing its resolution - when scale and distortion invariance is assumed, the exact location of a feature becomes less important and is allowed to exhibit slight positional variance - roughly speaking, an "approximate" translation invariance.

Although the combination of convolution and pooling performs well in many practical situations, it has multiple drawbacks. For example, the learned representations are not rotation invariant and thus, to mitigate this, the capacity of the network has to be increased and the training dataset must be enhanced to contain examples of rotated features, often extending the amount of data necessary and training time. A

[2]Whether it achieves this goal has been famously doubted by Geoffrey Hinton: "The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster." []

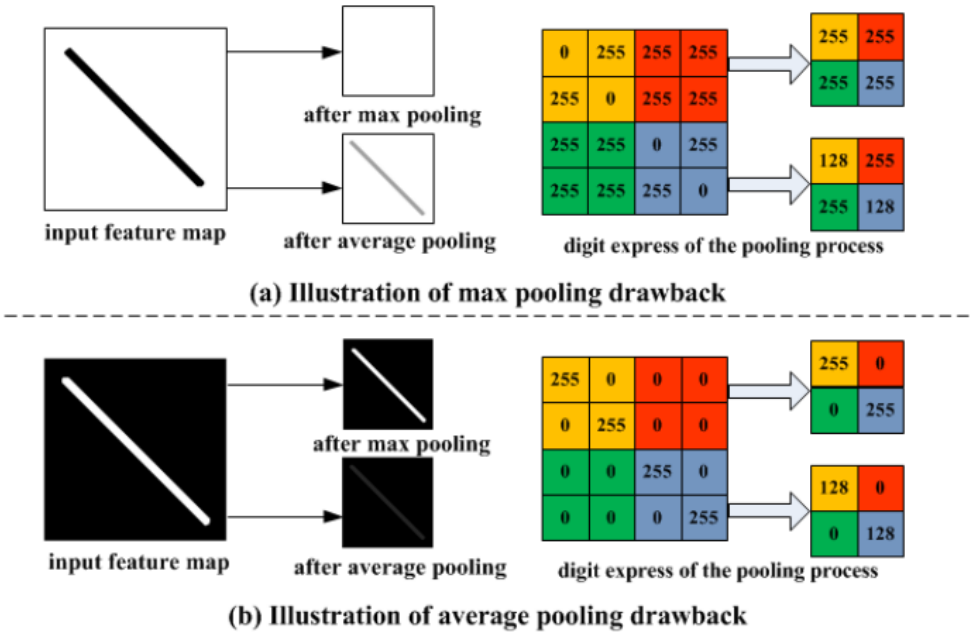

Figure 1.4: Examples of drawbacks of the pooling operation. Max pooling discards all except the maximum element, and valuable information may thus be lost. Average pooling considers all the values, and the information about their contrast is reduced. Moreover, extreme values may have undesired effects on the result. [10]

number of alternative approaches were suggested in the litarature.[3] For another example of a limitation, see Figure 1.4.

1.4 Applications

Maybe mention an example of how LeNet-5 was improved on subsequently (AlexNet, ResNet, etc.)? But this changes all the time...

1.5 CapsNets?

Does it make sense trying them? I found a only a few successful implementations. Maybe it would be better to try those after we have some results already, because it seems risky - we might end up with nothing.

[3] For instance, Hinton's *CapsNet*, described e.g. in [9], is an attempt to transform the manifold of ../Images of similar shape (which is highly non-linear in the space of pixel intensities) to a space where it is globally linear by the way of using so called capsules instead of traditional convolutional layers.

Chapter 2

Dynamical systems

2.1 Visual characterization of the dynamical system

2.1.1 Phase space plot

2.1.2 Poincare plot

2.1.3 Recurrence plot

When presented with a task of finding regularities in seemingly chaotic data, one possible approach is analysing at least approximate repetitions of simple patterns, which can be further used for reconstruction of more complicated rules. Recurrence plot is a method of visualizing obtained state-space trajectory segments in relation to each other to achieve this goal. Furthermore, it can be used to test necessary conditions for validity of dynamical parameters derivable from a non-linear time series such as the information dimension, entropy, Lyapunov exponents, dimension spectrum, etc. The information contained in recurrence plots is not easily obtainable by other known methods. [1]

Definition 4 ([1])**.** *Let N be the length of given time series, $\mathbf{s}_i$ for $i \in \{1, 2, \ldots, N\}$ be a i-th delay vector of any integer embedding dimension, $\|\cdot\|$ a norm, $\Theta(\cdot)$ a Heaviside step function, and $\epsilon \in \mathbb{R}_0^+$ a tolerance parameter. Then,* ***recurrence plot*** *is the matrix*

$$M_{ij} = \Theta(\epsilon - \|\boldsymbol{s}_i - \boldsymbol{s}_j\|)\,. \tag{2.1}$$

In other words, M_{ij} is a symmetric[1] binary NxN matrix, where $M_{ij} = 1$ when i-th and j-th points of the reconstructed trajectory enter each other's ϵ neighborhood.

The essential drawback of recurrence plot is their size - it is quadratic in the length of the time series. A simple way of reducing its dimension is to partition the time series into disjoined segments, and let M_{ij} represent the distance between those two segments. This is known as **meta-recurrence plot**. [5] (TODO: Find a justification for using them.)

(TODO: Cross-recurrence plots may be useful? Only between two series. Joint recurrence plots may be used to detect phase synchronization.)

[1] Although this is true for our definition, it may not be true for an alternative definition using a more general topology instead of a norm.

Bibliography

[1] J.-P Eckmann, S. Oliffson Kamphorst, and D Ruelle. Recurrence Plots of Dynamical Systems. *Europhysics Letters (EPL)*, 4(9):973–977, 1987.

[2] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[3] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[4] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 1968.

[5] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

[6] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

[7] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[8] Andrew M Pitts. *Nominal sets: Names and symmetry in computer science*. Cambridge University Press, 2013.

[9] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic Routing Between Capsules. (Nips), 2017.

[10] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *International Conference on Rough Sets and Knowledge Technology*, pages 364–375. Springer, 2014.