Convolutional and Recurrent Neural Networks

Johanni Brea

Introduction à l'apprentissage automatique

GYMINF 2021

Trying to Solve MNIST with Feature Engineering (Version I)

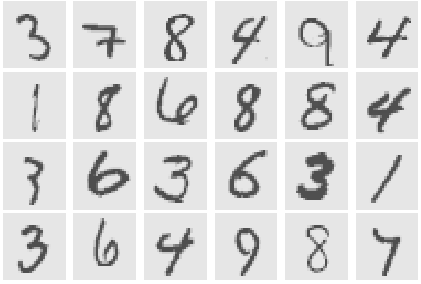
Idea: Pick randomly some training images j as feature vectors.

$H_{ij} = \text{relu}(x_i^T x_j)$ for i in training or test set indices and j in index set of randomly picked training images.

Perform multinomial regression on the feature representation.

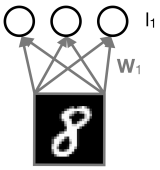
Test Misclassification Rate: $\approx 10.0\%$

Some hand-picked feature vectors (random subset of the training input)

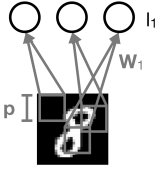


Trying to Solve MNIST with Feature Engineering (Version II)

Full connectivity

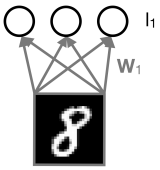


Localized connectivity

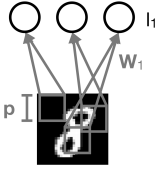


Trying to Solve MNIST with Feature Engineering (Version II)

Full connectivity

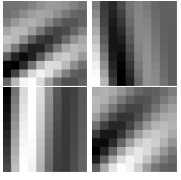


Localized connectivity



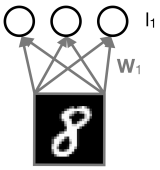
5000 localized random Gabor feature vectors

Random Gabor feature vectors

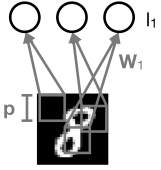


Trying to Solve MNIST with Feature Engineering (Version II)

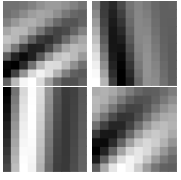
Full connectivity



Localized connectivity



Random Gabor feature vectors



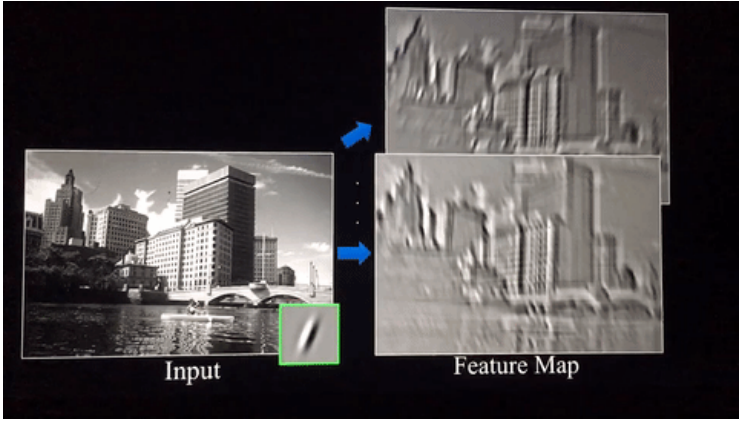
5000 localized random Gabor feature vectors

Test Misclassification Rate: $\approx 1.1\%$

Illing, Gerstner, Brea (2019)

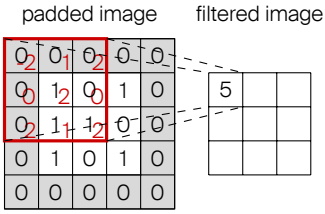
Table of Contents

Reusing Local Features: Convolutions



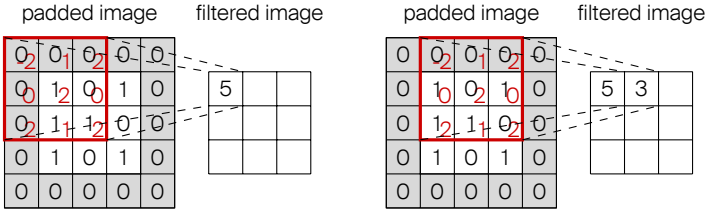
Convolution

filter size = 3×3**, stride = 1, padding = 1**



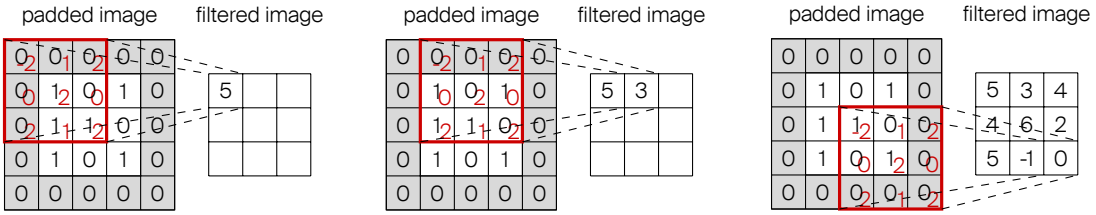
Convolution

filter size = 3×3**, stride = 1, padding = 1**



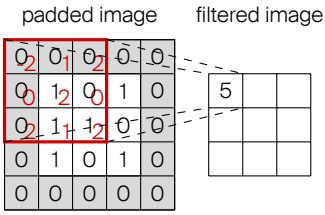
Convolution

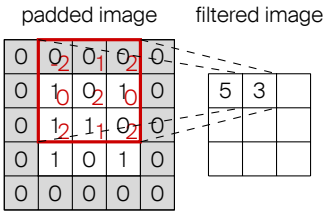
filter size = 3×3, stride = 1, padding = 1

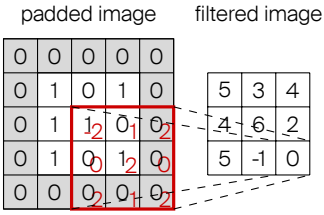


Convolution

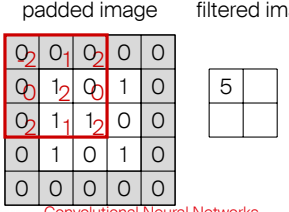
filter size = 3×3, stride = 1, padding = 1





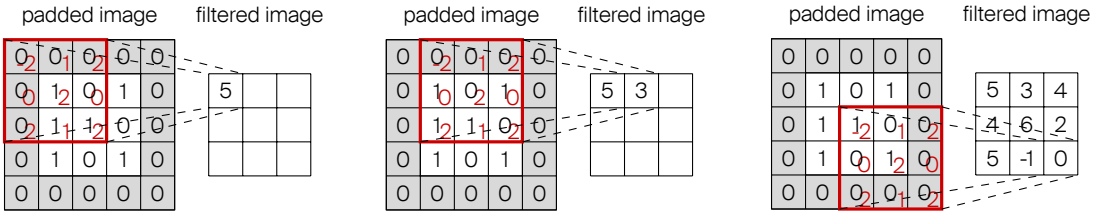


filter size = 3×3, stride = 2, padding = 1

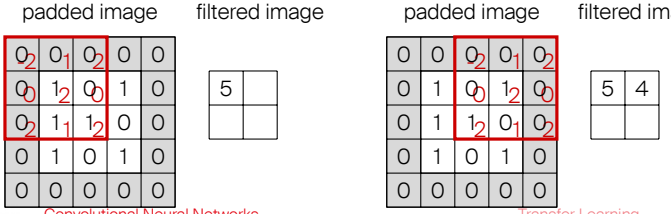


Convolution

filter size = 3×3, stride = 1, padding = 1

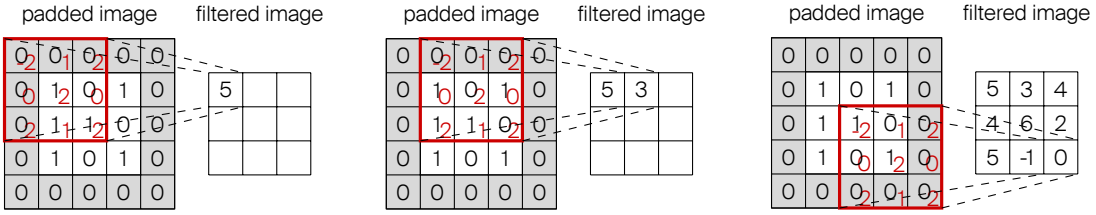


filter size = 3×3, stride = 2, padding = 1

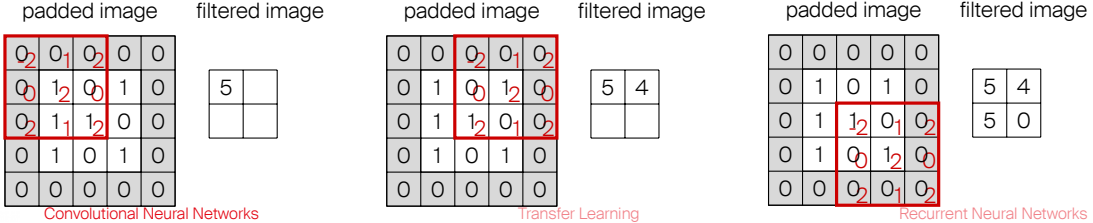


Convolution

filter size = 3×3, stride = 1, padding = 1



filter size = 3×3, stride = 2, padding = 1



Learning MNIST Features with a Convolutional Neural Network

Convolutional Neural Network

28×28 input neurons
1 hidden layer with 200 relu conv. filters
10 softmax neurons

Test Misclassification Rate: $< 1\%$

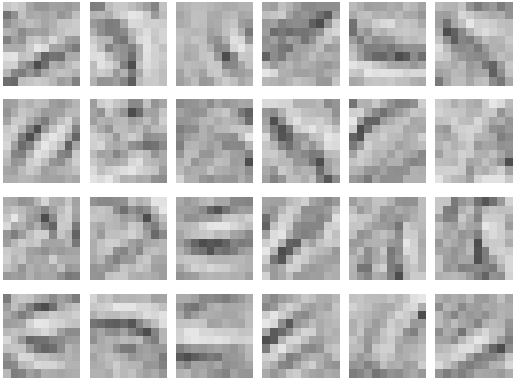
Learning MNIST Features with a Convolutional Neural Network

Convolutional Neural Network

28×28 input neurons
1 hidden layer with 200 relu conv. filters
10 softmax neurons

Test Misclassification Rate: $< 1\%$

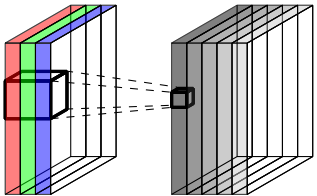
Examples of learned features



Convolution Volumes

From 3 (colour) channels to 5 (filter) channels

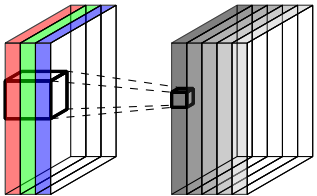
colour image multi-filtered image



Convolution Volumes

From 3 (colour) channels to 5 (filter) channels

colour image multi-filtered image



Input volume $n \times n \times c$ leads to output volume

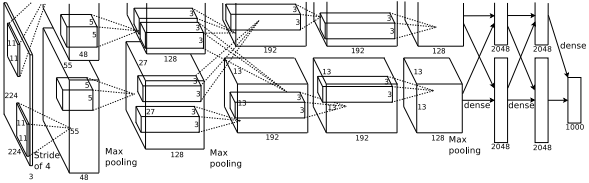
$$\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right) \times k\,.$$

padding p, filter size (f, f), stride s, number of filters k

Convolutional Networks: AlexNet

winner of imagenet competition 2012

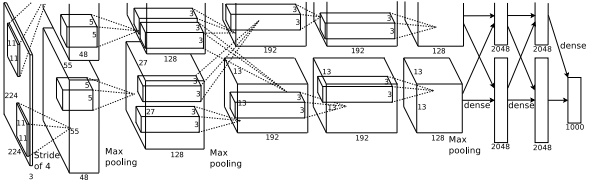
input: colour images $224 \times 224 \times 3$
output: 1000 classes
hidden layers: 5 convolutional layers,
3 dense layers

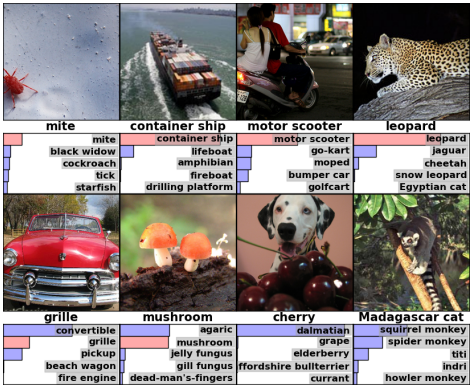


Convolutional Networks: AlexNet

winner of imagenet competition 2012

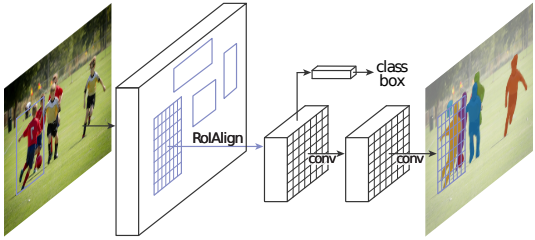
input: colour images $224 \times 224 \times 3$
output: 1000 classes
hidden layers: 5 convolutional layers,
3 dense layers





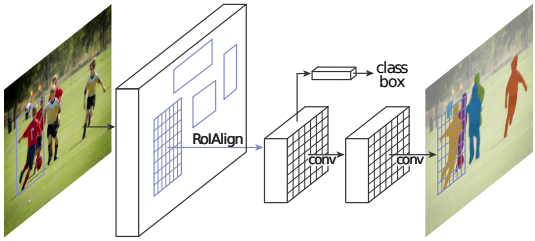
ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky, Sutskever, Hinton, 2012

Convolutional Networks: Mask R-CNN



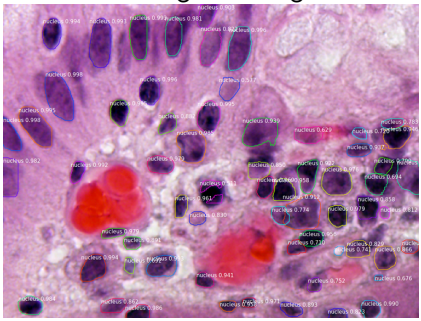
$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{class}} + \mathcal{L}^{\text{box}} + \mathcal{L}^{\text{mask}}$$

Convolutional Networks: Mask R-CNN



$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{class}} + \mathcal{L}^{\text{box}} + \mathcal{L}^{\text{mask}}$$

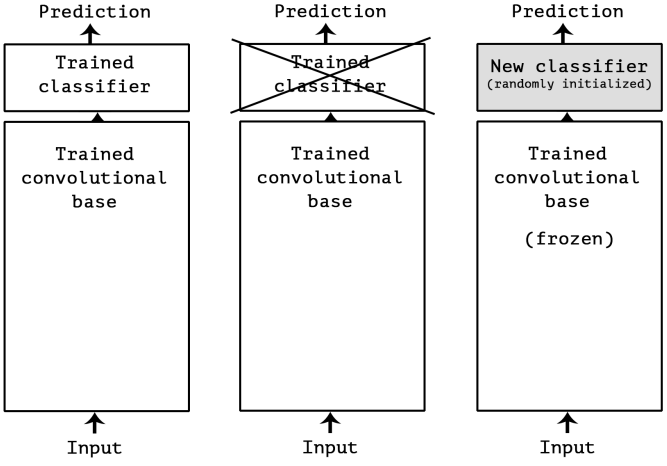
Nuclei Counting and Segmentation



Mask R-CNN, He Gkioxari, Dollár, Girshick, 2018
`https://github.com/matterport/Mask_RCNN`

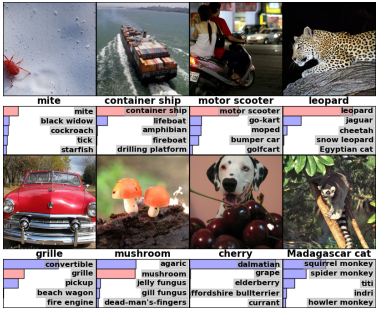
Table of Contents

Transfer Learning: Reusing Learned Features



Transfer Learning: Reusing Learned Features

Training Set for Features
Imagenet Dataset: more than 1 million images 1000 categories



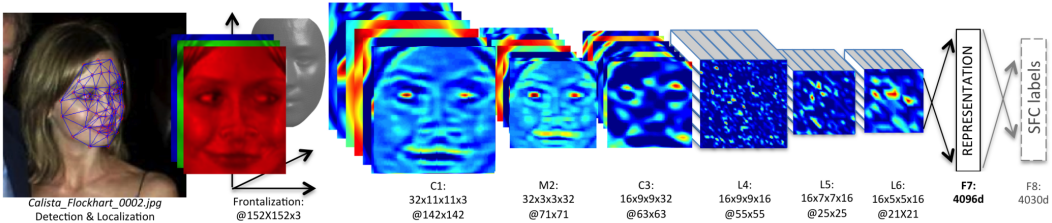
Training Set for Classifier
102 Category Flower Dataset (≈ 10'000 images)

102 Category Flower Dataset

Category	#ims	Category	#ims	Category	#ims
alpine sea holly	43	buttercup	71	fire lily	40
anthurium	105	californian poppy	102	foxglove	162
artichoke	78	camellia	91	frangipani	166
azalea	96	canna lily	82	fritillary	91

Transfer Learning: Reusing Learned Features

DeepFace: Closing the Gap to Human-Level Performance in Face Verification (2014)



- 4 million user-labeled faces on FaceBook images of 4000 individuals
- Retrain fully-connected layers at the top on Labeled Faces in the Wild (LFW) dataset reaching (human level) accuracy of 97.35%

Table of Contents

Recurrent Neural Networks(RNN)

Let us assume time dependent inputs $x(t) = (x(0), x(1), \ldots, x(T))$.
In a recurrent neural network the activity $a^{(l)}(t)$ of the hidden neurons at time t does not only depend on the input $x(t)$ at t, but also on the previous hidden activity $a^{(l)}(t-1)$, e.g.

$$a^{(l)}(t) = g^{(l)}(w^{(l)}a^{(l-1)}(t) + w^{(l,\text{rec})}a^{(l)}(t-1) + b^{(l)})$$

Application of a Recurrent Neural Network for Language Detection

inputs: sentences
output: language label

E.g. $x(1) = "t"$ $x(2) = "o"$ $x(3) = " "$ $x(4) = "b"$ $x(5) = "e"$ $x(6) = ","$
x(7) = "o" x(8) = "r" x(9) = " " x(10) = "n" x(11) = "o" x(12) = "t" x(13) = " "
x(14) = "t" x(15) = "o" x(16) = " " x(17) = "b" x(18) = "e" x(19) = "."
The final hidden activity $a^{(l)}(19)$ of the recurrent neural network (the scanner network) depends on the whole phrase.

The final hidden activity of the scanner network can be used as input to a classifier neural network (the encoder network).

The parameters of the scanner and the encoder are jointly fit with gradient descent to maximize the likelihood function.