Other Nonlinear Supervised Learning Methods

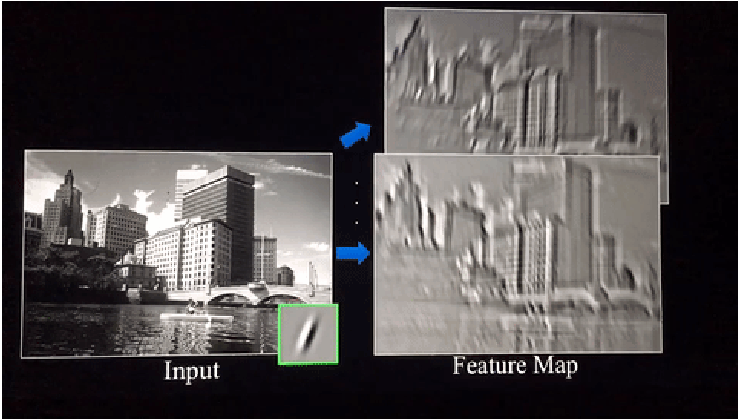
Johanni Brea

Introduction to Machine Learning

EPFL BIO322 2021

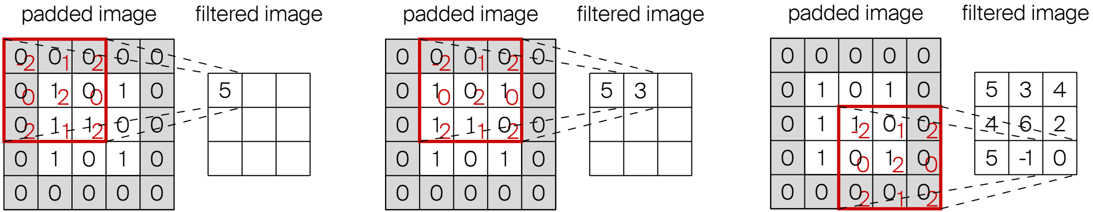
Table of Contents

Reusing Local Features: Convolutions

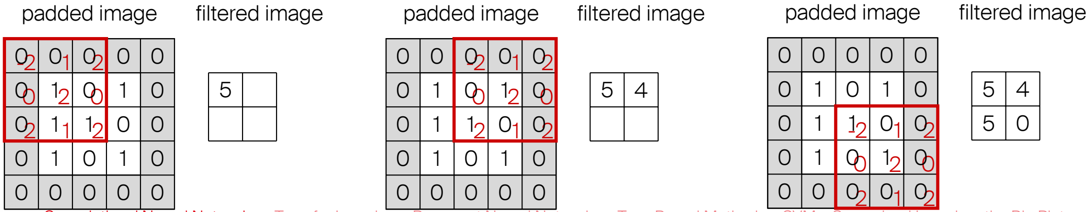


Convolution

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



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



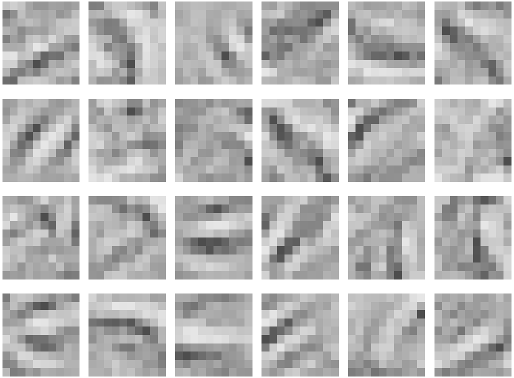
Learning MNIST Features with a Convolutional Neural Network

Convolutional Neural Network

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

Test Misclassification Rate: $< 1\%$

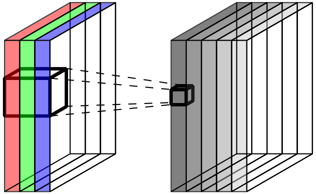
Examples of learned features



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 Layer

A convolutional layer l with $K^{(l)}$ filters of size $(2f+1, 2f+1)$, padding f and stride 1 computes

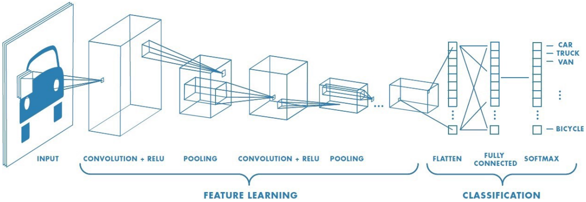
$$a_{xy,k}^{(l)} = g^{(l)}\left(w_{k0}^{(l)} + \sum_{q=-f}^{f}\sum_{r=-f}^{f}\sum_{s=1}^{K^{(l-1)}} w_{qrs,k}^{(l)} a_{x+q,y+r,s}^{(l-1)}\right),$$

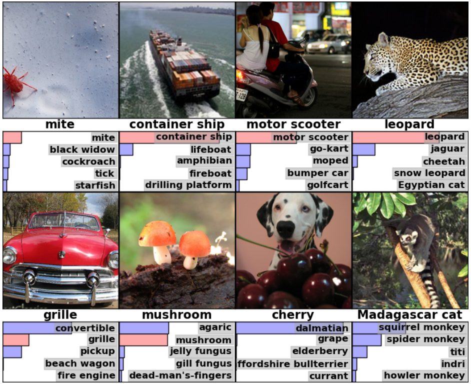
where $k = 1, \ldots, K^{(l)}$, $a_{x+q,y+r,s}^{(l-1)}$ are the activities in the previous layer, $w_{k0}^{(l)}$ is the bias of filter k, $w_{qrs,k}^{(l)}$ are the weights of filter k.

Convolutional Networks

winner of imagenet competition 2012

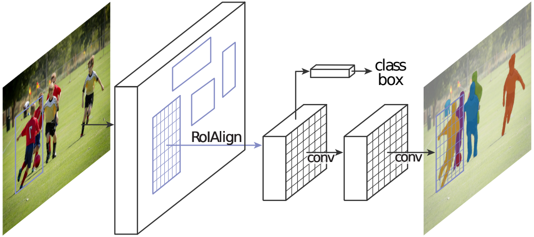
input: colour images $224 \times 224 \times 3$
output: 1000 classes
hidden layers: 5 convolutional layers,
3 dense layers





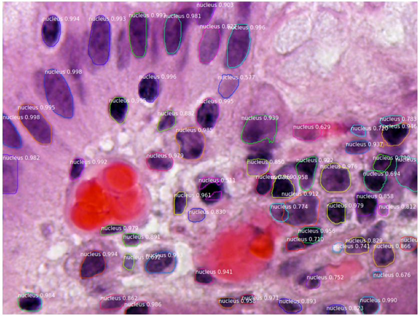
ImageNet Classification with Deep Convolutional Neural Networks, Krizhevsky, Sutskever, Hinton, 2012

Convolutional Networks: Mask R-CNN



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

Nuclei Counting and Segmentation

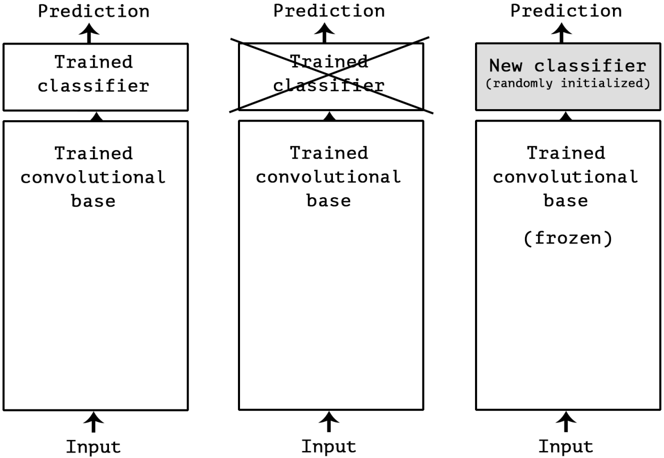


Mask R-CNN, He Gkioxari, Dollár, Girshick, 2018

`https://github.com/matterport/Mask_RCNN`

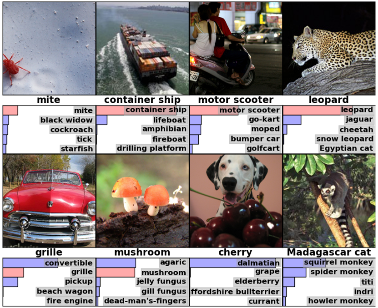
Table of Contents

Transfer Learning: Reusing Learned Features

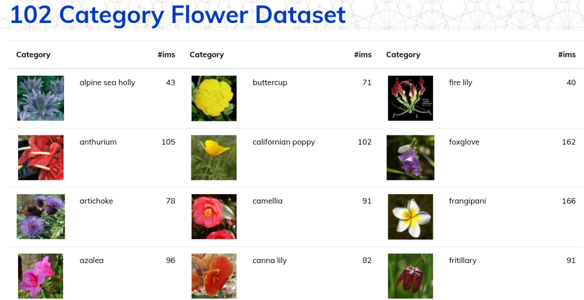


Transfer Learning: Reusing Learned Features

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



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

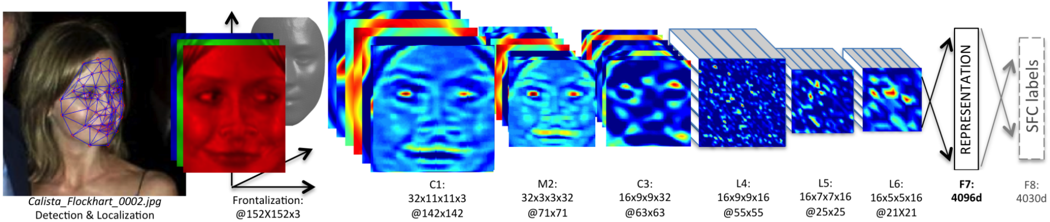


102 Category Flower Dataset

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 $x(t) = (x(0), x(1), \dots, x(T))$.
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.

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

Application of a Recurrent Neural Network for Language Detection

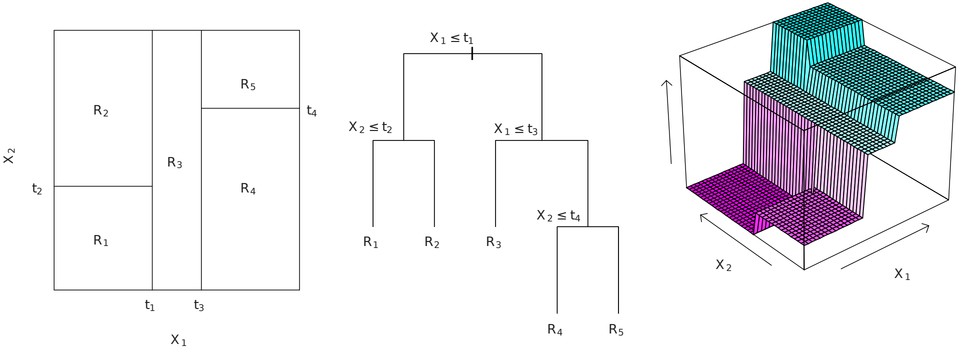
inputs: sentences, output: language label

time step	1	2	3	4	5	6	7	8	9	10	11	12
character	t	o		b	e		o	r		n	o	t
1-hot input	0	1	0	0	0	0	1	0	0	0	1	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	1	0	0	0	0	0	0	0	0	0	0	1
hidden layer	0.1	0.4	0.0	0.8	0.7	0.1	0.6	0.0	0.0	0.0	0.0	0.0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	0.3	0.7	1.0	0.3	0.0	0.8	0.3	0.3	1.1	1.3	1.3	1.4

Final hidden activity depends on whole phrase; can be used to predict the language.
All weights are jointly trained with gradient descent to maximize the likelihood function.

Table of Contents

Tree-Based Methods: Random Forests and Gradient Boosting Trees



Tree with 4 **internal nodes** and 5 **terminal nodes** or **leafs**.

Random Forests & Gradient Boosting Trees (GBT): $f(x) = \sum_{k=1}^{K} a_k \mathtt{tree}_k(x)$
The trees $\mathtt{tree}_k$ (and for GBT the a_ks) are learned.

Table of Contents

Support Vector Machines

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \kappa(x_i, x)$$

- $\kappa(x, y)$ is a called kernel; we can choose the type of kernel, e.g. radial kernel $\kappa(x, y) = \exp(-\gamma \sum_{i=1}^{p}(x_i - y_i)^2)$.
- β_0 and α_i are learned with a special loss function (not maximum likelihood).
- Most α_i's are zero after training.
- The training points x_i with non-zero α_i are called support vectors.

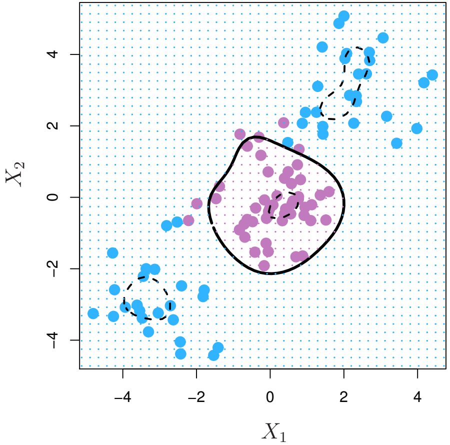
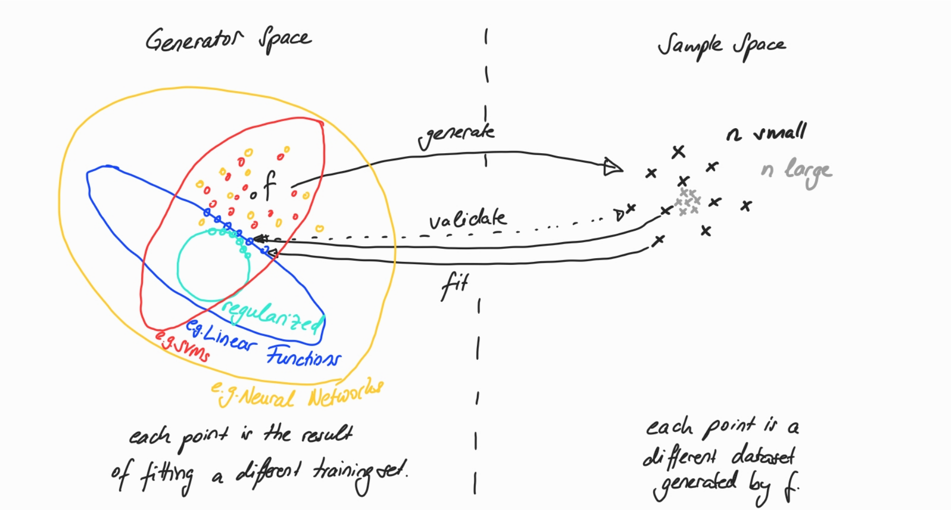


Table of Contents

Supervised Learning: the Big Picture



Supervised Learning: the Big Picture

In general it is not clear which method should be chosen for which task and data set.

Rules of Thumb

- Images as input: convolutional neural networks
- Sequences as input (text, music): transformers or recurrent neural networks
- With Gradient Boosting Trees or Random Forests one can often get decent results in little time (not much tuning is required). But well tuned neural networks tend often to be better.
- Carefully tuned regularization (L1, L2, early stopping or dropout) usually matters a lot.