Deep Learning - MAI

Feed-forward and convolutional neural networks

THEORY

Dario Garcia Gasulla
dario.garcia@bsc.es

A Bit of History

The not so distant origin story

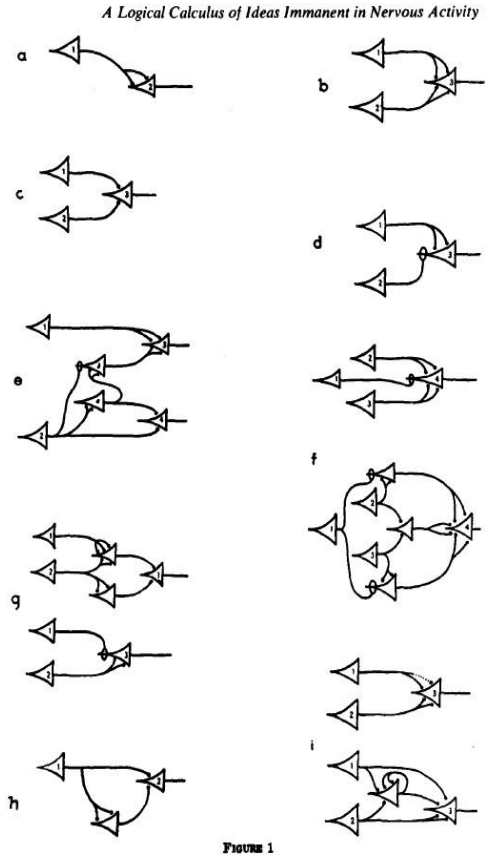
Connectionism & Hebbian Learning

Warren McCulloch & Walter Pitts (1943):

- From neurons to complex thought
- Binary threshold activations

Howard Hebb (1949):

- “Neurons that fire together wire together”
- Weights yield *learning* and *memory*



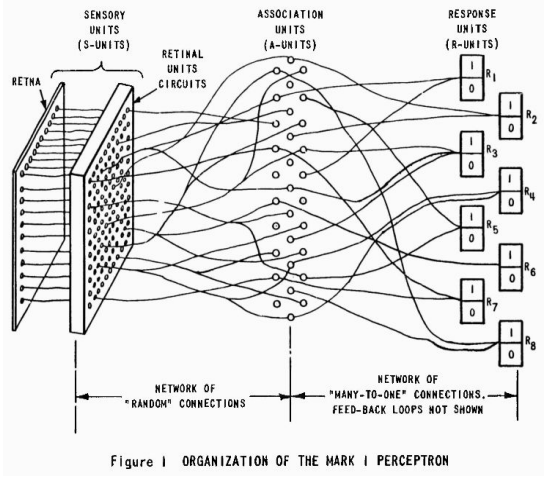
Ref: [1,31]

Rosenblatt's Perceptron

Rosenblatt (1948): Hebb's learning + McCulloch & Pitts design

Mark I Perceptron

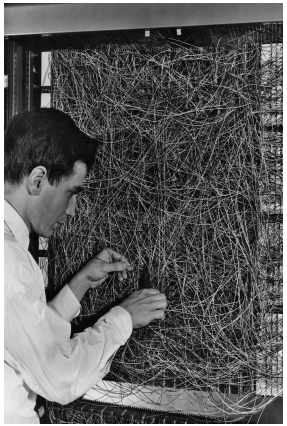
- 400 photosensitive receptors (sensory units)
- 512 stepping motors (association units, trainable)
- 8 output neurons (response units)



Threshold function

$$f(x) = \begin{cases} 1 \text{ if } w \cdot x + b > 0 \\ 0 \text{ otherwise} \end{cases}$$

w real-valued weights
$\cdot$ dot product
b real scalar constant



[2,3,4,5]

Minsky & Papert: The XOR affair

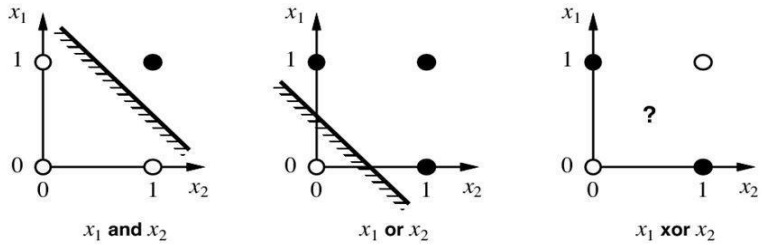
The perceptron capabilities were limited (Rosenblatt)

Linear separability

"Perceptrons: an introduction to computational geometry"

(Minsky & Papert, 1969):

- Perceptron cannot learn non-linearities
- Single Layer networks cannot be trained



NNs abandoned until the mid 80s: ***1st* AI WINTER**

Shift from connectionism to symbolism

Backpropagation Algorithm

How to optimize weights not directly connected to the error?

Backpropagation algorithm:

- Use the chain rule to find the derivative of cost with respect to any variable

Used for training MLPs in (Werbos in 1974, Rumelhart et al. in 1985)

End of NNs Winter (Begining of 2nd AI Winter)

[7,8]

Feedforward Neural Networks

Optimization & Learning

Training through gradient descent

- **Forward pass**
  - Compute output with a given input
  - Error measurement (loss function)
- **Backward pass**
  - Find gradients minimizing error layer by layer
  - Apply gradients

The Gradient Descent Family

Batch GD: Compute gradients of *all training samples* before descending

- Deterministic outcome
- Large memory cost

Stochastic GD: Apply gradient of *one random training sample* at a time

- Stochastic outcome
- Does not guarantee learning, poor parallelization

Mini-batch GD: Combine gradients of a *random subset of train samples*

- Stochastic
- Efficient computation
- Subset size aka *“Batch size”*

This naming is not universally respected!

Mini-batch Training Nomenclature

Num. samples computed together: **The batch size**

One feedforward/backward cycle (one batch): **A step**

$$N = \frac{dataset\_size}{batch\_size}$$

N steps (all training samples once): **An epoch**

Practical Tips I

Factors for defining the batch size (rarely)

- Instance size
- Computational efficiency
- Stochasticity
- Powers of 2

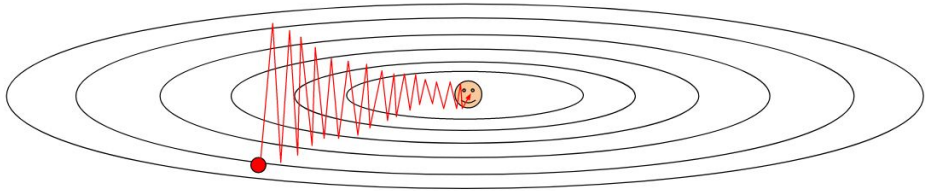
Factors for defining the number of epochs (constantly)

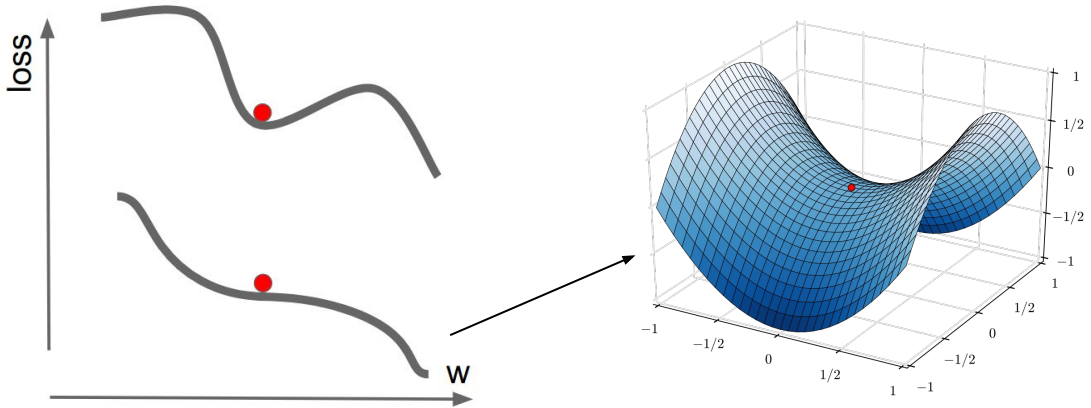
- Convergence
- Reliability
- Footprint

The Art of Descending

Hard to go down in a high dimensional space

- Different loss speed among dimensions
  - Slow and jitter
- Local minima & saddle points
  - Stuck gradient
- Mini-batches are noisy



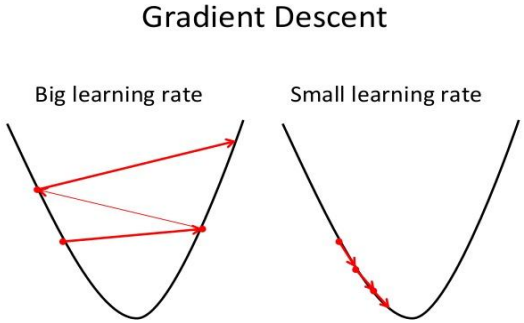


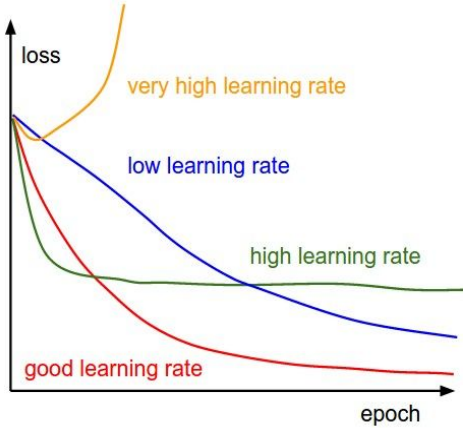
The Art of Descending

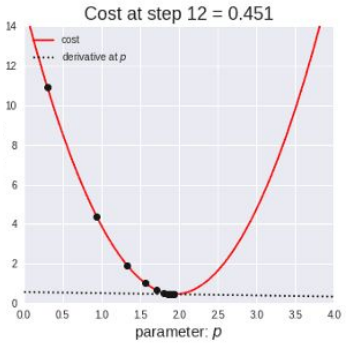
Learning rate: How much you move in the direction of the gradient

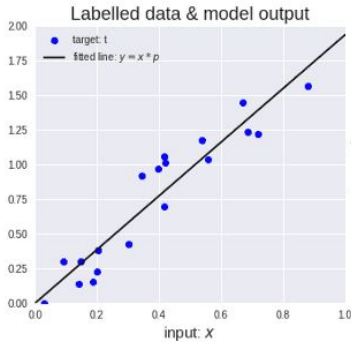
Direct effect on convergence and speed

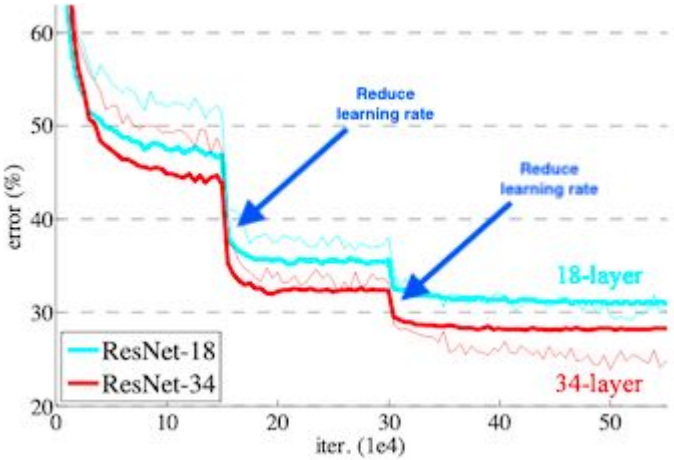
The same LR may not always be the right one











Barcelona Supercomputing Center
Centro Nacional de Supercomputación

HPAI
HIGH PERFORMANCE ARTIFICIAL INTELLIGENCE

Practical Tips II

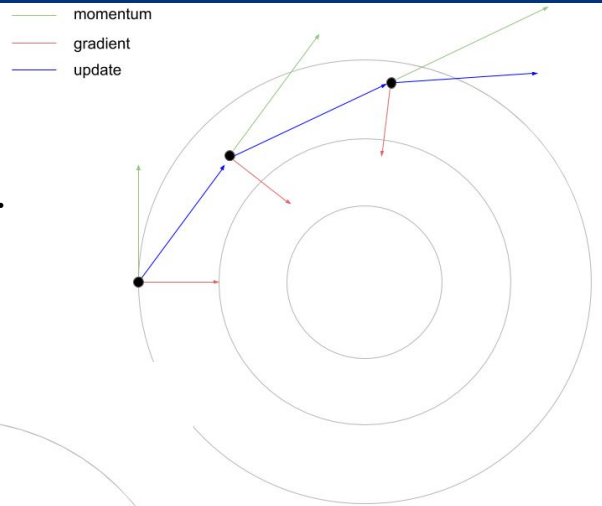
Tuning the Learning Rate

- Fix the batch size (or viceversa)
  - Theory: Double one, double the other
- Always smaller than 1
- Search by orders of magnitude
- Grid search and Random search
- In case of doubt, go small
- When stuck, reduce it

Inertia in Optimizers

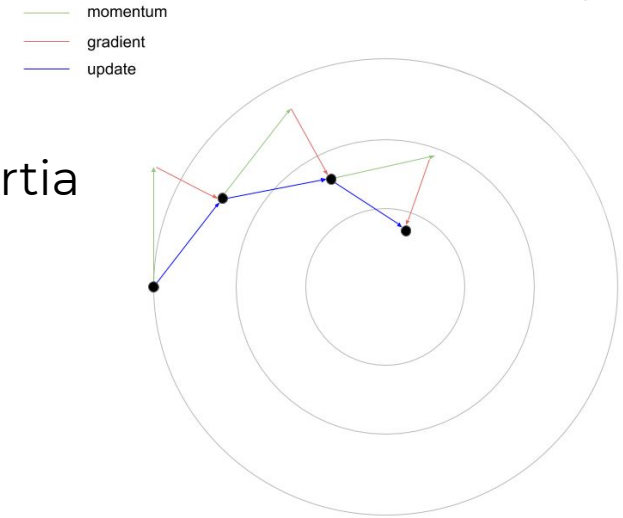
Momentum:

- Add a fraction of the previous gradient. *Inertia*.
- Decaying weight parameter. *Friction*.
- Faster, smoother convergence



Nesterov:

- Gradient computed after inertia
- See the slope ahead
- Faster convergence



Adaptative LR Optimizers

Adagrad:

- Parameter-wise LR considering past updates.
- High LR for infrequent ones. Low LR for frequent ones.
- Good for sparse data. Vanishing step size due to growing history.

Adadelta:

- Adagrad with a decaying average over history (typically around 0.9)

RMSprop:

- Similar to Adadelta

Adam:

- Adadelta + Momentum

Nadam:

- Adadelta + Nesterov

Practical Tips III

Optimizers

- Adam: Current popular default. Competitive without tuning
- SGD + Momentum: Great if LR is decayed properly

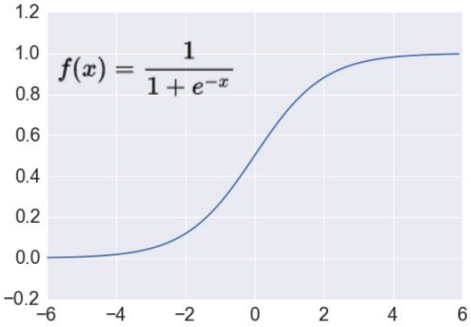
Hyperparameters incomplete list #1 (training)

1. Batch size
2. Number of epochs
3. Learning rate
4. Weight decay

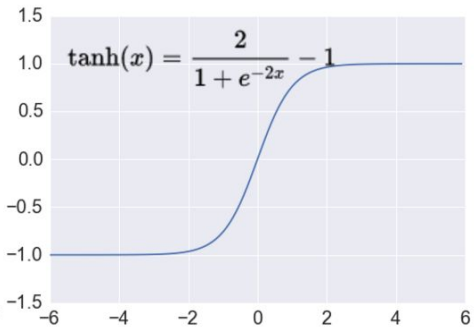
Activation Functions

Transform the output of a layer to a given range.

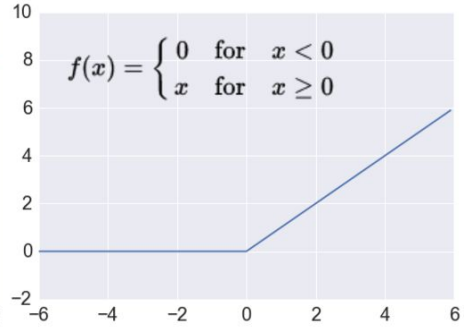
Sigmoid



TanH



ReLU



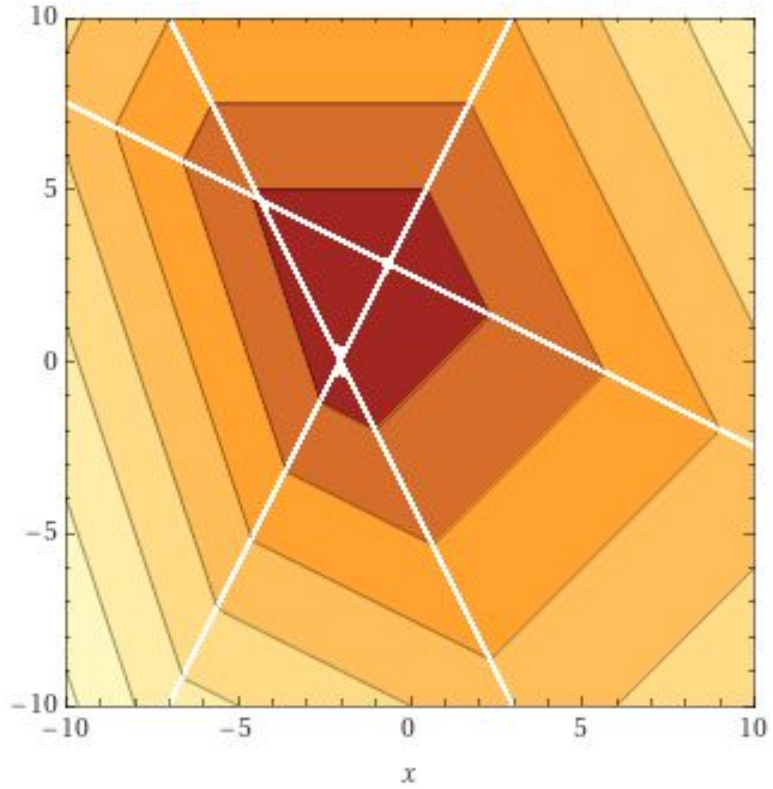
Does not saturate
Does not vanish
Faster computation
May die with high LR

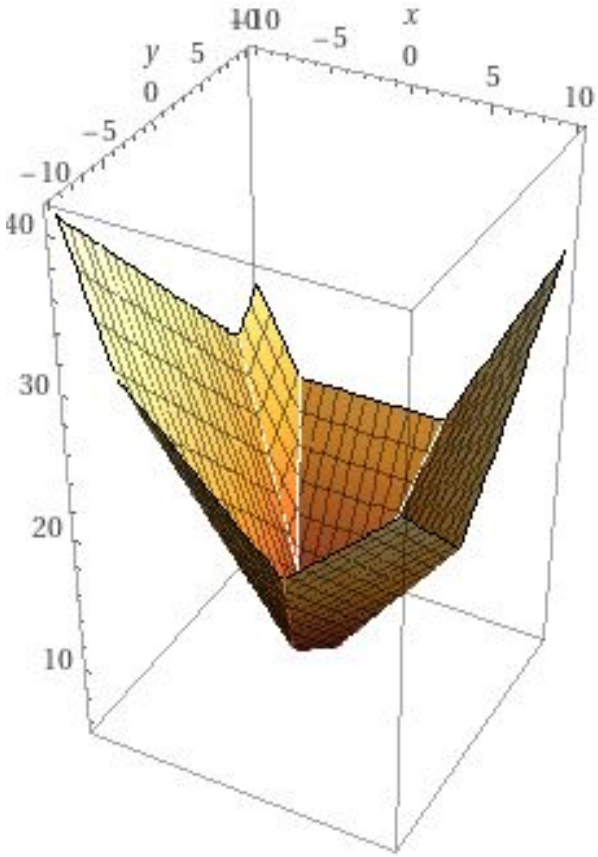
Zero gradient most of f(x): Saturates!
Gradient is 0.25 or 1 max. Vanishes!

ReLU is a safe choice in most cases
Undying alternatives: Leaky ReLU, PReLU, ELU, SELU, ...

ReLU: The linear non-linearity

ReLU(-4-2x+y) +
ReLU(4+2x+y) +
ReLU(5-x-2y)



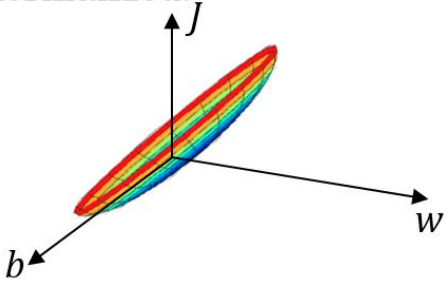


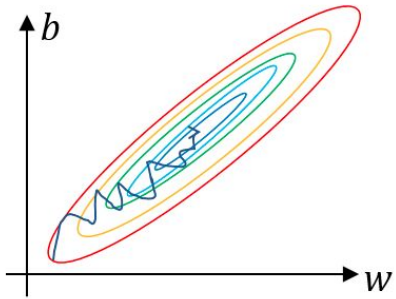
[30]

Input Pre-processing

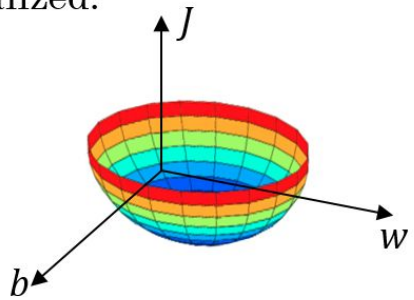
Make your model's life easier. She would do it for you.

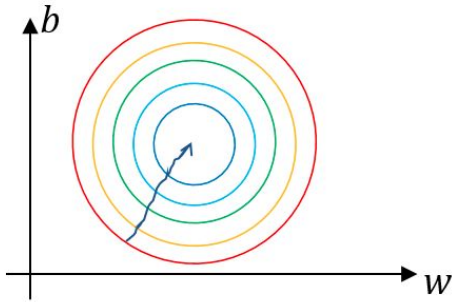
Unnormalized:





Normalized:





Options

- Mean substraction
- Normalization
  - - mean / std
  - Image-wise?
  - Channel-wise?

Weight Initialization

We want:

- Small numbers
- Half positive, half negative

What about bias?

- If weights are properly initialized, bias can be zero-init.

Options:

- ~~Constant values~~: No symmetry breaking
- Normal distribution sample: Ok for shallow, but deviation grows with size
- Glorot/Xavier: Normalize variance by number of inputs
- He/Kaming/MSRA: Specific for ReLUs.

Practical Tips IV

- Start with ReLUs. Explore variants if possible.
- Always zero-mean the data. Or normalize (init depends on it)
- If using ReLUs, *He* init. Otherwise *Glorot*.

Hyperparameters incomplete list #2 (initialization & preprocessing)

5. Activation function
6. Input normalization
7. Weight Initialization

Regularization

Why do we need regularization?

- **Generalization:** Difference between *Machine Learning* and *Optimization*
- We want to learn the “good” patterns
- A neural net will always go for the “easy” patterns
- Generalization is a sweet spot that may be unreachable

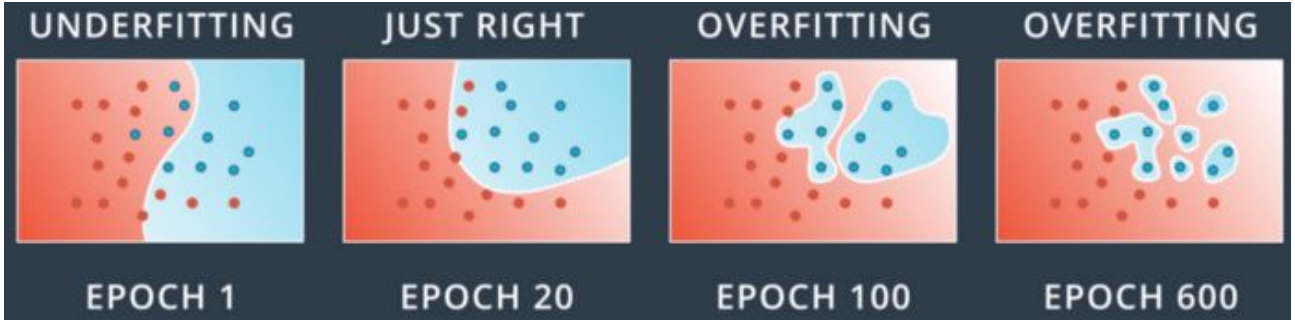
From underfit to overfit

Underfit: Insufficient learning of training data patterns

Overfit: “Excessive” learning of training data patterns ⟶

Key players:

- Model capacity
- Data input
- Regularizers



Train, Test and Val

Doing a good train/val/test split is not easy! Take your time & do it right.

Training set/split

- Data used by *the model* for learning parameters
- Keep an eye for variance (spurious patterns)
- As large and varied as possible
- Use mostly as a sanity check
- Overfitting is inevitable

Validation set/split

- Data used by *you* for tuning hyperparameters
- Size entails reliability
- Overfitting is possible

Test set/split

- Hide under a rock
- Must be 100% independent
- Run once. Cite forever.

Practical Tips V

Never, ever, ever

- Mix correlated data in train/val/test
- Process data in an order
  - Shuffle with seed!
- Believe train results generalize
- The dataset is free of bias

Practical Tips VI

Training milestones

1. Learn, **anything**! (Train set)
   - Little capacity makes it easier
   - Rough hyperparameter estimation
   - *Goal*: Underfit
     i. Better than random

2. Learn, **everything**! (Train set)
   - Growing capacity
   - Hyperparameter refinement
   - *Goal*: Overfit

3. Learn **the right thing** (Val set)
   - Regularization
   - *Goal*: Fit

Back to Regularization

Takes us from overfit to fit

The must do ones (if overfitting):

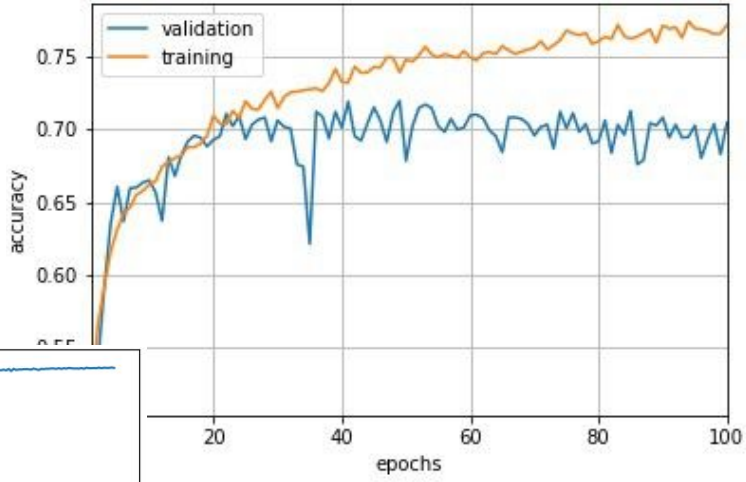
- **Early Stopping**
  - Overfitting is the end of the road
  - The guide: Validation loss/accuracy
  - Enough to understand the model (mind the footprint)

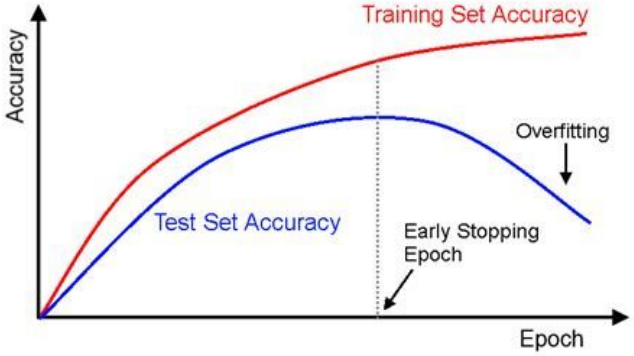
- **Data Augmentation**
  - More data for free
  - As any data preprocessing, think thoroughly!

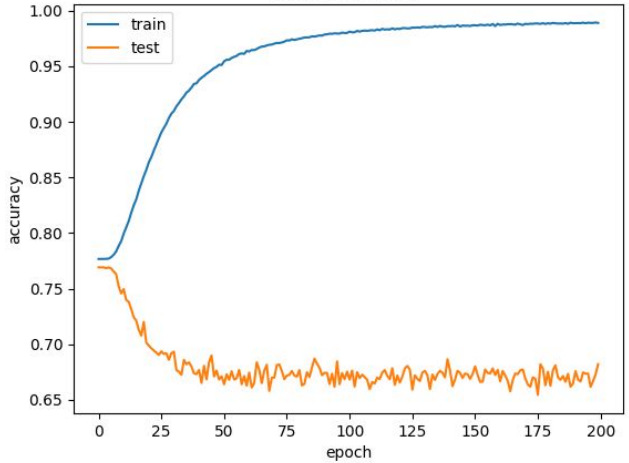
Practical Tips VII

Diagnosing the curves (loss & acc.*)

- Random performance?
- General trend?







Parameter Norm Penalty methods

Add a regularizing term to the loss function

- **L2/L1 norm** on weights by factor (another hyperparam!)
- Makes gradients and weights smaller (L1 sparsifies inputs)
- Safe for SGD, not so for adaptive learning optimizers (e.g., Adam)
  - Uneven normalization :S

Weight decay is similar to L2-norm (often confused)

- Scaling factor on the update rule (another hyperparam!)
- Analogous to L2-norm for SGD, not for adaptative optimizers
- Theoretically safe for all (if implemented!)

[10,11,12]

Max-Norm

Limits the magnitude of the weights vector

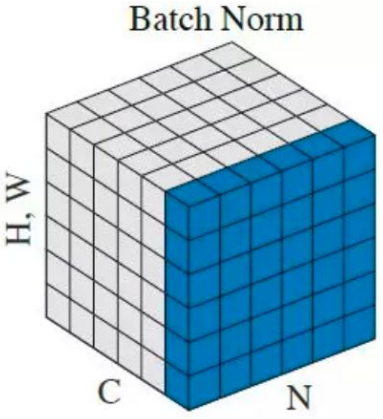
- Constant c (another hyperparam!)
- Typically around 3 or 4
- Goes well with dropout
- May be redundant with L2-norm/weight decay

[10,11,12]

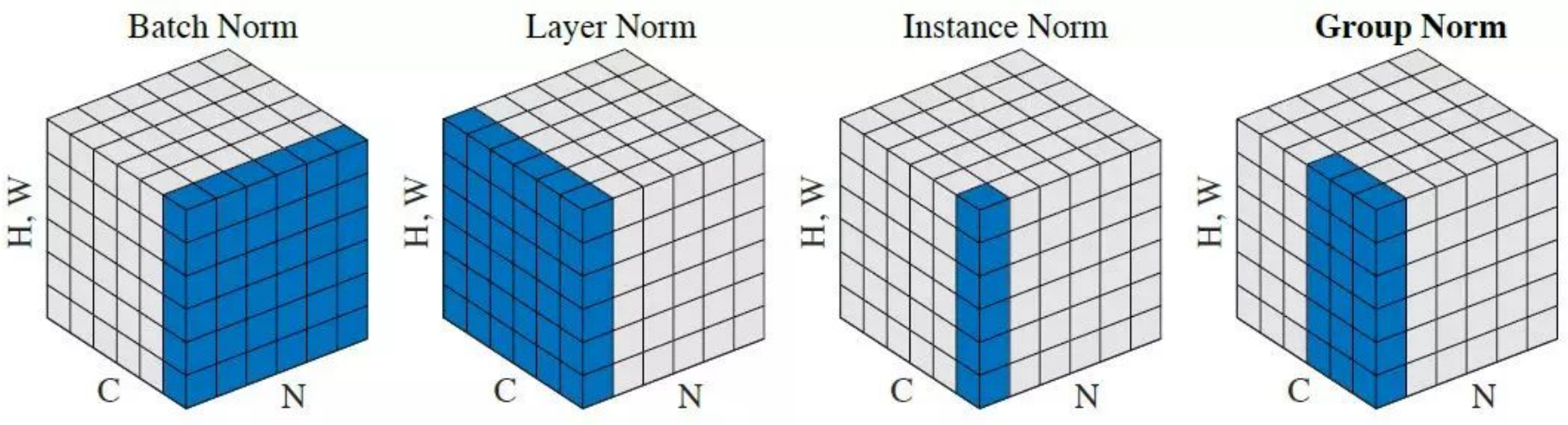
Batch Normalization

Force the *activations* within a normal distribution

- Applied between neurons and activation functions
- Statistics computed per mini-batch (practical reasons)
- On inference, use population of mini-batch statistics
- Helps with initialization and regularize (avoid the radicals)
- Allows higher LR
- Faster convergence
- Requires minimum & fixed batch size



Norms, Norms, Norms



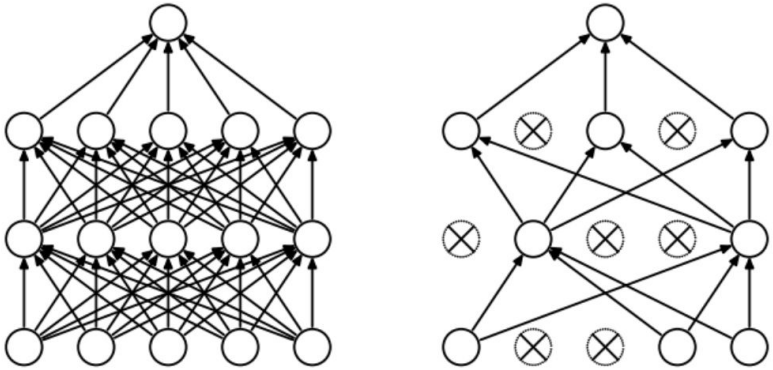
All instances	One instance	One instance	One instance
One var	All vars	One var	N vars

And more!!

Dropout

Cut-off neuron inputs with a given probability

- Dropout rate (typically between 0.2 and 0.5) applied on every step
- In practice trains an ensemble of networks with “shared” params
- Inference: Use all inputs, scaled by the same probability
- Reduces co-adaptation between neurons
- Slows down training (a lot)
- Affects many other hyperparameters
- Good on FC layers. Not on Convs.



(a) Standard Neural Net (b) After applying dropout.

Practical Tips VIII

Experimental cycle

1. Analysis
   - What is wrong/improvable?
   - How can it be solved/achieved?
2. Test
   - Which alternative works better?
   - Ablation study: Alone or combined?

Underfitting

- ❑ Initialization
- ❑ LR, batch size
- ❑ Complexity up

Overfitting

- ❑ Regularization
- ❑ Complexity down

Learning what?

Loss/Cost/Objective/Error function defines the optimization goal

- N-way Classification
  - Softmax (outs N probabilities) + Cross-Entropy (in N neurons)
- Regression
  - Mean Squared Error (in 1 neuron)
- And so many others!

Practical Tips IX

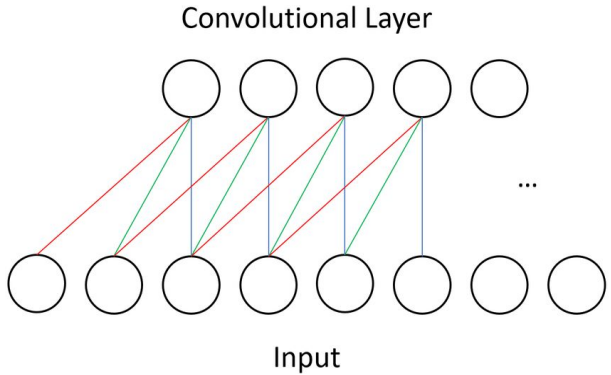
Hyperparameters incomplete list #3 (capacity, regularization and loss)

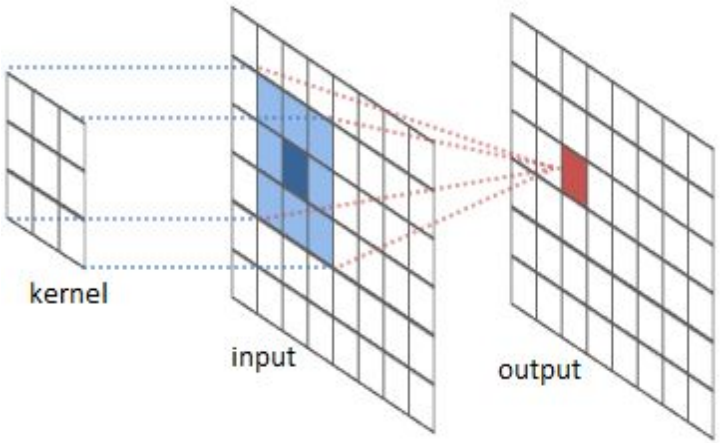
8. Network *capacity* (layers, neurons)
9. Early stopping policy
10. Data Augmentations
11. Normalization layers
12. Dropout rate
13. Loss function

Convolutional Neural Networks

Spatial Connectivity

- Some data has spatial correlations that can be exploited
  - 1D, 2D, 3D, …
- Near-by data points are more relevant than far-away.
- Sparsify connectivity to reduce complexity and ease the learning





Weight Sharing by Convolution

Sparse connectivity is nice, but we still want to apply filters everywhere.

Each filter will get convolved all over the image, generating a number of activations

In practice we have sets of neurons sharing weights

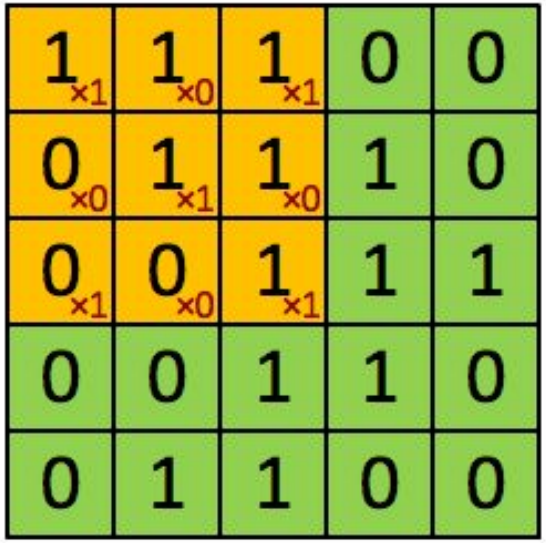
Notice each kernel generates a 2D matrix of values.

Now, what is a neuron?

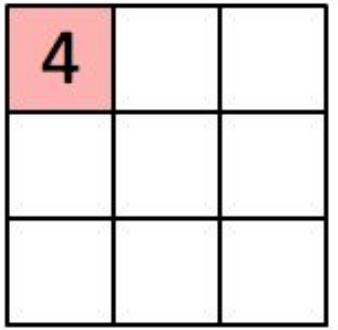
Convolution in Action

Kernel size 3x3
(neuron input = 9)

1 0 1
0 1 0
1 0 1



Image



Convolved Feature

Image Transformations

Convolutions transform the image

Let the model learn the kernels it needs

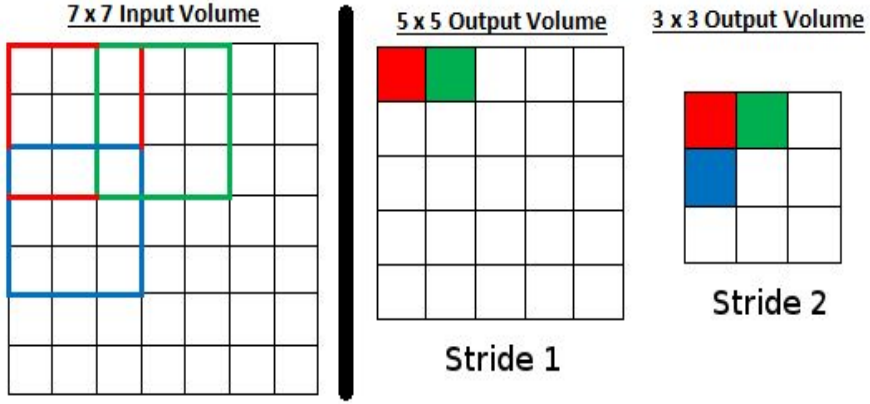
Edge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Gaussian blur 3 × 3	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Convolution Details

Kernel size: Size of the receptive field of convolutional neurons

Stride: Steps size of convolution

Padding: Allows focus on border



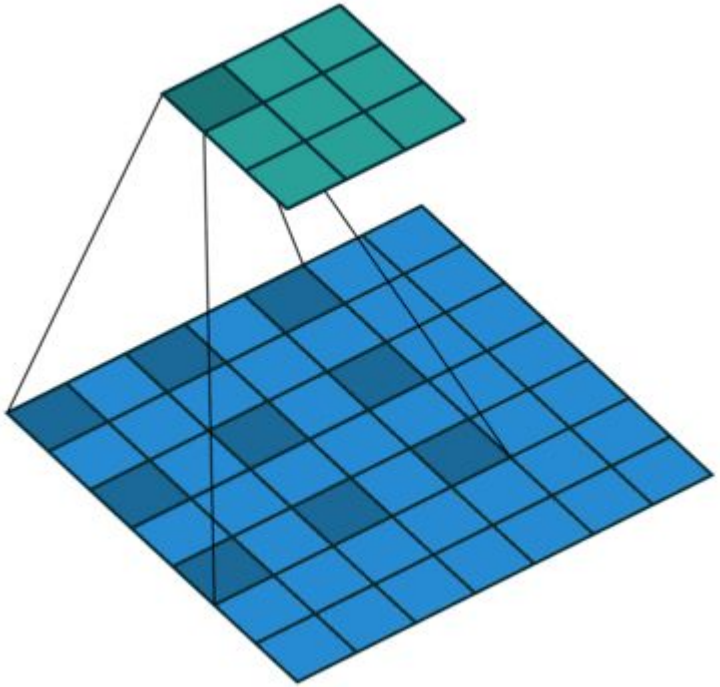
- Most common fill: Zeros
- Valid (no padding): Internal only. May skip data. Reduces dimensionality
- Same: Keep dimensionality with stride 1

$$OutputSize = \frac{InputSize - KernelSize + 2 * Padding}{Stride} + 1$$

Dilated/Atrous Convolutions

Sparsify the kernel

- Increases perceptive field without added complexity
- Loses details, gains context
- Another hyperparam :(
- Used for
  - Down/Upsampling (segmentation)
  - High Resolution inputs



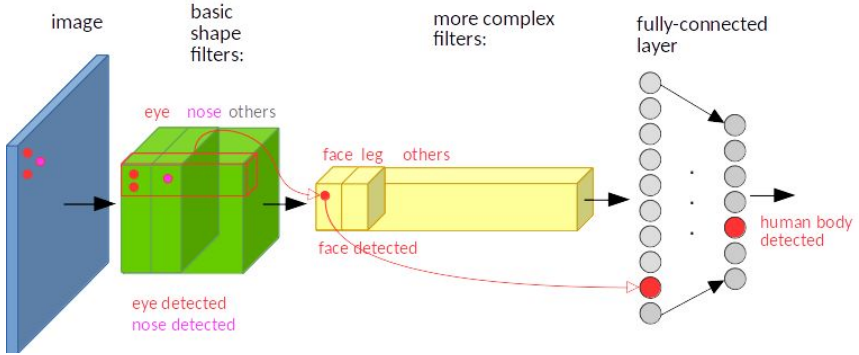
Output Volumes

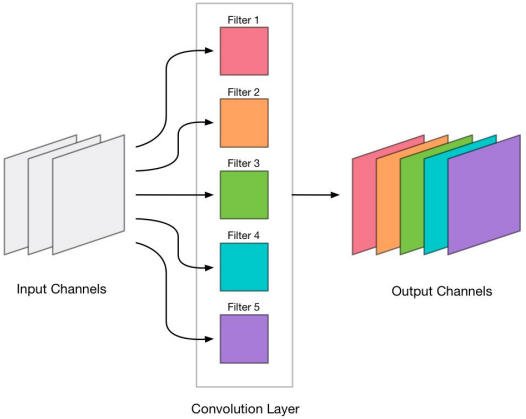
Typically, conv filters are full depth (**N*N*input_depth**)

Each 2D-conv filter (actually 3D) convolved generates a 2D plane of data

Depth provides all the neural views on a part of the input

Output volume: New representation of input with different dimensions



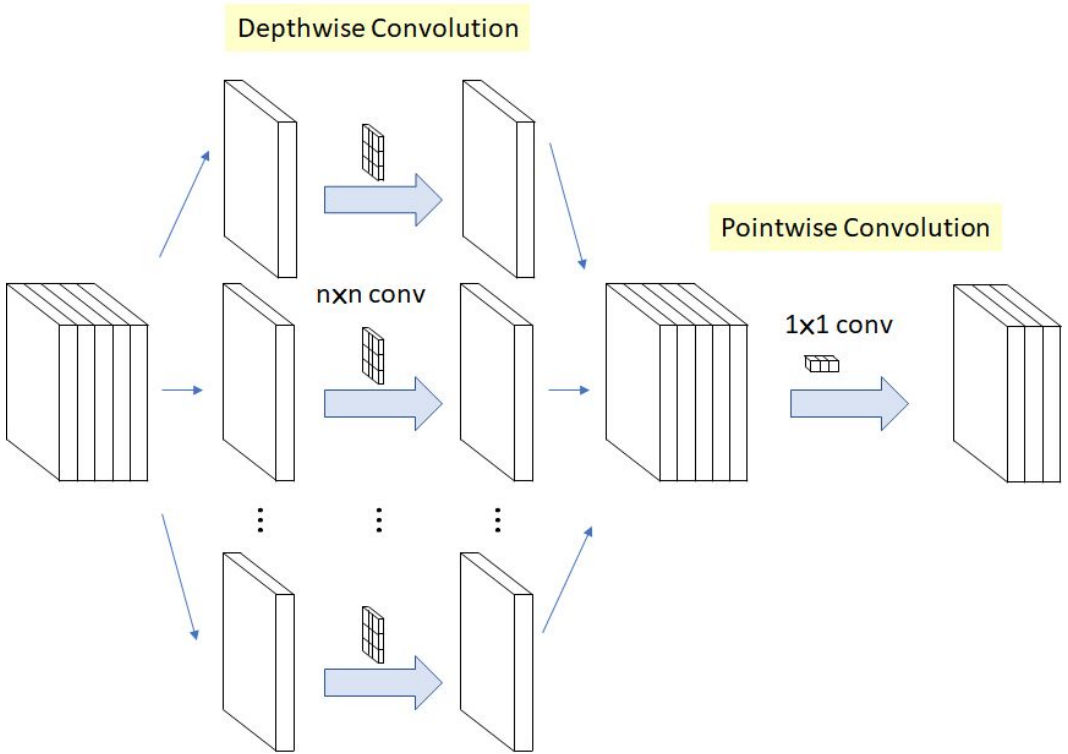


Depth-wise Separable Convolutions

Decreasing the complexity and cost of convolution

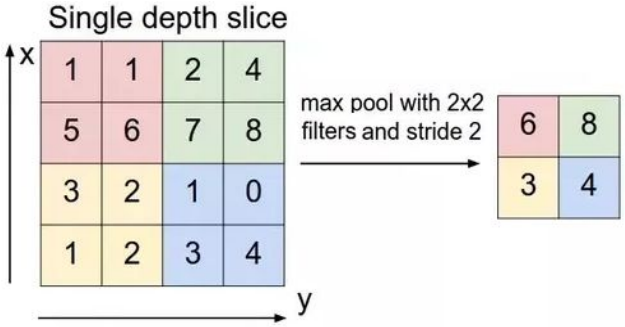
1. Depth-wise convolutions
   - Filters: **N*N*1**
2. Point-wise convolution
   - Filters: **1*1*input_depth**

Params: **N*N+N**



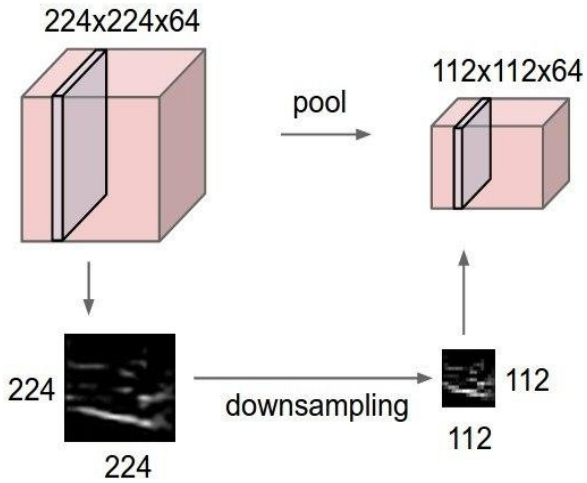
To Pool Or Not To Pool

- Operation: **Max** or Avg
- Dimensionality reduction (along x and y only)
- Never applied full depth!
- Parameter free layer
- Hyperparams: Size & Stride
- Loss in precision



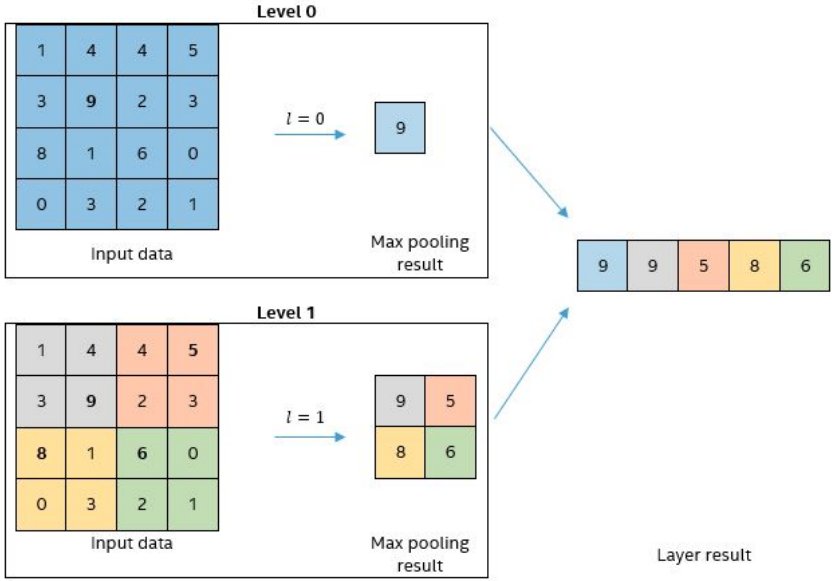
Other means to reduce complexity

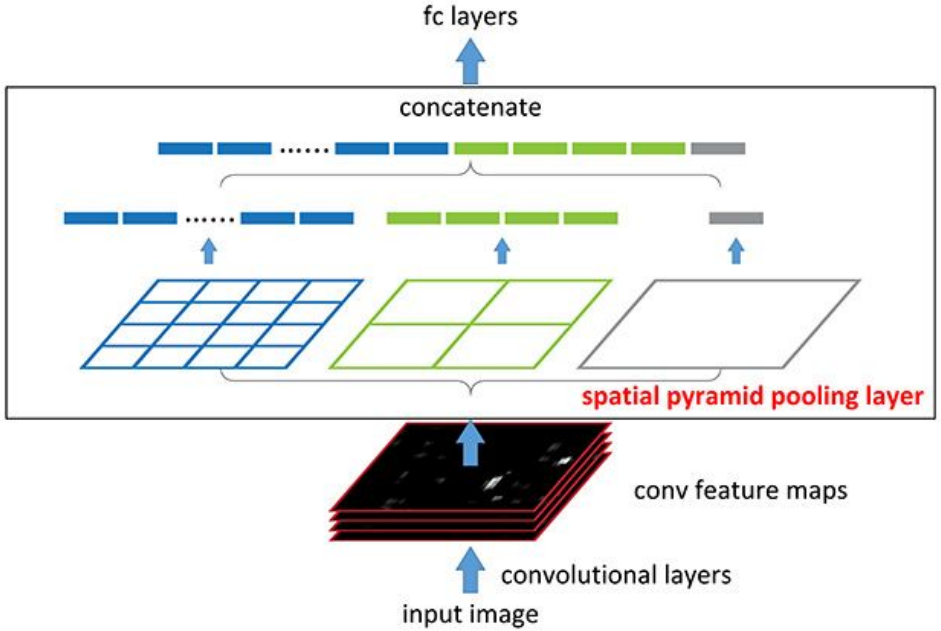
- Depth-wise separable convs, bigger conv. strides



Spatial Pyramid Pooling (SPP)

- Multi-scale Pool (by powers of 2)
- Often used between conv and fc





More alternatives: Atrous spatial PP, Global average pooling, Pyramid pooling module, Adaptive PP

Practical Tips X

Convolutional

- Small/big filters (**3x3**, 5x5, 7x7)
  - Cheap/Expensive
  - Local/General
  - Bigger/Smaller outputs (stride)
- Kernel Size = input size: fc
- Kernel size = 1x1: Alter depth)

Pooling

- 2x2, stride 1 is the least invasive

Hyperparameters incomplete list #4

- ❑ Kernel size (conv & pool)
- ❑ Stride (conv & pool)
- ❑ Padding (conv & pool)
- ❑ Num. filters
- ❑ Dilatation rate

The Challenge

ImageNet Large-Scale Visual Recognition Challenge 2012 (ILSVRC'12)

- Image Classification: 1,000 classes
- Training: 1.2M
- Living things + Human-made objects
  - 120 breeds of dogs

Data Augmentation for CNNs

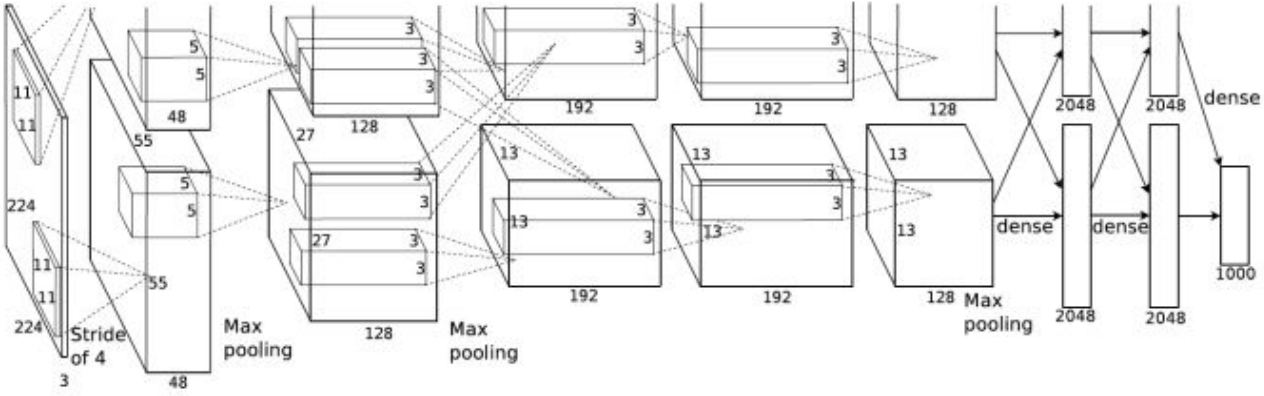
Apply what is safe for each case

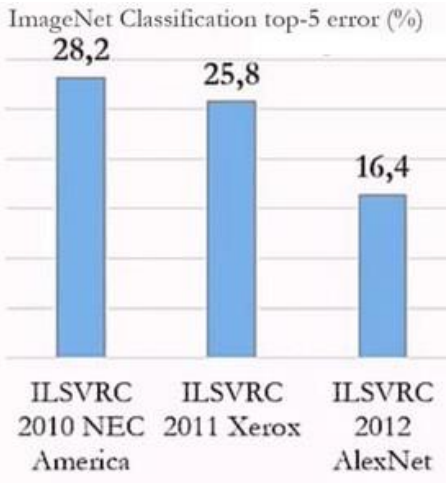
- Horizontal/Vertical flips
- Random Crops
- Color Jitter
- Rotation
- ...

CNNs Big Bang

AlexNet (2012)

- Breakthrough in ILSVRC
- 5 convs+pools, ReLU, 2 dense, and dropout
- 62M parameters



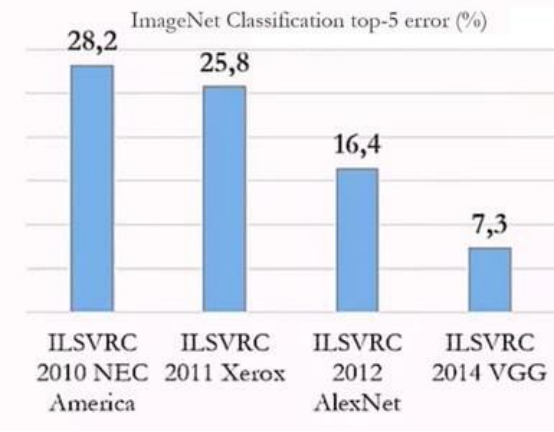


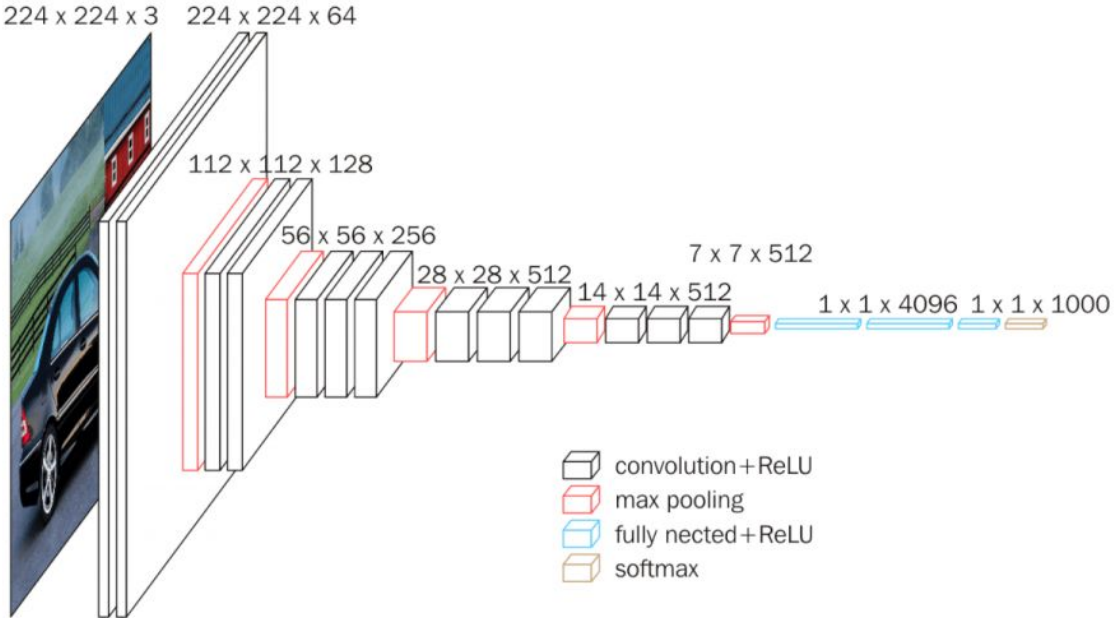
On the shoulders of giants

A new Standard for CNNs

VGG 11/13/16/19 (2014)

- Prototype of **(conv-pool)*+dense*** architecture
- 133-144M parameters
- 3x3 convs only

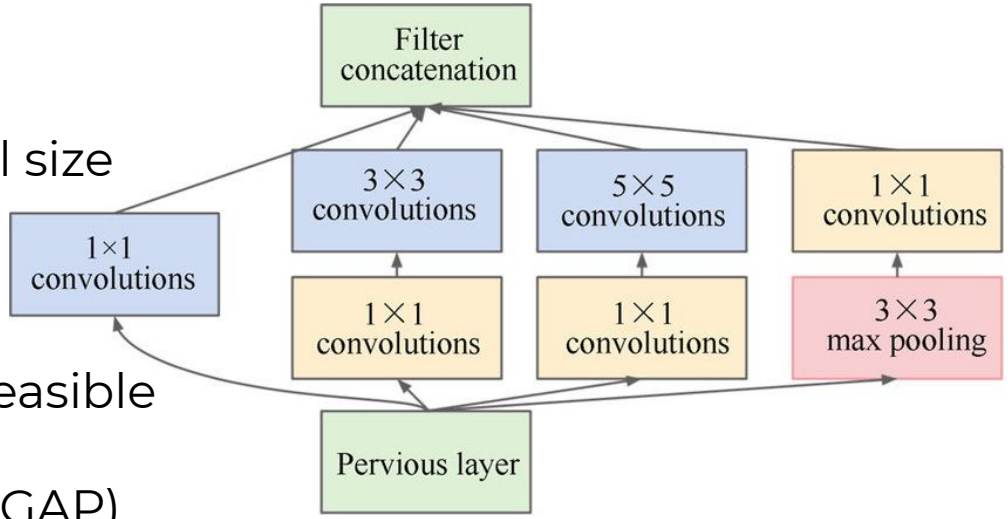




The Inception Family

GoogLeNet (2014)

- The Inception block
- Let the model decide the kernel size
- Better scale adaptation
- Bottleneck 1x1 conv to make it feasible
- No FC: Global Average Pooling (GAP)

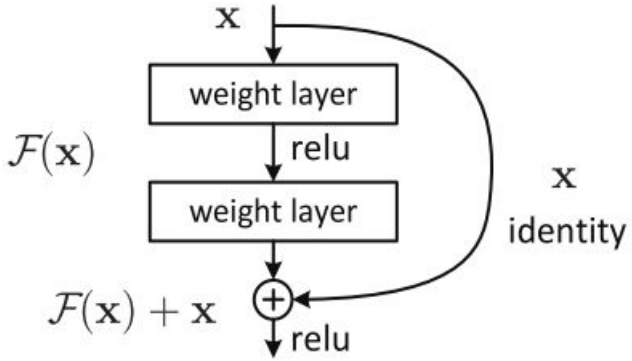


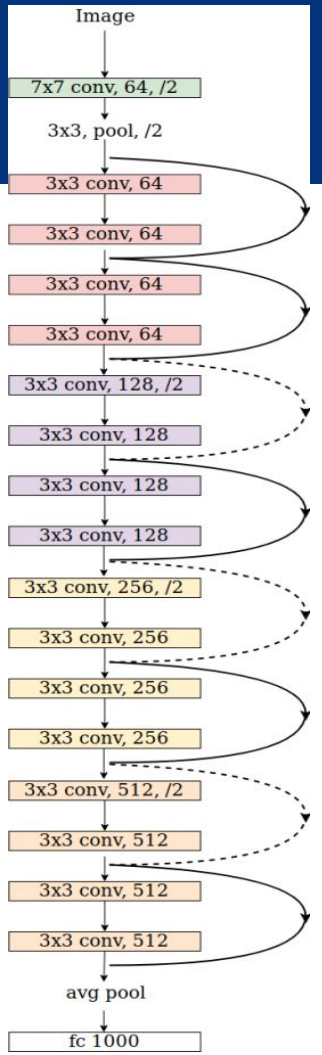
[Inception,15]
[ResNet,16]
[Huang,16]
[Xu,WWW]

The Skipped Connection

ResNet (2015)

- Residual blocks / Skip connections
- Deeper should never be worse
  - Learning the identity is hard
  - Learning to cancel out is easy
- Shallow ensemble of nets
- Train up to 1K layers (do not!)
- ILSVRC'12 human level

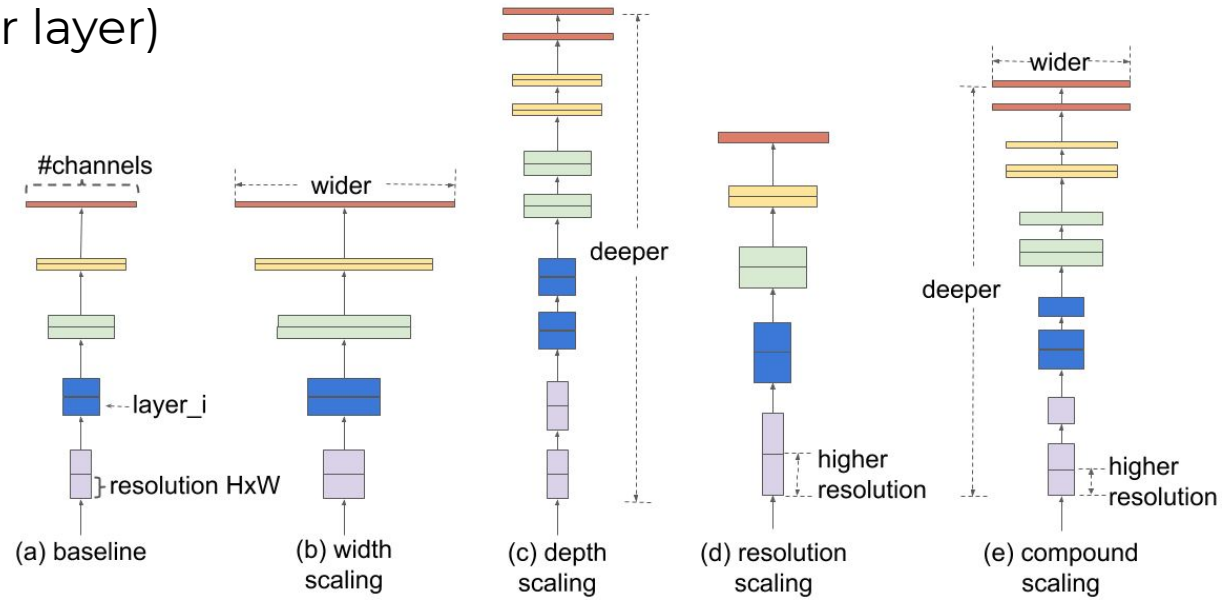




EfficientNet

Should I go deeper, wider or bigger?

- Find a balance between them (they are all related!)
  - Width (neurons per layer)
  - Depth (layers)
  - Resolution (input)
- Choose a size
  - EfficientNetB0-B7



Noisy Student

A semi-supervised training paradigm

1. Train model A (teacher) with the labeled data
2. Use A to generate pseudo-labels for an unlabeled data set
3. Train model B (student) with both labeled and pseudo-labeled data

- Iterate, re-labeling the unlabeled data each time
- Highly regularized (noise!) student to guarantee improvement
- Each student has more capacity than the previous

Visualizing CNNs

The Basics

- NN are representation learning techniques
- CNNs build hierarchically complex features
  - From Gabor filters to dog faces
  - Induced by convolution
  - Tend to focus on the “non obvious for humans”
    - Backgrounds, textures
- The closer to the loss, more classifier (task) and less representation (data)

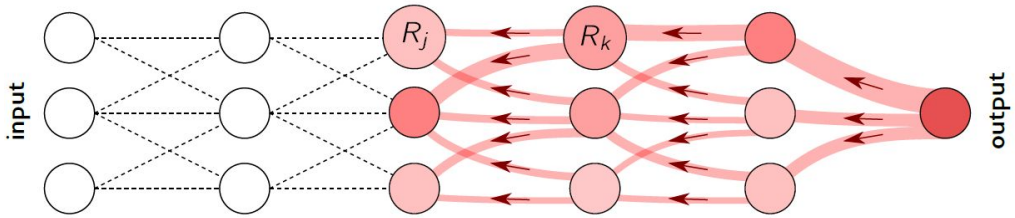
[25,26]

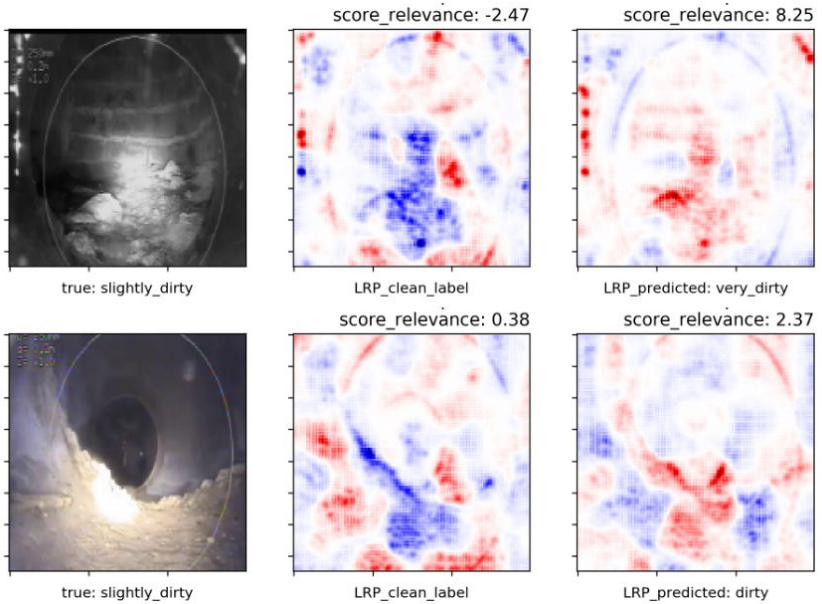
Ways of Looking at CNNs

- Attribution: Where is the network looking?
  - Grounded. Instance based.
  - Explainability in practice
- Feature Visualization: What is the network seeing?
  - Uncontextualized.
  - Diagnosys & Insight
- Exemplification: Which images cause a maximum activation?
  - Samples from a distribution

Attribution

- Finding the importance of pixels
- Layerwise Relevance Propagation (LRP)
  - Backpropagate *an* output. Find the relevance of each neuron
    - Weighted by CNN parameters



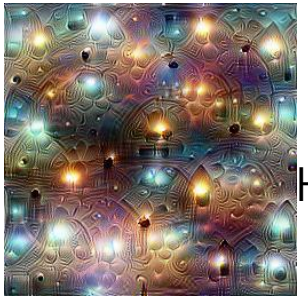


[29]

Feature Visualization

- Optimizing the input to maximize the output
  - A neuron
    - A channel
  - A layer (DeepDream)

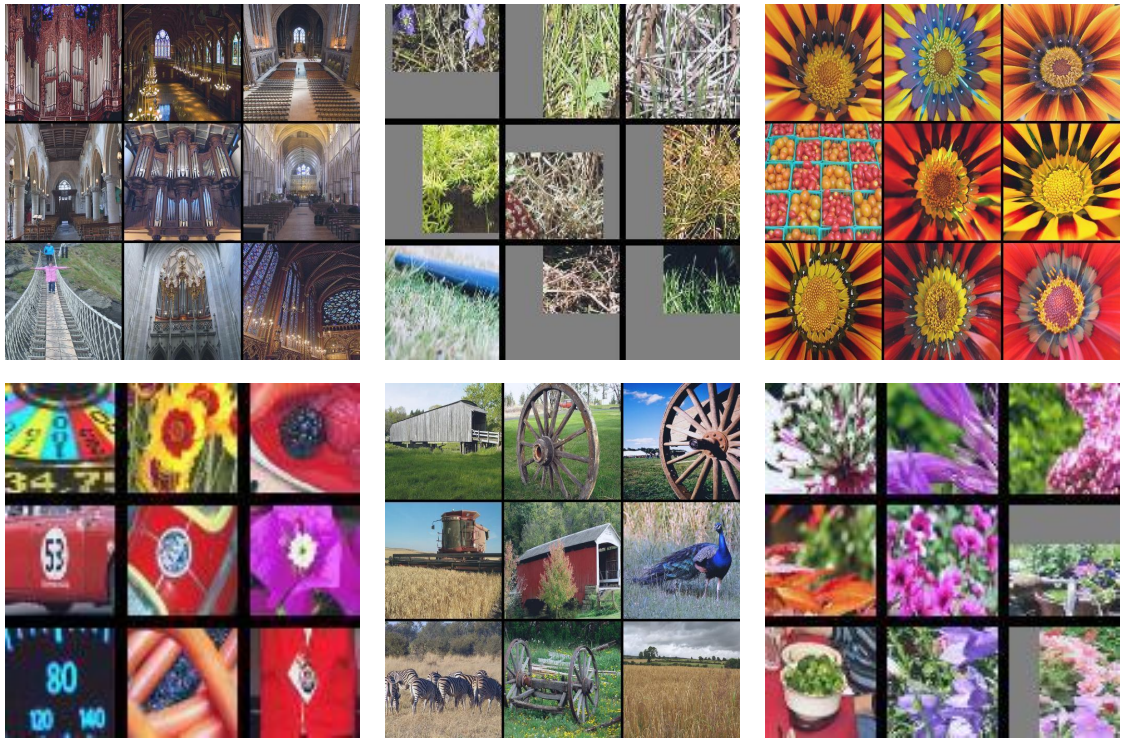
Low level



High level

Exemplification

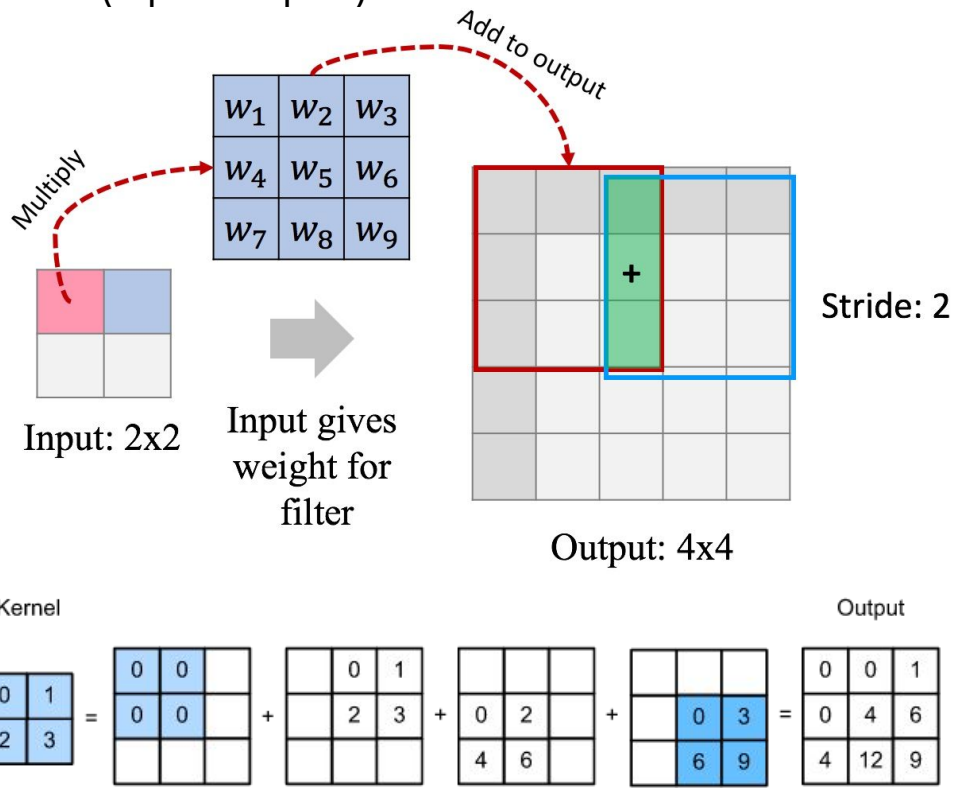
- Finding images within a dataset maximizing outputs
  - Subjective
  - Partial
  - Stochastic



Playing with CNNs

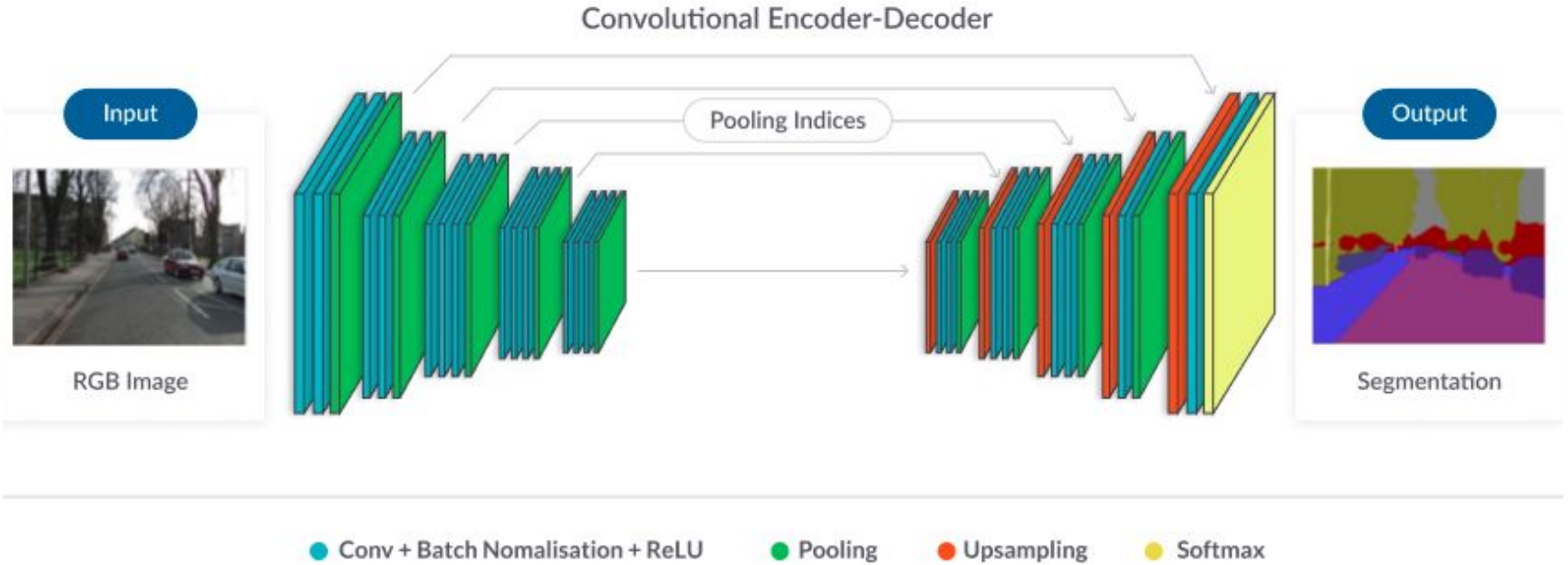
Transposed Convolution ~~Deconvolution~~

- Reverse effect of regular convolution (upsample)
- Learnt interpolation
- Applications
  - Segmentation
  - GANs
  - Super-Resolution
  - Conv. Autoencoders



Encoder-Decoder CNNs

- Pixel-wise classification task (image reconstruction loss)
- Bottlenecking makes it cheaper



[14]

Faster Segmentation

- ~~Pixel-wise classification~~ Object detection (bounding box)
  - Can be done with a “regular” CNN
- R-CNN: Propose crops (SVM). Extract features (CNN). Classify crops (SVM)
- Fast R-CNN: Extract features. Propose crops. Classify/Bounding Box (CNN)
- Faster R-CNN: Propose crops through a specific sub-net (RPN)
- YOLO v? (no regions, faster, less accurate)
  - Divide into grid. Predict class and bounding box for each cell.

Better Segmentation

- Mask R-CNN
  - Faster R-CNN for object detection
  - FCN for instance segmentation (pixel classification)
- Xception
  - Depth-wise separable Convs (inverted order & w/o non-linearity)
  - Skip connections
  - Atrous SPP

[17,18]

Automatic Image Colorization

- Another pixel-wise classification application

(a) Colorado National Park, 1941 (b) Textile Mill, June 1937 (c) Berry Field, June 1909 (d) Hamilton, 1936

[15,16]

Style Transfer

- What do the correlation of activations intra-layer tell us?
  - What if we force it on another image?

- Gram matrix represents the *style*
  - Channel-wise (cXc)
  - Several mid layers
- Activations represents the *content*
  - One mid layer

- Optimize the **input** to minimize 2 losses
- Use a pre-trained net frozen
- Improved and extended

[19,20,21,22]

References

[1] http://vordenker.de/ggphilosophy/mcculloch_a-logical-calculus.pdf
[2]http://www-public.tem-tsp.eu/~gibson/Teaching/Teaching-ReadingMaterial/Rosenblatt58.pdf
[3] http://www.dtic.mil/dtic/tr/fulltext/u2/236965.pdf
[4] https://en.wikipedia.org/wiki/Perceptrons_(book)
[5]http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/
[6] https://en.wikipedia.org/wiki/Perceptrons_(book)
[7] Werbos et al. "Beyond regression:" new tools for prediction and analysis in the behavioral sciences." Ph. D. dissertation, Harvard University (1974).
[8] Rummelhart et al. "Learning Internal Representations by Error Propagation". MIT Press (1986).

References

[9]https://towardsdatascience.com/effect-of-gradient-descent-optimizers-on-neural-net-training-d44678d27060
[10] https://arxiv.org/abs/1711.05101
[11] https://bbabenko.github.io/weight-decay/
[12] https://towardsdatascience.com/weight-decay-l2-regularization-90a9e17713cd
[13] Veit, Andreas, Michael J. Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow networks." Advances in neural information processing systems. 2016.
[14] https://thegradient.pub/semantic-segmentation/
[15] https://arxiv.org/pdf/1603.08511
[16] https://pdfs.semanticscholar.org/5c6a/0a8d993edf86846ac7c6be335fba244a59f8.pdf

References

[17] https://arxiv.org/pdf/1606.00915.pdf
[18] https://arxiv.org/pdf/1610.02357.pdf
[19] https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf
[20] https://arxiv.org/abs/1603.08155
[21] https://arxiv.org/abs/1603.03417
[22] https://ai.googleblog.com/2016/10/supercharging-style-transfer.html
[23] https://arxiv.org/pdf/1903.07291.pdf
[24] http://nvidia-research-mingyuliu.com/gaugan
[25] Geirhos, Robert, et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness." arXiv preprint arXiv:1811.12231 (2018).

References

[26] Beery, Sara, Grant Van Horn, and Pietro Perona. "Recognition in terra incognita." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
[27] https://distill.pub/2017/feature-visualization/
[28] https://distill.pub/2018/building-blocks/
[29] Montavon, Grégoire, et al. "Layer-wise relevance propagation: an overview." Explainable AI: interpreting, explaining and visualizing deep learning. Springer, Cham, 2019. 193-209.
[30] https://medium.com/machine-intelligence-report/how-do-neural-networks-work-57d1ab5337ce
[31] Hebb, D.O. (1949), The organization of behavior, New York: Wiley

References

[32] Dauphin, Yann N., et al. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization." Advances in neural information processing systems. 2014.
[33] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).

Dario Garcia-Gasulla (BSC)

dario.garcia@bsc.es