Deep Learning

Introduction to Machine Learning – GIF-7015

Professor: Christian Gagné

Week 8

8.1 Motivations for deep learning

History of neural networks

- 1957: proposal of the perceptron by Frank Rosenblatt
- 1967: demonstration by Marvin Minsky that the perceptron is unable of processing nonlinearly separable data, disinterest in neural approaches
- 1986: Rumelhart, Hinton and Williams demonstrate the use of gradient backpropagation for the training of multilayer perceptrons
- 1995-2005: development of SVMs, loss of interest in neural networks
- 2006: first deep neural network architectures
- 2012: results for object (Toronto, ImageNet) and speech (Microsoft) recognition demonstrate the potential of deep learning as a disruptive technology
- 2014: explosion of private investment in machine learning, especially in deep learning

Emergence of deep networks

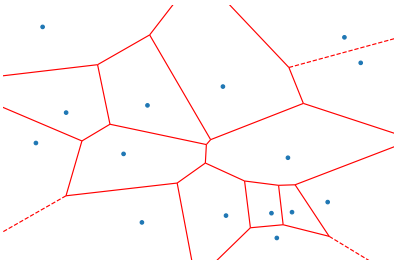
- Conditions that allowed the emergence of deep networks:
  1. Availability of very large datasets (*big data*)
  2. Availability of massive computing capacity (GPU)
  3. New very flexible learning models, with priors that deal better with the curse of dimensionality

Model composition

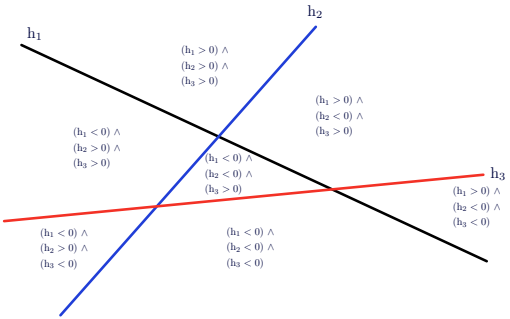
- Model compositionality is necessary in machine learning
  - Such as for language, we need to compose elements to define a language that gives meaning to complex notions
- Exploiting compositionality allows an exponential gain in representation power
  - Distributed representations, feature learning
  - Deep architectures: several levels of representation learning
- Model composition is useful to describe our world effectively

Local vs. distributed representation

- Set of distributed (not mutually exclusive) discriminants is exponentially more statistically efficient than local representations (k-nearest neighbours, clustering)



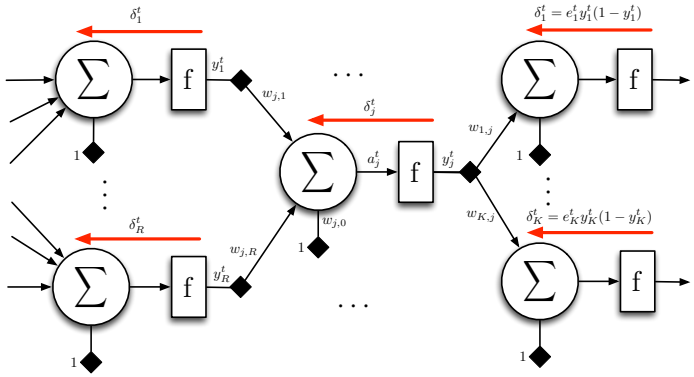
Local representation



Distributed representation

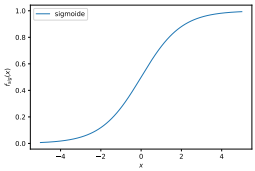
Vanishing gradient problem

- Multilayer perceptron training of more than two hidden layers with backpropagation does not work well
  - Saturated neurons, with very low gradient
  - *Vanishing gradient* from layer to layer

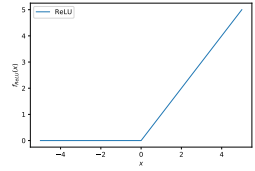


Transfer functions

- Sigmoid function
  - Probabilistic interpretation
  - Approximation of a *step* function (binary)
  - Gradient saturation problem



- Transfer functions must include nonlinearities
- ReLU function (*Rectified Linear Unit*), $f_{\mathrm{ReLU}}(a) = \max(0,a)$
  - Simple transfer function model with nonlinearity
  - Composition of ReLUs allows piecewize linear approximations
  - Biological motivation of deep networks with ReLU (*leaky integrate-and-fire model*)
  - Training deep networks with ReLU possible without unsupervised pre-training



Network depth

- Deep networks, when well trained, learn better than (*fat networks*)
  - Network capacity grows linearly with the width of a layer, exponentially with the depth of the network

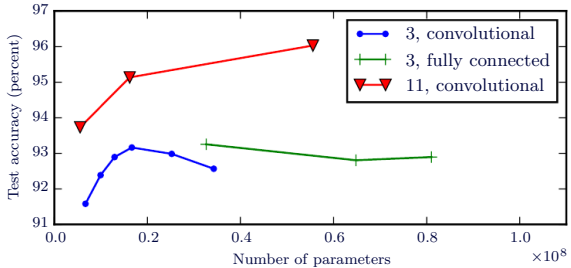


Figure 6.7 from *I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.* Accessed online on October 19, 2020 at `https://www.deeplearningbook.org/contents/mlp.html`.

- Fat networks overfit with 20M parameters, deep networks work well with 60M parameters

8.2 Autoencoders and RBM

Unsupervised pre-training

- Deep networks before 2011: unsupervised pre-training required
  - Random initialization of deep networks generates a wide variety of sub-optimal solutions (local minima)
  - Unsupervised pre-training allows to start the backpropagation in a good configuration (basin of attraction)

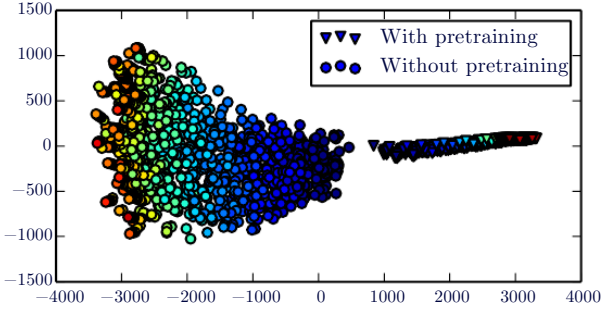
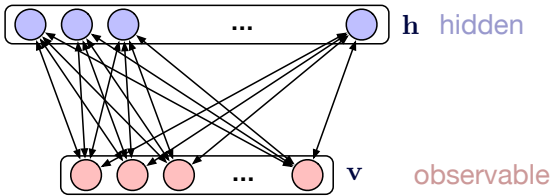


Figure 15.1 from *I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.* Accessed online on October 19, 2020 at `https://www.deeplearningbook.org/contents/representation.html`.

Restricted Boltzmann machine

- Restricted Boltzmann machine (RBM): generative neural network model
  - Can learn from distributions on input data
  - Layer of visible neurons (**v**) and hidden neurons (**h**)



- **h** and **v** are binary, the model can calculate $P(\mathbf{v},\mathbf{h})$, $P(\mathbf{v})$, $P(\mathbf{v}|\mathbf{h})$, $P(\mathbf{h}|\mathbf{v})$
- Used to learn *deep belief network*, with unsupervised learning through RBM layers, followed by error backpropagation refinement (supervised)

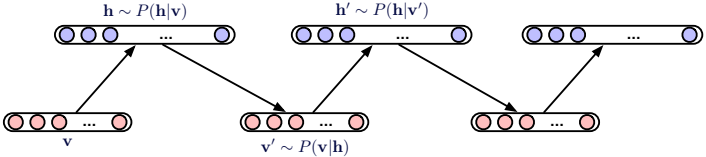
Restricted Boltzmann machine

- RBM energy function

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{a}^\top \mathbf{v} - \mathbf{b}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W}\mathbf{h}$$

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp[-E(\mathbf{v},\mathbf{h})]$$

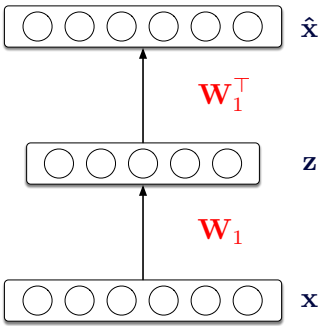
- Z is a partition function, allows to normalize the values so that the probabilities sum to 1
  - Calculation of Z, and thus $P(\mathbf{v},\mathbf{h})$, is intractable ($Z = \sum_{\forall \mathbf{v},\mathbf{h}} \exp[-E(\mathbf{v},\mathbf{h})]$)
  - Solution: Gibbs sampling (Monte Carlo)



- Computationally very heavy
  - RBM very rarely used nowadays as deep networks

Autoencoders

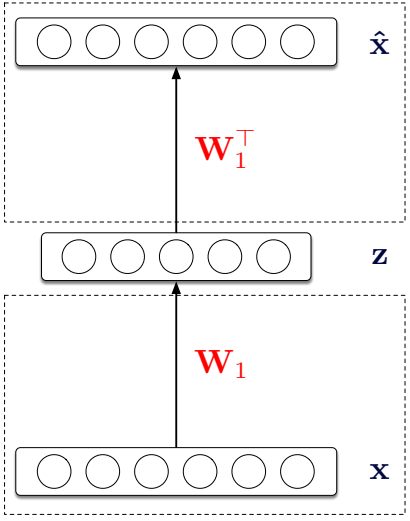
- Autoencoder: model allowing to compress the input (encoder) and decompress it (decoder).
  - Objective: compress while keeping the error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$ low
  - Decoder weights linked to encoder weights (usually transposed)



Autoencoders training

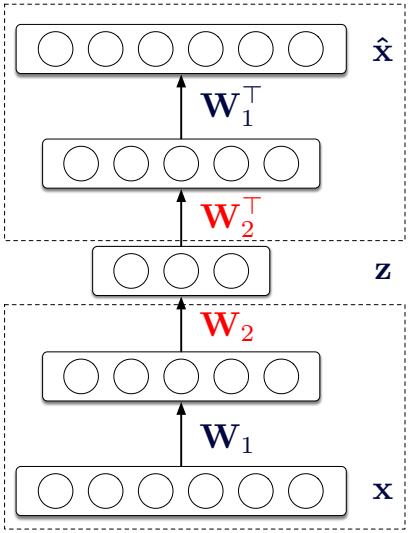
- Unsupervised training of the autoencodeur, to learn representation
  - Encoder used to extract a compact representation
- Greedy training, one layer at a time
  - Training of the outermost layer
  - Addition of a new layer, which is driven individually, with the outer layer being fixed, and so on
- Nonlinear transfer function between layers
  - Necessary, otherwise several linear layers could be simplified into a single layer
  - Weight learning by gradient descent
- Output layer added to encoder, with supervised training
  - Complete backpropagation training of the output layer
  - Adjustment of encoder weights by backpropagation (*fine-tuning*)

Autoencoder training example



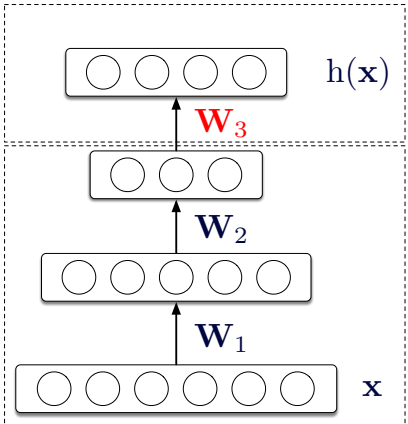
- Unsupervised weight training $\mathbf{W}_1$, weight $\mathbf{W}_1^\top$ linked
- Minimize error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- Intermediate representation in central values

Autoencoder training example



- Addition of two new layers (one in encoder and one in decoder)
- Unsupervised weight training $\mathbf{W}_2$, weight $\mathbf{W}_1$ fixed
- Always minimizes error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- New intermediate representation
- Can be repeated like this on several layers

Autoencoder training example



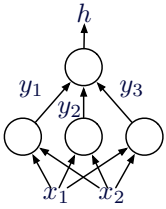
- Removal of the decoder part of the network
- Adding an output layer, with as many outputs as classes
- **Supervised** training of $\mathbf{W}_3$ by backpropagation
- Weights $\mathbf{W}_1$ and $\mathbf{W}_2$ also often fine-tuned by backpropagation

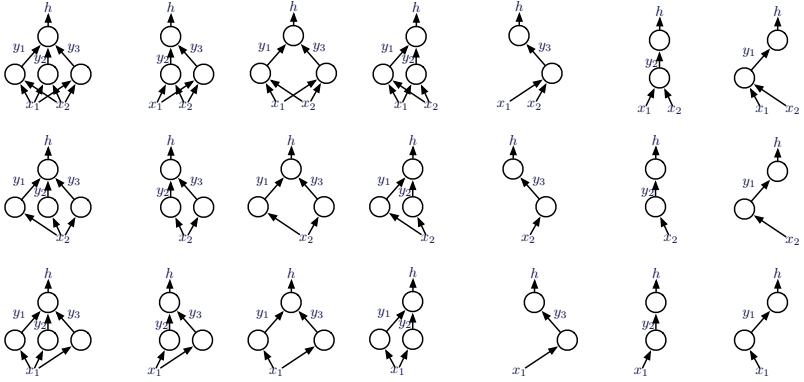
8.3 Dropout and batch normalization

Dropout

- Dropout: training method that consists in randomly deactivating neurons
  - Typically, half of the neurons in the hidden layers (80 % of the inputs) are activated at the presentation of each data during training
  - Random masks to select active neurons, a different one for each presentation
- Does a regularization of the network
  - Forces the learning of a representation distributed throughout the network
  - Makes it difficult for “grandmother cells” to emerge
  - Has proven to be very effective in improving the performance of deep networks
- Evaluation of new data in test by averaging over several selection masks
  - Analogy with ensemble learning (seen later in the semester), in particular to bagging

Dropout





Batch normalization

- Modification of a weight by backpropagation based on a local gradient
  - Weight of previous and following layers are also modified!
- *Batch normalization*: normalize activation of neurons between all the data of a mini-batch
  - Mini-batch: small subset of data instances from the training set (typically a few hundreds)
- Activation of neurons $\mathbf{H}$ normalized according to

$$\mathbf{H}' = \frac{\mathbf{H} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \quad \boldsymbol{\mu} = \frac{1}{m}\sum_i \mathbf{H}_{i,:}, \quad \boldsymbol{\sigma} = \sqrt{\epsilon + \sum_i (\mathbf{H} - \boldsymbol{\mu})_i^2}$$

  - $\mathbf{H}$: activation of neurons (row) of a layer for the data of the mini-batch (column)
  - ϵ: small value (typically 10^{-8}) to avoid division by zero when variance is zero

8.4 Text and image processing

Text processing

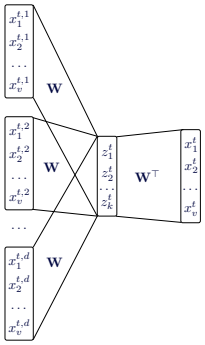
- How to give documents (sequence of strings) to a neural network (vector of fixed-size real values)?
- *Bag-of-Words* model (BoW)
  - Identify a dictionary of the most frequent / interesting words
  - Calculate the frequency of each word for each processed document (vector of integers of fixed size)

$$\mathbf{x}^t = [x_1^t, x_2^t, \ldots, x_v^t]^\top$$

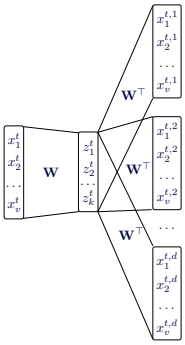
    where x_i^t is the number of occurrences (integer value) of the $i-$th word (according to the dictionary) in the document
- Does not take into account word order
  - Models with N-gram measures the frequency of adjacent word groups
  - Skip-gram: related words may not be adjacent

Neural representations for word processing

- Take into account the sequence to encode the text
  - *Continuous BoW*: predict the word according to those before and after (faster)
  - *Continuous skip-gram*: predict the words before and after according to the word of interest (more precise)



Continuous BoW

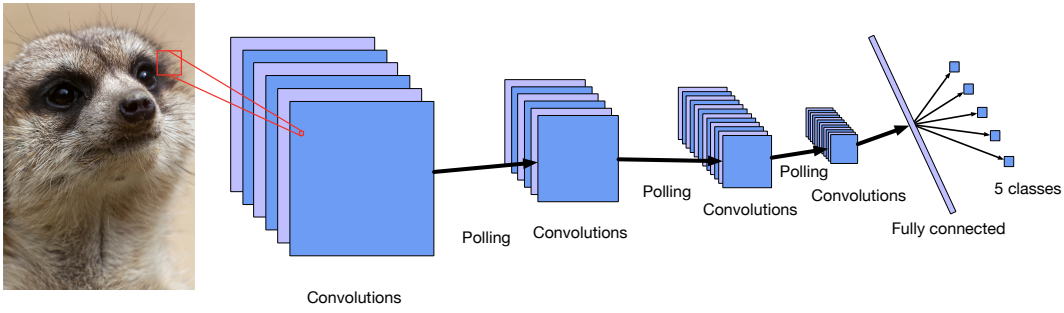


Continuous skip-gram

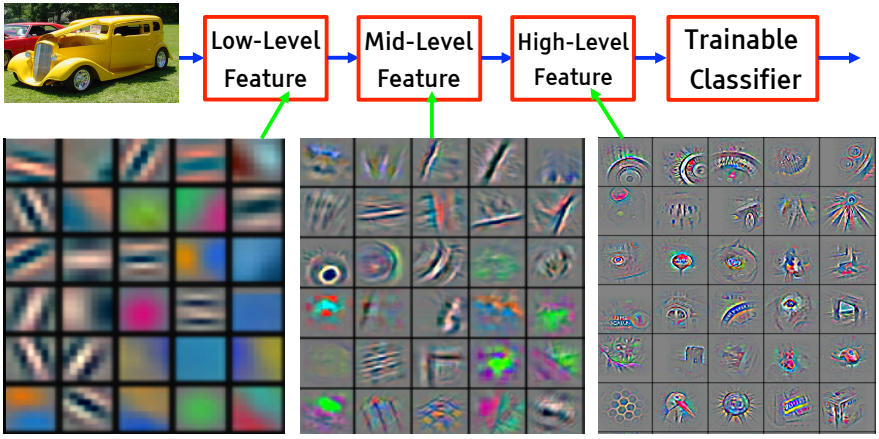
Convolution network

- Convolution network: processing temporal or spatial signals
  - Time signal: sound and speech
  - Spatial signal: image
- Convolution layer: filters convoluted on temporal/spatial data
  - Data can be network input values or outputs from previous layers
  - Convolution on each channel (multiple channels is possible)
  - Learning the filters by backpropagation
- *Pooling* layer: values selection (maximum of a window)
  - Allows to reduce the size of the values, otherwise the size of the model explodes!
- Fully connected neurons output for decision-making
- Next presentation in the course on that topic

Convolution network



Filters composition



From *G. Hinton, Y. Bengio and Y. LeCun, Deep Learning NIPS'15 Tutorial, 2015.* Accessed online on October 19, 2020 at `https://media.nips.cc/Conferences/2015/tutorialslides/DL-Tutorial-NIPS2015.pdf`.

Objects recognition

- *ImageNet Large Scale Visual Recognition Challenge*: recognize objects in an image (1000 classes), giving the right class in a top 5

ILSVRC 2012	
Team	% error
SuperVision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE / INRIA	27.0
UvA (Amsterdam)	29.6
INRIA / LEAR	33.4

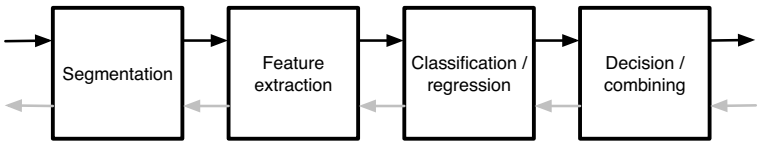
ILSVRC 2013	
Team	% error
Clarifai	11.7
NUS	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

ILSVRC 2014	
Team	% error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

8.5 Representation learning

Representation learning

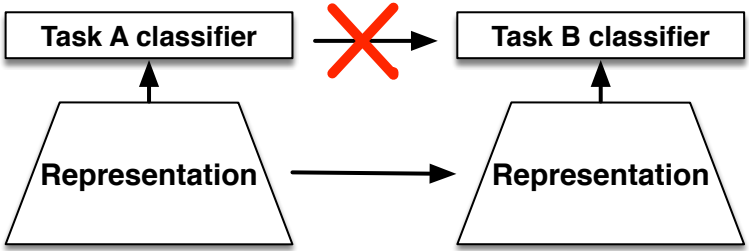
- Classic pattern recognition pipeline



- In the past, each module was designed independently
- Deep learning allows learning of the representations
  - Learning of all modules simultaneously
  - Ability to retrieve representations (segmentation, feature extraction) and use them with other classification and decision making modules

Transfer of representations

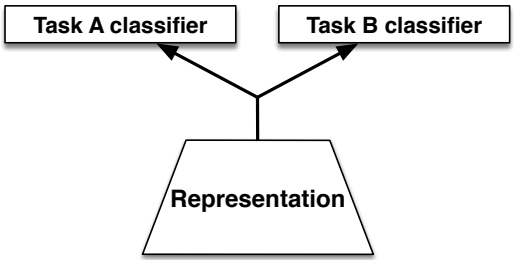
- Learning a deep network on task A
- New task B, based on data similar to task A
  - Retrieve task A representation
  - Train new classifier for task B



  - Allows a transfer of representation (*transfer learning*)

Multitask learning

- Multitask learning: *simultaneously* learning a representation for separate operations
  - Two-headed network, one for each task



  - Backpropagation comes from one head at a time
  - Mixing data and tasks during learning
  - Good performance for producing representations capturing general concepts

8.6 Image generation

Generation of examples

- Idea: generate input data based on a desired output
  - Generate a model of the data that can produce the output according to the neural network
- Approach: gradient descent on the input data

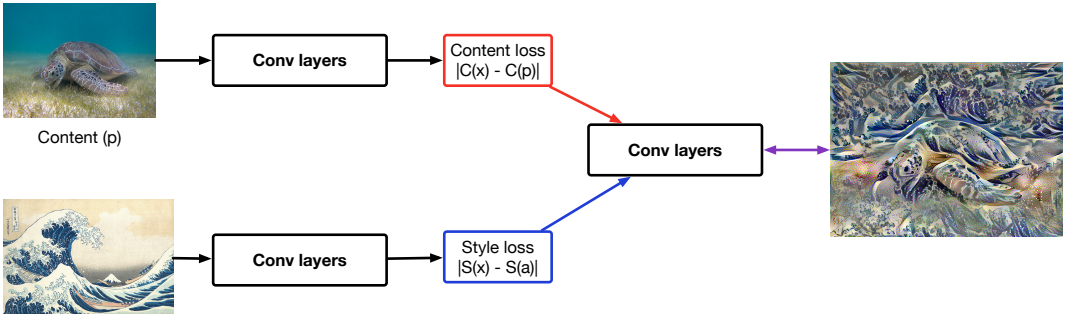
$$\Delta \mathbf{x} = -\eta \frac{\partial E(\mathbf{x}|\theta)}{\partial \mathbf{x}}$$

  - We will generate a new data from the initial value of **x** and the desired output **r**.
  - Network weights do not change
- Used in various circumstances
  - *Deep Dream* image generation from Google

Deep dream

Style transfer

- Idea: transfer the style of an image into a new image
  - Compare the content in the convolution layers (e.g. VGG19) and the style (Gram matrix)



Green Sea Turtle grazing seagrass, CC-BY-SA-3.0, https://commons.wikimedia.org/wiki/File:Green_Sea_Turtle_grazing_seagrass.jpg
The Great Wave off Kanagawa, public domain, https://commons.wikimedia.org/wiki/File:Tsunami_by_hokusai_19th_century.jpg

8.7 Adversarial approaches

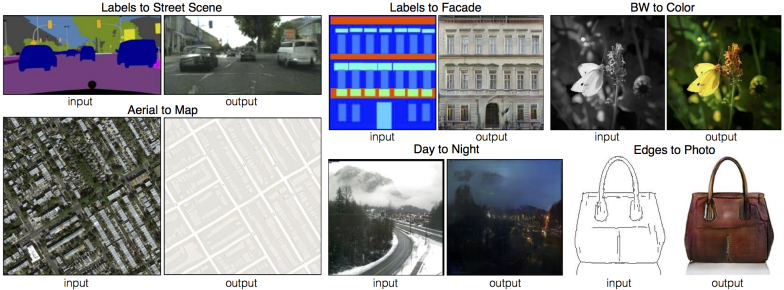
Adversarial data

- Use data generation to determine the smallest variation that would lead to a misclassification

- Caused by the use of distributed representation in a very high dimensionality space
- Illustrates a current difficulty with deep networks, robustness to adversary data needs to be improved

Generative Adversarial Networks (GAN)

- GAN model: putting in competition two neural networks
  - Discriminative network: distinguishing true data from the problem from generated data
  - Generative network: producing data that looks authentic
  - Allows various treatments based on unsupervised learning
- Example: image-to-image translation with conditional GANs

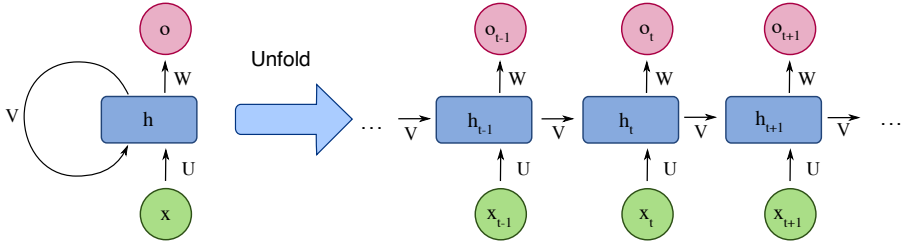


From *Isola, Zhu, Zhou and Efros, Image-to-Image Translation with Conditional Adversarial Networks, CVPR, 2017.* Accessed online on October 19, 2020 at `https://arxiv.org/pdf/1611.07004v3.pdf`.

8.8 Sequence processing

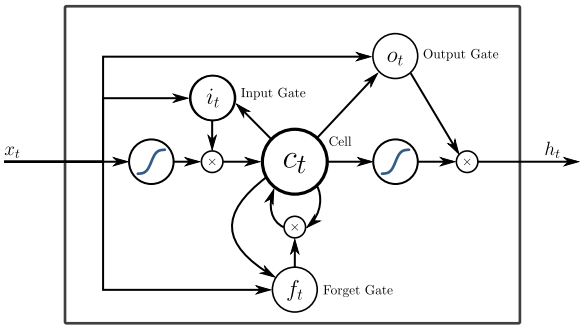
Recurrent network

- Usual networks (*feedforward*): data propagated in the network, independent of the following / previous data
  - Sequential data processing important in many contexts
- Recurrent networks: connections with previous values
  - Processing with usual algorithms by unrolling the network



By fdeloche, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg

Long Short-Term Memory (LSTM)



By Graves, Mohamed and Hinton, CC-SA 4.0, https://en.wikipedia.org/wiki/File:Peephole_Long_Short-Term_Memory.svg

- LSTM model: adding memory to the network
- Memory cell (state), with four neurons
  - Input
  - Input activation
  - Forgetfulness activation
  - Output activation

Deep reinforcement Learning

- Reinforcement learning: determining the right actions to take under current conditions
  - Guided by punctual rewards, without precise indications on decisive actions
  - More sophisticated (and more complex) form of intelligence than classification and regression tasks
- Deep networks are proving to be very promising for reinforcement learning
  - Massive simulations allow to learn how to perform certain actions
- Video games (Atari 2600): Deep Q-learning Network (Deepmind)
  - Input is a screenshot, reward is the score obtained
  - 49 different games, superhuman performances
- Go game: AlphaGo (Google Deepmind again)
  - Go: traditional Asian game, more complex than chess: `http://www.theverge.com/2016/3/9/11185030/google-deepmind-alphago-go-artificial-intelligence-impact`

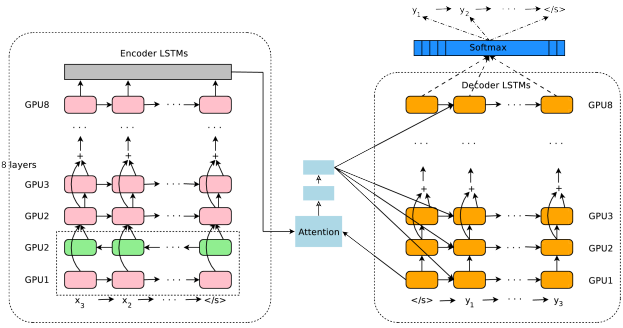
8.9 Deep learning applications

Automated Driving

- Autonomous vehicles: the next upheaval in the way of moving around
  - First tests with very expensive sensors (e.g. long-range LIDAR)
  - Development with more affordable technologies (e.g. video camera, RADAR, sonar)
- Great potential of deep learning for autonomous driving
  - Object and pedestrian detection
  - Reading of the signs (panels and lights)
  - Interpretation of the driving of other vehicles
  - Driving control
- Predictability of machine learning, especially deep learning, remains a problem
- Tesla's approach to autonomous driving
  - Equip all manufactured vehicles with sophisticated sensors and a network link
  - Collect information on driving for all these vehicles
  - Gradually automate driving by learning about these data as the quality of the learned models increases

Automated translation

- *Google's Neural Machine Translation System*: new version of Google Translate based on deep networks
  - 60 percent performance improvement over the previous version
  - Deployed on Google's systems simultaneously with its publication (fall 2016)



From *Wu et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, arXiv:1609.08144, 2016.* Accessed online on October 19, 2020 at `https://arxiv.org/pdf/1609.08144.pdf`.

Translation examples with GNMT

Source	When asked about this, an official of the American administration replied: "The United States is not conducting electronic surveillance aimed at offices of the World Bank and IMF in Washington."
PBMT (3.0)	Interrogé à ce sujet, un responsable de l'administration américaine a répondu: "Les États-Unis n'est pas effectuer une surveillance électronique destiné aux bureaux de la Banque mondiale et du FMI à Washington".
GNMT (6.0)	Interrogé à ce sujet, un fonctionnaire de l'administration américaine a répondu: "Les États-Unis n'effectuent pas de surveillance électronique à l'intention des bureaux de la Banque mondiale et du FMI à Washington".
Humain (6.0)	Interrogé sur le sujet, un responsable de l'administration américaine a répondu: "les États-Unis ne mènent pas de surveillance électronique visant les sièges de la Banque mondiale et du FMI à Washington".
Source	She was spotted three days later by a dog walker trapped in the quarry
PBMT (6.0)	Elle a été repéré trois jours plus tard par un promeneur de chien piégé dans la carrière.
GNMT (2.0)	Elle a été repérée trois jours plus tard par un traîneau à chiens piégé dans la carrière.
Humain (5.0)	Elle a été repérée trois jours plus tard par une personne qui promenait son chien coincée dans la carrière.

From *Wu et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, arXiv:1609.08144, 2016.* Accessed online on October 19, 2020 at `https://arxiv.org/pdf/1609.08144.pdf`.

8.10 Software implementation

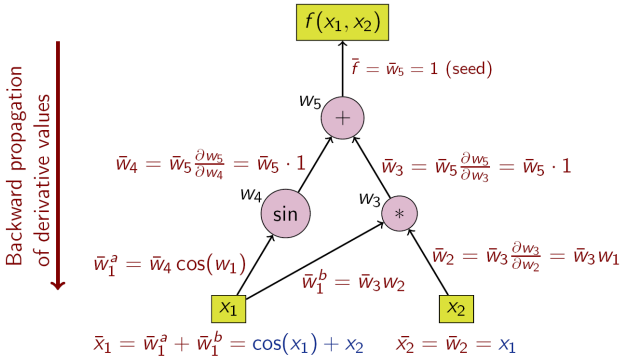
Automatic gradient

- Computational graph: representing the mathematical operations of a network in a graph
  - Captures the order and nature of operations
- Automatic gradient: calculate **analytical** gradients on the whole network automatically, via computational graphs
- Allows to define complex and heterogeneous network topologies without having to do the analytical derivatives manually!
- Also allows to optimize the processing on the targeted architecture (e.g. GPU)

Automatic gradient

$$\frac{\partial f(x_1,x_2)}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\sin(x_1) + x_1\, x_2\right) = \cos(x_1) + x_2$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} = \frac{\partial}{\partial x_2}\left(\sin(x_1) + x_1\, x_2\right) = x_1$$



By Berland, public domain, https://en.wikipedia.org/wiki/File:ReverseaccumulationAD.png

Tools for deep learning

- TensorFlow (Google): `https://www.tensorflow.org/`
  - Launched in November 2015, massive adoption by the community
  - Code in C++, with user interface in Python
  - Entirely organized around computational graphs
- PyTorch (open source): `https://pytorch.org/`
  - Based on Torch (Collobert and collaborators), a C++ library more than 15 years old: `http://torch.ch/`
  - Offers user-friendly Python programming interface, *programmatic* approach
  - Automatic differentiation made dynamically, more versatile than TensorFlow in some aspects

References

- Yann LeCun, Yoshua Bengio and Geoffrey Hinton. *Deep learning*. Nature, vol. 521, pages 436–444, 2015. https://doi.org/10.1038/nature14539
- Ian Goodfellow, Yoshua Bengio and Aaron Courville. "Deep Learning", MIT Press, 2016. http://www.deeplearningbook.org/
- Geoffrey Hinton, Yoshua Bengio and Yann LeCun, *Deep Learning NIPS'15 Tutorial*, 2015. https://nips.cc/Conferences/2015/Schedule?showEvent=4891