Deep Learning

Introduction to Machine Learning – GIF-7015

Professor: Christian Gagné

Week 8

UNIVERSITÉ LAVAL

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 non-linearly 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
- 2018: ACM Turing Award (“Nobel” prize of computer science) to Bengio, Hinton and LeCun for their work on deep learning
- 2020-2022: large generative models for text (ChatGPT) and images (DALL-E, Midjourney)

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

Representation learning

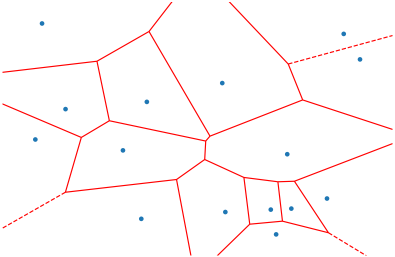
- Deep network motivation: learning a representation for weakly structured data
  - Weakly structured data: data whose information of interest is present in raw content, without being clearly identified (e.g., image, text, voice)
  - As opposed to tabular data, where each variable is clearly identified and has often been chosen according to the task of interest.
- Deep learning extracts a representation from the raw data that is adapted to the task in hand.
  - Avoids having to engineer a data representation by domain experts
  - However, requires a large amount of data to learn the representation from it

Model composition

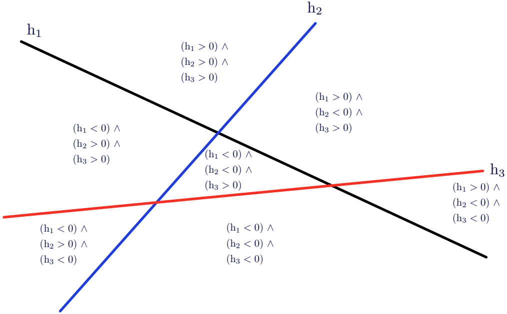
- Model compositionality is necessary in machine learning
  - Such as for language, we need to compose elements to define a language that gives meaning to complex notions
- Exploiting compositionality allows an exponential gain in representation power
  - Distributed representations, feature learning
  - 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

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

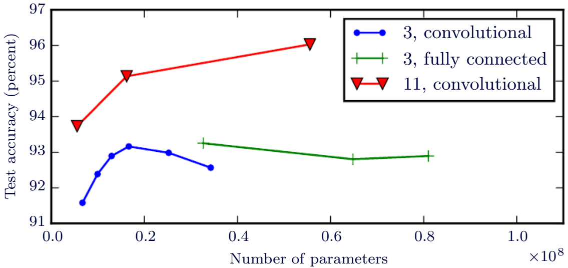


Figure 6.7 from *I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.* Accessed online on October 19, 2020 at `https://www.deeplearningbook.org/contents/mlp.html`.

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

8.2 Autoencoders

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)

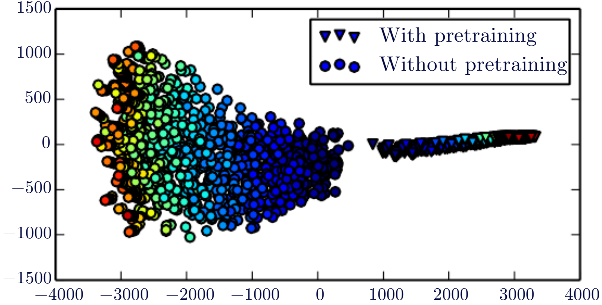
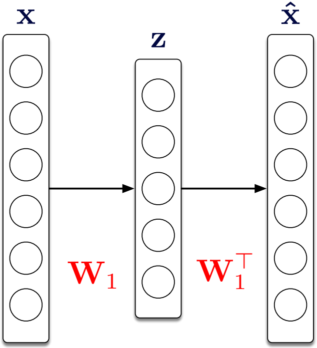


Figure 15.1 from *I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.* Accessed online on October 19, 2020 at `https://www.deeplearningbook.org/contents/representation.html`.

Autoencoders

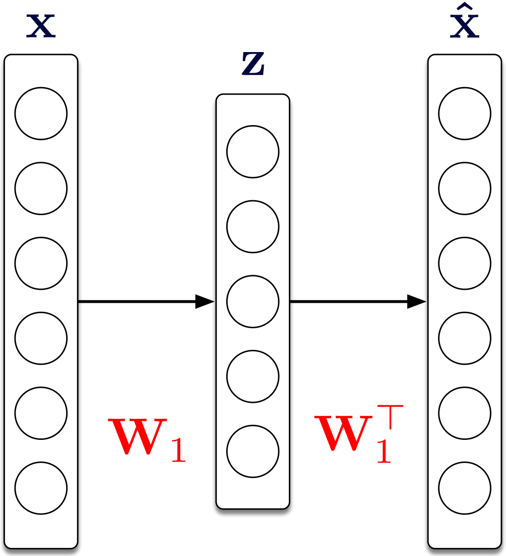
- Autoencoder: model allowing to compress the input (encoder) and decompress it (decoder).
  - Objective: compress while keeping the error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$ low
  - Decoder weights linked to encoder weights (usually transposed)



Autoencoders training

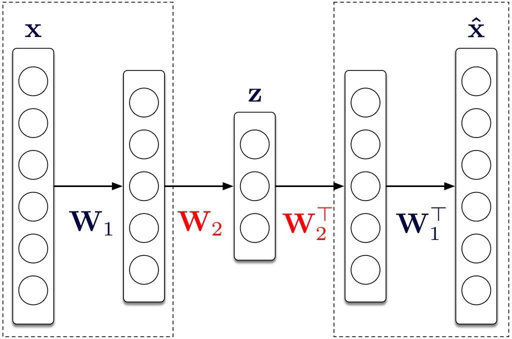
- Unsupervised training of the autoencoder, to learn representation
  - Encoder used to extract a compact representation
- Greedy training, one layer at a time
  - Training of the outermost layer
  - Addition of a new layer, which is driven individually, with the outer layer being fixed, and so on
- 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 the encoder, with supervised training
  - Complete backpropagation training of the output layer
  - Adjustment of encoder weights by backpropagation (*fine-tuning*)

Autoencoder training example



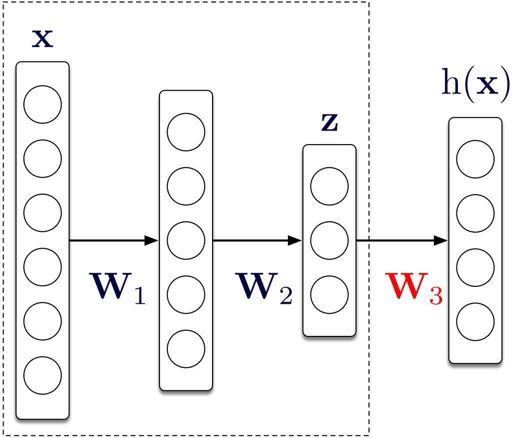
- Unsupervised weight training $\mathbf{W}_1$, weight $\mathbf{W}_1^\top$ linked
- Minimize error $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- Intermediate representation in central values (latent vector $\mathbf{z}$)

Autoencoder training example



- Addition of two new layers (one in the encoder and one in the 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

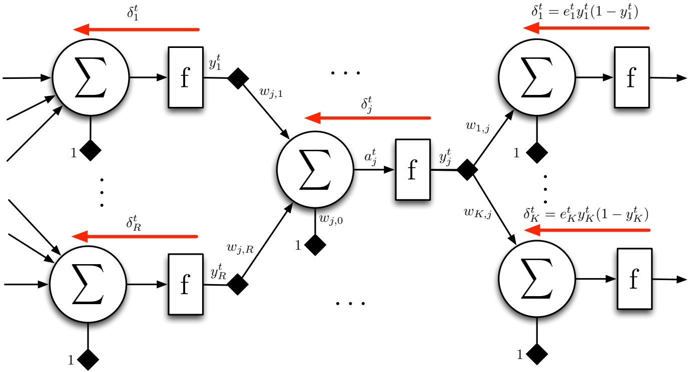
8.3 Elements of deep learning

Deep learning without unsupervised pre-training

- Unsupervised pre-training of deep networks is generally no longer required
- Various techniques enable direct training of deep networks
  - New transfer functions (e.g., ReLU) alleviate gradient dilution problem
  - Better initialization of network weights (Xavier and He techniques)
  - Random weight deactivation with dropout, enabling better distribution of processing across the network
  - Batch normalization, to renormalize values between layers, enabling some learning invariances
  - Residual links, to distribute input information more directly to deeper layers

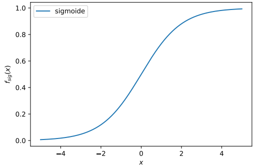
Vanishing gradient problem

- 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

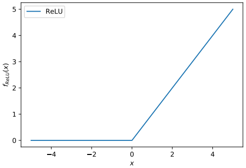


Transfer functions

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



- Transfer functions must include non-linearities
- ReLU function (*Rectified Linear Unit*), $f_{\text{ReLU}}(a) = \max(0, a)$
  - 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



Deep network initialization

- Initial weight values have a significant effect on gradient values used for learning
  - Initial weights too low $\Rightarrow$ gradient implosion, learning stagnation
  - Initial weights too high $\Rightarrow$ gradient explosion, learning instability
  - Make the right trade-off for initializing weights, taking into account the transfer functions used
- Mathematical justification: example of an L-layer deep network with linear transfer function

$$y = \mathbf{W}_L \, \mathbf{W}_{L-1} \, \ldots \, \mathbf{W}_2 \, \mathbf{W}_1 \, \mathbf{x}$$

  - Suppose $L-1$ first layers identical and equal to $\mathbf{W}$, $y = \mathbf{W}_L \, (\mathbf{W})^{L-1} \, \mathbf{x}$
  - With $\mathbf{W} = c \, \mathbf{I}$ and $c > 1$, explosion of output value, $\lim_{L \to \infty} y = \infty$
  - With $c < 1$, implosion of output value, $\lim_{L \to \infty} y = 0$

Initialization methods

- Xavier's method
  - Adapted with sigmoid transfer function and tanh.
  - Consists of uniform random values in $\left[-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}\right]$, where n_{in} is the number of inputs and n_{out} is the number of outputs of the neuron associated with the generated weight

$$w_{j,i} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right)$$

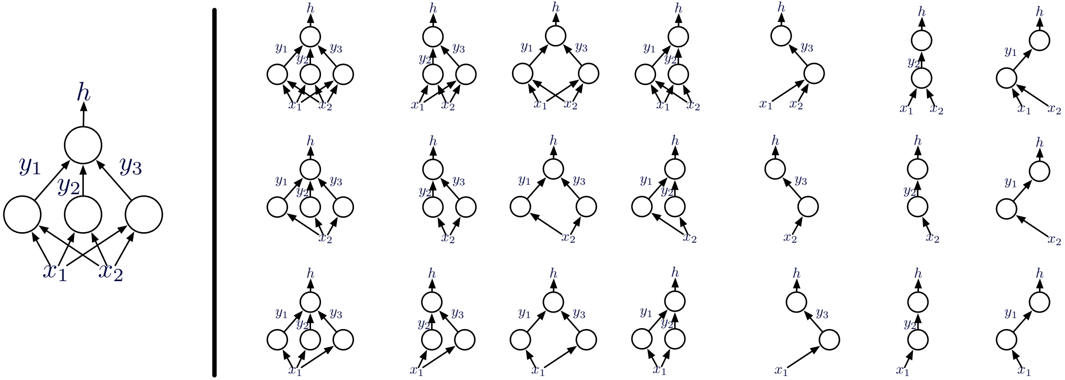
- He's method
  - For asymmetrical transfer functions such as ReLU, He's method is preferable
  - Initializes according to a Gaussian distribution that depends on the number of neuron inputs

$$w_{j,i} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$$

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 at test time by averaging over several selection masks
  - Analogy with ensemble learning (seen later in the semester), in particular to bagging

Dropout



Batch normalization

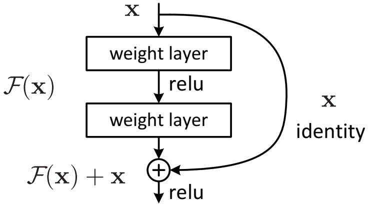
- Modification of a weight by backpropagation based on a local gradient
    - Weight of previous and following layers are also modified!
- *Batch normalization*: normalize activation of neurons between all the data of a mini-batch
    - Mini-batch: small subset of data instances from the training set (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

Residual links

- Residual links: allow direct connections between non-adjacent layers (*skip links*)



From *K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition. CVPR, 2016.* Accessed online November 6, 2020 at `https://arxiv.org/abs/1512.03385`.

- Enables much deeper and more powerful networks (ResNets)
  - ResNets: winners of ImageNet 2015 competition (3.57 % error top 5)
  - Facilitates signal optimization and propagation across the network
  - Residual block must do some processing to improve the output of the previous block, otherwise the current block may be ignored.

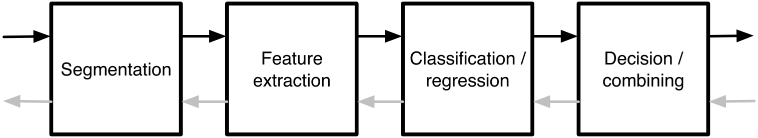
8.4 Learning from unstructured data

Learning from unstructured data

- Classical neural network models process fixed-size vectors
  - Assumes data are represented on a predefined number of variables
- Tabular data
  - Structured data, with a reasonable number of known variables
  - Directly usable by neural networks (e.g. MLP) and other models
- Images
  - Matrix of numbers, with each pixel represented by three real values (RGB)
  - Unstructured data, large number of variables (millions of pixels per image) and little significance of individual pixels
  - High locality of pixels in images, suitable operators (e.g. convolution) can take advantage of this
- Text
  - Collection of words forming sentences, variable-length sequences
  - Extensive vocabulary ($\sim 100\,000$ words in French), with synonyms, homonyms, related words

Representation learning

- Classic pattern recognition pipeline

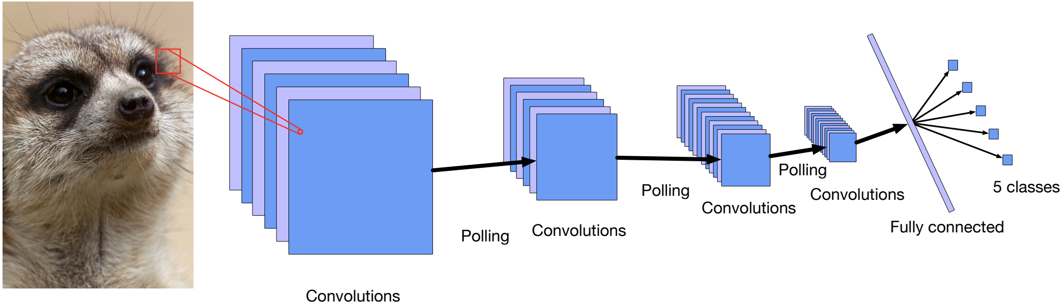


- In the past, each module was designed independently
- Deep learning allows learning of the representations
  - Learning of all modules simultaneously
  - Ability to retrieve representations (segmentation, feature extraction) and use them with other classification and decision-making modules

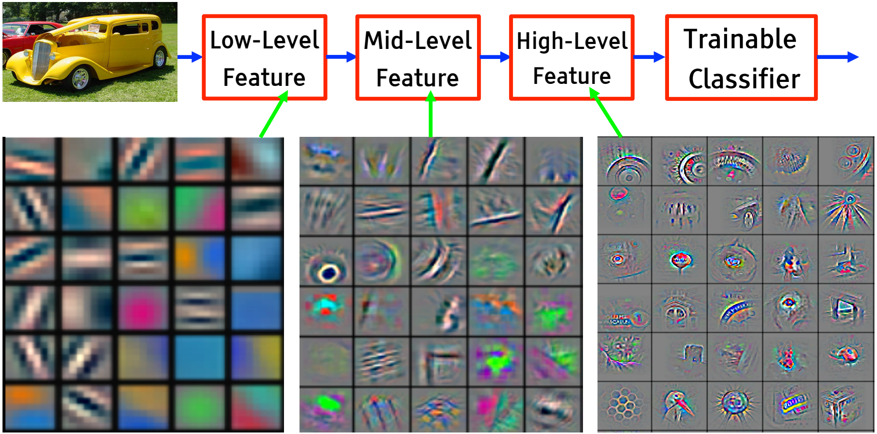
Convolution network

- 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: value 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
- Presented in detail in the next module

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

Text processing

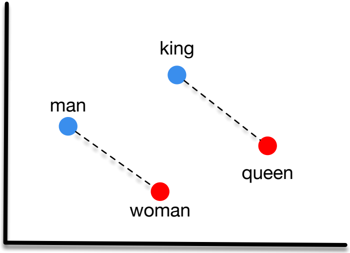
- How to give documents (sequence of strings) to a neural network (vector of fixed-size real values)?
- *Bag-of-Words* model (BoW)
  - Identify a dictionary of the most frequent / interesting words
  - Calculate the frequency of each word for each processed document (vector of integers of fixed size)

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

    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 order of the words
  - Models with N-gram: measures the frequency of adjacent word groups
  - Skip-gram: related words may not be adjacent

Word embedding

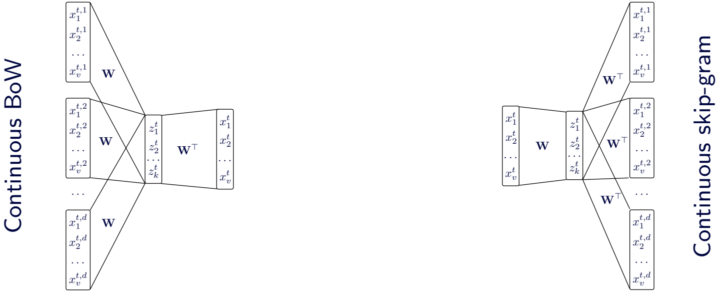
- Word embedding: projection of words into a vector space capturing semantic relations.
  - Words close together in the vector space have a similar or related meaning
  - Postulate that algebraic operations in this space respect a semantic logic.



- Construction of word embeddings generally done by unsupervised or self-supervised learning approaches
- Induced space is interesting for performing processing
  - For example, input of a neural network for document classification

Constructing word embeddings

- Idea: predict words in a sequence to encode text
    - *Continuous BoW*: predict the word according to those preceding and following (faster)
    - *Continuous skip-gram*: predict preceding and following words according to the word of interest (more accurate)

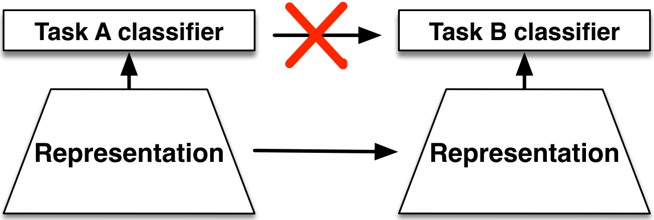


- word2vec: use *Continuous BoW* or *Continuous skip-gram* to build embeddings
    - MLP network, two hidden layers, embedding of a few hundred dimensions

8.5 Representation transfer and adaptation

Transfer of representations

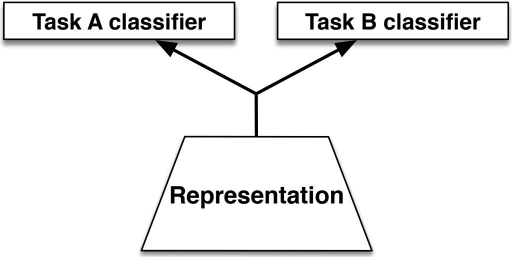
- Learning a deep network on task A
- New task B, based on data similar to task A
  - Retrieve task A representation
  - Train new classifier for task B



  - Allows a transfer of representation (*transfer learning*)
  - Fine-tuning of the representation on the new task possible
- Standard approach to learning object recognition model, using representation created with ImageNet

Multitask learning

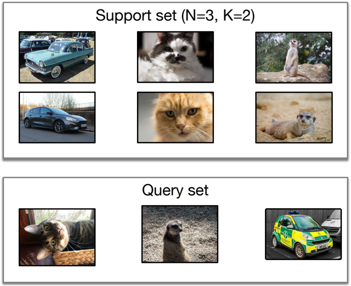
- Multitask learning: *simultaneously* learning a representation for separate operations
  - Two-headed network, one for each task



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

Few-shot learning

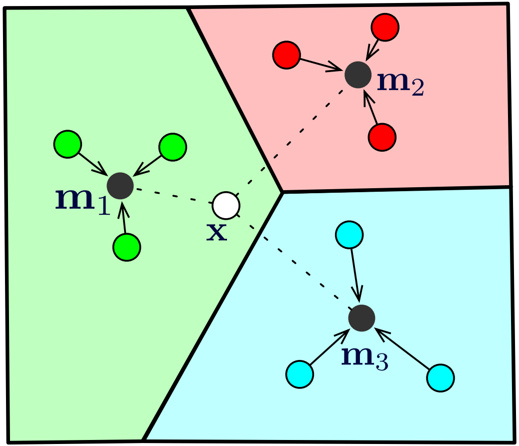
- How to learn with few data of each class?
  - General problem: learning to learn (meta-learning)
- Model with N classes of K instances each (N-*way*-K-*shot*)



  - Support set: K instances for each of the N classes to be processed
  - Query set: new instances to process
- Classes of support and query sets vary at each attempt
  - Learning models designed to work with classes unknown beforehand

Few-shot learning: prototypical network

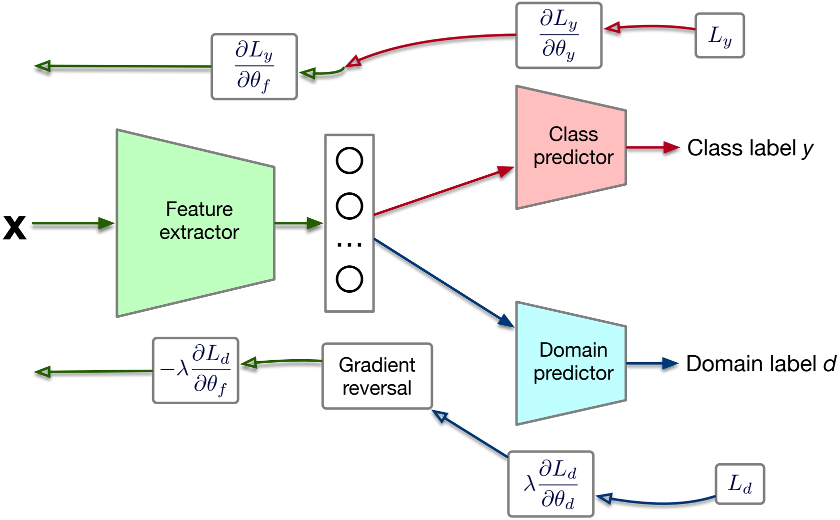
- Prototypical network
  - Summarize the support of a class by an average value (prototype)
  - Classify queries according to the nearest prototype



Domain adaptation

- Domain adaptation: use one or more source domains to better process the target domain
  - Model inputs and outputs are the same for all domains
  - Special case of transfer learning, where inputs/outputs may change in the general case.
- Domain Adversarial Neural Network (DANN)
  - Learning for multiple source domains, based on a multitasking learning model
  - One head associated to the classification task
  - A second head aims to discriminate between source domains in the shared part output
  - Learning the additional head involves a gradient reversal to force a common, general intermediate representation.

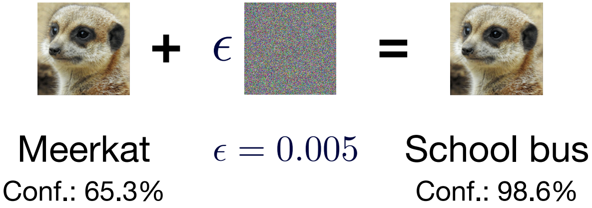
Domain adversarial neural network (DANN)



8.6 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

Attacks and defences

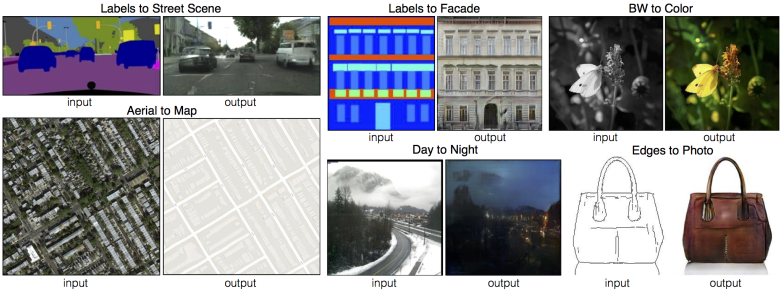
- Typical attacks: gradient descent on the data to deceive the network
  - Fast gradient sign method (FGSM):

$$\mathbf{x} = \mathbf{x} + \epsilon \operatorname{sign} \frac{\partial L(\mathbf{x}, y|\theta)}{\partial \mathbf{x}}$$

  - Several other variants proposed to produce adversarial data
- Defence mechanism: adversarial training
  - Augment the training set with adversarial data to make the network robust

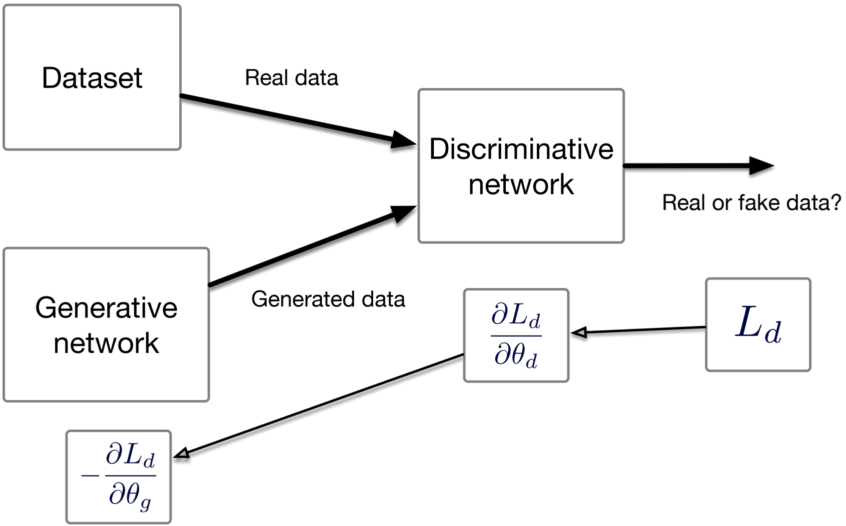
Generative Adversarial Networks (GAN)

- 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`.

Generative Adversarial Networks (GAN)



8.7 Software implementation

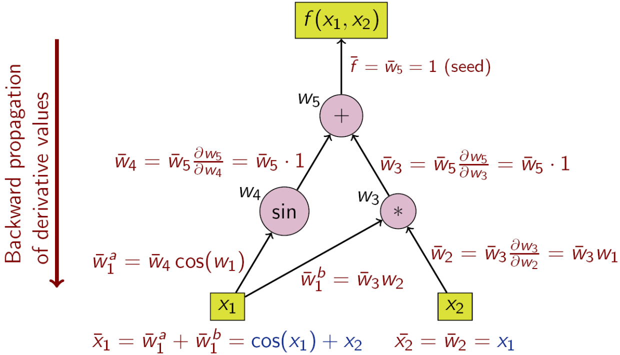
Automatic gradient

- 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

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

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



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

Tools for deep learning (open source)

- TensorFlow: `https://www.tensorflow.org/`
  - Code in C++, with a user interface in Python
  - Entirely organized around computational graphs
- PyTorch: `https://pytorch.org/`
  - Offers user-friendly programming interface in Python, *programmatic* approach.
  - Automatic differentiation done dynamically, more versatile than TensorFlow in some respects.
- Google JAX: `https://jax.readthedocs.io/`
  - Combines automatic gradient and high-performance numerical calculation in a standard interface (à la NumPy)
  - Not specifically designed for deep learning, but offers great flexibility
- Keras: `https://keras.io/`
  - User-friendly interface for deep learning, at the cost of reduced flexibility
  - On top of TensorFlow, PyTorch or JAX as underlying deep learning environment

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`