COMP30027 Machine Learning
"Deep" Learning: Part II

Semester 1, 2018

Tim Baldwin & Jeremy Nicholson & Karin Verspoor

What is Deep Learning?

- **Deep learning** is the combination of "deep" models (usu. meaning multiple hidden layers) with sufficient (= lots and lots of) data to train the models
- One key facet of deep learning is **representation learning**, i.e. the transformation of raw inputs into a (latent) representation that is more amenable to machine learning
- "(deep learning) usually work(s) well with large-scale datasets, especially when the model takes in low-level raw features ... (with) large-scale datasets ... ranging from hundreds of thousands to several millions of samples" [Zhang et al., 2015]

Document Representation

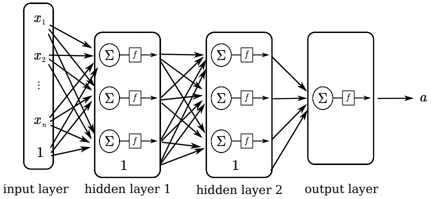
- As we saw in Project 2, a typical document representation is in terms of the words (“tokens”) within a document (discarding “context”)
- A “bag of words” representation of a document is a sparse vector of “term frequencies” (only a small proportion of the vocabulary is present)
- Other representations like “token *n*-grams” (pairs, triples, etc. of words) maintain more information from the document, but are more sparse
- A token/*n*-gram (frequency) can be suggestive of a class, but no more than suggestive

Image Representation

- Similar issues — an instance (image) is fundamentally comprised of pixel values (R/G/B, brightness, etc.)
- An individual pixel can be suggestive (?) of some image property
- Only really sensible in combination:
  - Some sequence of pixel values defines a line fragment
  - Some sequence of line fragments (roughly) defines a shape
  - Some sequence of shapes defines a ... dog
  - Requires lots of effort to try to identify meaningful features
  - **feature engineering** is very difficult

Representation Learning I

- In deep learning, at each layer of the model, it is possible to extract out a dense real-valued vector representation of a test instance (= **embedding**), e.g. from either hidden layer of the following MLP:



- This is generally the approach to deep representation learning, with the bigger question being how to train the model

Representation Learning II

- Each dimension of the embedding corresponds to ... something
    - The model has learned a generalise-able property
    - Sometimes the property has an interpretation ... but usually not
- The advantage of the embedding is three-fold:
    - We have fewer features = faster training/prediction
    - Feature engineering is "free"
    - With a careful choice of representation, we can represent an instance by a (combination of) embedding(s), allowing us to use the trained network for problems it wasn't designed for (!)

Representation Learning: word2vec I

- One particularly well-known example of representation learning in a language context is **word2vec** [Mikolov et al., 2013], e.g. the **CBOW** ("continuous bag of words") model:

$$J = \frac{1}{T}\sum_{i=1}^{T} \log \frac{\exp\left(\mathbf{w}_i^\top \sum_{j\in[-c,+c],j\neq 0} \tilde{\mathbf{w}}_{i+j}\right)}{\sum_{k=1}^{V} \exp\left(\mathbf{w}_k^\top \sum_{j\in[-c,+c],j\neq 0} \tilde{\mathbf{w}}_{i+j}\right)}$$

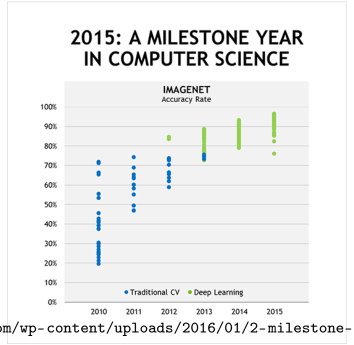
where $\mathbf{w}_i$ and $\tilde{\mathbf{w}}_i$ are vector representations of words (focus and context words, resp.), V is the vocabulary size, T is the number of tokens in the corpus, and c is the context window size

Representation Learning: word2vec II

- The basic idea is that for each word in a corpus, we attempt to predict it from its context words $w_{i-c}, w_{i-c+1}, \ldots w_{i+c}$ (ignoring sequence)
- This is framed as a classification task, with negative instances synthetically generated through **negative sampling**, by randomly selecting words from the vocabulary V (based on the assumption that they are not going to "fit" the original context of occurrence)
- The model used to learn the word embeddings is a simple feed-forward neural network (without a hidden layer), using logistic regression as the objective function

Deep Learning: Convolutional Neural Networks

- **Convolutional neural networks** ("convnets") are a staple of deep learning, especially for computer vision applications
- The power of convnets is perhaps best illustrated through a potted empirical history of ImageNet (the main annual "bakeoff" for computer vision):



Source(s): `https://blogs.nvidia.com/wp-content/uploads/2016/01/2-milestone-web1.gif`

Convnets: Overview

- Convnets are neural networks made up of the following layers:
  - convolutional layers
  - max pooling layers
  - fully-connected layers
- Part of the big breakthrough in 2012 with "AlexNet" [Krizhevsky et al., 2012] was stacking convolutional layers together
- With very deep convnets, ReLU is often used as the activation function, as it: (a) is fast to differentiate; and (b) tends to avoid the "vanishing gradient" problem

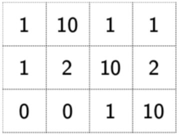
Source(s): `https://cs.nju.edu.cn/wujx/paper/CNN.pdf`

Convnets: Convolution I

- Convolutions are made up of two components:
  1. a **kernel**, in the form of a matrix which is overlaid on different sub-regions of the image, and combined through an **element-wise product**
  2. a **stride** $s \in \mathbb{Z}^+$ which defines how many positions in the image to advance the kernel on each iteration

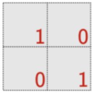
Convnets: Convolution II

- For example, assuming the following 3×4 image and $s = 1$:



1	10	1	1
1	2	10	2
0	0	1	10

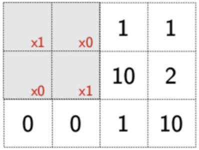
image



1	0
0	1

2x2 kernel

the first convolution would be:



x1	x0	1	1
x0	x1	10	2
0	0	1	10

convolution

$1 \times 10 + 10 \times 0 + 1 \times 0 + 2 \times 1 = 3$

Convnets: Convolution III

- The full convolutional output would be:

$$\begin{bmatrix} 3 & 20 & 3 \\ 1 & 3 & 20 \end{bmatrix}$$

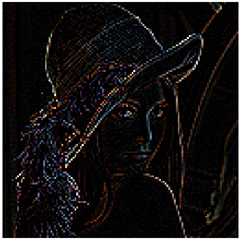
Convnets: Convolution IV

- In practice, multiple kernels are generally applied in a single convolutional layer, e.g.:

Original:

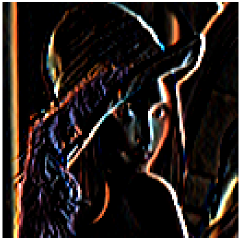
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

3×3 Laplacian:



$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3×3 Sobel:



$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Convnets: Pooling I

- Pooling is comprised of three components:
  (1) a **kernel**, in the form of a matrix which is overlaid on different sub-regions of the image to determine the extent of the "pool"
  (2) a **stride** $s \in \mathbb{Z}^+$ which defines how many positions in the image to advance the kernel on each iteration
  (3) the pooling basis: either **max** (usually 1-max, but sometimes k-max) or **average**

Convnets: Pooling II

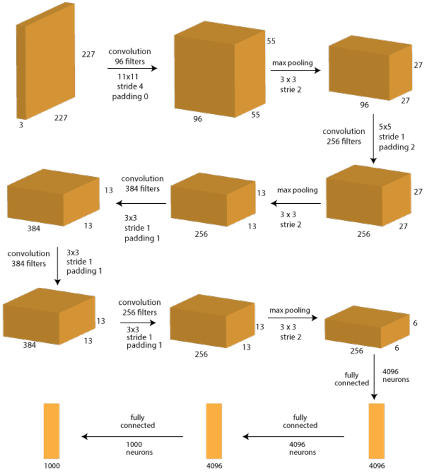
- For example, assuming the convolution output from earlier:

$$\begin{bmatrix} 3 & 20 & 3 \\ 1 & 3 & 20 \end{bmatrix}$$

and a 2×2 pooling layer with $s = 1$, the output in the case of 1-max and average pooling would be, resp.:

$$\begin{bmatrix} 20 & 20 \end{bmatrix} \qquad \begin{bmatrix} \frac{27}{4} & \frac{46}{4} \end{bmatrix}$$

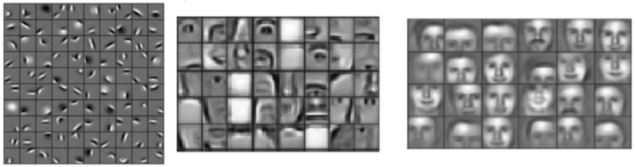
Real-world Convnet Example: AlexNet



Source(s): `https://www.analyticsvidhya.com/wp-content/uploads/2016/03/fig-8.png`

And What is Learned by a Convnet?

- Visualisation of representations learned by a convolutional neural network [Lee et al., 2011], showing how the model is able to hierarchically learn structure of images:



Convnets over Textual Data

- While text data is 1- rather than 2/3-dimensional, convnets have been successfully applied in document categorisation tasks (e.g. see Zhang and Wallace [2015]):
    - 1-dimensional convolutions of varying length over sliding windows over the text
    - (max-)pooling over different width windows
    - represent input as either character sequence or word2vec word-level embeddings (with updates)

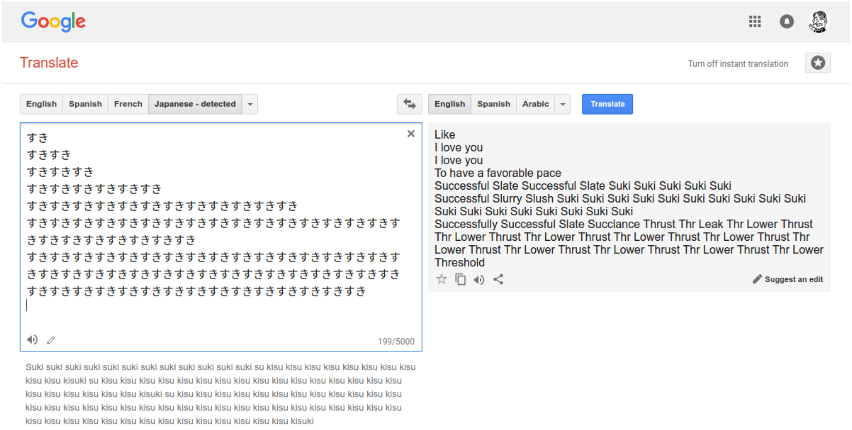
Deep Learning: The Good

- Huge impact on vision and speech recognition tasks in particular, with massive improvements in empirical accuracy over standard datasets
- Possible to model much larger contexts/neighbourhoods than conventional models, due to representation learning/generalisation
- Easy to combine different input modalities
- Easy to play around with using high-level libraries such as TensorFlow, PyTorch and keras
- Lots of vibe/demand for machine learning skills!

Deep Learning: The Bad

- Any savings in terms of feature engineering are outweighed by costs in terms of *architecture* engineering
- Very expensive to train over large datasets (with implications for hyperparameter tuning, architecture engineering, ...)
- Without code, reproducibility of results can be very low, due to the impact of various engineering tricks that have a big impact on results
- Overblown claims about the capabilities of deep learning (AI Armageddon, anyone?)

Deep Learning: The Ugly


Google
Translate
Turn off instant translation
English Spanish French Japanese - detected
English Spanish Arabic
Translate

すき
すきすき
すきすきすき
すきすきすきすきすきすき
すきすきすきすきすきすきすきすきすきすきすきすき
すきすきすきすきすきすきすきすきすきすきすきすきすきすきすきすきす
きすきすきすきすきすきすきすき
すきすきすきすきすきすきすきすきすきすきすきすきすきすきすきすきす
きすきすきすきすきすきすきすきすきすきすきすきすきすきすきすきすき
すきすきすきすきすきすきすきすきすきすきすきすきすきすきすき

199/5000

Suki suki suki suki suki suki suki suki suki suki suki suki su kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisuki su kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisuki su kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisu kisuki

Like
I love you
I love you
To have a favorable pace
Successful Slate Successful Slate Suki Suki Suki Suki Suki
Successful Slurry Slush Suki Suki Suki Suki Suki Suki Suki Suki Suki Suki
Suki Suki Suki Suki Suki Suki Suki Suki
Successfully Successful Slate Succlance Thrust Thr Leak Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Thrust Thr Lower Threshold

Source(s): Inspired by related posts to Language Log

Deep Learning: Current Trends

- End-to-end deep learning
- Multimodal learning
- Deep reinforcement learning
- More specialised hardware to train deep learning models
- Lines of work in learning from the successes of deep learning, and showing that conventional (computationally cheaper!) methods can achieve similar results for some tasks

Summary

- What is deep learning?
- What is representation learning?
- What are the basic elements of a convolutional neural network?
- How do convolution and pooling layers work?
- What are the strengths and weaknesses of deep learning?

References I

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103, 2011.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations, 2013*, Scottsdale, USA, 2013.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.

Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015. URL `http://arxiv.org/abs/1510.03820`.