Neural Networks and Deep Learning

Applications

Zoran Kostić
Columbia University
Electrical Engineering Department &
Data Sciences Institute

COLUMBIA ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Application of Deep Learning

*Large Scale Deep Learning
Computer Vision
Speech Recognition
Natural Language Processing
Other Applications

Applications
Large Scale Deep Learning

Fast CPU Implementations

GPU Implementations

Distributed Implementations

Model Compression

Dynamic Structures

Specialized Hardware Implementations

Applications
Large Scale Deep Learning

Philosophy of Connectionism

- **Intelligent behavior can be exhibited by a VERY large number of connected neurons**

ANN size increases have been exponential, but still mimic only the insect’s nervous system.

Applications
Large Scale DL: Fast CPU Implementations

As of 2019, single CPUs are not the best choice

Efficient CPU code:

- **Fixed point, as opposed to floating point arithmetic (up to order of magnitude improvement)**
- **Optimized algorithms, code and libraries are required**
- **Not a forte of a typical SW engineer**
- **But practiced by digital signal processing and real time embedded coding communities**

Applications
Large Scale DL: Fast CPU Implementations

A word on fixed point computations:

- **16-bit fixed point DSPs**
- **8-bit fixed point DSPs?**
- **<8-bit custom logic design**

Applications

Large Scale DL: Fast CPU Implementations

Note on "moveable fixed-point" computations

Content of a register: 101001

- **What number does it represent?**

show examples

Applications

Large Scale DL: GPU Implementations

GPU - Graphic Processing Unit

- **GPUs for 3D computer graphics –> mesh generation OpenGL**
- **Video game rendering: triangle vertices, lines and textures; 3D coordinate computation and transformation**
- **Matrix computations**

Applications
Large Scale DL: GPU Implementations

GPU - Graphic Processing Unit

- **Massively parallel computations**
- **Small number of branching (if then) instructions**
- **OpenGL standard (managed by Khronos)**
  - **https://www.khronos.org/**
- **Huge memory bandwidth**
- **Lower clock speeds**

Applications

Large Scale DL: GPU Implementations

Evolution of a GPU: from Video gaming computations to ANN computations

- **Each neuron has a similar computational need, and all can be evaluated in parallel**
- **Large number of localized parameters, activation and gradient values**
- **Update required every step of the training**

Applications
Large Scale DL: GPU Implementations

GPU -> GPGPU: General Purpose Graphic Processing Unit

- **IEEE floating point standards for ALU operations**
- **Arbitrary code (not only 3-D graphics)**
- **Data/instruction cache is not the key to performance**
- **Data transfer between memory and processing elements is critical**
- **Languages: NVIDIA CUDA, Khronos OpenCL (heterogeneous computing)**

Applications
Large Scale DL: GPU Implementations

Languages: NVIDIA CUDA, Khronos OpenCL (heterogeneous computing), Apple Metal

- **SIMD: single instruction multiple data**
- **Multi-threaded code (thread or work item)**
- **Notion of a thread-block (work group) is about data/memory management**
- **Data Alignment / Coallescing**
- **Efficient modular libraries are key**
  - **Are eigen or BLAS libraries in Tensorflow (theano) optimized for GPUs?**

Applications
Large Scale DL: GPU Implementations

SISD

SIMD

show examples

Applications
Large Scale DL: GPU Implementations

Data Coalescing

Data Alignment

show examples

Applications
Large Scale DL: GPU Implementations

Flash Memory -> DRAM -> On-chip Registers

show examples

Applications
Large Scale DL: GPU Implementations

- **Hardware: IBM/NVIDIA servers**
- **Chip: NVIDIA Xavier- 20 Watts**
  - https://en.wikichip.org/wiki/nvidia/tegra/xavier
  - https://www.nvidia.com/en-us/autonomous-machines/jetson-agx-xavier/
  - https://developer.nvidia.com/embedded-computing
- **Hardware: IBM/NVIDIA servers**
- **GPU + FPGA Assisted Server: Microsoft**
- **AMD Firepro**

Applications
Large Scale DL: Distributed Implementations

Workload: training and inference.

Inference can be run on separate machines

- **For each input example: Data parallelism**
- **Several machines working on single data point, each machine working on a different part of the model: Model Parallelism**

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Applications
Large Scale DL: Distributed Implementations

Workload: training and inference

Training:

- **Larger mini-batch for a single SGD step**
- **Standard SGD is a sequential algorithm (t depends on t-1)**
- **Use asynchronous stochastic gradient descent**
  - **Memory share between processor cores, read and write WITHOUT lock**
  - **Cores overwrite results of each other**
  - **Average amount of improvement is reduced for each SGD step**

Applications
Large Scale DL

Workload: training and inference

Training can be done a-priori, offline:
- **(for impersonalized applications)**

More important to minimize the cost of inference.

Example: speech recognition training on a computer cluster, inference on a phone.

Applications
Large Scale DL: Model Compression

Compression:

- **Replace the original model with a smaller memory/execution footprint**
- **Applicable for (large) original models where the size is driven by a need to prevent overfitting**

Applications
Large Scale DL: Model Compression

Original (large) model:

- **Often an ensemble of several independently trained models**
- **Learns function *f(x)*, but probably with many more parameters than necessary for the task (due to lack of training examples)**
  - **Fit the function first, then use the model on a number of random points to generate MANY new examples.**
  - **Then use a smaller model to train to all of those many new examples/points…**

Applications
Large Scale DL: Dynamic Structure

Dynamic ANN structure allows for changing the graph, as a function of input (“conditional computation”).

One has to determine which subset of a group of ANNs should be used for some input!

Applications
Large Scale DL: Dynamic Structure

Cascade Strategy:

- **Simple classifier to identify that a rare object is present (low capacity, high recall)**
- **Final classifier with high precision**
- **Stop work as soon as some classifier fails**
- **Full inference runs only when needed**
- **Two approaches: later stages have high capacity; many later stages have low capacity but system has high capacity**

Applications
Large Scale DL: Dynamic Structure

Gater ANN (mixture of experts):

- **Set of probabilities for each expert, final output uses weighted expert outputs**
- **Has to be managed to save computing**

Switch component:

- **Hidden units get input conditionally**

Problem with dynamic systems:

- **Use conditional flow, not good for parallelization**
- **Play with pipelining**

Applications
Large Scale DL: Specialized Architectures

Power Consumption

- FPGA-Assisted Server: Microsoft
- Chip: Resistive RAM
- Chip: Graphcore
- Chip: IBM TrueNorth/Synapse 70mW; deep dive
- Chip Research: Spike Sorting at Columbia EE Prof. Seok

How many bits is enough?

- 8 to 16 or ?
- Dynamic fixed point

Applications

Large Scale Deep Learning
*Computer Vision
Speech Recognition
Natural Language Processing
Other Applications

Applications
Computer Vision

Recognition, understanding

Preprocessing

- **Contrast Normalization**
- **Dataset Augmentation**

Applications
Computer Vision: Recognition, Understanding

Recognition

- **Object recognition**
- **Face recognition**
- **Optical character recognition**
- **Image annotation**
- **Detect sound from video**
- **Image synthesis (useful for image repair)**

Applications
Computer Vision: Preprocessing

- Pixel range normalization (0-255, 0-1)
- Scale adjustment (is often required
  - Research opportunities (Mellin transform, Fourier/spectral domain)
  - Some convolutional models can keep output constant
- Dataset augmentation
  - For reduction of the generalization error
  - Very useful for images (affine invariance)
  - “Fancy” augmentation: flip, color perturbation, nonlinear geometric distortions
  - Corresponding test strategy (ensemble test)
- AlexNet: subtract mean is the only preprocessing step

Applications
Computer Vision: Preprocessing - Contrast

Contrast Normalization

- Problematic variation in images
- Standard deviation of pixel values in a region of an image (shade,…)

Global contrast normalization (GCN)

- Is simple: subtract mean, rescale for constant standard deviation s
- But does not work if
  - For zero-contrast images, division by zero, messes up edges/corners
  - Needs regularization
- Local GCN: windowed approach, may be done per R,G,B

Applications

Large Scale Deep Learning
Computer Vision
*Speech Recognition
Natural Language Processing
Other Applications

Applications
Speech Recognition

The Task:

- **Map an acoustic signal containing a spoken utterance into the sequence of words**

Acoustic input (usually 10 or 20ms frame)

$$\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T)})$$

Use hand-designed features, or raw input

Output sequence of chars/words

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$$

What is the most probable linguistic sequence

$$f^*_{ASR}(\boldsymbol{X}) = \arg\max P^*(\boldsymbol{y} \mid \mathrm{X} = \boldsymbol{X})$$

Where P^* is the true conditional distribution

Applications
Speech Recognition

1980-2012 - Field dominated by

- **Hidden Markov Models (HMMs): waveforms modeled as sequences of phonemes with states (beginning, middle,end)**
- **Gaussian Mixture Models (GMMs): acoustic features <-> phonemes, transform discrete symbol into a brief audio waveform**
- **Attempts to use ANNs worked (TIMIT with 26% error, and better than GMM-HMM), but did not get embraced**

Applications
Speech Recognition

2009 -

- **Deep ANNs improved recognition accuracy**
- **Start: Based on training restricted Bolzmann machines (RBMs) to model the data**
- **RBMs are undirected probabilistic models**
- **Two steps: pre-training and training**
- **Take input acoustic signal in a frame, predict conditional probabilities of HMM states for the frame (error~20%)**

Applications
Speech Recognition

2009 -

- **Expand from phoneme recognition to large-vocabulary speech recognition (word sequences)**
- **Replaced Bolzmann with RELU + dropout**
- **Use of CNNs to replicate weights across time AND frequency (2D spectrogram)**
- **Removal of HMM by using LSTM RNN (error ~17.7%)**

Applications

Large Scale Deep Learning
Computer Vision
Speech Recognition
*Natural Language Processing
Other Applications

Language Modelling

- A B C A B C A B _
  What symbol comes next?
  What is its probability?

- Yesterday it was Sunday, so today it must be _
  How to predict the next word?
  What is this good for?

Applications: Natural Language Processing

Natural Language Processing

- **N-grams**
- **Neural Language Models**
- **High-Dimensional Outputs (Short Lists, Hierarchical Softmax, Importance Sampling, Noise-Contrastive Estimation and Ranking Loss)**
- **Combining Neural Language Models with n-grams**
- **Neural Machine Translation (Attention Mechanism and Aligning Pieces of Data)**
- **Historical Perspective**

Applications: Natural Language Processing

NLP Basics:

- **Natural languages can be ambiguous and can defy formal representation**
- **NLP = Use of human languages by computer: read and emit specialized languages, to allow parsing by simple programs**
- **Translation from one language to other**
- **Language models: probability distributions over sequences of words, characters or bytes**

Applications: Natural Language Processing

NLP and ANNs:

- **Generic ANN techniques can be applied to NLP**
- **Key: sequential data processing**
- **Need to include domain-specific knowledge (regularization, …)**
- **Choice: work with sequences of words (rather than characters) -> extremely large word-based, multi-dimensional and sparse spaces**

Applications: Natural Language Processing
n-grams

Language model: probability distribution over sequences of discrete tokens (word, character).

N-gram = sequence of n tokens:

- **Conditional probability of n-th token based on previous tokens (chain rule)**

$$P(x_1, \ldots, x_\tau) = P(x_1, \ldots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t \mid x_{t-n+1}, \ldots, x_{t-1}).$$

Training n-gram model is simple:

- **count how many times does n-gram occur in the training set.**

Applications: Natural Language Processing
n-grams

Simultaneous training of of Pn and Pn-1

$$P(x_t \mid x_{t-n+1}, \ldots, x_t) = \frac{P_n(x_{t-n+1}, \ldots, x_t)}{P_{n-1}(x_{t-n+1}, \ldots, x_{t-1})}$$

Example: trigram model

$$P(\text{THE DOG RAN AWAY}) = P_3(\text{THE DOG RAN})P_3(\text{DOG RAN AWAY})/P_2(\text{DOG RAN}).$$

Applications: Natural Language Processing
n-grams

Limitations:

P_{n-1} can often be zero

- undefined ratio (apply smoothing = choose unobserved "close" tuples)

Curse of dimensionality

- Many n-grams will not occur in the training set
- N-grams looks like a k-nearest neighbor lookup -> the model must be able to share knowledge between a word and (many of) its neighbors

Applications: Natural Language Processing n-grams - > Neural Models

Main deficiency of N-grams is the exponential growth of number of parameters with the length of the context

Neural networks address this problem by performing dimensionality reduction and parameter sharing.

Neural network language models are today state of the art, often applied to systems participating in competitions (ASR, MT)

There are two main types of neural network architectures for language modeling: feedforward and recurrent.

Applications: Natural Language Processing
Neural Language Models

NLMs to overcome the curse of dimensionality:

- **Able to recognize that 2 words have "similar features" although they are encoded differently**
- **Example: words dog and cat may map into some representation where they share many attributes -> then sentences that contain the word dog can inform the predictions for sentences with the word cat**
- **Generalizations can happen in many ways -> curse of dimensionality -> exponential relationship to sentence length**

Applications: Natural Language Processing
Neural Language Models

Word Embedding Space

- **Raw symbols are points in a space of dimensions equal to the vocabulary size, distance of *sqrt(2)***
- **Representations embed those points in a feature space of lower dimensions**
- **Word embeddings space: words that frequently appear in similar context are close to each other**
- **Useful to NLP: use hidden layers of ANNs to map words to some real-valued vector space, capture various embeddings**

Applications: Natural Language Processing
High Dimensional Outputs: Short Lists

Limit the vocabulary size (100K -> 10K)

- **Shortlist of L most frequent words (NN)**
- **Shortlist of tail T = V/L rare words (n-gram)**
- **Combine the two predictions**
  - **Have to predict probability that a word after context C belongs to the tail list (extra sigmoid unit at output)**

$$P(y=i \mid C) = 1_{i\in\mathbb{L}} P(y=i \mid C, i\in\mathbb{L})(1-P(i\in\mathbb{T}\mid C)) + 1_{i\in\mathbb{T}} P(y=i \mid C, i\in\mathbb{T}) P(i\in\mathbb{T}\mid C)$$

  - **Limited benefit of NN**

Applications: Natural Language Processing High Dimensional Outputs of NLMs

Often: words as a fundamental output

- **Large vocabulary (100K) -> expensive to represent distributions over word choice**
- **Naïve: affine from hidden to softmax out**
  - **Affine matrix large, full matrix multiplication needed both at training (all likelihoods and gradients) and test times (probabilities of all words) - complexity O (|V|n_h)**

$$a_i = b_i + \sum_j W_{ij} h_j \quad \forall i \in \{1, \ldots, |\mathbb{V}|\},$$

$$\hat{y}_i = \frac{e^{a_i}}{\sum_{i'=1}^{|\mathbb{V}|} e^{a_{i'}}}.$$

Applications: Natural Language Processing
High Dimensional Outputs: Hierarchical Softmax

Hierarchical decomposition of probabilities

- **Computational reduction to log |V|**
- **Nested: Categories (categories (categories))**
- **Tree with words as leaves**
- **With balanced trees, depth = *O* (log|V|)**
- **Multiply probabilities at every branch leading to a word, multiple branches possible -> addition**

Applications: Natural Language Processing
High Dimensional Outputs: Hierarchical Softmax

Hierarchical decomposition of probabilities

- **Use of logistic regression at each node – use supervised learning with cross-entropy loss (maximize log-likelihood of the correct decision sequence)**
- ***O*** **(log|V|) applies to gradient computations as well**
- **Information theory: choose optimal binary code for a word, number of bits approximately equal to the log of word frequency**

Applications: Natural Language Processing
High Dimensional Outputs: Hierarchical Softmax

$$P(y = w_4) = P(b_0 = 1, b_1 = 0, b_2 = 0)$$

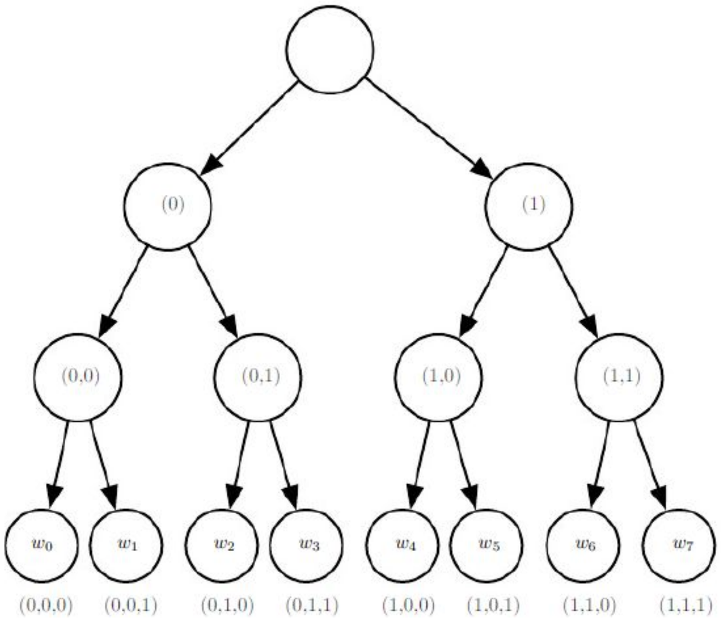
$$= P(b_0 = 1)P(b_1 = 0 \mid b_0 = 1)P(b_2 = 0 \mid b_0 = 1, b_1 = 0).$$

Hierarchy of 8 words

Words at bottom

Word frequency = weights

Perfectly balanced



Applications: Natural Language Processing
High Dimensional Outputs: Hierarchical Softmax

Simplification (since $\log_2(10^6)$~20):

- **Define a tree with depth 2 and branching factor of sqrt (|V|)**
- **Sets (classes) of mutually exclusive words!**

How to best define the word classes (word hierarchy)

- **Hard to learn, exact optimization using max likelihood is discrete and not amenable to gradient-based optimization**

Hierarchical approach brings computational benefits at both training and test time

- **Although even log complexity is expensive**
- **And performance can be better**

Applications: Natural Language Processing
High Dimensional Outputs: Importance Sampling

Speedup:

- **Avoid computing contribution of the gradient from all of the words that DO NOT appear in next position**
- **Use only a subset of those words, sampling from some other “proposal “ distribution *q*. Correct for the bias introduced by the imprecise distribution -> biased importance sampling**
- **Notion of positive and negative phase contribution to the gradient**
- **Negative proposal distribution is beneficial to reducing numerical complexity, by the fact that the number of possible samples from the proposal distribution is significantly smaller that the size of the output layer.**

Applications: Natural Language Processing
High Dimensional Outputs: Ranking Loss

View the output of NN for each word as a score, and try to make the score of the correct word be ranked high.

The ranking loss

$$L = \sum_i \max(0, 1 - a_y + a_i).$$

The gradient is zero for the i-th term if the score of the observed word ay is greater than the score of the negative word by margin of 1.

Applications: Natural Language Processing
Combine NLMs with n-grams

N-grams achieve high capacity with little computation to process an example (since frequencies of many tuples have been stored) – keep high model capacity.

For NN, doubling the number of parameters roughly doubles the computational time.

The add capacity: combine the two approaches in an ensemble -> will reduce the test error if making independent mistakes. Can have many ways of combining/weighting the two approaches

Applications: Natural Language Processing
Neural Machine Translation

Reading in one language, writing in another language, with the same meaning.

Issues: location of adjectives vs. nouns
Components:

- **Propose candidate translations**
- **Language model**
  - **Evaluates the proposed translations, and scores**

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Applications: Natural Language Processing Neural Machine Translation

Reading in one language, writing in another language, with the same meaning.

Original models used n-grams for the language model

- **Reporting the probability of a natural language sentence**

Extension to neural models, which can incorporate the conditional distribution over some variable given context C.

RNNs allow for having sentences of different sizes at input and output.

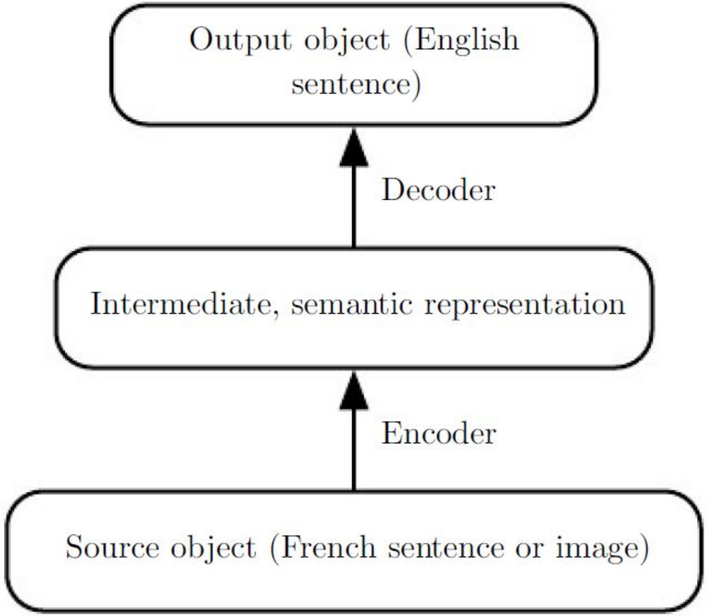
Applications: Natural Language Processing
Neural Machine Translation
Encoder-Decoder Architecture

Intermediate semantic representation captures the meaning.

Applied to translation, image captioning.

Capturing semantic meaning for a 60 word sentence is hard, needs a very large RNN.

More efficient to “read “ the sentence to get the gist and then produce the translated words one at a time.



Applications: Recommender Systems

Online advertising and recommendations

- **rely on predicting the association between a user and an item, to predict the probability of action or gain.**

Association:

- **supervised learning problem which can predict a proxy (clicks, …) which is a regression or a classification.**

Applications: Recommender Systems

Generalizations rely on similarity (user, item):

- **Predictions are obtained by the dot product between product embedding and user embedding**
- **Minimization of the squared error between predicted ratings and actual ratings**
- **Can be done using SVD**

Problem of collaborative filtering:

- **new item no history -> introduce extra info (user profile, item feature) -> CNNs**

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Backup Slides

Applications

Content

Large Scale Deep Learning

- **Fast CPU Implementations, GPU Implementations, Distributed Implementations, Model Compression, Dynamic Structure, Specialized Hardware Implementations**

Computer Vision

- **Preprocessing (Contrast Normalization, Dataset Augmentation)**

Speech Recognition

Natural Language Processing

- **N-grams, Neural Language Models, High-Dimensional Outputs (Short Lists, Hierarchical Softmax, Importance Sampling, Noise-Contrastive Estimation and Ranking Loss), Combining Neural Language Models with n-grams, Neural Machine Translation (Attention Mechanism and Aligning Pieces of Data), Historical Perspective**

Other Applications

- **Recommender Systems (Exploration vs. Exploitation), Knowledge Representation Reasoning and Question Answering (Knowledge, Relations and Question Answering)**