Neural Networks and Deep Learning

Introduction I

Nicholas Dronen

Department of Computer Science
dronen@colorado.edu

January 15, 2019

Course Overview

Examples of Deep Learning

- Natural Language Processing
- Computer Vision

Assignment 0

Course Goals

- Solid understanding of types of neural networks (forward and backward modes) and their strengths and weaknesses
  - Feed-forward
  - Recurrent
  - Convolutional
- Fundamental knowledge of the deep learning stack
  - Frameworks – especially automatic differentiation (AD), the powerful technique at the heart of most deep learning frameworks
  - Hardware – from data center to low-power, mobile environment
- Awareness of current research directions, limitations

Milestones

- Nine assignments
  1. Self assessment (not graded)
  2. Perceptrons
  3. Multi-layer Neural Networks
  4. Automatic Differentiation
  5. Optimization
  6. Regularization
  7. Recurrent Networks
  8. Convolutional Networks
  9. Compilers
- Mid-term exam
- Final exam

Grading

Category	Weight	Purpose
Participation	10%	Clarify or reinforce concepts, help peers
Assignments	25%	Reinforce concepts, help peers
Midterm exam	30%	Demonstrate mastery of concepts
Final exam	35%	Demonstrate mastery of concepts

Instructor Background

Natural Language Processing

PEARSON

- Automated scoring of writing for educational assessments
- Many tasks, many models, many datasets of varying sizes
- Feature engineering *and* feature learning, “wide and deep”
- Convolutional networks for error correction (CU thesis, 2016)

Computer Vision

here

- Object detection for automated construction of maps for automated driving
- Heavy use of convolutional neural networks
- Feature learning *only*
- Focus on trade-off between accuracy and speed for mobile deployment

Correcting Writing Errors

- In writing assessments, error correction performance requirements are high
- Interactive, formative setting
  - User submits example of writing
  - System response identifies errors, returns ranked candidate lists of corrections
  - $\implies$ ~95% rank-5 accuracy
- Automatic, summative setting
  - User submits example of writing
  - Rubric says to ignore errors in spelling, grammar
  - System automatically replaces each error with first element of ranked candidate list
  - System scores *corrected* writing, returns score
  - $\implies$ ~99% rank-1 accuracy

Proposed Approach

- Augment data by learning generative model of non-words
  - Curated error+correction corpora (smaller N, V)
  - Generated error+correction corpora (larger N, V)
  - Wikipedia
- Use convolutional neural networks (ConvNets)
  - Ensemble of n-gram feature detectors
  - Well-suited to correction of single-word errors
- Systematically evaluate ConvNets on canonical error detection/correction tasks.

Data Sets: Curated Non-word Corpora

Corpus	Errors	$\lvert V \rvert$
Aspell	531	450
Birbeck	36133	6136
Holbrook	1771	1199
Wikipedia	2455	1922

Table 1: Available from `http://www.dcs.bbk.ac.uk/~roger/`

Data Sets: Generated Non-word Corpora

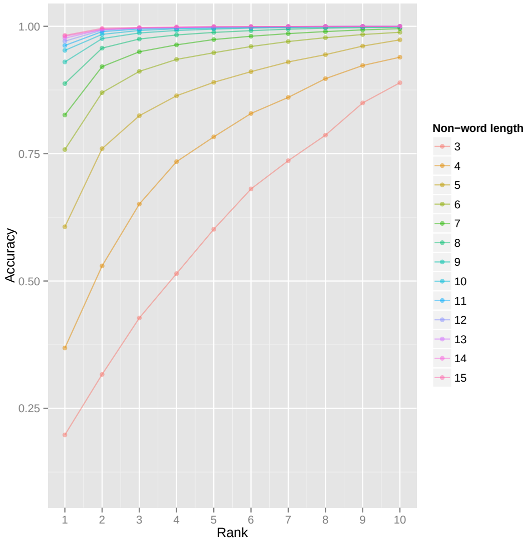
Edit	Probability	Non-word
ri $\rightarrow$ r	0.18	brck
i $\rightarrow$ e	0.15	breck
ic $\rightarrow$ is	0.14	brisk
ri $\rightarrow$ re	0.13	breck
c $\rightarrow$ s	0.10	brisk
ic $\rightarrow$ i	0.10	brik
ri $\rightarrow$ ry	0.06	bryck
ri $\rightarrow$ ra	0.06	brack
ck $\rightarrow$ c	0.04	bric
c $\rightarrow$ co	0.04	bricok

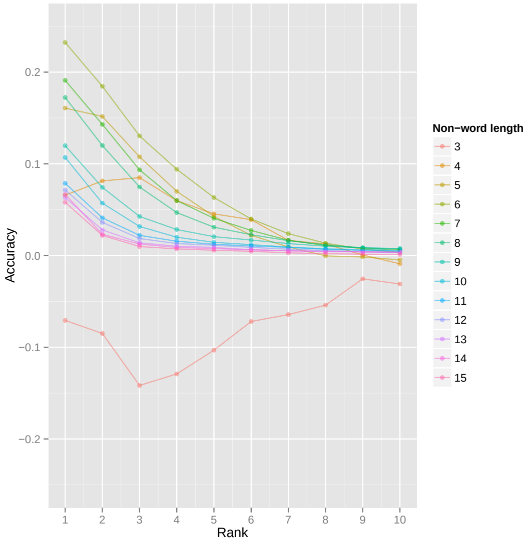
Table 2: Creating non-words from “brick” by sampling transformations with probability proportional to their frequency in curated corpora.

Random Forest RANK

Inputs	Description
All candidates.	Length of candidate list.
Candidate only.	Unigram probability (Google 1TB 5-gram Corpus).
Non-word and candidate, separately.	Length of string. Number of consonants, vowels, or capitals in string. Whether string contains space. Bag-of-words bigrams of string.
Tuple of non-word and candidate.	Levenshtein of strings or phonetic encodings. Damerau-Levenshtein of strings or phonetic encodings. Hamming of strings or phonetic encodings. Jaro of strings or phonetic encodings. Jaro-Winkler of strings or phonetic encodings.

Table 3: Feature set of Random Forest RANK model.


Accuracy
1.00
0.75
0.50
0.25
Rank
1
2
3
4
5
6
7
8
9
10
Non−word length
3
4
5
6
7
8
9
10
11
12
13
14
15


Accuracy
0.2
0.1
0.0
−0.1
−0.2
Rank
1
2
3
4
5
6
7
8
9
10
Non−word length
3
4
5
6
7
8
9
10
11
12
13
14
15

Corpus	Rank	Jaro-Winkler	ConvNet	Random Forest
Aspell	1	0.43	0.48	**0.64**
Aspell	2	0.58	0.66	**0.73**
Aspell	3	0.70	0.73	**0.78**
Aspell	4	0.77	0.77	**0.79**
Aspell	5	**0.81**	0.80	0.80
Birbeck	1	0.28	0.30	**0.39**
Birbeck	2	0.40	0.44	**0.46**
Birbeck	3	0.48	**0.50**	**0.50**
Birbeck	4	0.52	**0.53**	0.52
Birbeck	5	**0.56**	0.55	0.54
Holbrook	1	0.22	0.18	**0.36**
Holbrook	2	0.34	0.27	**0.45**
Holbrook	3	0.42	0.33	**0.50**
Holbrook	4	0.48	0.38	**0.52**
Holbrook	5	0.52	0.42	**0.53**
Wikipedia	1	0.67	0.72	**0.82**
Wikipedia	2	0.77	0.85	**0.90**
Wikipedia	3	0.87	0.89	**0.92**
Wikipedia	4	0.92	0.91	**0.93**
Wikipedia	5	**0.94**	0.92	**0.94**

Context-dependent Non-word Correction Models

- 1-4 Gram language model
  - Google Web 1T 5-gram Corpus
  - Good-Turing discounting
  - Backoff
- Feed-forward Word Embedding Network
- Word ConvNet
- Word and Character ConvNet (with $\mathcal{N}(0, \sigma)$ added to non-word)

Feed-forward Word Embedding Block

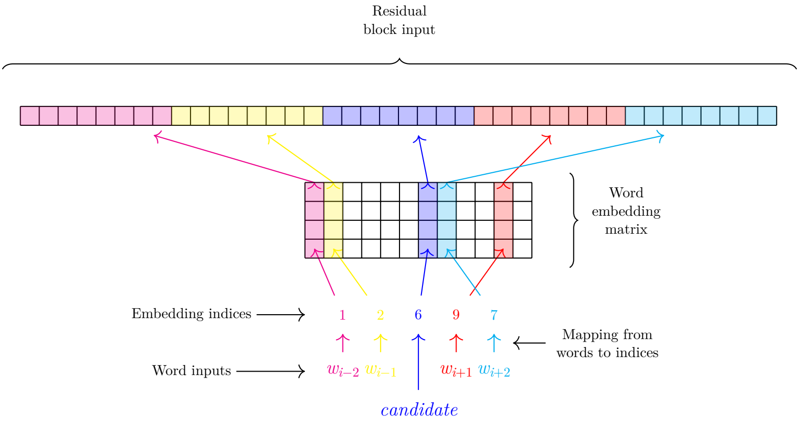


Figure 1: Window of 5 words centered on error, with non-word replaced by candidate. Word embeddings are 200 dimensions (=1000-dimensional input to fully-connected layer).

Word Convolutional Block

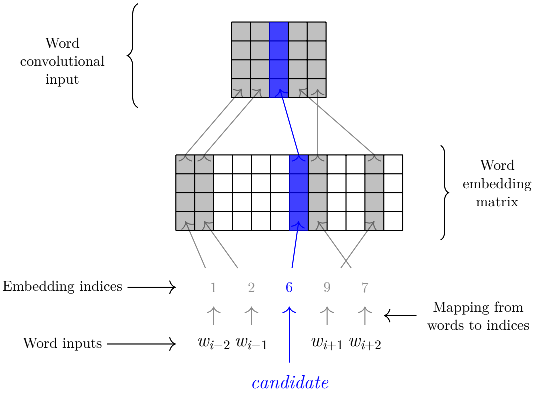


Figure 2: Window of 5 words centered on error, with non-word replaced by candidate. Filters are width 5; there are 1000 of them (=1000-dimensional input to fully-connected layer).

Character Convolutional Block

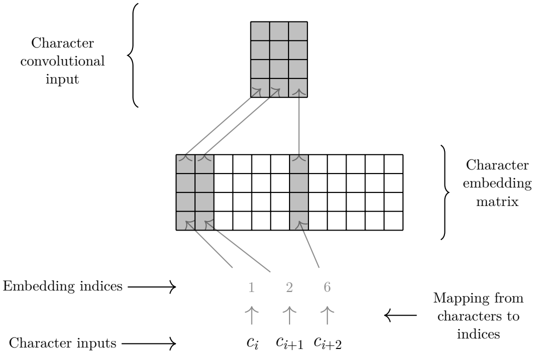
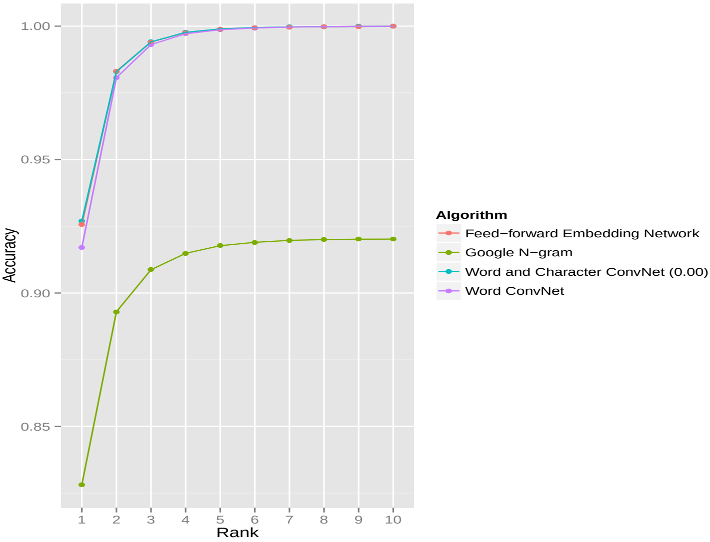


Figure 3: This block has 100 filters of width 3 (=200-dimensional input to fully-connected layer, 100 for non-word, 100 for candidate). It is applied separately to the non-word and the candidate word. A sample from $\mathcal{N}(0, \sigma)$ is added to the non-word after the convolutional operation.


Accuracy
1.00
0.95
0.90
0.85
1 2 3 4 5 6 7 8 9 10
Rank
Algorithm
Feed−forward Embedding Network
Google N−gram
Word and Character ConvNet (0.00)
Word ConvNet

	RANK		
σ^2	1	2	3
.00	.93	.98	.99
.01	.93	.98	.99
.02	.95	.99	.99
.03	.95	.99	.99
.04	.95	.99	.99
.05	.95	.99	.99
.06	.95	.99	.99
.07	.94	.99	.99
.08	.94	.99	.99
.09	.93	.98	.99
.10	.95	.99	.99

Table 4: Effect of varying the noise $\mathcal{N}(0, \sigma^2)$ with which the Word and Character ConvNet was trained.

Real Word Error Correction

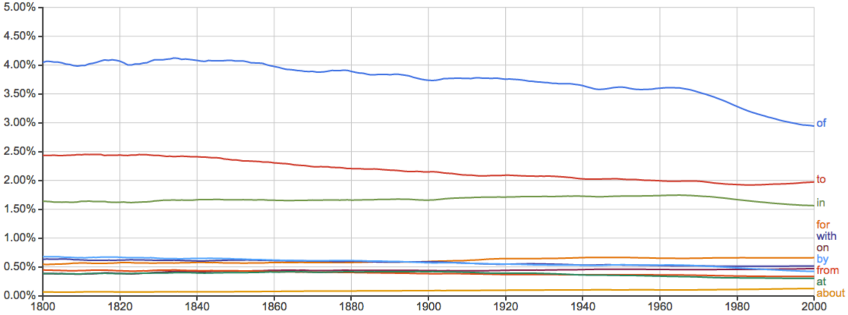


Figure 4: Relative frequencies of 10 prepositions. Our confusion set consists of all but the least frequent, “about”.

Contrasting Cases

Sentence	Target
This is justified **on** policy grounds.	**on**
This is justified **for** policy grounds.	**on**

Table 5: Training with contrasting cases prevents the trivial solution and forces the model to focus on the context of the preposition.

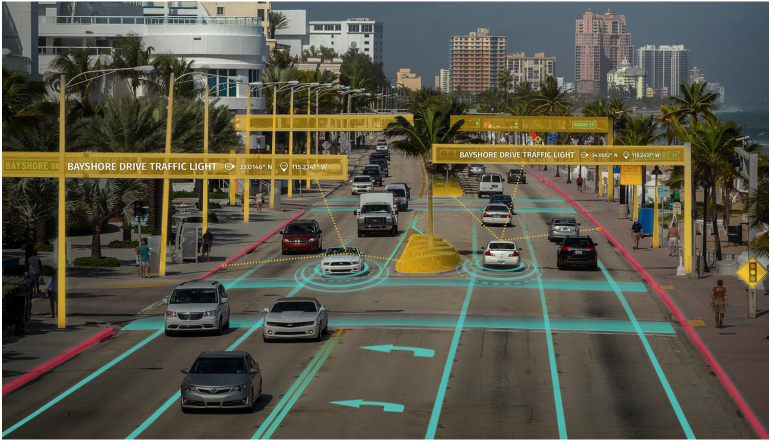
Human Judgments

	A1	A2	A3	A4
A2	.83	.	.	.
A3	.72	.79	.	.
A4	.70	.79	.77	.
ConvNet	.75	.78	.76	.75

Table 6: Cohen's κ of human annotators (A1-A4) and the ConvNet on Wikipedia test set examples. $N \sim 175$ for annotator-annotator κ, $N = 500$ for annotator-ConvNet κ.

HERE™ HD Live Map

Autonomous vehicles require very high accuracy (HD) maps that are updated frequently (Live).



Basic Strategy

1. Collect rich data with vehicles equipped with a variety of sensors, including inertial measurement unit (IMU), global navigation satellite system (GNSS), 4-perspective camera, and LiDAR.
2. Use convolutional neural networks to detect and localize features in imagery and point clouds.

Python and NumPy Self Assessment

Given $\mathbf{M} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^{n \times k}$, compute the matrix product $\mathbf{Mx} \in \mathbb{R}^{m \times k}$ twice – first using only native Python, then using NumPy.