CS 231n (2017)

perception in cats:

Simple cells - respond to oriented edges.

History

SUMMER PROJECT - MIT 1966

Reconstruction of vision system

David Marr (Vision Book 1970s)

Input image → Edge image → 2½-D sketch

↓

3 D model

Image segmentation → Normalized cut (Shi & Malik 1997)

Face detection - Viola & Jones 2001

used Adaboost

SIFT & Object Recognition, David Lowe 1999

→ feature based obj detection.

Spatial Pyramid matching, Lazebnik 2006

+ SVM

Human detection

Histogram of gradients, (2005)

Deformable Body part Model 2009

PASCAL VOC (visual Object Recognition) (20 do...

IMAGENET 22K class, 14M images

down using Amazon Mechanical Turk platform (crowdsourcing platform)

IMAGENET LSVRC → 1k class
1.4M images

Image classification using

i) KNNs:

take L1 or L2 distance between images and group classes by K value. L1 is coordinate dependent which L2 works generally.
Not ideal as O(N) for test can and can't understand subtle changes in image.

ii) Linear classifin.

$32 \times 32 \times 3 \longrightarrow f(x, W) \rightarrow 10$

$Wx + b$
W: 10×3072
x: 3072×1

Sathvik Hegde

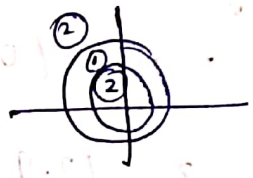
Linear classifier just averages all variances in input & comes up with one template to identify a class.

can imagine it as drawing linear boundaries b/w different categories

difficulty in cases such as

class 1: 1 <= L2 norm <= 2

class 2: everything else



can it classify.

Multiclass SVM loss:

	cat image	car image	frog image
cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

S_{y_i} → true class score

S_j → every class score

$$\text{Loss } L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } S_{y_i} \geq S_j + 1 \\ S_j - S_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, S_j - S_{y_i} + 1)$$

true score should be higher than all the other scores by a safety margin (1)

compare score of correct category and sum of scores of other categories

this type of loss fn is called Hinge loss

for cat:

$L_i = \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$

$= 2.9 + 0 = 2.9$

for car:

$L_i = 0$

for frog:

$= \max(0, 2.2 + 3.1) + \max(0, 2.5 + 3.1)$

$= 12.9$

$L = (2.9 + 0 + 12.9)/3 \approx 5.3$

Softmax can be viewed as KL divergence b/w target & computed y or as maximum likelihood estimate.

It is easier to maximize log than the prob distribution $= -\log\left(\frac{e^k}{\sum e^j}\right)$

	cat ins	exp			
cat	3.2		24.5	→	0.87
car	5.1	→	164		0
frog	-1.7		0.18		

0.13 -log 0.89

L) if all are small or zero, then softmax output should be log c. (sanity check)

c → no. of classes.

optimisation :

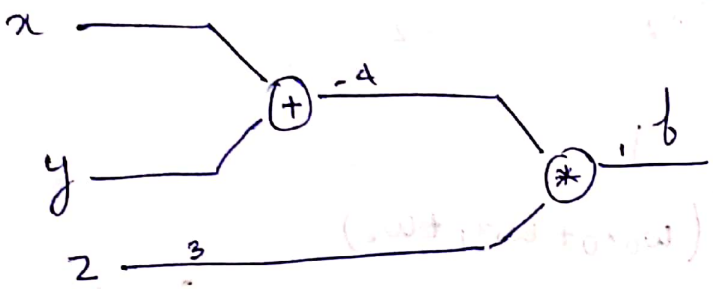
The slope of in any direction is the dot product of direction with gradient.

The direction of steepest gradient descent is the negitive gradient.

Backpropagation.

Eg:

$$f(x, y, z) = (x + y)\, z$$



$q = x + y$

$f = qz$

$\frac{\partial q}{\partial x} = 1$ $\quad \frac{\partial q}{\partial y} = 1$

$\frac{\partial f}{\partial q} = z$ $\quad \frac{\partial f}{\partial z} = q$

let $x = -2$

$y = 5$

$z = -4$.

to do backprop

start with $\frac{\partial f}{\partial f} = 1$

then $\frac{\partial f}{\partial z} = q = 3$

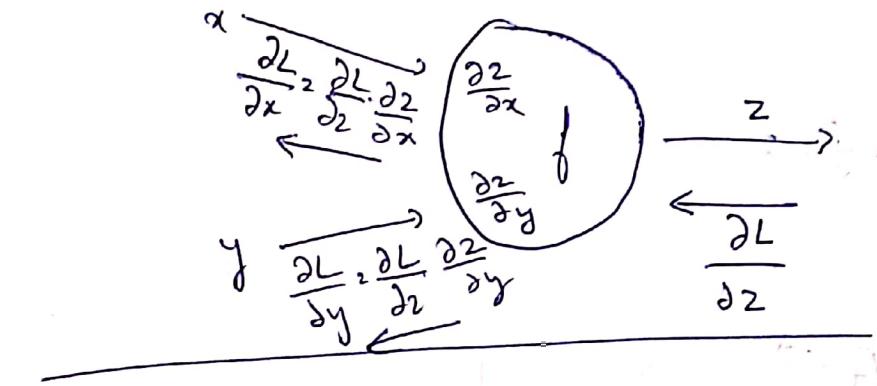
$\frac{\partial f}{\partial q} = z = -4$

then $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \times \frac{\partial q}{\partial y}$

$= 2 * y z - 4.$

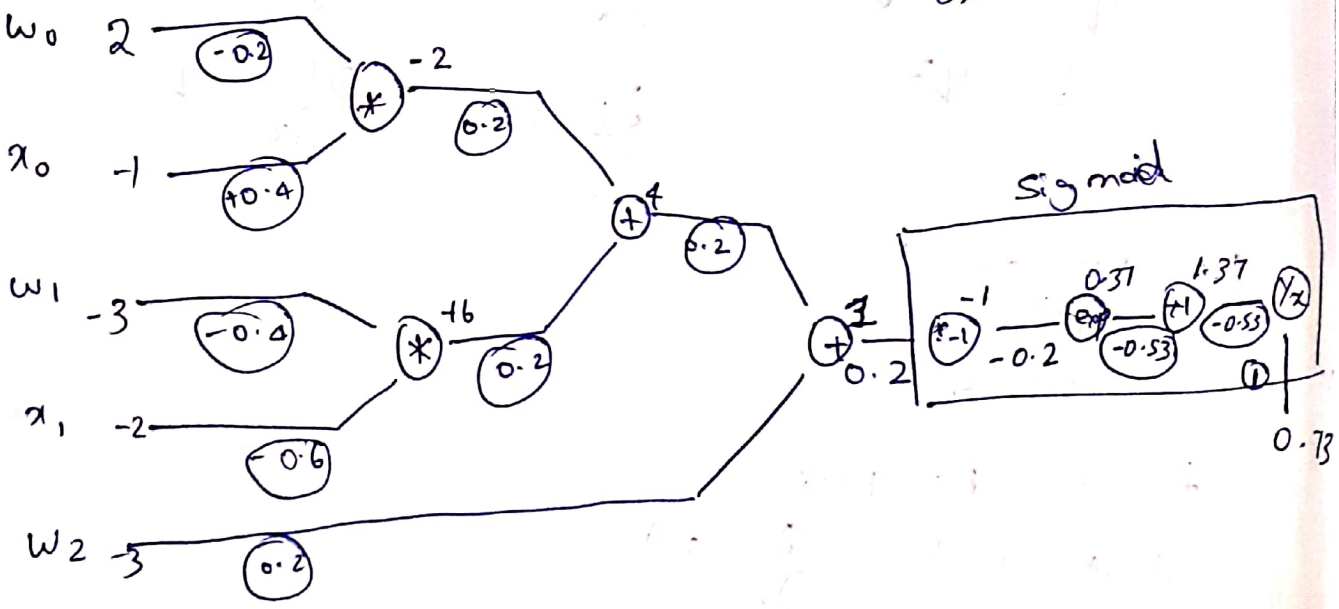
$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \times \frac{\partial q}{\partial x}$

$= 2x \times = -4$



$f(w,x) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$

$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$



$1 \times \frac{-1}{1.37^2} = -0.53$

$e^x (0.53) = e^{-1} \times (-0.53)$

for each node,
upstream gradient
× local gradient

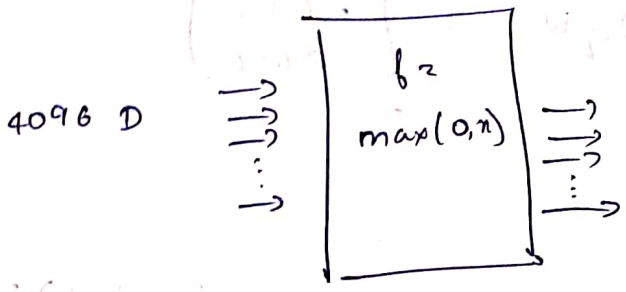
For addition, same gradient is passed down to all nodes

max gate → gradient routing (send to max i/p value)
mul gate → gradient switcher (take other value & mul gradient)
add gate → gradient distributor.

On backprop when upstream nodes converge to one, then gradients add up.

---

if



$\frac{\partial L}{\partial x} = \frac{\partial f}{\partial x} \cdot \frac{\partial L}{\partial f}$

$\frac{\partial f}{\partial x} \rightarrow 4096 \times 4096$ Jacobian

But only the diagonal elements in jacobian contribute to output.

$f(x, w) = ||w \cdot x||^2 = \sum_{i=1}^{n} (w \cdot x)_i^2$

$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix} w$

$\begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix} x$

$q = w \cdot x$

$\begin{bmatrix} 0.22 \\ 0.26 \end{bmatrix}$

$\begin{bmatrix} 0.44 \\ 0.52 \end{bmatrix}$

0.116

$w_{11}x_1 + \dots + w_{1n}x_n$

$w_{n1}x_1 \dots w_{nn}x_n$

$f(q) = ||q||^2 = q_1^2 + q_2^2 + \dots q_n^2$

$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$\nabla_q f = 2q = \begin{bmatrix} 2\times 0.22 \\ 2\times 0.26 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.52 \end{bmatrix}$$

$$\frac{\partial f}{\partial w_{ij}} = 2q_i x_j = 2q \cdot x^T$$

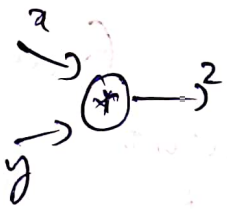
$$= \begin{bmatrix} 0.088 & 0.104 \\ 0.176 & 0.208 \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = 2qW = \begin{bmatrix} -0.112 \\ 0.636 \end{bmatrix}$$

$\Rightarrow$

coding them

Ex:




```
class mulgate(object):
    def forward(x,y):
        z = x * y
        return z
    def backward(dz) ← ∂L/∂z
        dx = ...
        dy = ...
        return [dx, dy]
                  ↑
                ∂L/∂x
```

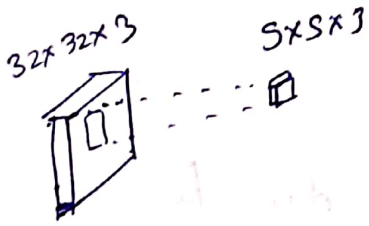
CNNs

Hinton 2006 -

backprop works with proper initialisation.

layers pre-trained using R Boltzman Machine

Filters always go down to full depth of image.



Conv filt not flipped (not like signal processing)

Why not sigmoid as activation?

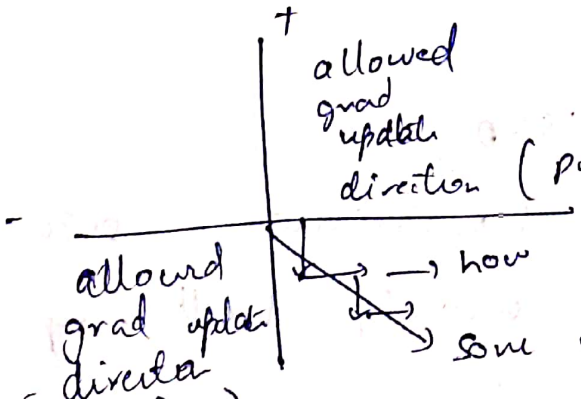
Saturated neurons kill grads

not zero centred

exp() is computationally exp.

if i/p is always +ve or -ve

then grad update will happen in same direction.



that's why zero mean data is preferred.

tanh(x)

kills neurons grads on saturation

Relu

Convergence (6x)

Does not saturate

Computationally efficient.

Not zero centered

Relu might not work well due to bad initialisation & high learning rate

Pre processing.

- zero mean is required.
- normalisation not really for images.
- mean image or per channel mean

initialisation

if $w = 0$, all will be same

Xavier Initialisation paper & He et al. 2015 & BN paper 2015

Why is SGDM > SGD

at saddle point / minimas, even though grad is zero, velocity built up gives it going

masking & saliency Maps to visualize importance

also guided backprop

gradient ascent to regularize generated image.

Adversarial - move on it

For efficient Inference:

Pruning
weight Sharing
Quantisation
Low Rank Approx.
Binary / Ternary Net
Winograd Transform

CS224N - NLP

Wordnet (nltk package available) → list of synonyms, hypernyms etc.

localist representation:

lexicon based models used before deep learning

Why not use one hot encodings?

→ large vector size

→ no context.

Distributional semantics

A words meaning is given by the words that frequently appear close by.

Vector Representation

Word2Vec

start with a centre word and predict context words around it within a range.

Using this the vector representation is learnt.

Likelihood $L(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m} P(w_{t+j} | w_t; \theta)$

T → total no of words

m → range.

To maximise prediction acc → minimise objective function.

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m} \log P(w_{t+j} | w_t ; \theta)$$

log changes product to sum of log.

How to calculate $P(w_{t+j} | w_t ; \theta)$?

1 vector for word w when it is center v_w

1 vector for word w when it is context u_w

for a new center word c, context word o,

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$ → normalize over entire vocab

$u^T v$ compares similarity of o and c.

larger dot product = larger probability

Compute grads of all vectors of words

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum \exp(u_o^T v_c)}$$

$$= \frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) - \frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp u_o^T v_c$$

$$\frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

$$\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_o^T v_c)$$

$$= \frac{1}{\sum_{w=1} \exp(u_o^T v_c)} \cdot \frac{\partial}{\partial v_c} \sum_{i}^{V} \exp(u_i^T v_c)$$

$$= \frac{1}{\sum_{w=1}^{V} \exp(u_o^T v_c)} \cdot \sum_{x=1}^{V} \exp(u_x^T v_c) \cdot \frac{\partial}{\partial u} u_x^T v_c$$

$$\therefore \frac{\partial}{\partial v_c} \log P(o|c) = u_o - \frac{\sum_{x=1}^{V} \exp(u_x^T v_c) \cdot u_x}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

$$= u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T v_c)}{\sum \exp(u_w^T v_c)} u_x$$

$$= u_o - \sum_{x=1}^{V} P(x|c) \cdot u_x$$

↓ observed rep of the context word

↓ what model thinks model should look like

- two vectors are used for easier optimisation. They are averaged at the end.
- It can also be done with one vector only

2 variants:

Skip gram: predict context based on center

Bagg of words: predict center based on all context

Negative sampling: for faster word2vec

- $P(o|c)$ denom takes too much time
- train binary logistic regressions for a true pair (center & context) versus many noise pairs (center with a random word)

$J_{-ve\ sampl}(o, v_c, U) = -\log(\sigma(u_o^T u_c)) - \sum_{k=1}^{K} \log(\sigma(-u_k^T u_c))$

- take k -ve samples.
- Maximise prop that real outside word appears, minimise prob that randn words appear around the centn.

window based co-current matrix

→ large size
→ Sparse
→ less robust models

→ Reduce dimension

Eg: SVD

→ for common words, use log & ceil fns.

→ peirson corr instead of counts, -ve to zero.

glove

- adding co-current (statistics) to context.
- Ratios of co-current prob can encode info
- capturing ratios of co-current prob as linear meaning components in word vector space

log bilinear model $w_i \cdot w_j = \log P(i|j)$

$$w_x \cdot (w_a - w_b) = \log \frac{P(x|a)}{P(x|b)}$$

for glove,

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

b_i and $\tilde{b}_j$: biases for each word

$\log X_{ij}$: to count.

- fast training
- scalable to huge corpus
- good performance even with small corpus & vector

good hyperparameter for glove:

→ dim 300
→ window size 8

NER

why it is hard?

- It can be difficult to know if something is an entity
- It can be difficult to know if words leading & prevailing an known entity should be considered as entity as well.

different ways:

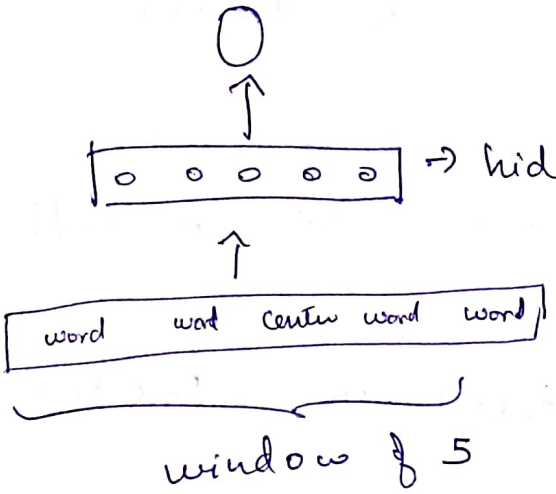
window classification: Softmax.

take a centre word & ±k word, take word vectors of the (2k+1) word window & classify

window, using score:

same idea as before, but let the model get a

single no in the last layer, a high value in case the center word of window is a required entity, a low value if not, train it in such a way.



$s = u^T h$

$h = f(z)$

$z = wx + b$

x

$$\frac{\partial u^T h}{\partial u} = h^T$$

to backprop

$s = u^T h$

$h = f(z)$

$z = wx + b$

x

$$\frac{\partial s}{\partial b} = \frac{\partial s}{\partial h} \frac{\partial h}{\partial z} \frac{\partial z}{\partial b}$$

$$= h^T \text{diag}(f'(z))I$$

$$= h^T \circ f'(z)$$

$$\frac{\partial s}{\partial w} = \frac{\partial s}{\partial h} \frac{\partial h}{\partial z} \frac{\partial z}{\partial w}$$

$$= h^T \text{diag}(f'(z)) x^T$$

Disadvantage of retraining word vector

→ Ex: Say TV, telly & television are in close space in original word vector Representation. Our data has only TV & telly in it. Then retraining might push TV & telly somewhere else losing its context to television

→ If dataset is small, Don't train word vectors
→ If dataset is large (1M words +) then retrain/fine-tune.

← dependencies & treebanks (Lec 5) →

Language Modelling

- Predicting what word comes next.
- $P(x^{(t+1)} \mid x^{(t)}, x^{(t-1)}, x^{(t-2)} \ldots)$ $x^{(t+1)} \in V$.

Pre-dup learning:

N-grams.

→ assume $x^{(t+1)}$ depends only on the preceding $n-1$ words

$$= P(x^{(t+1)} \mid x^{(t)}, \ldots, x^{(t-n+2)})$$

$$= \frac{P(x^{(t+1)}, x^{(t)}, \ldots)}{P(x^{(t)}, \ldots)}$$

← prob of n gram (Conditional prob)

→ prob of n-1 gram

These probs are obtained by count of appearence.

3) 4-gram model

as the proctor started the clock, the students opened their _____

∴ n-1 = 3

P(w | students opened their)

$$= \frac{\text{count(students opened their w)}}{\text{count(students opened their)}}$$

-> The answer could be exam if procto context was taken

-> if the event has not happened, then it would cod. to remove this, add small δ to all

-> if denom is zero, then back off - 4 gram -> 3 gram. (opened their)

-> increasing n increase model size

-> can generate text. ed

NN band

-> fixed window.

window size = 4.

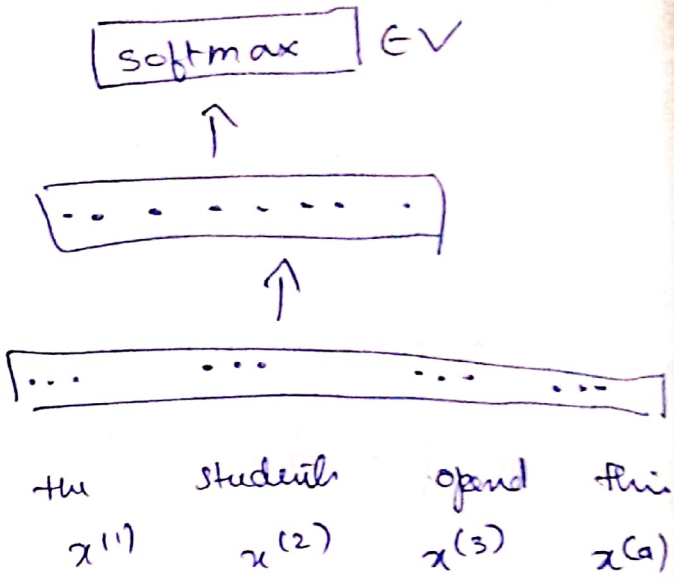
~~the students open~~

$\hat{y} = \text{softmax}(Uh + b_2)$

$h = f(We + b_1)$

$e = [e^1\ e^2\ e^3\ e^4]$ word embeddings

$x^1\ x^2\ x^3\ x^4$



→ advantages

- No sparsity problem
- No need to store all observed n-grams

→ disadvantages

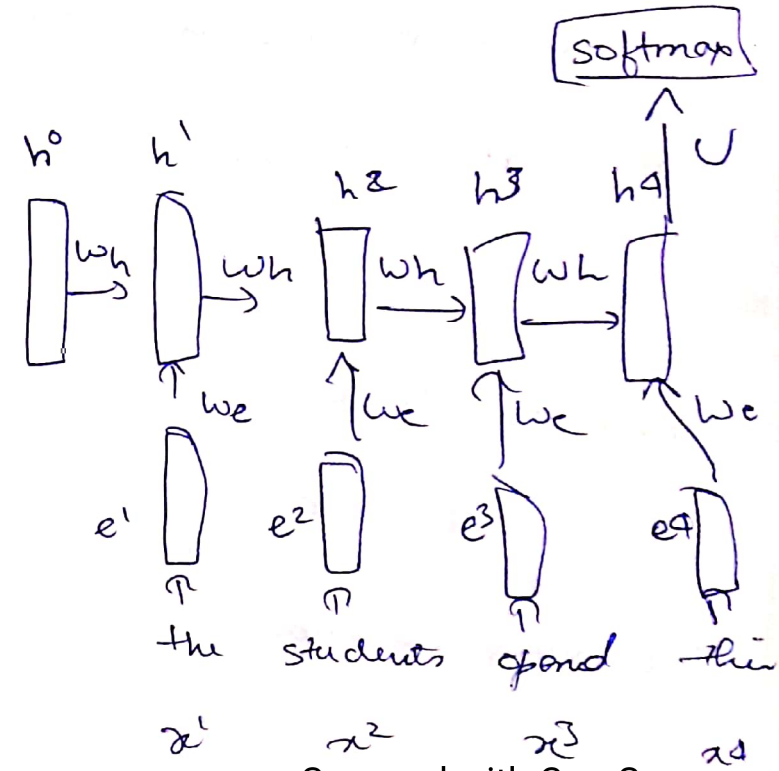
- window is small
  - → enlarging window increase weight W
  - → for each embedding input, the weight matrix doesn't share its weight

RNNs

$e^{(t)} = Ex^{(t)}$

$h^{(t)} = f(W_h h^{(t-1)} + We^{(t)} + b_1)$

$y^{(t)} = \text{softmax}(Uh^{(t)} + b)$



→ advantages:

- any length
- information transfer through many layers (Hierarchy)
- model size doesn't increase for longer i/p
- same weights on every timestep

→ disadvantage

- pretty slow, can't do in parallel

Standard evaluation metric → perplexity

Inverse prob of corpus, according to model

$$= \prod_{t=1}^{T} \left(\frac{1}{P_{LM}(x^{(t+1)} \mid x^{(t)} \cdots x^{(1)})} \right)^{1/T}$$

$$= \exp(J(\theta))$$

lower perplexity is better

gradient vanishing

- due to small Δ values everywhere
- if largest eigenvalue of $w_n < 1$, then grad will shrink exponentially. (1 reason sigmoid is used)
- $w_n > 1$, then exploding grads.
- due to this only nearby effects are taken into consideration.

LSTM vs GRU

- GRU is faster & less parameter
- No other notiable changes
- start with LSTM.

in Bi Di RNN

Forward & Backward o/ps are concatenated.

Machine Translation.

On the part:

find best English y for given french x.

$$\text{argmax}_y \; P(y|x)$$

Bayes Rule: $= \text{argmax}_y \; P(x|y) \; P(y)$

$P(x|y)$: Models how words & phrases should be translated

$P(y)$: Models how to write good english

$$P(x|y) \rightarrow P(x, a|y)$$

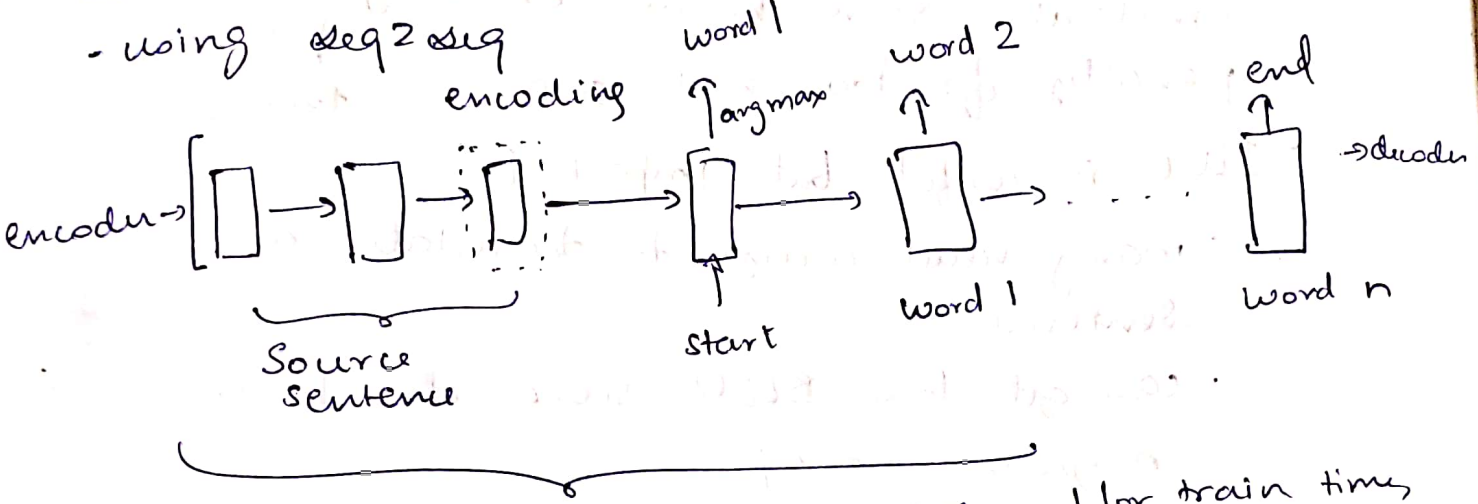
a is the alignment, word level correspondence b/w 2 languages.

- probability of a particular word aligning
- Probability of particular word having particular fertility (no of corresponding words)

the argmax is found by a heuristic search algo to search for best translation, disregarding hypotheses that are low-prob.
→ called decoding

Neural Machine Translation (NMT)

- using seq2seq



test time architecture (for train time, computations for each word in decoder)

many tasks by seq to seq
- Summarisation
- Dialogue
- Parsing
- Code generation

greedy decoding → argmax on every decoder output

To fix this, Beam search is used

keep track of k most probable partial translations
k usually ∈ [5,10]

- final Beam search prob output

$$= \frac{1}{t} \sum \log P(y_i | y_1, y_2 \cdots y_{i-1}, x)$$

Evaluation:

BLEU

- similarity score based on n-gram precision
- penalty for too short system translations
- BLUE is useful but imperfect
  - many valid ways to translate a sentence
  - can get low BLEU score due to less n-gram overlap

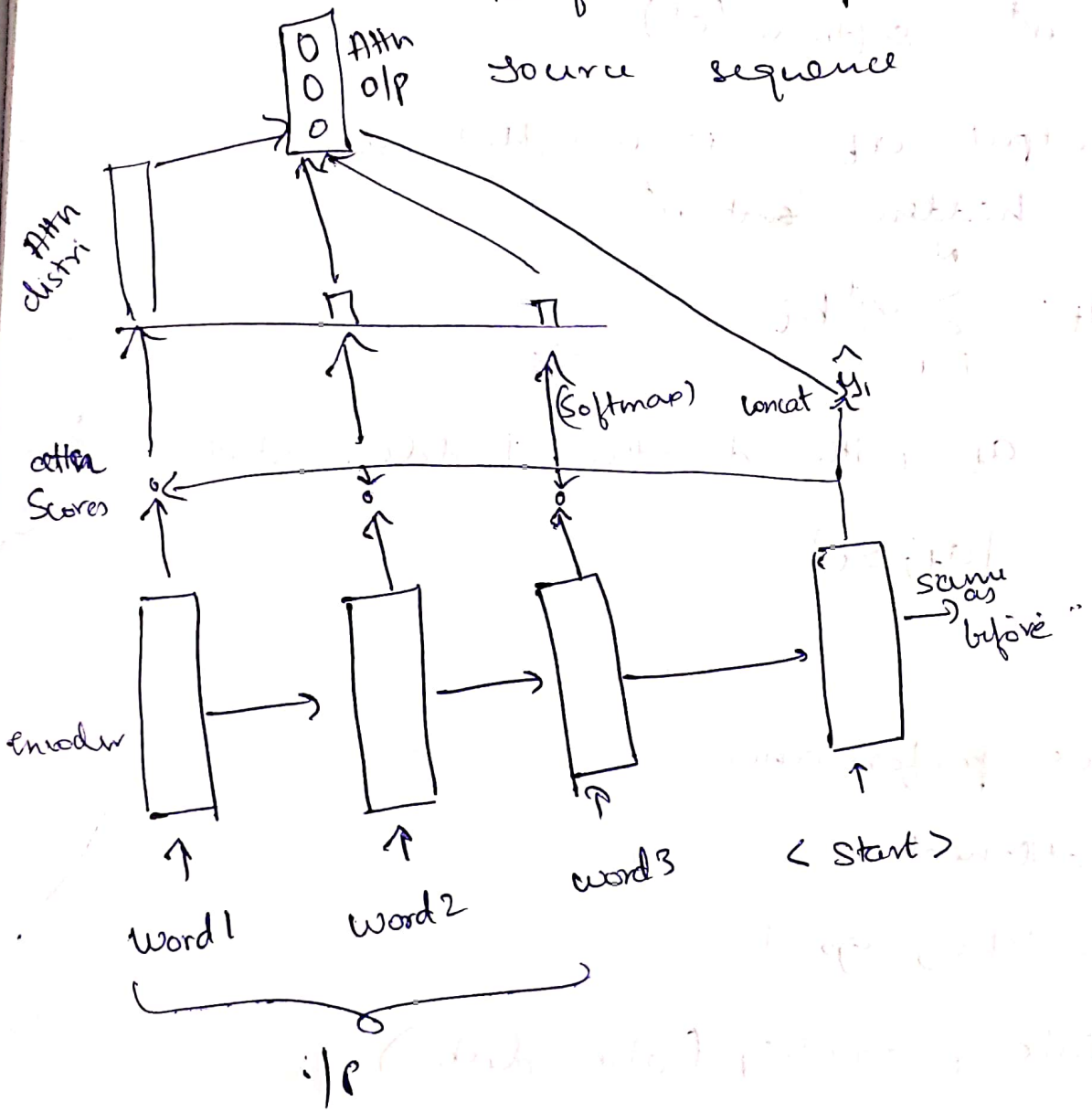
Difficulties

- out of vocab.
- Domain mismatch (train / test)
- Context over longer text
- low-resource language pairs
- no common sense
- biases in training data (Eg: gender)
- nonsensical i/p can give weird outputs (somali → english agagag)

Why attention ?

info bottleneck on decoder i/p

- all info needs to be captured by last encoding vector
- on each step of decoder, use direct connection to encoder to focus on particular part of source sequence



Attn equations

- encodr hidden $h_1, h_2, h_3 \dots h_N$
- on timestep t, decodr hidden s_t
- attn scores e^t,

$$e^t = [s_t^T h_1, s_t^T h_2, \dots, s_t^T h_N]$$

- softmax to get attn distribution α^t

$$\alpha^t = \text{softmax}(e^t)$$

- attn output a_t is weighted sum of encodr hidden ~~sum~~ α^t.

$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i$$

- concat a_t with decodr hidden stat

$$[a_t; s_t]$$

advantages

- improves performance
- no bottleneck
- no vanishing grad
- some interpretability (attn distri)

Variants

- Multiplicative attn

$e_i = s^T W h_i$

$W \rightarrow$ learnable

- Additive

$e_i = v^T \tanh(W_1 h_i + W_2 s)$

~~$W_1, W_2, v \rightarrow$ learnable~~

~~$v \rightarrow$ weight~~

$d_3 \rightarrow$ hyperparameter

all learnable

$W_1 \in R^{d_3 \times d_1}$

$W_2 \in R^{d_3 \times d_2}$

~~v~~ $v \in R^{d_3}$

Data

look at linguistic Data Consortium.

Sn - Ans

gained traction with MCTest Corpus → 2013

CNN/DM, SQUAD → 2015/16

IBM Jeopardy → 2011 → ensemble of many methods

SQUAD 1.0

- 100k examples
- passage, qn and ans
- ans must appear in passage (not yes or no)
- used Amazon Mechanical Turk.

SQUAD 2

dw & test set, half the qns has no answer

Stanford Attn Reader

for each qn: for represent using BiLSTM hidden state o/p.

for each passage, again use BiLSTM, then add attention from the qn vector

$$\alpha_i = \text{softmax}(q^T W_s \tilde{p}_i)$$

q^T → question dp vector
$\tilde{p}_i$ → passage hidden BiLSTM
W_s → learnable weight

→ predict start token

$$\alpha_i' = \text{softmax}(q^T W_s' \tilde{p}_i)$$

→ predict end of answer

SAR ++

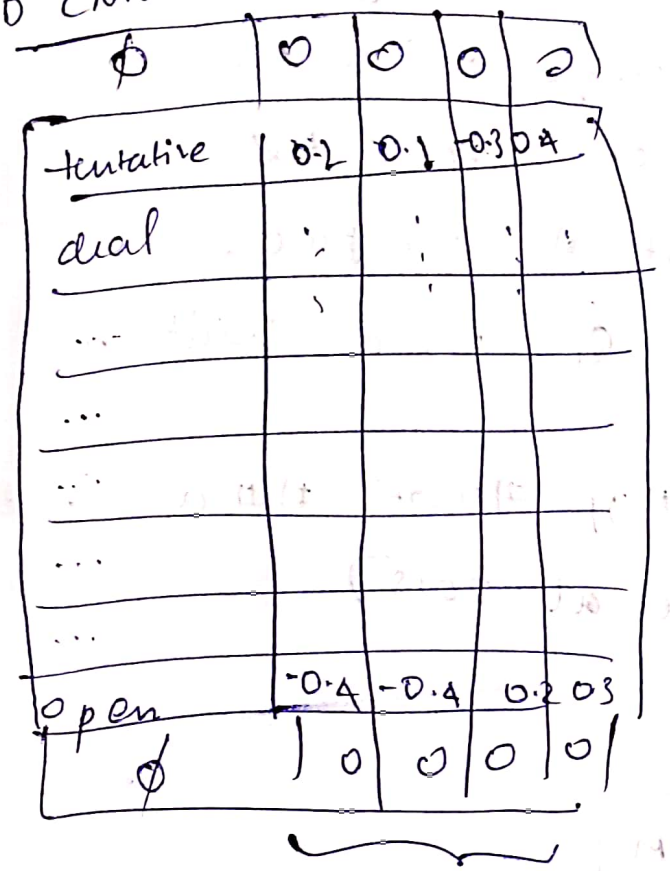
Get representation from all hidden units for qn attention. Much more complex model.

Vector Rep of each token in passage
- word embedding
- POS & NE tags, one-hot encoded
- term frequency
- whether word appears in question (exact, uncased, lemma)

- Aligned gen embeddings ('car' vs 'vehicle')
(sort of word similarity)

Conv for Text

1D CNN



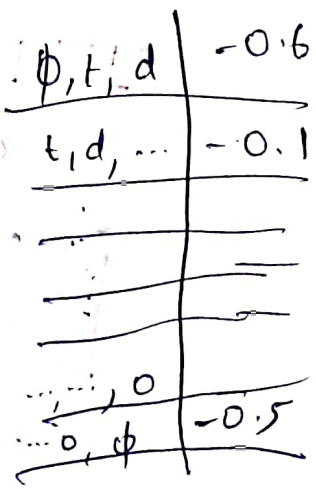
4D word vector.

Apply filter of size 3.

$$\begin{bmatrix} 3 & 1 & 2 & -3 \\ -1 & 2 & 1 & -3 \\ 1 & 1 & -1 & 1 \end{bmatrix}$$

use many filters to increase out dim

o/p



Quasi RNN

Combining CNN & RNN.

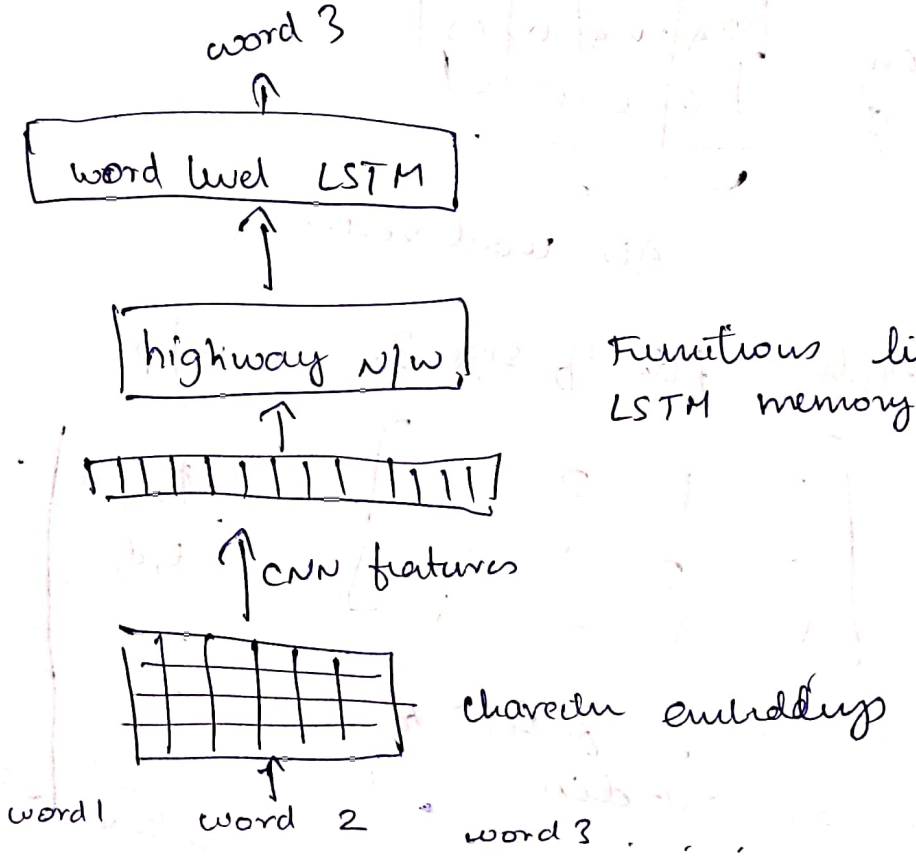
Wordpiece / Sentencepiece models

- variant of Byte pair encoding
- wordpiece model tokenizes inside words.
- sentencepiece works on raw text
- BERT uses a variant of wordpiece, has common words & words built on wordpieces.

hypatia = h ##yp ##ati ##a (in Bert)

character

word-level models (Yoon et al 2015)



- character level model works for out of vocab words
- CNNs + Highway N/w over characters can extract rich semantic & structural information.

Fast Text

- further work on word2vec
- took word2vec skip gram & added character n-grams
- where = <wh, whe, her, ere, re> ; <where>
- for center words in word2vec, all 6 of them are used

Contextual word embedding

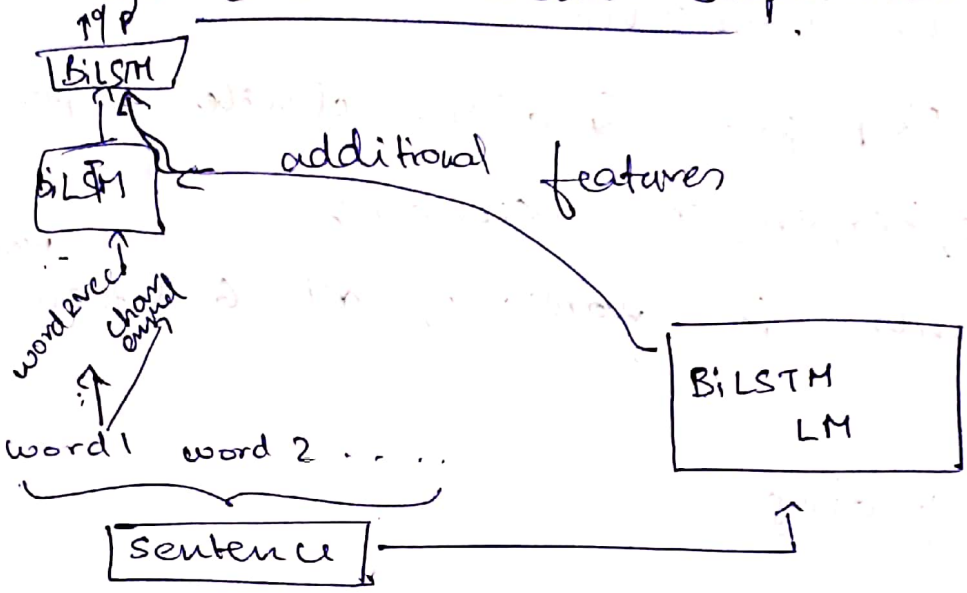
for unknown words:

- one idea is to train them as one vector <UNK>
- character embeddings as before
- if a <UNK> at test appears in embeddings, use it in test time
- for other words, assign a random vector, adding it to vocab
- Collapsing things to word class, like unknown number, thing etc

disadvantage of word vectors

- same word can have diff. meaning
- they can have diff parts of speech,
  eg: a ~~to~~ noun or a verb (arrive, arrival)

TagLM (2017) - semi supervised for NER



ELMo (similar structure to TagLM)

→ 2 Bi LM (char CNN → BiLSTM + ~~learned~~ pre-trained LSTM)

→ Only character CNN for initial word Representation
(2048 char n-gram & 2 highway layers, 512 dim projection

→ 4096 dim LSTM → 512 linear

→ Residual

→ parameter typing of token i/p & o/p

→ learns task specific combination of BiLM
(runs all hidden o/ps)

→ gave almost 3% improvement in a range of NLP tasks

ULMFit

- Train a big unsupervised corpus.
- Fine tune the model in domain you are working in
- Convert the LM to a text classifier
- less data required for other data (transfer learning)
- training → 1 GPU day

other works

GPT	240 GPU days
Bert	320-560 TPU days
GPT-2	2048 TPU v3 days

Transformers

→ The need to parallelise ~~ma~~ training to make it faster, RNNs are sequential.

→ Attention is all you need (original Transformer architecture)

- A complex encoder & decoder using only attention, no recurrent
- inputs: a query q, a set of key-value pairs to an output.
- dot product attn is used → query key similarities

$$A = \sum \frac{e^{q \cdot R_i}}{\sum_j e^{q \cdot k_j}} V_i$$

(softmax) → attn band weight on corresponding values

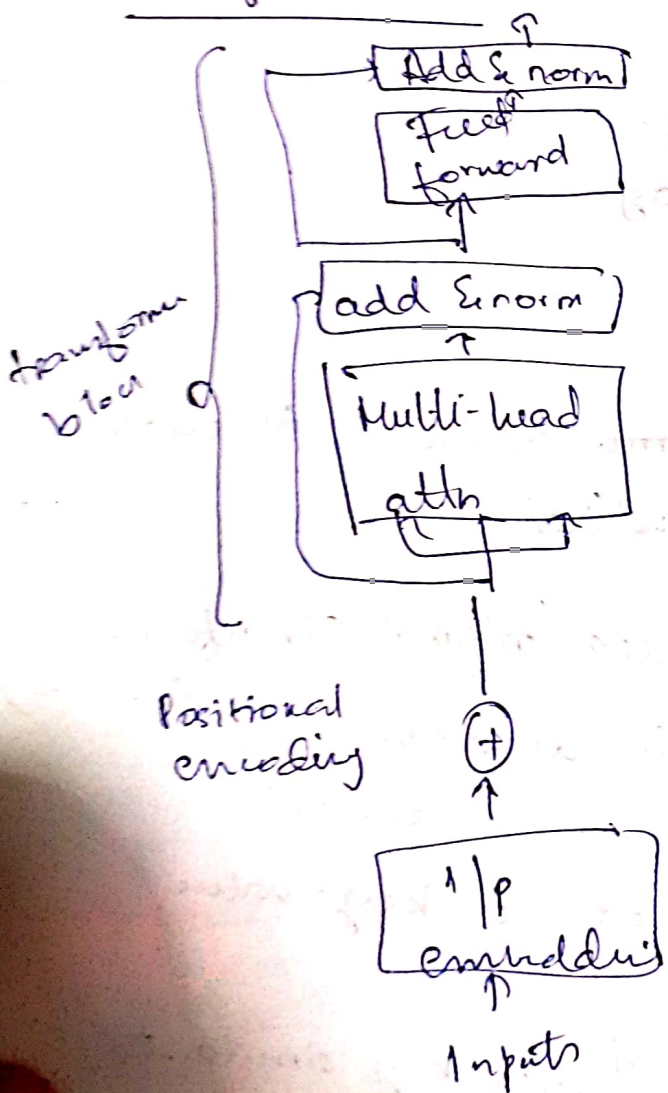
In encoder

-> i/p words are queries, key and value
-> word vectors self themselves

Multihead attn

Because one attn network focusing over a single word, you may need to look for more dependencies

transformer block



For encoder, at each block, same Q, K, V and from previous layer.

- 6 vertical blocks

Why is it soo good?
- for pronouns, attn points to its reference
- verbs points the chang it does (sort of)

Bert
they wanted bidirectional context w/o further words being seen in later hidden layers
- Mask out words in a sentence, then predict the masked words
- they ~~the~~ mask 15% of words

- Next Sentence Prediction

i/p -> pair of sentences, words are wordpiecy positional embeddings, segment embedding (which word is from sentence 1 or 2)

(Wikipedia + BookCorpus)

Bert Base - 12 layer, 768 hidden, 12 head
Bert large - 24 layer, 1024 hidden, 16 head

4x4 or 8x8 TPU for 4 days

Fine-tuning
- same as ULMfit.
- Remove top level (mask level or next sentence)
- Replace with our layer for the task

→ Bert large gave ~2% more than Bert-Base

- attn is permutation invariant
- Positional info added to maintain order.

- On decoder side, it mimics a language model. This is done by imposing causality on positions you can look at. Eg: First position, its illegal to look forward. Otherwise, it copies the i/p.

Multihead attn

- Diff Attn happens in parallels.

  (Intuition - diff things handles questions like who? did what? To whom?)

- Tf model were 3x faster than previous for MT tasks.

- Residuals carry positional info to higher layer
- Tf also used for probabilistic image generation.

Music generation using self-attn

- performance info is taken (not sound).
- initial motif is given & model is supposed to complete.
- Music Attn -> great stuff!!! (Magenta Google AI)
- Relative Attn used for graphs.

- Message Passing neural networks.

Natural language Generation

- Beam search on Wordword!?
- Increasing Beam search k size might decrease BLEU. (large k-beam search produces too short translations)
- In chit-chat (dialogues), low beam size might give nonsensical but on-topic o/p. High beam size might give safe, correct response, by generic & less relevant to the topic

alternative to Beam search

- Sampling based decoding.
- Pure sampling
  - randomly sample to obtain next word
- Top-n sampling
  - Take top n words & sample from that
- Both are more efficient than Beam search.

Softmax temperature

- temperature hyperparameter τ on softmax.

$$P_t(w) = \frac{\exp(s/\tau)}{\sum \exp(s/\tau)}$$

- Raising the ϵ makes Prob distribution more uniform.
  - Thus more diverse O/P.
- lower ϵ, Prob distri becomes spiky.
  - less diverse output.

Summarisation

Single document summarisation:

- gigaword dataset
  - sentences → headline (sentence comprehension)
- LCSTS
  - paragraph → sentence summary
- NYT, CNN/Dailymail
  - news article → sentence summary.
- wikihow
  - full how to article → summary sentences.

Sentence Simplification

- Simple wikipedia
  - Stand wiki sentence → Simple
- Newela
  - news article → written for children

extractive Summarisation

- Select parts to form summary.

abstractive Summarisation

-> generate new text.

Pre-neural methods

-> content selection using tf-idf, graph based

Evaluation

- ROUGE
- based on Recall
- no brevity penalty
- ROUGE-1 : unigram overlap
- ROUGE-2 : bigram overlap

Neural-based

- 2015 Rush et. al
- Seq2seq are good at copying but bad at copying over details.
- Copy based models copy too much.
  - when document is long
  - bad at content selection.

- bottom-up Summarisation
  - neural seq tagging model for content selection
  - seq2seq attn won't attend to words tagged 'don't'

- dul Reinforced summarisation.

→ ROUGE score can't be optimud during training bccacm its not differentiall so RL is usd fo the samm

Dialogue

- Assitive
- Co-operatilc
- Adversarial

} task oriented

- Chit chat
- Therapy

Seq2Seq baud

- boring responms
- Repetitions
- → Irrelevant (for eaitht)

Soln:

- optimis for Maxs mutual info b/w inpat S and supomm T

$$\frac{\log p(S,T)}{p(S)p(T)}$$

- Block repeating n-grams duris beamsearch

Better evaluation of NLG

- Fluency (compute prob wrt well-trained LM)
- Correct Style (prob wrt LM trained on target corpus)
- Diversity (rare word usage)
- Relevant to i/p (Semantic Similarity response)
- human quality ques

Coreference Resolution

- Identify entities that refer to the same real world entity
- Which of the mentions refer to same entity
- Applications:
  - Full text understanding
  - MT (for tracking pronouns)
  - Dialogue systems (carry reference across messages, (James Bond → Spectre))
- Detecting mentions is easy
- Clustering refernce is hard

3 kind of mention

- Pronouns
- Named Entities
  - → Noun phrases

- one way to do this is to train a classifier specifically for mention detection instead of a POS tagger, NER system, parser

Coreference -

2 mentions refer to same entity

Barak Obama went to .... Obama ...

→ both are same

anaphora

term refers to another term

Obama said he went

→ one refers to another

antecedent anaphor

anaphora without reference -

No dancer twisted her knee

bridging anaphora

we went to see a concert. The tickets ...

Cataphora

He was lying, smoking, as was his custom, Lord Henry ....

(need to look forward then)

- Hobbs Naive algo (1976) (tree based)

Winograd Schema

-> Proposed as an alternative to Turing Test (2013)

(if you can solve corefernce, you've solved AI)

Mention pair models

- train a binary classifier that assigns every pair of mentions a probability of being coreferent.

Neural Coref Models

- I/P -> Candidate + Mention embeddings + additional features (statistical)

↓

Hidden layer + Relu

↓

Score

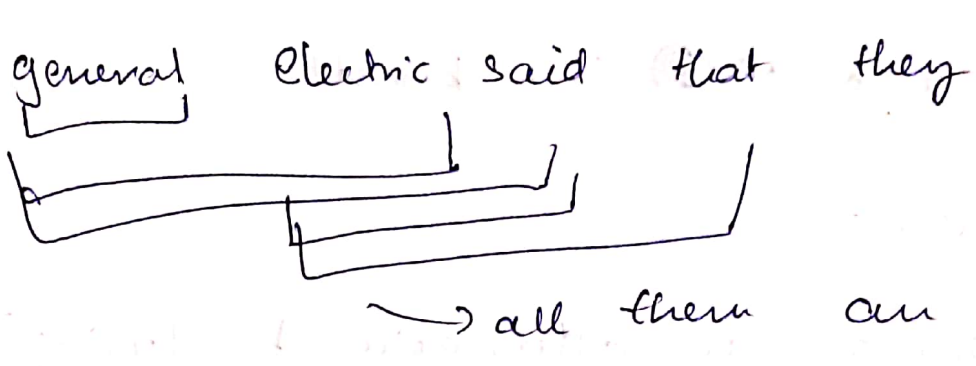
End-to-End (to SOTA?) (2017)

- LSTM + attention
- words as i/p -> embedding + CharCNN

↓ i/p representation

Deep BiLSTM

↳ Rep for span (continous subsequence of words)



→ all them are span

Deep BiLSTM

↓

span representation in 3 parts
+ Attn weights on diff words

Score every pair of spans to see if they are coherent mentions

$$s(i,j) = s_m(i) + s_m(j) + s_a(i,j)$$

- $s_m(i)$ ↓ is i a mention
- $s_m(j)$ ↓ is j a mention
- $s_a(i,j)$ ↓ Do they look Coreferent

Score fn $s()$ takes span ref as i/p

Clustering band

- agglomerative clustering
- start with each mention as singleton clusters
- Merge a pair of clusters at each step

Multi task learning.

- decaNLP (Natural langug Decathlon)
  (- langug modeling
  - Qn Ans
  - Summarisation)

- use RL for NAS & multitask ?

← TREE bank models for NLP →
- Treebank

← Bias in AI →

Using unlabled data

- Pre train encodr & decodr for diff languges (for MT)
- Use unlabled data (in this case, En → anothr lang) for pre-training
- Self training
  - Pre-training won't have interaction b/w encodr & decodr
  - load data from our model & send it to En-De model. (It can b circular, so not- and much)

Back- Translation

- Do En - Fr, then do Fr-en with target as the original english. (like an autoencoder)
- Then train it on ~~trained~~ labled data

- train 2 model on labled data, then do back-translation on large corpus, do it over and over again

Unsupervised word translation

- Cross lingual word embedding -
  - Shared embedding for both languages
  - Assumption is that language structure is same b/w the languages
  - Run word2vec on monolingual Corpora, getting word embeddings X & Y
  - Learn an orthogonal matrix W such that $WX \approx Y$
  - Our method to train W is called adversarial training. A Discriminator will predict if a word is english or french. Train W so that Discriminator will be confused. (It can't predict accurately)

- same en. dec used for both languages
- special token is used to diff languages

GPT-2

- Just a really big TF LM
- 40GB text i/p
- Data webpages from reddit links with high karma
- zero shot learning
  - generate from a prompt
  - Reading Compre..
    i/p <context> <qn> A:..
  - Summarisation
    <article> TL;DR:
  - Translation
    <sent1> = <sent2>
    :
    <sent> =
  - Qn Ans
    <qn> A:
- It can do translation because other lang are in training corpora in a very small amount

graph neural n/w

graph → vertice / node and edges

$G = (V, E)$ edges → unidirectional ⇒ bidirectional

node classification in GNN. (original paper)

every node → associated with a label, predict label w/o ground truth.

each node v is characterised by its feature x_v associated with a ground truth t_v.

given a partially labelled graph, then labelled nodes are used to predict the unlabelled.

learns to represent each node with a d-dim vector h_v

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]})$$

$x_{co[v]}$ → features of edges connecting v

$h_{ne[v]}$ → embeddings of neighbouring nodes of v

$x_{ne[v]}$ → feature of neigh nodes of v

f → puts it into a d-dim space.

This can be solved by message passing / neighbourhood aggregation

Banach fixed point theorem can be applied

$$H^{t+1} = F(H^t, x)$$

H, x -> concat of h, x.

gnn o/p :

$$o_v = g(h_v, x_v)$$

$$\text{loss} = \sum (t_i - o_i)$$

Limitations of this:

- if assumption of fixed point is relaxed, NN can learn more stable representations, and can remove iterative update process.
- can't process edge info
- Fixed point can discourage the diversification of node distribution, and thus may not be suitable for learning to represent nodes.

gnn survey – Zonghan et al.

Conv GNNs → spectral
→ spatial

Network embidding – represent n/w nodes as low-D vectors preserving n/w topology & node info → use this to perform classification, clustering, recomendation using simple algos

$G = (V, E)$ V → nodes, E → edges

$e_{ij} = (v_i, v_j)$ edge v_j to v_i

$N(v) = \{ u \in V \mid (v, u) \in E \}$ neighbourhood of node v

A graph may have node attributes X

$X \in R^{n \times d}$ → node feature matrix $x_v \in R^d$

A graph may have edge attributes x^e

$X^e \in R^{m \times c}$ → edge feature matrix

$x^e_{v,u} \in R^c$ ↓ feat vector of edge u-v

A graph is undirected if & only if the adjacency matrix is symmetric.

Spatio-Temporal graph:

attributed graph when node attribute changes dynamically over time $G^{(t)} = (V, E, X^{(t)})$ $X^{(t)} \in R^{n \times d}$

ConvGNN

generalise convolution from grid data to graph data generate a node v's repre by aggregating its feature x_v, neighbours feature x_u. ConvGNNs stack multiple grap conv layers to extract high-level node repre.

graph autoencoders

encodes nodes/graphs into a latent vector space, reconstruct data from that. It is used for n/w embeddings, graph generative distributions.

with graph structure & node content info as i/ps, o/ps of GNN can focus on diff tasks

- node level
  node regression, classification
- edge level
  edge classification, link prediction

with two nodes hidden vector can form GNN as i/p, Similarly fn of nn can be utilized to predict label / connection strength of an edge.

- graph level

graph classification tasks. To get a compact representation, GNNs can be combined with pooling, read-out ops.

CS231n

Data preprocessing:

- Mean subtraction
  - data is around origin in each dim
  (done separately across each channel)
- Normalisation
  - scaling all data to a single scale
    - mean + std
    - min max
      ↳ use if diff i/p are at diff scales but needs to equal importance.
- PCA
  - if many features are dependent
  - for eg: with NB.
  - and for small dataset
- whitening
  - take data in eigenbasis & divide every dim by EV.

-linear classifier learns only 1 template for each class

SVM loss / Hinge loss

	class 1	class 2
class 1	3.2	1.3
class 2	5.1	4.9
class 3	-1.7	2

1 $L = \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$

$= 2.9$

2 $L = \max(0, 1.3 - 4.9 + 1) + \max(0, 2 - 4.9 + 1)$

$= 0$

properly classified

Softmax

- prob dist of scores of the classes.
- map this to target distribution
- can be viewed as KL div & MLE.
- mathematically it is easier to maximize log than actual prob
- negative cause log goes high n high and we need to minimize

	class 1			
class 1	3.2		24.5	0.13
class 2	5.1	$\xrightarrow{e}$	164	$\xrightarrow{\text{normalize}}$ 0.87
class 3	-1.7		0.18	0

$-\log(\text{true class}) = -\log(0.13) = 0.87$

BN

have unit gaussians at every forward pass. per batch.

- get mean & var for each dim.
- usually inserted after Conv & before non-linearity.
- can recover identity matching
- Improved grad flow
- Allows high LR
- Reduces dependence on initialisation

Momentum

$$x_{t+1} = x_t - \alpha \nabla f(x_t) \quad \text{SGD.}$$

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1} \quad \rho \text{ (rho) (MP)} \rightarrow \text{friction}$$

Nesterov M

- look at grad at where vel is computed & combine it with present grad

$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t + \rho v_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

Adagrad

keep running sum of squared grads
divide by this while updating
- accelerate movement among slow dim
& slow down on fast dim

RMSprop

decay the squared estimate

Adam

Combine Momentum & Adagrad / RMSprop

Alexnet (8 layer)

- shared across 2 gpu
- Relu, SGD+M (BS 128)
- Layer norm

ZFNET

- Improving HP

VGGNet

- 16 - 19 Layers
- 3×3 Conv. 1,1 -> ~~bar~~
  2×2 MP , 2,0
- 3 3×3 Conv has 7×7 Receptive field
- 138M param (lot in fc)
- no local norm.

GoogLeNet

- 22 layer
- Efficient inception module
- no fc (only global avg pool)
- Concat((1×1) + (1×1, 3×3) + (1×1, 5×5), (3×3 MP, 1×1))

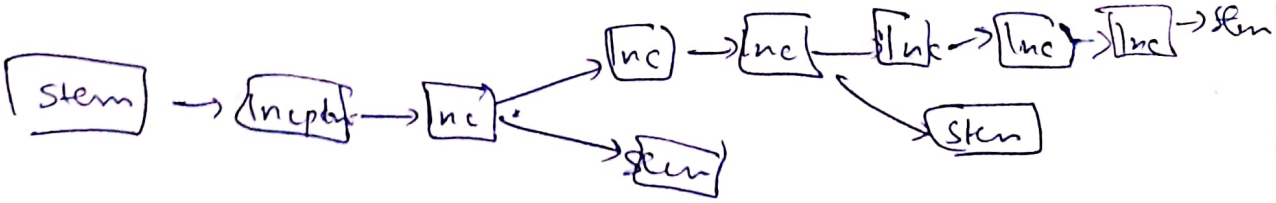
↓

good local n/w topology

Naive Inception.

Concat(1×1 + 3×3 + 5×5 + 3×3 MP)

1×1 bottleneck reduces param by $\frac{854}{388}$

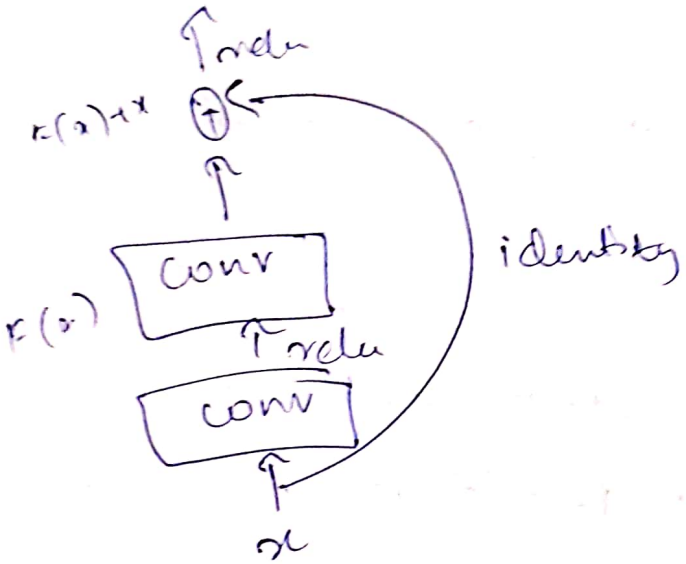


Stems -> auxiliary classi. o/p.

skips because more gradient injecting.

ResNet

- 152 layers (in Imagnet)
- Residual connections



- deeper models are harder to optimize.

generative Models

$$p(x) = \prod p(x_i | x_1, x_2 \ldots x_{i-1})$$

↓

Likelihood of image x.

$x \rightarrow$ pixel.

PixelRNN

- Generate Pixel starting from a corner.
- Sequential gen is slow

PixelCNN

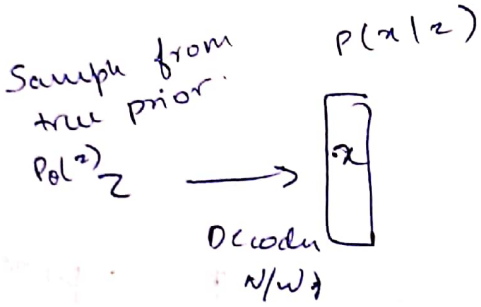
- CNN over a context region around a particular pixel.

VAE

density fn with a latent variable z

$$P_\theta(x) = \int P_\theta(z)\, P_\theta(x|z)\, dz$$

x is generated from z.



Choose gaussian prior.

represent $P(x|z)$ using a NN

But integral is intractable.

In addition to decoder, define encoder $q(z|x)$ that approx $P(z|x)$

loss - Reconstruction loss + KL Divergence

backprop can't go down on the latent space (which is a distribution)

$z = \mu + \sigma \odot \epsilon$

$\epsilon \sim$ Normal $(0, 1)$.

no need to run backprop on ϵ.