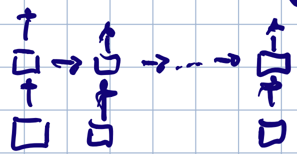
EEML – Day 5 – Bucharest

NLP Talk 1 – Razvan + Stephen

- RNN ⇒ Turing Complete → Expressivity: Can approx. every fct. given finite operands



↳ history h_{t-1} → ⊗ → h_t → ▷ → y_t out; x_t in, input/embedding; ↳ read out

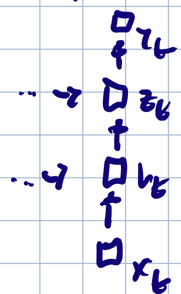
difficult to optimize → ↳ Can also replace fct. by a deep net ⇒ DT-RNN

↳ Skip connections not enough difficulty in optimization!

→ vs. short at input → TTTL → dense → safe

- How to make deep?

→ Stacking: (for addl. expressivity)



- Backprop: Only have to store vector in memory ⊕ easily compute $\partial L / \partial a_L \in \mathbb{R}^{d_L}$ while $\frac{\partial \sigma(.)}{\partial (.)} \in \mathbb{R}^{d \times d_L}$

↳ Also: Elementwise activation gives diagonal Jacobian

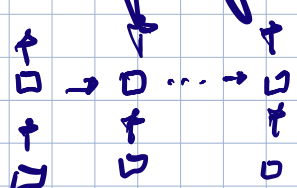
↳ Through time: $\frac{\partial h(t)}{\partial h(t-k)} = \prod_{j=k+1}^{t} \frac{\partial h(j)}{\partial h(j-1)}$ → Jacobian product to leading vs. vanishing expl. grad.

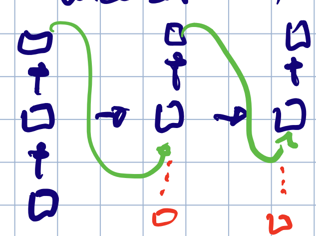
Problem: Gradient flow is not enough to ensure memory

$\frac{\partial (x_1 + x_2 + x_3)}{\partial x_3} = 1$ $\boxed{\text{But}}$ $x_1 + x_2 + x_3 = 10$ → Can't recover x_3!

↳ Not all info can be recovered from limited expressability of hidden state

→ Teacher forcing vs. Unconstrained





↳ Looks like Imitation L.

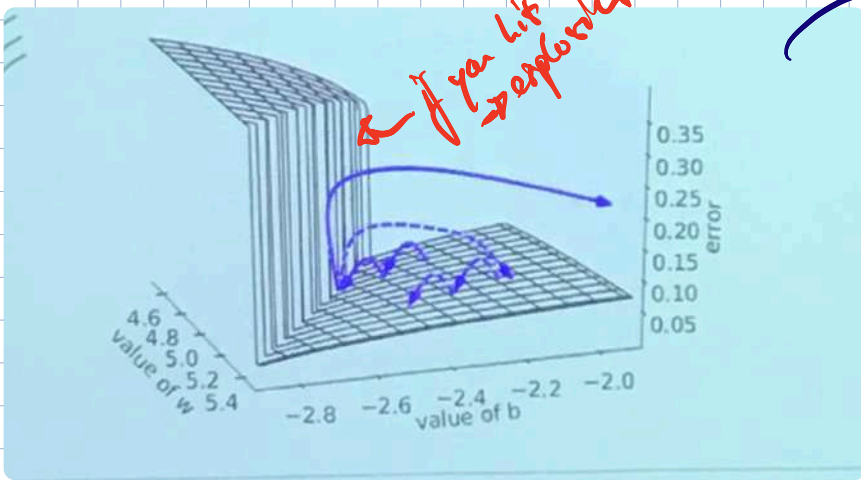
↳ Can also combine both and have model decide what input to feed!

↳ GENERAL: RL ⇔ RNN connection → many parallels → pick up & transfer tricks!

• Exploding gradients: Grad clipping ⇒ ad-hoc solution

→ loss surface usually not valley to-scale with grad descent/momentum

↳ More like:



→ Not solved by 2nd order!

↳ All derivatives explode!

⇒ Clipping: Different versions

↳ Clip full gradient

↳ Clip at each layer

⇒ Does not really matter but you need something!

• Vanishing gradients: Not slow learning problem,

→ Components of gradient vanish ⇒ Makes problem hard

↳ Can't be detected by simply looking at the norm!

↳ $$g_T = \frac{\partial C(T)}{\partial h(T)} \rightarrow g_{T-1} = \frac{\partial C(T)}{\partial h(T)} \frac{\partial h(T)}{\partial h(T-1)} + \frac{\partial C(T-1)}{\partial h(T-1)}$$

SGD: Problem → Never have vanishing parts stored independently! → Memory footprint

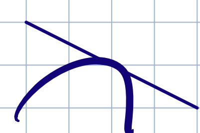
↳ Also: practically never explicitly compute $\partial h(T) / \partial h(T-1)$

→ exploit elementwise activations, etc.

• Weight matrix restriction to be orthonormal!

↳ Stay on sphere!

↳ all eigenvalues = 1



⇒ what happens to expressivity

→ linearize everything in all directions

⇒ what do we actually learn?

↳ Saxe et al 2014, Henaff et al 2016, Arjovsky et al 2016

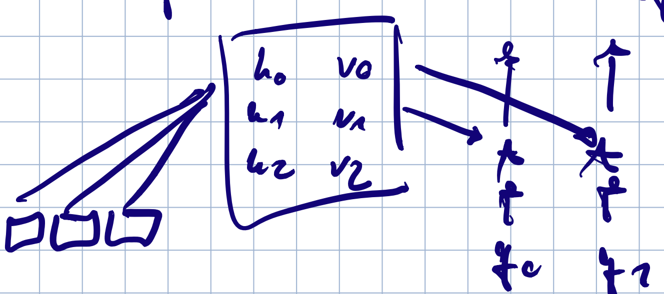
• LSTMs Criticism: Intuition of filtering the hidden state
→ Non-discrete gates // How to set gates up?
→ GRUs simplify → $h_t = (1-z) \circ h_{t-1} + z \circ \tilde{h}$
→ Echo state networks → Learn mixing

• Hierarchical Approaches: Higher layers work on slower timescale
→ Clockwork RNN → Koutnik et al 2014
→ Feeding targets as inputs vs auxiliary loss
↳ loss seems to work better!

• Language: – Sutskever et al (2014) → Google
– Bahdanau et al (2015) → Bengio
↳ Attention! MLP → Weights → Aggregate

• WaveNet: Nal Kalchbrenner – try to use CNNs in place of RNNs ↳ less layers very cons
→ Easily parallelizable but no layer truly looking at temporal order only cut!
↳ Generalize locality assumption by learning via attention

• Transformer Architecture: Attention = flexible Convolution
→ Dot product attention: $A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$ key value



$= \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$ temperature → saturate softmax → helps w. optimization

↳ Problem: Softmax → Amplifies small differences
Unimodal ↳ Sometimes want to attend to multiple things
↳ Solution: Transformers have separate heads! → at the same time helps to do credit assignment!
Reason for why sigmoid does not exp!

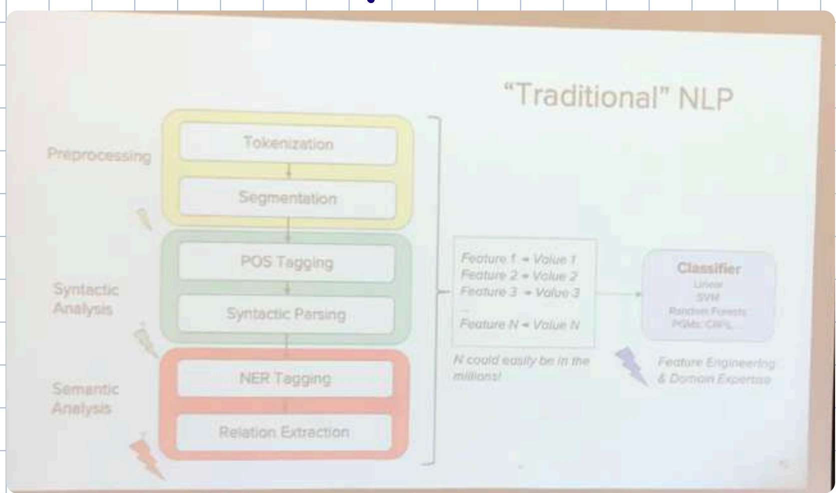
NLP Talk 2 – Antoine Bordes – (Machine Reading & Q Answering)

- Machine Reading: Extract representations to answer questions! (from text)
  - → Before 2014: Symbolic approaches ⇒ Translate into SQL query + database
  - → After 2014: E-to-E DL (First RNNs → Now Transfs)

TEXT → Q → A

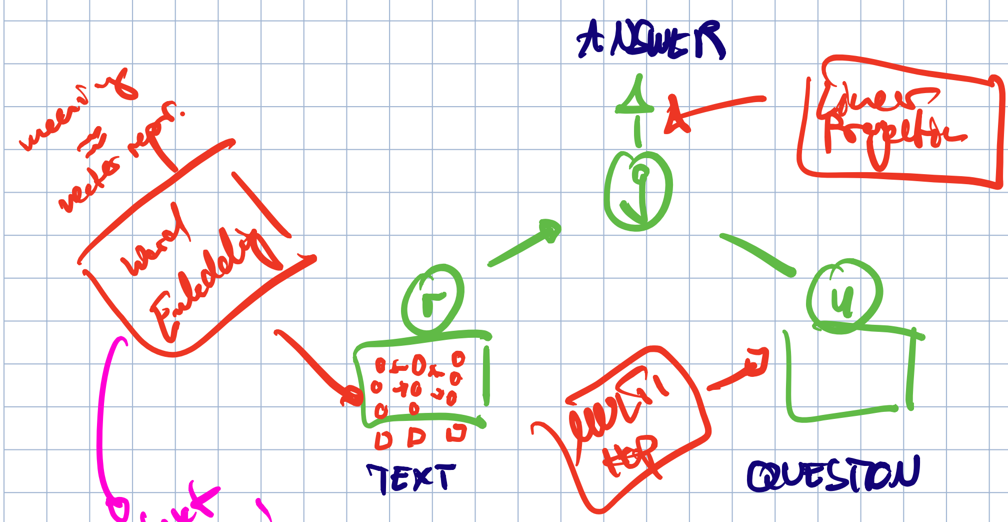
↳ REPRESENTATIONS!

- Challenge: A lot of common-sense encoded for us before starting to read!



- Independent steps in traditional NLP!
  ↳ Error propagation at each step
- MNIST for QA
  ↳ SQuAD → Stanford
  ↳ Highlights through Gathering A

- Attentive Reader Model – Hermann et al 2015



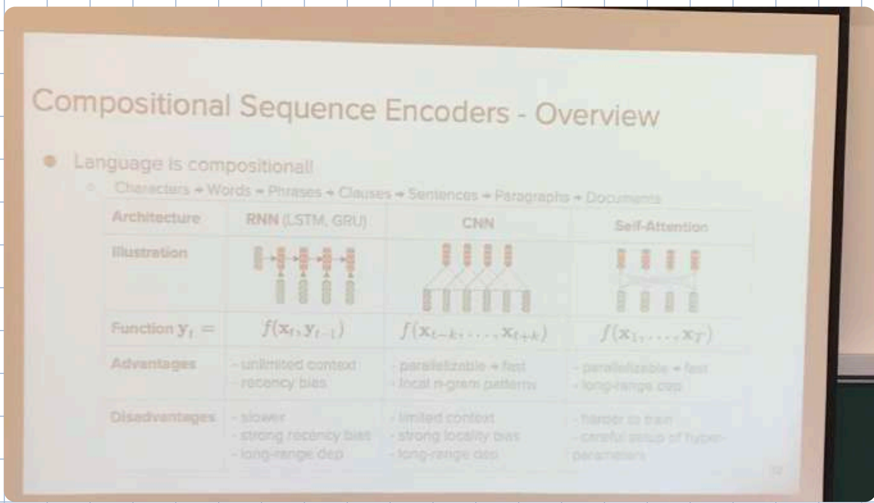
Not Context Sensitive Repr.!
⇒ Want Contextuality

→ Language ⇒ Compositional!
↳ Inductive Biases ⇒ Which composition function?

→ Tradeoff ⇒ Expressivity vs. Trainability!

→ Bi-Dir. LSTM: Concatenate representations from l-to-r and r-to-l

- Self-Attention ⇒ Problems: hyperparameter search, Memory costs
  ↳ Transformer: Vaswani et al 2017 → Form a graph with weighted edges



within sequence!

↳ Multi-Head
⇒ Form of different Kernels in CNN without local constraint!

↳ Can be trained in an unsupervised fashion!
⇒ 'PRETRAINED'

• Pretrained Contextualized Embeddings ⇒ ELMO, BERT
→ BERT: Randomly mask 15% of tokens in input ⇒ Predict!
↳ Learn large transformer to fill blanks → DROPOUT
↳ Robustness!
→ Needs lots of high quality data + lots compute!
↳ Can also add data in/output
Also language dep. notebooks
↳ Can only be trained on few computers ⇒ No mobile real-time inference

• Memory Networks → Multi-hop attention ⇒ Sequence reasoning
↳ Use question explicitly to attend to paragraph!
↳ Go back further (multi-read!)

• Answers ⇒ Probability distr. over answer options ⇒ Linear Projection!
↳ Cross-Entropy / Rank loss!

• Early future: Paragraph Answer Generation / Latent Representation / Knowledge graph
↳ Das et al., 2019 ⇒ Multi-Step Retriever Reader

EEML - Day 6 - Bucharest

"How to generate stuff and learn representations?" - Nal Kalchbrenner

- Generation problem => center/focus on info exchange!
- Too many bits in X to model X directly → split in pieces & sample/predict pieces instead

$$\Rightarrow p(X) = \prod_i p(x_i \mid x_{<i}) \longrightarrow \text{EXACT FACTORIZATION}$$

smaller

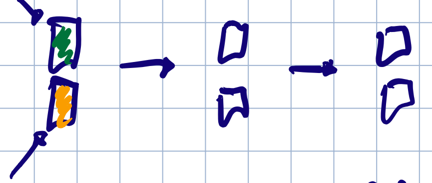
→ Reduce modelling complexity!
=> Conditioning

1D => Audio + Language

- RNN state captures $x_{<i}$ => Wave RNN

256 Repr. 256 Repr.
10100011 | 01100101
→ independence reduces problem

8 fine bits

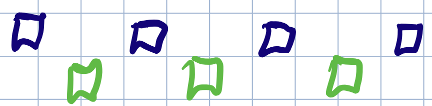


8 coarse bits

=> Still very slow!
→ Machines optimized for parallel generation → not sequential!
→ weights not units!
→ Sparsification => During training → threshold & die out but allow to get back if gradient in backprop crosses the threshold
→ Sweet spot of sparsity threshold!

SPEED UP!
Optimize for Mobile

- Subscale Wave RNN: Local dependence >> Global dependence
  → Reshape input tensor with gaps

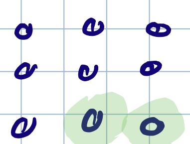


1st Sample w. steps in between
2nd Sample conditional

- Byte Net: Beginning of end of RNN
  → want to train faster => 1d Conv architecture → Masked Convolutions
- Importance of input features: Granularity vs. Complexity of what to model
  → Exploit hierarchy: Characters - Word-Pieces - Words

2D => Vision

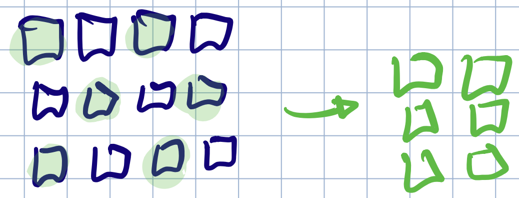
- Pixel RNN / Pixel CNN
  → Masked 2D Conv



→ Mask out → Softmax on top of masked filtering

↳ Model on pixel level!

→ Subscale Pixel Networks: Slicing of image into pieces ⇒ conditioned on previously sliced



↳ Multidim. Upscaling: Size + depth
⇒ Bits per channel

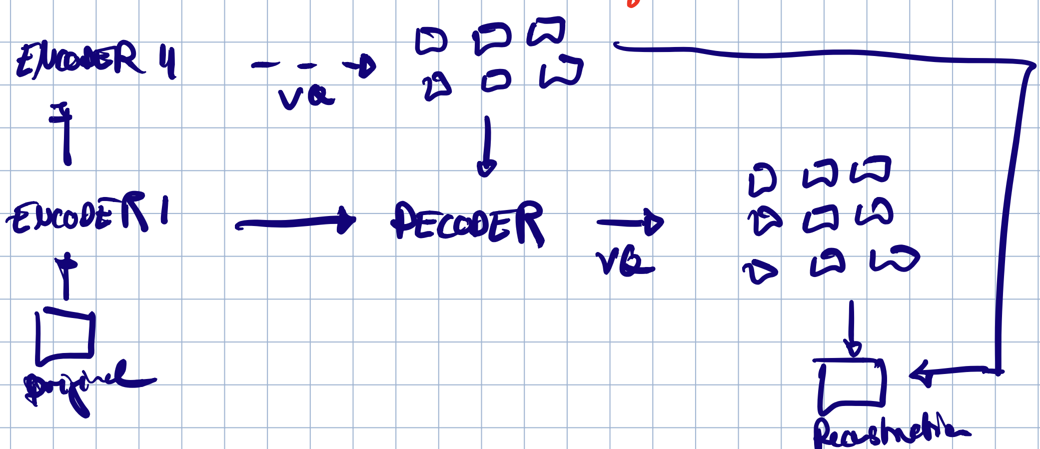
↳ Works a lot better for images than audio

↳ Working with spectogram fourier representation using own approach
⇒ phase looks/seems fairly random → try modeling conditionally!

↳ Put slices into large encoder - decoder structure

• VQ-VAE 2: $P(f(X))$

compressed latents ⇒ discrete - hard to propagate ⇒ take nearest neighbor



lossy transformation fct.

↳ No downstream task

→ Latent representations: do not seem to disentangle ⇒ Instead: low/downsampled full energy representations

3D ⇒ Videos

• Video Pixel Networks + Video Transformer → How to generalize!

↳ Masked convs

↳ Feed frames over time

↳ For self-Attention

→ Predict missing pieces

$P(x_i \mid x_{<i}, y_{>i})$

Learning Representations

• Representation Problem: Good fct. $g(X)$ representing X in useful way for classification/understanding

↳ Contrastive Predictive Coding

↳ Contextual learning! // Hard to discover tricks ⇒ Powerful!