CS772: Deep Learning for Natural Language Processing (DL-NLP)

LM, CNN, Stable Diffusion

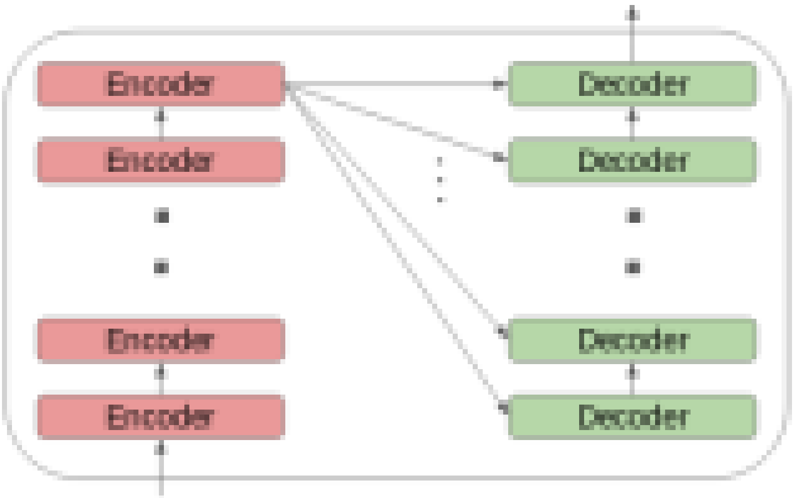
Pushpak Bhattacharyya

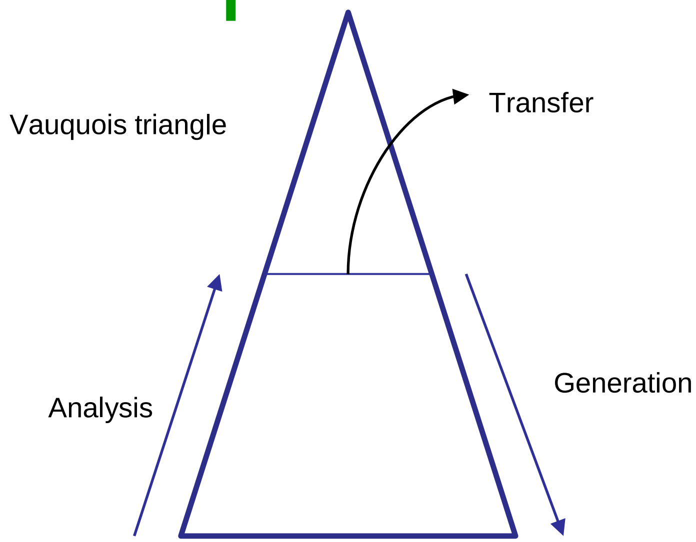
Computer Science and Engineering Department

IIT Bombay

Week 14 of 8apr24

1-slide recap





Efficient and elastic Large Models (Dr. Prateek Jain, Google Research**)**

Key challenges in improving efficiency of LLM serving.

Matformers: train one model but read-off 100s of smaller models + speed up decoding in LLMs
Depends on Matryoshka Representation Learnining, Neurips 24

What is Language Model

- **Model of Language**: what is that?
- Recall properties of human language
  - **Displacement** (Indicators that change with time and place: I saw him yesterday at the market; I will see him tomorrow in the school)
  - **Arbitrariness** (name → Meaning; water, chair)
  - **Productivity/creativity** (potentially infinite no. of sentences)
  - **Cultural Transmission** (child acquires parent's language)
  - **Discreteness** (sound and meaning units separated)
  - **Duality** (Surface structure, deep structure)

What does a Model do?

- **The only thing Models** do is **PREDICT!**
- E.g.,

$$s = ut + \frac{1}{2}at^2$$

- *S:distance; u: initial velocity; a:acceleration; t: time*
- Now, given <*u, a, t*> we can compute (predict!) *s*

What does a Language Model do?

- **Predicts** properties of language objects, called classification ~~of~~ regression
  - E.g., sentiment analysis: *The movie was wonderful* → *+ve sentiment*
- **Predicts** "next" language objects
  - E.g.-1, Answer to Question: *what is the capital of India* → *Delhi*
  - E.g.-2, Summary of input text: *Delhi is a large city. The city has large number of people residing and passing through. Large businesses are transacted here. The cultural activities are numerous.* → *Delhi is a large, populous and busy city*

The foundation

- All predictions on language objects are FOUNDED on ONE prediction task
- **Predict** the next word given a sequence of words

$$P(W_N \mid W_{N-1}W_{N-2}\ldots W_1W_0)$$

$$= \frac{\#(W_0W_1\ldots W_{N-1}W_N)}{\#(W_0W_1\ldots W_{N-1})}$$

What is Curse of Dimensionality

- As the number of dimensions (features) increases, the amount of data required to adequately sample the space grows exponentially
- E.g., No. of Boolean Functions: 2^{2^N}
- No. of Threshold functions: 2^{N^2}

Curse of Dimensionality and LM

- N-gram LM
- $P(W_N|W_0 W_1 W_2 W_3 \ldots W_{N-1})$
- What happens as *N* increases?

$$P(W_N \mid W_{N-1} W_{N-2} \ldots W_1 W_0) = \frac{\#(W_0 W_1 \ldots W_{N-1} W_N)}{\#(W_0 W_1 \ldots W_{N-1})}$$

Curse of Dimensionality and LM, cntd.

- N-gram LM, equivalent to computing $P(W_0 W_1 W_2 W_3 \ldots W_{N-1} W_N)$
- How many such parameters need to be computed?
- Let $|V|$ be the vocab size
- Each position of the n-word string can be filled in $|V|$ ways
- Hence the number of parameters is $|V|^N$
- *Curse of dimensionality*

Neural LM

- Solves curse of dimensionality; how?
- The number of parameters are not $P(W_N|W_0W_1W_2W_3\ldots W_{N-1})$s
- The parameters are weights and biases in the neural net

Neural LM: #parameters

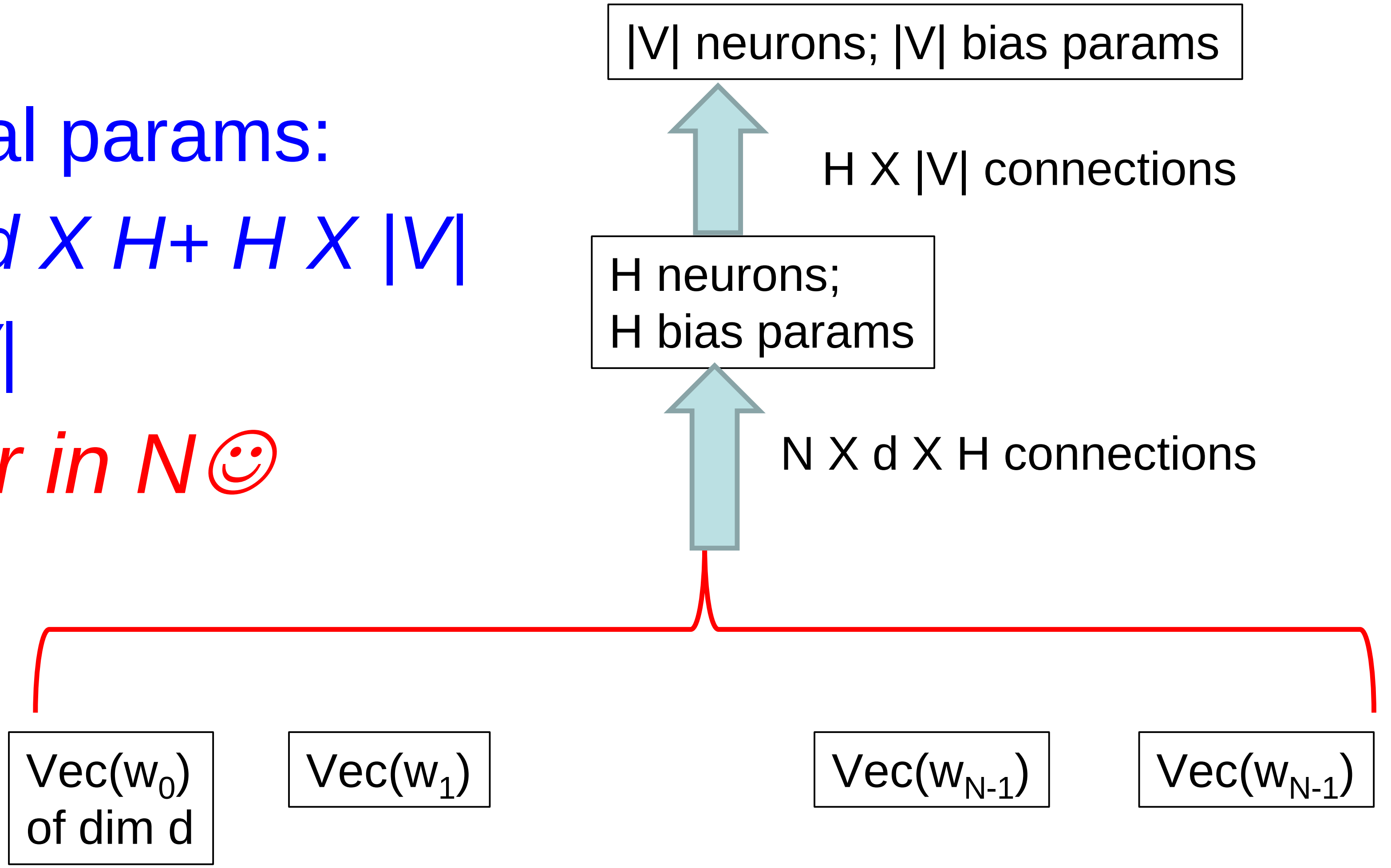
- Solves curse of dimensionality; how?

- Total params:

$= N\ X\ d\ X\ H + H\ X\ |V|$

$+ H + |V|$

Linear in N☺



Grammar as LM and Computation

Computation is called **PARSING**

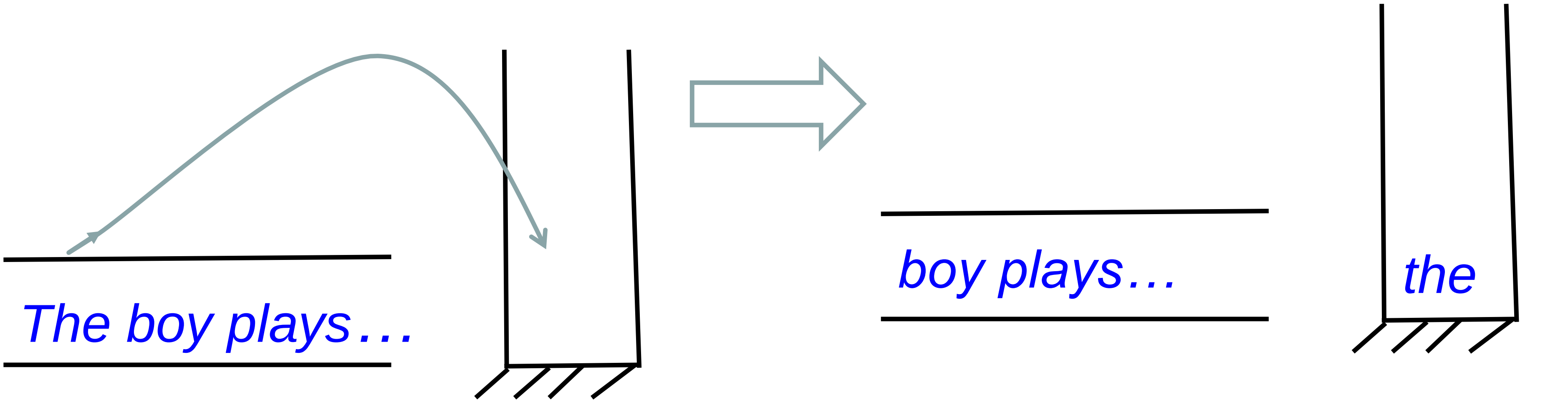
A segment of English

- $S \rightarrow NP\ VP$
- $NP \rightarrow DT\ NN$
- $NP \rightarrow NNS$
- $NP \rightarrow NP\ PP$
- $PP \rightarrow P\ NP$
- $VP \rightarrow VP\ PP$
- $VP \rightarrow VBD\ NP$

- DT $\rightarrow$ the
- NN $\rightarrow$ gunman
- NN $\rightarrow$ building
- VBD $\rightarrow$ sprayed
- NNS $\rightarrow$ bullets

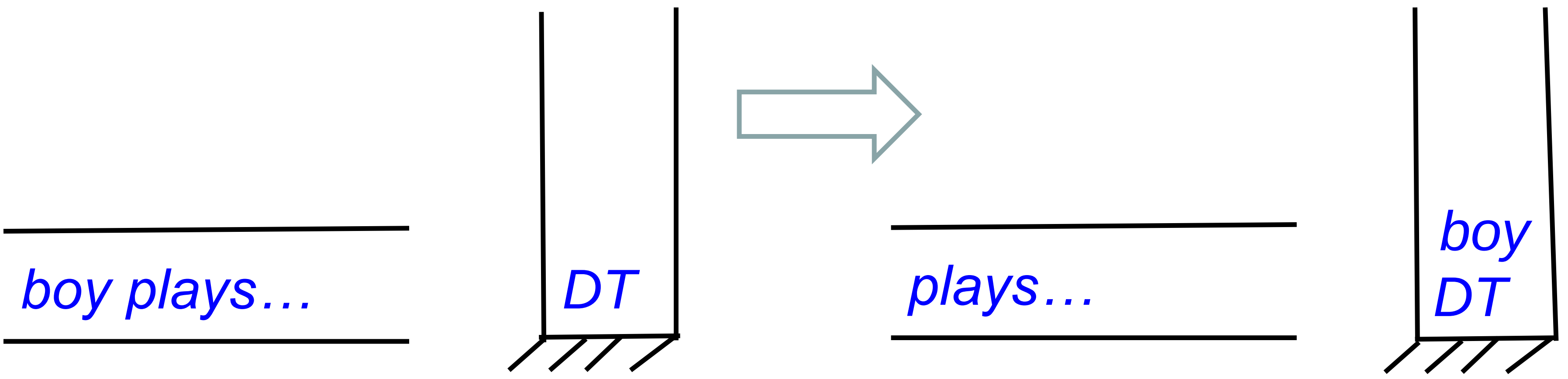
GENERATIVE GRAMMAR, due to Noam Chomsky

Shift Reduce (1/3)



(a) Shift

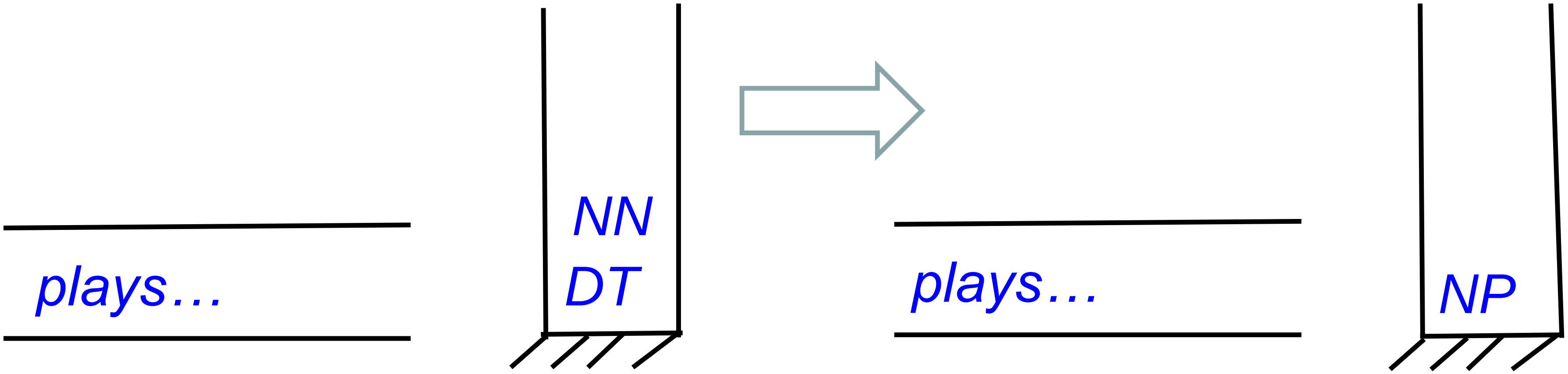
Shift Reduce (2/3)



(b) Reduce

(c) Shift

Shift Reduce (3/3)



(d) Reduce

(e) Reduce

A segment of English

- S → NP VP
- NP → DT NN
- NP → NNS
- NP → NP PP
- PP → P NP
- VP → VP PP
- VP → VBD NP

- DT → the
- NN → gunman
- NN → building
- VBD → sprayed
- NNS → bullets

GENERATIVE GRAMMAR, due to Noam Chomsky

Foundational Question

- Grammar rules are context free grammar (CFG) rules
- Is CFG enough powerful to capture language?

- CFG cannot accept/generate $a^n b^n c^n$
- Corresponding language phenomenon: Jack, Mykel and Mohan play tennis, soccer and cricket respectively.

Indexed sentence

$_0$The $_1$ gunman $_2$ sprayed $_3$ the $_4$

$_4$building $_5$ with $_6$ bullets $_7$. $_8$

CYK Parsing: Start with (0,1)

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7.**

To From	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	DT						
1	--------						
2	--------	----------					
3	--------	----------	---------				
4	---------	----------	---------	----------			
5	---------	----------	---------	----------	----------		
6	---------	----------	---------	----------	----------	----------	

CYK: Keep filling diagonals

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7**.

To From	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	DT						
1	-------	NN					
2	-------	---------					
3	-------	---------	--------				
4	--------	---------	--------	---------			
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK: Try getting higher level structures

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7**.

To **From**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	DT	NP					
1	-------	NN					
2	-------	---------					
3	-------	---------	--------				
4	--------	---------	--------	---------			
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK: Diagonal continues

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To **From**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	DT	NP					
1	-------	NN					
2	-------	---------	VBD				
3	-------	---------	--------				
4	--------	---------	--------	---------			
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7.**

To **From**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	DT	NP	---------				
1	-------	NN	---------				
2	-------	---------	VBD				
3	-------	---------	--------				
4	--------	---------	--------	---------			
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7**.

To From	1	2	3	4	5	6	7
0	DT	NP	--------				
1	-------	NN	--------				
2	-------	---------	VBD				
3	-------	---------	--------	DT			
4	--------	---------	--------	---------			
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7.**

To **From**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	DT	NP	--------	---------			
1	-------	NN	--------	---------			
2	-------	---------	VBD	---------			
3	-------	---------	--------	DT			
4	--------	---------	--------	---------	NN		
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK: starts filling the 5th column

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------			
1	-------	NN	--------	---------			
2	-------	---------	VBD	---------			
3	-------	---------	--------	DT	NP		
4	--------	---------	--------	---------	NN		
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7*.*

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------			
1	-------	NN	--------	---------			
2	-------	---------	VBD	---------	VP		
3	-------	---------	--------	DT	NP		
4	--------	---------	--------	---------	NN		
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7***.*

To From	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	**DT**	**NP**	**--------**	**---------**			
1	**-------**	**NN**	**--------**	**---------**	**---------**		
2	**-------**	**---------**	**VBD**	**---------**	**VP**		
3	**-------**	**---------**	**--------**	**DT**	**NP**		
4	**--------**	**---------**	**--------**	**---------**	**NN**		
5	**--------**	**---------**	**--------**	**---------**	**---------**		
6	**--------**	**---------**	**--------**	**---------**	**---------**	**---------**	

CYK: S found, but NO termination!

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7**.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S		
1	-------	NN	--------	---------	---------		
2	-------	---------	VBD	---------	VP		
3	-------	---------	--------	DT	NP		
4	--------	---------	--------	---------	NN		
5	--------	---------	--------	---------	---------		
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S		
1	-------	NN	--------	---------	---------		
2	-------	---------	VBD	---------	VP		
3	-------	---------	--------	DT	NP		
4	--------	---------	--------	---------	NN		
5	--------	---------	--------	---------	---------	P	
6	--------	---------	--------	---------	---------	---------	

CYK (cont…)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	
1	-------	NN	--------	---------	---------	---------	
2	-------	---------	VBD	---------	VP	---------	
3	-------	---------	--------	DT	NP	---------	
4	--------	---------	--------	---------	NN	---------	
5	--------	---------	--------	---------	---------	P	
6	--------	---------	--------	---------	---------	---------	

CYK: Control moves to last column

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	
1	-------	NN	--------	---------	---------	---------	
2	-------	---------	VBD	---------	VP	---------	
3	-------	---------	--------	DT	NP	---------	
4	--------	---------	--------	---------	NN	---------	
5	--------	---------	--------	---------	---------	P	
6	--------	---------	--------	---------	---------	---------	NP NNS

CYK (cont…)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7*.*

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	
1	-------	NN	--------	---------	---------	---------	
2	-------	---------	VBD	---------	VP	---------	
3	-------	---------	--------	DT	NP	---------	
4	--------	---------	--------	---------	NN	---------	
5	--------	---------	--------	---------	---------	P	PP
6	--------	---------	--------	---------	---------	---------	NP NNS

CYK (cont…)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	
1	-------	NN	--------	---------	---------	---------	
2	-------	---------	VBD	---------	VP	---------	
3	-------	---------	--------	DT	NP	---------	NP
4	--------	---------	--------	---------	NN	---------	---------
5	--------	---------	--------	---------	---------	P	PP
6	--------	---------	--------	---------	---------	---------	NP NNS

CYK (cont…)

0 *The* **1** *gunman* **2** *sprayed* **3** *the* **4** *building* **5** *with* **6** *bullets* **7**.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	
1	-------	NN	--------	---------	---------	---------	
2	-------	---------	VBD	---------	VP	---------	VP
3	-------	---------	--------	DT	NP	---------	NP
4	--------	---------	--------	---------	NN	---------	---------
5	--------	---------	--------	---------	---------	P	PP
6	--------	---------	--------	---------	---------	---------	NP NNS

CYK: filling the last column

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7.

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	
1	-------	NN	--------	---------	---------	---------	---------
2	-------	---------	VBD	---------	VP	---------	VP
3	-------	---------	--------	DT	NP	---------	NP
4	--------	---------	--------	---------	NN	---------	---------
5	--------	---------	--------	---------	---------	P	PP
6	--------	---------	--------	---------	---------	---------	NP NNS

CYK: terminates with S in (0,7)

0 *The* 1 *gunman* 2 *sprayed* 3 *the* 4 *building* 5 *with* 6 *bullets* 7*.*

To From	1	2	3	4	5	6	7
0	DT	NP	--------	---------	S	---------	S
1	-------	NN	--------	---------	---------	---------	---------
2	-------	---------	VBD	---------	VP	---------	VP
3	-------	---------	--------	DT	NP	---------	NP
4	--------	---------	--------	---------	NN	---------	---------
5	--------	---------	--------	---------	---------	P	PP
6	--------	---------	--------	---------	---------	---------	NP NNS

LM with probability; probabilistic parsing

Main source:

Christopher Manning and Heinrich Schutze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

Noisy Channel Modeling

Source sentence → Target parse

$$T^* = \underset{T}{\operatorname{argmax}}\, [P(T|S)]$$

$$= \underset{T}{\operatorname{argmax}}\, [P(T).P(S|T)]$$

$$= \underset{T}{\operatorname{argmax}}\, [P(T)]$$, ***since given the parse the sentence is completely determined and P(S|T)=1***

Probability of a sentence (2/2)

Probability of a sentence = $P(w_{0,l})$

(0 is the index before the first word and l the index after the last word. All other indices are between words)

$$=\Sigma_t (P(w_{0,l}, t)$$

$$=\Sigma_t (P(t). (P(w_{0,l} | t)$$

$$=\Sigma_t P(t).1$$

where *t* is a parse tree of the sentence

If t is a parse tree for the sentence $w_{0,l}$, this will be 1 !!

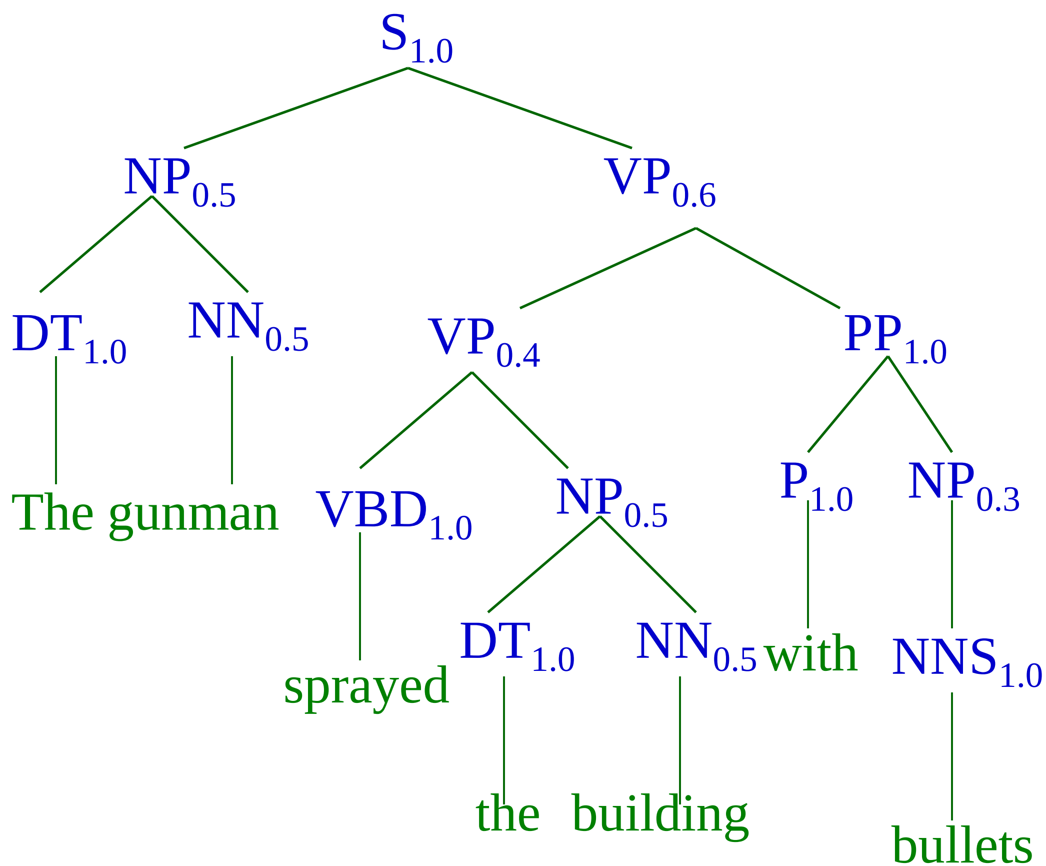
Probabilistic Context Free Grammars

- $S \rightarrow NP\ VP$ 1.0
- $NP \rightarrow DT\ NN$ 0.5
- $NP \rightarrow NNS$ 0.3
- $NP \rightarrow NP\ PP$ 0.2
- $PP \rightarrow P\ NP$ 1.0
- $VP \rightarrow VP\ PP$ 0.6
- $VP \rightarrow VBD\ NP$ 0.4

- $DT \rightarrow$ the 1.0
- $NN \rightarrow$ gunman 0.5
- $NN \rightarrow$ building 0.5
- $VBD \rightarrow$ sprayed 1.0
- $NNS \rightarrow$ bullets 1.0

Example Parse t_1`

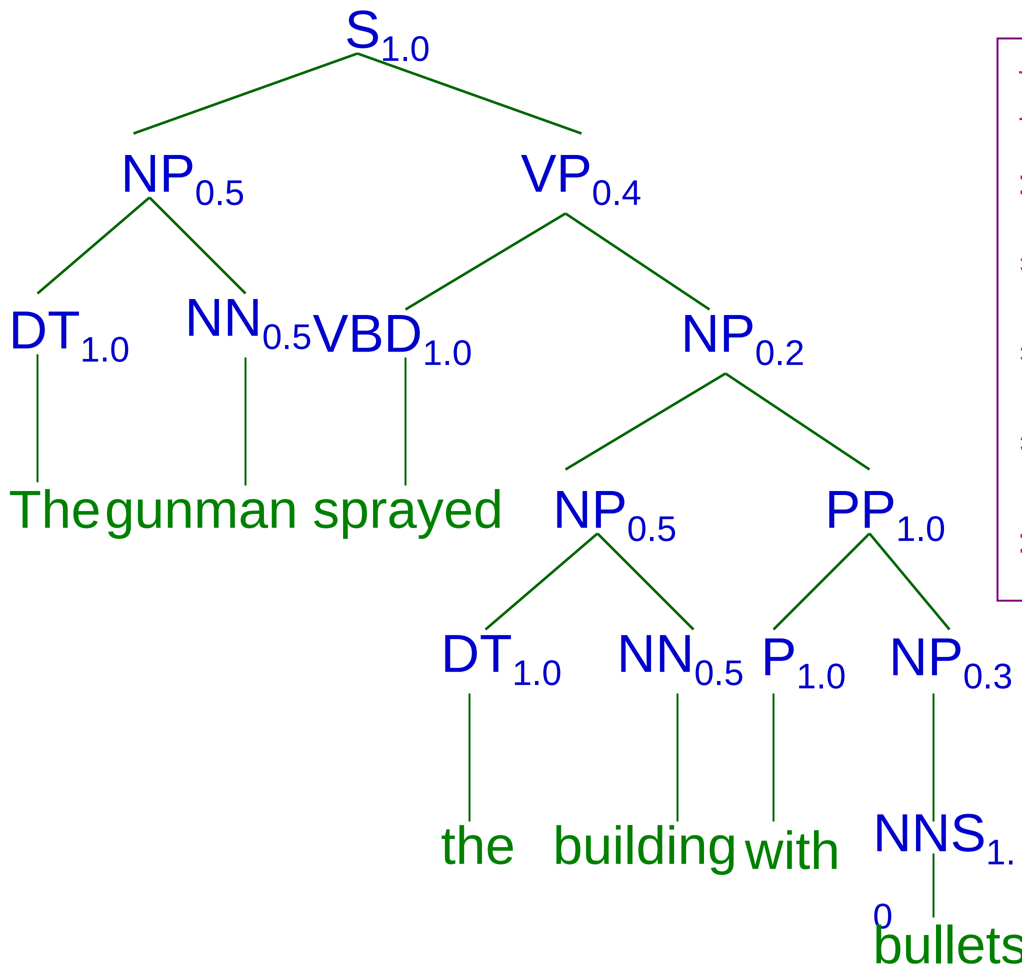
The gunman sprayed the building with bullets.



$P(t_1) = 1.0 * 0.5 * 1.0 * 0.5 * 0.6 * 0.4 * 1.0 * 0.5 * 1.0 * 0.5 * 1.0 * 1.0 * 0.3 * 1.0 = 0.00225$

Another Parse t_2

The gunman sprayed the building with bullets.



$P(t_2)$

$= 1.0 * 0.5 * 1.0 * 0.5$

$* 0.4 * 1.0 * 0.2 * 0.5$

$* 1.0 * 0.5 * 1.0 * 1.0$

$* 0.3 * 1.0$

$= 0.0015$

Summary of generations of LMs

Gen1: Context Free Grammar

- S → NP VP
- NP → DT NN
- NP → NNS
- NP → NP PP
- PP → P NP
- VP → VP PP
- VP → VBD NP

- DT → the
- NN → gunman
- NN → building
- VBD → sprayed
- NNS → bullets

GENERATIVE GRAMMAR, due to Noam Chomsky

Gen2(a): N-grams

- $P(W_0 W_1 W_2 W_3 \ldots W_{N-1} W_N)$
- Eqv to $P(W_N | W_0 W_1 W_2 W_3 \ldots W_{N-1})$

$$P(W_N \mid W_{N-1} W_{N-2} \ldots W_1 W_0)$$
$$= \frac{\#(W_0 W_1 \ldots W_{N-1} W_N)}{\#(W_0 W_1 \ldots W_{N-1})}$$

Gen2(b): Probability of a sentence

Probability of a sentence = $P(w_{0,l})$

(0 is the index before the first word and l the index after the last word. All other indices are between words)

$$=\Sigma_t(P(w_{0,l}, t)$$

$$=\Sigma_t(P(t).\ (P(w_{0,l}|\ t)$$

$$=\Sigma_t P(t).1$$

where *t* is a parse tree of the sentence

If t is a parse tree for the sentence $w_{0,l}$, this will be 1 !!

Gen3: Neural LM

- Solves curse of dimensionality; how?

- Total params:

$= N \times d \times H + H \times |V|$

$+ H + |V|$

Linear in N☺

|V| neurons; |V| bias params

H X |V| connections

H neurons;
H bias params

N X d X H connections

Vec(w_0)
of dim d

Vec(w_1)

Vec(w_{N-1})

Vec(w_{N-1})

CNN

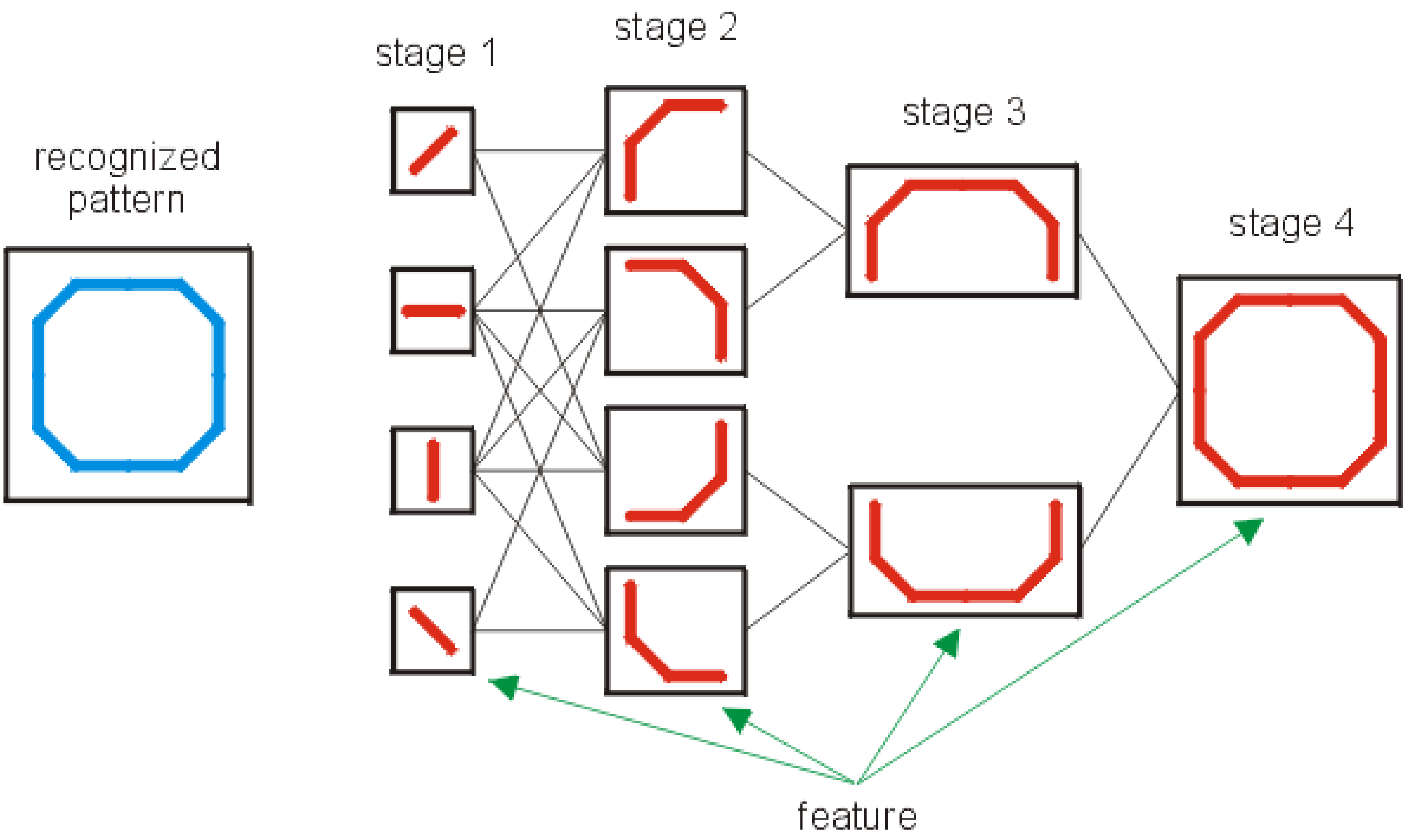
Two motivation points

- 1. Reduced number of parameters
- 2. Stepwise extraction of features
- These two are applicable to any AI situation, and not only vision and image processing

CNN= feedforward like + recurrent like!

- Whatever we learnt so far in FF-BP is useful to understand CNN
- So also is the case with RNN (and LSTM)
- Input divided into regions and fed forward
- Window slides over the input: input changes, but ‘filter’ parameters remain same
- That is like RNN

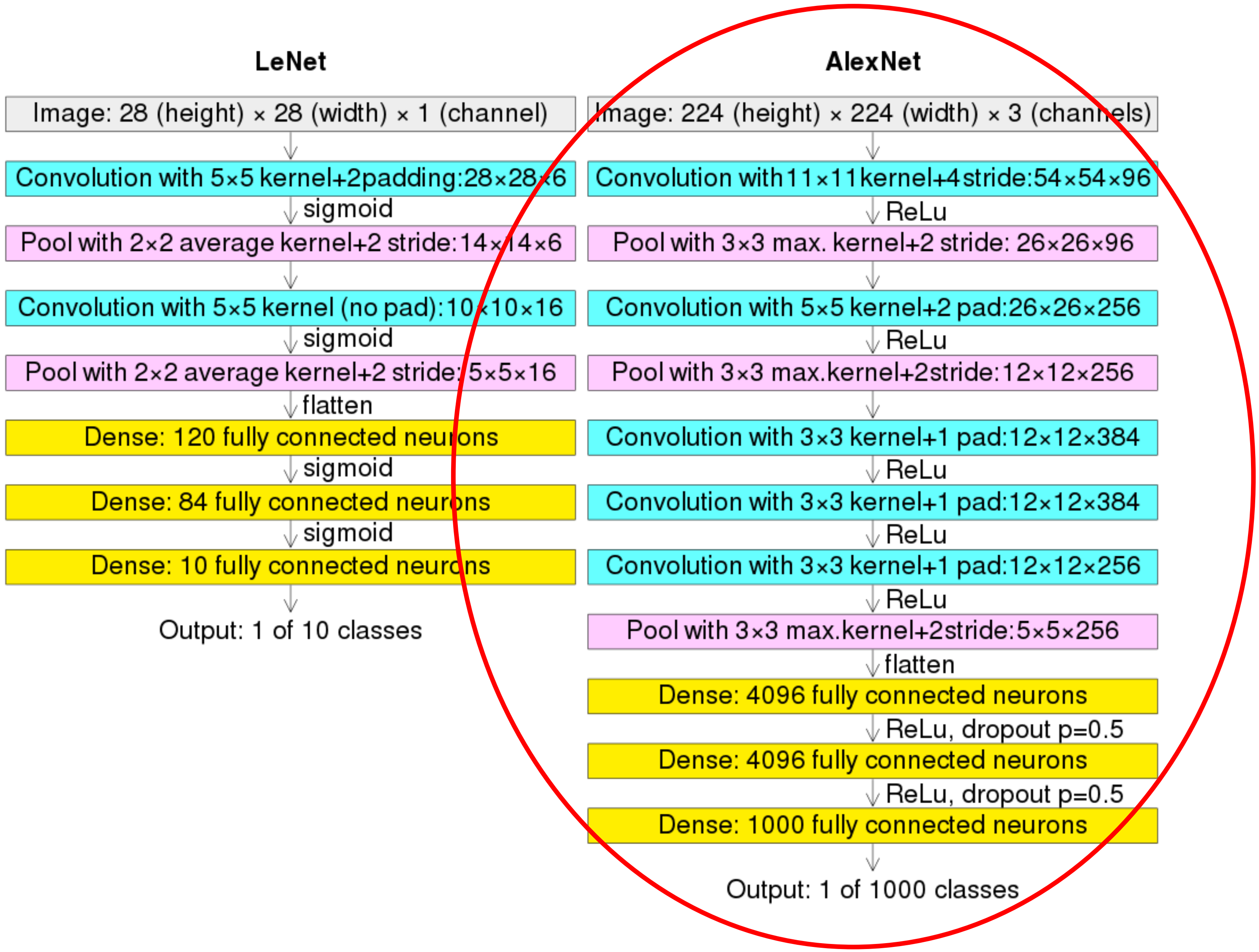
Genesis: Neocognitron (Fukusima, 1980)



Inspiration from biological processes

- Connectivity pattern between neurons resembles the organization of the animal visual cortex
- Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field
- Receptive fields of different neurons partially overlap such that they cover the entire visual field

The classic CNN (Wikipedia)



Convolution

1	**0**	**1**
0	1	0
1	0	1

Filter/kernel/

feature-detector

1 $_{\times 1}$	1 $_{\times 0}$	1 $_{\times 1}$	0	0
0 $_{\times 0}$	1 $_{\times 1}$	1 $_{\times 0}$	1	0
0 $_{\times 1}$	0 $_{\times 0}$	1 $_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

B/W Image

4	3	4
2	4	3
2	3	4

Convolved Feature

4= 1.1+1.0+1.1
+0.0+1.1+1.0
+0.1+0.0+1.1

Convolution basics

Convolution: continuous and discrete

$$(f*g)(t)=\int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

This is the area under the curve $f(\tau)$ **weighted by** $g(t-\tau)$

$$(f*g)[n]=\sum_{m=-\infty}^{+\infty} f(m)g(n-m)$$

Convolution of two vectors

V_1: <0, 1, 2, 3, 4, 5, 6, 7, 8, 9>

V_2: <1, 1, 1>

$V_1 \oplus V_2 =$

<(0.1+1.1+2.1), (1.1+2.1+3.1),
(2.1+3.1+4.1), (3.1+4.1+5.1),
(4.1+5.1+6.1), (5.1+6.1+7.1),
(6.1+7.1+8.1), (7.1+8.1+9.1)>

=<3, 6, 9, 12, 15, 18, 21, 24>

Receptive field and selective emphasis/de-emphasis

- The filter <1,1,1> given equal “emphasis” to constituents of the “receptive field” which means region of interest
- Sliding of the filter corresponds to taking different receptive fields
- By designing the filter as <0,1,0>, we emphasise the center of the receptive field

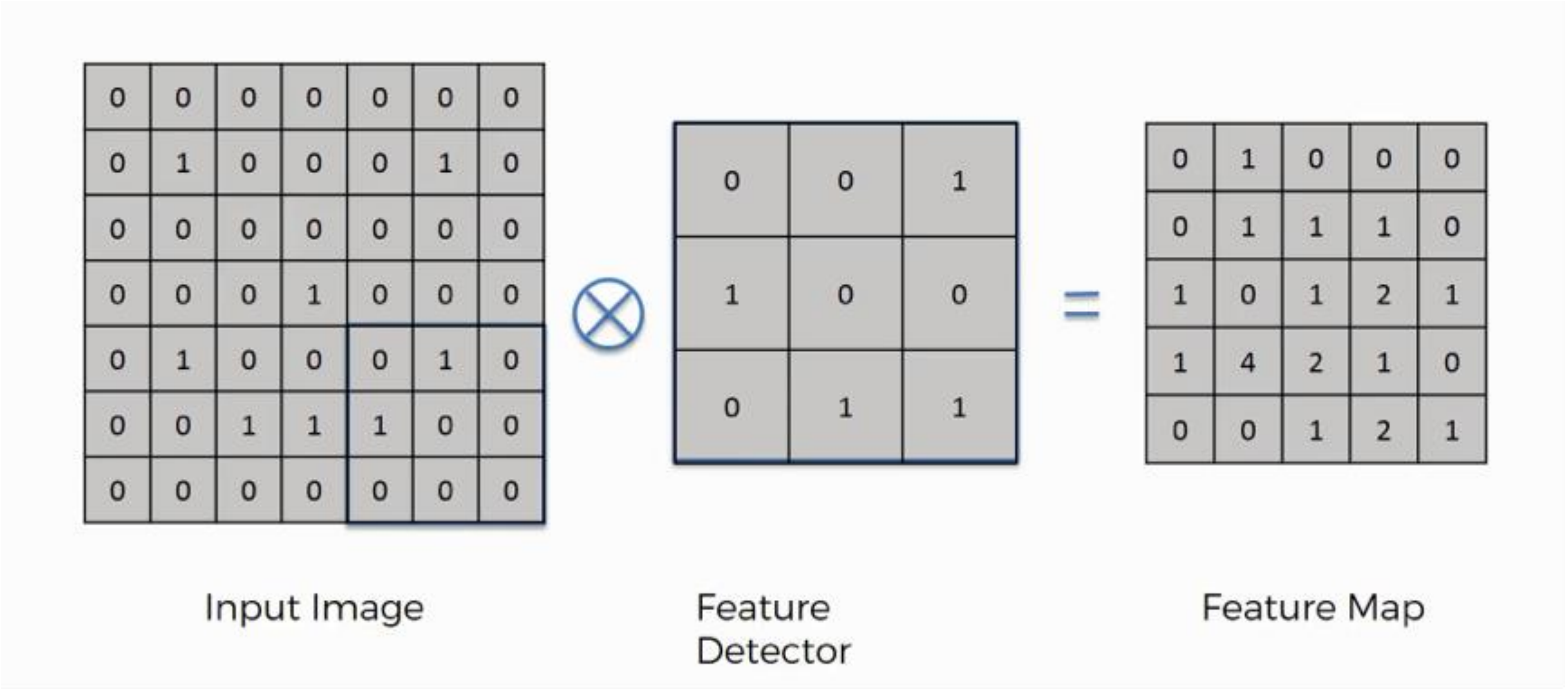
“dog” image and “cat” image

- For dog, the face is of conical shape
- For cat, the shape is round
- So, this distinguishing feature is important for classification
- The filter should have the ability of detecting this kind of feature

Interpretation of convolution

- The filter/kernel/feature_extractor highlights features and obtains those features
- The sliding achieves the effect of focussing on “region” after “region”
- This resembles sequence processing
- The filter components are **LEARNT**

Convolution as feature extractor



CNN architecture

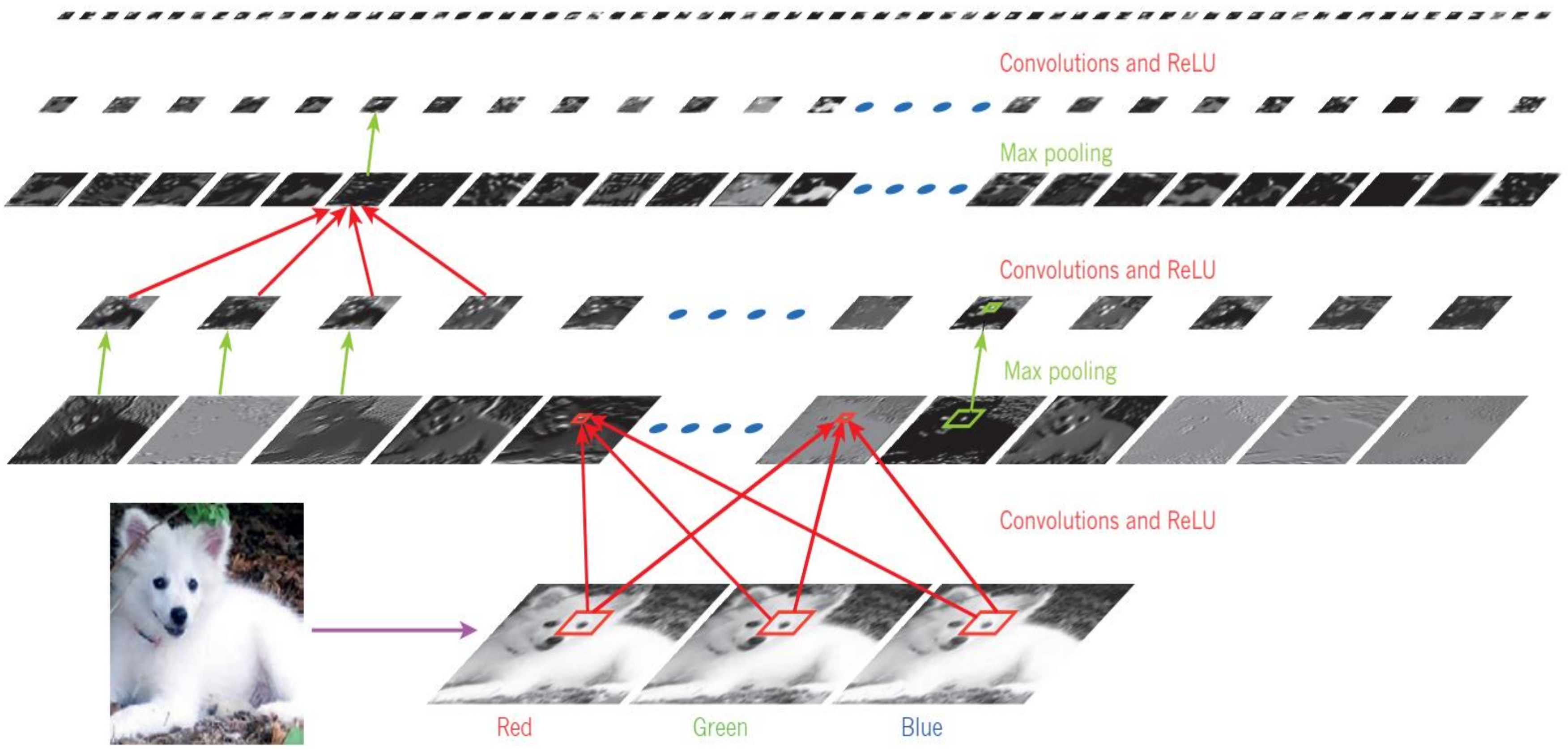
- Several layers of convolution with *tanh* or *ReLU* applied to the results
- In a traditional feedforward neural network we connect each input neuron to each output neuron in the next layer. That's also called a fully connected layer, or affine layer.
- In CNNs we use convolutions over the input layer to compute the output.
- This results in local connections, where each region of the input is connected to a neuron in the output

Key Ideas

Four key ideas that take advantage of the properties of natural signals:

- local connections,
- shared weights,
- pooling and
- the use of many layers

A typical ConvNet



Lecun, Bengio, Hinton, Nature, 2015

Why CNN became a rage: image

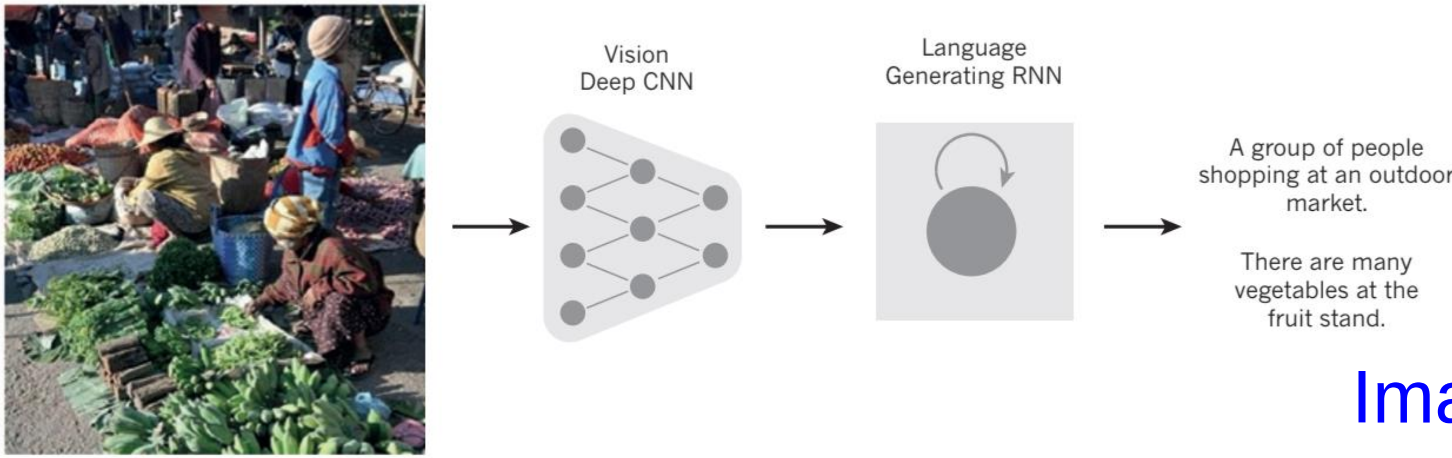


Image Captioning-1

A **stop** sign is on a road with a mountain in the background

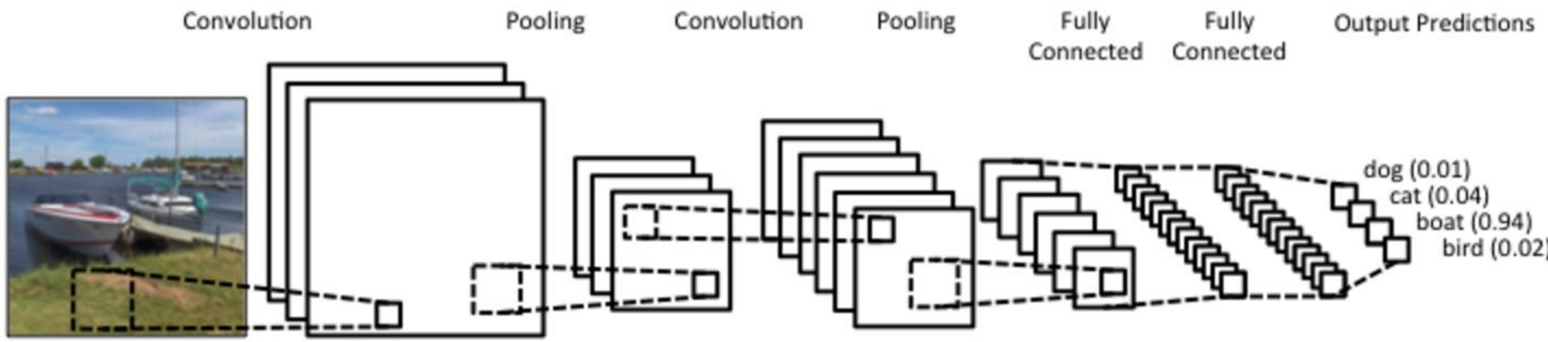
Image Captioning-2

Role of ImageNet

- Million images from the web
- 1,000 different classes
- Spectacular results!
- Almost halving the error rates of the best competing approaches1.

Learning in CNN

- **Automatically learns the values of its filters**
- For example, in Image Classification learn to
  - detect edges from raw pixels in the first layer,
  - then use the edges to detect simple shapes in the second layer,
  - and then use these shapes to detect higher-level features, such as facial shapes in higher layers.
  - The last layer is then a classifier that uses these high-level features.



http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Pooling

- Gives invariance in translation, rotation and scaling
- Important for image recognition
- Role in NLP?

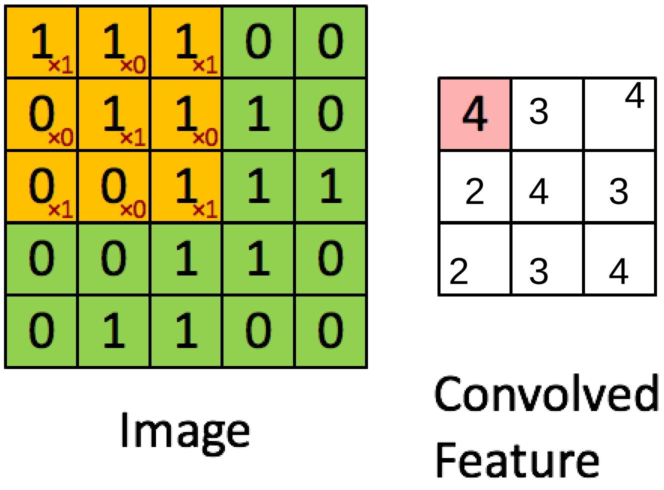
CNN for NLP

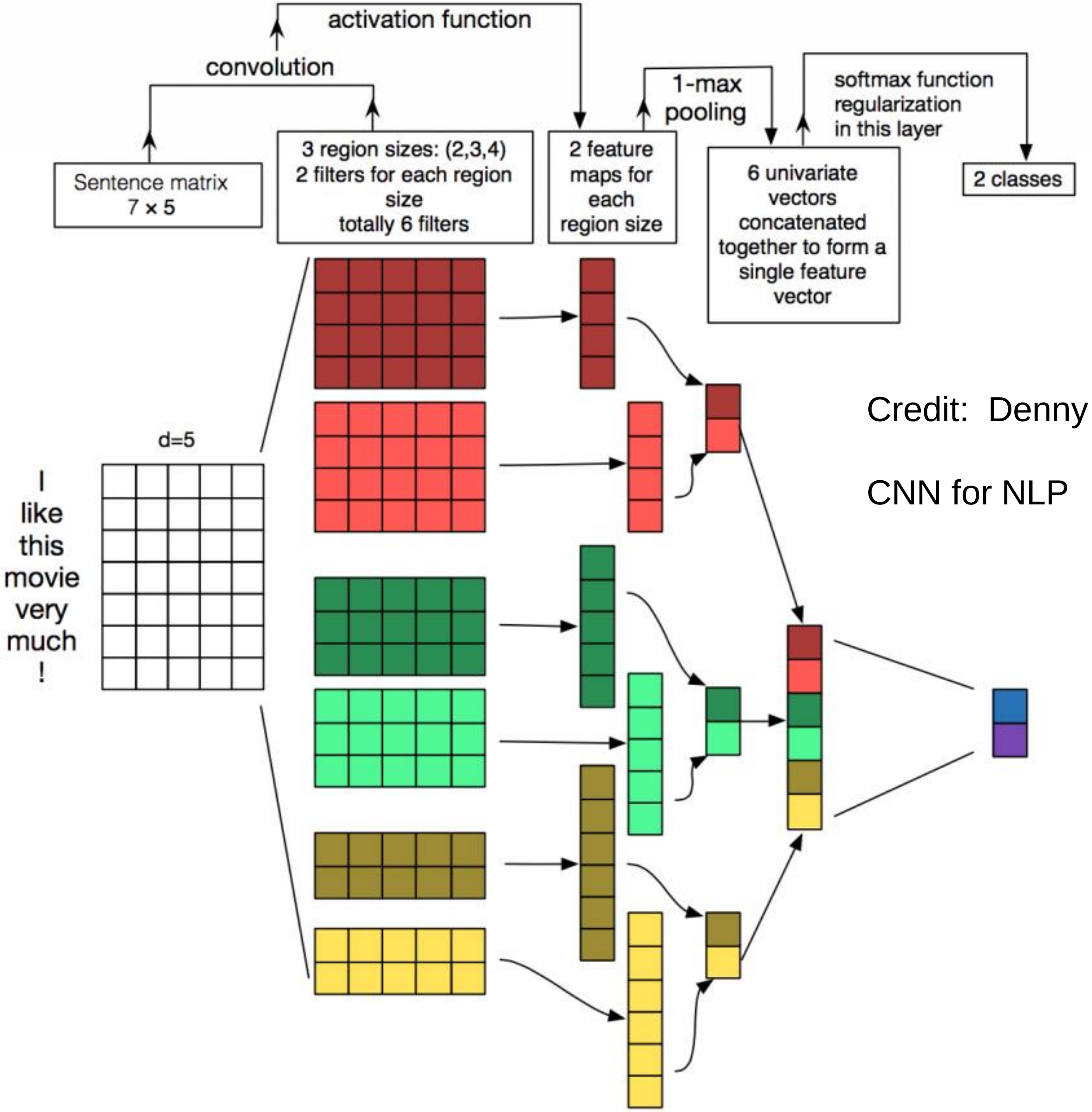
Input matrix for CNN: NLP

- “image” for NLP ←→ word vectors in the rows

- For a 10 word sentence using a 100-dimensional Embedding,

- we would have a 10×100 matrix as our input





Credit: Denny Britz

CNN for NLP

Role of multiple filters in CNN

- In the last slide- 2 filters per n-gram (n=2, 3, 4)
- In multitask learning setting, for tasks such as sentiment analysis and emotion analysis multiple filters can be used.
- Multiple filters allow multiple views and emphasis angles for each task. For instance one filter for sentiment analysis and another for emotion analysis.
- The number of filters should be equal to the number of tasks.

Role of lower order ngrams

- Lower order ngrams play an important role in vocabulary matching.
- Lower order ngrams give importance to lexical properties. For instance:
  - Unigram: I like this movie.
  - Bigram: I do not like this movie.

Role of higher order ngrams

- Higher order ngrams give emphasis to syntactic structure of the sentence and the dependencies.
- For instance:
  - Trigram: *I like this movie* (like ←→ movie)
  - Quadrigram and pentagram capture more dependencies and syntactic structure and play an important role in tasks like sentiment analysis, emotion detection, machine translation, etc.
  - Example: *John watched a movie with James yesterday in Melbourne* (Who did what to whom when and where type dependency)

CNN Hyper parameters

- Narrow width vs. wide width
- Stride size
- Pooling layers
- Channels

Detailing out CNN layers

Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

CNN stages

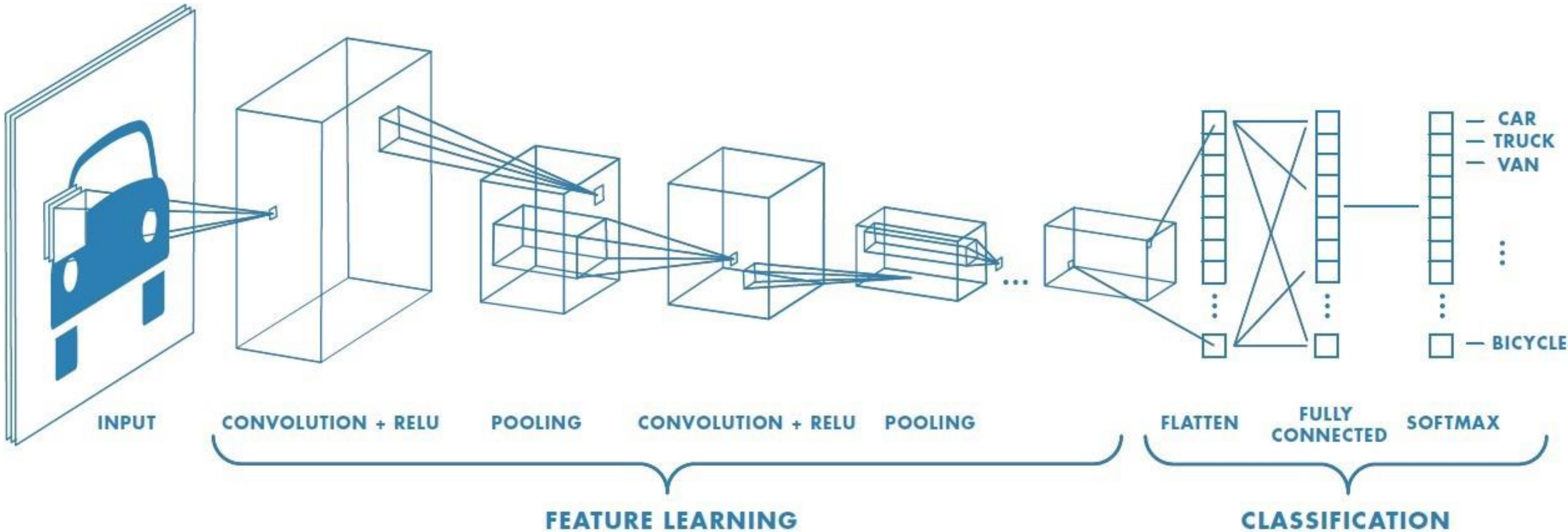


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Another depiction

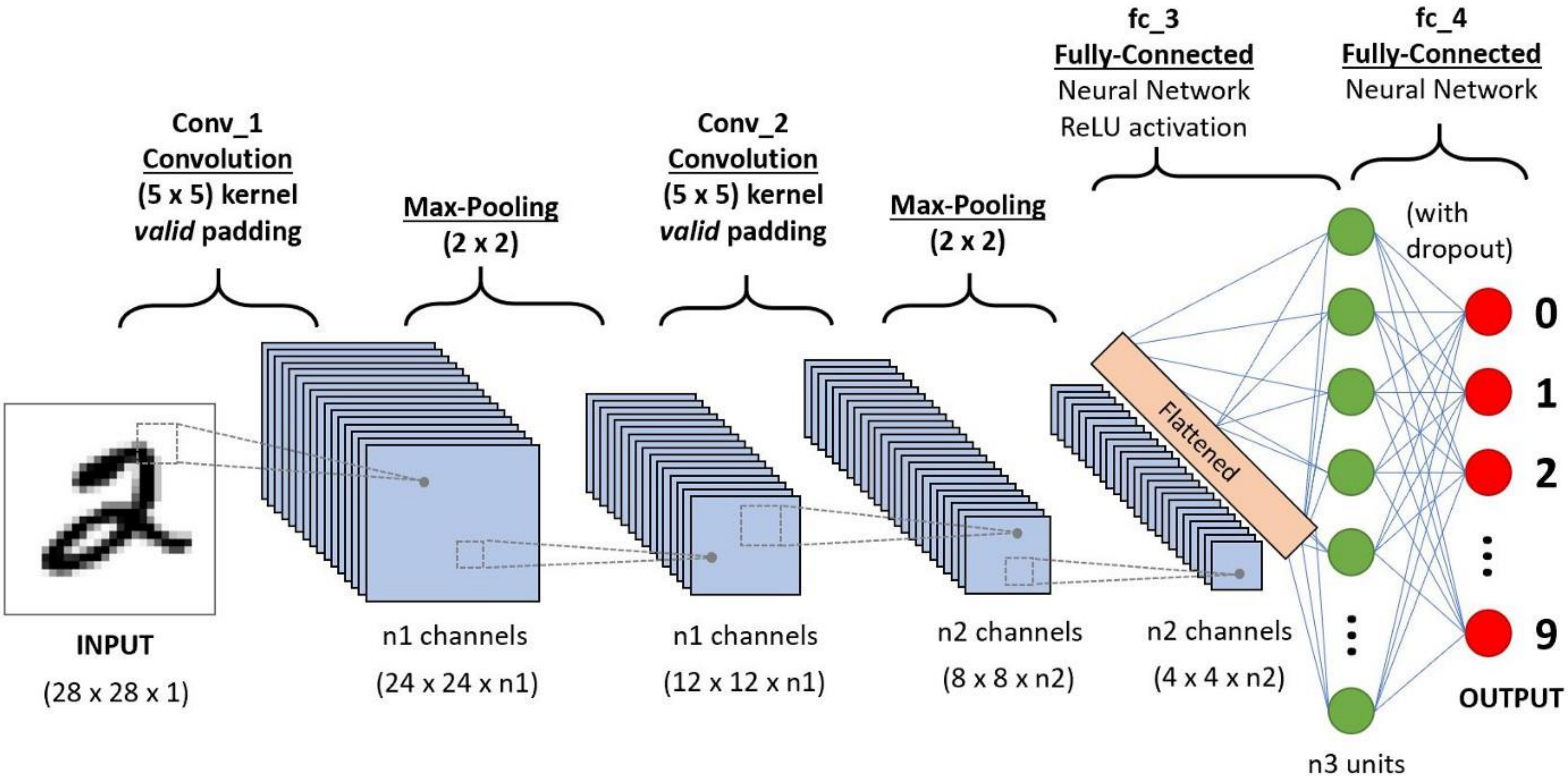
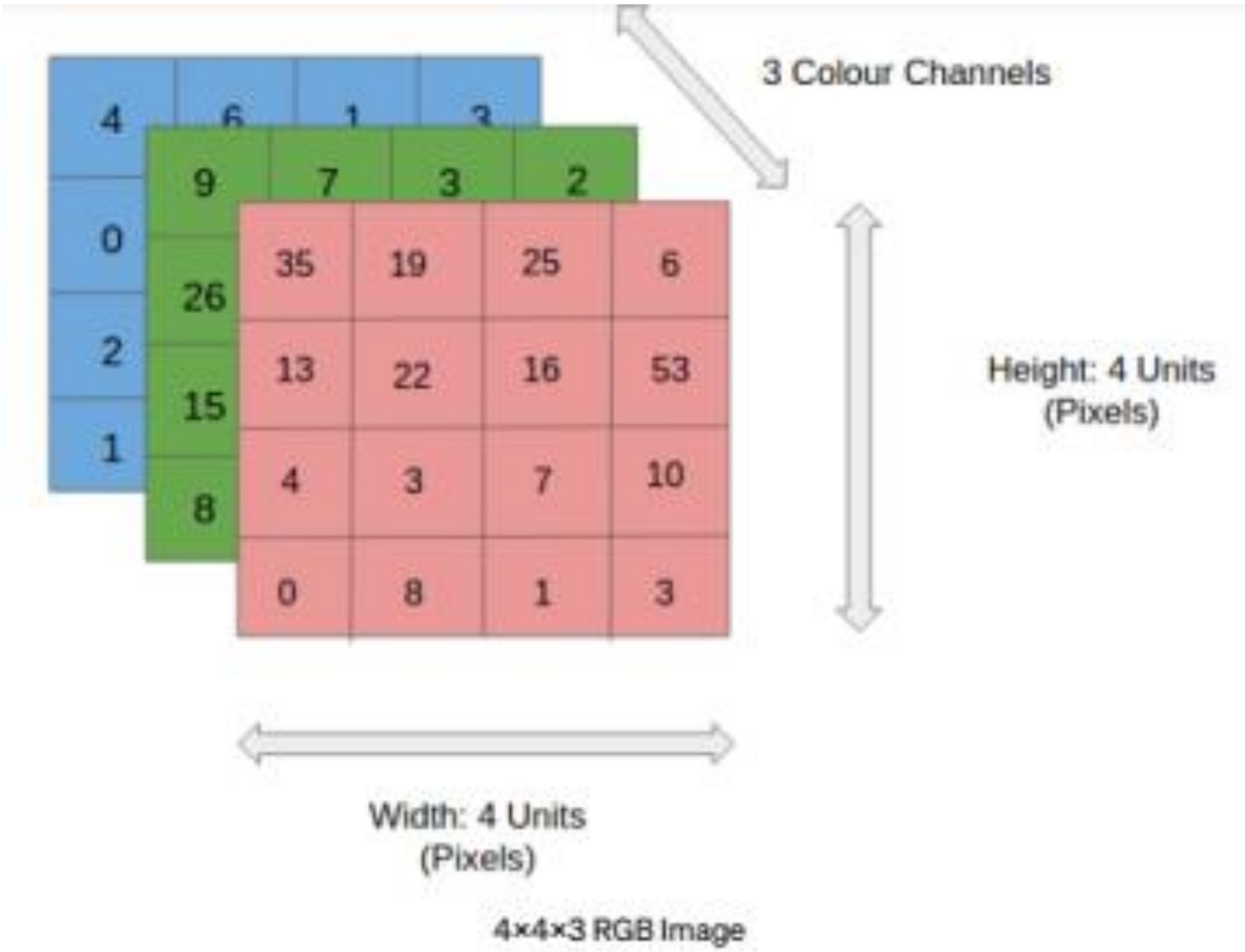
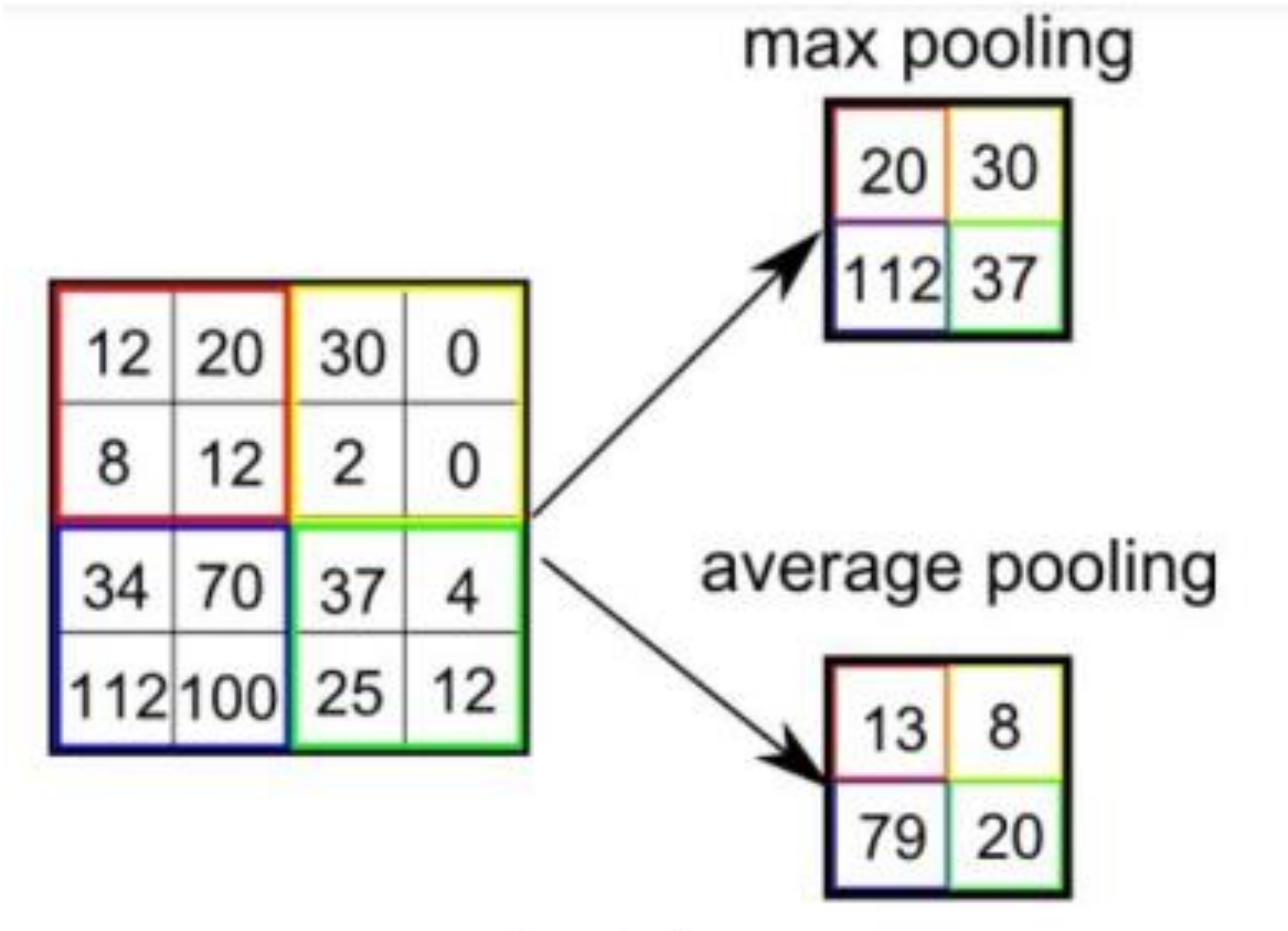


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Channelized Image

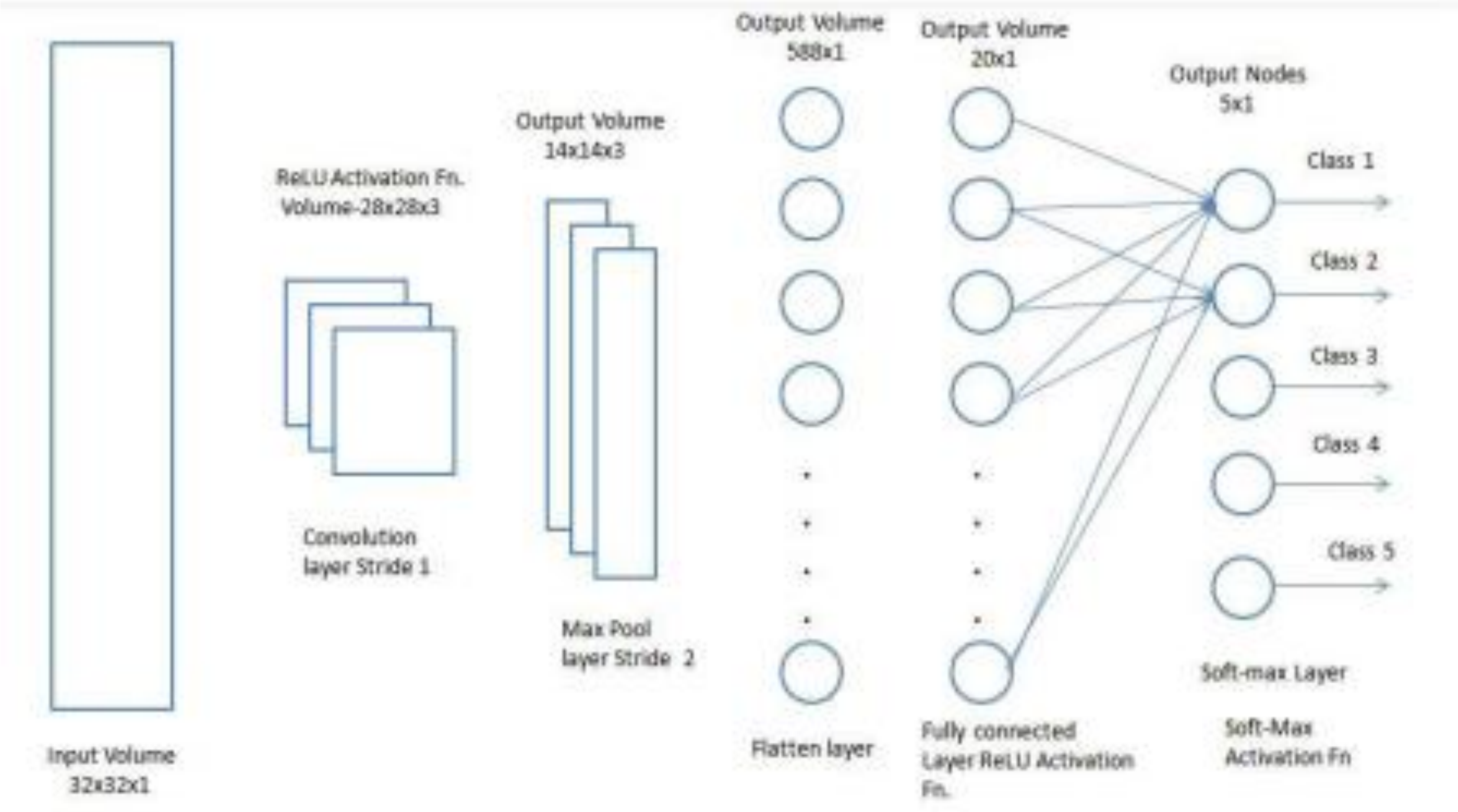


Pooling



Types of Pooling

Complete Architecture



Convolution Layer

- Input is a tensor with a shape
  - (number of inputs) x (input height) x (input width) x (input channels)
- After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape
  - (number of inputs) x (feature map height) x (feature map width) x (feature map channels).

Tensors and Vectors

- **Tensors**: vectors of vectors
- Vector, *V*: <1, 2, 3, 4, 5>
- Tensor, T1: <<1, 2, 3>, <4, 5, 6>>
- Tensor, T2: <<<1,2>, <3>>, <<4>, <5,6>>>
- **Channels**: R, G, B
- Each image consists of Red, Green and Blue channels- that is, 3 different matrices of pixel values

Pooling Layer

- "Pooling" involves sliding a two-dimensional filter over each channel of feature map
- Effect: summarizing the features
- For a feature map having dimensions $\mathbf{n_h}$ **x** $\mathbf{n_w}$ **x** $\mathbf{n_c}$, the output dimension after pooling is

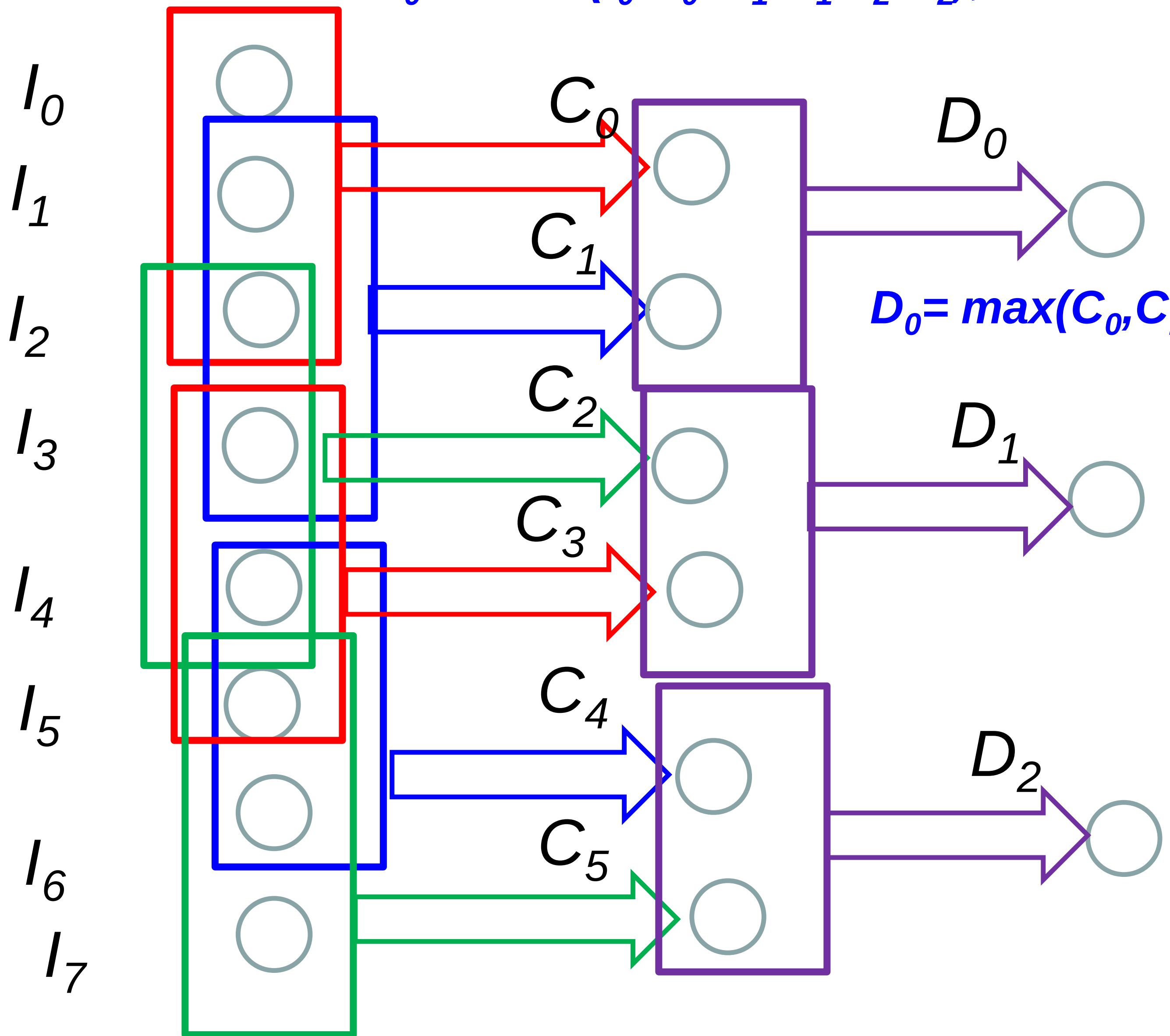
$$\left(\frac{n_h - f_h + 1}{s}\right) . \left(\frac{n_w - f_w + 1}{s}\right) (.n_c)$$

where, n_h= height of feature map, n_w=width, n_c= number of channels, f_h=height of filter, f_w=width of filter, s=stride length

Learning in CNN

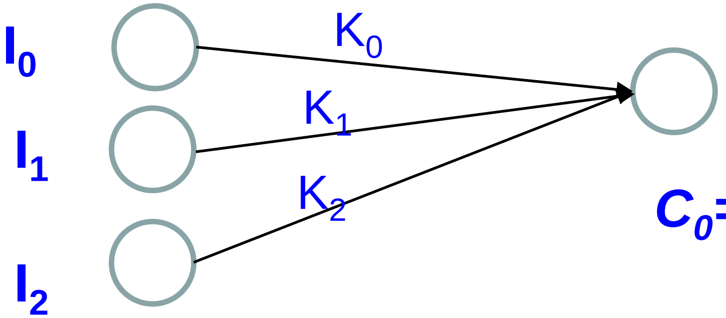
First Kernel+RELU+POOLING

C_0= RELU($I_0.K_0$+ $I_1.K_1$+$I_2.K_2$); Ks are kernel "weights"



D_0= max(C_0,C_1)

Fleshing out the details



$C_0 = RELU(I_0.K_0 + I_1.K_1 + I_2.K_2)$

Input vector I

New K_0= old K_0+sum of ΔK_0s across C_0, C_1..C_5

This addition does not violate gradient descent rule

Normal BP works

- Backpropagate from the final layer of softmax.
- When it comes to the first convolution layer, post the changes in the weights, maintaining the constraint that kernel values are parameter-shared
- Nothing special needs to be done for RELU and MAX functions

Another depiction

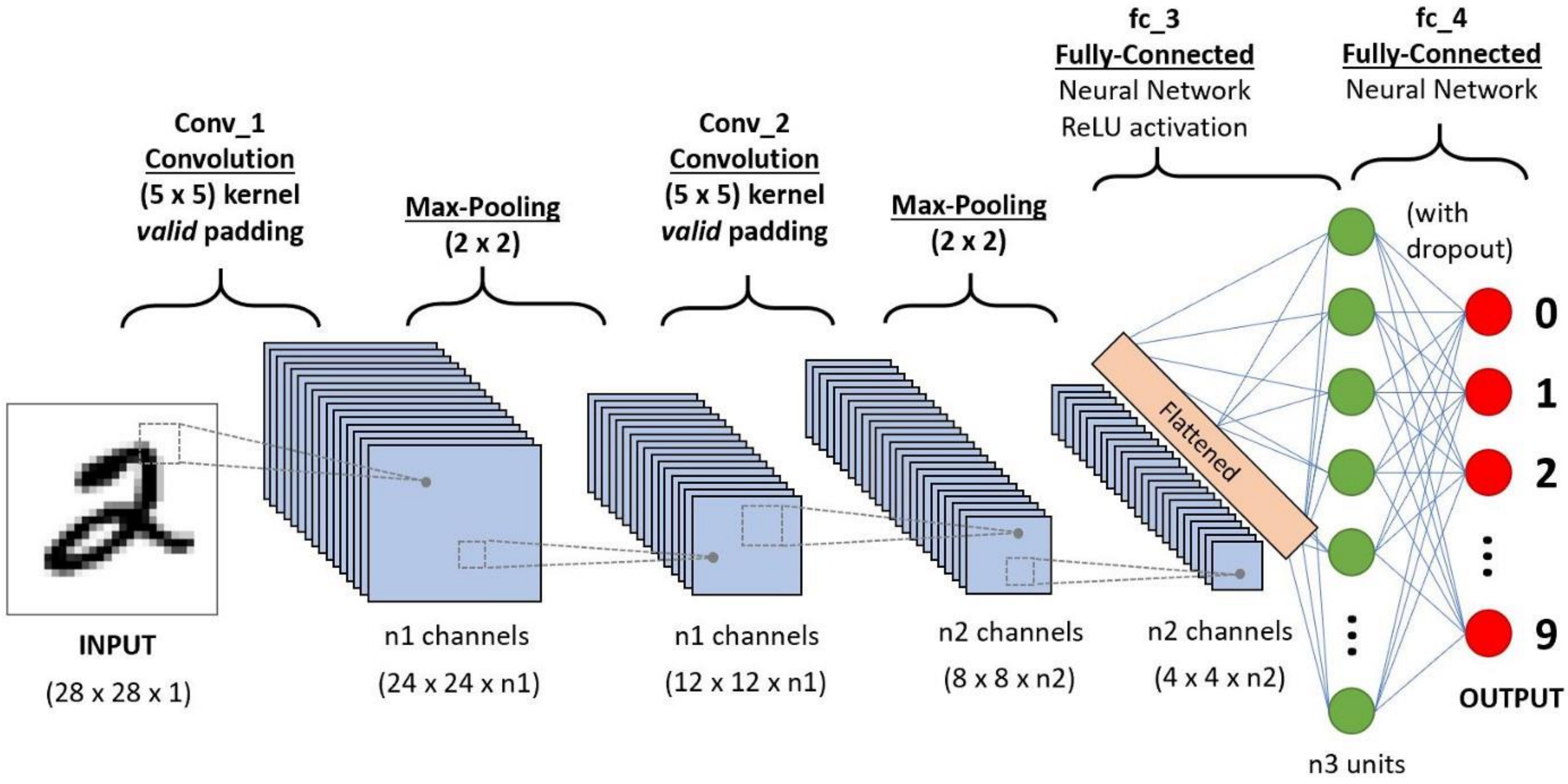


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

An application: Sarcasm Detection

Illustrates use of CNN Channels

Sarcasm Detection: a sub-problem of Sentiment and Emotion Analysis

Sentiment Analysis: The task of identifying if a certain piece of text contains any opinion, emotion or other forms of affective content.

Machine Learning based approach: classifiers and features

- SVM, KNN and Random Forest classifiers

○ Sentiment-based features

- Number of
  - positive words
  - negative words
  - highly emotional positive words,
  - highly emotional negative words.

- Positive/Negative word is said to be highly emotional if it’s POS tag is one amongst : ’JJ', ‘JJR', ‘JJS', ‘RB', ‘RBR', ‘RBS', ‘VB', ‘VBD', ‘VBG', ‘VBN', ‘VBP', ‘VBZ'.

Emotion Features

- Positive emoticon
- Negative emoticon

- Boolean features that are 1 if both positive and negative words are present in the tweet.
- Boolean features that are 1 when positive (negative) word and negative (positive) emoji are simultaneously present

Punctuation features

- number of exclamation marks.
- number of dots
- number of question mark.
- number of capital letter words.
- number of single quotations.

- Number in the tweet: This feature is simply the number present in the tweet.
- Number unit in the tweet : This feature is a one hot representation of the type of unit present in the tweet. Example of number unit can be hour, minute, etc.

Comparison of results (1: sarcastic, 0: non-sarcastic)

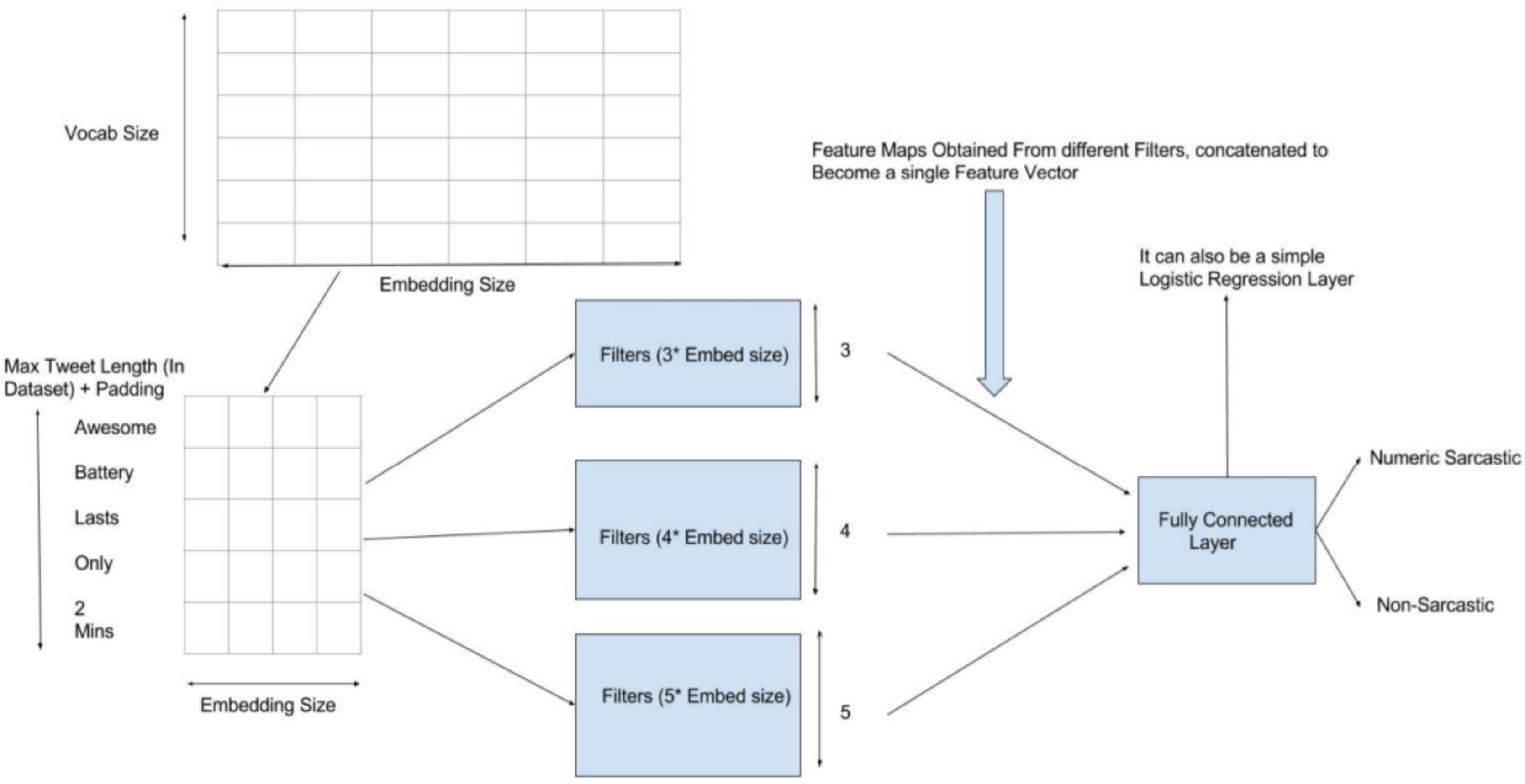
Approaches	Precision			Recall			F-score		
	P(1)	P(0)	P(avg)	R(1)	R(0)	R(avg)	F(1)	F(0)	F(avg)
Past Approaches									
Buschmeier et.al.	0.19	0.98	0.84	0.99	0.07	0.24	0.32	0.13	0.16
Liebrecht et.al.	0.19	1.00	0.85	1.00	0.07	0.24	0.32	0.13	0.17
Gonzalez et.al.	0.19	0.96	0.83	0.99	0.06	0.23	0.32	0.12	0.15
Joshi et.al.	0.20	1.00	0.86	1.00	0.13	0.29	0.33	0.23	**0.25**
Rule-Based Approaches									
Approach-1	0.53	0.87	0.81	0.39	0.92	0.83	0.45	0.90	**0.82**
Approach-2	0.44	0.85	0.78	0.28	0.92	0.81	0.34	0.89	0.79
Machine-Learning Based Approaches									
SVM	0.50	0.95	0.87	0.80	0.82	0.82	0.61	0.88	**0.83**
KNN	0.36	0.94	0.84	0.81	0.68	0.70	0.50	0.79	0.74
Random Forest	0.47	0.93	0.85	0.74	0.81	0.80	0.57	0.87	0.82

Deep Learning based

- Very little feature engg!!

- EmbeddingSize of 128
- Maximum tweet length 36 words
- Padding used
- Filters of size 3, 4, 5 used to extarct features

Deep Learning based approach: CNN-FF Model

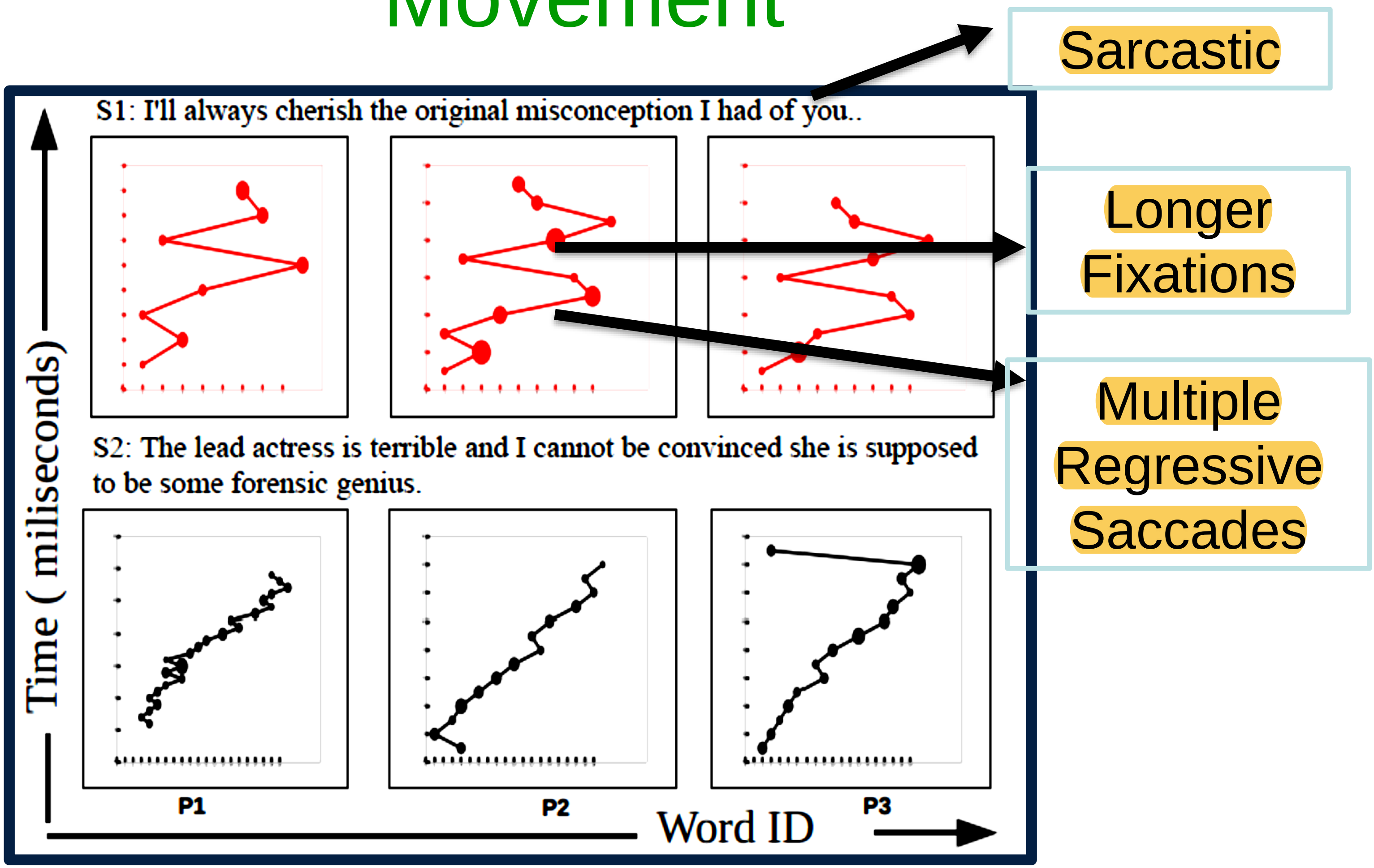


Comparison of results (1: sarcastic, 0: non-sarcastic)

Approaches	Precision			Recall			F-score		
	P(1)	**P(0)**	**P(avg)**	**R(1)**	**R(0)**	**R(avg)**	**F(1)**	**F(0)**	**F(avg)**
Past Approaches									
Buschmeier et.al.	0.19	0.98	0.84	0.99	0.07	0.24	0.32	0.13	0.16
Liebrecht et.al.	0.19	1.00	0.85	1.00	0.07	0.24	0.32	0.13	0.17
Gonzalez et.al.	0.19	0.96	0.83	0.99	0.06	0.23	0.32	0.12	0.15
Joshi et.al.	0.20	1.00	0.86	1.00	0.13	0.29	0.33	0.23	**0.25**
Rule-Based Approaches									
Approach-1	0.53	0.87	0.81	0.39	0.92	0.83	0.45	0.90	**0.82**
Approach-2	0.44	0.85	0.78	0.28	0.92	0.81	0.34	0.89	0.79
Machine-Learning Based Approaches									
SVM	0.50	0.95	0.87	0.80	0.82	0.82	0.61	0.88	**0.83**
KNN	0.36	0.94	0.84	0.81	0.68	0.70	0.50	0.79	0.74
Random Forest	0.47	0.93	0.85	0.74	0.81	0.80	0.57	0.87	0.82
Deep-Learning Based Approaches									
CNN-FF	**0.88**	**0.94**	**0.93**	**0.71**	**0.98**	**0.93**	**0.79**	**0.96**	**0.93**
CNN-LSTM-FF	0.82	0.94	0.92	0.72	0.96	0.92	0.77	0.95	0.92
LSTM-FF	0.76	0.93	0.90	0.68	0.95	0.90	0.72	0.94	0.90

back

Sentiment Annotation and Eye Movement

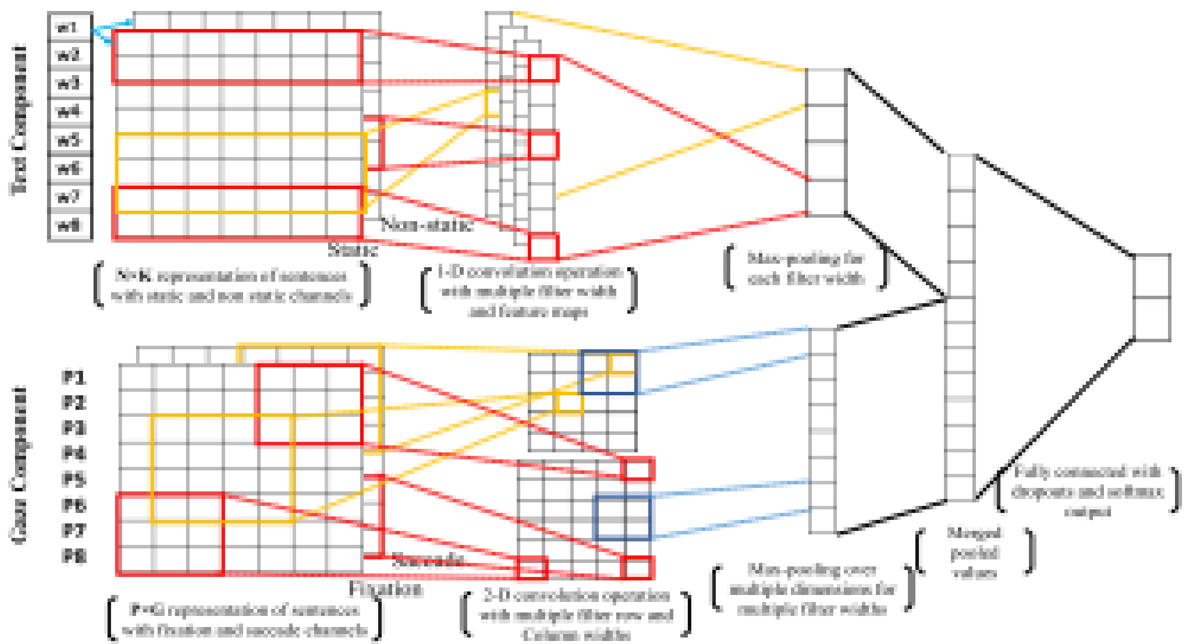


Datasets

- Two publicly available datasets released by us (Mishra et al, 2016; Mishra et al., 2014)
- **Dataset 1: (Eye-tracker: Eyelink-1000 Plus)**
  - 994 text snippets : 383 positive and 611 negative, 350 are sarcastic/ironic
  - Mixture of Movie reviews, Tweets and sarcastic/ironic quotes
  - Annotated by 7 human annotators
  - Annotation accuracy: **70%-90%** with Fleiss kappa IAA of **0.62**
- **Dataset 2: (Eye-tracker: Tobi TX300)**
  - 843 snippets : 443 positive and 400 negative
  - Annotated by 5 human subjects
  - Annotation accuracy: **75%-85%** with Fleiss kappa IAA of **0.68**

CNN Based Sarcasm Detection

Abhijit Mishra, Kuntal Dey and Pushpak Bhattacharyya, *Learning Cognitive Features from Gaze Data for Sentiment and Sarcasm Classification Using Convolutional Neural Network*, **ACL 2017**, Vancouver, Canada, July 30-August 4, 2017.



Learning Cognitive Features from Gaze Data for Sentiment and Sarcasm Classification

- In complex classification tasks like sentiment analysis and sarcasm detection, extraction and choice of features should be learnt
- CNN channels exploited
- CNN learns features from both gaze and text and uses them to classify the input text

Central Idea

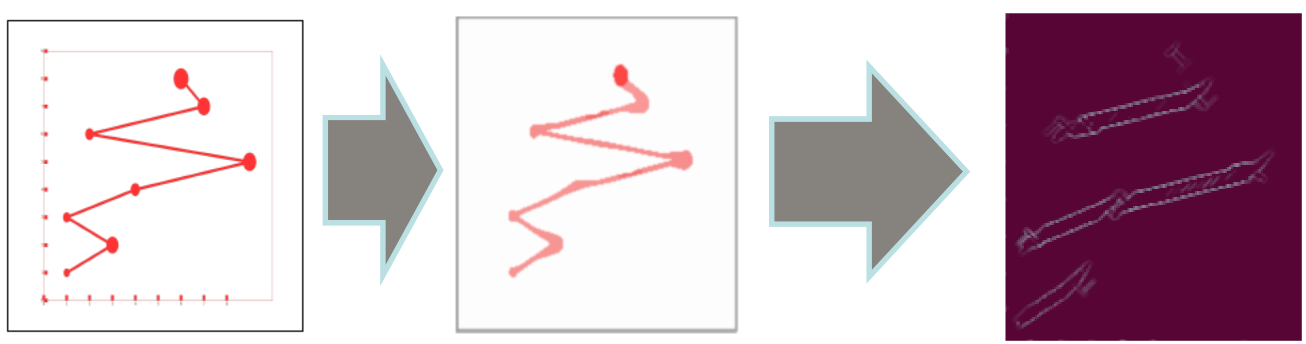
- Learn features from Gaze sequences (fixation duration sequences and gaze-positions) and Text automatically using Deep Neural Networks.
- Deep NNs have proven to be good at learning feature representations for Image and Text classification tasks (Krizhevsky et al., 2012;Collobert et al., 2011).
- Use Convolutional Neural Network (already used for sentiment classification, Kim, 2014)

Why Convolutional NNs

- Convolutional Layers good at capturing compositionality (Lawrence et al, 1997).

Image

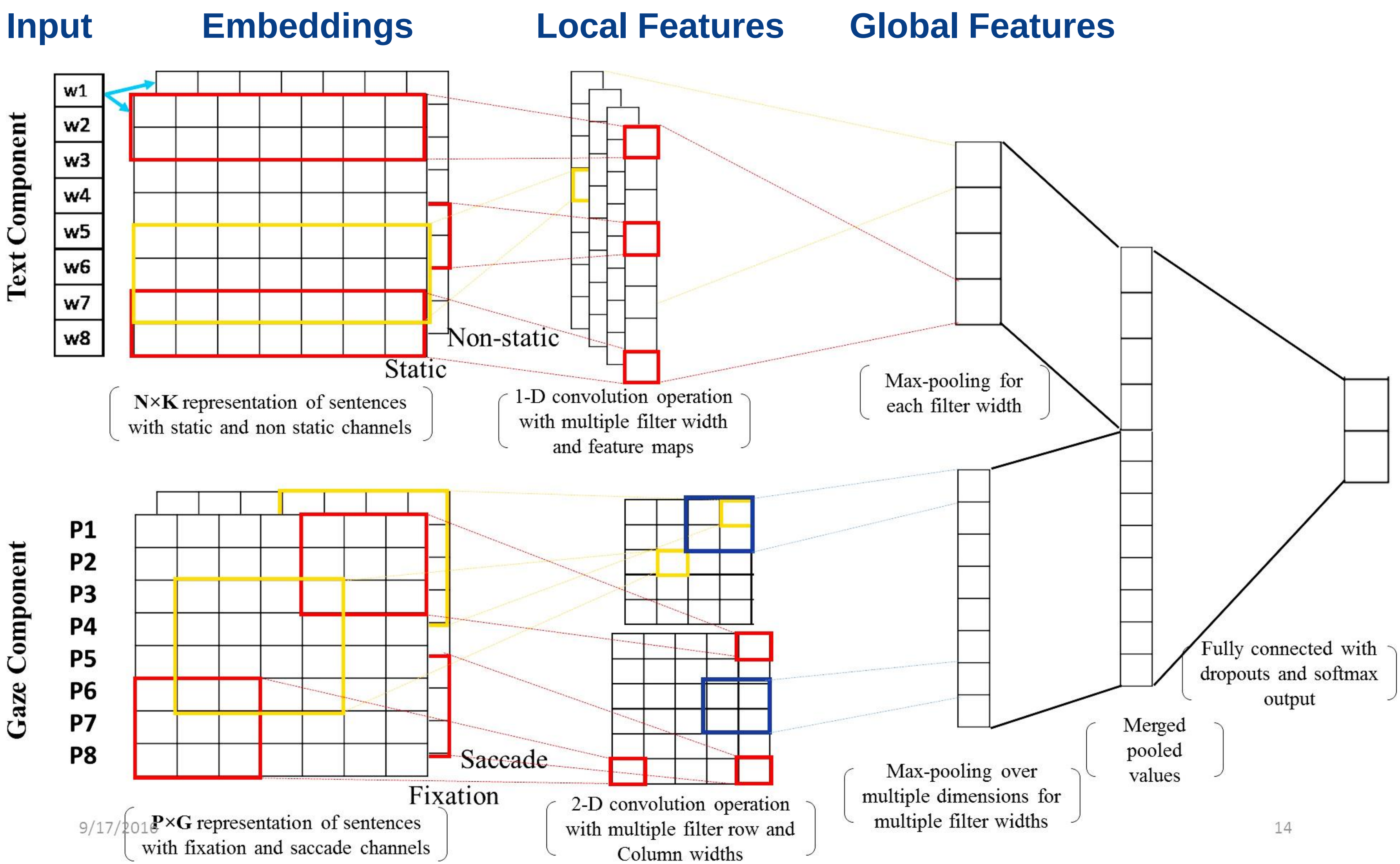
Gaze



??

Images taken from: mrulafi.blogspot.com

Neural Network Architecture



Why both Static and Non-static embedding

- Non-static embedding channel for tuning embeddings for SA/Sarcasm (e.g., produce similar embeddings for adjectives like good and excellent)
- Static embedding channel: to prevent over-tuning of embeddings due to collocation (e.g., words such as I and love are often collocated but should not share similar vector representation).

Fixation and Saccade Channels

- Fixation channel: Lexical Complexity (pertaining to length, frequency and predictability of words while annotation)
- Saccade channel: Syntactic Complexity and Incongruity

Datasets (1/2)

- Two publicly available datasets released by us (Mishra et al, 2016; Mishra et al., 2014)
- Dataset 1: (Eye-tracker: Eyelink-1000 Plus)
  - 994 text snippets : 383 positive and 611 negative, 350 are sarcastic/ironic
  - Mixture of Movie reviews, Tweets and sarcastic/ironic quotes
  - Annotated by 7 human annotators
  - Annotation accuracy: 70%-90% with Fleiss kappa IAA of 0.62

Datasets (2/2)

- Dataset 2: (Eye-tracker: Tobi TX300)
- 843 snippets : 443 positive and 400 negative
- Annotated by 5 human subjects
- Annotation accuracy: 75%-85%with Fleiss kappa IAA of 0.68

Experimental Setup: Configurations

- Text Only: (Only Text Component is Used)
  - Text_Static: Word embeddings are kept static and not updated during back propagation.
  - Text_Non-static: Embeddings are updated during back propagation.
  - Text_Multi Channel: Two channels (one taking input from static and one from dynamic embeddings) are used.

- Gaze Only: (Only Gaze Component is Used)
  - Gaze_Fixation_Duration: Sequence of fixation durations are used as input
  - Gaze_Saccade: Sequence of gaze locations (in terms of word ID used as input)
  - Gaze-Multi Channel: Two channels (one taking input from Fixation and one from saccade) are used

- Both text and Gaze (9-Configs)

Experiment Setup (Model Details)

- Word Embeddings: Word2Vec (Mikolov et.al), trained on Amazon Movie Review Data, Embedding dimensions: 300
- Convolution: Filter sizes: 3,4 (Best), Number of filters used for each filter size: 150 (Better than smaller values)
- Feed-Forward: Number of hidden neurons: 150 (Better than smaller values), Dropout probability: 0.25
- Training: Number of epochs: 200 (change in loss negligible after 200 epochs), Optimizer: Adadelta, LR: 0.1

Results – Sentiment Analysis

	Configuration	Dataset1 P	Dataset1 R	Dataset1 F	Dataset2 P	Dataset2 R	Dataset2 F
Traditional systems based on textual features	Näive Bayes	63.0	59.4	61.14	50.7	50.1	50.39
	Multi-layered Perceptron	69.0	69.2	69.2	66.8	66.8	66.8
	SVM (Linear Kernel)	72.8	73.2	72.6	70.3	70.3	70.3
Systems by Mishra et al. (2016c)	Gaze based (Best)	61.8	58.4	60.05	53.6	54.0	53.3
	Text + Gaze (Best)	**73.3**	**73.6**	**73.5**	**71.9**	**71.8**	**71.8**
CNN with only text input (Kim, 2014)	STATICTEXT	63.85	61.26	62.22	55.46	55.02	55.24
	NONSTATICTEXT	72.78	71.93	72.35	60.51	59.79	60.14
	MULTICHANNELTEXT	72.17	70.91	71.53	60.51	59.66	60.08
CNN with only gaze Input	FIXATION	60.79	58.34	59.54	53.95	50.29	52.06
	SACCADE	64.19	60.56	62.32	51.6	50.65	51.12
	MULTICHANNELGAZE	65.2	60.35	62.68	52.52	51.49	52
CNN with both text and gaze Input	STATICTEXT + FIXATION	61.52	60.86	61.19	54.61	54.32	54.46
	STATICTEXT + SACCADE	65.99	63.49	64.71	58.39	56.09	57.21
	STATICTEXT + MULTICHANNELGAZE	65.79	62.89	64.31	58.19	55.39	56.75
	NONSTATICTEXT + FIXATION	73.01	70.81	71.9	61.45	59.78	60.60
	NONSTATICTEXT + SACCADE	77.56	73.34	75.4	**65.13**	**61.08**	**63.04**
	NONSTATICTEXT + MULTICHANNELGAZE	**79.89**	**74.86**	**77.3**	63.93	60.13	62
	MULTICHANNELTEXT + FIXATION	74.44	72.31	73.36	60.72	58.47	59.57
	MULTICHANNELTEXT + SACCADE	**78.75**	**73.94**	**76.26**	63.7	60.47	62.04
	MULTICHANNELTEXT + MULTICHANNELGAZE	**78.38**	**74.23**	**76.24**	64.29	61.08	62.64

Results – Sarcasm Detection

	Configuration	P	R	F
Traditional systems based on textual features	Näive Bayes	69.1	60.1	60.5
	Multi-layered Perceptron	69.7	70.4	69.9
	SVM (Linear Kernel)	72.1	71.9	72
Systems by Riloff et al. (2013)	Text based (Ordered)	49	46	47
	Text + Gaze (Unordered)	46	41	42
System by Joshi et al. (2015)	Text based (best)	70.7	69.8	64.2
Systems by Mishra et al. (2016b)	Gaze based (Best)	73	73.8	73.1
	Text based (Best)	72.1	71.9	72
	Text + Gaze (Best)	76.5	75.3	75.7
CNN with only text input (Kim, 2014)	StaticText	67.17	66.38	66.77
	NonStaticText	84.19	**87.03**	85.59
	MultiChannelText	84.28	**87.03**	85.63
CNN with only gaze input	Fixation	74.39	69.62	71.93
	Saccade	68.58	68.23	68.40
	MultiChannelGaze	67.93	67.72	67.82
CNN with both text and gaze Input	StaticText + Fixation	72.38	71.93	72.15
	StaticText + Saccade	73.12	72.14	72.63
	StaticText + MultiChannelGaze	71.41	71.03	71.22
	NonStaticText + Fixation	**87.42**	85.2	86.30
	NonStaticText + Saccade	84.84	82.68	83.75
	NonStaticText + MultiChannelGaze	84.98	82.79	83.87
	MultiChannelText + Fixation	87.03	86.92	**86.97**
	MultiChannelText + Saccade	81.98	81.08	81.53
	MultiChannelText + MultiChannelGaze	83.11	81.69	82.39

Observations (1/2)

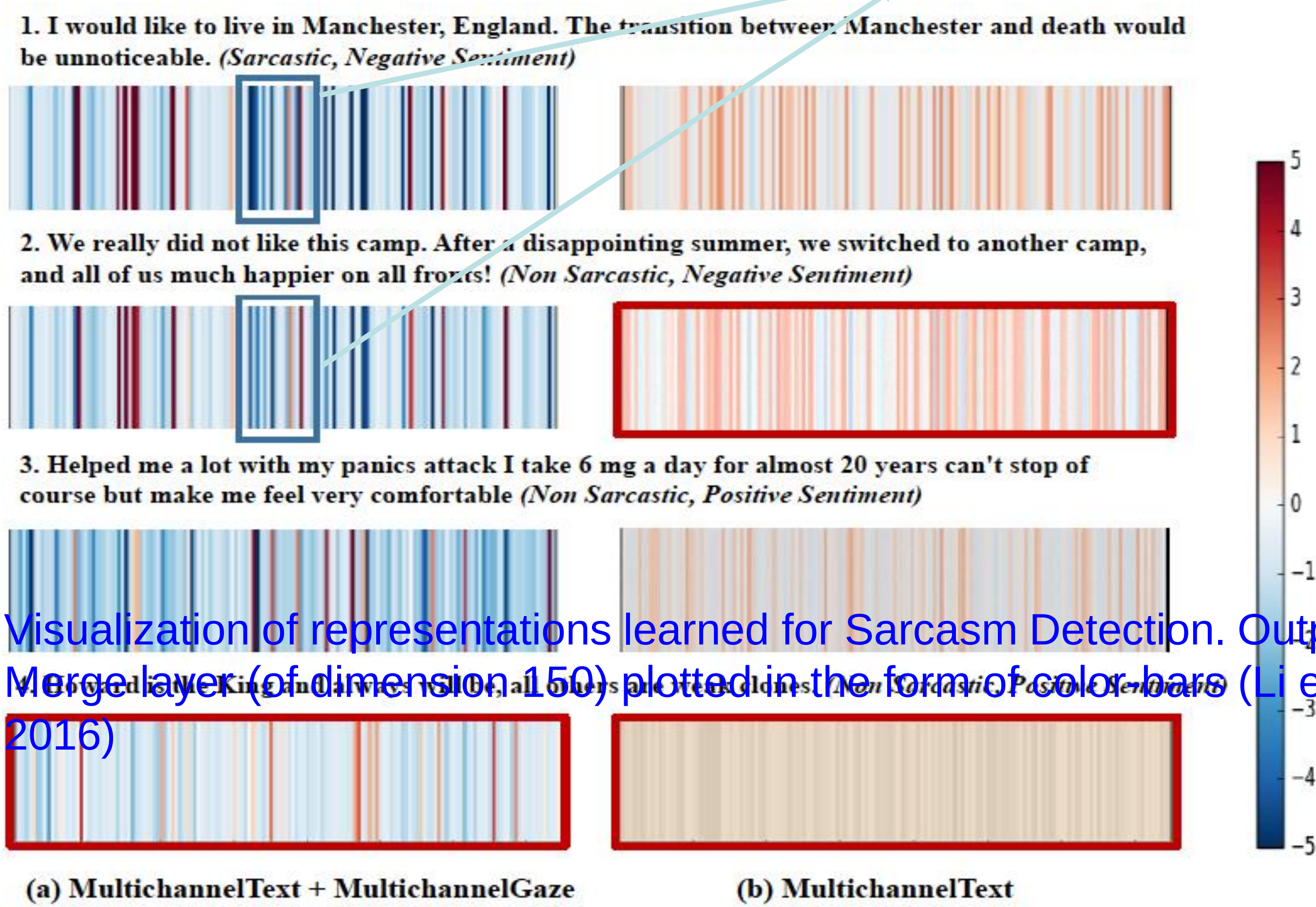
- Overfitting for SA dataset 2: Training accuracy reaches 100 within 25 epochs with validation accuracy still at around 50%. Better dropout/regularization configuration required.
- Better classification accuracy for Sarcasm detection: Clear differences between vocabulary of sarcasm and non-sarcasm classes in our dataset. Captured well by non-static embeddings.
- Effect of dimension variation: Reducing embedding dimension improves by a little margin.

Observations (2/2)

- Increasing filters beyond 180 decreases accuracy (possibly over-fits). Decreasing beyond 30 decreases accuracy.
- Effect of static / non static text channels: Better for non static (word embeddings with similar sentiment come closer in non static channels, e.g., *good* ~ *nice*
- Effect of fixation / saccade channels: Saccade channel alone handles nuances like incongruity better.
- Fixation channel does not help much, may be because of higher variance in fixation duration.

Analysis of Features Learned (1/2)

Capturing intensity variation in sarcasm VS no-sarcasm better



- Visualization of representations learned for Sarcasm Detection. Output of the Merge layer (of dimension 150) plotted in the form of color-bars (Li et al. , 2016)

Analysis of Features Learned (2/2)

- Addition of gaze information helps to generate features with more subtle differences.
- Features for the sarcastic texts exhibit more intensity than the non-sarcastic ones- perhaps capturing the notion that sarcasm typically conveys an intensified negative opinion.
- Example 4 is incorrectly classified by both the systems– lack of context?
- Addition of gaze information does not help here, as it becomes difficult for even humans to classify such texts

Stable Diffusion

Lecture taken by Tejomay Padole, PhD student CFILT

Diffusion Models for Image and Text Generation

Guide: Prof. Pushpak Bhattacharyya

Tejomay Kishor Padole
Department of Computer Science and Engineering
IIT Bombay

Background

Generative modeling

- Estimating $p_{data}(x)$: the probability of observing a data instance x.

No analytical form available for real-world data!

- Desirable properties of a generative model:-
  - **Flexible:** should be able to estimate any data distribution
  - **Tractable:** easy sampling procedure

Timeline

1967 — **ELIZA chatbot:** exploring conversations between a human and a machine

1986 — **Recurrent Neural Networks** : Processes sequential data by using the output from previous steps as inputs for the current step

2003 — **Energy-based models:** inspired from statistical physics

2013 — **Variational Autoencoders:** modeling latent space with a known distribution

2014 — **Generative Adversarial Networks:** directly modeling the data generation process

present — **LLMs:** high quality generation of text.

Diffusion: high-resolution image synthesis

Why diffusion models?

Two ways to model data

Model the data distribution

Model the generating process

VAEs, language models, normalizing flows

GANs

Highly restrictive structure!

Cannot estimate the probability values!

Diffusion models offer flexibility along with allowing estimation of probability.

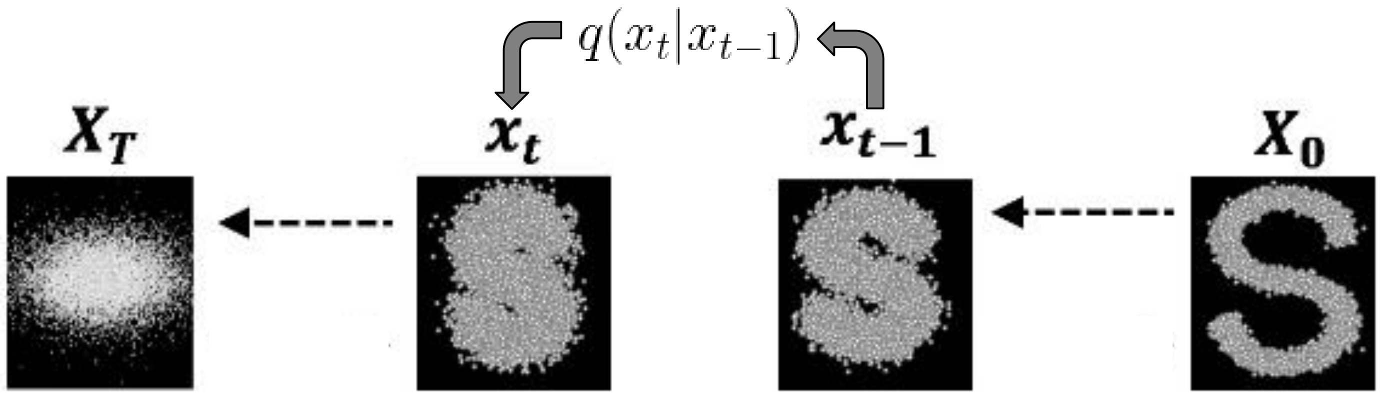
Diffusion Models for Image generation

Main idea

- Inspired from non-equilibrium statistical physics.
- **Forward process:** slowly and iteratively destroys the structure of the input until it gets completely transformed to random noise.
- **Reverse process:** learns to iteratively construct the data from random noise using a Neural Network.
- The noising process can be seen as similar to a drop of ink diffusing into clear water.

Image from https://chemistryclinic.co.uk/1-3-diffusion/

Forward process



Define a markov chain of perturbations using Gaussian distribution

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Gaussian distribution

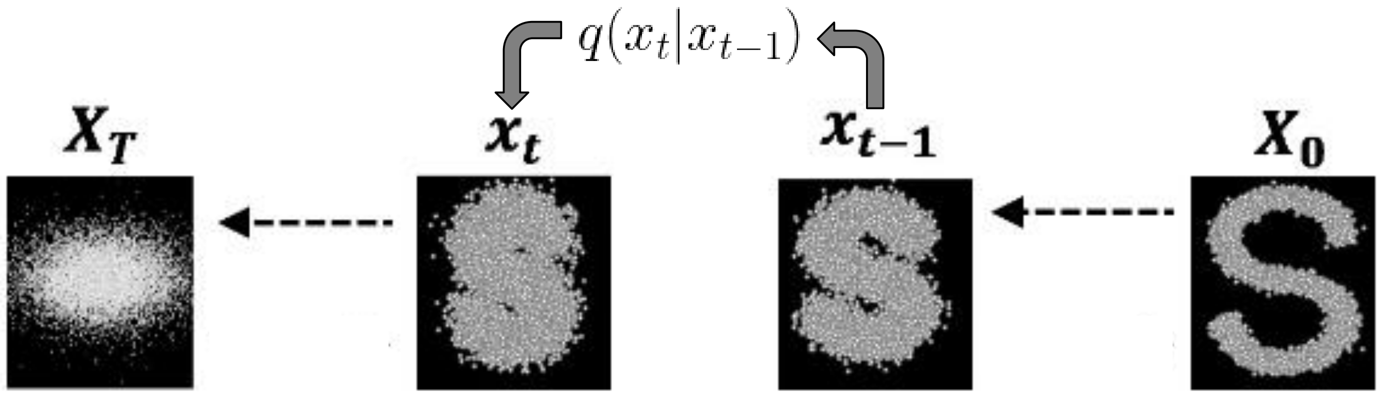
$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I)$$

Part of the previous image

Part of gaussian random noise

[Images taken from https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da]

Forward process



Define a markov chain of perturbations using Gaussian distribution

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Gaussian distribution

$$\beta_t \in (0,1) \quad \forall t \in \{0, T\}$$

Defined by some noise schedule which increases β with *t.*

$$\beta_t \to 0 : \mathcal{N}(x_t; x_{t-1}, 0)$$

$$\beta_t \to 1 : \mathcal{N}(x_t; 0, I)$$

With large number of discrete timesteps, we reach pure random gaussian noise in the end of the forward process

[Images taken from https://towardsdatascience.com/diffusion-models-made-easy-8414298ce4da]

Forward process

We can obtain *$q(x_t|x_0)$* in a closed form to directly sample *x_t* from *x_0* for a timestep t.

$$\text{We define } \alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1 - \alpha_t} \epsilon_{t-1} \text{ ; where } \epsilon_{t-1} \sim \mathcal{N}(0, I)$$

$$\text{substituting } x_{t-1} = \sqrt{\alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_{t-1}} \epsilon$$

$$x_t = \sqrt{\alpha_t \alpha_{t-1}} x_{t-2} + \sqrt{1 - \alpha_t \alpha_{t-1}} \epsilon_{t-2}$$

$$\vdots$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon$$

$$\boxed{q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)}$$

Forward process

We can obtain $q(x_t|x_0)$ in a closed form to directly sample x_t from x_0 for a timestep t.

$$\text{We define } \alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

$$\boxed{q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)}$$

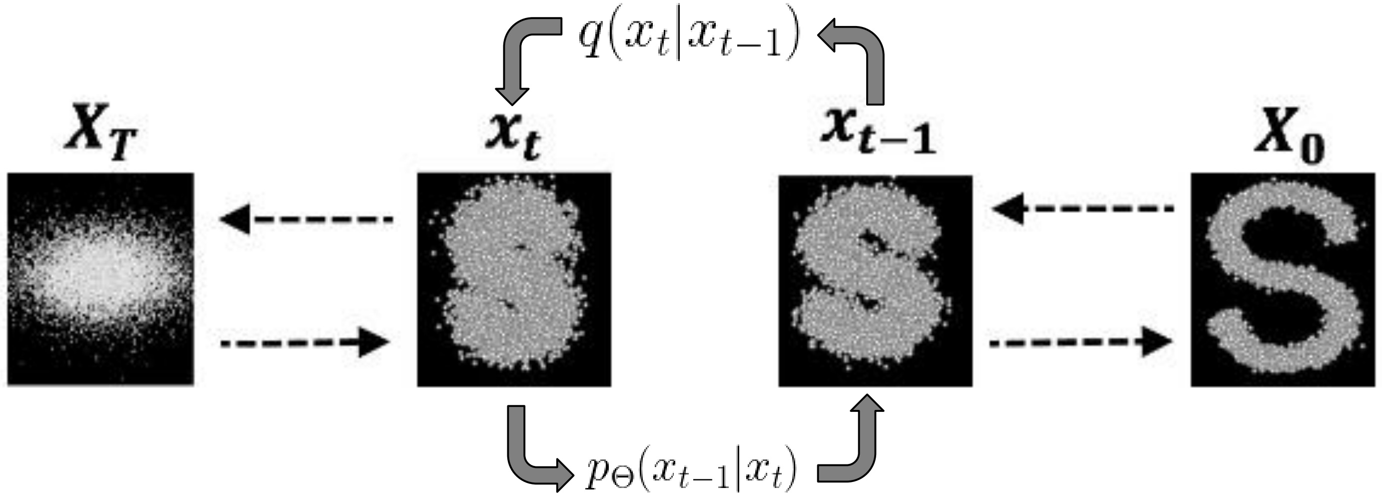
$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{(1 - \bar{\alpha}_t)} \epsilon$$

Part of the original image

Part of gaussian random noise

As *t* increases, the original image component decreases and the noise component increases.

Reverse process



$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Estimates the true reverse distribution $q(x_{t-1} \mid x_t)$ using a neural network parameterized by θ.

WHY?

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}$$

Intractable since it involves computing $q(x_t)$ and $q(x_{t-1})$

Calculating the Posterior Distribution

Fortunately, the posterior $q(x_{t-1}|x_t,x_0)$ is tractable and follows a gaussian distribution.

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

Since the forward process is markovian

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$$

Where,

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t)$$ and $$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Variational Lower Bound (Loss Function)

$$x_0 \sim \text{data} \Rightarrow p_\theta(x_0) \text{ must be high}$$
$$-\log p_\theta(x_0) \text{ must be minimized}$$

Can't be directly estimated!

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big]
\end{aligned}$$

$$\text{Let } L_{\mathrm{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)$$

Variational Lower Bound

Break down VLB into timesteps

$$
\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \boxed{\sum_{t=2}^T \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\Big)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\Big] \\
&= \mathbb{E}_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}
$$

Parameterization of L_t

$$L_t = D_{KL}(q(x_t|x_{t+1}, x_0) \,||\, p_\theta(x_t|x_{t+1})) \text{ for } 1 \le t \le T-1$$

$$\text{Goal is to learn } p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$\text{Recall that we had a tractable posterior } q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I)$$

$$\text{where } \tilde{\mu}_t(x_t, x_0) = \tfrac{1}{\sqrt{\alpha_t}}(x_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t)$$

$$\text{Similarly, we parameterize } \mu_\theta(x_t, t) = \tfrac{1}{\sqrt{\alpha_t}}(x_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t))$$

KL-divergence term reduces to an L2-norm between the difference of the true and predicted mean (assuming we fix the variance of the learned reverse distribution)

$$\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\Big] \\
&= \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]
\end{aligned}$$

Converting from L2 norm of mean difference to noise difference

Final Training Objective

$$L_t = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\Big[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]$$

The authors found that dropping the weighted term led to better sample quality which gives us the final objective:

$$\begin{aligned} L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big] \end{aligned}$$

What about variance of the reverse distribution?

The authors also fixed the the variance for $p(x_{t-1}|x_t)$ to β_t or $\tilde{\beta}_t$ and did not learn it in the diffusion process as it gave them little benefit.

Training and Sampling

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Sample x_0 from training set
Sample a random timestep t

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Start with some random noise
Predict and denoised sample at each step

[Algorithms taken from *Ho et al., 2020*.]

Resources

- Diffusion Models explanation by Outlier
- Lilian Weng’s blog on Diffusion Models
- CVPR tutorial on Diffusion Models

Diffusion Models for Text Generation

Challenges with Textual Data

- Text is discrete in nature unlike images that can be represented as continuous data.
- Most of the advancements in diffusion models are made for images (which are continuous). Hence, they are not directly applicable to text.
- It is difficult to define what noise means in terms of text. Incorrect structure of the sentence or randomly replaced words from a sentence can be considered as noise for text. But it is not so trivial to incorporate this in the diffusion process.

Classification of Text Diffusion Approaches

Text Diffusion

Discrete space

- Directly perturbs the tokens with a categorical distribution
- More interpretable, but difficult to design

Embedding space

- Maps tokens to continuous embedding space and uses continuous diffusion techniques
- Less intuitive, but enables incorporation of existing techniques on image diffusion

Discrete Diffusion Models

Forward Process

- Consider a vocabulary of tokens of size *V* and a sentence *X* where each word is a token from the vocabulary.

- The diffusion process at a timestep *t* for a single token $x \in X$ is defined as:

$$q(x_t|x_{t-1}) = \mathcal{C}\left(x_t|(1-\beta_t)x_{t-1} + \frac{\beta_t}{V}\right)$$

The token remains the same with probability $(1-\beta_t)$

Token is replaced by a random token from the vocabulary with probability β_t

- This is applied to all the tokens in the sentence simultaneously until the sentence becomes a completely random set of words.

Discrete Diffusion Models (Example)

Vocabulary = {a, an, the, He, likes, eat, to, sleep, is, was, hates, has}
Sentence = He likes to eat.

t = 0	He	likes	to	eat
	↓	↓	↓	↓
t = 1	He	was	to	eat
	Unchanged with probability $1\text{-}\beta_1$	Randomly replaced with probability β_1	Unchanged with probability $1\text{-}\beta_1$	Unchanged with probability $1\text{-}\beta_1$

This goes on for *T* steps after which each token in the sentence looks like it is uniform randomly sampled from the vocabulary.

Re-designing the Training Objective

For each token x_t at timestep t, the true posterior is derived as:

Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$ (Similar to DDPMs)

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})\, q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{C}(x_{t-1}|\theta_{post}(x_t, x_0)), \text{ where } \theta_{post}(x_t, x_0) = \frac{\tilde{\theta}}{\sum_{v=1}^{V} \tilde{\theta}_v}$$

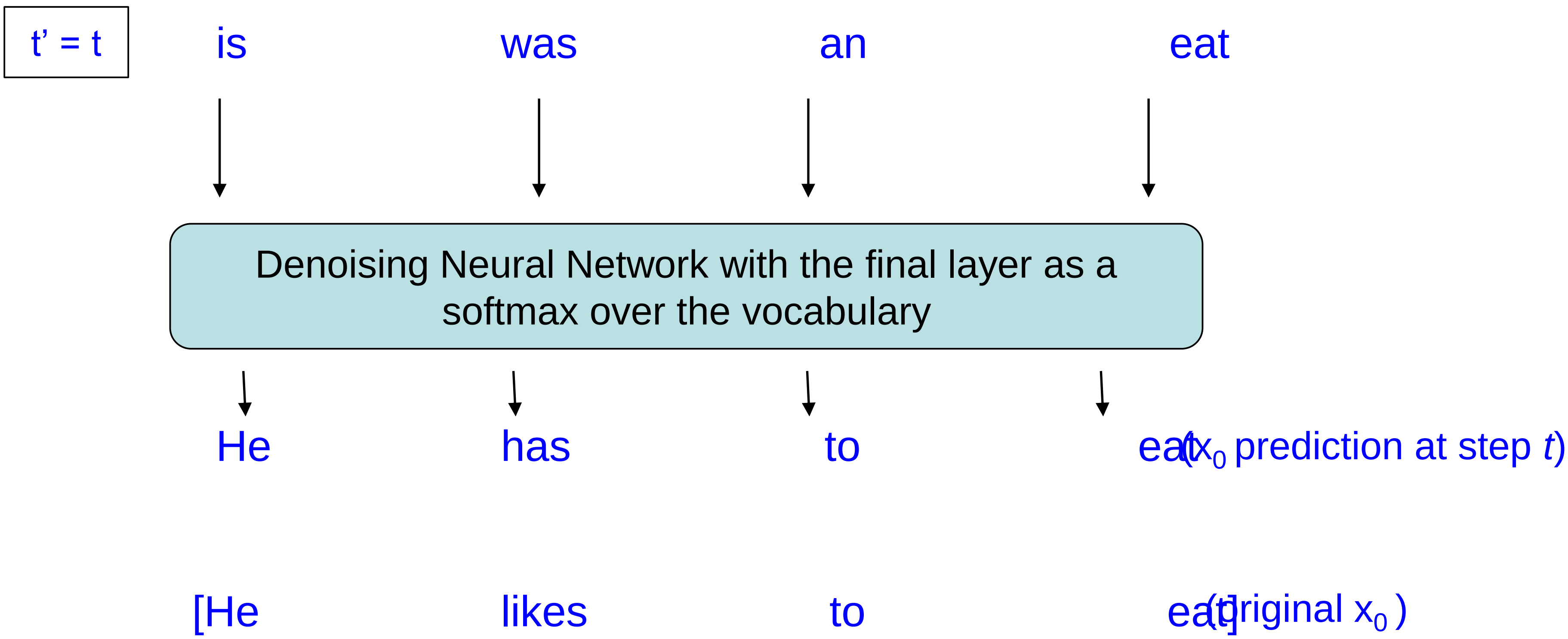
$$\tilde{\theta} = \left[\alpha_t x_t + \frac{1-\alpha_t}{V}\right] \odot \left[\bar{\alpha}_t x_0 + \frac{1-\bar{\alpha}_{t-1}}{V}\right]$$

where θ_{post} consists of probability values for each token in the vocabulary (representing a categorical distribution)

Problem: Difficult to define the notion of noise since we are individually perturbing discrete tokens.

Re-designing the Training Objective

Hoogeboom et al. 2021 proposes to bypass the noise calculation by directly predicting the original tokens at each step.



How to compare these two?

Re-designing the Training Objective

Using the original x_0 and the predicted x_0, we first obtain the true posterior and the predicted posterior.

These two are compared using KL-divergence (thus giving us a training objective)

$$D_{\mathbf{KL}}(q(x_{t-1}|x_t, x_0)\,||\,p_\theta(x_{t-1}|x_t)) = D_{\mathbf{KL}}(\mathcal{C}(x_{t-1}|\theta_{post}(x_t, x_0))|\mathcal{C}(x_{t-1}|\theta_{post}(x_t, \hat{x_0})))$$

True posterior

Predicted posterior where x_0^{hat} is the predicted x_0

Generalizing Discrete Diffusion Models

- The discrete perturbations can also be represented in terms of transition matrices.
- Transition matrix defines the probability of transitioning from one token to another at each step.

Forward process in terms of transition matrices:

$$q(x_t|x_{t-1}) = \mathcal{C}(x_t|p = x_{t-1}Q_t) \quad \text{where} \quad [Q_t]_{ij} = q(x_t = j|x_{t-1} = i).$$

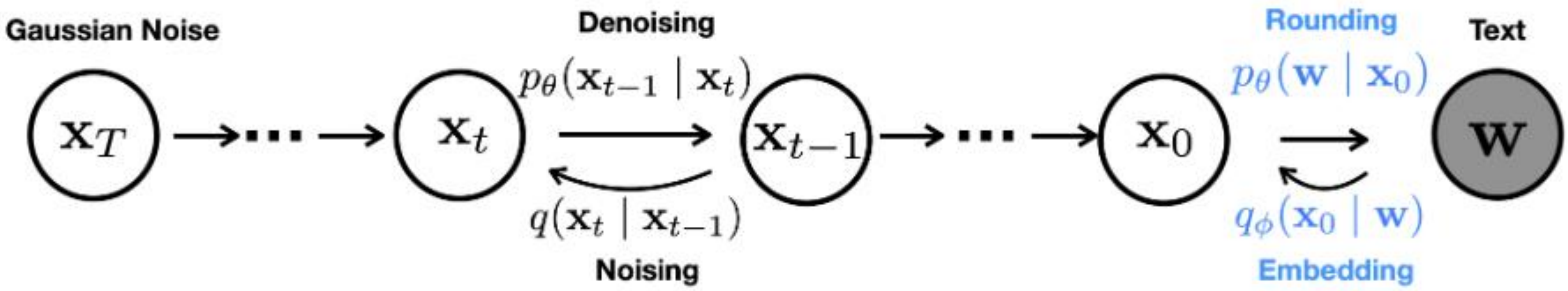
$$q(x_t|x_0) = \mathcal{C}(x_t|p = x_0\bar{Q}_t) \quad \text{where} \quad \bar{Q}_t = Q_1Q_2\cdots Q_t$$

The true reverse distribution can now be written as:

$$q(x_{t-1}|x_t, x_0) = \mathcal{C}(x_{t-1}|p = \frac{x_tQ_t^\top \odot x_0\bar{Q}_{t-1}}{x_0\bar{Q}_tx_t^\top})$$

Embedding Diffusion Models

Forward Process



- **Embedding step:** Maps discrete tokens to a continuous embedding space.

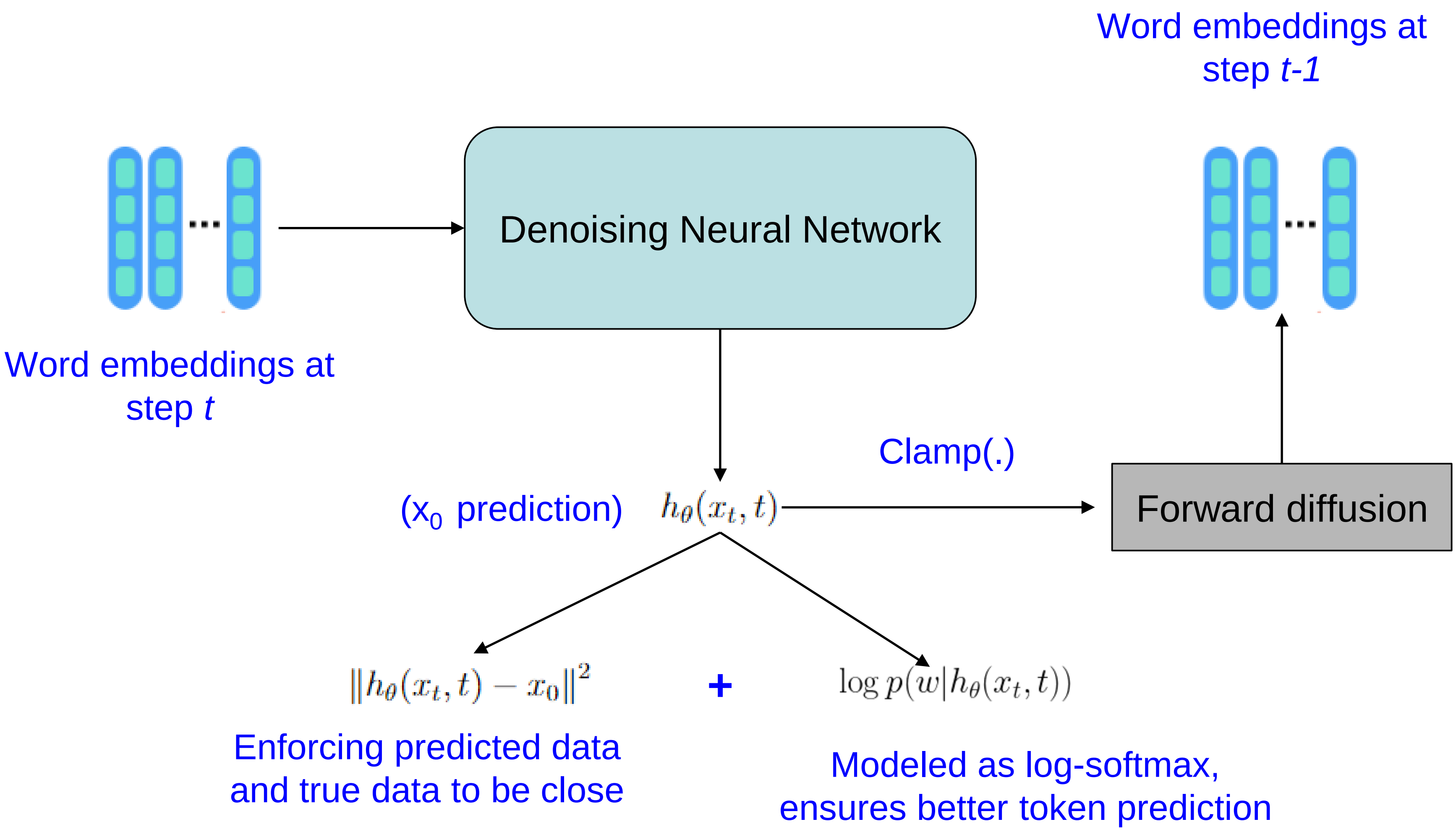
- **Rounding step:** Maps the output vectors from the diffusion process to the nearest token.

[Figure taken from Li. et al., 2022]

Jointly Learning the Embeddings with Diffusion Process

- Embedding step only occurs at the beginning of the diffusion.
- Not enough training signals considering we sample a random timestep during training!
- **Solution:** force the denoising model to commit to a token at each step. This also reduces rounding step errors.
- During training, this can be achieved by enforcing the predicted embeddings and true embeddings to be similar by a distance measure.
- During sampling, we simply find the nearest token embedding and round off the output to it (also called the clamping trick).

Illustration of Embedding Diffusion Models



Summary

- In this presentation, we give an overview of diffusion models in image generation and how it has been extended for textual data.
- We begin with putting forth the notion of generative modeling, briefly trace back into solutions before diffusion, and look at their drawbacks.
- Next, we describe the general formulation of diffusion models for images in detail. We also mention some improvement techniques and faster sampling methods.
- We mentioned some challenges that would occur when adapting diffusion models for textual data
- In text diffusion, we described two approaches that can be taken: discrete diffusion and embedding diffusion.
- Finally, we briefly discuss the current state of diffusion models compared to autoregressive language models.

Conclusion

- Diffusion models have seen excellent growth in image generation and are currently studied quite extensively.
- However, diffusion models for text is fairly underexplored due to the advent of Language Models (LLMs) that are quite capable in generating fluent text.
- Adapting diffusion models for text is a non-trivial task. One reason for this is that it doesn’t look like a natural framework for generating text as text is usually an ordered sequence.
- Even so, the benefits of flexibility and high output diversity are very desirable in the textual domain even when we already have excellent text generators in place.

Thank you

References

- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 574, 6840–6851.
- Nichol, A.Q. and Dhariwal, P.. (2021). Improved Denoising Diffusion Probabilistic Models. *Proceedings of the 38th International Conference on Machine Learning, in Proceedings of Machine Learning Research* 139:8162-8171
- Hoogeboom, Emiel, Didrik Nielsen, Priyank Jaini, Patrick Forr'e and Max Welling. “Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions.” *Neural Information Processing Systems (2021).*
- Austin, Jacob, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow and Rianne van den Berg. “Structured Denoising Diffusion Models in Discrete State-Spaces.” ArXiv abs/2107.03006 (2021): n. pag.
- Li, Xiang Lisa, John Thickstun, Ishaan Gulrajani, Percy Liang and Tatsunori Hashimoto. “Diffusion-LM Improves Controllable Text Generation.” ArXiv abs/2205.14217 (2022): n. Pag.
- Song, Jiaming, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." arXiv preprint arXiv:2010.02502 (2020).