Deep Learning & LLMs

Text Mining, Transforming Text into Knowledge

Ayoub Bagheri

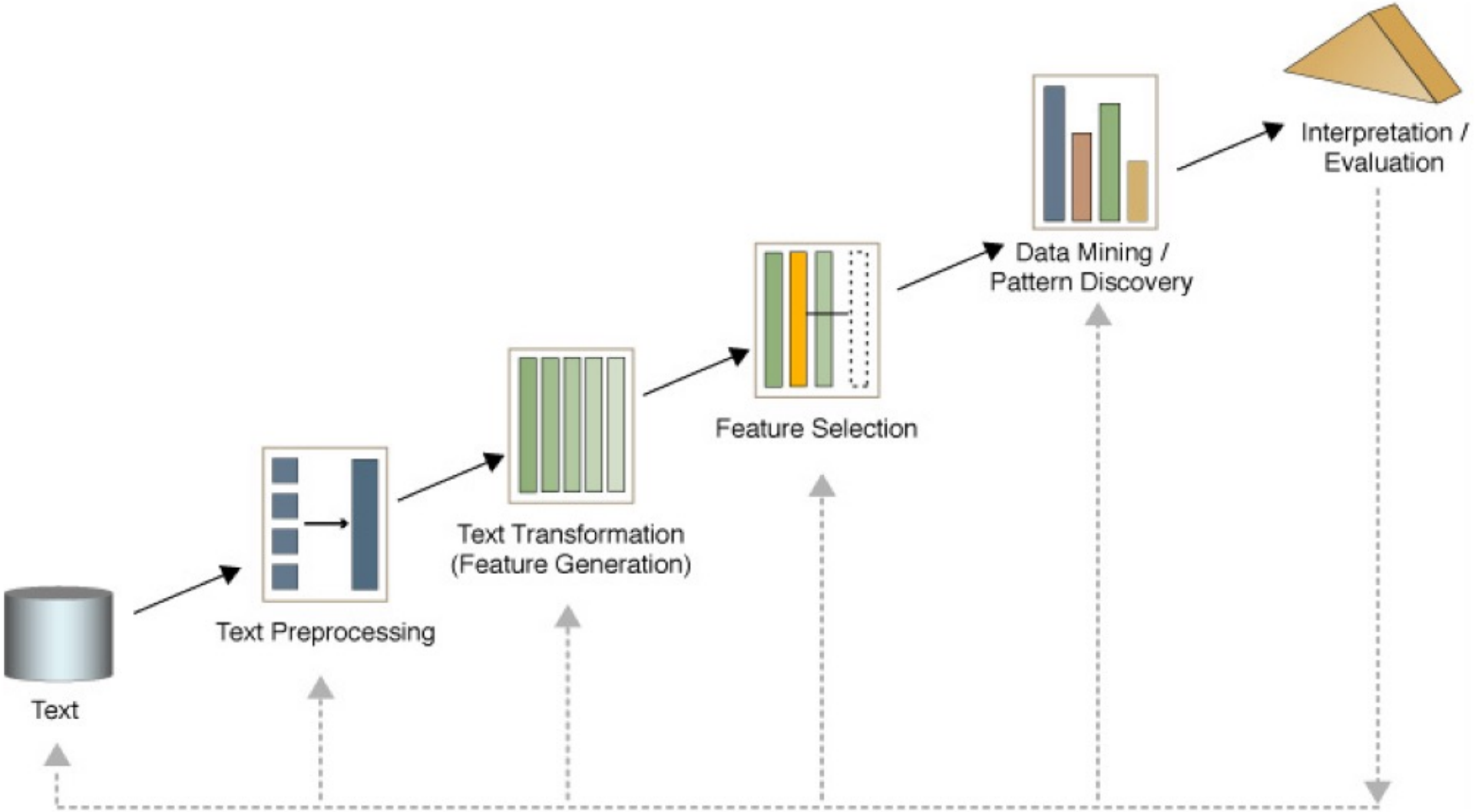
Last week

- Text representation
- Word embedding
  - Skipgram learning
  - Pre-trained embeddings

Today

- Introduction to neural networks
- Feed-forward & deep neural networks
- State-of-the-art methods

Text mining process



Text mining process

- **Data: Text**
- **Text Preprocessing:** is the process of cleaning, normalizing, and structuring raw text data into a format suitable for analysis or input into NLP models. (week 2)
- **Text transformation, feature generation:** involves converting text data into a different format or structure, such as numerical vectors or simplified forms, to make it suitable for analysis or modeling. (weeks 1, 2, 3, 6, 7, 8)
- **Feature selection**: is the process of identifying and selecting the most relevant features from a dataset to improve model performance and reduce complexity. (week 4)
- **Data mining, pattern discovery**: is the process of extracting meaningful patterns and knowledge from text. (weeks 3, 5, 7, 8, 9)
- **Interpretation / Evaluation:** is the process of understanding and explaining the model and patterns / is the assessment process to measure performance and quality. (weeks 3-9)

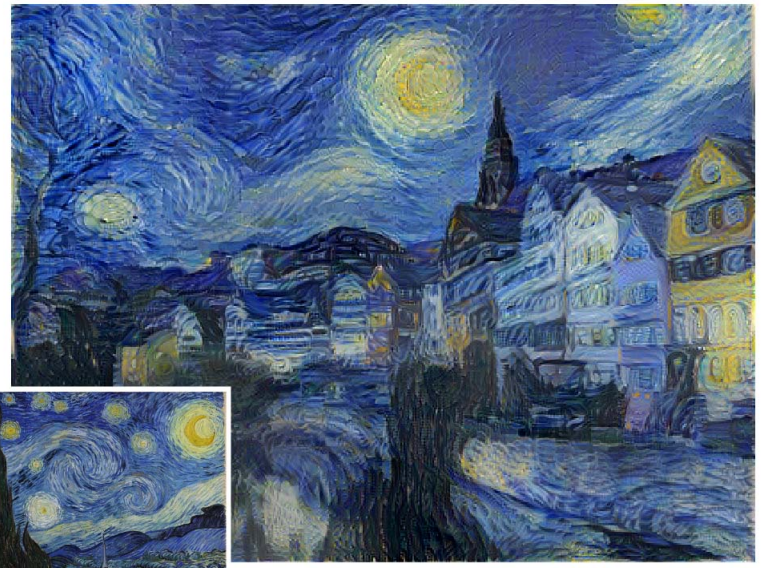
Introduction

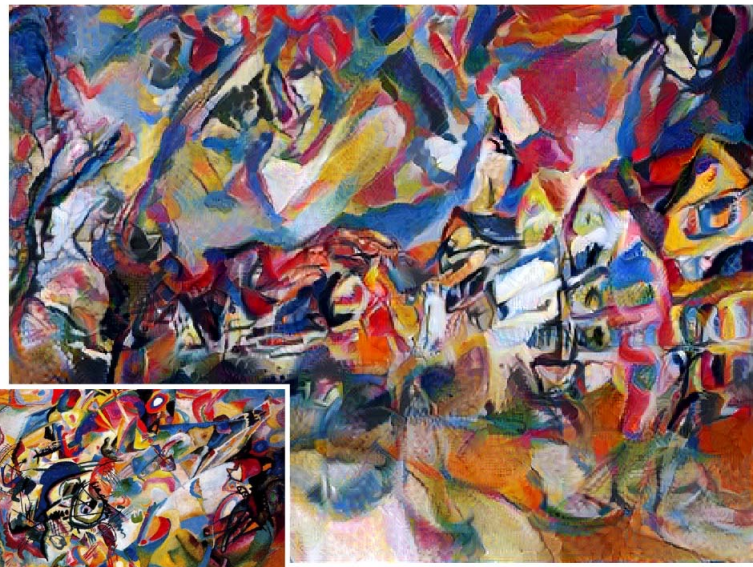
Why should we learn this?

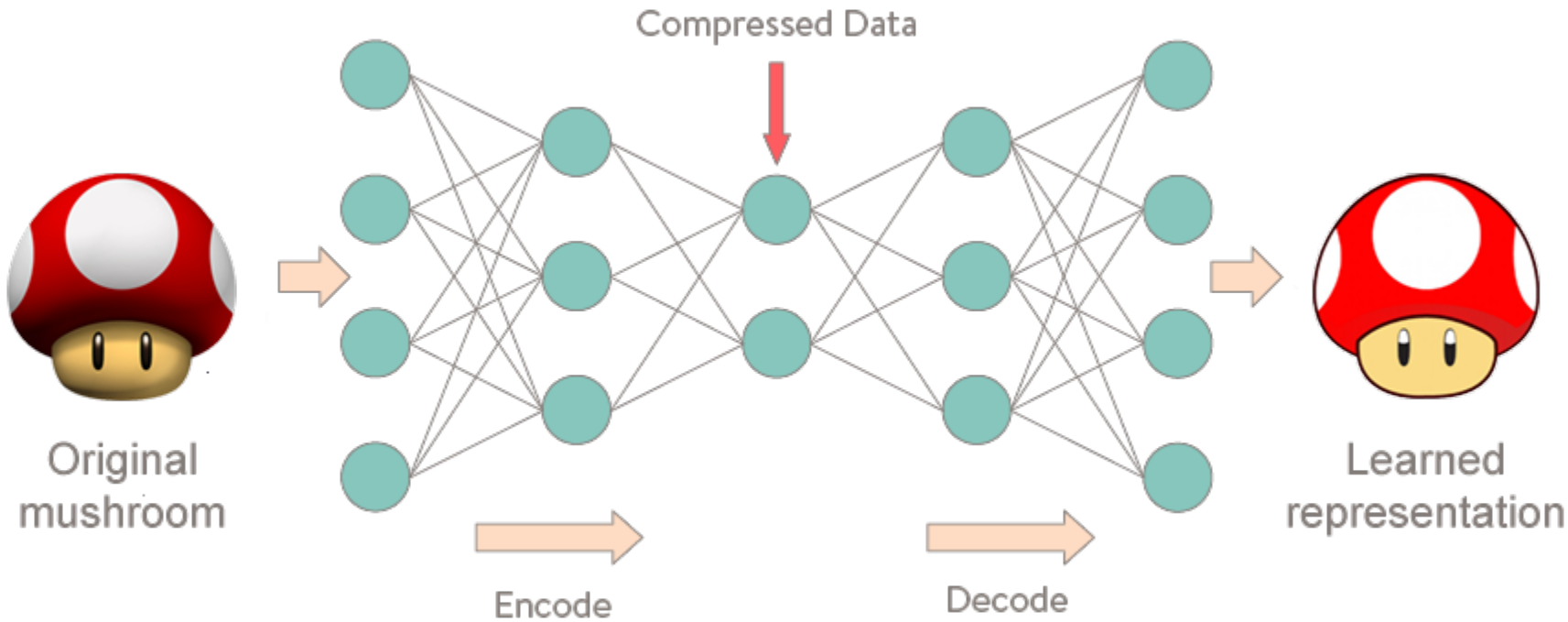
State-of-the-art performance on various tasks

- Text prediction (your phone’s keyboard)
- Text mining
- Forecasting
- Spam filtering
- Compression (dimension reduction)
- Text generation
- Translation
- …

https://thispersondoesnotexist.com/



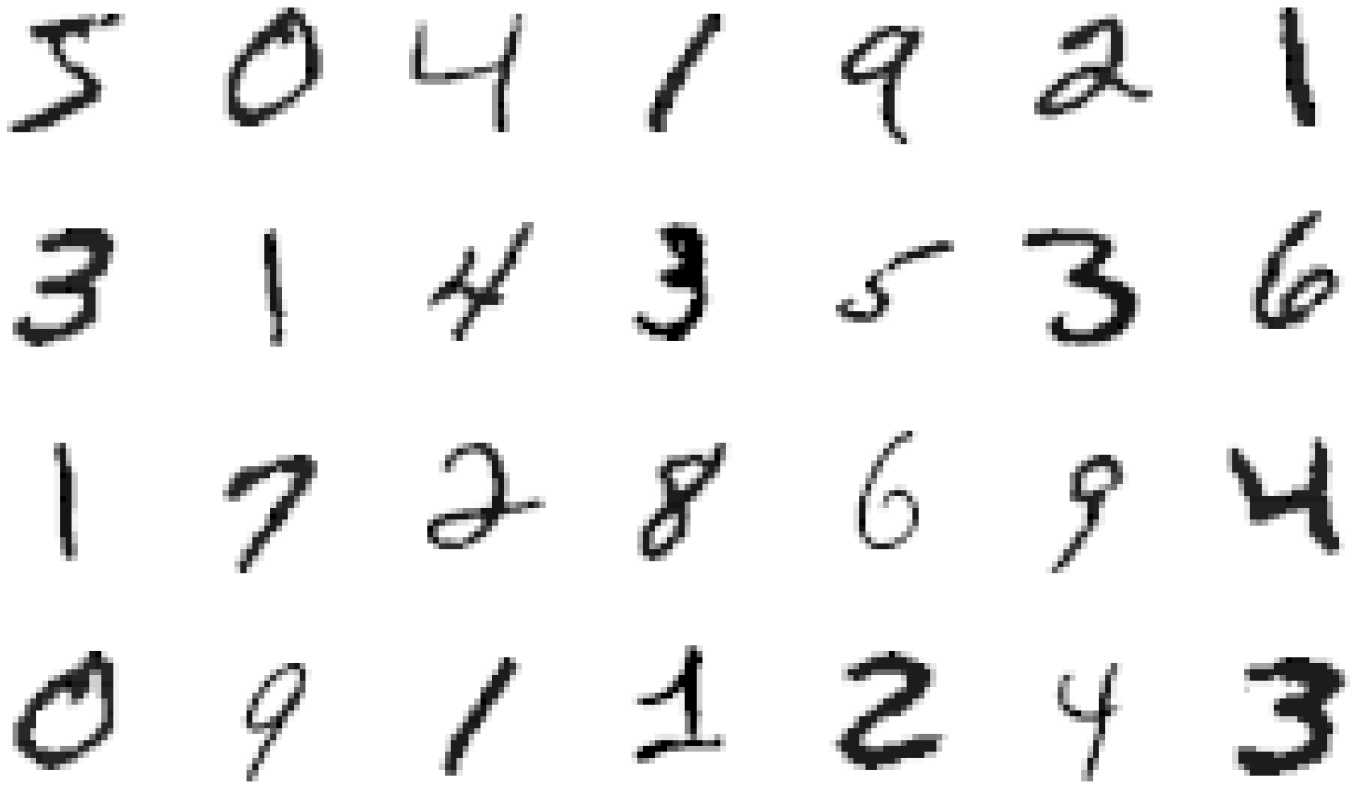




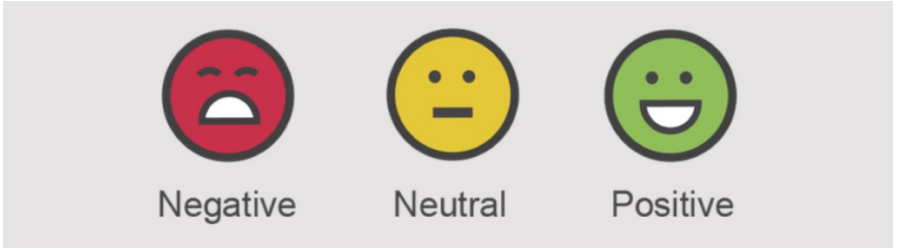
https://community.canvaslms.com/t5/Canvas-Developers-Group/Canvas-LMS-Cheat-Detection-System-In-Python/m-p/118134

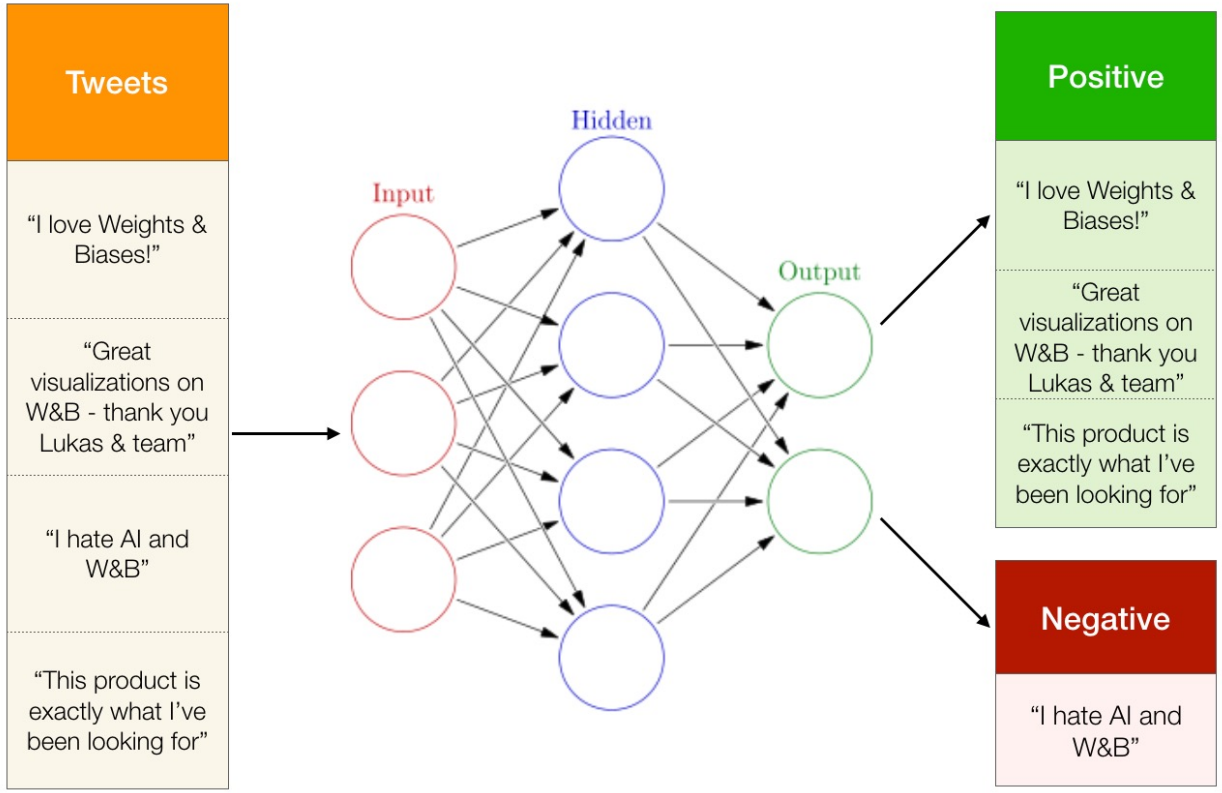
“Hello world” of neural networks

- MNIST (Modified National Institute of Standards and Technology)
- Handwritten digits
- 28 * 28 pixels
- 60 000 training images and 10 000 testing images



“Hello world” of neural networks for text: Sentiment classification with LSTM





So what is a neural network?

Neural networks

$$y = f(X) + \epsilon$$

- Neural networks are a way to specify $f(X)$
- You can display $f(X)$ graphically

- Let's graphically represent linear regression!

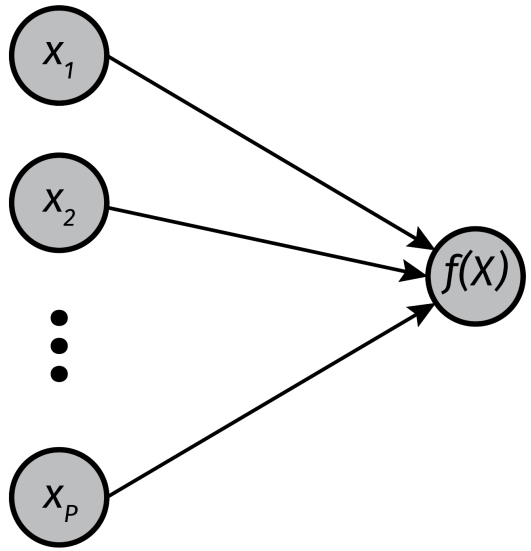
$$f(X_i) = \sum_{p=1}^{P} \beta_p x_{pi}$$

Linear regression as neural net

Graphical representation

- Parameters are arrows
- Arrows ending in a node are summed together
- Intercept is not drawn

$$f(X_i) = \alpha + \sum_{p=1}^{P} \beta_p x_{pi}$$

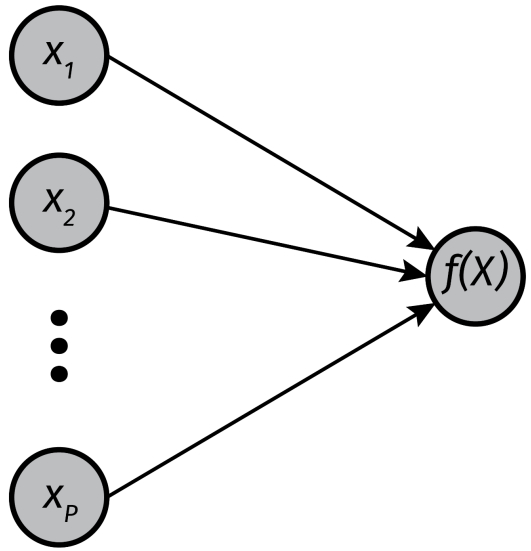


Linear regression as neural net

Neural network jargon

- Parameter = **weight**
- Intercept = **bias**

$$f(X_i) = \boldsymbol{\beta} + \sum_{p=1}^{P} \boldsymbol{w}_p x_{pi}$$



Single layer neural networks

$$y = f(X) + \epsilon$$

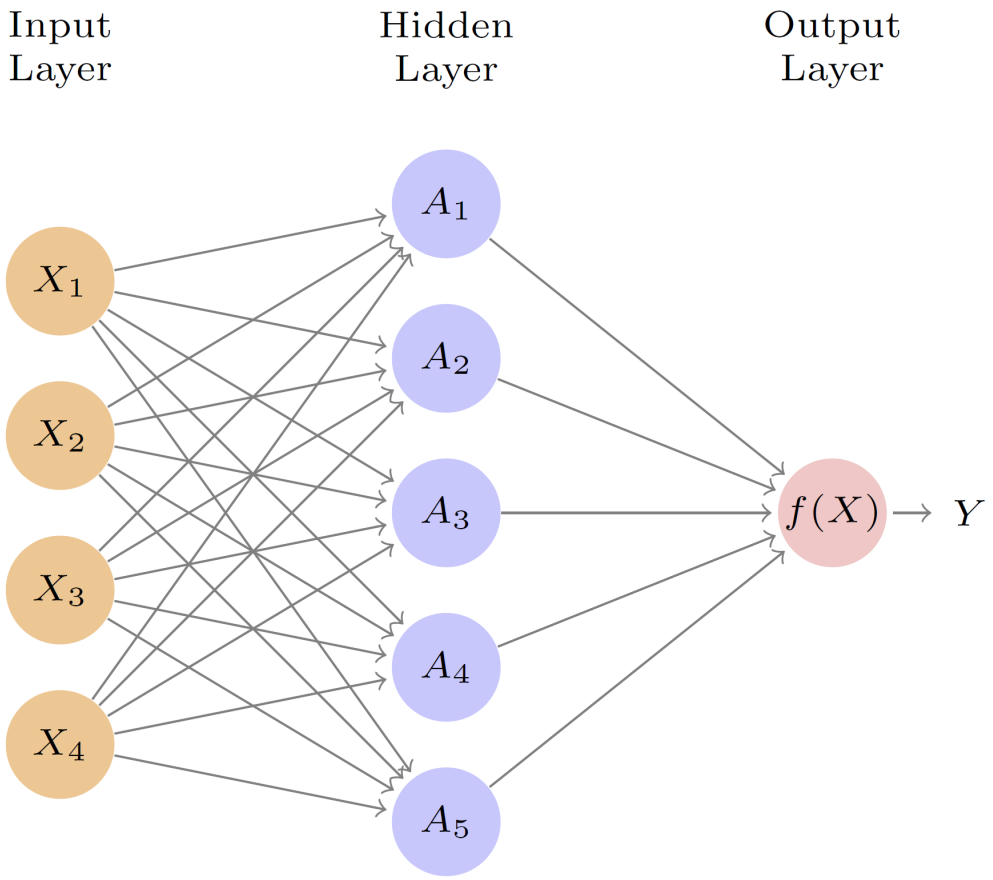
Specify a layer with K *hidden units* called A

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k A_k$$

Where

$$A_k = h_k(X) = g\left(w_{0k} + \sum_{p=1}^{P} w_{pk} x_p\right)$$

Single layer neural networks

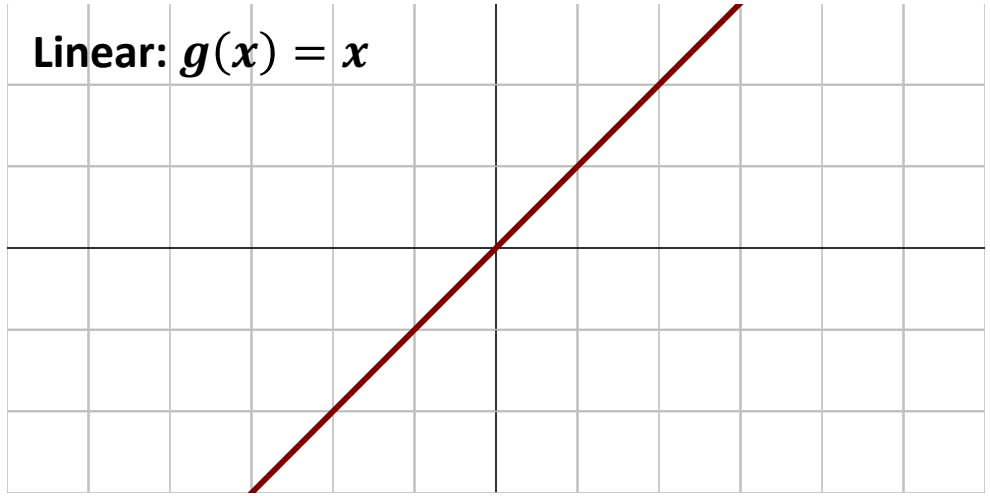


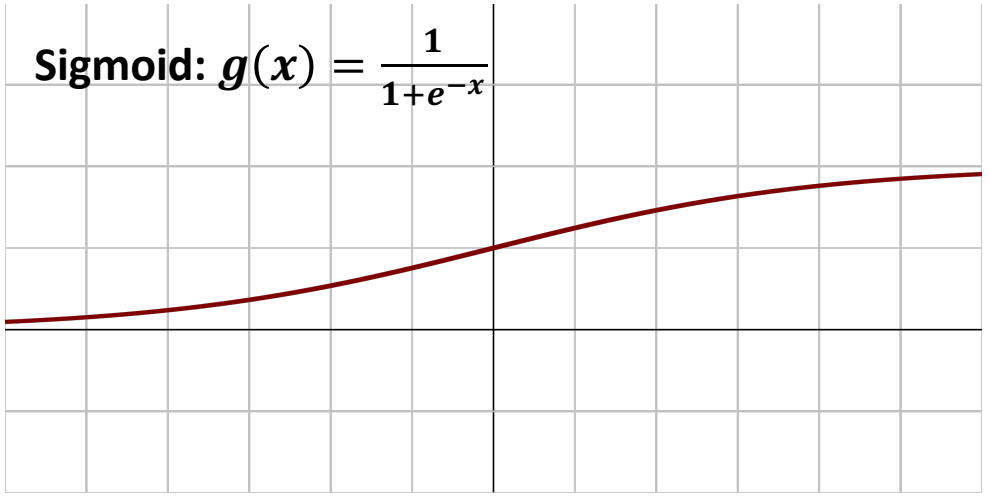
Single layer neural networks

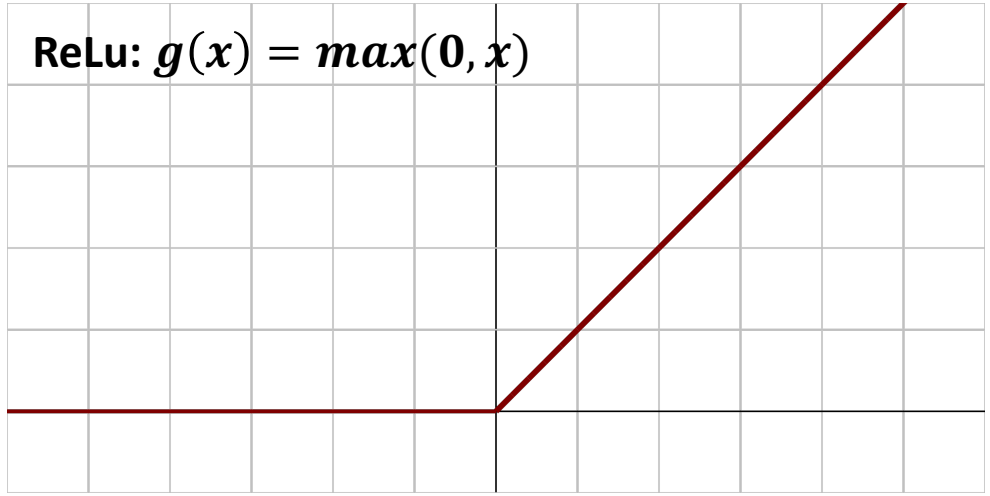
- What about the function $g(\cdot)$?
- This is called the **activation function**
- A transformation of the linear combination of predictors

$$h_k(X) = g\left(w_{0k} + \sum_{p=1}^{P} w_{pk} x_p\right)$$

Activation functions

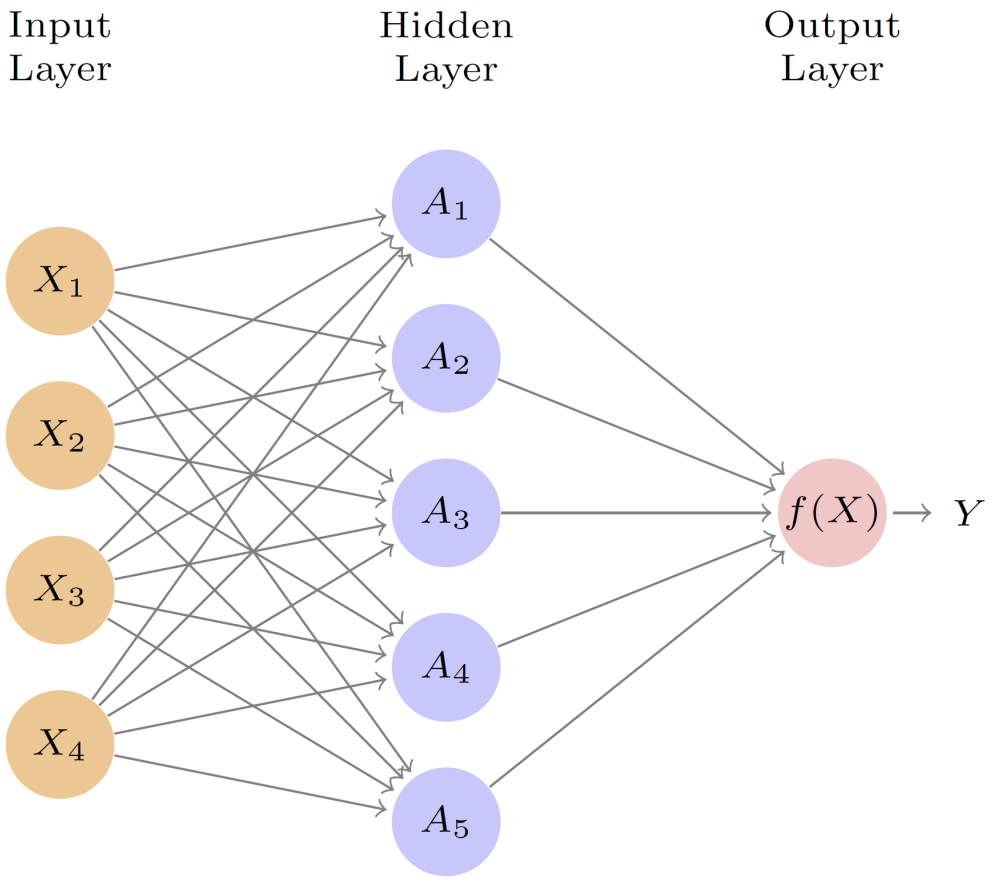






- Rectified linear (ReLu) is most popular nowadays
- Nonlinearity necessary! Otherwise: collapse to linear regression

Single layer neural networks



Feed-forward Neural Networks

Feed-forward neural networks

We can go deeper

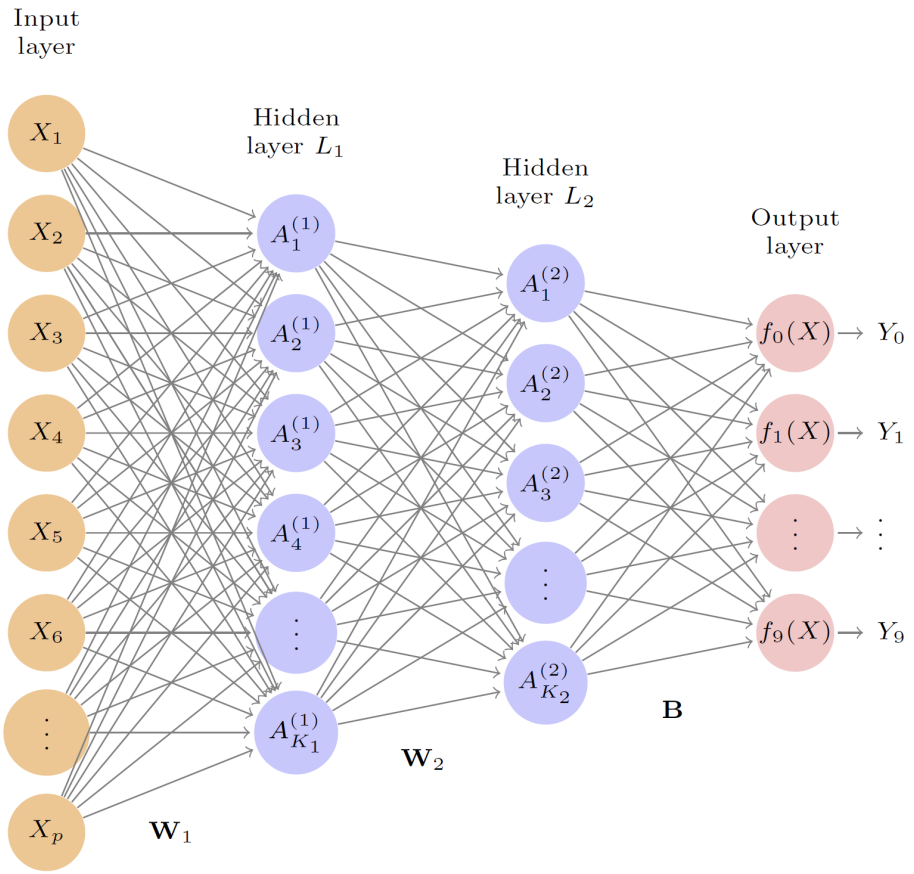
- More hidden layers after one another
- Higher-order features composed of lower-order features

Universal function approximation theorem, version 2

Any “well-behaved” function can be represented by neural net of sufficient *depth* with nonlinear activation function

(deep neural nets may be more tractable than wide)

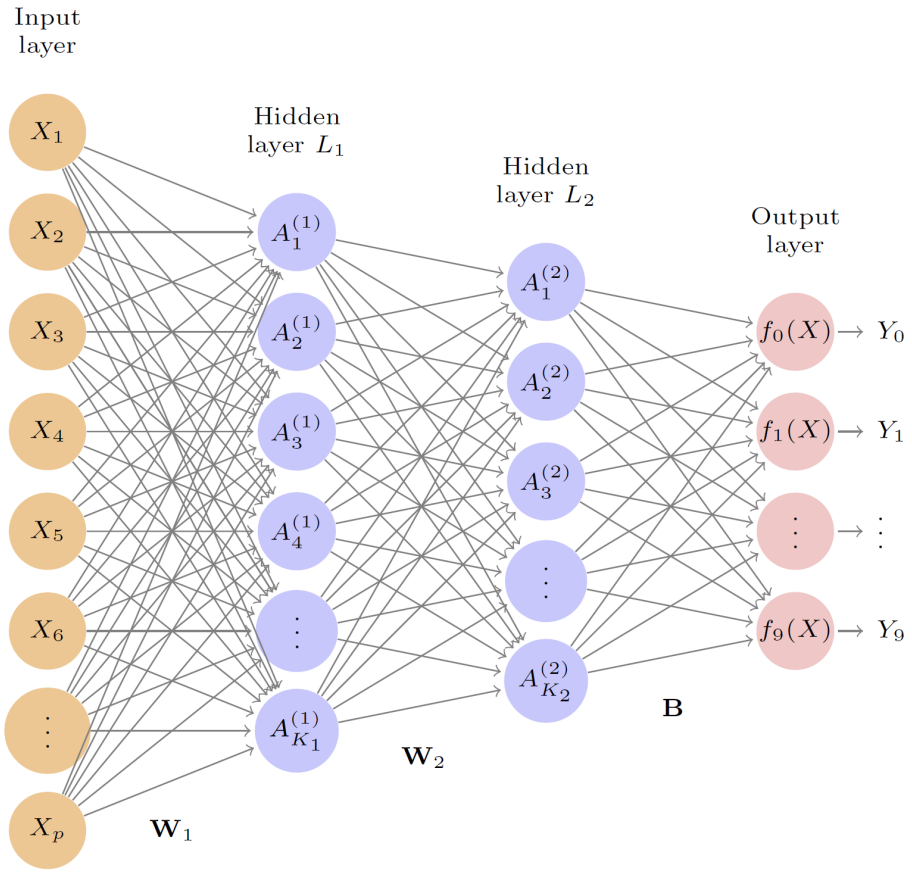
Feed-forward neural networks



Feed-forward neural networks

Feed-forward network architecture defined by:

- Number of layers
- Number of hidden units in each layer
- Activation function for each layer
- Activation function for output layer



Keras!

```
library(keras)


model_dff <-
  keras_model_sequential() %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 256, activation = "relu") %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(10, activation = "softmax")
```

Keras!

```
summary(model_dff)

Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               200960
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
=================================================================
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0
_________________________________________________________________
```

How to estimate parameters?

Estimating parameters

- We need some way to measure how well the network does
- Parameters that make the network perform well are good!

Loss function

- For continuous outcomes you can use squared error (same as linear regression!)

$$L(\theta) = (f(X_i; \theta) - y_i)^2$$

- For binary outcomes you can use binary cross-entropy (same as logistic regression!)

$$L(\theta) = -\big(y_i \log\big(f(X_i; \theta)\big) + (1 - y_i) \log\big(f(X_i; \theta)\big)\big)$$

Gradient descent

Iteration: step of size λ in the direction of the negative gradient

$$\theta^{(j+1)} = \theta^{(j)} - \lambda \cdot g\left(\theta^{(j)}\right)$$

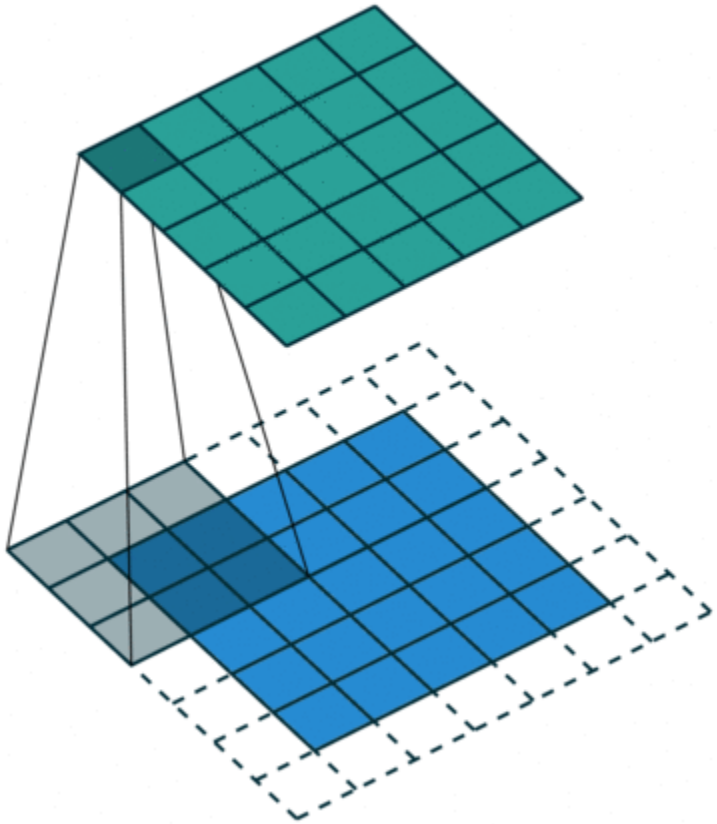
- But in neural networks, how do we compute gradients?
- We have functions of functions!
- Software like tensorflow / Keras / torch does this for you!
- **Backpropagation**: smart repeated use of the *chain rule* to compute derivatives

Convolutional Neural Networks

What is a convolution

- Convolution is applying a **kernel** (filter) over data (text, image, etc.)
- The kernel (filter) defines which **feature** is important in the data

What is a convolution

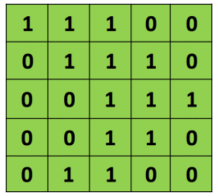


https://github.com/vdumoulin/conv_arithmetic

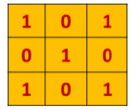
What is a convolution

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

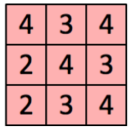
4		



5x5 input.



3x3 filter/kernel/feature detector.



3x3 convolved feature/
activation map/feature map

Convolution layers

- A convolutional neural network is a NN with one or more **convolution layers**
- The parameters / weights in a convolution layer are the elements of the filter
- The filter is **learnt** by the network!

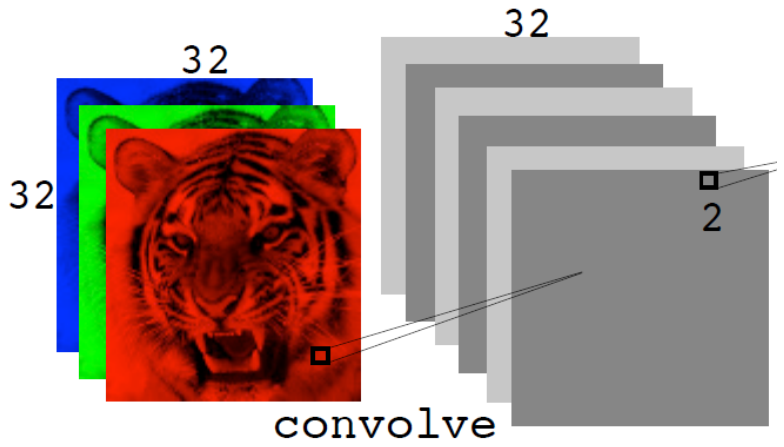


FIGURE 10.8. *Architectu*
Convolution layers are inte
size by a factor of 2 in both

Pooling layer

- Convolution layers are usually followed by a **pooling layer**
- Reduces dimensionality
- **Location invariance**: Robustness against pixel shift / small rotations
- **Max pool** most common

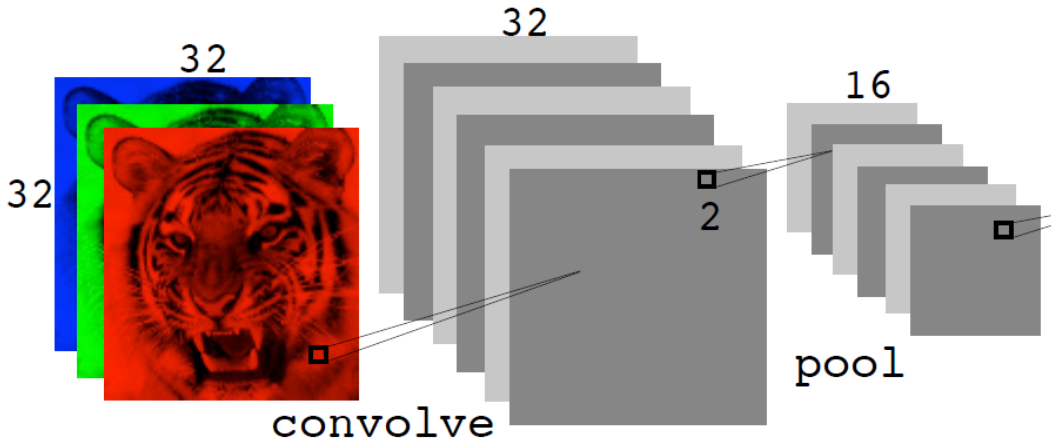


FIGURE 10.8. *Architecture of a de*
Convolution layers are interspersed u
size by a factor of 2 in both dimensio

Pooling layer

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}.$$

Architecture of a CNN

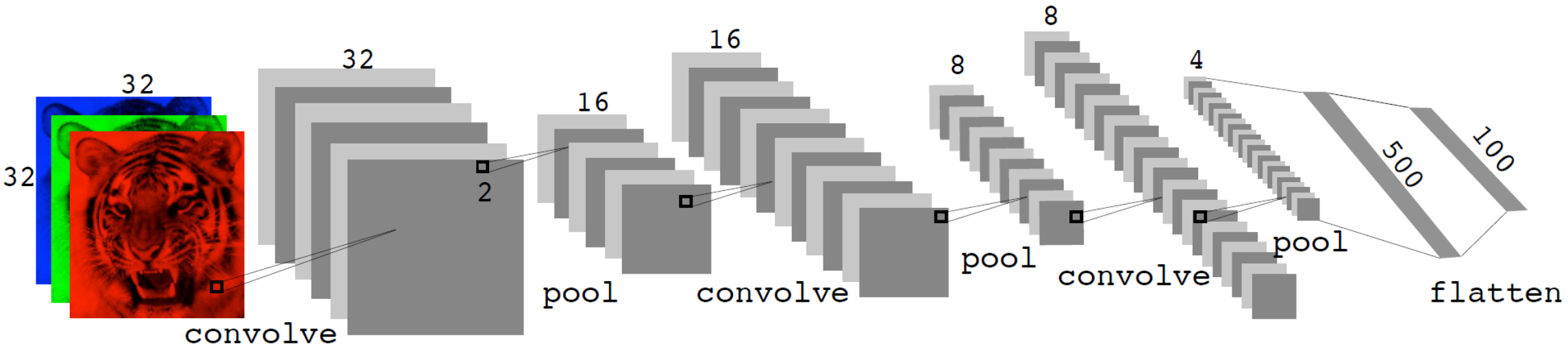


FIGURE 10.8. *Architecture of a deep CNN for the* `CIFAR100` *classification task. Convolution layers are interspersed with* 2×2 *max-pool layers, which reduce the size by a factor of 2 in both dimensions.*

Break

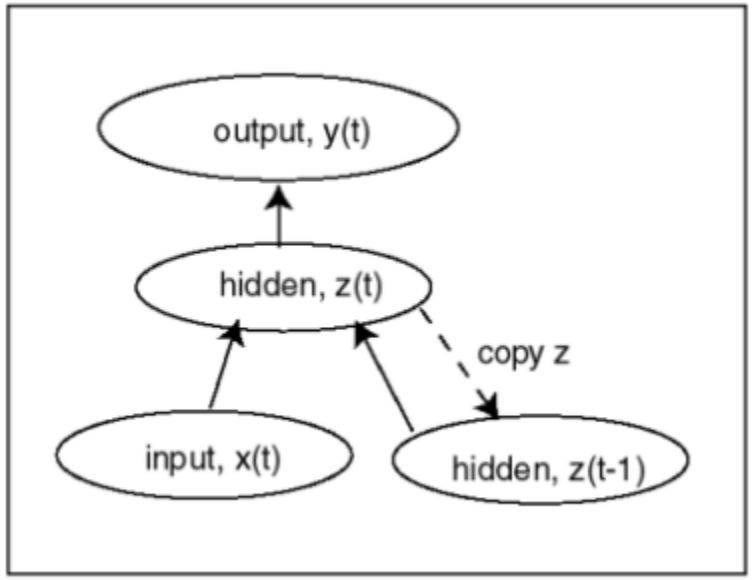
Recurrent Neural Network (RNN)

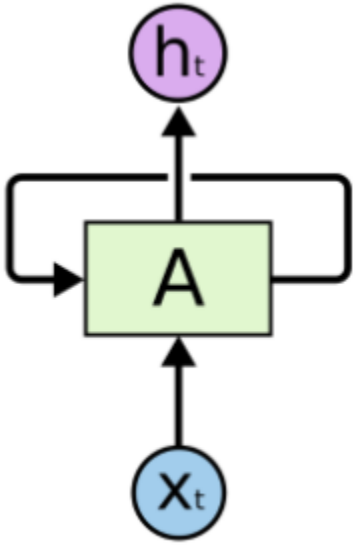
Recurrent Neural Network

- Another famous architecture of **Deep Learning**
- Preferred algorithm for sequential data
  - time series, speech, **text**, financial data, audio, video, weather and much more.
  - **text**: sentiment analysis, sequence labeling, speech tagging, machine translation, etc.

- Maintains **internal memory**, thus can remember its previous inputs

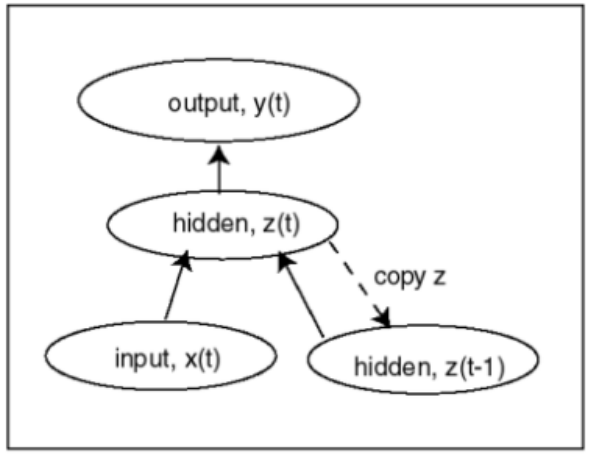
Simple recurrent network

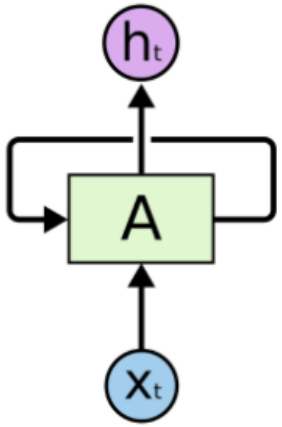




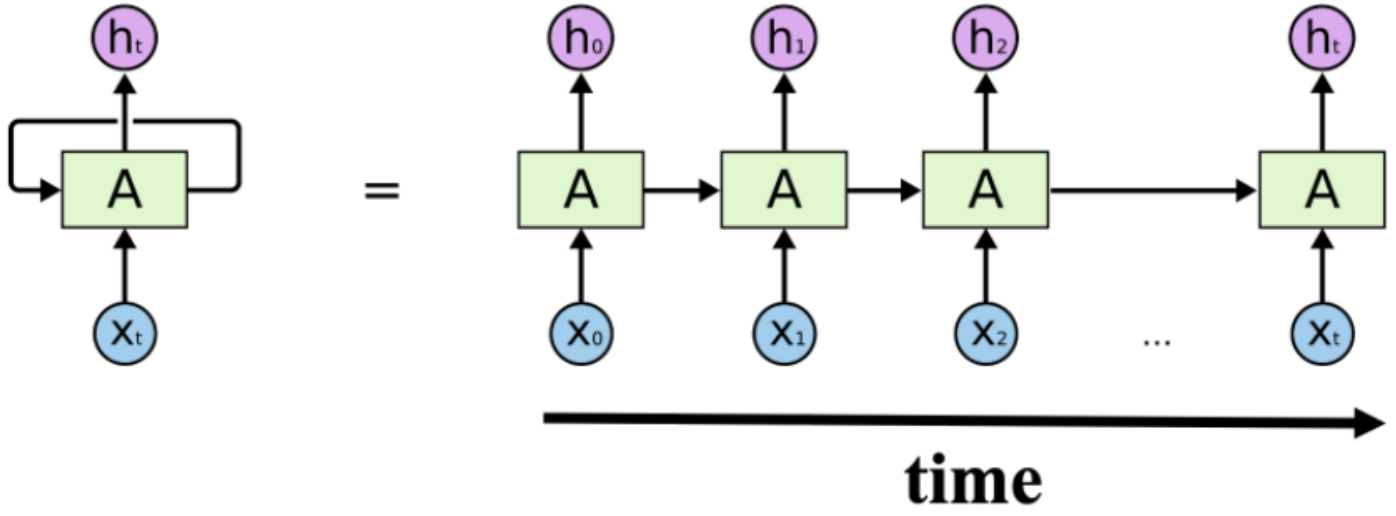
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Simple recurrent network



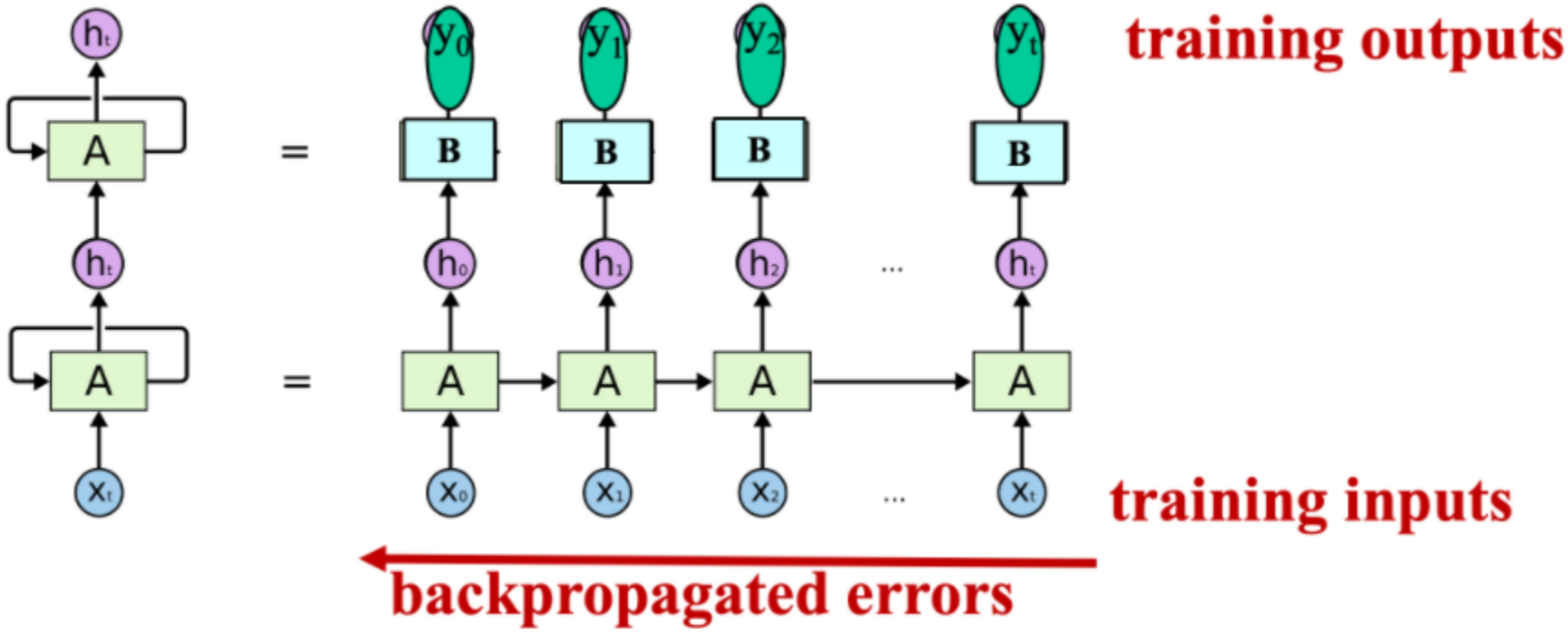


http://colah.github.io/posts/2015-08-Understanding-LSTMs/



Training RNNs

- RNNs can be trained using “backpropagation through time.”
- Can viewed as applying normal backprop to the unrolled network.



The problem of Vanishing Gradient

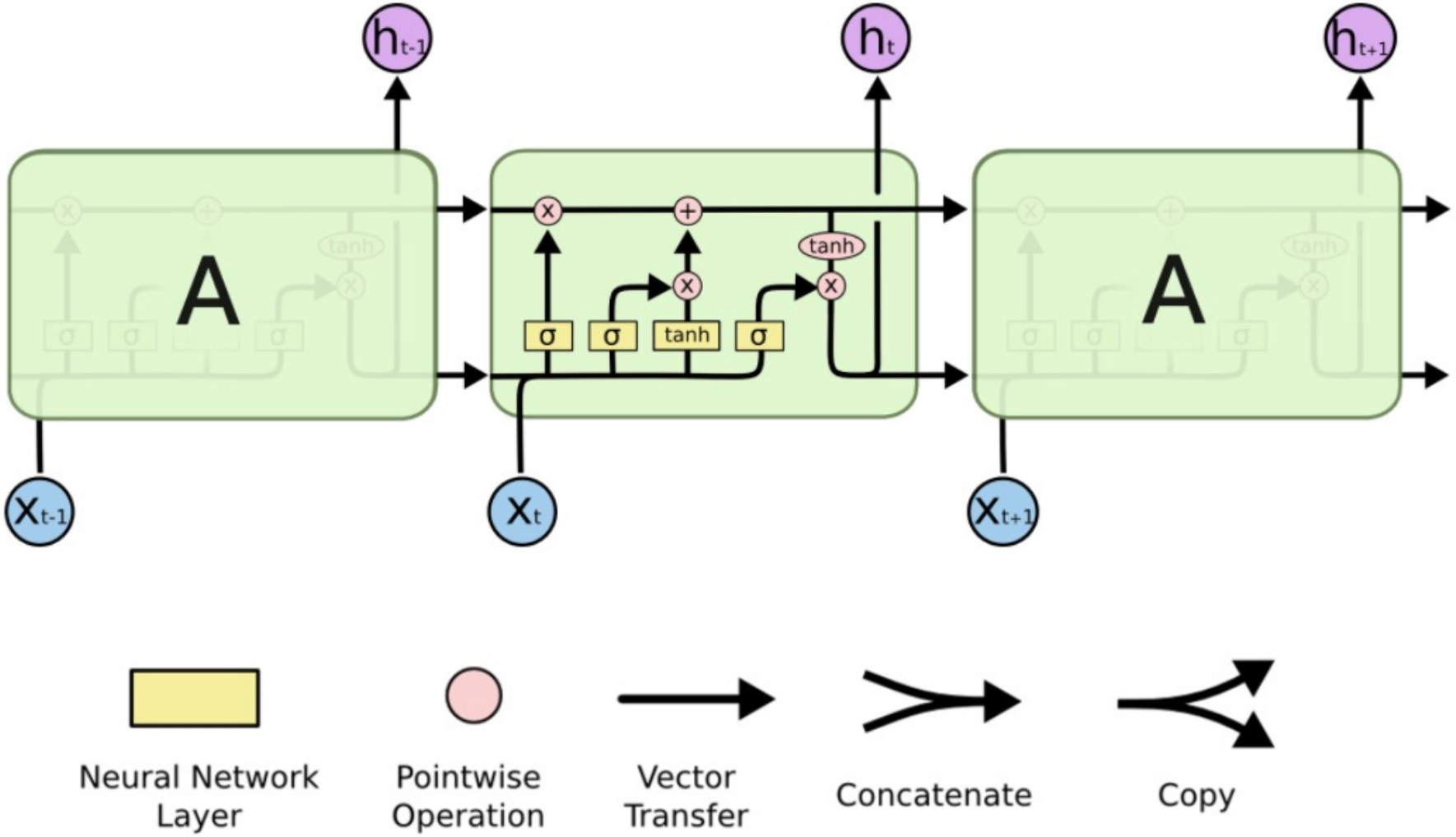
- Consider a **RNN** model for a **machine translation** task from English to Dutch.
- It has to read an English sentence**, store as much information as possible** in its hidden activations, and output a Dutch sentence.
- The information about the first word in the sentence doesn't get used in the predictions until it starts generating Dutch words.
- There's **a long temporal gap** from when it sees an input to when it uses that to make a prediction.
- It can be hard to learn **long-distance dependencies**.
- In order to adjust the input-to-hidden weights based on the first input, the error signal needs to travel backwards through this entire pathway.

Long Short-Term Memory (LSTM)

Long Short-Term Memory

- Prevents vanishing/exploding gradient problem by:
  - introducing a gating mechanism
  - turning multiplication into addition
- Designed to make it easy to remember information over long time periods until it's needed.
- The activations of a network correspond to short-term memory, while the weights correspond to long-term memory.

LSTM architecture



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Extensions

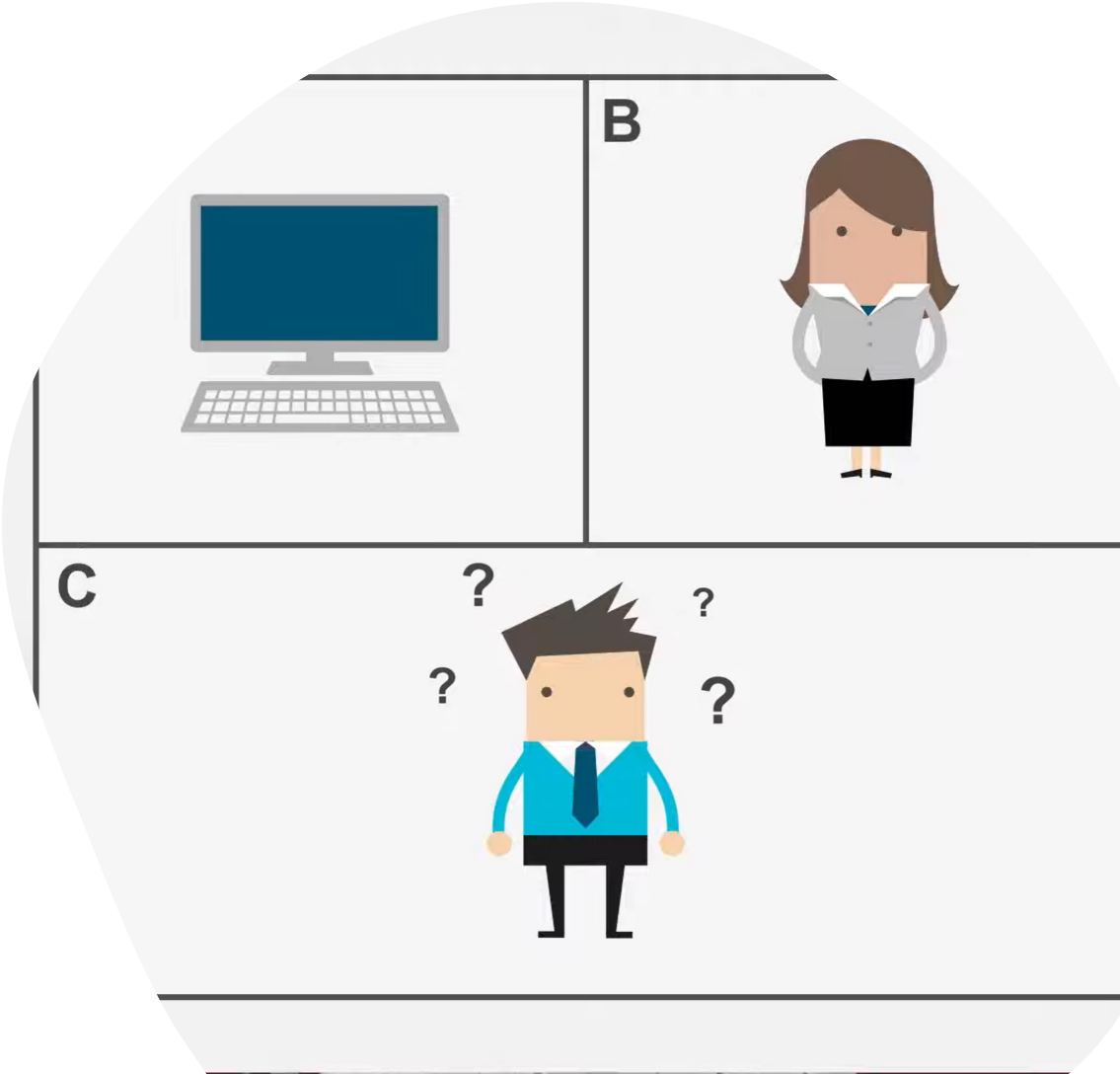
- **Bi-directional** network: separate LSTMs process forward and backward sequences, and hidden layers at each time step are concatenated to form the cell output.
- **Gated Recurrent Unit (GRU):** alternative RNN to LSTM that uses fewer gates, combines forget and input gates into “update” gate, eliminates cell state vector.
- **Attention:** Allows network to learn to attend to different parts of the input at different time steps, shifting its attention to focus on different aspects during its processing.

State-of-the-Art

- Recurrent neural networks
  - LSTM
  - GRU
  - Bi-directional network
- Transformers
- Contextual embeddings
- Large Language Models --> ChatGPT

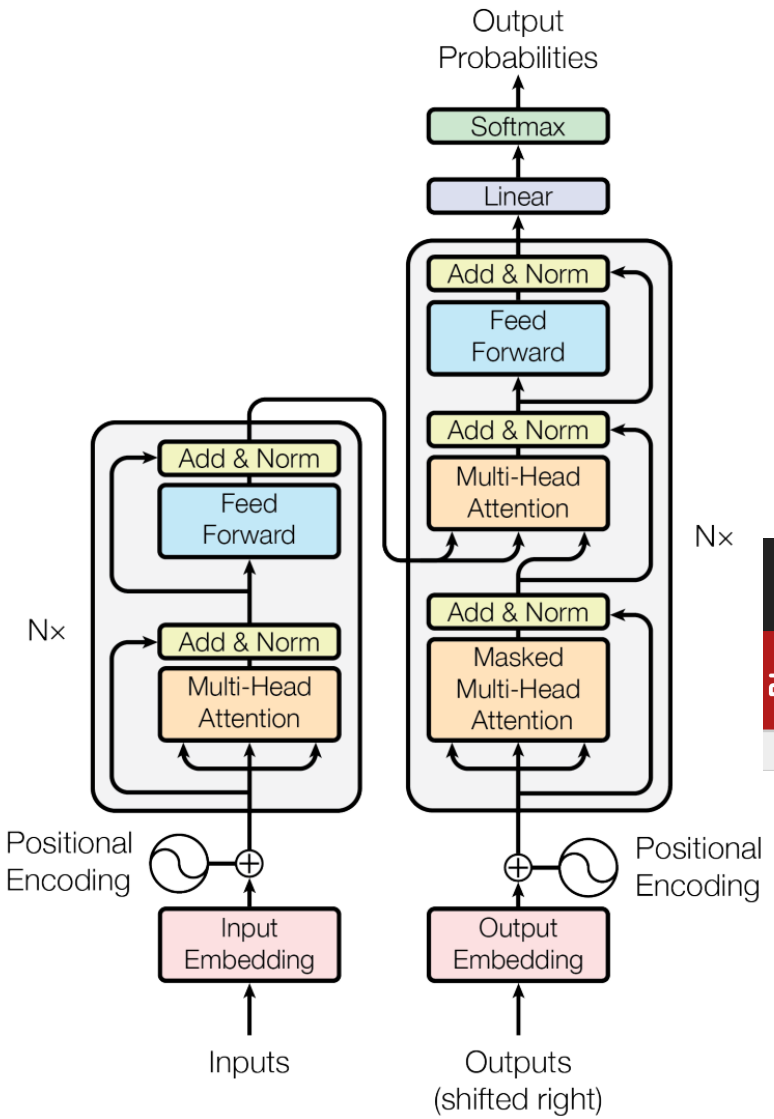
Large Language Models

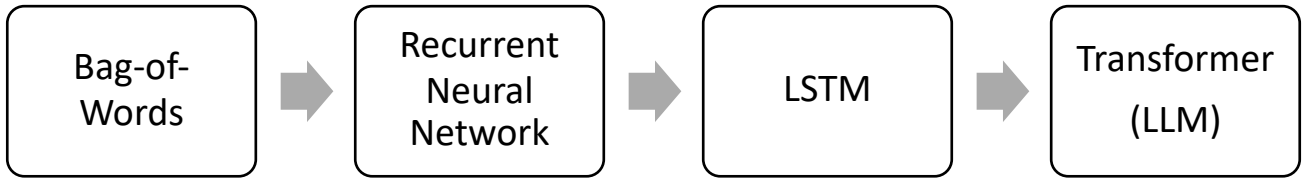

ONCE
UPON A TIME


B
C
?
?
?
?

Transformers!

Transformers!





Cornell University

We gratefully acknowledge

arXiv > cs > arXiv:1706.03762

Search...

Help | Adva

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 2 Aug 2023 (this version, v7)]

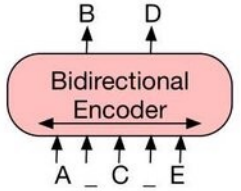
Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

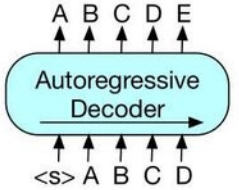
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Transformer foundation models: BERT, GPT, BART

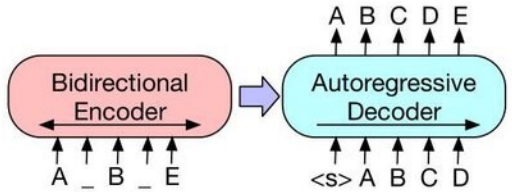
- **BERT**: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.
  - *Masked word prediction, text representation*
- **GPT**: **G**enerative **P**re-trained **T**ransformer.
  - *Next word prediction, text generation, chat*
- **BART** = "BERT+GPT": **B**idirectional encoder and **A**uto-**R**egressive decoder **T**ransformers.
  - *Noised text reconstruction, summarization, translation, spelling correction*



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with a mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

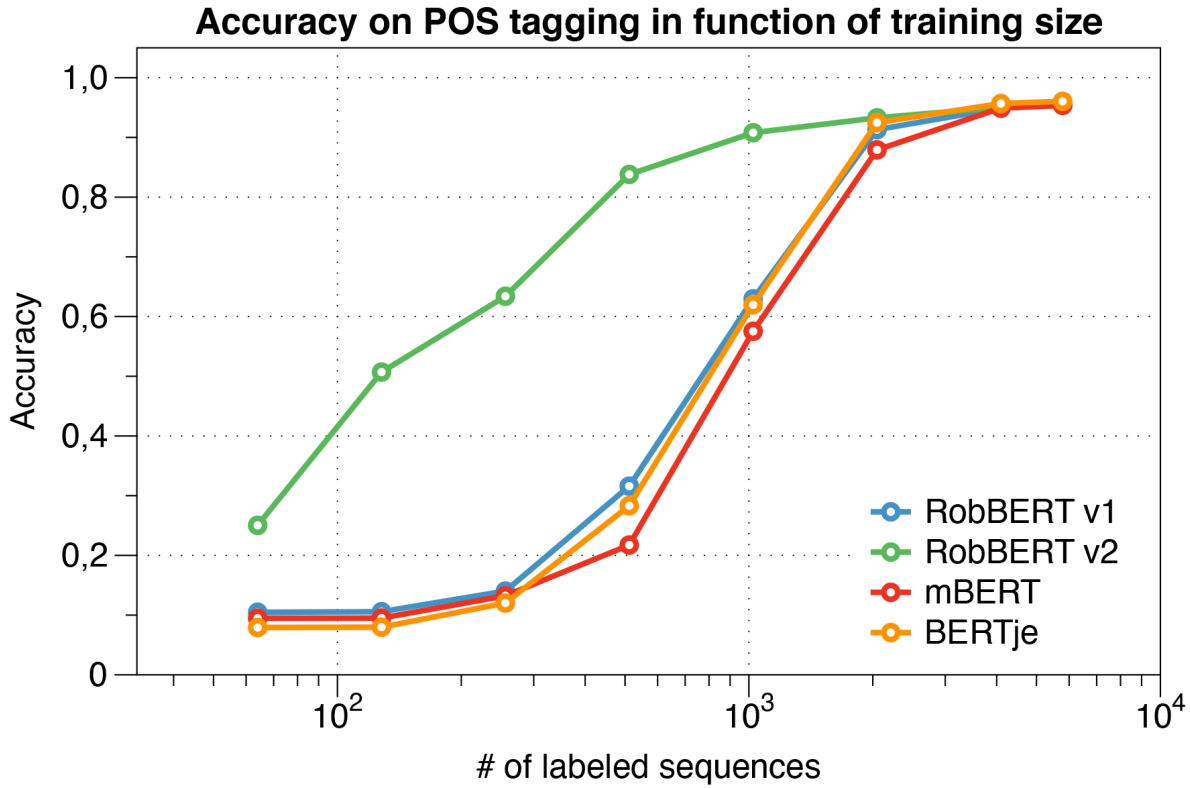
Dutch Transformers!

RobBERT
A Dutch RoBERTa-based Language Model

RobBERT: Dutch RoBERTa-based Language Model.

RobBERT is the state-of-the-art Dutch BERT model. It is a large pre-trained general Dutch language model that can be fine-tuned on a given dataset to perform any text classification, regression or token-tagging task. As such, it has been successfully used by many researchers and practitioners for achieving state-of-the-art performance for a wide range of Dutch natural language processing tasks, including:

- Emotion detection
- Sentiment analysis (book reviews, news articles*)
- Coreference resolution
- Named entity recognition (CoNLL, job titles*, SoNaR)
- Part-of-speech tagging (Small UD Lassy, CGN)
- Zero-shot word prediction
- Humor detection
- Cyberbulling detection



Conclusion

- Neural networks are popular methods especially for text mining
- Feed-forward & RNN & CNN
- RNN works better for text data
- Large Language Models such as GPT are based on RNN and attention deep learning layer.

Practical 7

Application and comparison of deep neural networks for text classification.

Questions?