Deep Learning Part I

Semyon Polyakov

Outline

Intro
- ML vs DL
- DL history

Architectures
- Feedforward networks
- CNN
- RNN
- Transformers

ML vs DL

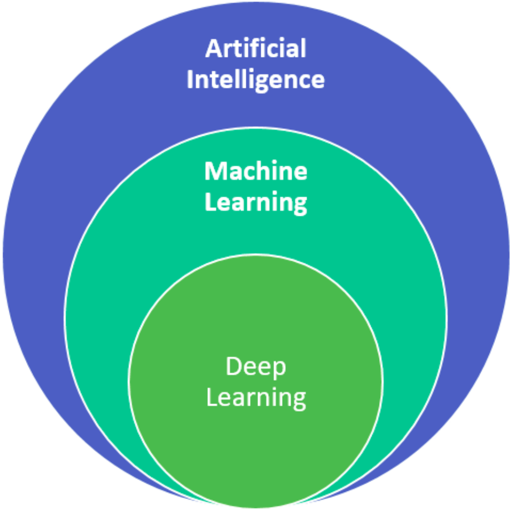


Figure: Big picture of Deep learning

ML vs DL

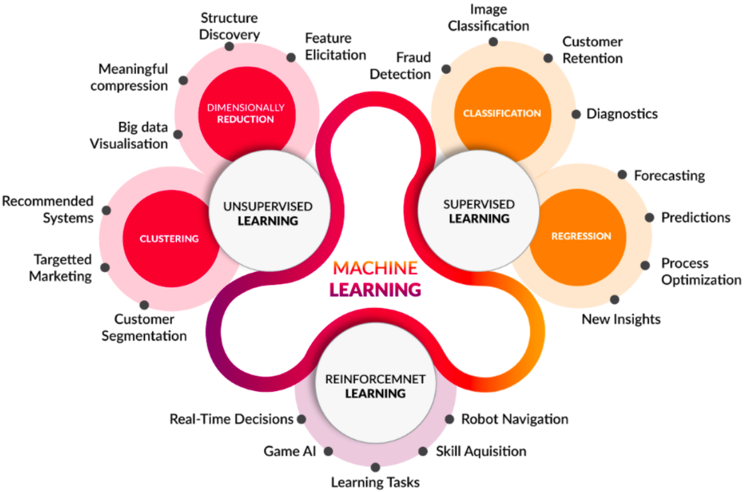


Figure: Big picture of ML

ML vs DL

ML feature engineering steps

1. Collect raw features
2. Design complex features
3. Repeat if not enough

ML vs DL

ML feature engineering steps

1. Collect raw features
2. Design complex features
3. Repeat if not enough

DL feature engineering steps

1. Feed raw features
2. Stack more layers

DL history

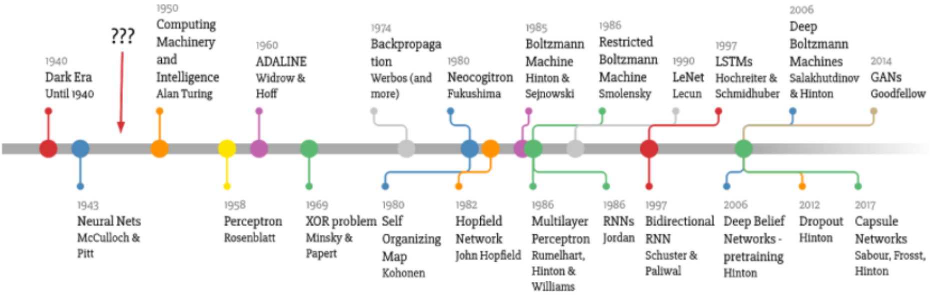


Figure: Brief Deep learning timeline

Perceptron

Perceptron description

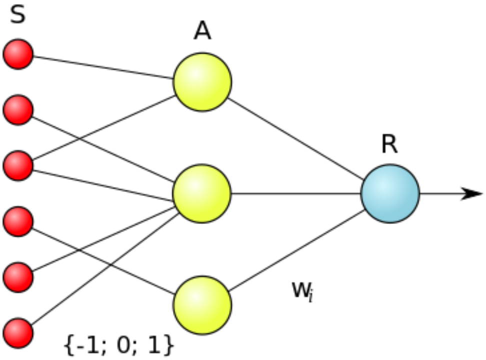


Figure: Perceptron scheme

Perceptron

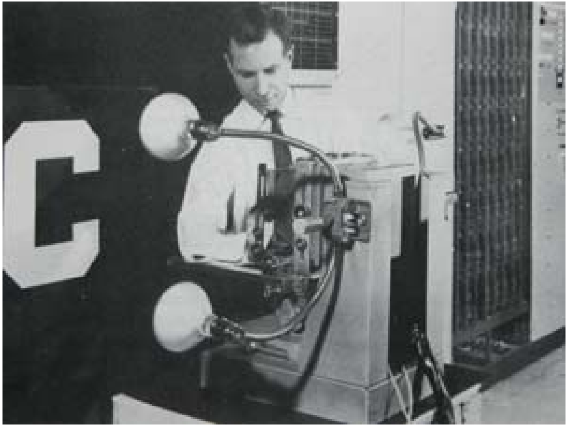


Figure: Mark I with camera system and C symbol

Perceptron learning algorithm

$$y(x) = f(w^T \phi(x))$$

$$\phi_0(x) = 1$$

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

$$E(w) = -\sum_{n \in M} w^T \phi_n t_n$$

$$M - \mathit{misclassified\ examples},\ t \in \{-1, +1\}$$

$$\phi(x) - \mathit{fixed\ nonlinear\ function\ of\ x}$$

Perceptron learning algorithm

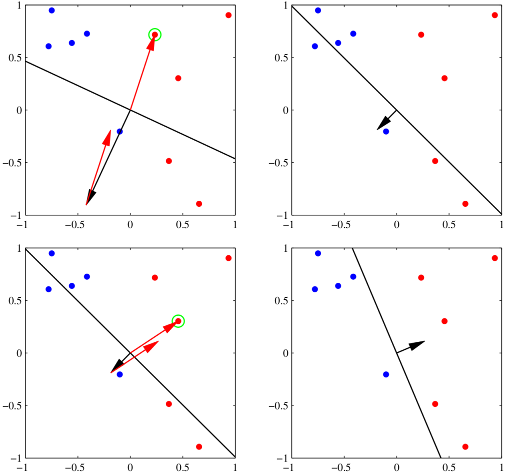


Figure: Perceptron learning example

Perceptron limitations

Limitations

- Only binary classification for linearly separable classes
- Slow convergence

Note

Logistic regression - similar, but better

Error backpropagation algorithm

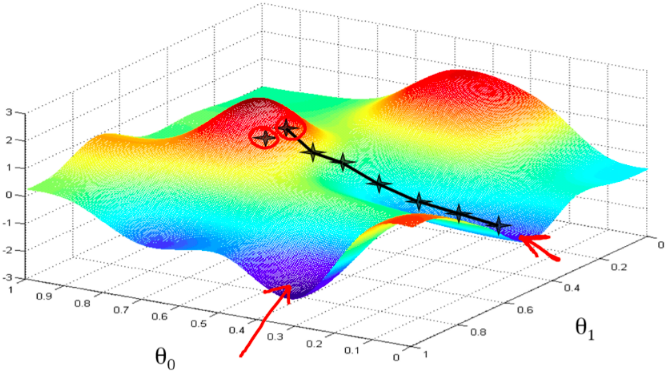


Figure: Gradient descent

Error backpropagation algorithm

Gradient descent optimization

$$w_{(\tau+1)} = w_{(\tau)} - \eta \nabla E(w_{(\tau)})$$

Limitations

- Convergence depends on learning rate
- Finds local minimum
- Can find saddle point instead of local minimum
- Fast convergence only for univariate normal distributed weights

Note

Warmup and LR scheduling for solving problems above

Error backpropagation algorithm

Backpropagation - effective way to calculate the gradient of neural network

$$a_j = \sum_i w_{ji} z_j$$

$$z_j = h(a_j)$$

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

$$\delta_j = \frac{\partial E_n}{\partial a_j}$$

$$\frac{\partial a_j}{\partial w_{ji}} = z_i$$

$$\delta_k = y_k - t_k$$

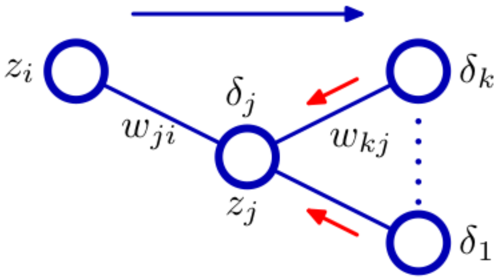


Figure: Backpropagation

Error backpropagation algorithm

Notes

- Much faster than finite differences approximation
- Applied to any neural networks
- One can use the same way to calculate Hessian
- Need to store activations in memory, but there is a gradient checkpoints method
- Can vanish or explode gradients
- Initialiation is non-trivial

What is next?

Neural networks architectures count is innumerable

Feedforward networks

Applications

- Tabular data classification (works worse than ML algorithms)
- Tabular data encoding (better than PCA)
- Part of other neural networks

Feedforward networks

Layers:

- Dense (Matrix multiplication)
- Activations: tanh, ReLU, ReLU-6, Leaky ReLU, ELU, SELU, Swish, ..
- Dropout, BatchNormalization, LayerNormalization, ..

Activations strategy:

1. Use ReLU. Be careful with your learning rates
2. Try out Leaky ReLU / ELU / SELU / Swish to squeeze out some marginal gains
3. Don't use sigmoid or tanh

Preprocessing

- Gaps filling + categorical encoding (only numbers)
- Outliers cleaning
- Normalization (zero mean and unite variance)
- Whitening (zero mean and identity covariance)
- Class balancing

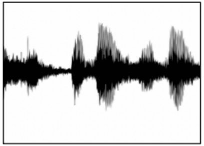
Structured data

Structured Data

Size	#bedrooms	...	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	...	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮
27	71244		1

Unstructured Data



Audio

Image

Four scores and seven
years ago...

Text

Figure: Structured data vs unstructured

CNN

Convolutional neural networks

CV Tasks

Computer vision tasks:

- Image Classification
- Object Detection
- Image Classification With Localization
- Object Segmentation
- Image Style Transfer
- Image Colorization
- Image Reconstruction
- Image Super-Resolution
- Image Synthesis
- OCR
- Other Problems

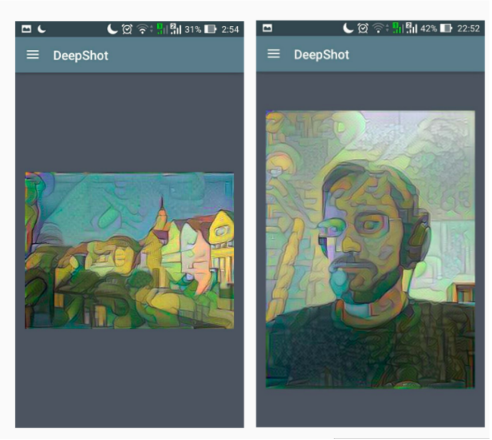


Figure: Style transfer

CNN

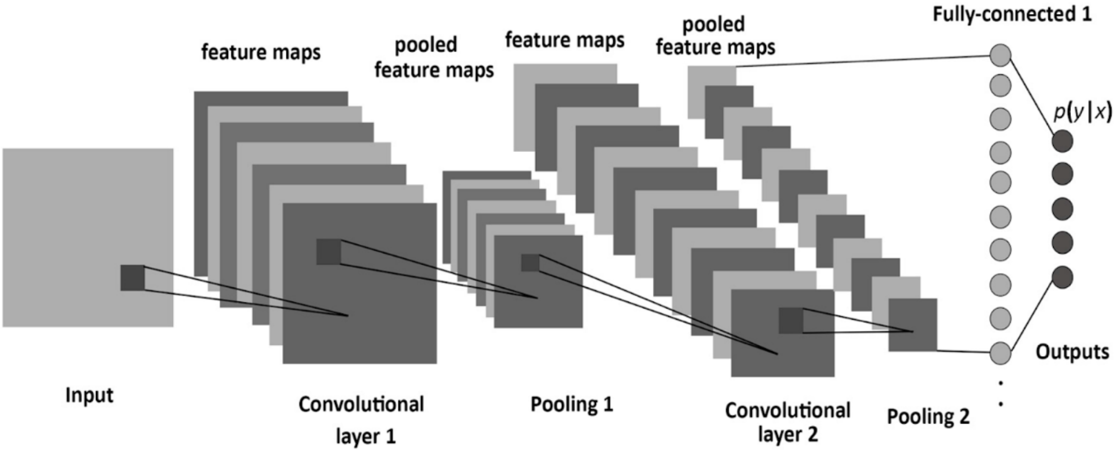


Figure: One of the first CNN - LeNet

CNN

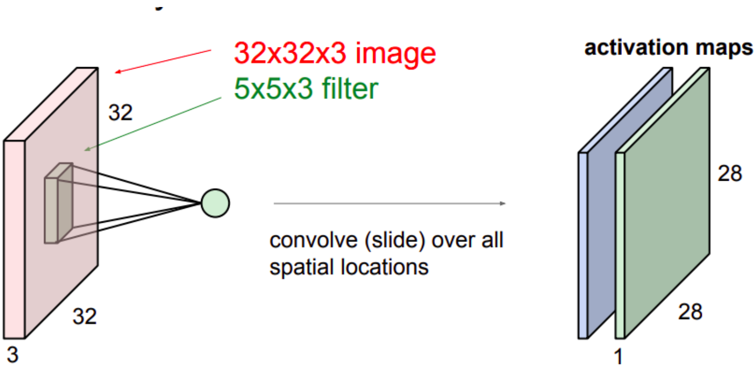


Figure: Convolutional layer scheme

CNN features

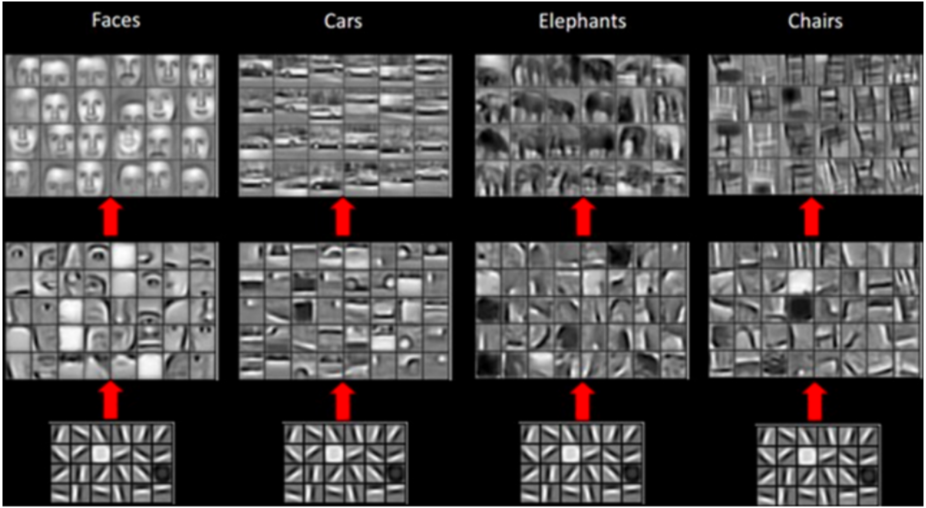


Figure: CNN patterns at different layers

Preprocessing

Preprocessing:

1. Look at the data
2. Outliers cleaning
3. Normalization (grayscale or histogram equalization)
4. Old-school CV methods (blurring, background removal..)
5. Augmentation (rotations, flips, crops, style changes, ..)
6. Resize

Layers and hacks

Layers:

1. Convolutions (+ upconv, + depthwise separable convolutions)
2. Pooling (Avg, Min, Max, + Global)

Hacks:

1. Inception-like blocks
2. Factorization (1 big Conv to 2 small Conv)
3. 1x1 convolutions
4. Residual connections
5. Train top weights with full resolution
6. FixRes augmentation with correct train resizing

Models

Classification:

1. VGG-16 (2014)
2. ResNeXt-50 (2017)
3. MobileNetV2 (2018)
4. EfficientNet (2019)
5. FixEfficientNet-L2 (2020)

Object detection:

1. Faster R-CNN (2016)
2. Mask R-CNN (2018)
3. EfficientDet (2019)
4. CSP-p7 + Mish (2019)

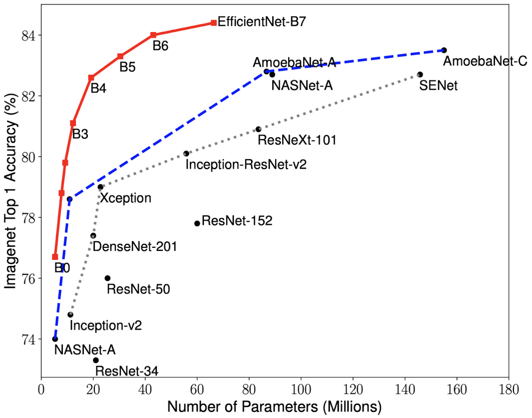


Figure: Imagenet accuracy timeline

RNN

Recurrent neural networks

RNN Tasks

Sequence processing tasks:

- Text sequences: characters, words, n-grams, BPE
- Video (classification, detection, captioning,..)
- Audio (classification, Speech2Text, Text2Speech, ..)
- Time series: financial, social networks (forecasting, classification, ..)
- Other

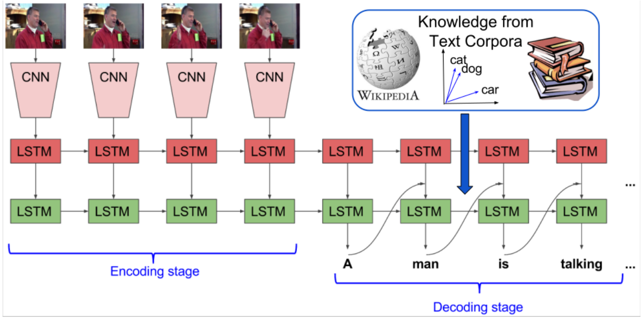


Figure: Video captioning

RNN

Common formula: $h_t = f(h_{t-1}, x_t)$

Vanilla RNN:

$$h_t = tanh(W_{hh}h_{t-1}) + W_{xh}x_t$$

$$y_t = W_{hy}h_t$$

GRU:

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

$$\tilde{h}_t = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1} + b_h)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

$\odot$ - Hadamard product

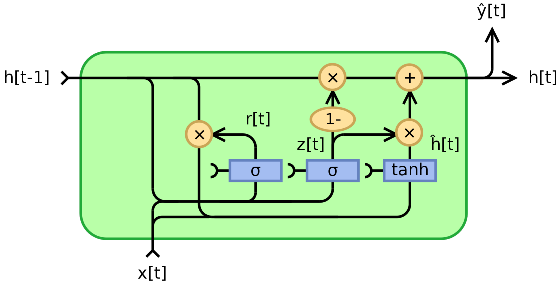
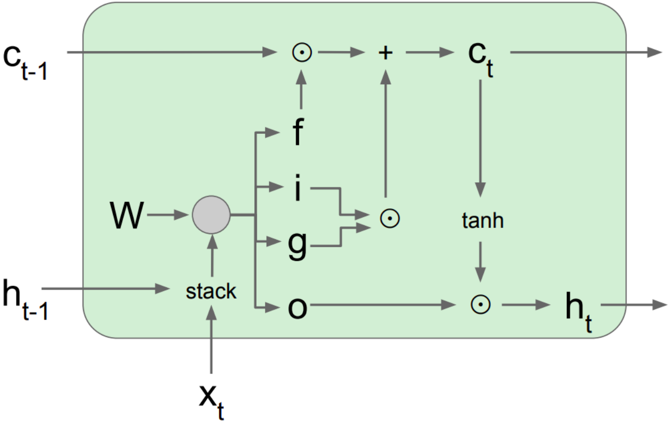


Figure: Gated recurrent unit scheme

RNN

LSTM (1997)



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Figure: Long-short term memory scheme

RNN

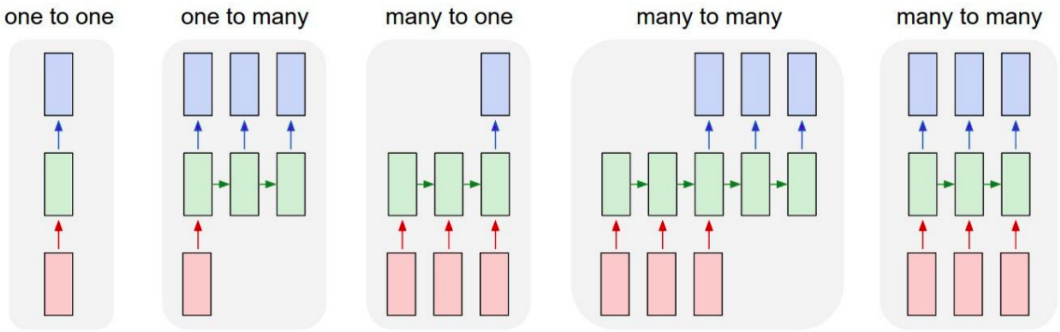


Figure: Recurrent neural network types

RNN

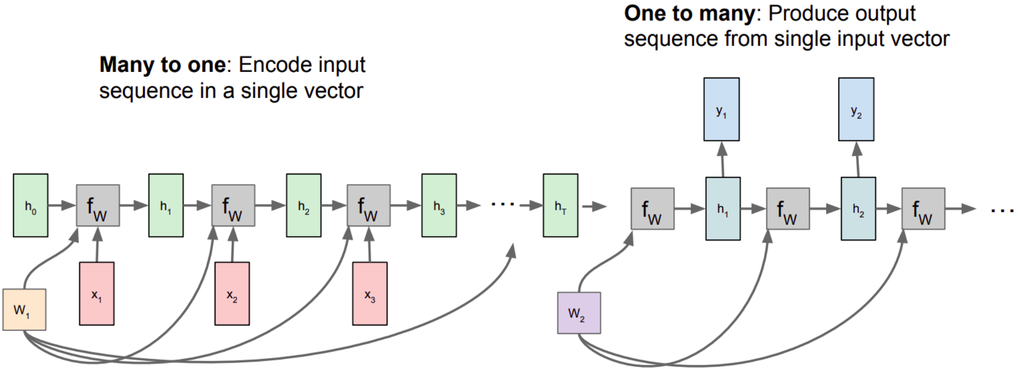


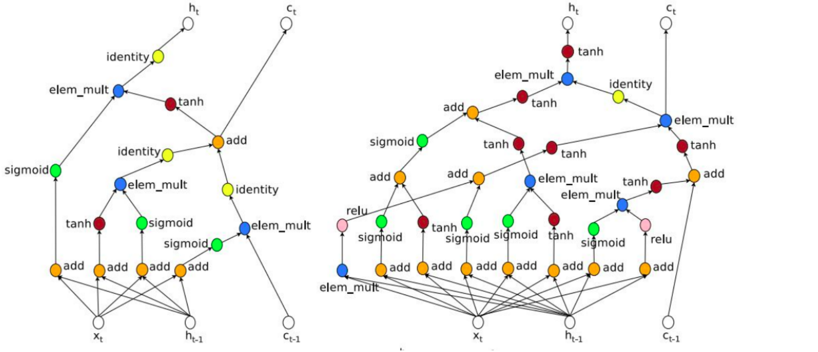
Figure: Sequence2Sequence architecture. Example - machine translation

RNN notes

Notes:

1. Additive operations helps to avoid vanishing gradients (see also residual and highway layers)
2. Gradient exploding is controlled by gradient clipping
3. Train initial state
4. Tie inputs and outputs when it is possible
5. One can stack RNN layers as any other layer (ELMO RNN)
6. Other NN can be input for the NN (W2V, CNN + RNN for video tasks)
7. Bidirectional RNN often outperforms
8. Stephen Merity's SHA-RNN paper - https://arxiv.org/pdf/1911.11423.pdf

RNN



LSTM cell

Cell they found

Figure: Neural architecture search, applied for language modeling, ICLR 2017

Attention and Transformers

Transformers

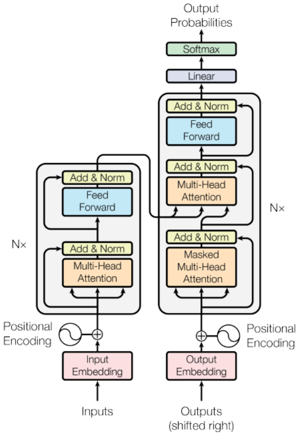


Figure 1: The Transformer - model architecture.

Figure: Trasnformer architecture from paper "Attention is All you need"

Transformers Tasks

- All NLP tasks (classification, autoencoding, synthesis, ..)
- Attention mechanism applied everywhere - CV, speech, tabular data, ..
- One of the biggest models (GPT-2, GPT-3, ..)

Transformers: notes

Notes:

1. No vanishing gradients in comparison to RNN
2. A lot of models is computationally expensive
3. Almost never meet overfitting
4. There is optimizations for long sequences (Transformer-XL, Reformer)

Transformers

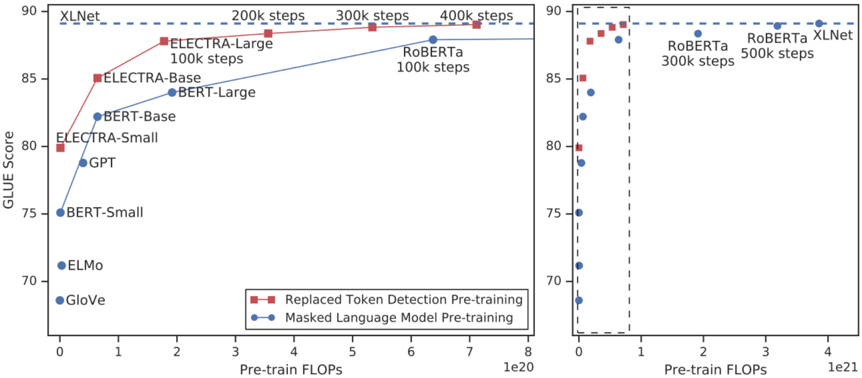


Figure: Replaced token detection vs masked language models. ICLR 2020

Thank you for attention!

SOTA ML School

State-Of-The-Art Machine Learning School