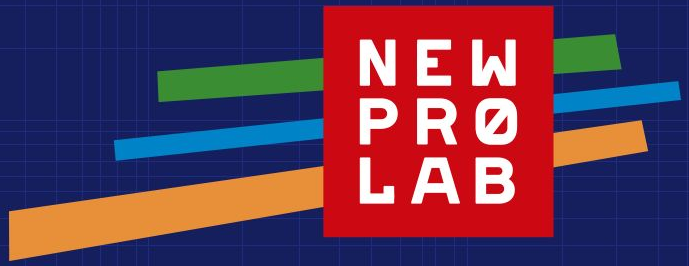


Введение в Deep Learning

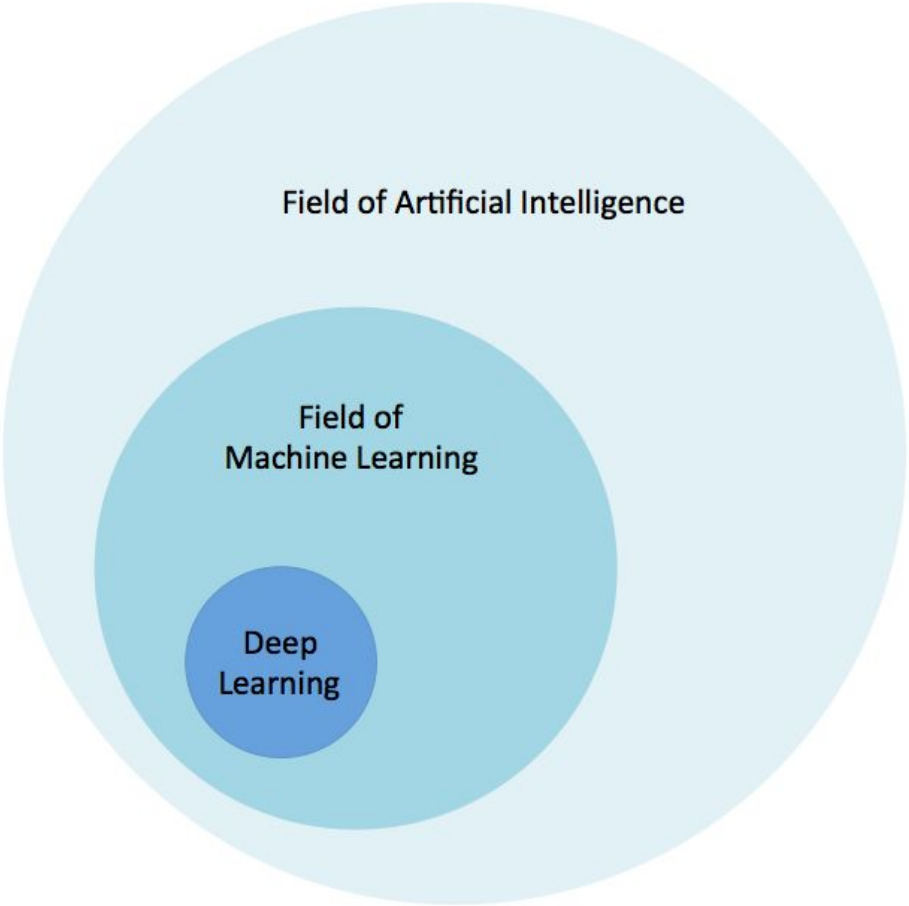
Григорий Сапунов
Со-основатель и CTO в Inten.to

NEWPROLAB.COM

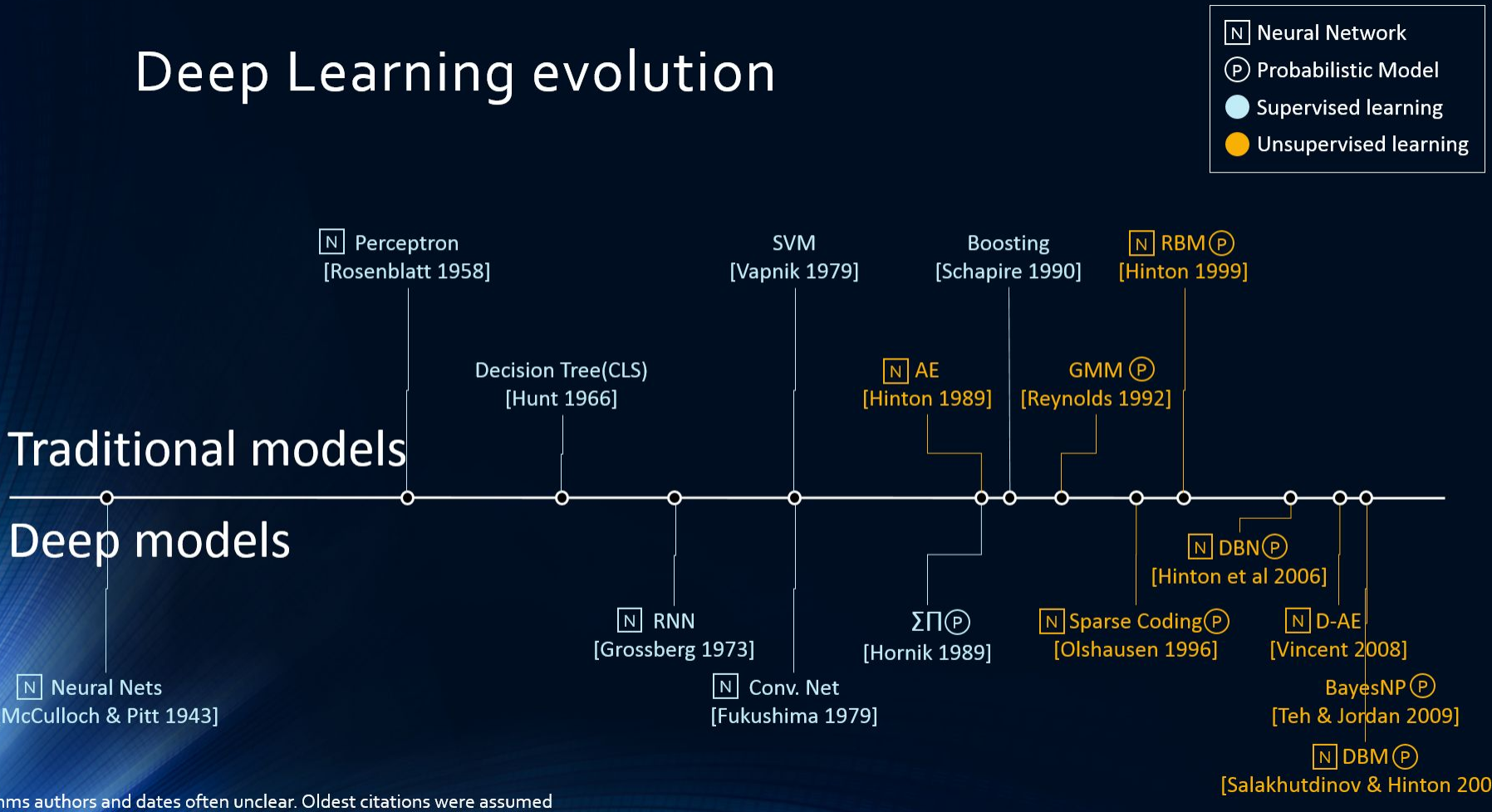
Deep Learning vs. Machine Learning

В чём различия и что общее?

Место Deep Learning среди других областей



Deep Learning evolution



Algorithms authors and dates often unclear. Oldest citations were assumed
Classifications based on Yann LeCun's Deep Learning class at NYU – spring 2014

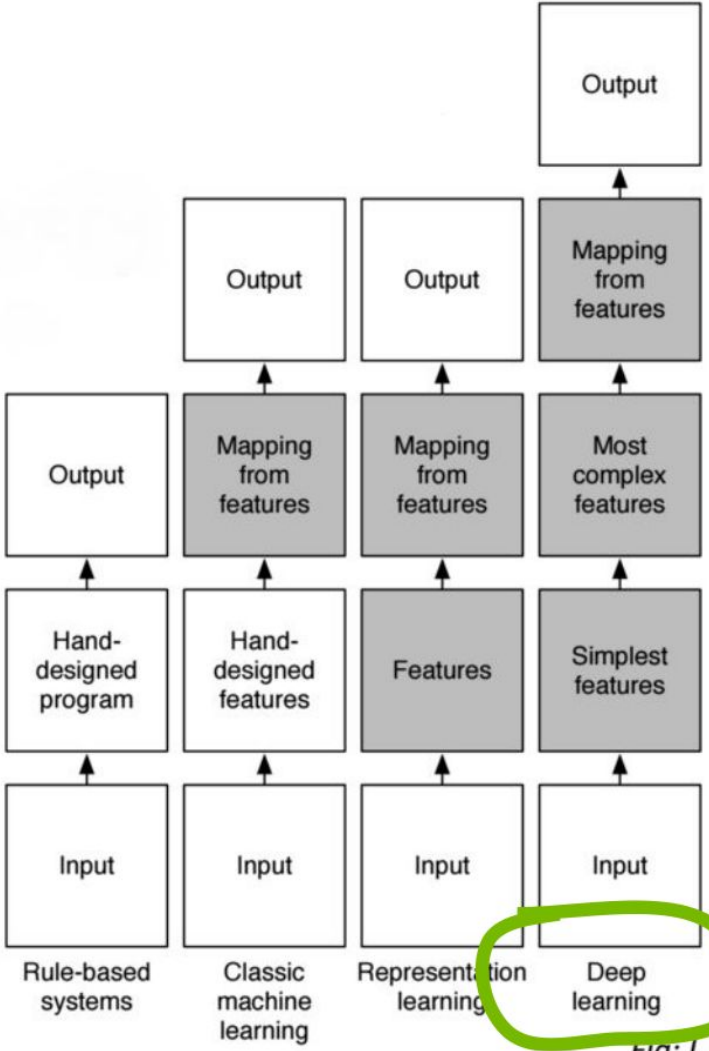
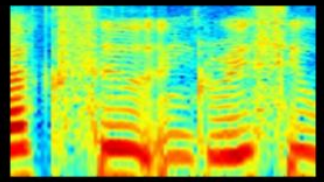
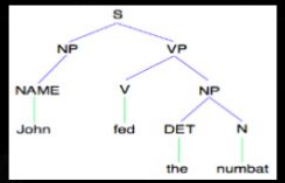
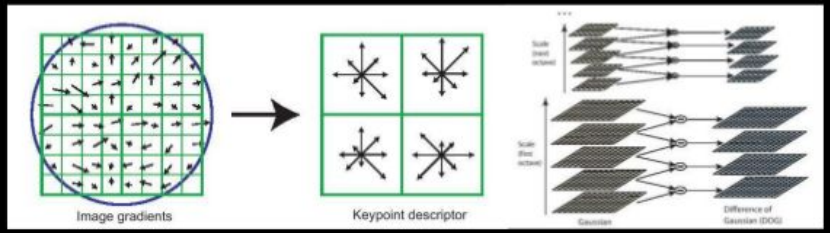
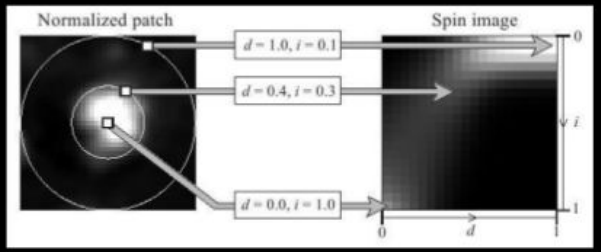


Fig: I. Goodfellow

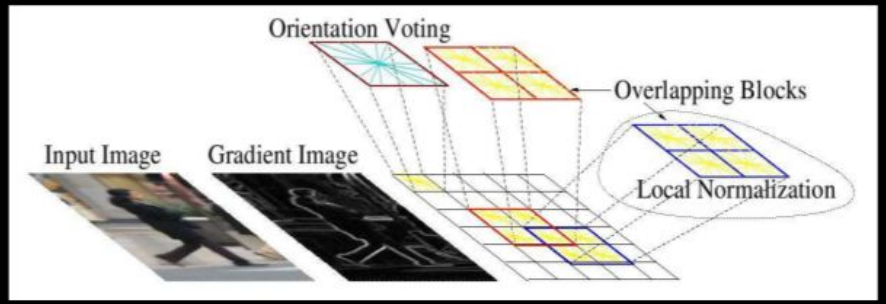
Raw data	Feature extraction	Classifier / detector	Result
		SVM, shallow neural net, …	
		HMM, shallow neural net, …	Speaker ID, speech transcription, …
		Clustering, HMM, LDA, LSA …	Topic classification, machine translation, sentiment analysis…



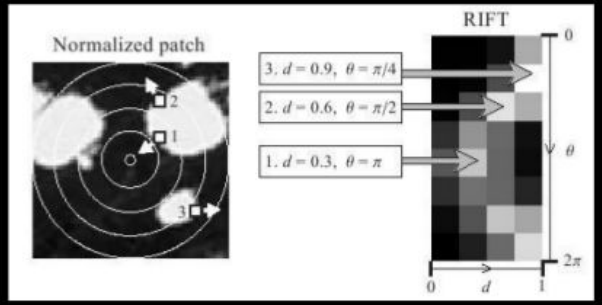
SIFT



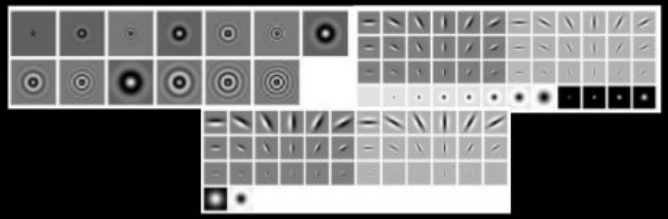
Spin image



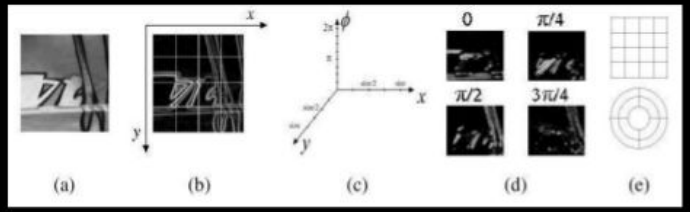
HoG



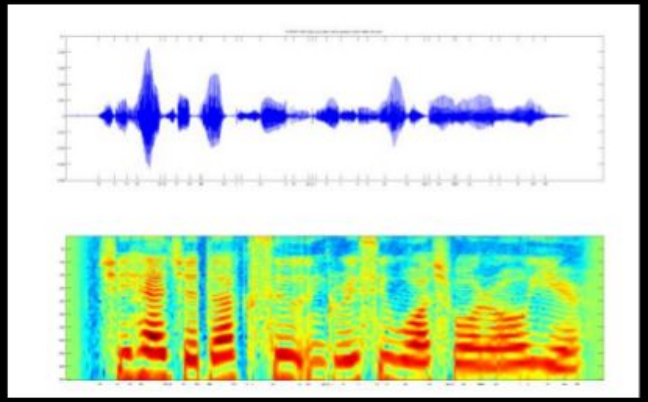
RIFT



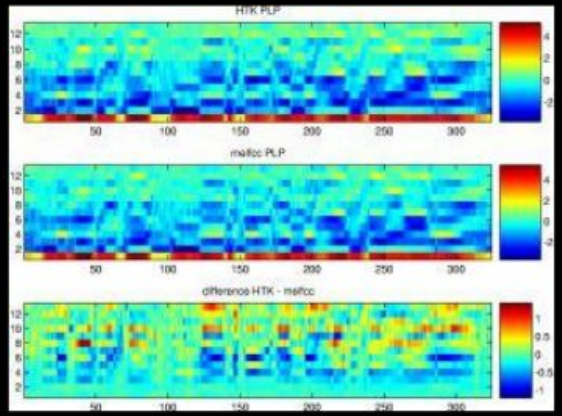
Textons



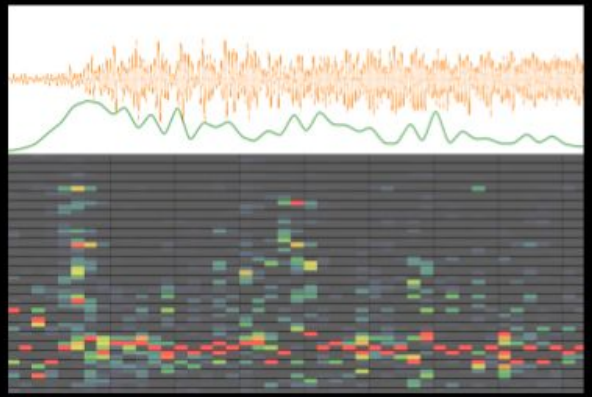
GLOH



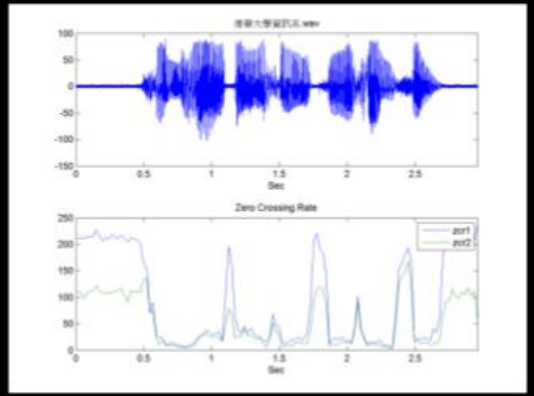
Spectrogram



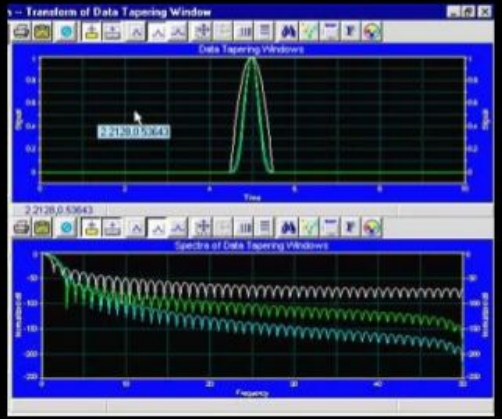
MFCC



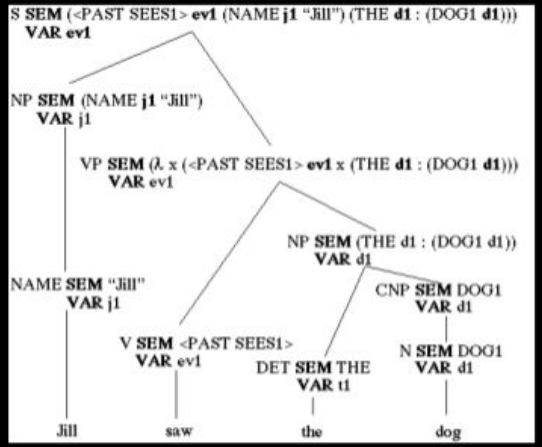
Flux



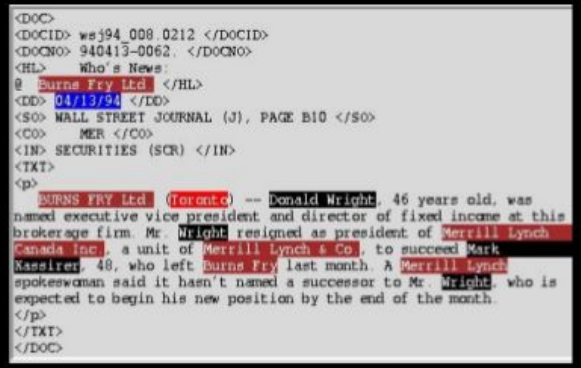
ZCR



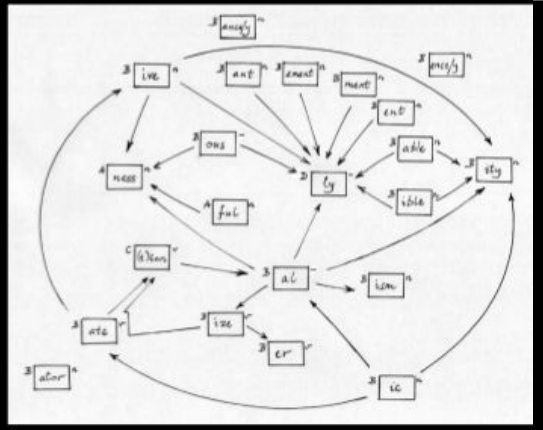
Rolloff



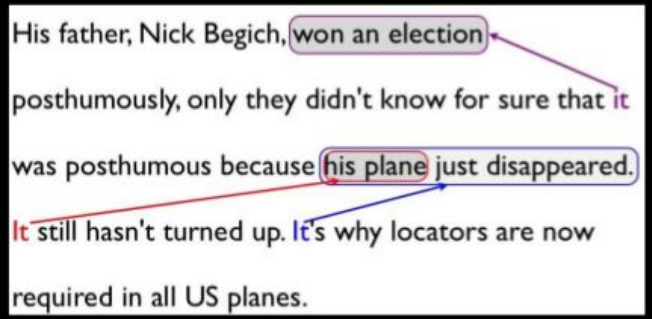
Parser features



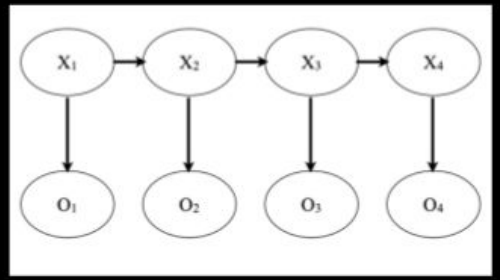
NER/SRL



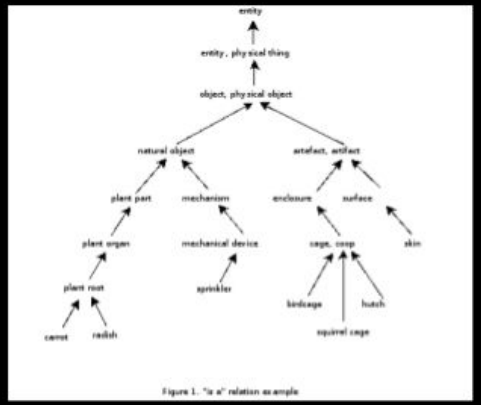
Stemming



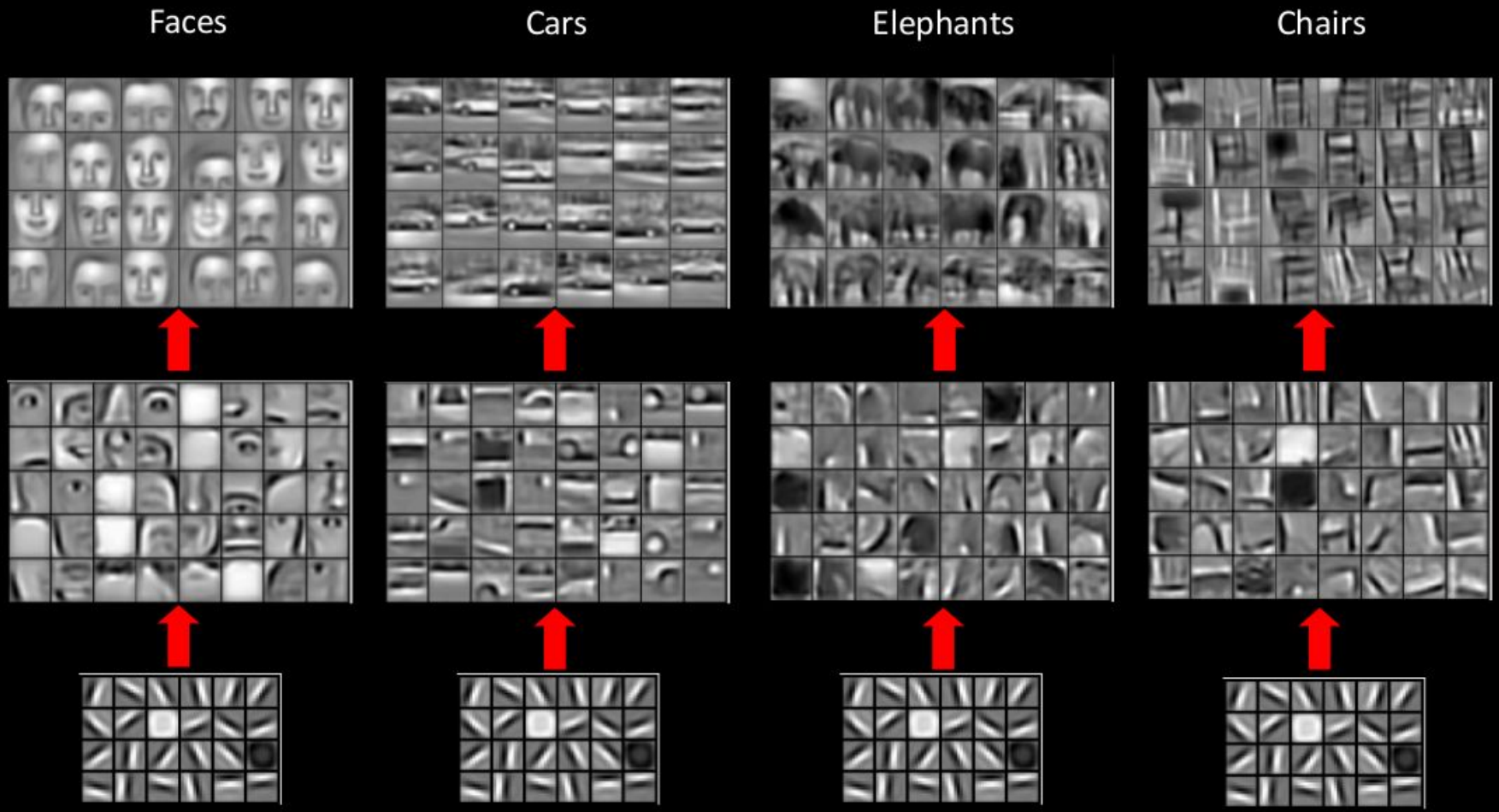
Anaphora



POS tagging



WordNet features

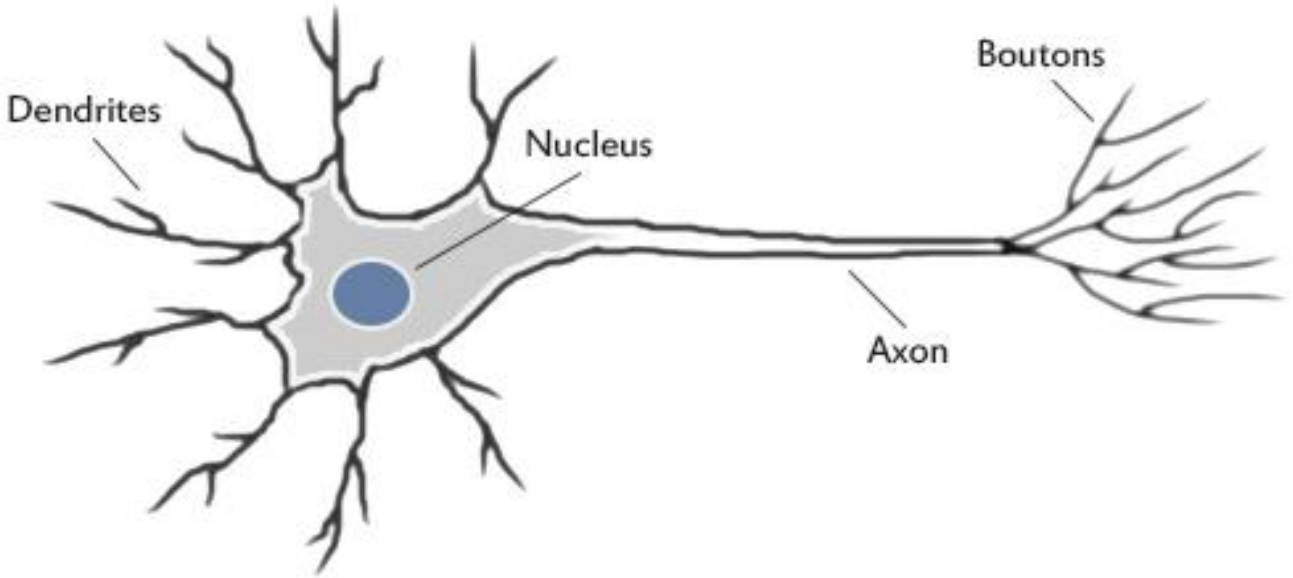


[Lee, Grosse, Ranganath & Ng, 2009]

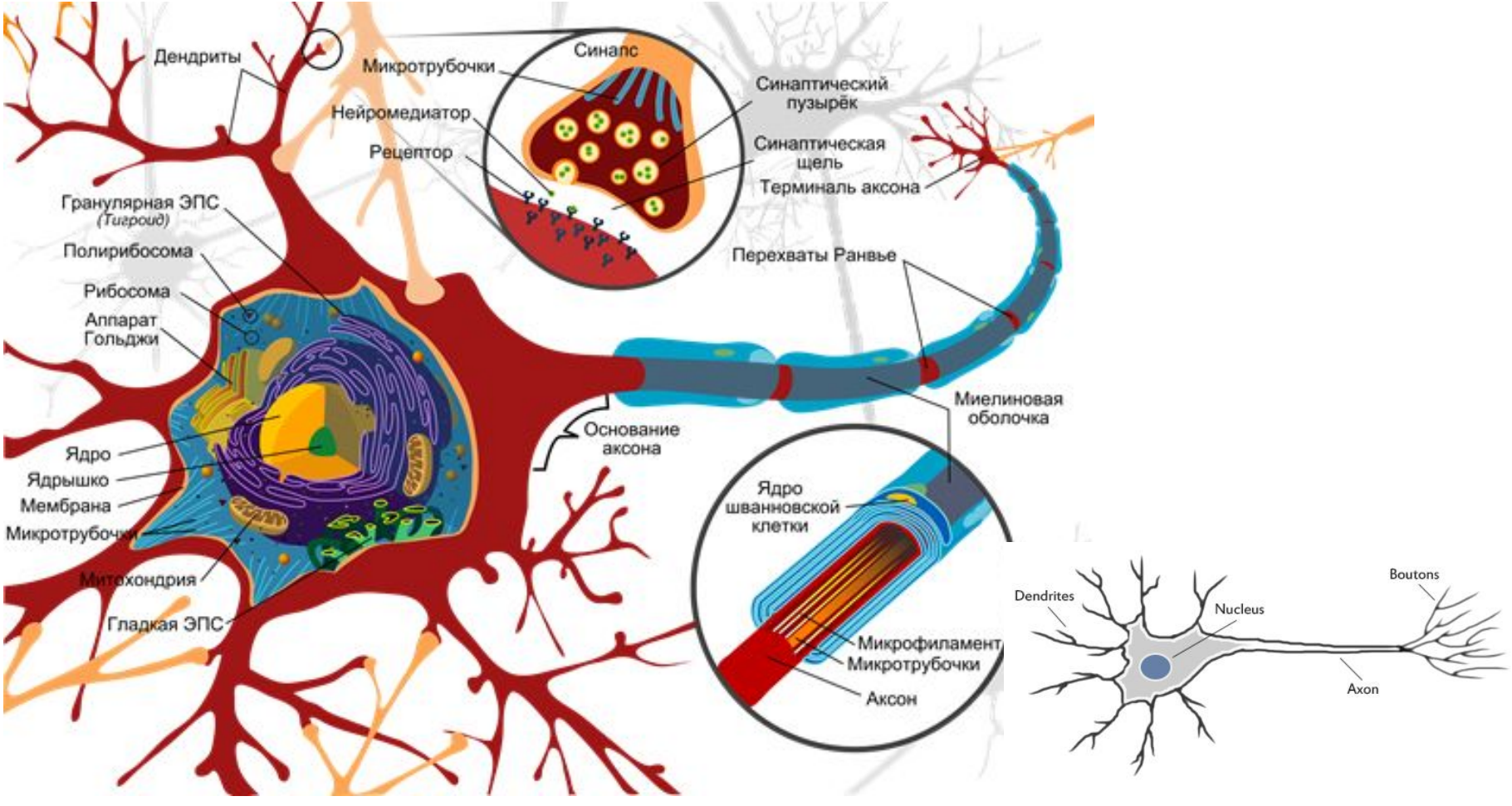
Нейрон и нейросеть

Естественный нейрон

Нейрон — клетка нервной системы, способная к возбуждению и передаче сигнала.

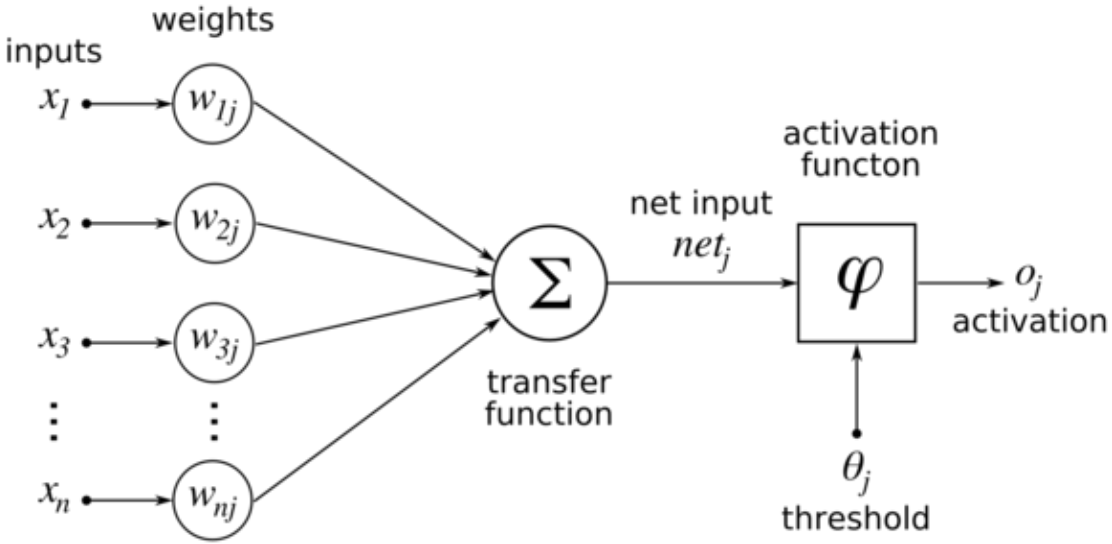


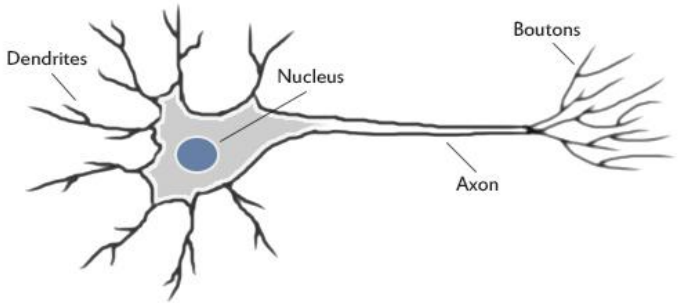
В реальности всё посложнее...



Искусственный нейрон

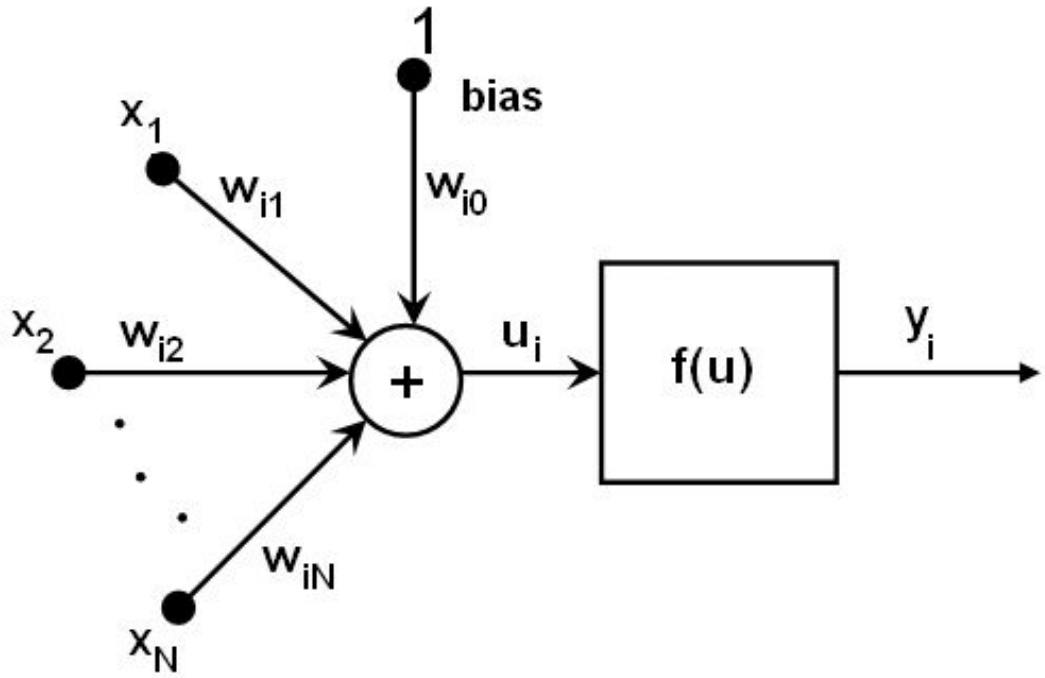
Искусственный нейрон — отдалённое подобие биологического.
Базовый элемент искусственной нейронной сети





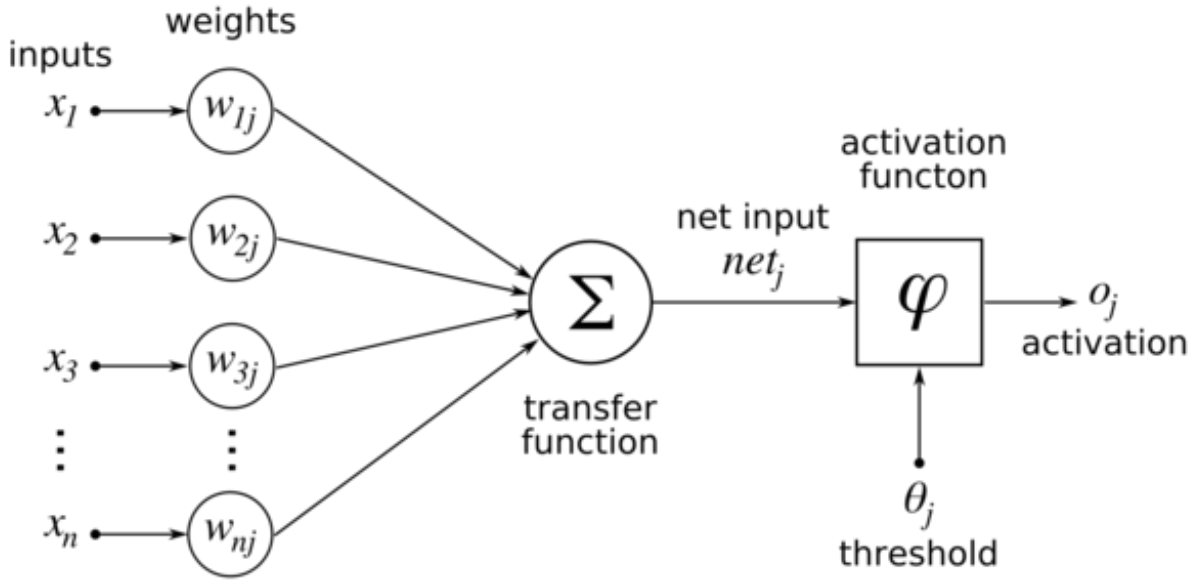
Модель искусственного нейрона

Обычно у нейрона также есть bias-вход, константное смещение, вес на котором также обучается.



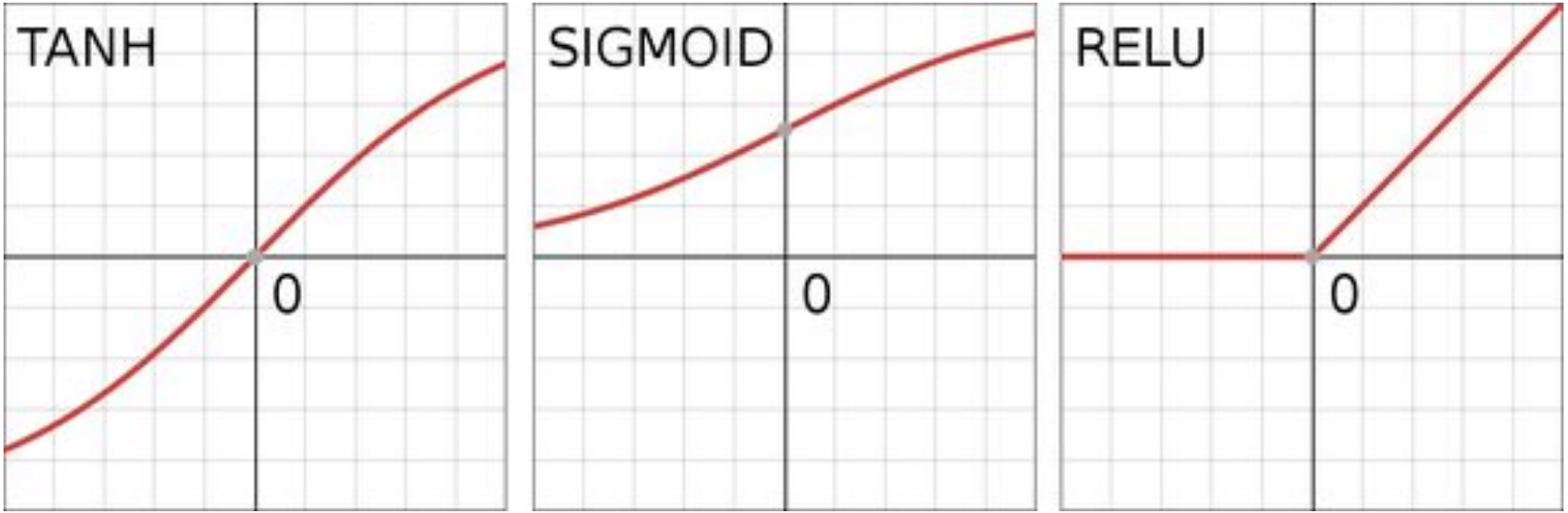
Модель искусственного нейрона

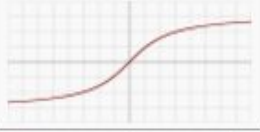
Отдельный нейрон математически полностью аналогичен логистической регрессии (в случае если используется функция активации сигмоида)



$$f(z) = \frac{1}{1 + e^{-z}}.$$

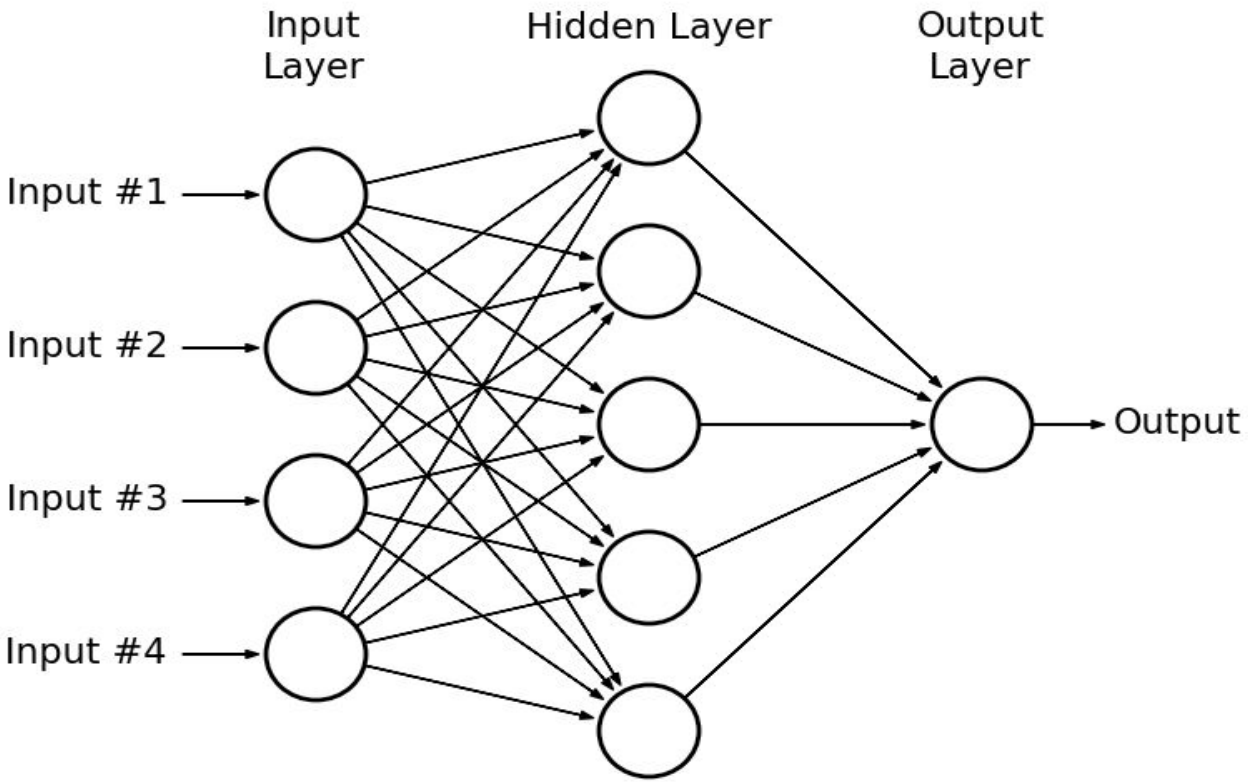
Функции активации: Tanh, Sigmoid, ReLU



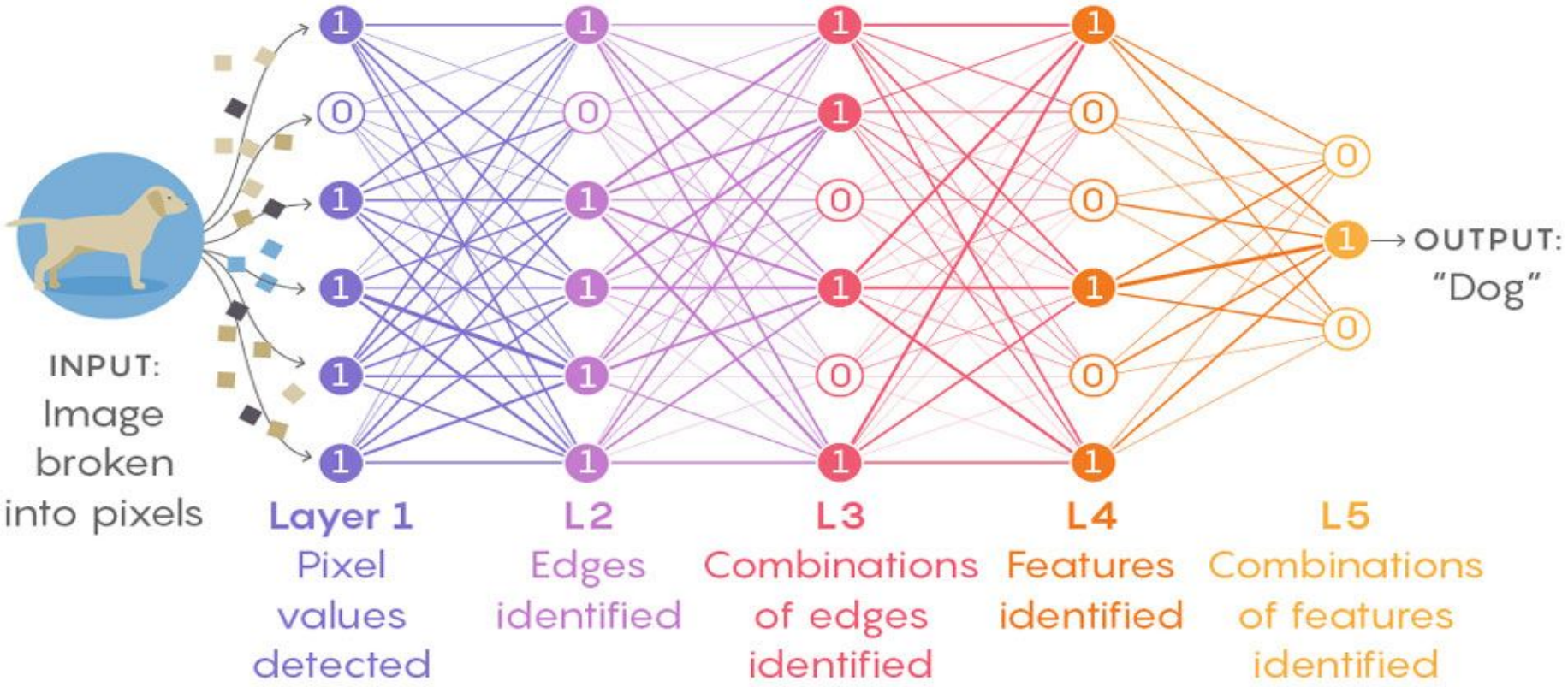
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

https://en.wikipedia.org/wiki/Activation_function

Искусственная нейросеть: терминология



Иерархические признаки



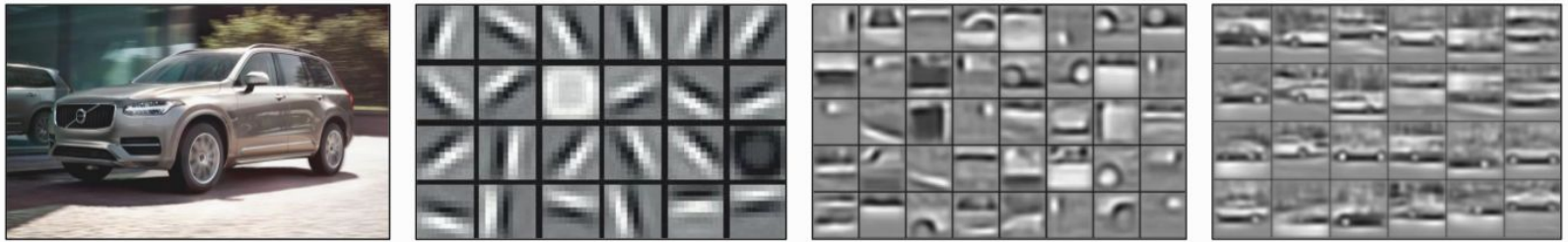
https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921

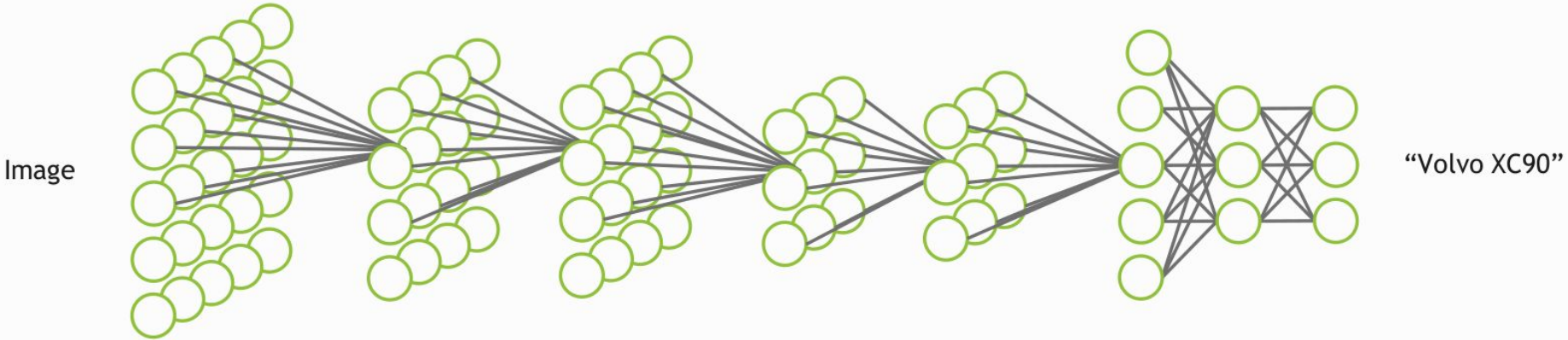
Использование нейросети

- На вход нейросети для каждого объекта подаётся вектор признаков
- На выходе нейросети получается результат. Вид результата определяется архитектурой нейросети
  - Например, в случае многоклассовой классификации (n классов) выходной слой содержит n нейронов, каждый из которых выдаёт “степень принадлежности” к данному классу (если активация была сигмоидальной)
  - Если нужно получить распределение вероятностей классов, используется функция softmax, нормализующая все выходные значения, так чтобы их сумма была равна 1.

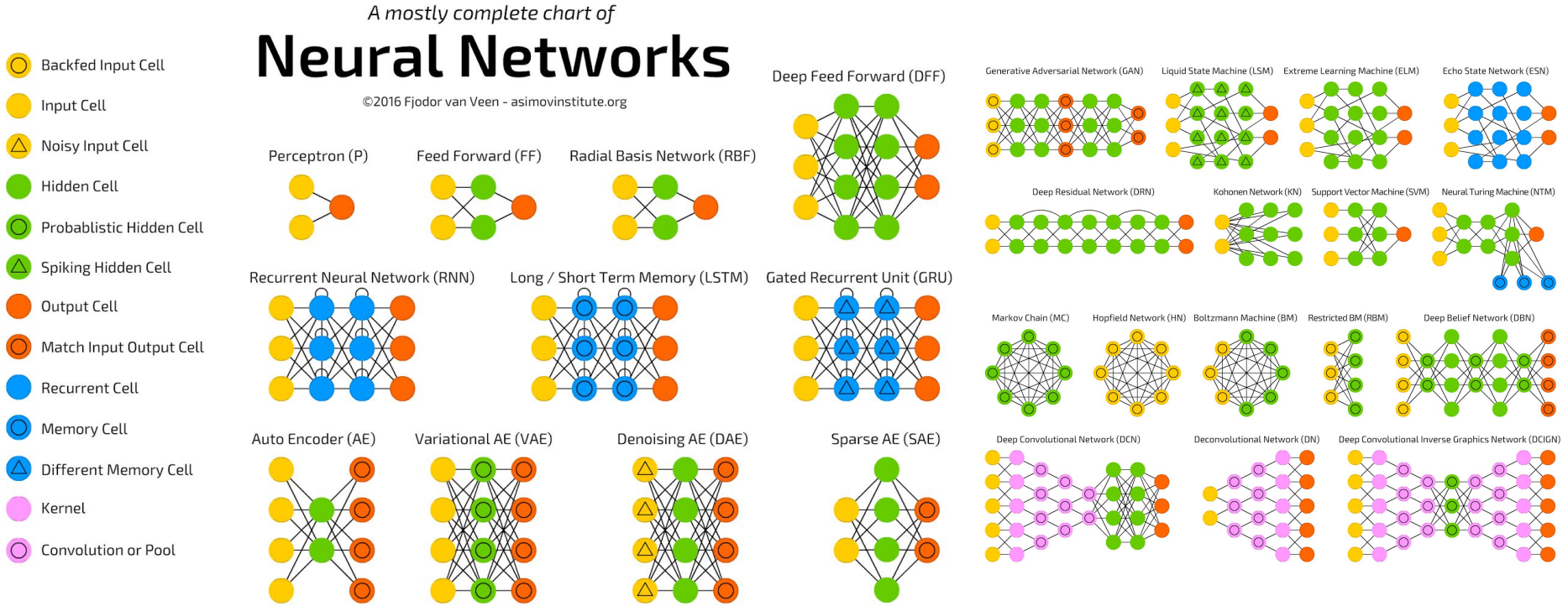
Глубокая нейросеть

Наличие более одного скрытого слоя даёт возможность строить иерархические представления.



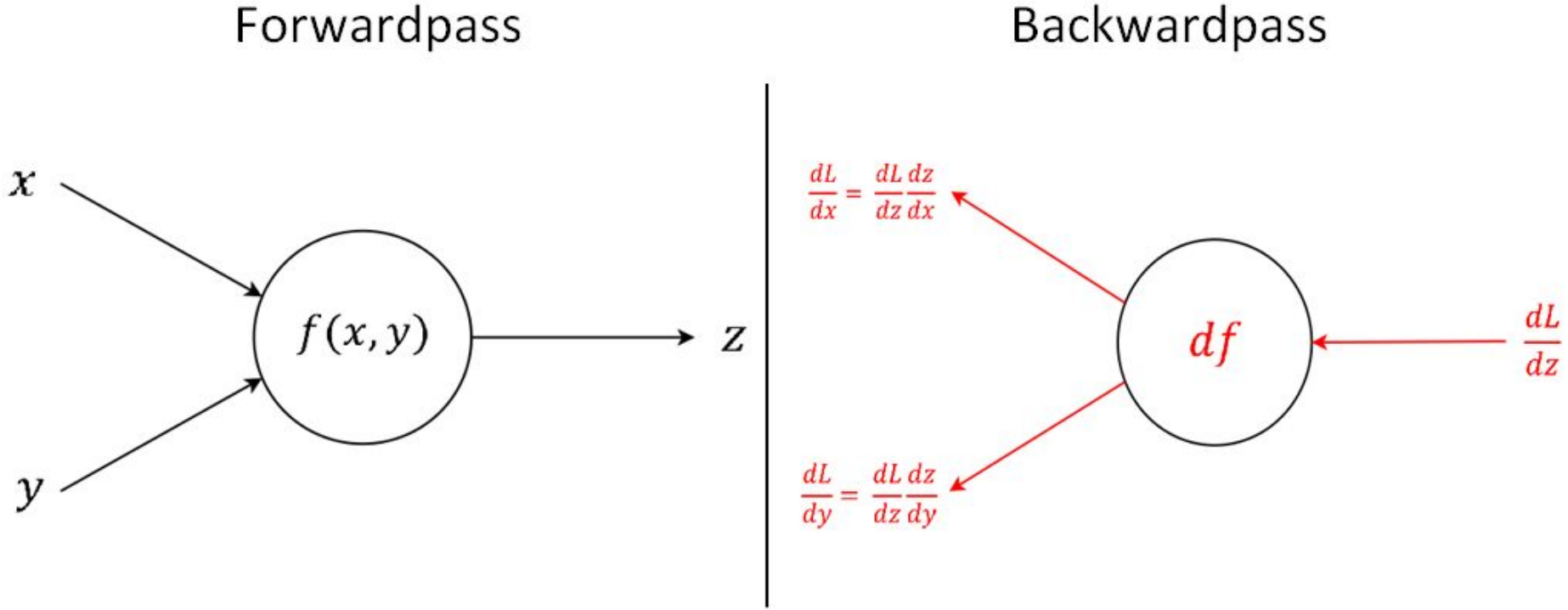


Существует многообразие архитектур



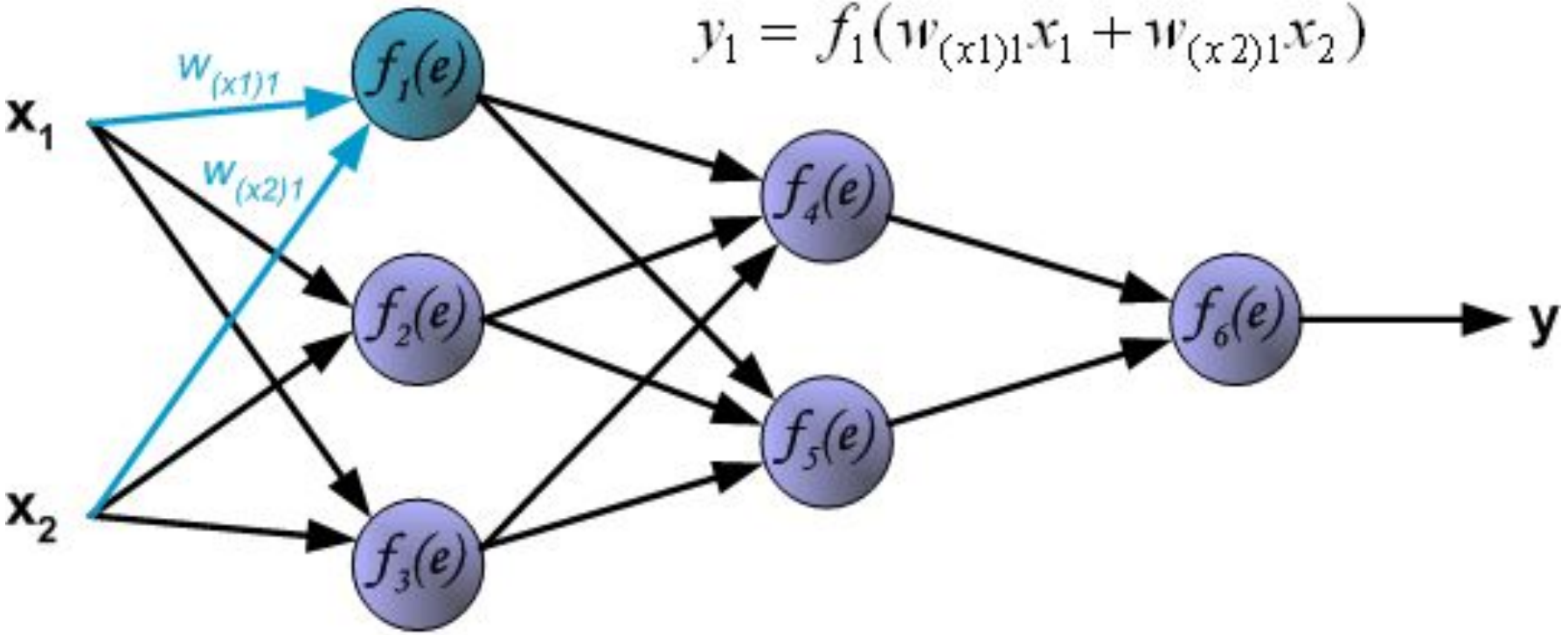
http://www.asimovinstitute.org/neural-network-zoo/

Два режима работы нейрона и нейросети



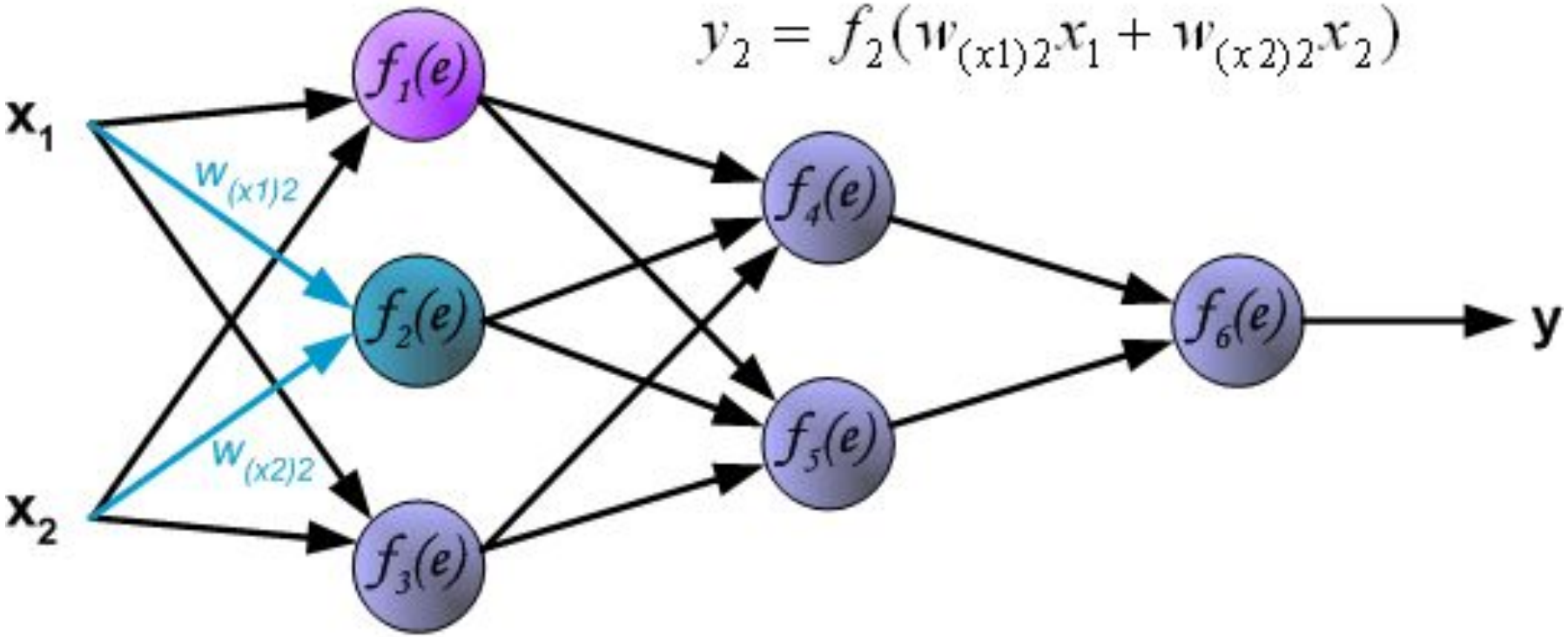
Механизмы работы
1. Forward propagation

Forward propagation



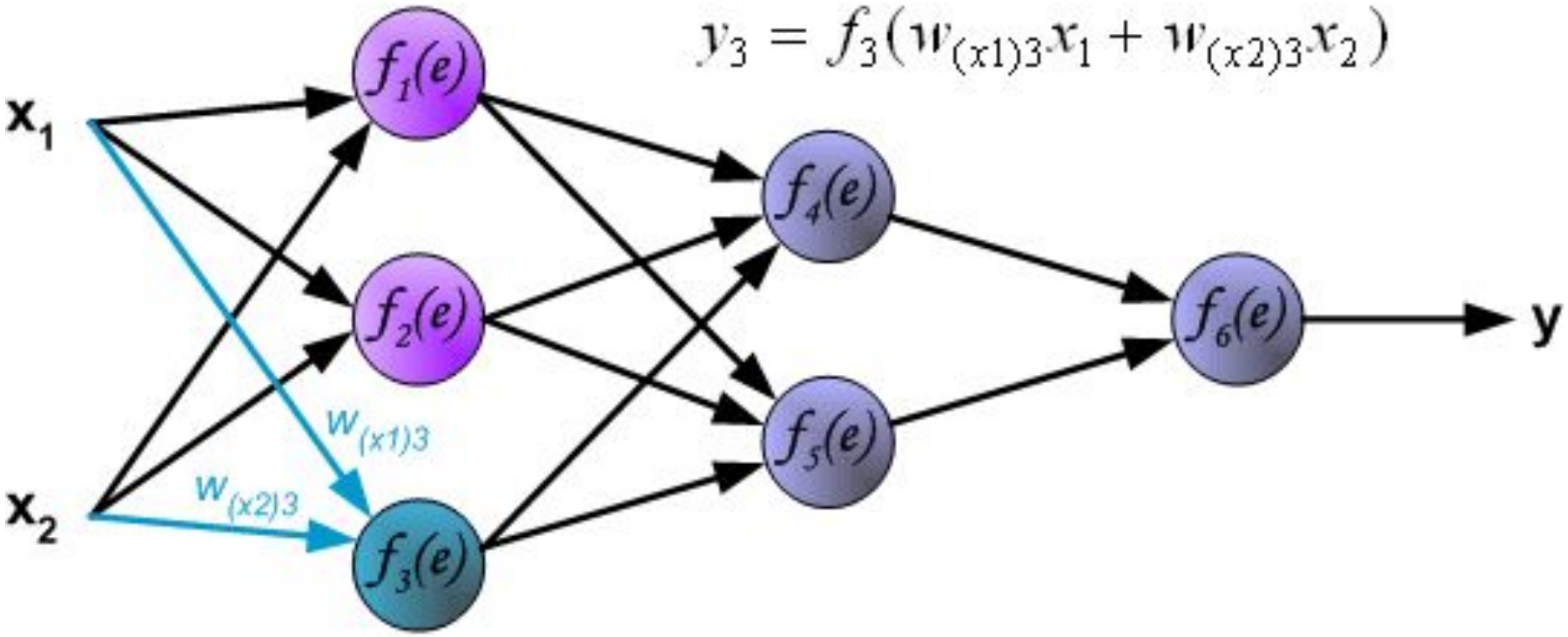
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Forward propagation



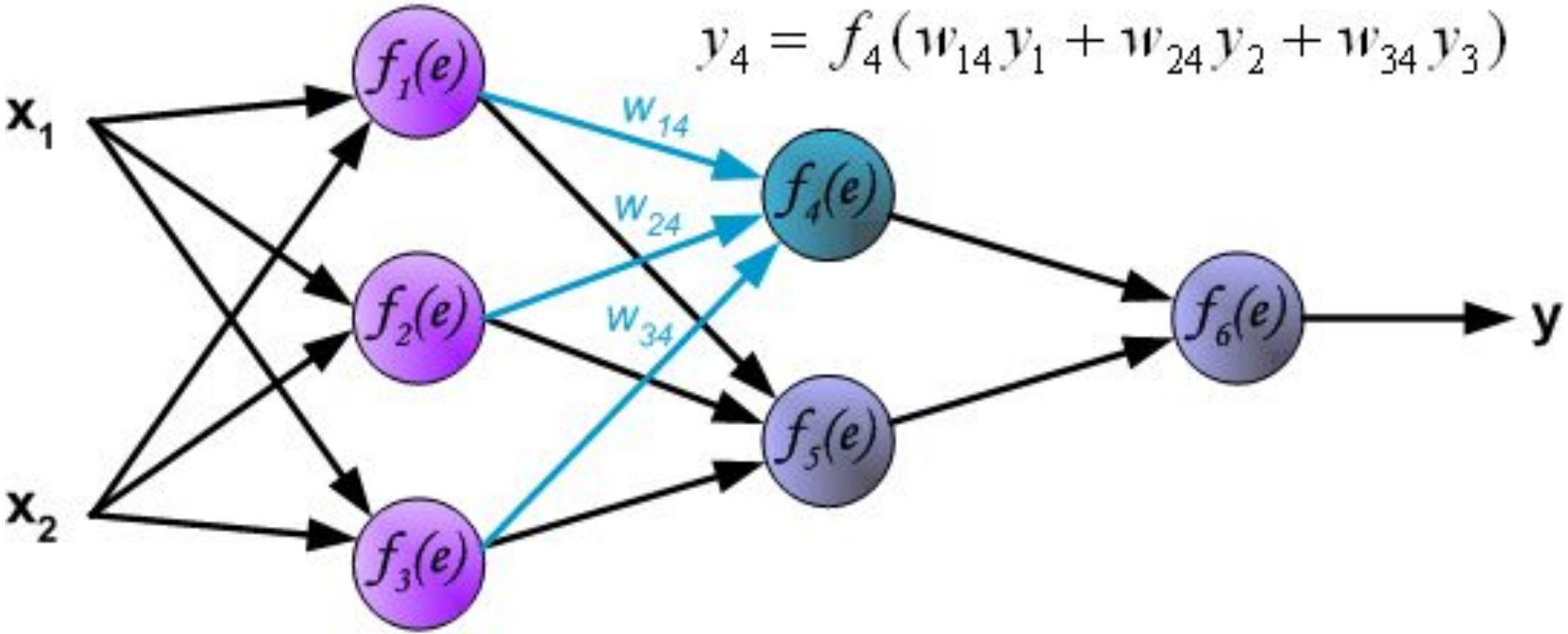
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Forward propagation



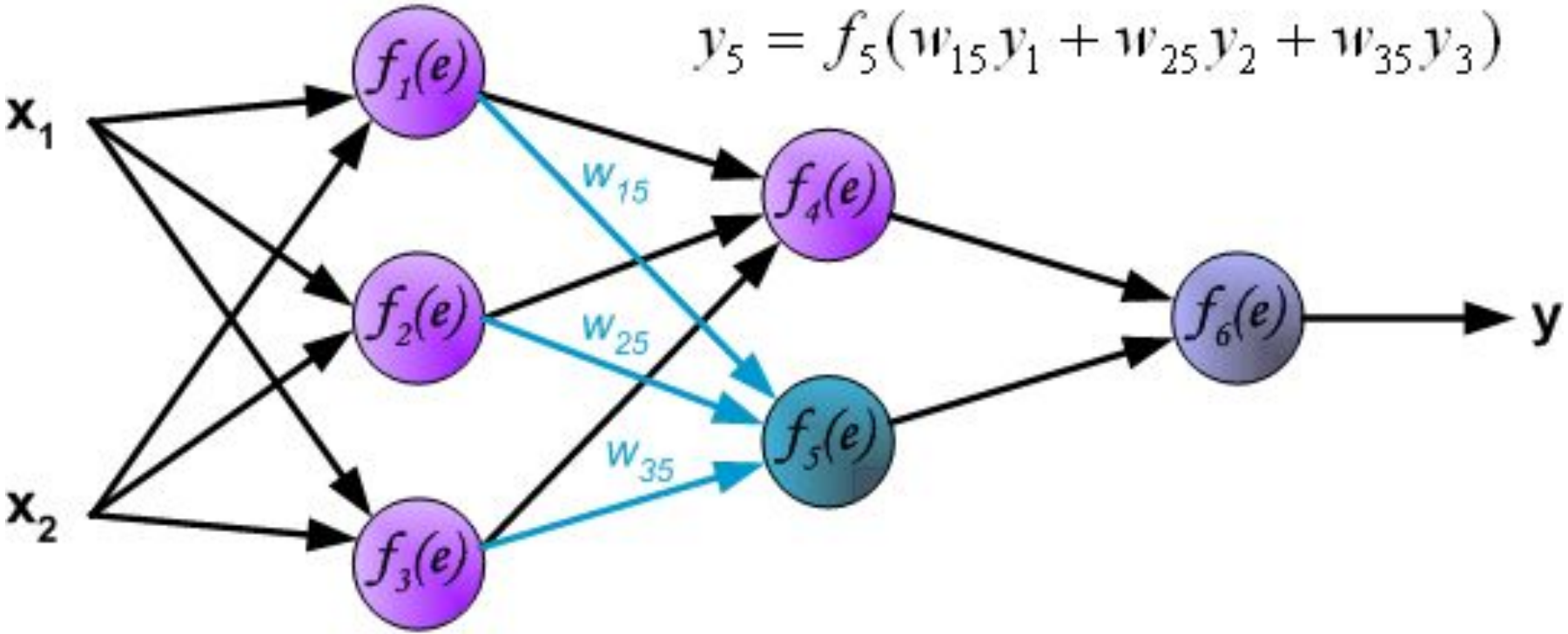
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Forward propagation



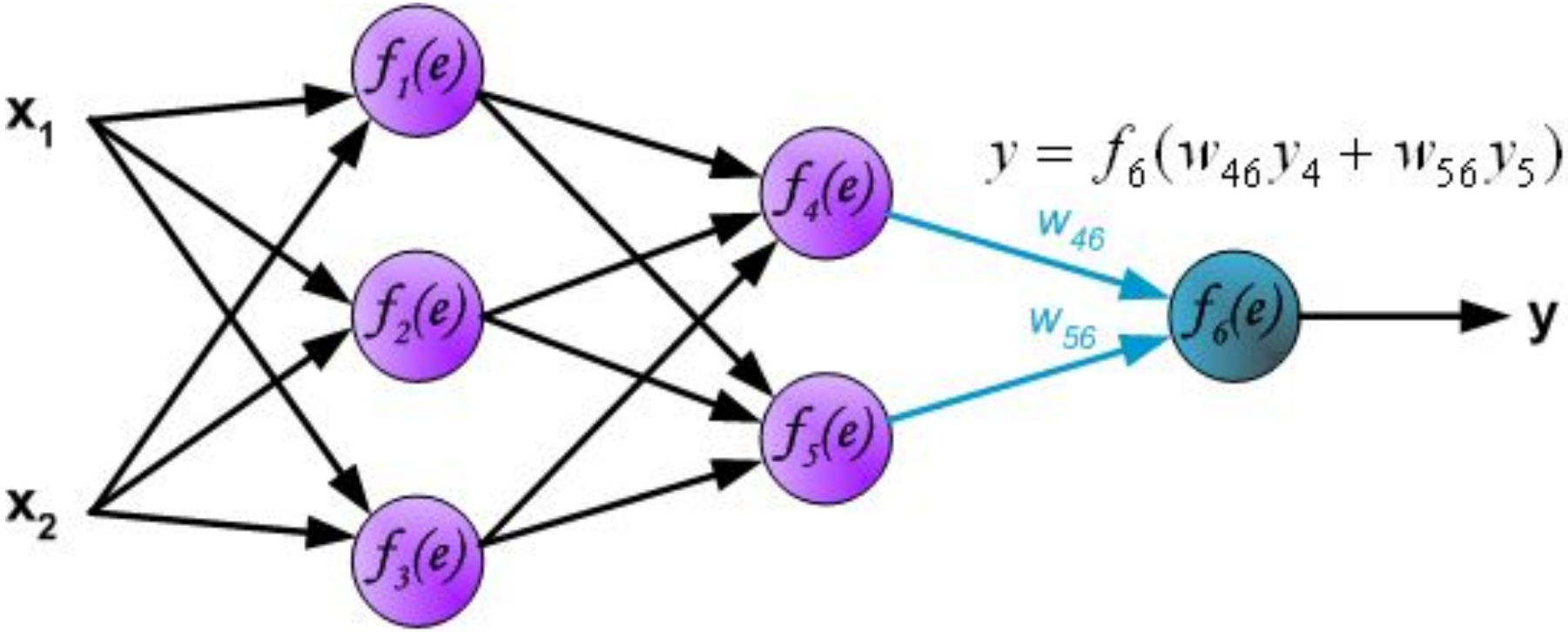
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Forward propagation



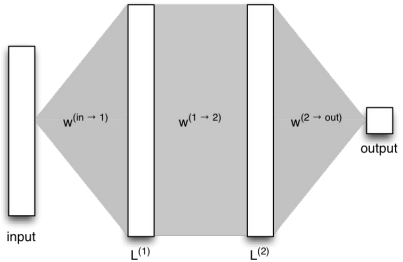
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Forward propagation

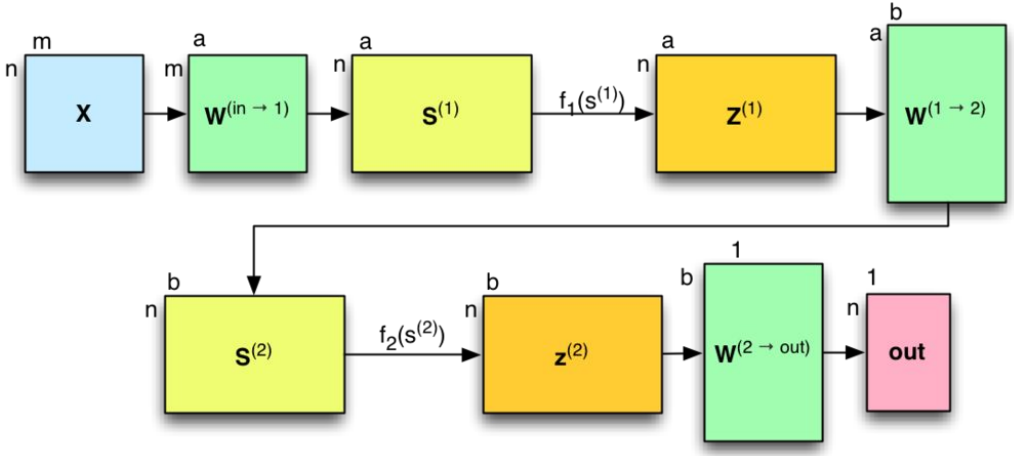


http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Forward propagation: matrix form



В реальности код векторизуется и вычисляется с помощью матричных операций:



$$S^{(1)} = XW^{(in \to 1)}$$
$$Z^{(1)} = f_1(S^{(1)})$$
$$S^{(2)} = Z^{(1)}W^{(1 \to 2)}$$
$$Z^{(2)} = f_2(S^{(2)})$$
$$\hat{y} = f_{out}\left(Z^{(2)}W^{(2 \to out)}\right)$$

http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Механизмы работы
2. Backward propagation

Оптимизационная задача

Цель обучения: например, научиться хорошо классифицировать объекты (задача обучения с учителем, классификация) или предсказывать значения переменной (задача обучения с учителем, регрессия)

Задача обучения: минимизировать некую функцию потерь (Loss function, Cost function, Objective), которая отражает близость к идеалу классификации, регрессии или иной задачи.

С помощью хитрого выбора функции потерь можно решать довольно сложные задачи. В будущем увидим это на примере CTC в seq2seq или на примере переноса стиля изображения.

Функции потерь

В зависимости от типа задачи выбирается архитектура нейросети и в частности функция потерь (http://cs231n.github.io/neural-networks-2/#losses)

Примеры для классификации (http://cs231n.github.io/linear-classify/#loss)

- Multi-class SVM (hinge) loss
  http://www.pyimagesearch.com/2016/09/05/multi-class-svm-loss/

$$L_i = \sum_{j \neq y_i} \max(0, f_j - f_{y_i} + 1)$$

- Multi-class squared SVM (hinge) loss

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)^2$$

- Multi-class cross-entropy (softmax) loss
  http://www.pyimagesearch.com/2016/09/12/softmax-classifiers-explained/

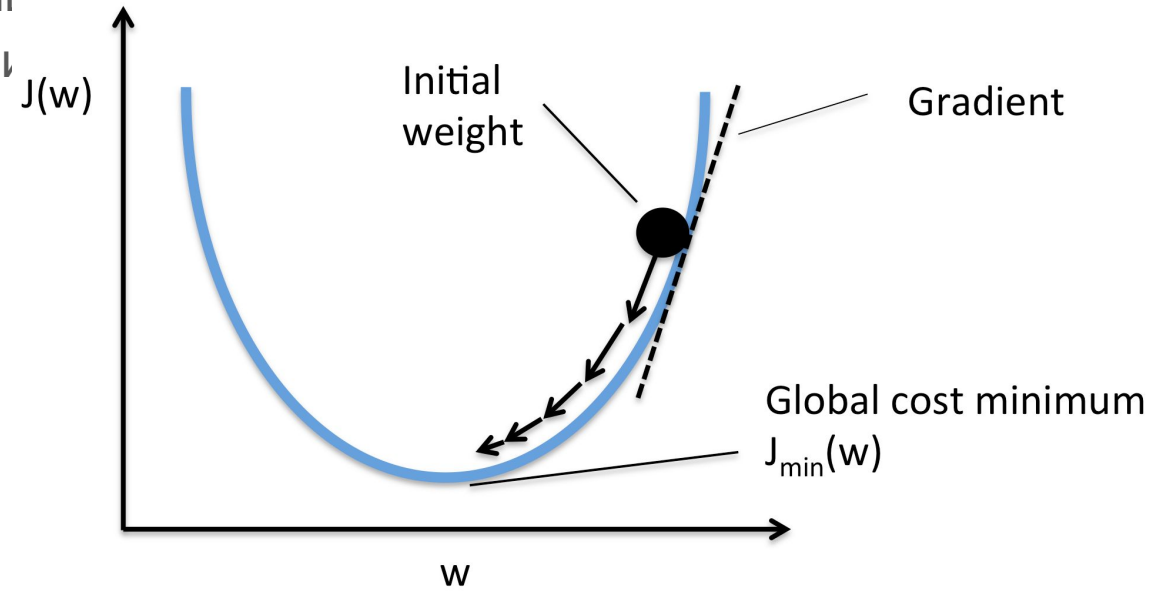
$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

К функции потерь может добавляться регуляризация (L1/L2) для предотвращения переобучения.

Градиентный спуск

Градиентный спуск — это способ оптимизации дифференцируемых функций.

Идея: если мы можем определить направление наибольшего роста функции потерь (а это и есть градиент этой функции), то, двигаясь в
проти ию (что приблизит
нас к



Градиентный спуск

Обычный градиентный спуск (GD или batch gradient descent) подразумевает расчёт градиента по полному датасету. Это довольно дорогая процедура, которая приводит к медленной сходимости алгоритма.

Есть несколько разновидностей градиентного спуска:

- **Stochastic gradient descent (SGD, стохастический градиентный спуск)**: обновления происходят на каждом обучающем примере.
- **Mini-batch gradient descent (MB-GD, градиентный спуск с мини-батчами)**: обновления происходят по некоторому числу обучающих примеров. Промежуточный вариант между обычным и стохастическим градиентным спуском.

http://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

Общий вид алгоритма градиентного спуска

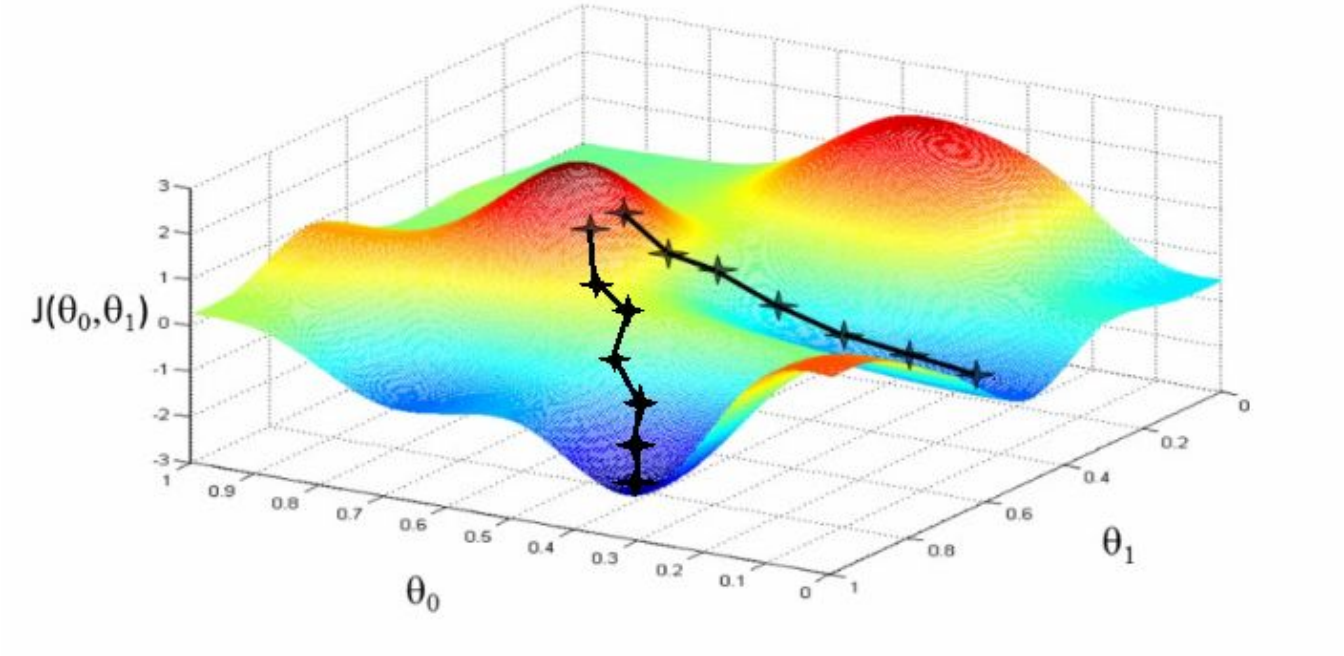
- Задаём параметр η (скорость обучения, learning rate)
- Инициализируем веса нейросети W
- Повторяем пока не достигнут критерий остановки:
  - Получаем мини-батч обучающих примеров <X,y>
  - Вычисляем градиент функции потерь g
  - Корректируем веса $W = W - \eta * g$

Правильный выбор скорости обучения очень важен.

http://www.deeplearningbook.org/contents/optimization.html

Градиентный спуск

В случае нейросетей ландшафт функции потерь очень сложный, поэтому с градиентным спуском много тонкостей.

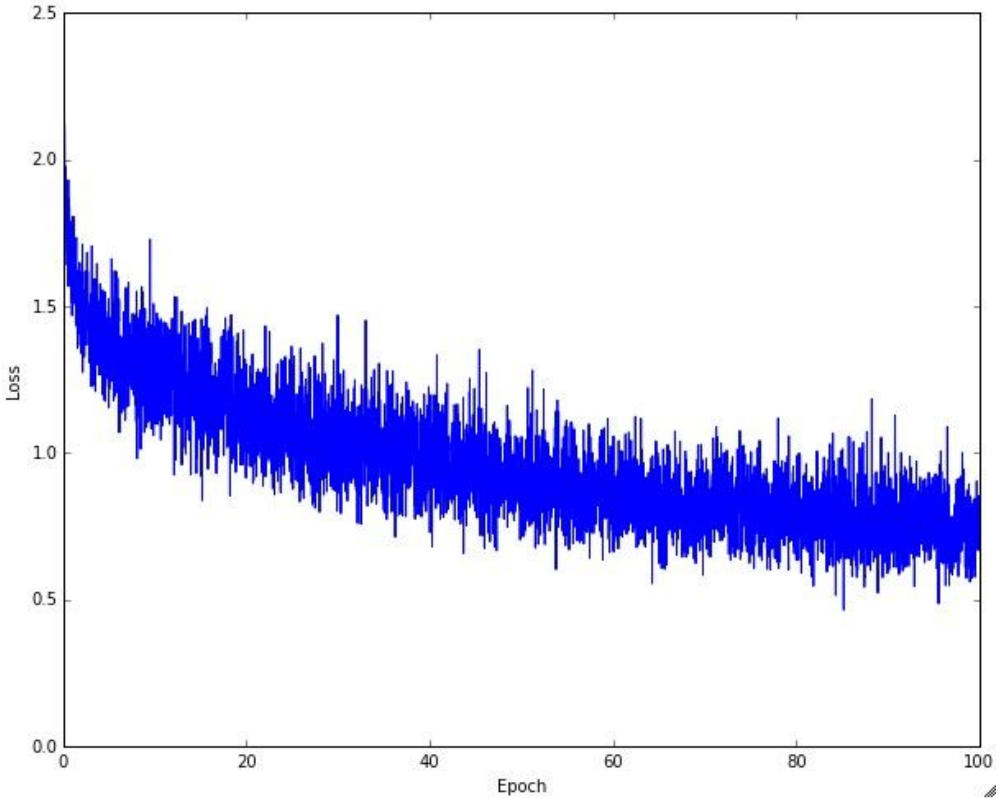


http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

Градиентный спуск

Реальный пример уменьшения со временем функции потерь при обучении нейросети градиентным спуском.

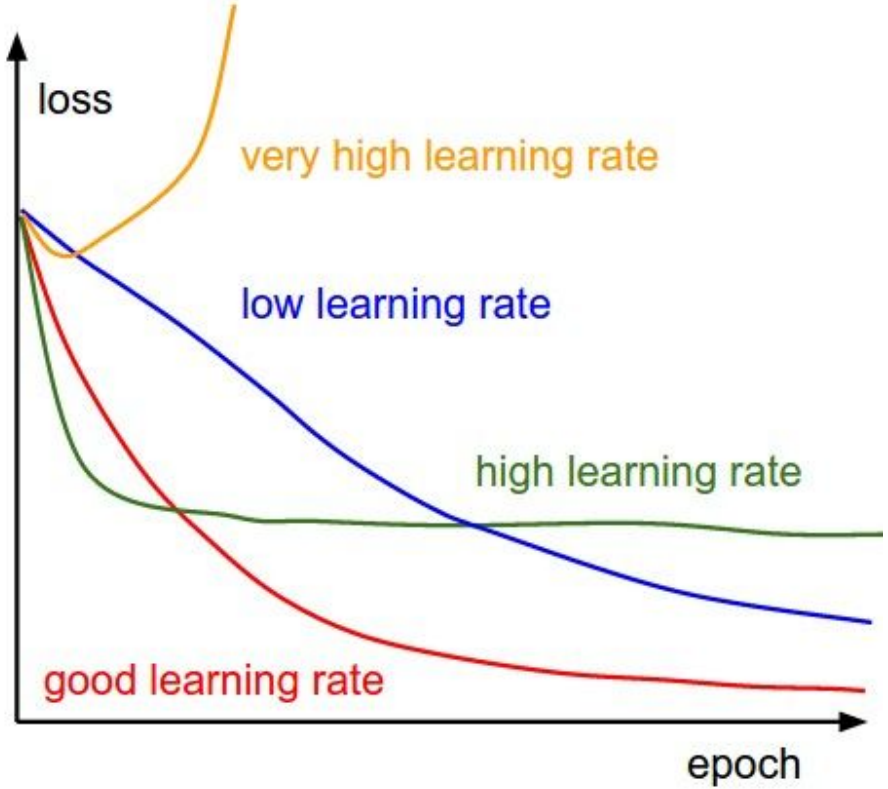
Эпоха = один полный цикл обучения нейросети на всех примерах обучающей выборки.



http://cs231n.github.io/neural-networks-3/

Градиентный спуск

Важность правильного выбора параметра скорости обучения (learning rate) и влияние его на скорость обучения нейросети.

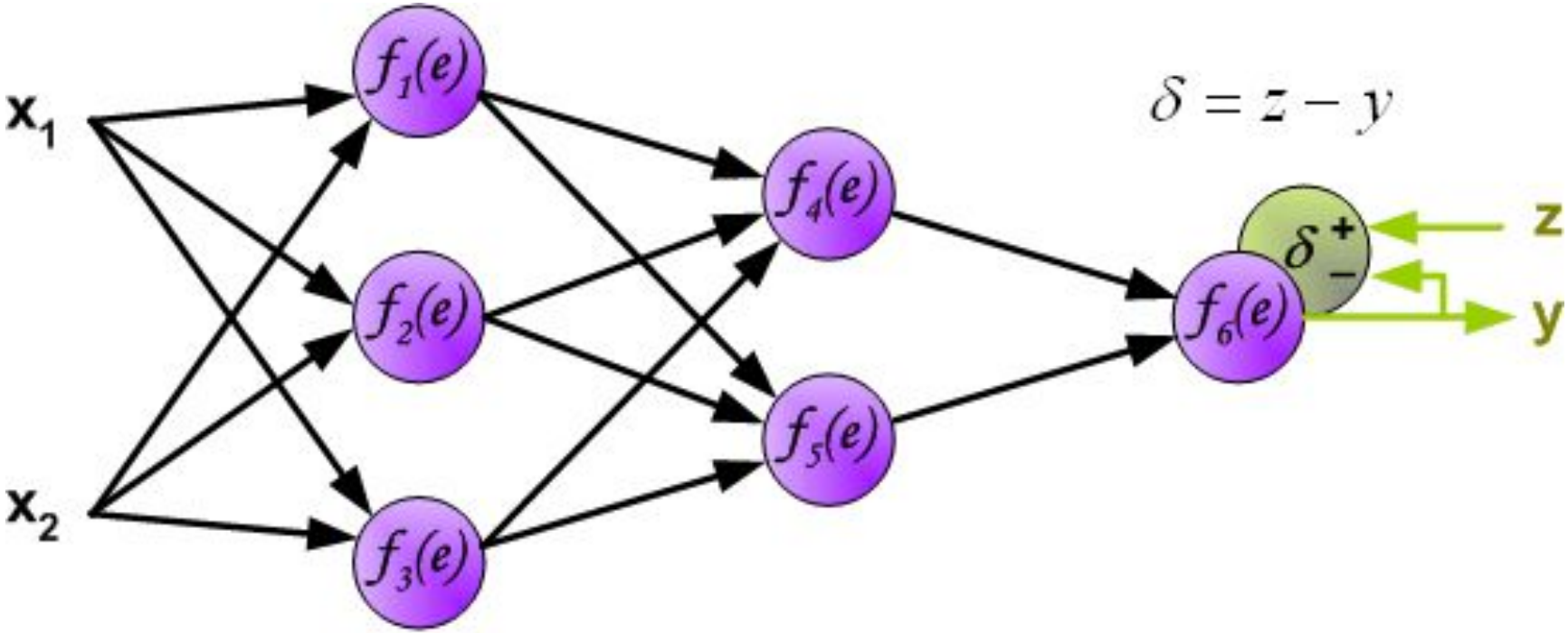


http://cs231n.github.io/neural-networks-3/

Алгоритм back-propagation

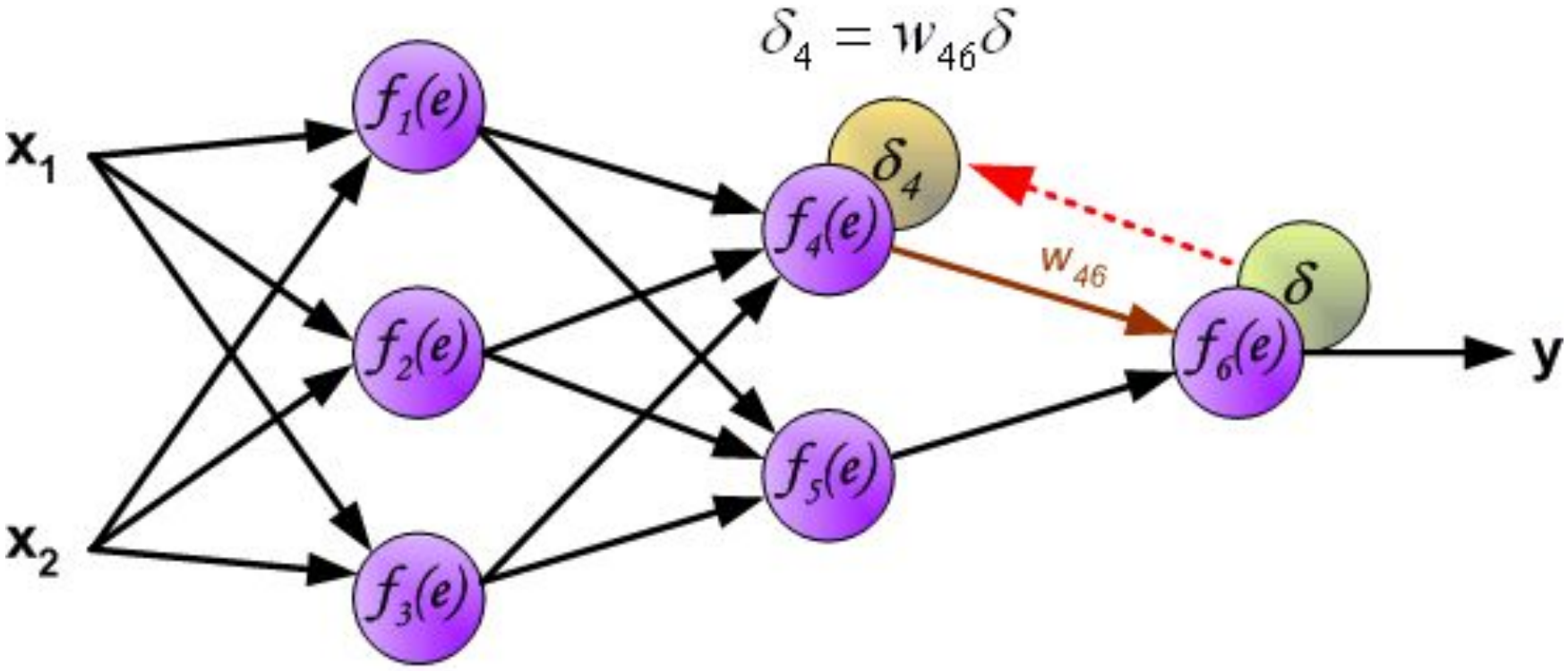
Алгоритм обратного распространения ошибки (back-propagation, backprop) — это способ расчёта градиента ошибки по каждому из весов в нейросети, чтобы затем каждый вес скорректировать в соответствии с алгоритмом градиентного спуска.

Backward propagation



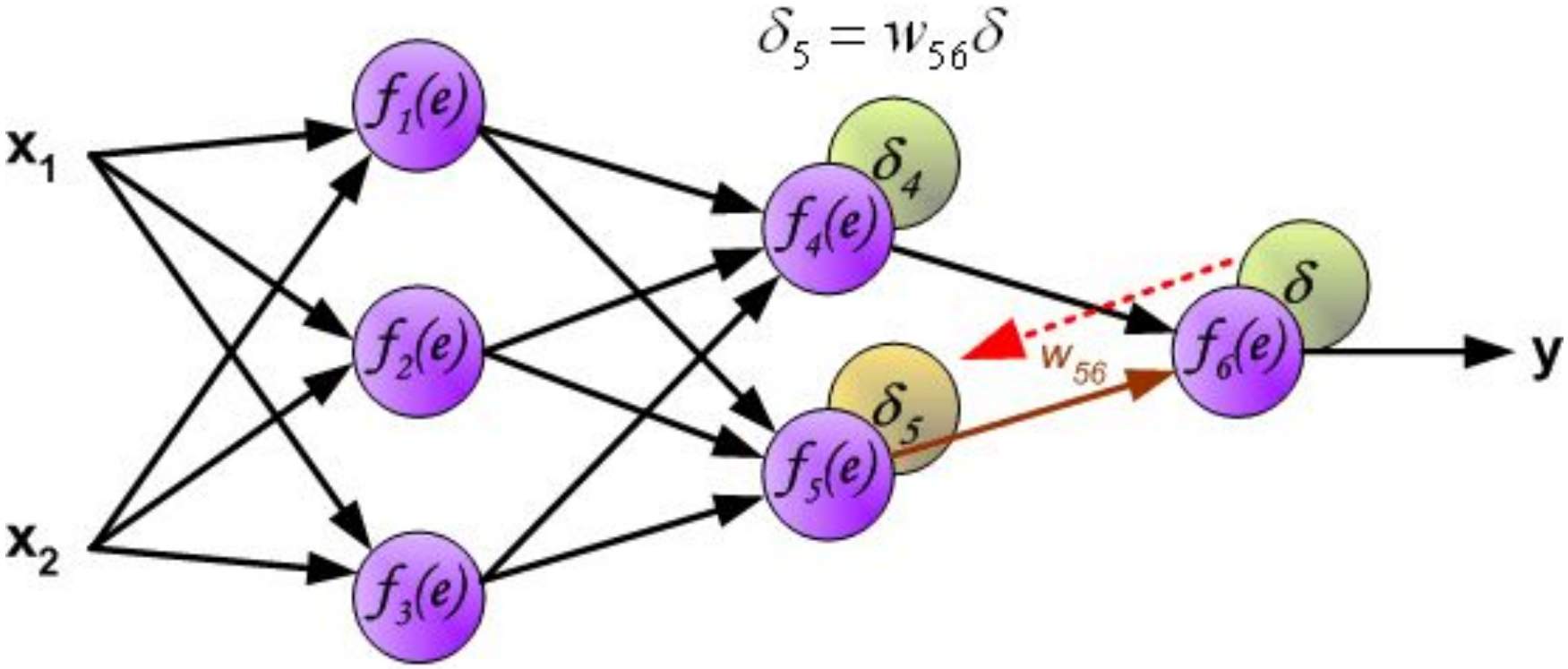
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation



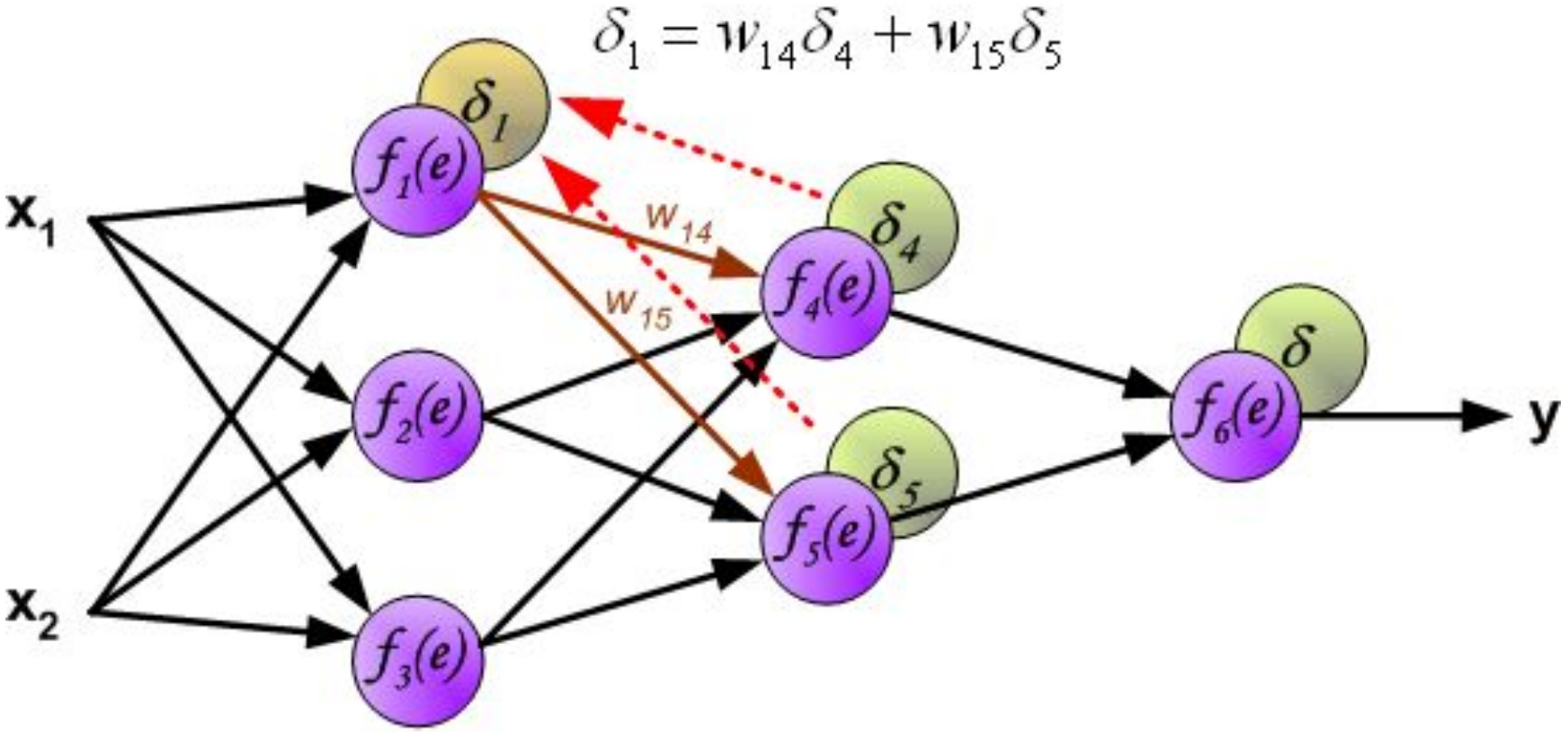
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation



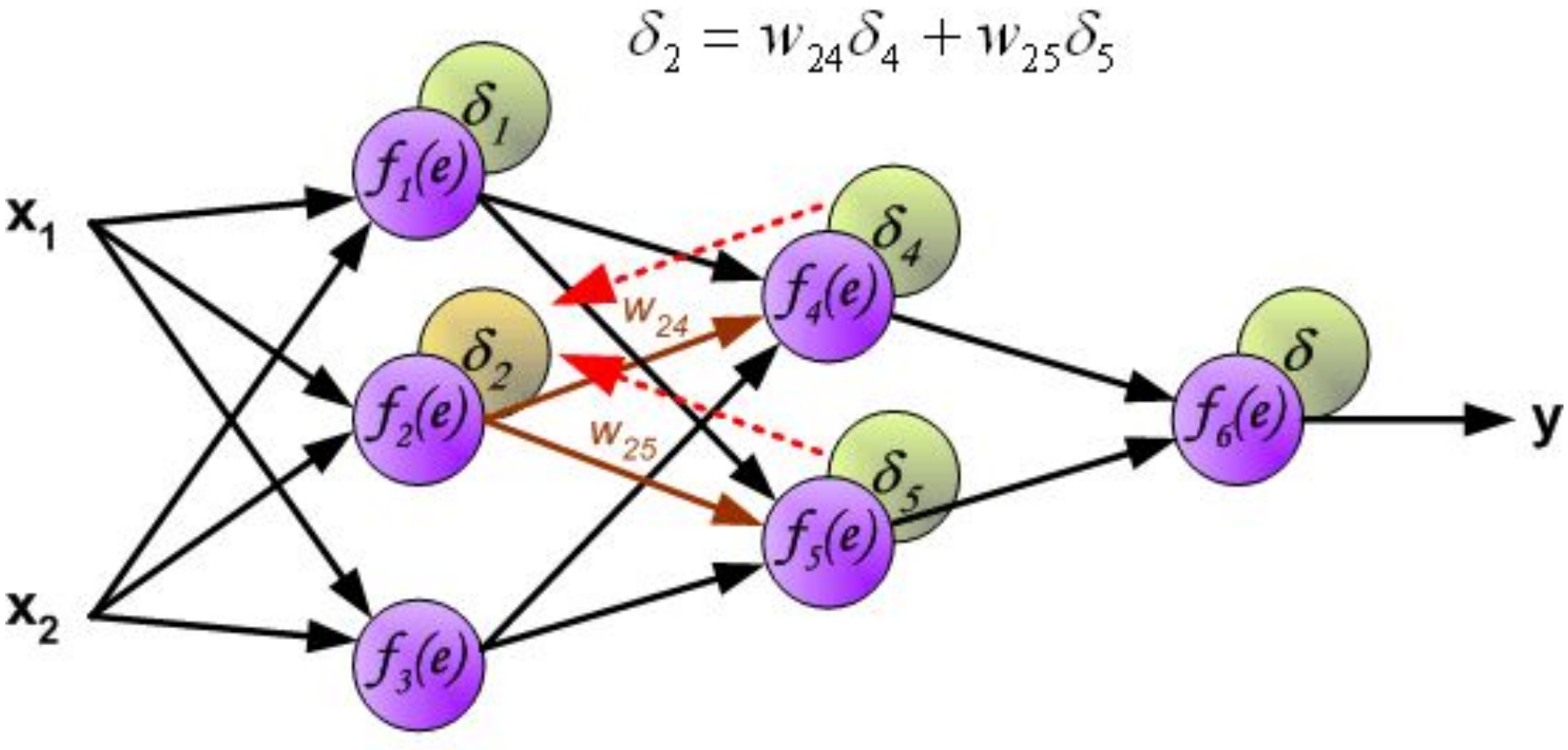
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation



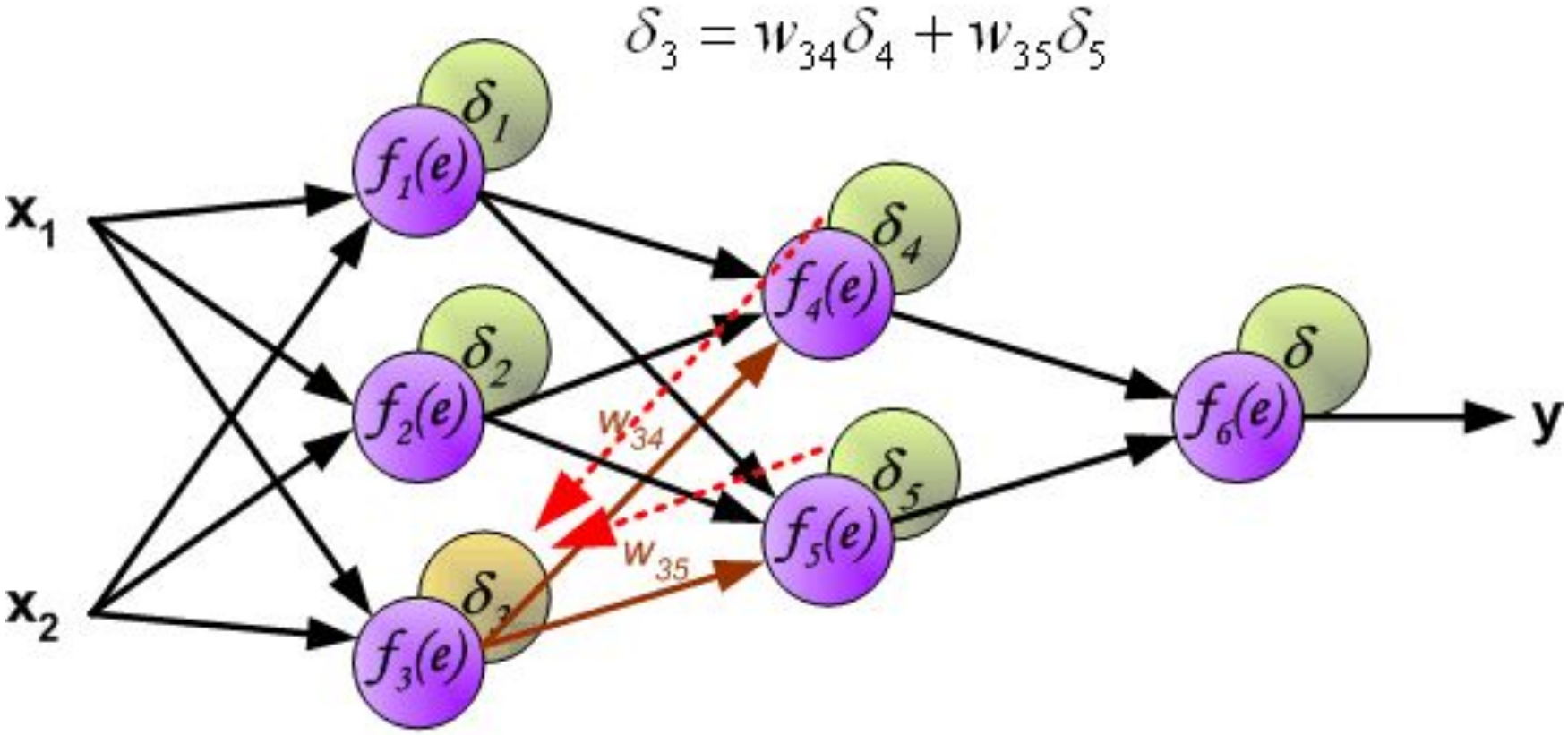
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation



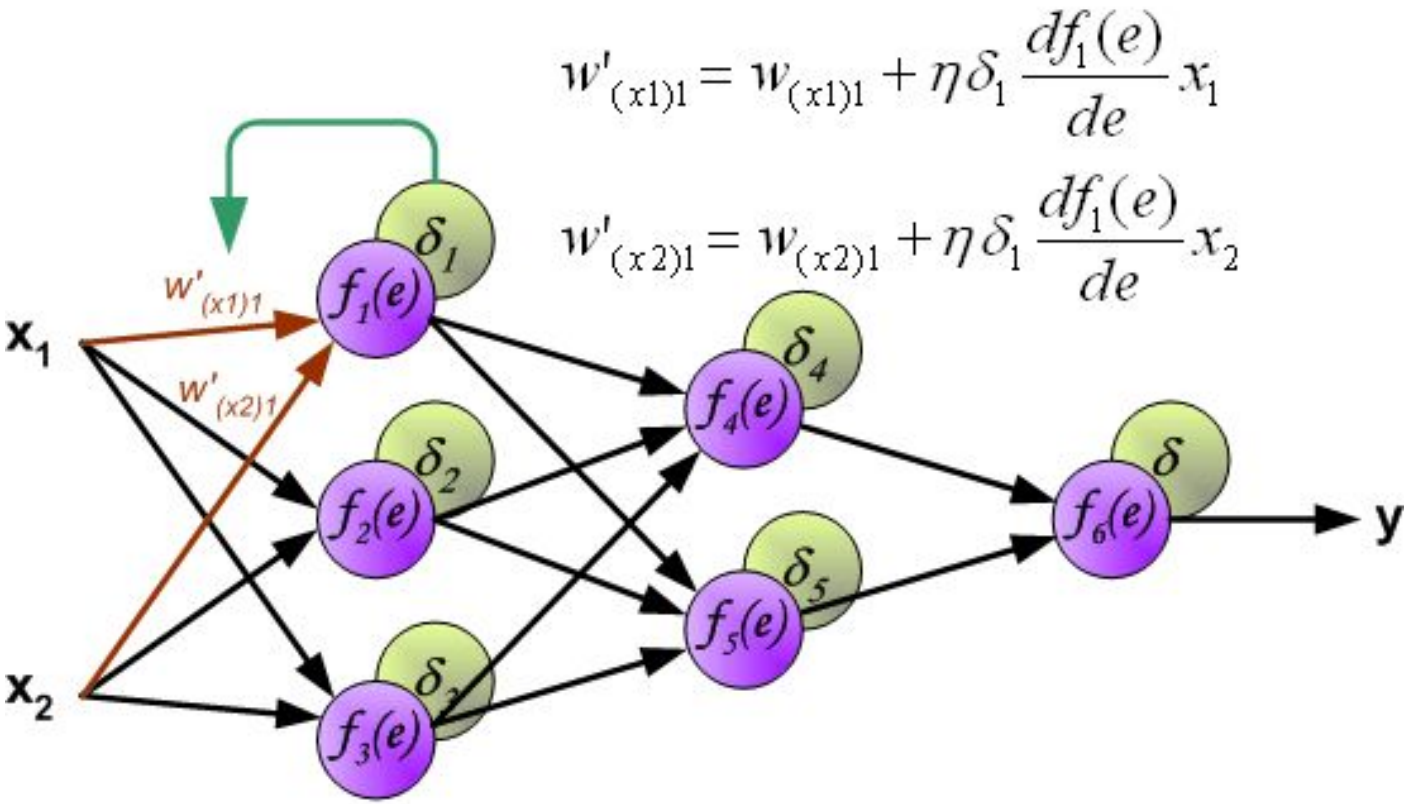
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation



http://prog3.com/sbdm/blog/nan_xiaoyang/article/details/50321873

Backward propagation

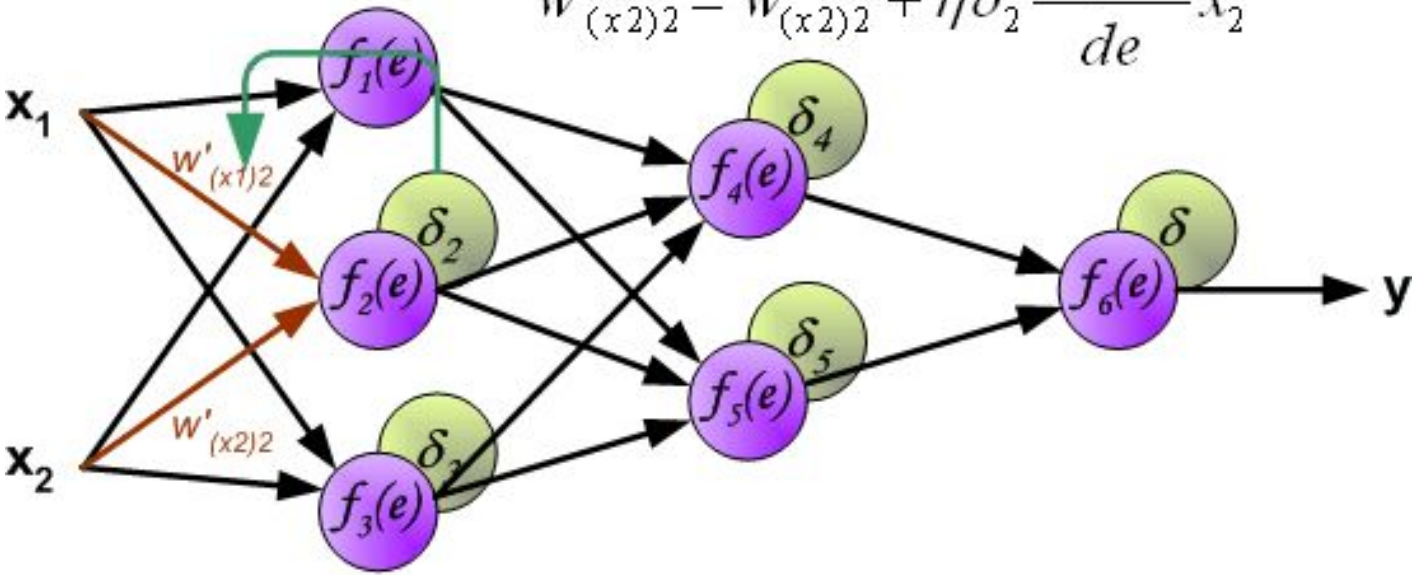


http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

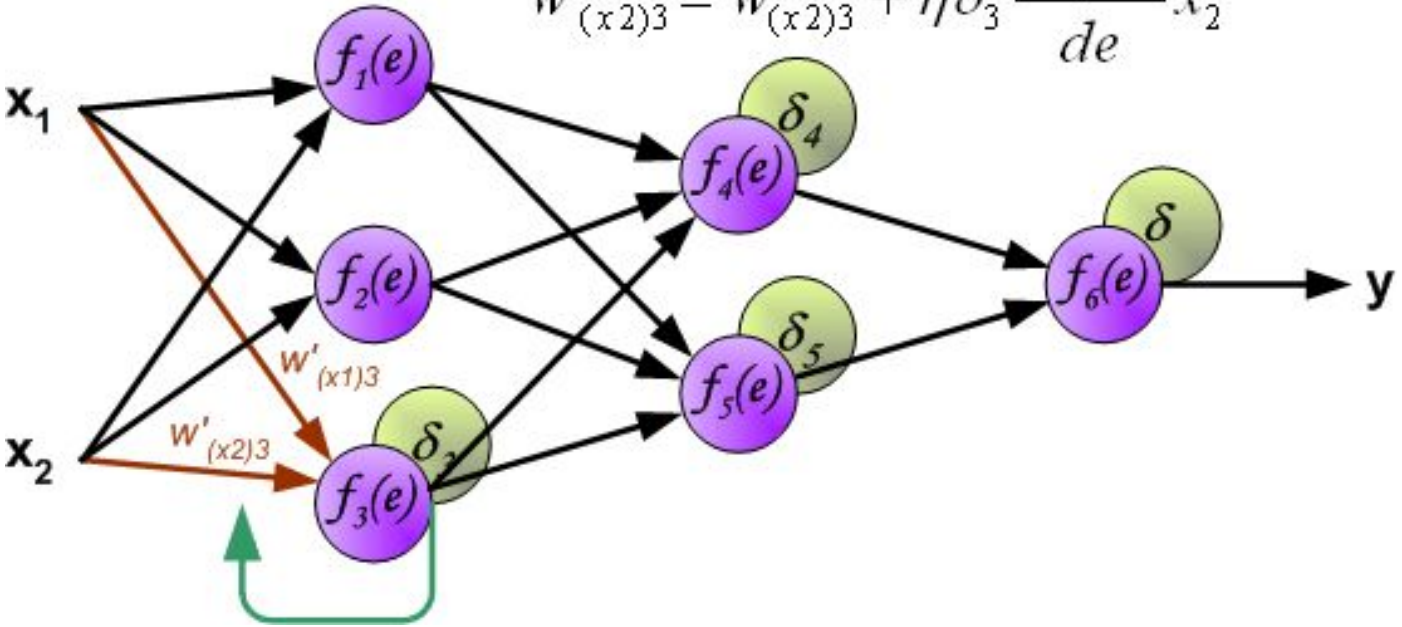


http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation

$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$



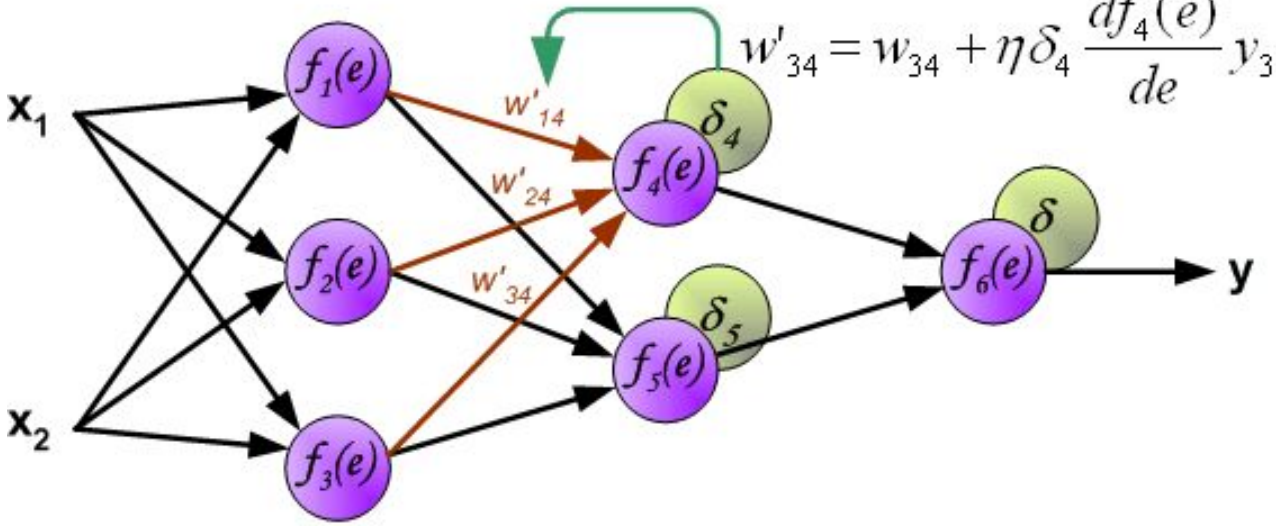
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$



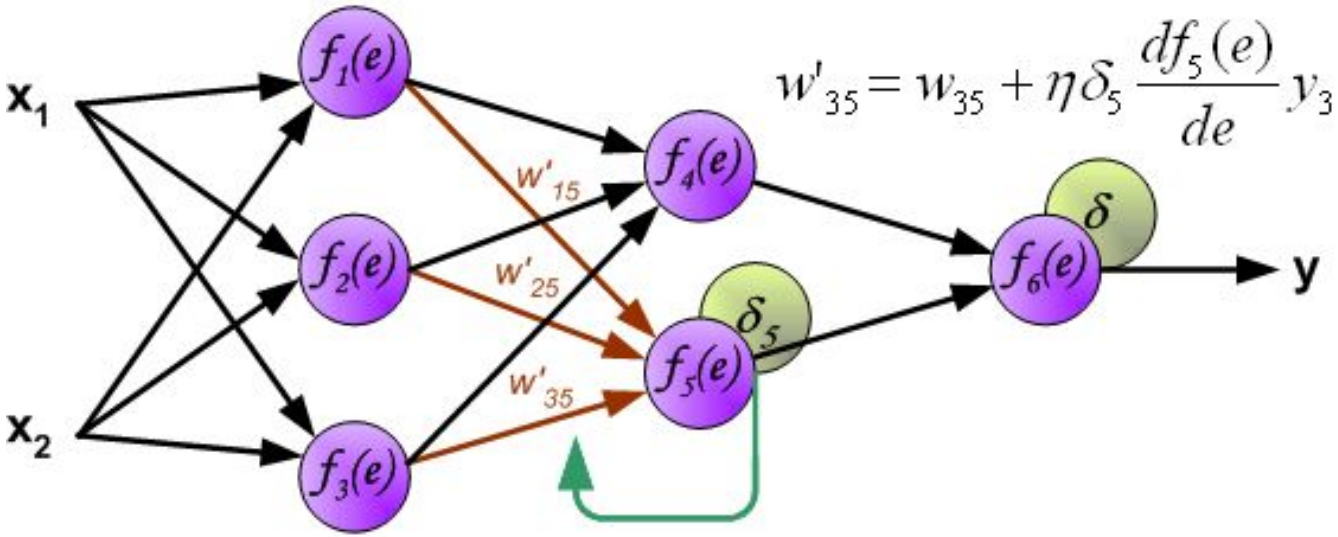
http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation

$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$

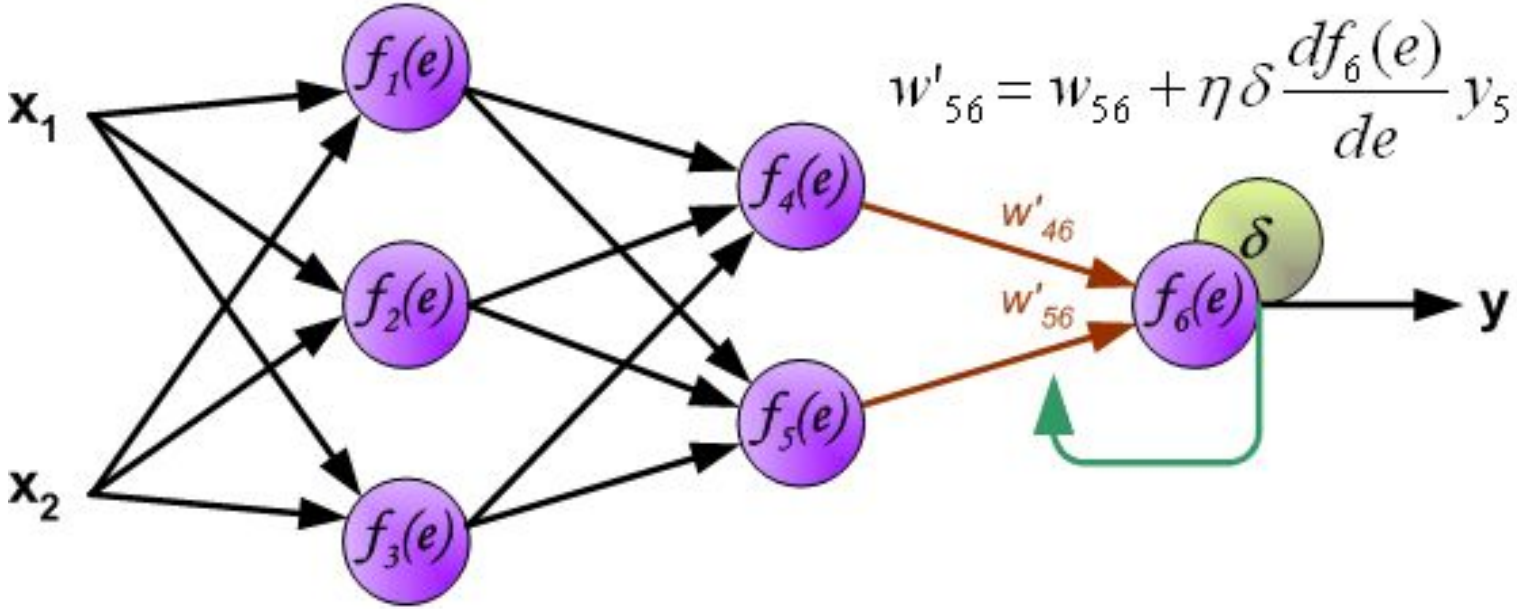


http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

Backward propagation

$$w'_{46} = w_{46} + \eta\delta\frac{df_6(e)}{de}y_4$$

$$w'_{56} = w_{56} + \eta\delta\frac{df_6(e)}{de}y_5$$



http://prog3.com/sbdm/blog/han_xiaoyang/article/details/50321873

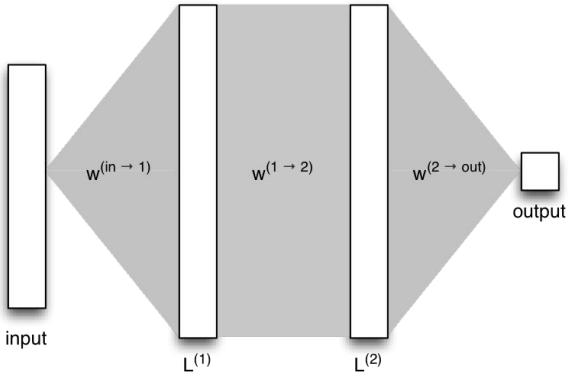
Backward propagation: matrix form

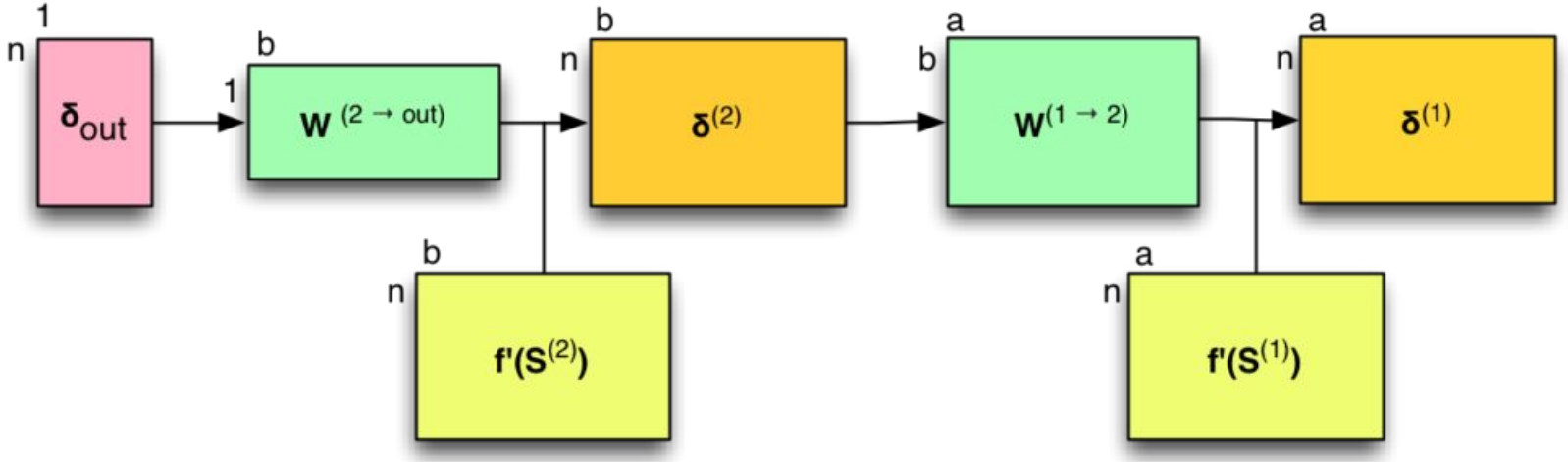
$$D^{(out)} = (Z^{(out)} - Y)^T$$

$$D^{(i)} = F^{(i)} \odot W^{(i)} D^{(j)}$$

$$\Delta W^{(in \to 1)} = -\eta (D^{(1)} X)^T$$

$$\Delta W^{(i \to j)} = -\eta (D^{(j)} Z^{(i)})^T$$





http://briandolhansky.com/blog/2014/10/30/artificial-neural-networks-matrix-form-part-5

Дифференцирование

Ещё несколько лет назад программирование нейросети заключалось в ручном задании всех слоёв, функций активации и функции потерь, аналитического расчёта их производных, а также проверки корректности этих производных с помощью численного дифференцирования. Эта процедура была подвержена ошибкам и имела высокий порог входа.

В последние годы необходимости в ручном задании производных нет, так как большинство библиотек поддерживают один из двух вариантов автоматизации этого процесса:

- Символьное дифференцирование (Theano)
- Автоматическое дифференцирование (Tensorflow, Torch)

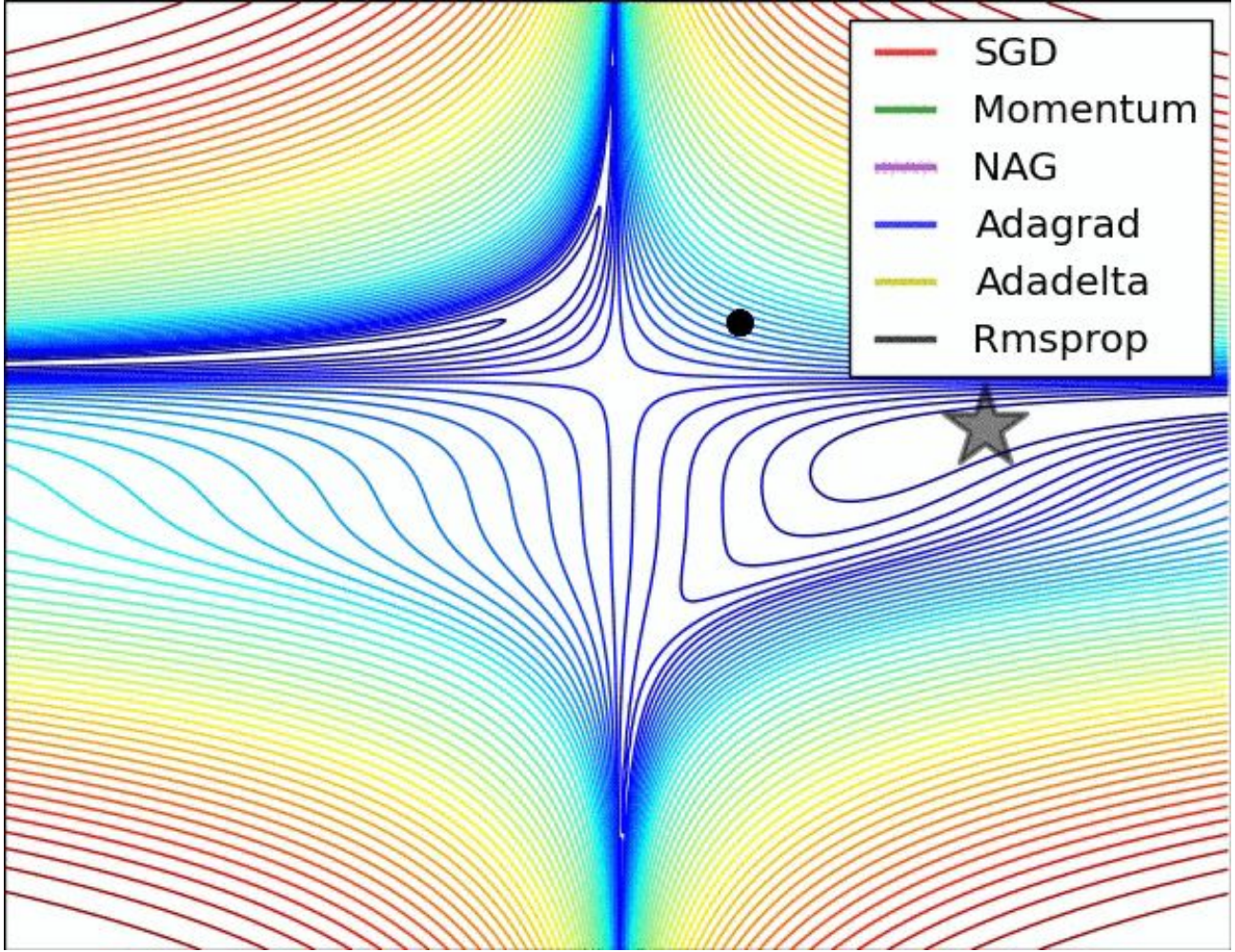
https://domluna.me/automatic-differentiation-a.k.a-please-save-me-from-backprop/

Вариации и улучшения backprop

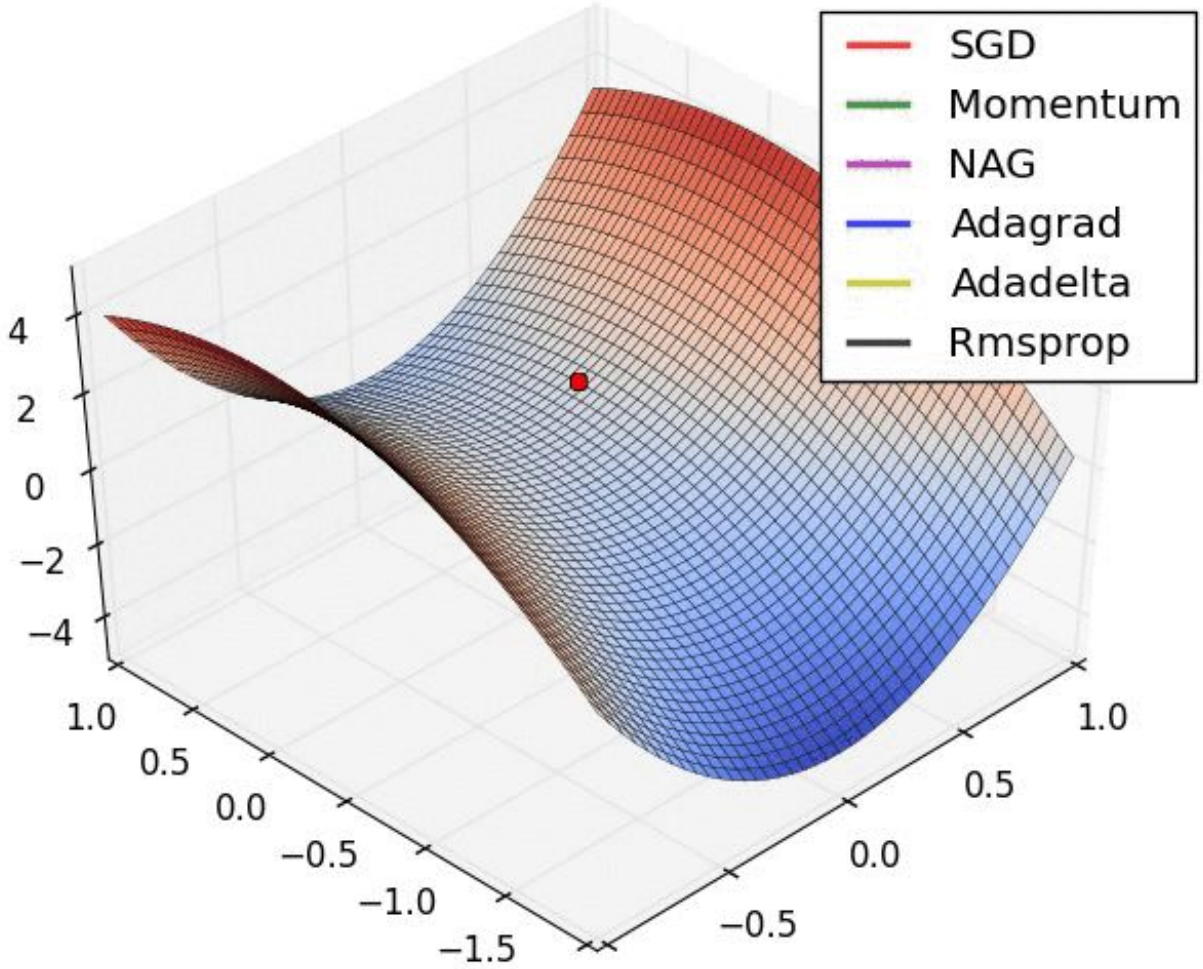
Существует множество модификаций backprop

- Изменяемый learning rate (постепенное уменьшение, уменьшение на порядок каждые N эпох, …)
- Использование момента инерции (momentum update, Nesterov momentum)
- Адаптивные методы (Adagrad, Adadelta, RMSprop, Adam, …)
- Методы второго порядка (L-BFGS, Conjugate gradient, Hessian Free)

http://cs231n.github.io/neural-networks-3/

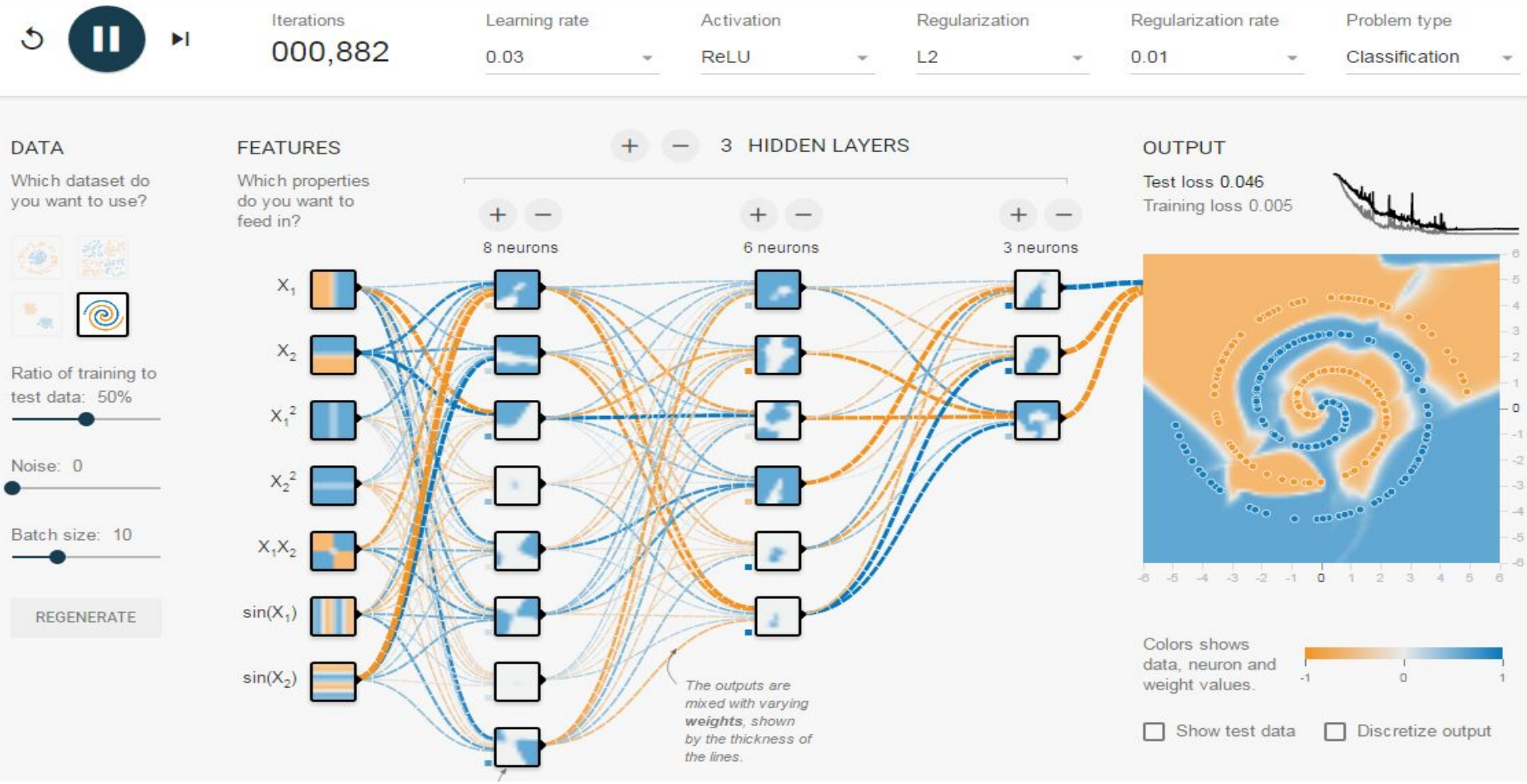


http://cs231n.github.io/neural-networks-3/



http://cs231n.github.io/neural-networks-3/

Демонстрация работы

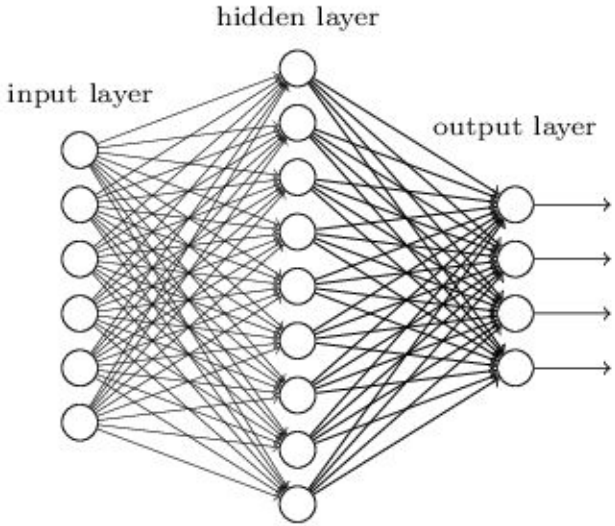


http://playground.tensorflow.org/

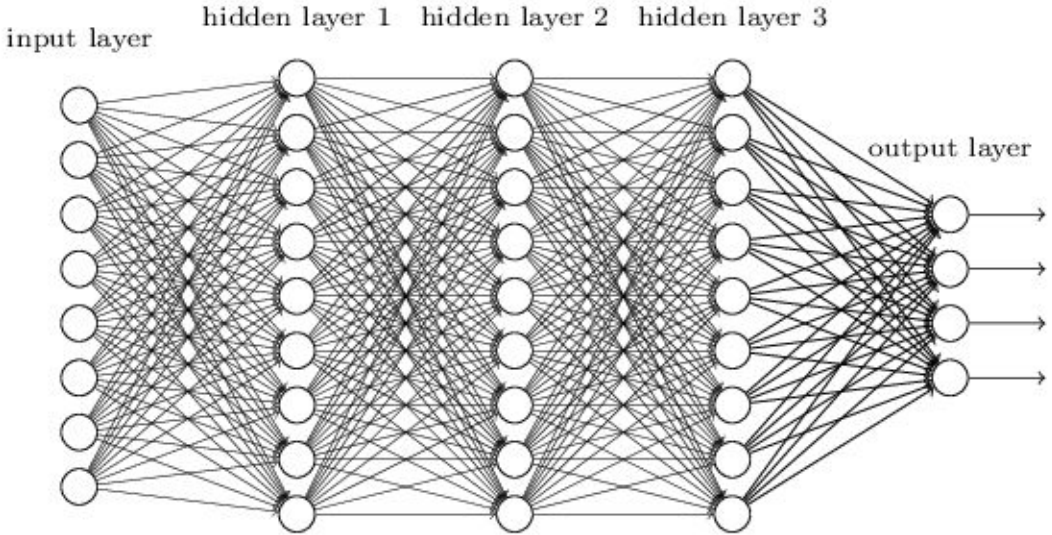
Классические нейросети прямого распространения (Feed-Forward Neural Networks, FNN)

Полносвязная нейросеть

"Non-deep" feedforward neural network



Deep neural network

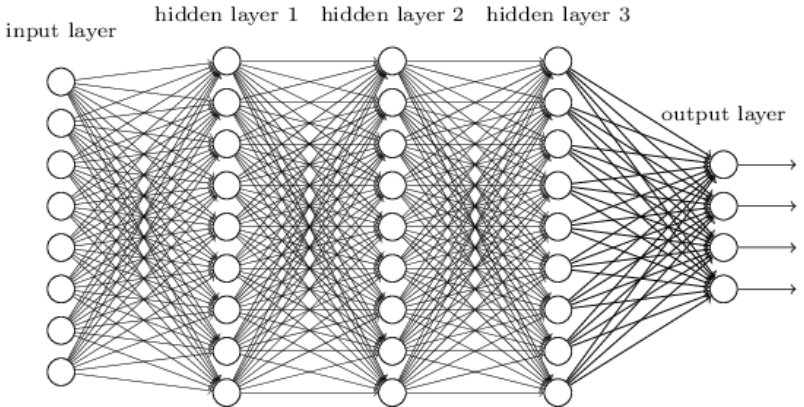


Multilayer Perceptron (MLP)

Классика FNN. Хорошо работают для классификации, но есть трудности:

- Много параметров
  - Для сети, у которой на входе картинка 100x100, три скрытых слоя по 100 нейронов каждый, и выходом на 10 классов, число параметров будет примерно 1M
    (10000*100 + 100*100 + 100*100 + 100*10)
- Затухающие градиенты (если слоёв много)

Как следствие — трудно обучать.



Example: Neural Net in Spark ML

```
from pyspark.ml.classification import
MultilayerPerceptronClassifier,
MultilayerPerceptronClassificationModel

mlp = MultilayerPerceptronClassifier(
   maxIter=100, layers=[4125, 10, 10, 2])

model_mlp = mlp.fit(trainingData)

predictions = model_mlp.transform(testData)
```

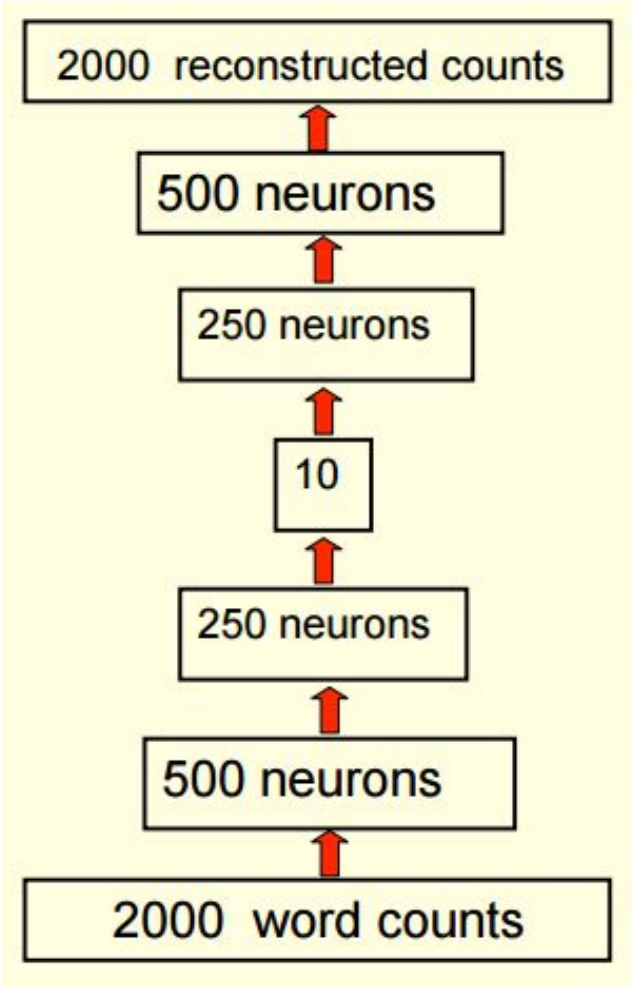
https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier

Вариации FNN: Автоэнкодер (AE)

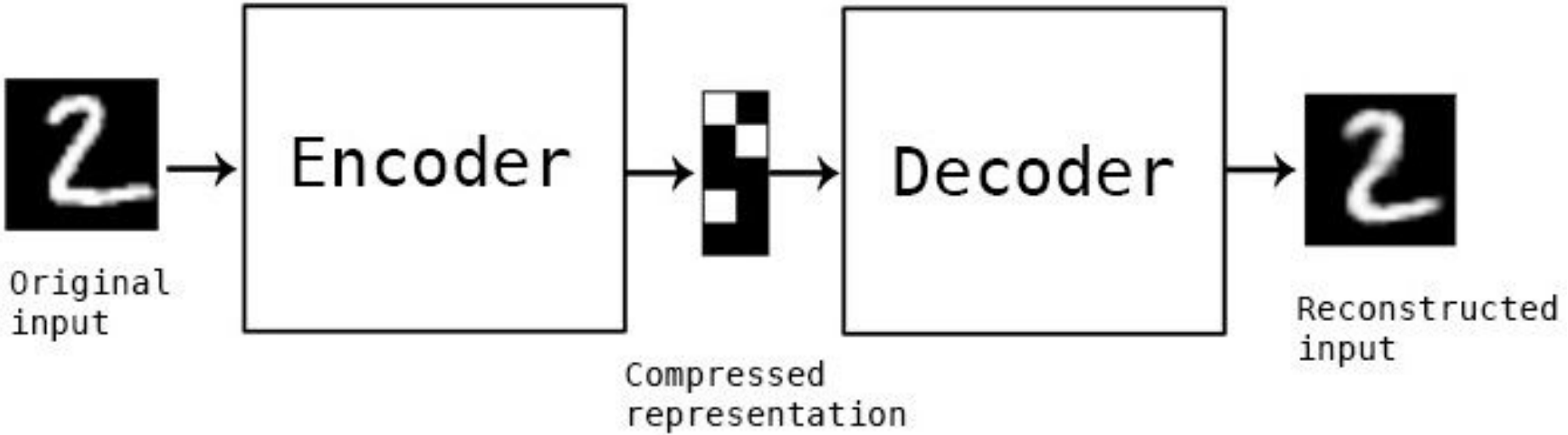
Учится создавать компактное описание входных данных.

Используется для уменьшения размерности и получения новых высокоуровневых признаков.

Может быть глубоким (многослойным).

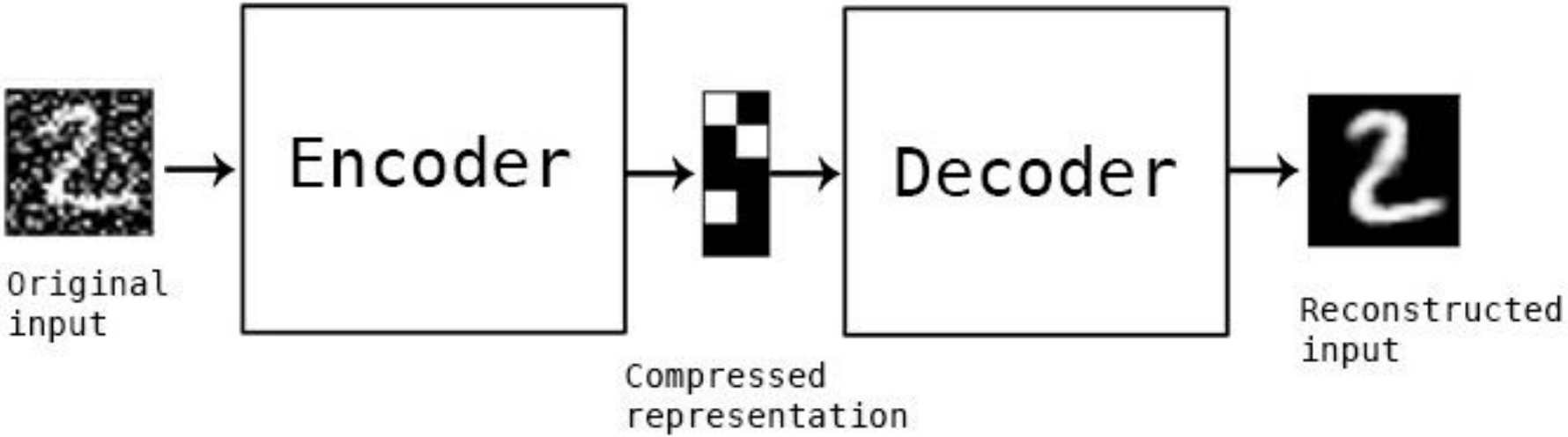


Autoencoder



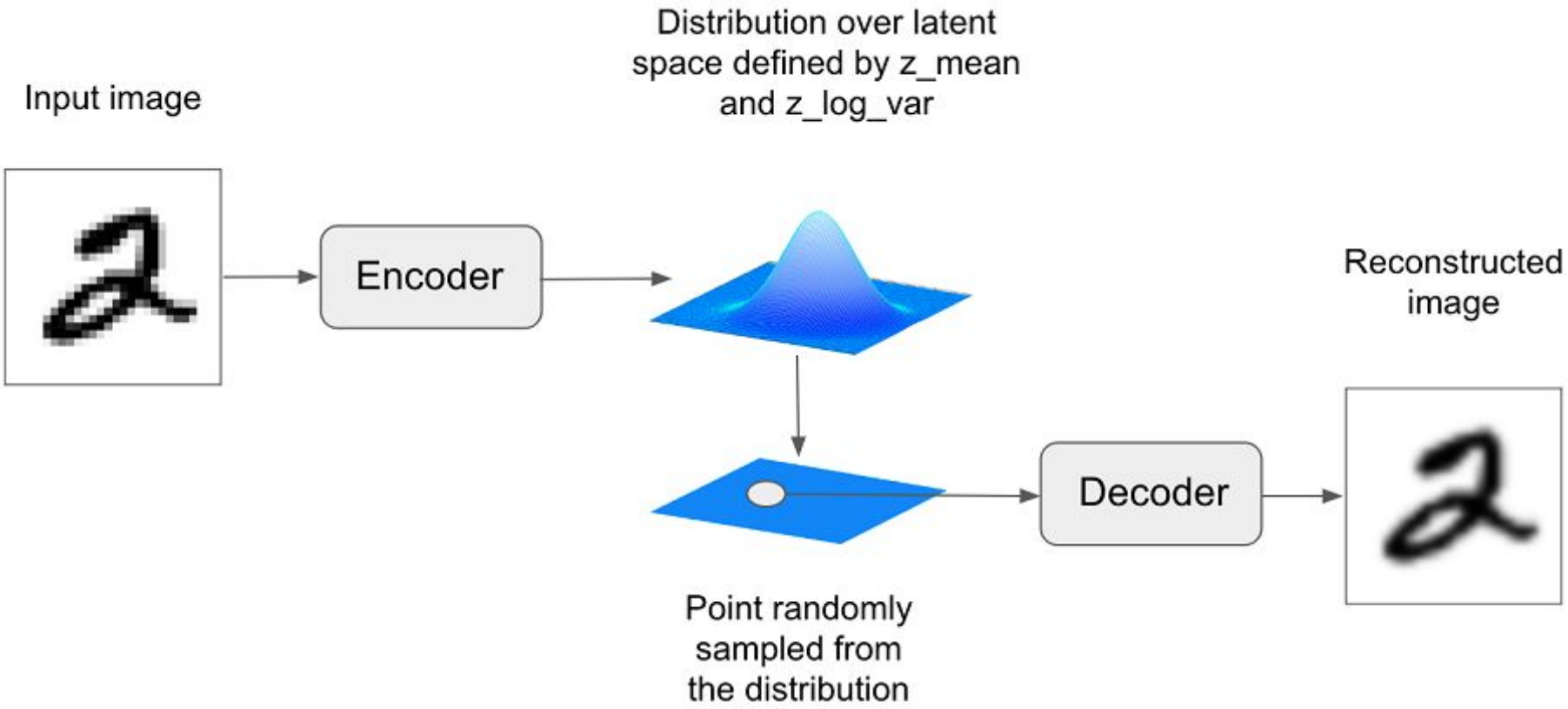
https://www.manning.com/books/deep-learning-with-python

Denoising Autoencoder



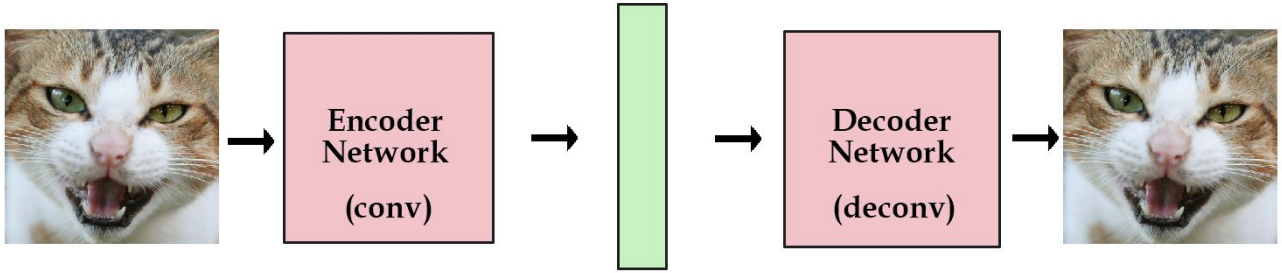
https://www.manning.com/books/deep-learning-with-python

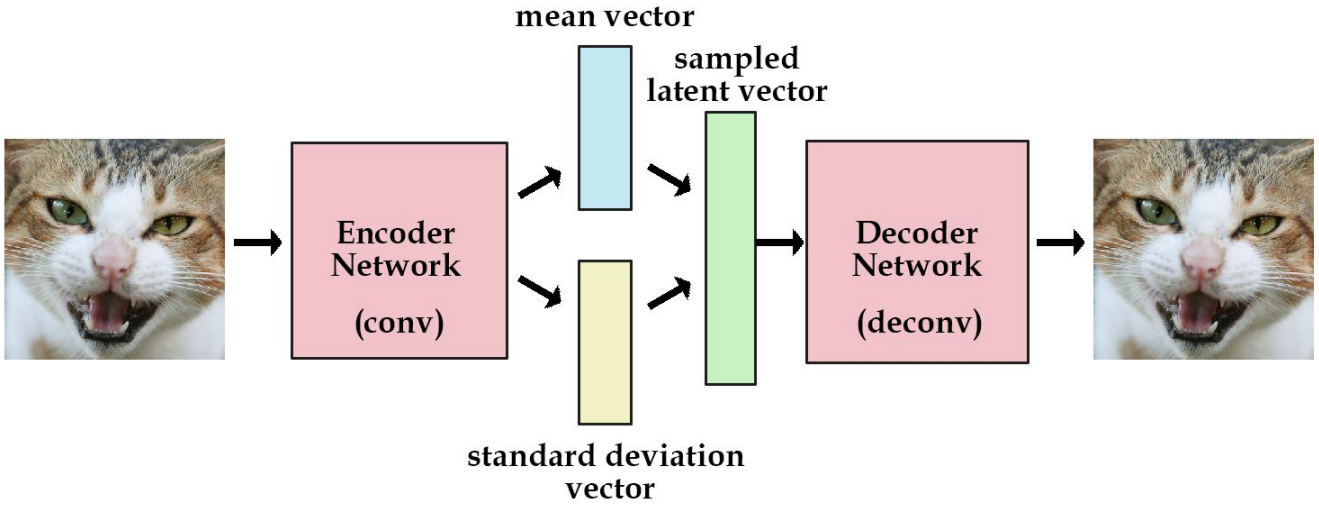
Variational Autoencoder (VAE)



https://www.manning.com/books/deep-learning-with-python

AE vs. VAE





http://kvfrans.com/variational-autoencoders-explained/

VAE on images

https://www.manning.com/books/deep-learning-with-python

VAE on images and the “smile vector”

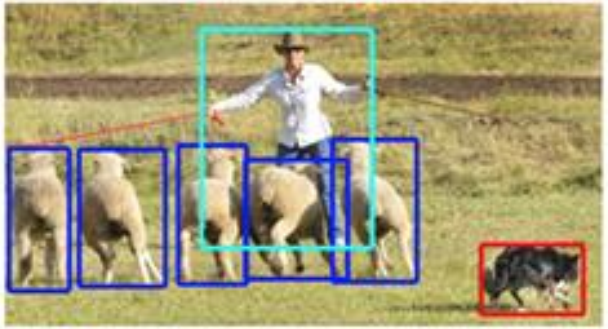
https://www.manning.com/books/deep-learning-with-python

Свёрточные нейросети

Convolutional Neural Networks, CNN

Классические задачи для CNN

(a) classification



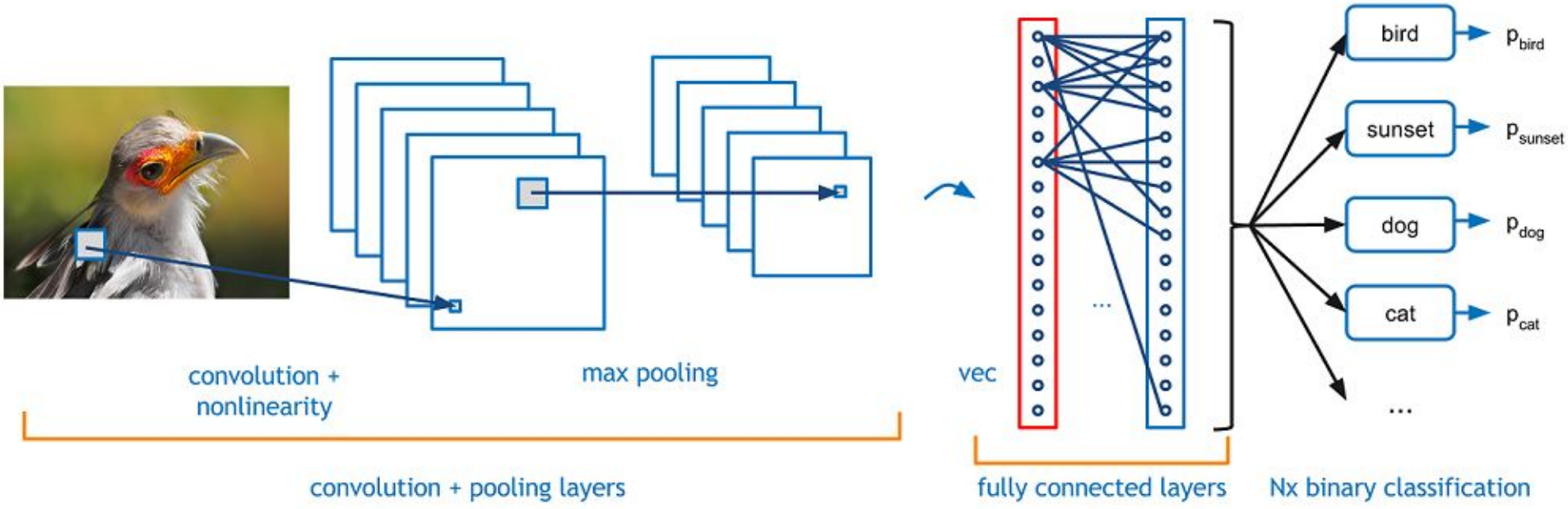
(b) detection

(c) segmentation

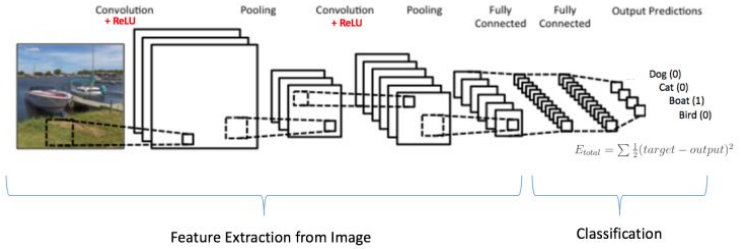
https://research.facebook.com/blog/learning-to-segment/

Свёрточная нейросеть: общий вид

Свёрточная нейросеть (CNN) — это Feed-Forward сеть специального вида:



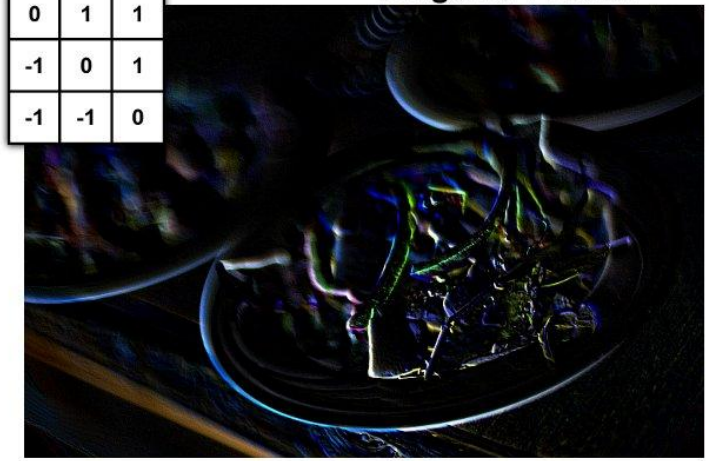
Состав CNN



- **Свёрточные слои**: каждая плоскость в свёрточном слое — это один нейрон, реализующий операцию свёртки (convolution) и являющийся матричным фильтром небольшого размера (например, 5x5).
- **Слои субдискретизации** (subsampling, spatial pooling): уменьшают размер изображения (например, в 2 раза).
- **Полносвязные слои** (MLP) на выходе модели (используются для классификации).

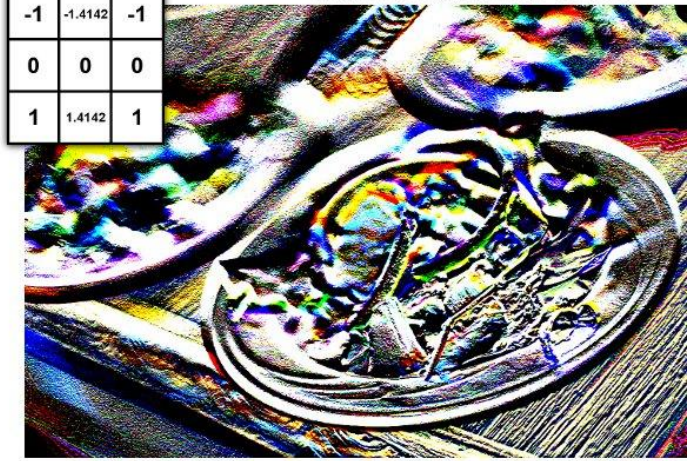
Diagonal Prewitt

0	1	1
-1	0	1
-1	-1	0



Horizontal Frei-Chen

-1	-1.4142	-1
0	0	0
1	1.4142	1

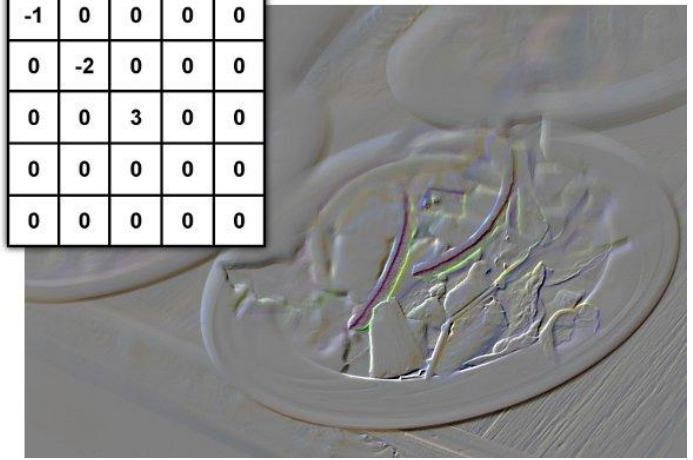


Sharpen

0	-1	0
-1	5	-1
0	-1	0

Emboss

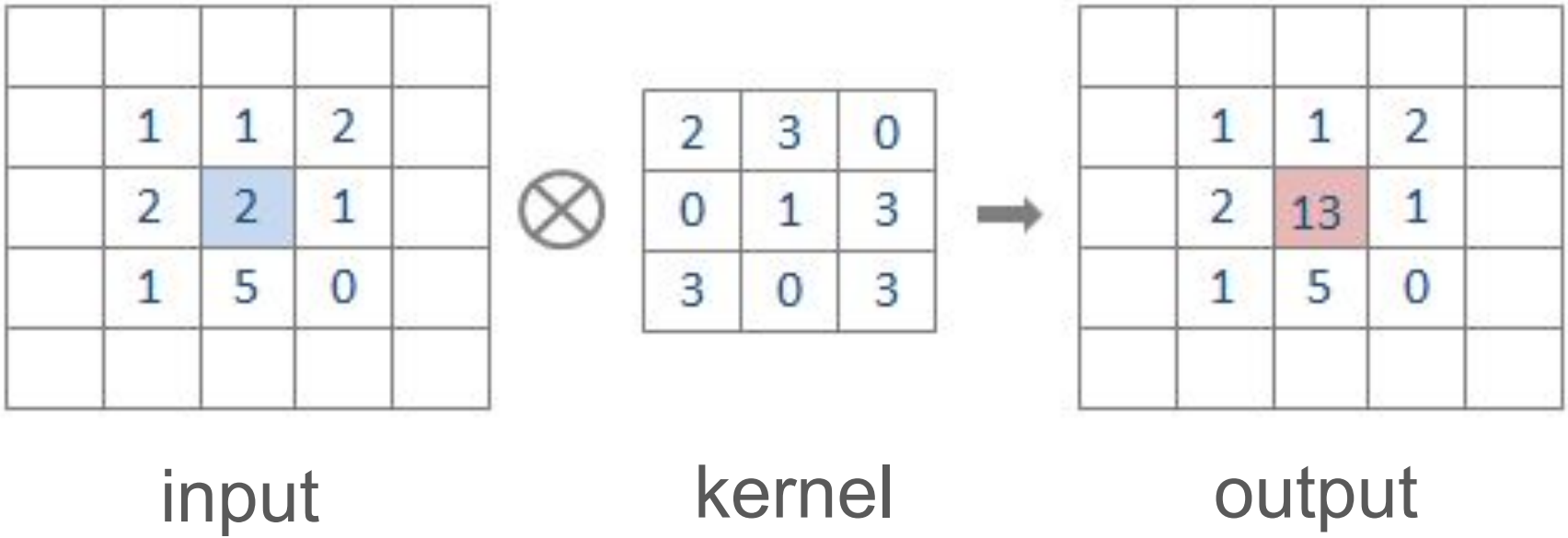
-1	0	0	0	0
0	-2	0	0	0
0	0	3	0	0
0	0	0	0	0
0	0	0	0	0



http://www.gimpbible.com/files/convolution-matrix/

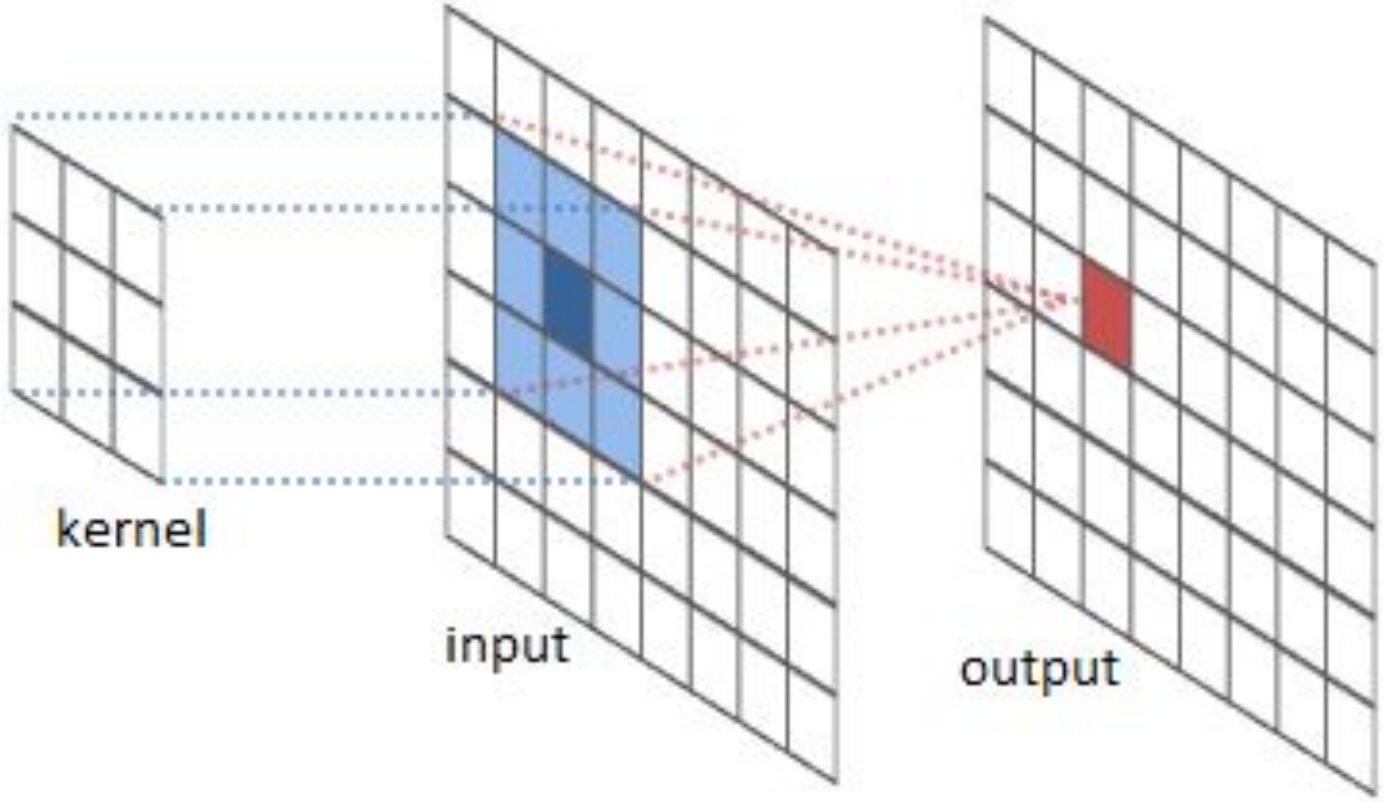
Визуализация операции свёртки

Знакомые по фотошопу фильтры blur, emboss, sharpen и другие — это именно матричные фильтры.



http://intellabs.github.io/RiverTrail/tutorial/

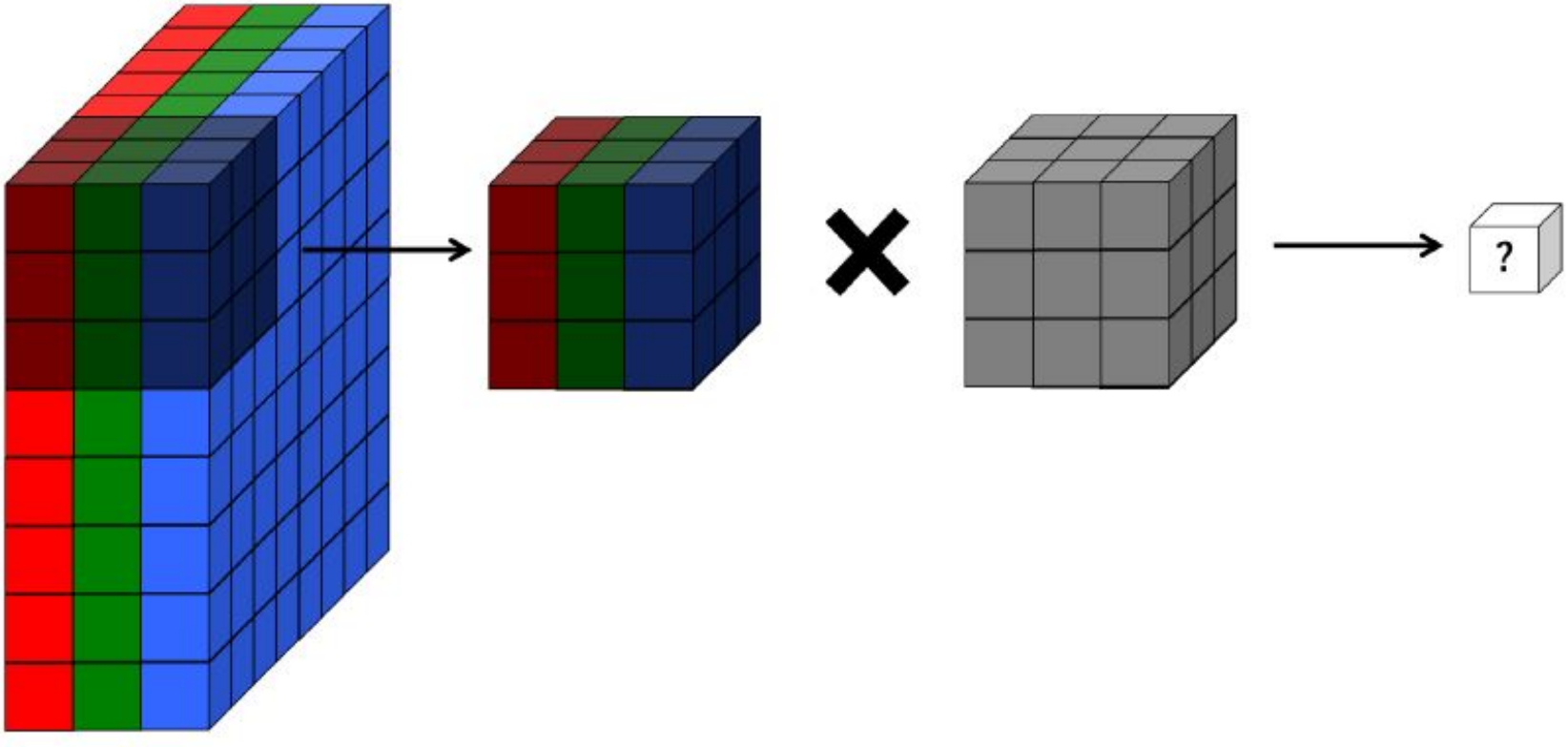
Визуализация операции свёртки



http://intellabs.github.io/RiverTrail/tutorial/

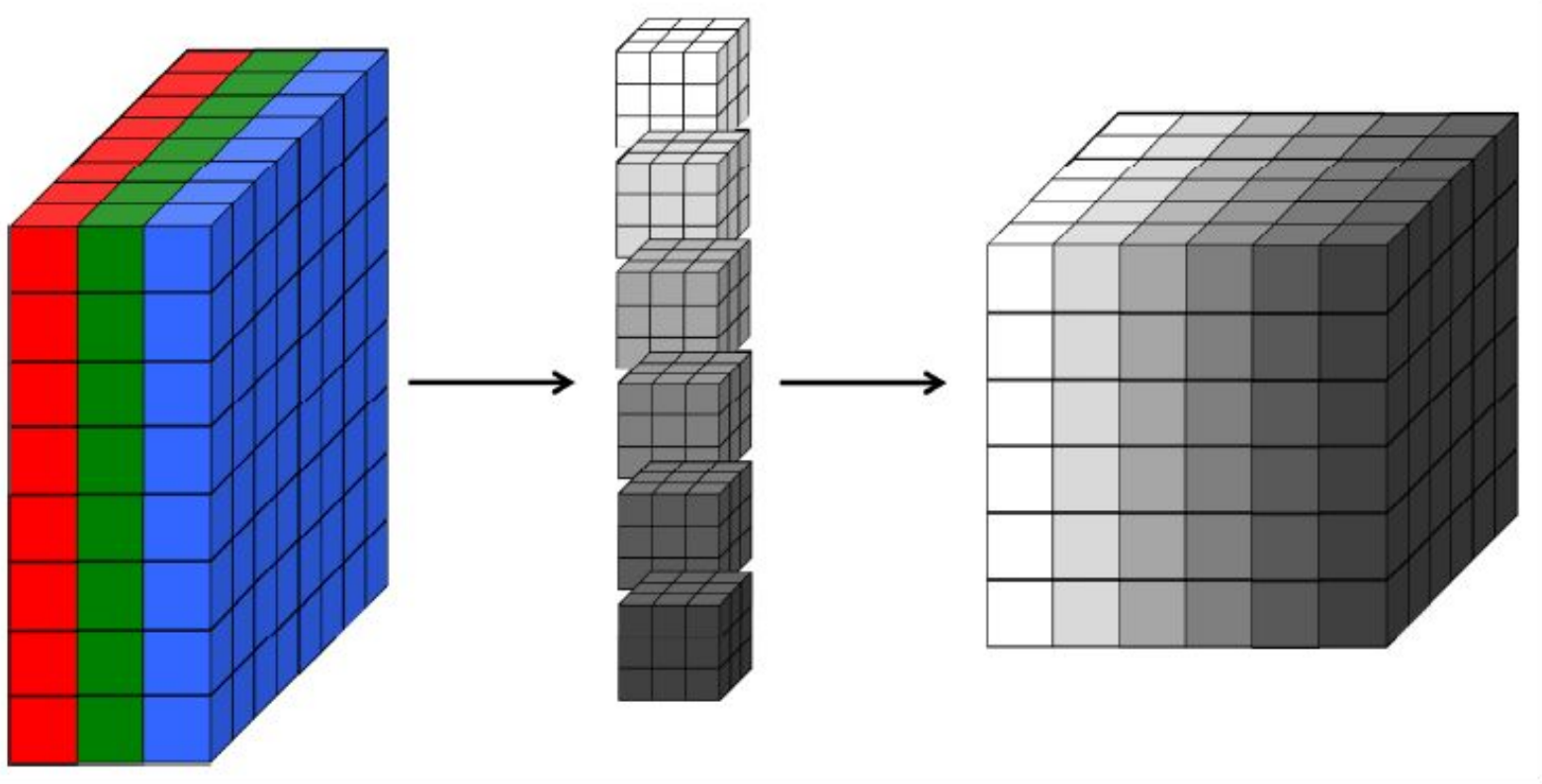
Свёртка работает над объёмами

В реальности фильтр трёхмерный.

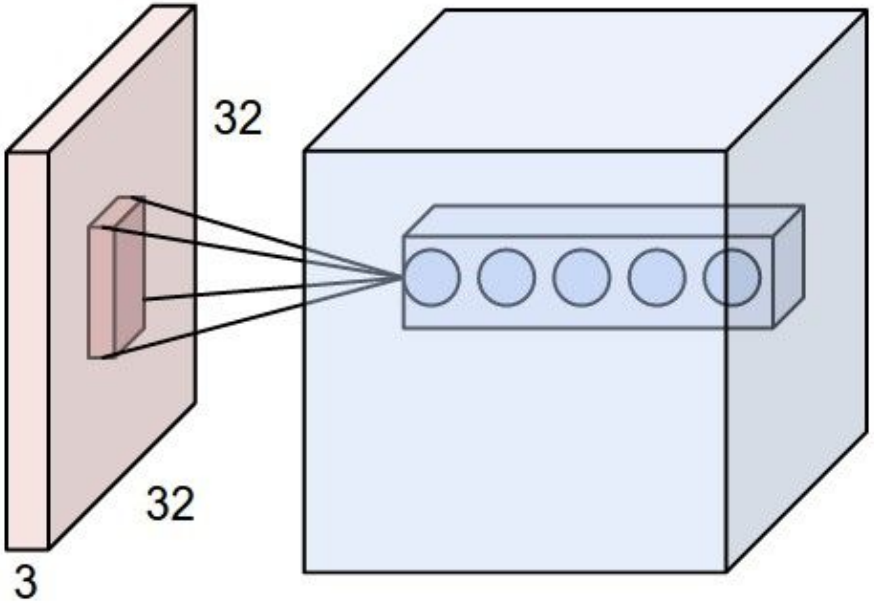


Свёртка работает с объёмами

Несколько фильтров (образующих один слой) создают “объём”



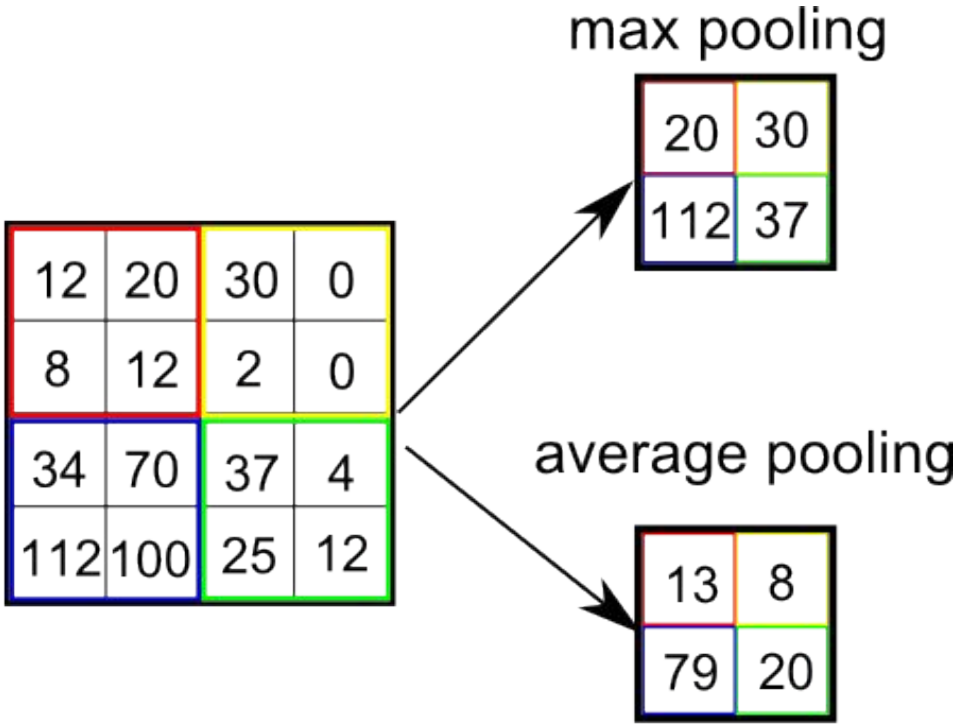
CNN: Свёрточный слой (5 нейронов)



Веса нейронов — это коэффициенты ядра свёртки. Каждая “обучаемая” свёртка выделяет одинаковые локальные признаки во всех частях изображения.

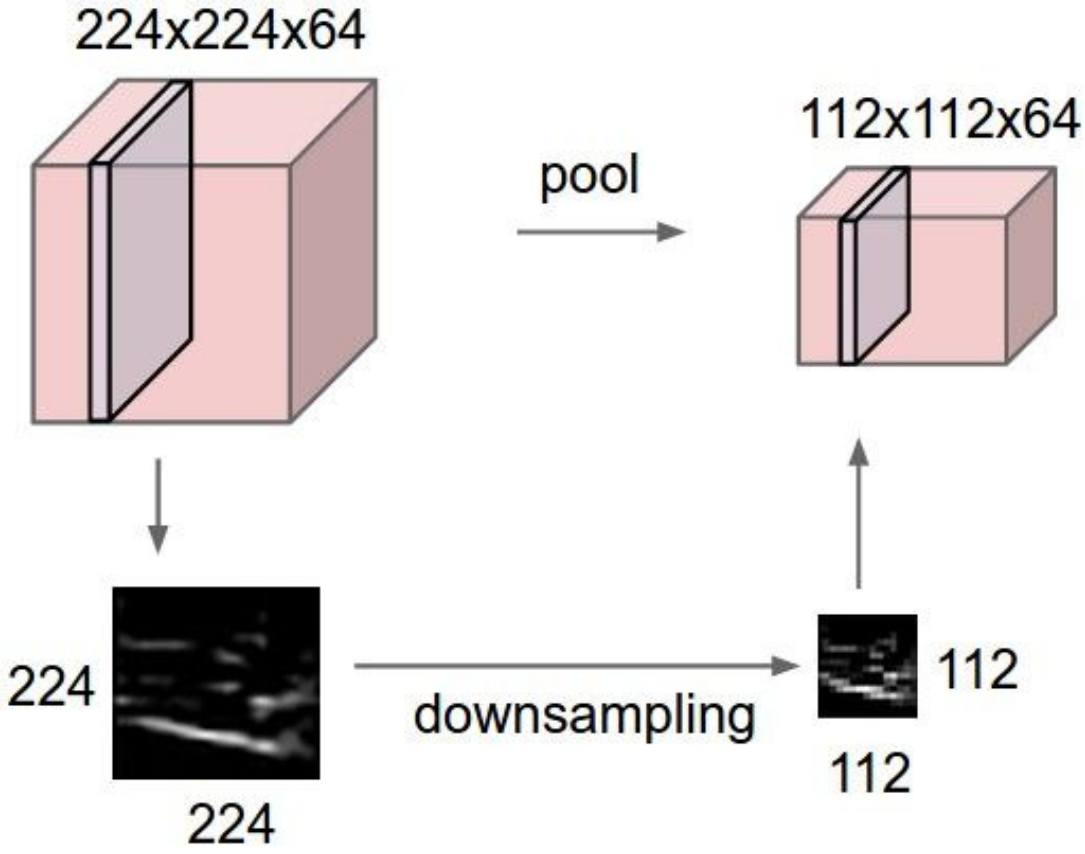
http://cs231n.github.io/convolutional-networks/

Операция pooling (max pool, avg pool)



http://vaaaaaanquish.hatenablog.com/entry/2015/01/26/060622

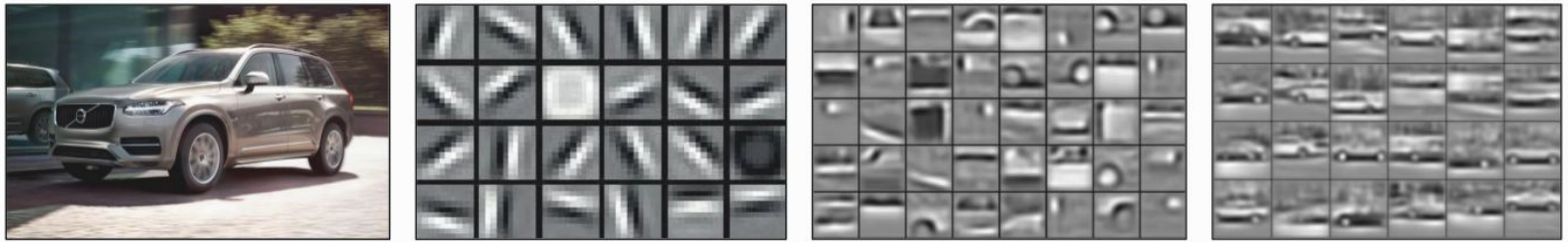
CNN: Pooling слой (downsampling)

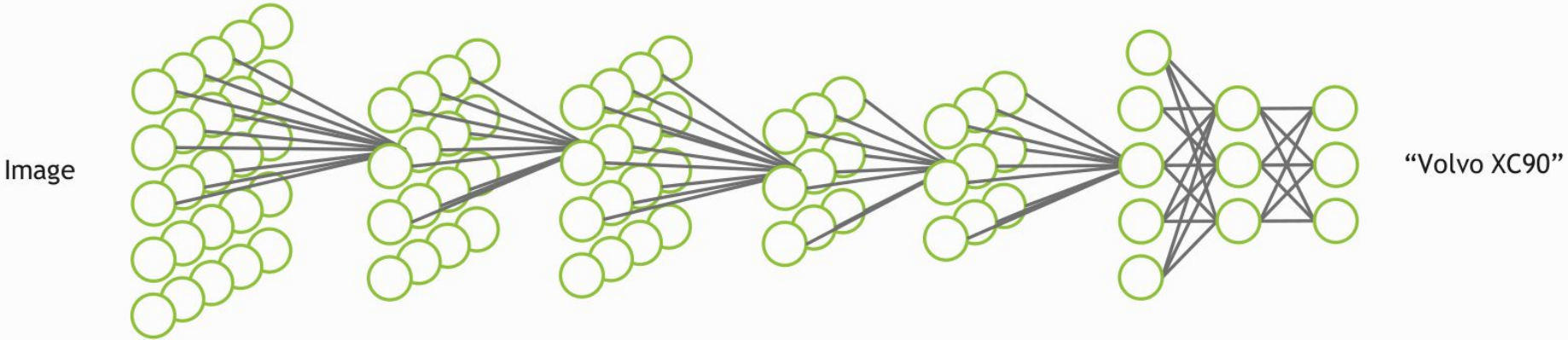


http://cs231n.github.io/convolutional-networks/

Свёрточная нейросеть

Свёрточные слои учат иерархические признаки для изображений, а spatial pooling даёт некоторую инвариантность к перемещениям.





Пример сети: LeNet-5

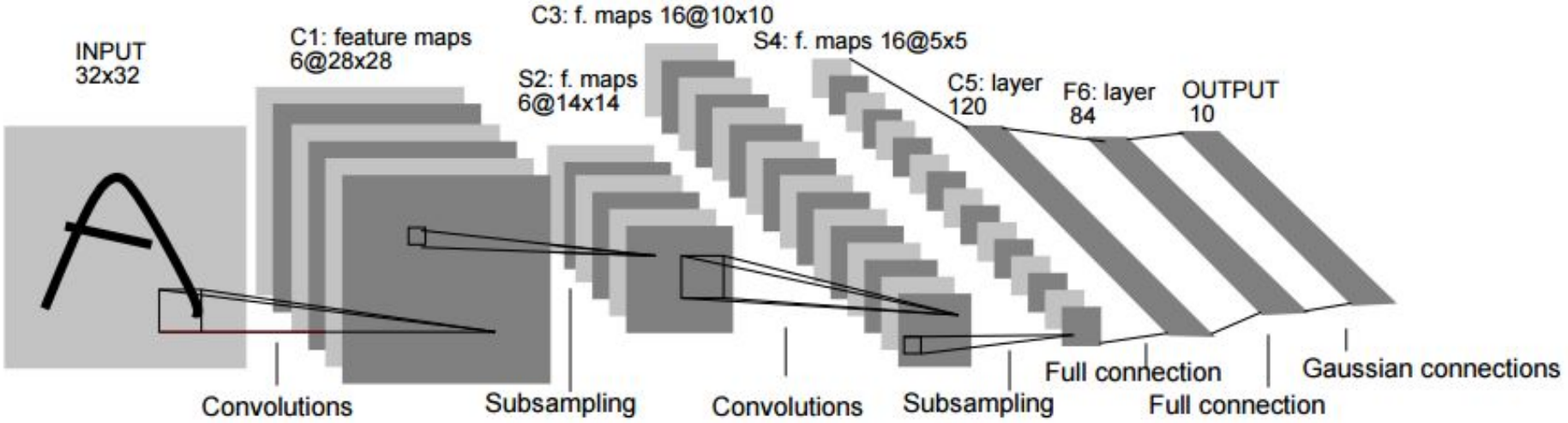


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Example: Neural Net in Keras

```
model = Sequential()
model.add(Convolution2D(20, 5, 5, border_mode="same",
  input_shape=(depth, height, width)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Convolution2D(50, 5, 5, border_mode="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(500))
model.add(Activation("relu"))
model.add(Dense(classes))
model.add(Activation("softmax"))
```

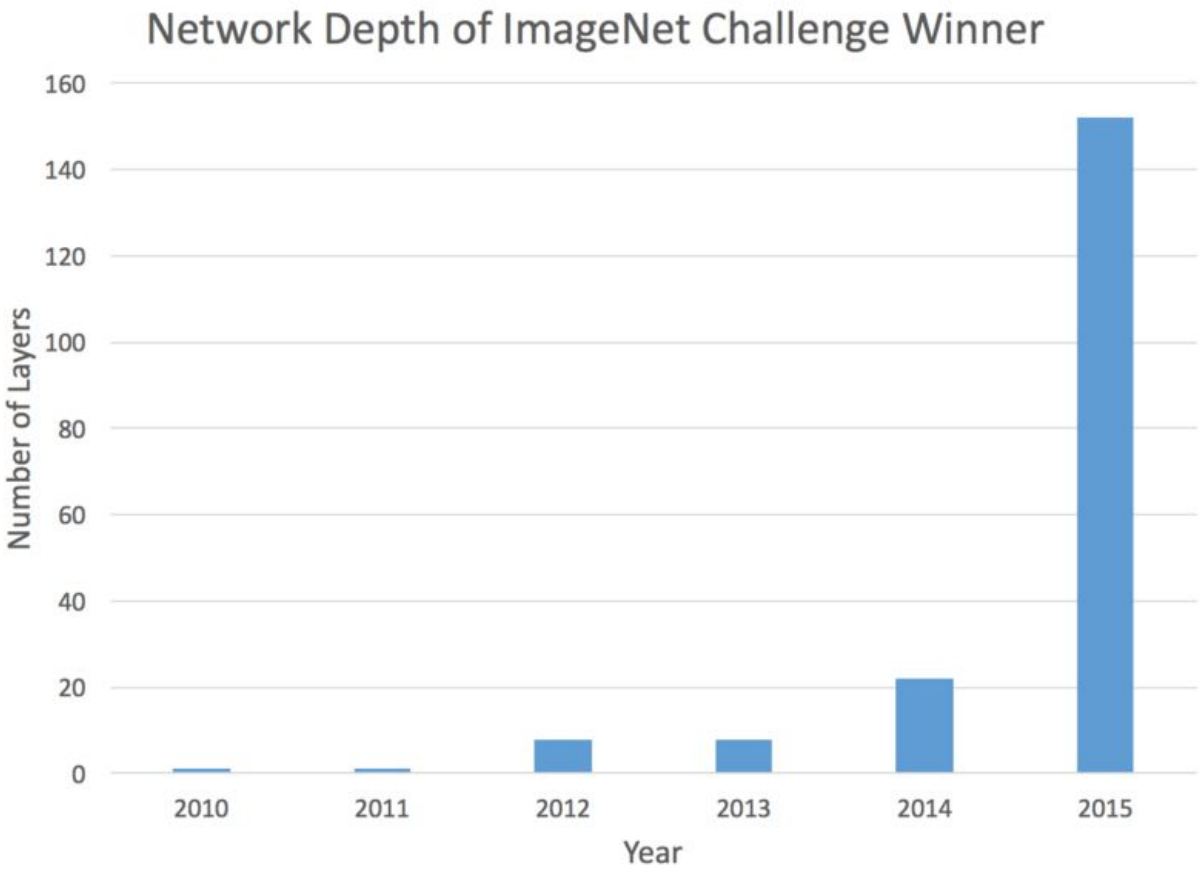
Example: Neural Net in Keras

```
model.compile(loss=keras.losses.categorical_crossentropy,
    optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
    verbose=1, validation_data=(x_test, y_test))

score = model.evaluate(x_test, y_test, verbose=0)
```

Network complexity increases



У CNN меньше параметров, чем у FNN

CNN

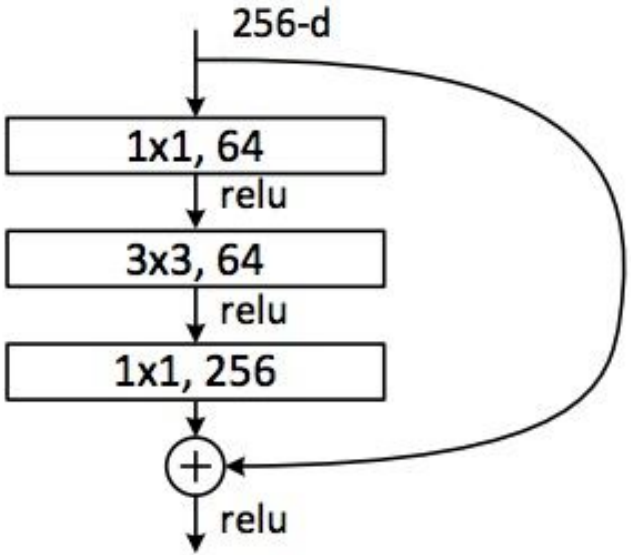
- вход ч/б картинка 100x100
- три свёрточных слоя по 100 плоскостей каждый (conv 5x5 и subsampling 2)
- выход: 10 классов
- число параметров примерно **650K** (5*5*1*100 + 5*5*100*100 + 5*5*100*100 + 12*12*100*10)

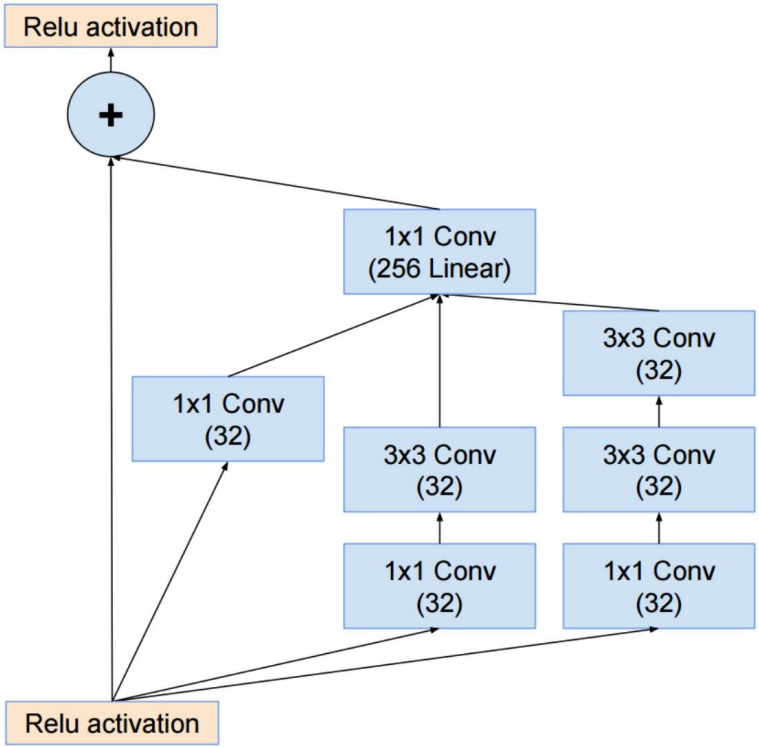
FNN

- вход: ч/б картинка 100x100
- три скрытых слоя по 100 нейронов каждый
- выход: 10 классов
- число параметров примерно **1M** (10000*100 + 100*100 + 100*100 + 100*10)

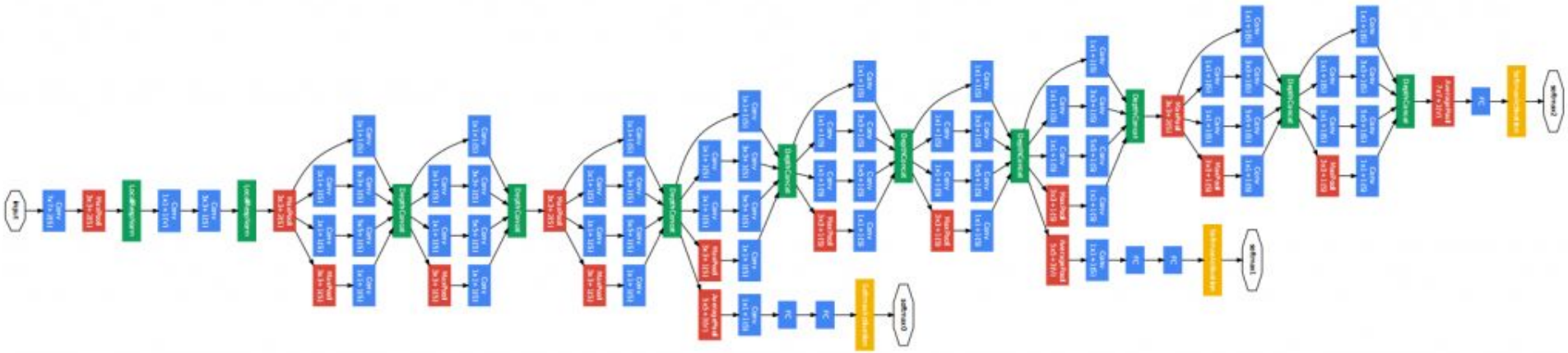
Современные архитектуры

Inception, ResNet и другие современные архитектуры содержат специальные блоки слоёв.



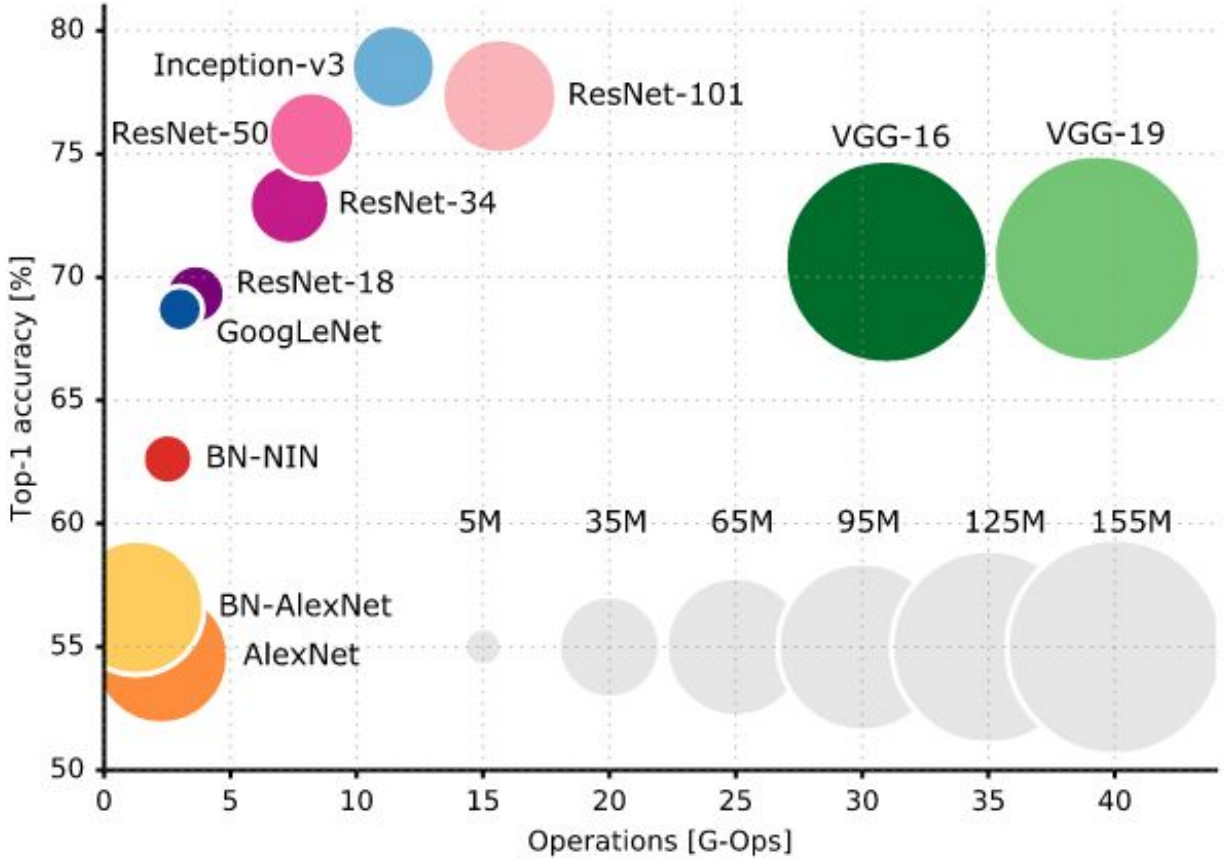


Network complexity increases



GoogLeNet (2014)
http://cs.unc.edu/~wliu/papers/GoogLeNet.pdf

Network complexity increases



https://culurciello.github.io/tech/2016/06/04/nets.html

Case: IJCNN 2011

The German Traffic Sign Recognition Benchmark

- Classification, >40 classes
- >50,000 real-life images
- First Superhuman Visual Pattern Recognition
  - 2x better than humans
  - 3x better than the closest artificial competitor
  - 6x better than the best non-neural method

Method	Correct (Error)
1 Committee of CNNs	**99.46 % (0.54%)**
2 Human Performance	**98.84 % (1.16%)**
3 Multi-Scale CNNs	98.31 % (1.69%)
4 Random Forests	96.14 % (3.86%)

http://people.idsia.ch/~juergen/superhumanpatternrecognition.html

Case: ImageNet

GT: horse cart
1: horse cart
2: minibus
3: oxcart
4: stretcher
5: half track

GT: birdhouse
1: birdhouse
2: sliding door
3: window screen
4: mailbox
5: pot

GT: forklift
1: forklift
2: garbage truck
3: tow truck
4: trailer truck
5: go-kart

GT: coucal
1: coucal
2: indigo bunting
3: lorikeet

GT: komondor
1: komondor
2: patio
3: llama

GT: yellow lady's slipper
1: yellow lady's slipper
2: slug
3: hen-of-the-woods

Точность распознавания растёт

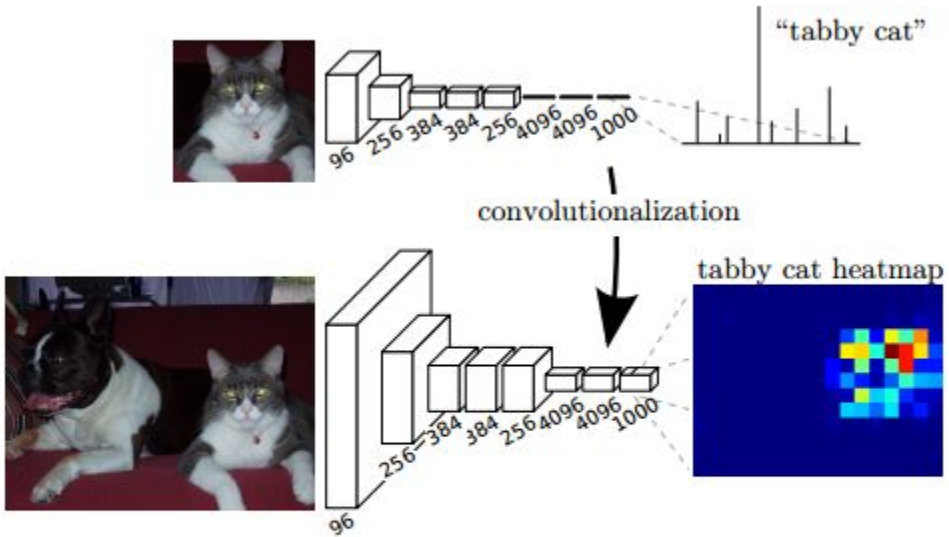
Человеческое качество ~5.1% (0.051)

Examples: Activity Recognition

Fully-convolutional networks (FCN)

Обычная свёрточная сеть, но без MLP сверху (нет полносвязных слоёв).

Позволяет работать с изображениями произвольного размера и выдавать на выходе тепловую карту классификации.

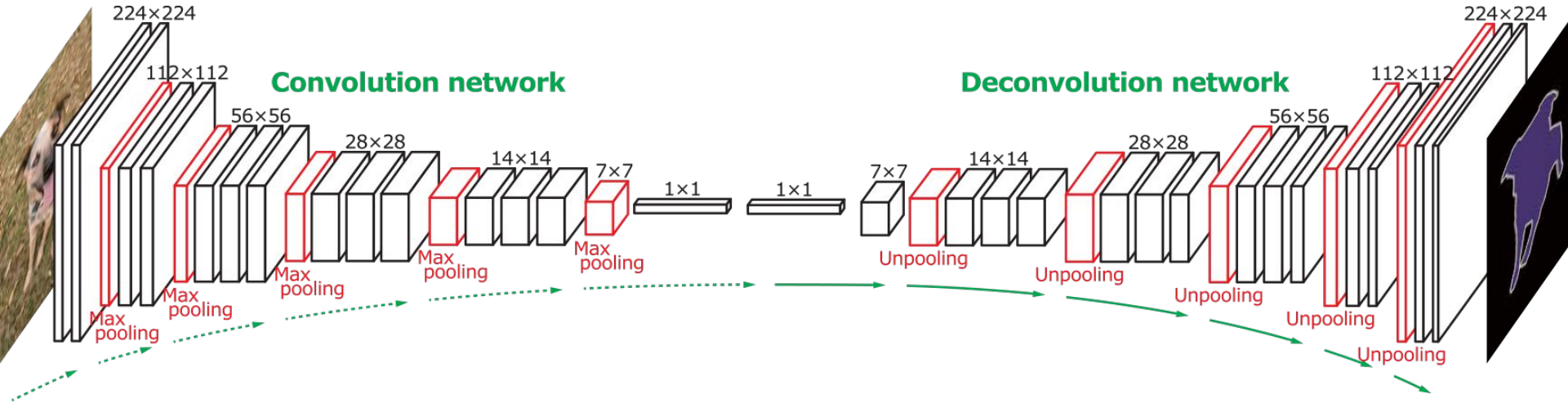


https://arxiv.org/abs/1411.4038

Deconvolution networks

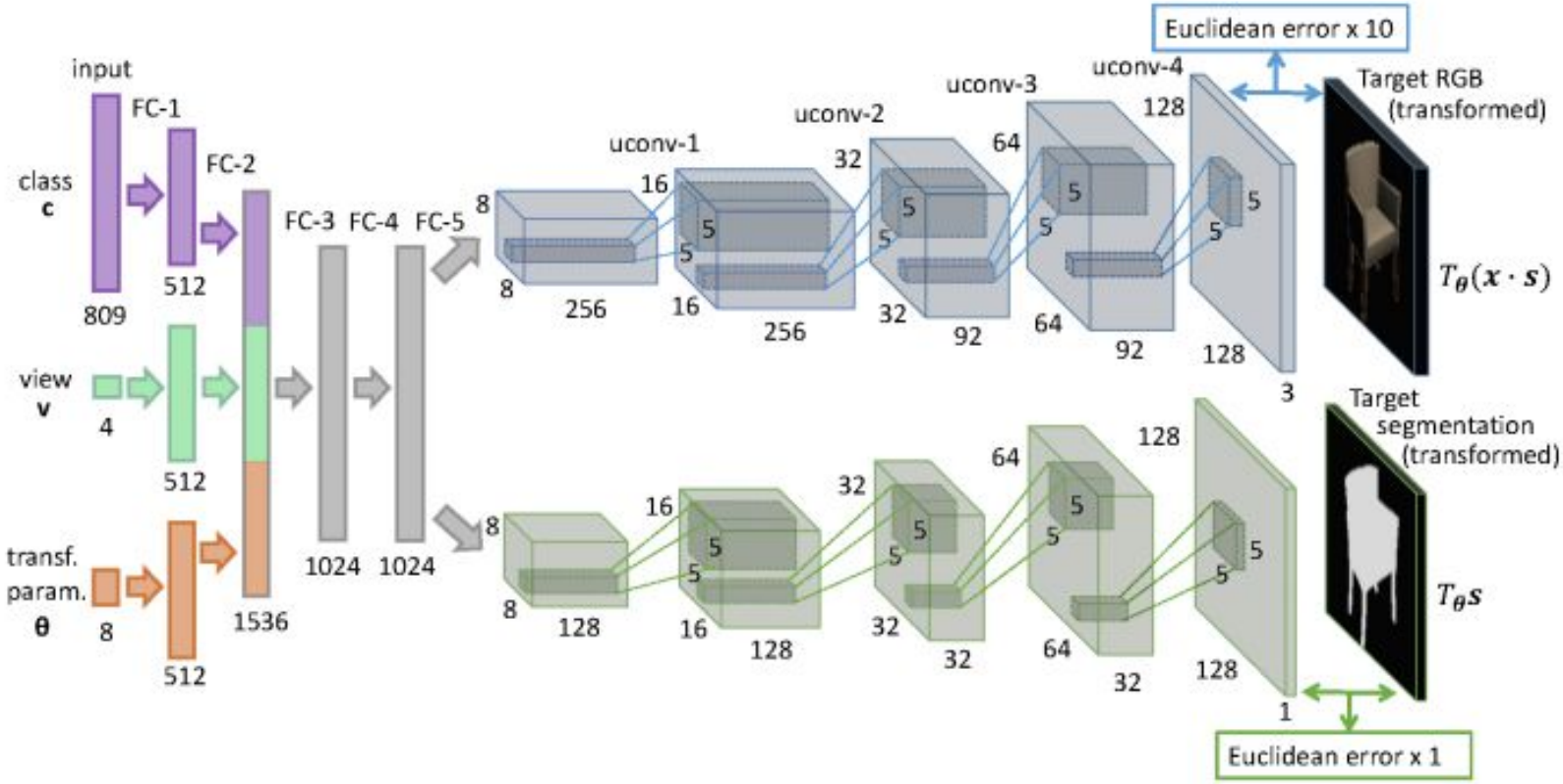
Правильнее называть это Transposed convolution, а не Deconvolution (это слово уже занято в цифровой обработке сигналов для обратной операции).

По сути, реализован обучаемый upsampling.



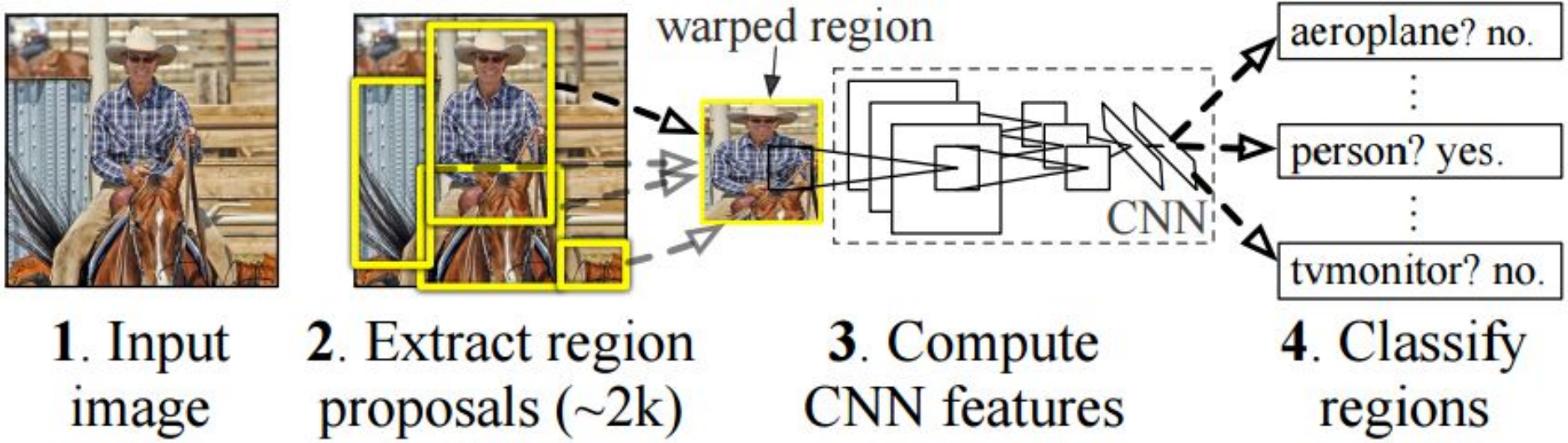
http://cvlab.postech.ac.kr/research/deconvnet/

Генерация изображений



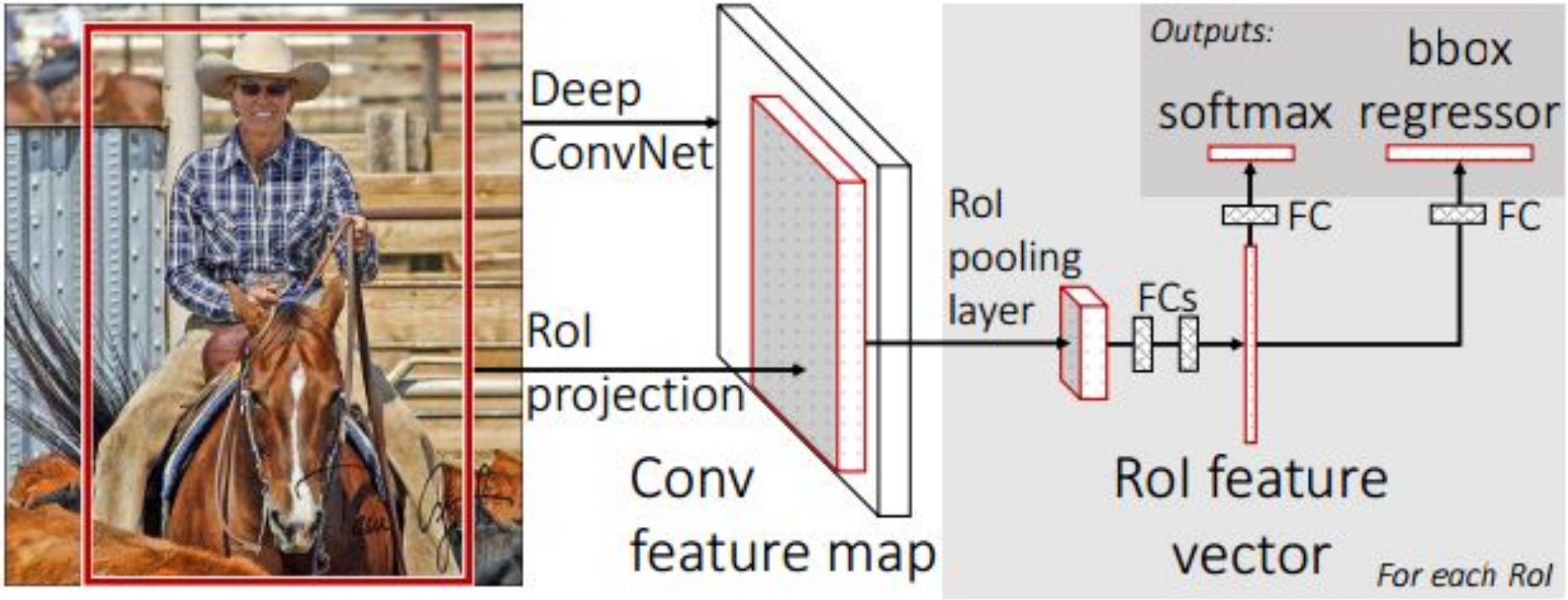
https://arxiv.org/abs/1411.5928

R-CNN: Region-based Convolutional Network



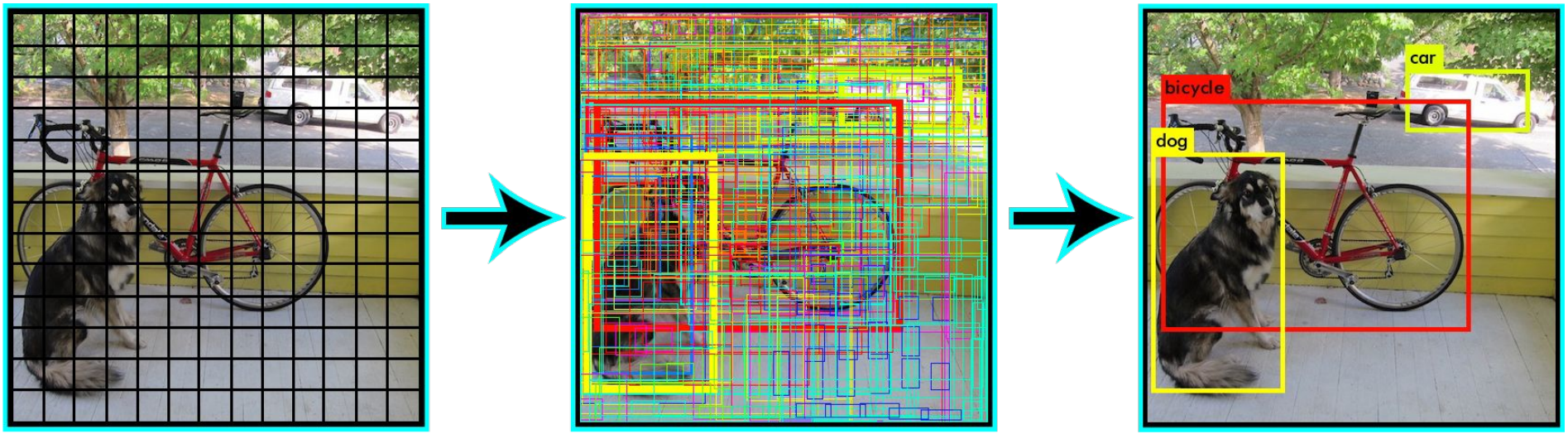
https://github.com/rbgirshick/rcnn
https://people.eecs.berkeley.edu/~rbg/papers/pami/rcnn_pami.pdf

Fast R-CNN



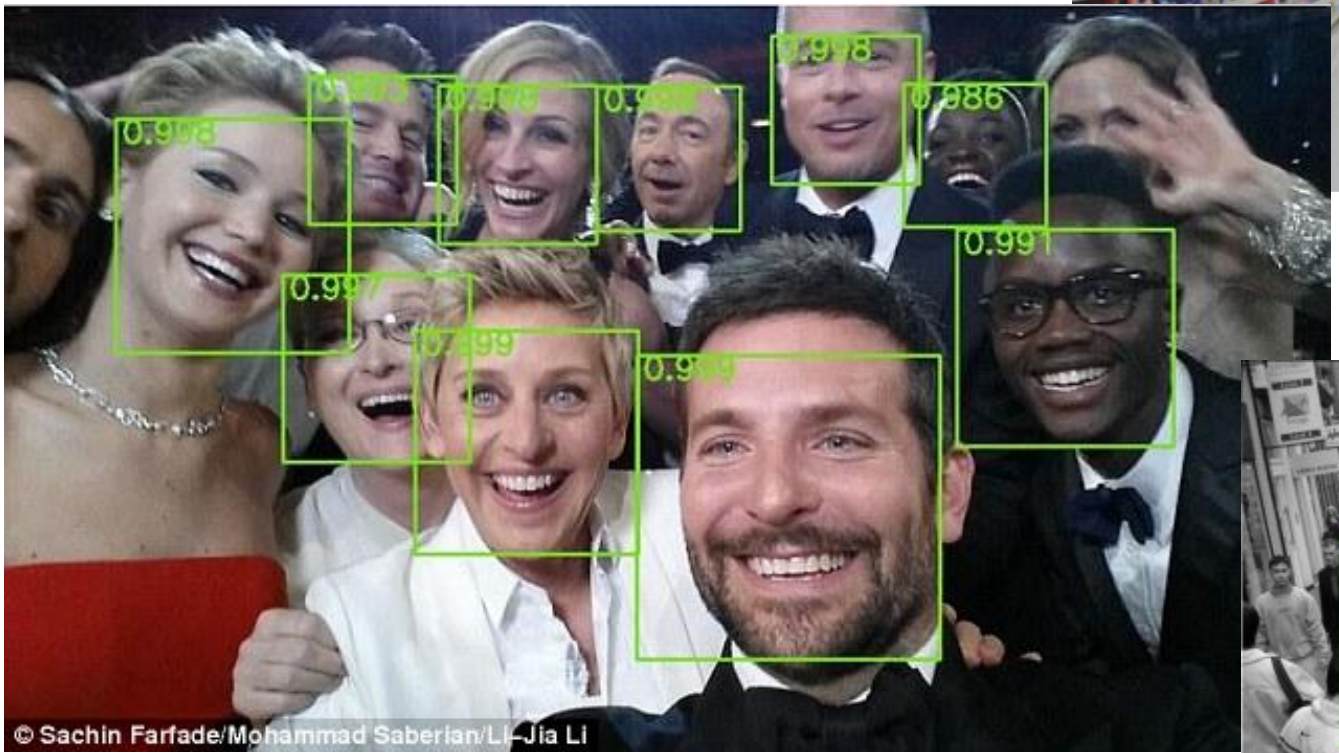
http://tutorial.caffe.berkeleyvision.org/caffe-cvpr15-detection.pdf

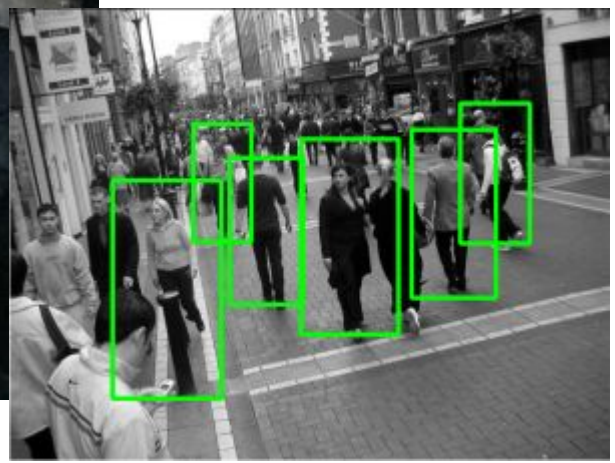
YOLO: Real-Time Object Detection



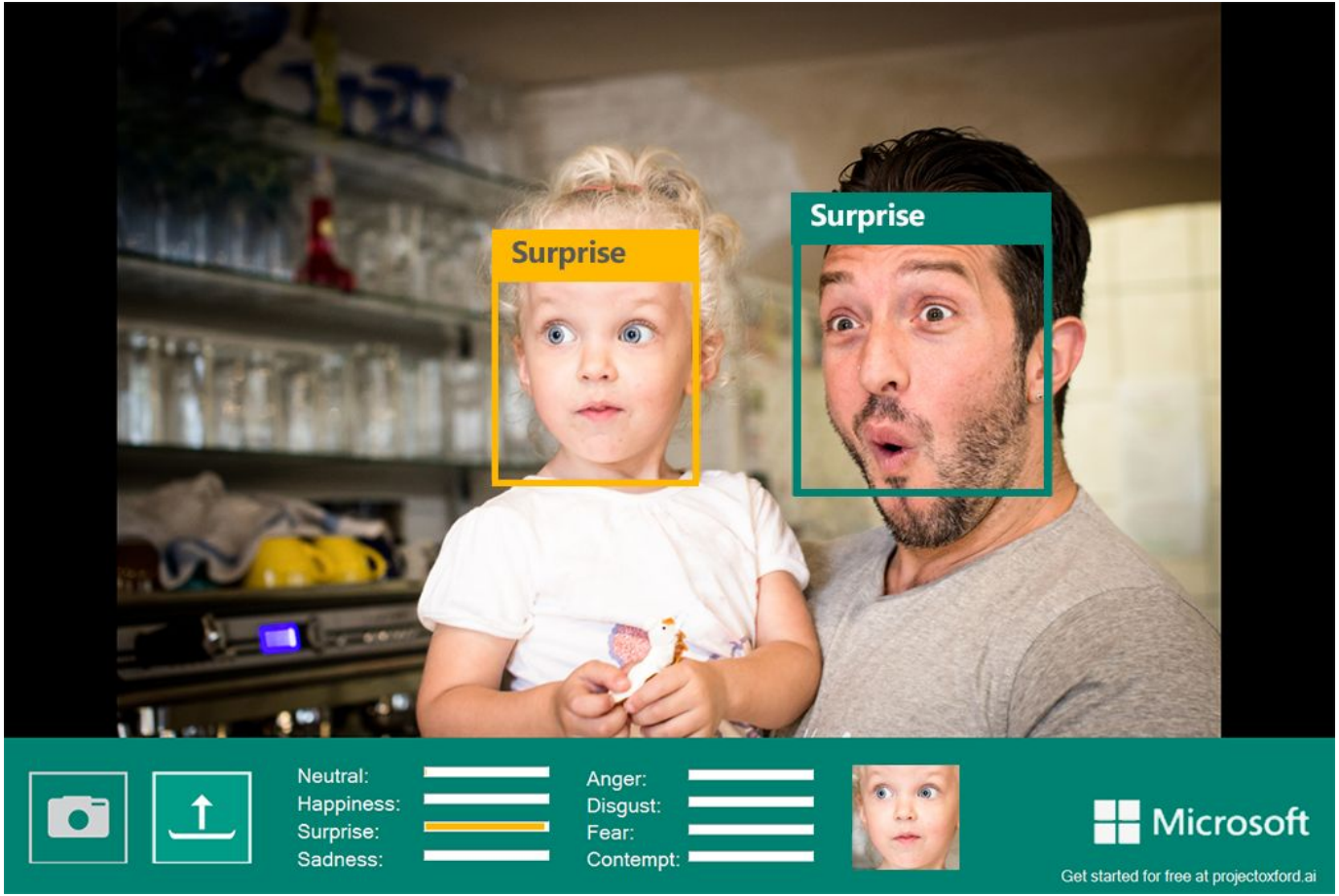
https://pjreddie.com/darknet/yolo/

Examples: Object Detection

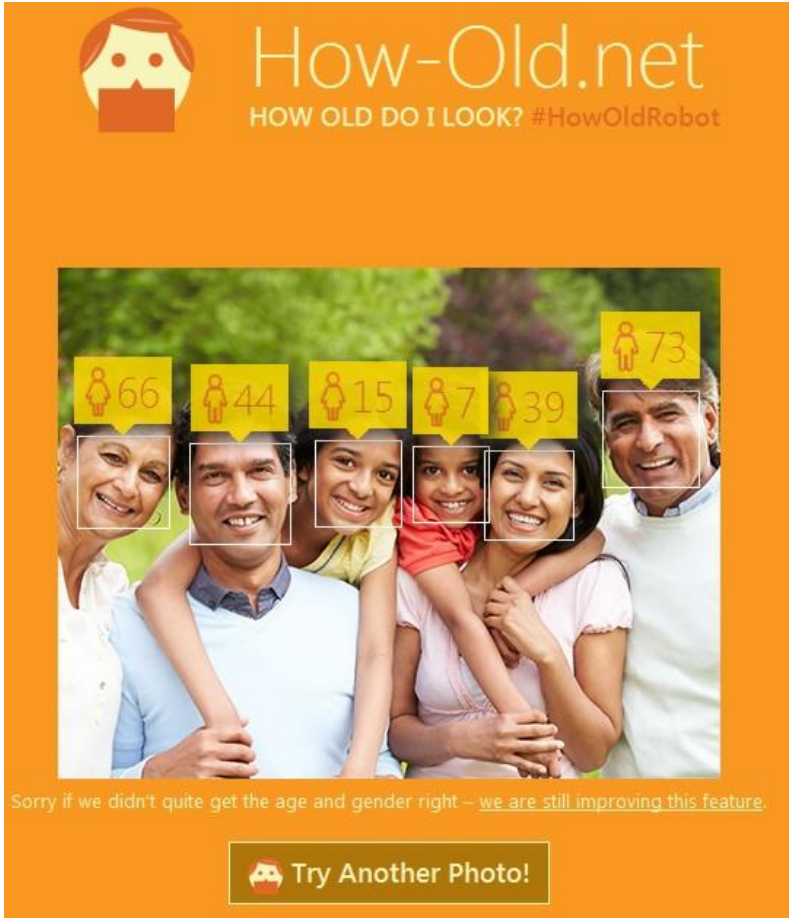




Example: Face Detection + Emotion Classification



Example: Face Detection + Classification + Regression

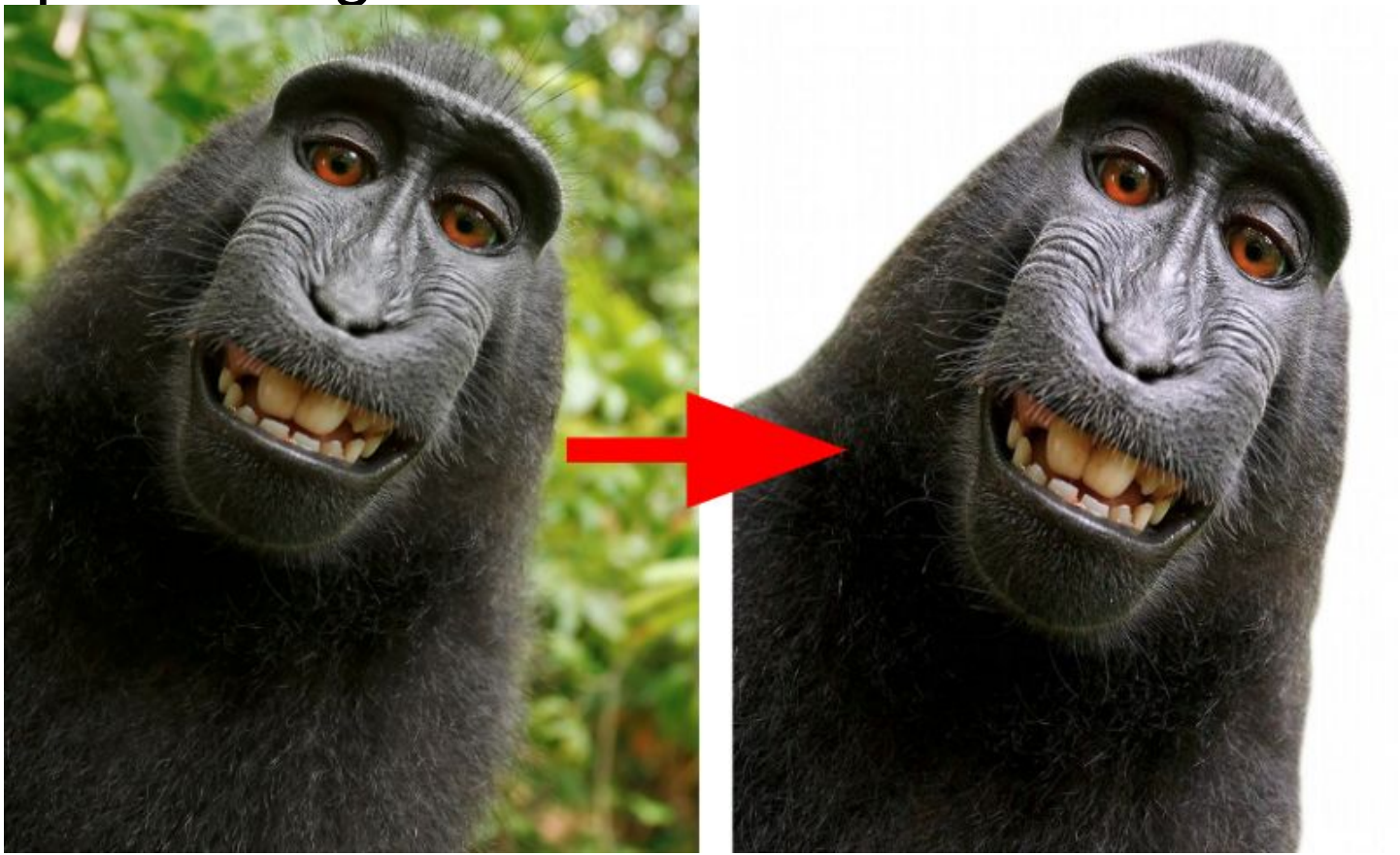


Example: Semantic Segmentation

Example: Image Colorization

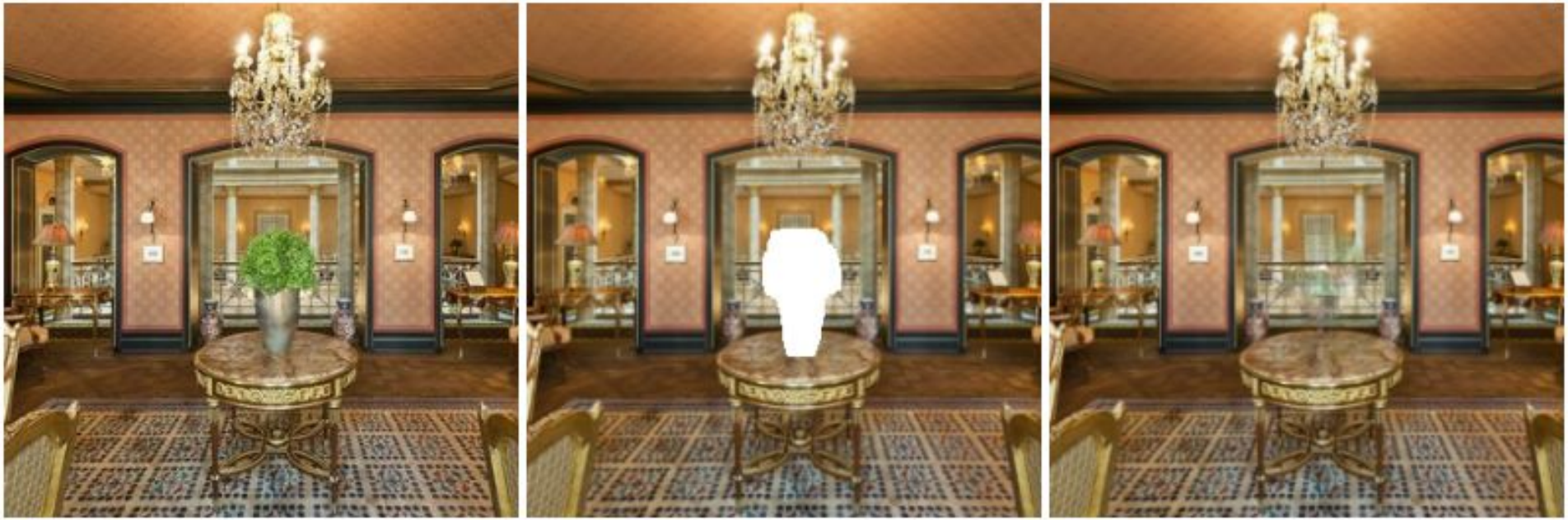
Learning Representations for Automatic Colorization https://arxiv.org/abs/1603.06668

Example: Background removal



https://towardsdatascience.com/background-removal-with-deep-learning-c4f2104b3157

Example: Object removal + Inpainting



Original | Input | Output

http://hi.cs.waseda.ac.jp/~iizuka/projects/completion/en/

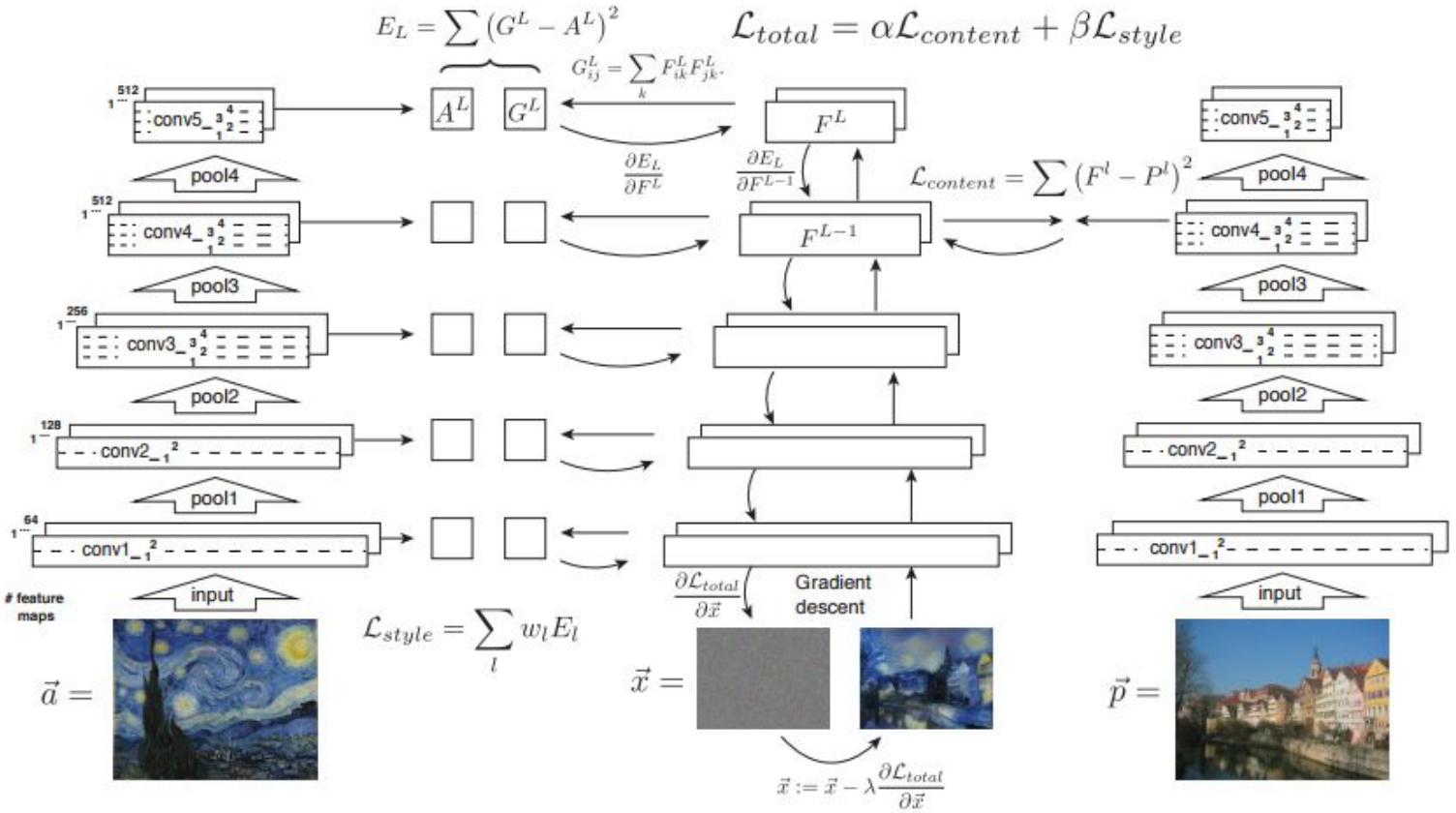
Неклассические задачи: перенос стиля

https://arxiv.org/abs/1508.06576

Example: Photo-realistic Style Transfer

https://arxiv.org/abs/1703.07511 Deep Photo Style Transfer

Перенос стиля: оригинальный алгоритм



https://arxiv.org/abs/1508.06576

Photorealistic content generation

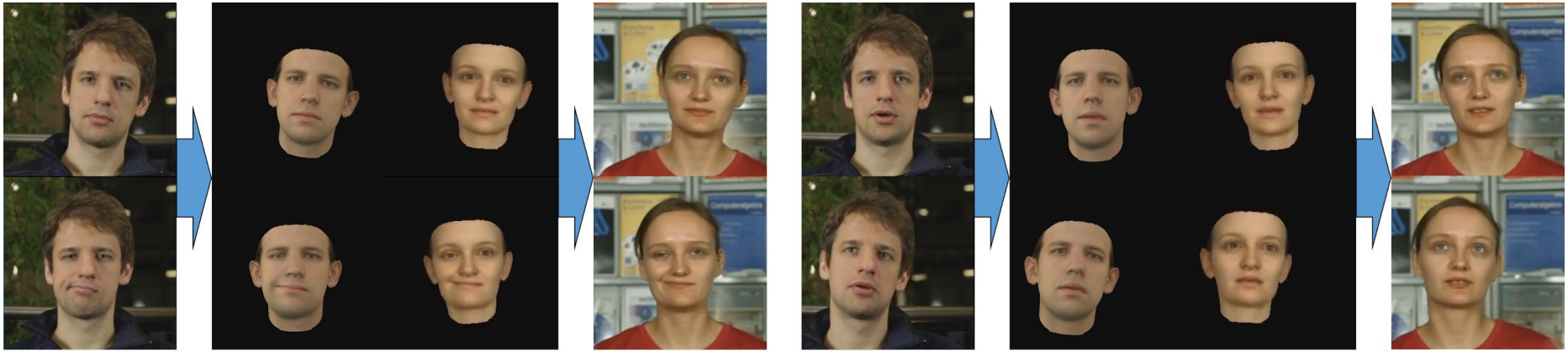
Example: Learning Lip Sync from Audio

http://grail.cs.washington.edu/projects/AudioToObama/
https://www.youtube.com/watch?v=9Yq67CjDqvw

Example: DeepFakes, FakeApp

https://thenextweb.com/artificial-intelligence/2018/02/21/deepfakes-algorithm-nails-donald-trump-in-most-convincing-fake-yet/

Deep Video Portraits



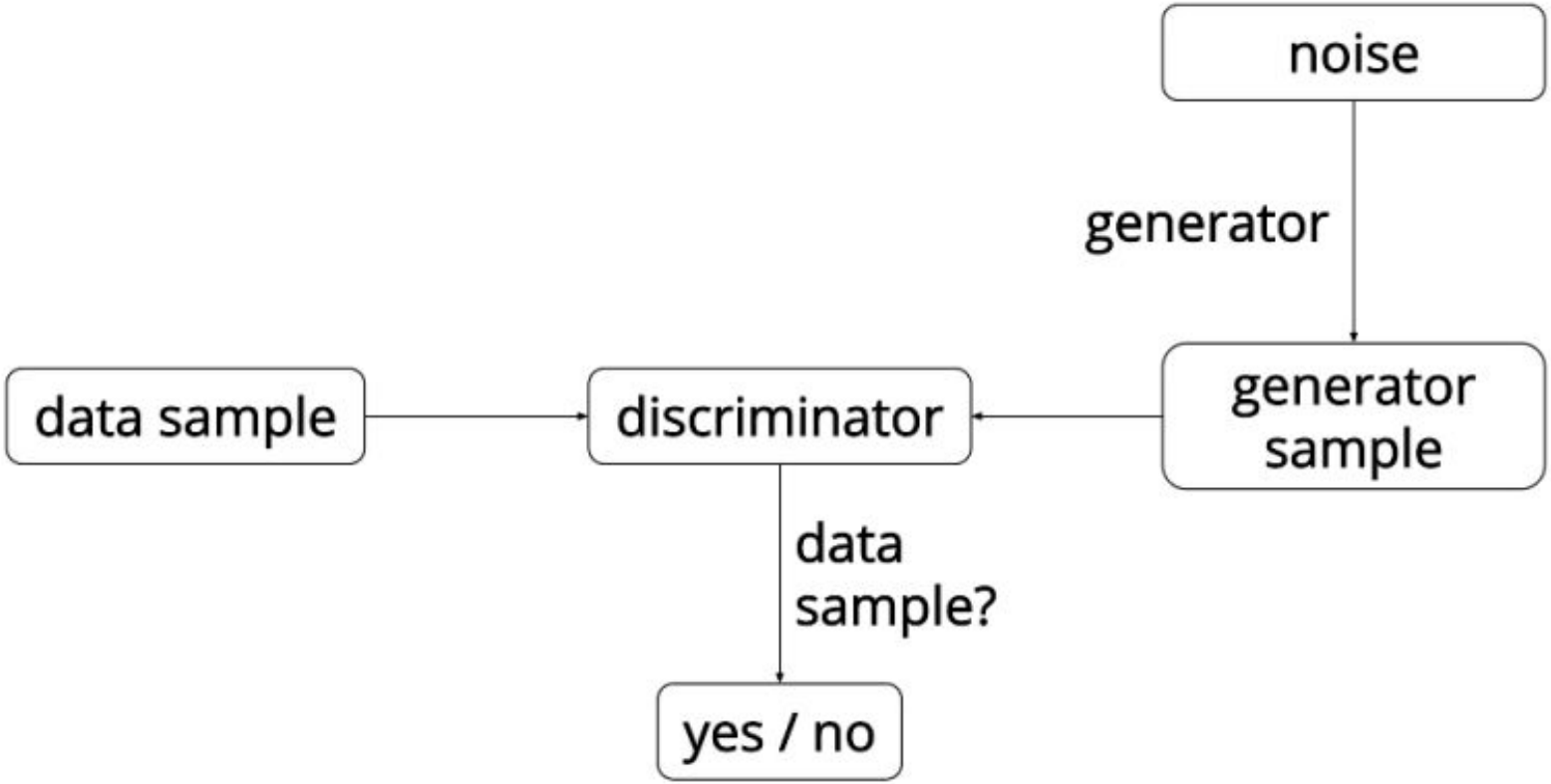
“We present a novel approach that enables photo-realistic re-animation of portrait videos using only an input video. In contrast to existing approaches that are restricted to manipulations of facial expressions only, we are the first to transfer the full 3D head position, head rotation, face expression, eye gaze, and eye blinking from a source actor to a portrait video of a target actor.”

https://web.stanford.edu/~zollhoef/papers/SG2018_DeepVideo/page.html
https://techcrunch.com/2018/06/04/forget-deepfakes-deep-video-portraits-are-way-better-and-worse/

New kid on the block: GAN

https://www.technologyreview.com/lists/technologies/2018/

Generative Adversarial Networks (GANs)



https://arxiv.org/abs/1411.5928

Generative Adversarial Networks (GANs)

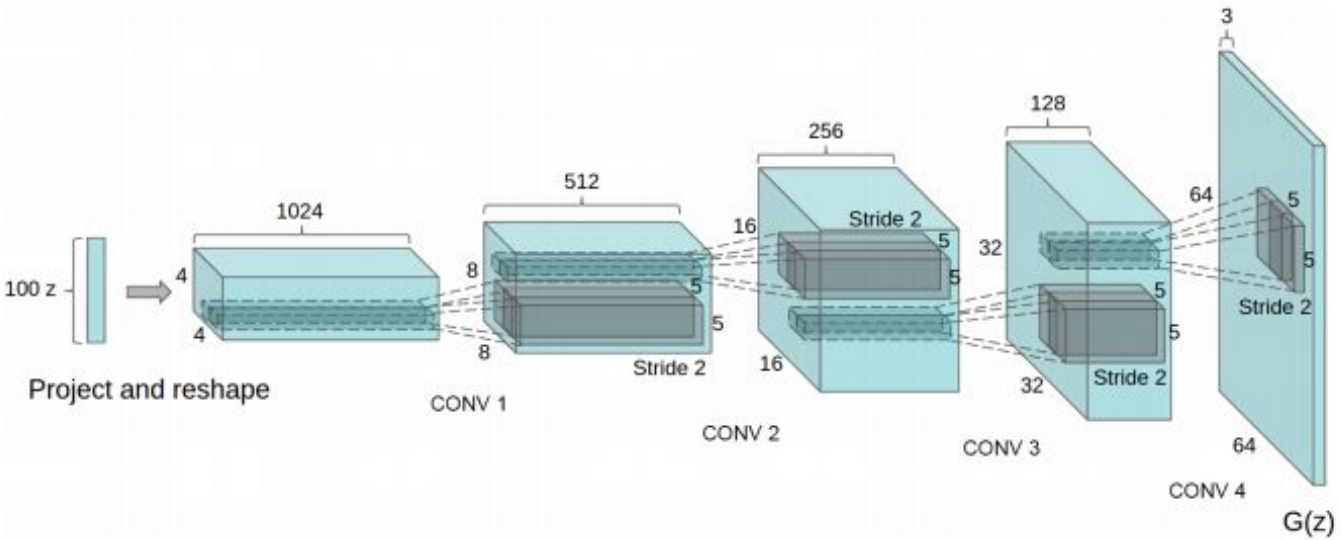


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

https://arxiv.org/pdf/1511.06434v2.pdf

Generative Adversarial Networks (GANs)

Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

https://arxiv.org/pdf/1511.06434v2.pdf

Generative Adversarial Networks (GANs)

volcano

http://www.evolvingai.org/ppgn

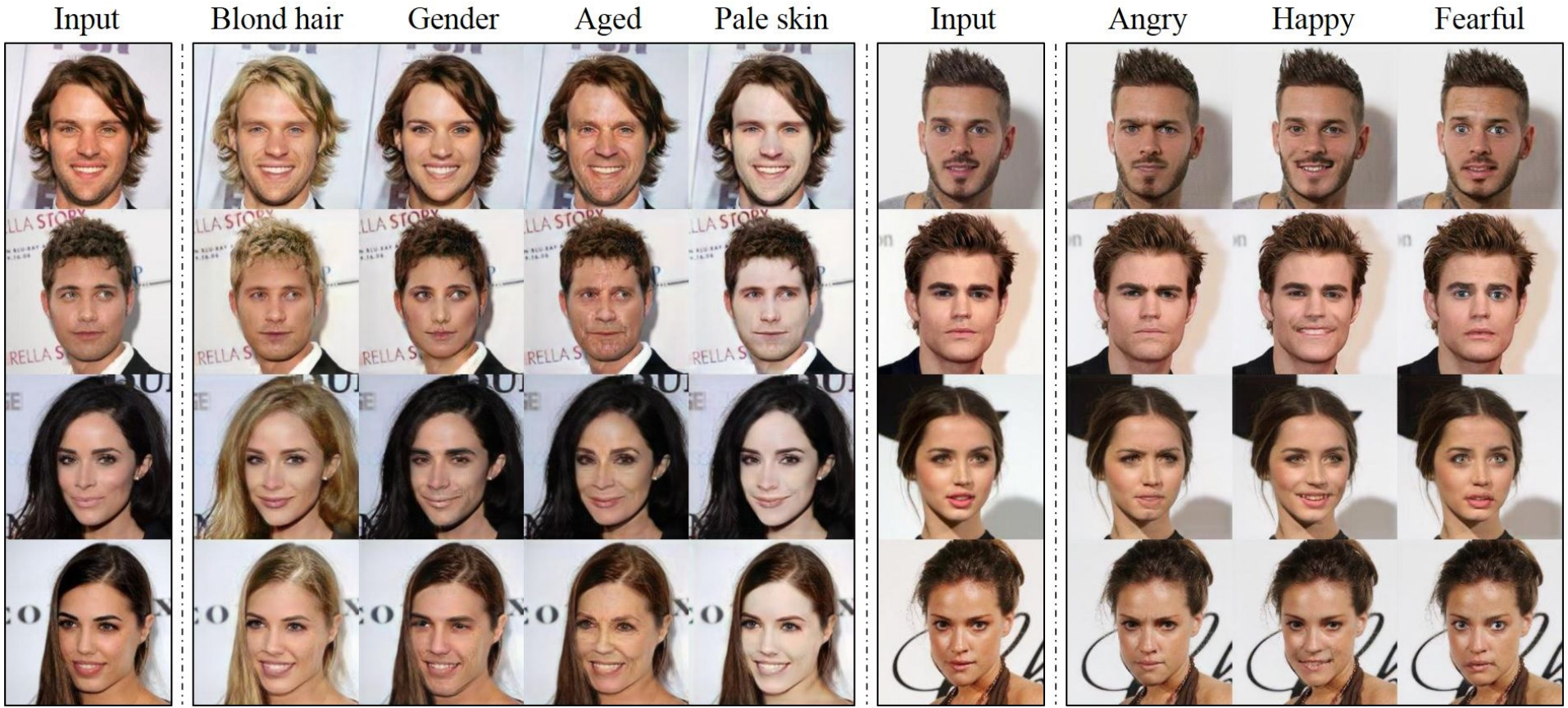
Example: Generating images by GAN

Progressive Growing of GANs for Improved Quality, Stability, and Variation,
https://github.com/tkarras/progressive_growing_of_gans
https://www.youtube.com/watch?v=XOxxPcy5Gr4

GAN rapid evolution

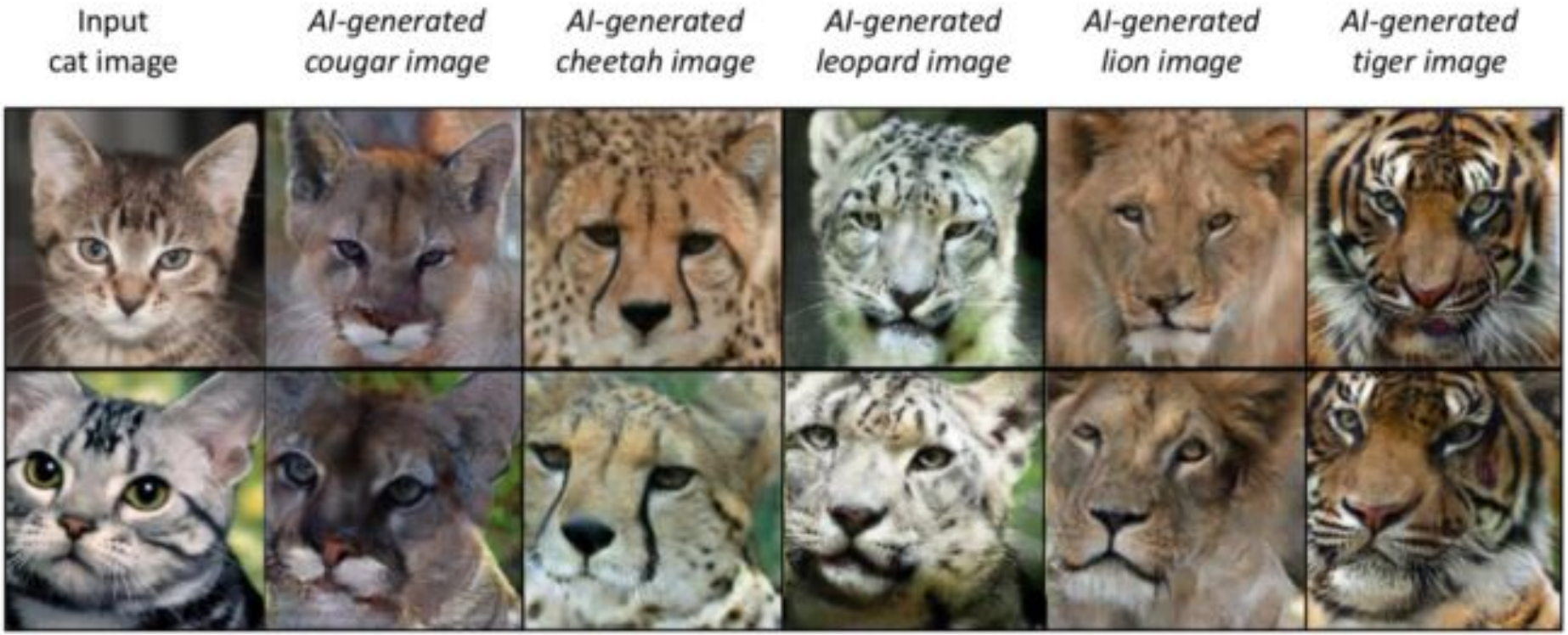
The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation
https://arxiv.org/abs/1802.07228

Example: Multi-Domain Image-to-Image Translation



https://github.com/yunjey/StarGAN

Example: **Unsupervised** Image-to-Image Translation



http://research.nvidia.com/publication/2017-12_Unsupervised-Image-to-Image-Translation
https://www.youtube.com/watch?v=nlyXoX2aIek
https://arxiv.org/abs/1703.00848

Input winter image

AI-generated summer image

Input sunny image

AI-generated rainy image

NVIDIA Real-Time Ray Tracing

https://developer.nvidia.com/rtx

But...

What’s with the Big Picture?

https://www.engadget.com/2018/01/23/photo-stitch-ai-fail-the-big-picture/

Still some issues exist: Reasoning

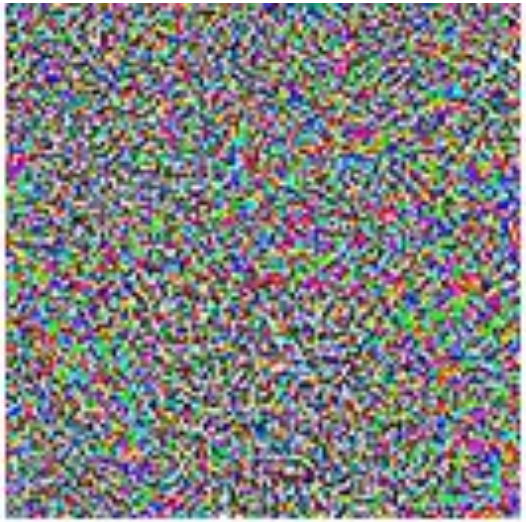
Deep learning is mainly about perception, but there is a lot of inference involved in everyday human reasoning.

- Neural networks lack common sense
- Cannot find information by inference
- Cannot explain the answer
  - It could be a must-have requirement in some areas, i.e. law, medicine.
  - GDPR is coming

The most fruitful approach is likely to be a hybrid neural-symbolic system. Topic of active research right now.

Adversarial Examples

$+\epsilon$



$=$

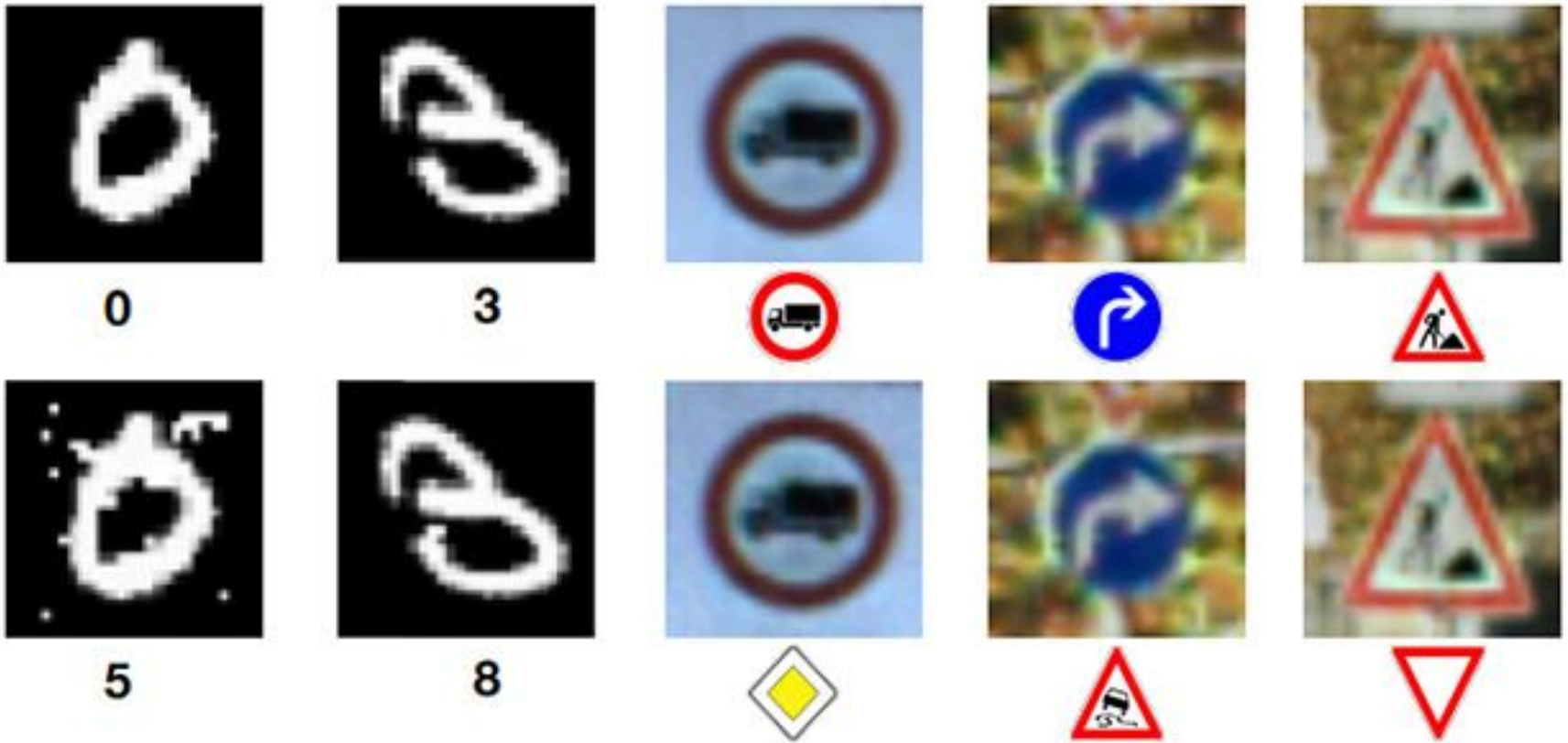
"panda"

57.7% confidence

"gibbon"

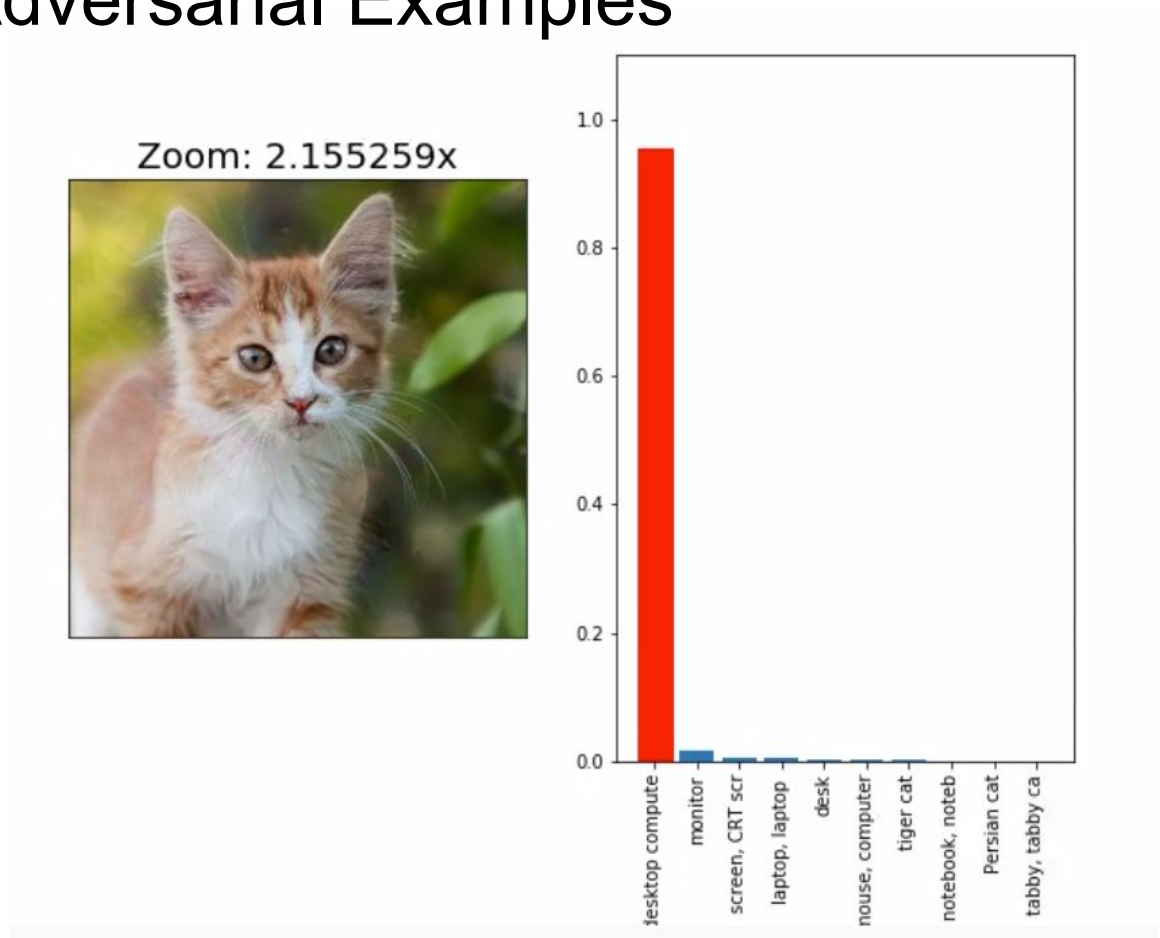
99.3% confidence

Adversarial Examples



https://spectrum.ieee.org/cars-that-think/transportation/sensors/slight-street-sign-modifications-can-fool-machine-learning-algorithms

Robust Adversarial Examples

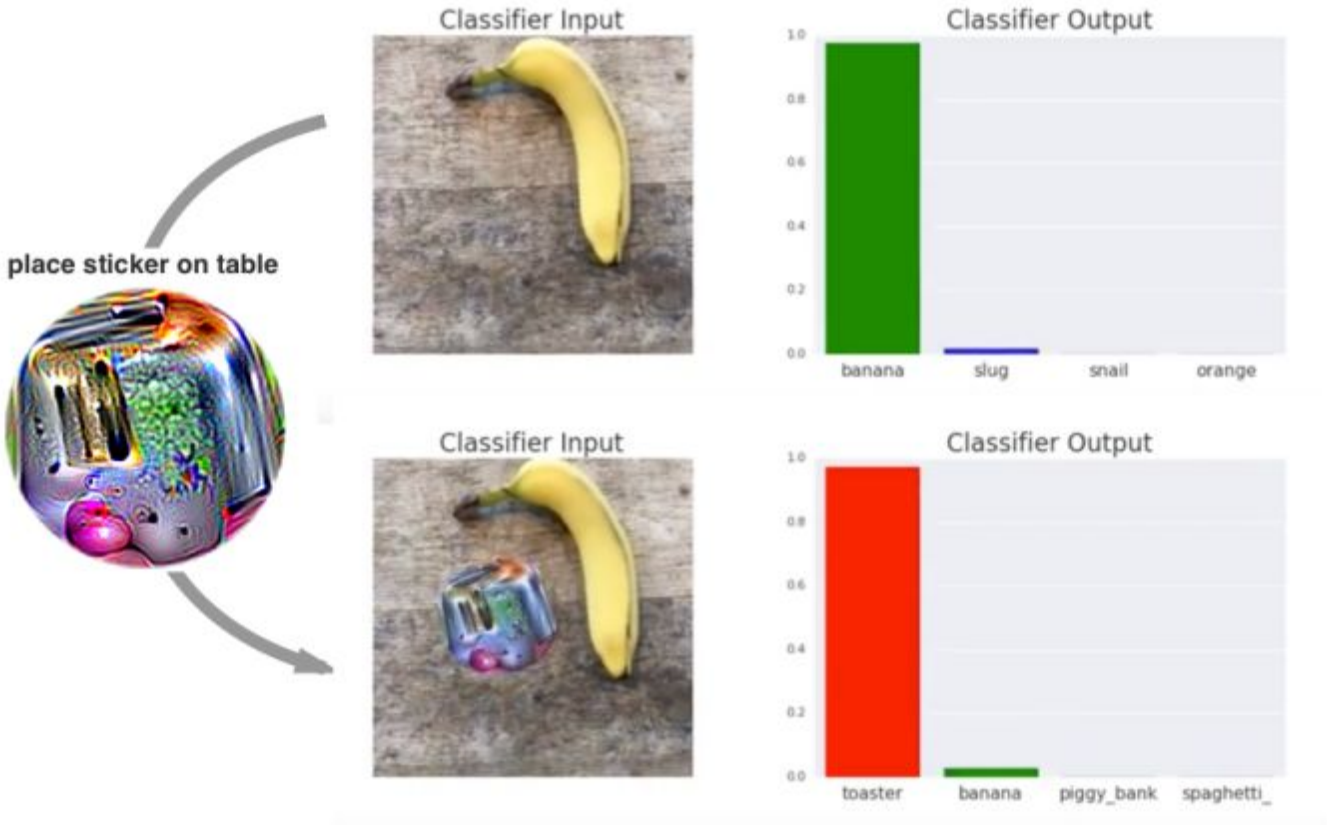


https://blog.openai.com/robust-adversarial-inputs/

Physical Adversarial Examples

http://www.labsix.org/physical-objects-that-fool-neural-nets/

Adversarial Patch



https://arxiv.org/abs/1712.09665

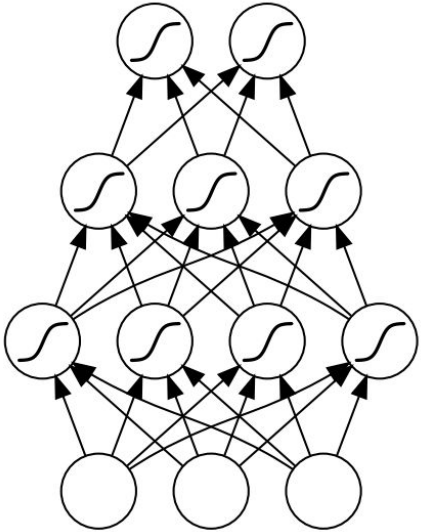
Computer & Human Adversarial Examples

https://spectrum.ieee.org/the-human-os/robotics/artificial-intelligence/hacking-the-brain-with-adversarial-images

Рекуррентные нейросети

Recurrent Neural Networks, RNN

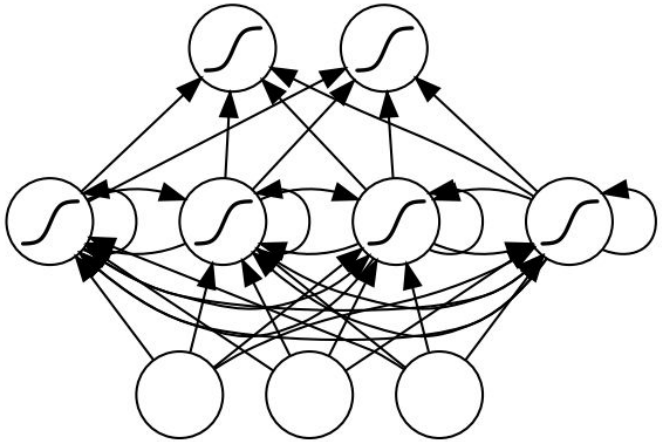
Feedforward NN vs. Recurrent NN



Output Layer

Hidden Layers

Input Layer



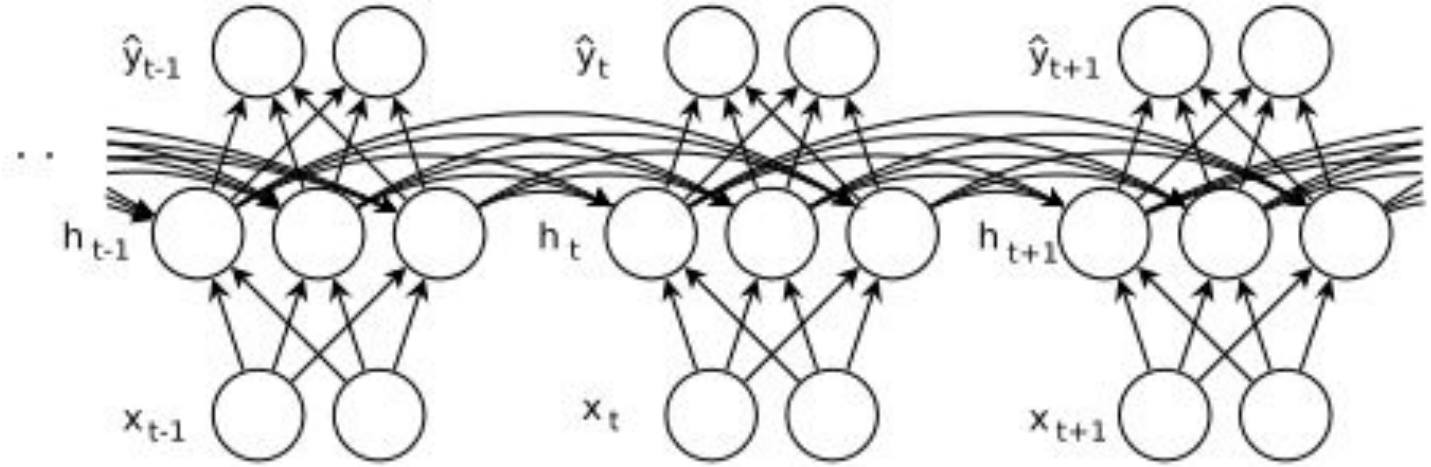
Output Layer

Hidden Layer

Input Layer

В RNNs разрешены циклические связи

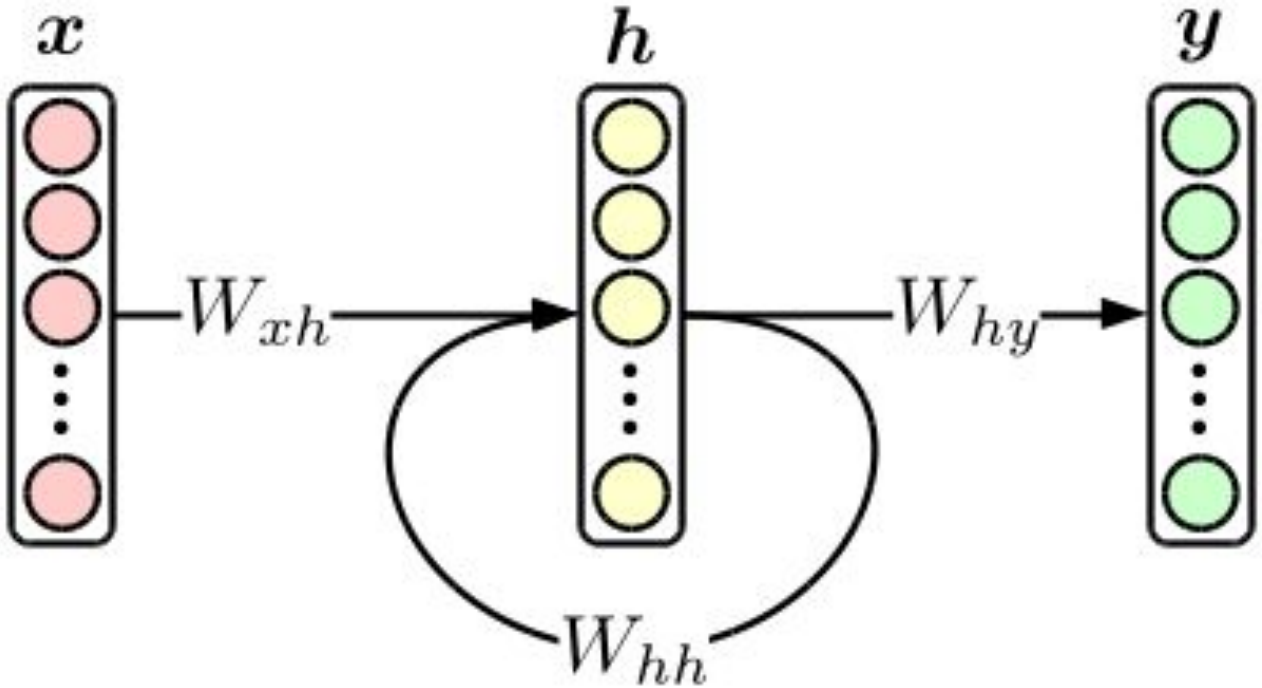
Что такое циклическая связь?



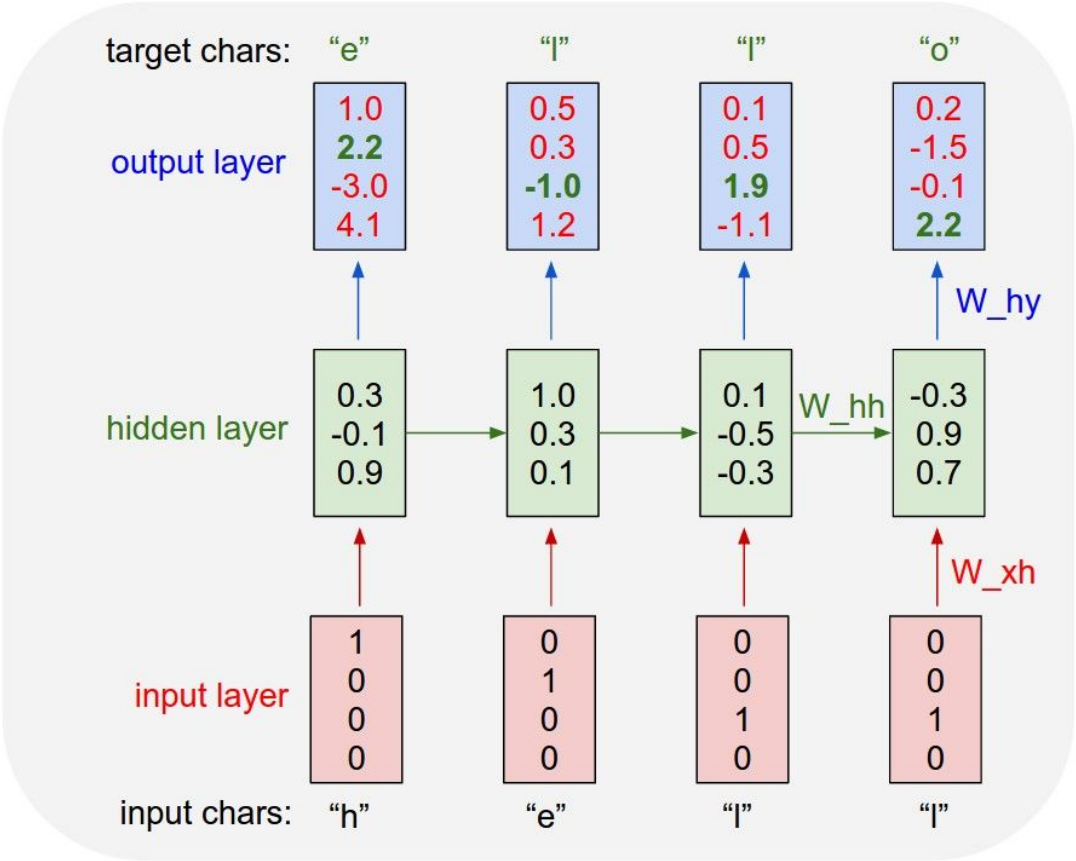
Циклические связи существуют во времени

RNN: Матричная формулировка

Главное отличие RNN от FNN в наличии циклических связей (W_{hh} на рисунке)



Пример работы на генерации текста



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Свойства нейросетей

Feedforward NN (FNN):

- FFN — это универсальный аппроксиматор: однослойная нейросеть с конечным числом нейронов может аппроксимировать непрерывную функцию на компактных подмножествах R^n (Теорема Цыбенко, универсальная теорема аппроксимации).
- FFN не имеют естественной возможности учесть порядок во времени.
- FFN не обладают памятью, кроме полученной во время обучения.

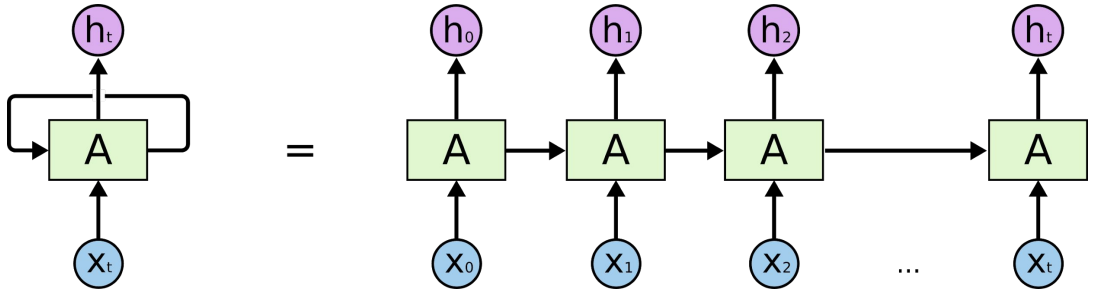
Recurrent NN (RNN):

- RNN Тьюринг-полны: можно реализовать любую вычислимую функцию.
- RNN обладают определённым видом памяти и гораздо лучше подходят для работы с последовательностями, моделированием контекста и временными зависимостями.

Backpropagation through time (BPTT)

Для обучения RNN используется специальный вариант backpropagation: (backpropagation through time, BPTT) и “разворачивание” нейросети.

Из-за этого есть проблема с затуханием градиентов при большой глубине. Для её решения вместо простых нейронов используют более сложные ячейки памяти — LSTM или GRU.



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

Unfolding the RNN and training using BPTT

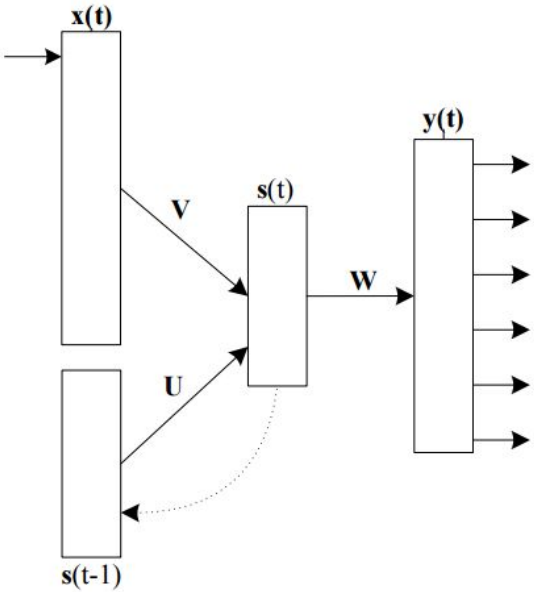


Figure 1: A simple recurrent neural network.

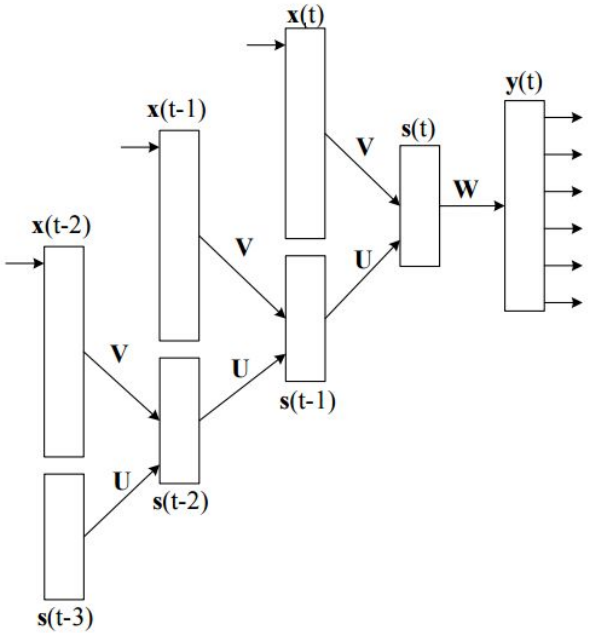


Figure 2: An unfolded recurrent neural network.

Can do backprop on the unfolded network: Backpropagation through time (BPTT)

http://ir.hit.edu.cn/~jguo/docs/notes/bptt.pdf

RNN problem: Vanishing gradients

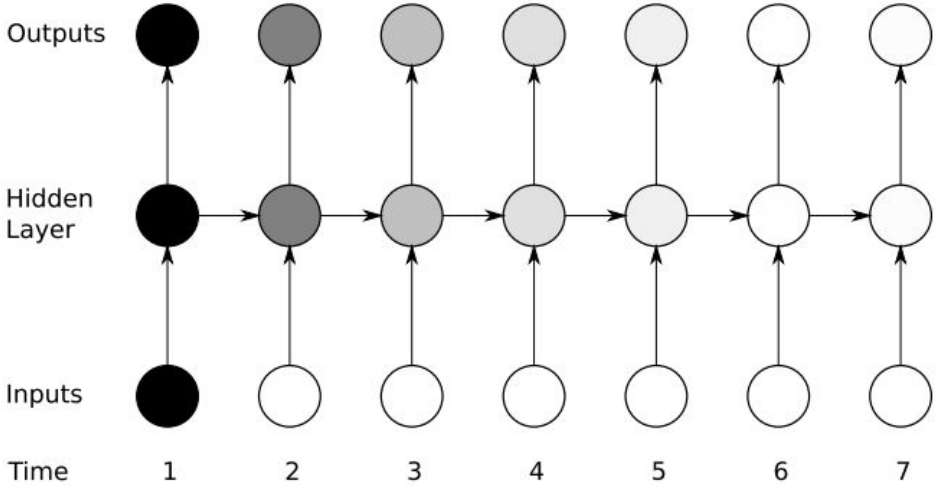
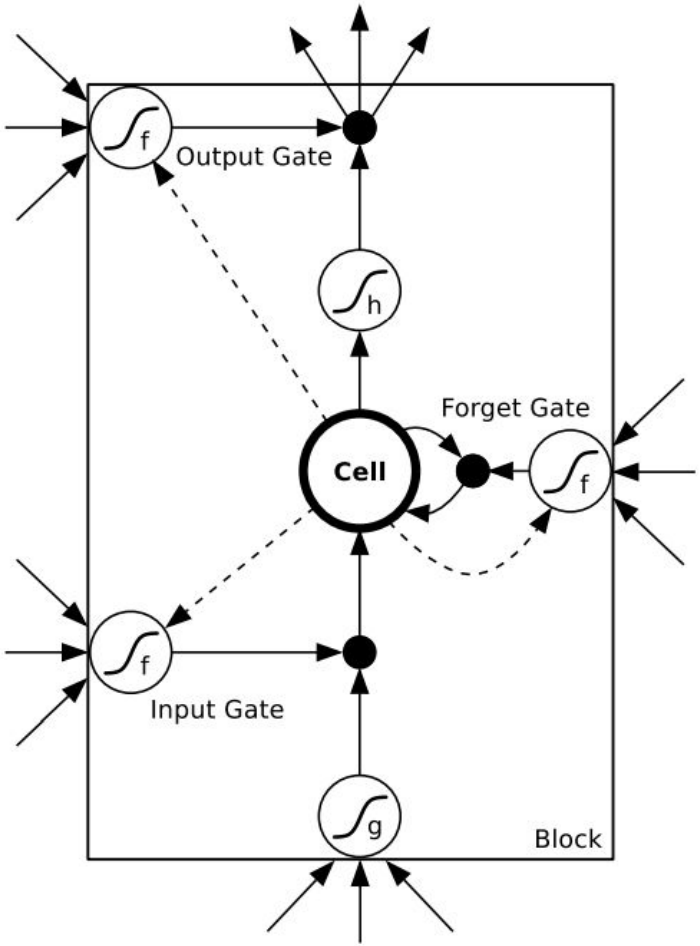


Figure 4.1: **The vanishing gradient problem for RNNs.** The shading of the nodes in the unfolded network indicates their sensitivity to the inputs at time one (the darker the shade, the greater the sensitivity). The sensitivity decays over time as new inputs overwrite the activations of the hidden layer, and the network 'forgets' the first inputs.

Solution: Long short-term memory (LSTM, Hochreiter, Schmidhuber, 1997)

LSTM cell



LSTM network

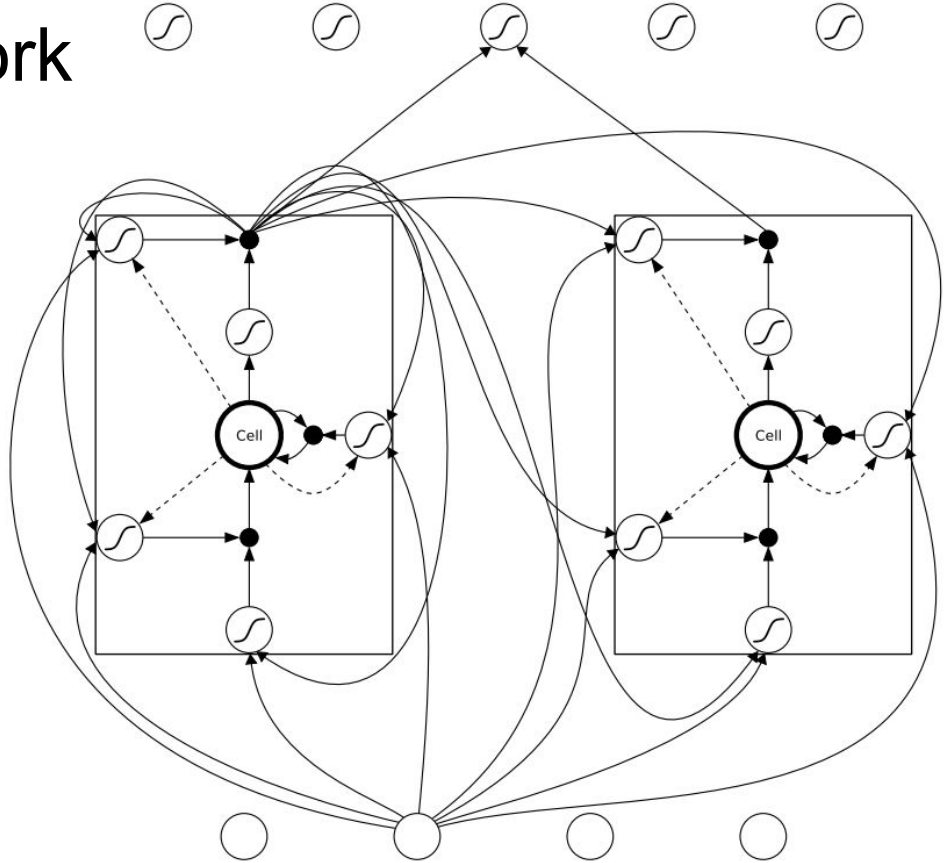


Figure 4.3: **An LSTM network.** The network consists of four input units, a hidden layer of two single-cell LSTM memory blocks and five output units. Not all connections are shown. Note that each block has four inputs but only one output.

Сравнение ячейки LSTM и обычного нейрона

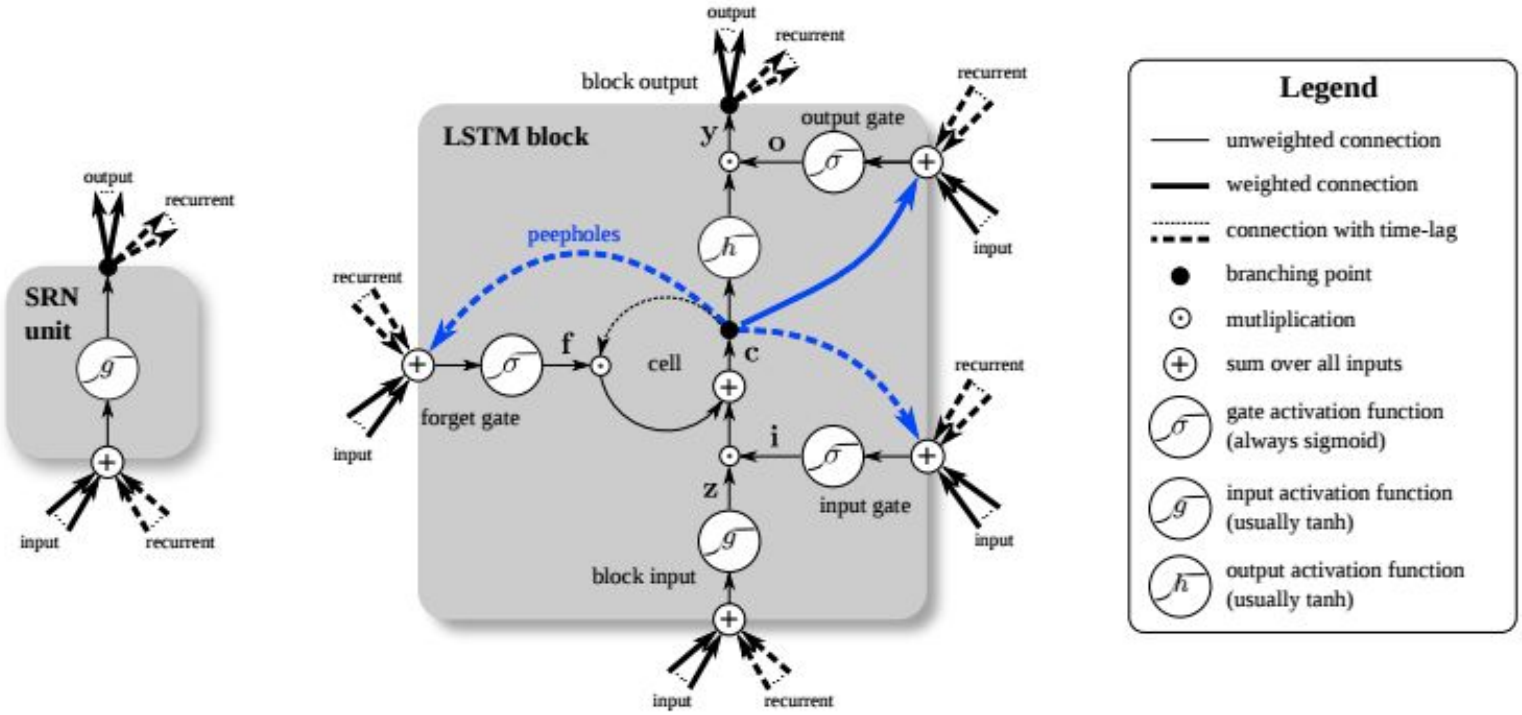


Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

https://deeplearning4j.org/lstm

LSTM: Fixing vanishing gradient problem

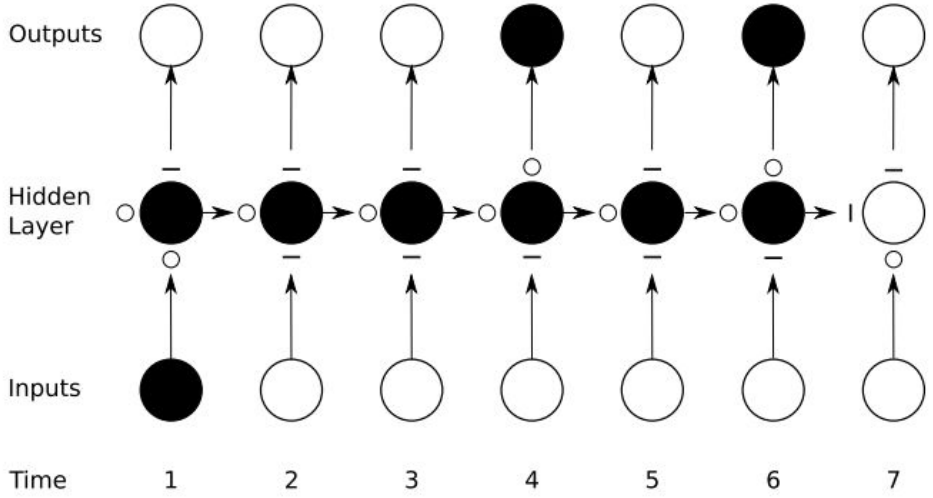
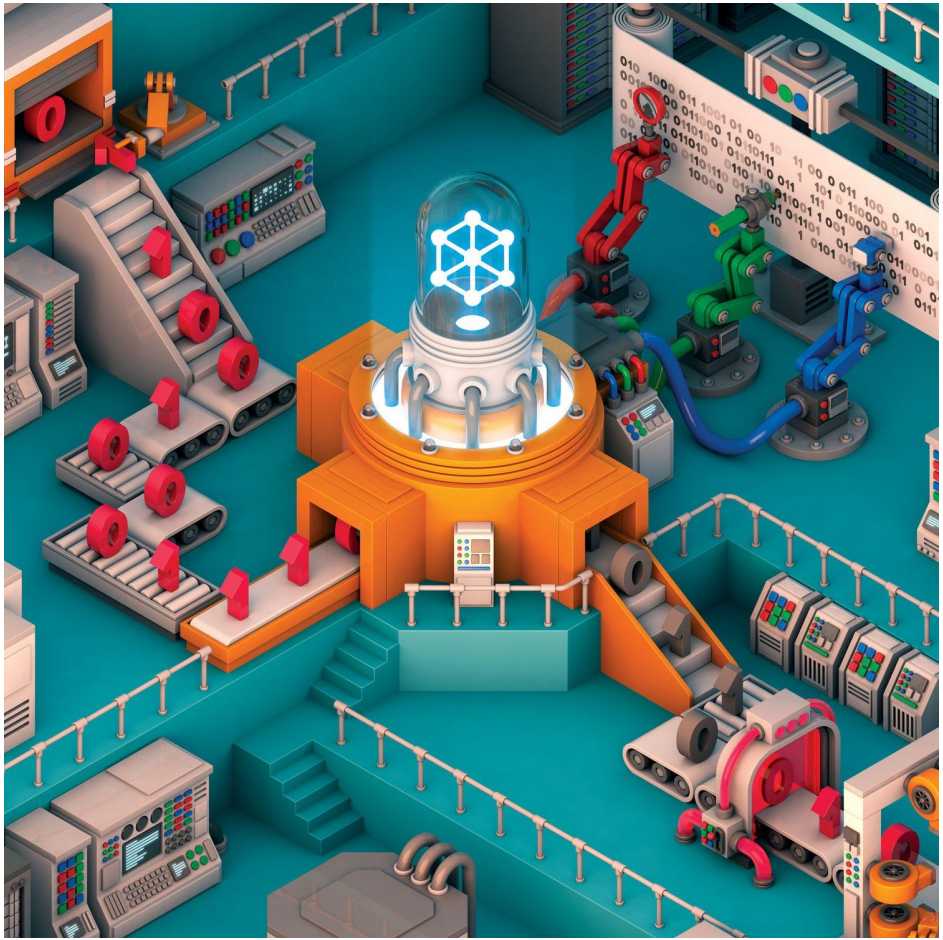
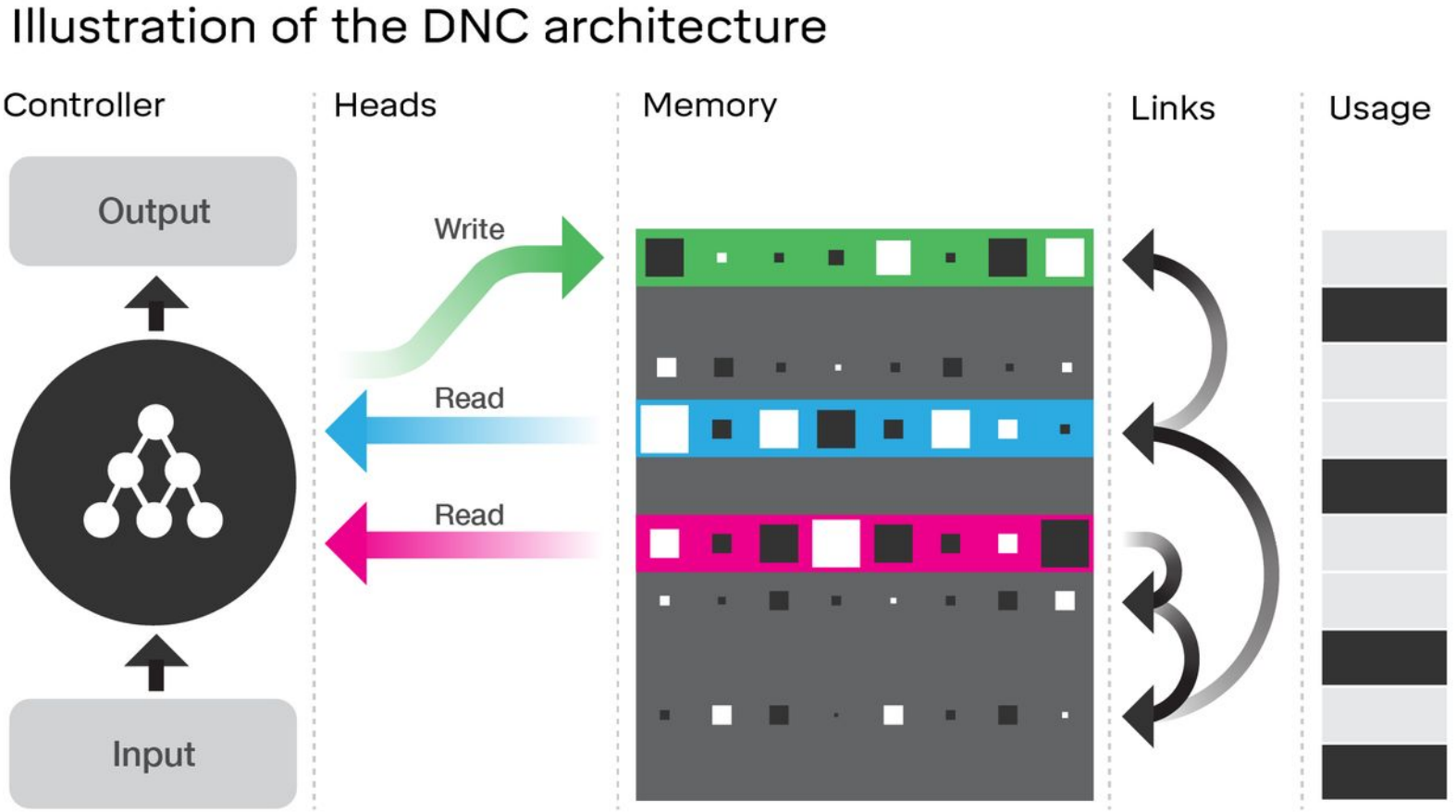


Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

More augmented RNNs

- Attentional Interfaces (Hard attention, Soft attention)
- Differentiable Memory (Neural Turing Machines, Differentiable neural computer, Hierarchical Attentive Memory, Memory Networks, ...)
- Adaptive Computation Time
- Differentiable Data Structures (structured memory: stack, list, queue, …)
- Differential Programming (Neural Programmer, Differentiable Functional Program Interpreters, …)
- ...



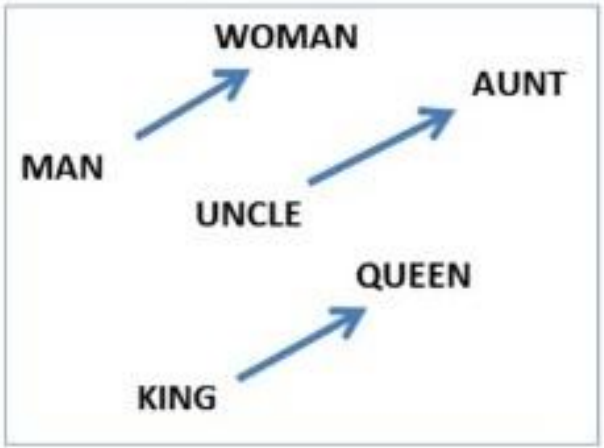


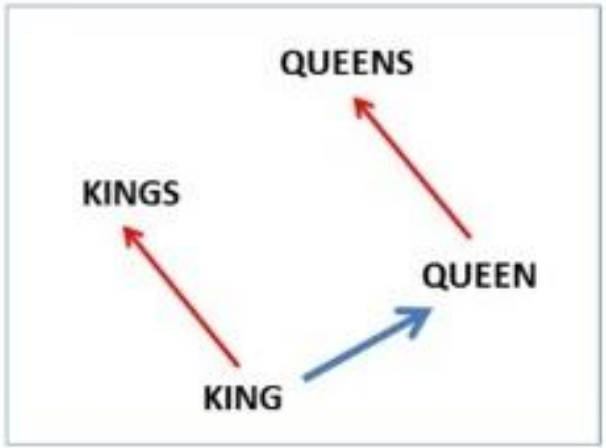
https://deepmind.com/blog/differentiable-neural-computers/

Мультимодальное обучение (Multimodal Learning)

Example: Semantic Spaces (word2vec, GloVe)

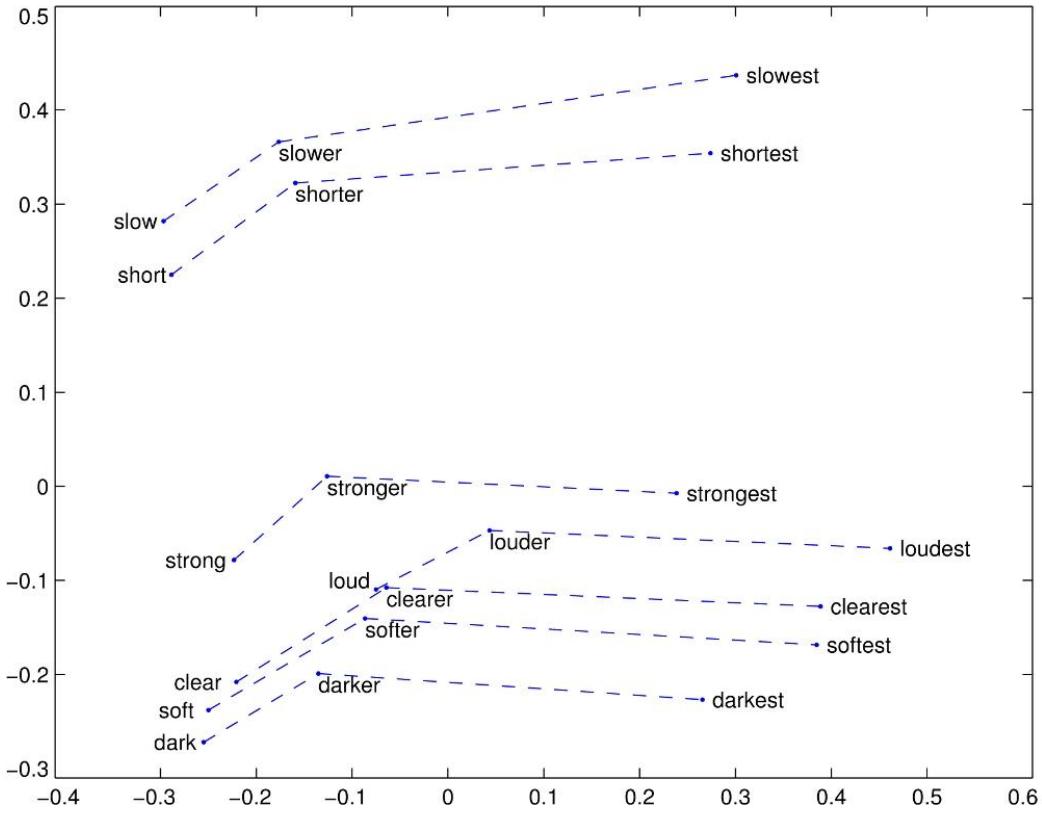
vec(“man”) – vec(“king”) + vec(“woman”) = vec(“queen”)





https://code.google.com/archive/p/word2vec/

Example: Semantic Spaces (word2vec, GloVe)



http://nlp.stanford.edu/projects/glove/

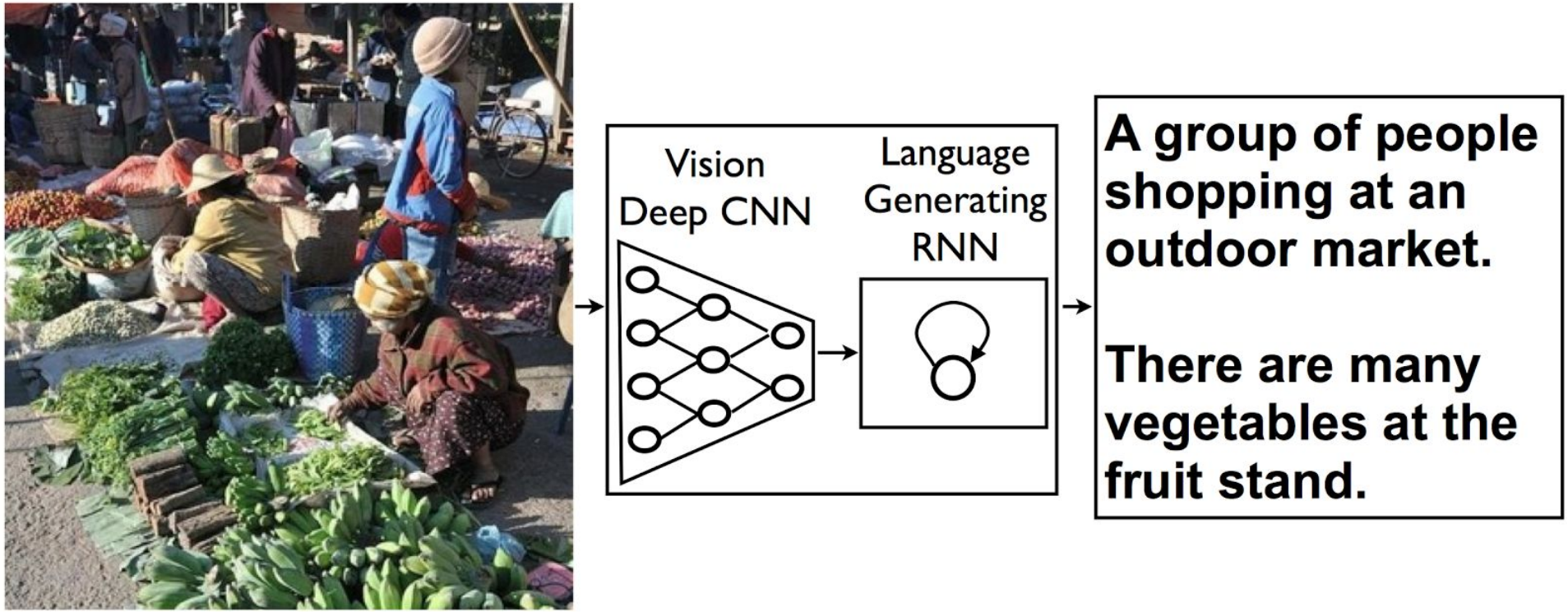
Encoding semantics

Using word2vec instead of word indexes allows you to better deal with the word meanings (e.g. no need to enumerate all synonyms because their vectors are already close to each other).

But the naive way to work with word2vec vectors still gives you a “bag of words” model, where phrases “The man killed the tiger” and “The tiger killed the man” are equal.

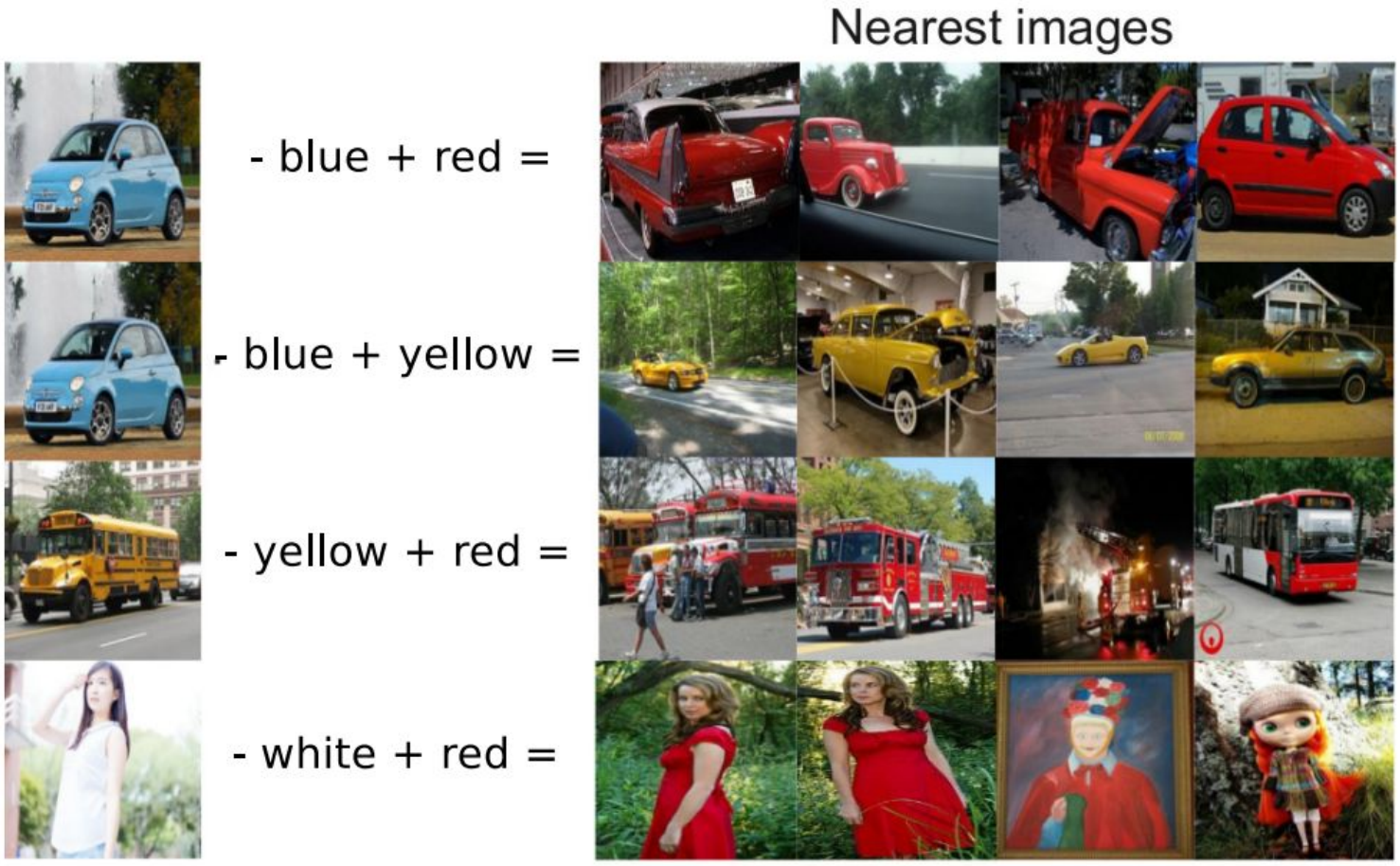
Need models which pay attention to the word ordering: paragraph2vec, sentence embeddings (using RNN/LSTM), even World2Vec (LeCunn @CVPR2015).

Генерация описаний картинок

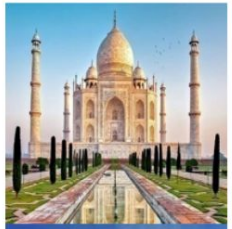


http://arxiv.org/abs/1411.4555 “Show and Tell: A Neural Image Caption Generator”

Example: More multi-modal learning



Nearest Images



- day + night =

- flying + sailing =

- bowl + box =

- box + bowl =

(Kiros, Salakhutdinov, Zemel, TACL 2015)

Мультимодальное обучение

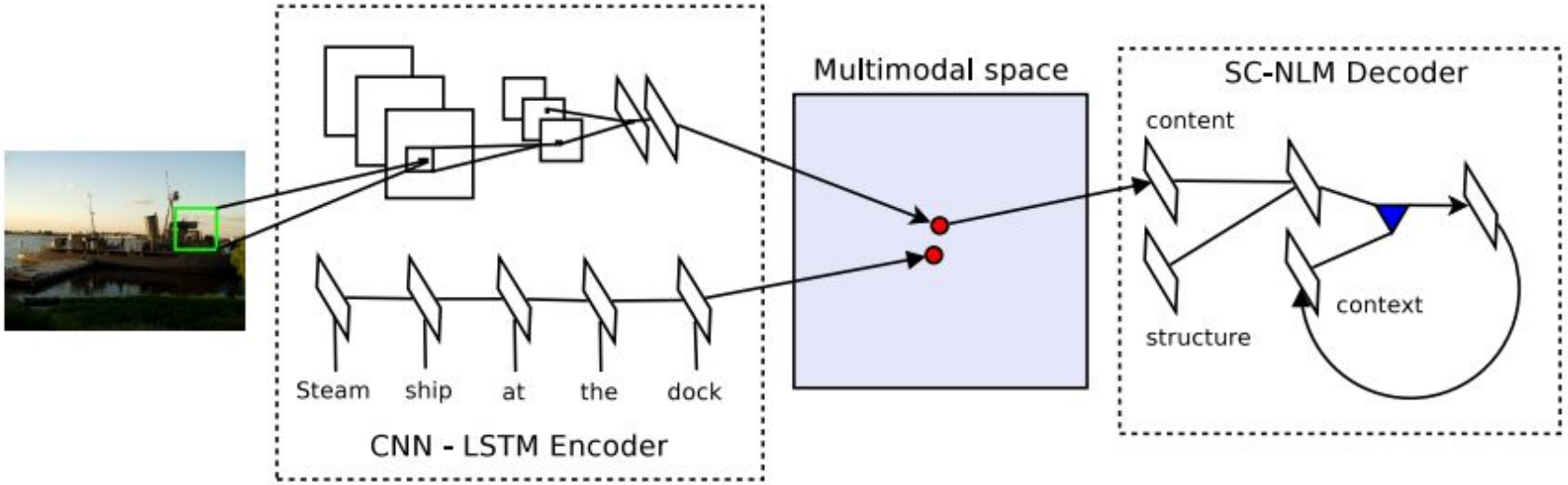


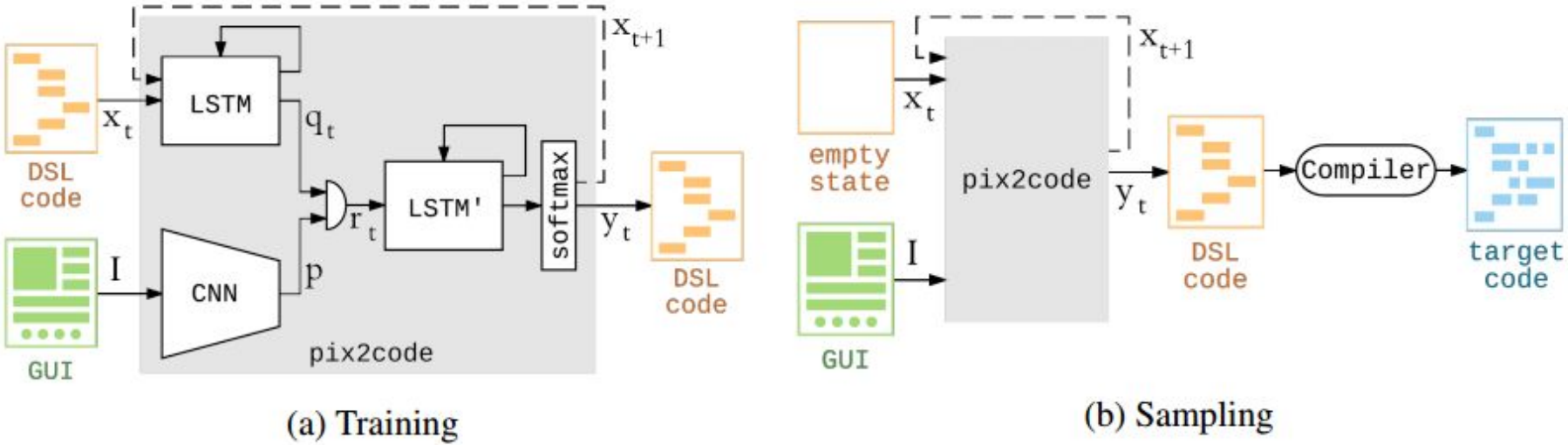
Figure 2: **Encoder:** A deep convolutional network (CNN) and long short-term memory recurrent network (LSTM) for learning a joint image-sentence embedding. **Decoder:** A new neural language model that combines structure and content vectors for generating words one at a time in sequence.

http://arxiv.org/abs/1411.2539 Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models

Example: Image generation by text

StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, https://arxiv.org/abs/1612.03242

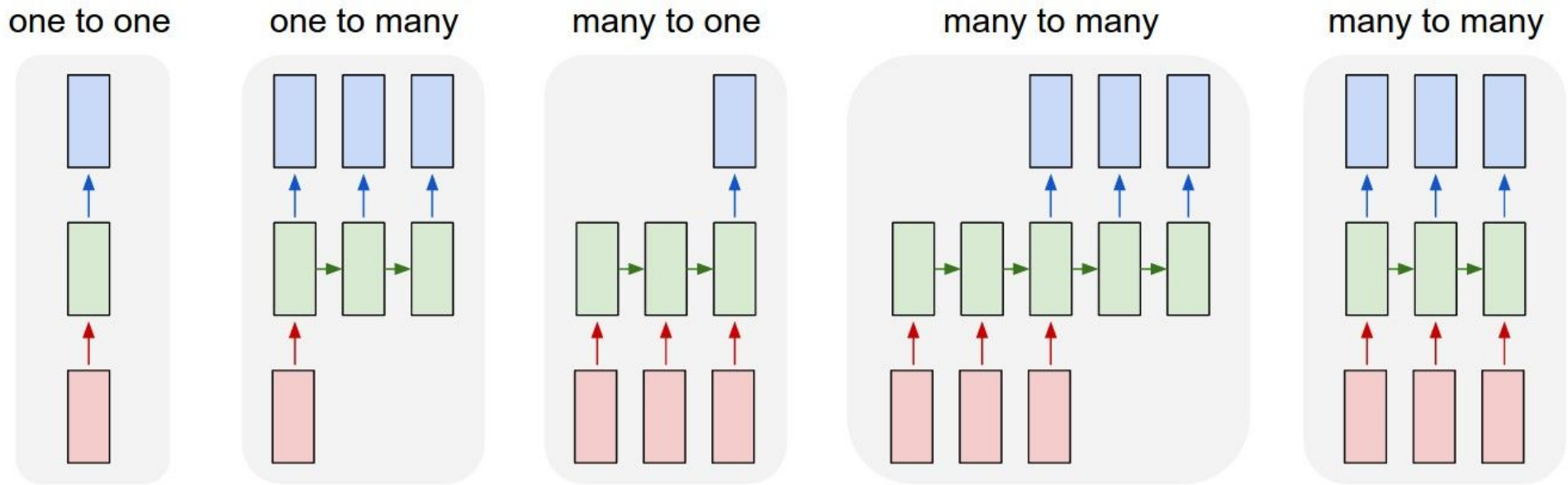
Example: Code generation by image



pix2code: Generating Code from a Graphical User Interface Screenshot,
https://arxiv.org/abs/1705.07962

Sequence Learning и парадигма seq2seq

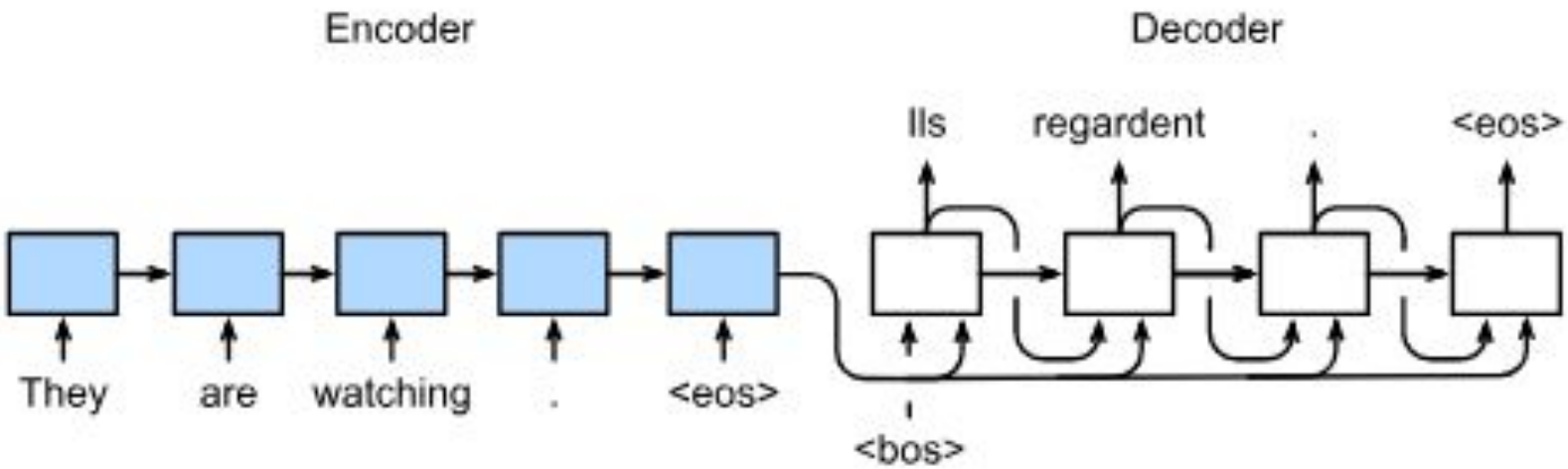
Sequence to Sequence Learning (seq2seq)



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Encoder-Decoder

Encoder-Decoder architecture



http://en.diveintodeeplearning.org/chapter_natural-language-processing/seq2seq.html

Encoder-Decoder architecture

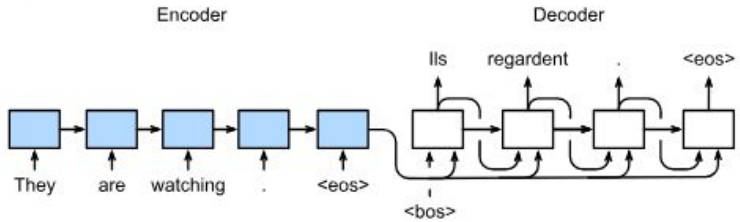
The encoder transforms the hidden state of each time step into **context variables** through custom function ***q***:

$$\boldsymbol{c} = q(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$$

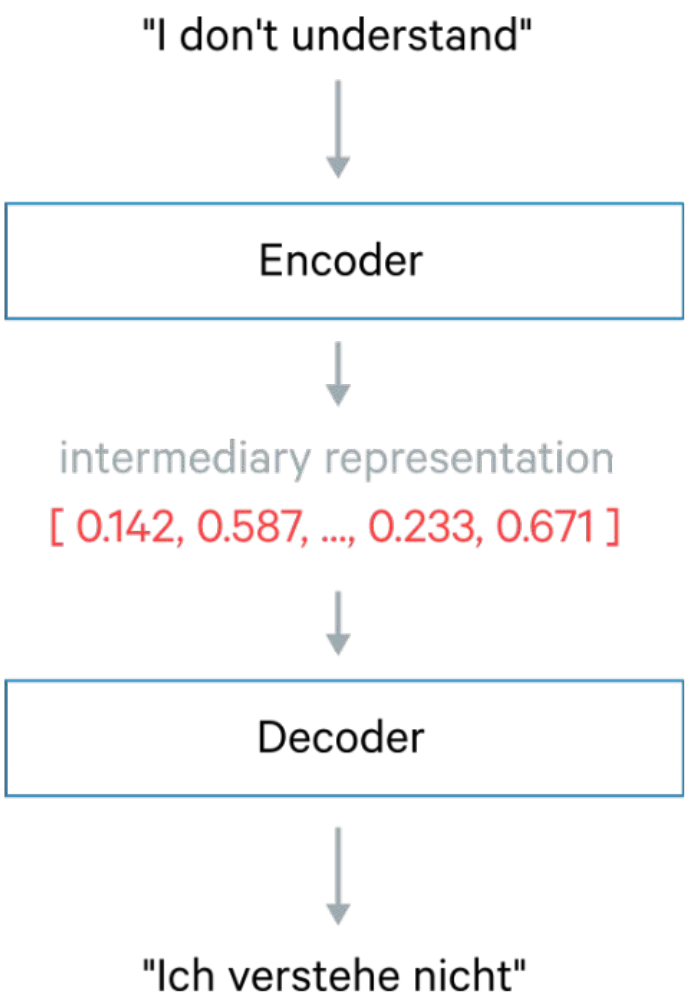
For example, when we select $q(\boldsymbol{h}_1,\ldots,\boldsymbol{h}_T)=\boldsymbol{h}_T$, the context variable is the hidden state of input sequence $\boldsymbol{h}_T$ for the final time step.

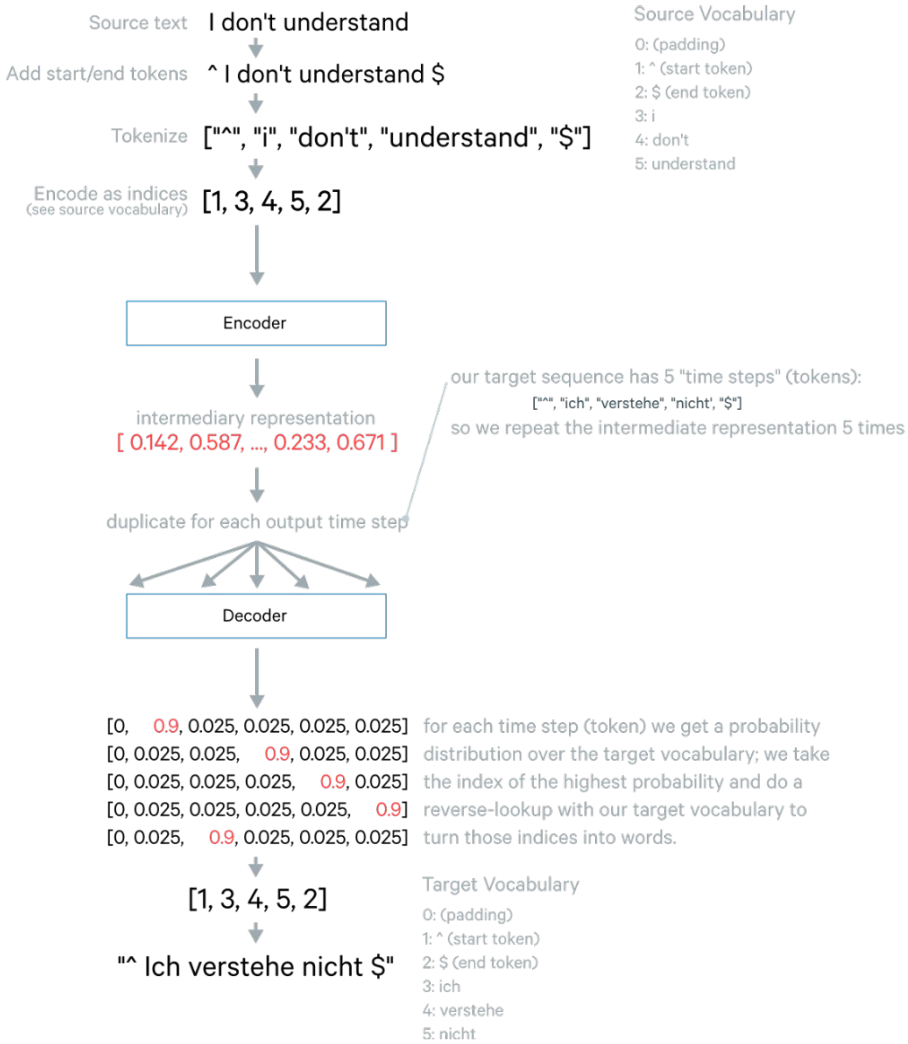
The transformation of the decoder's hidden layer:

$$\boldsymbol{s}_{t'} = g(y_{t'-1}, \boldsymbol{c}, \boldsymbol{s}_{t'-1})$$



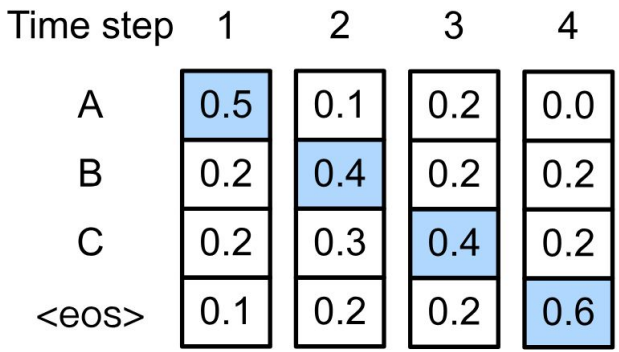
http://en.diveintodeeplearning.org/chapter_natural-language-processing/seq2seq.html


"I don't understand"
Encoder
intermediary representation
[0.142, 0.587, ..., 0.233, 0.671]
Decoder
"Ich verstehe nicht"

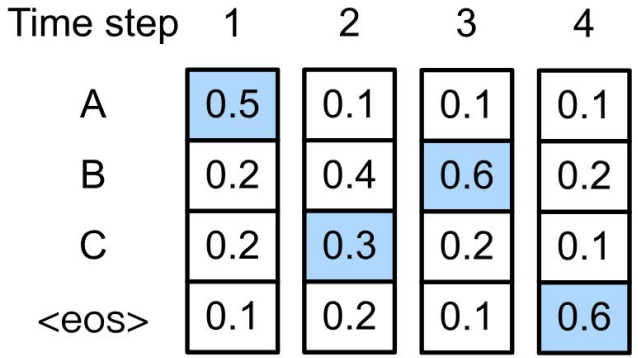

Source text
I don't understand
Add start/end tokens
^ I don't understand $
Tokenize
["^", "i", "don't", "understand", "$"]
Encode as indices
(see source vocabulary)
[1, 3, 4, 5, 2]
Source Vocabulary
0: (padding)
1: ^ (start token)
2: $ (end token)
3: i
4: don't
5: understand
Encoder
intermediary representation
[0.142, 0.587, ..., 0.233, 0.671]
duplicate for each output time step
our target sequence has 5 "time steps" (tokens):
["^", "ich", "verstehe", "nicht', "$"]
so we repeat the intermediate representation 5 times
Decoder
[0, 0.9, 0.025, 0.025, 0.025, 0.025]
[0, 0.025, 0.025, 0.9, 0.025, 0.025]
[0, 0.025, 0.025, 0.025, 0.9, 0.025]
[0, 0.025, 0.025, 0.025, 0.025, 0.9]
[0, 0.025, 0.9, 0.025, 0.025, 0.025]
for each time step (token) we get a probability distribution over the target vocabulary; we take the index of the highest probability and do a reverse-lookup with our target vocabulary to turn those indices into words.
[1, 3, 4, 5, 2]
"^ Ich verstehe nicht $"
Target Vocabulary
0: (padding)
1: ^ (start token)
2: $ (end token)
3: ich
4: verstehe
5: nicht

Decoding: Greedy search

Greedy search is not optimal. Exhaustive Search is not possible.



P = 0.5×0.4×0.4×0.6 = 0.048

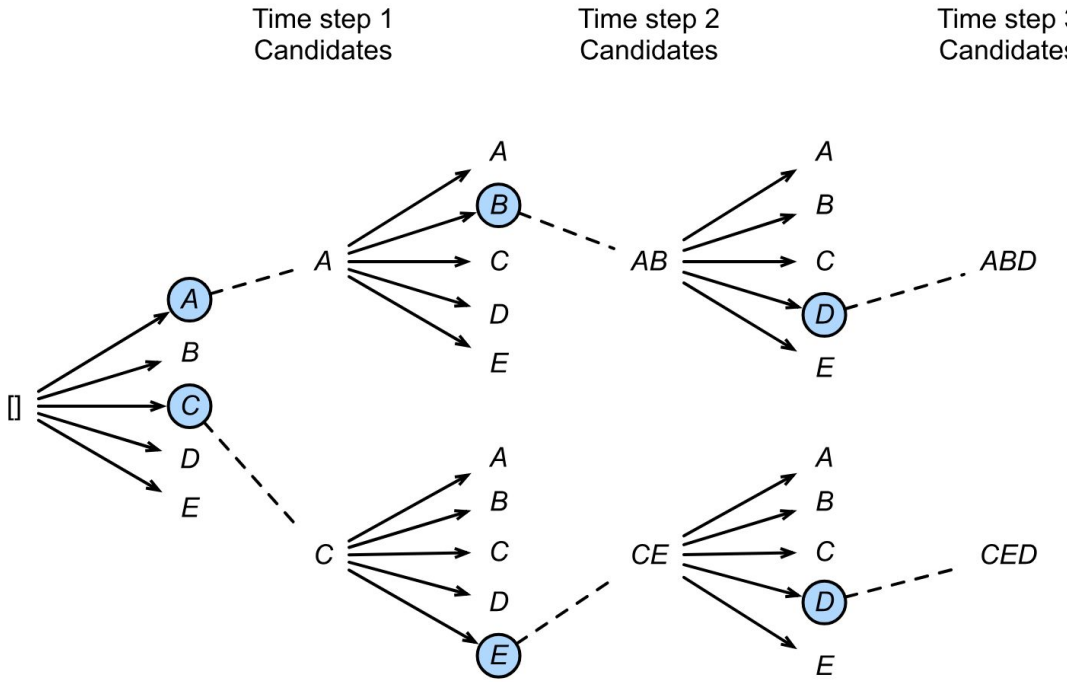


P = 0.5×0.3×0.6×0.6 = 0.054

http://en.diveintodeeplearning.org/chapter_natural-language-processing/beam-search.html

Decoding: Beam search

Beam search can give better solutions. Hyper-parameter: beam size.



http://en.diveintodeeplearning.org/chapter_natural-language-processing/beam-search.html

Neuraltalk2

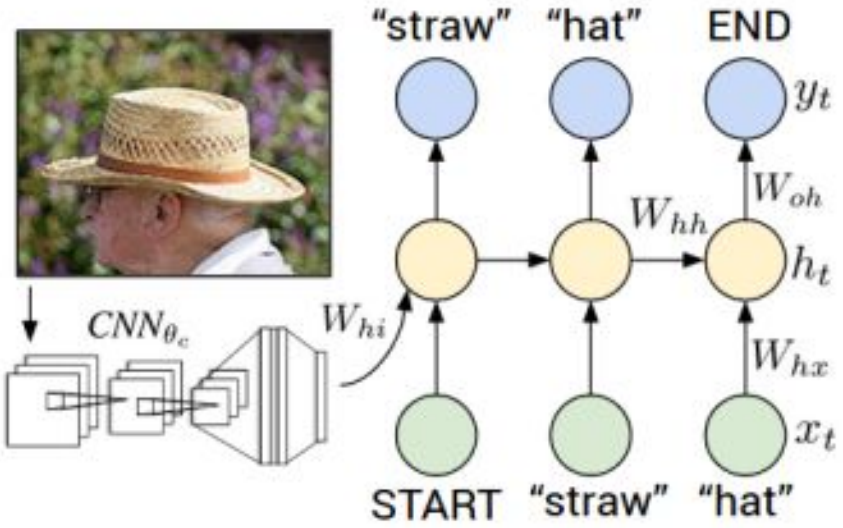


Figure 4. Diagram of our multimodal Recurrent Neural Network generative model. The RNN takes a word, the context from previous time steps and defines a distribution over the next word in the sentence. The RNN is conditioned on the image information at the first time step. START and END are special tokens.

http://cs.stanford.edu/people/karpathy/deepimagesent/

Deep Visual-Semantic Alignments for Generating Image Descriptions

Neuraltalk2

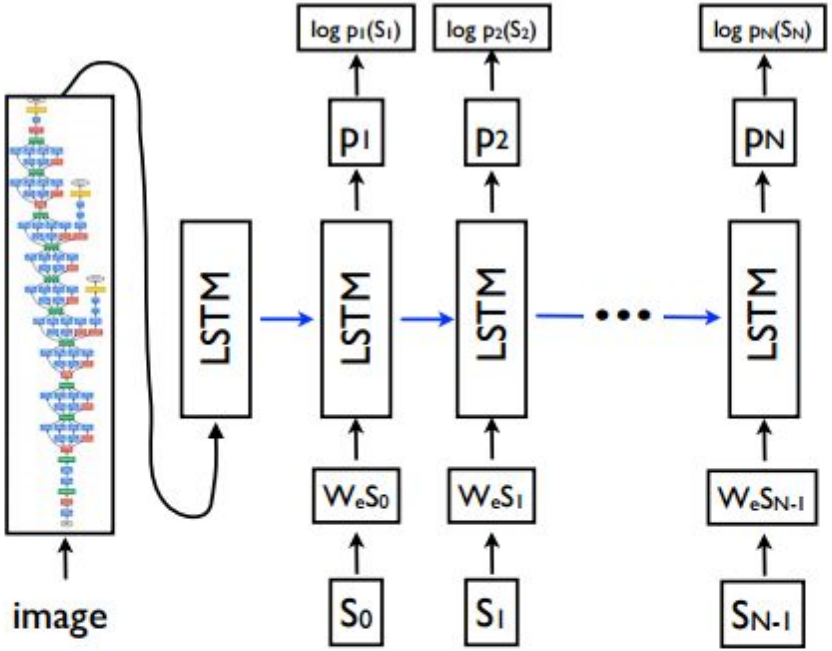


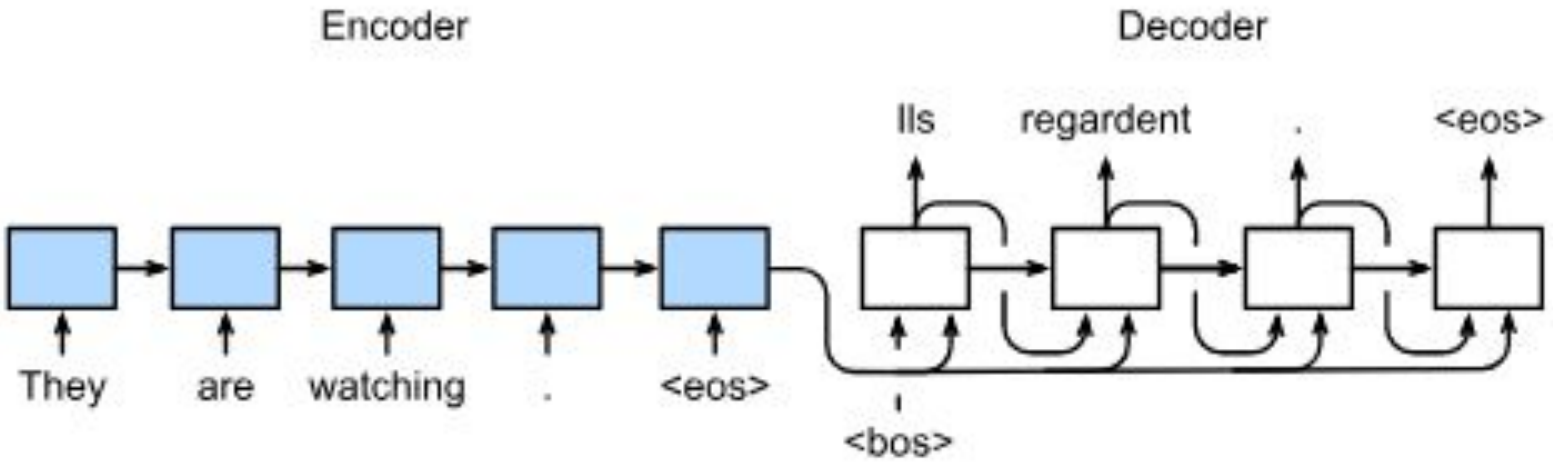
Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

https://arxiv.org/abs/1411.4555 Show and Tell: A Neural Image Caption Generator

Attention mechanisms

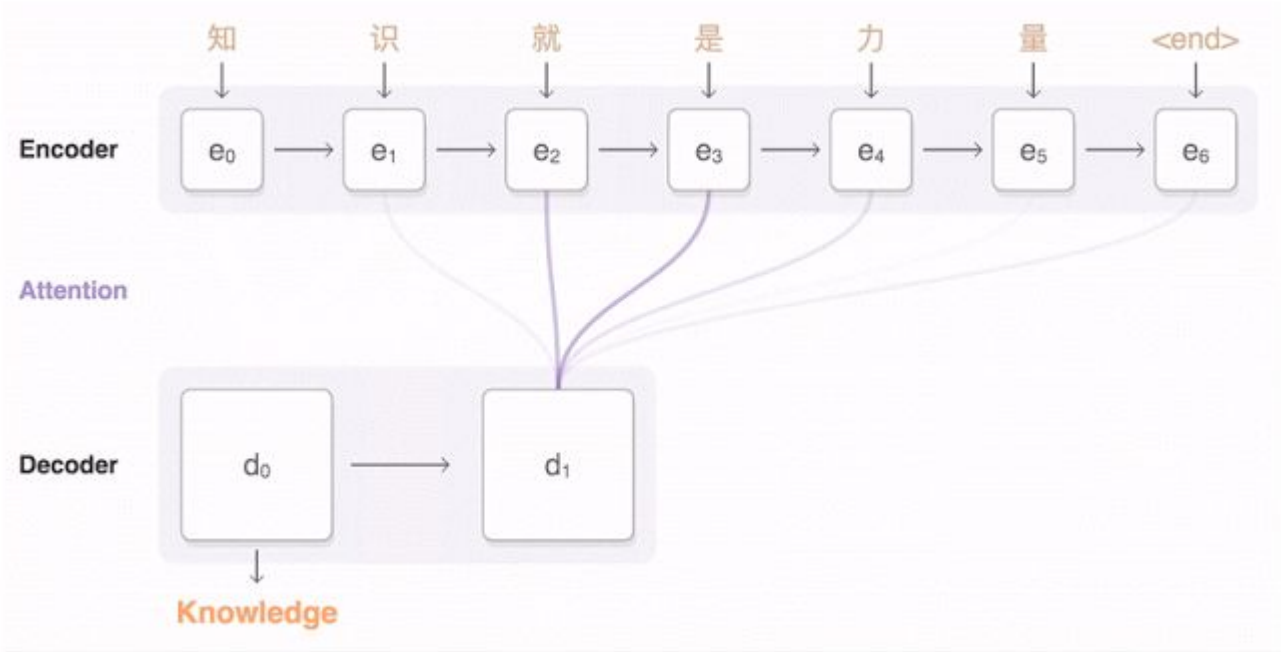
Encoder-Decoder shortcomings

Encoder-Decoder can be applied to N-to-M sequence, yet an Encoder reads and encodes a source sentence into a fixed-length vector. Is one hidden state really enough? A neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector.



Encoder-Decoder with Attention

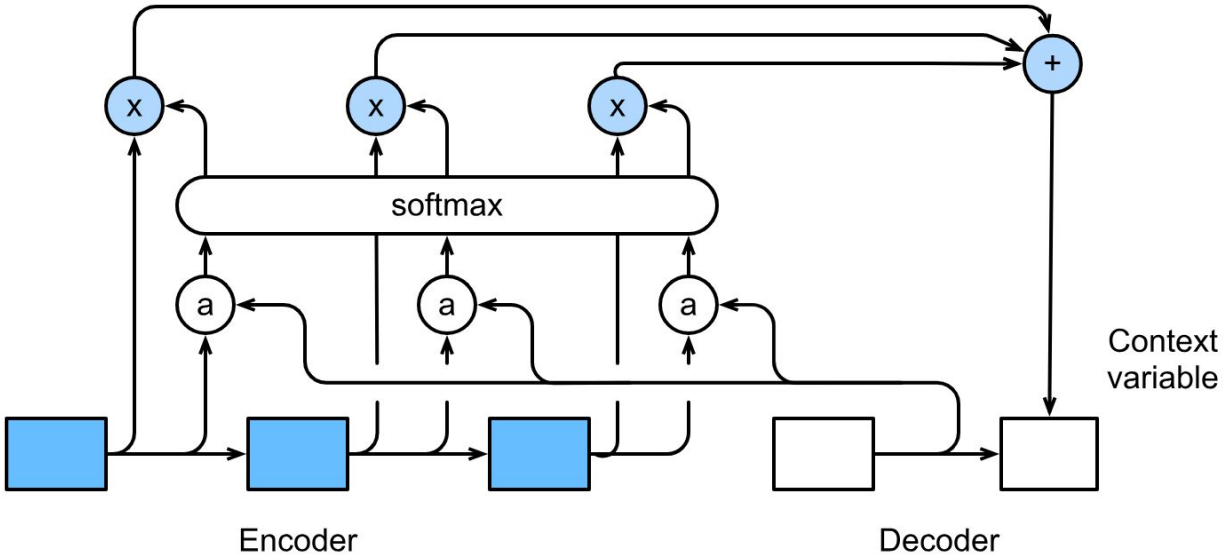
Attention Mechanism allows the decoder to attend to different parts of the source sentence at each step of the output generation.



https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

Attention Mechanism

The attention mechanism obtains the context variable by weighting the hidden state of all time steps of the encoder.

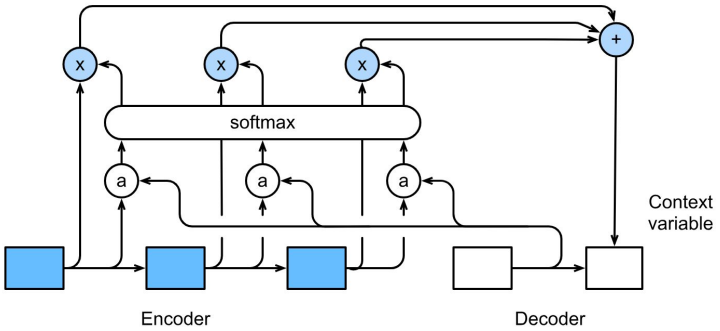


http://en.diveintodeeplearning.org/chapter_natural-language-processing/attention.html

Attention Mechanism

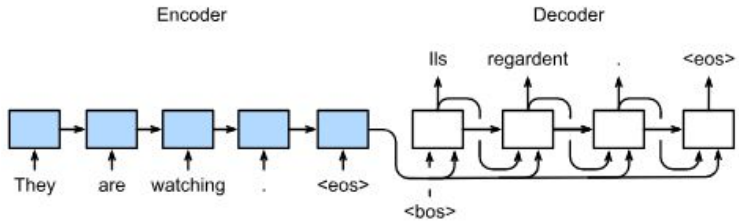
Now the hidden state of the decoder at time step ***t'*** can be rewritten as:

$$\boldsymbol{s}_{t'} = g(\boldsymbol{y}_{t'-1}, \boldsymbol{c}_{t'}, \boldsymbol{s}_{t'-1})$$



Compare to simple encoder-decoder:

$$\boldsymbol{s}_{t'} = g(y_{t'-1}, \boldsymbol{c}, \boldsymbol{s}_{t'-1})$$



http://en.diveintodeeplearning.org/chapter_natural-language-processing/attention.html

Attention Mechanism

The context variable of the decoder at time step ***t'*** is the weighted average on all the hidden states of the encoder ($\boldsymbol{h}_t$).

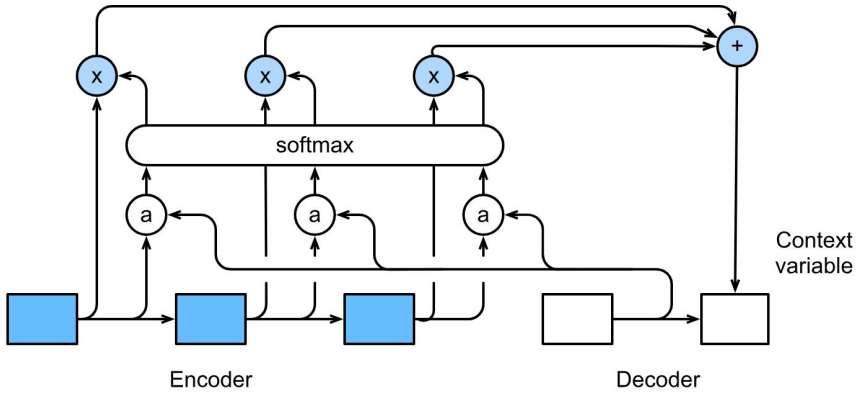
$\boldsymbol{s}_{t'-1}$: decoder's hidden state at that time step ***t'−1***

$$\boldsymbol{c}_{t'} = \sum_{t=1}^{T} \alpha_{t't} \boldsymbol{h}_t$$

$$\alpha_{t't} = \frac{\exp(e_{t't})}{\sum_{k=1}^{T} \exp(e_{t'k})}, \quad t = 1, \ldots, T$$

$$e_{t't} = a(\boldsymbol{s}_{t'-1}, \boldsymbol{h}_t)$$

$$a(\boldsymbol{s}, \boldsymbol{h}) = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_s \boldsymbol{s} + \boldsymbol{W}_h \boldsymbol{h})$$



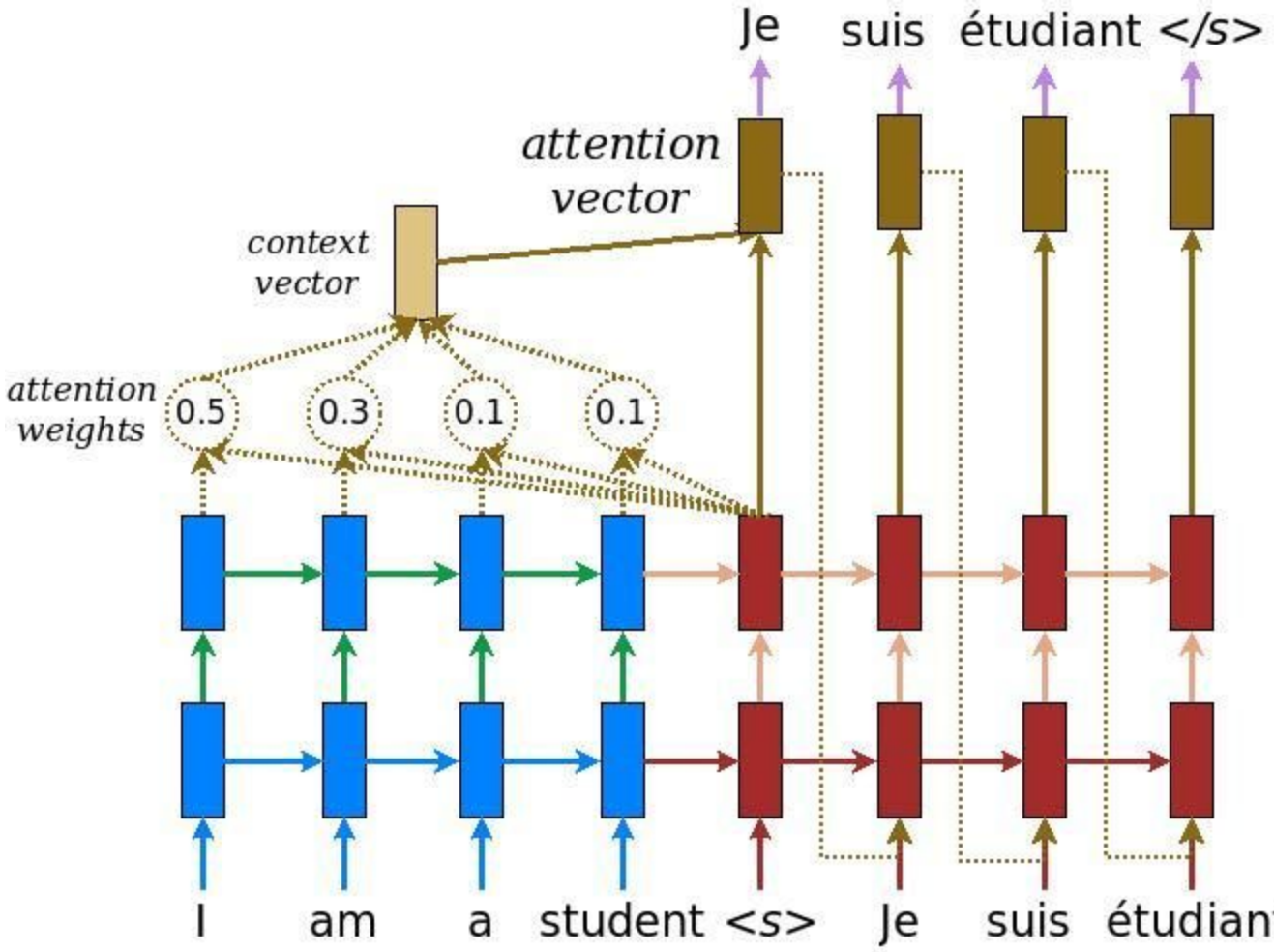
http://en.diveintodeeplearning.org/chapter_natural-language-processing/attention.html

Attention Mechanism

Attention is simply a vector, often the outputs of dense layer using softmax function.

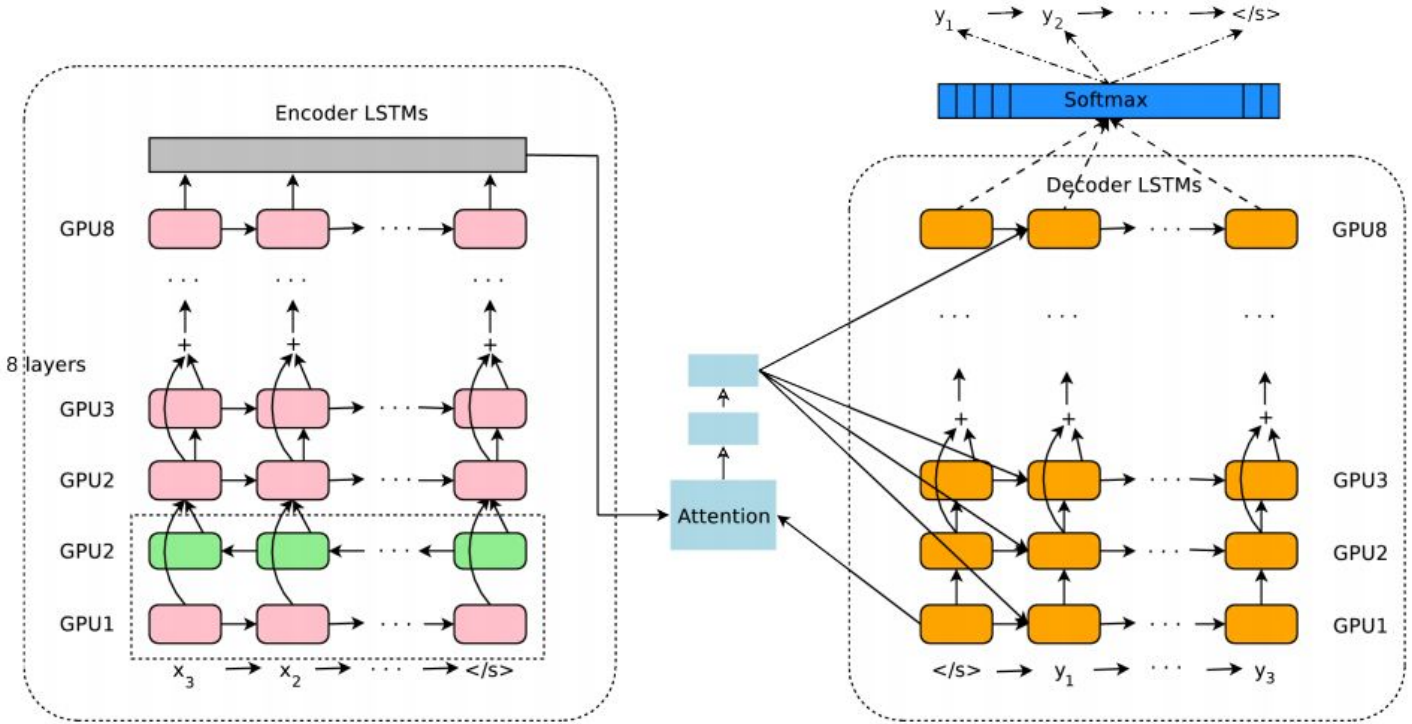
It is just an interface, you could plug it anywhere you find it suitable.

Can use different functions: tahn, sigmoid, dot product, learned transformations



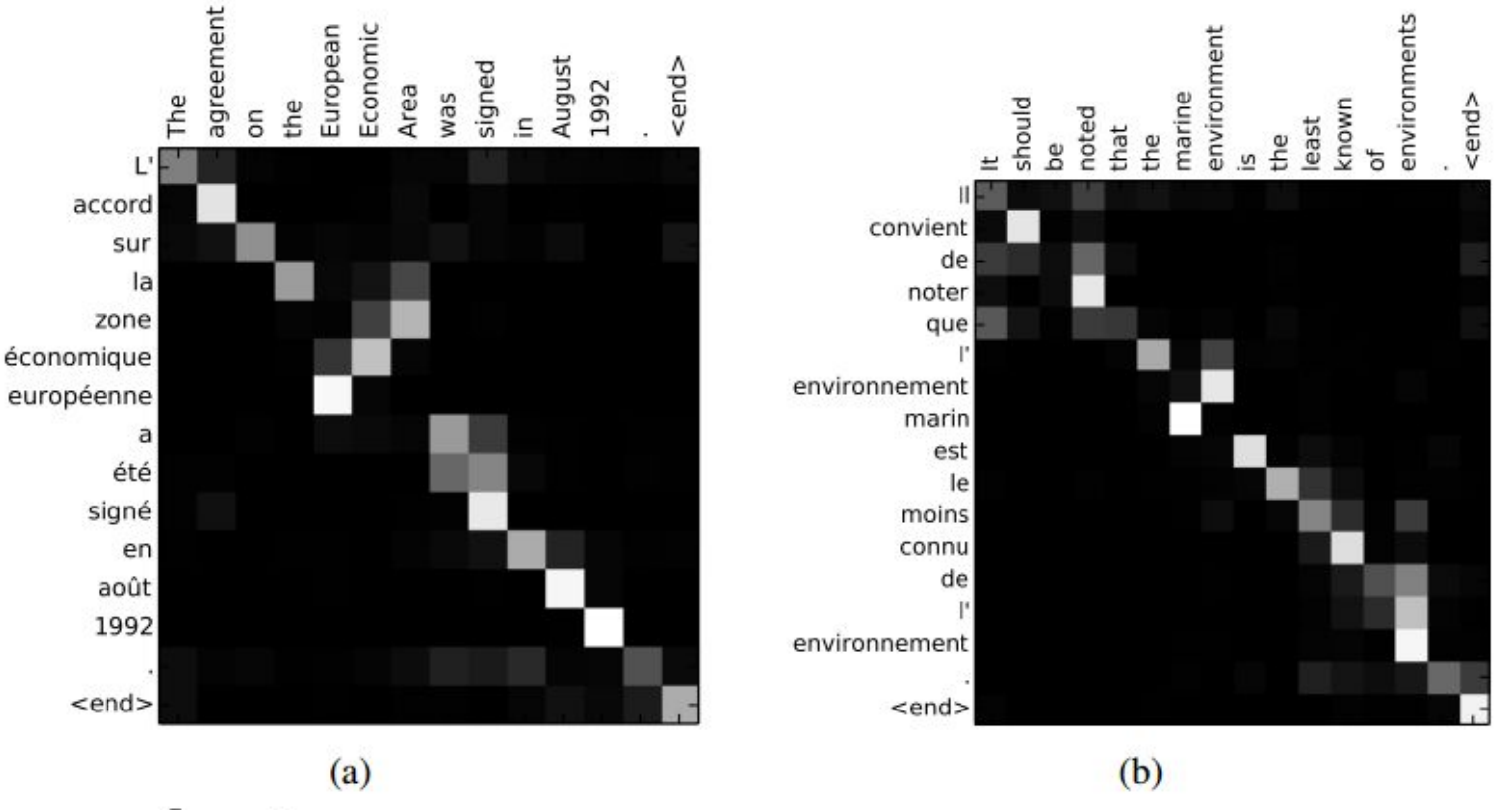
https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

Encoder-Decoder: modern architecture



Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation,
https://arxiv.org/abs/1609.08144
Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation
https://arxiv.org/abs/1611.04558

Visualizing RNN attention weights α_{ij} on MT



Neural Machine Translation by Jointly Learning to Align and Translate, https://arxiv.org/abs/1409.0473

Visualizing RNN attention heat maps on QA

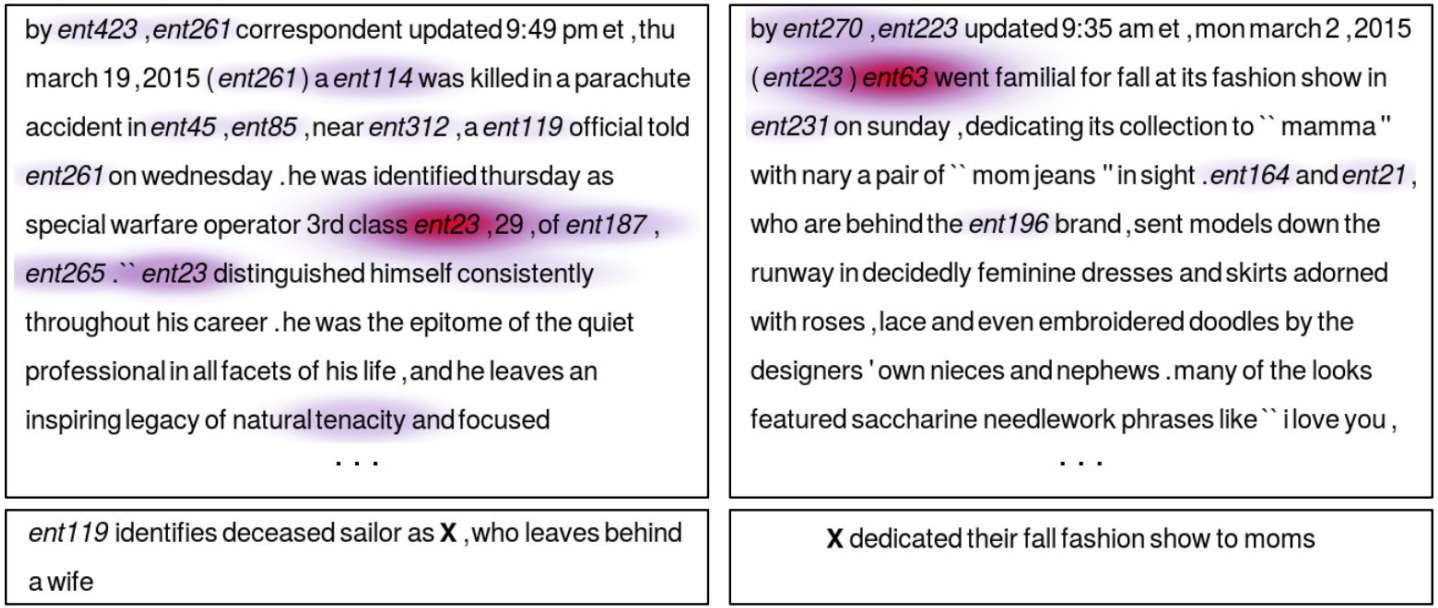
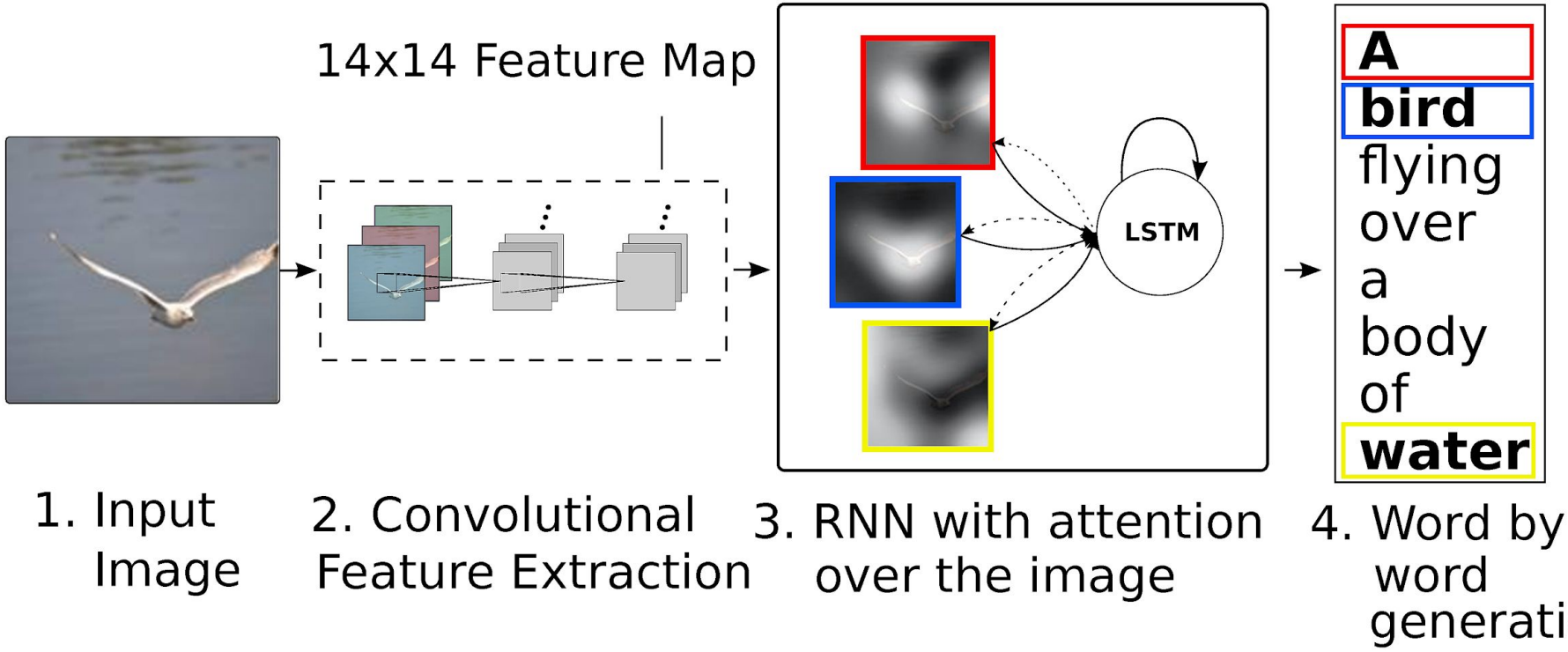


Figure 3: Attention heat maps from the Attentive Reader for two correctly answered validation set queries (the correct answers are *ent23* and *ent63*, respectively). Both examples require significant lexical generalisation and co-reference resolution in order to be answered correctly by a given model.

Teaching Machines to Read and Comprehend, https://arxiv.org/abs/1506.03340

CNN+RNN with Attention



http://kelvinxu.github.io/projects/capgen.html

CNN+RNN with Attention

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

A woman is throwing a frisbee in a park.

http://kelvinxu.github.io/projects/capgen.html

Hard vs. Soft attention

Soft means differentiable, Hard means non-differentiable.

The mechanism described previously is called **Soft attention** because it is a fully differentiable deterministic mechanism that can be plugged into an existing system, and the gradients are propagated through the attention mechanism at the same time they are propagated through the rest of the network.

Hard attention is a stochastic process: instead of using all the hidden states as an input for the decoding, the system samples a hidden state y_i with the probabilities s_i. In order to propagate a gradient through this process, we estimate the gradient by Monte Carlo sampling. Alternatively you can use Reinforcement Learning approaches.

https://blog.heuritech.com/2016/01/20/attention-mechanism/
https://jhui.github.io/2017/03/15/Soft-and-hard-attention/

Self-attention (Intra-Attention)

Each element in the sentence attends to other elements. It gives context sensitive encodings.

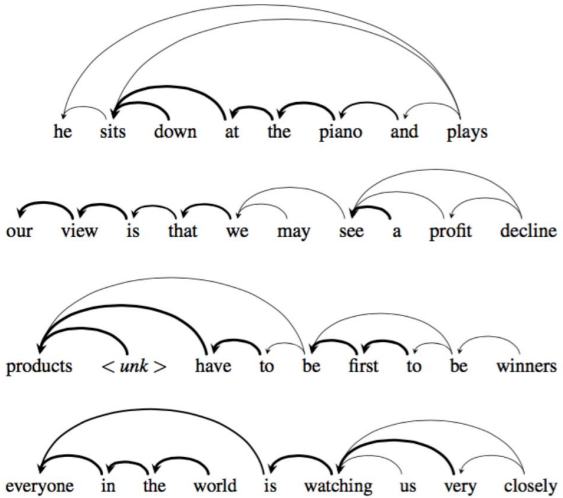


Figure 4: Examples of intra-attention (language modeling). Bold lines indicate higher attention scores. Arrows denote which word is being focused when attention is computed, but not the direction of the relation.

Long Short-Term Memory-Networks for Machine Reading, https://arxiv.org/abs/1601.06733

Convolutional Sequence Learning

CNN encoder

Convolutional Encoder / Recurrent Decoder

Encoder	Words/s	BLEU
BiLSTM	139.7	22.4
Deep Conv. 6/3	187.9	23.1

(a) IWSLT'14 German-English generation speed on *tst2013* with beam size 10.

Encoder	Words/s	BLEU
2-layer BiLSTM	109.9	23.6
Deep Conv. 8/4	231.1	23.7
Deep Conv. 15/5	203.3	24.0

(b) WMT'15 English-German generation speed on *newstest2015* with beam size 5.

Table 3: Generation speed in source words per second on a single CPU core.

A Convolutional Encoder Model for Neural Machine Translation, https://arxiv.org/abs/1611.02344

CNN encoder

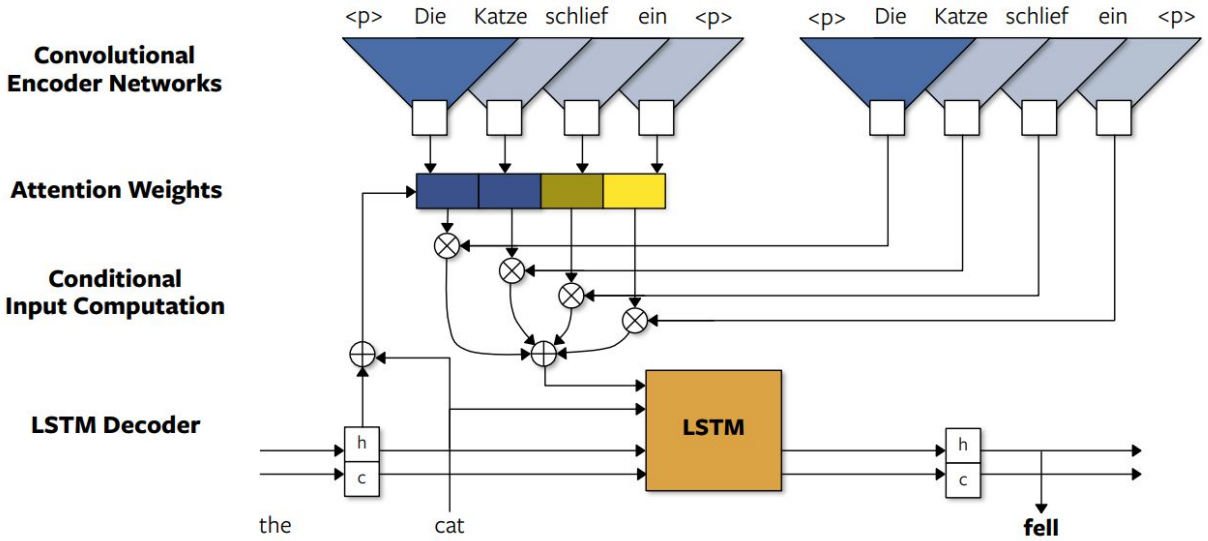


Figure 1: Neural machine translation model with single-layer convolutional encoder networks. CNN-a is on the left and CNN-c is at the right. Embedding layers are not shown.

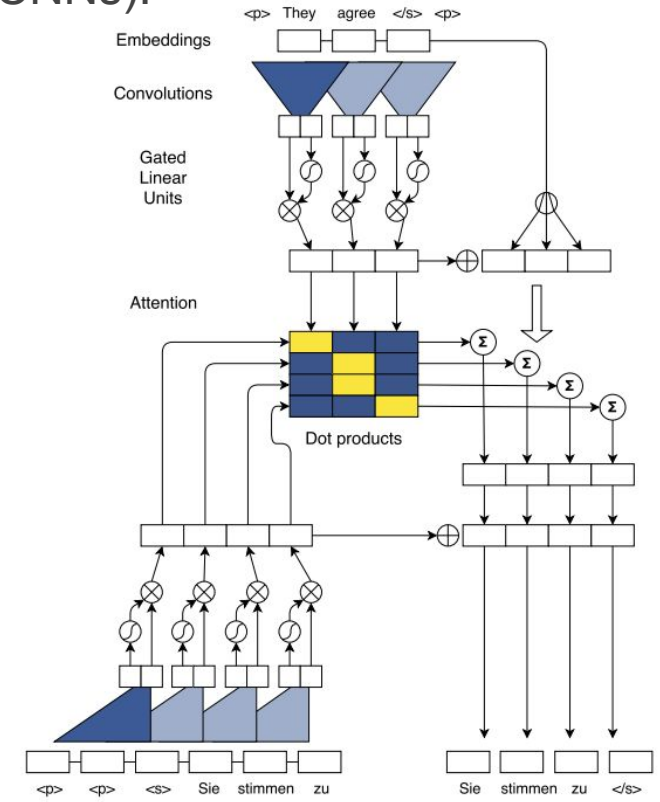
A Convolutional Encoder Model for Neural Machine Translation, https://arxiv.org/abs/1611.02344

CNN encoder + decoder

Actually no RNN here (Facebook AI Research loves CNNs).

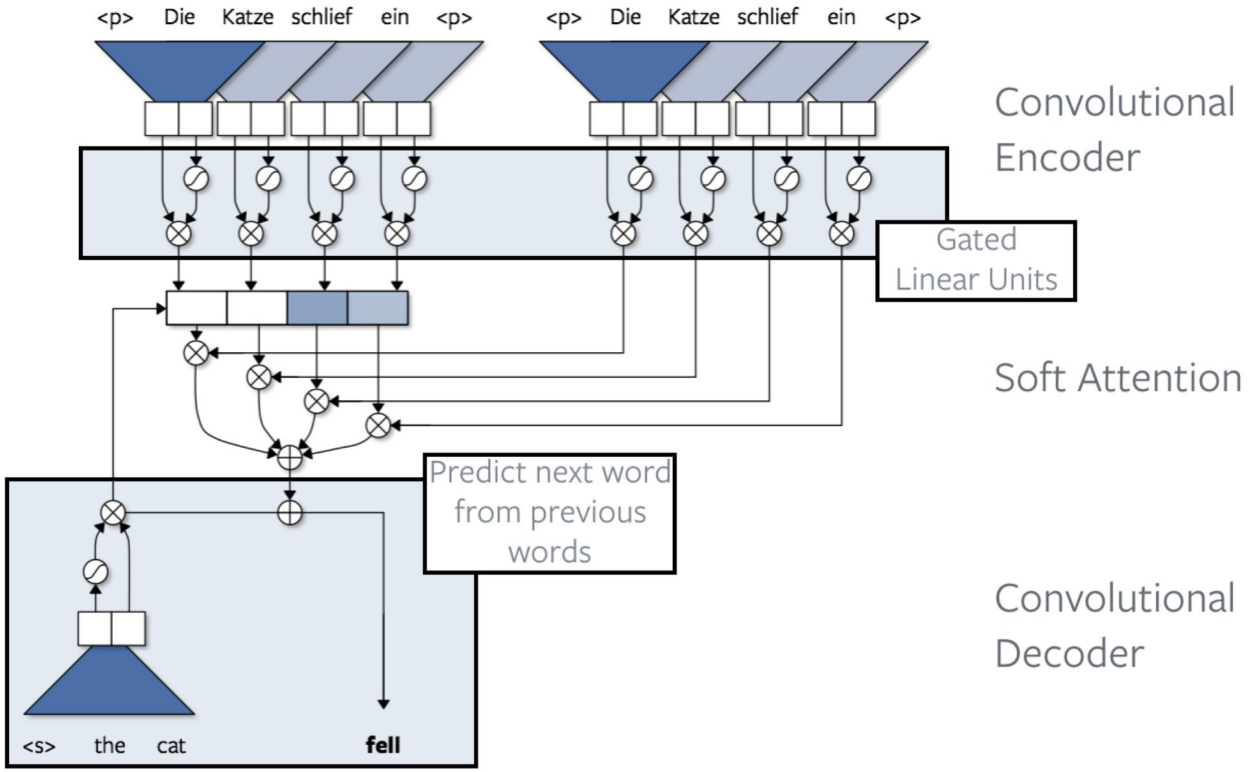
	BLEU	Time (s)
GNMT GPU (K80)	31.20	3,028
GNMT CPU 88 cores	31.20	1,322
GNMT TPU	31.21	384
ConvS2S GPU (K40) $b = 1$	33.45	327
ConvS2S GPU (M40) $b = 1$	33.45	221
ConvS2S GPU (GTX-1080ti) $b = 1$	33.45	142
ConvS2S CPU 48 cores $b = 1$	33.45	142
ConvS2S GPU (K40) $b = 5$	34.10	587
ConvS2S CPU 48 cores $b = 5$	34.10	482
ConvS2S GPU (M40) $b = 5$	34.10	406
ConvS2S GPU (GTX-1080ti) $b = 5$	34.10	256

Table 3. CPU and GPU generation speed in seconds on the development set of WMT'14 English-French. We show results for different beam sizes b. GNMT figures are taken from Wu et al. (2016). CPU speeds are not directly comparable because Wu et al. (2016) use a 88 core machine compared to our 48 core setup.



Convolutional Sequence to Sequence Learning, https://arxiv.org/abs/1705.03122

CNN encoder + decoder



https://aiukraine.com/wp-content/uploads/2017/10/2_1-Yarats.pdf

Self-Attention Neural Networks (SAN): Transformer Architecture

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

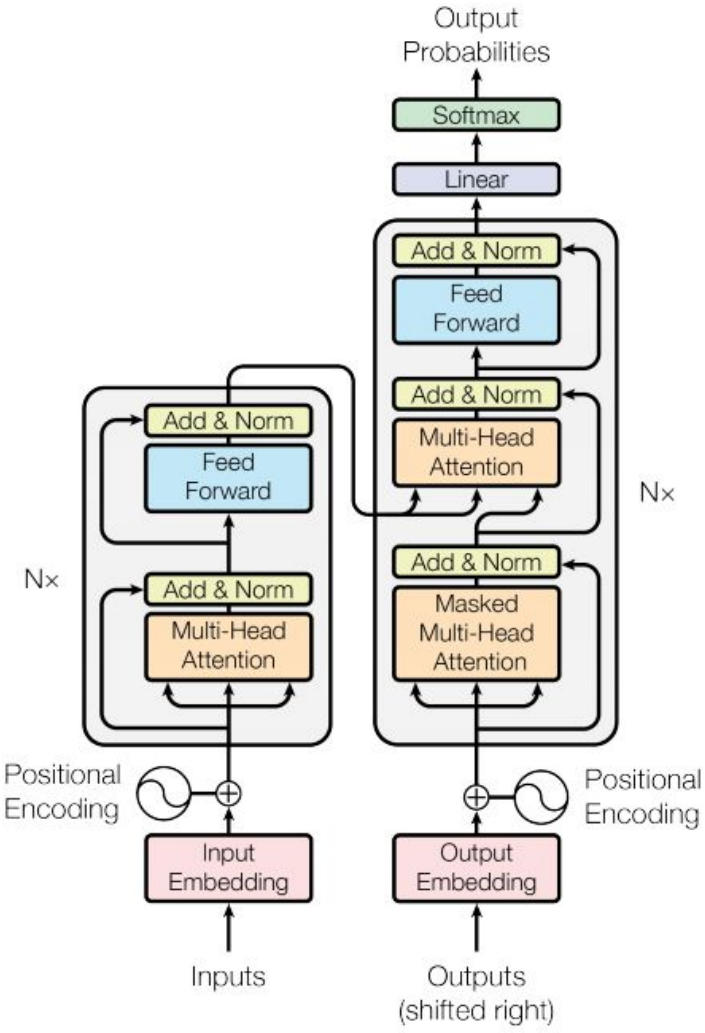
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, 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.

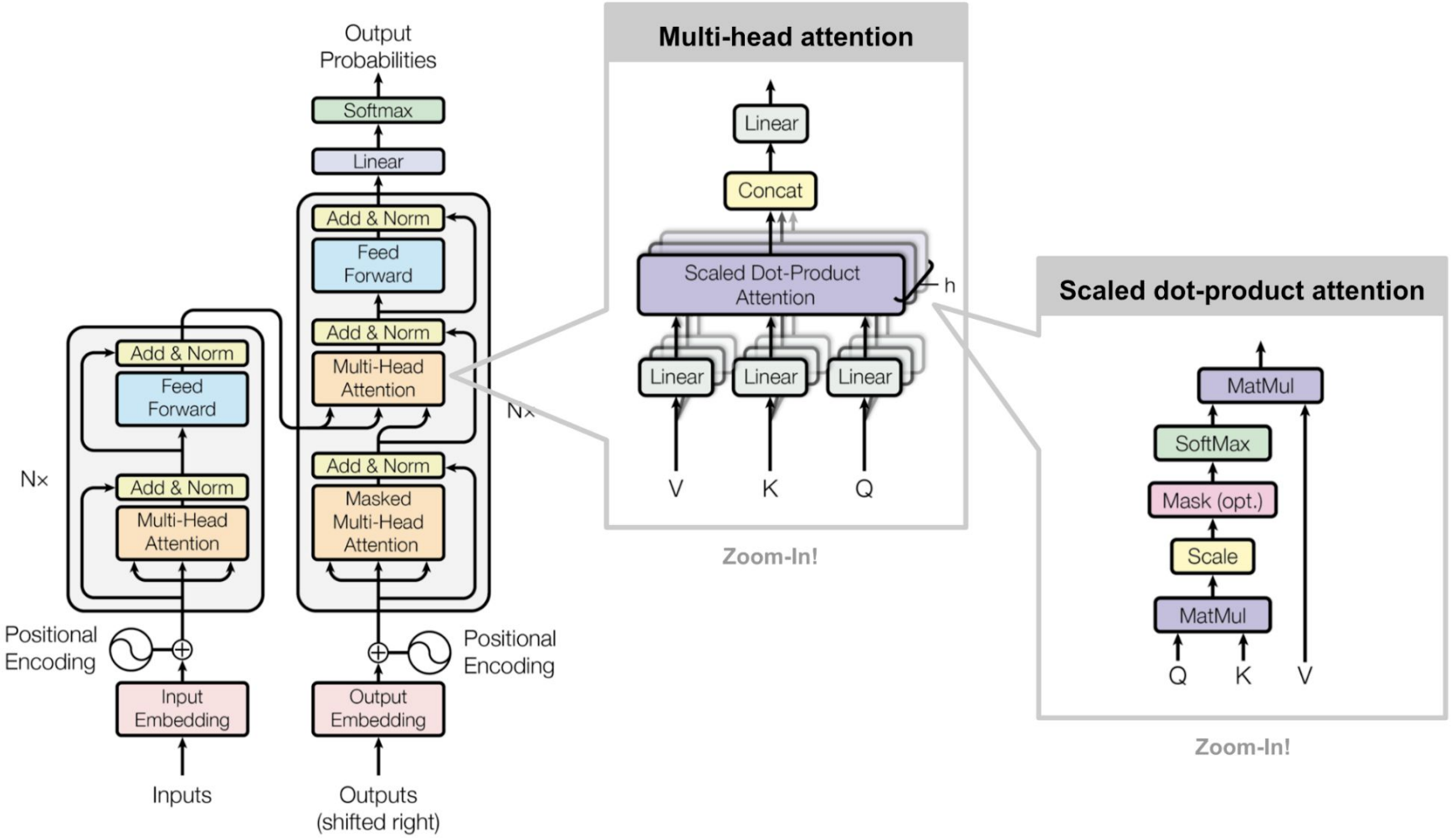
Attention Is All You Need, https://arxiv.org/abs/1706.03762

Transformer

A new simple network architecture,
the Transformer:

- Is a Encoder-Decoder architecture
- Based solely on attention mechanisms (no RNN/CNN)
- The major component in the transformer is the unit of multi-head self-attention mechanism.
- Fast: only matrix multiplications
- Strong results on standard WMT datasets





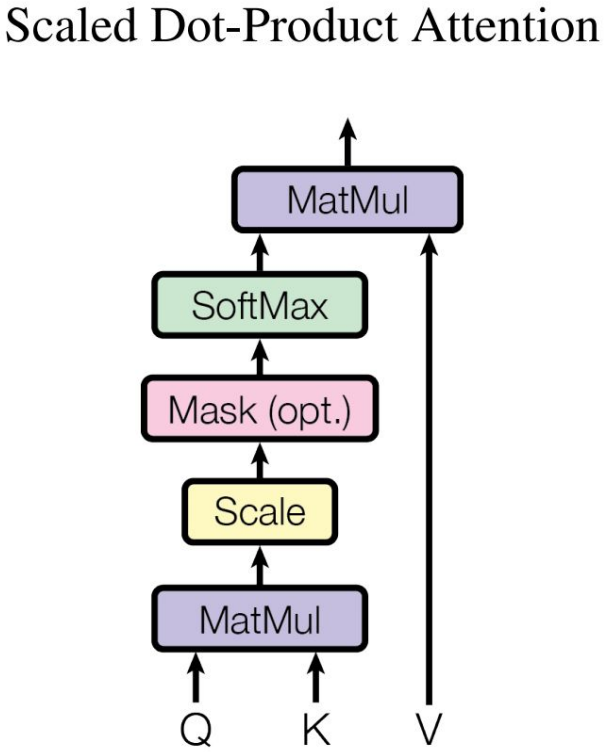
https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Scaled dot-product attention

The transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

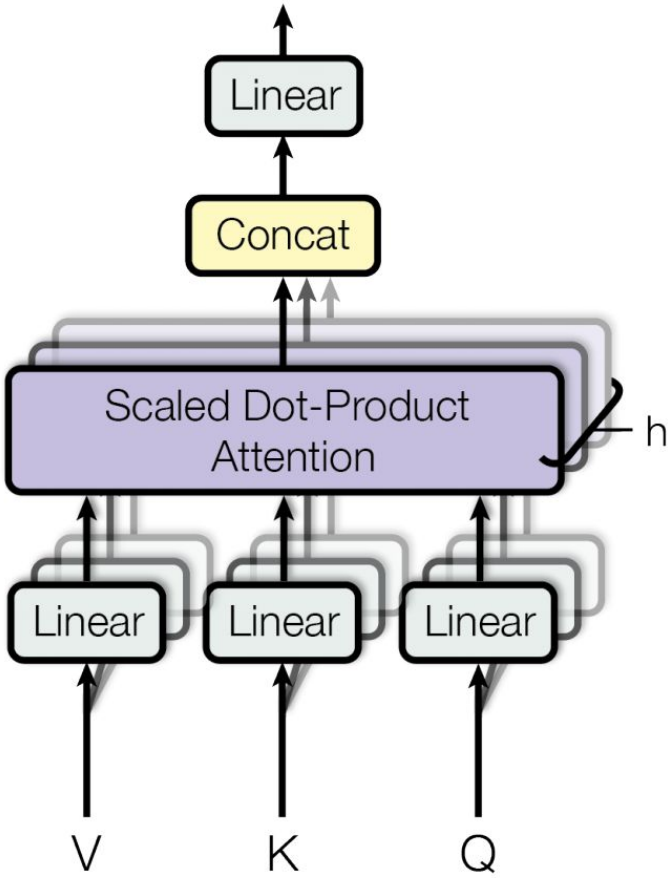
The input consists of queries and keys of dimension d_k, and values of dimension d_v.



Multi-head self-attention mechanism

Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel. The independent attention outputs are simply concatenated and linearly transformed into the expected dimensions.

According to the paper, “multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.”



Multi-head self-attention mechanism

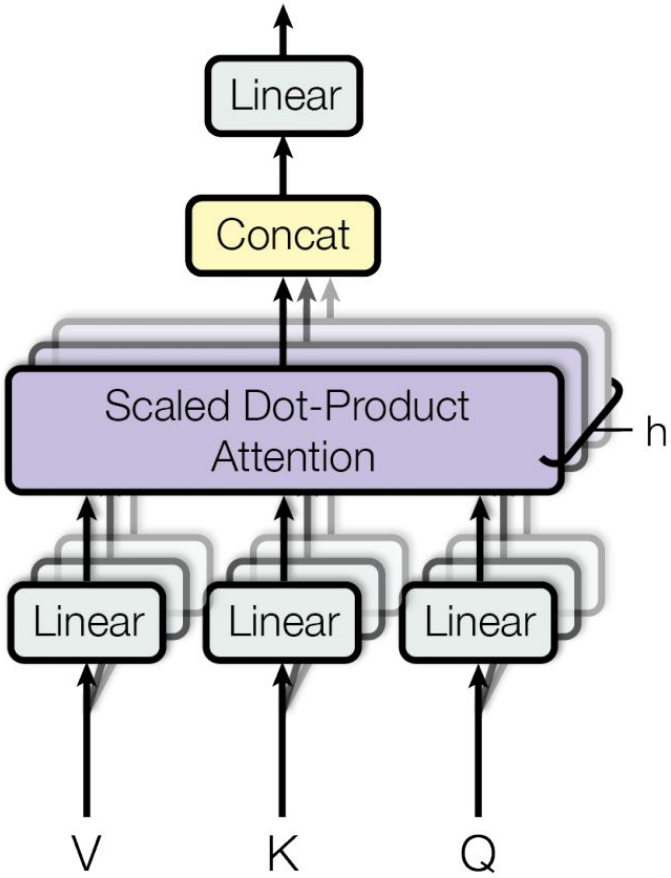
$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O$$
$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

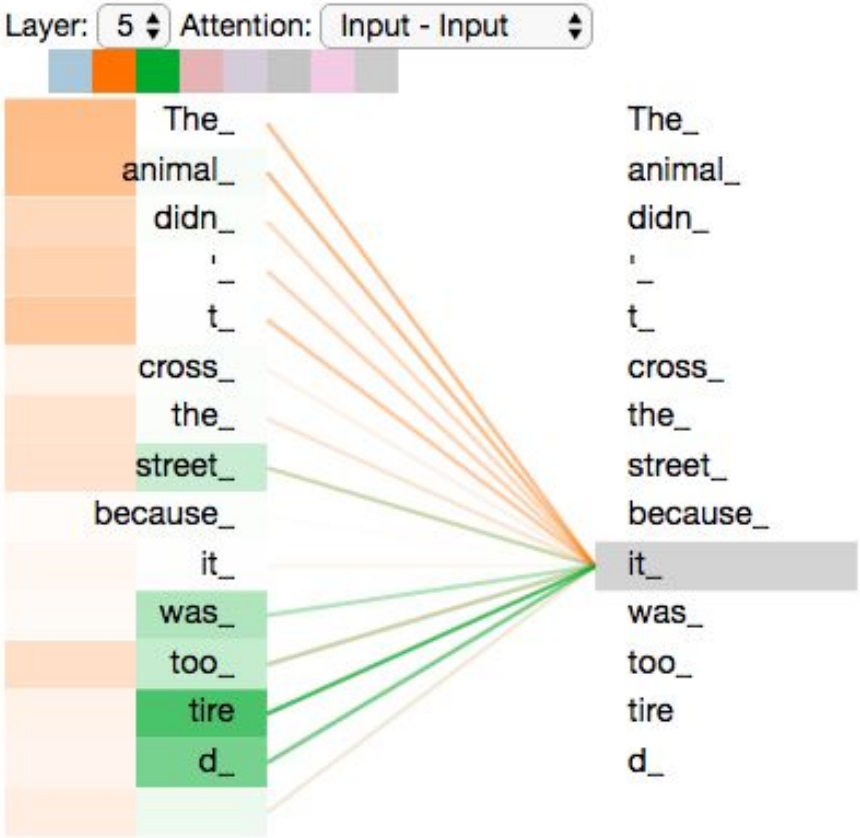
$$W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

In this work we employ h=8 parallel attention layers, or heads. For each of these we use $d_k=d_v=d_{model}/h=64$.

Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.



Multi-head self-attention example (2 heads shown)



http://jalammar.github.io/illustrated-transformer/

Different architectures for NMT

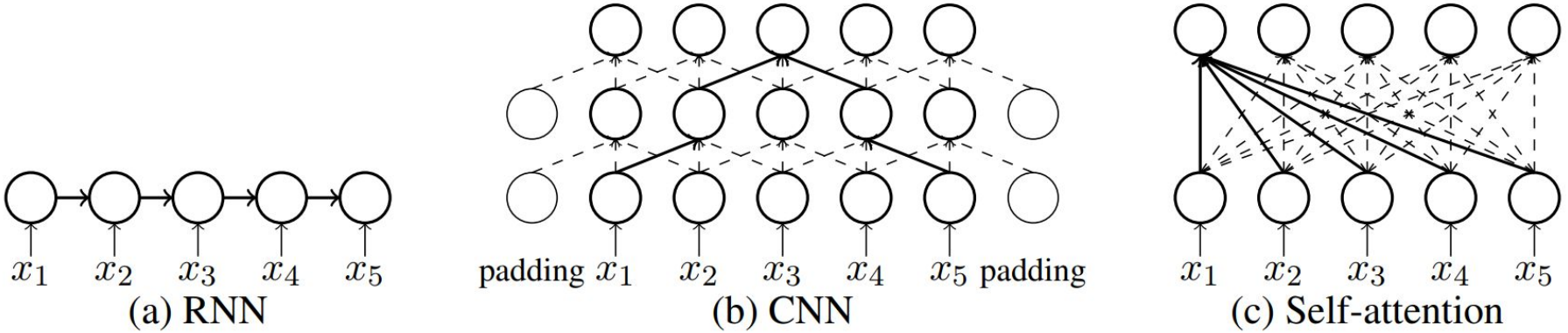


Figure 1: Architectures of different neural networks in NMT.

Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures,
https://arxiv.org/abs/1808.08946

Image Transformer

In this work, we generalize a recently proposed model architecture based on self-attention, the Transformer, to a sequence modeling formulation of image generation with a tractable likelihood. By restricting the self-attention mechanism to attend to local neighborhoods we significantly increase the size of images the model can process in practice, despite maintaining significantly larger receptive fields per layer than typical convolutional neural networks.

Image Transformer, https://arxiv.org/abs/1802.05751

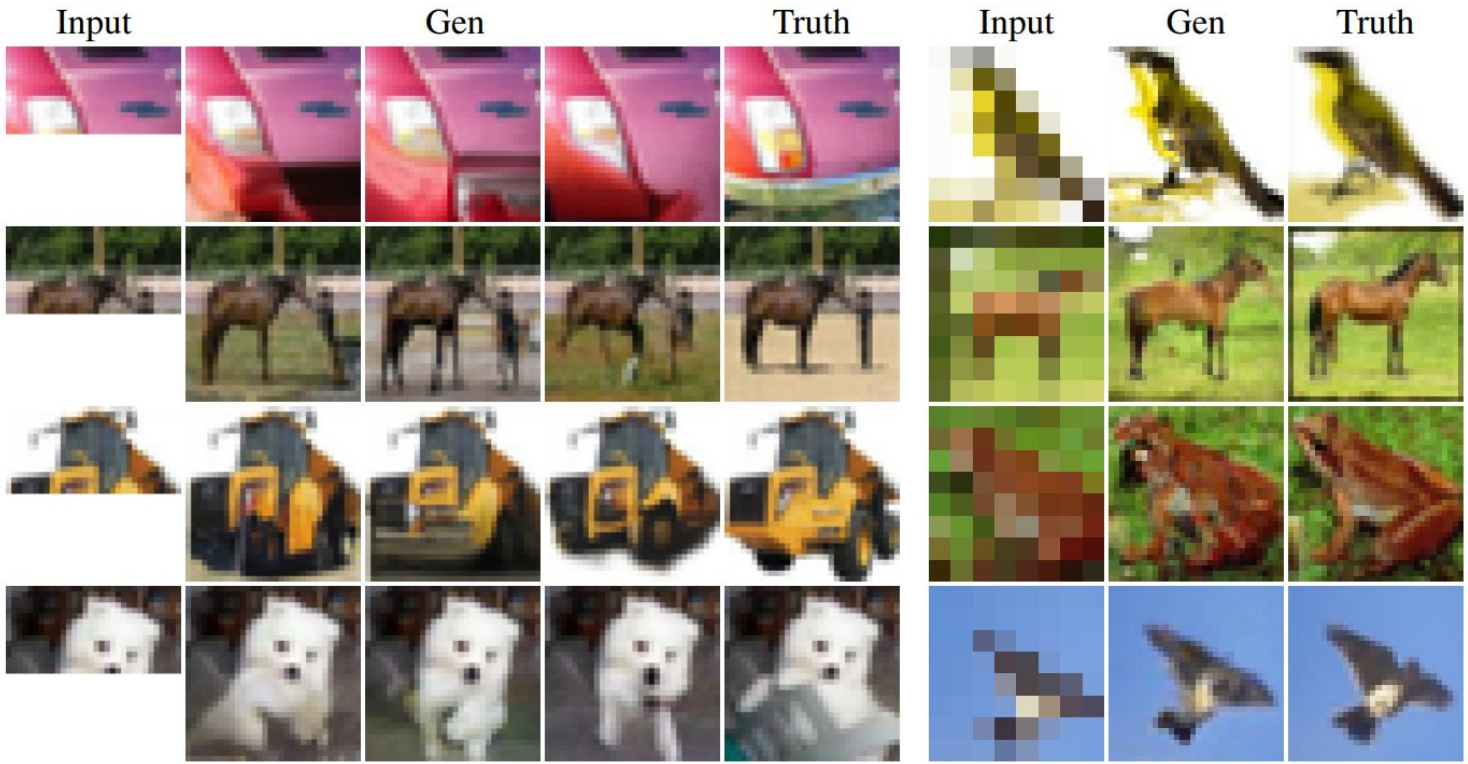


Table 2. On the left are image completions from our best conditional generation model, where we sample the second half. On the right are samples from our four-fold super-resolution model trained on CIFAR-10. Our images look realistic and plausible, show good diversity among the completion samples and observe the outputs carry surprising details for coarse inputs in super-resolution.

Image Transformer, https://arxiv.org/abs/1802.05751

BERT

Bidirectional Encoder Representations from Transformers, or BERT.

BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT uses only the encoder part of the Transformer.

Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing,
https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html
Best NLP Model Ever? Google BERT Sets New Standards in 11 Language Tasks
https://medium.com/syncedreview/best-nlp-model-ever-google-bert-sets-new-standards-in-11-language-tasks-4a2a189bc155

BERT

Bidirectional Encoder Representations from Transformers, or BERT

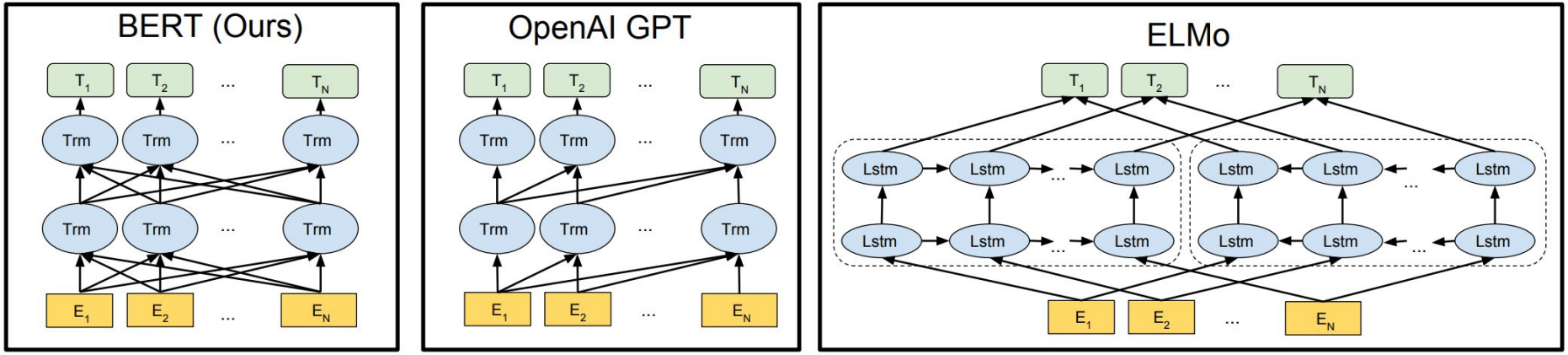


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,
https://arxiv.org/abs/1810.04805

BERT

Pre-training tasks:

- **Masked Language Model**: predict random words from within the sequence, not the next word for a sequence of words.
- **Next Sentence Prediction**: give the model two sentences and ask it to predict if the second sentence follows the first in a corpus or not.

Input =

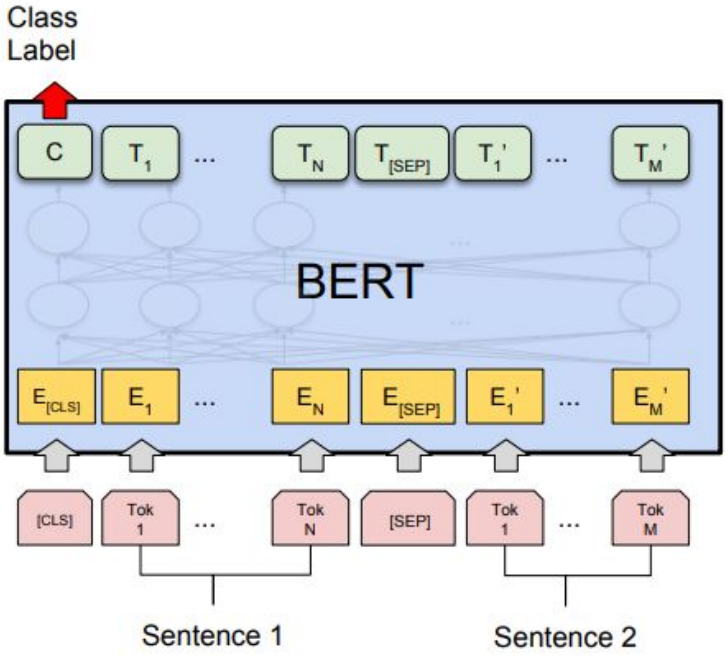
[CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

BERT

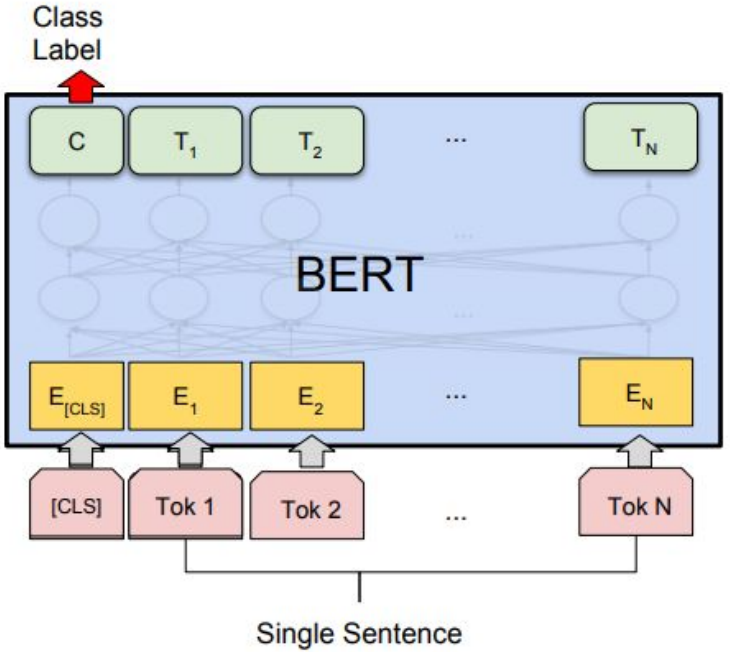
How to use:

- **Fine-tuning approach**: pre-train some model architecture on a LM objective before fine-tuning that same model for a supervised downstream task.
  - Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch.
- **Feature-based approach**: learned representations are typically used as features in a downstream model.
  - Not all NLP tasks can be easily be represented by a Transformer encoder architecture, and therefore require a task-specific model architecture to be added.
  - There are major computational benefits to being able to pre-compute an expensive representation of the training data once and then run many experiments with less expensive models on top of this representation

BERT: using fine-tuning approach

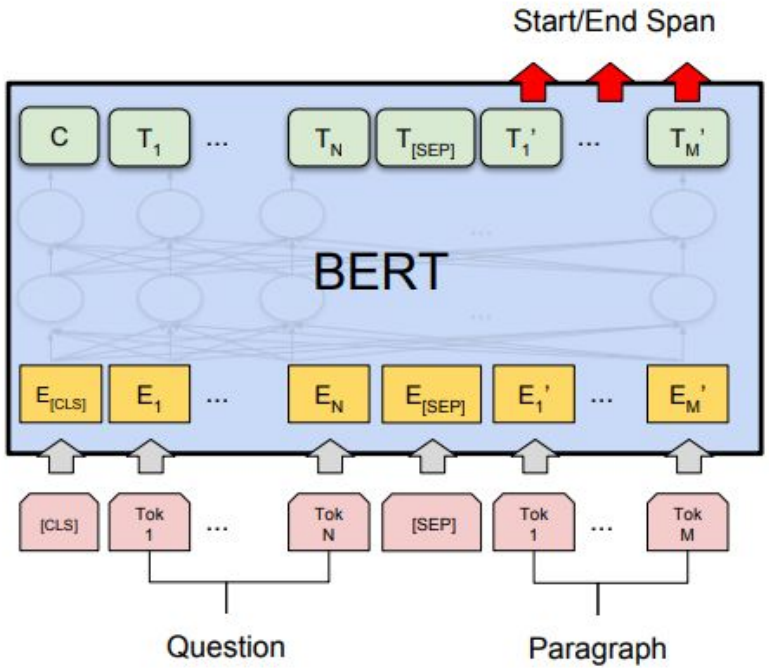


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

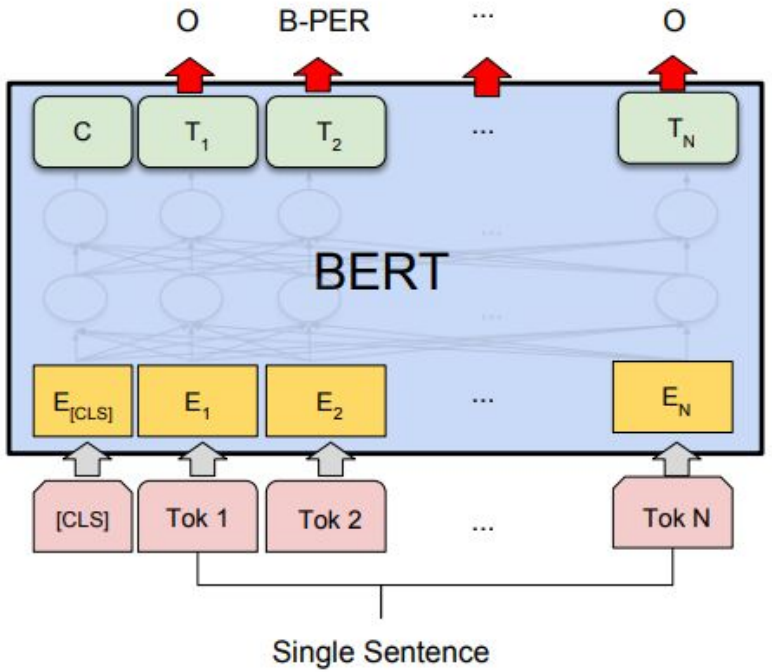


(b) Single Sentence Classification Tasks: SST-2, CoLA

BERT: using fine-tuning approach

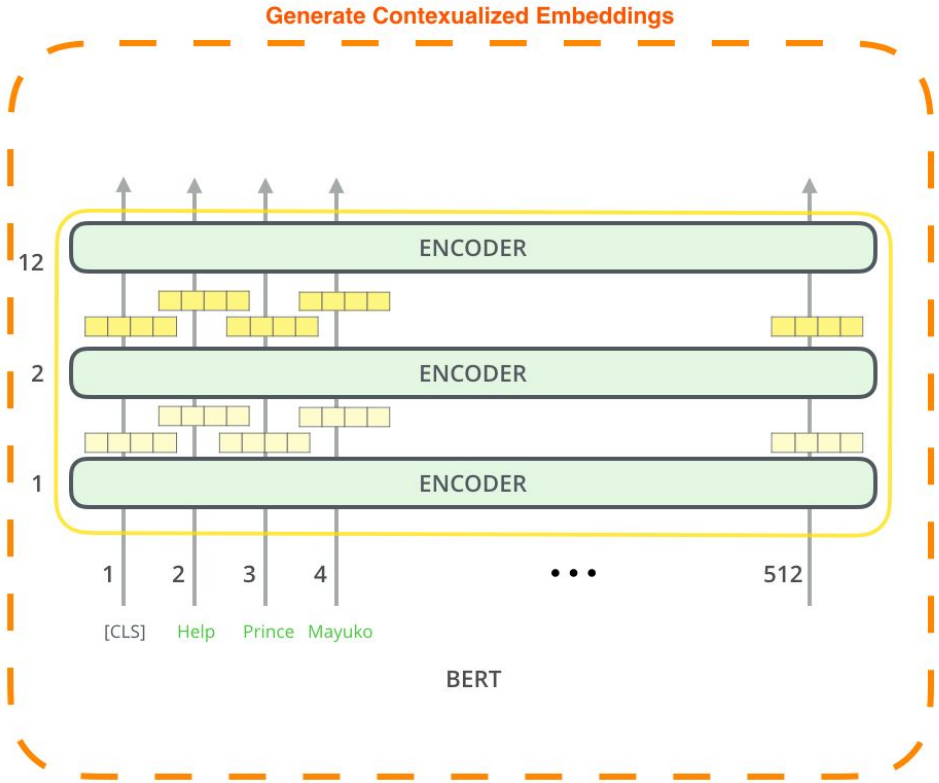


(c) Question Answering Tasks: SQuAD v1.1

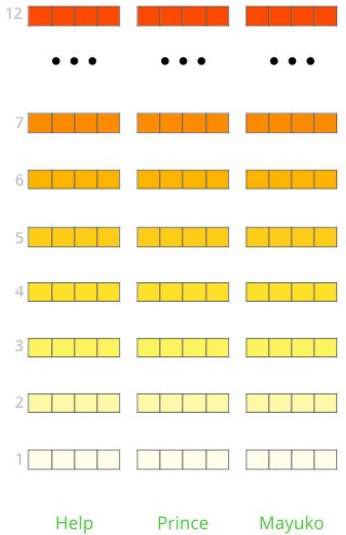


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT: feature extraction



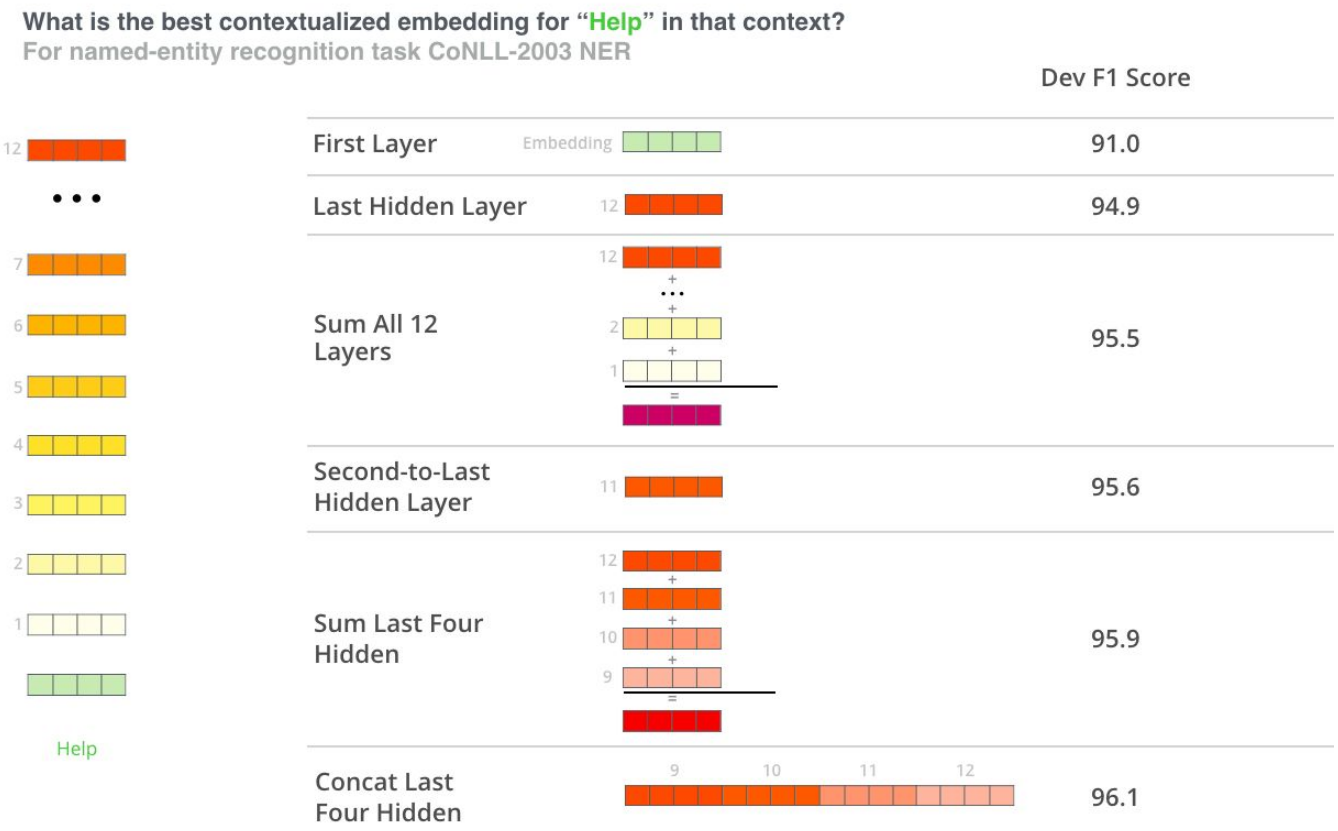
The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

https://jalammar.github.io/illustrated-bert/

BERT: feature extraction



https://jalammar.github.io/illustrated-bert/

Universal Transformer

In “Universal Transformers” we extend the standard Transformer to be computationally universal (**Turing complete**) using a novel, efficient flavor of parallel-in-time recurrence which yields stronger results across a wider range of tasks.

Crucially, where an RNN processes a sequence symbol-by-symbol (left to right), the Universal Transformer processes all symbols at the same time (like the Transformer), but then refines its interpretation of every symbol in parallel over a variable number of recurrent processing steps using self-attention.

This parallel-in-time recurrence mechanism is both faster than the serial recurrence used in RNNs, and also makes the Universal Transformer more powerful than the standard feedforward Transformer.

Moving Beyond Translation with the Universal Transformer
https://ai.googleblog.com/2018/08/moving-beyond-translation-with.html

Universal Transformer

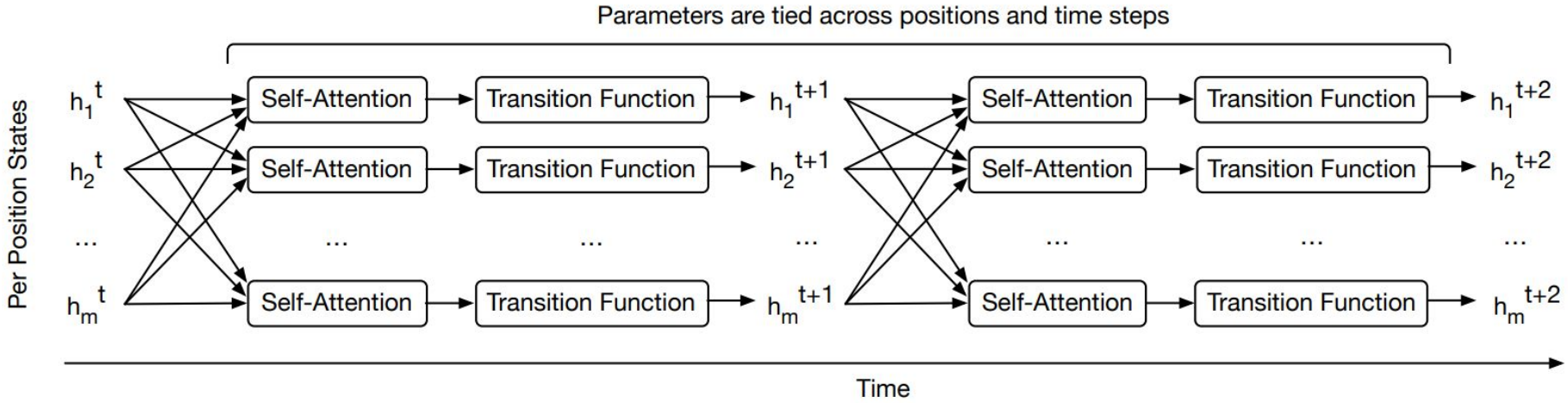


Figure 1: The Universal Transformer repeatedly refines a series of vector representations for each position of the sequence in parallel, by combining information from different positions using self-attention and applying a recurrent transition function. We show this process over two recurrent time-steps. Arrows denote dependencies between operations. Initially, h^0 is initialized with the embedding for each symbol in the sequence. h_i^t represents the representation for input symbol $1 \leq i \leq m$ at recurrent time-step t.

Universal Transformers, https://arxiv.org/abs/1807.03819

Universal Transformer

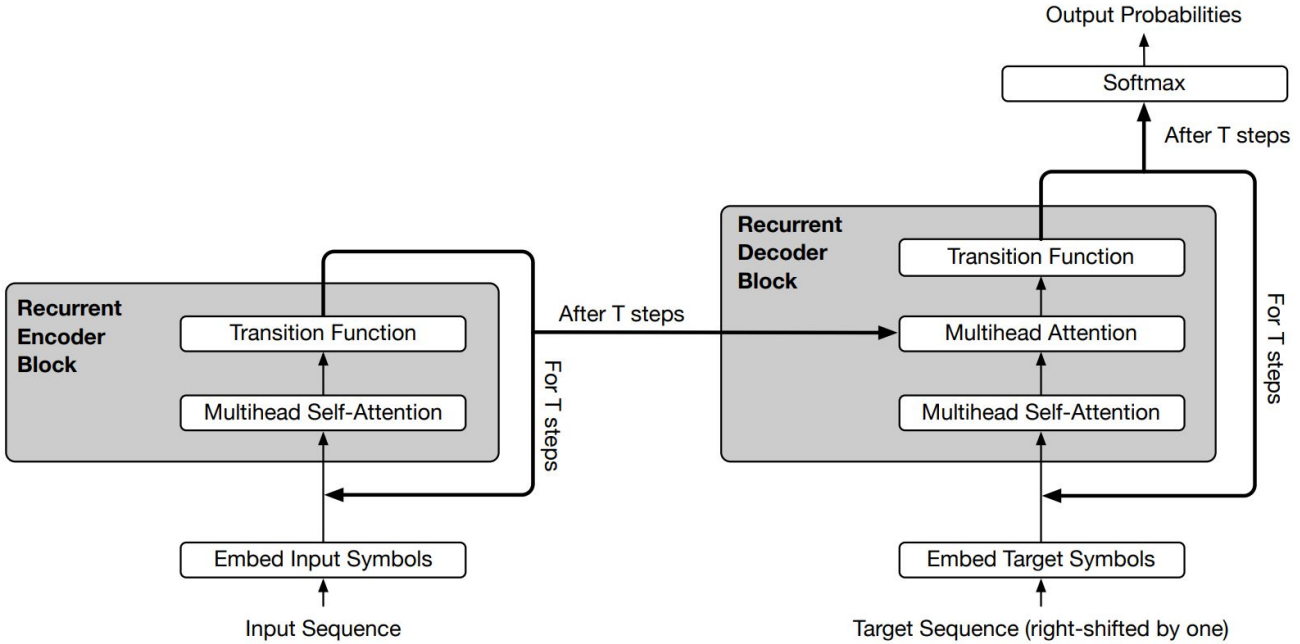


Figure 2: The recurrent blocks of the Universal Transformer encoder and decoder. This diagram omits position and time-step encodings as well as dropout, residual connections and layer normalization. A complete version can be found in the appendix. The Adaptive Universal Transformer dynamically determines the number of steps T for each position using ACT.

Universal Transformers, https://arxiv.org/abs/1807.03819

Summary

DL/Multi-modal Learning

Deep Learning models become multi-modal: they use 2+ modalities simultaneously, i.e.:

- Image caption generation: images + text
- Search Web by an image: images + text
- Video describing: the same but added time dimension
- Visual question answering: images + text
- Speech recognition: audio + video (lips motion)
- Image classification and navigation: RGB-D (color + depth)

Where does it aim to?

- Common metric space for each concept, “thought vector”. Will be possible to match different modalities easily.

DL/Transfer of Ideas

Methods developed for one modality are successfully transferred to another:

- Convolutional Neural Networks, CNNs (originally developed for image recognition) work well on texts, speech and some time-series signals (e.g. ECG).
- Recurrent Neural Networks, RNNs (mostly used on language and other sequential data) seem to work on images.

If the technologies successfully transfer from one modality to another (for example, image to texts for CNNs) then probably the ideas worked in one domain will work in another (style transfer for images could be transferred to texts).

Why Deep Learning is helpful? Or even a game-changer

- Works on raw data (pixels, sound, text or chars), no need to feature engineering
  - Some features are really hard to develop (requires years of work for group of experts)
  - Some features are patented (i.e. SIFT, SURF for images)
- Allows end-to-end learning (pixels-to-category, sound to sentence, English sentence to Chinese sentence, etc)
  - No need to do segmentation, etc. (a lot of manual labor)

⇒ **You can iterate faster (and get superior quality at the same time!)**

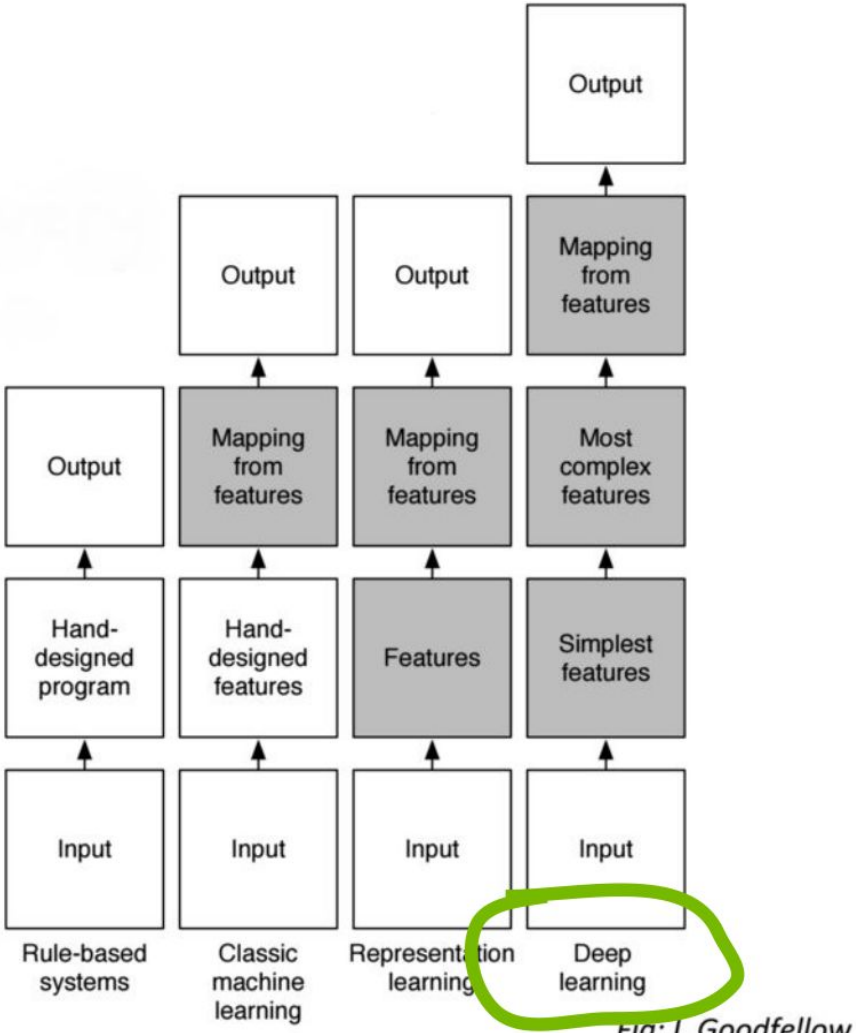


Fig: I. Goodfellow

Still some issues exist: Datasets

- No dataset -- no deep learning

  There are a lot of data available (and it’s required for deep learning, otherwise simple models could be better)

  - But sometimes you have no dataset…
    - Nonetheless some hacks available: Transfer learning, Data augmentation, Mechanical Turk, …

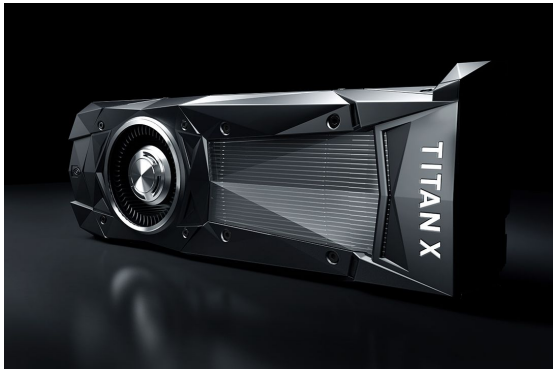
Still some issues exist: Datasets

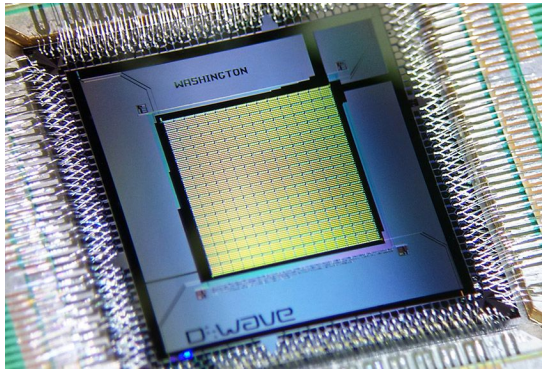
Year	Breakthroughs in AI	Datasets (First Available)	Algorithms (First Proposed)
1994	Human-level spontaneous speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka "The Extended Book" (1991)	Negascout planning algorithm (1983)
2005	Google's Arabic- and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! champion	8.6 million documents from Wikipedia, Wiktionary, Wikiquote, and Project Gutenberg (updated in 2010)	Mixture-of-Experts algorithm (1991)
2014	Google's GoogLeNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolution neural network algorithm (1989)
2015	Google's Deepmind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning algorithm (1992)
Average No. of Years to Breakthrough:		**3 years**	**18 years**

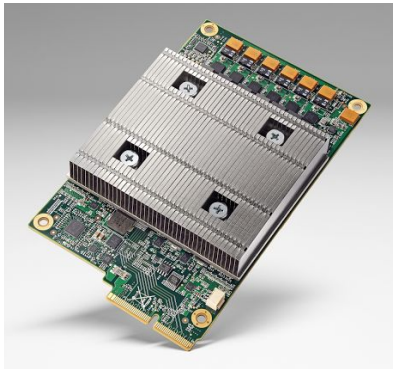
http://www.spacemachine.net/views/2016/3/datasets-over-algorithms

Still some issues exist: Computing power

- Requires a lot of computations. No cluster or GPU machines -- much more time required
  - Currently GPUs (mostly NVIDIA) is the only choice
  - Waiting FPGA/ASIC coming into this field (Google TPU gen.2, Intel 2017+). The situation resembles the path of Bitcoin mining
  - Neuromorphic computing is on the rise (IBM TrueNorth, memristors, etc)
  - Quantum computing can benefit machine learning as well (but probably it won't be a desktop or in-house server solutions)



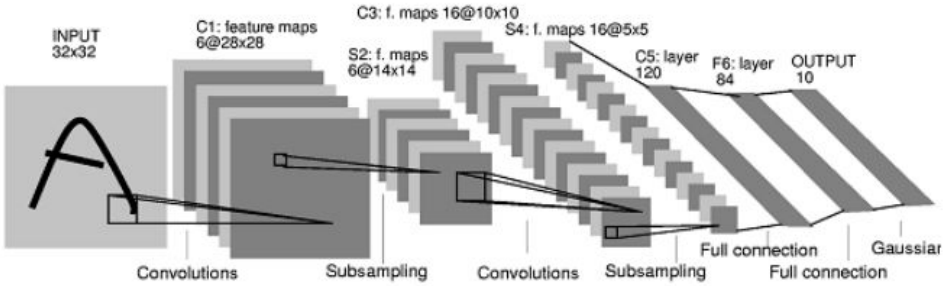




Datasets and computing power are growing

1998

LeCun et al.



# of transistors

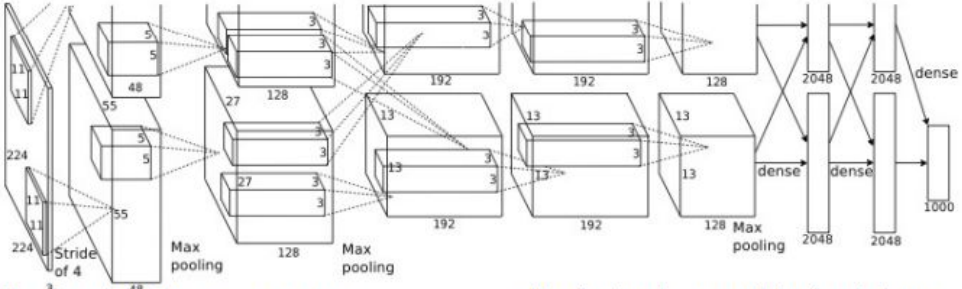
10^6

# of pixels used in training

10^7 NIST

2012

Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

Still some issues exist: Reasoning

Deep learning is mainly about perception, but there is a lot of inference involved in everyday human reasoning.

- Neural networks lack common sense
- Cannot find information by inference
- Cannot explain the answer
  - It could be a must-have requirement in some areas, i.e. law, medicine.

Still some issues exist: Reasoning

The most fruitful approach is likely to be a hybrid neural-symbolic system. Topic of active research right now.

And it seems all major players are already go this way (Watson, Siri, Cyc, …)

There is a lot of knowledge available (or extractable) in the world. Large knowledge bases about the real world (Cyc/OpenCyc, FreeBase, Wikipedia, schema.org, RDF, ..., scientific journals + text mining, …)

Фреймворки и библиотеки для работы с нейросетями

Libraries and Frameworks

A larger list: http://deeplearning.net/software_links/

Deep Learning в Spark

- **MLLib** (MultilayerPerceptronClassifier) [Python, Java, Scala]
- **BigDL** (https://github.com/intel-analytics/BigDL) [Python, Java, Scala]
- **CaffeOnSpark** (https://github.com/yahoo/CaffeOnSpark) [Scala, Java]
- **TensorFlowOnSpark** (https://github.com/yahoo/TensorFlowOnSpark) [Python]
- **Deeplearning4j** (https://deeplearning4j.org/) [Scala, Java]
- **Elephas**: Distributed Deep Learning with Keras & Spark (https://github.com/maxpumperla/elephas) [Python]
- **DeepDist** (https://github.com/dirkneumann/deepdist/) [Python, Java, Scala]
- SparkNet (https://github.com/amplab/SparkNet) [Scala, Java]
- + local installations of DL libraries (but it's better to use DL frameworks)

Универсальные библиотеки и сервисы

- ~~**Torch7**~~**, PyTorch** (http://torch.ch/, http://pytorch.org) [Lua, Python]
- **TensorFlow** (https://www.tensorflow.org/) [Python, C++]
- **Keras** (http://keras.io/) [Python]
- ~~**Theano** (http://deeplearning.net/software/theano/) [Python]~~
  - ~~**Lasagne** (https://github.com/Lasagne/Lasagne)~~
  - ~~**blocks** (https://github.com/mila-udem/blocks)~~
  - ~~**pylearn2** (https://github.com/lisa-lab/pylearn2)~~
- **Microsoft Cognitive Toolkit (CNTK)** (http://www.cntk.ai/) [Python, C++, C#, BrainScript]
- **Neon** (http://neon.nervanasys.com/) [Python]
- **Deeplearning4j** (http://deeplearning4j.org/) [Java]
- **MXNet** (http://mxnet.io/) [C++, Python, R, Scala, Julia, Matlab, Javascript]
- …

Обработка изображений и видео

- **OpenCV** (http://opencv.org/) [C, C++, Python]
- **Caffe2** (https://caffe2.ai/)
- **Caffe** (http://caffe.berkeleyvision.org/) [C++, Python, Matlab]
- **clarifai** (https://www.clarifai.com/)
- **Google Vision API** (https://cloud.google.com/vision/)
- …
- + all universal libraries

Распознавание речи

- **Microsoft Cognitive Toolkit (CNTK)** (http://www.cntk.ai/) [Python, C++, C#, BrainScript]
- **KALDI** (http://kaldi-asr.org/) [C++]
- **Google Speech API** (https://cloud.google.com/)
- **Yandex SpeechKit** (https://tech.yandex.ru/speechkit/)
- **Baidu Speech API** (http://www.baidu.com/)
- **wit.ai** (https://wit.ai/)
- …

Обработка текстов

- **Torch7** (http://torch.ch/)
- **Theano/Keras/**…
- **TensorFlow** (https://www.tensorflow.org/)
- **Google Translate API** (https://cloud.google.com/translate/)
- **Salesforce Einstein** (https://www.salesforce.com/products/einstein/overview/)
-
- **Machine Translation Benchmark (https://bit.ly/mt_jul2018)**
- **Intent Detection Benchmark (August 2017)** (https://www.slideshare.net/KonstantinSavenkov/nlu-intent-detection-benchmark-by-intento-august-2017)

Что читать и смотреть

- **CS231n: Convolutional Neural Networks for Visual Recognition**, Fei-Fei Li, Andrej Karpathy, Stanford (http://vision.stanford.edu/teaching/cs231n/index.html)
- **CS224d: Deep Learning for Natural Language Processing**, Richard Socher, Stanford (http://cs224d.stanford.edu/index.html)
- **Neural Networks for Machine Learning**, Geoffrey Hinton (https://www.coursera.org/course/neuralnets)
- **Подборка курсов по компьютерному зрению** (http://eclass.cc/courselists/111_computer_vision_and_navigation)
- **Подборка курсов по deep learning (**http://eclass.cc/courselists/117_deep_learning)
- “**Deep Learning**”, Ian Goodfellow, Yoshua Bengio and Aaron Courville (http://www.deeplearningbook.org/)

Что читать и смотреть

- **Google+ Deep Learning community** (https://plus.google.com/communities/112866381580457264725)
- **VK Deep Learning community** (http://vk.com/deeplearning)
- **Quora** (https://www.quora.com/topic/Deep-Learning)
- **FB Deep Learning Moscow** (https://www.facebook.com/groups/1505369016451458/)
- **Twitter Deep Learning Hub** (https://twitter.com/DeepLearningHub)
- **NVidia blog** (https://devblogs.nvidia.com/parallelforall/tag/deep-learning/)
- **IEEE Spectrum blog** (http://spectrum.ieee.org/blog/cars-that-think)
- http://deeplearning.net/
- **Arxiv Sanity Preserver** http://www.arxiv-sanity.com/
- ...

За кем следить?

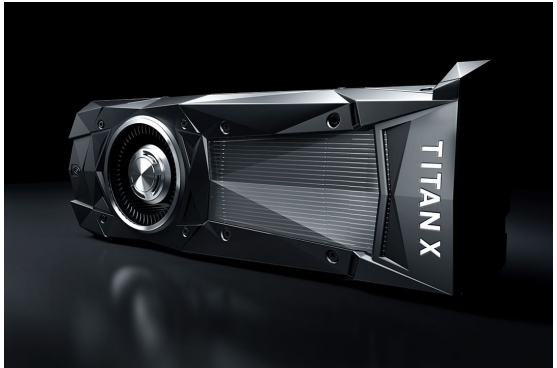
- **Jürgen Schmidhuber** (http://people.idsia.ch/~juergen/)
- **Geoffrey E. Hinton** (http://www.cs.toronto.edu/~hinton/)
- **Yann LeCun** (http://yann.lecun.com, https://www.facebook.com/yann.lecun)
- **Yoshua Bengio** (http://www.iro.umontreal.ca/~bengioy, https://www.quora.com/profile/Yoshua-Bengio)
- **Andrej Karpathy** (http://karpathy.github.io/)
- **Andrew Ng** (http://www.andrewng.org/)
- **Google DeepMind** (http://deepmind.com/)
- **OpenAI** (https://blog.openai.com/)
- ...

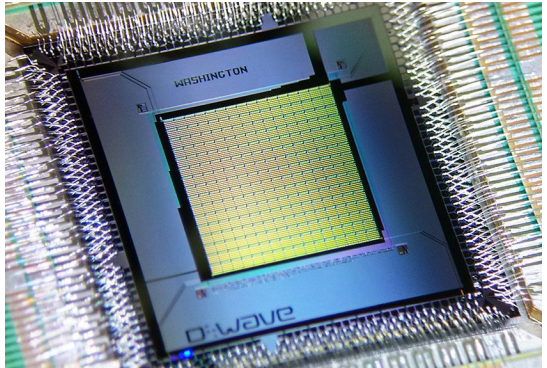
Hardware

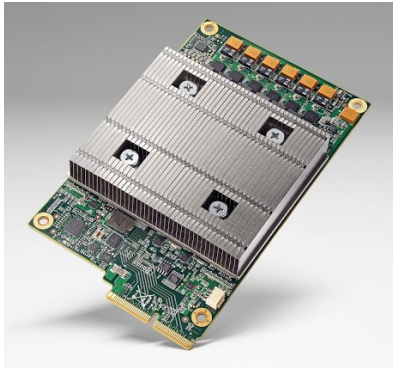
Computing power is a strong requirement

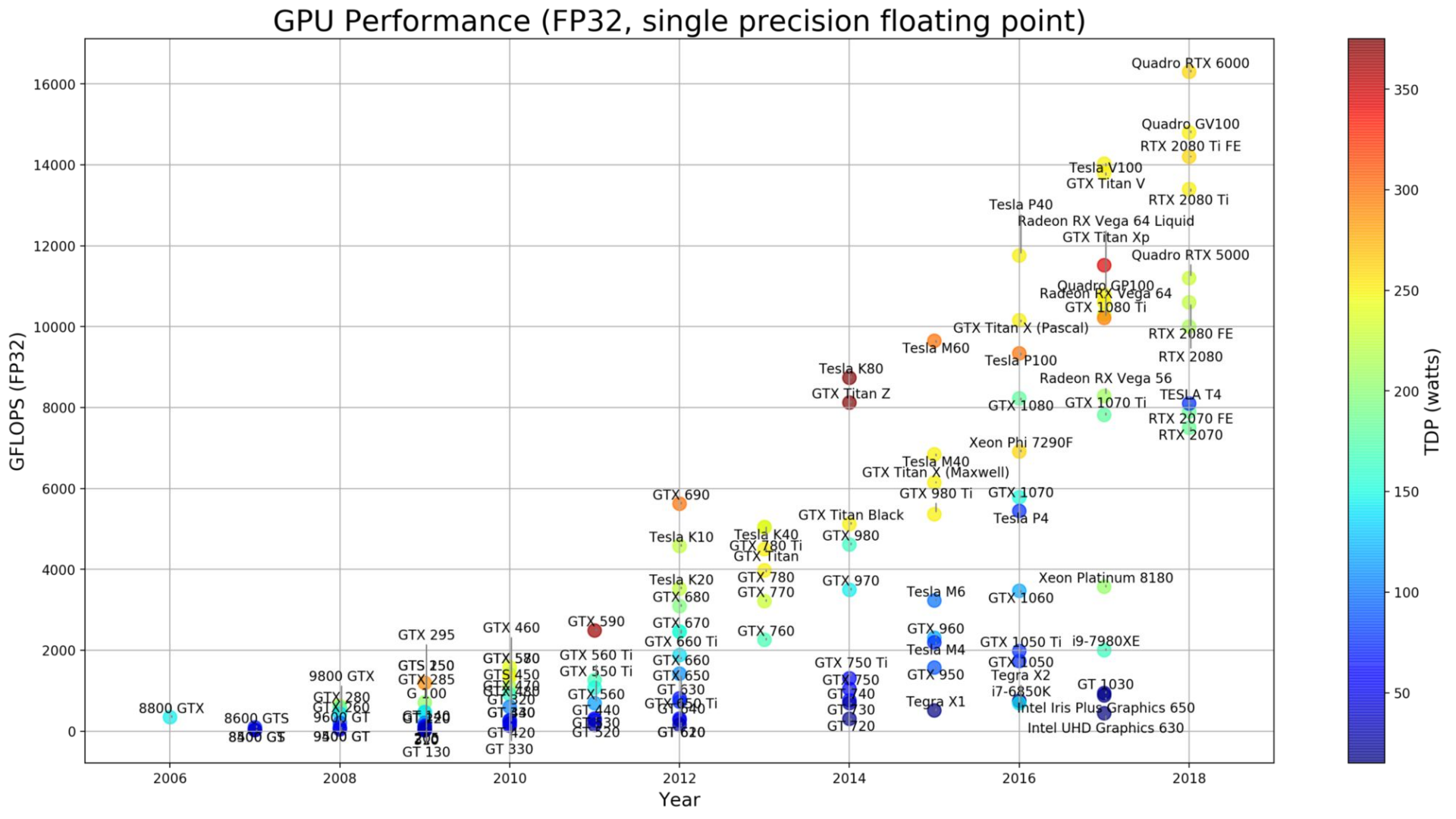
DL requires a lot of computations. Without a cluster or GPU machines much more time is required.

- Currently GPUs (mostly NVIDIA) is the only choice
- FPGA/ASIC are coming into this field (Google TPU gen.3, Bitmain Sophon, Intel 2018+). The situation resembles the path of Bitcoin mining
- Neuromorphic computing is on the rise (IBM TrueNorth, Intel, memristors, etc)
- Quantum computing can benefit machine learning as well (but probably it won't be a desktop or in-house server solutions)

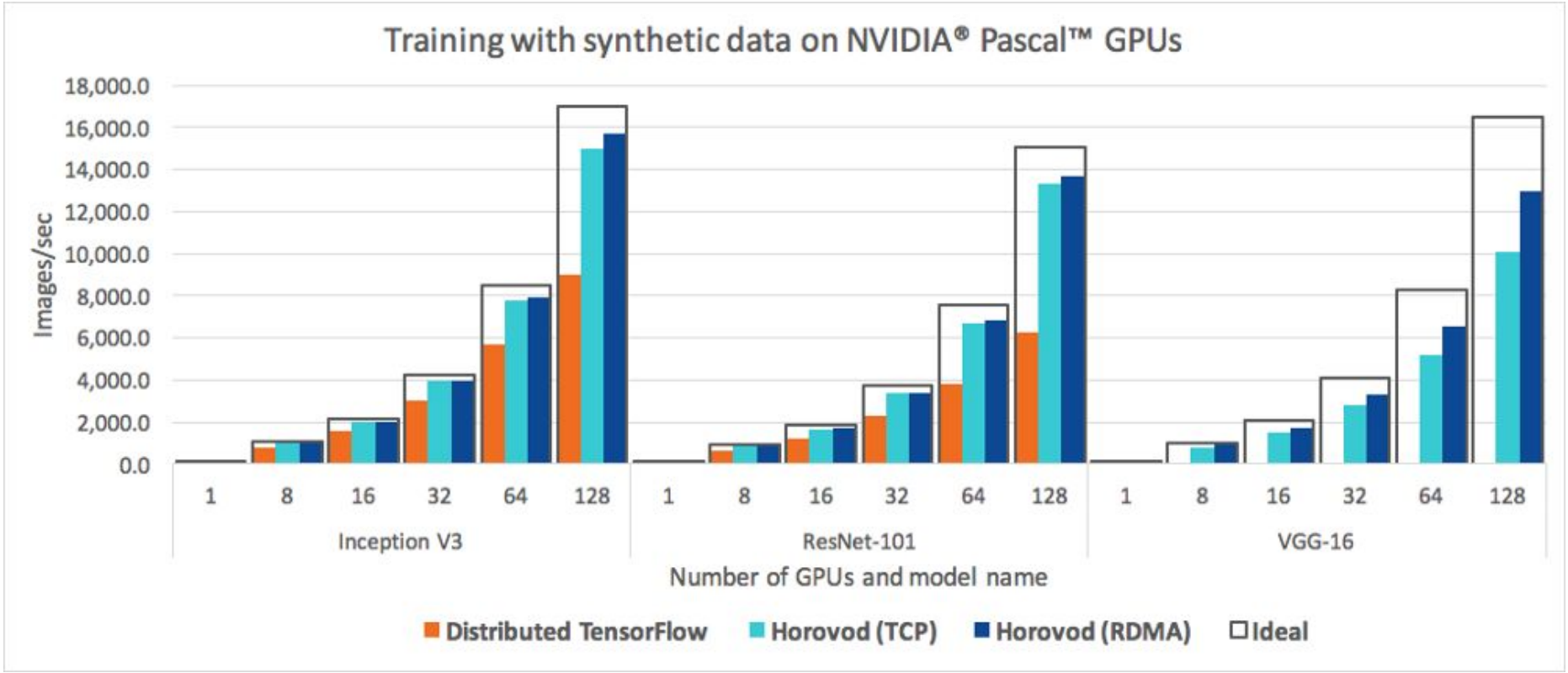







GPU Performance (FP32, single precision floating point)
GFLOPS (FP32)
Year
TDP (watts)
Quadro RTX 6000
Quadro GV100
RTX 2080 Ti FE
Tesla V100
GTX Titan V
RTX 2080 Ti
Tesla P40
Radeon RX Vega 64 Liquid
GTX Titan Xp
Quadro RTX 5000
Quadro GP100
Radeon RX Vega 64
GTX 1080 Ti
GTX Titan X (Pascal)
RTX 2080 FE
Tesla M60
Tesla P100
RTX 2080
Tesla K80
Radeon RX Vega 56
GTX Titan Z
TESLA T4
GTX 1080
GTX 1070 Ti
RTX 2070 FE
RTX 2070
Xeon Phi 7290F
Tesla M40
GTX Titan X (Maxwell)
GTX 980 Ti
GTX 1070
GTX 690
GTX Titan Black
Tesla P4
Tesla K10
Tesla K40
GTX 980
GTX 780 Ti
GTX Titan
Xeon Platinum 8180
Tesla K20
GTX 780
GTX 970
GTX 680
GTX 770
Tesla M6
GTX 1060
GTX 590
GTX 460
GTX 670
GTX 760
GTX 960
GTX 295
GTX 660 Ti
Tesla M4
GTX 1050 Ti
i9-7980XE
GTX 560 Ti
GTX 660
GTX 750 Ti
GTX 1050
9800 GTX
GTX 550 Ti
GTX 650
GTX 950
Tesla X2
GTX 285
GTX 280
GTX 560
GTX 630
GTX 750
i7-6850K
GT 1030
GTX 260
9600 GT
GT 440
GT 740
Tesla X1
Intel Iris Plus Graphics 650
8800 GTX
GT 430
GT 730
Intel UHD Graphics 630
8600 GTS
GT 520
GT 720
GT 420
GT 330
GT 130
2006
2008
2010
2012
2014
2016
2018
0
2000
4000
6000
8000
10000
12000
14000
16000
50
100
150
200
250
300
350

Distributed training is a commodity now



https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664

GPUS MAKE DEEP LEARNING ACCESSIBLE

Deep learning with COTS HPC systems

A. Coates, B. Huval, T. Wang, D. Wu, A. Ng, B. Catanzaro

ICML 2013

"*Now You Can Build Google's $1M Artificial Brain on the Cheap*"

WIRED

GOOGLE DATACENTER

1,000 CPU Servers
2,000 CPUs • 16,000 cores

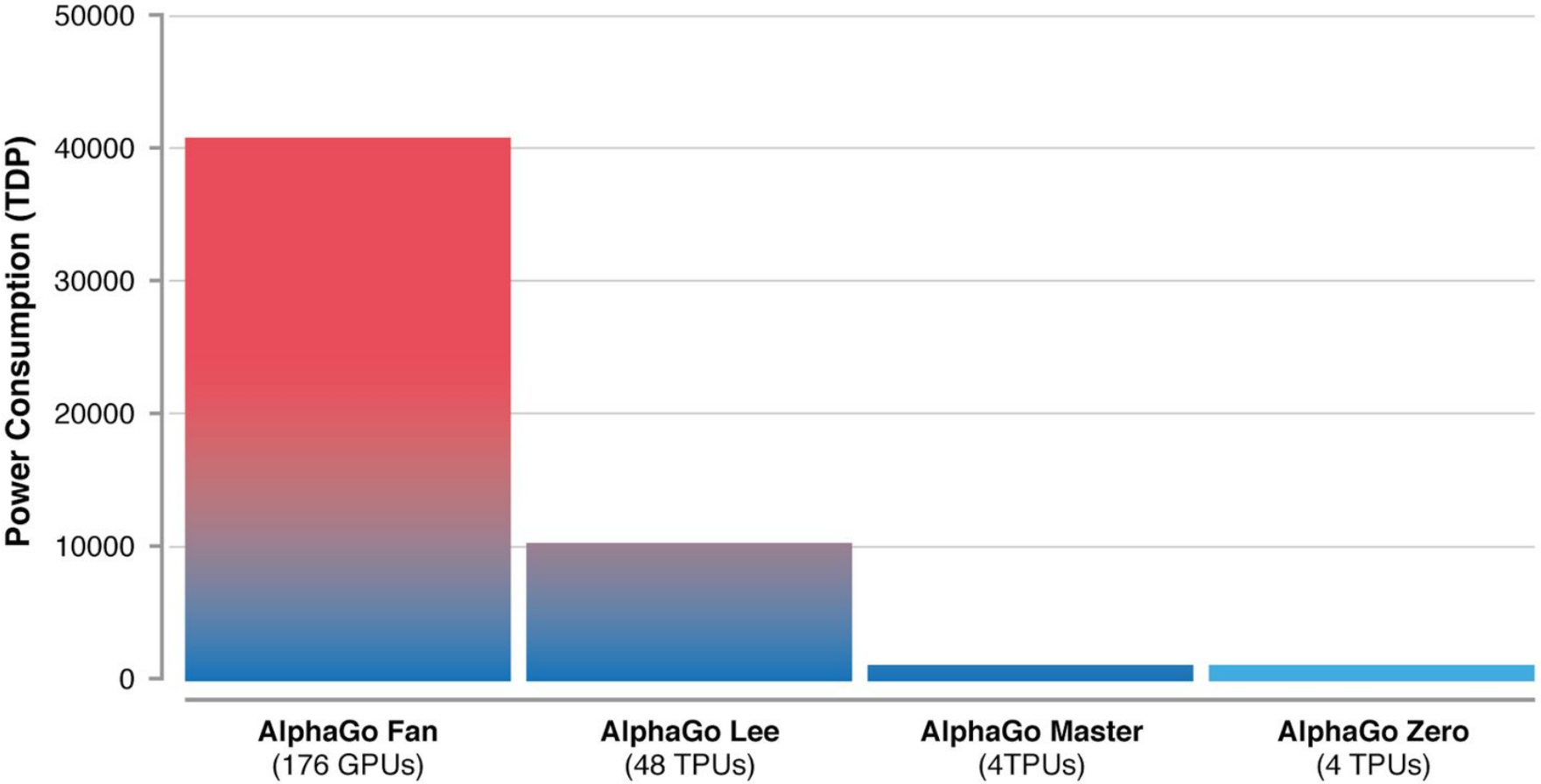
600 kWatts
$5,000,000

STANFORD AI LAB

3 GPU-Accelerated Servers
12 GPUs • 18,432 cores

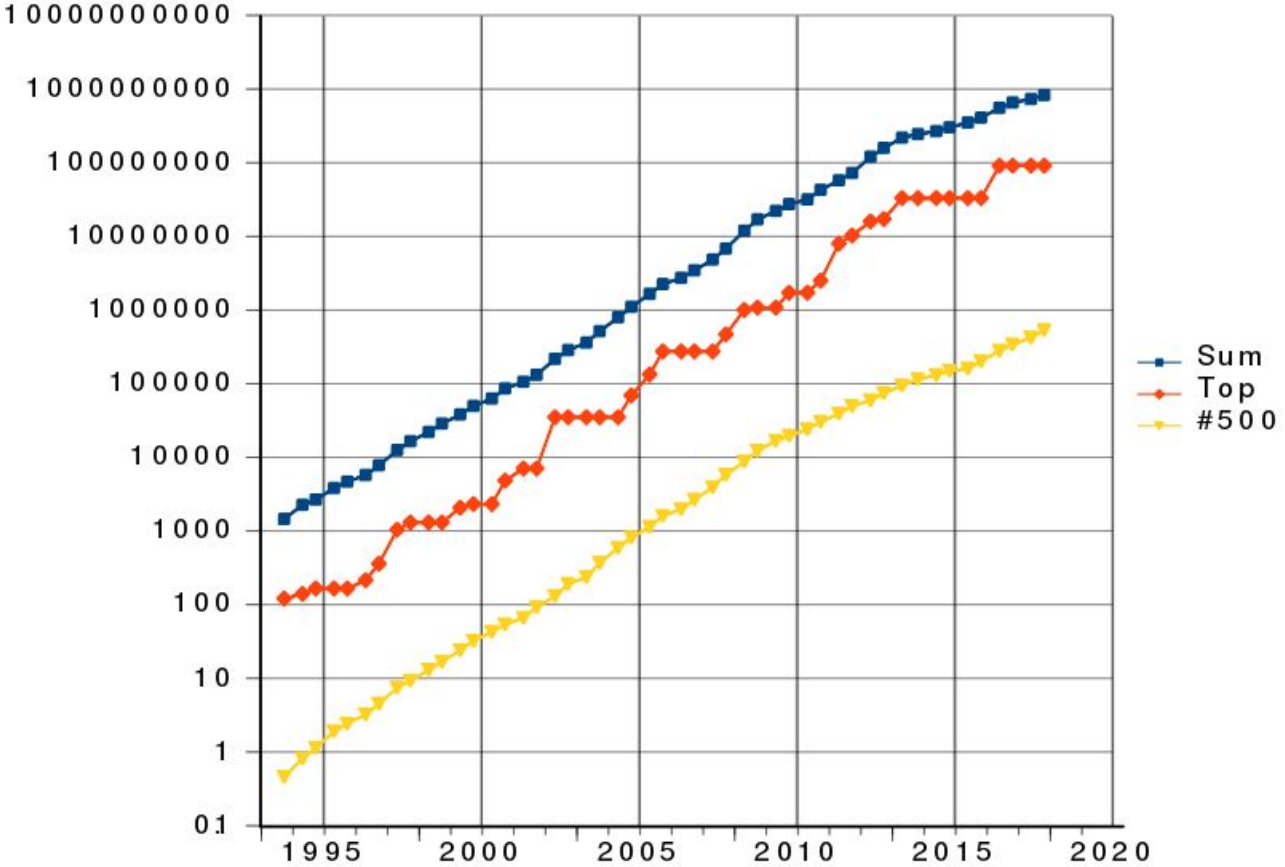
4 kWatts
$33,000

Case: AlphaGo Zero



https://deepmind.com/blog/alphago-zero-learning-scratch/

Trends: Supercomputer performance (GFLOPS)



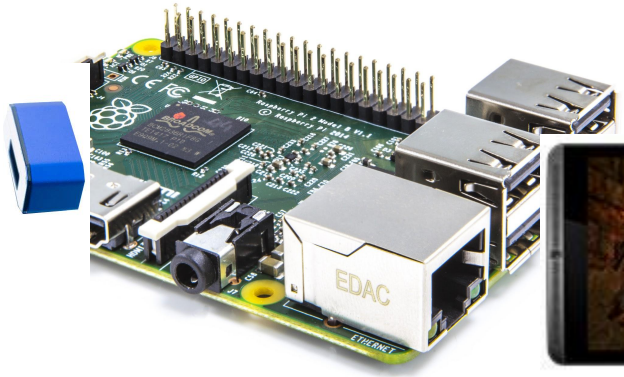
Personal Supercomputers

- **NVIDIA DGX-1 Server** ($149,000)
  Performance: 1000 TFLOPS FP16, 125 TFLOPS FP32
- **DeepLearning11** ($16,500, contains 10x NVIDIA GeForce GTX 1080 Ti)
  Performance: 100 TFLOPS FP32
- **NVIDIA GTX Titan V** gaming card ($3000) 6.9 TFLOPS FP64
  - Corresponds to the best supercomputer in the world at 2001–2002 (IBM ASCI White with 7.226 TFLOPS peak speed) and a supercomputer on 500th place (still a cool supercomputer) of the TOP500 list in November 2007 (the entry level to the list was the 5.9 TFlop/s)
- For comparison: **Huawei Mate 10 smartphone** with Kirin 970 Neural Network Processing Unit, 1.92 TFLOPS FP16
  - A similar performance (but FP64) had the top performing supercomputer of 1997

https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664

Deep Learning at the edge

- NVidia Jetson TK1/TX1/TX2/Xavier
  - 192/256/256/512 CUDA Cores
  - 4/4/6/8-Core ARM CPU, 2/4/8/16 Gb Mem
- Tablets, Smartphones
  - Qualcomm Snapdragon 845, Apple A11 Bionic, Huawei Kirin 970
- Raspberry Pi 3 (1.2 GHz 4-core)
- Movidius Neural Compute Stick
- Google Edge TPU



References:

Hardware for Deep Learning series of posts:
https://blog.inten.to/hardware-for-deep-learning-current-state-and-trends-51c01ebbb6dc

- Part 1: Introduction and Executive summary
- Part 2: CPU
- Part 3: GPU
- Part 4: FPGA
- Part 5: ASIC
- Part 6: Mobile AI
- Part 7: Neuromorphic computing
- Part 8: Quantum computing

Thanks!

https://ru.linkedin.com/in/grigorysapunov
gs@inten.to