CIML Summer Institute 2022
Scalable Machine Learning

SDSC SAN DIEGO SUPERCOMPUTER CENTER

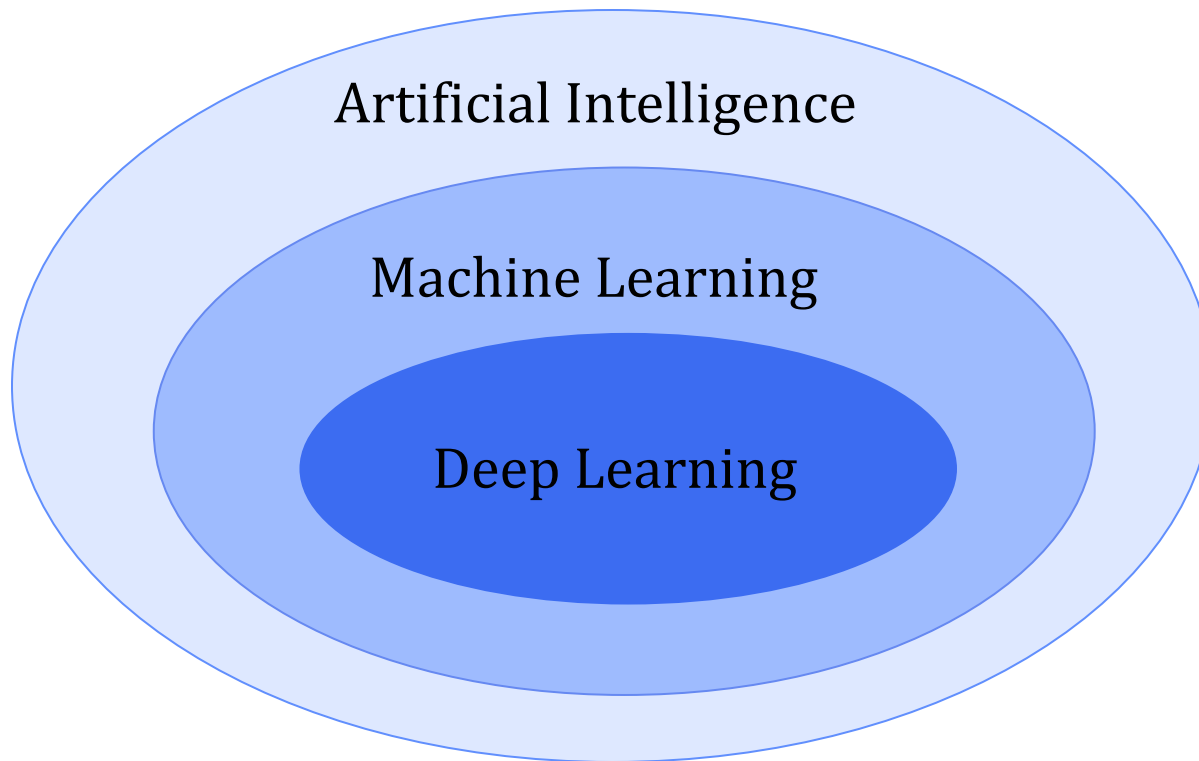UC San Diego

Deep Learning Layers & Architectures

Mai H. Nguyen, Ph.D.

DEEP LEARNING OVERVIEW

- **Neural Network Basics**
- **Deep Network Layers**
- **Deep Learning Architectures**
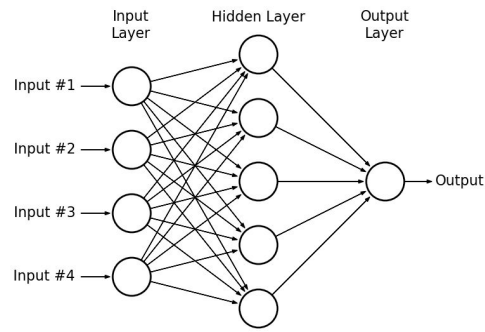- **Deep Learning Libraries**

DEEP LEARNING
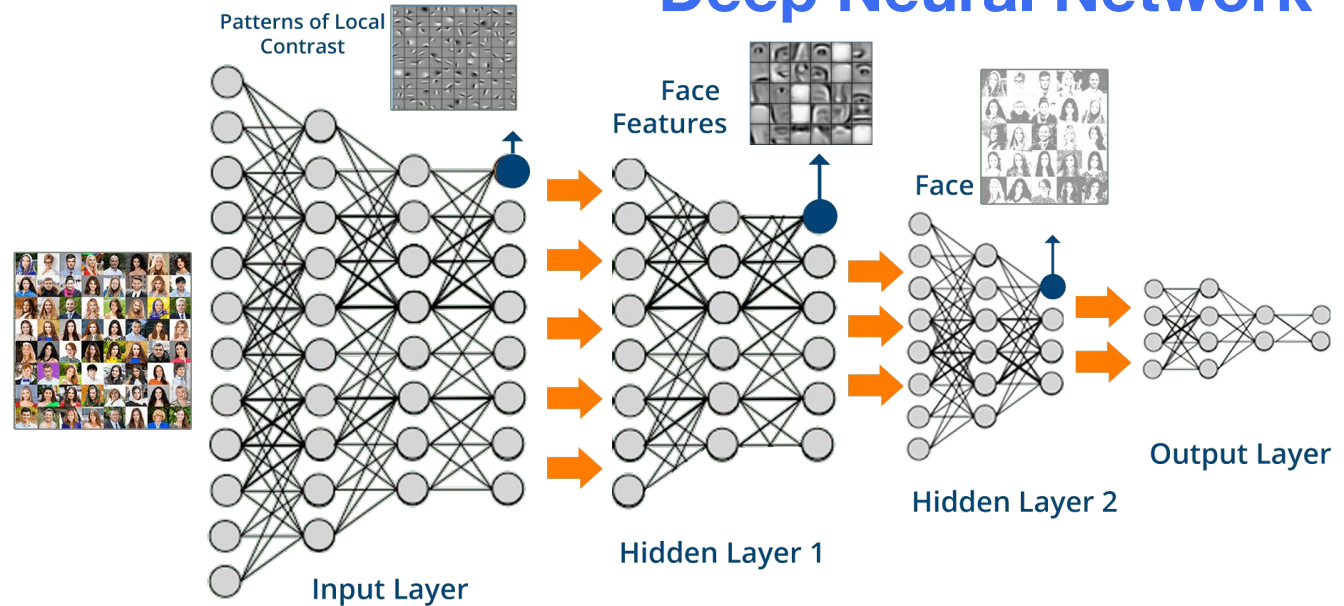
Deep Learning is a subfield of Machine Learning

DEEP LEARNING

Neural Network

http://www.astroml.org/book_figures/appendix/fig_neural_network.html

Deep Neural Network

Patterns of Local Contrast

Face Features

Face

Input Layer

Hidden Layer 1

Hidden Layer 2

Output Layer

https://cdn.edureka.co/blog/wp-content/uploads/2017/05/Deep-Neural-Network-What-is-Deep-Learning-Edureka.png

- **'Deep' refers to the many layers in model**
  - Allows for learning at different levels of abstraction
  - Leads to automatic feature learning & excellent performance

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

APPLICATIONS OF DEEP LEARNING

- **Image classification**
- **Speech recognition**
- **Text summarization**
- **Self-driving cars**
- **Face recognition**
- **Drug design**
- **Precision medicine**
- **Fraud detection**
- **Targeted ads**
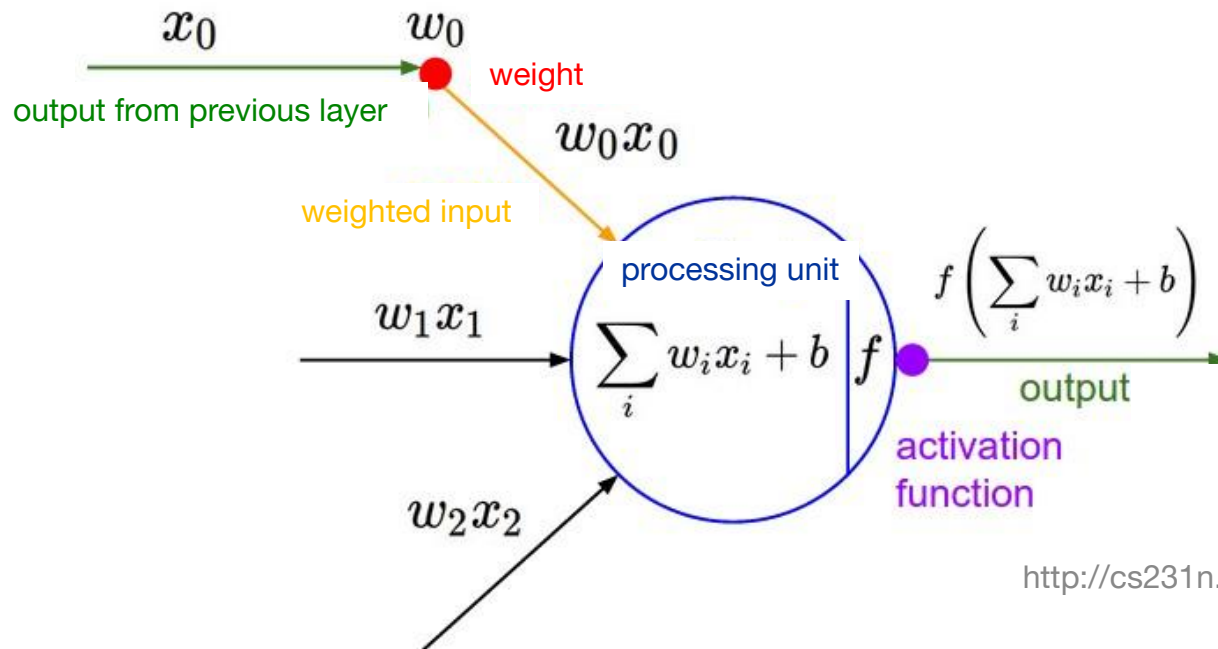- **Stock market analysis**
- **Many others ...**

NEURAL NETWORK
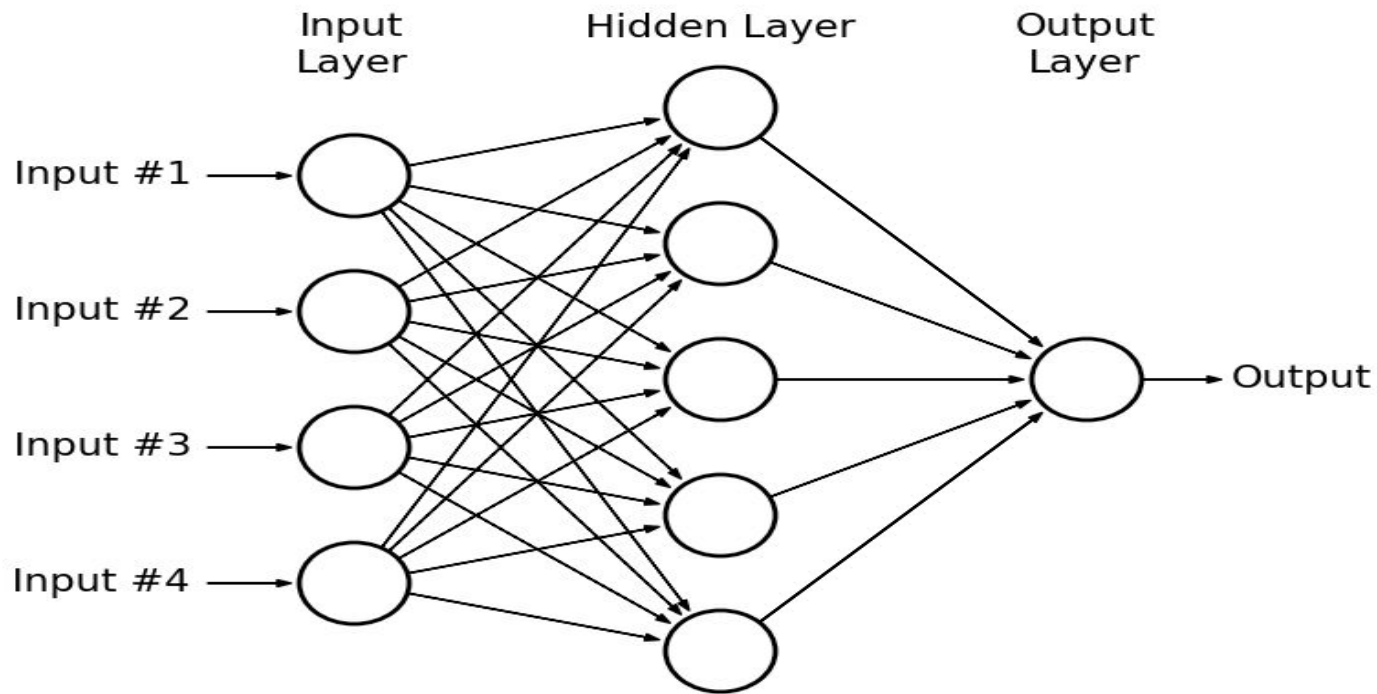
http://www.astroml.org/book_figures/appendix/fig_neural_network.html

- Machine learning model
- Consists of processing units connected by weights
- Learns mapping from input to output based on training data
- Inspired by biological neural systems

PROCESSING UNIT IN NEURAL NETWORK

x_0

w_0

weight

output from previous layer

$w_0 x_0$

weighted input

processing unit

$w_1 x_1$

$$\sum_i w_i x_i + b \quad f$$

$$f\left(\sum_i w_i x_i + b\right)$$

output

activation function

$w_2 x_2$

http://cs231n.github.io/neural-networks-1/

Steps Performed by Each Unit

- Compute dot product of inputs and weights
- Add bias
- Apply activation function
- Feed output to next layer of units

NEURAL NETWORK TRAINING

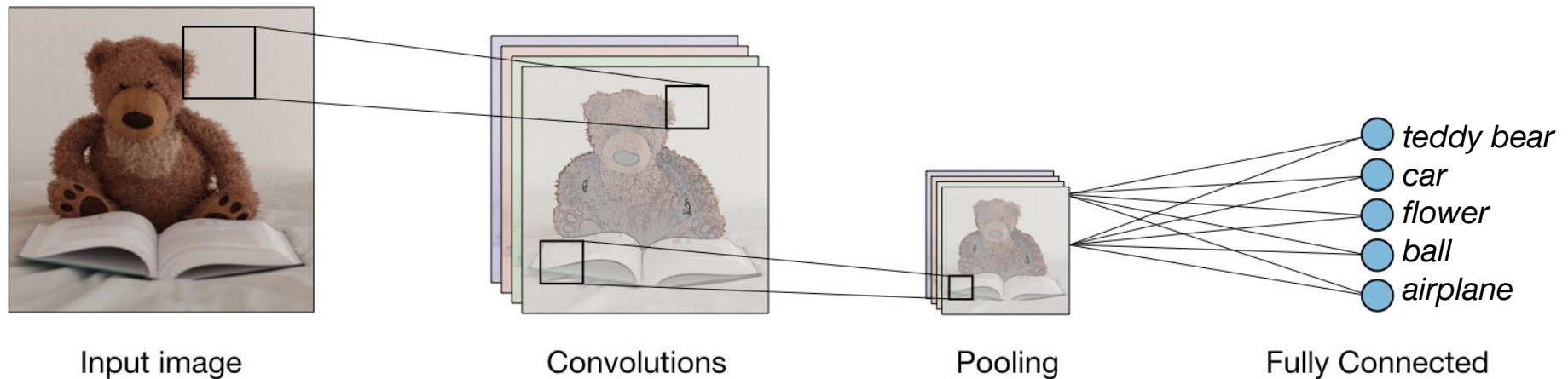https://www.guru99.com/backpropogation-neural-network.html
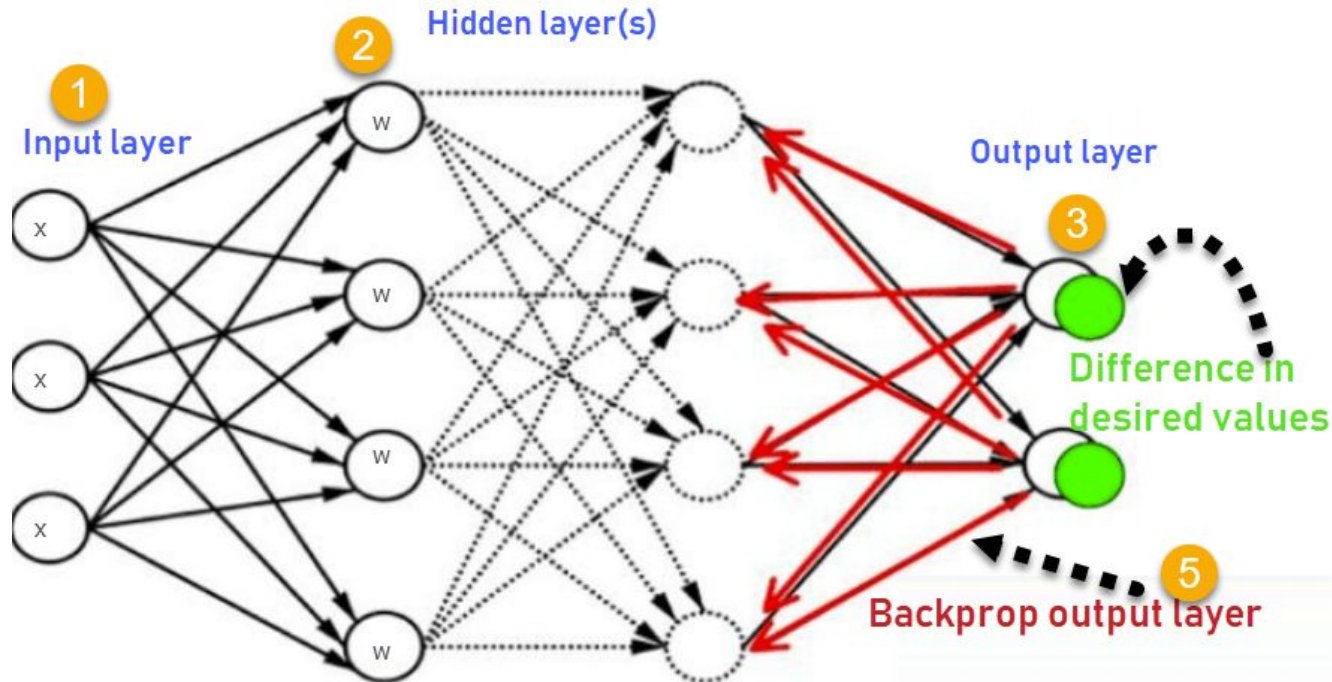
1. Input is fed to network
2. Input is multiplied by weights (i.e., model parameters)
3. Output of one layer is fed as input to the next (forward pass)
4. Error is calculated at output layer
5. Error is backpropagated to adjust weights in order to decrease error based on loss function
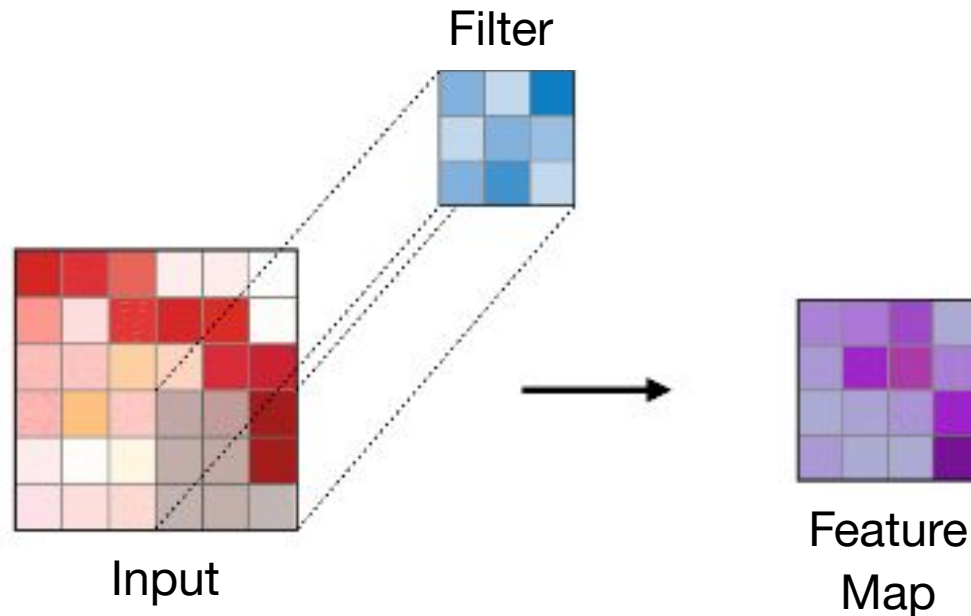
DEEP LEARNING MODELS

- ## General Deep Network Architecture:
  - Has sequence of layers
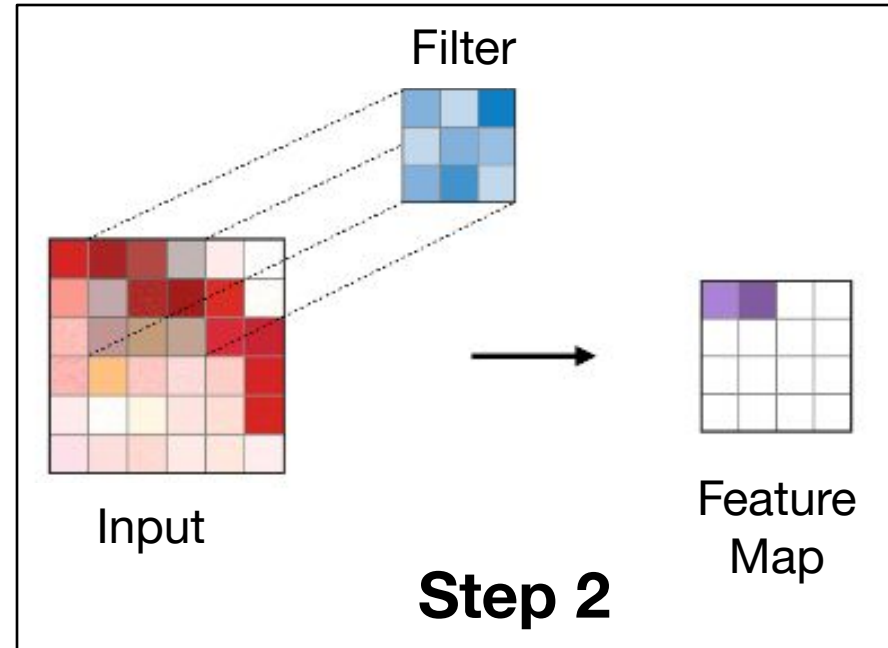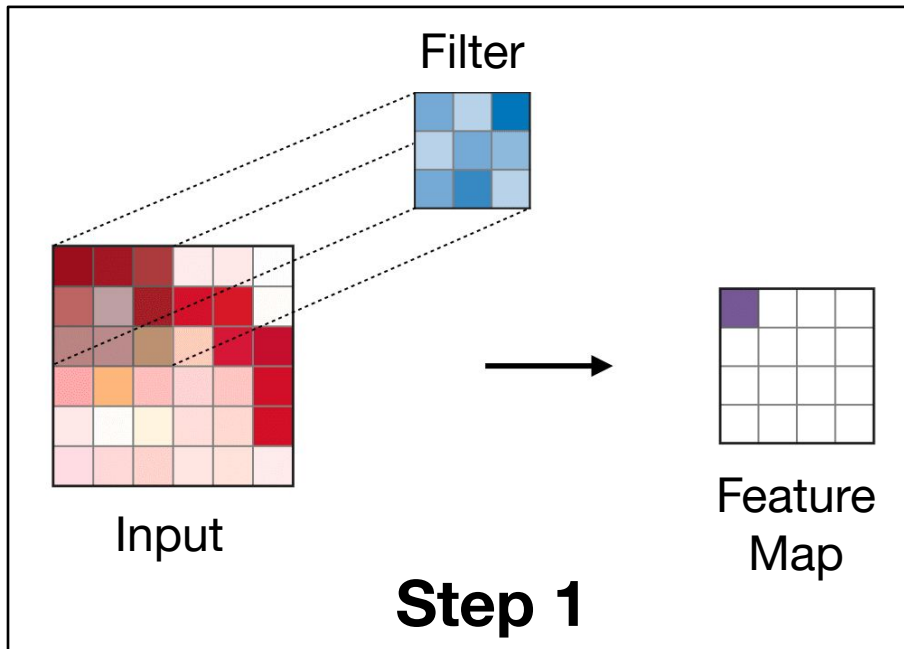 - Each layer transforms its input to generate an output through nonlinear function

Input image   Convolutions   Pooling   Fully Connected

teddy bear
car
flower
ball
airplane

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer

SDSC SAN DIEGO SUPERCOMPUTER CENTER   UC San Diego

CONVOLUTION LAYER

- Core building block of CNN
- Performs convolution operations on input using convolution filters
- Filter operates on local region of input and slides over input
- Filters have parameters that are adjusted during training
- Filters learn to detect features in input important for prediction task

Filter

Input

Feature
Map

CONVOLUTION FILTER

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer

- Filter size = receptive field of filter
- Stride = sliding amount, i.e., # pixels by which filter is moved over image
- Padding = padding around input volume
- Depth = number of filters

CONVOLUTION OPERATION

1	0	1
0	1	0
1	0	1

3 x 3 Filter

Steps

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

→

4		

Input

Feature Map

http://ufldl.stanford.edu/tutorial/supervised/Feat
ureExtractionUsingConvolution/

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

CONVOLUTION OPERATION

3 x 3 Filter

Steps

Input

Feature Map

Step 2

Input

Feature Map

http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/

CONVOLUTION OPERATION
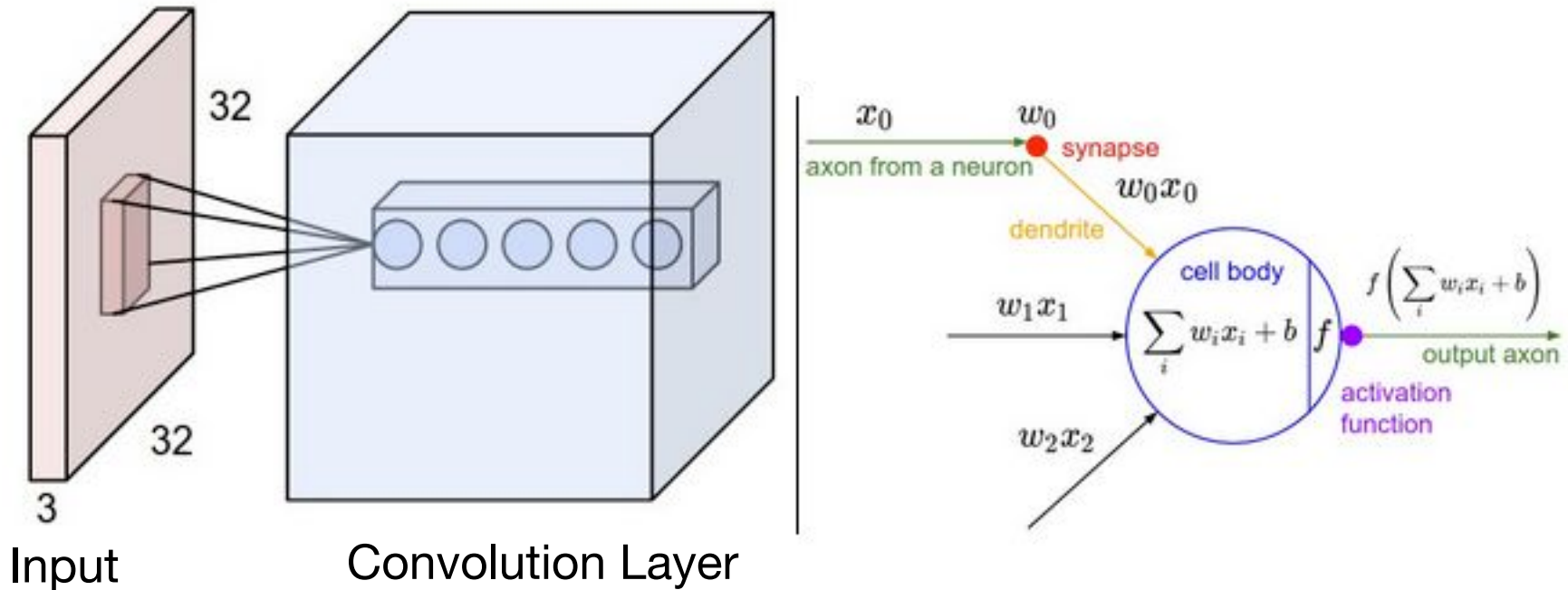

3 x 3 Filter

Steps

Input

Feature Map
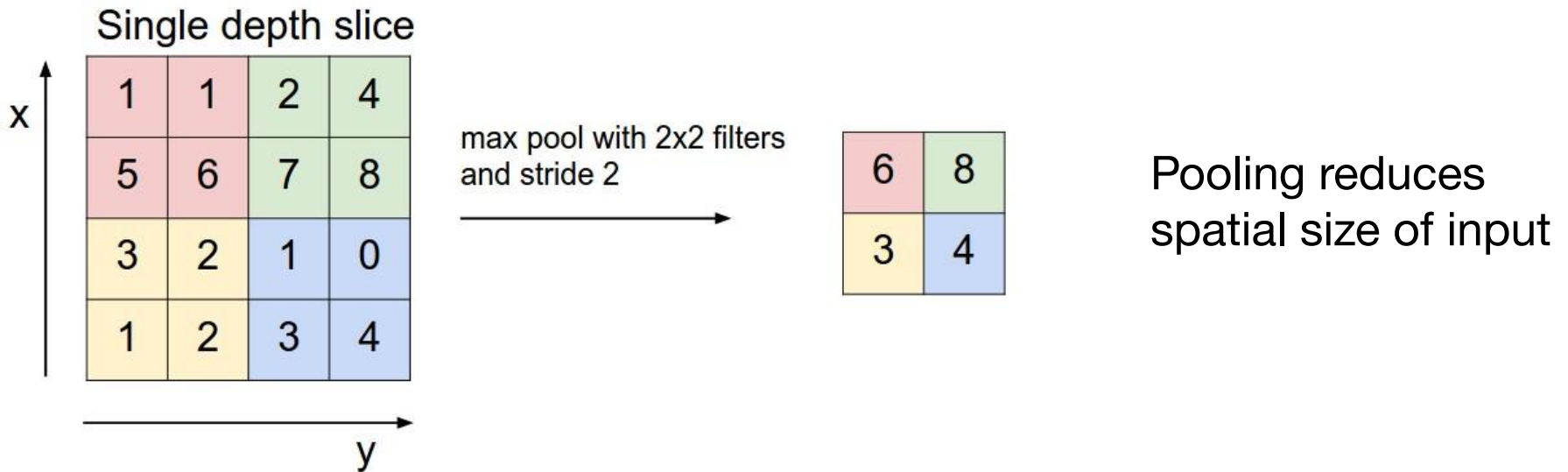
Step 9

Input

Feature Map

CONVOLUTION LAYER

- Performs convolution on input volume (height X width X channels) with filters
- Each filter in convolution layer is connected to local region in input
- Result of convolution is passed through nonlinear activation function
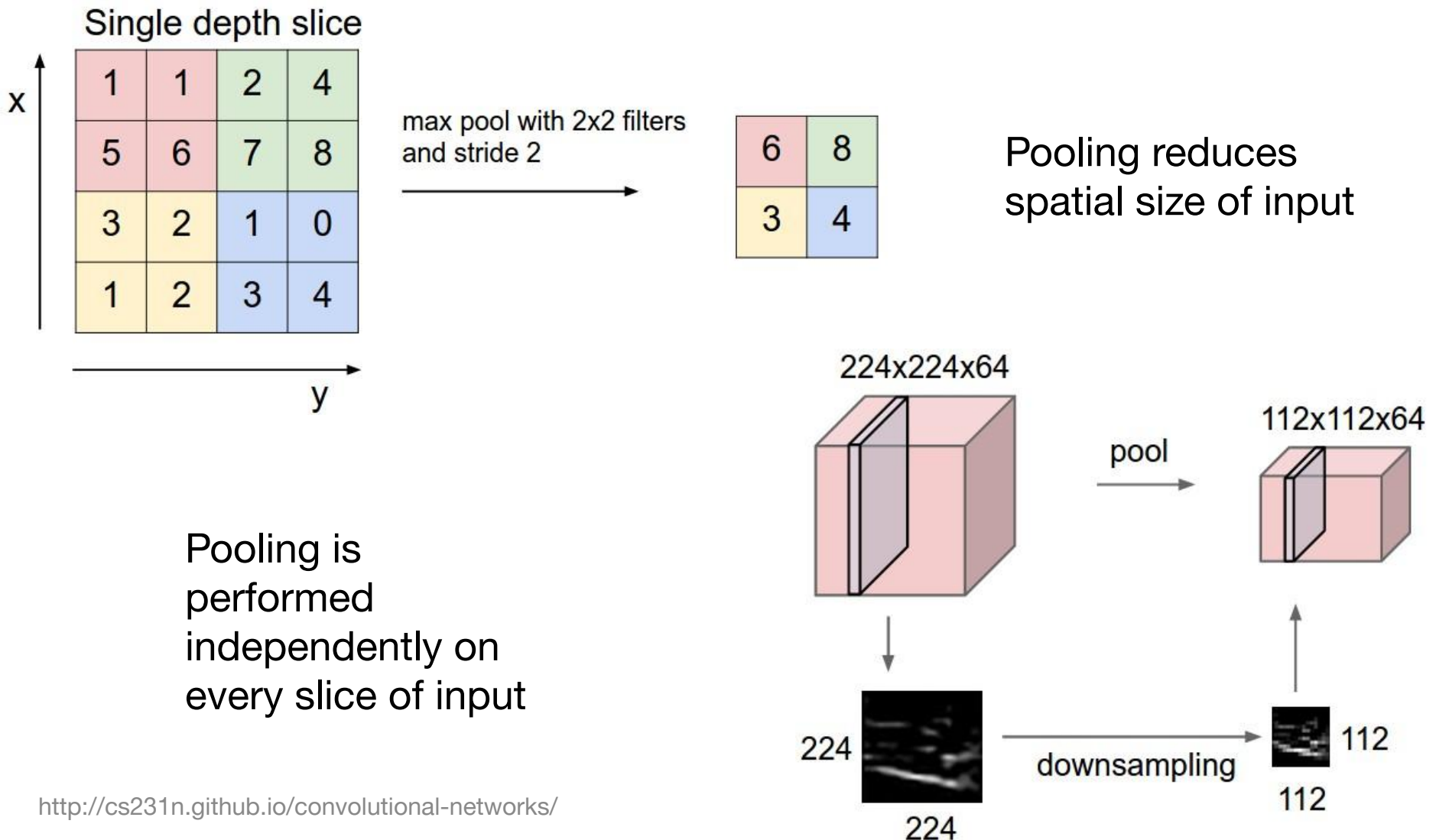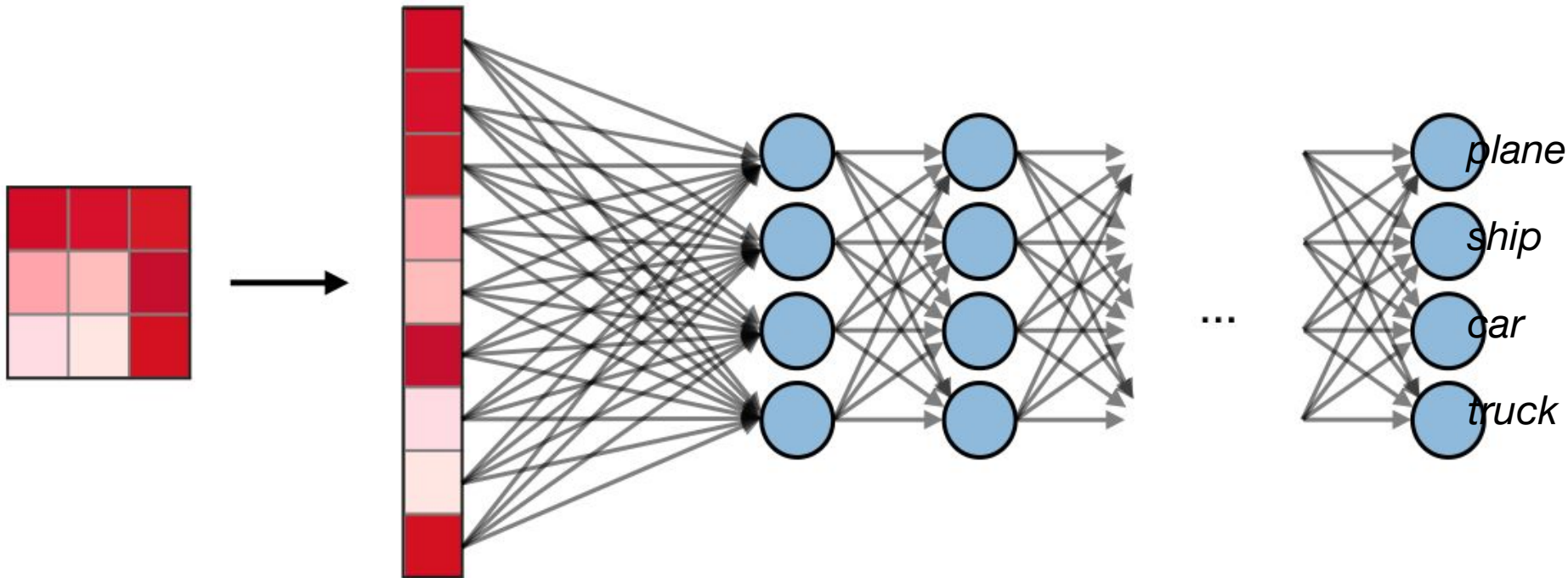- Depth = number of filters = number of feature maps in convolution layer

Input   Convolution Layer

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

POOLING LAYER

Single depth slice

max pool with 2x2 filters and stride 2

Pooling reduces spatial size of input

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

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

POOLING LAYER

Single depth slice

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

x ↑

→ y

max pool with 2x2 filters
and stride 2

→

6	8
3	4

Pooling reduces
spatial size of input

Pooling is
performed
independently on
every slice of input

224x224x64

pool →

112x112x64

224

224

downsampling →

112

112

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

FULLY CONNECTED LAYER

- Fully connected (FC) layer takes flattened input.
- Every input is connected to all processing units.
- Output of FC layer is typically vector with probabilities for categories.
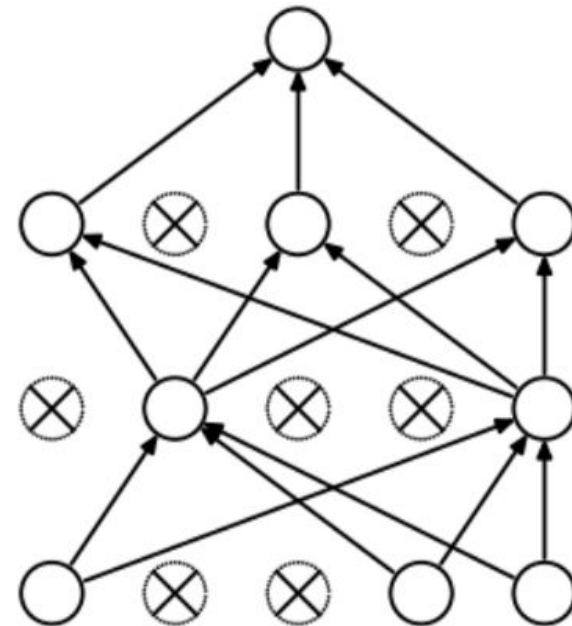

plane
ship
car
truck

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer

DROPOUT

- **Randomly drop units during training**
- **Prevents units from co-adapting**
- **Helps to address overfitting**
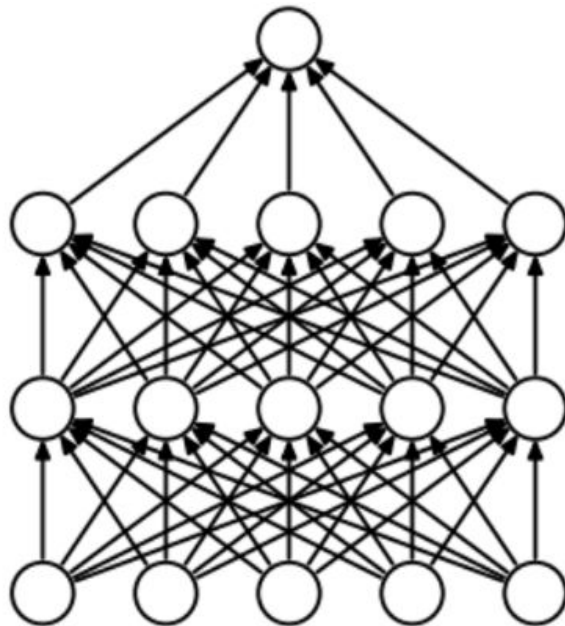
(a) Standard Neural Net   (b) After applying dropout.

BATCH NORMALIZATION

- **Normalizes input to layer**
  - Subtract mean and divide by standard deviation for each mini-batch
- **Benefits**
  - Increased stability
  - Faster convergence
  - Less sensitive to weight initialization
  - Reduces overfitting

CONVOLUTIONAL NEURAL NETWORK (CNN)

- Model consists of several repeating sets of layers called 'blocks'
- Input volume is tensor of size width X height X # of channels
- Output is vector of numbers representing class probabilities

VGG-16 Network

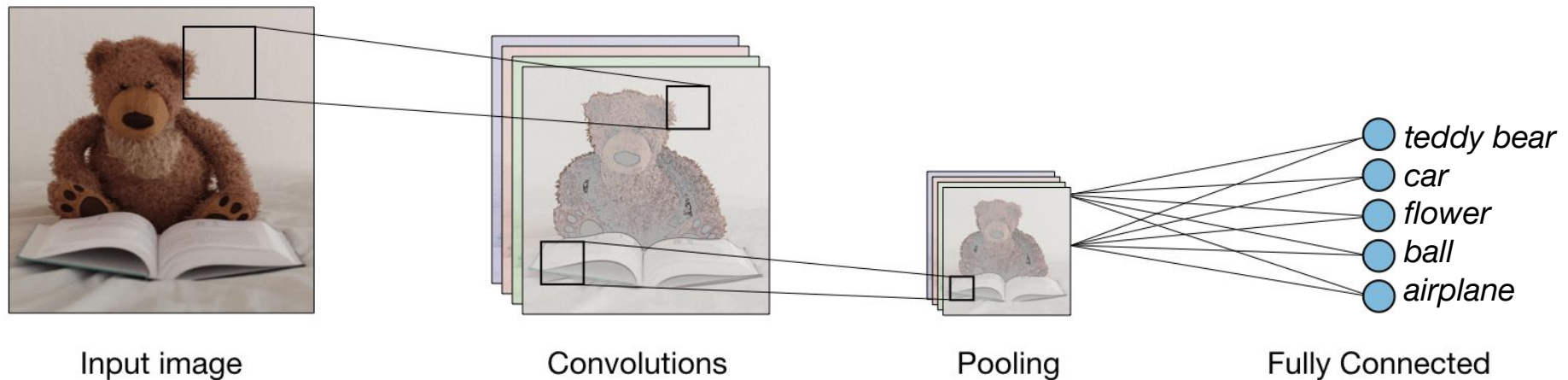https://www.cs.toronto.edu/~frossard/post/vgg16/
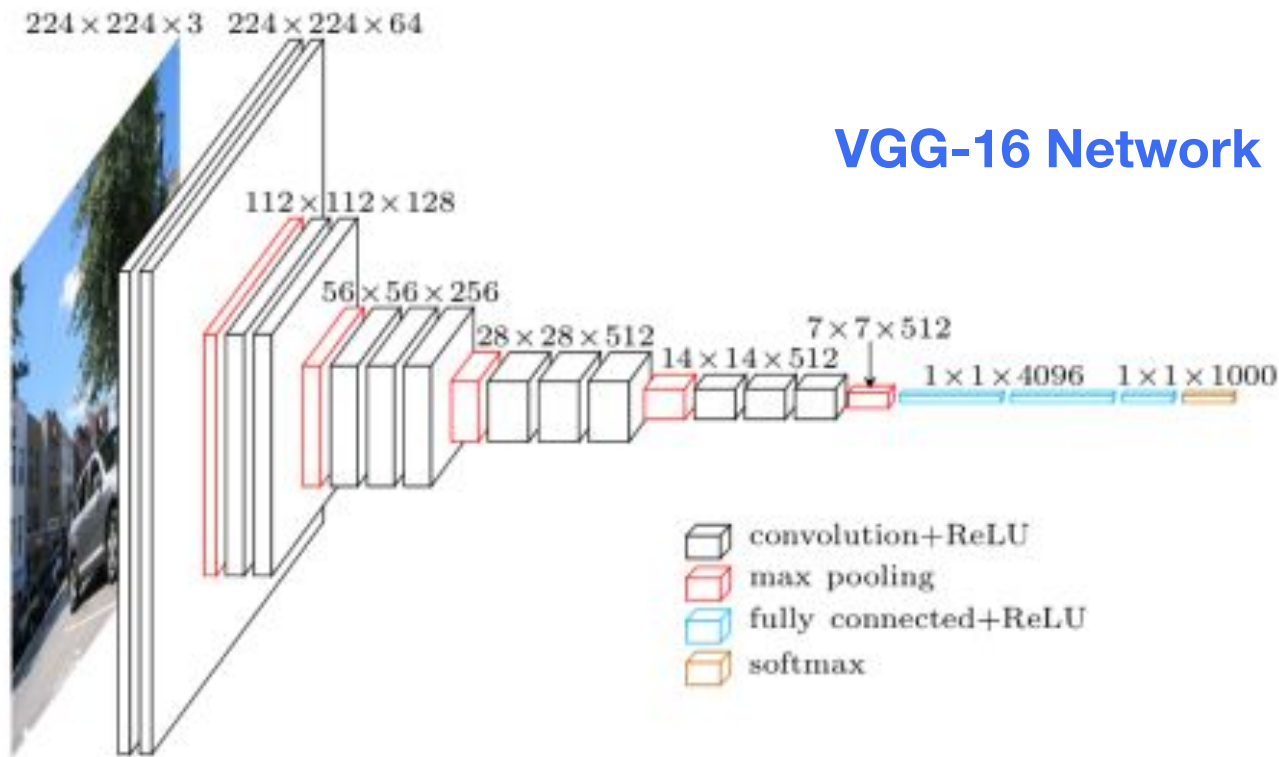
CNN

- **General CNN Architecture**
  - Has sequence of layers
  - Each layer transforms its input to generate an output through nonlinear function
  - Has different types of layers

Input image   Convolutions   Pooling   Fully Connected

teddy bear
car
flower
ball
airplane

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer

CNN Models

- **LeNet**
- **AlexNet**
- **VGG**
- **Inception**
- **ResNet**
- **XceptionNet**
- **Inception-ResNet**
- **…**

CNN Applications

- **Image Analysis**
  - Object classification, localization, detection
  - Face recognition
  - Text classification
- **Natural Language Processing**
  - Topic modeling
  - Part-of-speech tagging
- **Others**
  - Drug design
  - Crime hot spots identification
  - House price prediction

AUTOENCODER

- Input is fed to hidden layer
- Output is reconstructed version of input
- Model learns to reconstruct input data

AUTOENCODER

https://www.jeremyjordan.me/autoencoders/
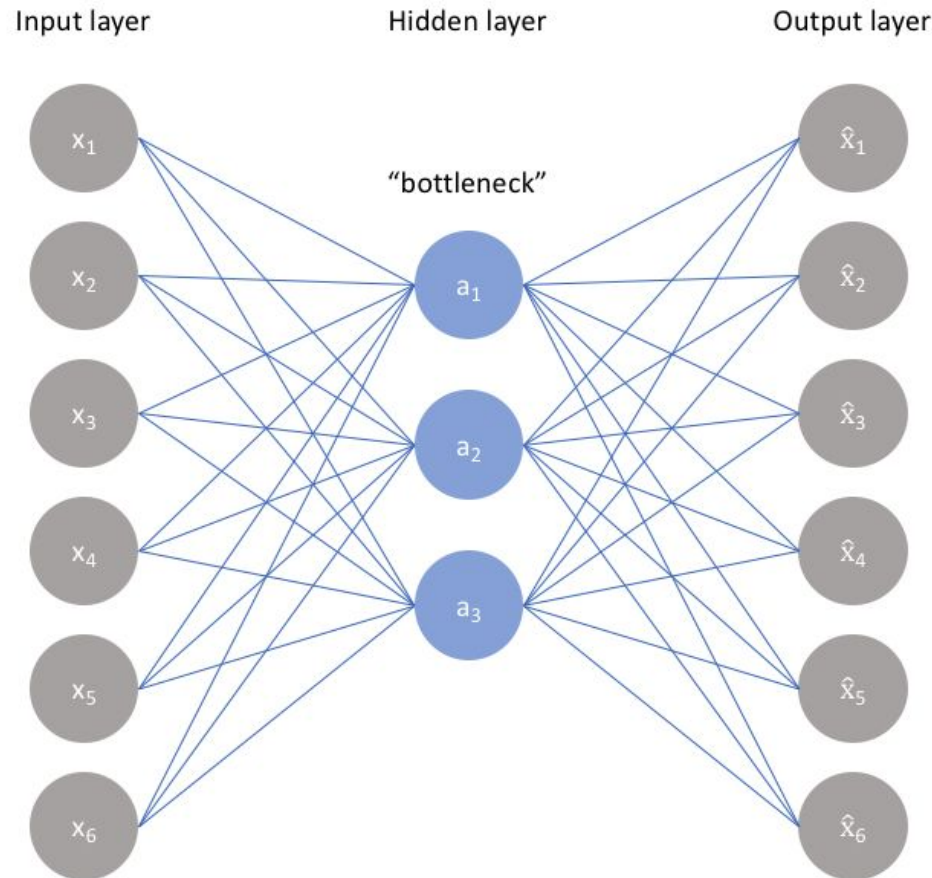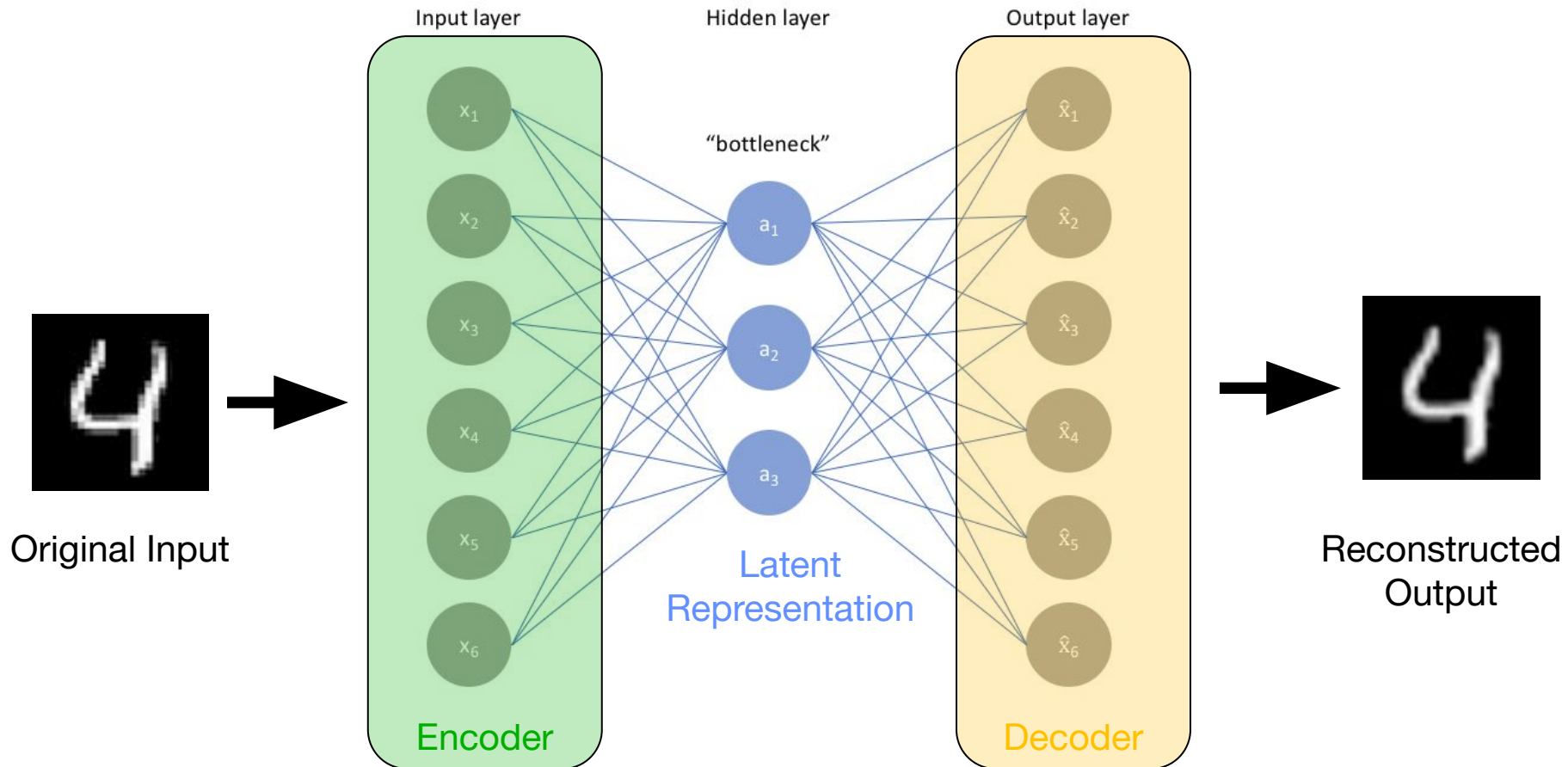
- "Bottleneck" layer provides encoding of input
- Used to generate latent representation of data

AUTOENCODER

- **Uses**
  - Feature learning
    - Generated features useful for downstream tasks (e.g., classification, anomaly detection, clustering)
  - As part of larger deep learning model

- **Variations**
  - Sparse
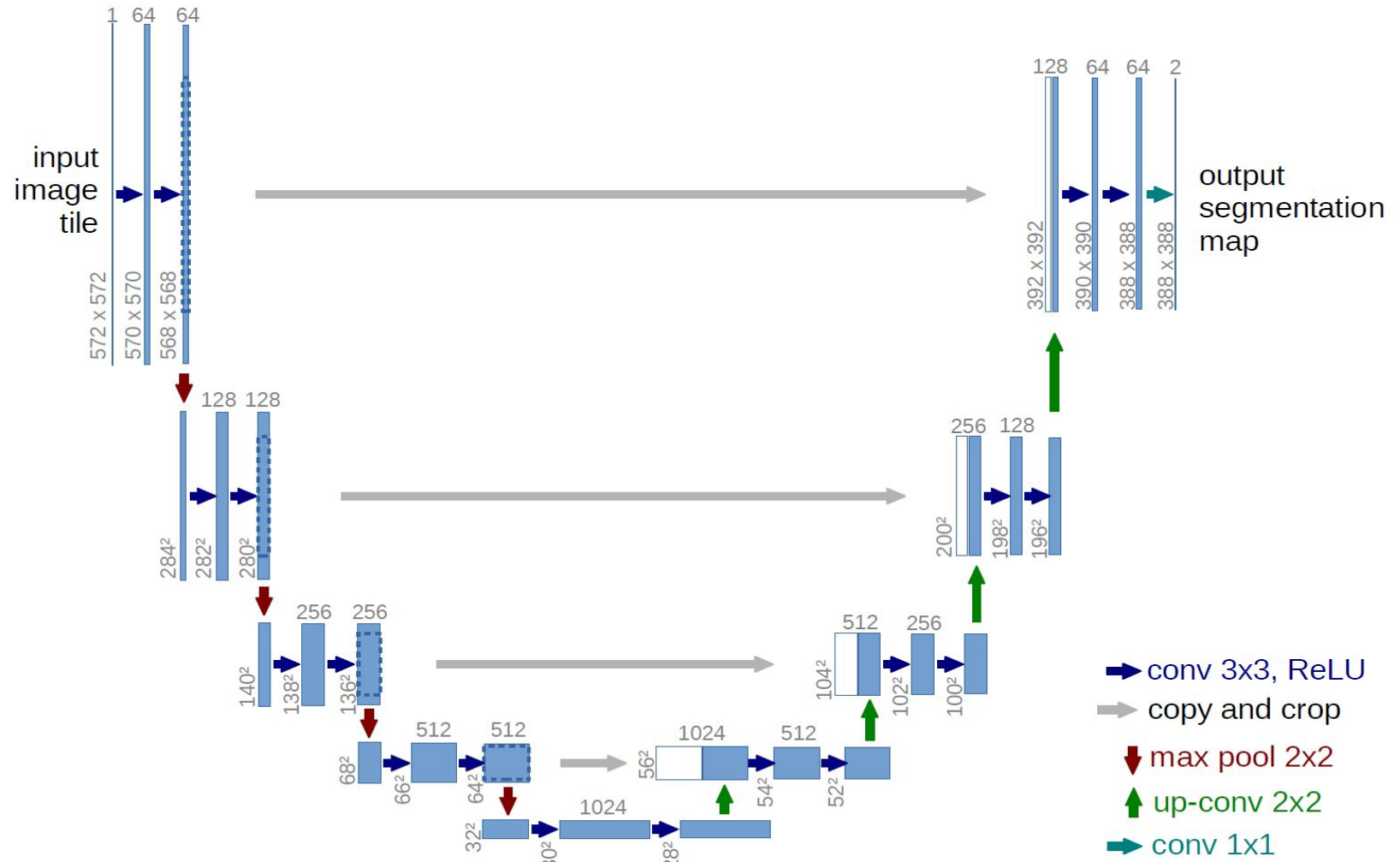 - Denoising
  - Contractive
  - Variational

U-NET

- **Semantic Segmentation**
  - Dividing image into multiple salient image regions
  - Assign label to every pixel in image
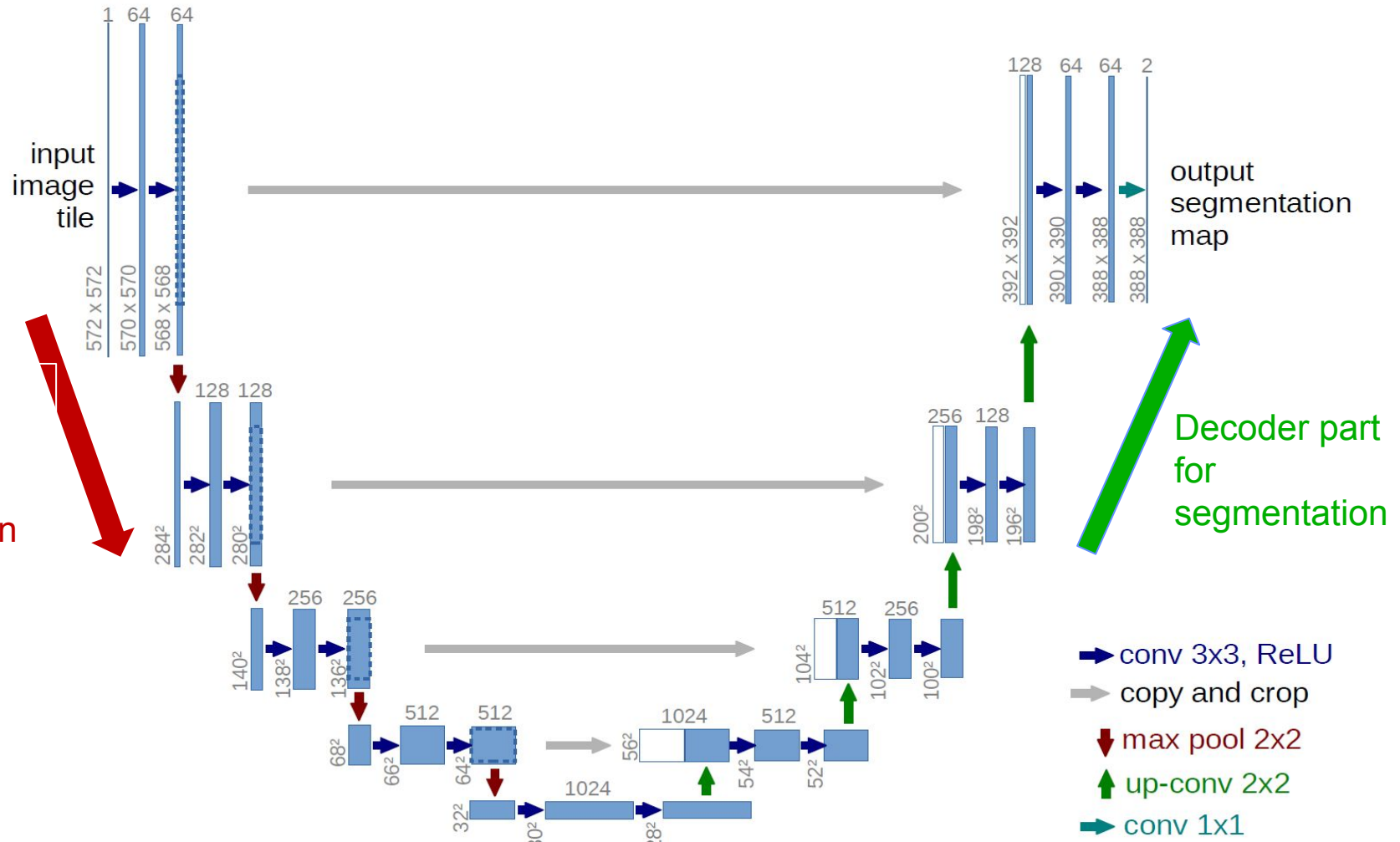 - Pixels with same label are similar

30 FPS

https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066
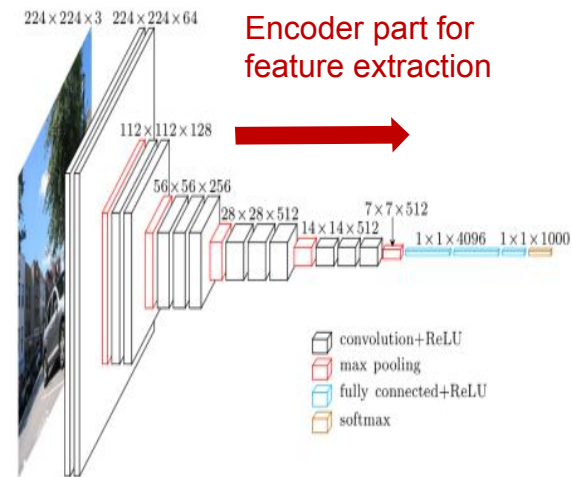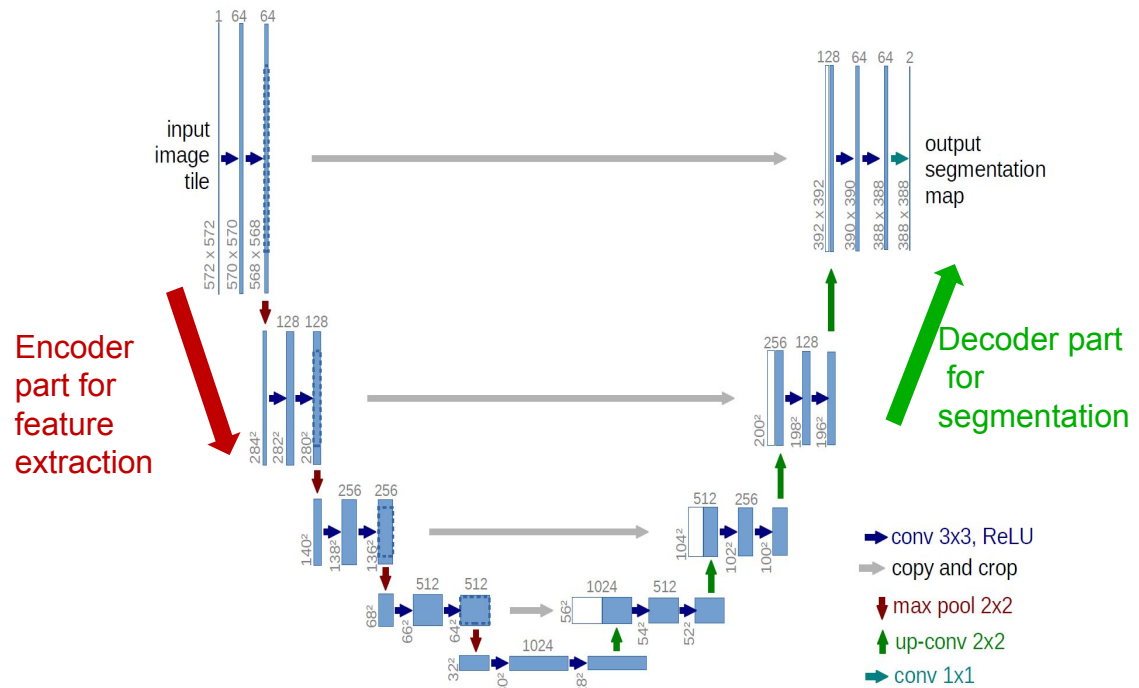
U-NET ARCHITECTURE

U-NET ARCHITECTURE

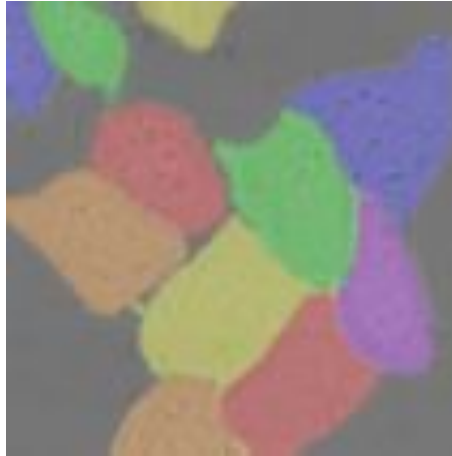https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

U-NET ARCHITECTURE

VGG16 CNN Architecture
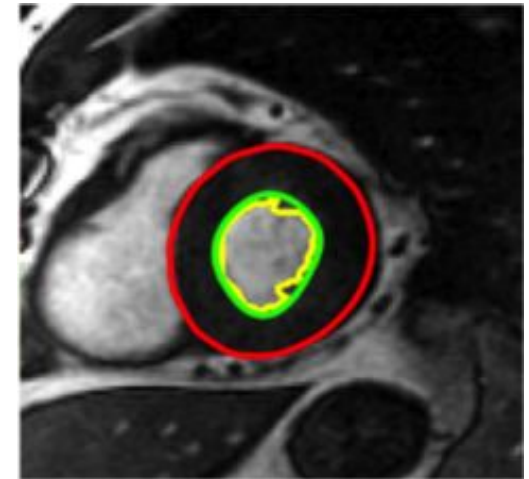
U-Net Architecture

U-NET APPLICATIONS

Biomedical Segmentation
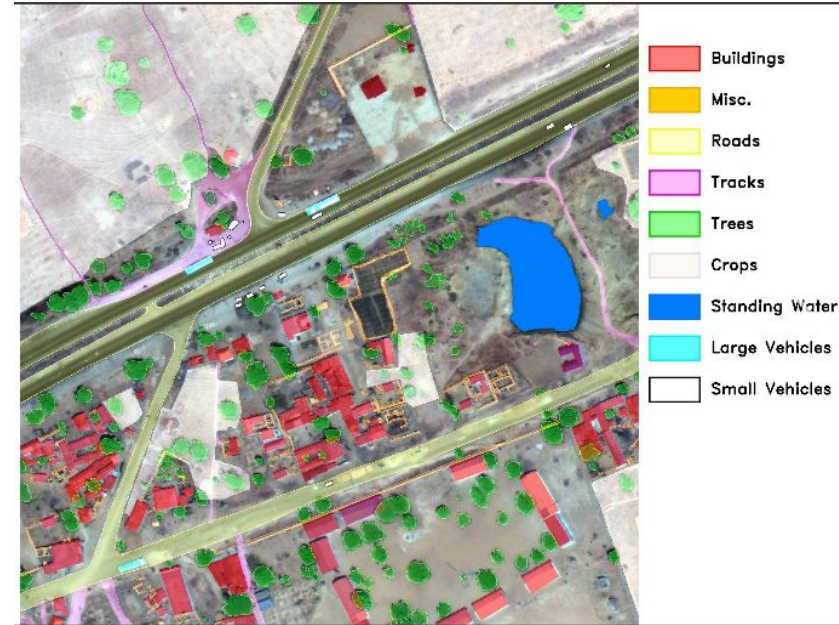
Satellite Image Processing

Buildings
Misc.
Roads
Tracks
Trees
Crops
Standing Water
Large Vehicles
Small Vehicles

Medical Image Analysis

Object Detection

LSTM

- **Long Short-Term Memory**
- **Used for sequence learning**
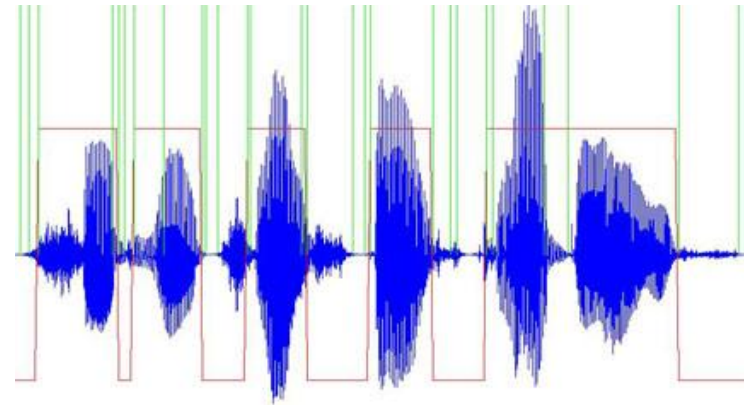- **Type of Recurrent Neural Network (RNN)**

LSTM

- **Sequence Learning**
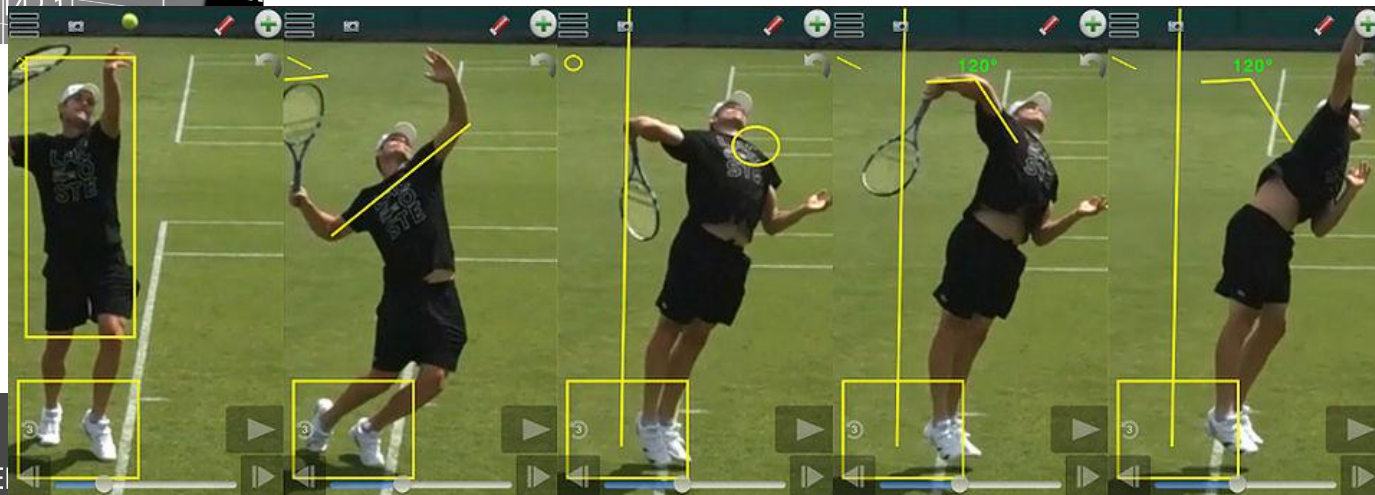 - Learning a signal with an ordering or time component


Stock Price

Speech

Video

RECURRENT NEURAL NETWORK (RNN)

- **Can model sequences and time-dependent signals**
- **Have cyclic connections that feed previous activations as part of input back to network**
  - Allows for temporal contextual information to be stored
  - Predictions at current time step depend on current input and previous predictions
  - Context required must be learned

LSTM ARCHITECTURE

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
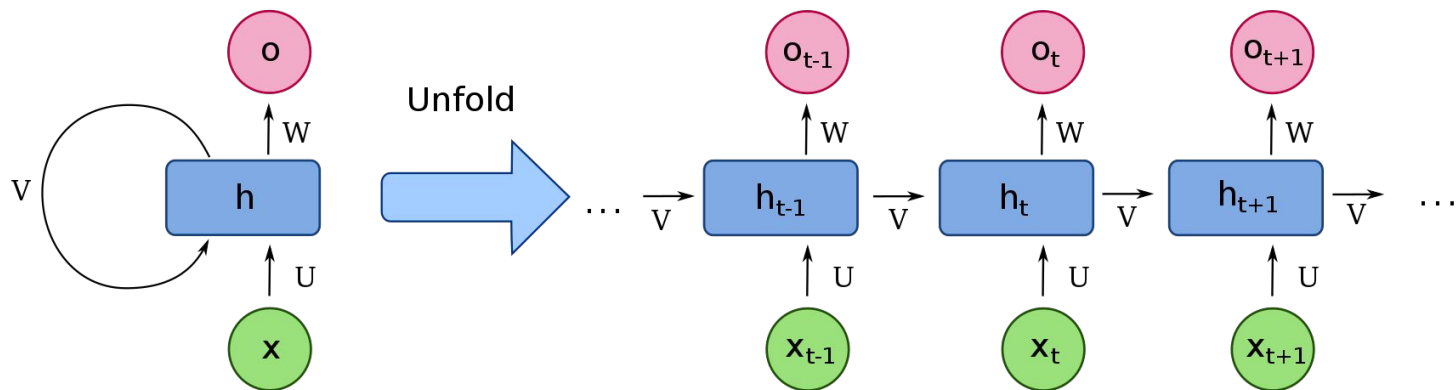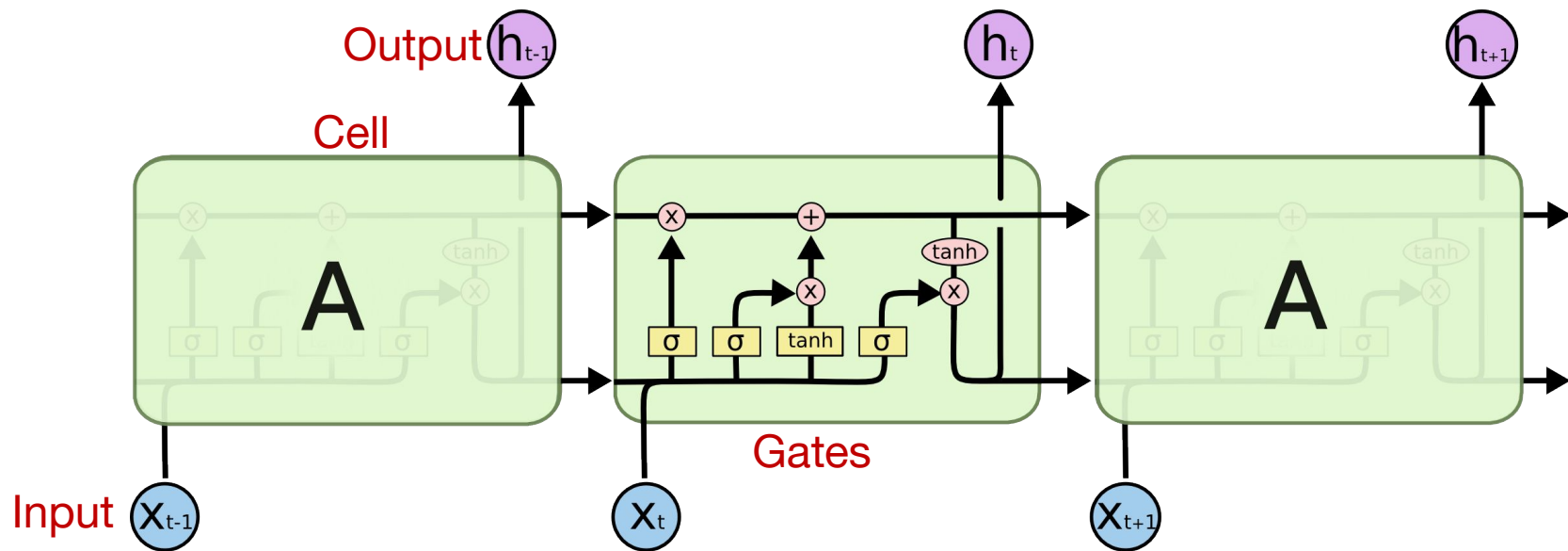
- Info flows through memory blocks called 'cells'
- Structure of cell allows LSTM to selectively remember/forget pieces of information
- Each cell manipulates memory through 'gates'

LSTM Applications

- **Speech recognition**
- **Machine translation**
- **Language modeling**
- **Speech synthesis**
- **Handwriting recognition**
- **Text generation**
- **Video analysis**
- **Protein structure prediction**
- **Stock price prediction**

SEQ2SEQ

- Converts input sequence to output sequence
  - machine translation, question-answering
- Encoder & decoder are RNNs
- Issue: Difficult to capture long-range dependencies
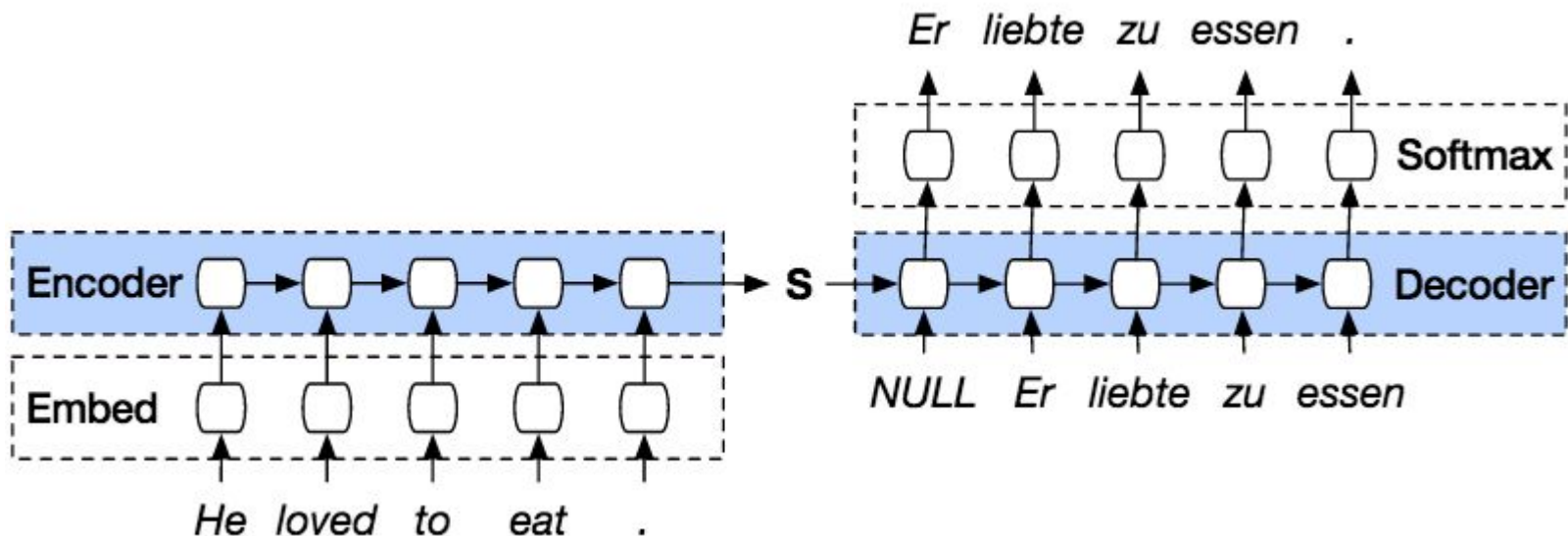
ATTENTION MECHANISM

For each word, attention is used to determine which other words in sequence are important
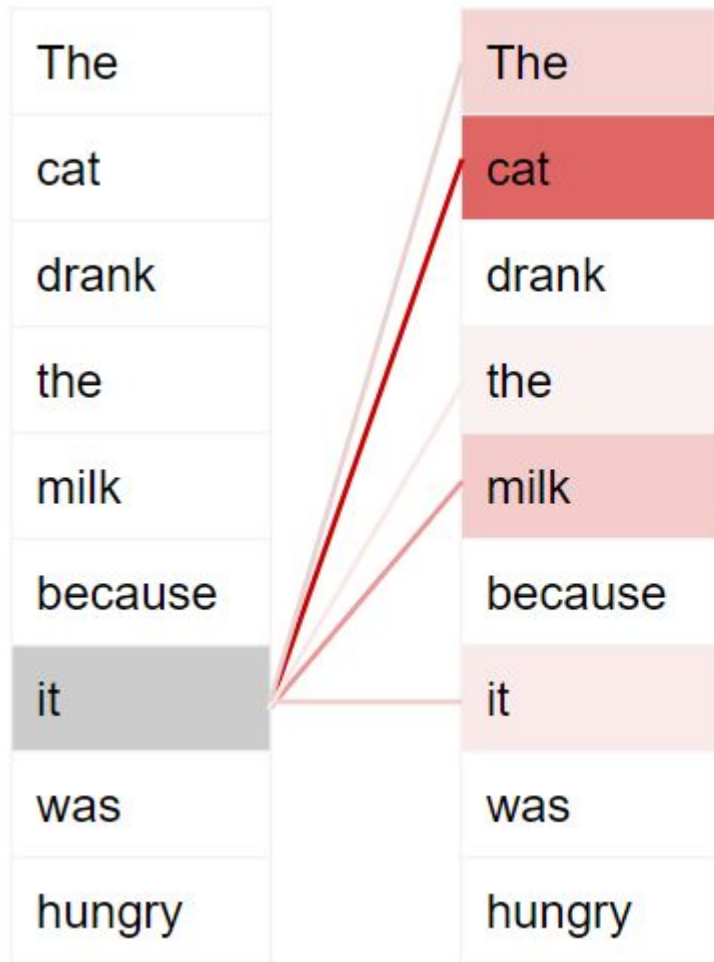
TRANSFORMER

- Encoder-decoder model
- Uses only attention to capture relationships between words in sentence ("Attention is All You Need")
- No recurrence or convolutions

https://arxiv.org/abs/1706.03762

Encoder

Decoder
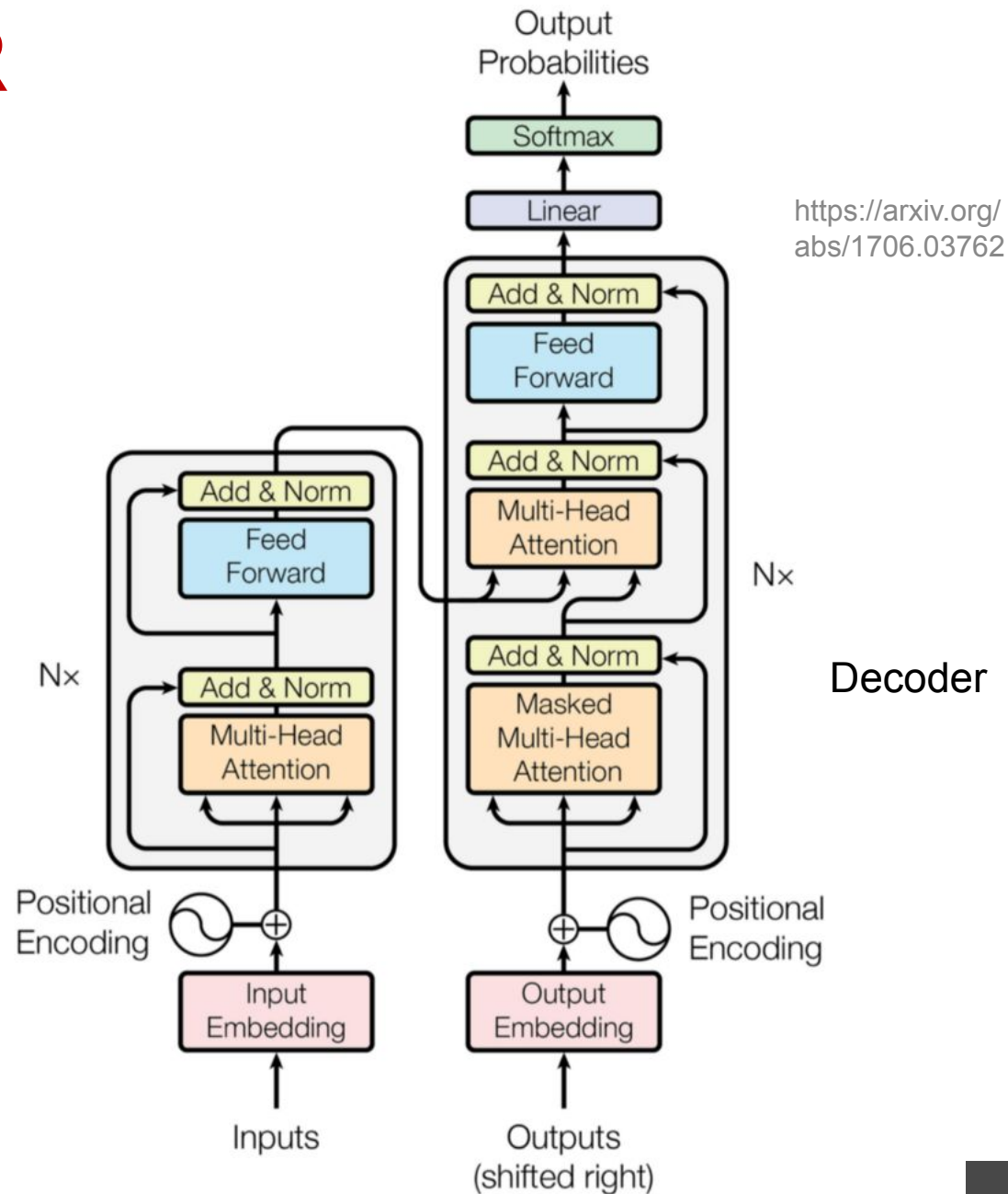
Figure 1: The Transformer - model architecture.

TRANSFORMER ADVANTAGES OVER RNN

- Long-range dependencies can be captured
- All words in sequence are processed in parallel
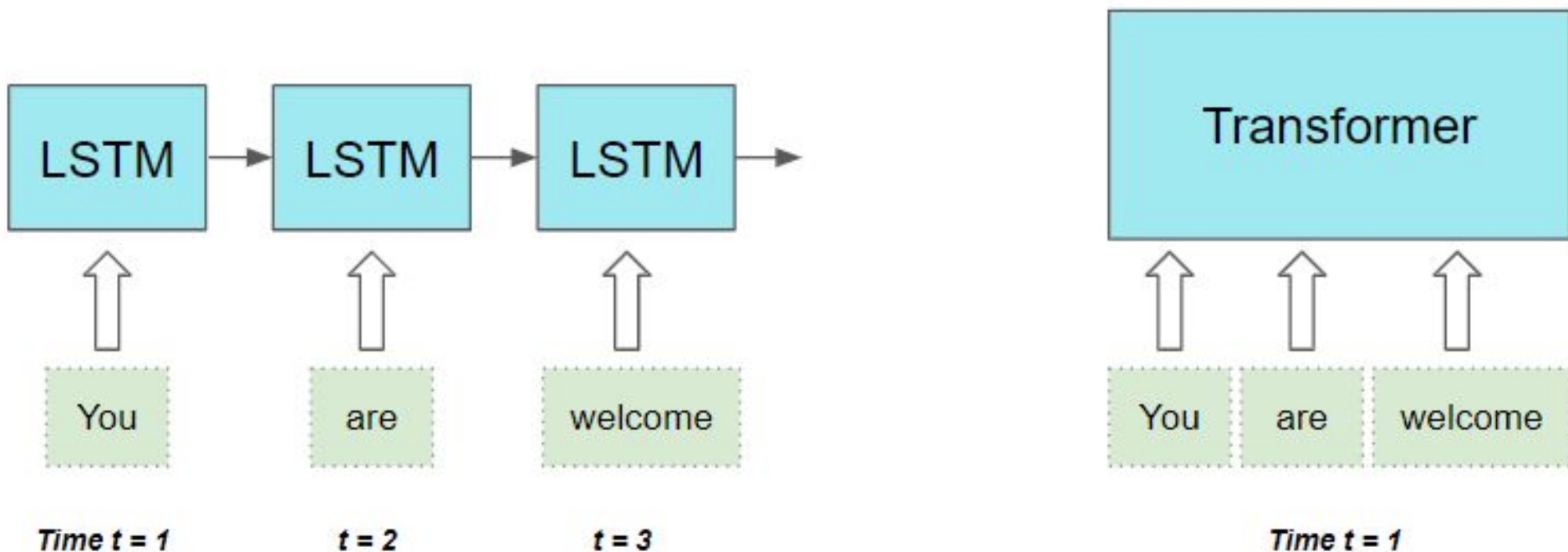
BERT

- Bidirectional Encoder Representations for Transformers
- Transformer trained as a language model
  - Encoding part only
- Pre-trained on Wikipedia and Books Corpus
- Can be fine-tuned for various NLP tasks
  - e.g., named entity recognition, relation extraction, question-answering, sentiment analysis

GPT-3

Generative Pre-trained Transformer 3

Can generate text to answer questions, write essays, summarize text, translate languages

OpenAI's GPT-3 may be the biggest thing since bitcoin

OpenAI, a non-profit artificial intelligence research company backed by Peter Thiel, Elon Musk, Reid Hoffman, Marc Benioff, Sam Altman and others, released its third generation of language prediction model (GPT-3) into the open-source wild. Language models allow computers to produce random-ish sentences of approximately the same length and grammatical structure as those in a given body of text.

In my early experiments with GPT-3 I found that GPT-3's predicted sentences, when published on the bitcointalk.org forum, attracted lots of positive attention from posters there, including suggestions that the system must have been intelligent (and/or sarcastic) and that it had found subtle patterns in their posts. I imagine that similar results can be obtained by republishing GPT-3's outputs to other message boards, blogs, and social media.

https://maraoz.com/2020/07/18/openai-gpt3/

This was written by GPT-3!

Prompt:

<Author's Bio>

Title: OpenAI's GPT-3 may be the biggest thing since bitcoin

tags: tech, machine-learning, hacking

Summary: I share my early experiments with OpenAI's new language prediction model (GPT-3) beta. I explain why I think GPT-3 has disruptive potential comparable to that of blockchain technology.
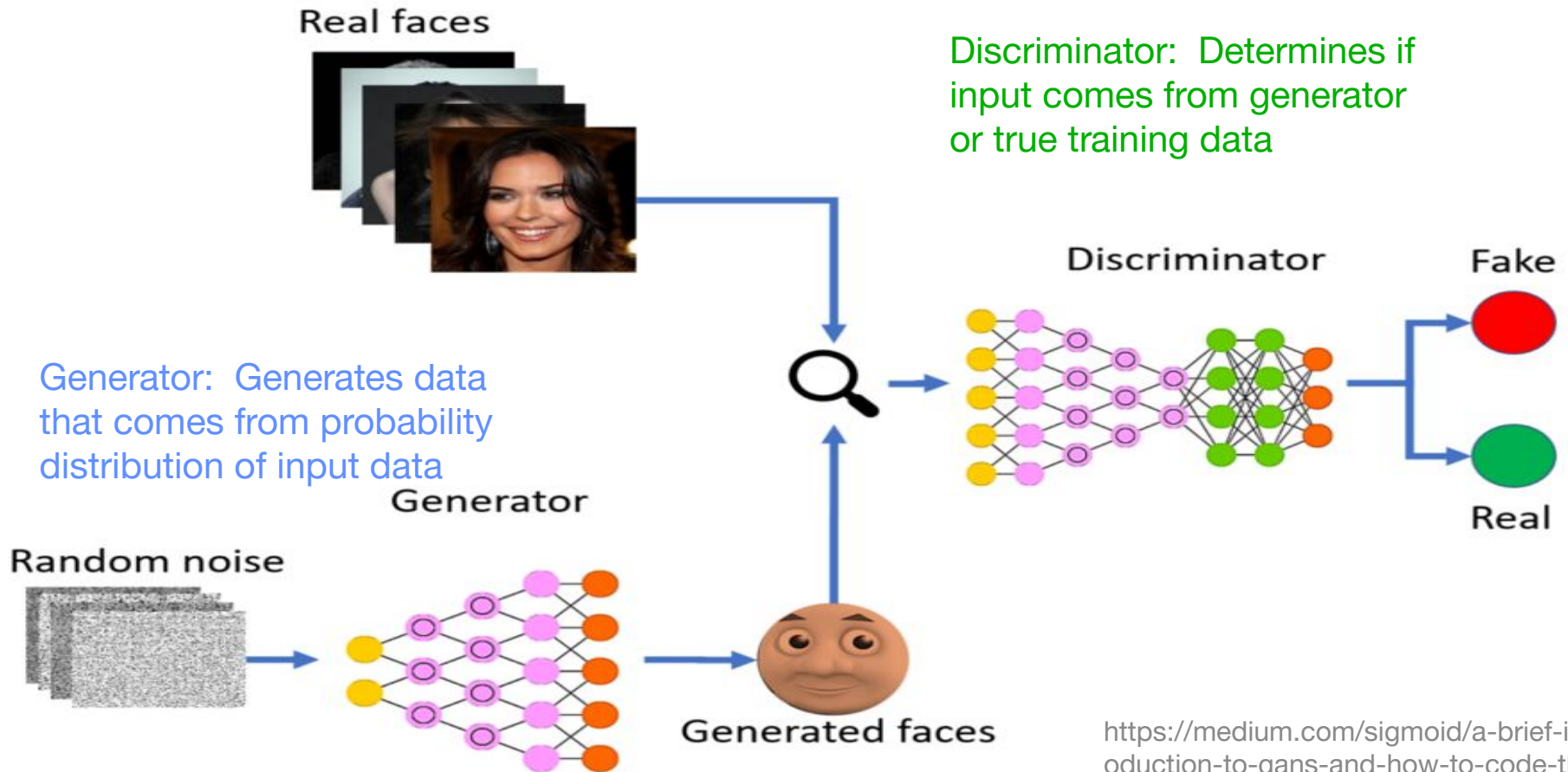
TRANSFORMER APPLICATIONS

- **NLP tasks**
  - machine translation
  - text summarization
  - question-answering
  - named entity recognition
- **Vision tasks**
  - video classification
  - object detection
  - image classification
  - image generation
- **Both**
  - image captioning

GENERATIVE ADVERSARIAL NETWORKS (GANs)

- **Deep learning approach to generative modeling**
- **Allows for model to generate data**
  - Model learns structure of input data to generate new data with similar characteristics as input data
- **Consists of two models**
  - **Generator**: Generates new samples
  - **Discriminator**: Determines if sample is generated (fake) or from input data (real)
  - Trained in an adversarial way

GAN ARCHITECTURE

Real faces

Discriminator: Determines if
input comes from generator
or true training data

Discriminator

Fake

Generator: Generates data
that comes from probability
distribution of input data

Generator

Real

Random noise

Generated faces
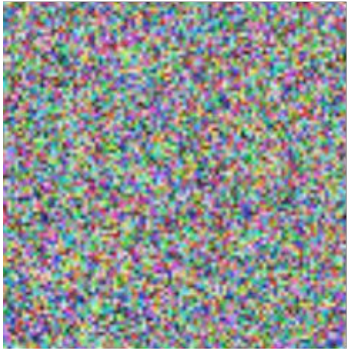
https://medium.com/sigmoid/a-brief-introduction-to-gans-and-how-to-code-them-2620ee465c30

GAN APPLICATIONS

Noise ~ N(0,1)

Generative Model

https://arxiv.org/pdf/1710.10196.pdf

GAN APPLICATIONS

- **Image-to-Image Translation**
  - Transform images from one domain (e.g., real scenery) to another domain (Monet paintings)

https://junyanz.github.io/CycleGAN/
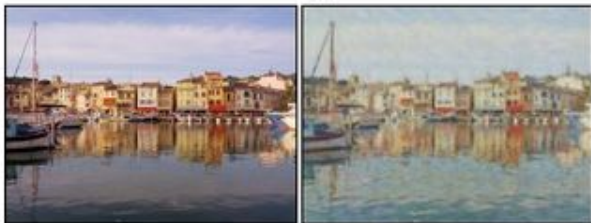
GAN APPLICATIONS

- **Superresolution**
  - Create high-resolution images from lower-resolution images

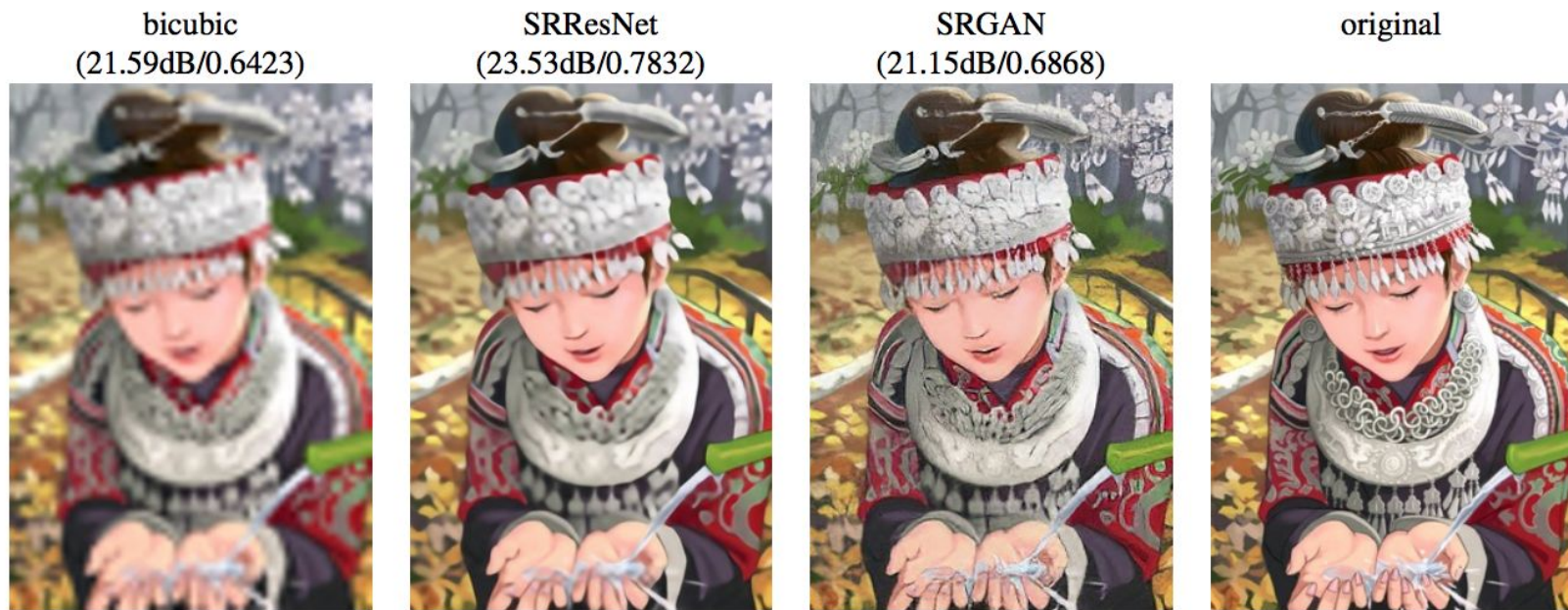| bicubic (21.59dB/0.6423) | SRResNet (23.53dB/0.7832) | SRGAN (21.15dB/0.6868) | original |

Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

GAN APPLICATIONS

- **Others**
  - Text-to-image translation
  - Face view generation
  - Pose generation
  - Photos to emojis
  - Face aging
  - …

PYTHON DEEP LEARNING LIBRARIES

- **TensorFlow**
  - https://www.tensorflow.org/
  - ML framework developed by Google
  - Keras: High-level NN API. Now part of TensorFlow.

- **PyTorch**
  - https://pytorch.org/
  - ML framework developed by Facebook
  - PyTorch Lightning: High-level API for PyTorch

- **Apache MXNet**
  - https://mxnet.apache.org/
  - DL framework used by AWS

OTHER DEEP LEARNING LIBRARIES

- **Java**
  - Deeplearning4j
- **R**
  - TensorFlow, MXNet
- **Cloud**
  - Google Cloud ML
  - AWS SageMaker
  - Microsoft Azure
  - IBM Watson ML

RESOURCES

- CS231n Convolutional Neural Networks for Visual Recognition: http://cs231n.github.io/

- TensorFlow Getting Started. https://www.tensorflow.org/get_started/

- TensorFlow Neural Network Playground. http://playground.tensorflow.org/

- PyTorch Tutorials: https://pytorch.org/tutorials/

- U-Net Paper: https://arxiv.org/abs/1505.04597

- LSTM Paper: https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

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

- Transformer Paper: https://arxiv.org/abs/1706.03762

- The Illustrated Transformer: https://jalammar.github.io/illustrated-transformer/

- GAN Paper: https://arxiv.org/abs/1406.2661

- GAN Introduction: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/