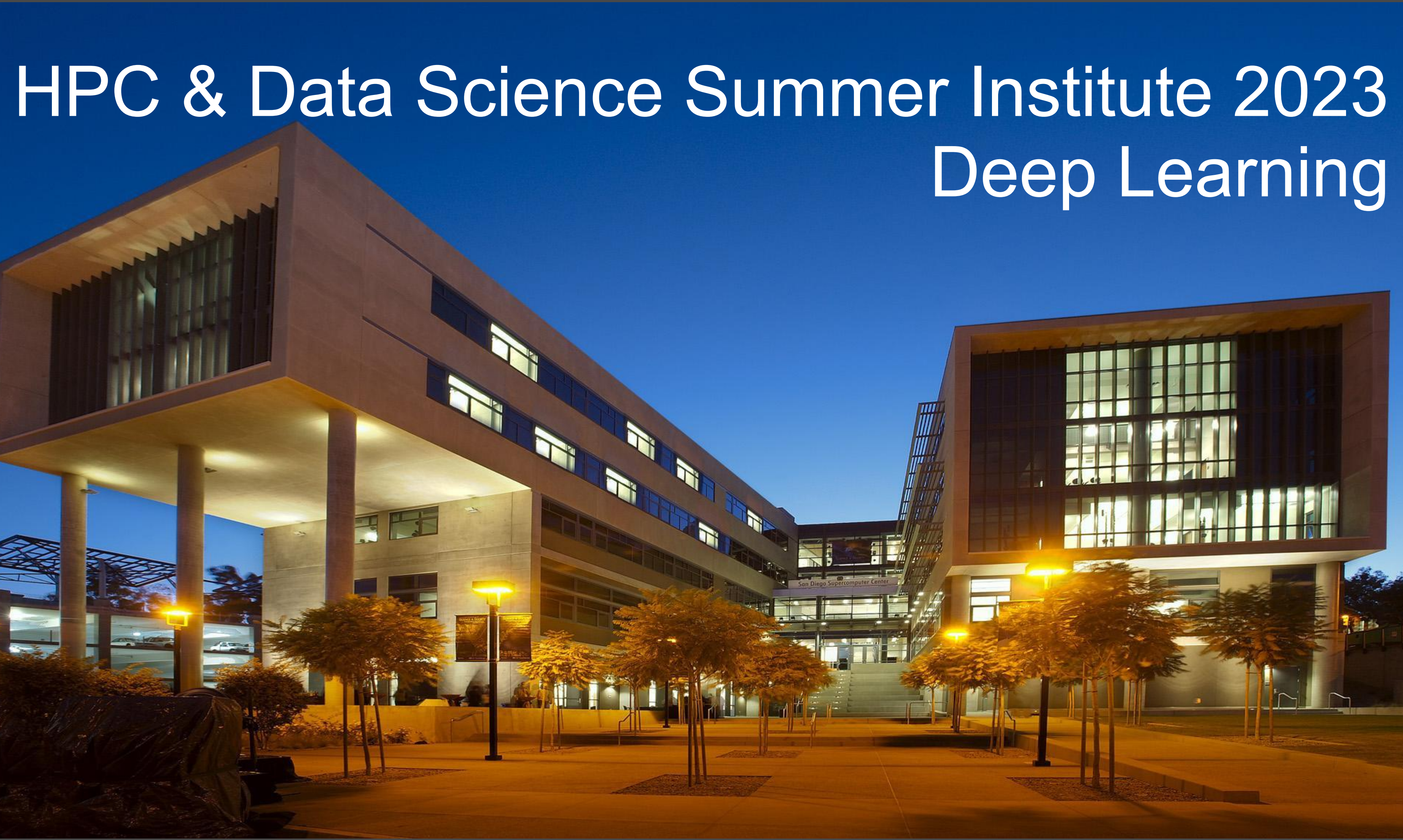
HPC & Data Science Summer Institute 2023
Deep Learning



SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Deep Learning Layers & Architectures

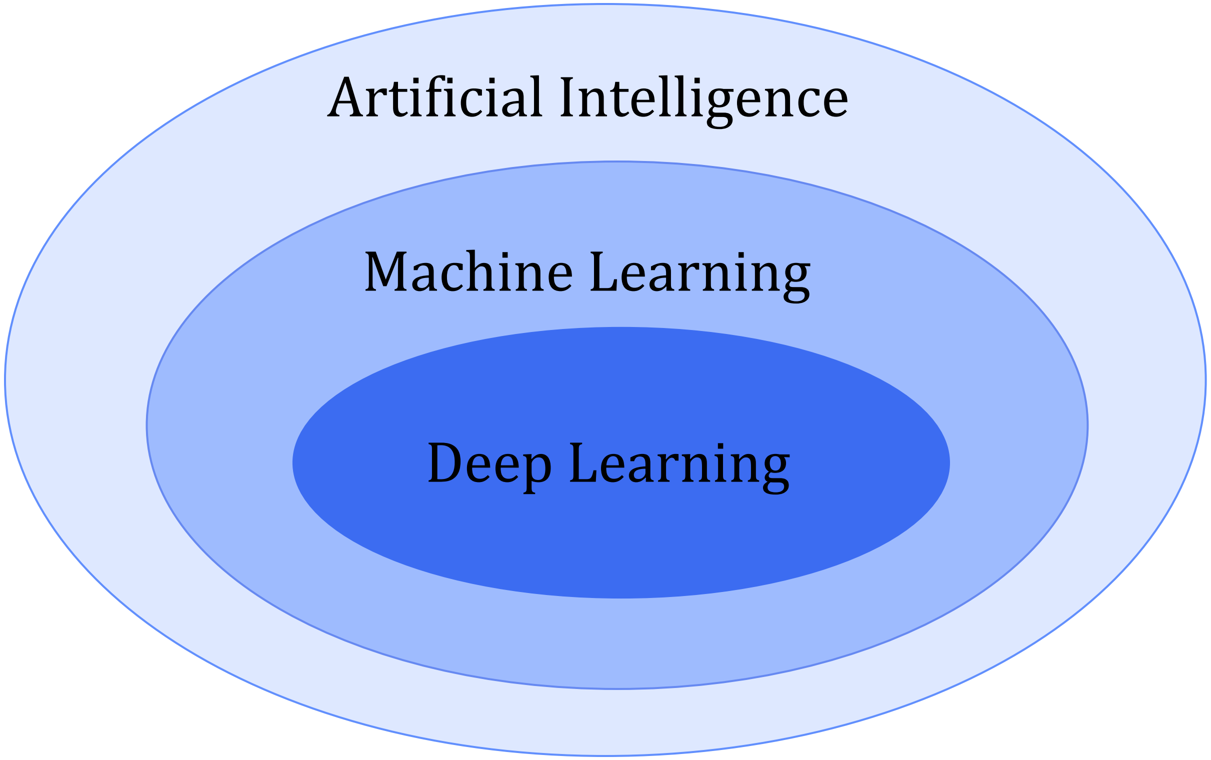
Mai H. Nguyen, Ph.D.

DEEP LEARNING OVERVIEW

- **Neural Network Basics**
  - Processing Unit
  - Activation Function
  - Loss Function
- **Deep Learning Fundamentals**
  - Deep Network Layers
  - DL Architectures
  - DL Libraries
- **Transfer Learning**
  - Transfer Learning Concepts
  - Transfer Learning Demo

DEEP LEARNING

Deep Learning is a subfield of Machine Learning



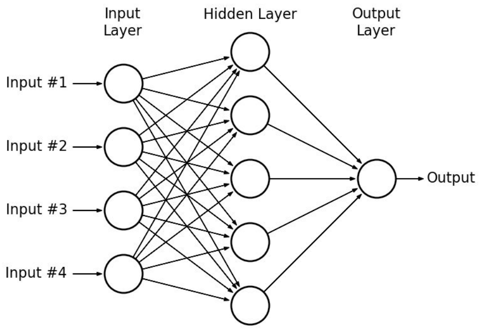
Artificial Intelligence

Machine Learning

Deep Learning

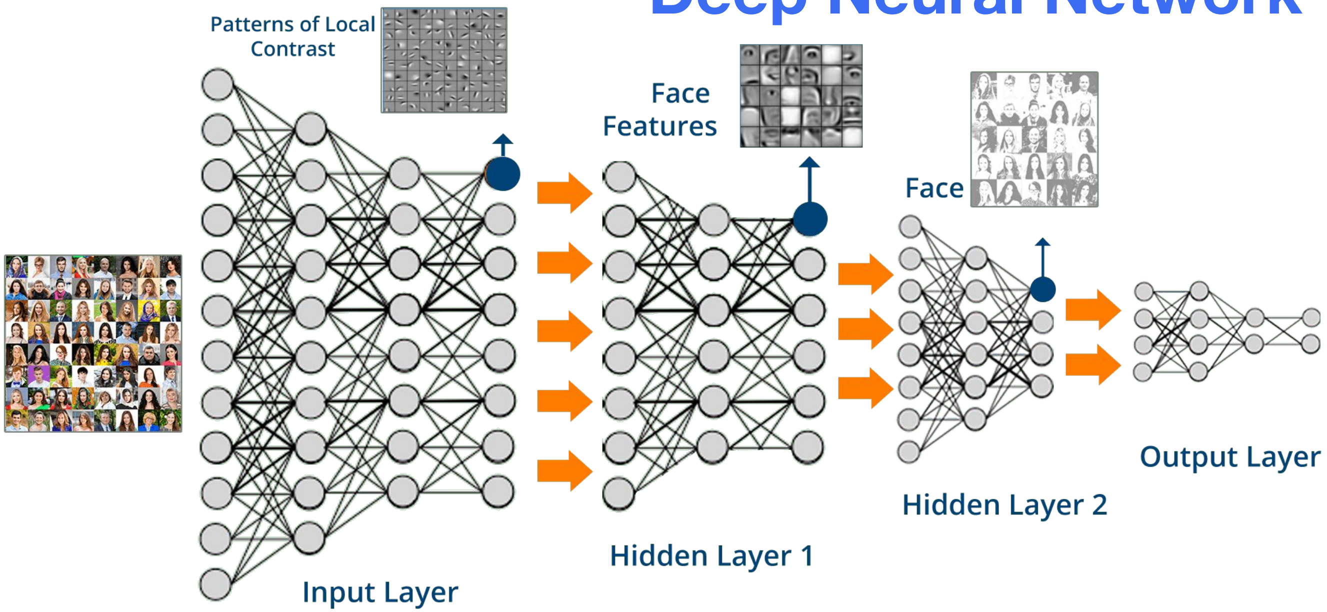
DEEP LEARNING

Neural Network



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

Deep Neural Network



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 & state-of-the-art performance

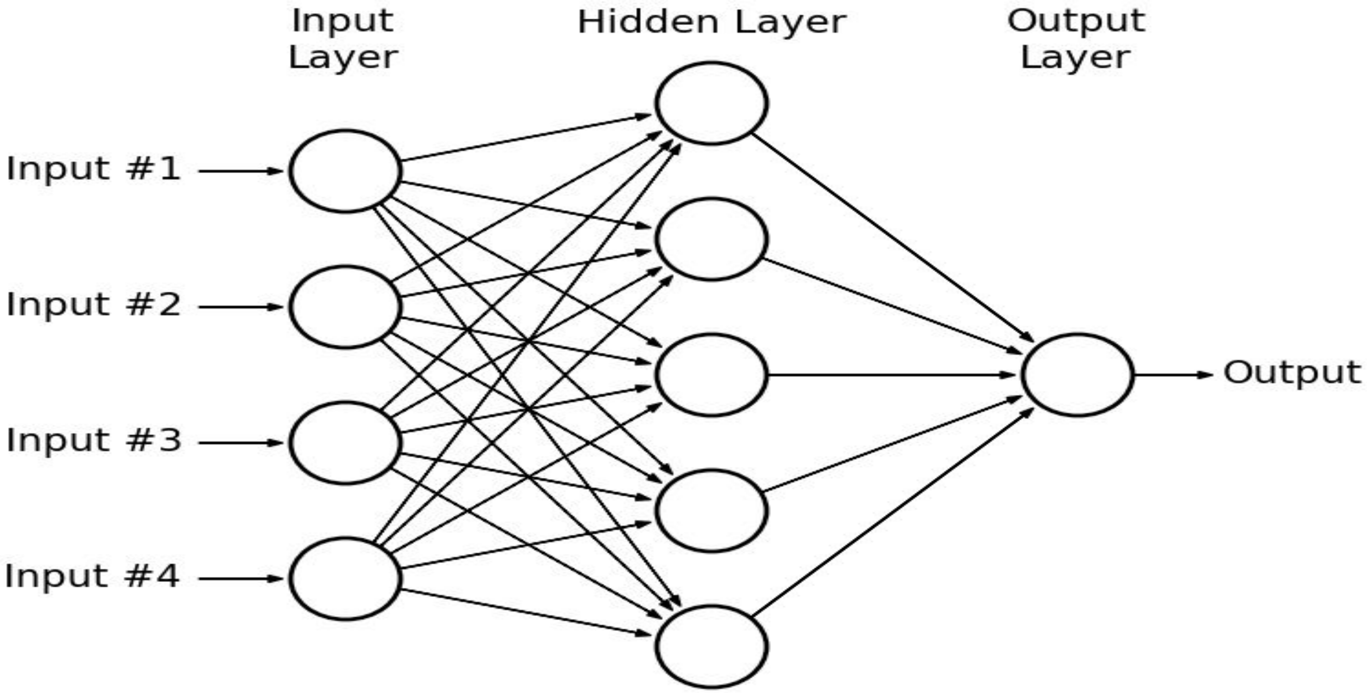
APPLICATIONS OF DEEP LEARNING

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

DEEP LEARNING OVERVIEW

- **Neural Network Basics**
  - Processing Unit
  - Activation Function
  - Loss Function
- **Deep Learning Fundamentals**
  - Deep Network Layers
  - DL Architectures
  - DL Libraries
- **Transfer Learning**
  - Transfer Learning Concepts
  - Transfer Learning Demo

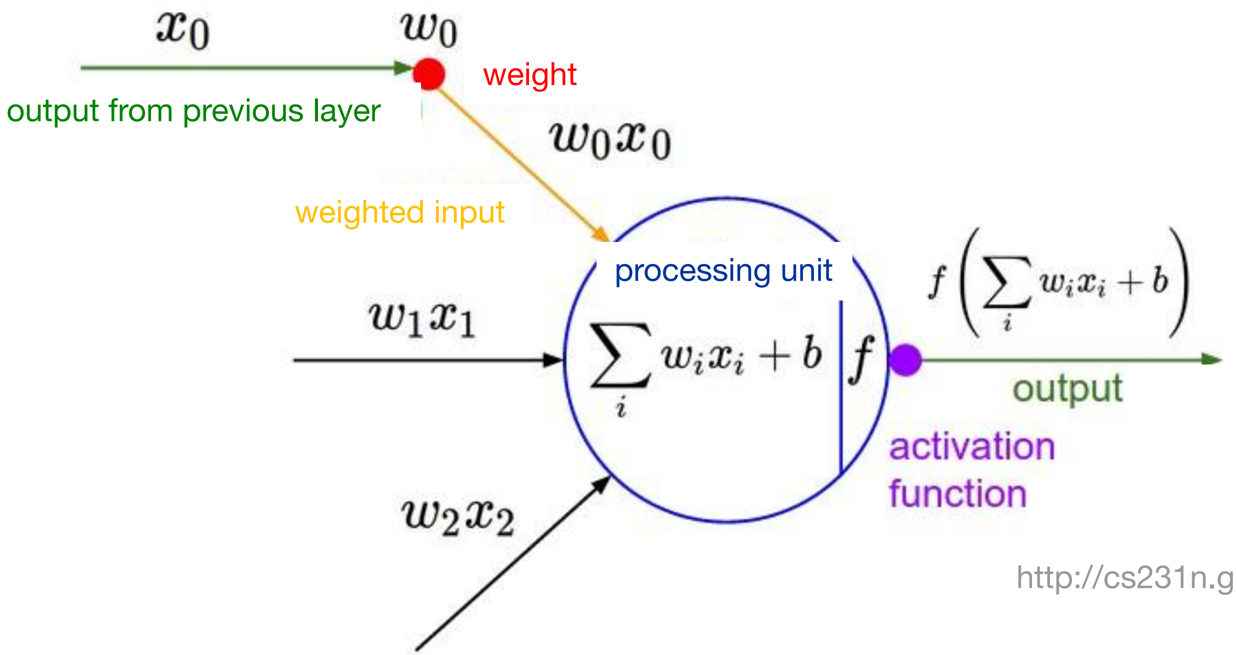
NEURAL NETWORK



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



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

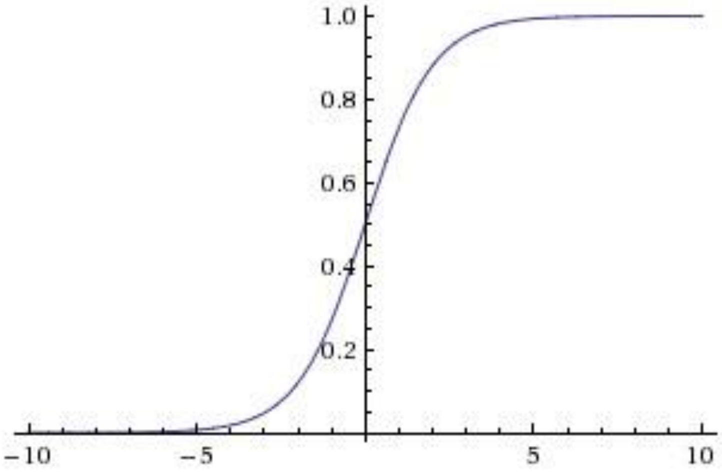
Steps Performed by Each Unit

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

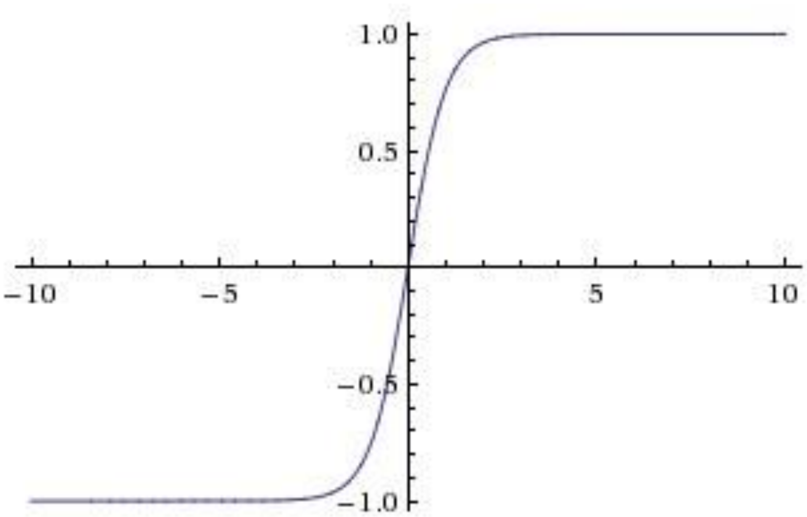
ACTIVATION FUNCTION

- **Converts input of node to output**
  - Implements input-to-output mapping
- **Nonlinear**
  - Necessary to represent complex arbitrary functional mappings between input and output
- **Differentiable**
  - Necessary to propagate loss through model to adjust parameters

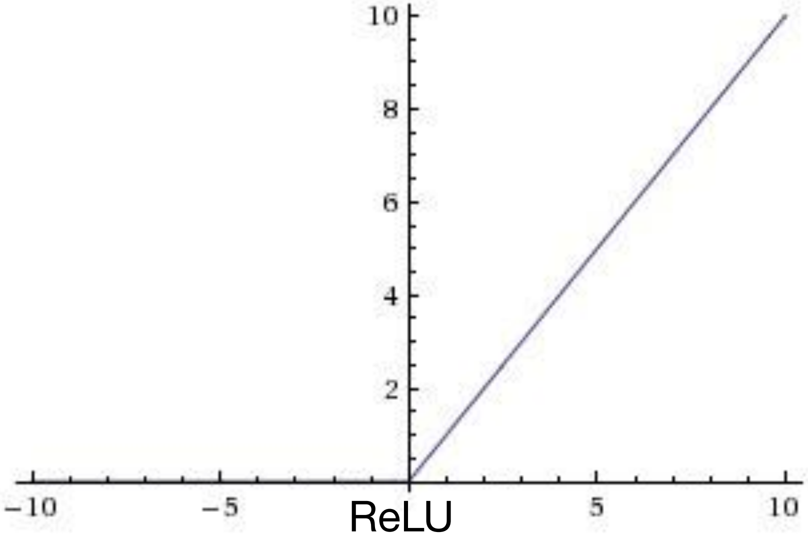
COMMON ACTIVATION FUNCTIONS



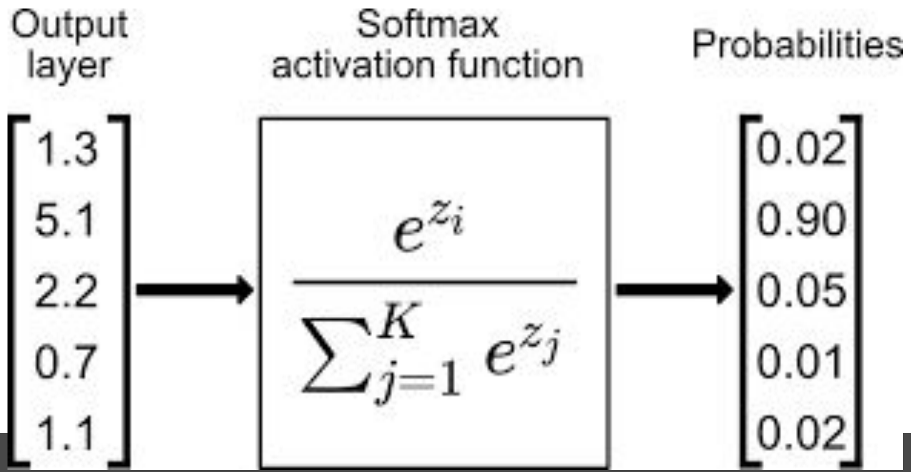
sigmoid



tanh



ReLU



$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

LOSS FUNCTION

- **Allows evaluation of model's performance**
  - Quantifies agreement between model's predictions and targets
  - Provides single scalar value that captures performance of complex model
  - Used to optimize model parameters
- **Other names**
  - cost function
  - objective function

COMMON LOSS FUNCTIONS

- **Cross Entropy (CE)**
  - Average of differences between predicted and target probability distributions for all classes

$$S = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

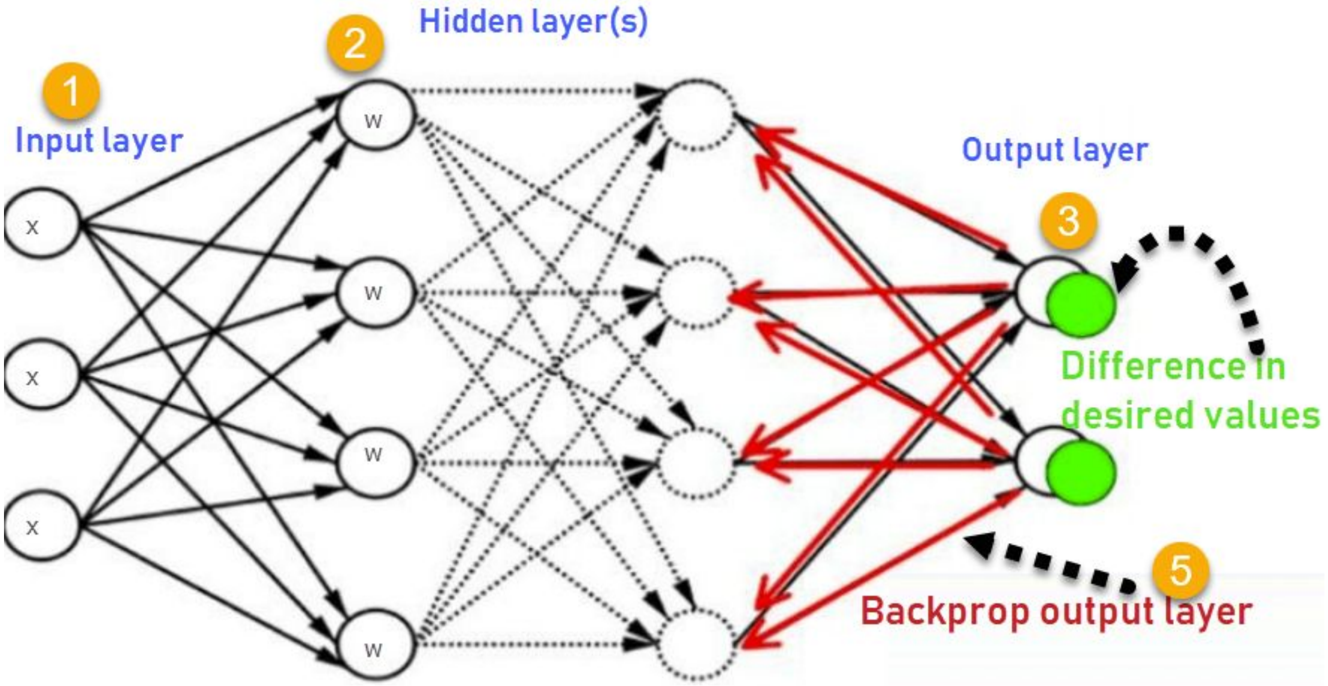
- **Mean Squared Error (MSE)**
  - Average of squared differences between predicted and target values

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- **Mean Absolute Error (MAE)**
  - Average of absolute differences between predicted and target values

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|.$$

NEURAL NETWORK TRAINING



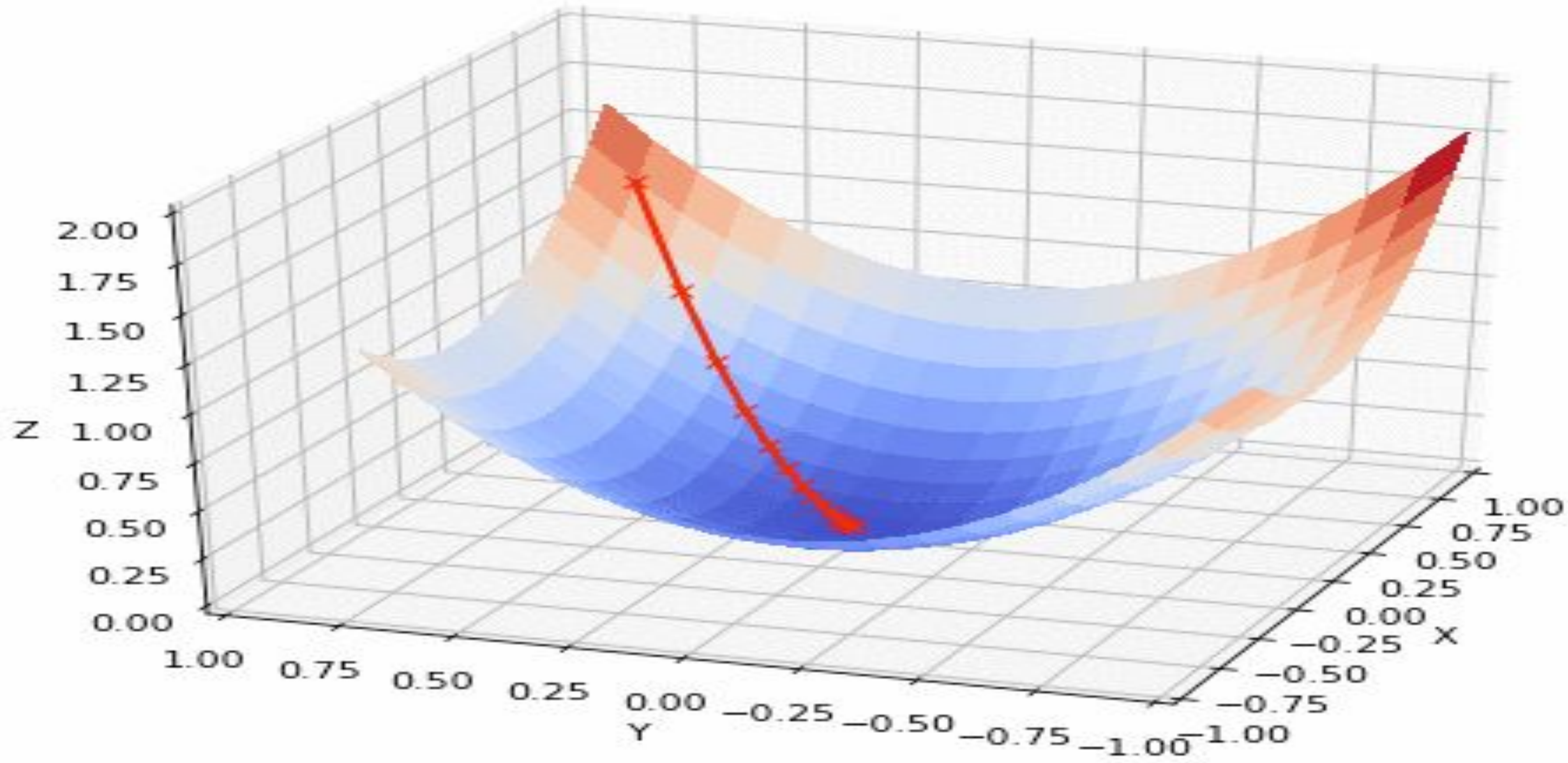
https://www.guru99.com/backpropogation-neural-network.html

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

OPTIMIZATION

- **Training neural network = optimization problem**
  - Adjust weights to change loss function
  - Goal: Find best set of model parameters ("weights") that minimize loss function
- **Challenge**
  - Finding best set of weights difficult/impossible
- **Approach**
  - Iterative refinement
  - Iteratively adjust model's weights to lower loss
  - This is what happens during training of neural networks

GRADIENT DESCENT



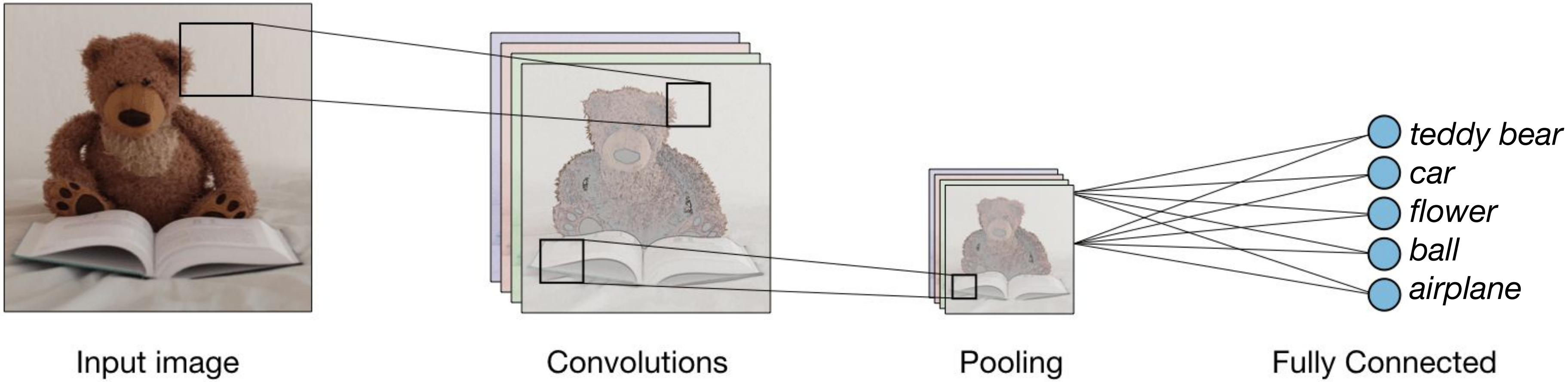
https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

DEEP LEARNING OVERVIEW

- **Neural Network Basics**
  - Processing Unit
  - Activation Function
  - Loss Function
- **Deep Learning Fundamentals**
  - Deep Network Layers
  - DL Architectures
  - DL Libraries
- **Transfer Learning**
  - Transfer Learning Concepts
  - Transfer Learning Demo

DEEP LEARNING MODELS

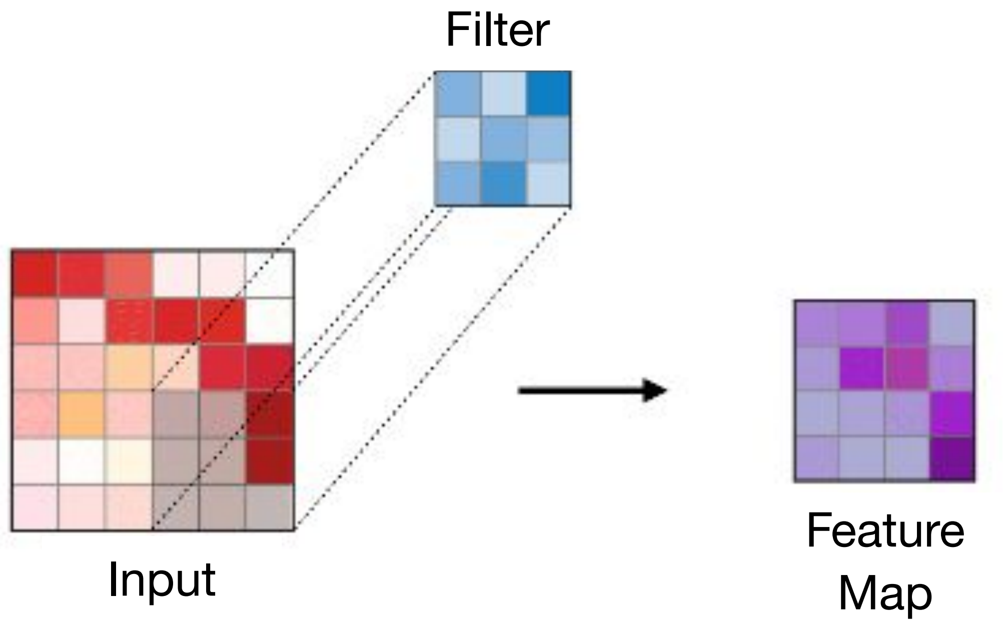
- **General Deep Network Architecture:**
  - Has sequence of layers
  - Each layer transforms its input to generate an output through nonlinear function
  - Has different types of layers



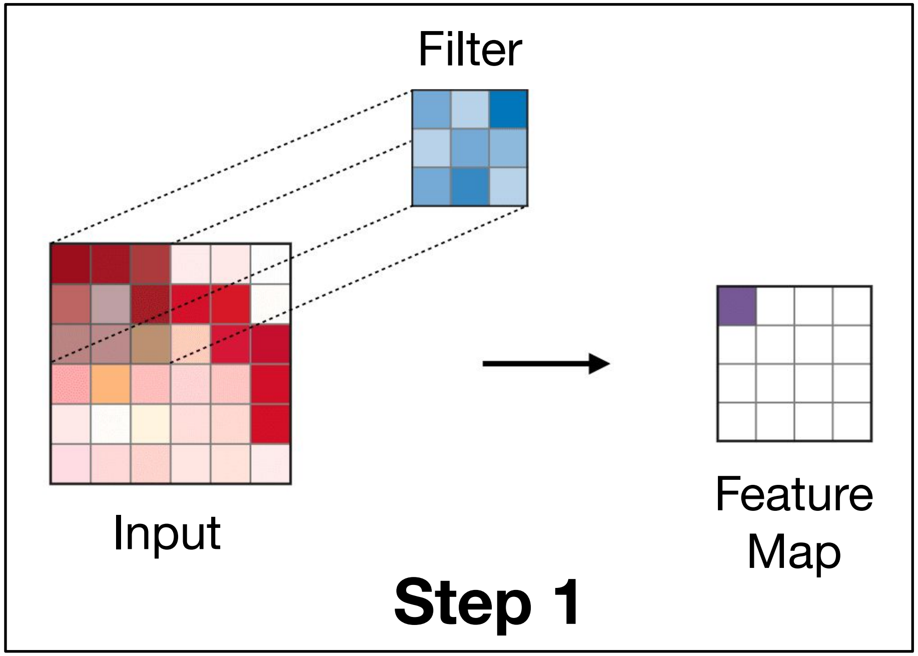
https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer

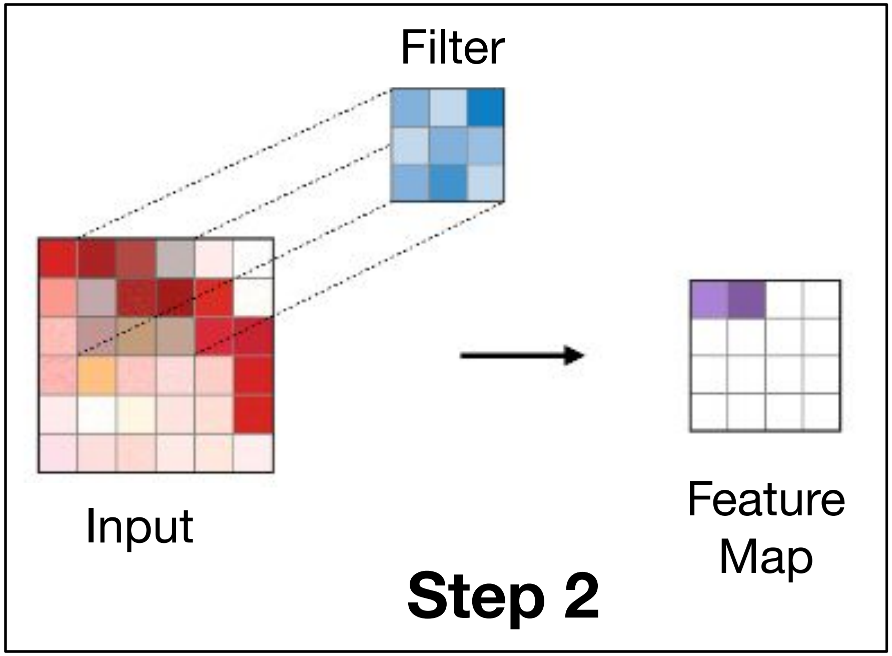
CONVOLUTION LAYER

- 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



CONVOLUTION FILTER





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

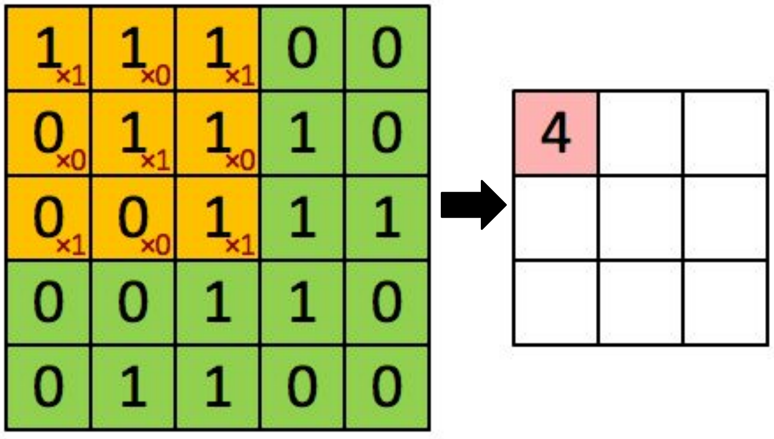
- Filter size = receptive field of filter
- Stride = sliding amount, i.e., # pixels by which filter is moved over image
- Padding = padding around input volume
- Depth = number of filters

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

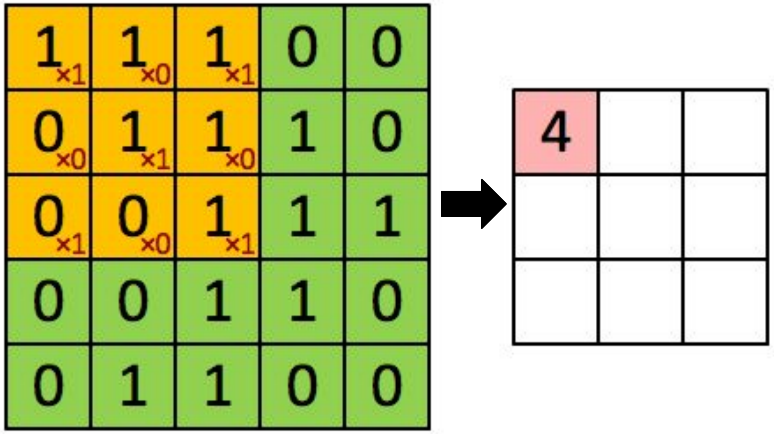
http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/

CONVOLUTION OPERATION

1	0	1
0	1	0
1	0	1

3 x 3 Filter

Steps



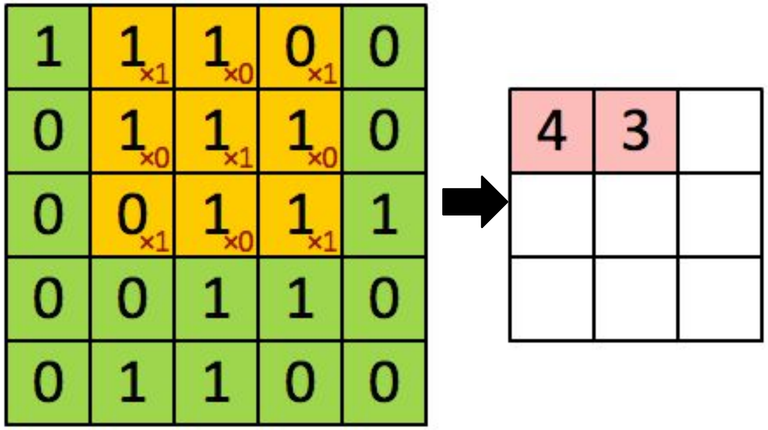
1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Input

Feature Map

Step 2



1	1 ×1	1 ×0	0 ×1	0
0	1 ×0	1 ×1	1 ×0	0
0	0 ×1	1 ×0	1 ×1	1
0	0	1	1	0
0	1	1	0	0

4	3	

Input

Feature Map

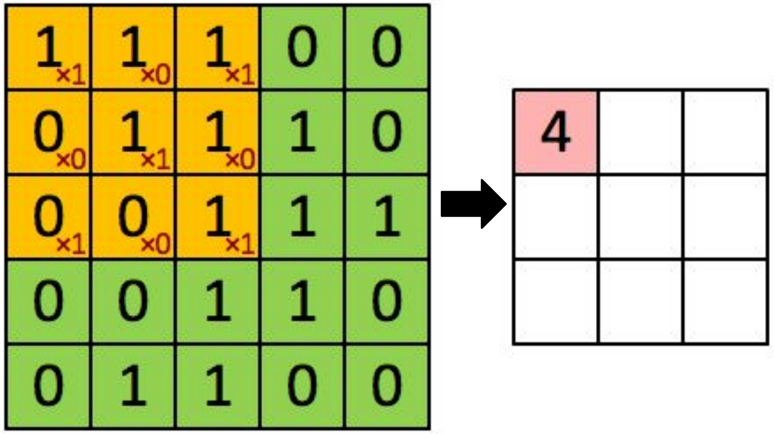
http://ufldl.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/

CONVOLUTION OPERATION

1	0	1
0	1	0
1	0	1

3 x 3 Filter

Steps



$1_{\times1}$	$1_{\times0}$	$1_{\times1}$	0	0
$0_{\times0}$	$1_{\times1}$	$1_{\times0}$	1	0
$0_{\times1}$	$0_{\times0}$	$1_{\times1}$	1	1
0	0	1	1	0
0	1	1	0	0

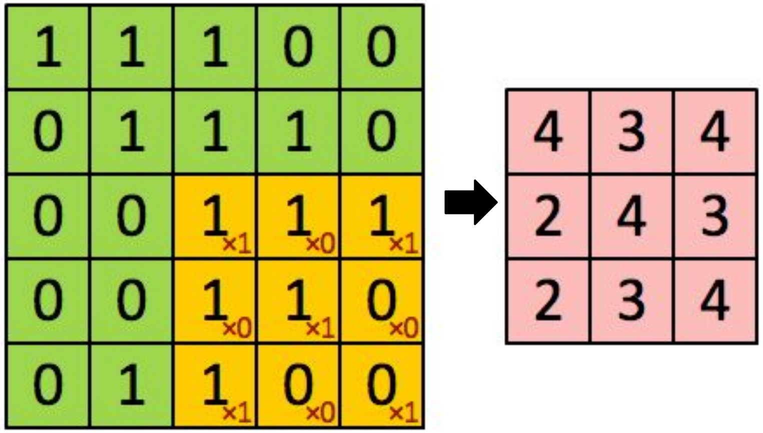
4		

Input

Feature Map

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

Step 9



1	1	1	0	0
0	1	1	1	0
0	0	$1_{\times1}$	$1_{\times0}$	$1_{\times1}$
0	0	$1_{\times0}$	$1_{\times1}$	$0_{\times0}$
0	1	$1_{\times1}$	$0_{\times0}$	$0_{\times1}$

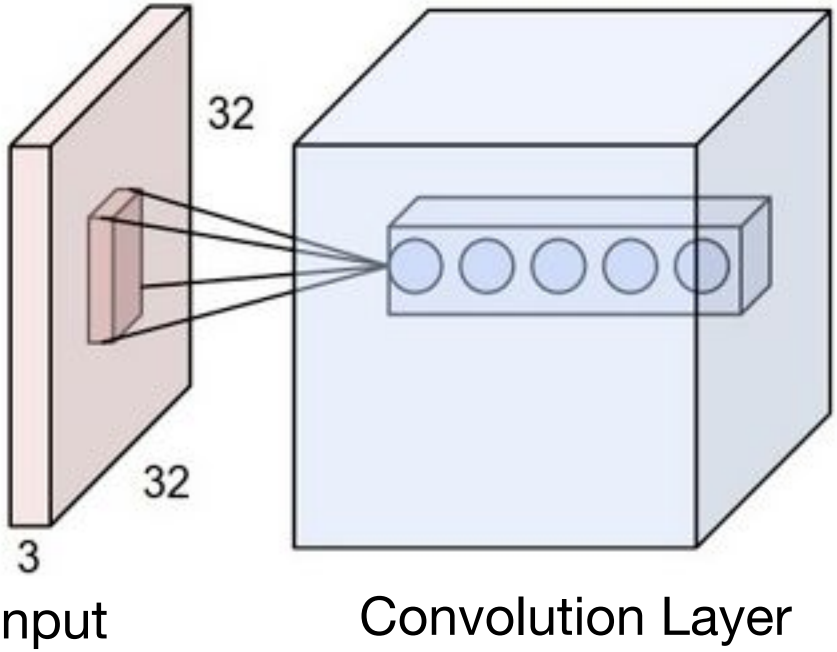
4	3	4
2	4	3
2	3	4

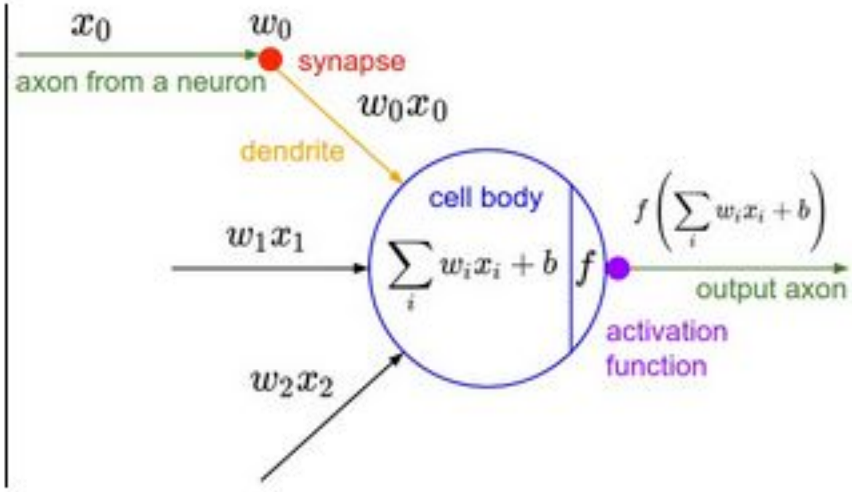
Input

Feature Map

CONVOLUTION LAYER

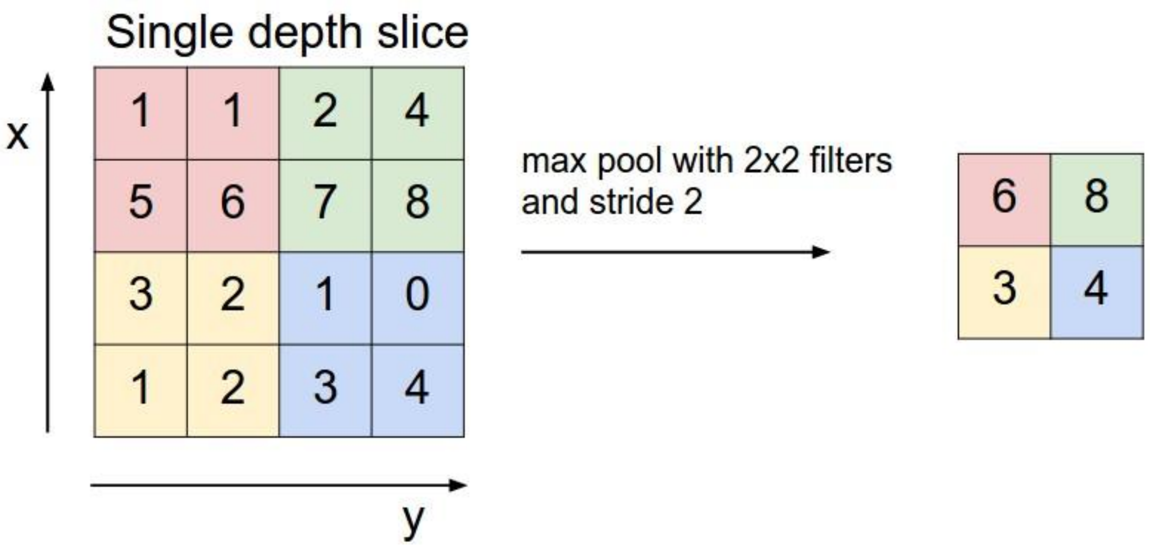
- Performs convolution on input volume (height X width X channels) with filters
- Each filter in convolution layer is connected to local region in input
- Result of convolution is passed through nonlinear activation function
- Depth = number of filters = number of feature maps in convolution layer





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

POOLING LAYER



Single depth slice

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

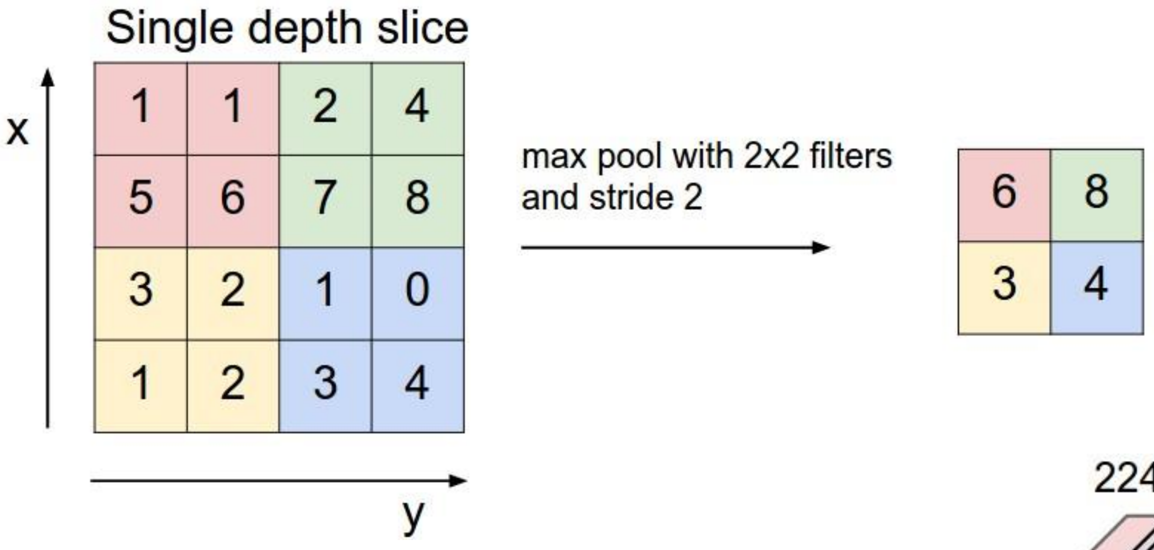
max pool with 2x2 filters and stride 2

6	8
3	4

Pooling reduces spatial size of input

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

POOLING LAYER



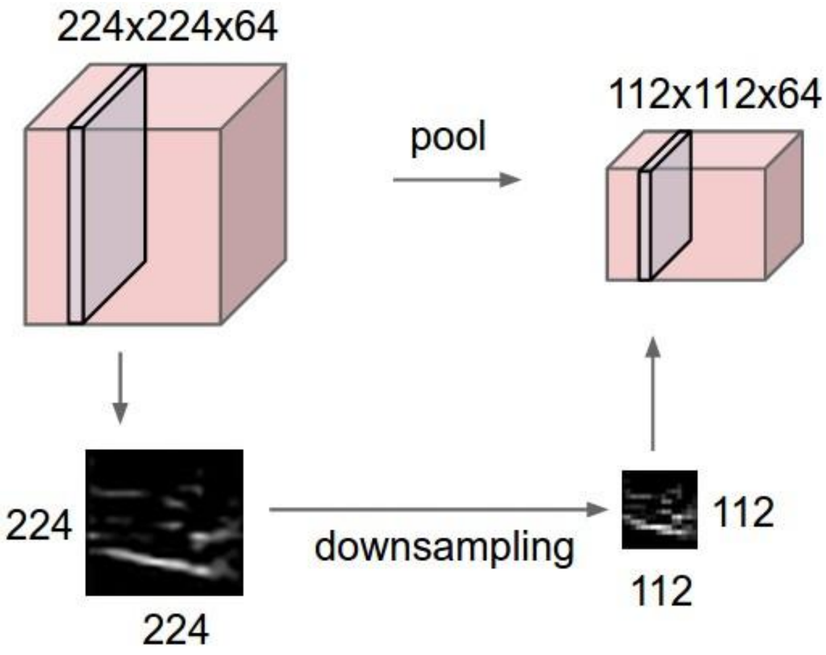
Single depth slice

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

max pool with 2x2 filters and stride 2

6	8
3	4

Pooling reduces spatial size of input

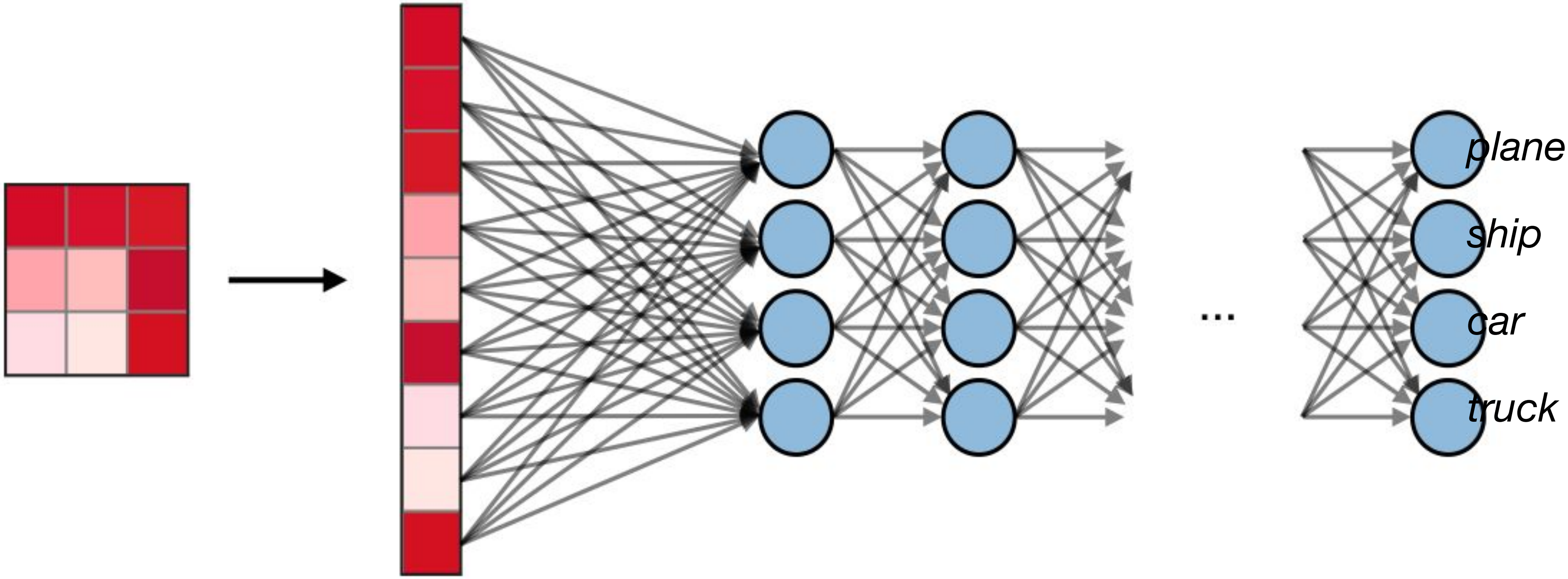


Pooling is performed independently on every slice of input

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

FULLY CONNECTED LAYER

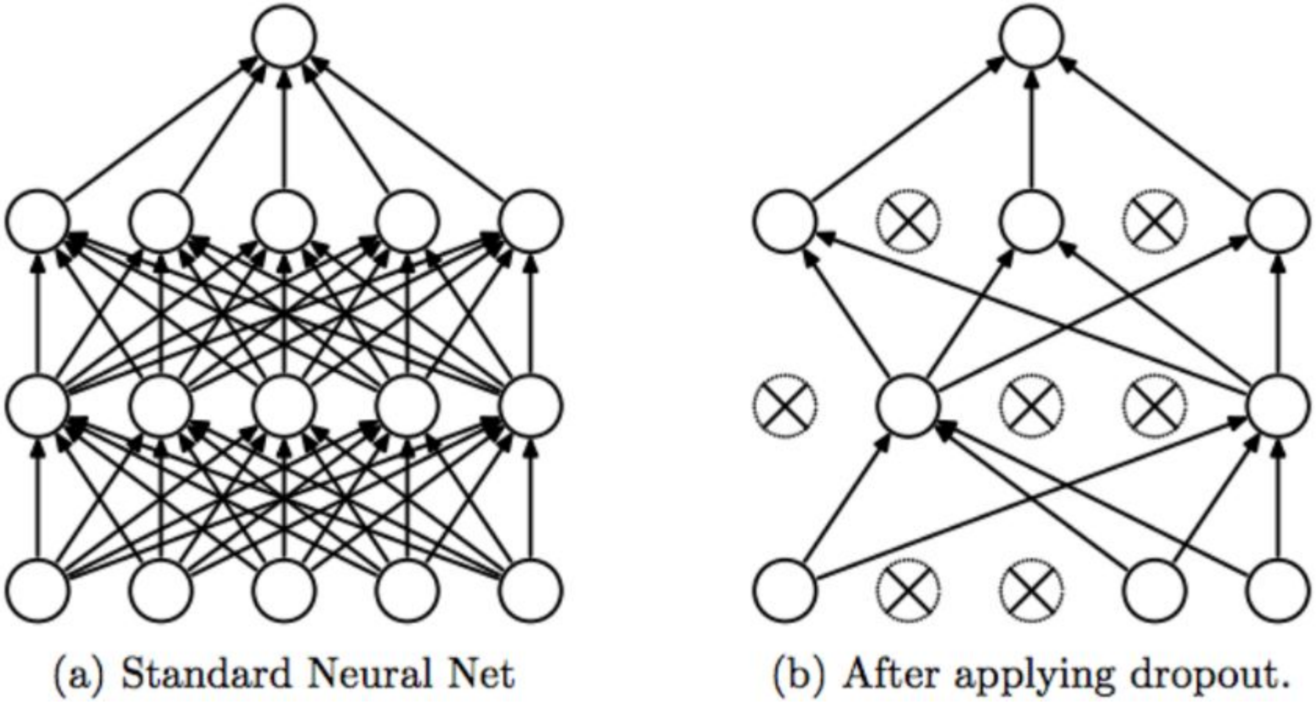
- Fully connected (FC) layer takes flattened input.
- Every input is connected to all processing units.
- Output of FC layer is typically vector with probabilities for categories.



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.

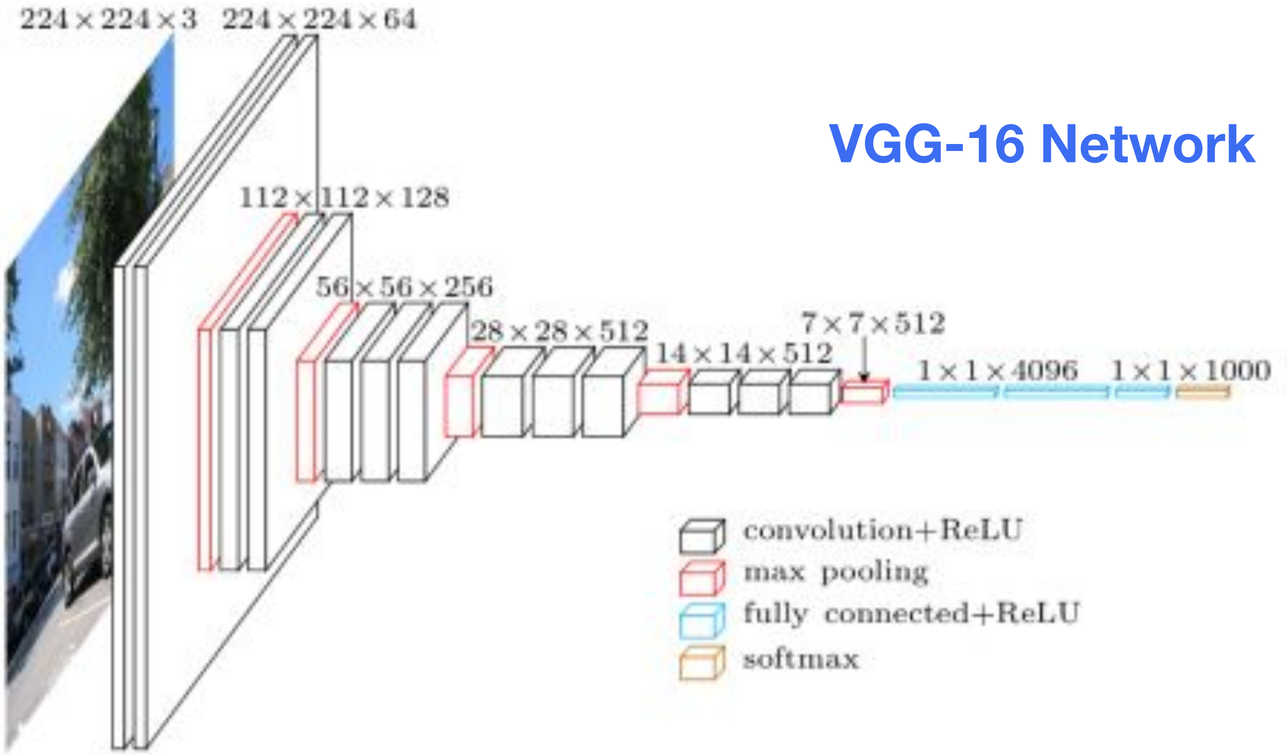
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

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

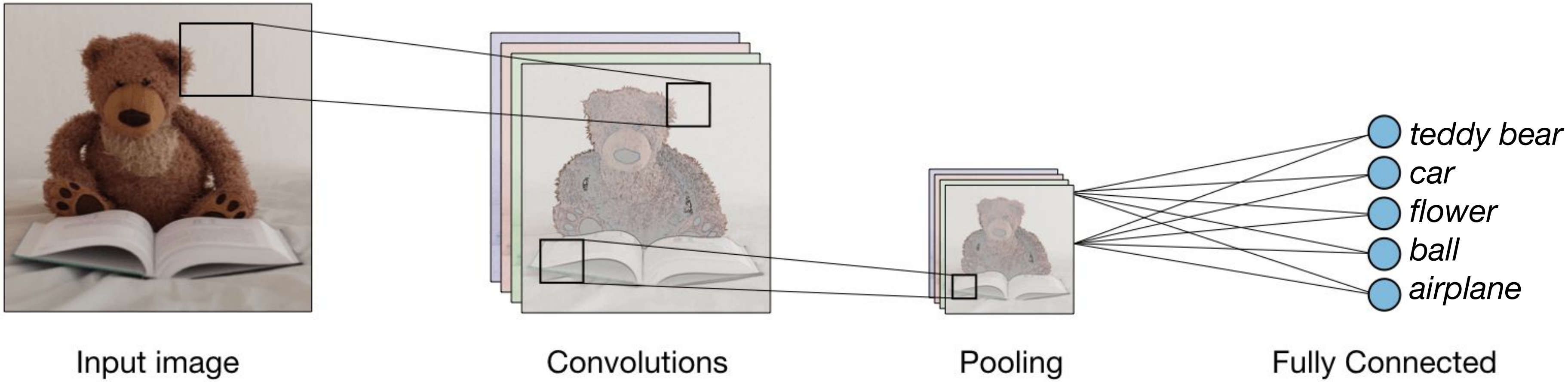


VGG-16 Network

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



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

CNN Models

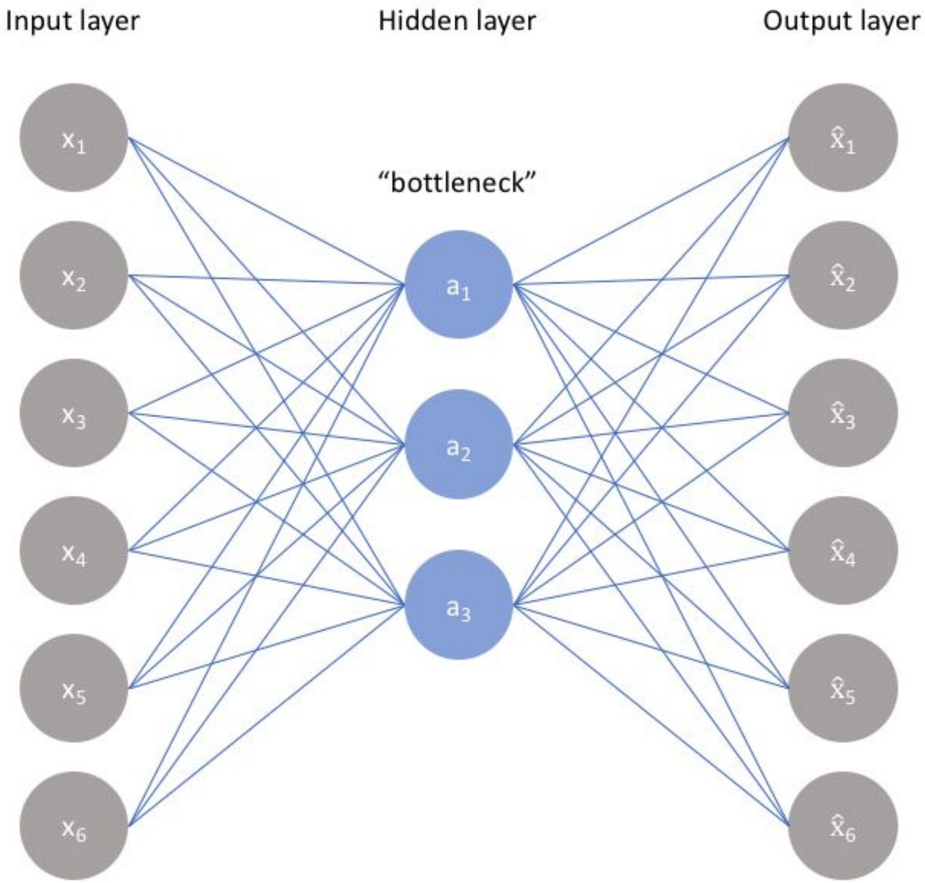
- LeNet
- AlexNet
- VGG
- Inception
- ResNet
- XceptionNet
- Inception-ResNet
- …

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

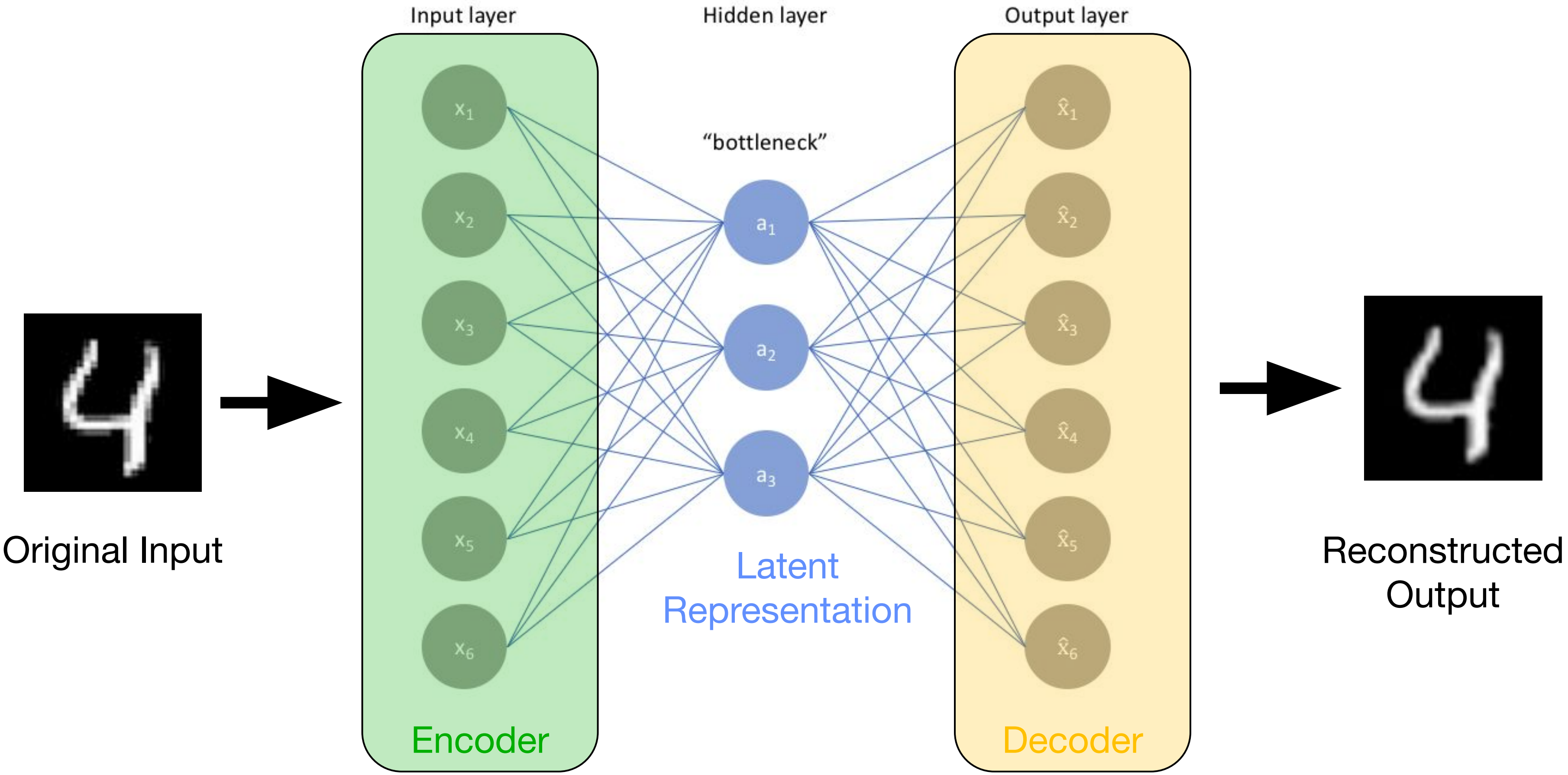
AUTOENCODER

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



https://www.jeremyjordan.me/autoencoders/

AUTOENCODER



https://www.jeremyjordan.me/autoencoders/

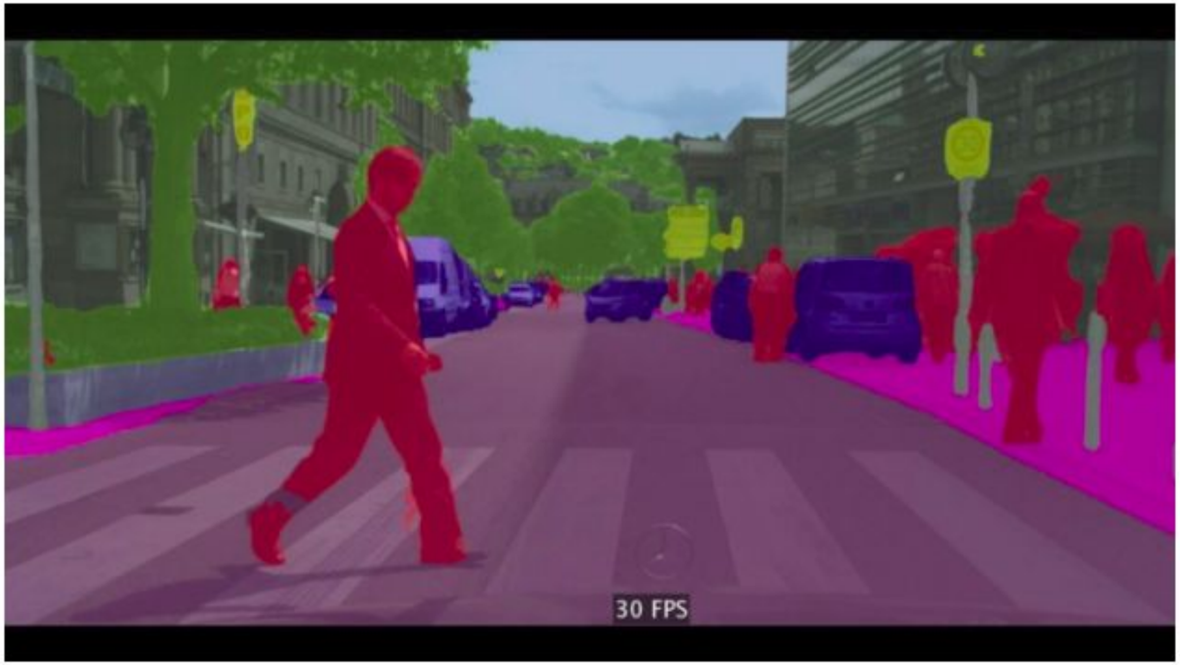
- ”Bottleneck” layer provides encoding of input
- Used to generate latent representation of data

AUTOENCODER

- **Uses**
  - Feature Learning
    - Generated features useful for downstream tasks (e.g., classification, anomaly detection, clustering)
  - Data Compression
    - Learn lower-dimensional representation of data
  - Denoising
    - Remove noise to reconstruct data from noisy inputs
  - Anomaly Detection
    - Detect samples that deviate from canonical representation
- **Variations**
  - Denoising
  - Sparse
  - Contractive
  - Variational

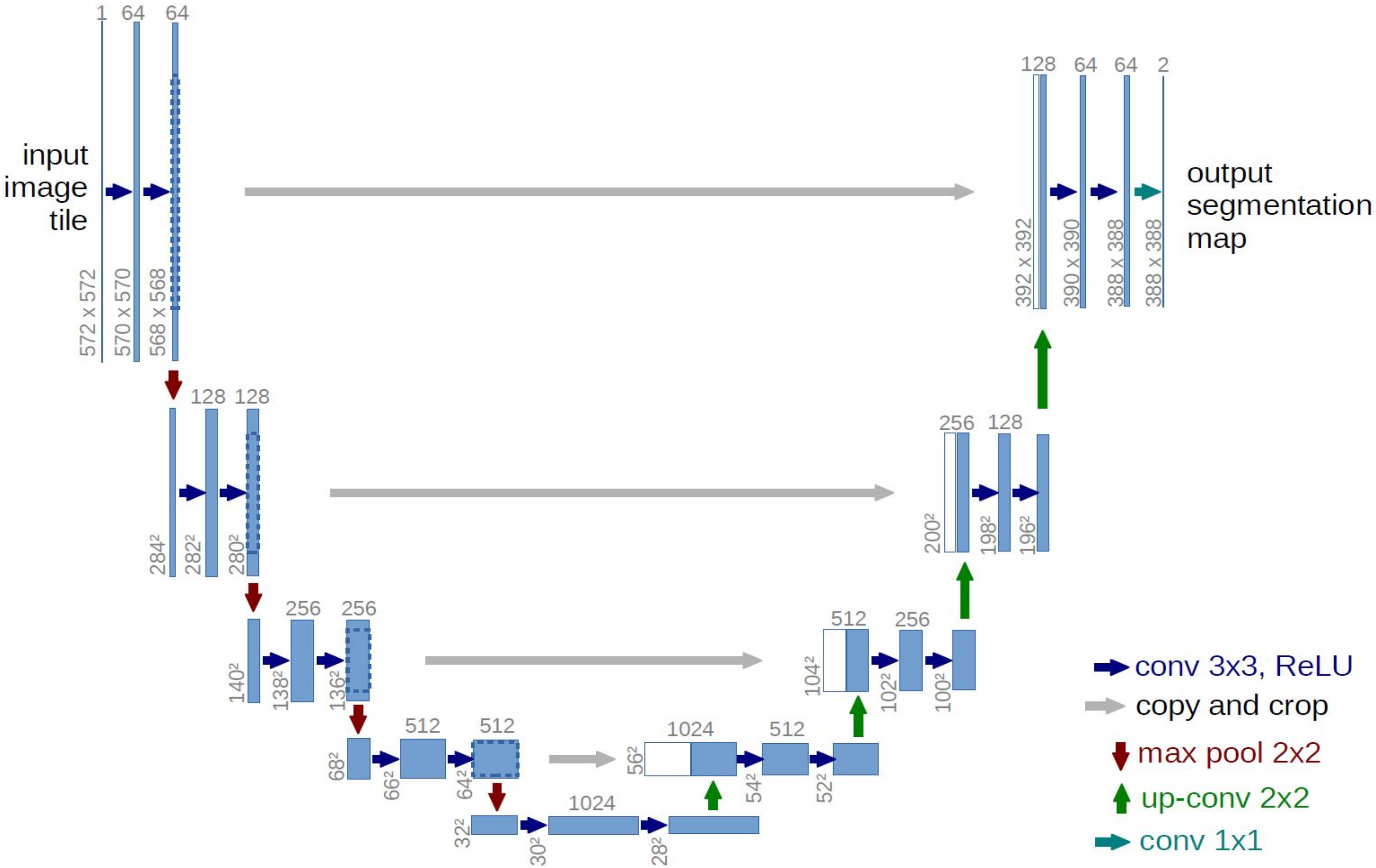
U-NET

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



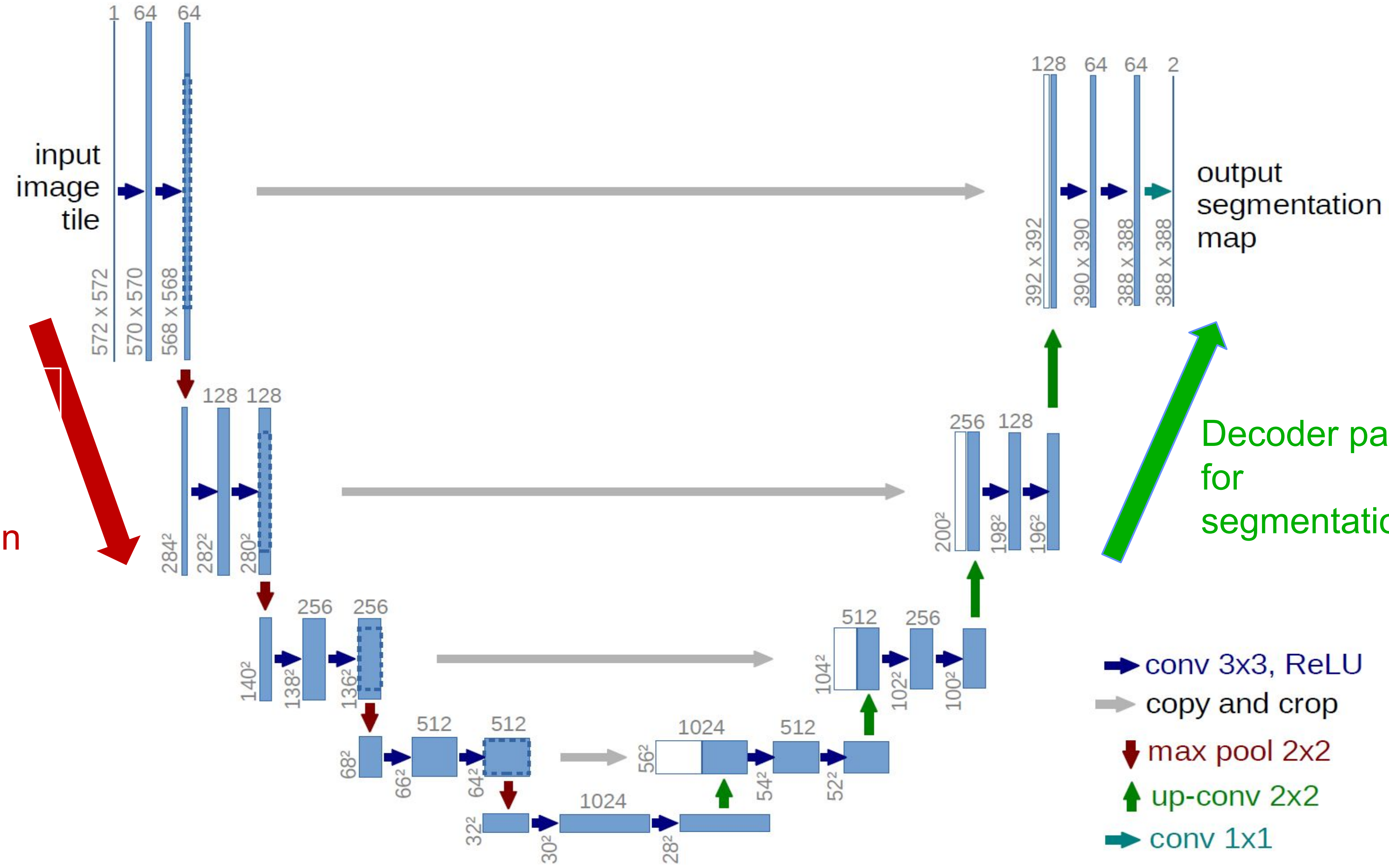
https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066

U-NET ARCHITECTURE



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

U-NET ARCHITECTURE



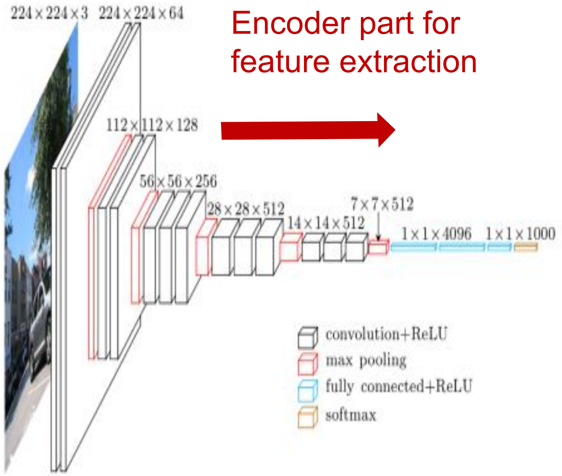
Encoder part for feature extraction

Decoder part for segmentation

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

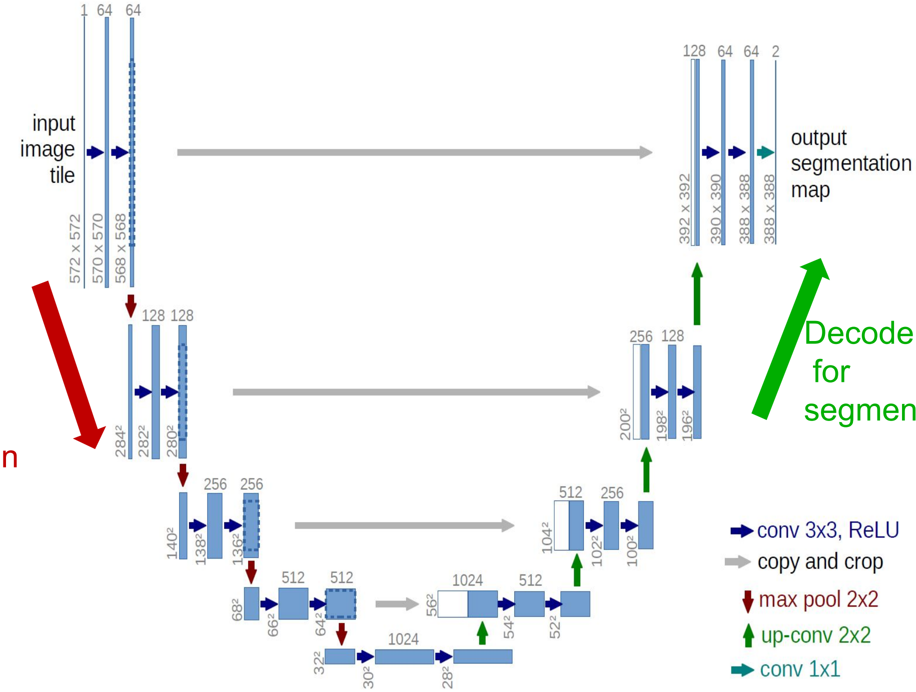
U-NET ARCHITECTURE

VGG16 CNN Architecture



Encoder part for feature extraction

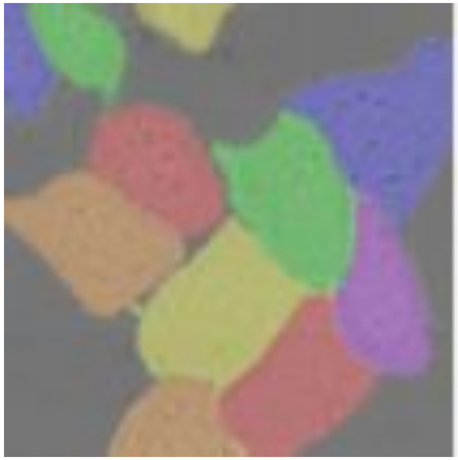
U-Net Architecture



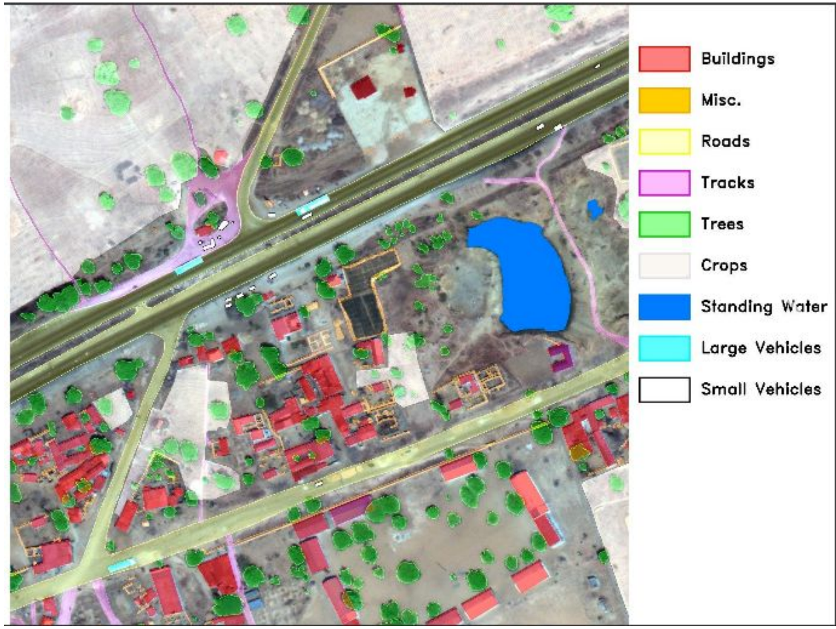
Encoder part for feature extraction

Decoder part for segmentation

U-NET APPLICATIONS

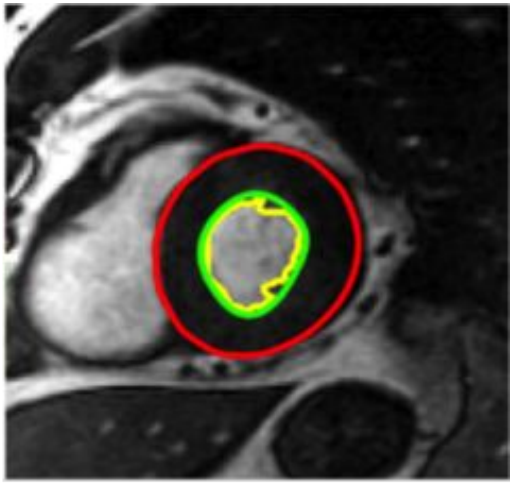


Biomedical Segmentation



Satellite Image Processing

Object Detection



Medical Image Analysis

LSTM

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

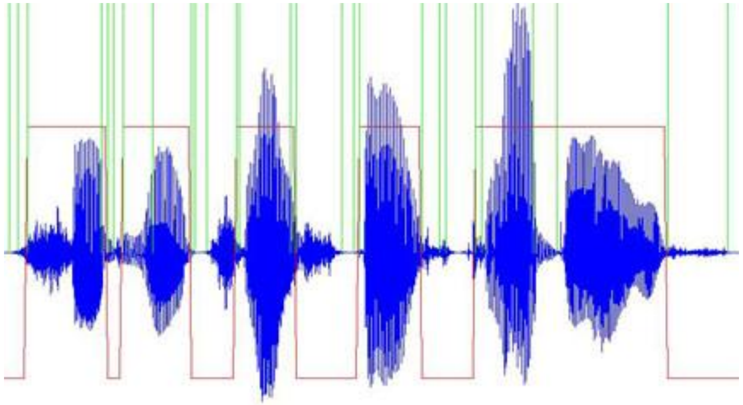
LSTM

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

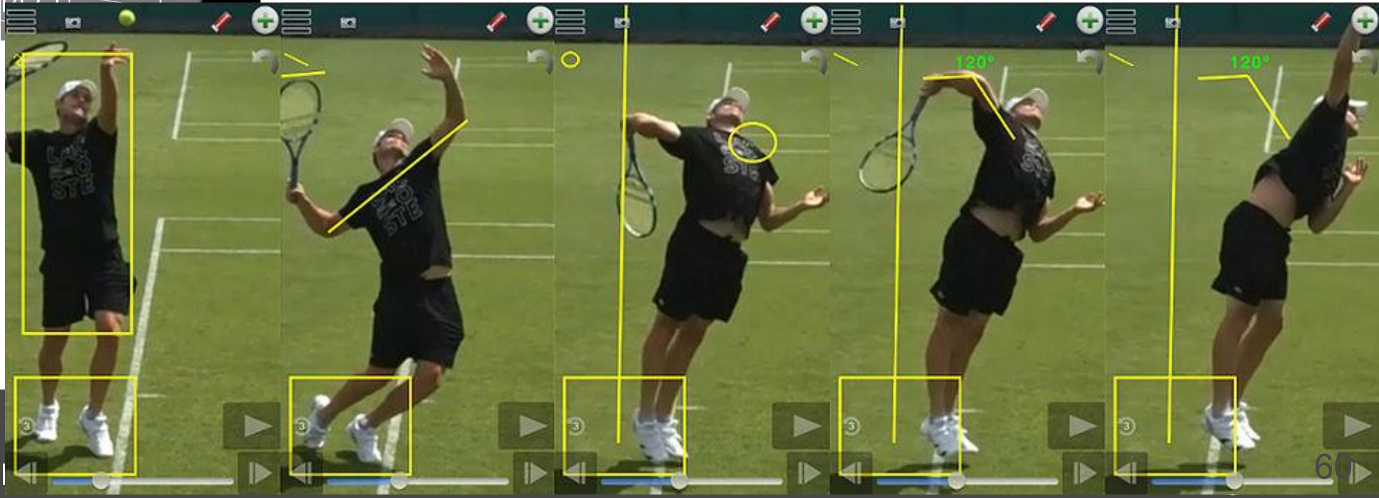
197.3
194.5
192.2
205.1
202.1
200.5
236.5
230.2
130.8
132.8
134.8
140.8

Stock Price

Speech

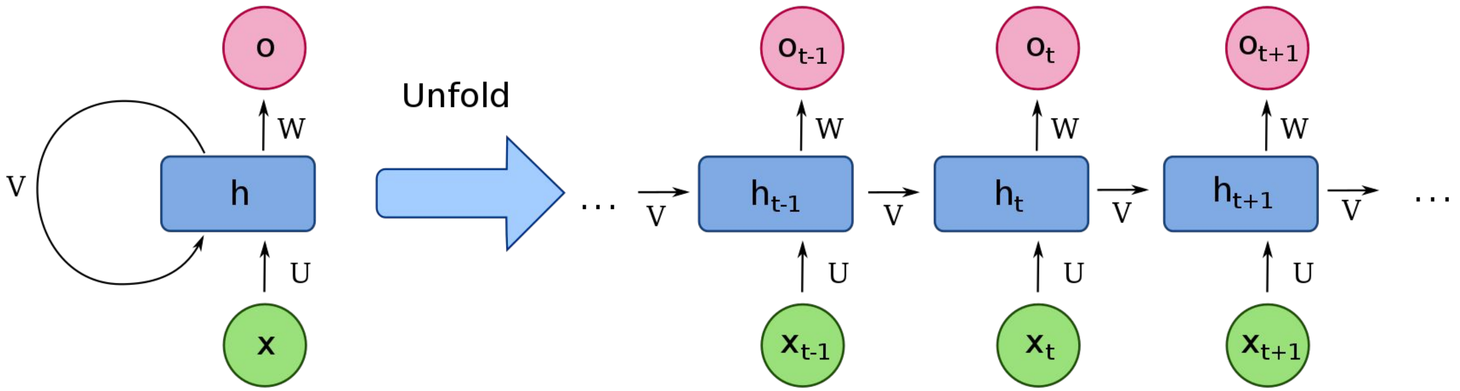


Video

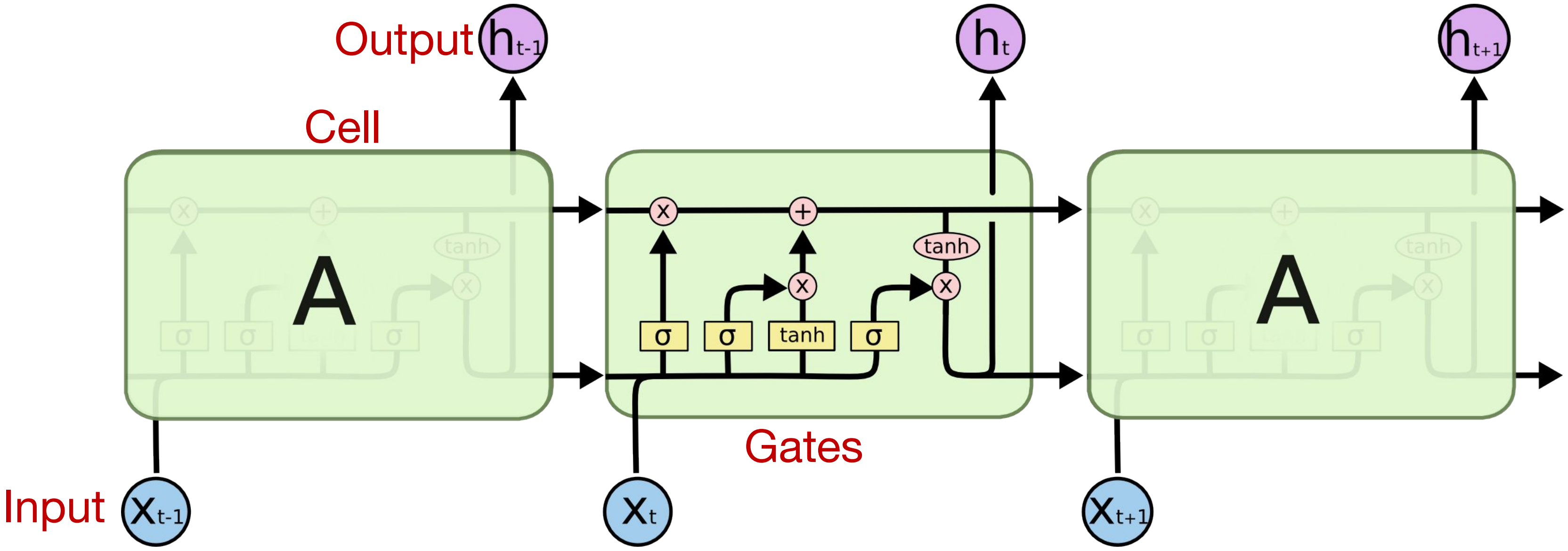


RECURRENT NEURAL NETWORK (RNN)

- **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/

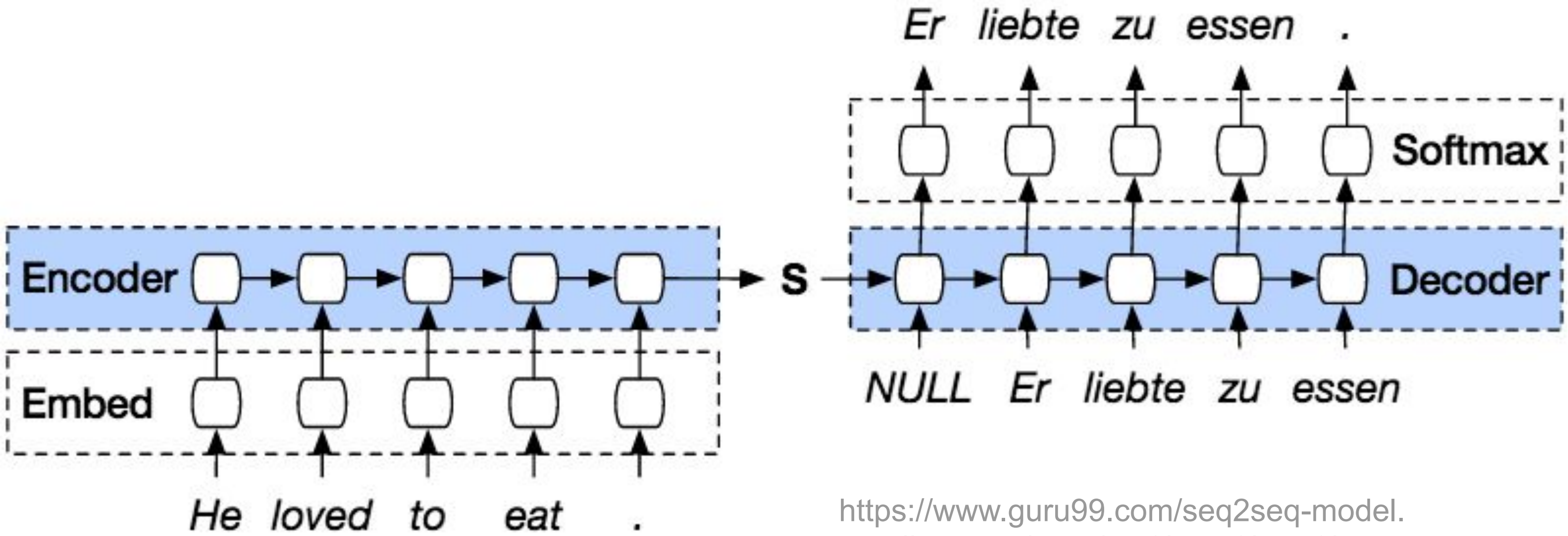
- Info flows through memory blocks called ‘cells’
- Structure of cell allows LSTM to selectively remember/forget pieces of information
- Each cell manipulates memory through ‘gates’

LSTM Applications

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

SEQ2SEQ

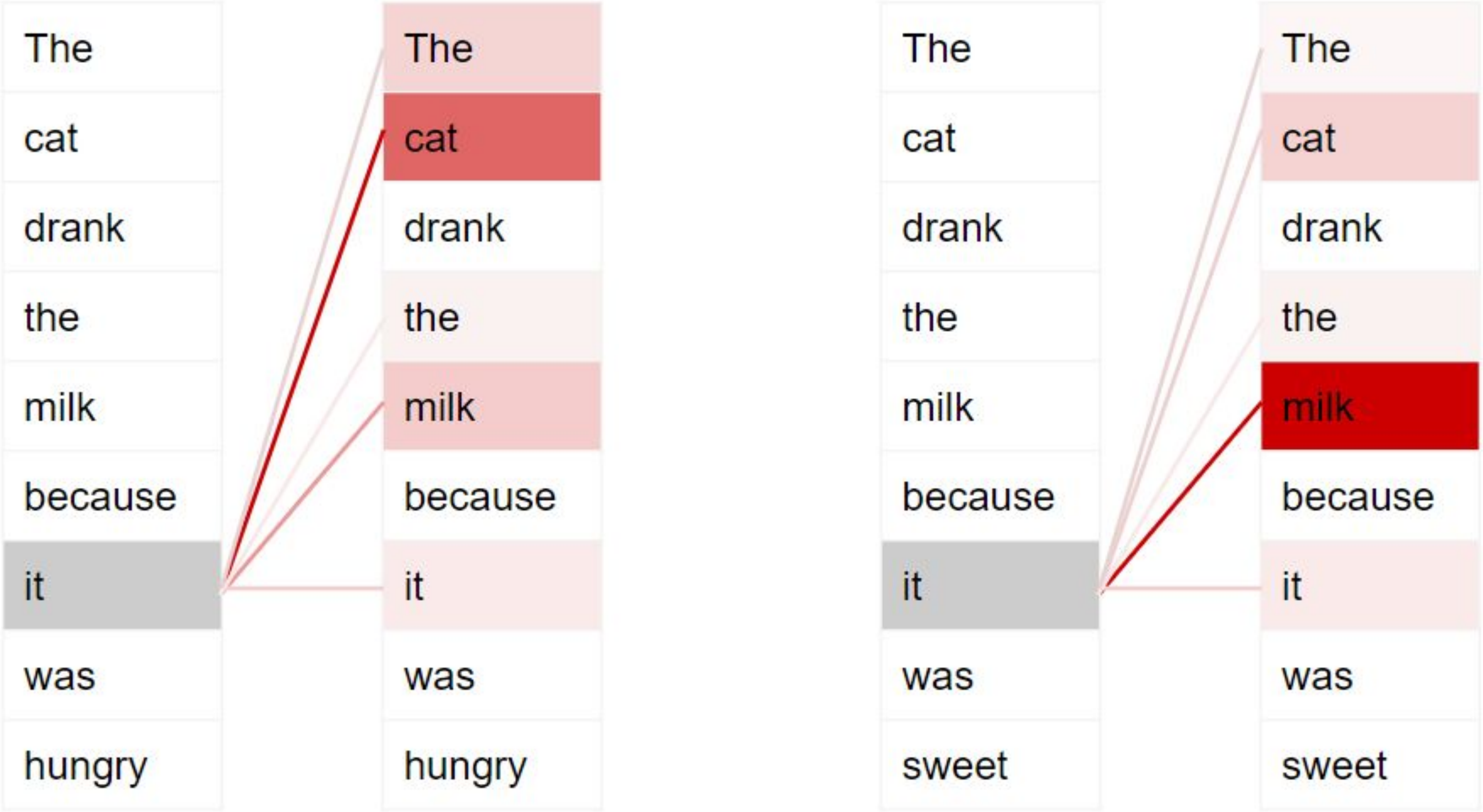
- Converts input sequence to output sequence
  - machine translation, question-answering
- Encoder & decoder are RNNs
- Issue: Difficult to capture long-range dependencies



https://www.guru99.com/seq2seq-model.html#:~:text=Seq2Seq%20is%20a%20method%20of,sequence%20from%20the%20previous%20sequence.

ATTENTION MECHANISM

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



https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

TRANSFORMER

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

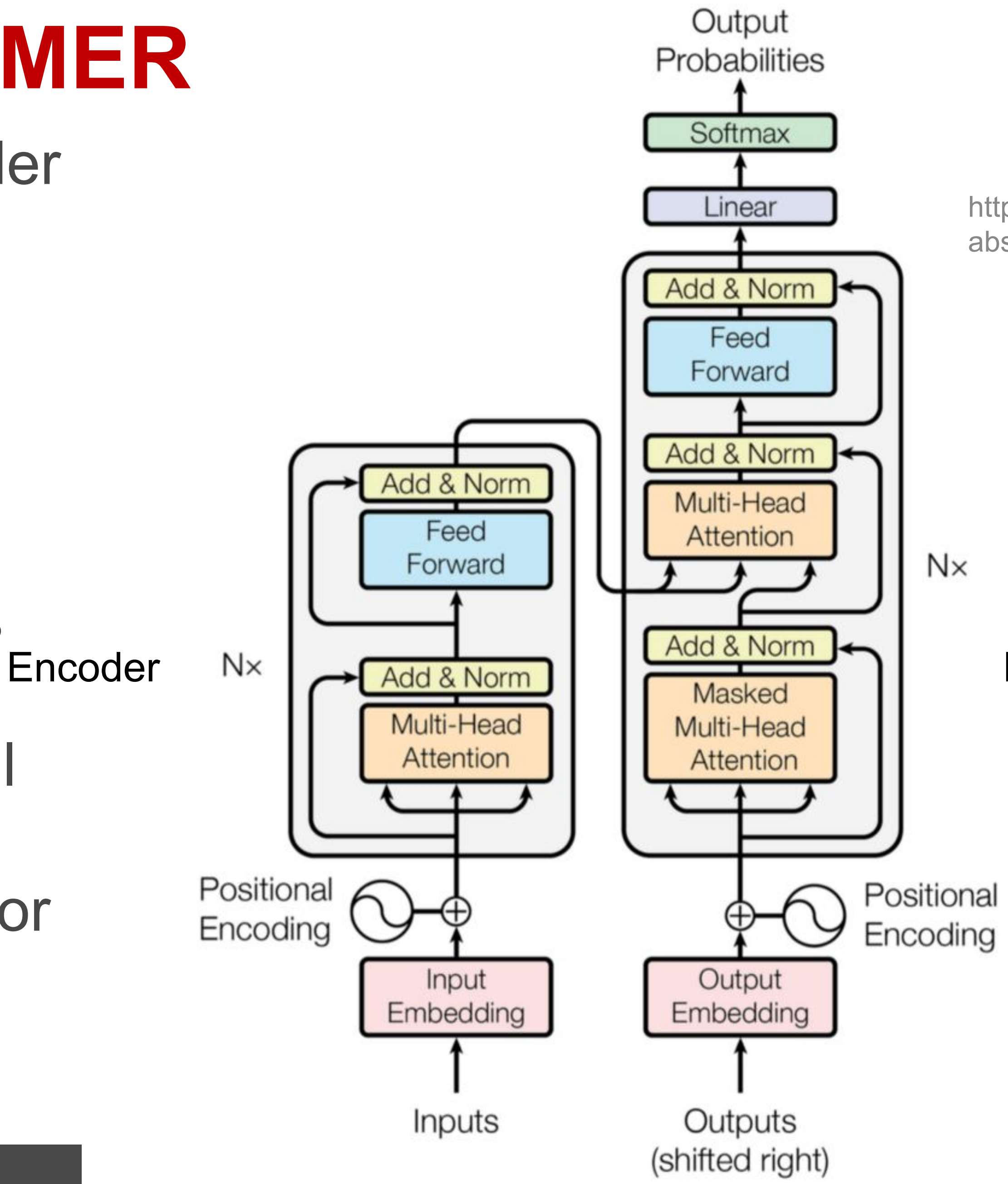
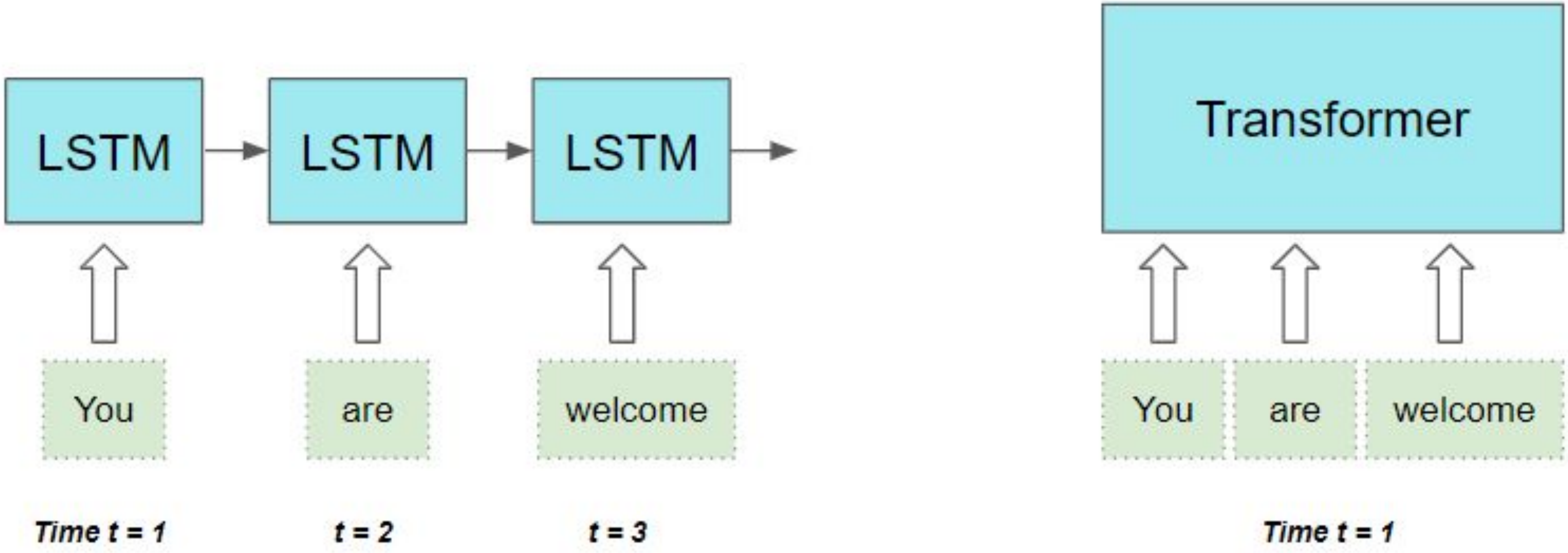


Figure 1: The Transformer - model architecture.

https://arxiv.org/abs/1706.03762

TRANSFORMER ADVANTAGES OVER RNN

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



https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

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

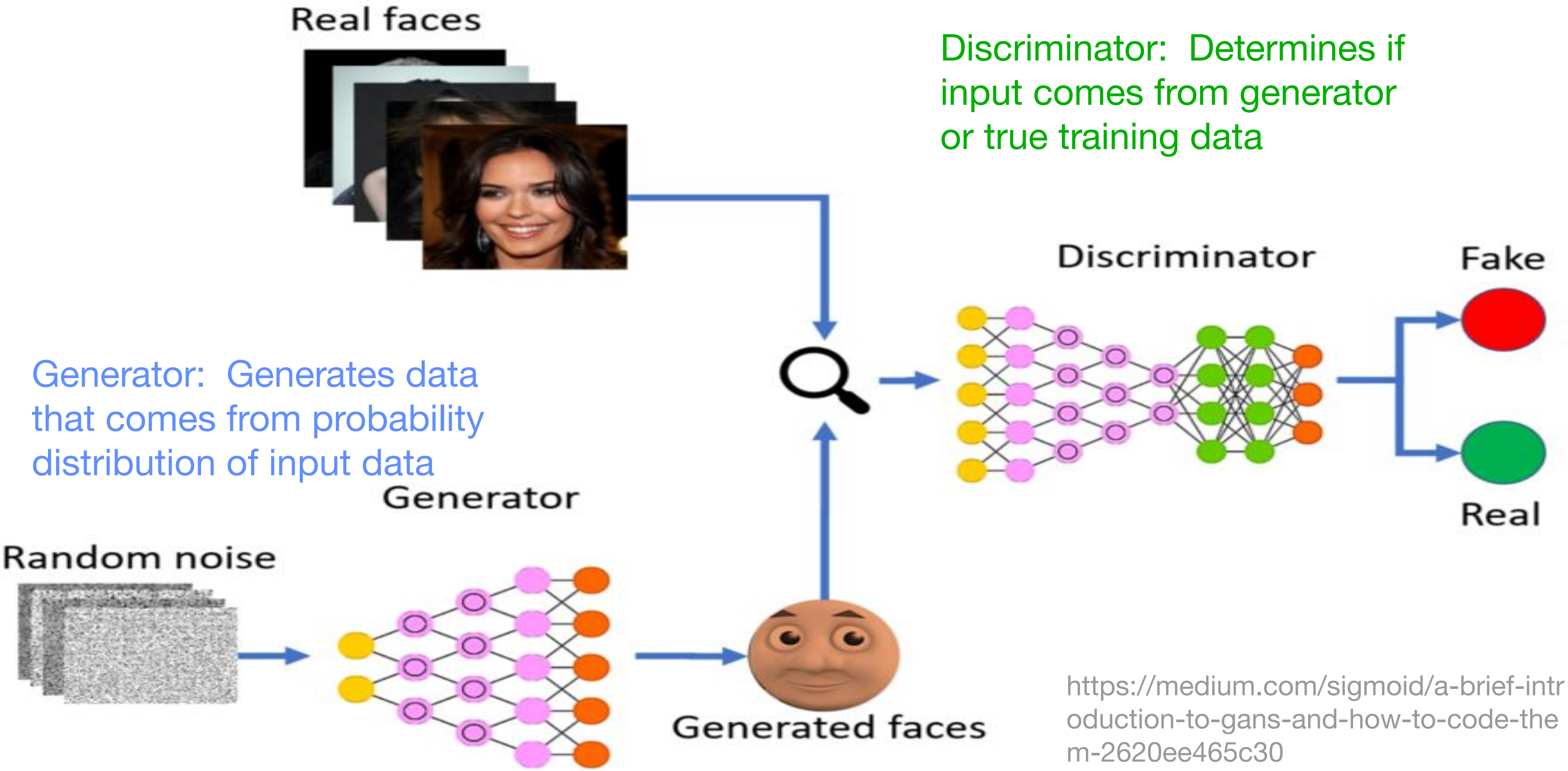
TRANSFORMER APPLICATIONS

- **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

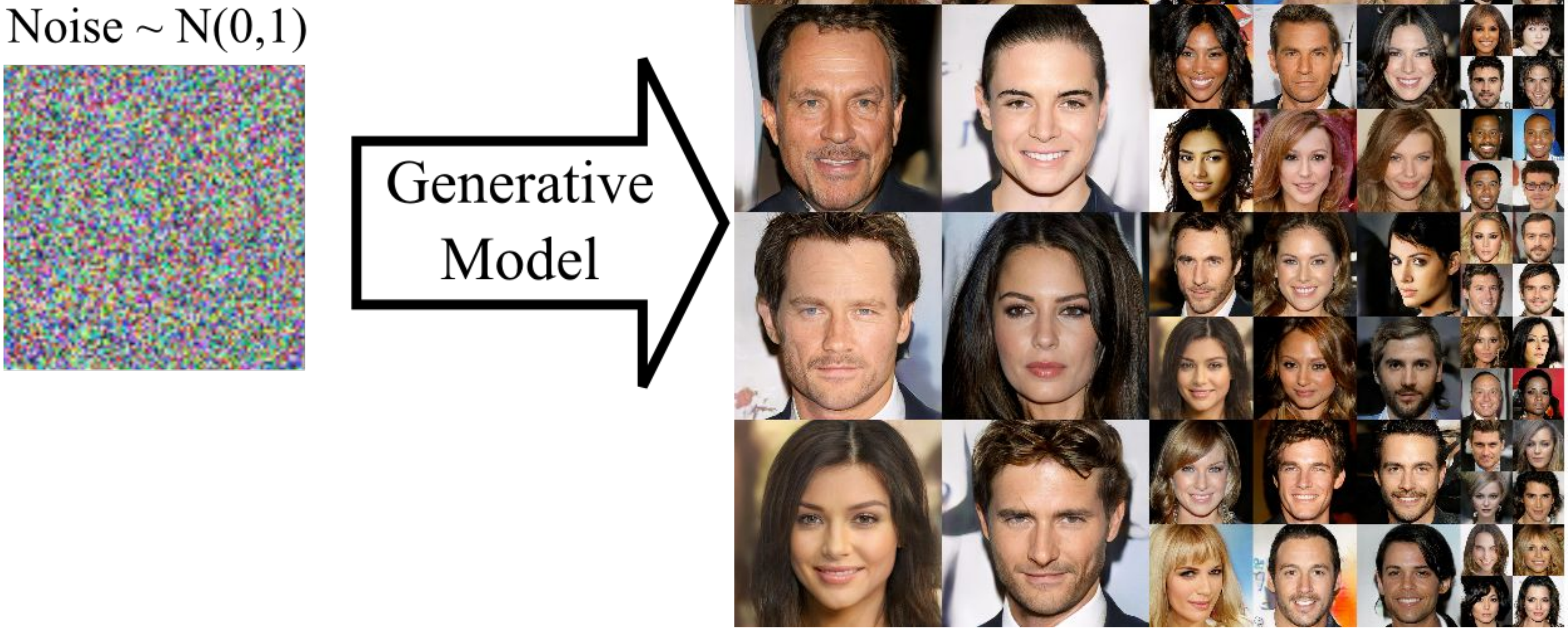


Discriminator: Determines if input comes from generator or true training data

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

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

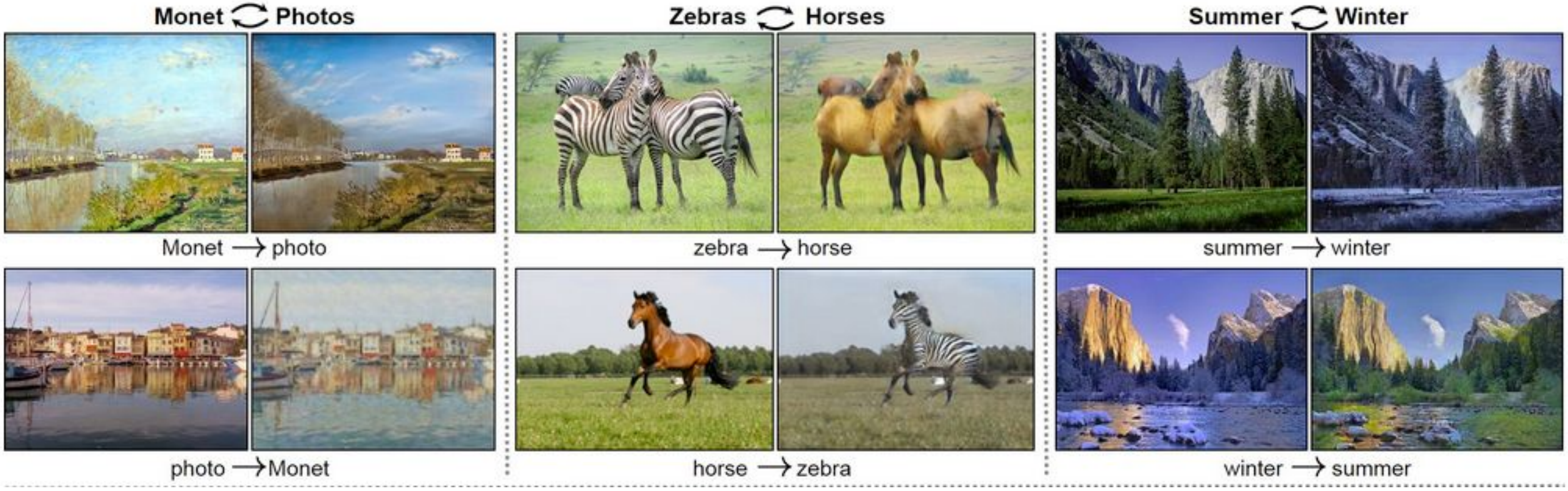
GAN APPLICATIONS



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

GAN APPLICATIONS

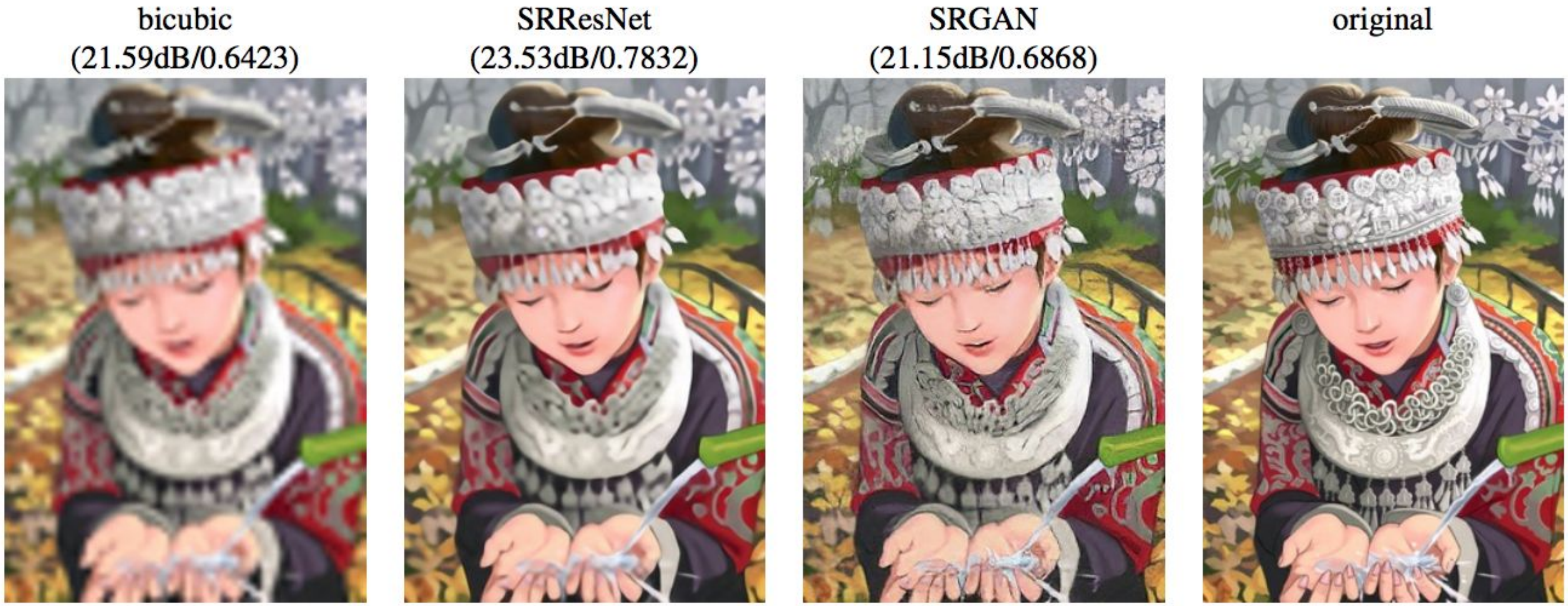
- **Image-to-Image Translation**
  - Transform images from one domain (e.g., real scenery) to another domain (Monet paintings)



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]

https://arxiv.org/pdf/1609.04802.pdf

GAN APPLICATIONS

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

GENERATIVE AI

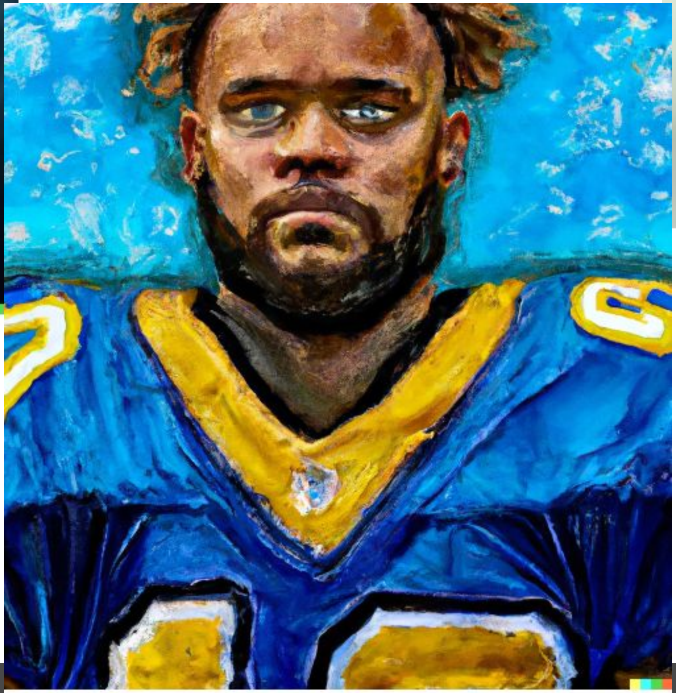
- **AI that can create new content resembling human-generated content**
    - Various modalities (e.g., text, images, audio)
- **Generative AI models**
    - Learn patterns from existing data to generate new data that is similar
- **Applications**
    - Text generation
    - Text-to-image generation
    - Programming code generation
    - Music generation
    - Many others …

DALL-E

https://openai.com/product/dall-e-2

Generates image from text description

A bowl of soup that is also a portal to another dimension, digital art



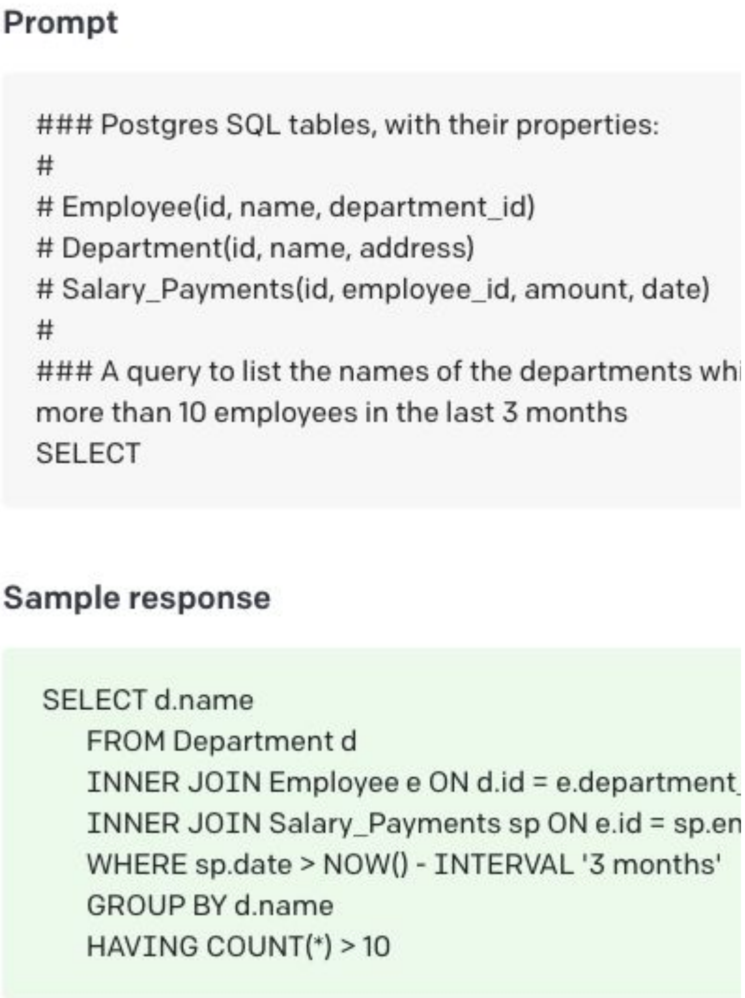
A van Gogh style painting of an American football player

An armchair in the shape of an avocado

CODEX

https://openai.com/blog/openai-codex

Convert natural language to executable code



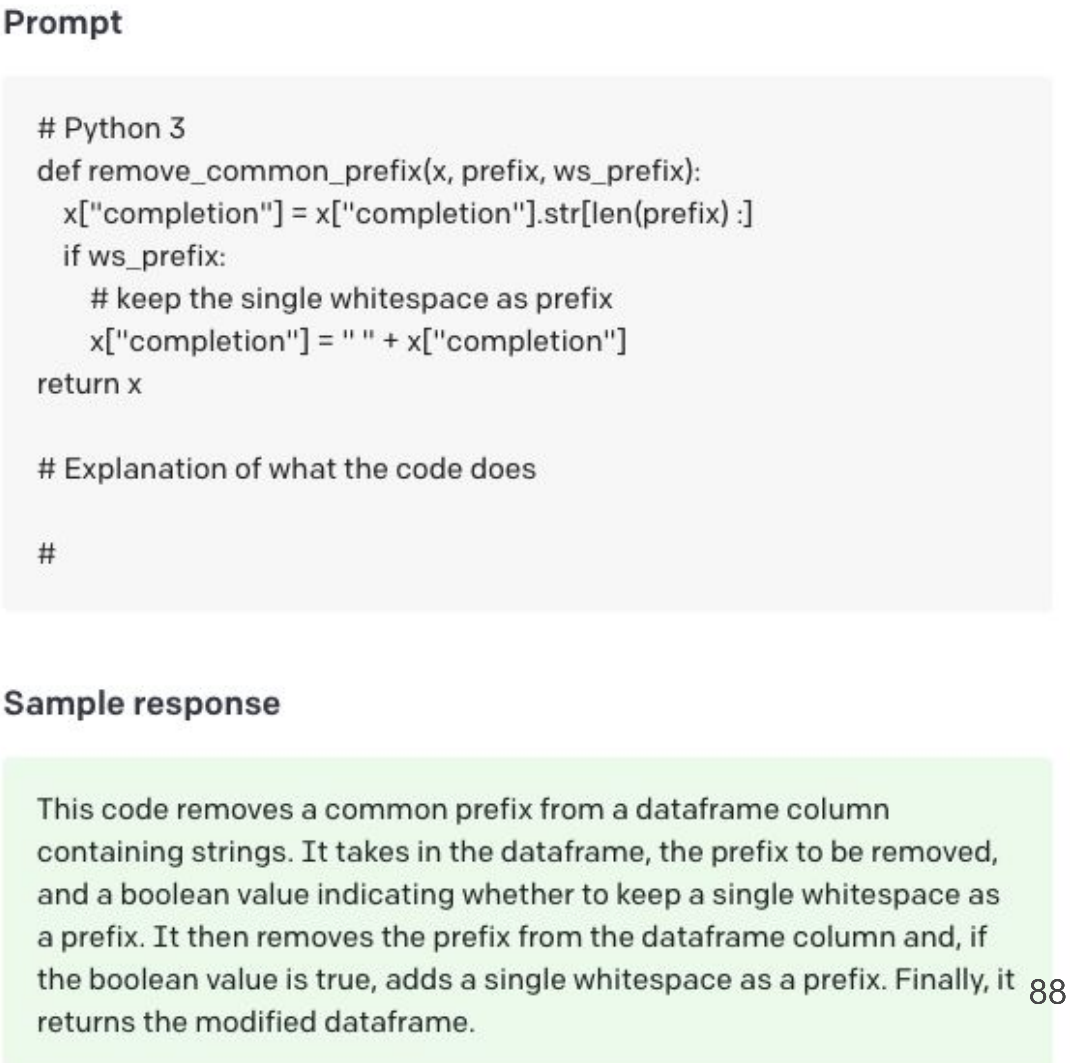
Prompt

```
### Postgres SQL tables, with their properties:
#
# Employee(id, name, department_id)
# Department(id, name, address)
# Salary_Payments(id, employee_id, amount, date)
#
### A query to list the names of the departments whi
more than 10 employees in the last 3 months
SELECT
```

Sample response

```
SELECT d.name
    FROM Department d
    INNER JOIN Employee e ON d.id = e.department_
    INNER JOIN Salary_Payments sp ON e.id = sp.en
    WHERE sp.date > NOW() - INTERVAL '3 months'
    GROUP BY d.name
    HAVING COUNT(*) > 10
```

Explain code segment in understandable language



Prompt

```
# Python 3
def remove_common_prefix(x, prefix, ws_prefix):
  x["completion"] = x["completion"].str[len(prefix) :]
  if ws_prefix:
     # keep the single whitespace as prefix
     x["completion"] = " " + x["completion"]
return x

# Explanation of what the code does

#
```

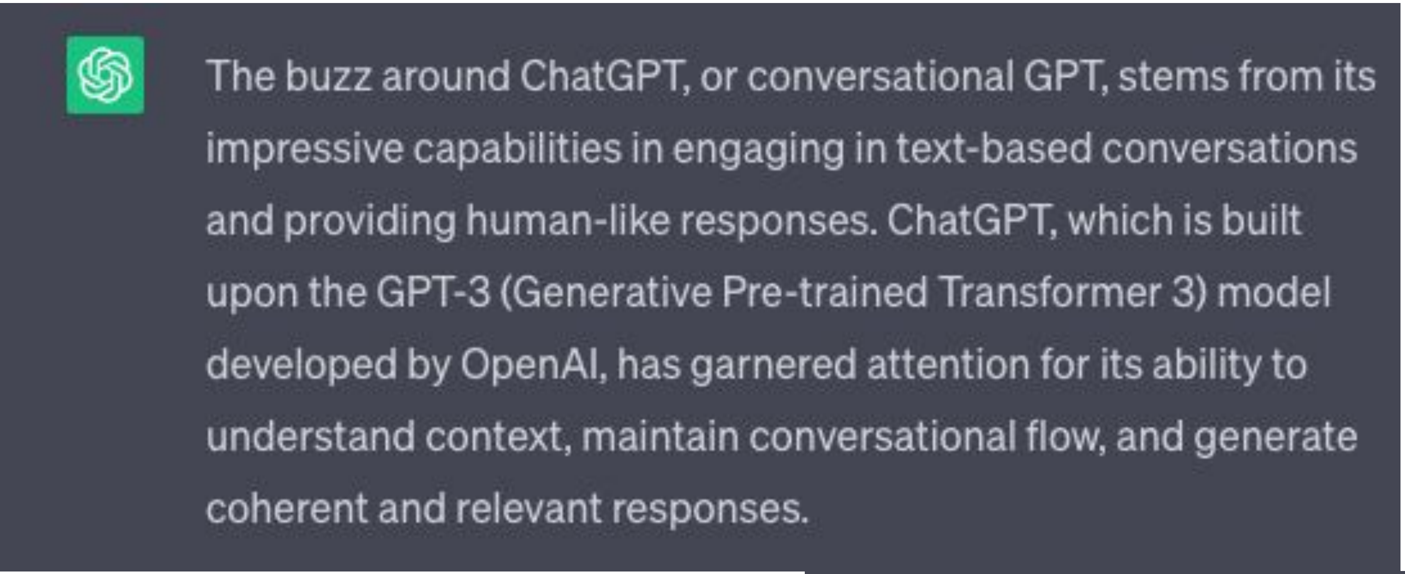
Sample response

This code removes a common prefix from a dataframe column containing strings. It takes in the dataframe, the prefix to be removed, and a boolean value indicating whether to keep a single whitespace as a prefix. It then removes the prefix from the dataframe column and, if the boolean value is true, adds a single whitespace as a prefix. Finally, it returns the modified dataframe.

CHAT-GPT

https://openai.com/blog/chatgpt

Chatbot that can engage in conversational interactions with users

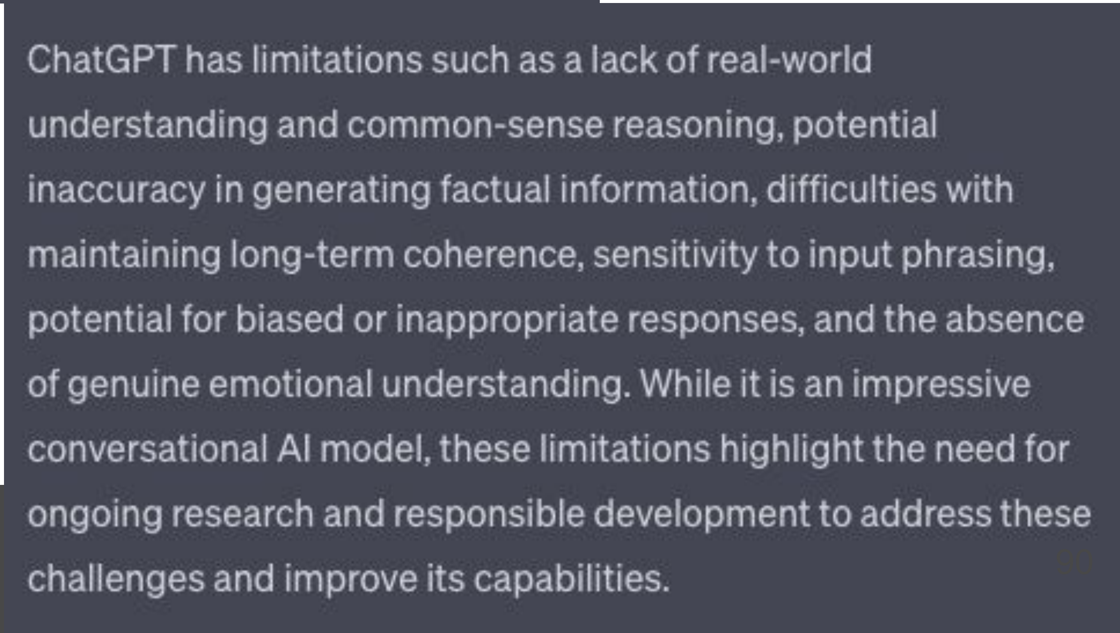


The buzz around ChatGPT, or conversational GPT, stems from its impressive capabilities in engaging in text-based conversations and providing human-like responses. ChatGPT, which is built upon the GPT-3 (Generative Pre-trained Transformer 3) model developed by OpenAI, has garnered attention for its ability to understand context, maintain conversational flow, and generate coherent and relevant responses.

Prompt:
What is all the buzz around ChatGPT?

Follow-up prompts:

- What are somethings that ChatGPT cannot do?
- Summarize in 1 paragraph



ChatGPT has limitations such as a lack of real-world understanding and common-sense reasoning, potential inaccuracy in generating factual information, difficulties with maintaining long-term coherence, sensitivity to input phrasing, potential for biased or inappropriate responses, and the absence of genuine emotional understanding. While it is an impressive conversational AI model, these limitations highlight the need for ongoing research and responsible development to address these challenges and improve its capabilities.

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

DEEP LEARNING OVERVIEW

- **Neural Network Basics**
  - Processing Unit
  - Activation Function
  - Loss Function
- **Deep Learning Fundamentals**
  - Deep Network Layers
  - DL Architectures
  - DL Libraries
- **Transfer Learning**
  - Transfer Learning Concepts
  - Transfer Learning Demo

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/