

SDSC Summer Institute 2021

Deep Learning

Mai H. Nguyen & Paul Rodriguez
2021-August-05



Deep Learning Agenda

8:00 - 8:45 -- Intro to Neural Networks / CNNs

8:45 - 9:45 -- MNIST & TensorBoard Hands-On

9:45 - 10:00 -- Break

10:00 - 10:45 -- DL Layers & Architectures

10:45 - 11:15 -- Lunch

11:15 - 12:30 -- Transfer Learning Hands-On

12:30 - 12:45 -- Break

12:45 - 1:45 -- Deep Sequence Learning

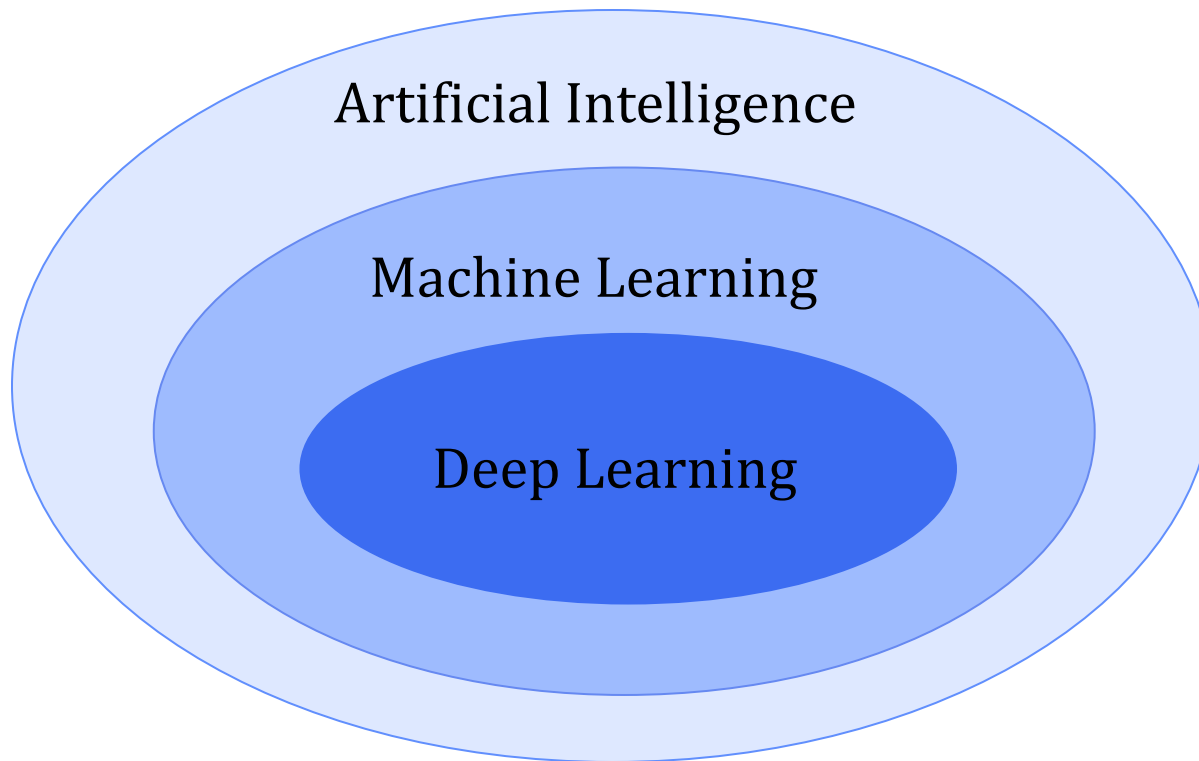
1:45 - 2:00 -- Wrap-Up

Deep Learning Layers & Models

Mai H. Nguyen, Ph.D.

DEEP LEARNING

Deep Learning is a subfield of Machine Learning

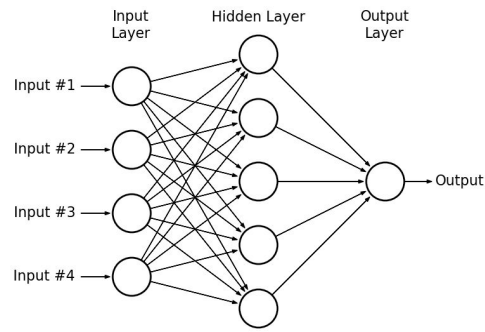


APPLICATIONS OF DEEP LEARNING

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

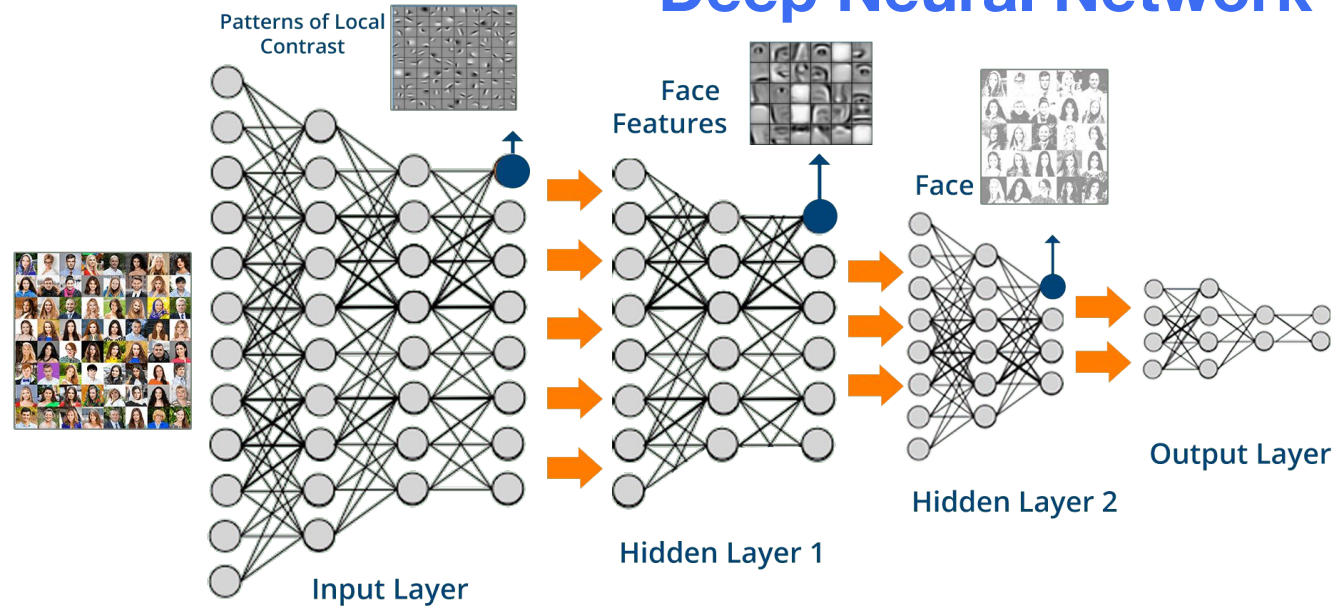
DEEP LEARNING

Neural Network



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

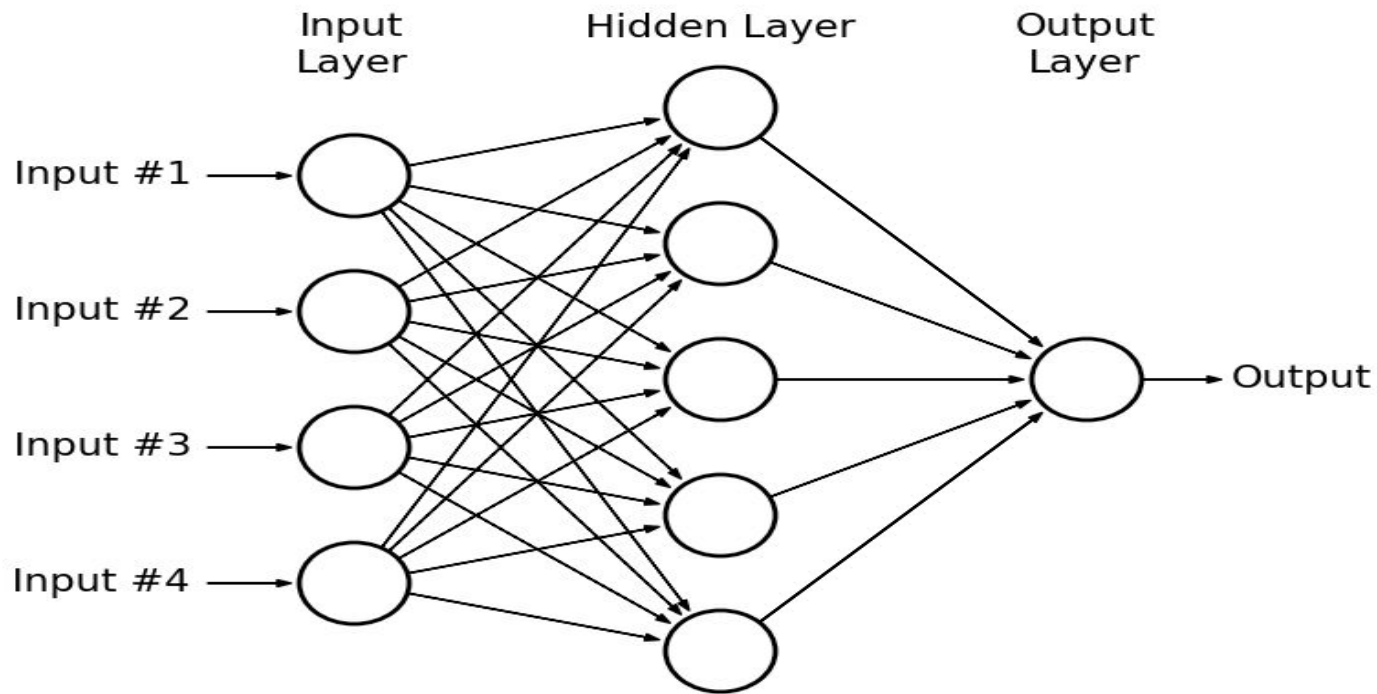
Deep Neural Network



<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

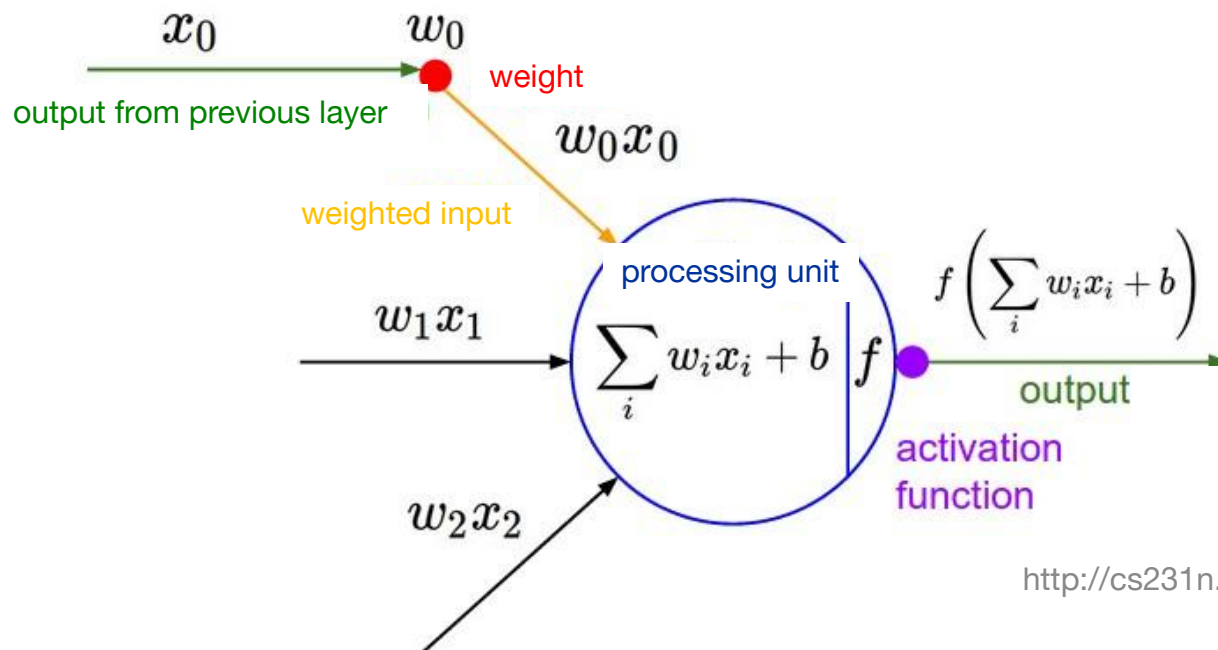
NEURAL NETWORK



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

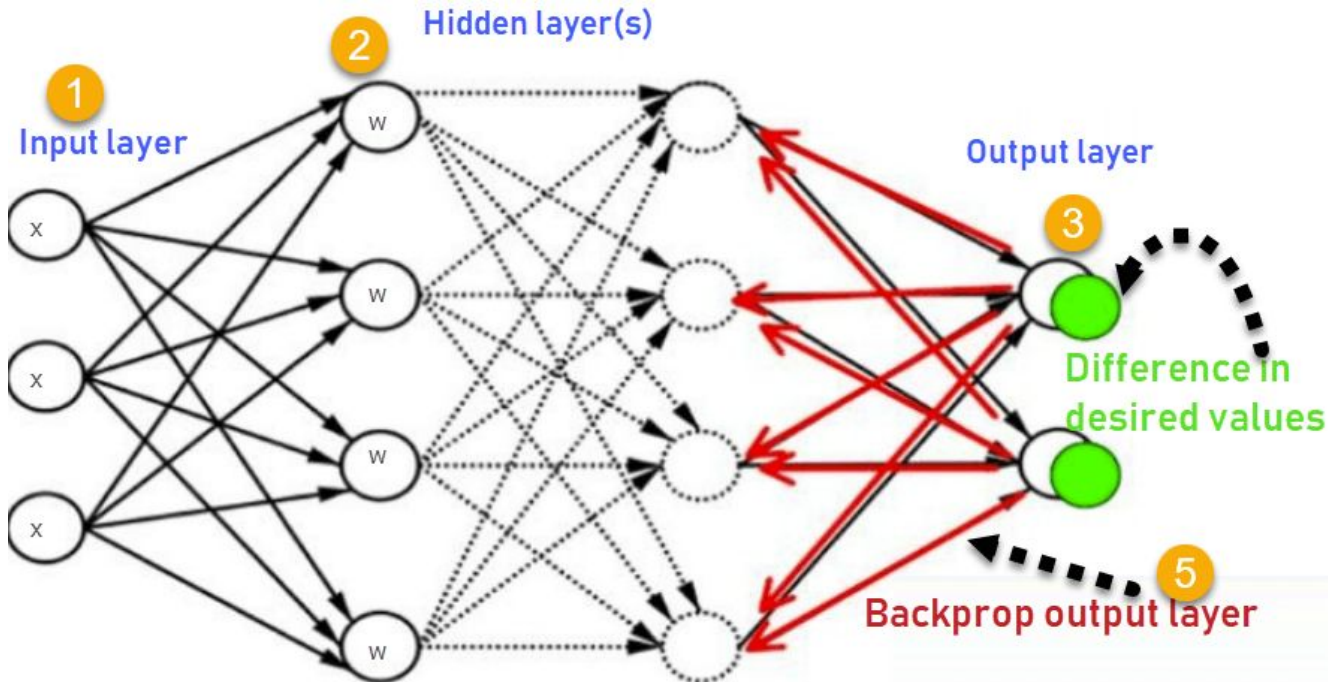
PROCESSING UNIT IN NEURAL NETWORK



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

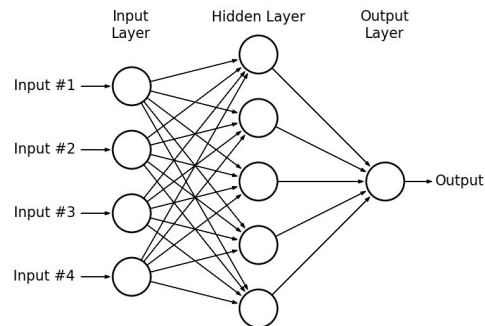


<https://www.guru99.com/backpropagation-neural-network.html>

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

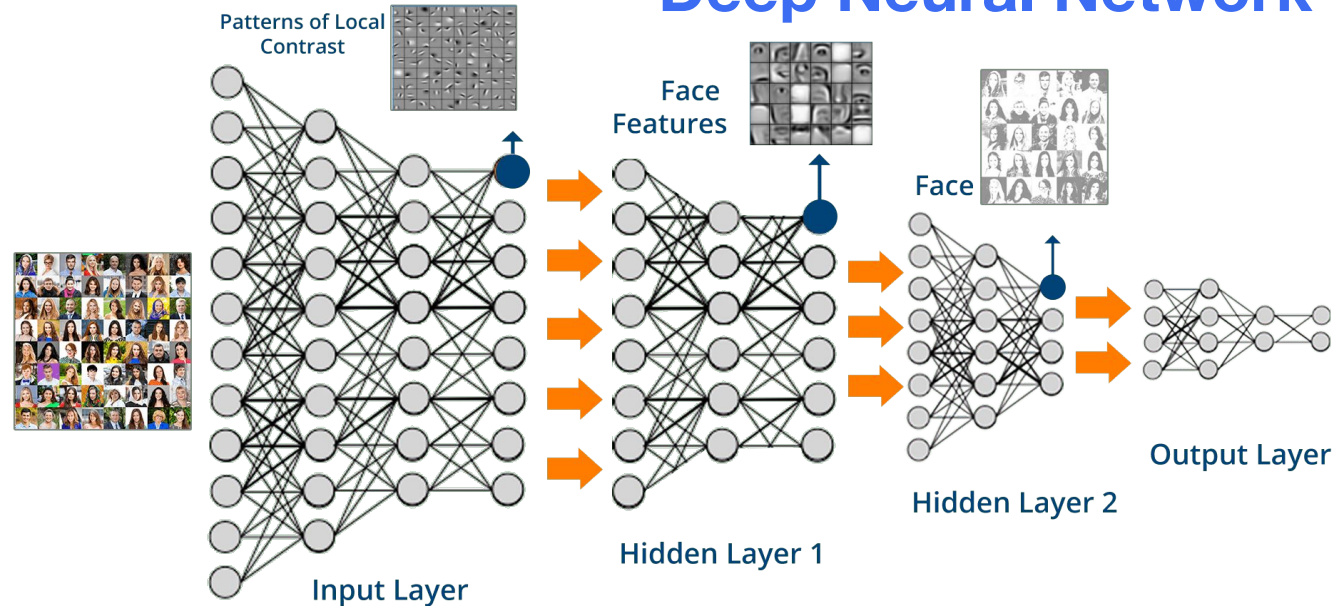
DEEP LEARNING

Neural Network



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

Deep Neural Network

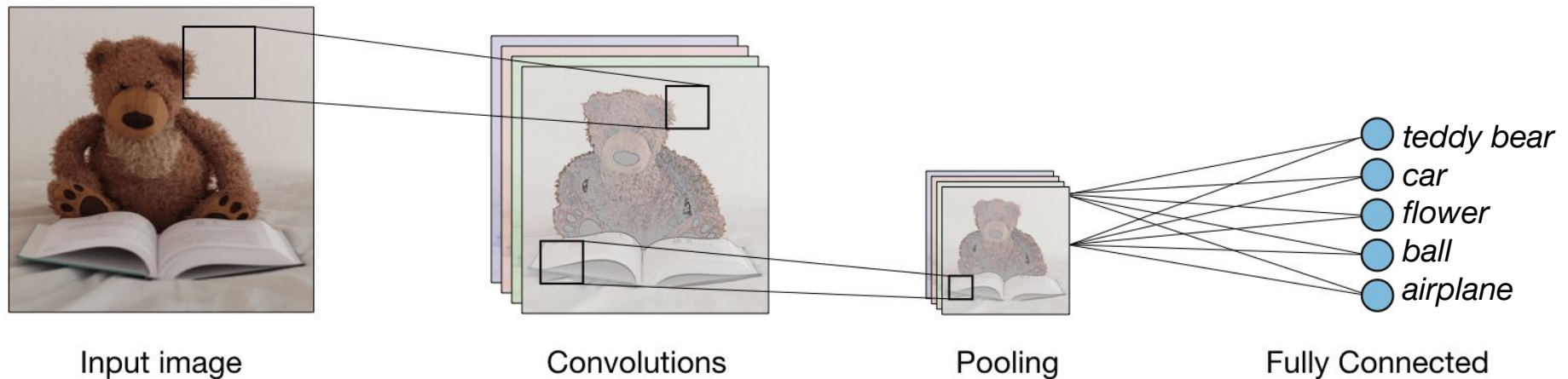


<https://cdn.edureka.co/blog/wp-content/uploads/2017/05/Deep-Neural-Network-What-is-Deep-Learning-Edureka.png>

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

DEEP LEARNING MODELS

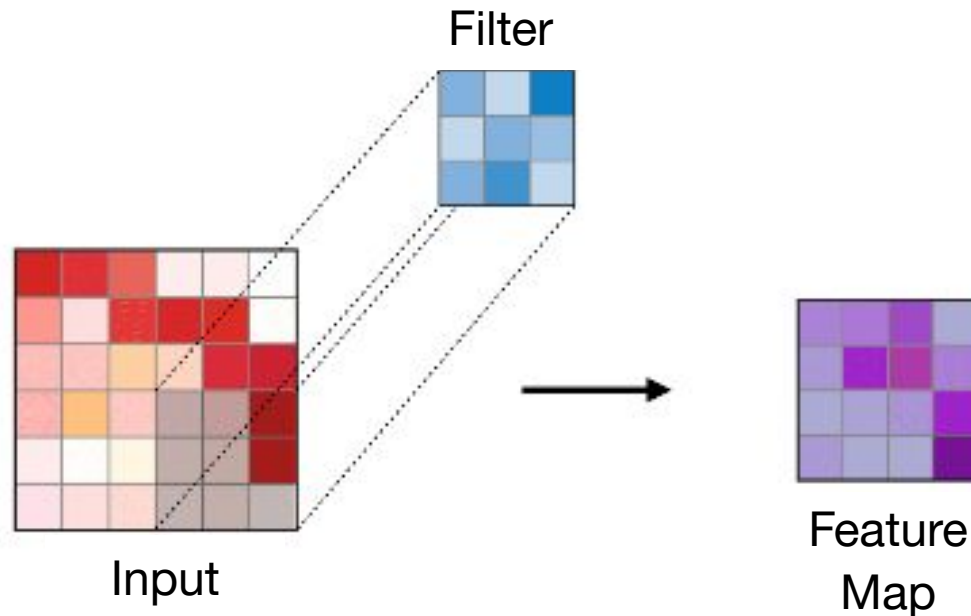
- **General Deep Network Architecture:**
 - Has sequence of layers
 - Each layer transforms its input to generate an output through nonlinear function



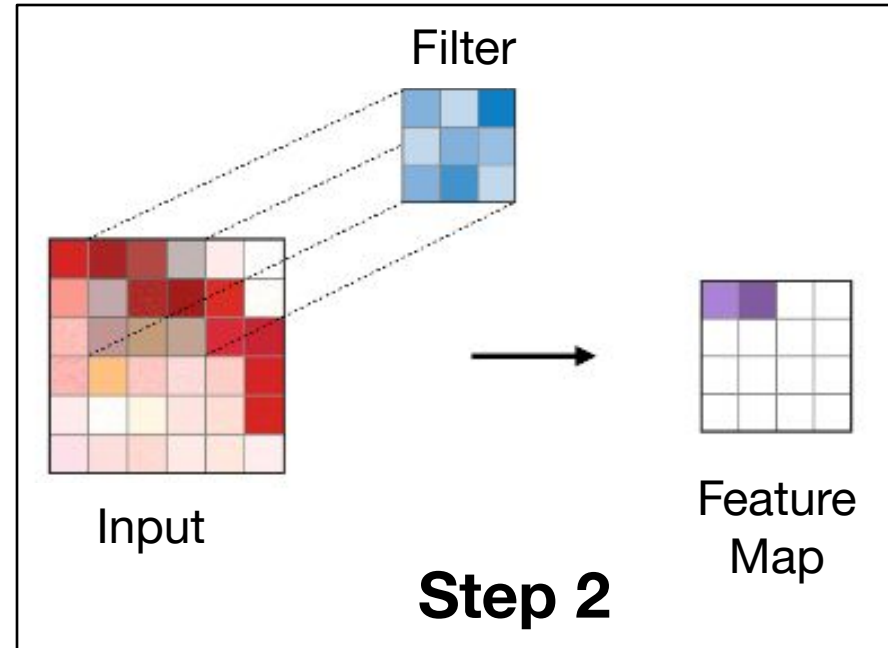
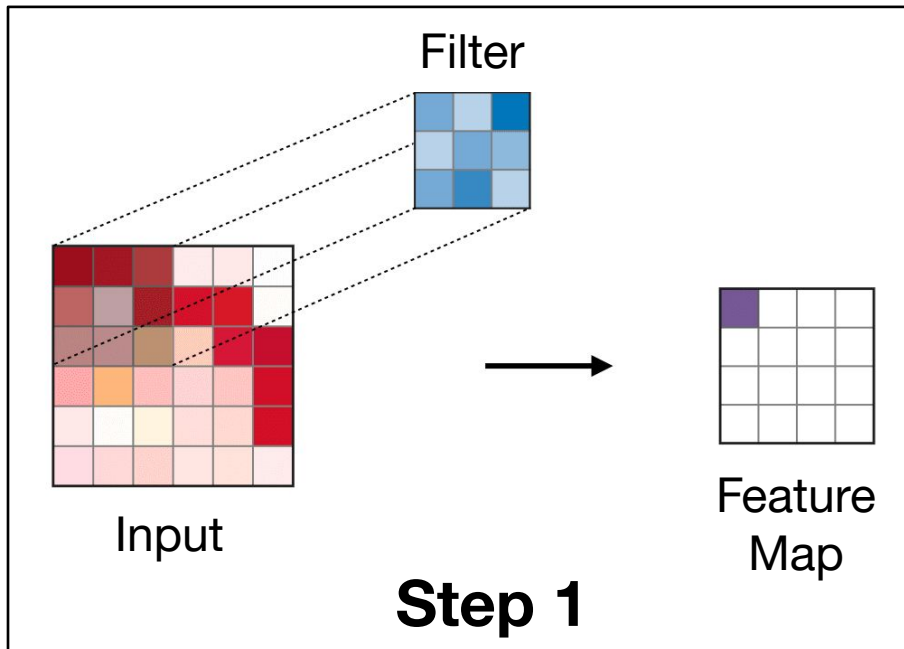
<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer>

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 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Input



4		

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 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0



4		

Input

Feature
Map

Step 2

1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	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 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Input



4		

Feature Map

Step 9

1	1	1	0	0
0	1	1	1	0
0	0	1 _{x1}	1 _{x0}	1 _{x1}
0	0	1 _{x0}	1 _{x1}	0 _{x0}
0	1	1 _{x1}	0 _{x0}	0 _{x1}

Input



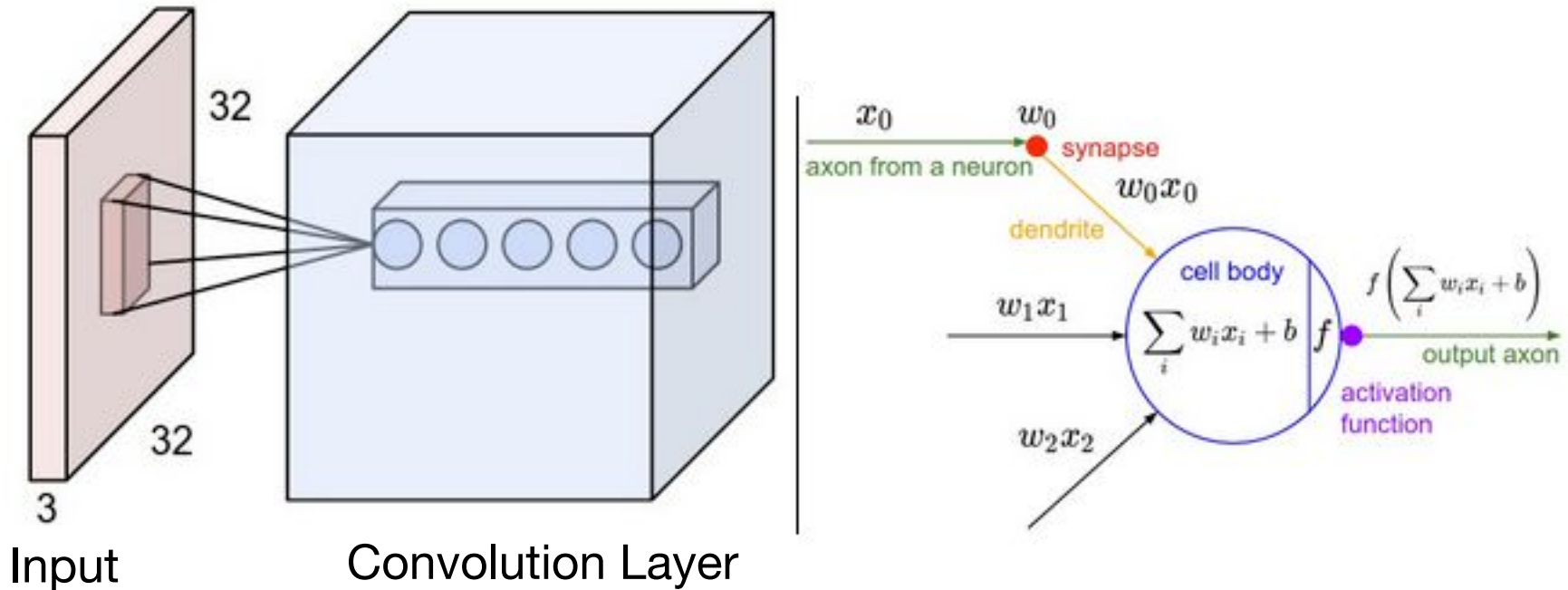
4	3	4
2	4	3
2	3	4

Feature Map

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

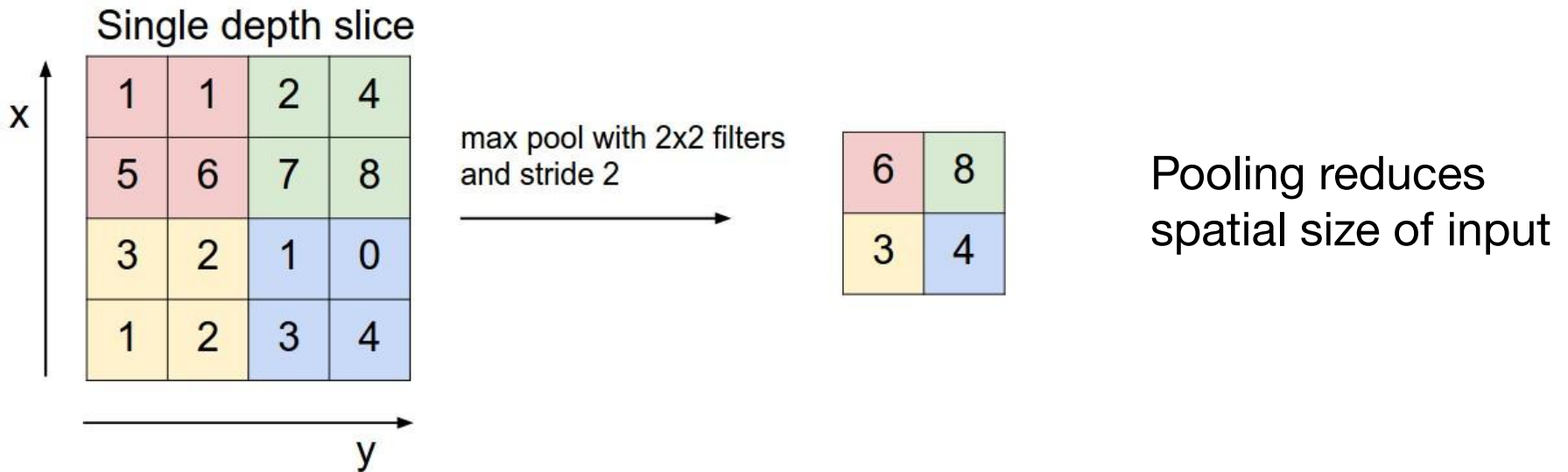
CONVOLUTION LAYER

- 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



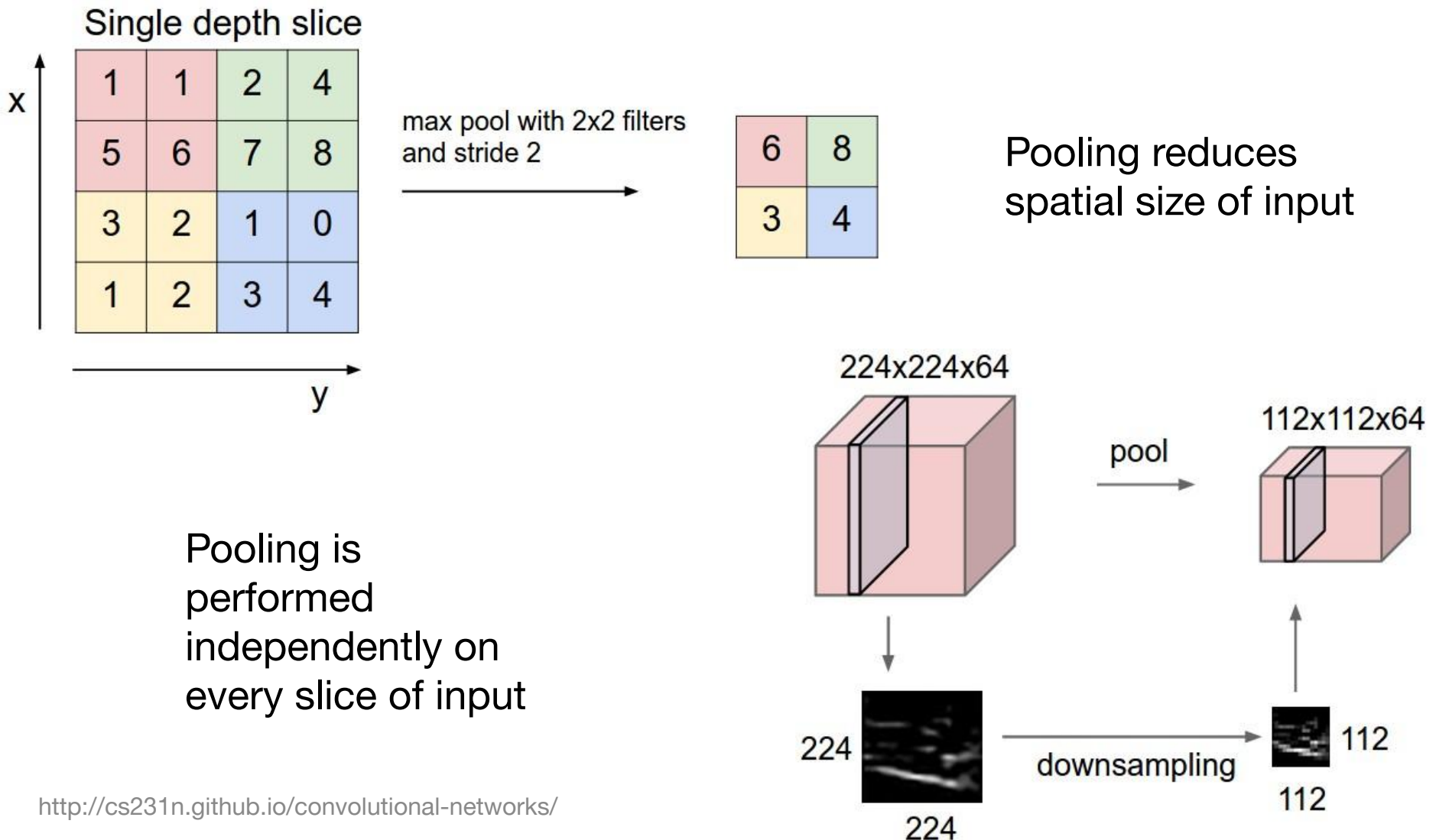
<http://cs231n.github.io/convolutional-networks/>

POOLING LAYER



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

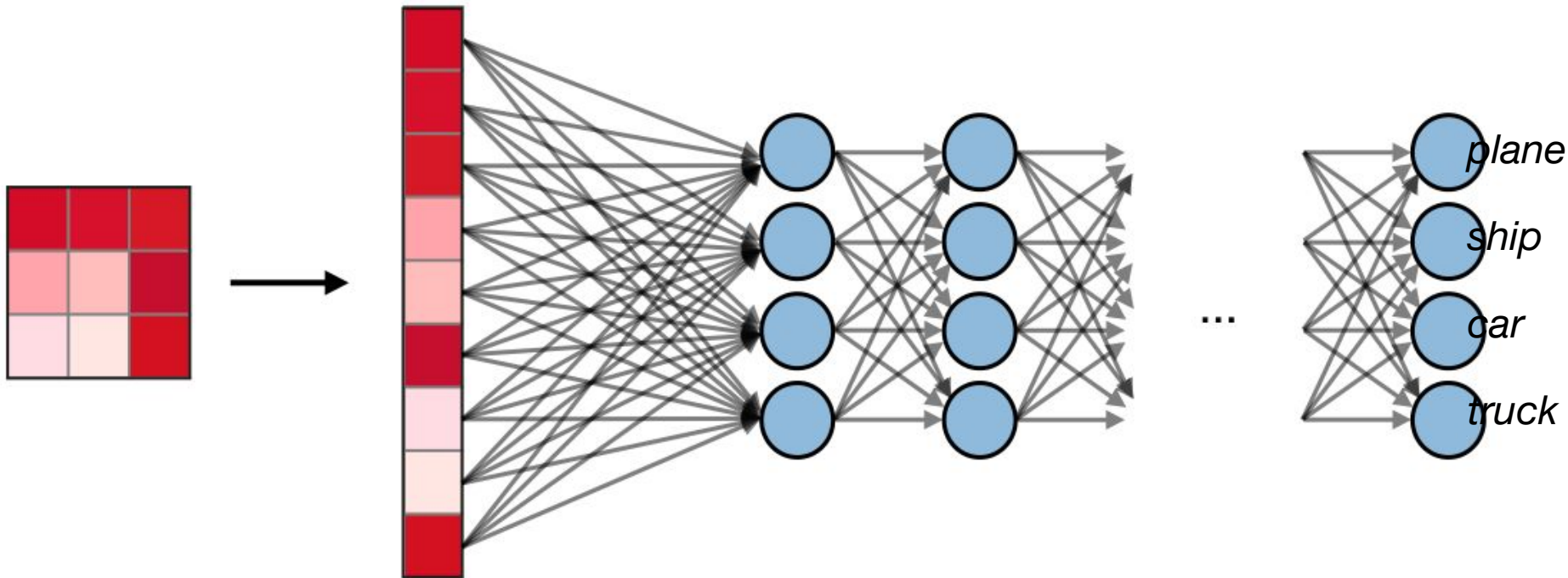
POOLING LAYER



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

FULLY CONNECTED LAYER

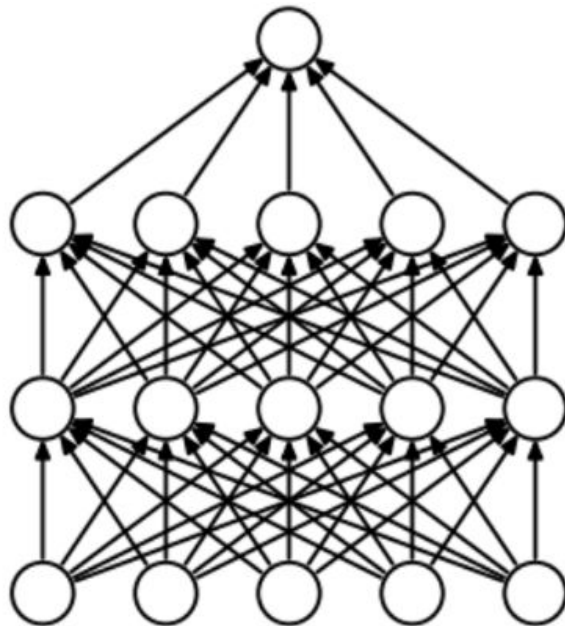
- Fully connected (FC) layer takes flattened input.
- Every input is connected to all processing units.
- Output of FC layer is typically vector with probabilities for categories.



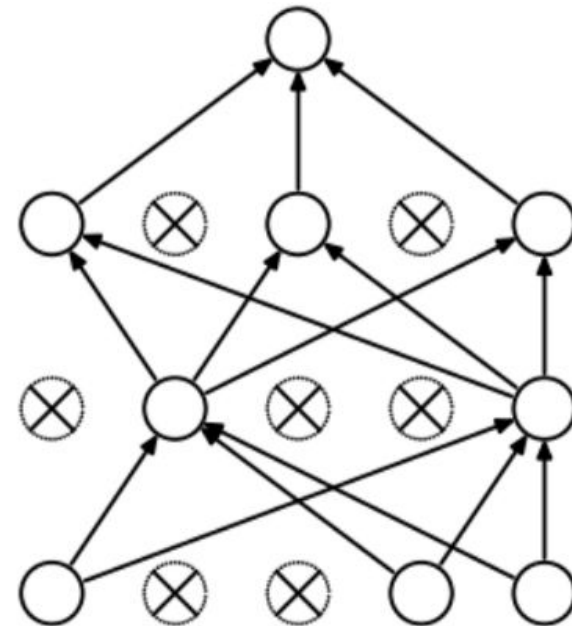
<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks#style-transfer>

DROPOUT

- Randomly drop units during training
- Prevents units from co-adapting
- Helps to address overfitting



(a) Standard Neural Net



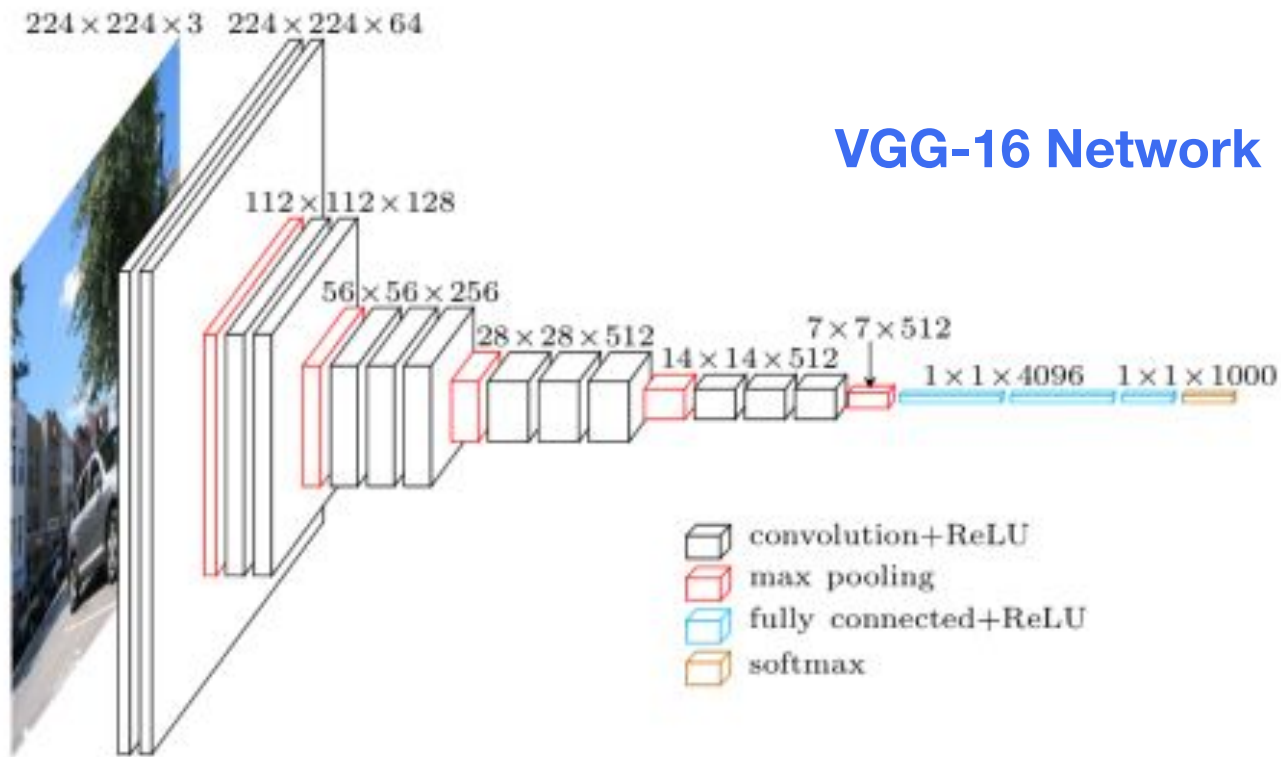
(b) After applying dropout.

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 image of size width X height X # of channels
- Output is vector of numbers representing class probabilities

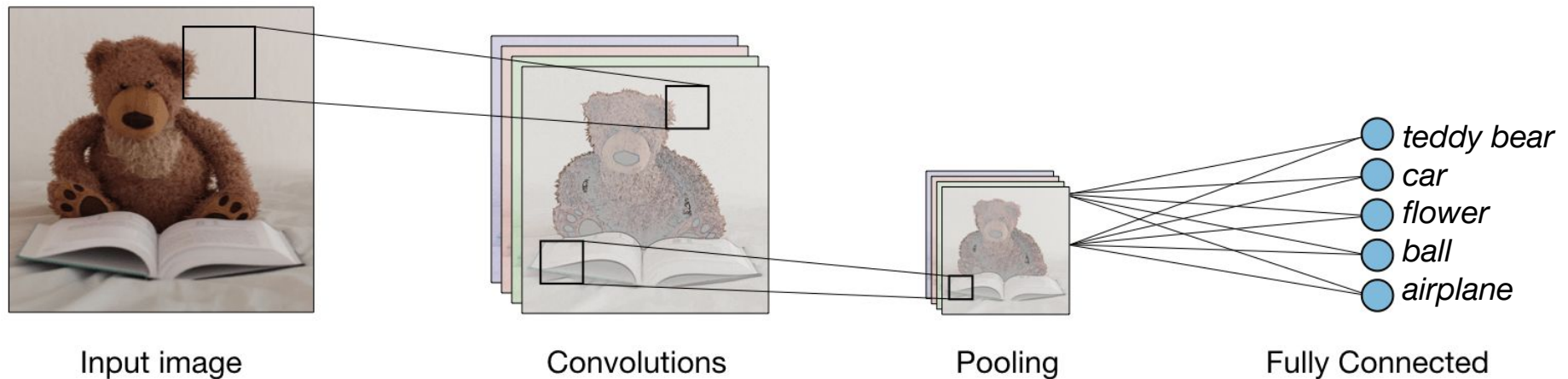


<https://www.cs.toronto.edu/~frossard/post/vgg16/>

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

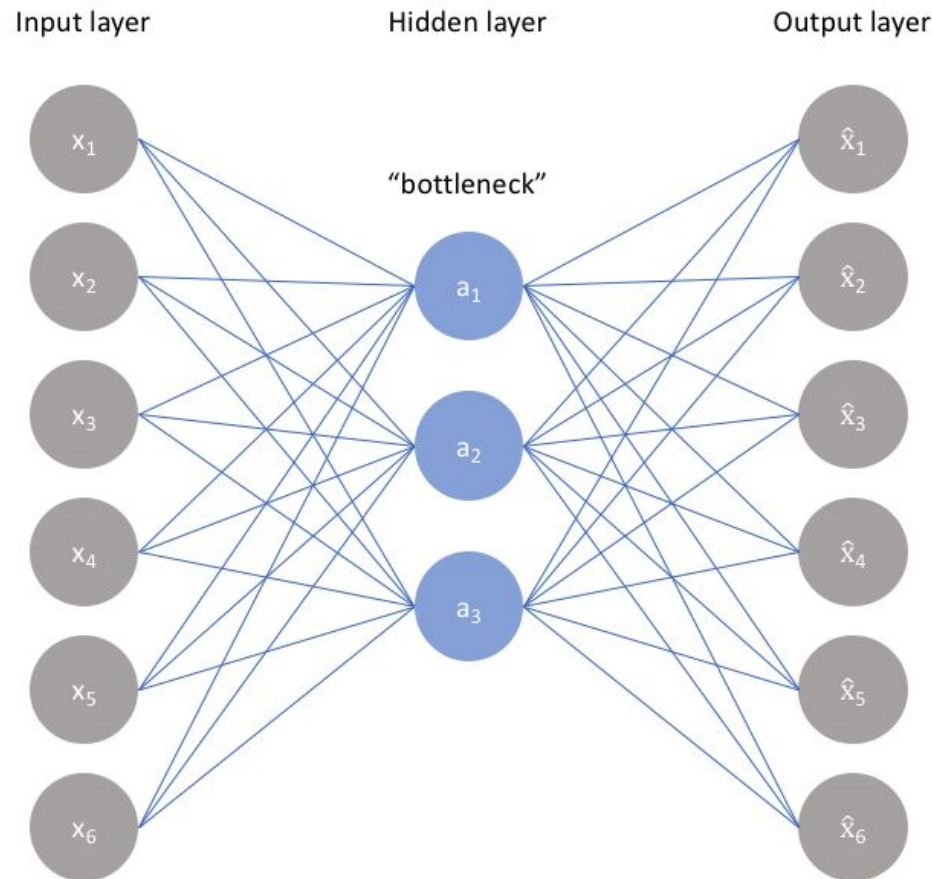
- LeNet
- AlexNet
- VGG
- Inception
- ResNet
- XceptionNet
- Inception-ResNet
- ...

CNN Applications

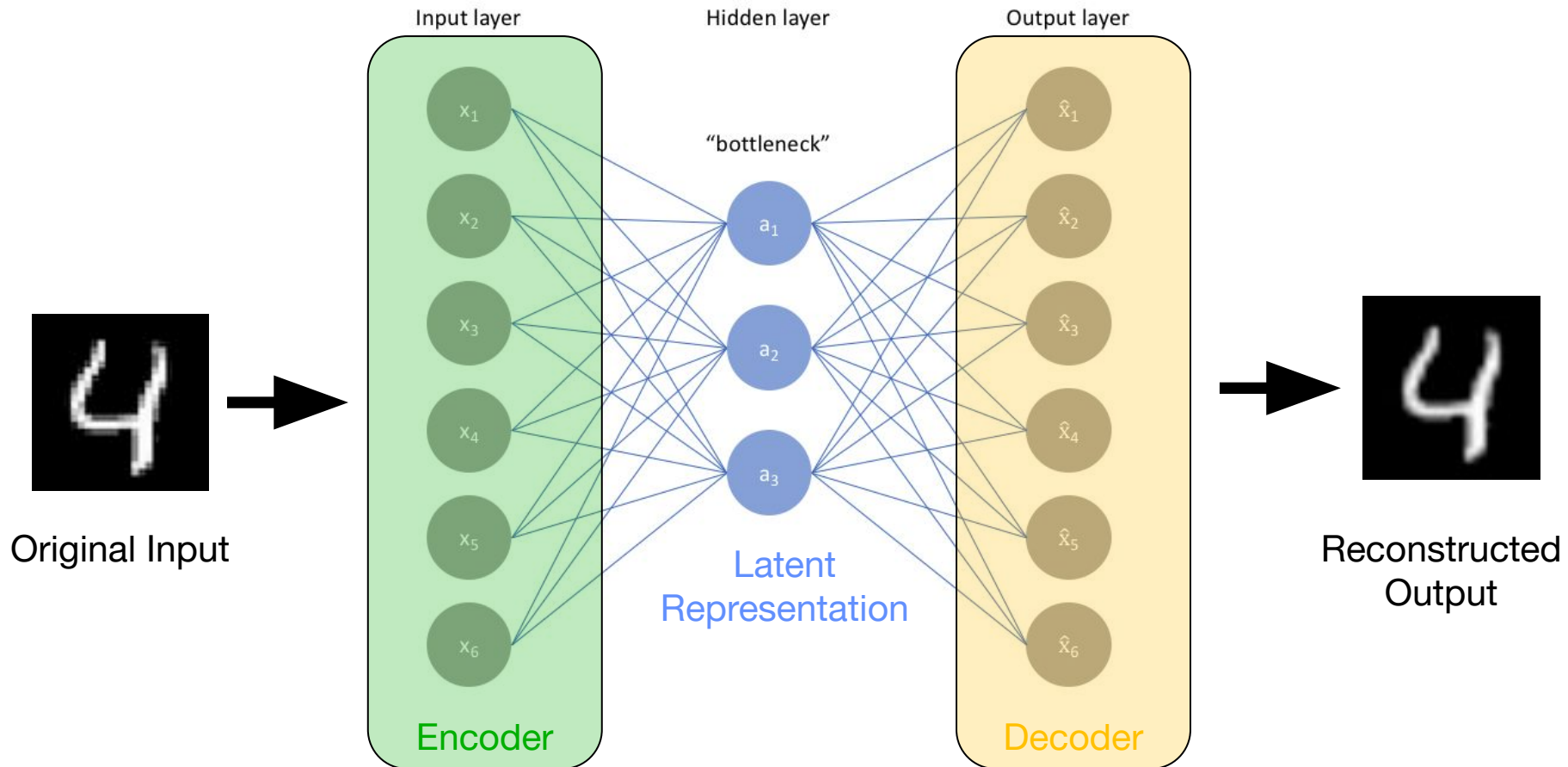
- **Image Analysis**
 - Object classification, localization, detection
 - Face recognition
 - Text classification
- **Natural Language Processing**
 - Topic modeling
 - Part-of-speech tagging
- **Others**
 - Drug design
 - Crime hot spots identification
 - House price prediction

AUTOENCODER

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



AUTOENCODER



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

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

AUTOENCODER

- **Variations**

- Sparse
- Denoising
- Contractive
- Variational

- **Uses**

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

U-NET

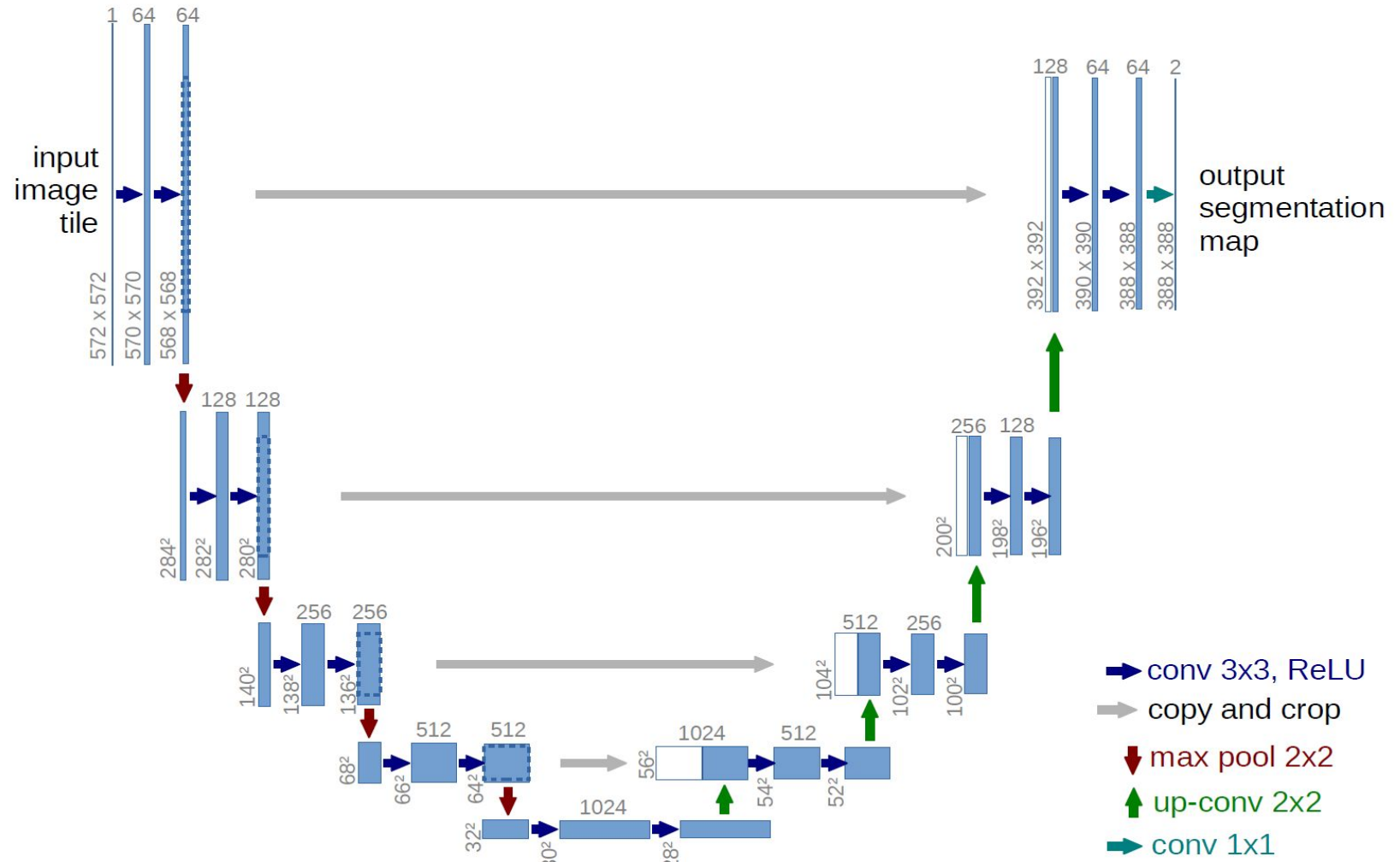
- **Semantic Segmentation**

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

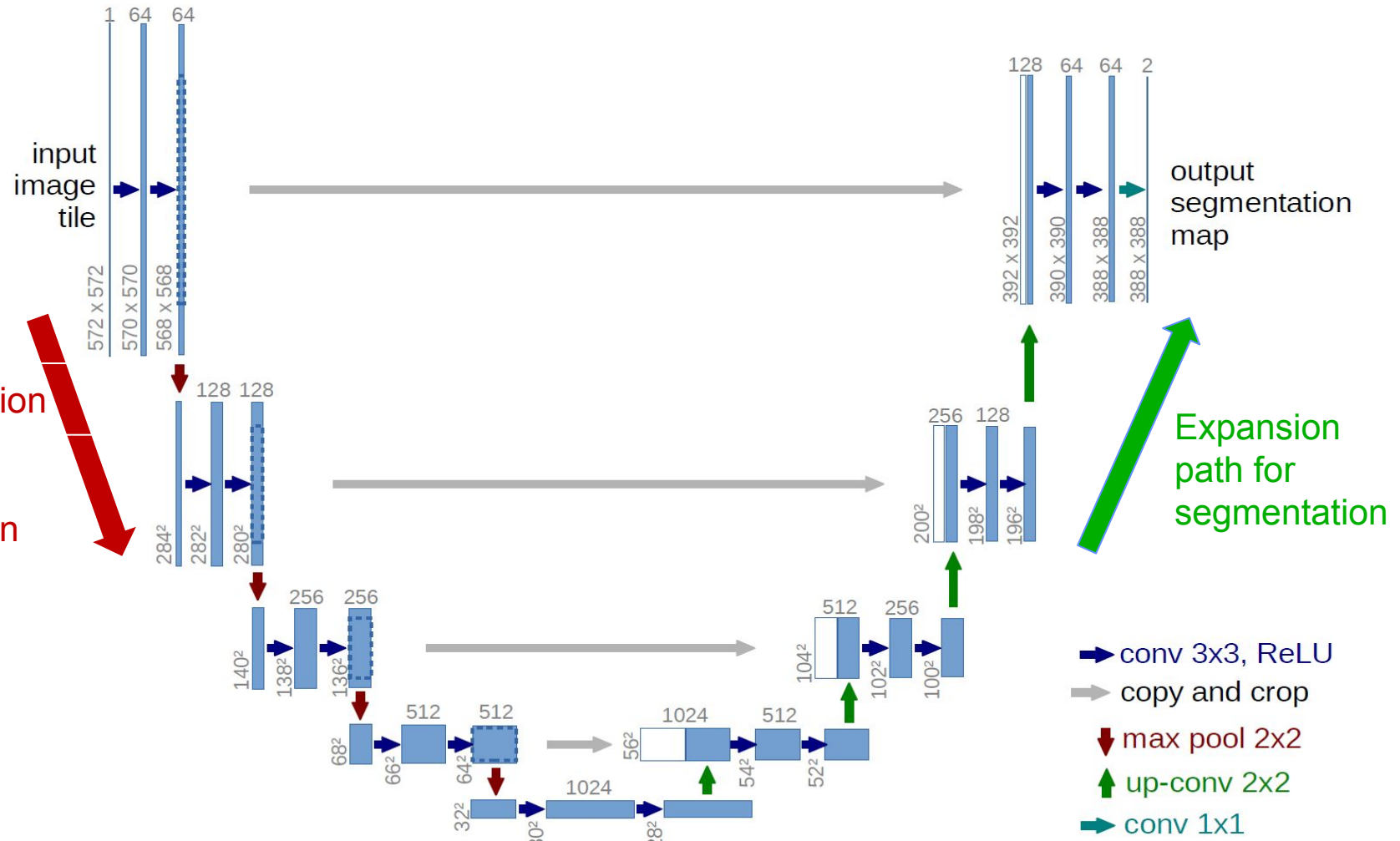


<https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066>

U-NET ARCHITECTURE

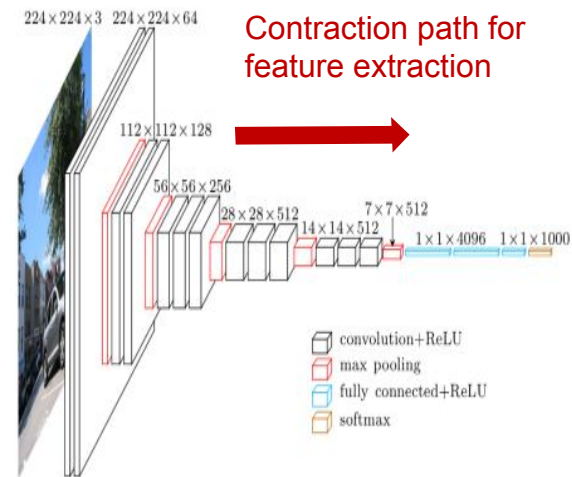


U-NET ARCHITECTURE

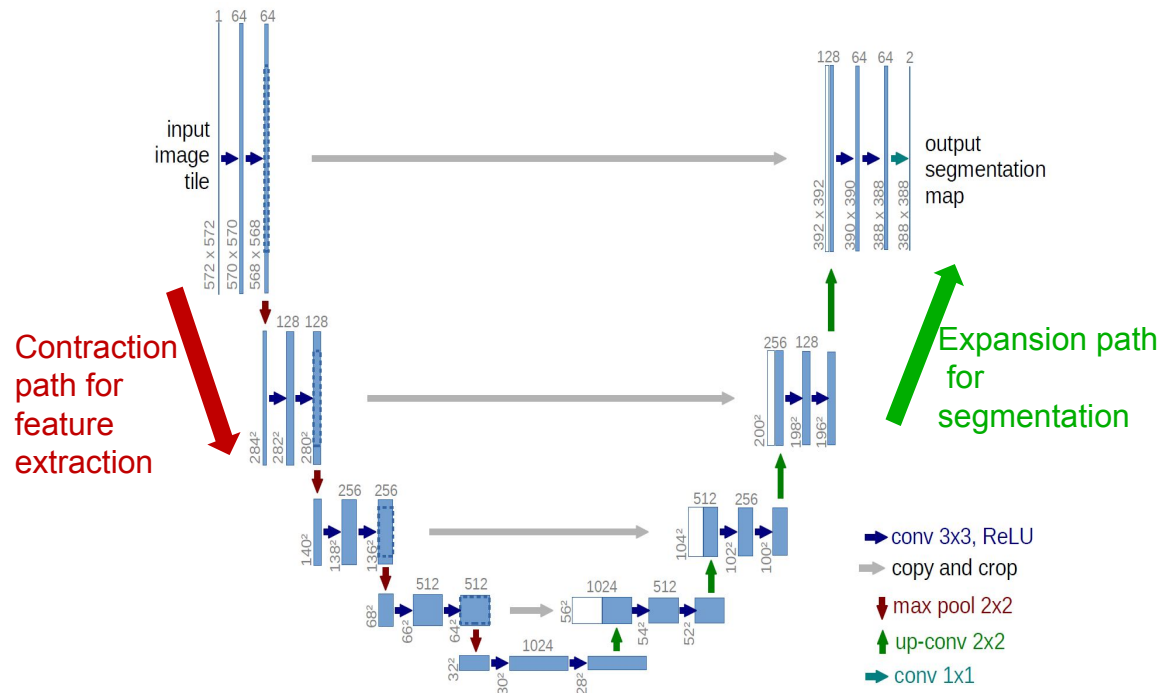


U-NET ARCHITECTURE

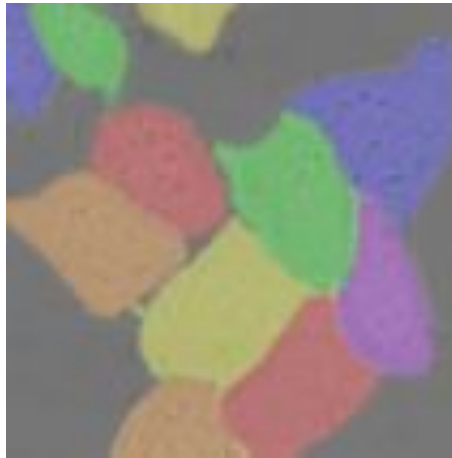
VGG16 CNN Architecture



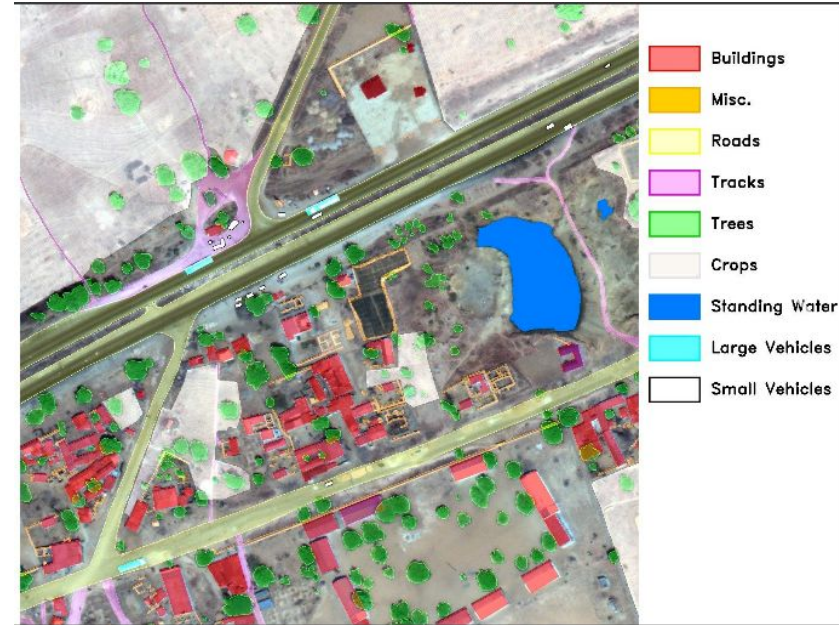
U-Net Architecture



U-NET APPLICATIONS



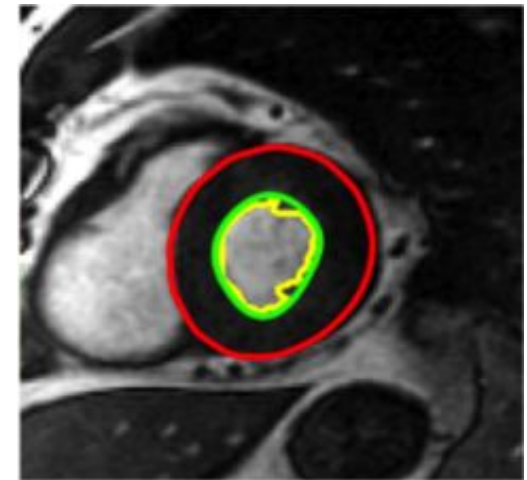
Biomedical Segmentation



Satellite Image Processing



Object Detection



Medical Image Analysis

LSTM

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

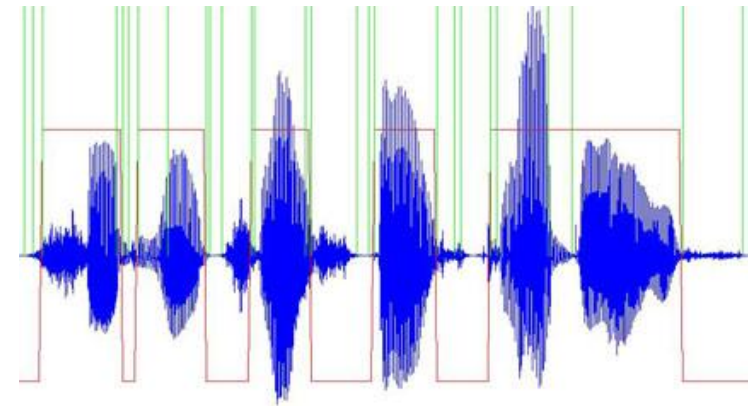
LSTM

- **Sequence Learning**

- Learning a signal with an ordering or time component

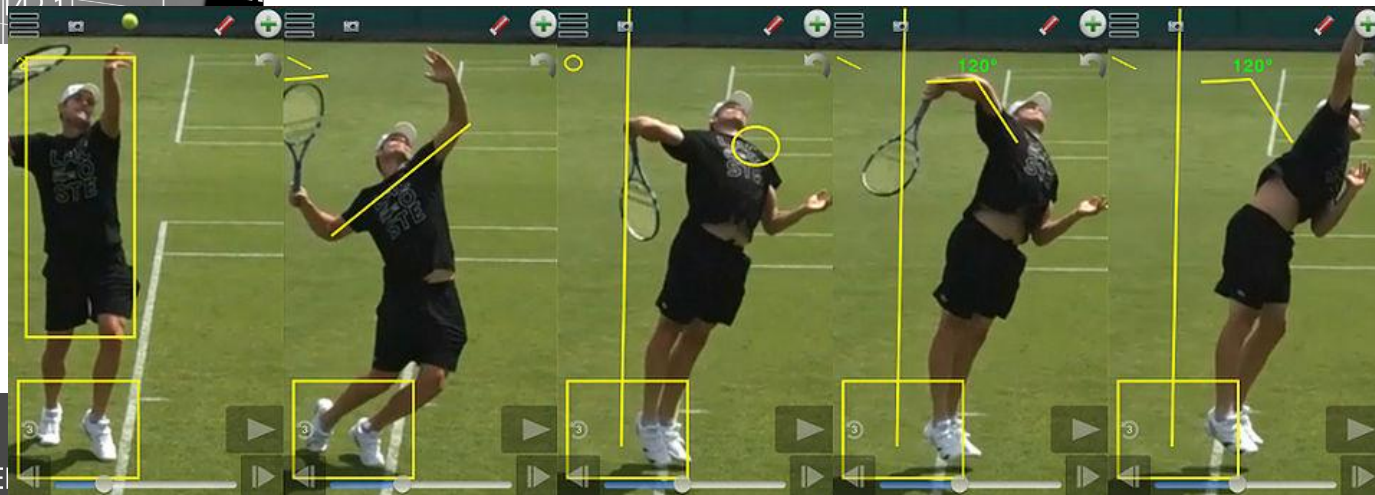


Stock Price



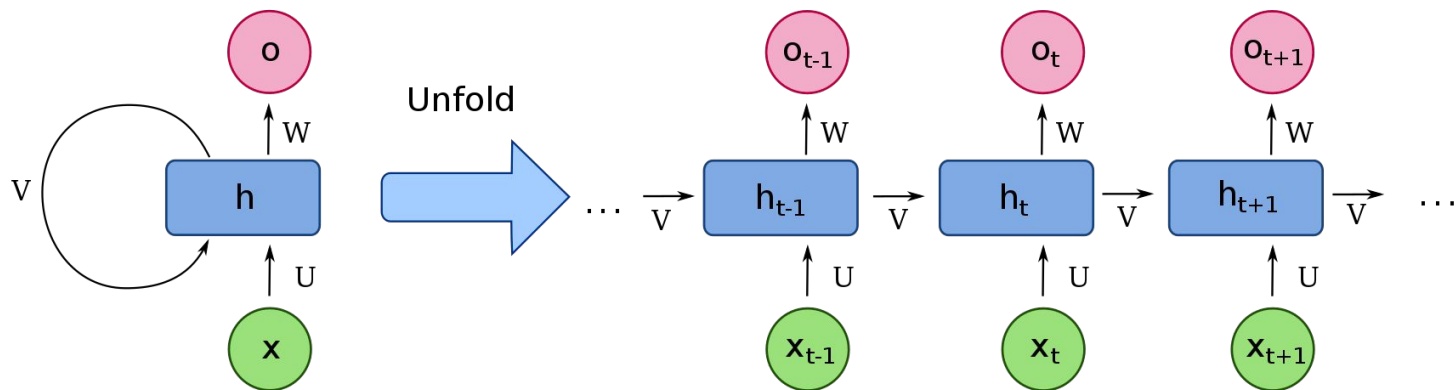
Speech

Video



RECURRENT NEURAL NETWORK (RNN)

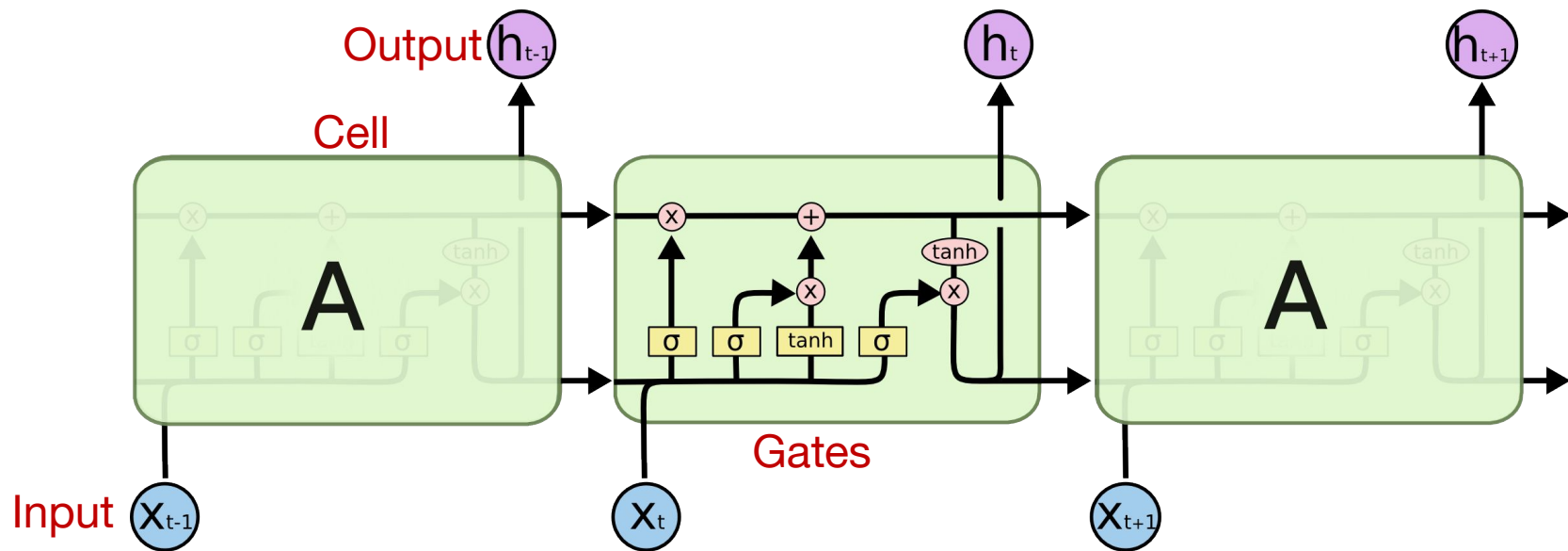
- Can model sequences and time-dependent signals
- Have cyclic connections that feed previous activations as part of input back to network
 - Allows for temporal contextual information to be stored
 - Predictions at current time step depend on current input and previous predictions
 - Context required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



LSTM

- **Issues with RNN training**
 - Vanishing gradients and exploding gradients
 - Weight of contextual input decays or blows up
 - Thus, contextual info that can be learned is limited in practice
- **LSTM**
 - Type of RNN
 - Addresses (some of) gradient issues with conventional RNN training

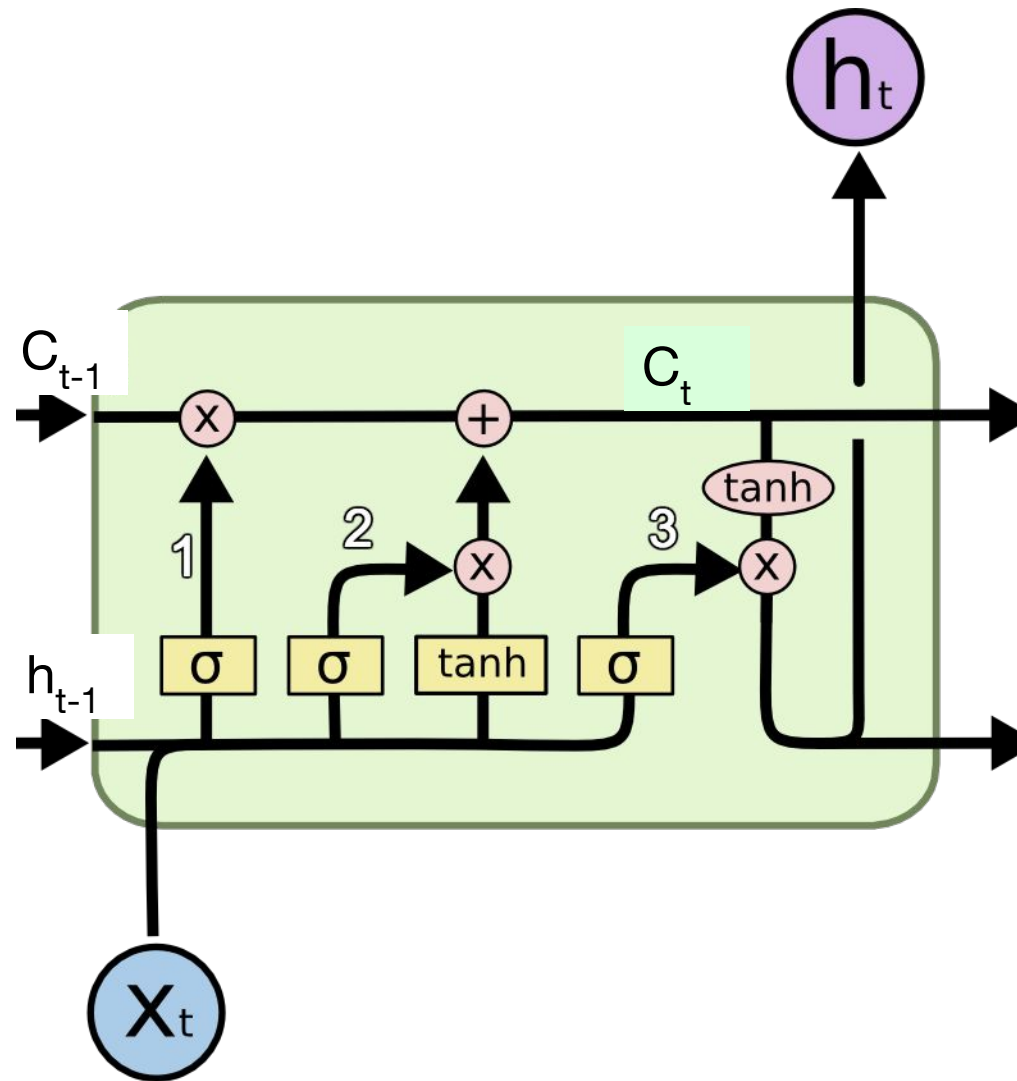
LSTM ARCHITECTURE



<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 CELL



X_t

Current input

C_{t-1}

Previous cell state
Long-term memory

h_{t-1}

Previous hidden state
Output from last cell
Working memory

h_t

Current output

1: forget gate

Removes info not relevant

2: input gate

Adds info to update cell
state

3: output gate

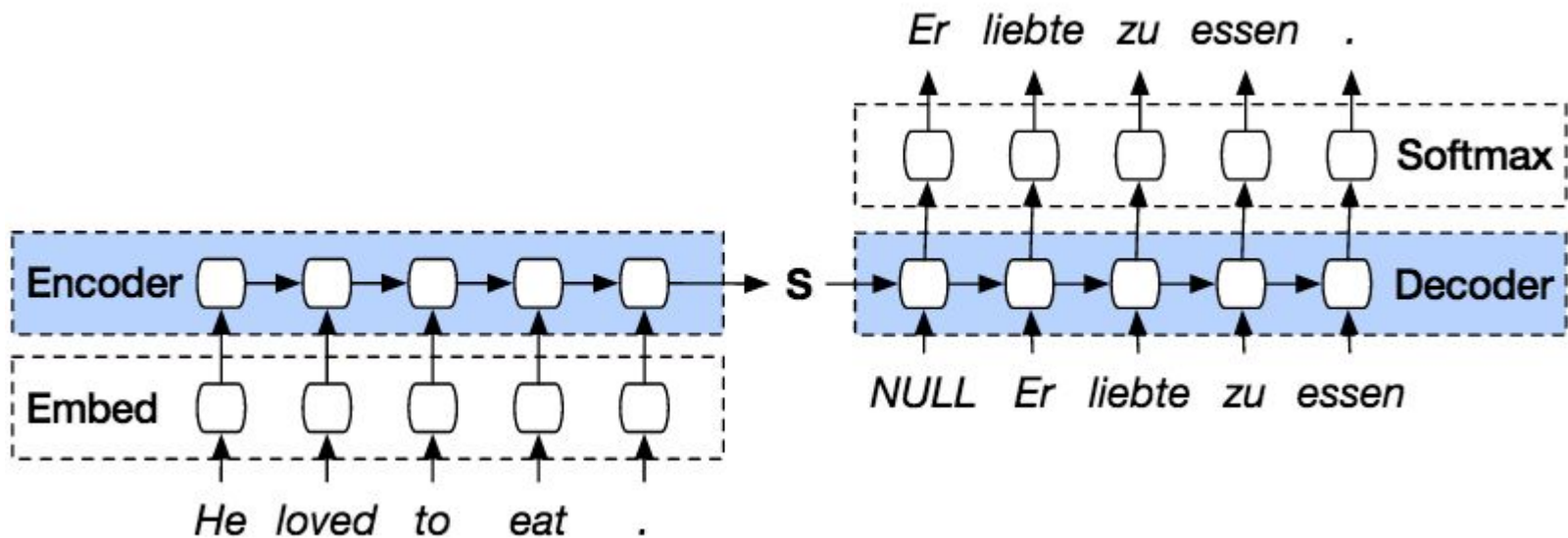
Selects useful info as
output

LSTM Applications

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

SEQ2SEQ

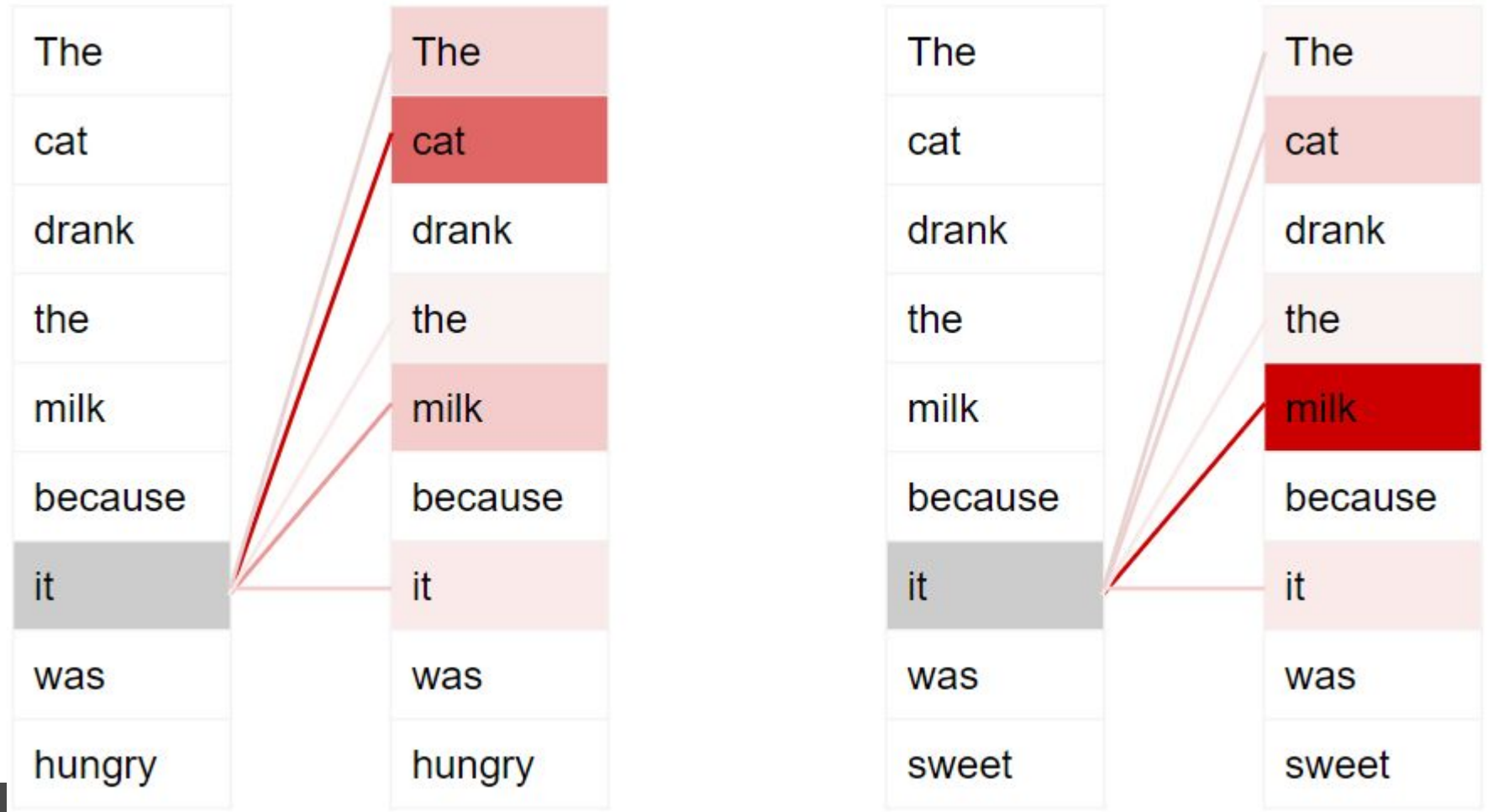
- **Converts input sequence to output sequence**
 - machine translation, question-answering
- **Encoder & decoder are RNNs**
- **Issue: Difficult to capture long-range dependencies**



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

ATTENTION MECHANISM

- For each part in sequence, attention is used to determine importance of other parts in sequence



TRANSFORMER

- Encoder-decoder model
- Uses only attention to capture relationships between words in sentence
- No recurrence or convolutions

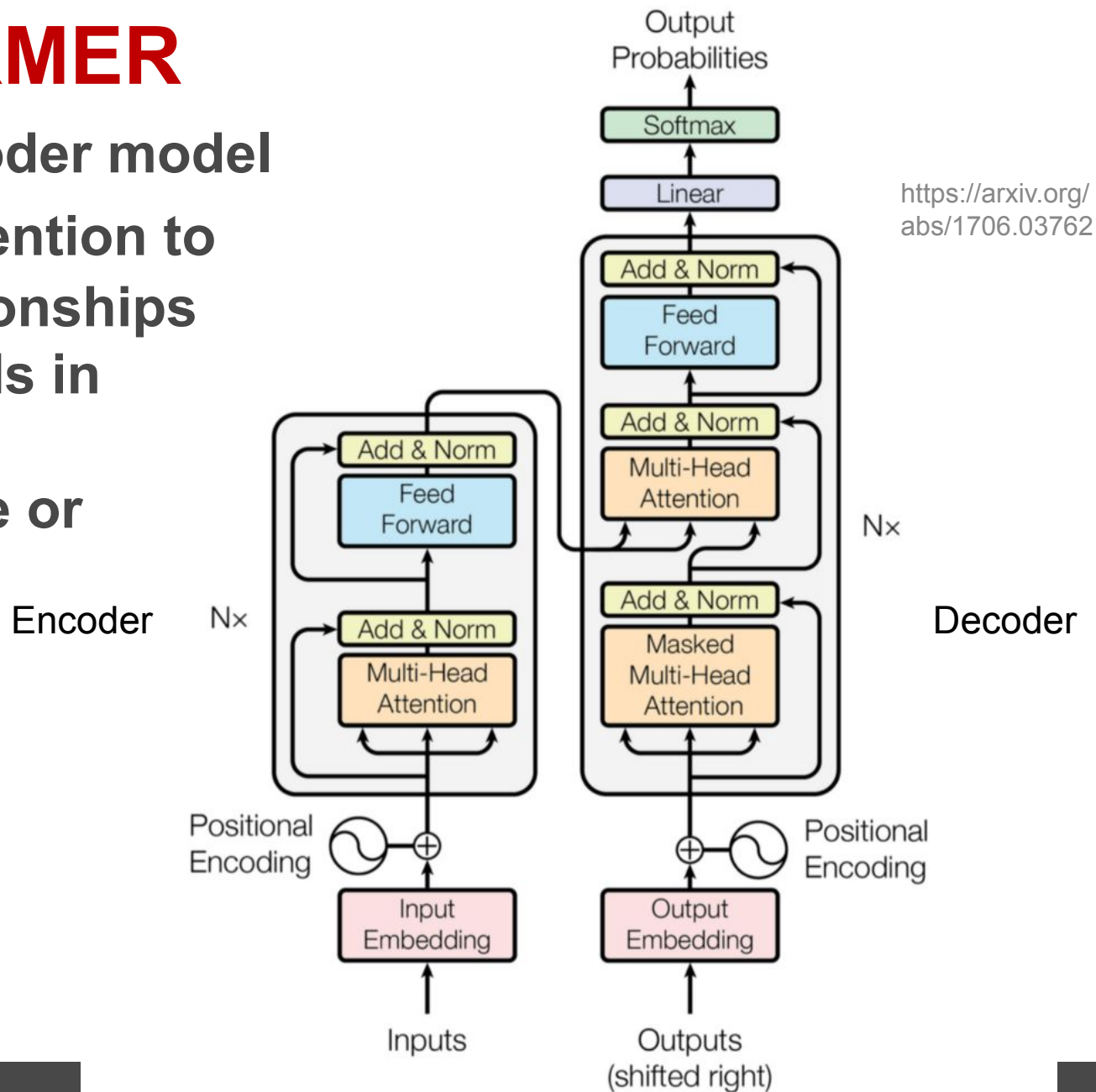
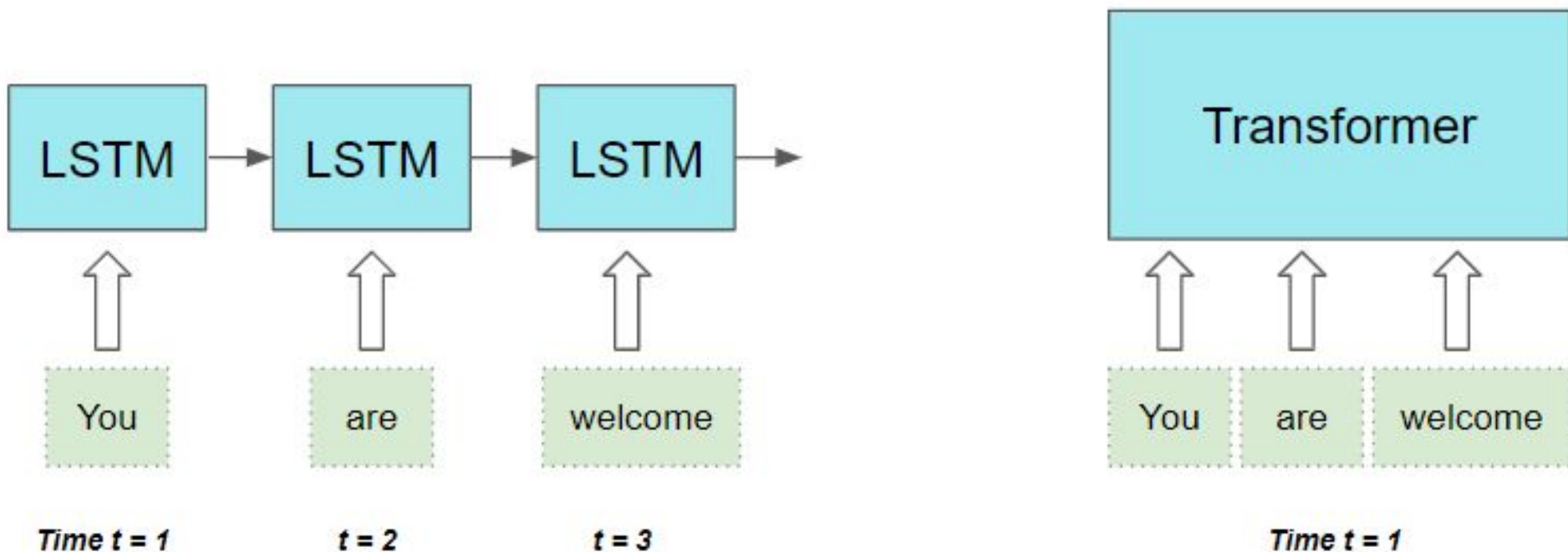


Figure 1: The Transformer - model architecture.

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.

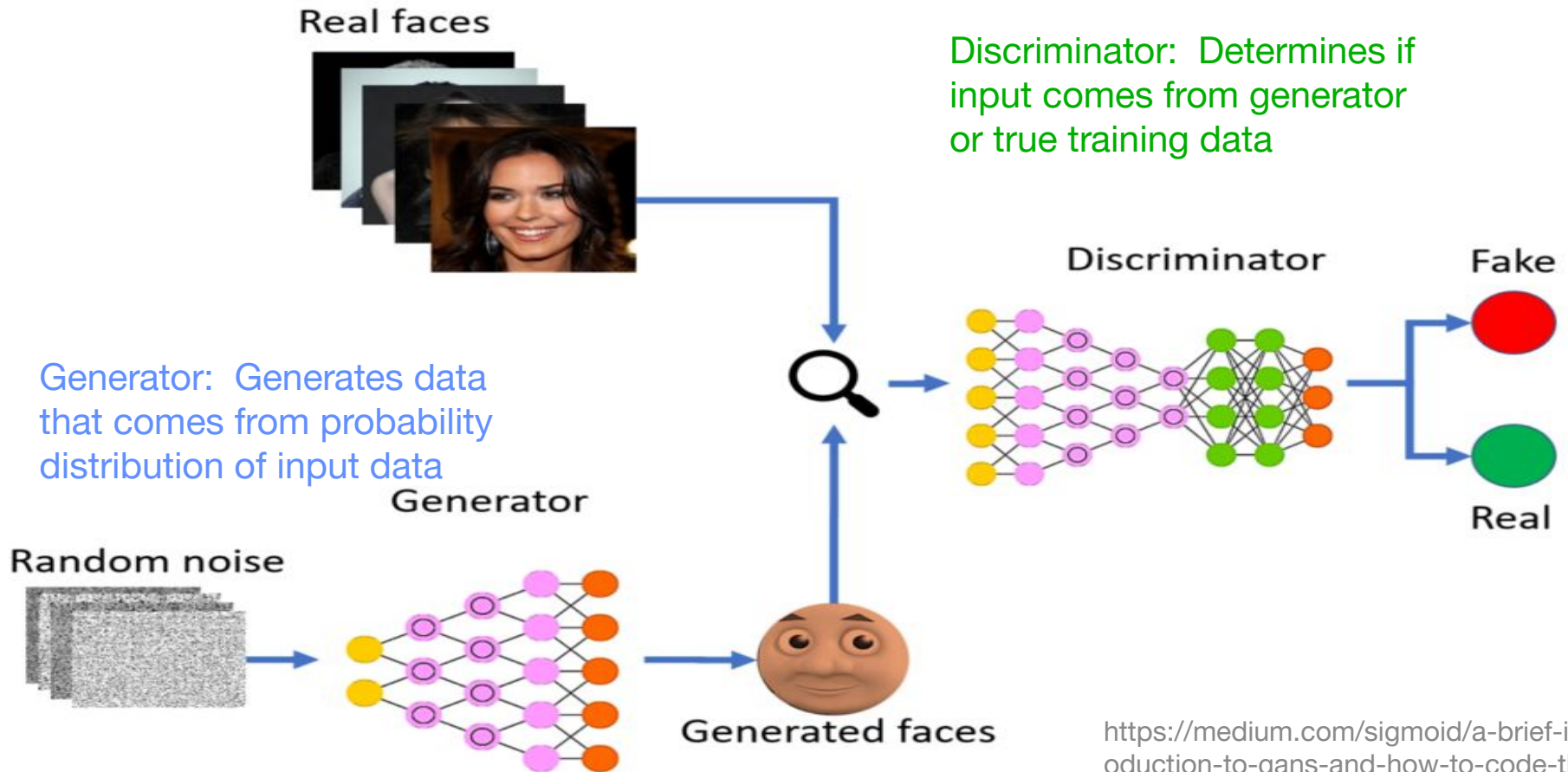
TRANSFORMER APPLICATIONS

- **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

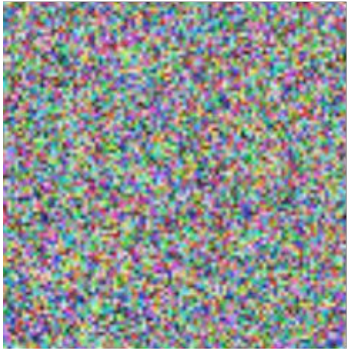
GAN ARCHITECTURE



<https://medium.com/sigmoid/a-brief-introduction-to-gans-and-how-to-code-the-m-2620ee465c30>

GAN APPLICATIONS

Noise $\sim N(0,1)$



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)

Monet ↔ Photos



Monet → photo

Zebras ↔ Horses



zebra → horse

Summer ↔ Winter



summer → winter

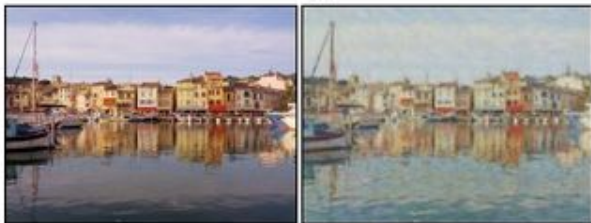


photo → Monet



horse → zebra



winter → summer

<https://junyanz.github.io/CycleGAN/>

GAN APPLICATIONS

- **Superresolution**

- Create high-resolution images from lower-resolution images

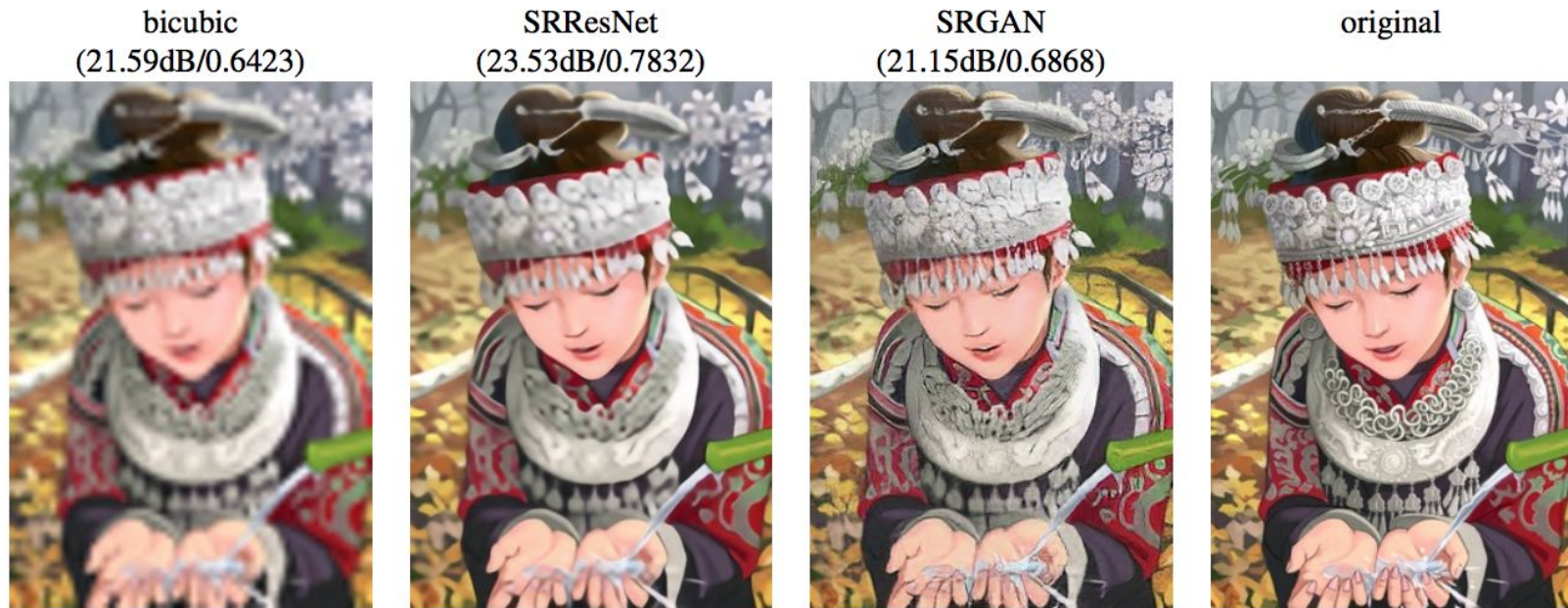


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

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

GAN APPLICATIONS

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

PYTHON DEEP LEARNING LIBRARIES

- **TensorFlow**

- <https://www.tensorflow.org/>
- ML framework developed by Google
- Keras: High-level NN API

- **PyTorch**

- <https://pytorch.org/>
- ML framework developed by Facebook

- **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

TOPICS COVERED

- **Deep Learning**
 - Deep neural networks have many layers
- **Deep Network Layers**
 - convolution, pooling, fully connected, etc.
- **Deep Learning Architectures**
 - CNN, auto-encoder, transformer, etc.
- **Deep Learning Libraries**
 - TensorFlow, PyTorch, cloud-based

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-net-works-gans/>