Week 9 | Lecture 10
Modern Architectures

Wan Fang
Southern University of Science and Technology

Today’s Agenda

- **Regularization for Deep Learning**
  - Regularization in General
  - Data Preprocessing
  - Dataset Augmentation
  - Early Stopping and Dropout
- **Modern Architectures**
  - Major Application of AI
  - AlexNet
  - YOLO
  - RNN
- **Exercise**

Regularization for Deep Learning

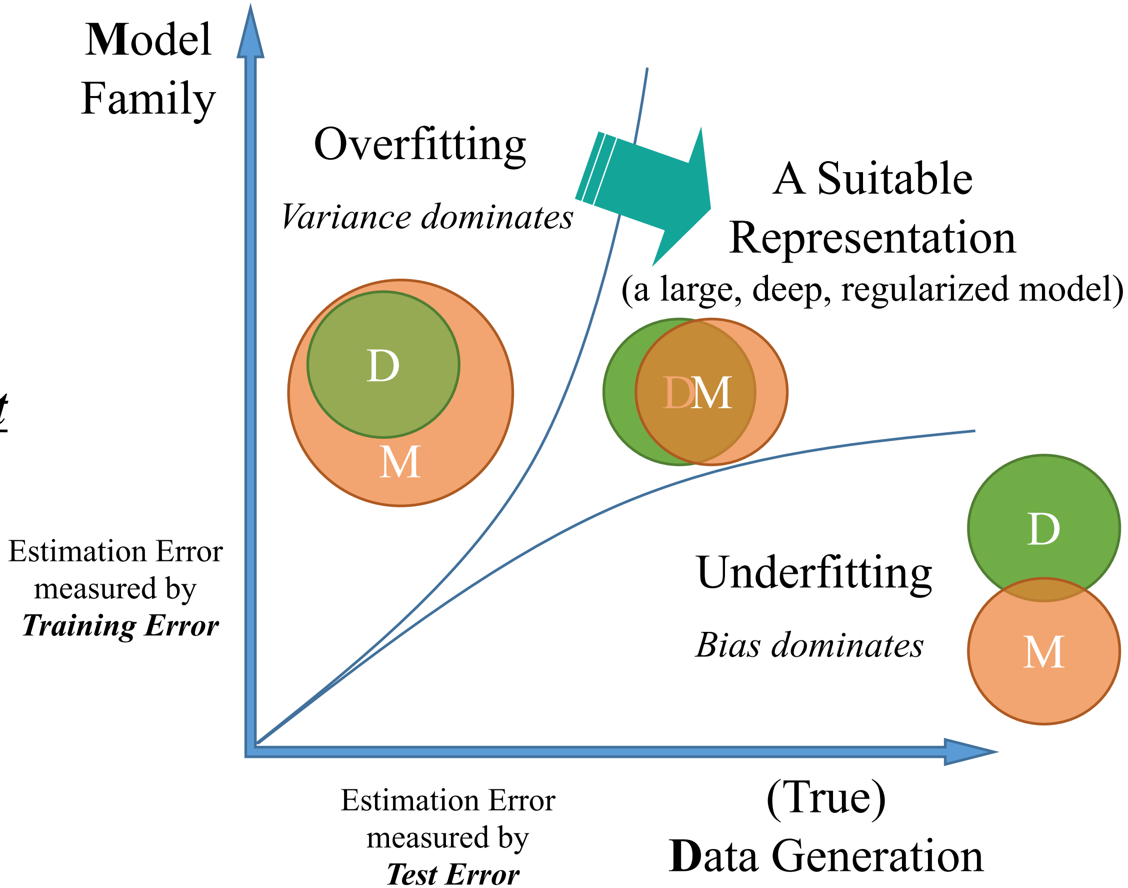
AncoraSIR.com

Regularization in General

Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

Regularization of a mismatch

- **Data generation**: almost never have access to the true data generation process
- **Model representation**: not sure if our model family covers the data generation or not
- More memorization capacity naturally tends to ***overfit***
  - Limited memorization capacity won't be able to learn the mapping, causing ***underfitting***
- The best fitting model is a large model that has been ***regularized appropriately***



AncoraSIR.com

Data Preprocessing

How to preprocess image data?

Mean subtraction

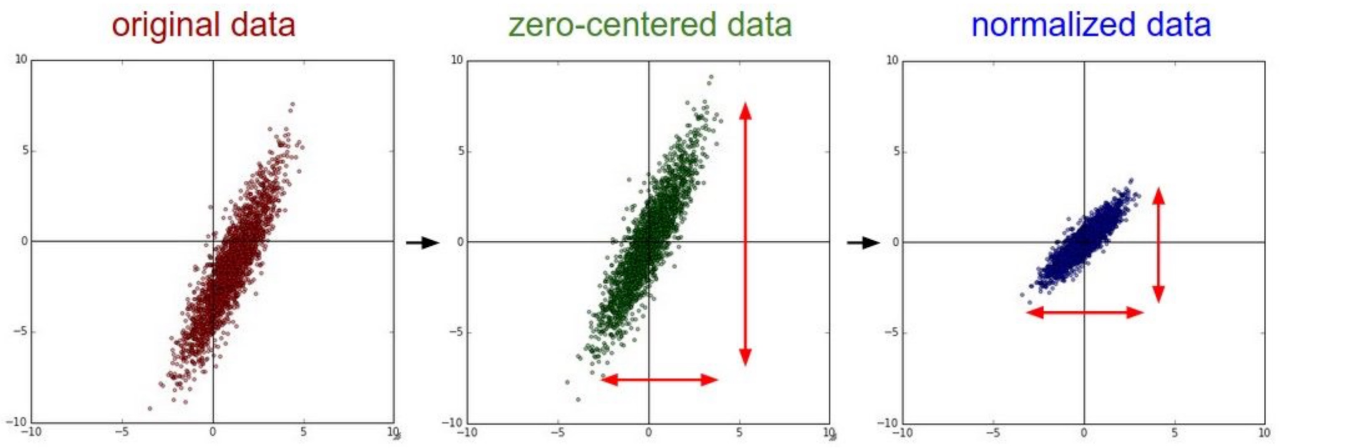
- Subtracting the mean across every individual feature in the data

Normalization

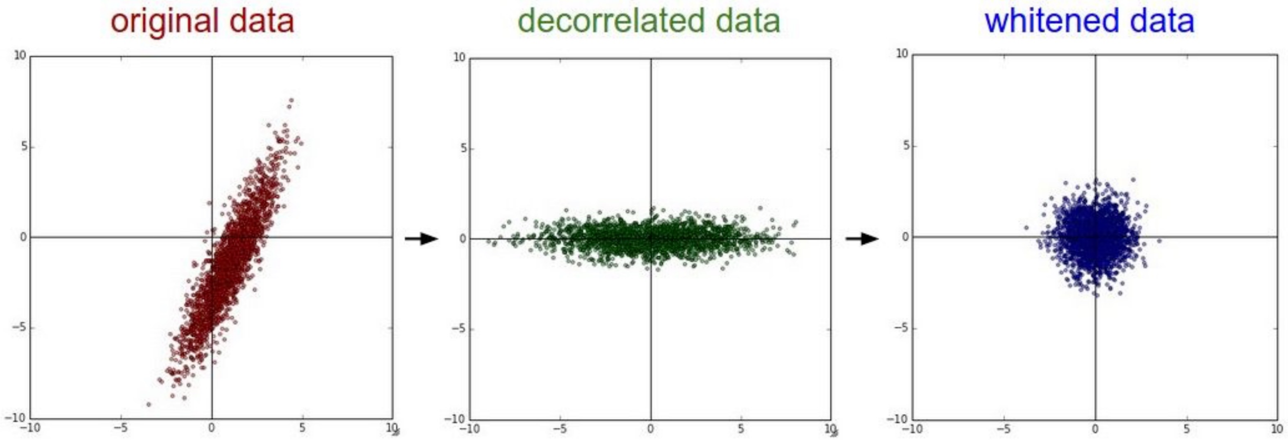
- Normalizing the data dimensions so that they are of approximately the same scale.
  - One is to divide each dimension by its standard deviation, once it has been zero-centered.
  - Another is to normalize each dimension so that the min and max along the dimension is -1 and 1 respectively.

PCA and Whitening

- In this process, the data is first centered as described above.
- Then, we can compute the covariance matrix that tells us about the correlation structure in the data




```
X -= np.mean(X, axis = 0)
X /= np.std(X, axis = 0)
```



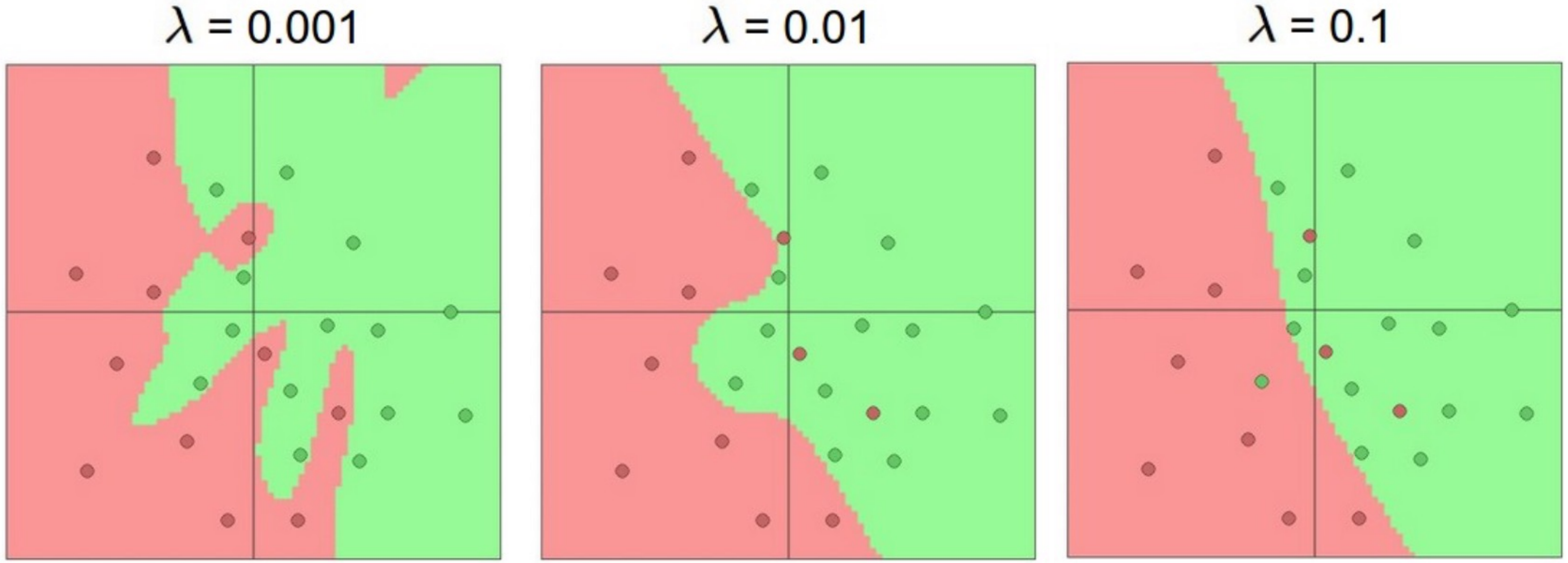
(data has diagonal covariance matrix)

(covariance matrix is the identity matrix)

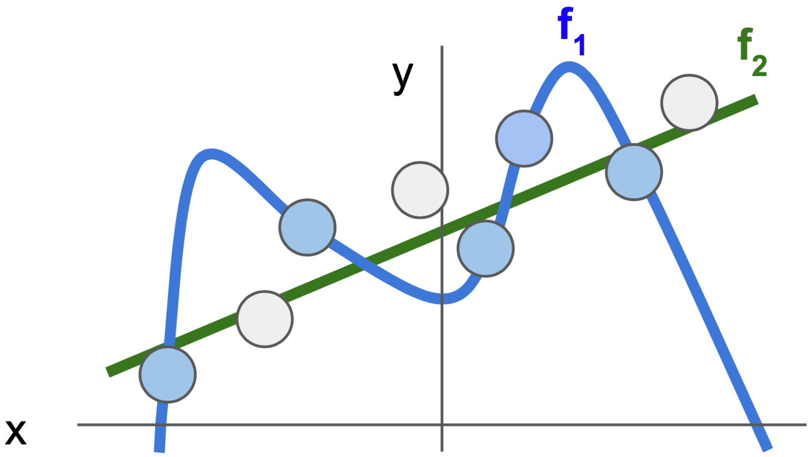
Norm Penalties as Constrained Optimization

Weight Regularization

- **Weight Regularization**
  - L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$
  - L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$



AncoraSIR.com



Regularization pushes against fitting the data *too* well so we don't fit noise in the data

$$L(W) = \boxed{\frac{1}{N}\sum_{1}^{N} L_i(\hat{y}_i, y_i)} + \boxed{\lambda R(W)}$$

λ as strength of Regularization (*hyperparameter*)

Data loss
Model predictions should match training data

Regularization
Prevent the model from doing too well on training data

Elastic Net (L1+L2)

- $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Dataset Augmentation

Create fake data and add it to the training set, particularly effective for object recognition

- We are always limited by the amount of data available for generalization
- Why Images?
  - high dimensional
  - include an enormous variety of factors, many of which can be easily simulated

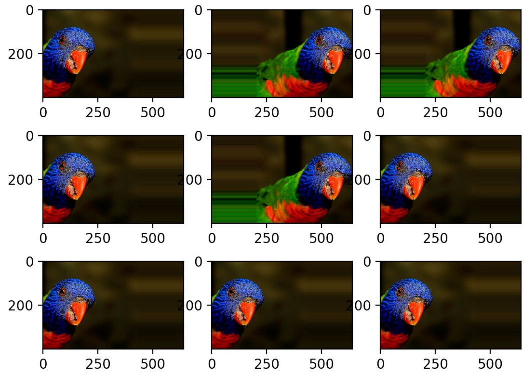
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

AncoraSIR.com

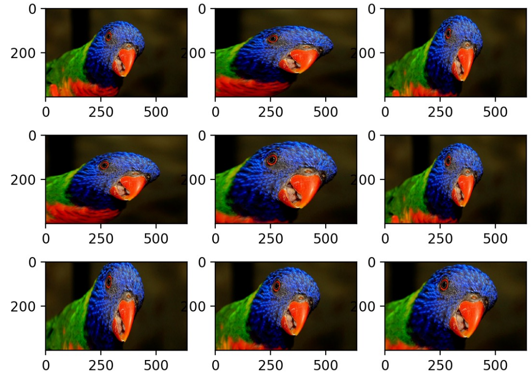
Dataset Augmentation

Create fake data and add it to the training set, particularly effective for object recognition

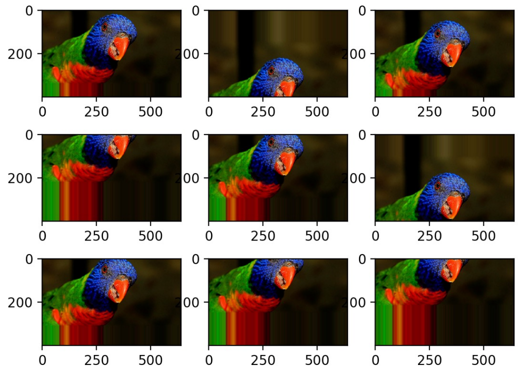
Horizontal Shift



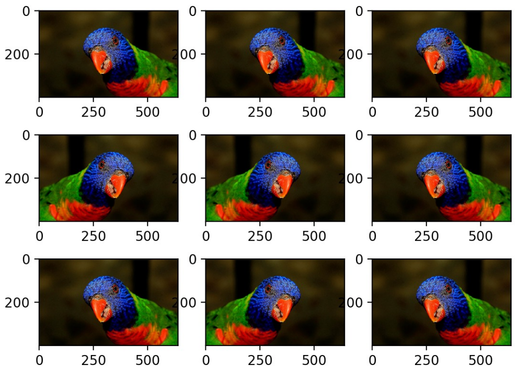
Random Zoom (Noise)



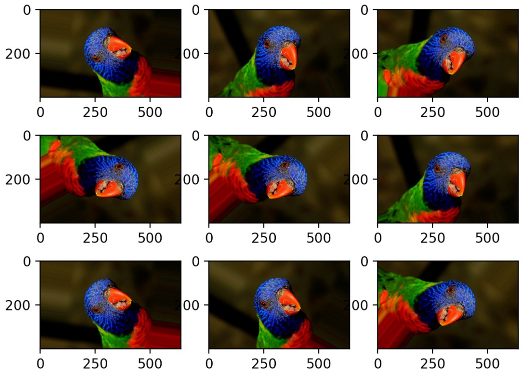
Vertical Shift



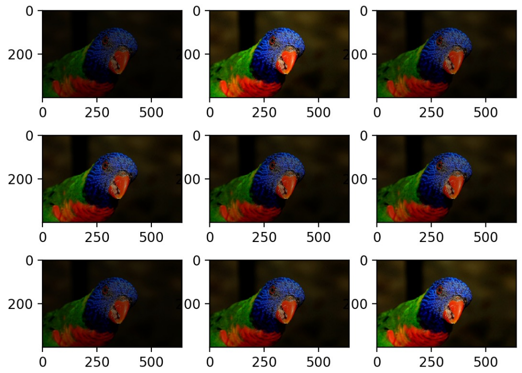
Flip with H/V Shift



Rotation Shift



Brightness Shift (Noise)

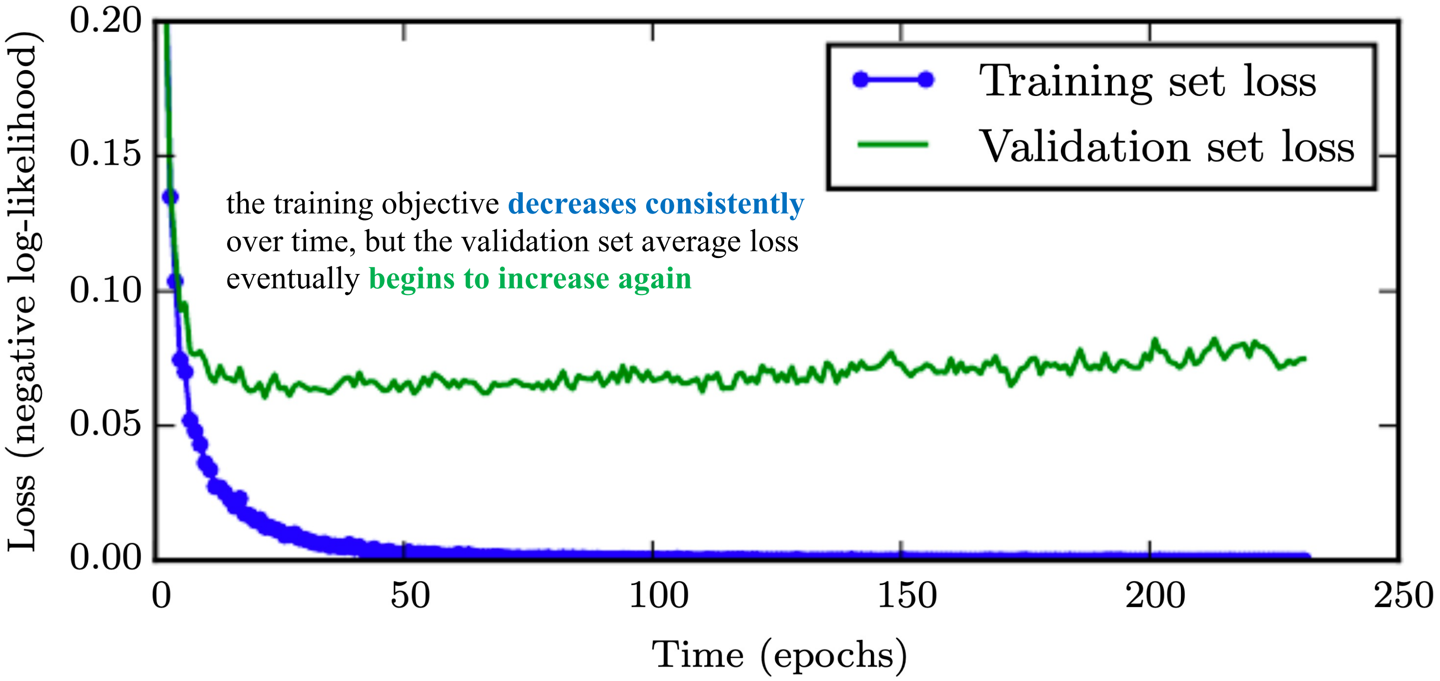


AncoraSIR.com

Early Stopping

Due to its simplicity and effectiveness, it is probably the most commonly used form of regularization in deep learning

- We can obtain a model with better validation set error (and thus, hopefully better test set error)
- Every time the error on the validation set improves, we store a copy of the model parameters
- As a very efficient hyperparameter selection algorithm
  - The number of training steps is just another hyperparameter



Learning curves showing how
the negative log-likelihood loss changes over time

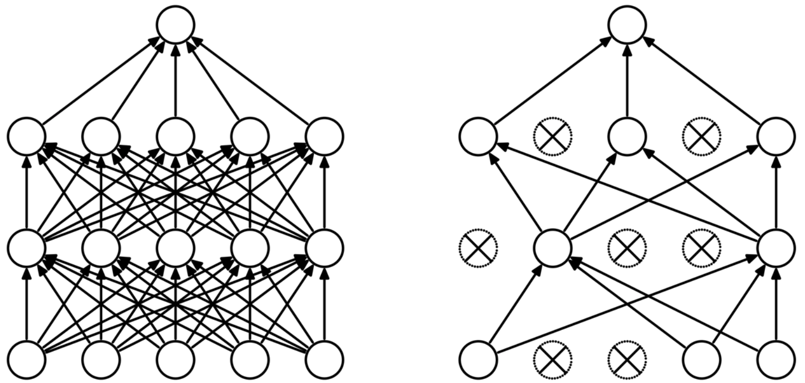
(***epochs***: the number of training iterations over the dataset)

AncoraSIR.com

Dropout

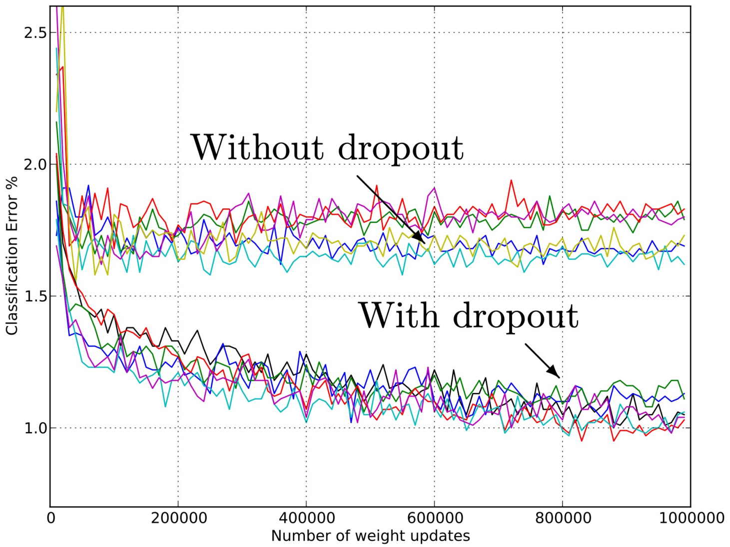
Randomly drop units (along with their connections) from the neural network during training

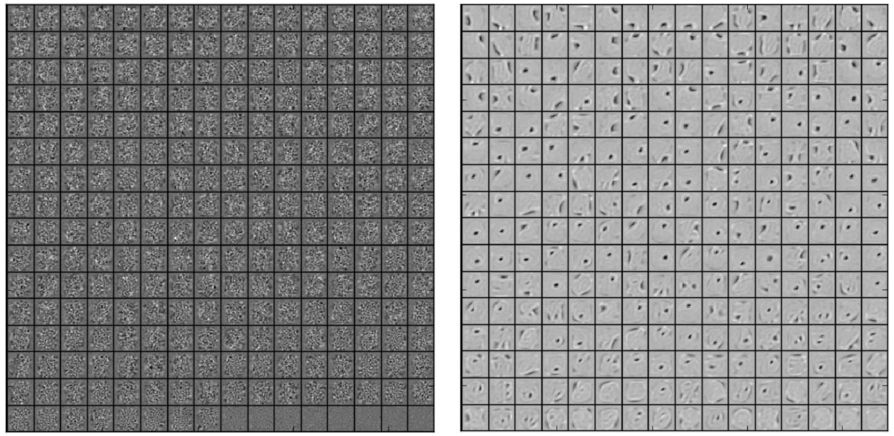
- The dropout rate
  - The fraction of the features that are zeroed out;
  - Usually set between 0.2 and 0.5.
- Dropout improves the performance of neural networks on supervised learning tasks significantly



(a) Standard Neural Net (b) After applying dropout.

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5×240) units	0.94
DBN + finetuning (Hinton and Salakhutdinov, 2006)	Logistic	500-500-2000	1.18
DBM + finetuning (Salakhutdinov and Hinton, 2009)	Logistic	500-500-2000	0.96
DBN + dropout finetuning	Logistic	500-500-2000	0.92
DBM + dropout finetuning	Logistic	500-500-2000	**0.79**





(a) Without dropout (b) Dropout with $p = 0.5$.

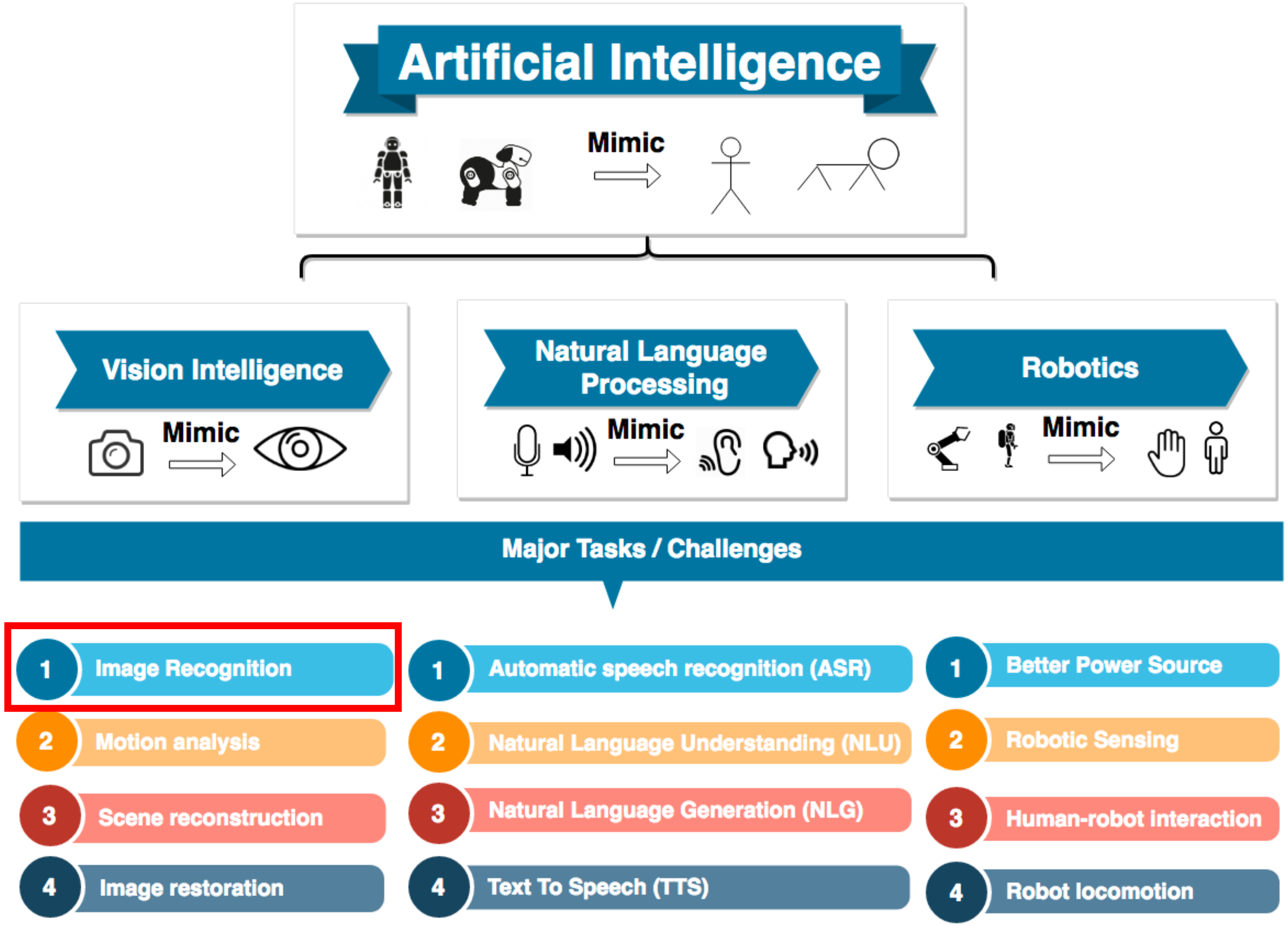
AncoraSIR.com http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

SUSTech Southern University of Science and Technology

Modern Architectures

AncoraSIR.com

Major Application of AI

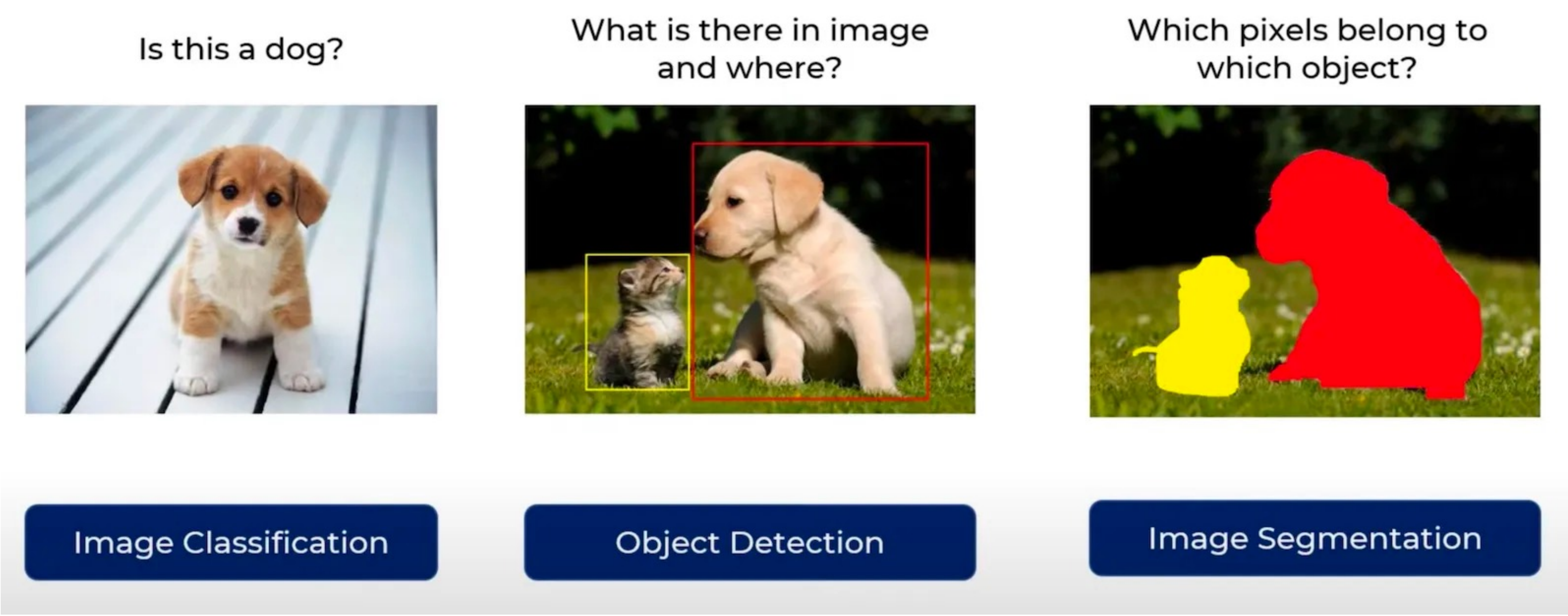


[Source]

AncoraSIR.com

SUSTech Southern University of Science and Technology

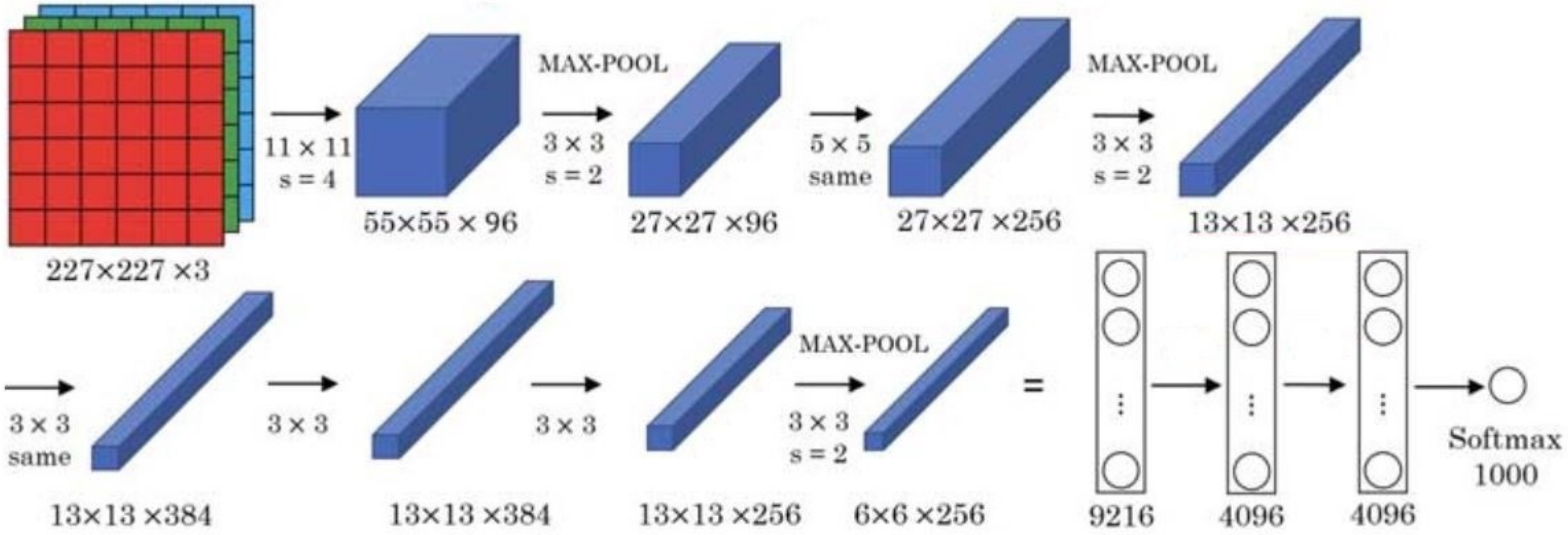
Image Recognition



Source: [codebasics]

AncoraSIR.com

AlexNet - Architecture



AncoraSIR.com

AlexNet - Application

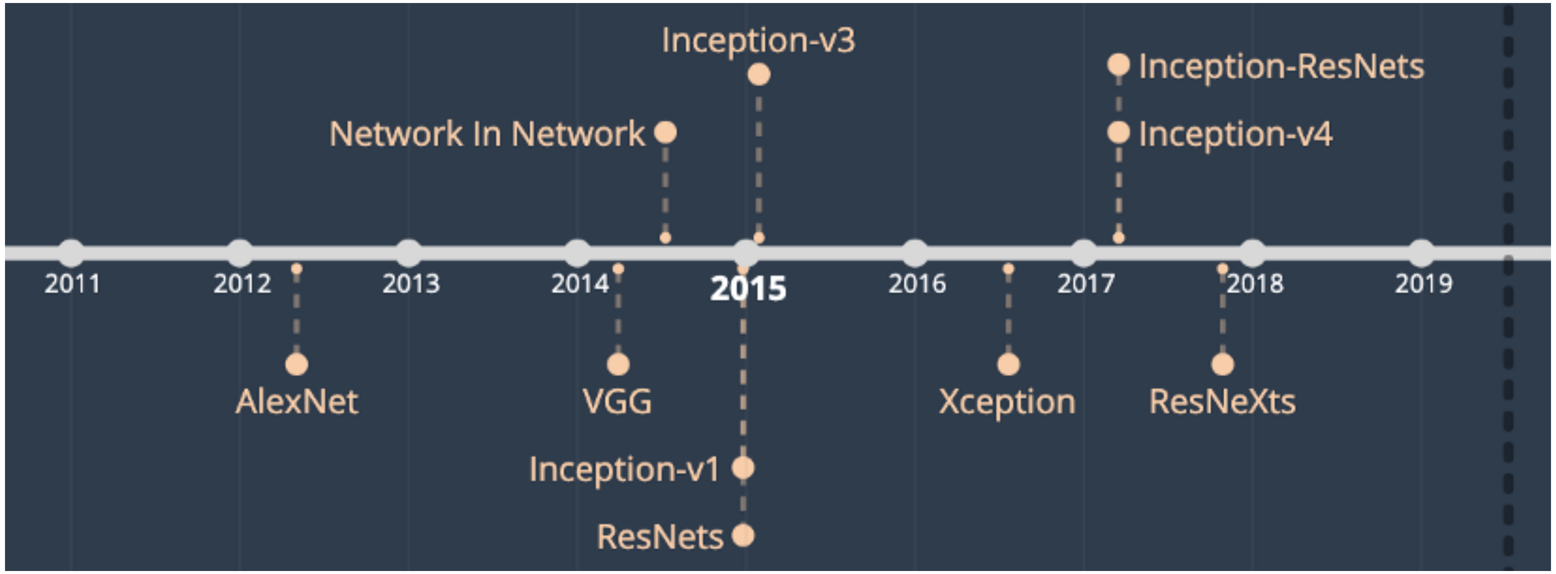
Finetune AlexNet

- Downloaded the train.zip file from
  - the [Kaggle Dogs vs. Cats Redux Competition].

AncoraSIR.com

Some Notes

- Why still Alexnet now?
  - Simple yet works quite well on many of the projects in our lab: ***classify rotation angles, recognize good grasp position, recognize certain patterns to trigger robot actions***
  - No reason to switch to any of the more heavy-weight models.
  - **Outperforms more complex model like VGG and Inception when the training data is small in size.**



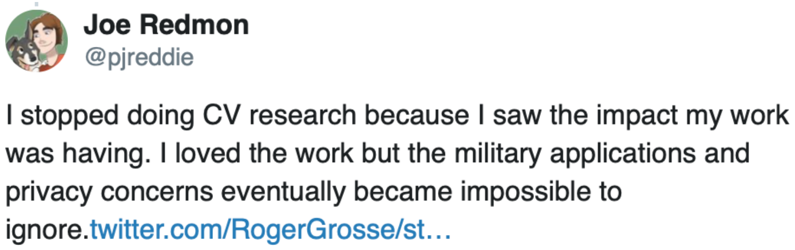
AncoraSIR.com


Object Detection
starring
YOLOv3

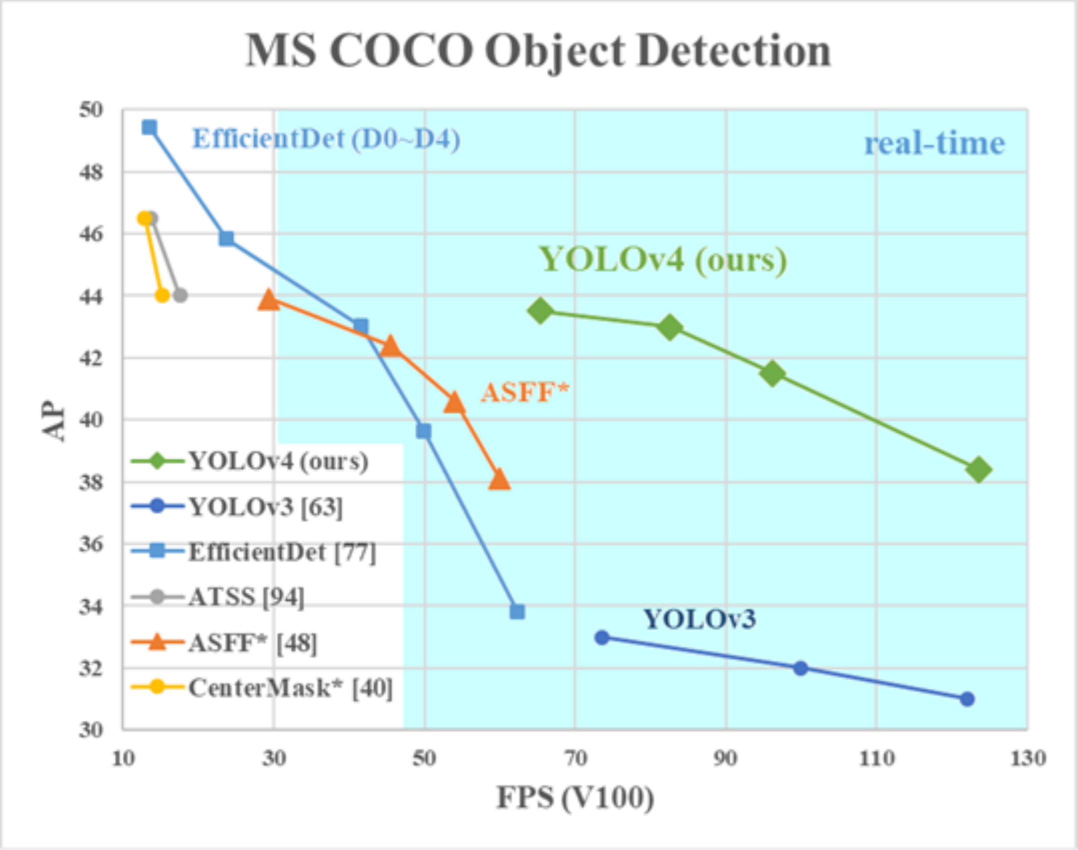
YOLO

History

- **You only look once (YOLO)** is an object detection system targeted for real-time processing.



- Evolvement
  - YOLO: 2015, “ [You Only Look Once: Unified, Real-Time Object Detection](#) ”
  - YOLOv2: 2017, “[YOLO9000: Better, Faster, Stronger](#)”
  - YOLOv3: 2018, “[YOLOv3: An Incremental Improvement](#)”
  - YOLOv4: 2020, “[YOLOv4: Optimal Speed and Accuracy of Object Detection](#)”



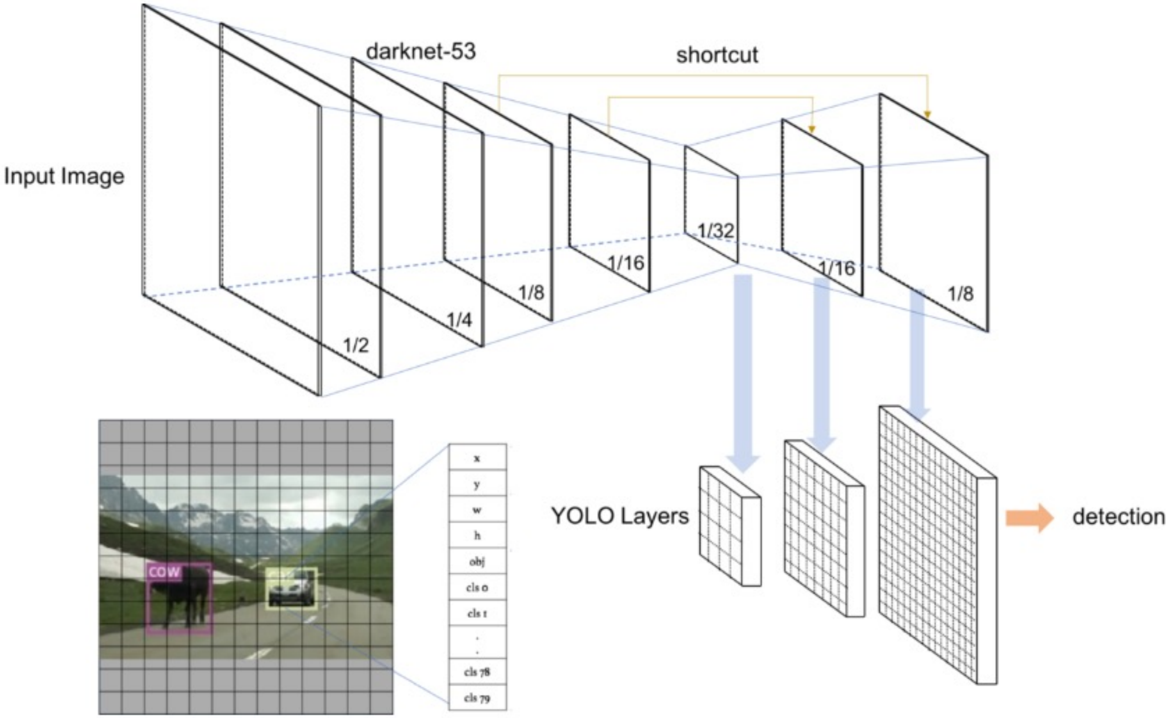
AncoraSIR.com

Architecture of pipeline

YOLOv3

- YOLO is a fully convolutional network (FCN) only convolutional layers.

1. A feature extracting network: darknet-53
   - 3 × 3 and 1 × 1 filters with skip connections
   - final feature map has 1/32 times smaller
2. Upsampling network
   - 1/32, 1/16, 1/8 of the input image
3. YOLO layers
   - Makes detections at three different scales.
   - capable of capturing **large/smaller** objects



AncoraSIR.com

More Details on Architecture

- Backbone Network : darknet53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Anco

- Upsampling network and YOLO layers

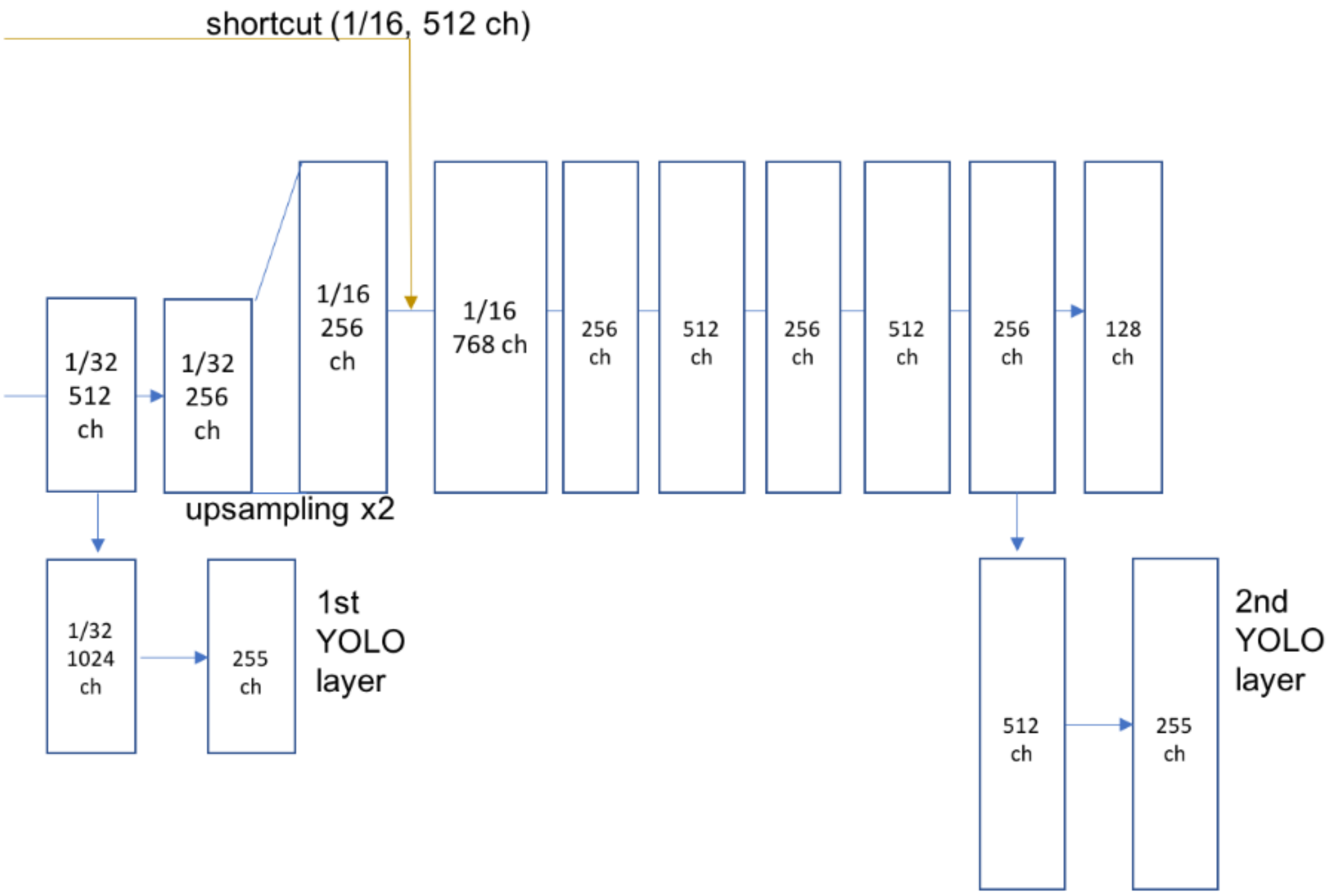


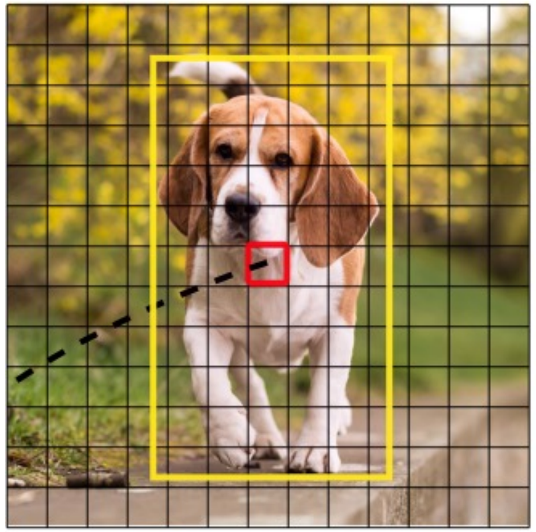
Fig. 2 Upsampling network and YOLO layers. The rectangles stand for feature maps.

Architecture

Interpreting the Output

- YOLOv3 makes detections at three different scales
- The detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes

416x416

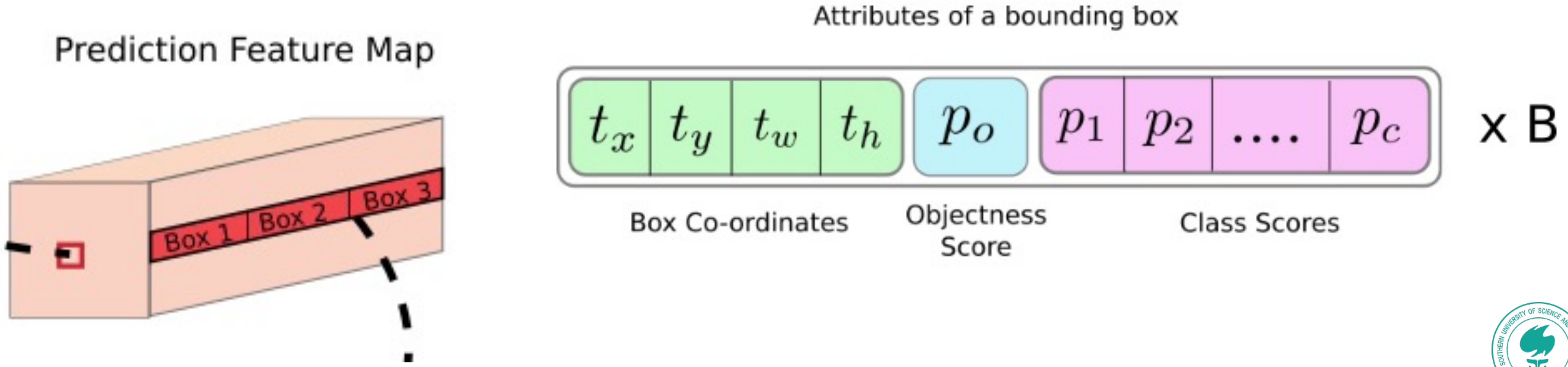


AncoraSIR.com

1/32: 13x13

Number of channel

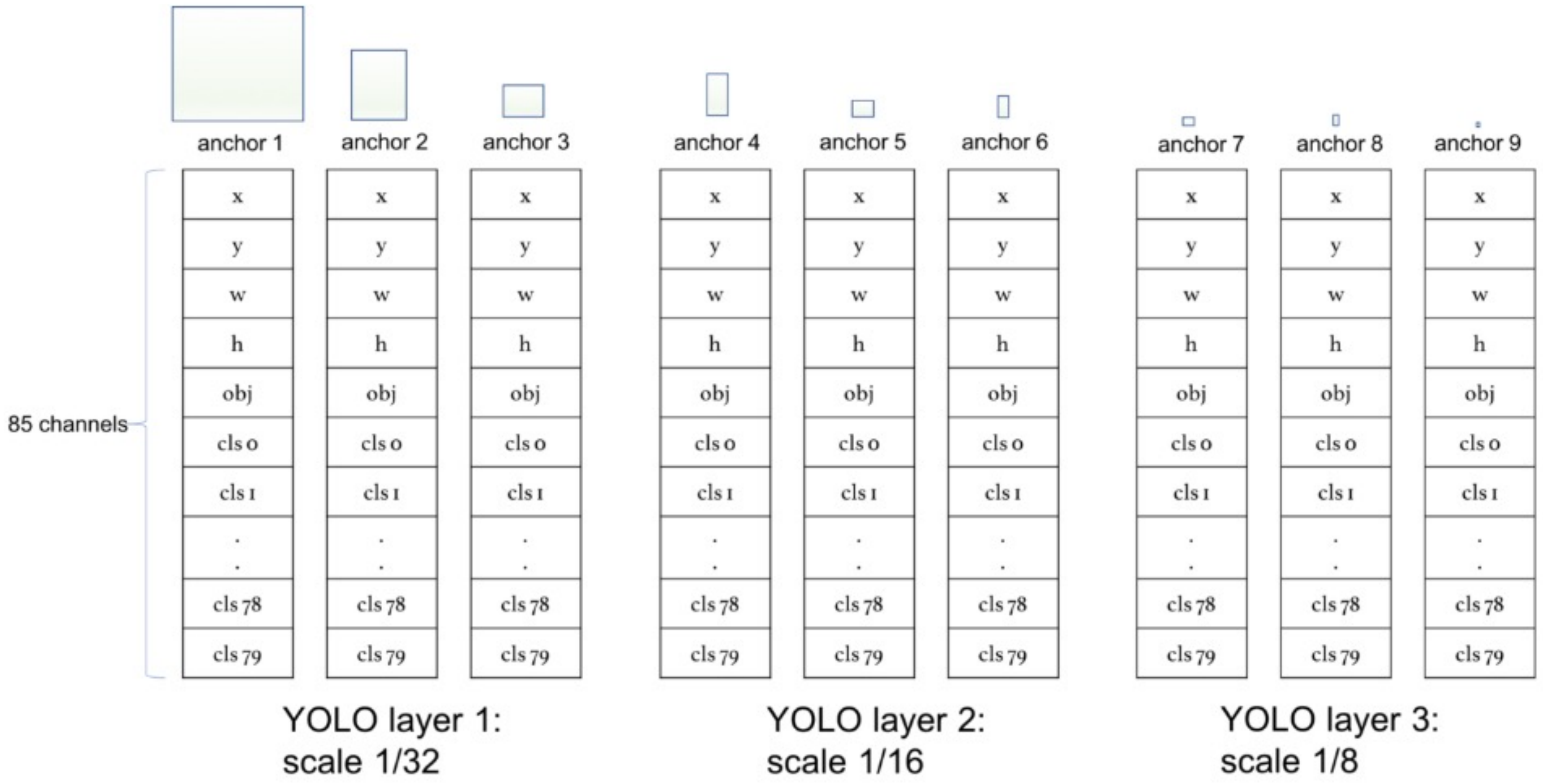
B x (5 + C)



Source

Choice of anchor boxes

- YOLO v3 uses 3 anchor boxes for each scale.
- With COCO 80 classes, Number of channel = B x (5 + C) = 3 x (5+80)



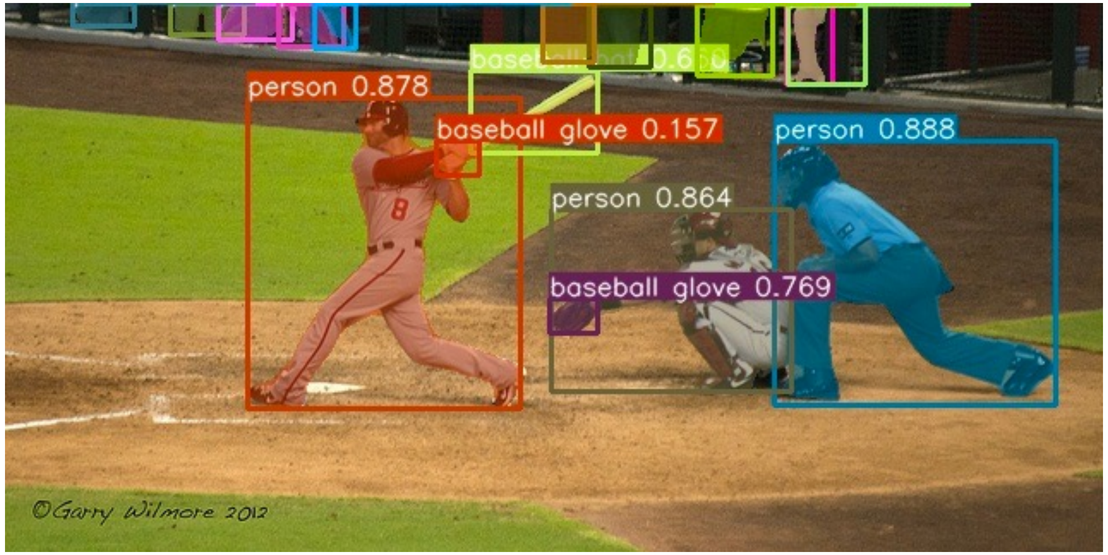
AncoraSIR.com

Summary

- YOLOv3 detects objects of different sizes at three YOLO layers.
- Each YOLO layer has grids of different resolution and three anchors with different shapes.
- Each anchor of one grid has the following information: box center location, box size, objectness likelihood, and class probability.
- YOLO is fast because it does not depend on regional proposal and run the neural network forward once on an image.

Resource

- YOLO v3 and v4 in C++: https://github.com/AlexeyAB/darknet
- YOLO is supported in Nvidia DeepStream: https://news.developer.nvidia.com/deepstream-sdk-4-now-available/
- YOLO v7 in Python and Pytorch: https://github.com/WongKinYiu/yolov7



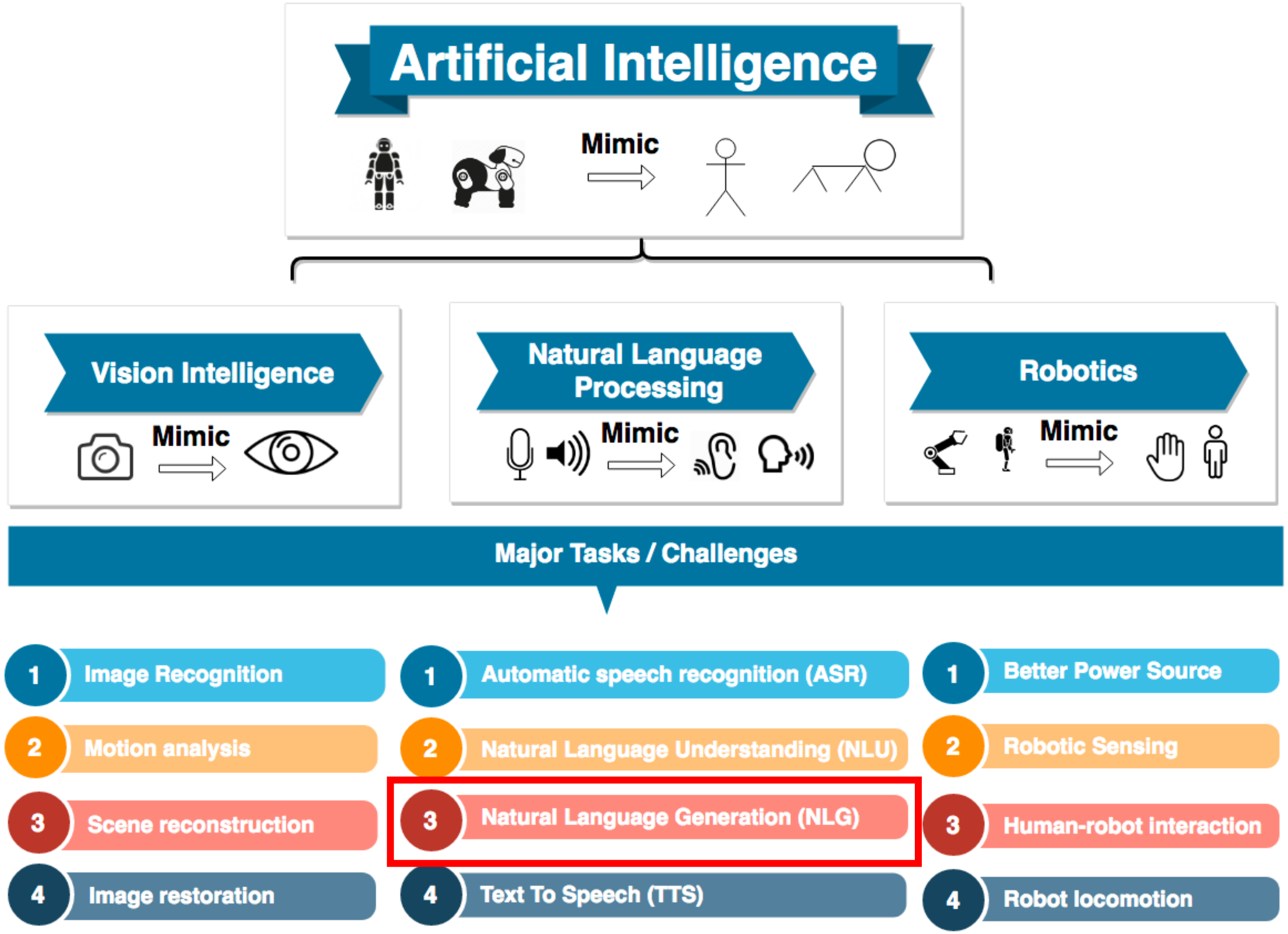
AncoraSIR.com

Application: Real-time Redaction

source

AncoraSIR.com

Major Application of AI



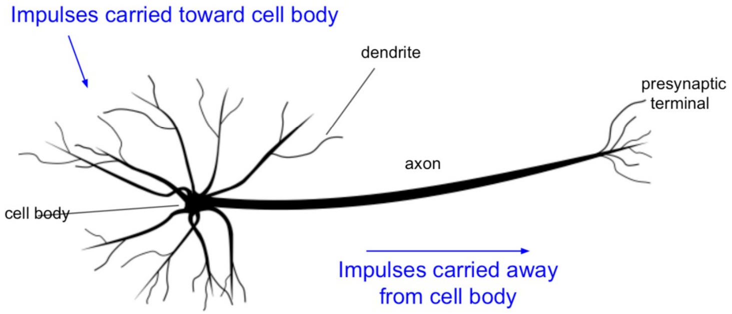
[Source]

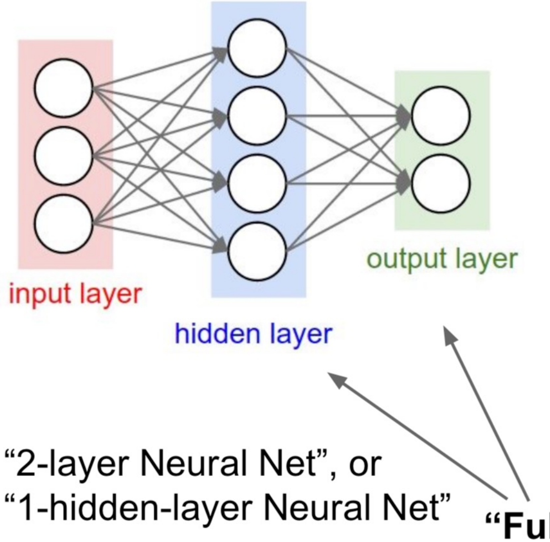
AncoraSIR.com

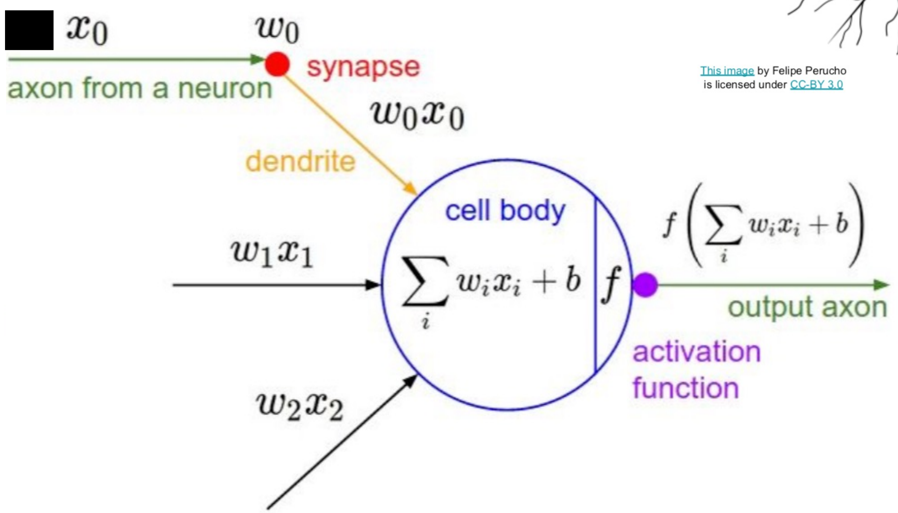
Recurrent Neural Networks

Why Recurrent Neural Networks?

- RNN were created because there were a few issues in the feed-forward neural network:
  - Cannot handle sequential data
  - Considers only the current input
  - Cannot memorize previous inputs

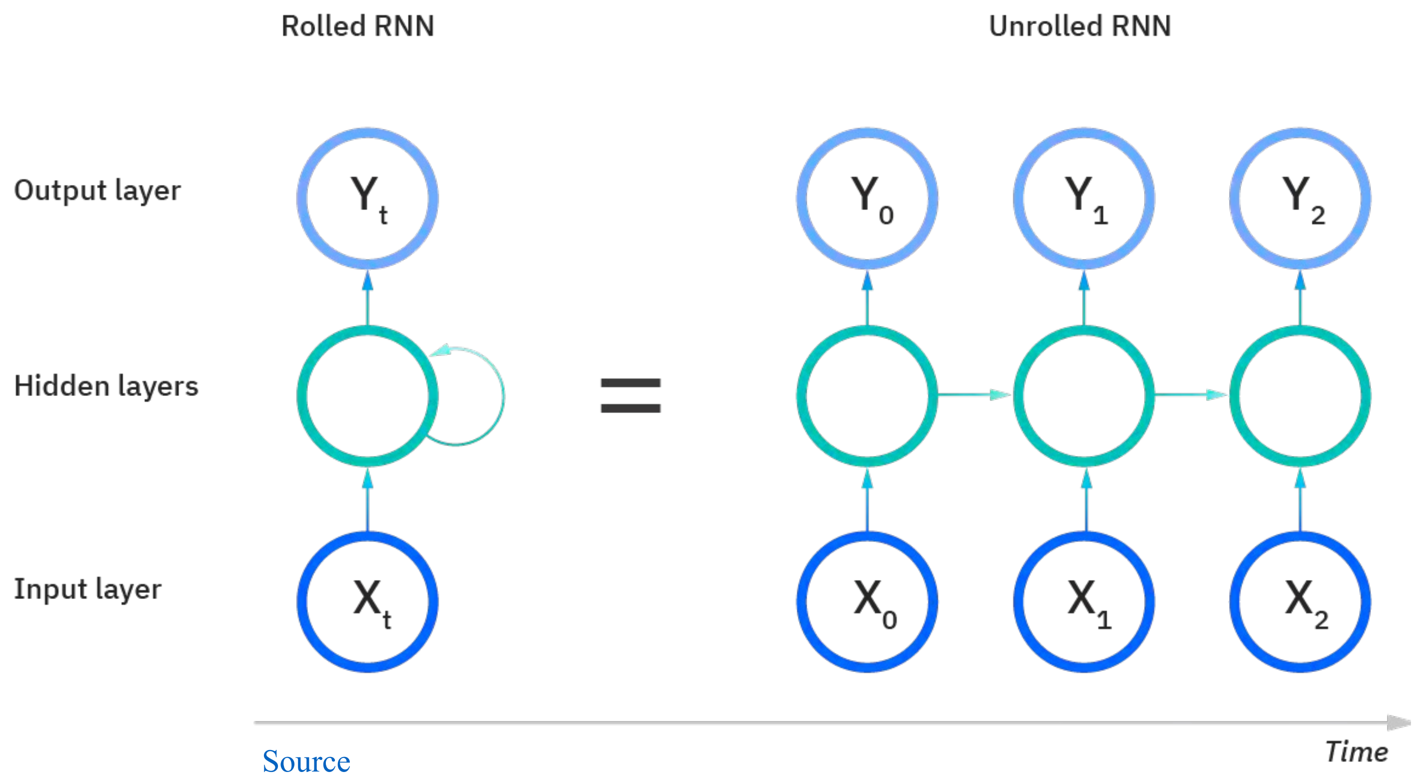






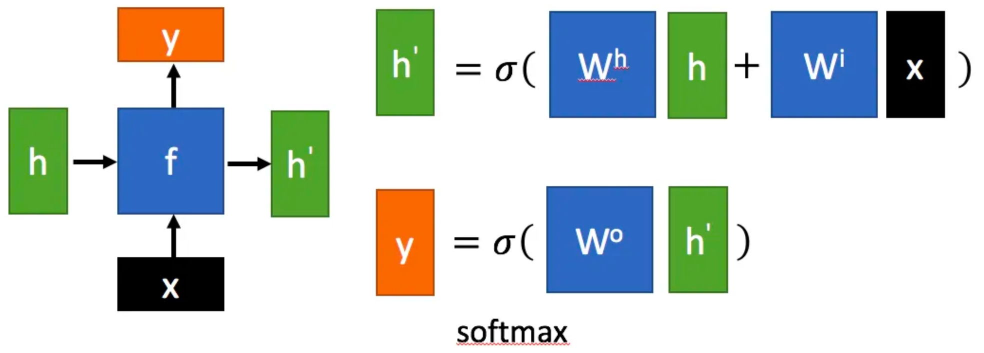
AncoraSIR.com

What Is a Recurrent Neural Network (RNN)?



Source

- Given function f: $h', y = f(h, x)$



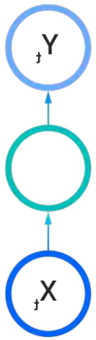
- RNN are distinguished by their "memory" as they take information from prior inputs to influence the current input and output.
- RNN share parameters across each layer of the network

AncoraSIR.com

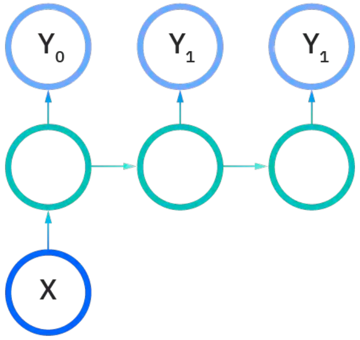
SUSTech Southern University of Science and Technology

Types of Recurrent Neural Network?

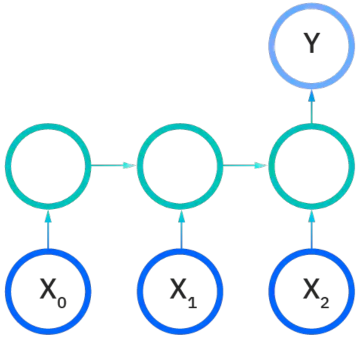
One-to-one:



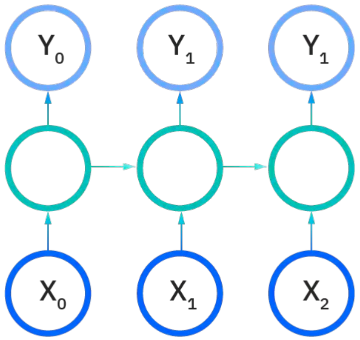
One-to-many:



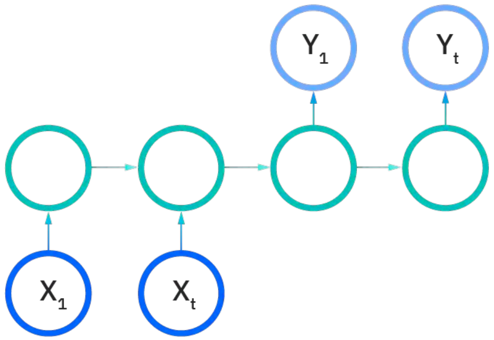
Many-to-one:



Many-to-many:



Many-to-many:

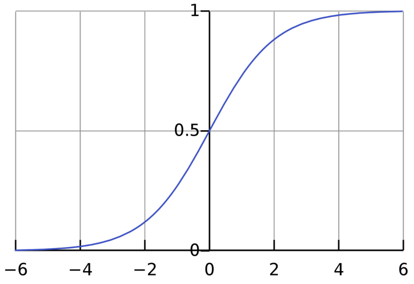


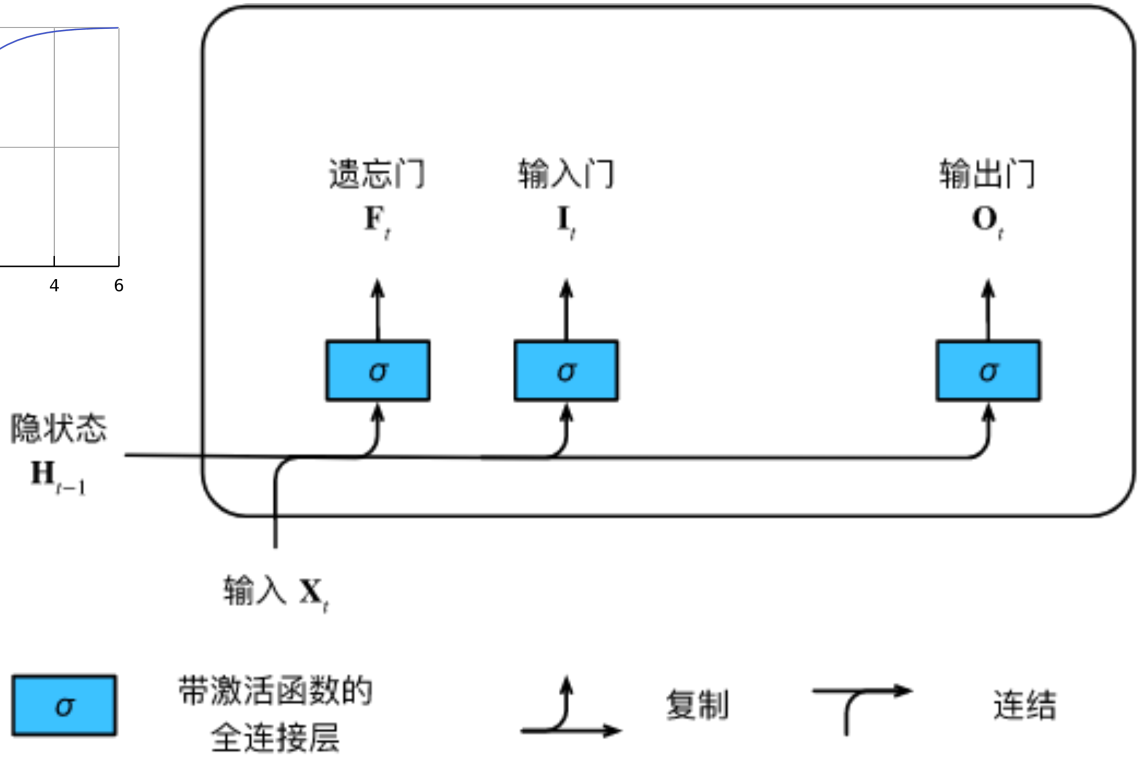
AncoraSIR.com

https://www.ibm.com/cloud/learn/recurrent-neural-networks

Variant RNN architectures

- Long Short-Term Memory Networks
  - LSTMs are a special kind of RNN — capable of learning long-term dependencies by remembering information for long periods is the default behavior.





f_t = forget gate
Decides which information to delete that is not important from previous time step

i_t = input gate
Determines which information to let through based on its significance in the current time step

o_t = output gate
Allows the passed in information to impact the output in the current time step

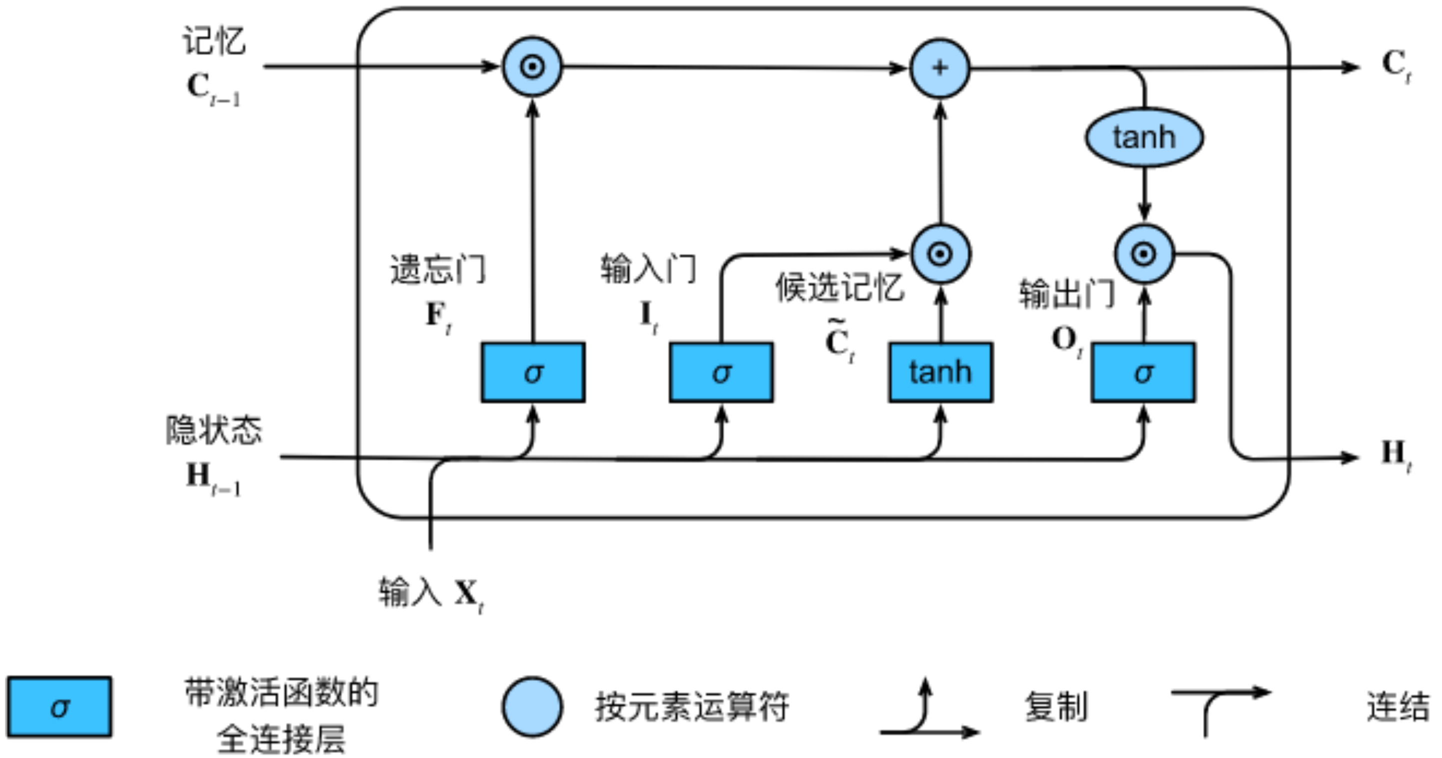
$$
\begin{aligned}
\mathbf{I}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i),\\
\mathbf{F}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f),\\
\mathbf{O}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o),
\end{aligned}
$$

AncoraSIR.com

Source

Variant RNN architectures

- Long Short-Term Memory Networks
  - LSTMs are a special kind of RNN — capable of learning long-term dependencies by remembering information for long periods is the default behavior.



$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t$$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t)$$

AncoraSIR.com

Generative Adversarial Networks

- GANs can be trained on the images of
  - humans to generate realistic faces.
  - cartoon characters for generating faces of anime characters as well as Pokemon characters.

AncoraSIR.com

SUSTech Southern University of Science and Technology

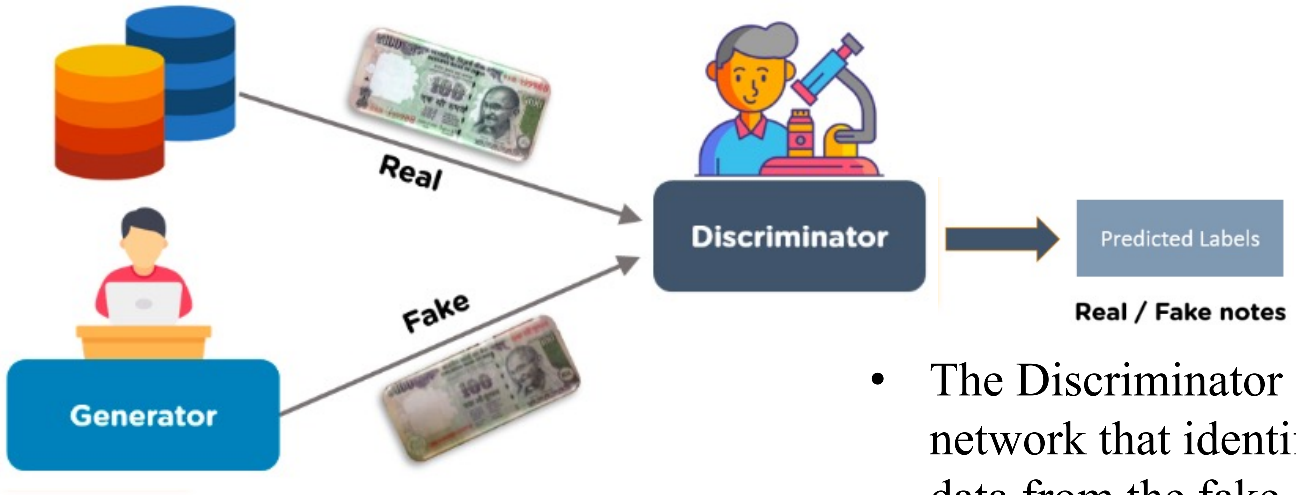
Text to Image

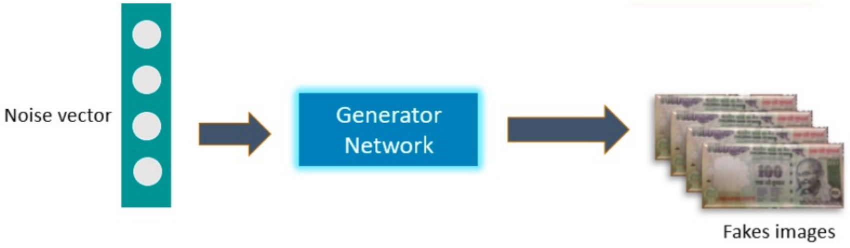
AncoraSIR.com

What are Generative Adversarial Networks?

- Generative Adversarial Networks (GANs) were introduced in 2014 by Ian J. Goodfellow
- GANs perform unsupervised learning tasks in machine learning.
- It consists of 2 models that automatically discover and learn the patterns in input data.

- A Generator in GANs is a neural network that creates fake data to be trained on the discriminator. It learns to generate plausible data.

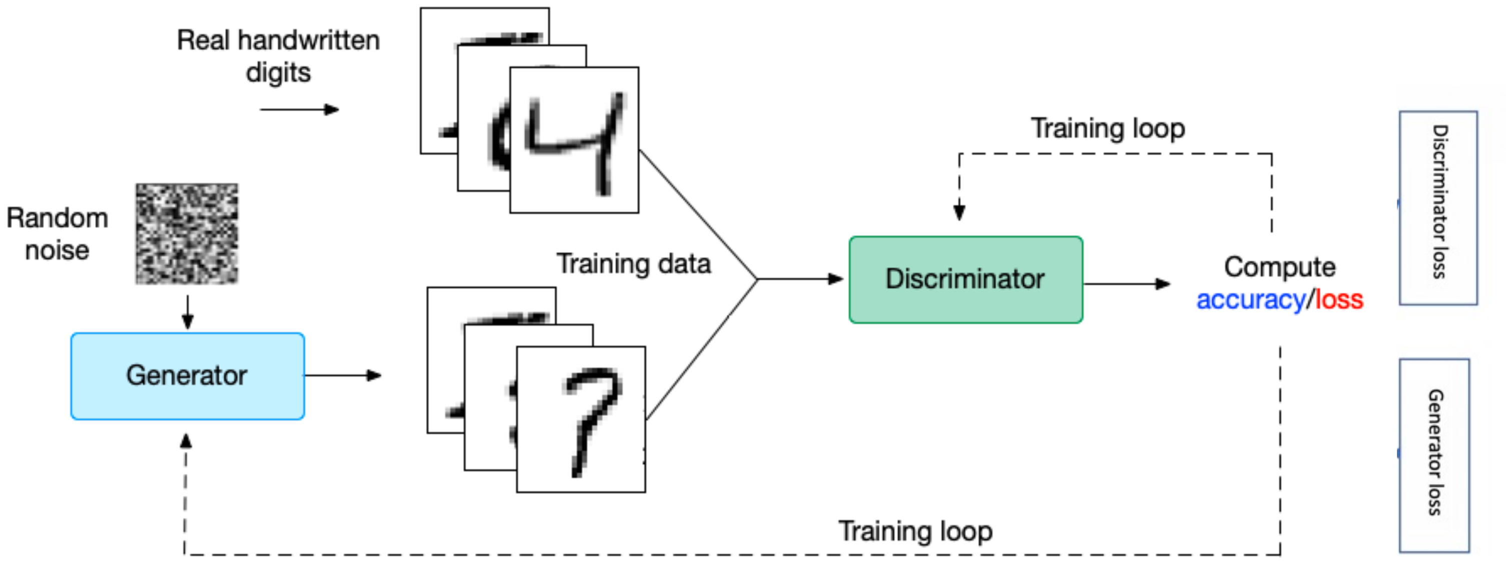




- The Discriminator is a neural network that identifies real data from the fake data created by the Generator. The discriminator's training data comes from different two sources

Steps for Training GAN

1. Define the problem
2. Choose the architecture of GAN
3. Train discriminator on real data
4. Generate fake inputs for the generator
5. Train discriminator on fake data
6. Train generator with the output of the discriminator

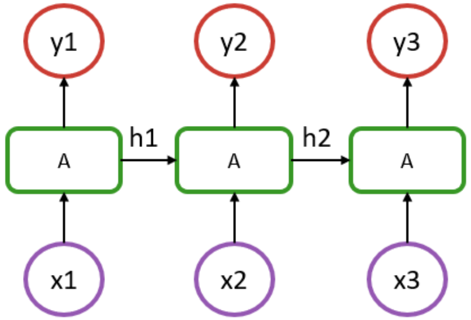


AncoraSIR.com

Exercise with Julia

- CNN: Handwritten digits classification

- RNN: AI Generates Shakespeare-like text



- Deep Convolutional GANs (DCGANs): Generate images from noise

AncoraSIR.com

Homework

- Start assembly your team’s portion of Reachy and identify parts that could be redesigned with generative design.
  - 装配手册（文字、截图、视频）
- On Tue Nov 15, each team will give a presentation of 6 mins + Q&A 4 mins
  1. 从个人的学习、生活经验、实习工作经验中，找到可能可以用智能机器人解决的痛点/需求。
  2. 探索一个或多个可能可以集成/改进到Reachy中的机器人/人工智能应用
     - 感知能力：视觉、触觉、语音？
     - 学习能力：强化学习、模仿学习？
     - 执行能力：移动底盘、双足、软体手？

AncoraSIR.com

SUSTech
Southern University of Science and Technology

Thank you~

Wan Fang
Southern University of Science and Technology