SCHOOL OF ENGINEERING
& APPLIED SCIENCE

Optimization and Troubleshooting in Machine Learning

Shuyue (Frank) Guan

The Medical Imaging & Image Analysis (MIA) Laboratory

April 2022

Choose the right model

Deep learning is NOT the panacea.

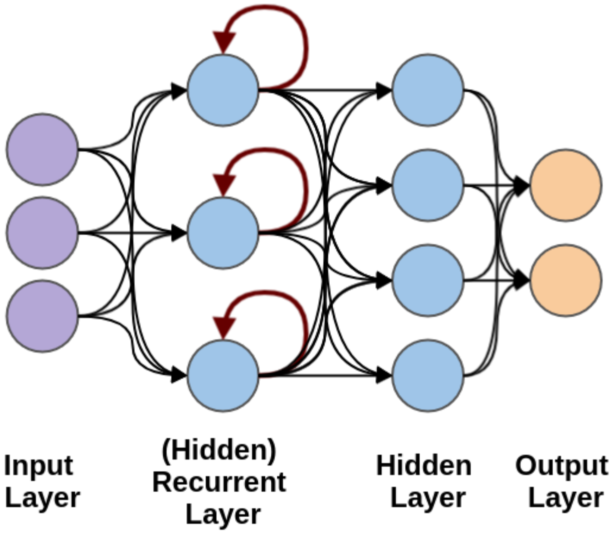
non-DL vs. DL models?

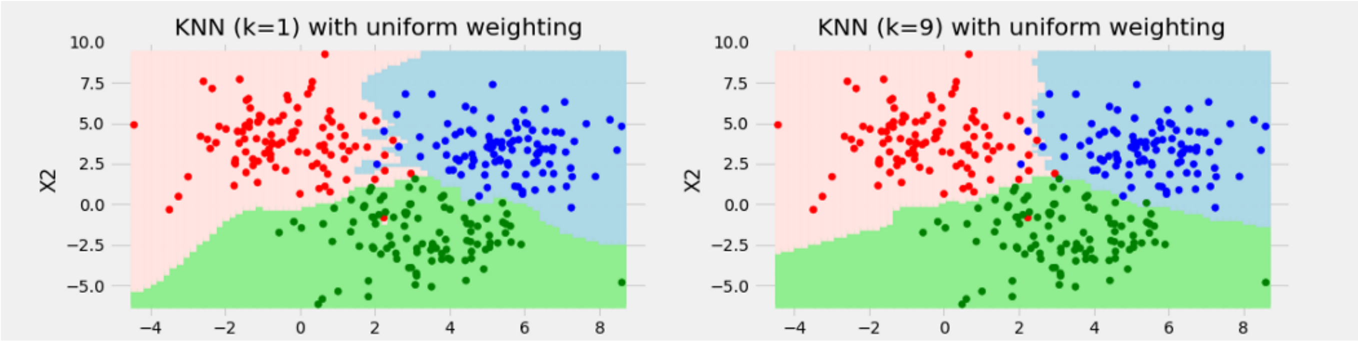
- Not many features -> non-DL
- Not enough data -> non-DL
- High separability by some features -> non-DL

Choose the right model (cont.)

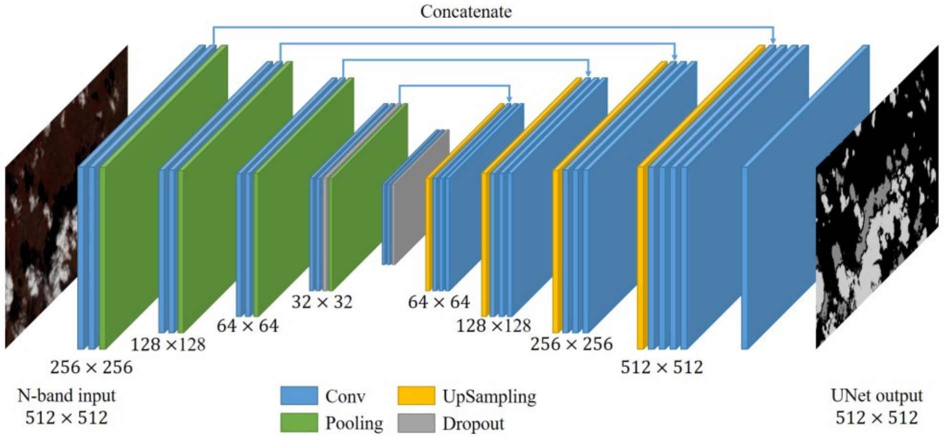
Problem type

- Regression problems
  - Logistic regression
  - DNN
  - Recurrent Neural Network (RNN)
- Classification problems
  - KNN
  - Random Forest
  - SVM
  - DNN/CNN
- Segmentation problems
  - Thresholding
  - Clustering
  - U-Net (DL)





John Clements, 2021



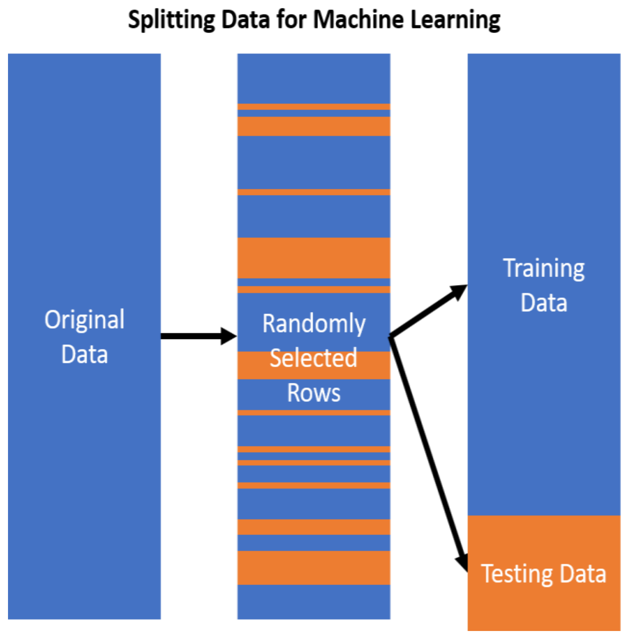
Libin Jiao et al., 2020

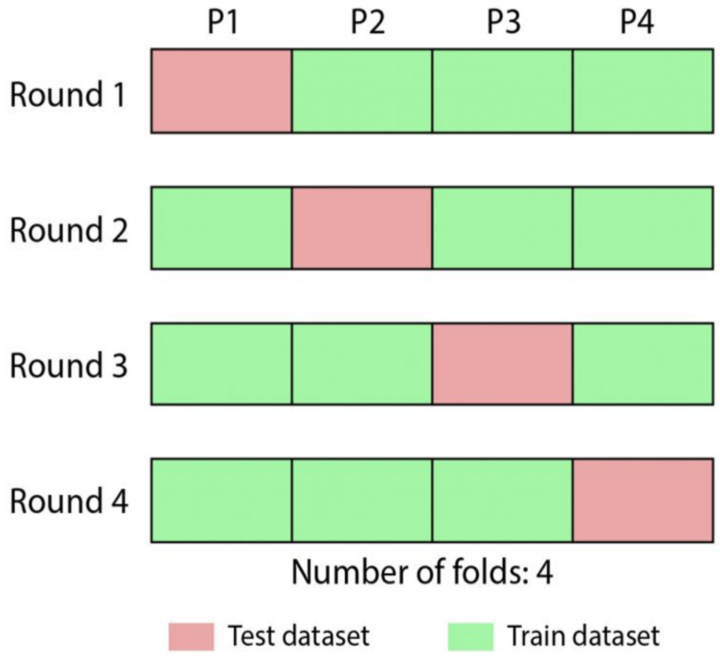
Model selection/adjust

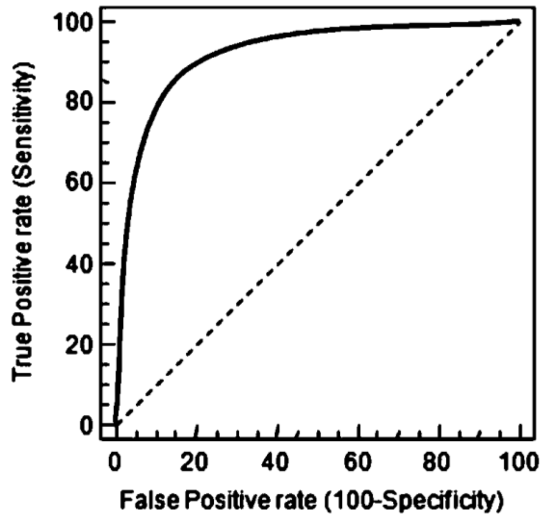
- Validation
  - Training – Test datasets
  - K-fold Cross Validation

- Choose metrics
  - **Accuracy**
  - Sensitivity(TPR), FPR,…

- Receiver Operating Characteristic (ROC)
  - **AUC**



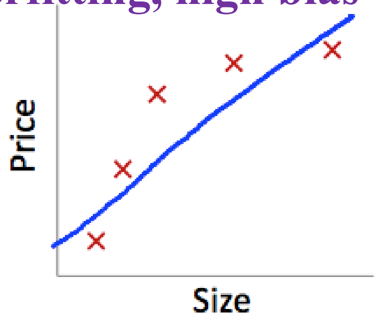




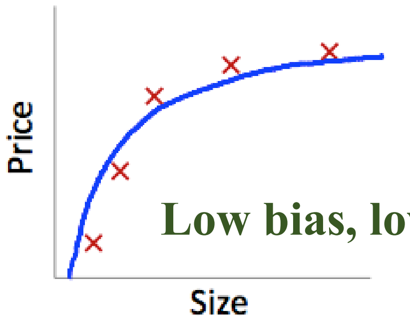
Model selection/adjust (cont.)

The bias-variance tradeoff

Underfitting, high bias

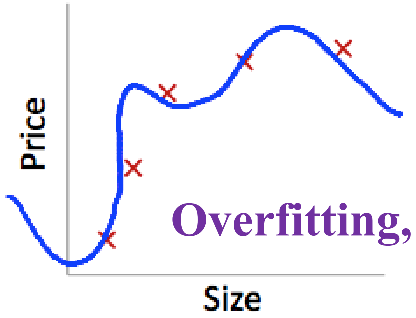


$\theta_0 + \theta_1 x$



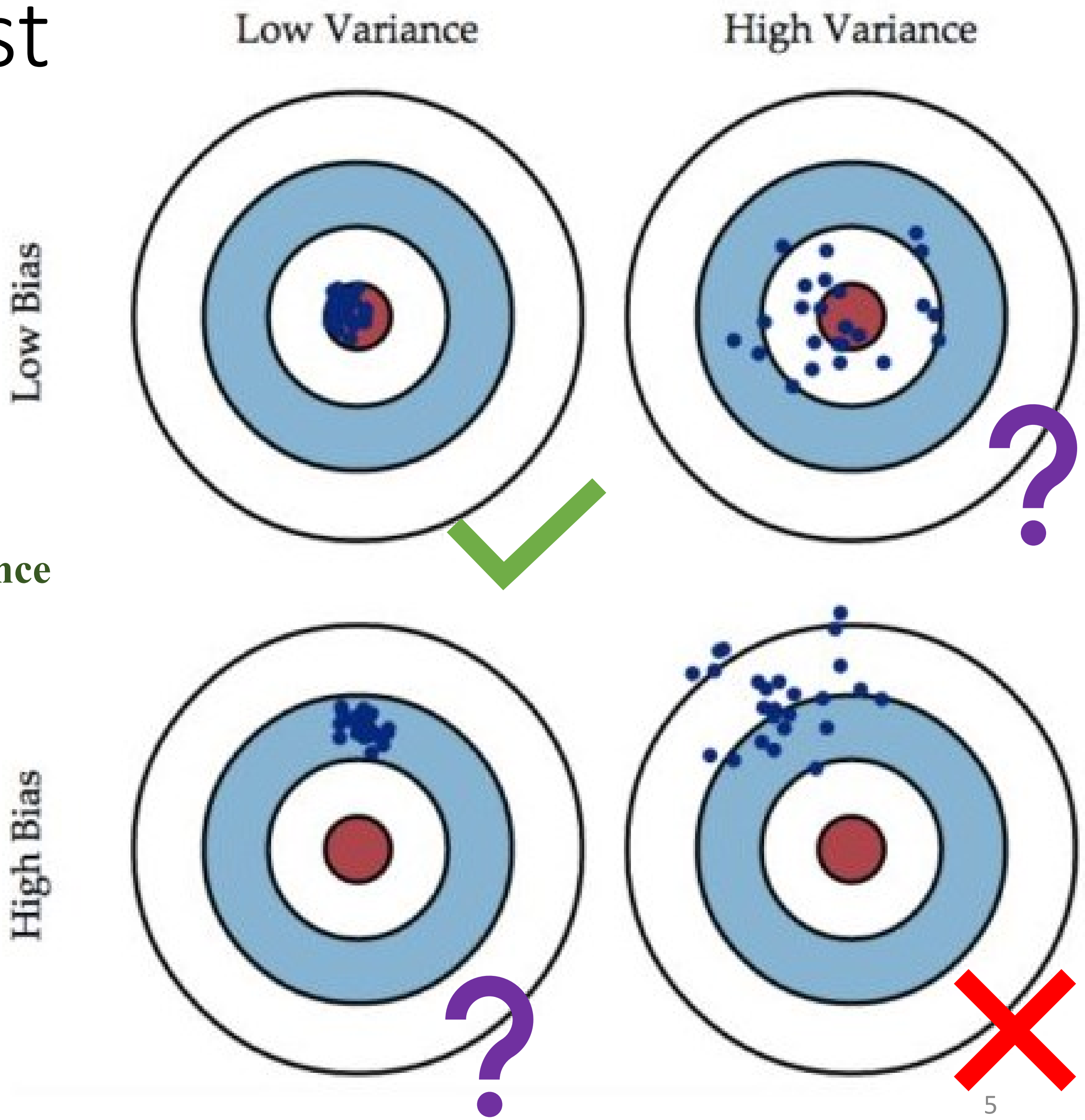
Low bias, low variance

$\theta_0 + \theta_1 x + \theta_2 x^2$



Overfitting, high variance

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



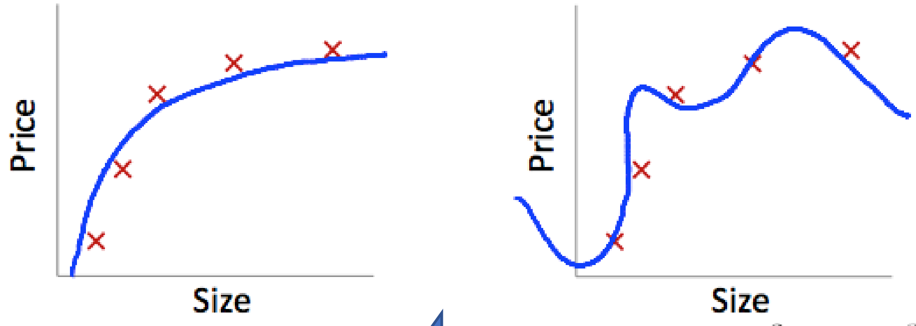
Methods of model adjust

- High Variance (overfitting)
  - **To increase training data**
  - Dimensionality reduction
  - To simplify the model

Tradeoff

- High Bias (underfitting)
  - To increase dimensionality
  - To use more complicated model

To simplify the model



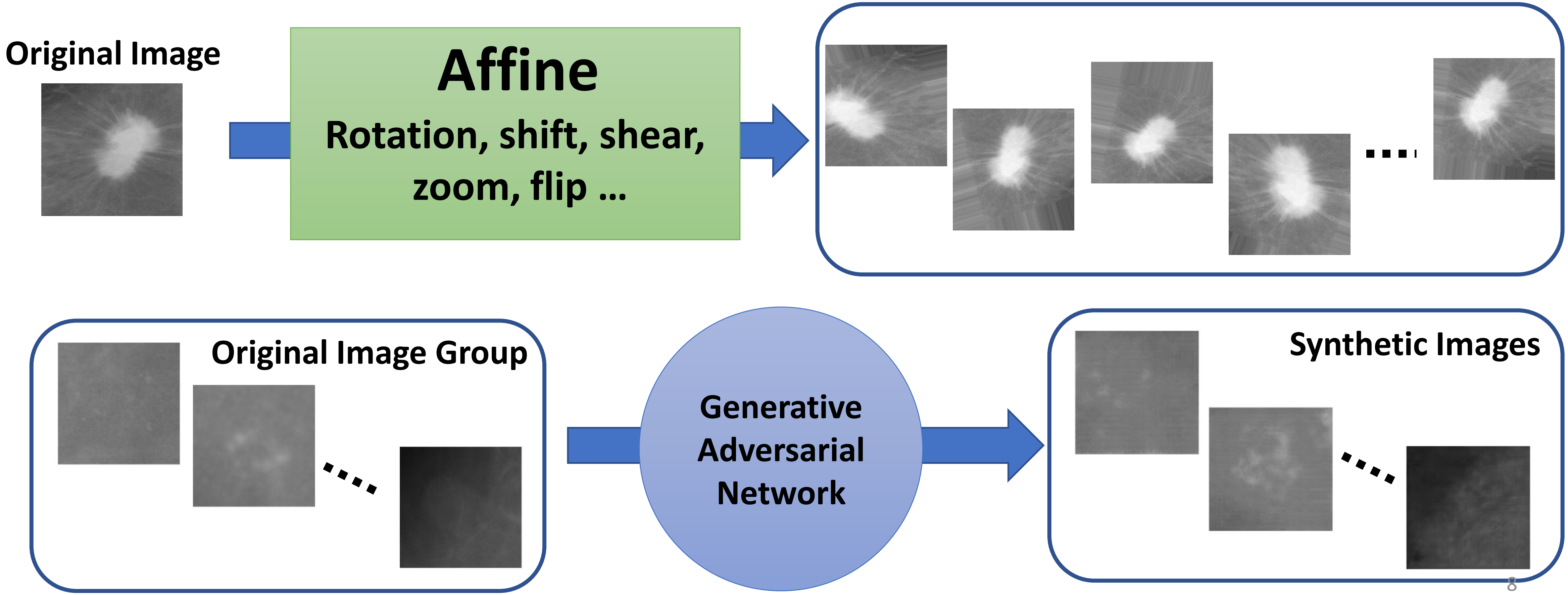
$\theta_0 + \theta_1 x + \theta_2 x^2$ ⇦ $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Solutions to improve performance

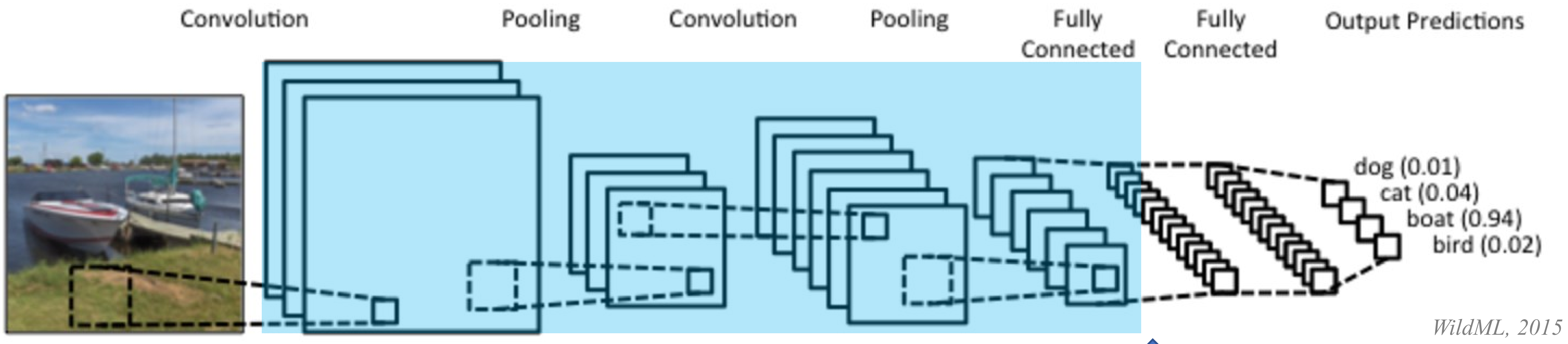
- Overfitting (commonly happens)
  - To increase training data
    - Collect more data
    - **Data augmentation**
    - **Transfer learning**
  - **Dimensionality reduction**
    - **Feature reduction (PCA)**
    - **Feature selection**
    - **Manifold learning**
  - **To simplify the model**
    - **Dropout**
    - **Regularization (L1, L2 - norms)**
- Underfitting
  - To increase dimensionality
    - Add features
    - Mapping functions
  - To use more complicated model
    - More training
    - Larger scale models (**boosting**)

Data augmentation

Create different but related images.



Transfer learning



Pre-trained model: weights frozen

Input new images

Feature extraction

Train new classifiers

Dimensionality reduction

- Solve overfitting

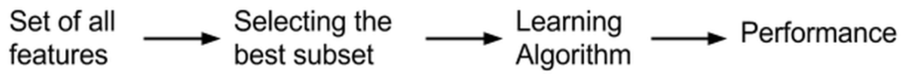
And,

- Reduce computing time and storage space
- Remove of linear correlations (multi-collinearity) to improve the performance of the machine learning algorithms
- Easier to visualize or explain the data (such as in 2D or 3D)

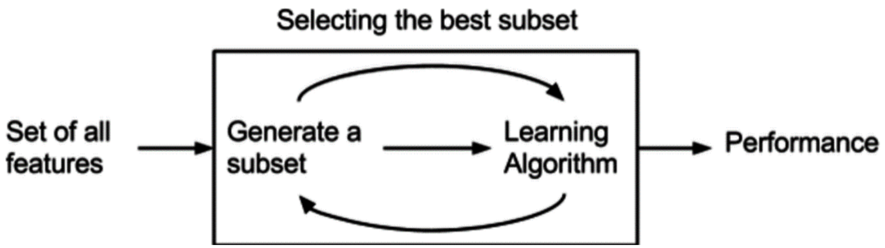
Dimensionality reduction (cont.)

DR

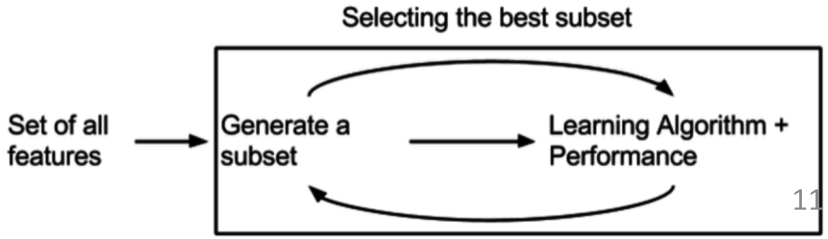
- Feature **Extraction**
  - Principal Components Analysis - PCA (Representation)
  - Linear Discriminant Analysis (Classification)
  - t-SNE / UMAP
  - Autoencoder
  - ……
- Feature **Selection**
  - Filter (Evaluation)



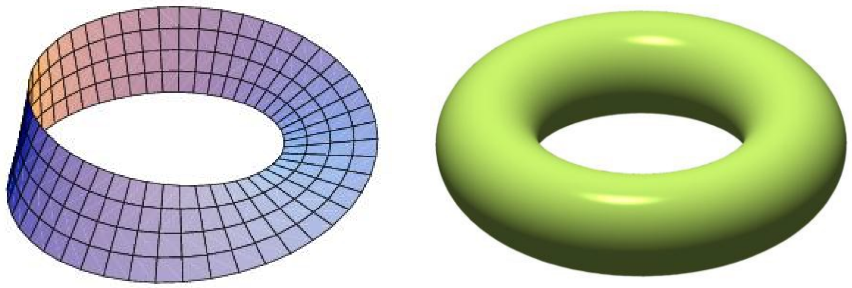
  - Wrapper (Optimization)



  - Embedded (*e.g.*, LASSO)



Manifold Learning



$M \subset R^N$

Dimensionality reduction:

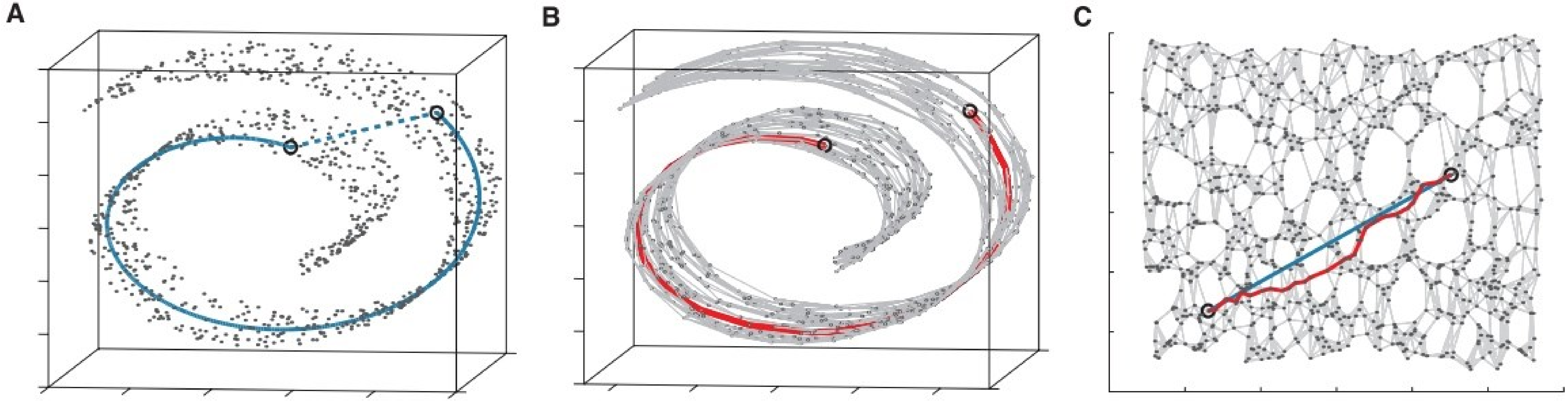
$M \to R^m, \qquad m \ll N$

Manifold

degree of freedom = 2

- Lower dimension shapes embedding
- Smaller degrees of freedom

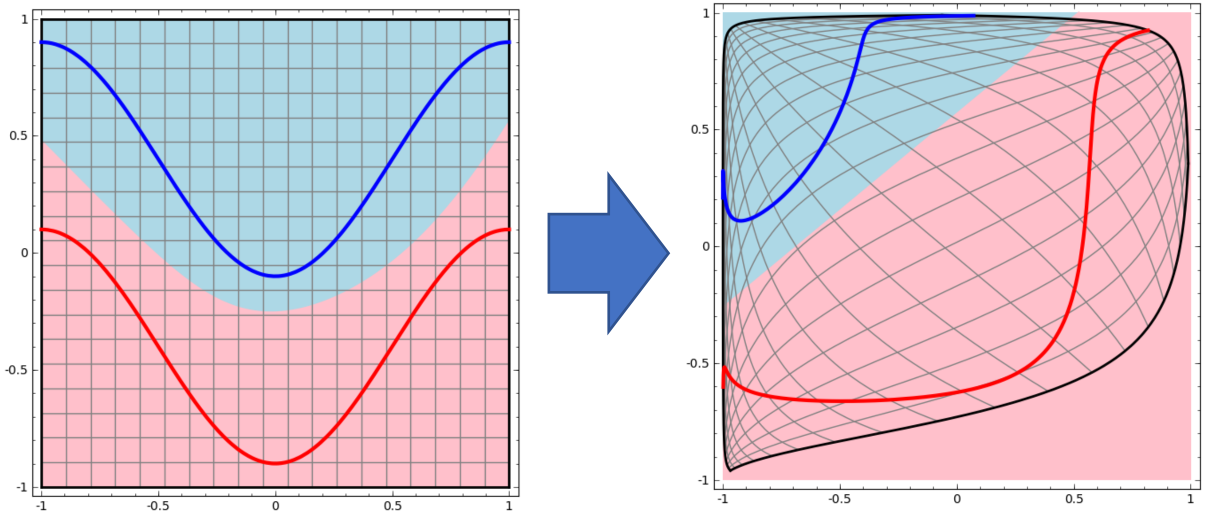
Euclidean distances → Geodesic distances

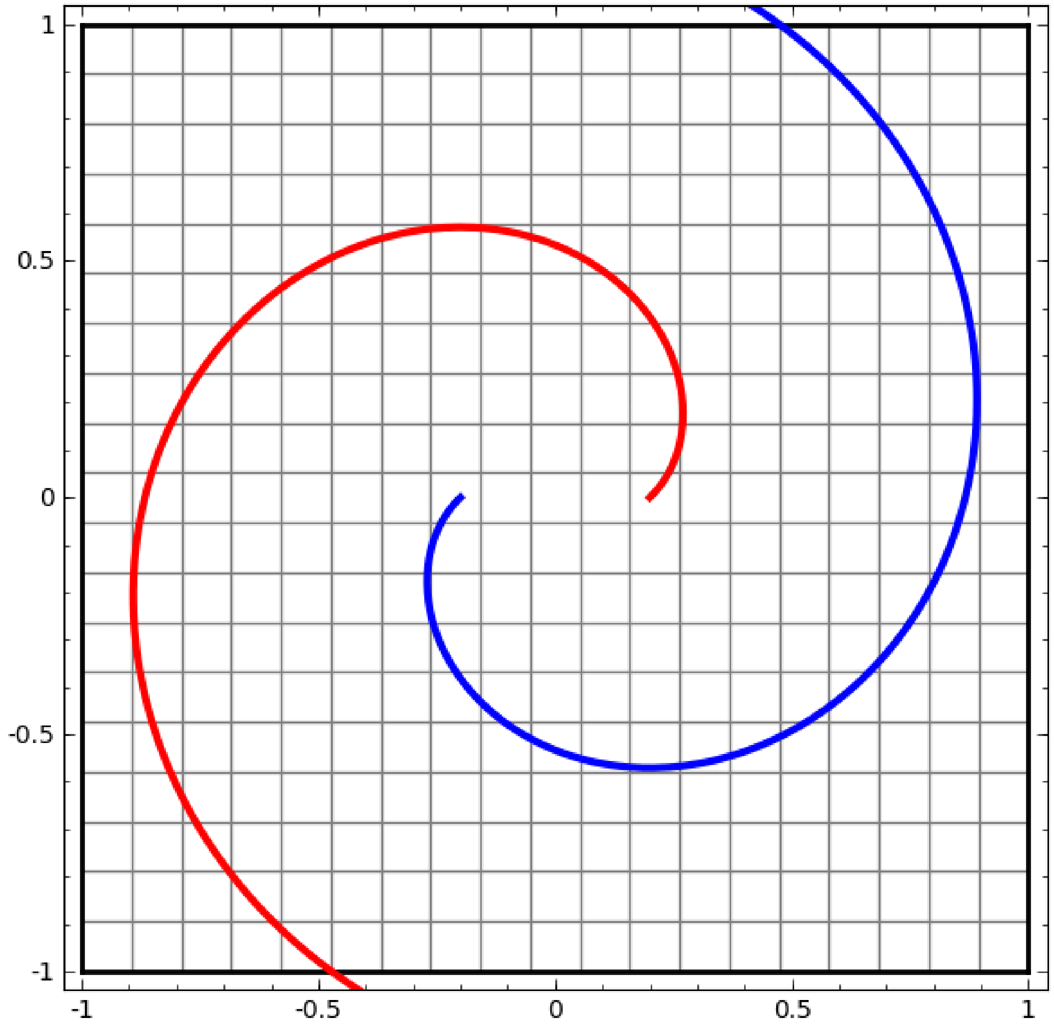


Classification for manifolds

colah's blog

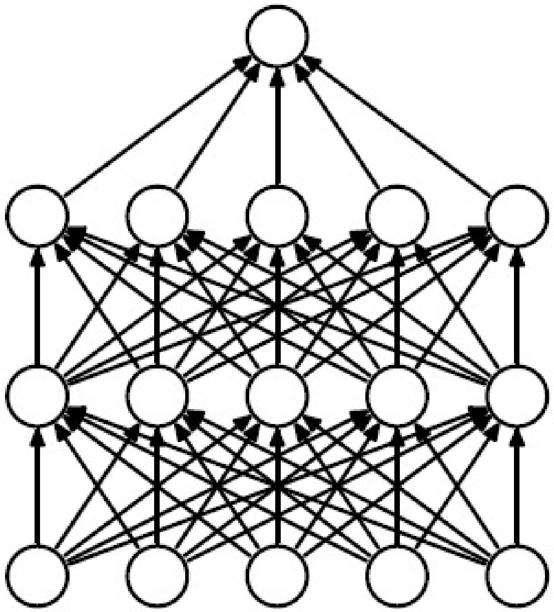
Linearly separable by homeomorphic transformation.



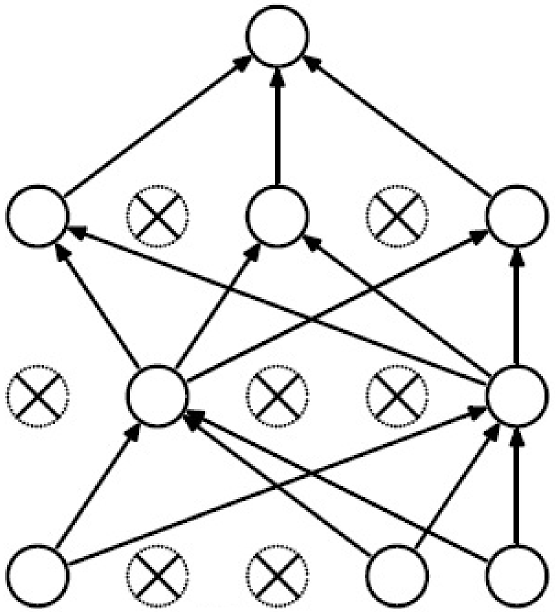


To simplify the model – dropout

For training neural networks



(a) Standard Neural Net



(b) After applying dropout.

dropout_layer = **tf.keras.layers.Dropout**(rate=**0.2**)

To simplify the model – regularization

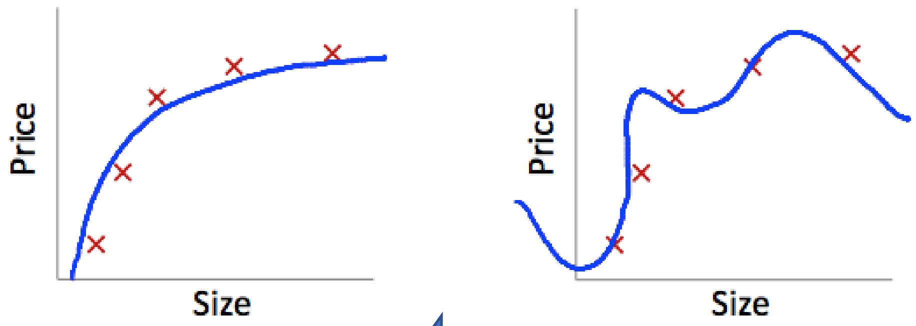
No regularization:

$$Loss_i = \left(y_i - f(x_i)\right)^2$$

With regularization:

$$Loss_i = \left(y_i - f(x_i)\right)^2 + \boldsymbol{\lambda r(\theta)}$$

To simplify the model



$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \Leftarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

L1-norm (LASSO):

$$r(\theta) = \sum_j |\theta_j|$$

L2-norm (Ridge):

$$r(\theta) = \sqrt{\sum_j \theta_j^2}$$

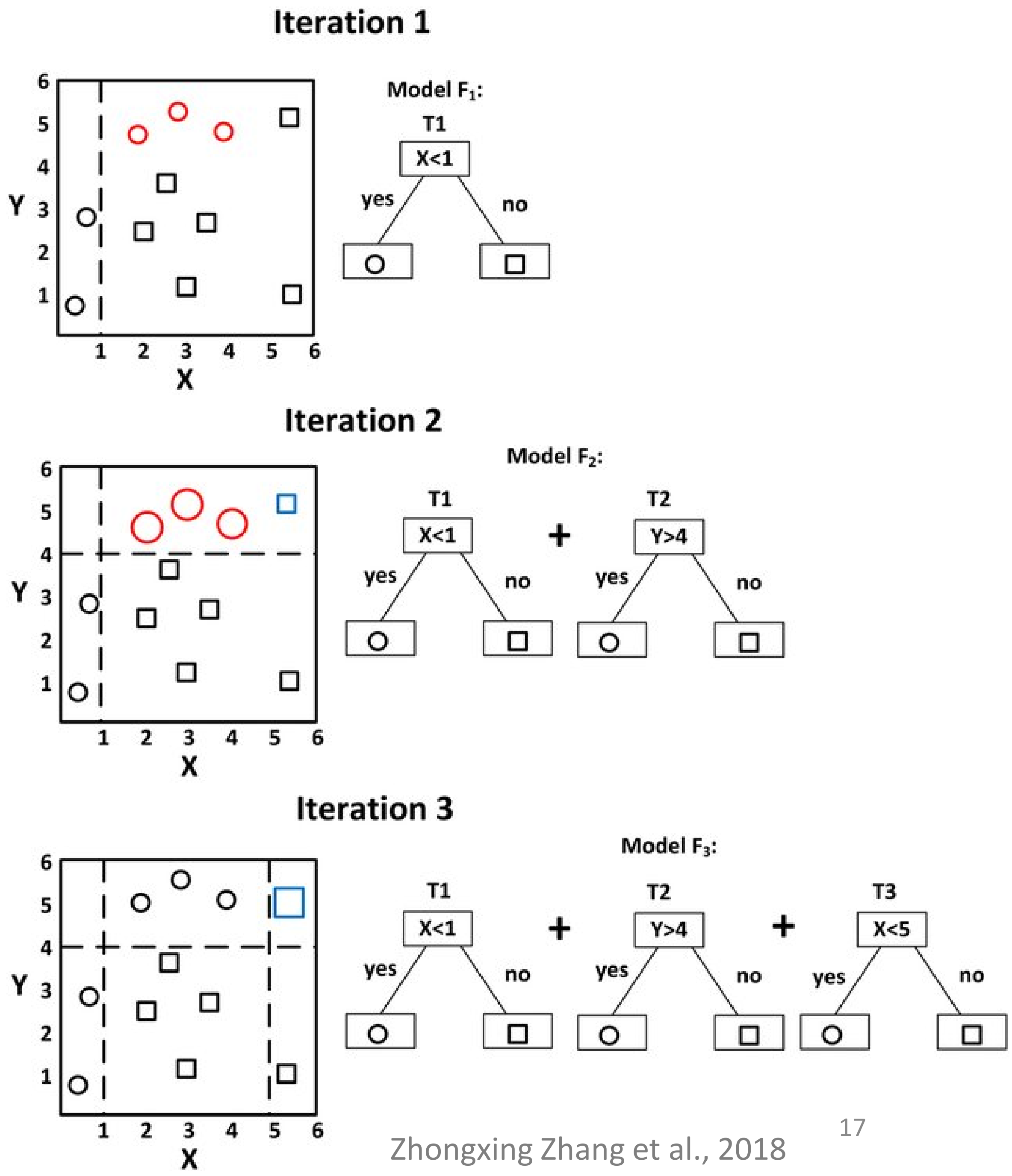
Regularization in deep learning

```
# Import L2-norm: weight decay
from keras.regularizers import l2
# Weight Regularization for Dense Layers
model.add(Dense(32, kernel_regularizer=l2(0.01),
bias_regularizer=l2(0.01)))
# Weight Regularization for Convolutional Layers
model.add(Conv2D(32, (3,3), kernel_regularizer=l2(0.01),
bias_regularizer=l2(0.01)))
```

To complicate the model – boosting

To put a set of weak (simple) models together to create a strong (complicated) model.

- Adaboost
- Gradient Boosted Decision Tree
- Gradient Boosted Regression Tree
- ……



Zhongxing Zhang et al., 2018

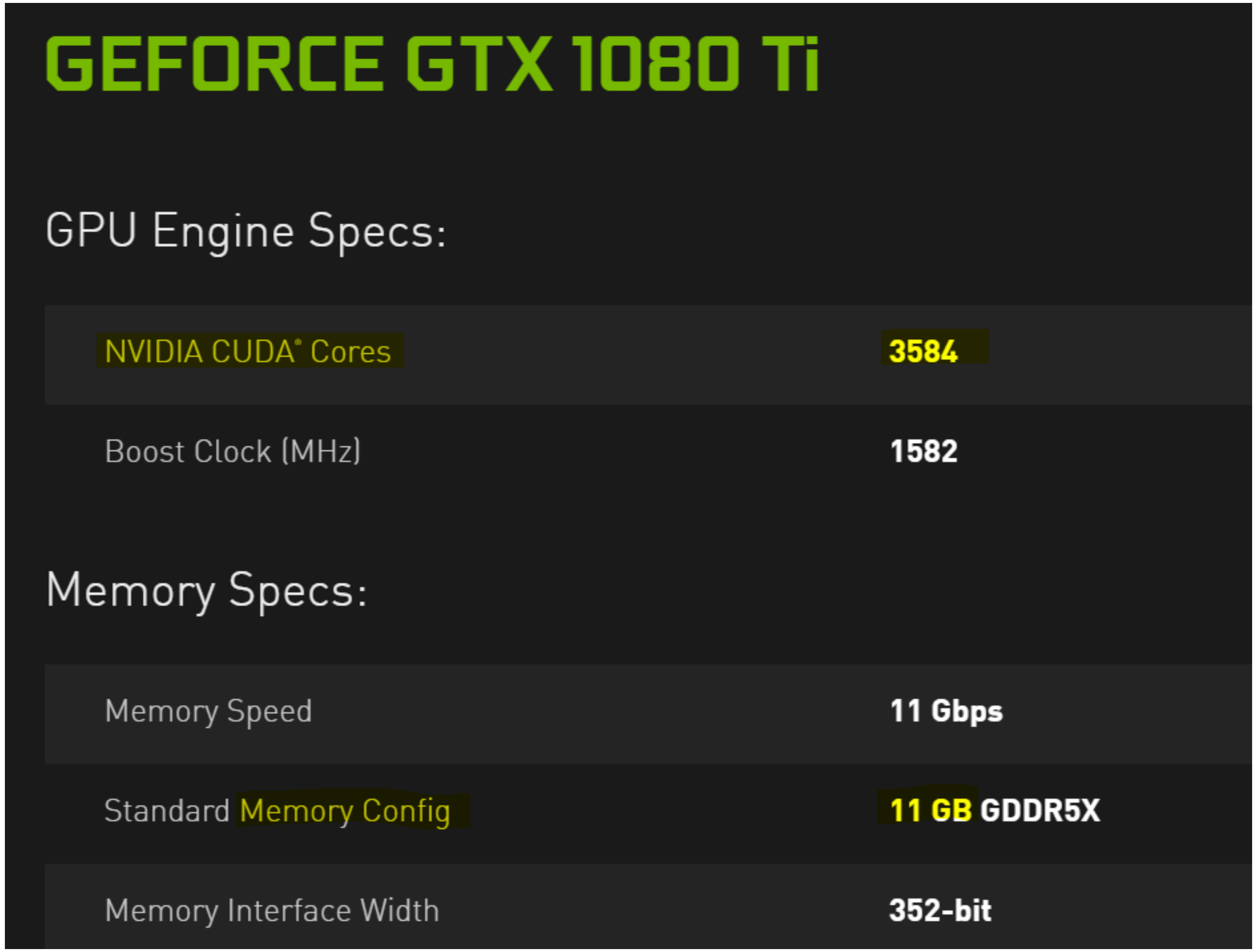
Troubleshooting and suggestions

Hardware resource

- GPU for DL
  - Buffer (Memory) – **<u>Crucial</u>**
  - CUDA Cores – Speed

  Solutions to out of memory (OOM) issues:
  - Model scale: # of layers, # of neurons, layer types.
  - Input size
  - Batch size

- Disk: store data in HDD, load/run data in **solid-state disk (SSD)**.



GEFORCE GTX 1080 Ti

GPU Engine Specs:

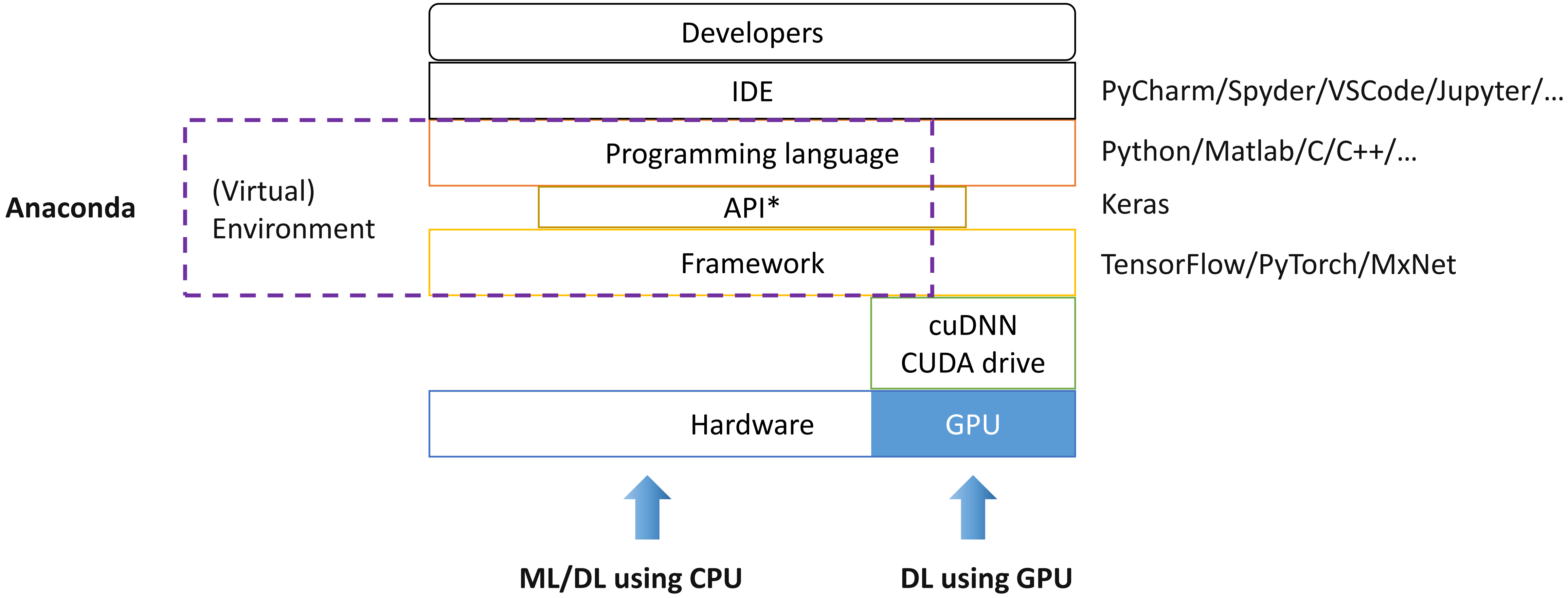
NVIDIA CUDA® Cores	3584
Boost Clock (MHz)	1582

Memory Specs:

Memory Speed	11 Gbps
Standard Memory Config	11 GB GDDR5X
Memory Interface Width	352-bit

Tips: SSD (512GB) + HDD (2TB)

Software overview



The version problem!!

Configuration for DL

DL using CPU

1. GPU without CUDA cores
2. Install Anaconda
3. (Conda) Install Python
4. (Conda) Install TensorFlow
5. (Conda) Install Keras
6. Install PyCharm
7. (PyCharm) Import Conda environment
8. Ready to use!

DL using GPU

1. Nvidia GPU with CUDA cores
2. Install CUDA drive
3. Set cuDNN
4. Install Anaconda
5. (Conda) Install Python
6. (Conda) Install TensorFlow-GPU
7. (Conda) Install Keras
8. Install PyCharm
9. (PyCharm) Import Conda environment
10. Ready to use!

Version problem for DL

According to:

- OS (Win, Linux, macOS)
- CPU-based/GPU-based
- **CUDA**
- **cuDNN**
- **Framework**
- **Programming language**
- ...

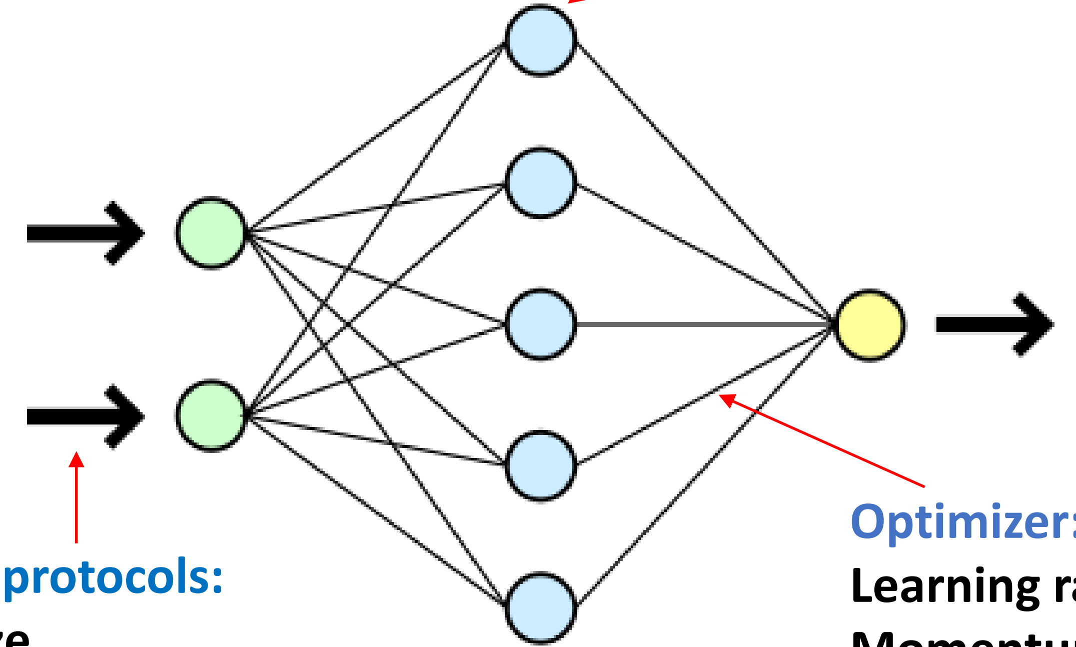
Version	Python version	cuDNN	CUDA
tensorflow_gpu-2.3.0	3.5-3.8	7.6	10.1
tensorflow_gpu-2.2.0	3.5-3.8	7.6	10.1
tensorflow_gpu-2.1.0	3.5-3.7	7.6	10.1
tensorflow_gpu-2.0.0	3.5-3.7	7.4	10
tensorflow_gpu-1.15.0	3.5-3.7	7.4	10
tensorflow_gpu-1.14.0	3.5-3.7	7.4	10
tensorflow_gpu-1.13.0	3.5-3.7	7.4	10
tensorflow_gpu-1.12.0	3.5-3.6	7	9
tensorflow_gpu-1.11.0	3.5-3.6	7	9
tensorflow_gpu-1.10.0	3.5-3.6	7	9
tensorflow_gpu-1.9.0	3.5-3.6	7	9
tensorflow_gpu-1.8.0	3.5-3.6	7	9
tensorflow_gpu-1.7.0	3.5-3.6	7	9
tensorflow_gpu-1.6.0	3.5-3.6	7	9
tensorflow_gpu-1.5.0	3.5-3.6	7	9
tensorflow_gpu-1.4.0	3.5-3.6	6	8
tensorflow_gpu-1.3.0	3.5-3.6	6	8

Windows GPU

Troubleshoot neural networks

Activation function

Input size
Noise
Data augmentation



Target value range
Loss function
Stop training
…

Training protocols:
Batch size
Epochs
…

Optimizer:
Learning rates
Momentum
Weight decay
…

# of neurons
# of hidden layers
Special layers: <u>dropout</u>, <u>regularization</u>, <u>normalization</u>, …

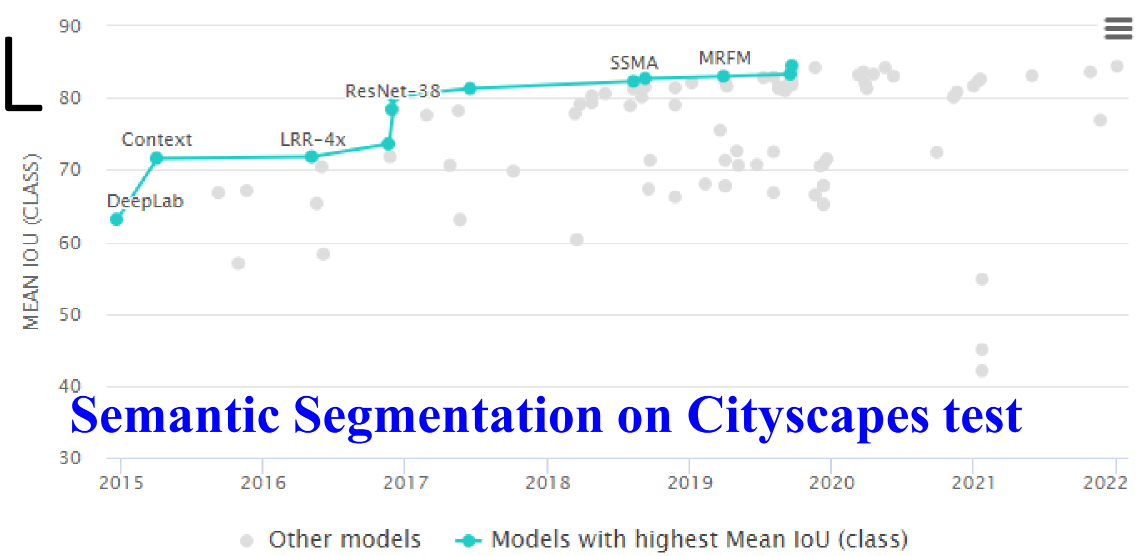
Data issues for machine learning

- **Enough data for the model?**
- Class imbalanced?
- **Need pre-processing?** – normalization, image enhancement, …
- Have some outliers?
- Cleaned/clear labels?
- Can training data well represent overall data distribution? (good sampling)
- …

Useful websites for ML/DL

- Guidebooks of each tools/software online – look-up books
- Github – without reinventing the wheel
- **Stack Overflow** – solve problems
- Kaggle – datasets
- **paperswithcode.com** – SOTA performance ranks with papers & codes
- Google – anything!
- …

View Mean IoU (class) by Date for All models



Semantic Segmentation on Cityscapes test

Rank	Model	Mean IoU (class)	Category mIoU	Time (ms)	Extra Training Data	Paper	Code	Result	Year
1	HRNetV2 + OCR +	84.5%			✓	Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation			2019
2	Lawin+	84.4%			×	Lawin Transformer: Improving Semantic Segmentation Transformer with Multi-Scale Representations via Large Window Attention			2022
3	EfficientPS	84.21%			✓	EfficientPS: Efficient Panoptic Segmentation			2020
4	Panoptic-DeepLab	84.2%			✓	Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation			2019
5	HRNetV2 + OCR (w/ ASP)	83.7%			✓	Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation			2019

Epilogue

- The Medical Imaging & Image Analysis Laboratory

  SEH 5290 W: https://loewlab.seas.gwu.edu/

- Speaker: **Shuyue Guan**

  https://orcid.org/0000-0002-3779-9368