Advanced Research Talk:
Learning-based Visual Content Analysis and Processing

Chaoyi Zhang

*School of Computer Science,
University of Sydney, Australia*

THE UNIVERSITY OF SYDNEY

Outline

- Classification & Segmentation
- Learning based Framework
- Learning based Image Content Classification
- Deep Learning based Image Segmentation
- Introduction of Deep Learning for Visual Content Analysis and Processing

Classification & Segmentation

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Cat vs Dog

Inputs — **Computerized Algorithm** → Output

$$Y = F(X)$$

Cat vs Dog

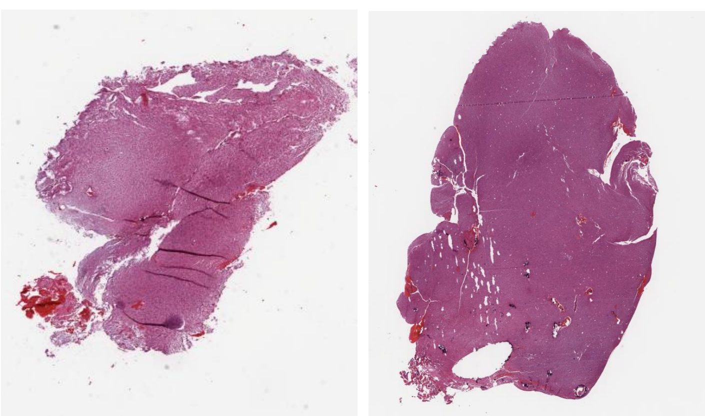
Classification

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Cat vs Dog

Classification



Astrocytoma vs Oligodendroglioma

Inputs — **Computerized Algorithm** → Output

$$Y = F(X)$$

Q: Whether the centroid pixel is a part of building or not?

Classification

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017, July). Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS) (pp. 3226-3229). IEEE.

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Q: Whether the centroid pixel is a part of building or not?

Yes vs No

Classification

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Q: Whether the centroid pixel is a part of building or not?

Yes vs No

Classification

In this case => Yes

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Q: Whether the centroid pixel is a part of building or not?

Yes vs No

Classification

In this case => Yes

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Q: Whether the centroid pixel is a part of building or not?

Yes vs No

Classification

In this case => Yes

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

Q: Whether the centroid pixel is a part of building or not?

Yes vs No

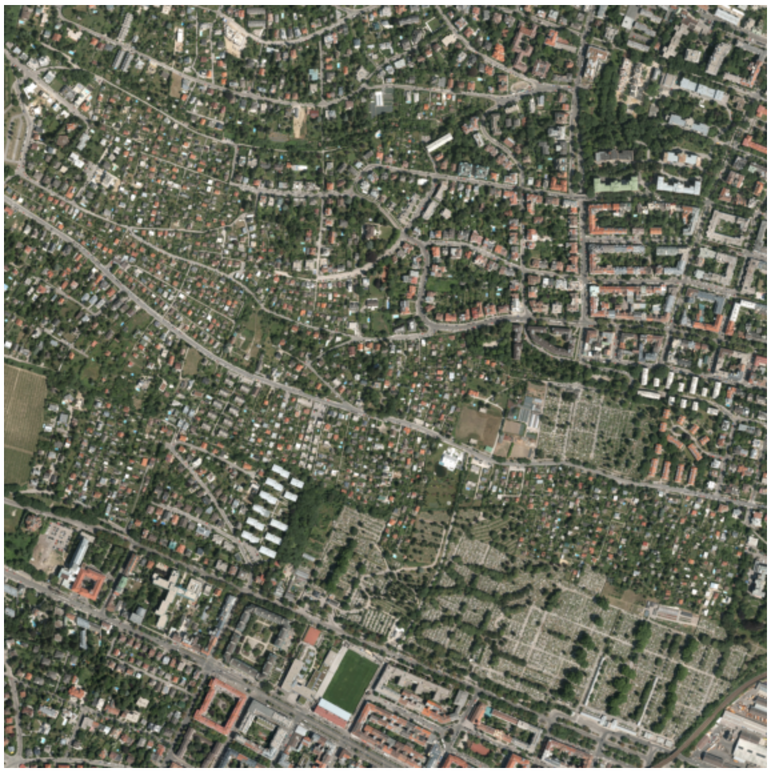
Classification

In this case => Yes

Computerized Algorithm

Inputs → Computerized Algorithm → Output

$$Y = F(X)$$



Q: Whether the centroid pixel is a part of building or not?

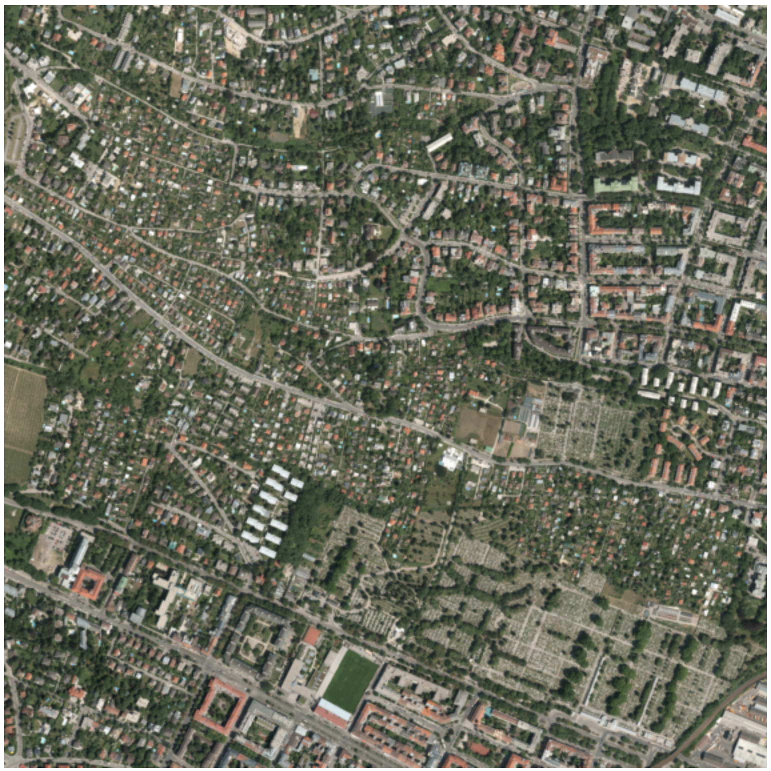
Yes vs No

Classification

In this case => Still, yes

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

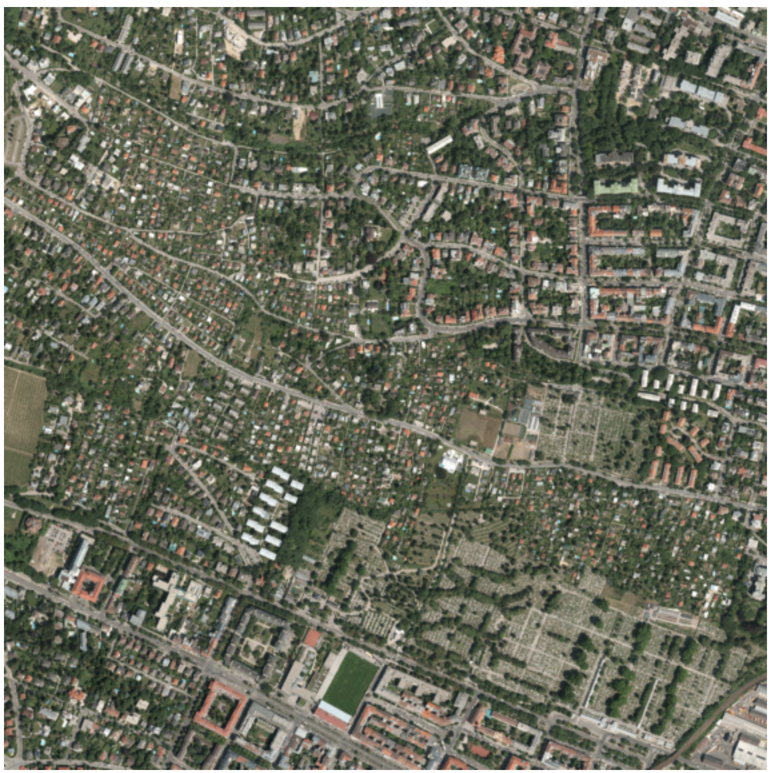


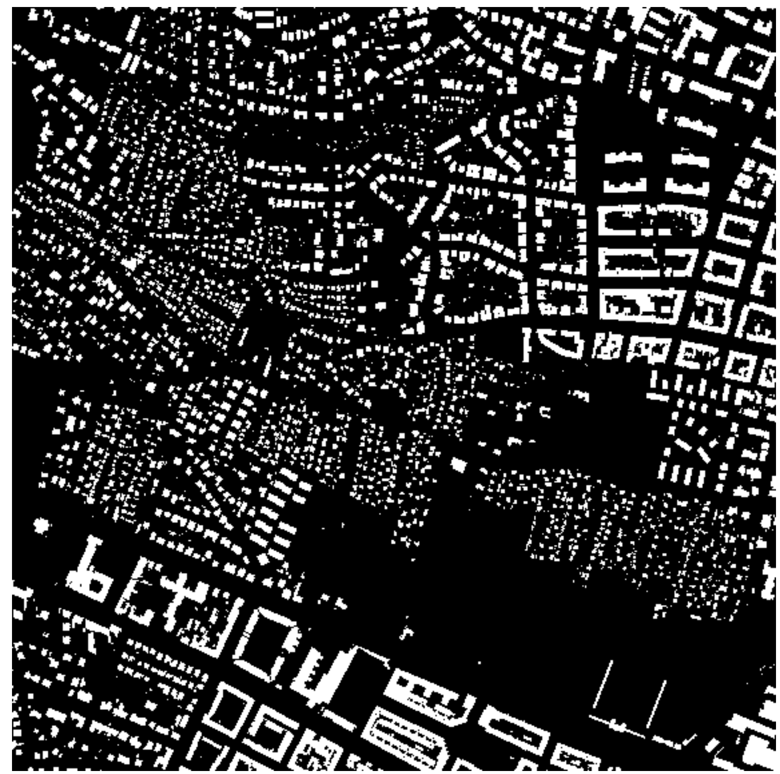
Q: Whether the centroid pixel is a part of building or not?

Q: How about the other pixels?

$$Y = F(X)$$

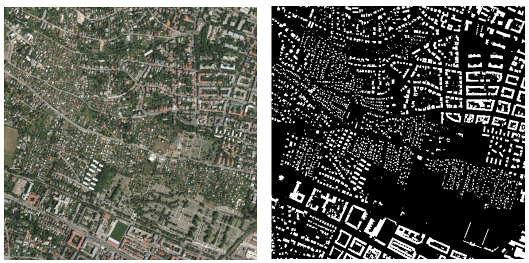
Q: How about the other pixels?



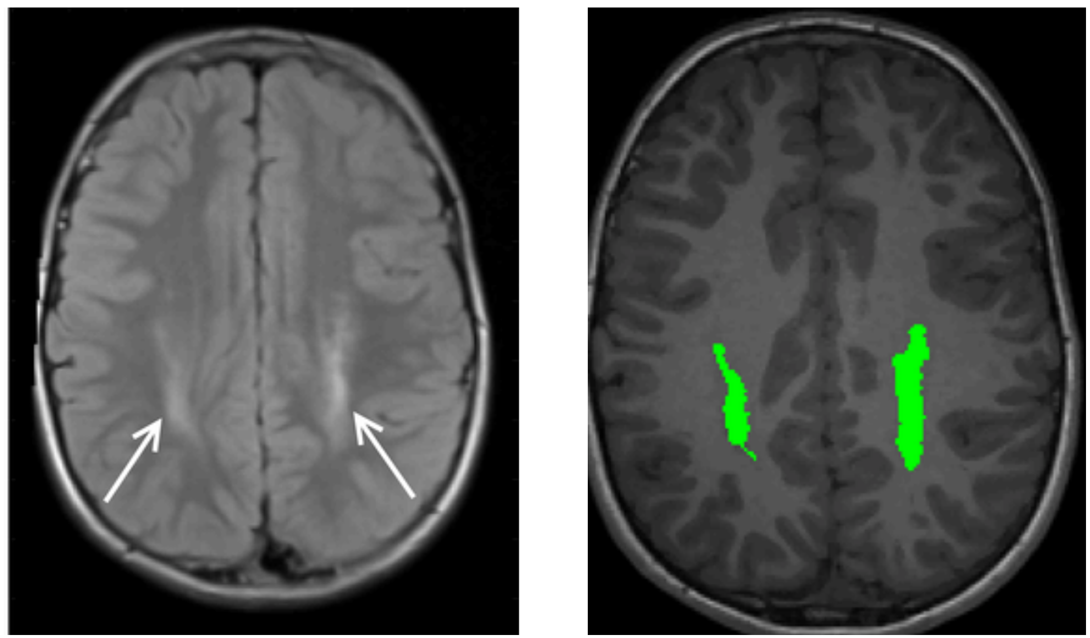


Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

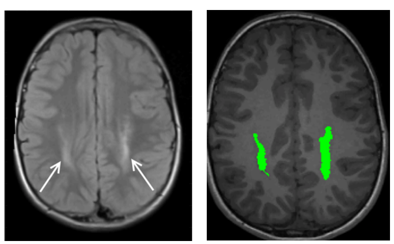


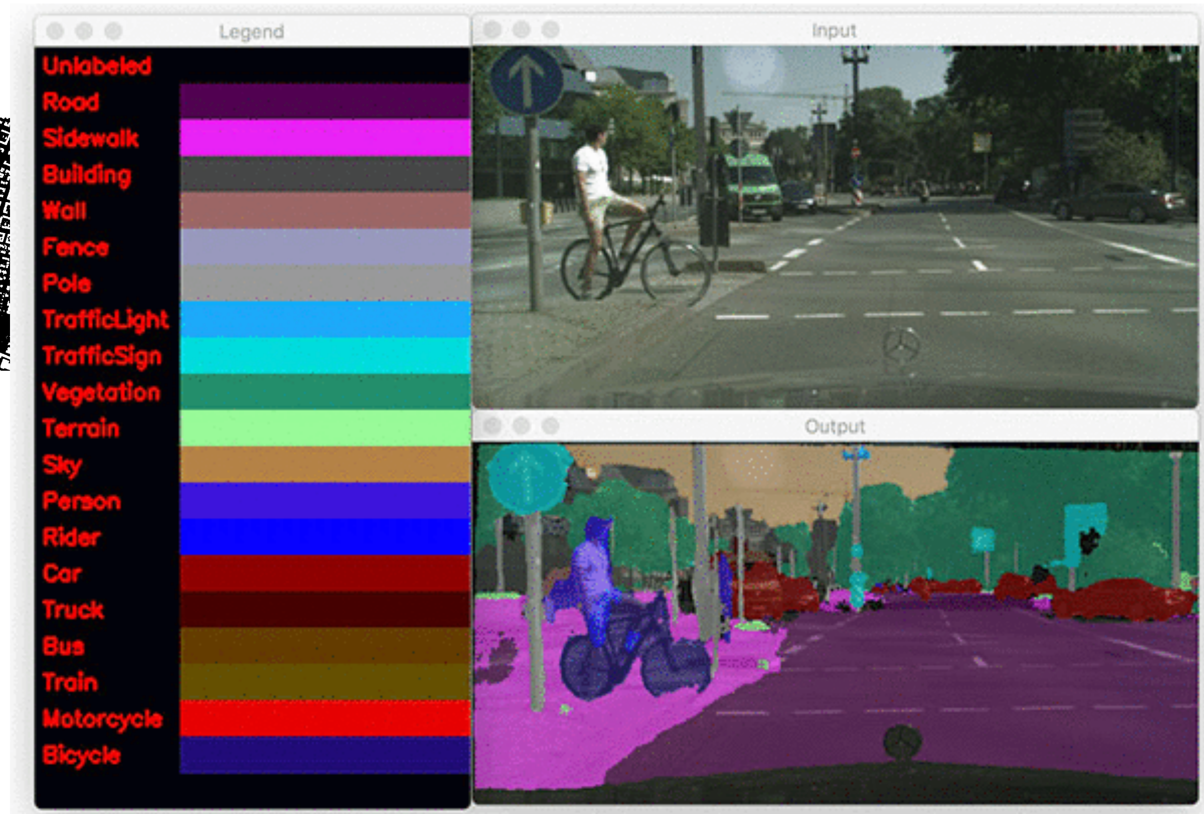
lesion



Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$



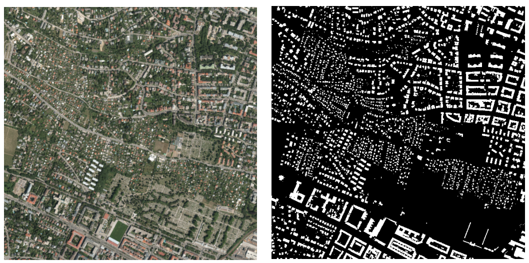


Multiple classes for urban scene understanding

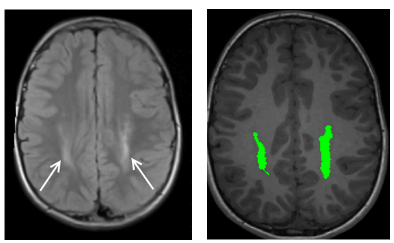
https://www.pyimagesearch.com/2018/09/03/semantic-segmentation-with-opencv-and-deep-learning/

Inputs — **Computerized Algorithm** → Output

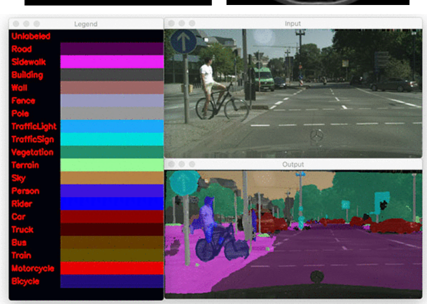
$$Y = F(X)$$



building



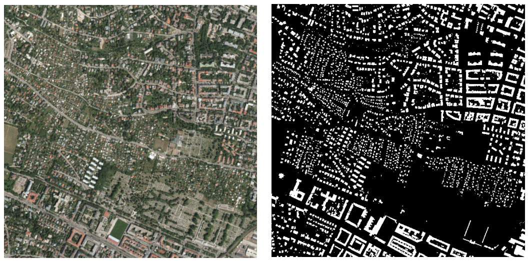
lesion



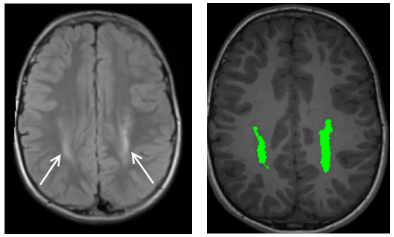
multiple classes
for urban scene understanding

Inputs → **Computerized Algorithm** → Output

$$Y = F(X)$$

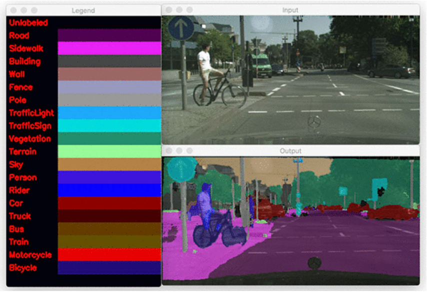


building



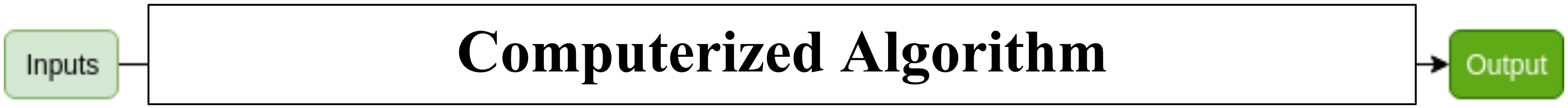
lesion

Segmentation

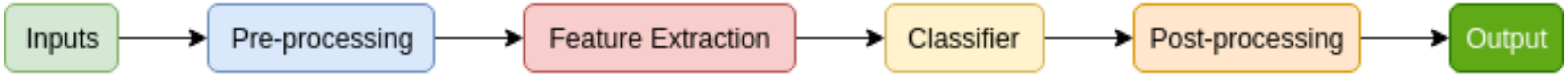


multiple classes
for urban scene understanding

Learning based Framework

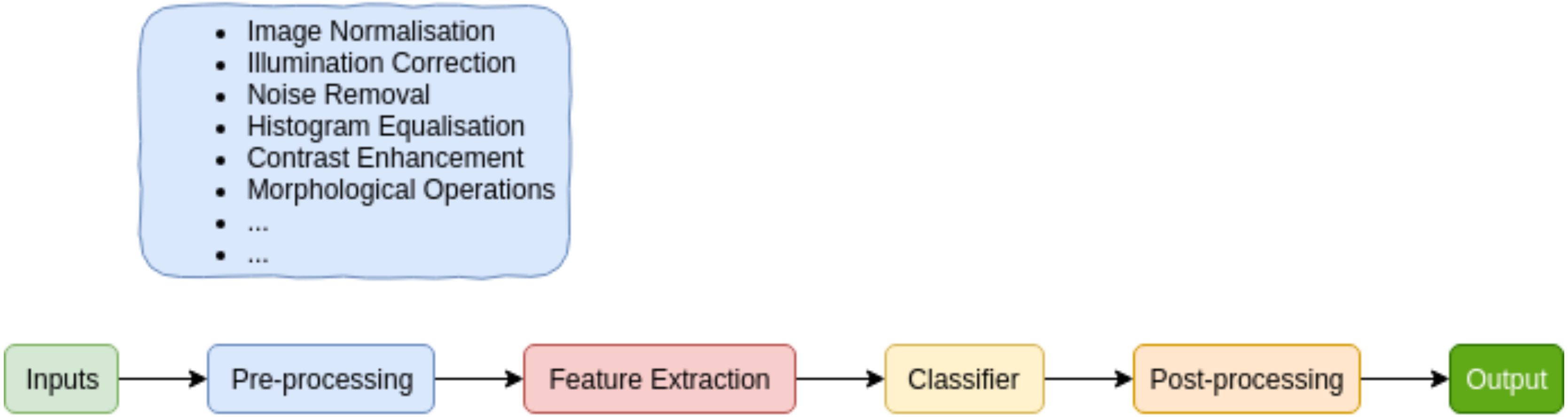


Learning based Framework



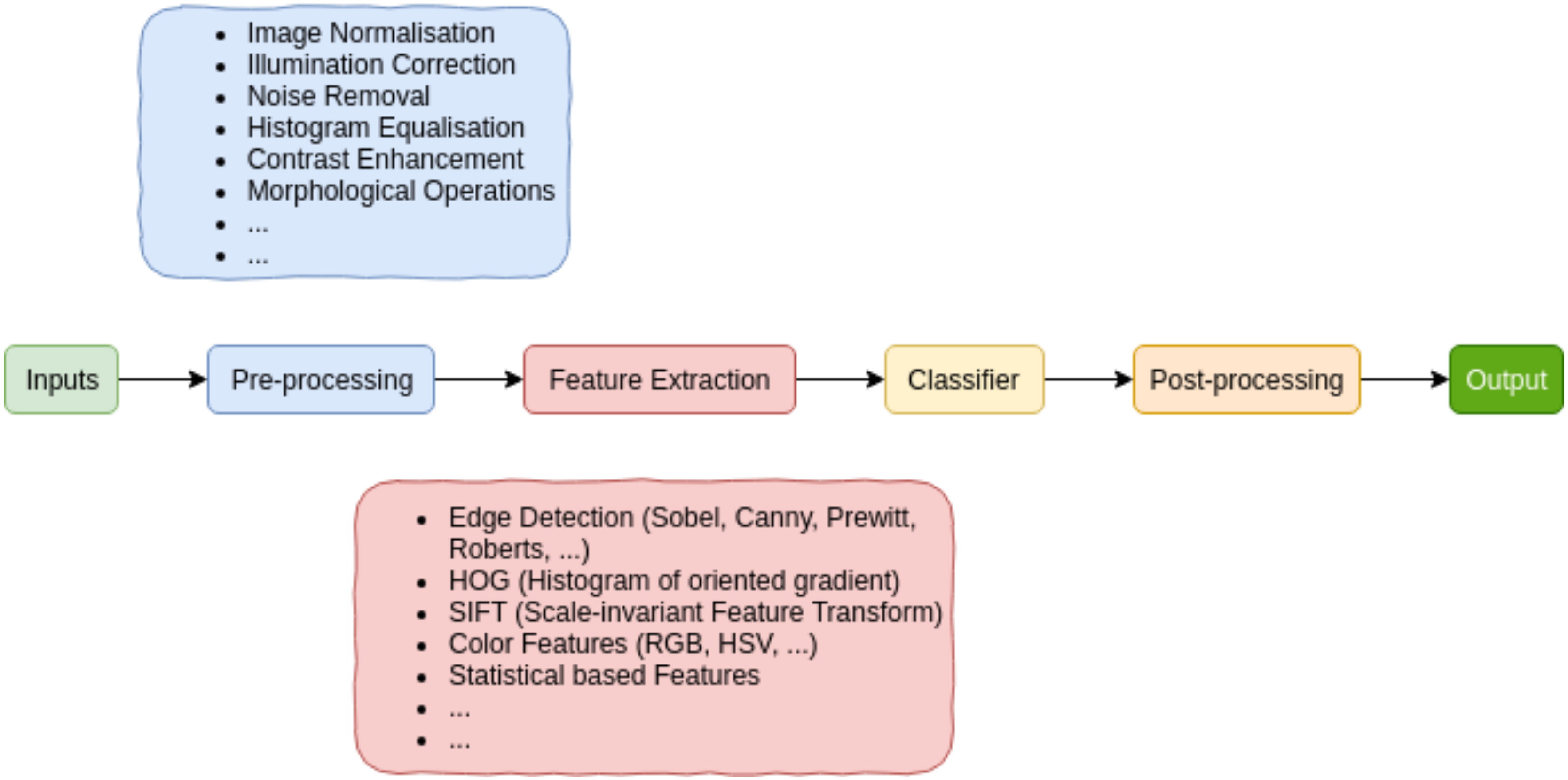
Learning based Framework

** Recall W01/02/03-Lecture*



Learning based Framework

** Recall W01/02/03-Lecture*

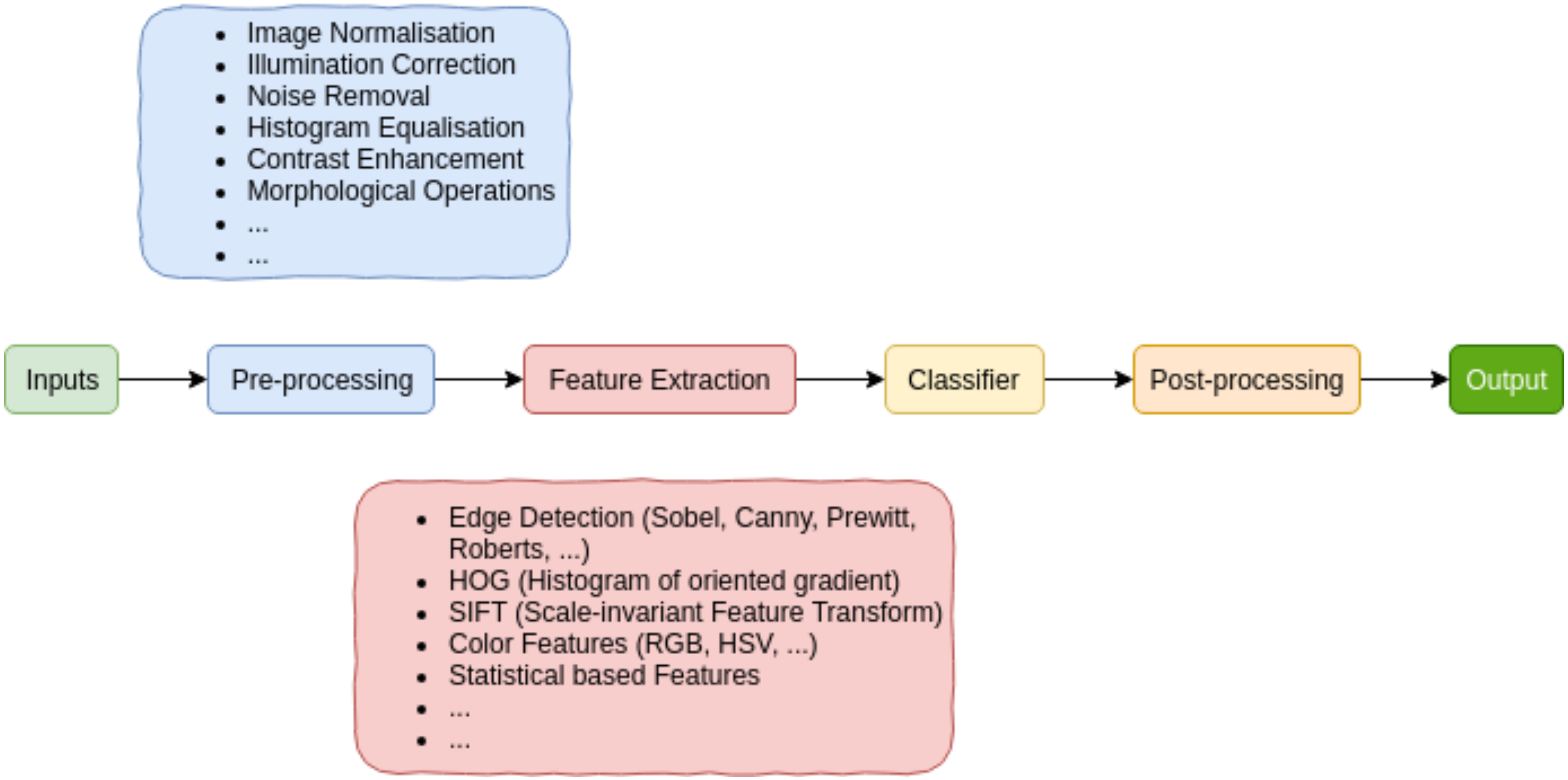


** Recall W08-Lecture*

Learning based Framework

$$Y = F(X)$$

** Recall W01/02/03-Lecture*



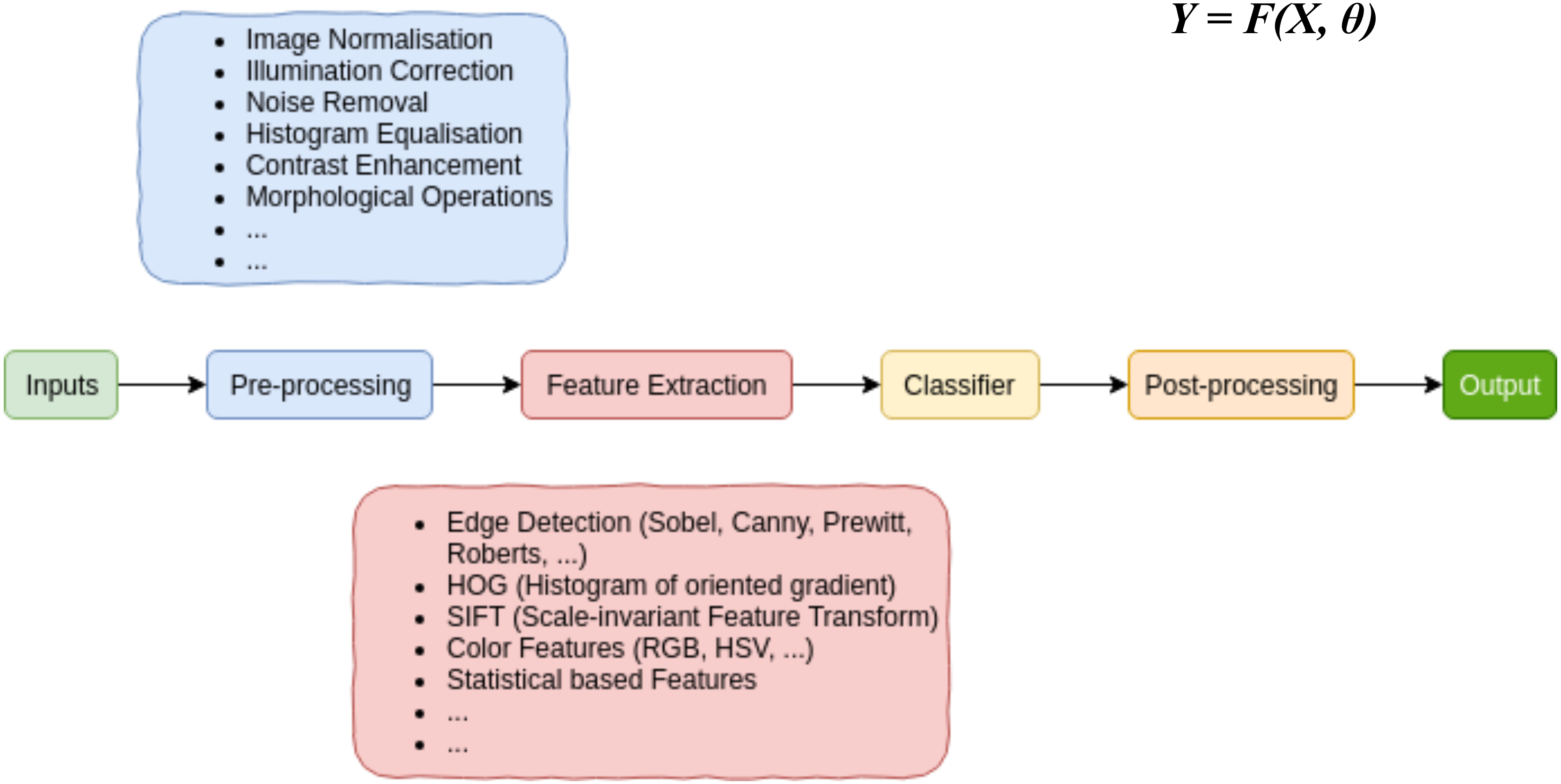
** Recall W08-Lecture*

Learning based Framework

$\cancel{Y = F(X)}$

$Y = F(X, \theta)$

** Recall W01/02/03-Lecture*



** Recall W08-Lecture*

Learning based Framework

$\cancel{Y = F(X)}$

$$Y = F(X, \theta)$$

- *Y – Output/Label*
- *X – Feature Vectors*
- *F – Classifier*
- *Θ – Classifier's Learnable Parameters*

** Recall W01/02/03-Lecture*

- Image Normalisation
- Illumination Correction
- Noise Removal
- Histogram Equalisation
- Contrast Enhancement
- Morphological Operations
- ...
- ...

- Edge Detection (Sobel, Canny, Prewitt, Roberts, ...)
- HOG (Histogram of oriented gradient)
- SIFT (Scale-invariant Feature Transform)
- Color Features (RGB, HSV, ...)
- Statistical based Features
- ...
- ...

** Recall W08-Lecture*

Learning based Framework

** Recall W01/02/03-Lecture*

- Image Normalisation
- Illumination Correction
- Noise Removal
- Histogram Equalisation
- Contrast Enhancement
- Morphological Operations
- ...
- ...

- SVM
- kNN
- Decision Tree
- Random Forest
- Multi-layer Perceptron
- Naive Bayes
- ...
- ...

$$Y = F(X, \theta)$$

- *Y – Output/Label*
- *X – Feature Vectors*
- ***F – Classifier***
- *Θ – Classifier's Learnable Parameters*

Inputs → Pre-processing → Feature Extraction → Classifier → Post-processing → Output

- Edge Detection (Sobel, Canny, Prewitt, Roberts, ...)
- HOG (Histogram of oriented gradient)
- SIFT (Scale-invariant Feature Transform)
- Color Features (RGB, HSV, ...)
- Statistical based Features
- ...
- ...

** Recall W08-Lecture*

Learning based Framework

** Recall W01/02/03-Lecture*

- Image Normalisation
- Illumination Correction
- Noise Removal
- Histogram Equalisation
- Contrast Enhancement
- Morphological Operations
- ...
- ...

- SVM
- kNN
- Decision Tree
- Random Forest
- Multi-layer Perceptron
- Naive Bayes
- ...
- ...

$$Y = F(X, \theta)$$

- *Y – Output/Label*
- *X – Feature Vectors*
- *F – Classifier*
- *Θ – Classifier's Learnable Parameters*

Inputs → Pre-processing → Feature Extraction → Classifier → Post-processing → Output

- Edge Detection (Sobel, Canny, Prewitt, Roberts, ...)
- HOG (Histogram of oriented gradient)
- SIFT (Scale-invariant Feature Transform)
- Color Features (RGB, HSV, ...)
- Statistical based Features
- ...
- ...

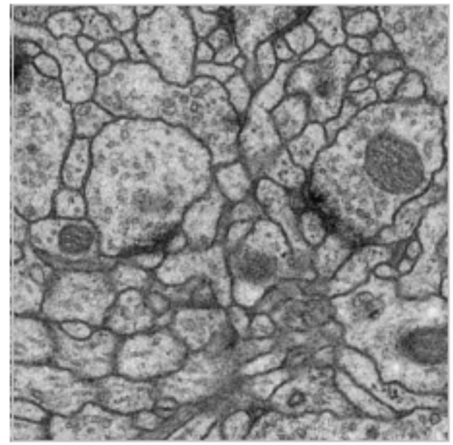
- CRF (Conditional Random Fields)
- Morphological Operations
- Refinement
- ...
- ...

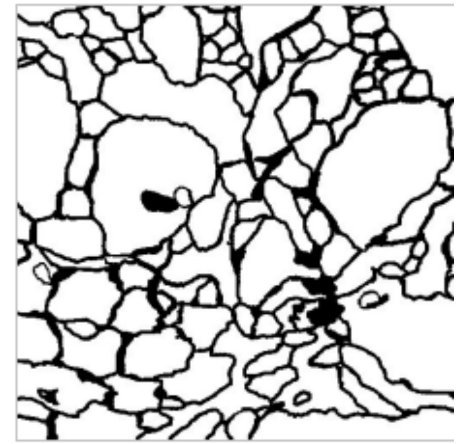
** Recall W08-Lecture*

Learning based
Image Content Classification

Task Formulation

Segmentation



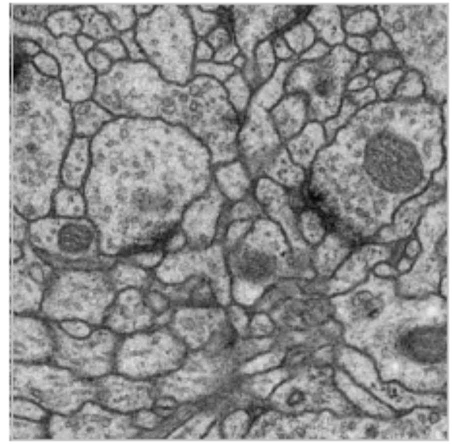


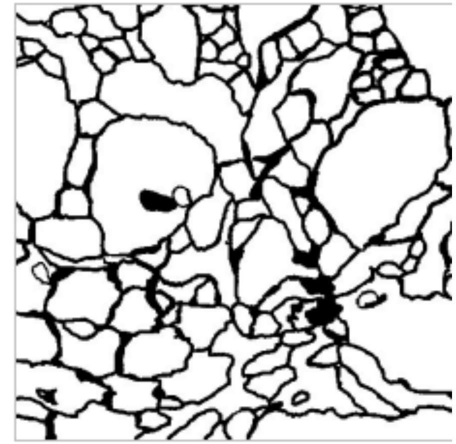
- The ISBI 2012 EM Segmentation Challenge dataset contains 30 ssTEM images taken from Drosophila larva ventral nerve cord (VNC).
- The objective is to segment the **neuron membranes** as indicated in the ground truth masks.

Albert Cardona, Stephan Saalfeld, Stephan Preibisch, Benjamin Schmid, Anchi Cheng, Jim Pulokas, Pavel Tomancak and Volker Hartenstein (10, 2010), "An Integrated Micro- and Macroarchitectural Analysis of the Drosophila Brain by Computer-Assisted Serial Section Electron Microscopy", PLoS Biol (Public Library of Science) 8 (10): e1000502, doi:10.1371/journal.pbio.1000502

Learning based
Image Content Classification

Segmentation

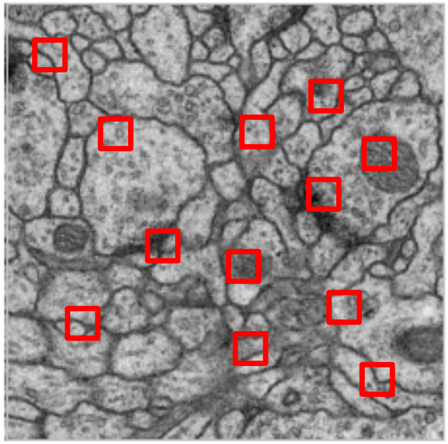


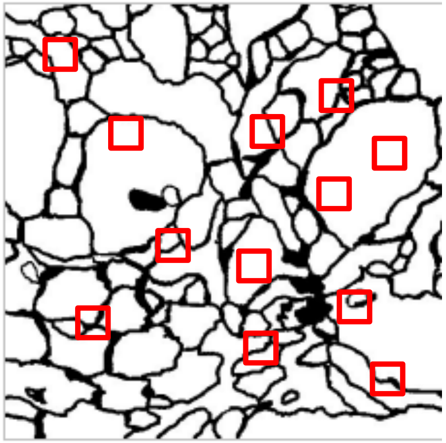


- The ISBI 2012 EM Segmentation Challenge dataset contains 30 ssTEM images taken from Drosophila larva ventral nerve cord (VNC).
- The objective is to segment the neuron membranes as indicated in the ground truth masks.
- **Patch Extraction**: Crop <u>small patches of size 3 x 3</u> from the original ssTEM images

Learning based
Image Content Classification

Segmentation





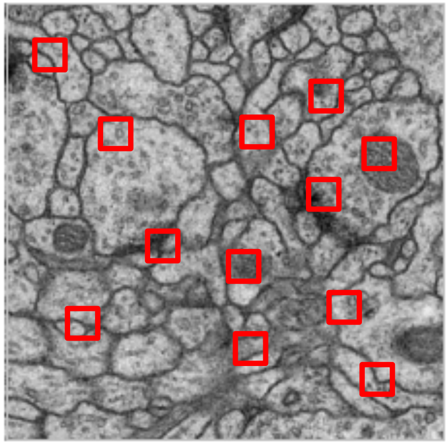
- The ISBI 2012 EM Segmentation Challenge dataset contains 30 ssTEM images taken from Drosophila larva ventral nerve cord (VNC).
- The objective is to segment the neuron membranes as indicated in the ground truth masks.
- **Patch Extraction**: Crop small patches of size 3 x 3 from the original ssTEM images

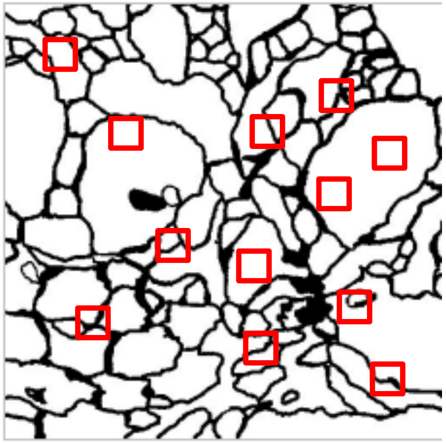
Learning based Image Content Classification

Task Formulation

~~Segmentation~~

Patch Classification





- The ISBI 2012 EM Segmentation Challenge dataset contains 30 ssTEM images taken from Drosophila larva ventral nerve cord (VNC).
- The objective is to segment the neuron membranes as indicated in the ground truth masks.
- **Patch Extraction**: Crop small patches of size 3 x 3 from the original ssTEM images
- Q: For each patch extracted, whether its centroid pixel is part of neuron membranes?

Learning based Image Content Classification

Framework Design

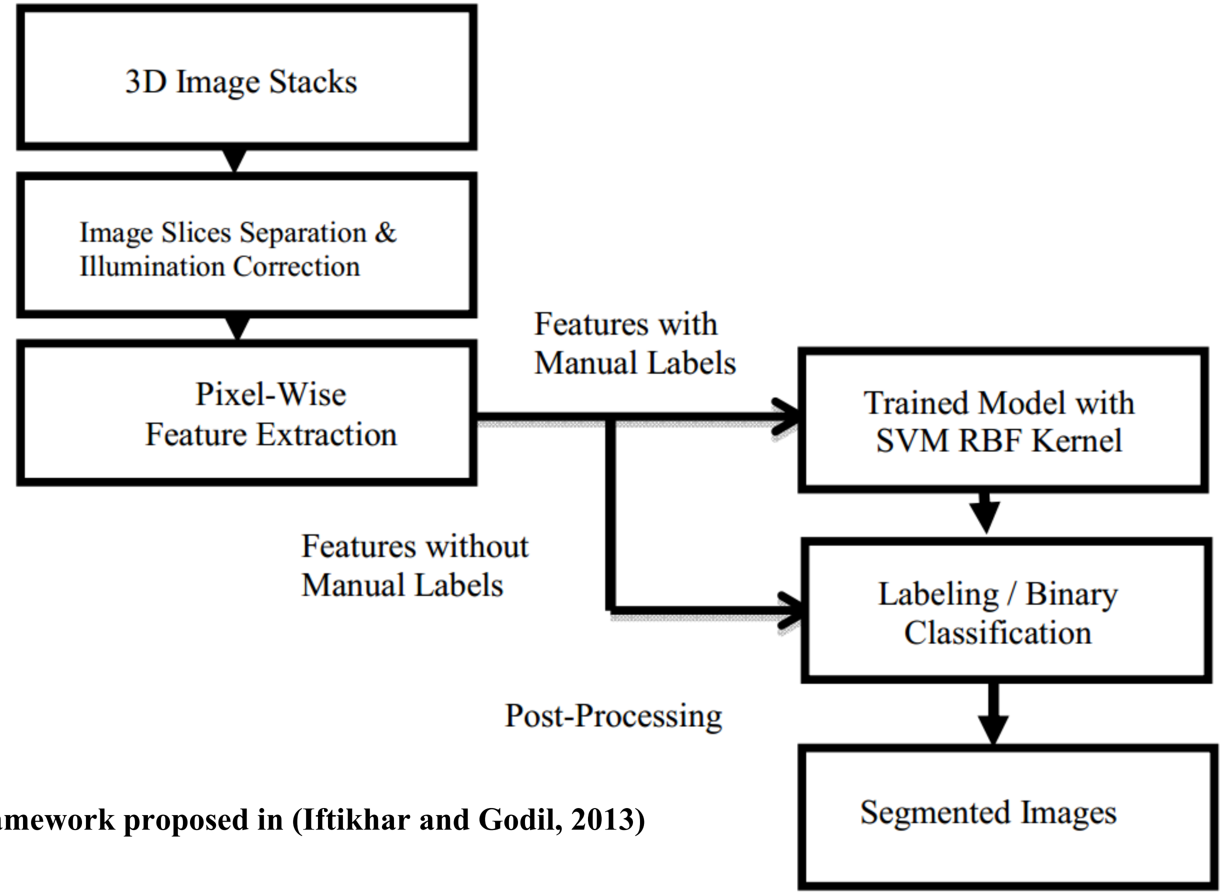


Figure: Pipeline of framework proposed in (Iftikhar and Godil, 2013)

Iftikhar, S. and Godil, A. (2013). Feature measures for the segmentation of neuronal membrane using a machine learning algorithm. In Sixth International Conference on Machine Vision (ICMV 2013), volume 9067, page 90670V. International Society for Optics and Photonics.

Learning based Image Content Classification

Framework Design

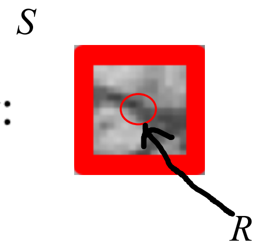
- **Image Pre-processing**: illumination correction.
- **Patch Subsetting**
  - 3 by 3 neighbourhood patches extracted from the raw images.
- **Feature Selection (for pixels)**
  - A set of 24 distinct features is computed from each neighbourhood patches, followed by the feature normalization step.
- **Pixel Classifier**
- **SegMask Post-processing**

Framework Design

- **Image Pre-processing**
- **Patch Subsetting**
- **Feature Selection**
- **Pixel Classifier**
  - An SVM-RBF based classifier.
- **SegMask Post-processing**
  - Morphological operations
  - Shape-related thresholding

Learning based Image Content Classification

- **Details of Feature Selection**
  - CandidatePixel **R** → NeighbourPatch **S** → FeatureVector **T**
  - To perform a pixel-level classification for a **candidate pixel R**, its 3 by 3 **neighbourhood patch S** is extracted firstly, then a **feature vector T** of length 24 is computed, including:
    - the intensities (3x3) of patch S.
    - the median value (1) of the intensities of patch S.
    - the range (1) of patch S, which equals to $Max(S) - Min(S)$.
    - the energy (1) of patch S, which equals to $\sum_{i \in S} i^2$.
    - the 2rd, 3nd and 4th spatial moments (3) of patch S, which equals to $\frac{1}{9} \sum_{i \in S} (i - \mu)^r$ for $r = 2, 3$ and 4 respectively, where μ is the mean value of patch S.
    - the gradients (3x3) of patch S, which can be used to increase the visibility of edges and other details.

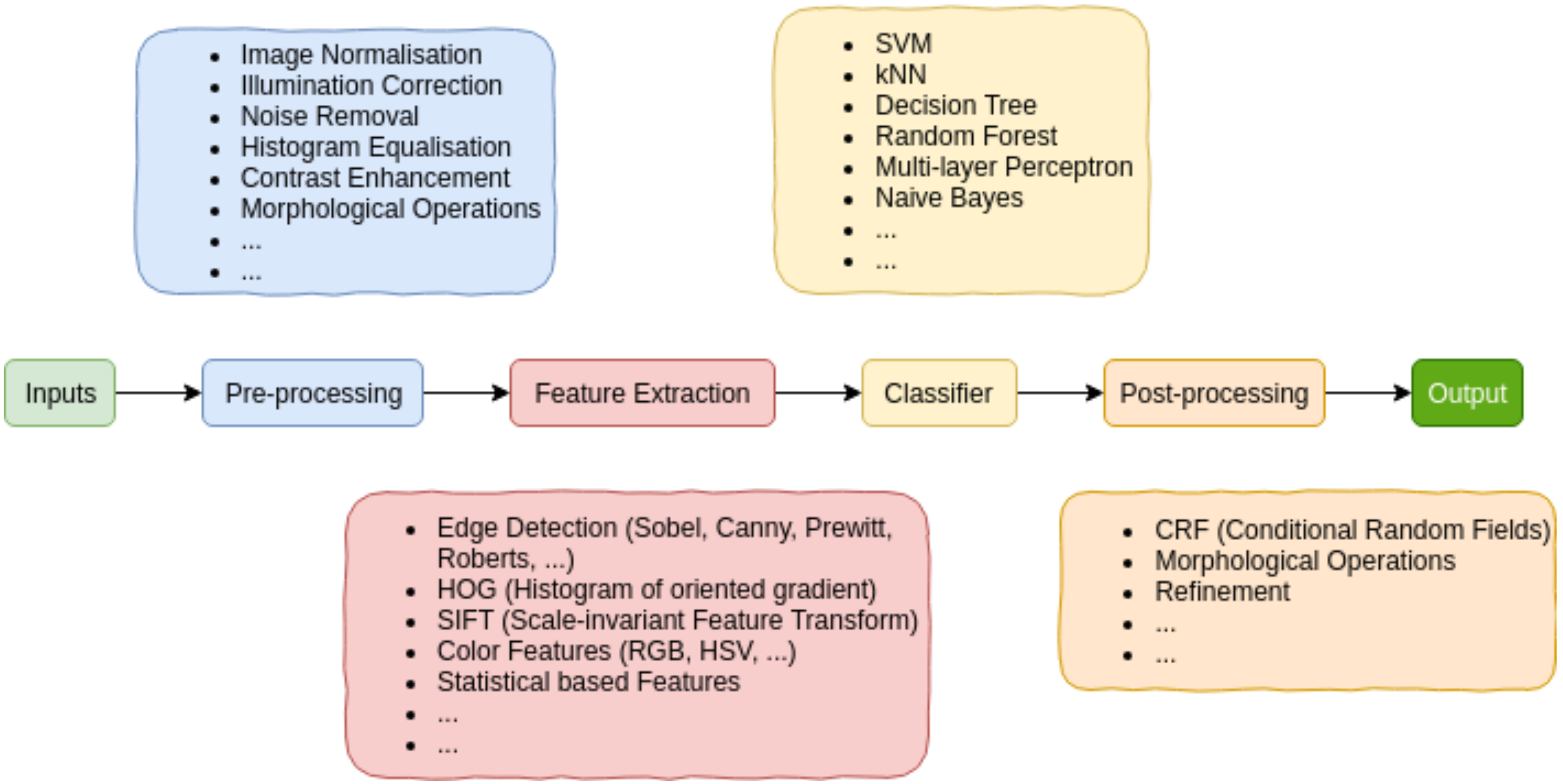


  - **Feature normalization** is essential to make sure that each feature is normalized to a range between [-1, +1].

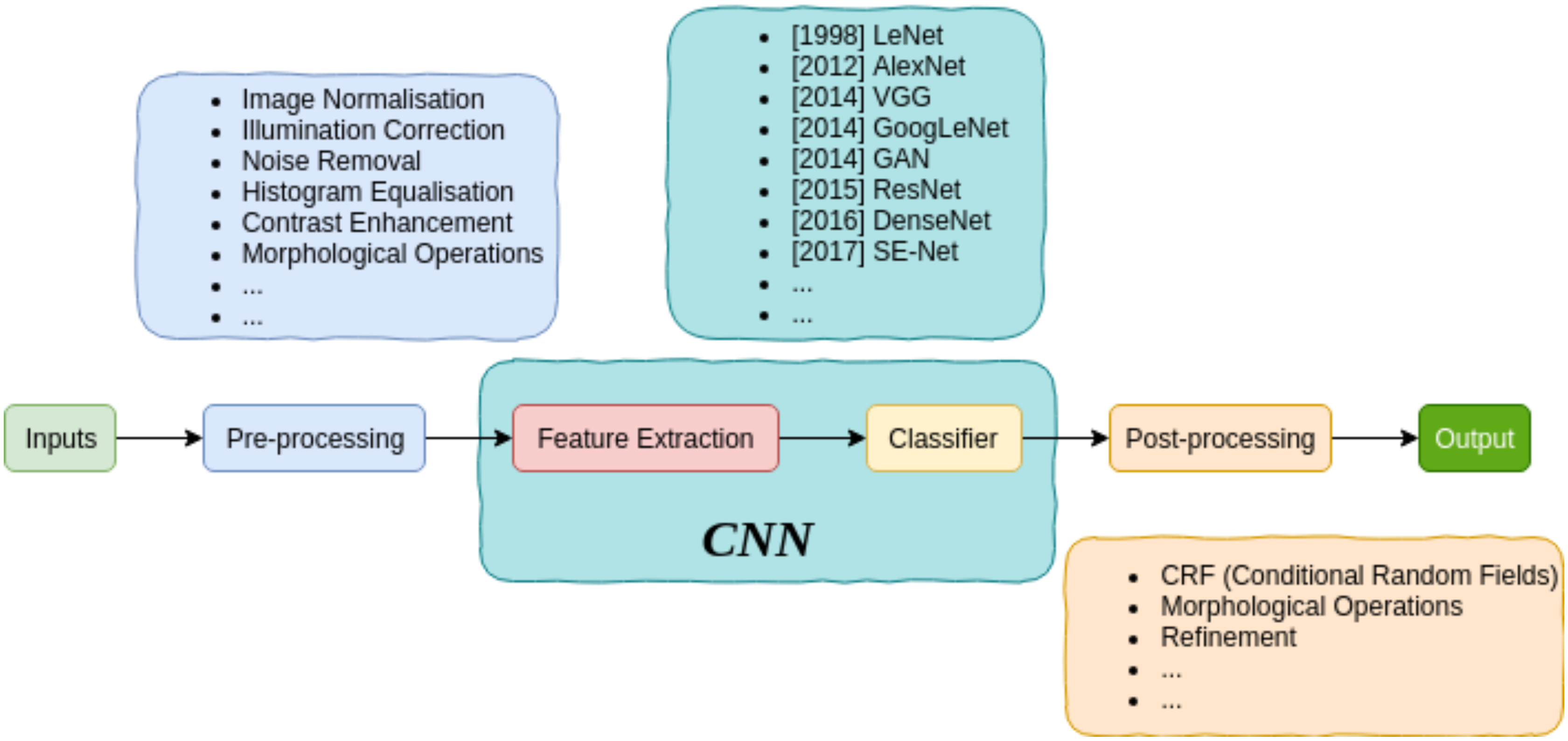
Learning based Image Content Classification

- **Details of Pixel Classifier**
  - An SVM with RBF kernel:
    - SVM: support vector machine.
    - RBF (radial basis function): $K(x, x') = exp(-\gamma||x - x'||_2^2)$, where $\gamma = \frac{1}{2\sigma^2}$.
  - The optimal setting of the related parameters, including gamma (width of the kernel), cost and weight, was obtained by setting different ranges of them from 10-fold cross validation.
  - LIBSVM package.

Learning based Framework



(Deep) Learning based Framework



Deep Learning based Image Segmentation

U-Net Architecture Design

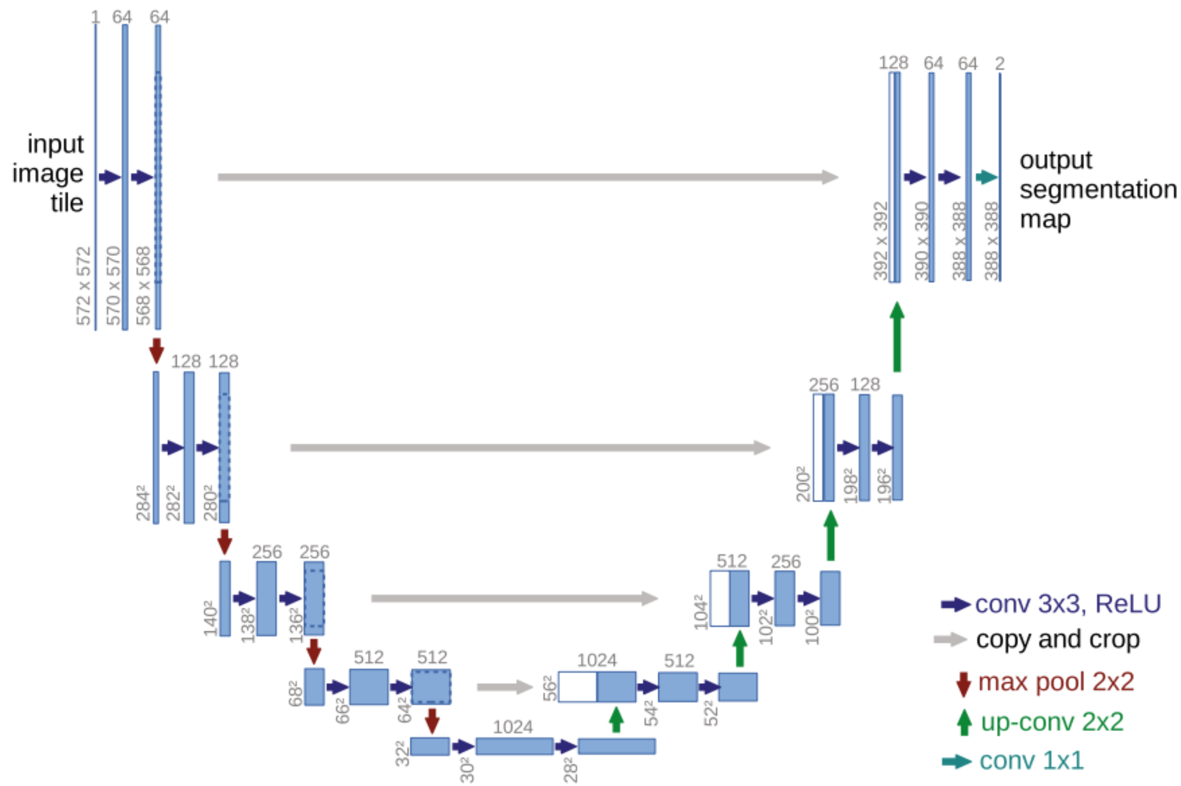


Figure: U-Net Architecture (Ronneberger et al., 2015)

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer.

Hold on… Hold on … Hold on …

Deep Learning for Visual Content Analysis and Processing

---- A Quick Introduction

Deep (Convolutional) Neural Network

Deep (Convolutional) Neural Network

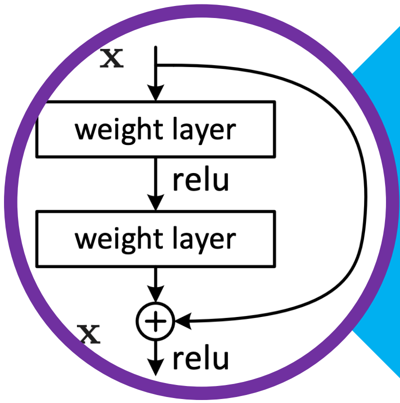
Network **Learning** Procedure

Deep Convolutional Neural Network & Its Learning Procedure

---- A Quick Introduction

Deep Convolutional Neural Network & Its Learning Procedure

---- A Quick Introduction

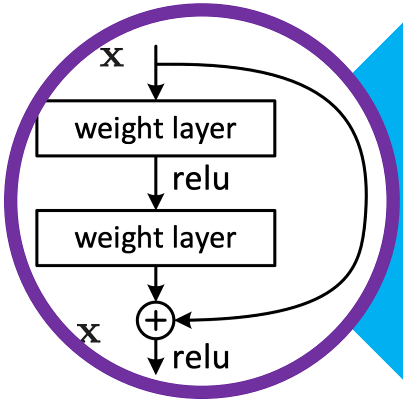


CNN Components:

- Convolution Layer
- Activation Layer
- Pooling Layer
- Batch Normalization Layer
- Dropout Layer
- Fully Connected Layer
- …

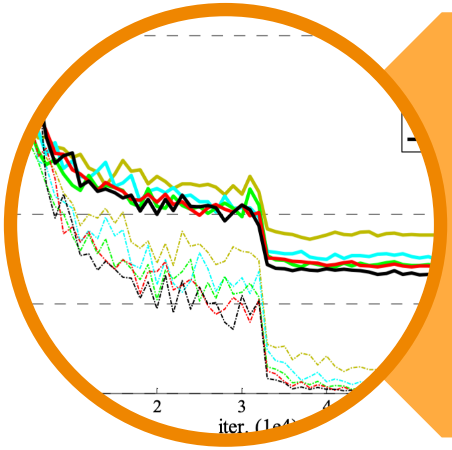
Deep Convolutional Neural Network & Its Learning Procedure

---- A Quick Introduction



CNN Components:

- Convolution Layer
- Activation Layer
- Pooling Layer
- Batch Normalization Layer
- Dropout Layer
- Fully Connected Layer
- …

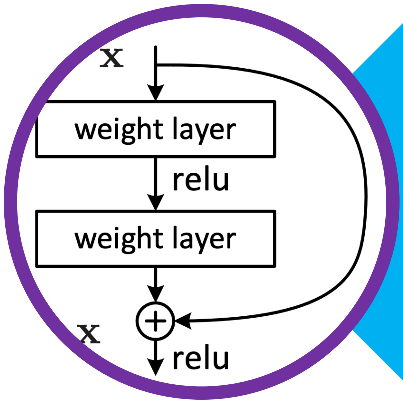


Learning Elements:

- Loss Functions
- Training Policy: Back-propagation
- Optimizer
- Scheduler
- Data Augmentation
- …

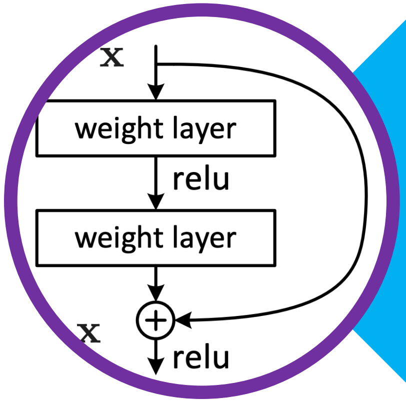
Deep Convolutional Neural Network & Its Learning Procedure

---- A Quick Introduction



CNN Components:

- **Convolution Layer**
- **Activation Layer**
- **Pooling Layer**
- **Batch Normalization Layer**
- **Dropout Layer**
- **Fully Connected Layer**
- **…**



CNN Components:

- Convolution Layer
- Activation Layer
- Pooling Layer
- Batch Normalization Layer
- Dropout Layer
- Fully Connected Layer
- …

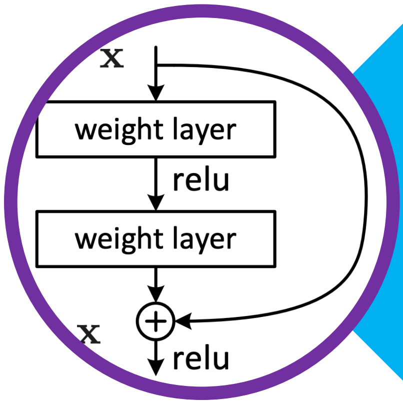
Convolution Layer

- Number of kernels: more than 1 kernel
- Learnable weights (instead of hand-crafted ones, like smooth filter or sharpen filter)

**Recall Convolution Operation From W01-Lecture*

Deep Convolutional Neural Network & Its Learning Procedure

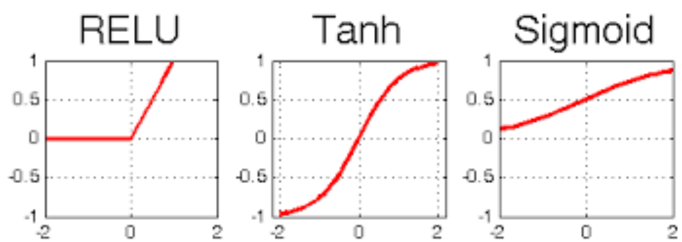
---- A Quick Introduction



CNN Components:

- Convolution Layer
- Activation Layer
- Pooling Layer
- Batch Normalization Layer
- Dropout Layer
- Fully Connected Layer
- …

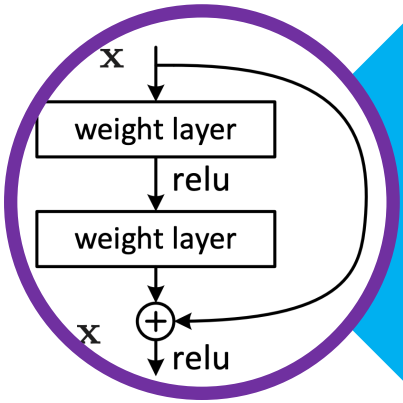
Activation Layer



- **Sigmoid**
- **Relu**
- **Tanh**
- **Softmax**
- **LeaklyRelu**
- **…**

Deep Convolutional Neural Network & Its Learning Procedure

---- A Quick Introduction



CNN Components:

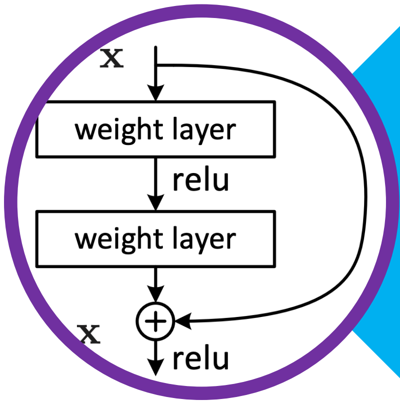
- Convolution Layer
- Activation Layer
- Pooling Layer
- Batch Normalization Layer
- Dropout Layer
- Fully Connected Layer
- …

Pooling Layer

- **Max-pooling**
- **Average-pooling**
- **Global Max-pooling**
- **Global Average-pooling**
- **…**

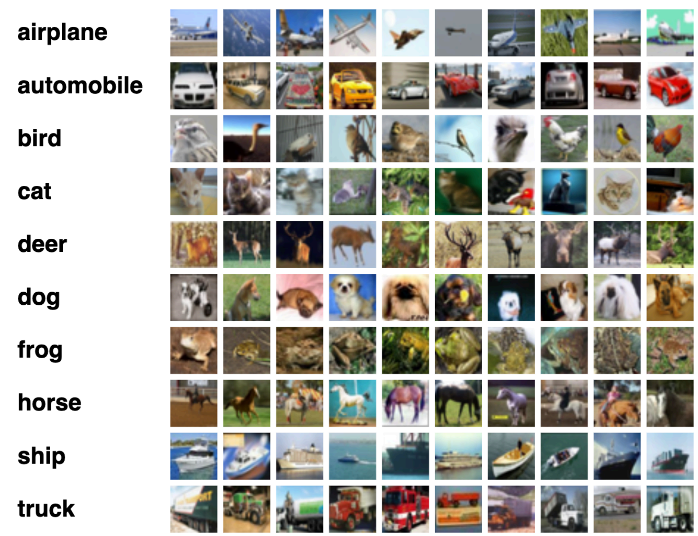
Deep Convolutional Neural Network & Its Learning Procedure

---- A Quick Introduction



CNN Components:

- **Convolution Layer**
- **Activation Layer**
- **Pooling Layer**
- **Batch Normalization Layer**
- **Dropout Layer**
- **Fully Connected Layer**
- **…**



Let’s design a toy CNN!

for an image classification task:

- Input (X): images (3, 32, 32)
- Output (Y): 10 classes

Krizhevsky, Alex. (2012). Learning Multiple Layers of Features from Tiny Images. University of Toronto.

Deep Convolutional Neural Network & Its Learning Procedure

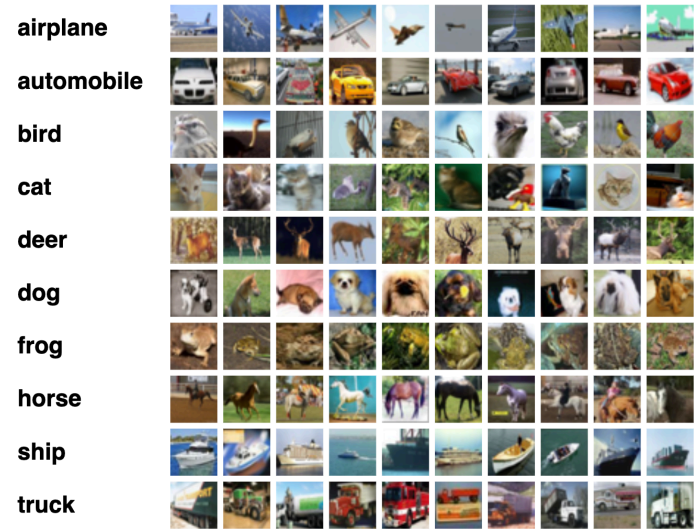
---- A Quick Introduction

Let’s design a toy CNN!

for an image classification task:

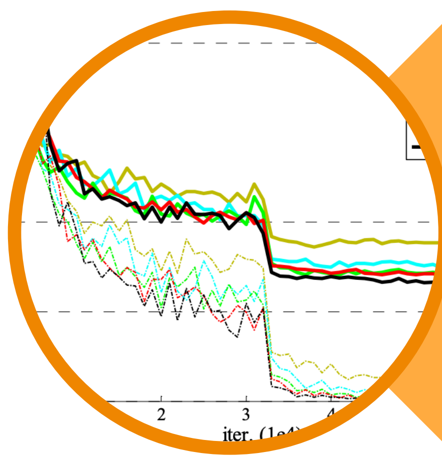
- Input (X): images (3, 32, 32)
- Output (Y): 10 classes

```
# Define the model
model = Sequential()
model.add(Convolution2D(48, 3, 3, border_mode='same', input_shape=(3, 32, 32)))
model.add(Activation('relu'))
model.add(Convolution2D(48, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Convolution2D(96, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(96, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Convolution2D(192, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(192, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```



(Personal Preference) Pytorch > Tensorflow > Keras

How to train it?



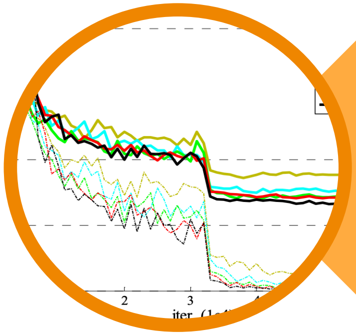
Learning Elements:

- Loss Functions
- Training Policy: Back-propagation
- Optimizer
- Scheduler
- Data Augmentation
- …

Loss Functions

- **A loss function is used to compute the model's prediction accuracy from the outputs**

- The training objective is to minimise this loss,
  via iteratively *updating the network parameters*
- The loss guides the backpropagation process to train the CNN model



Learning Elements:

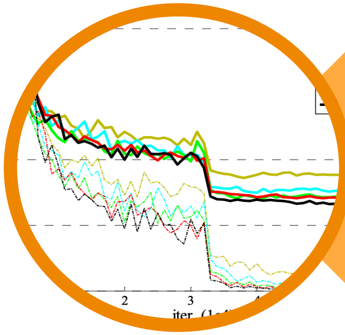
- **Loss Functions**
- **Training Policy: Back-propagation**
- **Optimizer**
- **Scheduler**
- **Data Augmentation**
- **…**

Loss Functions

- **A loss function is used to compute the <u>model's prediction accuracy</u> from the outputs**
  - Most commonly used: categorical cross-entropy loss function

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$

- The training objective is to <u>minimise this loss</u>,
  via iteratively *updating the network parameters*
- The loss guides the <u>backpropagation</u> process to train the CNN model



Learning Elements:

- **Loss Functions**
- **Training Policy: Back-propagation**
- **Optimizer**
- **Scheduler**
- **Data Augmentation**
- ...

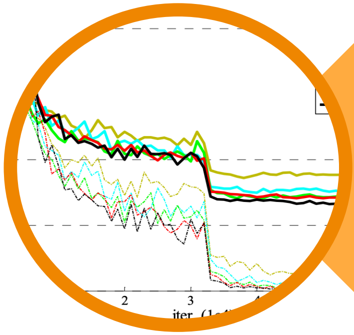
Loss Functions

- **A loss function is used to compute the <u>model's prediction accuracy</u> from the outputs**

- Most commonly used: categorical cross-entropy loss function

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$

- **The training objective is to <u>minimise this loss</u>, via iteratively *updating the network parameters***
- The loss guides the <u>backpropagation</u> process to train the CNN model



Learning Elements:

- **Loss Functions**
- **Training Policy: Back-propagation**
- **Optimizer**
- **Scheduler**
- **Data Augmentation**
- **…**

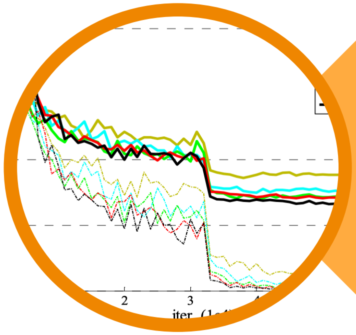
Loss Functions

- **A loss function is used to compute the model's prediction accuracy from the outputs**

  - Most commonly used: categorical cross-entropy loss function

$$H(y, \hat{y}) = \sum_i y_i \log \frac{1}{\hat{y}_i} = -\sum_i y_i \log \hat{y}_i$$

- **The training objective is to minimise this loss,**
  via iteratively *updating the network parameters*
- **The loss guides the backpropagation process to train the CNN model**



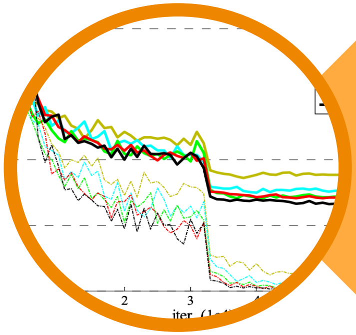
Learning Elements:
- **Loss Functions**
- **Training Policy: Back-propagation**
- **Optimizer**
- **Scheduler**
- **Data Augmentation**
- **…**

Loss Functions

Goal: obtain an optimal set of weights, resulting in the minimum loss.

How to achieve that?

- Find the weights that make the derivative of loss function equals zero, i.e., local extrema.
- Iterative approach: Gradient Descent optimization algorithm



Learning Elements:

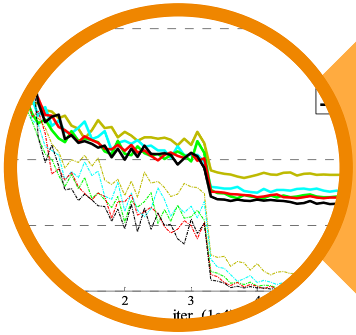
- **Loss Functions**
- **Training Policy: Back-propagation**
- **Optimizer**
- **Scheduler**
- **Data Augmentation**
- **…**

Loss Functions

Goal: obtain an optimal set of weights, resulting in the minimum loss.

How to achieve that?

- Find the weights that make the derivative of loss function equals zero, i.e., local extrema.
- Iterative approach: Gradient Descent optimization algorithm



Learning Elements:

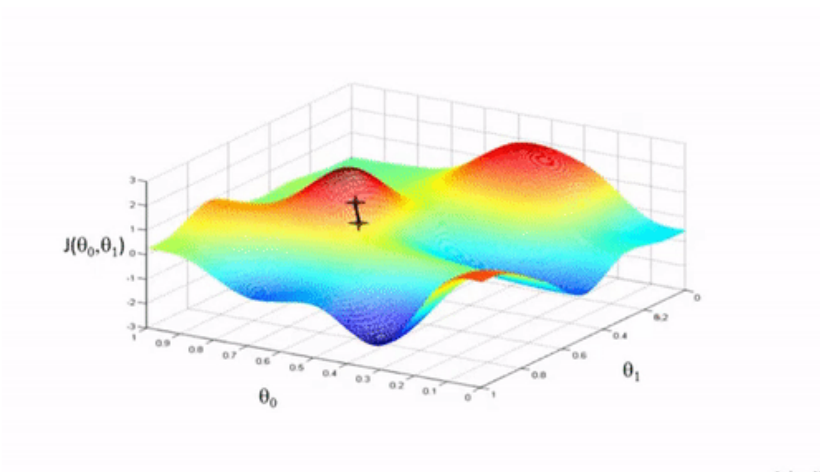
- **Loss Functions**
- **Training Policy: Back-propagation**
- **Optimizer**
- **Scheduler**
- **Data Augmentation**
- **…**

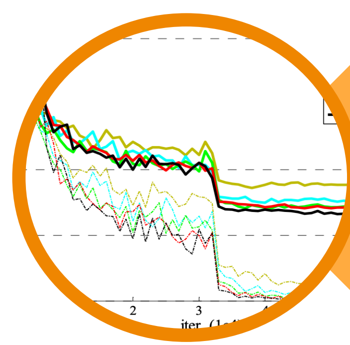
Loss Functions

Goal: obtain an optimal set of weights, resulting in the minimum loss.

How to achieve that?

- Find the weights that make the derivative of loss function equals zero, i.e., local extrema.
- Iterative approach: Gradient Descent optimization algorithm





Learning Elements:
- **Loss Functions**
- Training Policy: Back-propagation
- Optimizer
- Scheduler
- Data Augmentation
- …

Training with Back-propagation

Backpropagation is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function;

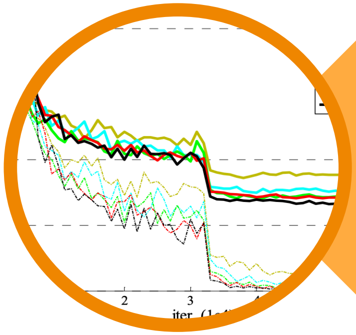
Old weight

Derivative of Error with respect to weight

$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

New weight

Learning rate



Learning Elements:

- Loss Functions
- Training Policy: Back-propagation
- Optimizer
- Scheduler
- Data Augmentation
- …

http://hmkcode.com/ai/backpropagation-step-by-step/

Optimizer

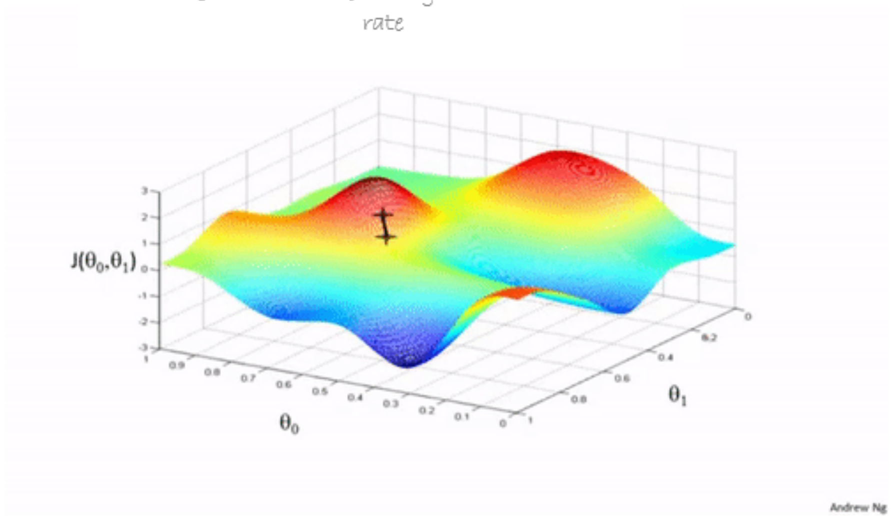
Derivative of Error with respect to weight

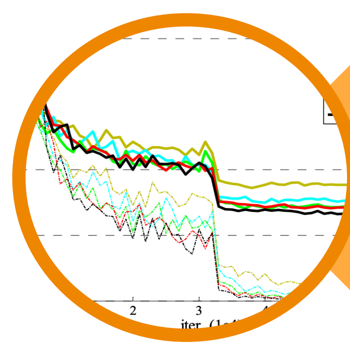
Old weight

$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$

New weight

Learning rate

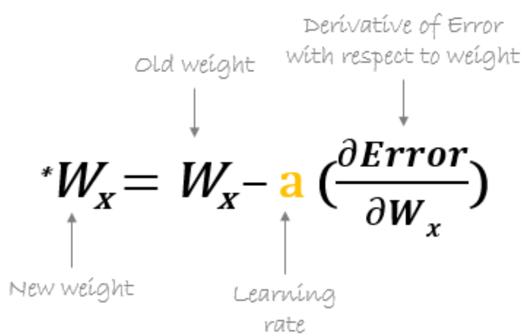




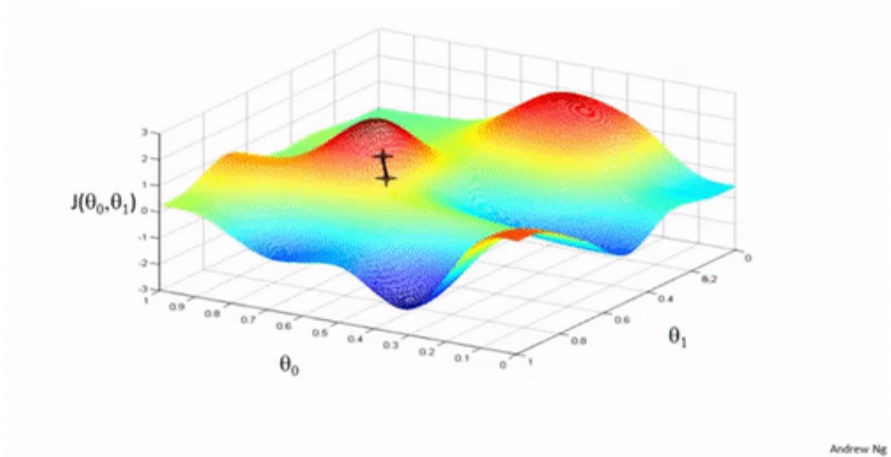
Learning Elements:

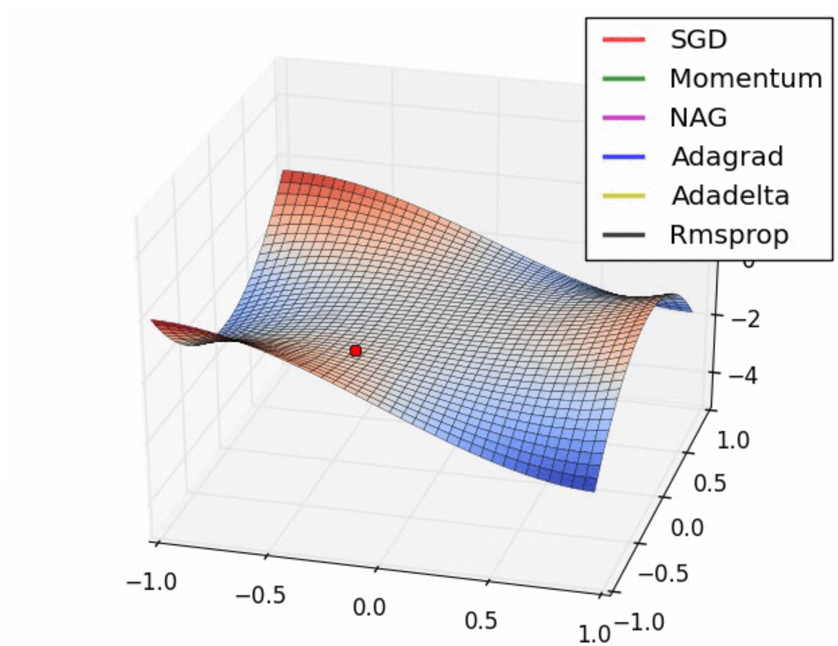
- Loss Functions
- Training Policy: Back-propagation
- **Optimizer**
- Scheduler
- Data Augmentation
- …

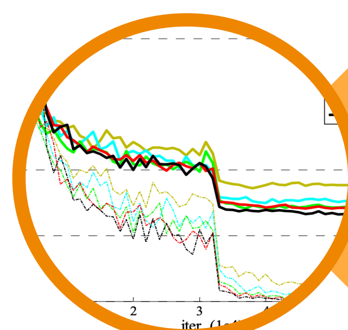
Optimizer



$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right)$$







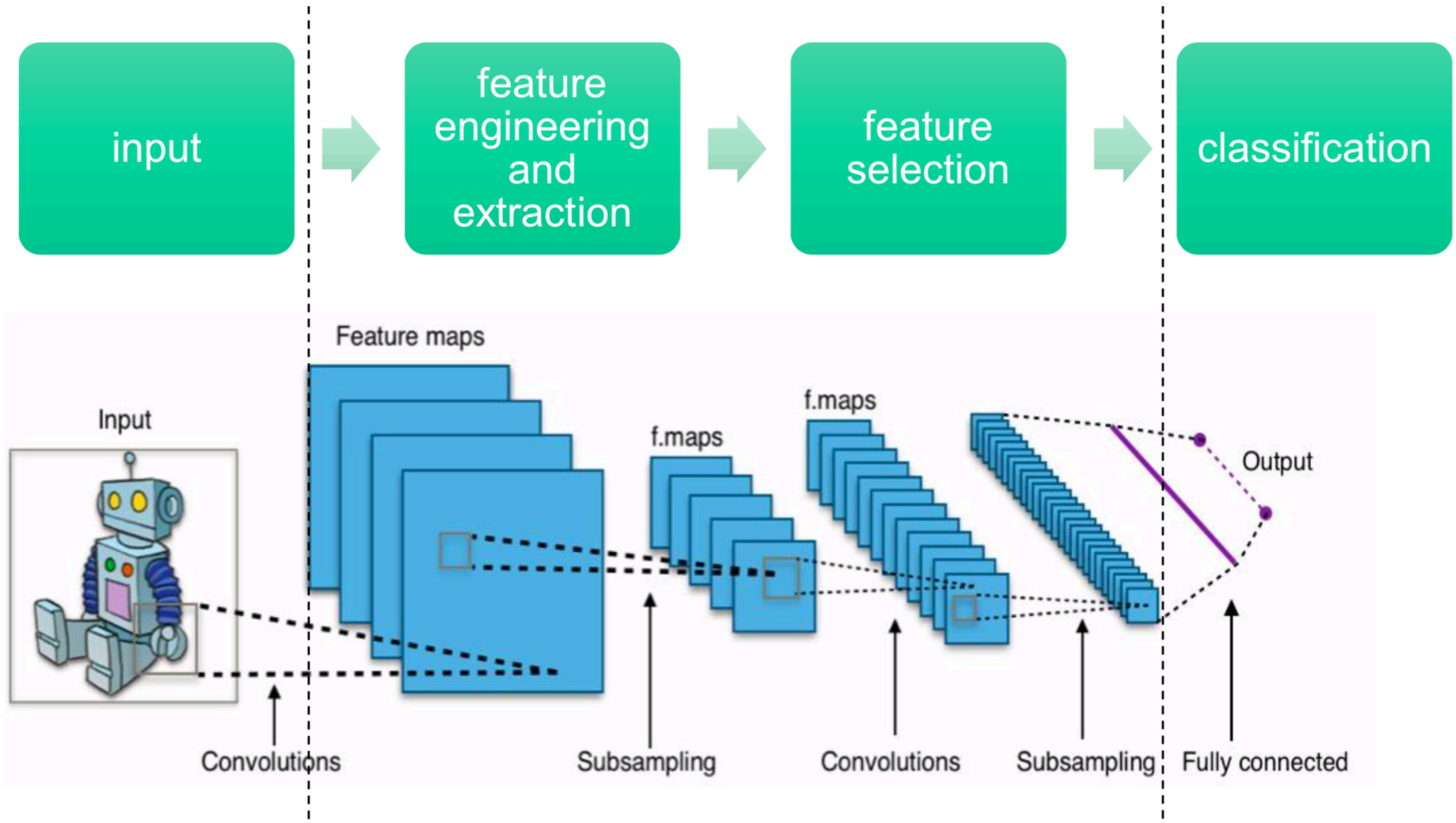
Learning Elements:

- Loss Functions
- Training Policy: Back-propagation
- **Optimizer**
- Scheduler
- Data Augmentation
- …

Deep Convolutional Neural Network & Its Learning Procedure

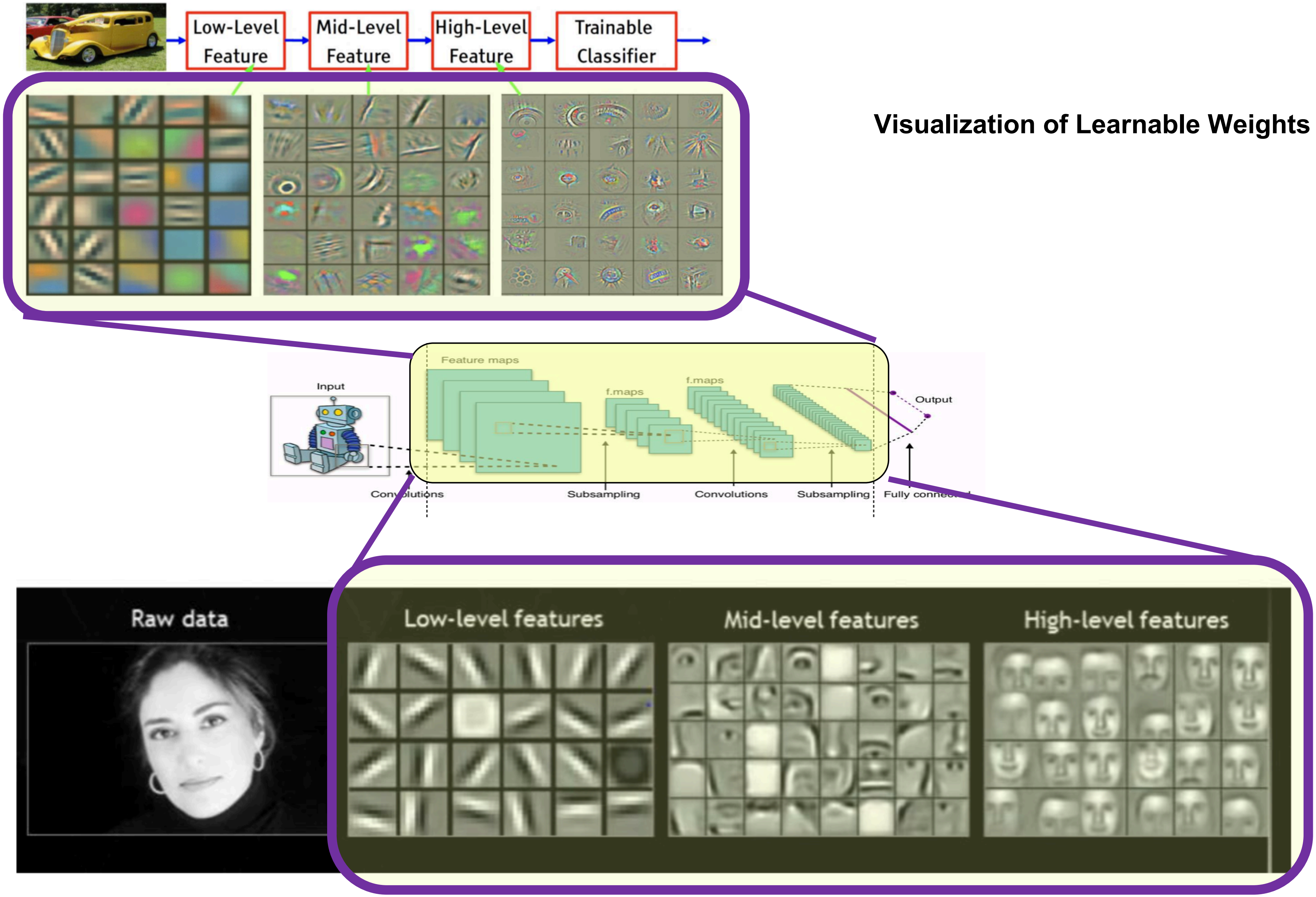
---- A Quick Introduction

Traditional Approach vs DL



https://towardsdatascience.com/convolutional-neural-networks-for-all-part-i-cdd282ee7947

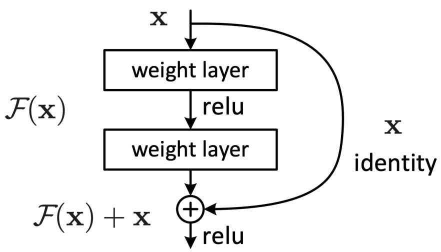
Visualization of Learnable Weights



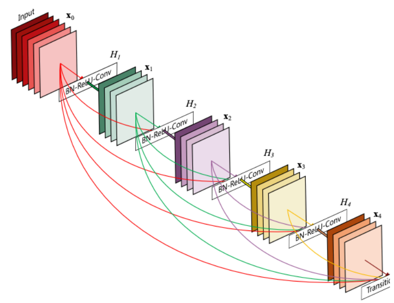
https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/

Deep Convolutional Neural Network & Its Learning Procedure

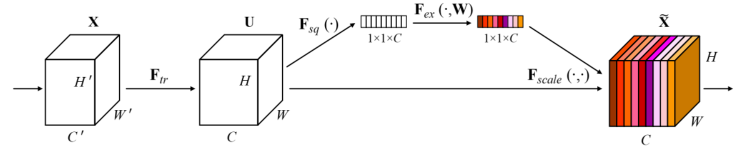
---- A Quick Introduction



ResNet



DenseNet



SENet

- *He, K., Zhang, X., Ren, S., & Sun, J. (2016). **Deep residual learning for image recognition.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).*
- *Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). **Densely connected convolutional networks.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).*
- *Hu, J., Shen, L., & Sun, G. (2018). **Squeeze-and-excitation networks.** In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).*

Back to our case study of deep learning based image segmentation…

Deep Learning based Image Segmentation

U-Net Architecture Design

- **Skip connections**. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization.
- An efficient solution with **limited dataset provided**.
- To output a 2D segmentation mask (rather than a global image label), the network does not have any fully connected layers, and only uses the valid part of each convolution, which is quite similar to FCN (**Fully Convolutional Network**) proposed as (Long et al., 2015).

Deep Learning based Image Segmentation

U-Net Architecture Design

Details of Objective/Loss Function

- **Softmax function** applied pixel-wisely over the final featuremap of the network, to convert the outputs of last activation function to probabilities.
- **Weighted cross entropy function** adopted as the loss function for the back-propagation step.

$$E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$

Figure: Energy function for the training of U-Net

- **Weight map** $w(x)$ [next slide] is pre-computed for each ground truth segmentation.

Deep Learning based Image Segmentation

U-Net Architecture Design

Details of Weight Map $w(x)$

$$E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x}))$$

- Motivation: to force the network to learn the small separation borders between touching cells.

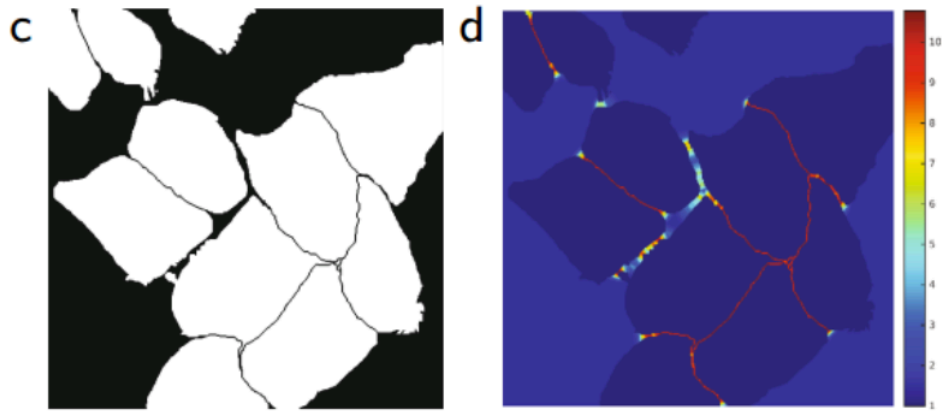


Figure: Left: binary GT mask; Right: weight map $w(x)$ pre-computed.

Deep Learning based Image Segmentation

U-Net Architecture Design

Details of Weight Map $w(x)$

- How: to assign **the separating background labels between touching cells** with **a large weight** in the loss function.

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$

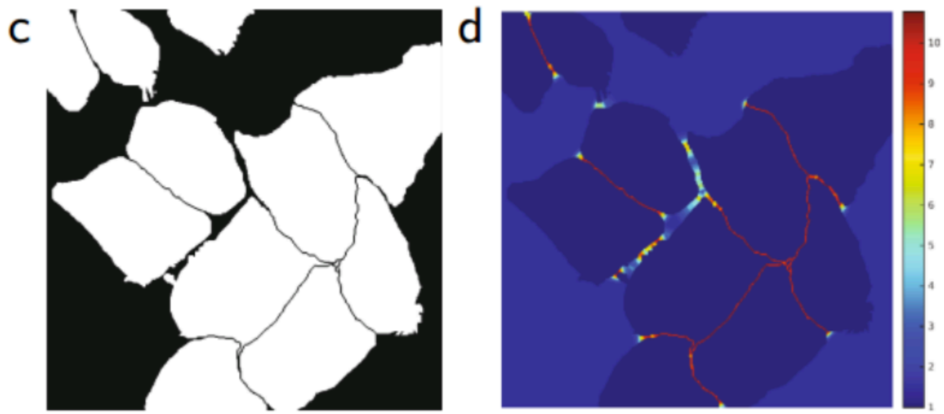


Figure: Border pixels, which are visualized in red, are assigned with large weights.

Deep Learning based Image Segmentation

Details of Hyper-parameters and Experimental Settings
(used in the original implementation of U-Net)

- Input image size: 512×512 (raw)
- Input image size: 572×572 (pre-processed via overlay-tile strategy [next slide])
- Output image size: 388×388 (unpadded convolutions)
- Weight map pre-computation: $w_0 = 10$ and $\sigma = 5$.
- Batch size: 1
- Optimizer: SGD (stochastic gradient descent) with momentum set as 0.99
- Weight initialization: gaussian distribution.

Deep Learning based Image Segmentation

Details of Data Augmentation Data augmentation is essential for the network training, especially when only few training samples are available.

- Random rotation ($[90^\circ, 180^\circ, 270^\circ]$ or other random degrees)
- Random shifting (x pixels horizontally or vertically)
- Random flipping (horizontally and vertically)
- Random elastic deformations (using random displacement vectors sampled from a Gaussian distribution with 10 pixels standard deviation, followed by bicubic interpolation)
- ...

For image segmentation tasks, both inputs and GTs should be augmented accordingly (joint transform).

Deep Learning based Image Segmentation

Training of CNN

- Record the training loss and testing loss simultaneously. Draw their curves during the training process to check whether/when the overfitting issue appears.
- Fine-tune learning rate. (Suggested by Andrew Ng's Machine Learning Course, an appropriate LR can be found via *dividing it by 3*. Put differently, try 1, 0.3, 0.1, 0.03, 0.01, 0.003, . . .).
- Weighted CE > CE, for problem with imbalanced class distribution.
- Data augmentation (on-the-fly).
- Optimizer (Try Adam at beginning).
- Dropout layers (dropout rate).
- LR warm up, LR scheduler, Weight initialization, . . .

Deep Learning based Image Segmentation

Training of CNN

- **Bag of Tricks for Image Classification with Convolutional Neural Networks** (He et al., 2019)
- https://towardsdatascience.com/a-bunch-of-tips-and-tricks-for-training-deep-neural-networks (NO.21 suggestion is very good).

Deep Learning based Image Segmentation

Evaluation - Cross Validation

- Cross-validation is required for the evaluation of your learning-based approaches.

Steps towards K-fold Cross Validation

1. Split dataset into K folds and assign $ID_{fold} \in [1, 2, \ldots, k]$ to each fold.
2. Repeat experiments for K times: for the mth time of the experiments, train and evaluate your methods proposed by choosing the mth fold ($ID_{fold} == m$) as the testing dataset and the rest $K - 1$ folds as the training dataset.
3. Calculate the average of the evaluation metrics for each experiment performed, such as accuracy,dice and IoU metrics.

Deep Learning based Image Segmentation

Evaluation - Cross Validation

- Cross-validation is required for the evaluation of your learning-based approaches.
- In other words, each single image from the dataset given should be treated as testing data (and once only) during the K-fold cross validation.