Model Inversion for Impersonation in Behavioral Authentication Systems

Md Morshedul Islam

University of Calgary

June 12, 2020

Motivation

- Behavioral Authentication (BA)
  - Use behavioral data for authentication

  (i) Registration → user profile
    - Profile → m samples, n features' value

  (ii) verification → user is verified

Feature

	F_1	F_2	F_3	....	F_n
x_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	....	$x_{1,n}$
x_2	$x_{2,1}$	$x_{2,2}$	$x_{2,2}$	....	$x_{2,n}$
...	...	...	...	....	...
x_m	$x_{m,1}$	$x_{m,2}$	$x_{m,2}$	....	$x_{m,n}$

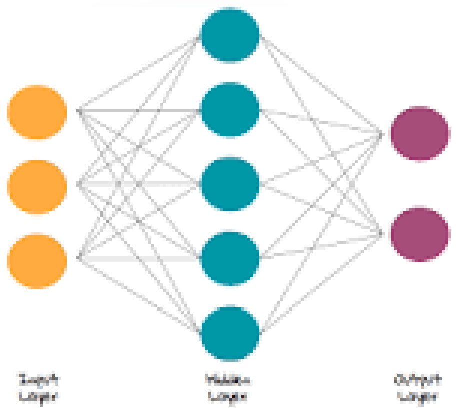
Data sample

Profile $\boldsymbol{X}$

- Artificial Neural Networks (ANNs)
  - Many applications: classification, prediction..
  - For BA authentication:

    (i) Input: user profile

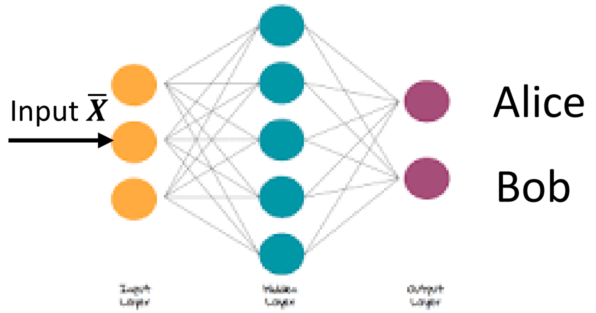
    (ii) decision: Alice or Bob



- Our Goal: Use ANN to break BA system → generate profiles that are accepted

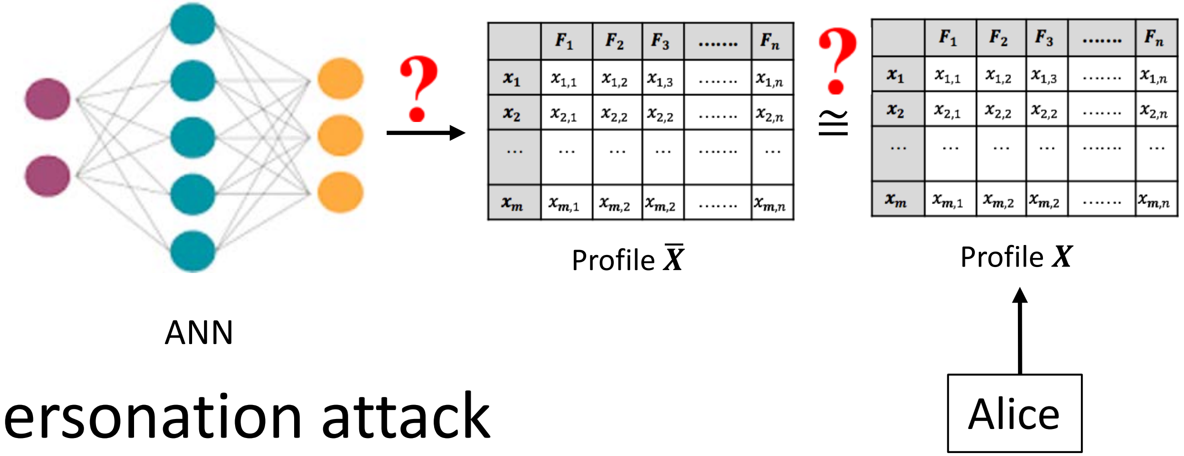
Research Question

- Attacker has
  - Access of BA classifier



Classifier

- Can attacker train an ANN
  - To generate an artificial profile $\overline{\boldsymbol{X}}$?
- Is $\overline{\boldsymbol{X}}$ close to a profile $\boldsymbol{X}$ of a target user Alice?



- BA classifier will accept claim $(Alice, \overline{\boldsymbol{X}})$ → Impersonation attack

Part I: Background

Machine Learning (ML)

- “Field of study that gives computers the ability to *learn without being explicitly programmed*” -Arthur Samuel (1959)
- Samuels wrote a checkers playing program
  - Program play 10000 games against itself

- “A computer program is said to *learn from experience E* with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, *improves with experience E*” -Tom Michel (1999)
  - The checkers example,
    - E = 10000s games
    - T is playing checkers
    - P if you win or not

Machine Learning (ML)

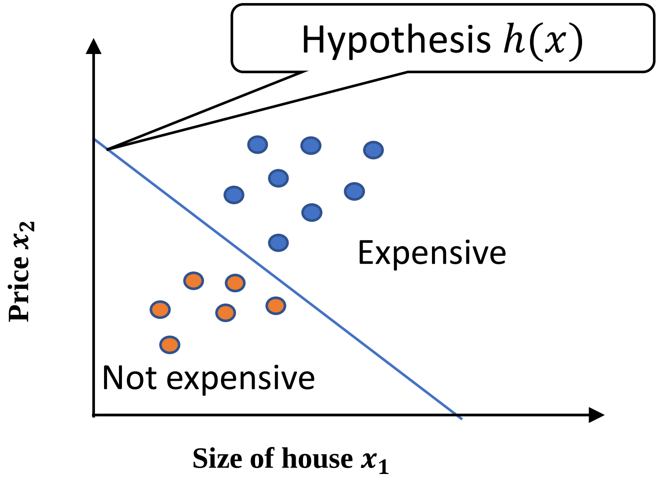
- Goal of ML: Given a data set → Build a model by ML algorithm
  - ML algorithm: Linear regression, logistic regression, k-means clustering,…

- Unsupervised learning: No label data
  - Model discover the structure of data

- Supervised learning: Data samples correctly label
  - Regression problem → Predict continuous value
    - Example: House price problem
  - Classification problem → Predict class (discrete value)
    1. Binary classification
    2. Multiclass classification
    - Example: House price category (Expensive/Not Expensive?)

ML Classification

- Binary classification → One of two classes
- Given tranning data $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
  - Features $x^{(i)} = \left(x_1^{(i)}: \text{size of house}, x_2^{(i)}: \text{pricse of house}\right)$
  - Label $y^{(i)} \in$ {Expensive=1, Not Expensive=0}



<u>Two phases:</u>

- Tanning phase: From training data
  i. Build a model → find a hypothesis function $h(x)$
  ii. Goal → minimize the classification error
- Inference phase: Infer the class of x

ML Classification

Training Phase:

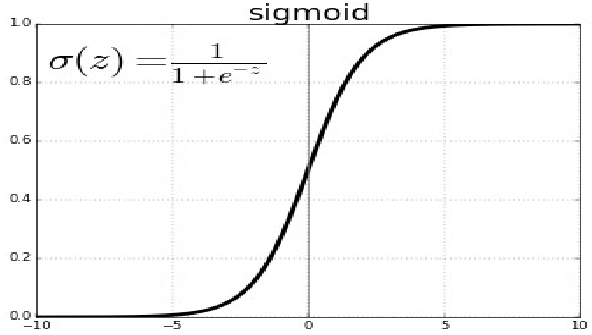
- Estimate $h(x)$→ Logistic regression
  - A classification algorithm

- A random $h(x)$

$$h(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2) = \sigma(z)$$

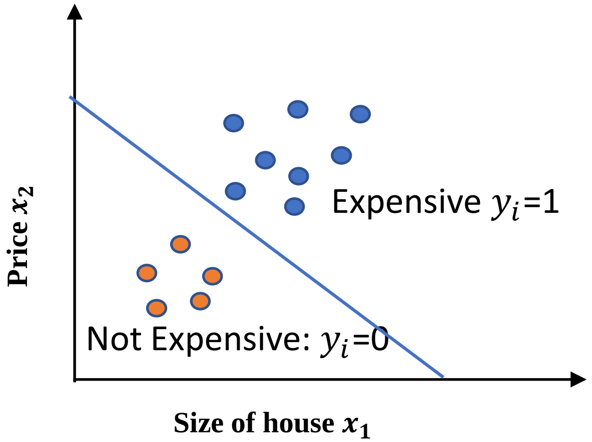
$$z = w_0 + w_1 x_1 + w_2 x_2$$

  - Activation function:
    - Sigmoid function $\sigma(.)$



- For data samples above the line
  - $h(x) \geq 0.5$

- For data samples below the line
  - $h(x) < 0.5$

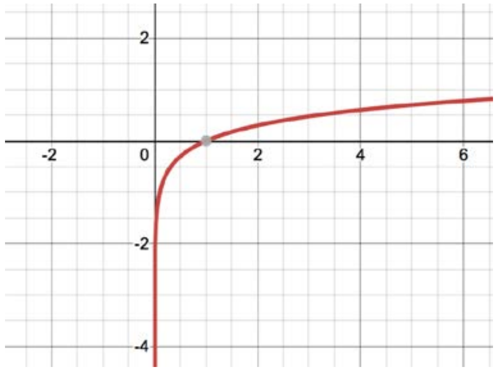


- From tanning data calculate
  - $h(x)$ → $\mathrm{p}(y = 1|(x_1, x_2); (w_0, w_1, w_2))$

ML Classification

- Loss → Use an error function (non-convex function)

$$L(y^{(i)}, h(x^{(i)})) = \begin{cases} -\log\left(h\left(x^{(i)}\right)\right) & if\ y^{(i)} = 1 \\ -\log\left(1 - h\left(x^{(i)}\right)\right) & if\ y^{(i)} = 0 \end{cases}$$



$log_{10}x$ graph

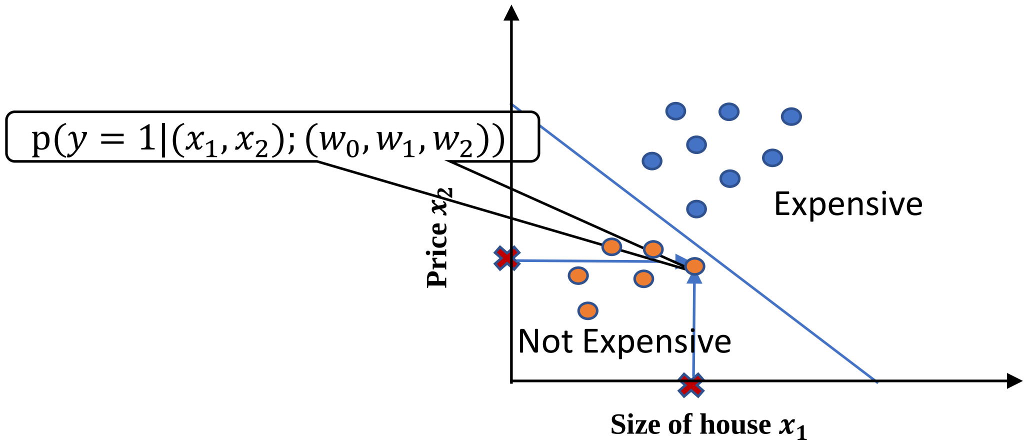
- Cost→ average loss over all samples
  - Classification error E
- Minimizing errror→Gradient decent algorithm

  Repeat {

  $$w_i = w_i - \alpha \frac{\partial}{\partial w_i} J(E)$$

  }
  - Derivation of error $\frac{\partial}{\partial w_i} J(E)$
    - with respect to all parameters

- Inference Phase:
  - Infer class of $x = (x_1, x_2)$

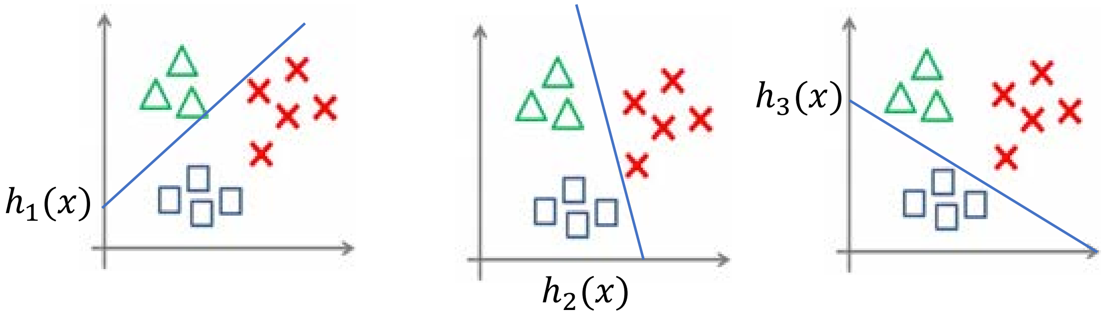


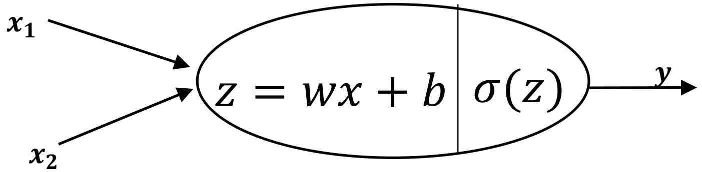
ML Classification

- Multi-class classification:
  - One of more than two discrete classes

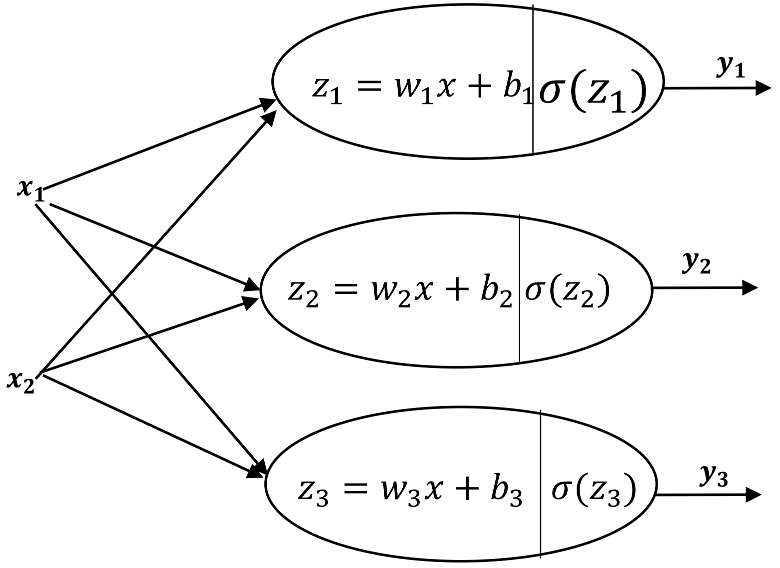
- Training phase →Train $h_i(x)$
  - $h_i(x)$ →$\mathrm{p}(y = \mathrm{i}|(x_1, x_2); (w_0, w_1, w_2))$

- Inference phase → make a prediction for x
  - Pick class i that maximizes the probability $h_i(x)$





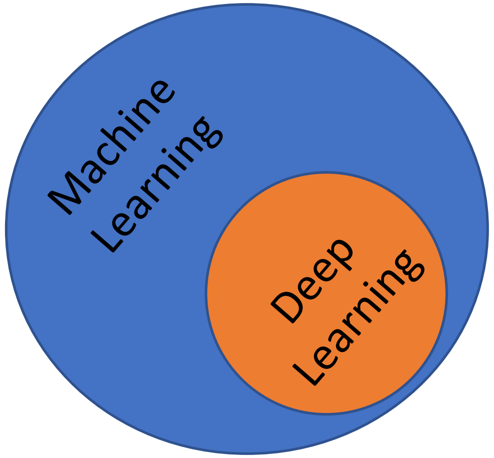
Binary classifier

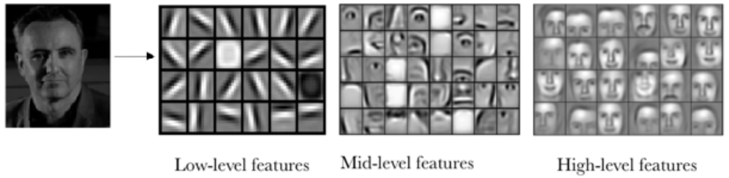


Multi-class classifier

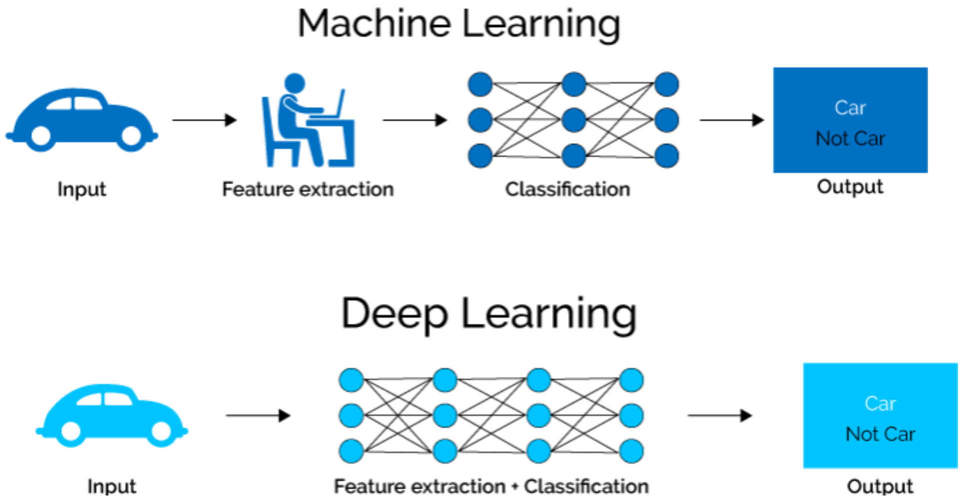
Deep Learning

- A class of machine learning algorithms
  - Uses Artificial Neural Networks (ANNs)
    - Inspired by how human brain works





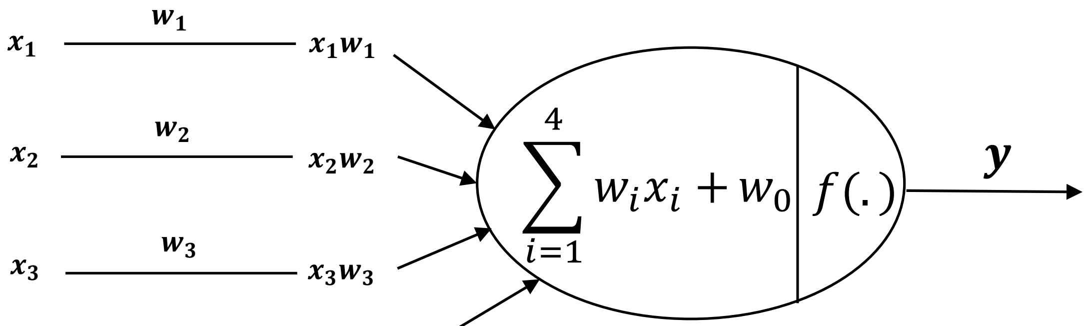
Feature extraction and classification:



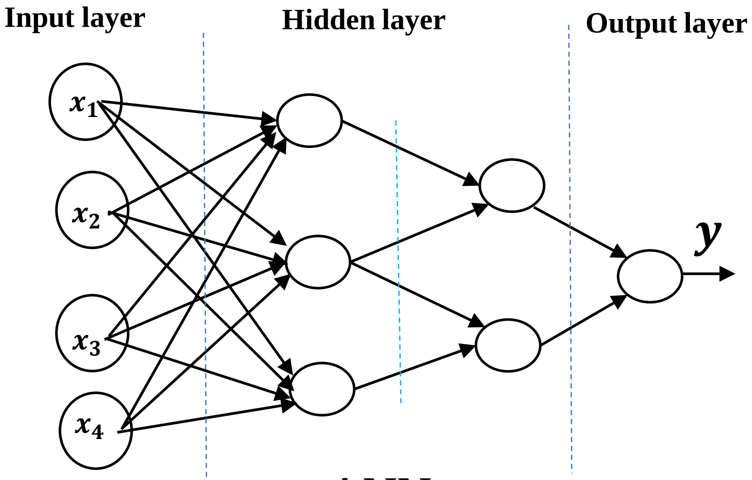
- **Algorithms:**
  - ML algorithm: Not suitable for complex problem
  - Deep learning algorithm:
    - Level of abstraction increases gradually by non-linear transformations of input data

- Applications:
  - Google Assistant
  - Amazon Alexa

Artificial Neural Networks (ANNs)



A single Neuron



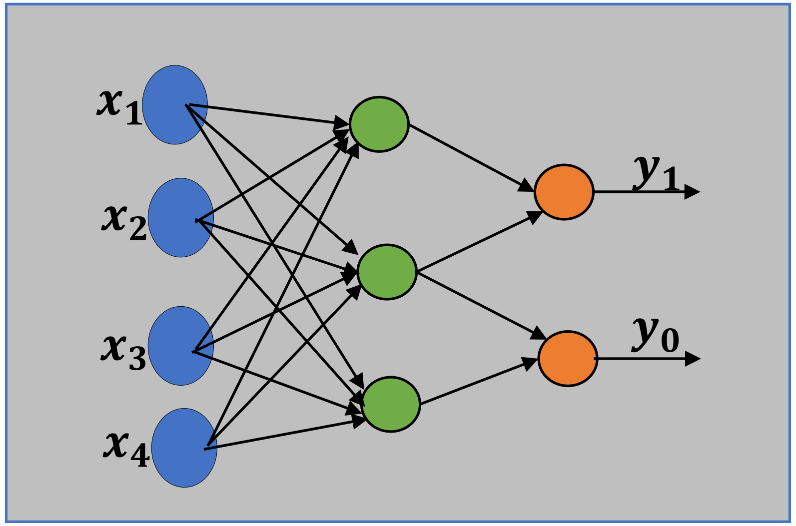
ANN

- ANNs Parameters: Weight w_i & Bias w_0/b
- Weight:
  - Numeric value multiply with inputs
  - Update weight to reduce loss
- Bias:
  - Help model to best fit with data
  - Update bias to reduce loss

- Input layer: Dimensions of input vector
- Hidden layer:
  - Divide input space into soft boundaries
- Output layer: Output of the neural network

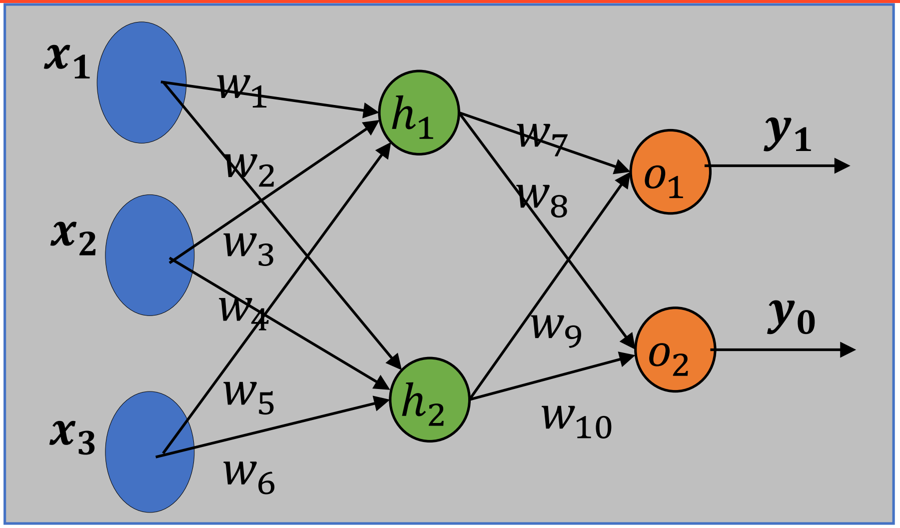
Multilayer Perceptron (MLP)

- A fully connected ANN → single/multi hidden layer
- Forward propagation → predict outputs
- Backward propagation →update parameters
- Non-linear activation function
  - ReLU, Leaky ReLU, tanh
  - Output layer uses a softmax function
- Application:
  - Complex classification, speech recognition,...



- Softmax function → prediction vector
  - Represents the probability distributions of outputs

MLP Example



Given that ,
Features value x_1=1, x_2=4, x_3=5
Output label $y_1=0.1$, $y_2=0.05$

<u>Tanning Phase:</u>

- **Step1** → Choose random values for all parameters
  $w_1=0.1, w_2=0.2, w_3=0.3, w_4=0.4, w_5=0.5,$
  $w_6=0.6, w_7=0.7, w_8=0.8, w_9=0.9, w_{10}=0.1$
  Bias of both layers $b_1=b_2=0.5$

- **Step2** → Forward propagation
  <u>Output layer</u>:
  o_1=0.8896 and o_2=0.8004

- **Step 3** → Cost (Sum of square error)
  $E = 0.593$

- **Step 4** → Backward propagation
  - Compute error derivatives with respect to all parameters (chain rule)
  - $\frac{dE}{dw_7} = 0.0765; \ldots \ldots \ldots \ldots, \frac{dE}{db_2} = 0.1975$
  - $\frac{dE}{dw_1} = 0.0020, \ldots \ldots, \frac{dE}{db_1} = 0.0008$

- **Step 5** → Update the parameters
  - Gradient descent (learning rate α=0.01)

  $w_1 \coloneqq w_1 - \alpha \frac{dE}{dw_1}$=0.1000

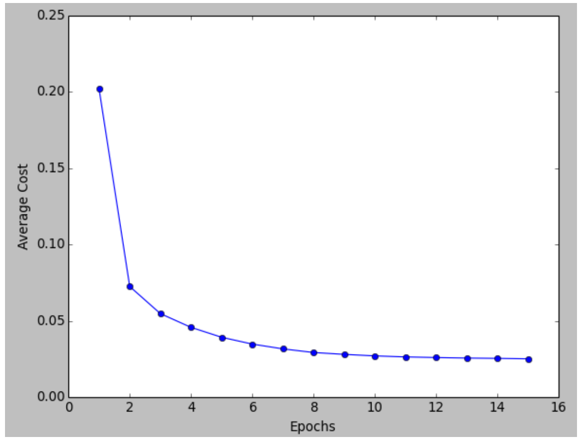
  ....

  $w_{10} \coloneqq w_{10} - \alpha \frac{dE}{dw_{10}}$=0.0988

- **Step 6** → Forward propagation
  o_1=0.8277 and o_2=0.7066

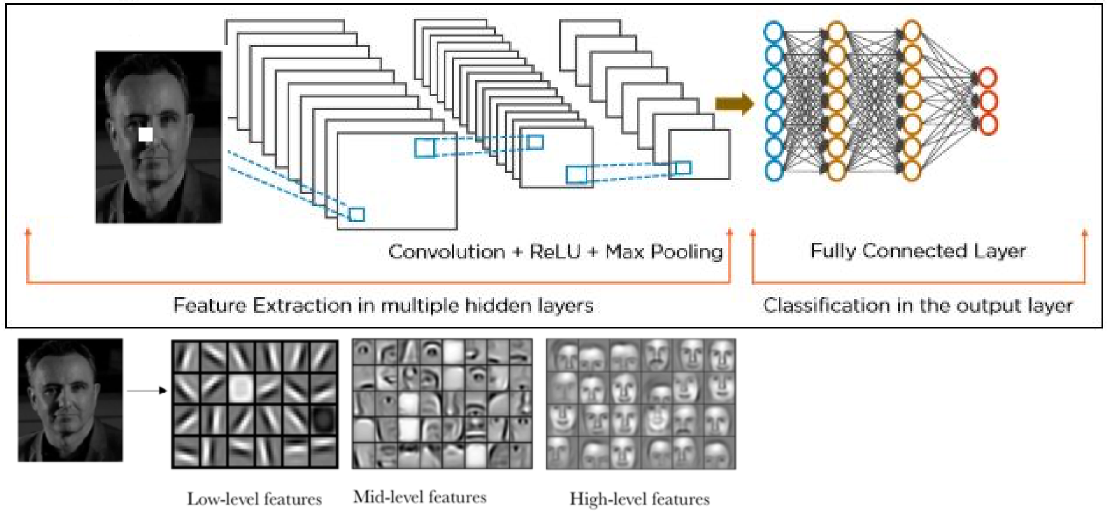
- **Step 7** → Cost(error)
  $E = 0.4803$

- Repeat step 4-7
  - Till get a reasonable cost



Convolution Neural Network (CNN)

- Two layers:
  - Convolution layer →recover features
  - MLP (fully connected) → classification

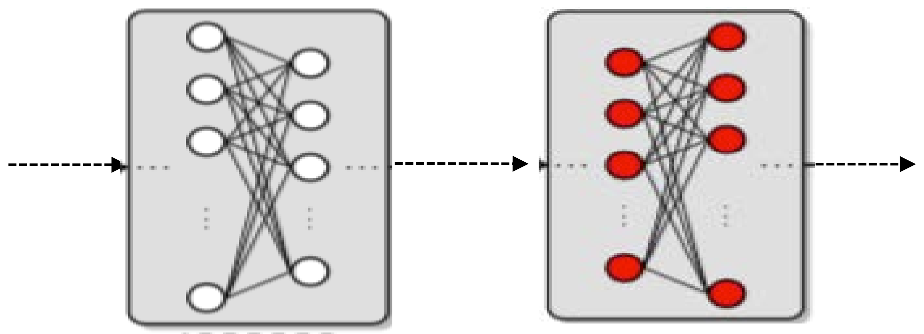


A convolutional Neural Network

Part II: Model Inversion for Impersonation

Model Inversion

- Given an ANN classifier C_w → How to generate $\bar{\boldsymbol{X}}$?
- Model Inversion of C_w
  - Optimization-based model inversion
    - Alexander Linden et. al. (1989)
  - Training-based model inversion
    - Ziqi Yang et. al. (2019)



Classifier C_w Inverse classifier I_w

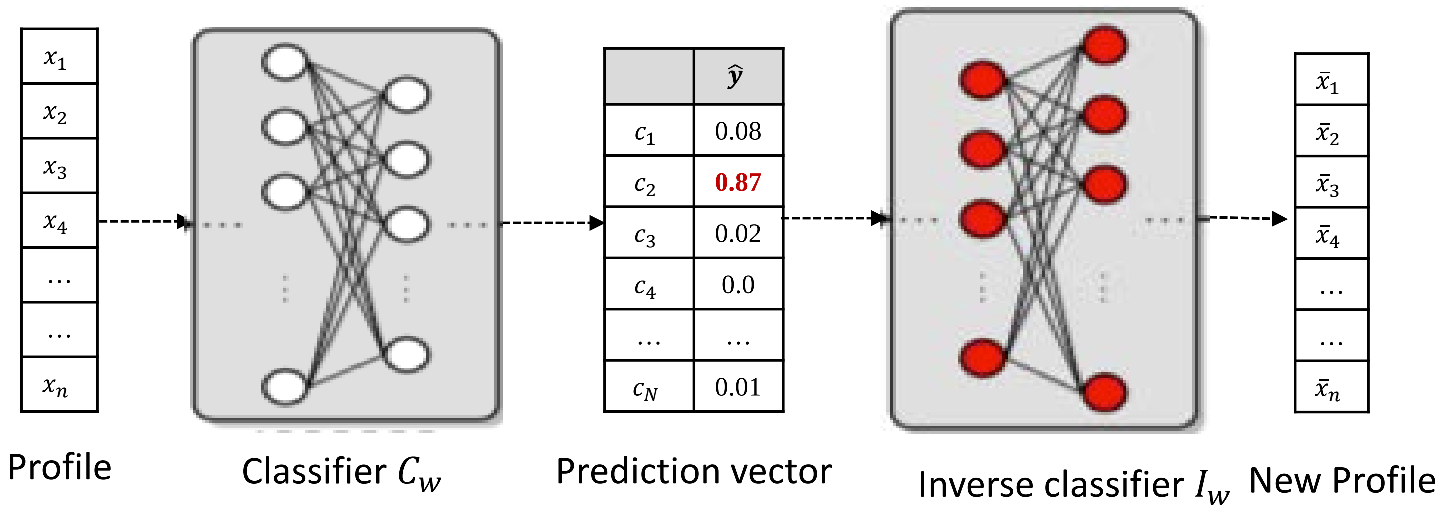
- **Optimization-based model inversion:**
  i. Find a profile $\bar{\boldsymbol{X}}$ of a target user i
  ii. Loss → between $C_w(\bar{\boldsymbol{X}})$ and $C_w(\boldsymbol{X})$
  iii. Update $\bar{\boldsymbol{X}}$ → gradients decent
  - Need white-box access of C_w

- **Training-based model inversion:**
  - Train an inverse classifier I_w
    - I_w will work opposite to C_w
  - Train I_w → black-box access of C_w

Training-based Model Inversion

- Input of C_w → An n dimensional vector
  - A BA profile $\boldsymbol{X}$ with m samples
    - Produces m input vectors
- Output of C_w → Prediction vector
  - An N dimensional vectors
  - N probability values for N classes
  - Valid class has
    - highest probability value
  - $\sum_{i=1}^{N} p_i = 1.0$
  - For a $\boldsymbol{X}$, C_w produces
    - m prediction vectors
      - Prediction table

- Input of I_w → Prediction vector
  - A prediction table produces
    - m inputs vectors
- Output of I_w →an n dimensional vector
  - A prediction table produces
    - an artificial profile $\bar{\boldsymbol{X}}$



Training-based Model Inversion (Ziqi Yang et. al.)

- Face Image classifier
- Assumption: Unlimited access of C_w
- Knowledge of an Adversary:
  - Type of classifier (face classifier)
  - Input and output format of C_w
  - Prediction vector for auxiliary image
    - Top $5/10$ probability values

- Model inversion → 3 steps process
  1. Construct an inverse classifier I_w
  2. Capture auxiliary images
     - Not used to train C_w
  3. Train the inverse classifier I_w

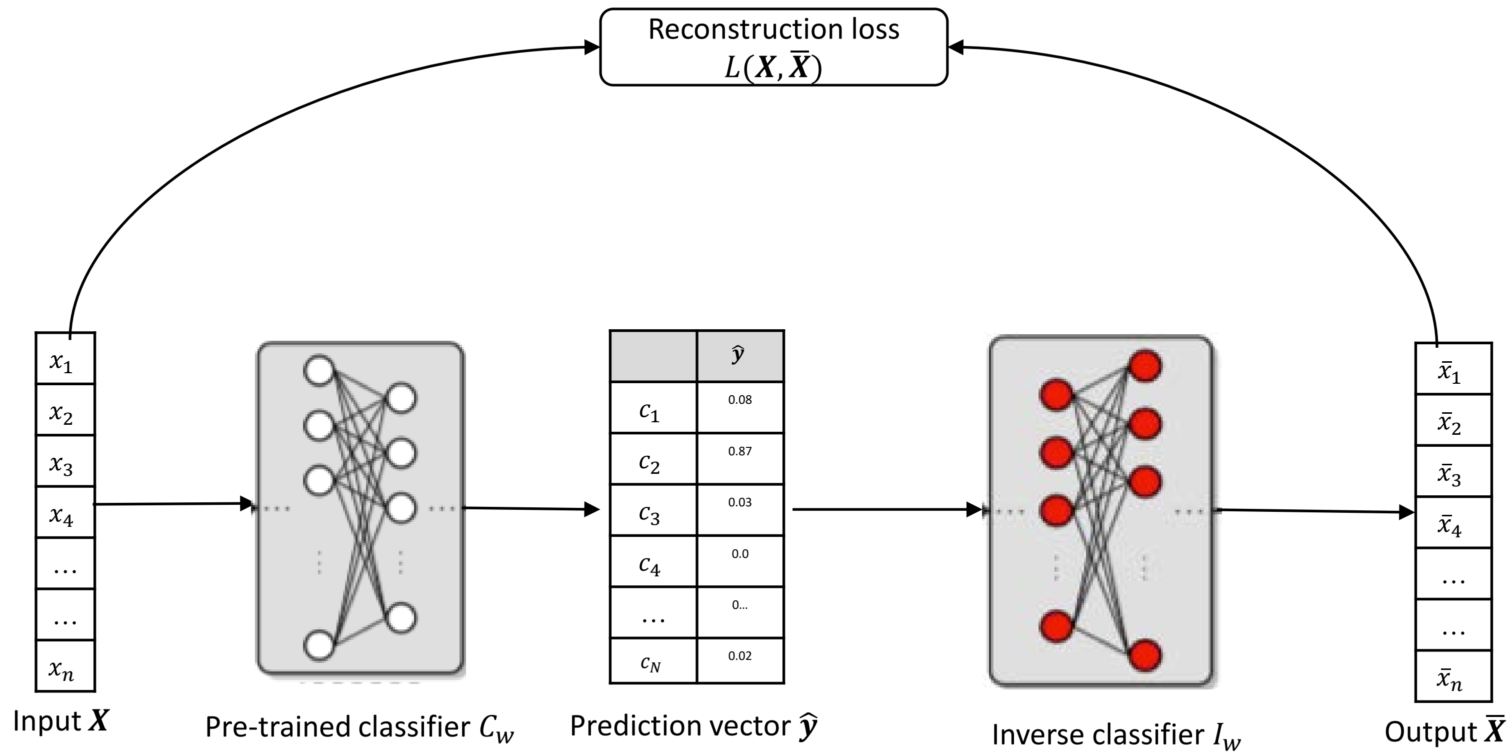
i. Construct an inverse classifier I_w:

- Convolution layer→inverse convolution layer
- MLP→inversion of MLP
- Tune layers, nodes number
  - For a good inverse classifier I_w

ii. Auxiliary Images:

- For facial recognition classifier
  - Collects public facial images of random individuals from Internet

Training-based Model Inversion (Ziqi Yang et. al.)



Our Approach

- Model inversion of C_w→ Inverse classifier I_w → An artificial BA profile $\overline{\boldsymbol{X}}$
- Is $\overline{\boldsymbol{X}} \cong \boldsymbol{X}$?

- Training of I_w depends on:
  - Auxiliary profiles
    - Close auxiliary profile→more correct information about soft boundary
  - Prediction vector:
    - Truncated prediction vector→information loss about soft boundary
    - Srivastava et. al. (2015):
      - Majority classes has small probability value →negligible information
      - Top 5 to 10 class carries all probability value → carries maximum information

Our Approach

- Unlimited black-box access to C_w
- Attacker's Knowledge:
  - Type of classifier (BA classifier)
  - Input and output format of C_w
  - Prediction vector
    - Partial information: highest non-zero probability value

Impersonation Attack: 4 steps process

I. Construct an I_w
  - C_w is a data classifier
    - I_w has almost same but inverse structure

ii. Collects auxiliary profiles
  - BA app →publicly available
    - Ask people for the BA profiles

iii. Tanning the inverse classifier I_w
  - Input of I_w→Need sufficient information
    - Updated the prediction vector

iv. Generate data samples in inference phase
  - I_w needs m probability vectors as input

Update the Prediction Vector

- k-NN classifier → a new prediction vector

- In both prediction vectors
  - The corresponding probability value may not be same
  - Order of all classes based on their probability values
    - Will be same with high probability

- In k-NN classifier
  - Attacker need data samples of N classes
- Use data samples of auxiliary profile to
    - Choose $\overline{m}$ representative for each class
      - Based on non-zero probability value in prediction vectors

Update the Prediction Vector

- Given
  - A set of auxiliary' profiles
  - Prediction tables of auxiliary profiles
  - N classes and their representatives → from auxiliary profile

	c_1	c_2	....	c_N
$\hat{y}_1$	0.0	0.81	....	0.0
$\hat{y}_2$	0.0	0.87	....	0.0
…	…	…	....	…
$\hat{y}_m$	0.0	0.0	....	0.54

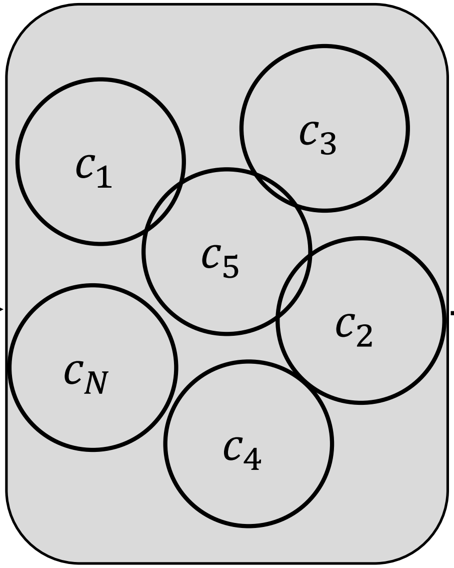
Prediction table from C_w

	F_1	F_2	F_3	....	F_n
x_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	....	$x_{1,n}$
x_2	$x_{2,1}$	$x_{2,2}$	$x_{2,2}$	....	$x_{2,n}$
…	…	…	…	....	…
x_m	$x_{m,1}$	$x_{m,2}$	$x_{m,2}$	....	$x_{m,n}$

Profile $\boldsymbol{X}$

x_i
$x_{i,1}$
$x_{i,2}$
$x_{i,3}$
$x_{i,4}$
$x_{i,5}$
…
…
$x_{i,n}$

Input $\boldsymbol{x_i \epsilon X}$



K-NN Classifier

	$\hat{y}_i$
c_1	0.09
c_2	0.63
c_3	0.03
c_4	0.13
..	…
c_N	0.12

Prediction vector $\hat{y}_i$

	c_1	c_2	....	c_N
$\hat{y}_1$	0.0	0.81	....	0.0
$\hat{y}_2$	0.0	0.87	....	0.0
…	…	…	....	…
…	…	…	....	…
$\hat{y}_m$	0.0	0.0	....	0.54

Prediction table from C_w

	c_1	c_2	....	c_N
$\hat{y}_1$	0.15	0.51	....	0.12
$\hat{y}_2$	0.09	0.63	....	0.12
…	…	…	....	…
$\hat{y}_m$	0.14	0.23	....	0.34

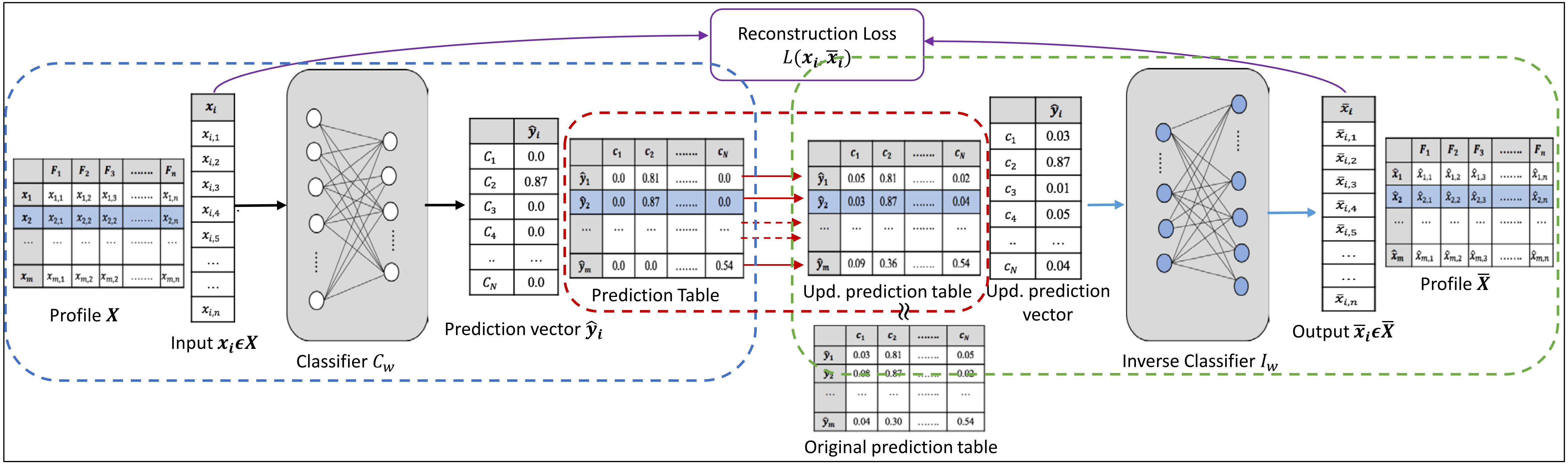
Prediction table from k-NN

	c_1	c_2	....	c_N
$\hat{y}_1$	0.05	0.81	....	0.02
$\hat{y}_2$	0.03	0.87	....	0.04
…	…	…	....	…
$\hat{y}_m$	0.09	0.36	....	0.54

Updated prediction table

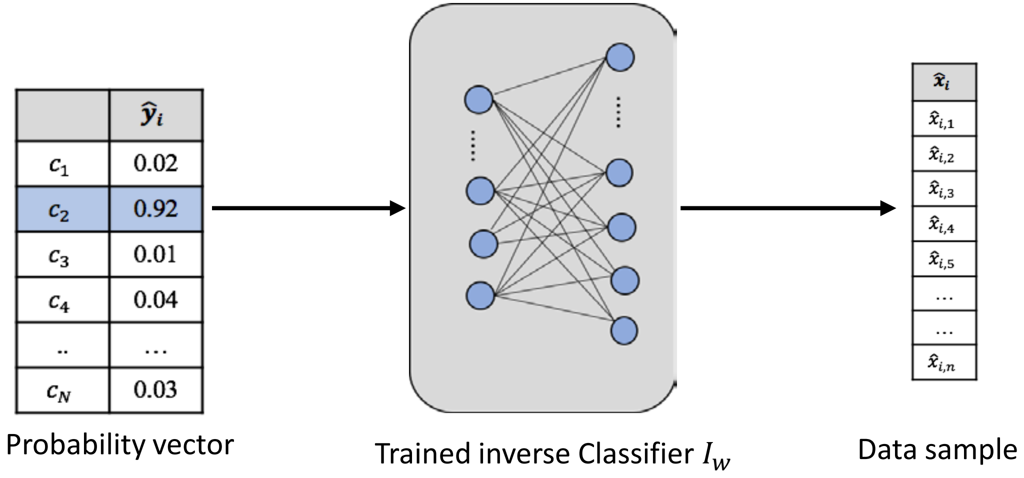
Train an Inverse Classifier

- Given that
  - A pre-trained classifier C_w
  - A set of auxiliary users' profiles
  - An inverse classifier (not trained yet)



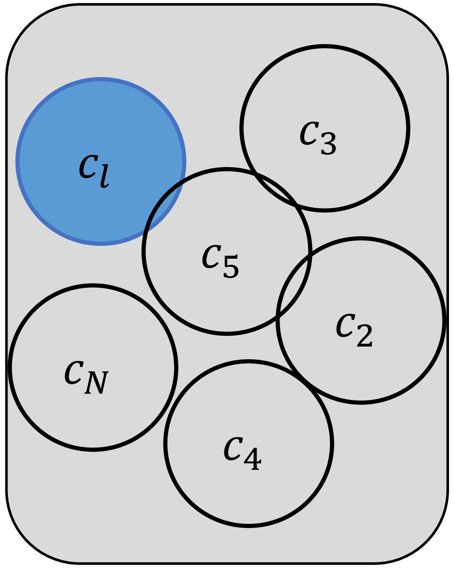
Generate Data Sample for Impersonation

- Trained I_w → generate data samples for $\overline{\boldsymbol{X}}$
  - I_w requires m probability vectors as input



Probability vectors generation:

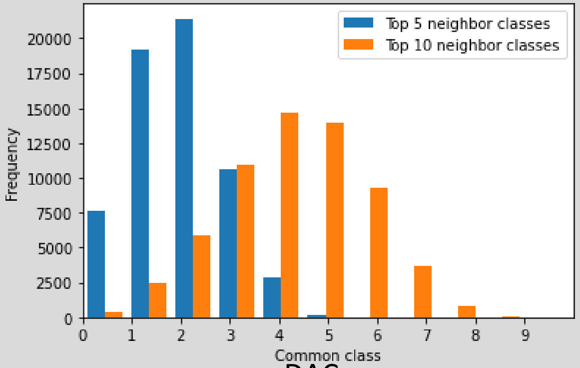
- Step1: For a class c_l
  - Calculate its avg. distance from all other classes
- Step2: Assign a highest probability value to c_l
- Step3: Distributed other probability values
  - To top 5 to 10 neighbor classes of c_l



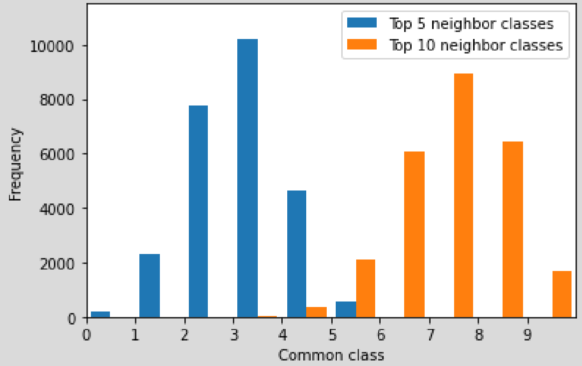
Experimental Results

- Two BA systems
  - DAC (Draw a Circle) (Morshedul Islam et. al. 2016)
  - Touchalytics (Mario Frank et. al. 2013)

- **Common class in top 5/10 classes of the prediction vectors of both k-NN and ANNs classifiers**

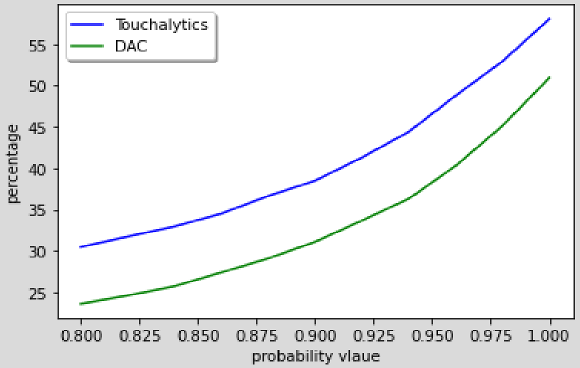


DAC

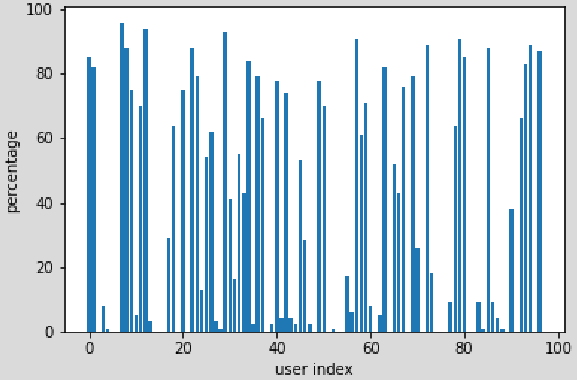


Touchalytics

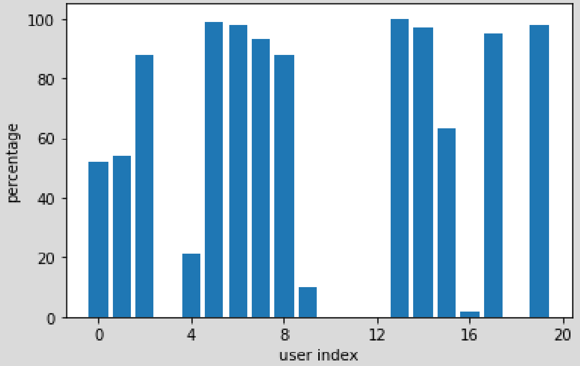
- **Data sample acceptance rate versus probability value of a target user's class**



- **Data sample acceptance rate for different profiles of both BA systems**



DAC



Touchalytics

References

- Yang, Ziqi, et al. "Neural network inversion in adversarial setting via background knowledge alignment." *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019.
- Mandelbaum, Amit, and Daphna Weinshall. "Distance-based confidence score for neural network classifiers." *arXiv preprint arXiv:1709.09844* (2017).
- Srivastava, Rupesh K., Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks." *Advances in neural information processing systems*. 2015.
- Linden, Alexander, and J. Kindermann. "Inversion of multilayer nets." *Proc. Int. Joint Conf. Neural Networks*. Vol. 2. 1989.
- Cover, Thomas, and Peter Hart. "Nearest neighbor pattern classification." *IEEE transactions on information theory* 13.1 (1967): 21-27
- Islam, Md Morshedul, and Reihaneh Safavi-Naini. "POSTER: a behavioural authentication system for mobile users." *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016.
- Frank, Mario, et al. "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication." *IEEE transactions on information forensics and security* 8.1 (2012): 136-148.

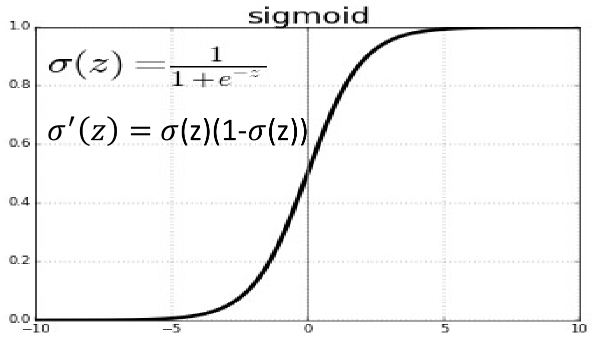
Thank You

Appendix

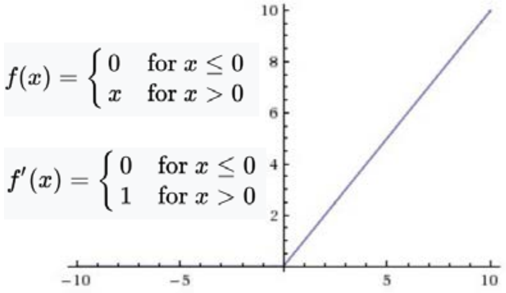
Activation Functions

Activation function → switch ON/OFF neuron

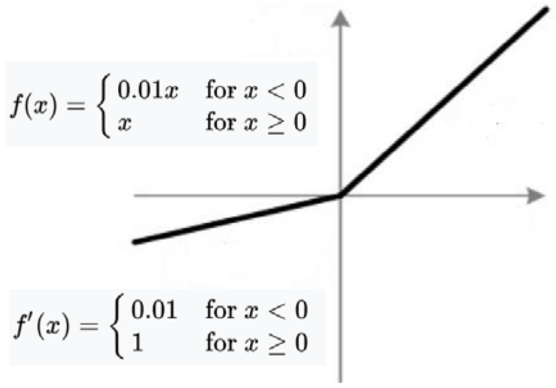
- Sigmoid activation function



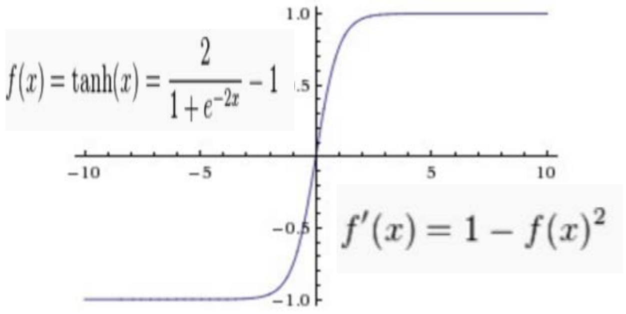
- Rectifier Linear Unit (ReLU)



- Leaky ReLU



- tanh



MLP Example

For an MLP, given that

- Features value x_1=1, x_2=4, x_3=5
- Output label y_1=0.1, y_2=0.05

<u>Input layer to hidden layer:</u>

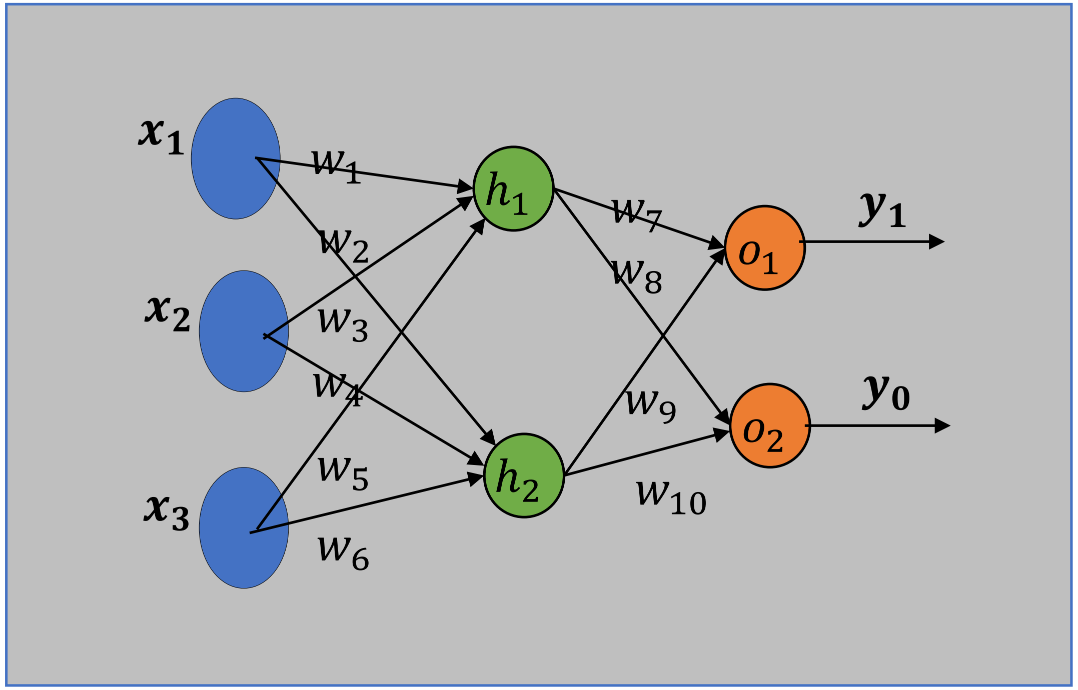
$w_1x_1 + w_3x_2 + w_5x_3 + b_1 = z_{h_1}$ and $h_1 = \sigma(z_{h_1})$

$w_2x_1 + w_4x_2 + w_6x_3 + b_1 = z_{h_2}$ and $h_2 = \sigma(z_{h_2})$

<u>Hidden layer to output layer:</u>

$w_7h_1 + w_9h_2 + b_2 = z_{o_1}$ and $o_1 = \sigma(z_{o_1})$

$w_8h_1 + w_{10}h_2 + b_2 = z_{o_2}$ and $o_2 = \sigma(z_{o_2})$



<u>Tanning Phase:</u>

Step1: Choose random value for each parameter

w_1=0.1, w_2=0.2, w_3=0.3, w_4=0.4, w_5=0.5,

w_6=0.6, w_7=0.7, w_8=0.8, w_9=0.9, w_{10}=0.1

Bias of both layers b_1=b_2=0.5

Step2: Forward propagation (predict the output)

<u>Hidden layer:</u>

z_{h_1}=0.1(1)+0.3(4)+0.5(5)=4.3 and $h_1 = \sigma(4.3)$=0.986

z_{h_2} = 5.3 and $h_2 = \sigma(5.3)$=0.9950

<u>Output layer</u>:

z_{o_1} = 2.0862 and $o_1 = \sigma(2.0862)$=0.8896

z_{o_2} = 1.3888 and $o_2 = \sigma(1.3888)$=0.8004

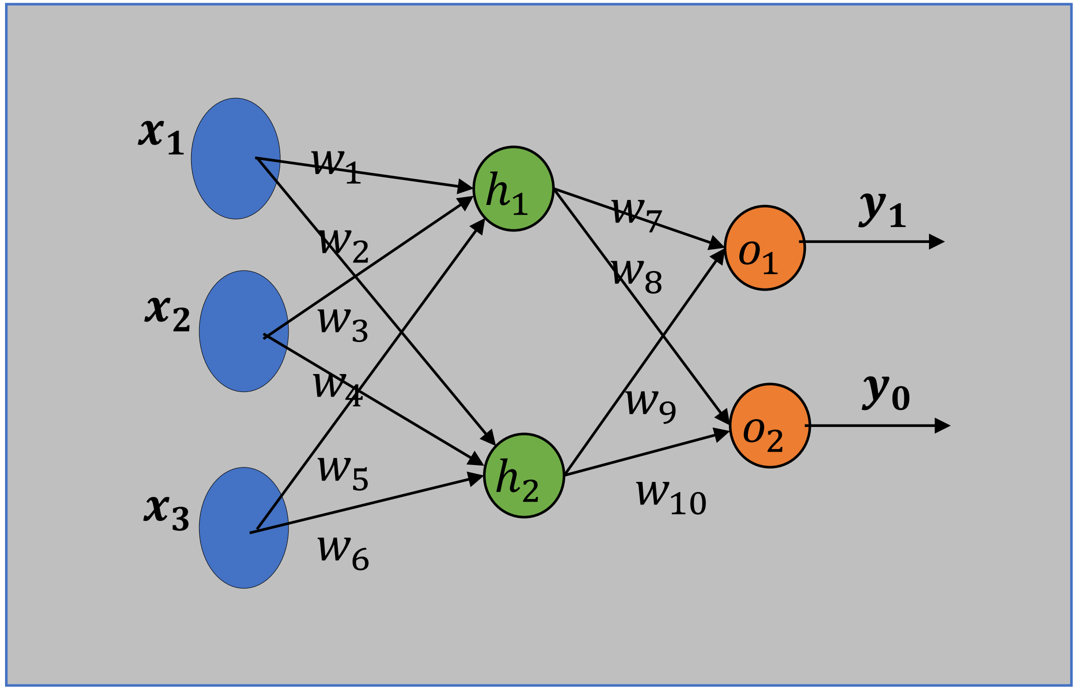
MLP Example

Step 3: Cost calculation (Sum of square error)

$$E = \frac{1}{2}[(o_1 - y_1)^2 + (o_2 - y_2)^2] = 0.593$$

Step 4: Backward propagation

Compute error derivatives with respect to all parameters



Derivative on output layer:

$$\frac{dE}{do_1} = o_1 - y_1 \text{ and } \frac{dE}{do_2} = o_2 - y_2$$

Derivative on hidden layer:

$$\frac{dz_{o_1}}{dw_7} = h_1; \frac{dz_{o_2}}{dw_8} = h_1; \frac{dz_{o_1}}{dw_9} = h_2; \frac{dz_{o_2}}{dw_{10}} = h_2; \frac{dz_{o_1}}{db_2} = 1; \text{ and } \frac{dz_{o_2}}{db_2} = 1$$

Use chain rule

$$\frac{dE}{dw_7} = \frac{dE}{do_1}.\frac{do_1}{dz_{o_1}}.\frac{dz_{o_1}}{dw_7} = (o_1 - y_1)(o_1(1 - y_1))\, h_1 = 0.0765$$

$$\frac{dE}{dw_8} = 0.1183; \frac{dE}{dw_9} = 0.0772; \frac{dE}{dw_{10}} = 0.1193$$

$$\frac{dE}{db_2} = \frac{dE}{do_1}.\frac{do_1}{dz_{o_1}}.\frac{dz_{o_1}}{db_2} + \frac{dE}{do_2}.\frac{do_2}{dz_{o_2}}.\frac{dz_{o_2}}{db_2} = 0.1975$$

Derivative on input layer:

$$\frac{dE}{dw_1} = 0.0020; \frac{dE}{dw_2} = 0.0004; \frac{dE}{dw_3} = 0.0079;$$

$$\frac{dE}{dw_4} = 0.0016; \frac{dE}{dw_5} = 0.0099; \frac{dE}{dw_6} = 0.0020;$$

$$\text{and } \frac{dE}{db_1} = 0.0008$$

MLP Example

Step 5: Update weight (gradient descent)

Given the learning rate α=0.01

$w_1 := w_1 - \alpha \frac{dE}{dw_1}$=0.1-0.01(0.0020)=0.1000

$w_2 := w_2 - \alpha \frac{dE}{dw_2}$=0.2000

$w_3 := w_3 - \alpha \frac{dE}{dw_3}$=0.2999

$w_4 := w_4 - \alpha \frac{dE}{dw_4}$=0.4000

$w_5 := w_5 - \alpha \frac{dE}{dw_5}$=0.4999

$w_6 := w_6 - \alpha \frac{dE}{dw_6}$=0.6000

$w_7 := w_7 - \alpha \frac{dE}{dw_7}$=0.6992

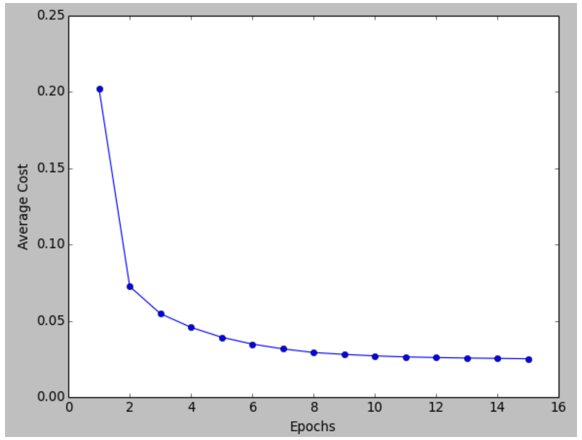
$w_8 := w_8 - \alpha \frac{dE}{dw_8}$=0.7988

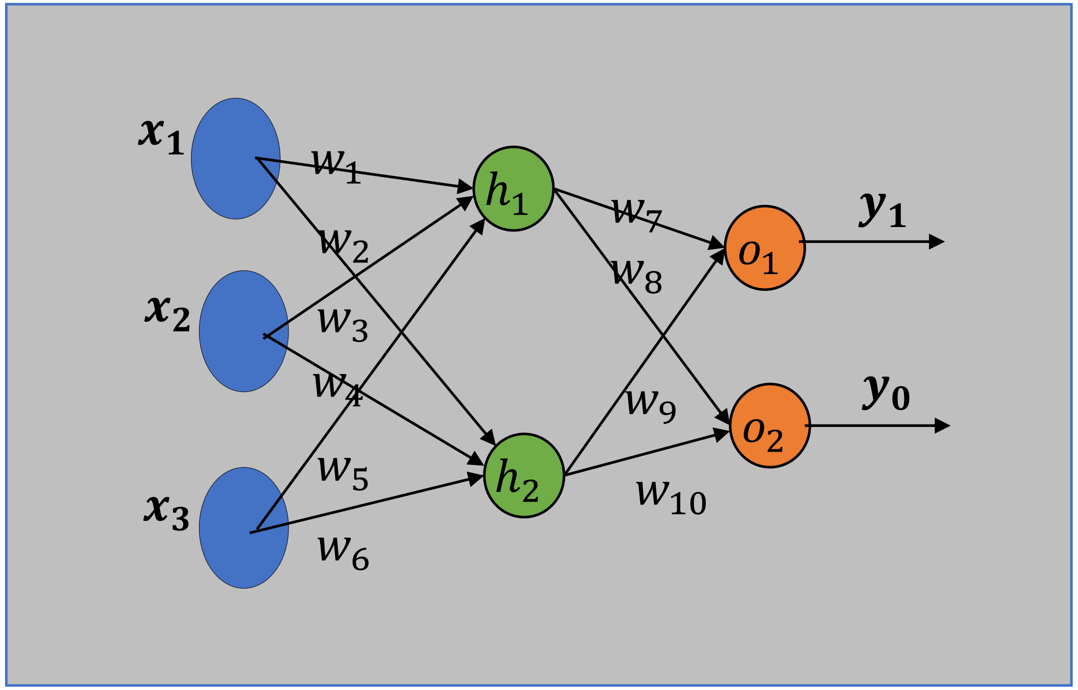
$w_9 := w_9 - \alpha \frac{dE}{dw_9}$=0.8992

$w_{10} := w_{10} - \alpha \frac{dE}{dw_{10}}$=0.0988

$b_1 := b_1 - \alpha \frac{dE}{db_1}$=0.5000

$b_2 := b_2 - \alpha \frac{dE}{db_2}$=0.4980





Step 6: Forward propagate

z_{h_1}=3.799 and $h_1 = \sigma(3.799)$=0.9781

z_{h_2} = 4.8 and $h_2 = \sigma(4.8)$=0.9918

z_{o_1} = 1.57 and $o_1 = \sigma(2.0862)$=0.8277

z_{o_2} = 0.8792 and $o_2 = \sigma(1.3888)$=0.7066

Step 3: Cost calculation (Sum of square error)

$$E = \frac{1}{2}[(o_1 - y_1)^2 + (o_2 - y_2)^2] = 0.4803$$

Convolution Neural Network (CNN)

- Two layers:
  - Convolution layer →recover features
  - MLP (fully connected) → classification

<u>Convolutional layer operations:</u>

- Step 1: Convolution operations
  - Filter → feature detectors

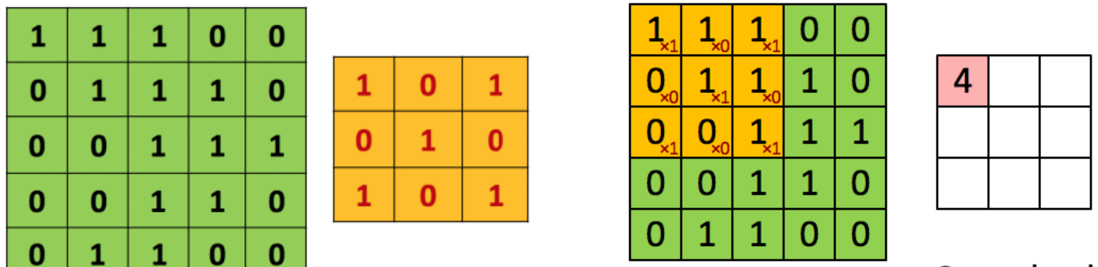
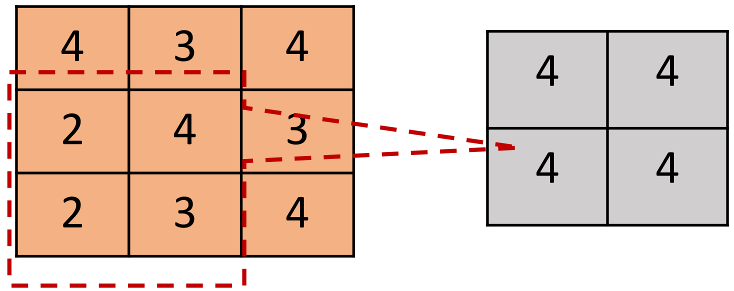
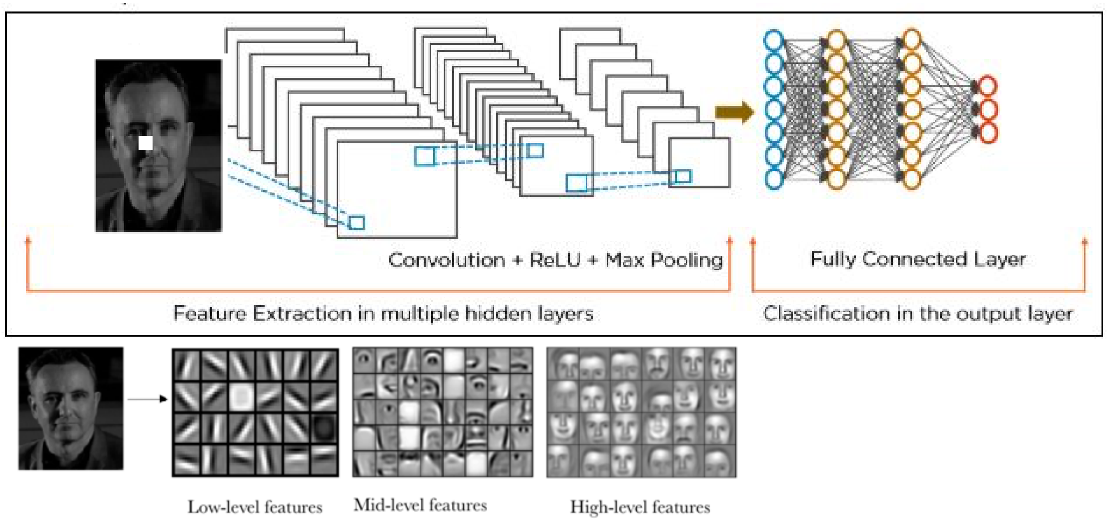


Image pixel value | Filter/Kernel | Image | Convolved Feature

- Step 2: Apply ReLU
  - Replaces negative pixel by zero

- Step3: Pooling step →reduces dimensionality
  - Retains most important information





A convolutional Neural Network