THE UNIVERSITY OF MELBOURNE

Autoencoders and their Applications

COMP90073
Security Analytics

Sarah Erfani, CIS

Semester 2, 2021

Outline

- Introduction to Neural Networks
- Gradient Decent Learning
- Autoencoders and their architectures
- Denoising Autoencoder (DAE)
- Variational Autoencoder (VAE)

THE UNIVERSITY OF MELBOURNE

Artificial Neural Networks

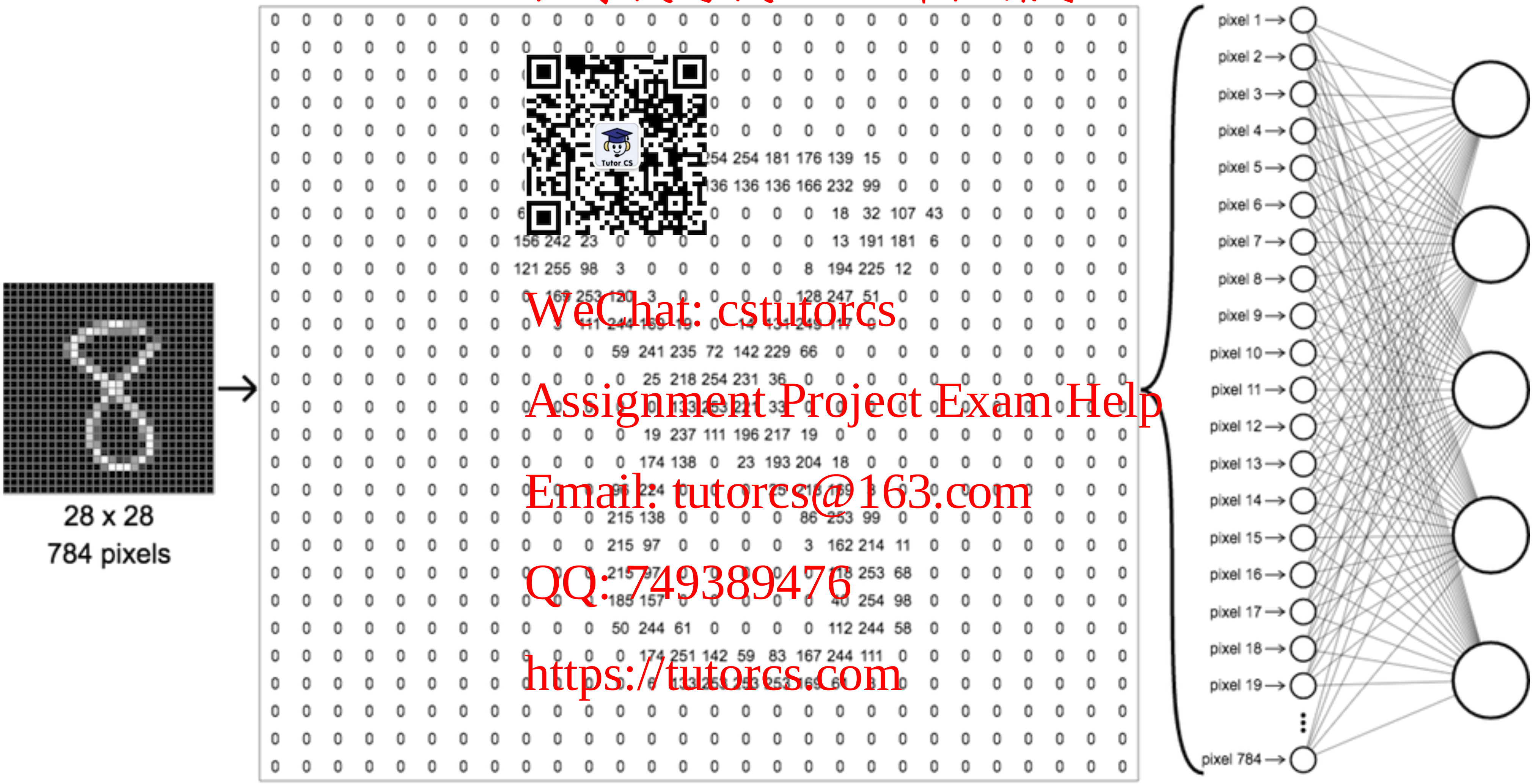
- A collection of simple, trainable mathematical units that collectively learn complex functions

- Given sufficient training data an artificial neural network can approximate very complex functions mapping raw data to output decisions

Artificial Neural Networks

THE UNIVERSITY OF MELBOURNE

Artificial Neural Networks



Artificial Neural Networks

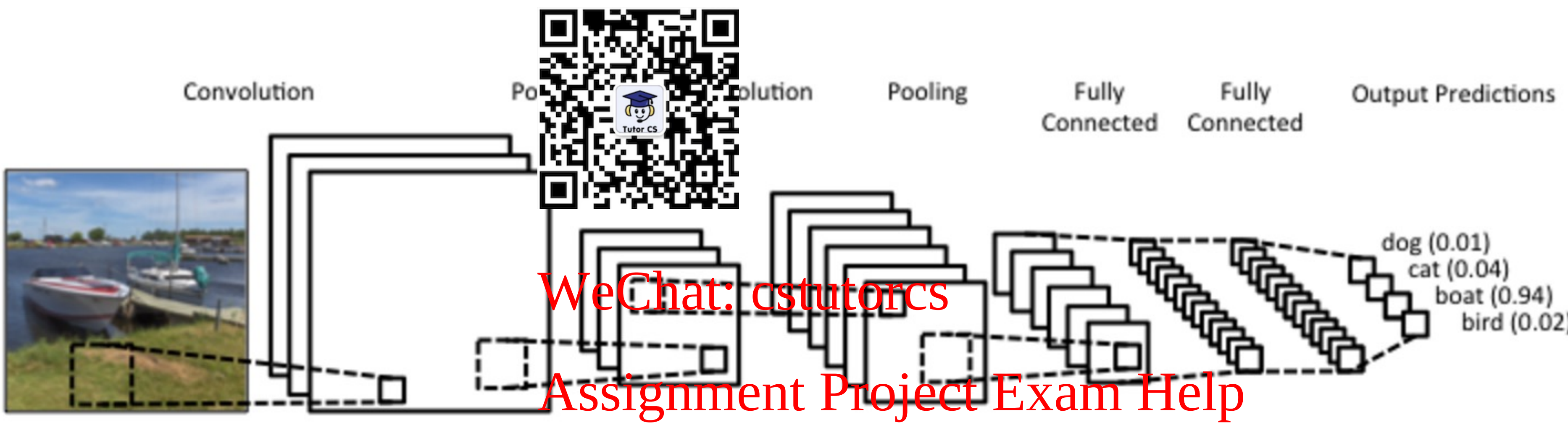
Types of Deep Neural Networks (DNNs)

1. DNN – all fully connected layers

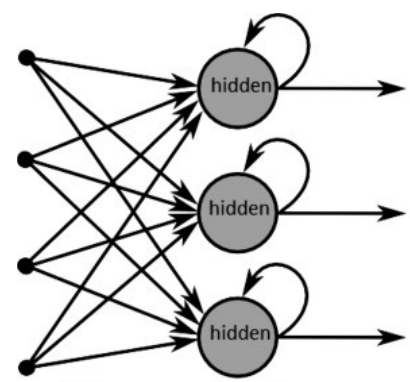
THE UNIVERSITY OF MELBOURNE

Types of Deep Neural Networks (DNNs)

2. CNNs (Convolutional Neural Networks) – some convolutional layers

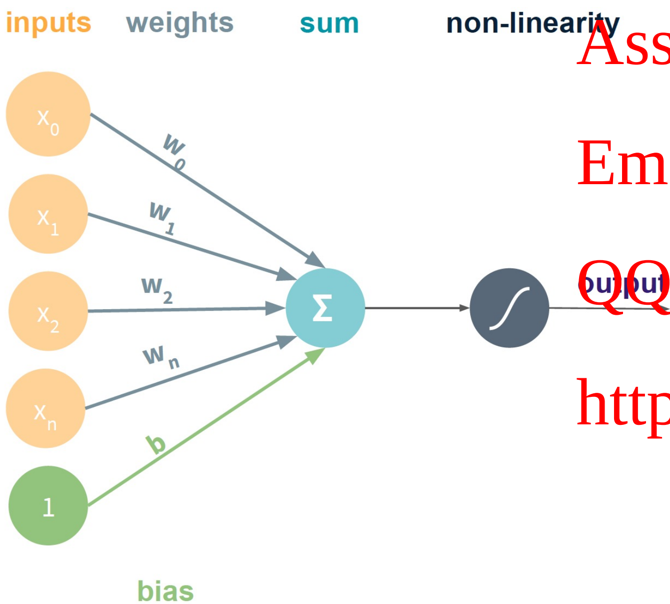


3. RNNs (Recurrent Neural Networks) – LSTM



Fundamentals of Neural Networks

- Receive signals from input neurons: $x_1, x_2, \dots, x_n$
- Weight signals according strength between neurons: $w_1x_1, w_2x_2, \dots, w_nx_n$
- Add the input signals and $_1, w_2x_2, \dots, w_nx_n + b = \sum_{i=1}^{n} w_i x_i + b$
- Emit an output signal: activation function f



Activation Function

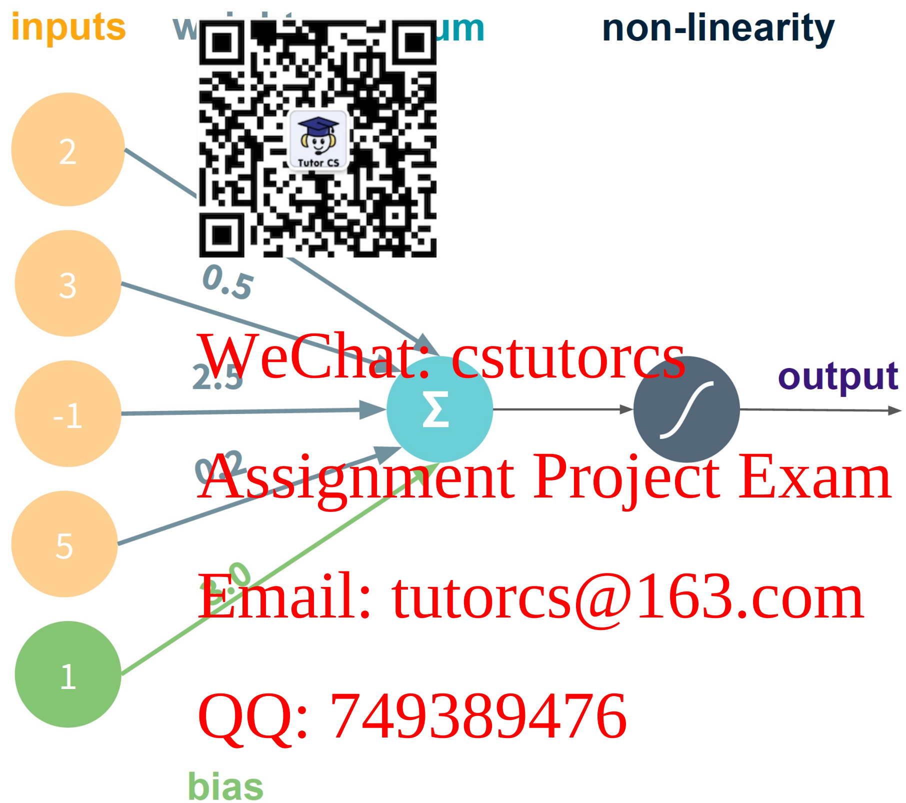
$$output = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Activation Functions

- Activation functions add non-linearity to our network's function
- Most real-world problems are non-linear

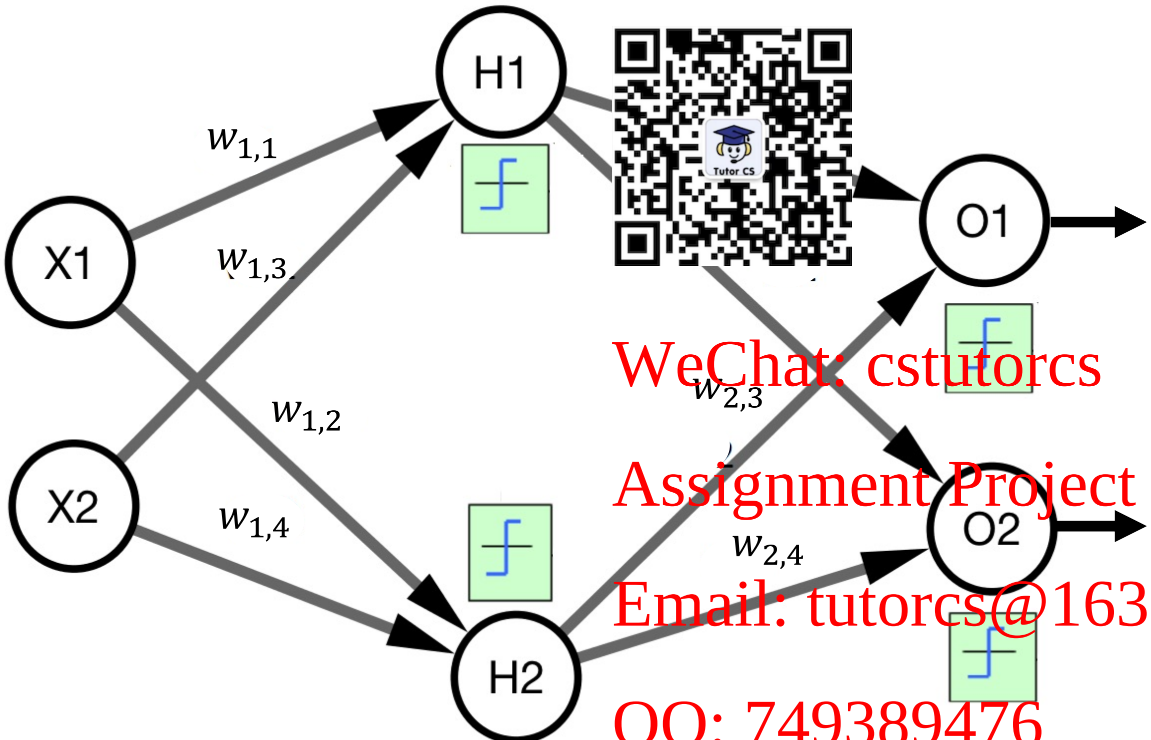
Common Activation Functions:

Fundamentals of Neural Networks



$$output = sigmoid(2\times0.1 + 3\times0.5\ + (-1)\times2.5 + 5\times0.2 + 1\times3) = 0.96$$

Functional Form of Neural Networks



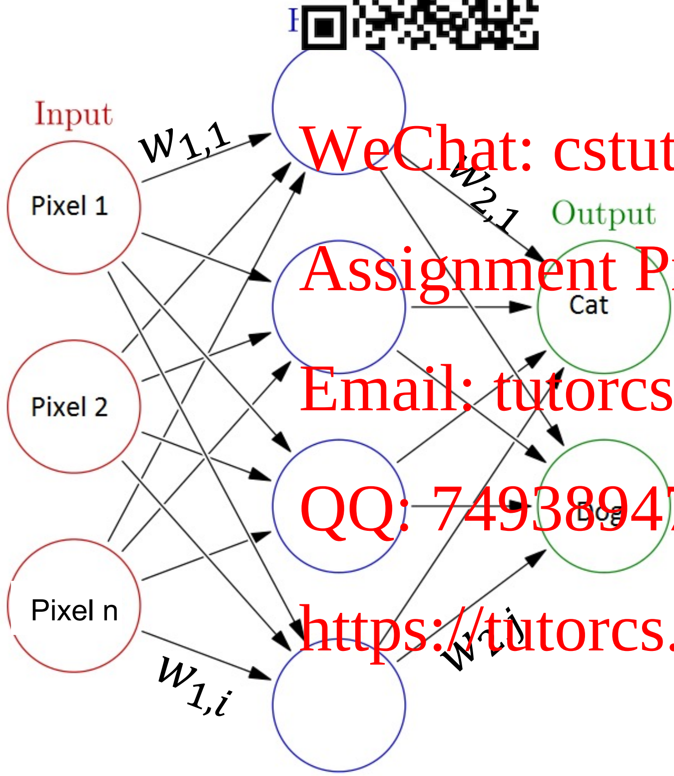
Feed-Forward pass:

$$H_1 = f_1(X_1 w_{1,1} + X_2 w_{1,3})$$
$$H_2 = f_1(X_1 w_{1,2} + X_2 w_{1,4})$$
$$O_1 = f_2(H_1 w_{2,1} + H_2 w_{2,3})$$
$$O_2 = f_2(H_1 w_{2,2} + H_2 w_{2,4})$$

Training a Neural Network

- Find a set of weights so that the network exhibits the desired behaviour
- Example: cat vs. dog



Output	Label	
0.8,0.7	1,1	(Cat & dog)
0.08,0.6	0,1	(Dog)
0.92,0.01	1,0	(Cat)
0.02, 0.0	0,0	(No cat or dog)

Error Function

- Measure the difference between actual output and expected output
- One popular measure: sum-squared error

$$E(input, weights, label) = \sum (output - label)^2$$

- Note: Neural network is a composite/nested function that map the input to the output.

$$output = f_{NN}(input, weights)$$

Gradient Decent Learning

- A training example is a vector of inputs and the desirable output(s), i.e., $(\{x_1, x_2, \dots, x_n\}, \{t_1, t_2, \dots, t_m\})$, $t_i = 1$ iff the data point $\{x_1, x_2, \dots, x_n\}$, belongs to the ith class.
- **Objective:** finding the weights that minimise the difference between t and o (predicted output) for each of our training inputs.
- Define an error function E to be the sum of squared errors.

$$E = \frac{1}{2} \sum_{k=1} (o_k - t_k)^2$$

- If we think of E as height, it defines an error landscape on the weight space. *The aim is to find a set of weights for which E is very low.*

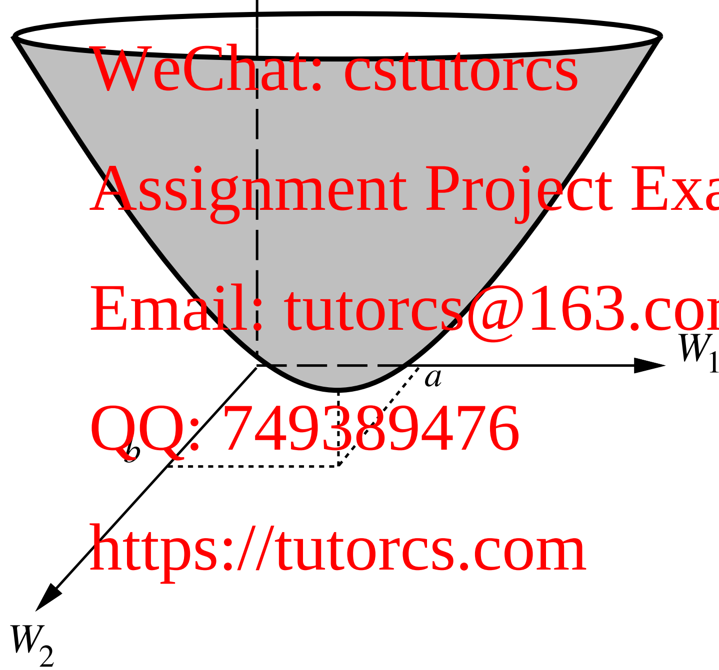
- This is done by moving in the steepest downhill direction (Gradient descent), i.e., $-\frac{\partial E}{\partial w_i}$

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

Weight update rule
η : Learning rate

Example Error Landscape

- An example error landscape for gradient descent search in weight space



Intuition of Gradient Descent Learning

$$w_1 = w_0 - \eta \frac{\partial E}{\partial w}(w_0)$$

$$w_2 = w_1 - \eta \frac{\partial E}{\partial w}(w_1)$$

$$\vdots$$

$$w_{\text{optimal}}$$

Negative gradient points towards a local minimum.

Weight update rule

To derive the weight update rule, we need to remember the chain rule from differential calculus:

if $y = y(u)$ and $u = u$

then $\partial y/\partial x = (\partial y/\partial u)($

So, if $\mathrm{E} = \frac{1}{2}\sum(o-t)^2$, $o = \sum w_i x_i + b$, and $\mathrm{t} = $ true output

$$\frac{\partial E}{\partial w_i} = \frac{\partial\left[\frac{1}{2}(o-t)^2\right]}{\partial o}\frac{\partial o}{\partial w_i}$$

$$= (o-t)\ \frac{\partial o}{\partial w_i}$$

$$= (o-t)x$$

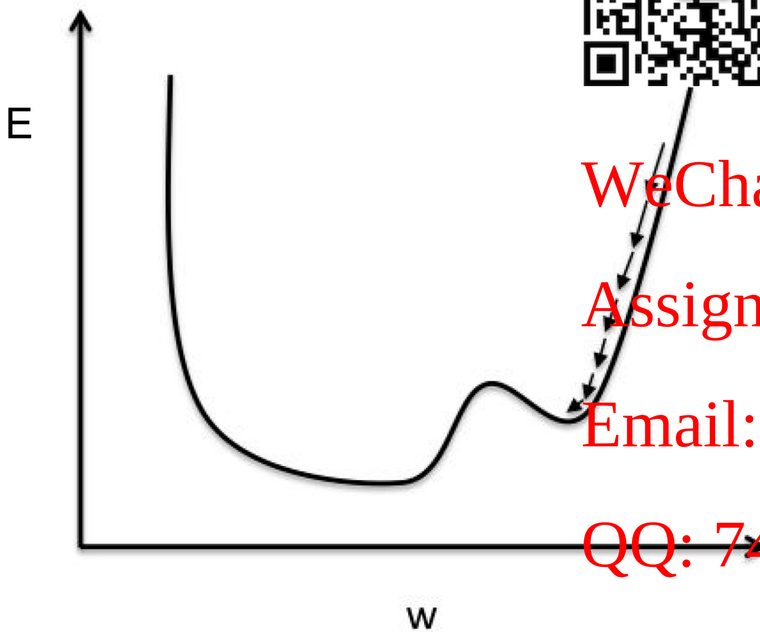
Normally we include a learning rate parameter η to control the update of weights in a stable manner

$$w_i \leftarrow w_i - \eta(o-t)x$$

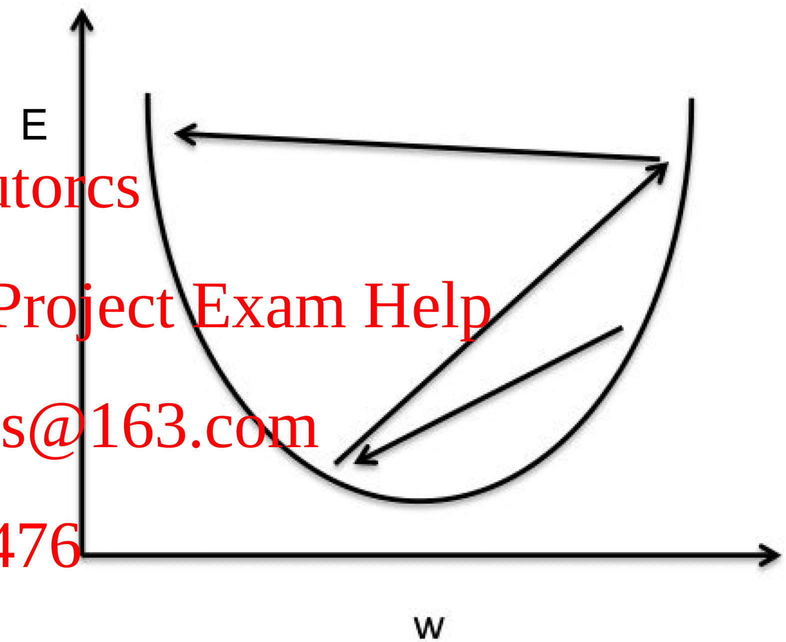
We repeatedly update the weights based on each example until the weights converge

Adjusting Learning Rate

- Learning rate parameter η is a small value (usually $0.0001 - 0.1$) to control the update of weights in a sta[illegible]er.



Small learning rate: Many iterations until convergence and trapping in local minima.

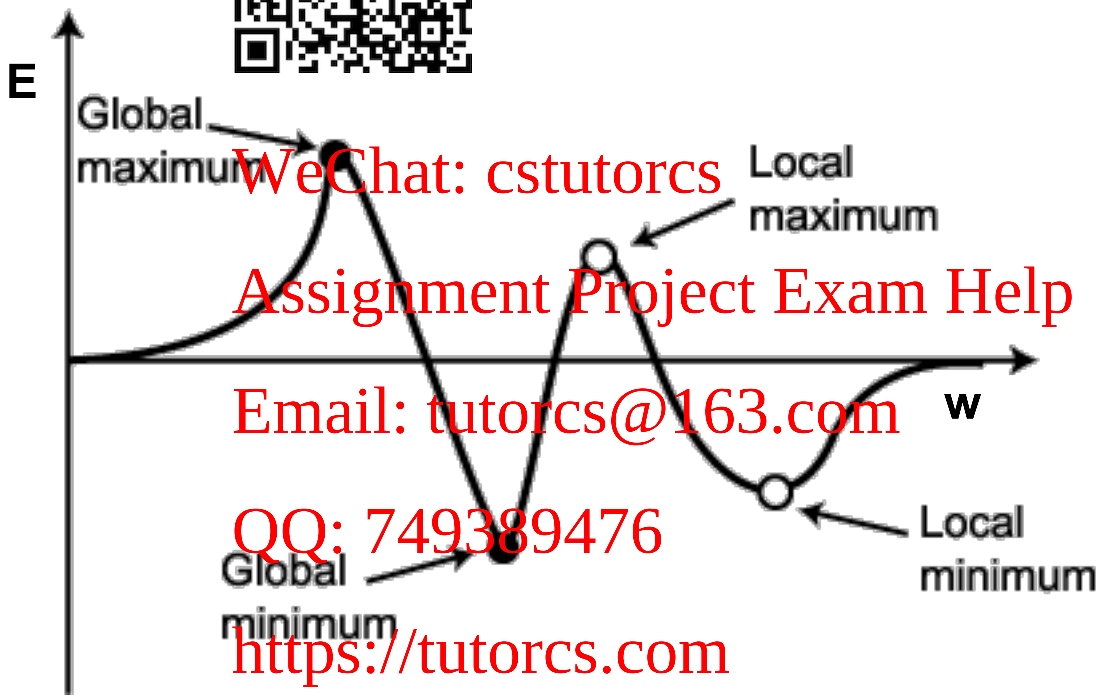


Large learning rate: Overshooting.

THE UNIVERSITY OF MELBOURNE

Minimizing Error by Gradient Descent

- The iterative algorithm might converge to one of the many local minima



Backpropagation Algorithm

Backpropagation(network, training data D, label T)

Initialise the weights ***w*** to small random values

Repeat

For each $d \in D, d =< x_1, \ldots, x_n, t_c >$

Forward pass: calculate hidden h_i and output O_j values

Backward pass:

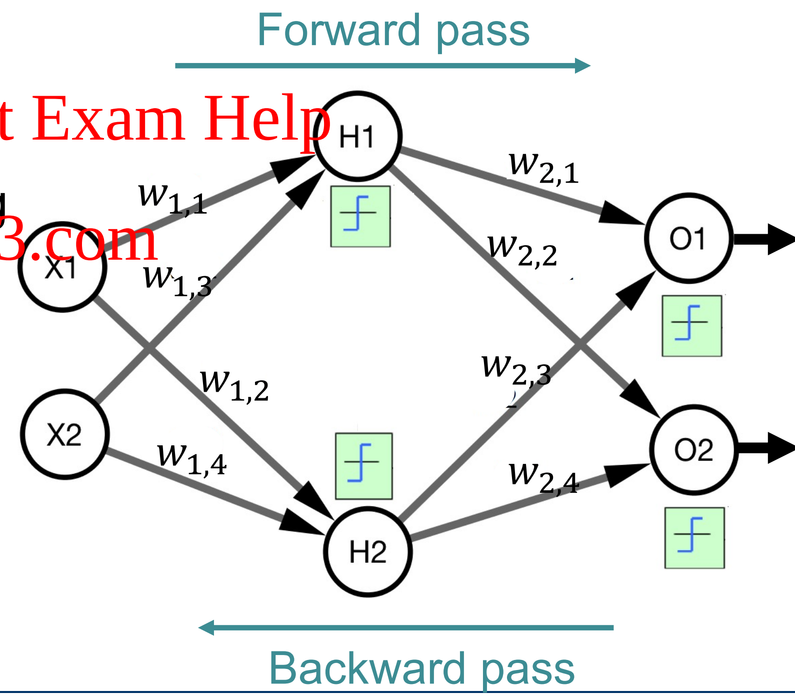
Calculate error between o_k and t_k at output

Update the weights in each layer $w_{i,j}$

(in proportion to their effect on the error using

gradient descent $w_i \leftarrow w_i - \eta(o - t)x$)

Until network has converged



Network Architectures/Parameters

- Number of input nodes = number of features
- Number of output nodes = [illegible] of classes
  - *Neural networks can [illegible] rally with multi-class classification problems (compare [illegible])*
- Number of hidden layers
- Number of nodes in each hidden layers
- Learning rate
- Regularization parameters (similar to C in SVM): control the complexity of the model, preventing overfitting.

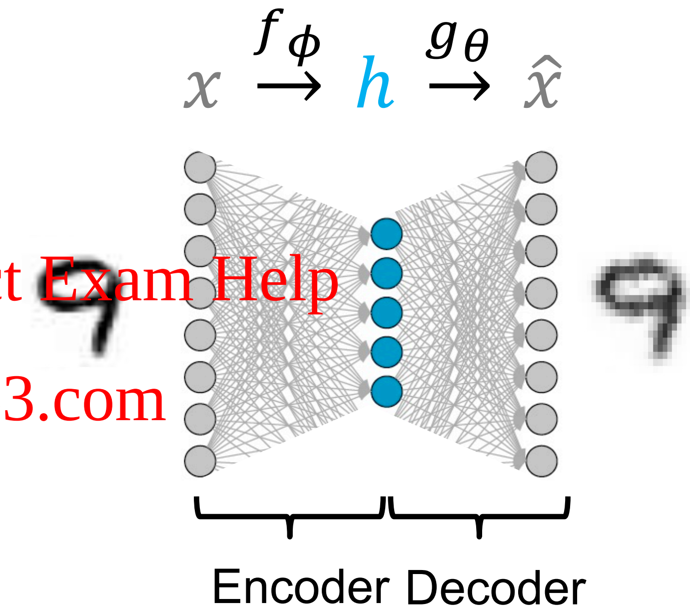
Live Demo:
http://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.13820&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTim

Autoencoder [1]

- A a neural net which aim is to take an input $x \in \mathbb{R}^d$ and reproduce it $\hat{x} \in \mathbb{R}^d$.
- To make this non-trivial, we [illegible]d a *bottleneck* layer $h \in \mathbb{R}^k$ whose dimension is much smaller t[illegible]ut, $k \ll d$.
- Architecture:
  - Encoder: $h = f_\phi(x) = sigmoid(Wx + b)$
  - Decoder: $\hat{x} = g_\theta(h) = sigmoid(\widehat{W}h + \hat{b})$
  - Often use tied weights, $\widehat{W} = W^T$
- Minimize a loss function

$$\mathcal{L}(\theta, \phi) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i - \overbrace{g_\theta\left(f_\phi(x_i)\right)}^{\hat{x}_i}\right)^2$$

- Parameters are obtained using backpropagation
- Demo: https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html



Using AEs Unsupervised Problems

Only input vectors $\{x_1, x_2, \dots, x_n\}$ are available, not corresponding labels.

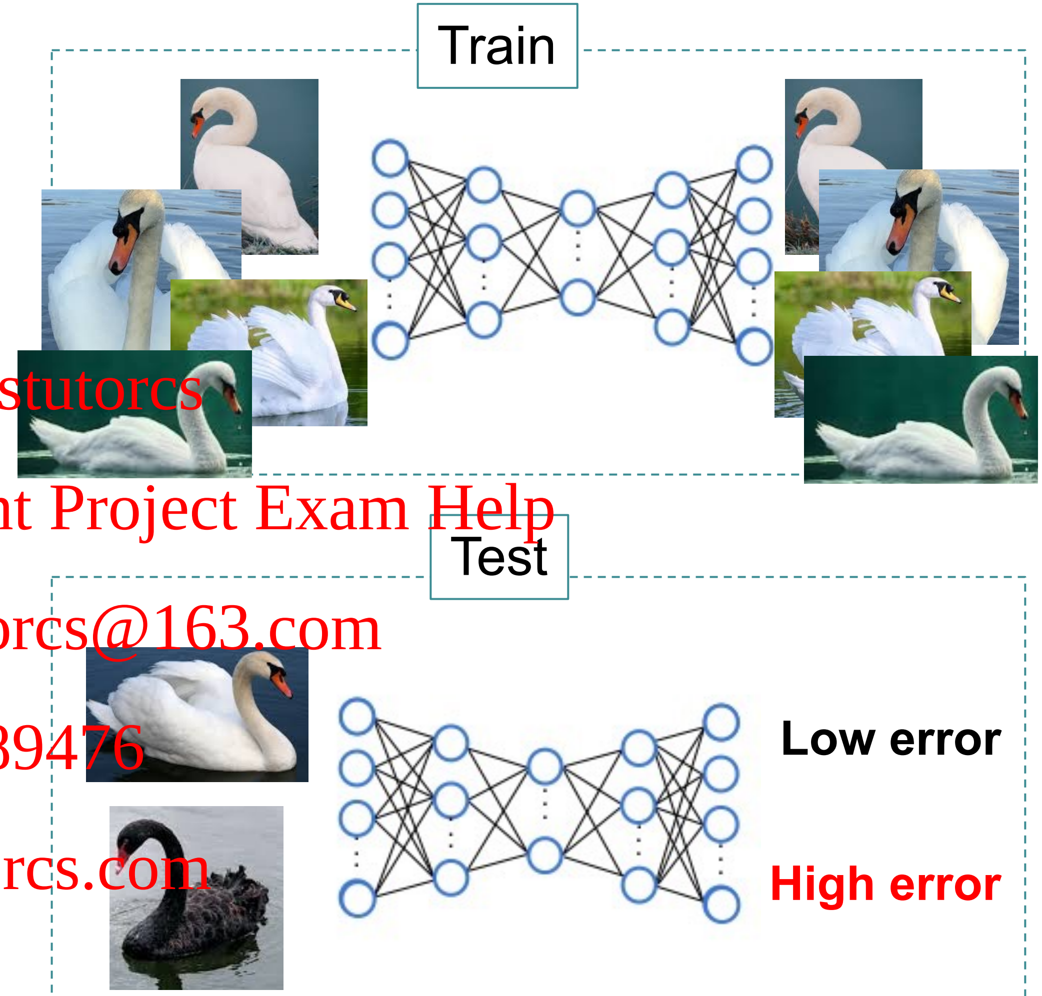
- Anomaly detection
- Extracting interesting infor[illegible]m data
  - Data compression (dimension reduction)
  - Clustering
  - Visualisation
- Representations learning

Using Autoencoder for Anomaly Detection

- Train an autoencoder on normal data
- Learn the distribution of normal data
- μ: mean of error in training data
- σ: standard deviation of error in training data
- **Identifying anomalies with 3σ rule:**

$$\left\| x_{anomaly} - \hat{x}_{anomaly} \right\|^2 \geq \mu + 3\sigma$$

  – Generate high error for anomalies.



Using Autoencoder for Anomaly Detection – Example

- Train samples

Sample	x	[illegible]
1	30, 12, 85	[illegible], 78
2	22, 18, 83	25, 13, 89
3	32, 21, 68	28, 18, 74

- Test samples

Sample	x_t	$\hat{x}_t$
1	32, 16, 81	29, 12, 79
2	19, 28, 63	27, 16, 88

Using Autoencoder for Anomaly Detection – Example

- Train samples

Sample	x	[illegible]	Error
1	30, 12, 85	[illegible], 78	8.3
2	22, 18, 83	25, 13, 89	8.4
3	32, 21, 68	28, 18, 74	7.8

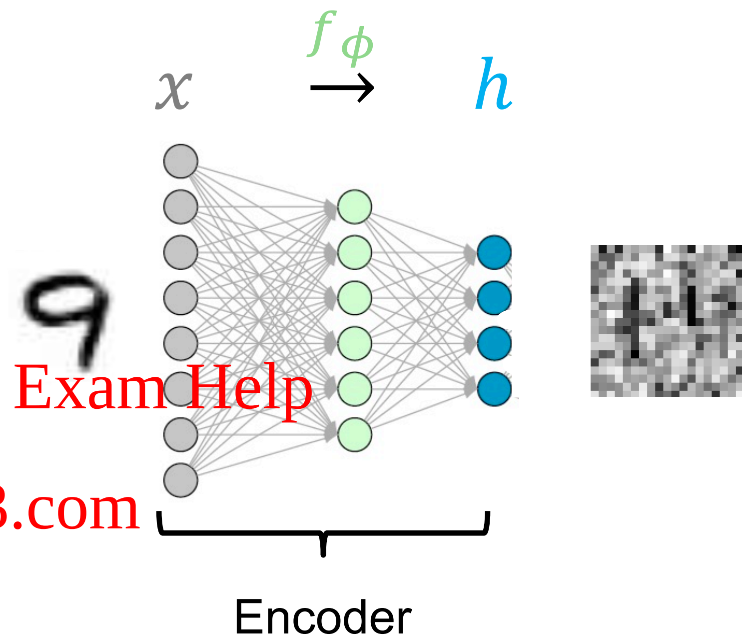
$$8.2 + 3\times 0.26 = 8.98$$

- Test samples

Sample	x	$\hat{x}$	Error
1	32, 16, 81	29, 12, 79	5.3
2	19, 28, 63	27, 16, 88	28.5

Data Compression

- **Objective:** Find a way to appropriately compress our input into a … “…ttleneck” vector of smaller dimensi… …coder).
  - Learns a Lossy Com… …f the input data.



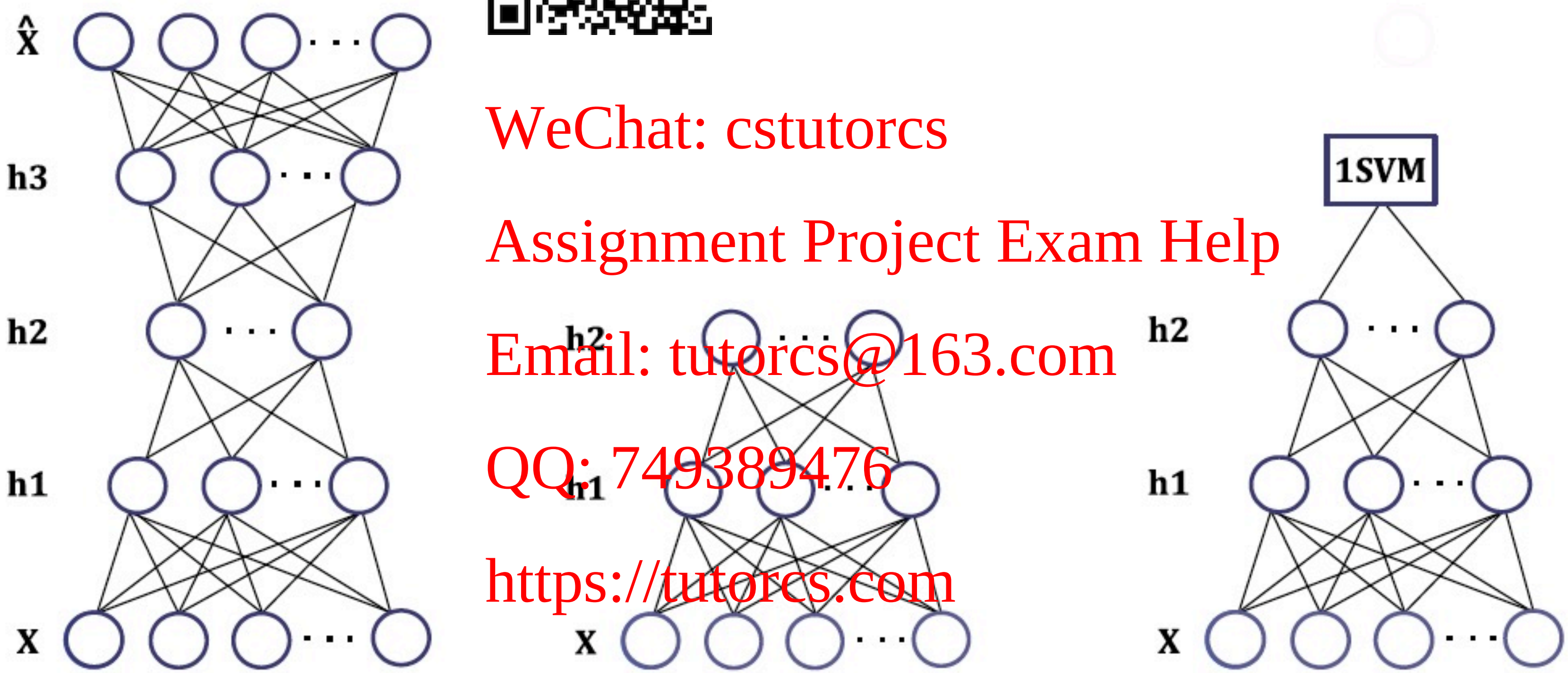
- **Special case:** f, g linear, $\mathcal{L}$ mean square error: reduces to Principal Component Analysis (PCA)
  - PCA: Data compression methods that reduces the dimensionality of the data while maintaining its essence.

Unsupervised Representation Learning

- Force the representations to better model input distribution
  - Not just extracting fe classification
  - Asking the model to representing the data and not overfitting to a particular task
  - Potentially allowing for better generalization

Application: Hybrid Anomaly Detection Model [3]

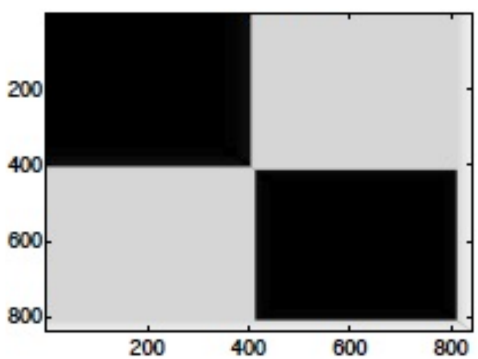
- Feed the output of the bottleneck into a simple model (e.g. k-NN, 1SVM, logistic regression. . .).



Application: Hybrid Anomaly Detection Model

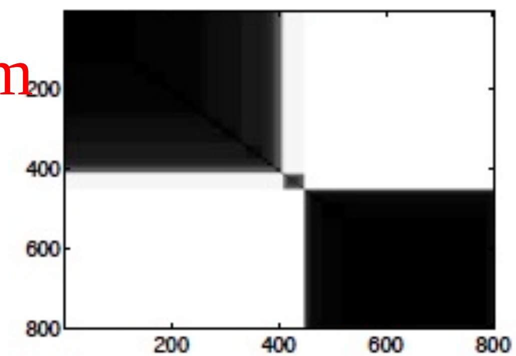
- Example of 2 dimensional Banana (two moon) dataset:

- iVAT images of 100 dimensional Banana dataset (with 5% random anomaly)



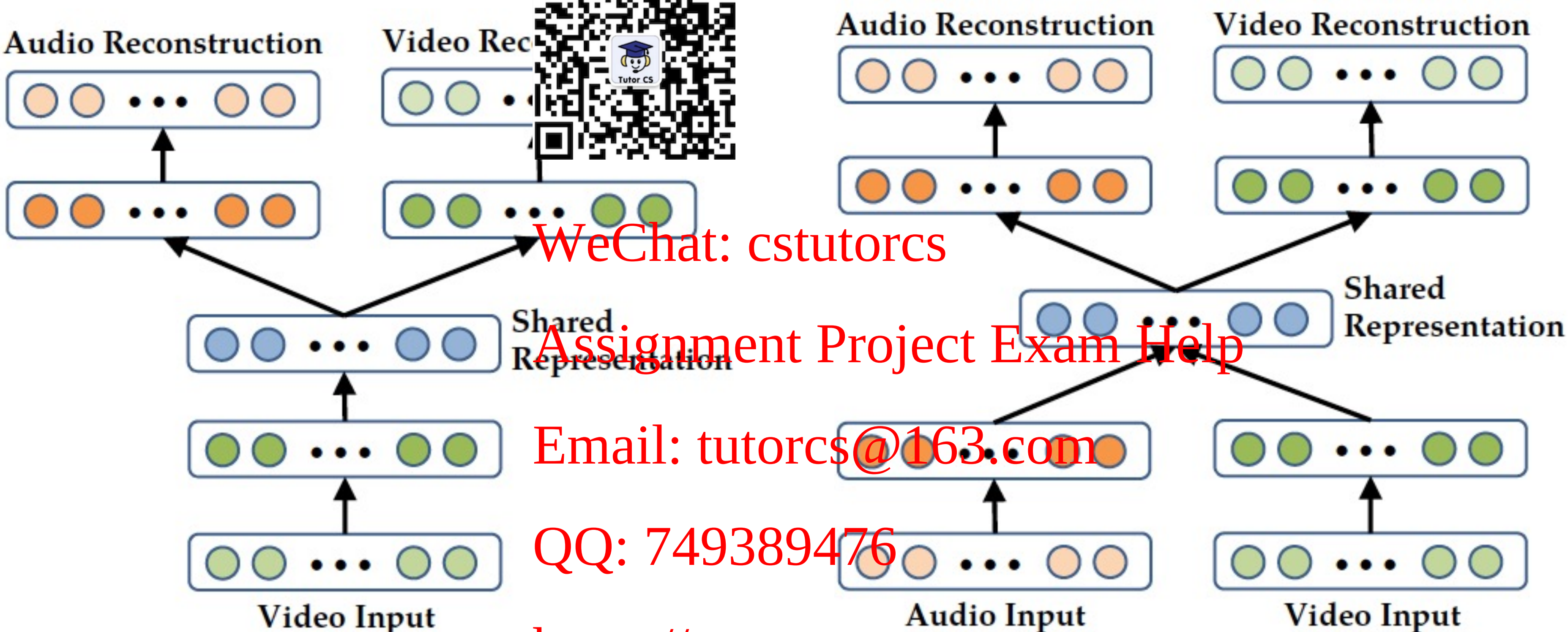
Input data

Output of 1st hidden layer



Output of 2nd hidden layer

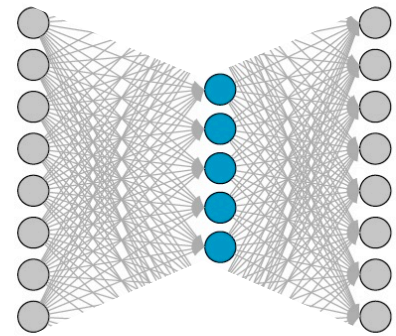
Application: Multimodal Learning [4]



Undercomplete vs. Overcomplete Hidden Layer

Undercomplete Representation:

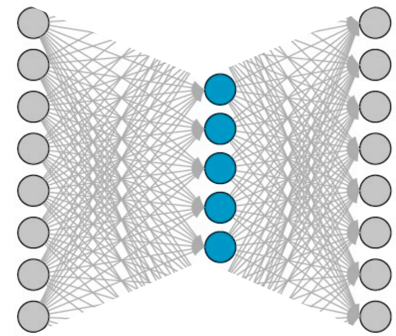
- Hidden layer is undercom[illegible]aller than the input layer
- Hidden layer compresses [illegible]
- Compresses well only for the training distribution
- Hidden units will be good features for the training distribution, but bad for other inputs



Undercomplete vs. Overcomplete Hidden Layer

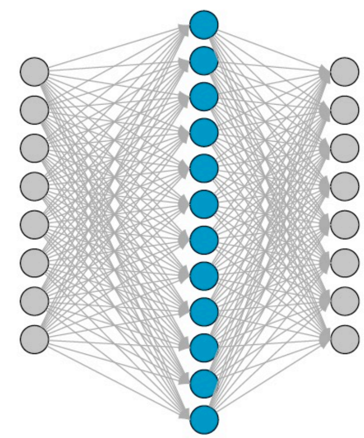
Undercomplete Representation:

- Hidden layer is undercom aller than the input layer
- Hidden layer compresses
- Compresses well only for the training distribution
- Hidden units will be good features for the training distribution, but bad for other inputs



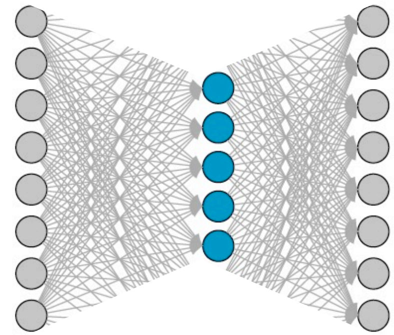
Overcomplete Representation:

- Hidden layer is overcomplete if greater than the input layer
- No compression in the hidden layer
- Each hidden unit could copy a different component
- No guarantee that the hidden unit will extract meaningful structure



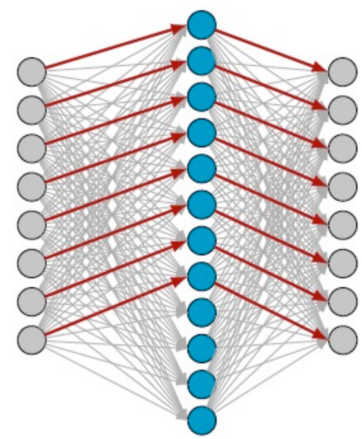
Undercomplete vs. Overcomplete Hidden Layer

Undercomplete Representation:



- Hidden layer is undercom aller than the input layer
- Hidden layer compresses
- Compresses well only for the training distribution
- Hidden units will be good features for the training distribution, but bad for other inputs

Overcomplete Representation:

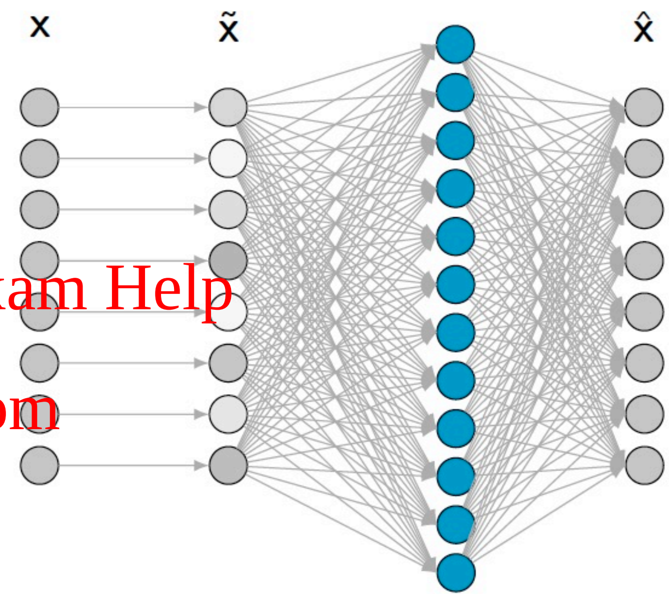


- Hidden layer is overcomplete if greater than the input layer
- No compression in the hidden layer
- Each hidden unit could copy a different component $x = \hat{x}$
- No guarantee that the hidden unit will extract meaningful structure

Denoising Autoencoder (DAE)

- **Idea:** Add noise to input but learn to reconstruct the original
  - Randomly assign a s [illegible] ata to 0, with probability ρ
  - Add Gaussian noise
- Reconstruct $\hat{x}$ from corrupted $\tilde{x}$
- Loss function minimises error between $\hat{x}$ and original sample x

$$\mathcal{L}(\theta,\phi) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i - \overbrace{g_\theta\left(f_\phi(\tilde{x}_i)\right)}^{\hat{x}_i}\right)^2$$

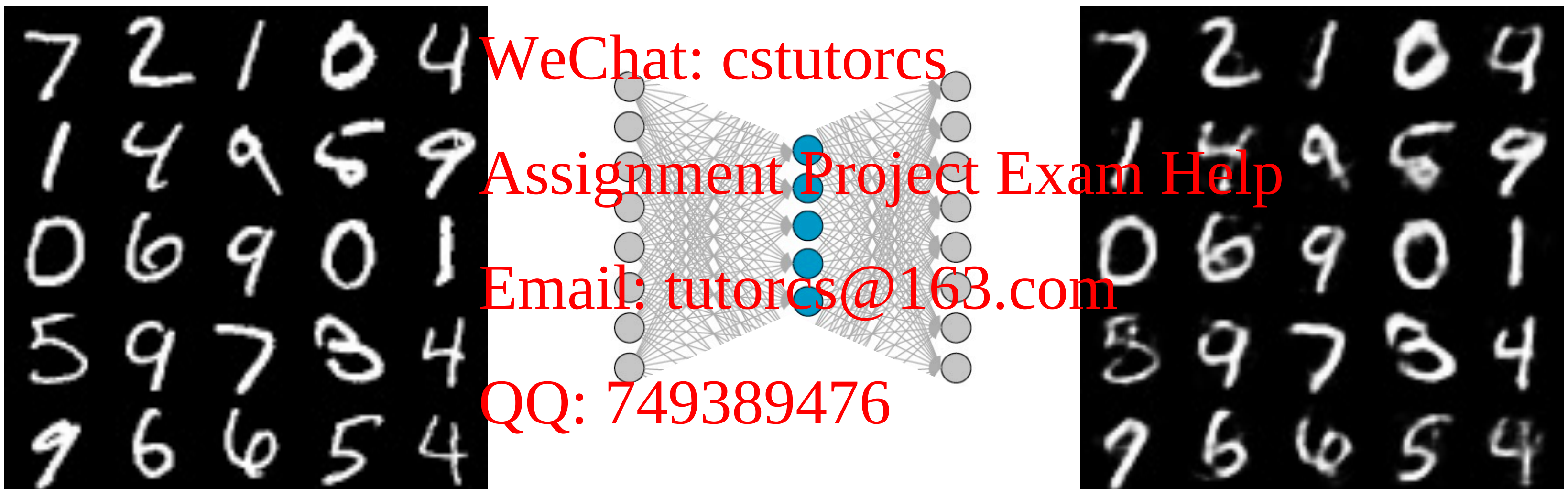


- Prevents copying
- Improves the representations and robustness
- **Note:** different noise is added during each epoch

Example of Autoencoder

Input x

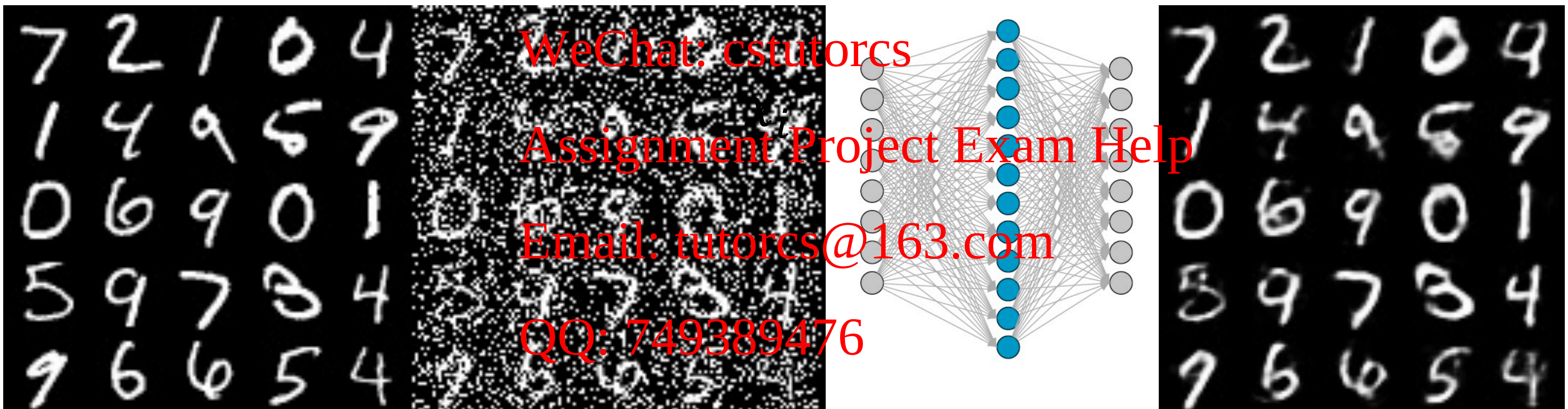
Output $\hat{x}$



$$\mathcal{L}(\theta, \phi) = \frac{1}{2}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

Example of Denoising Autoencoder

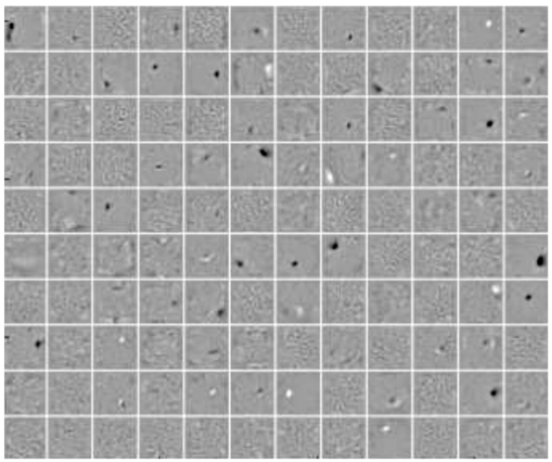
Original Data x Noisy Input $\tilde{x}$ Output $\hat{x}$



$$\mathcal{L}(\theta, \phi) = \frac{1}{2}\sum_{i=1}^{n}(x_i - \hat{x_i})^2$$

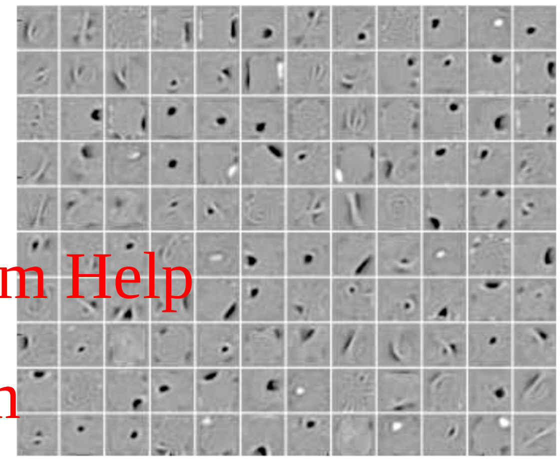
Example of Denoising Autoencoder

0% Noise

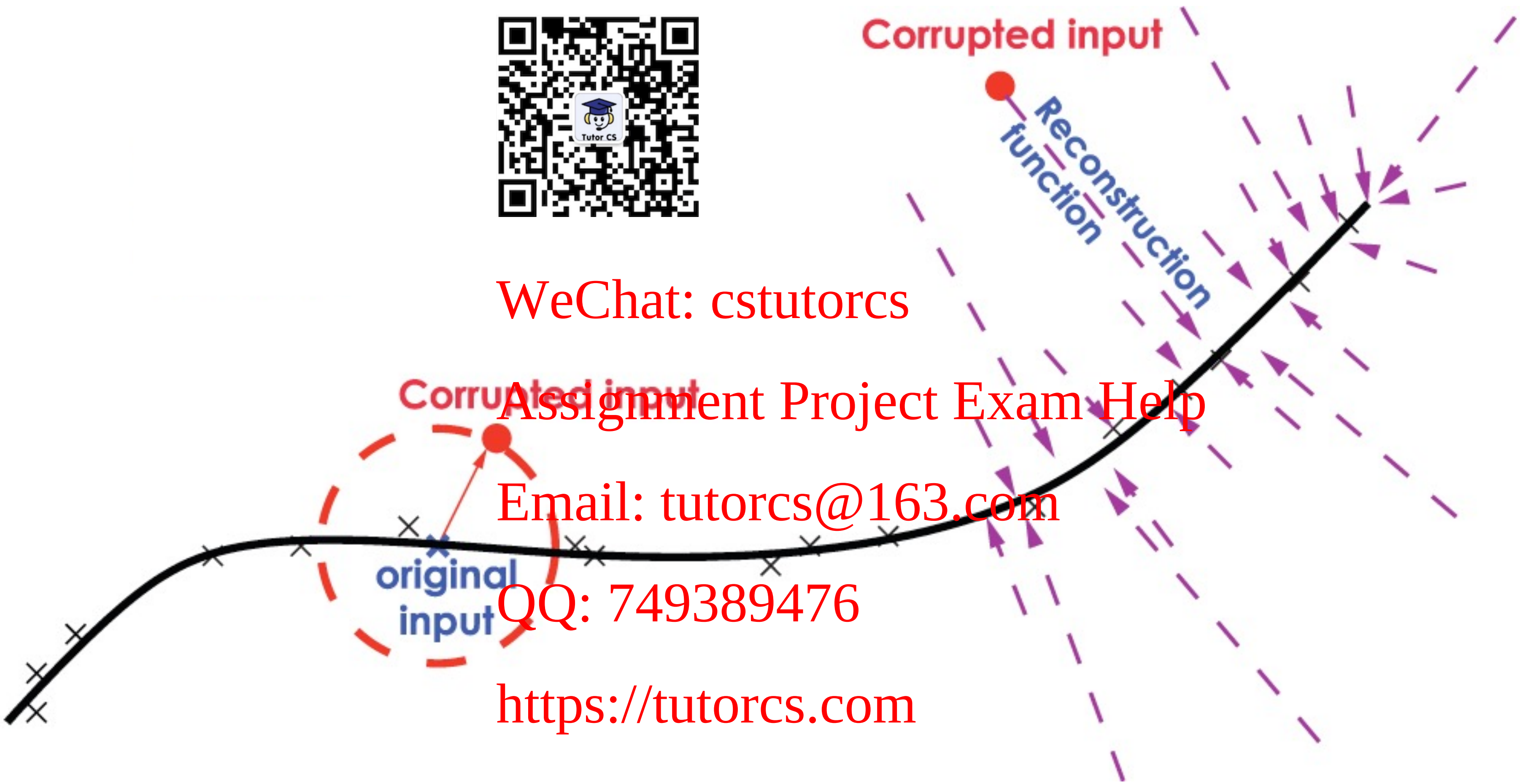


% Noise

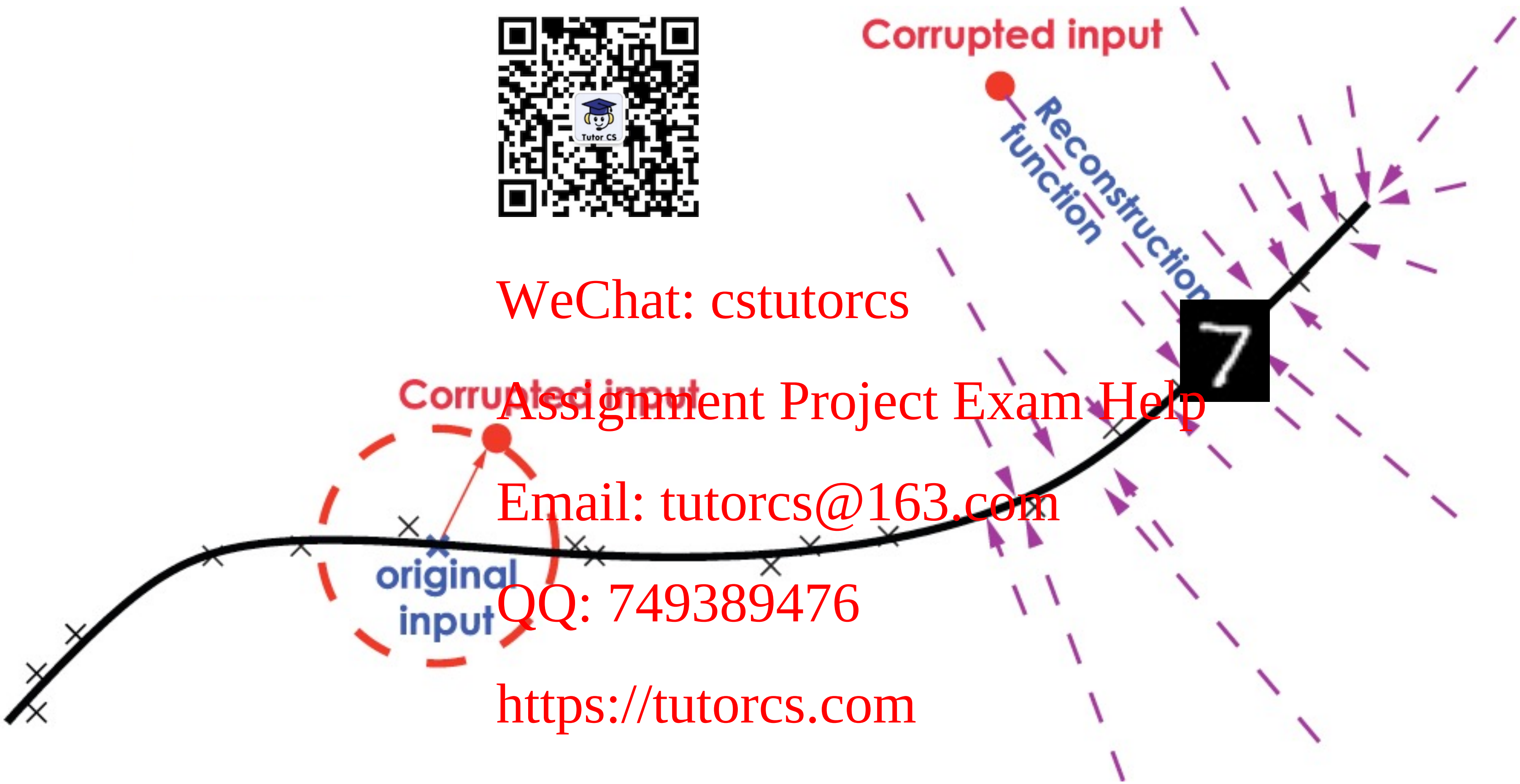
50% Noise



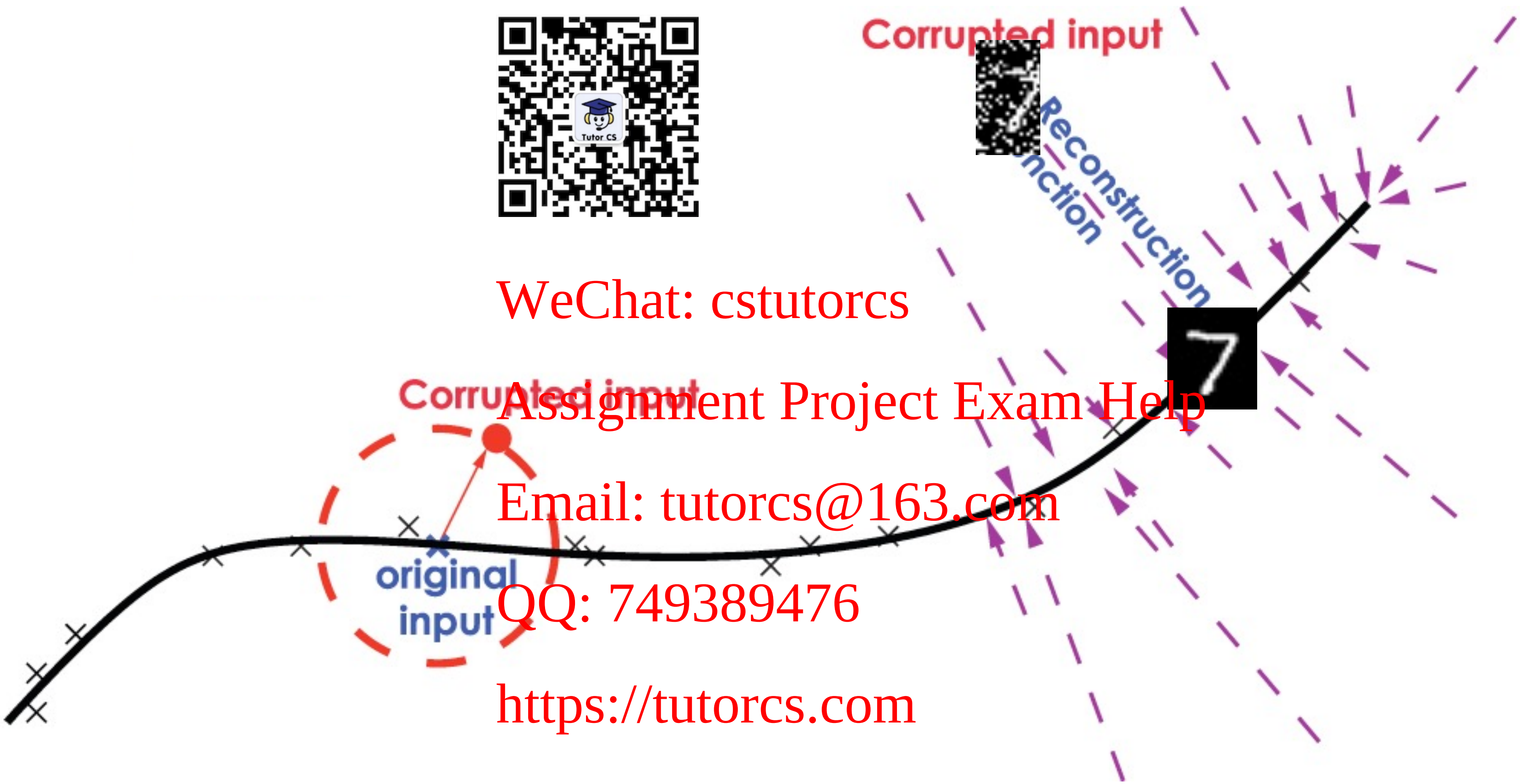
Intuition of Denoising Autoencoder



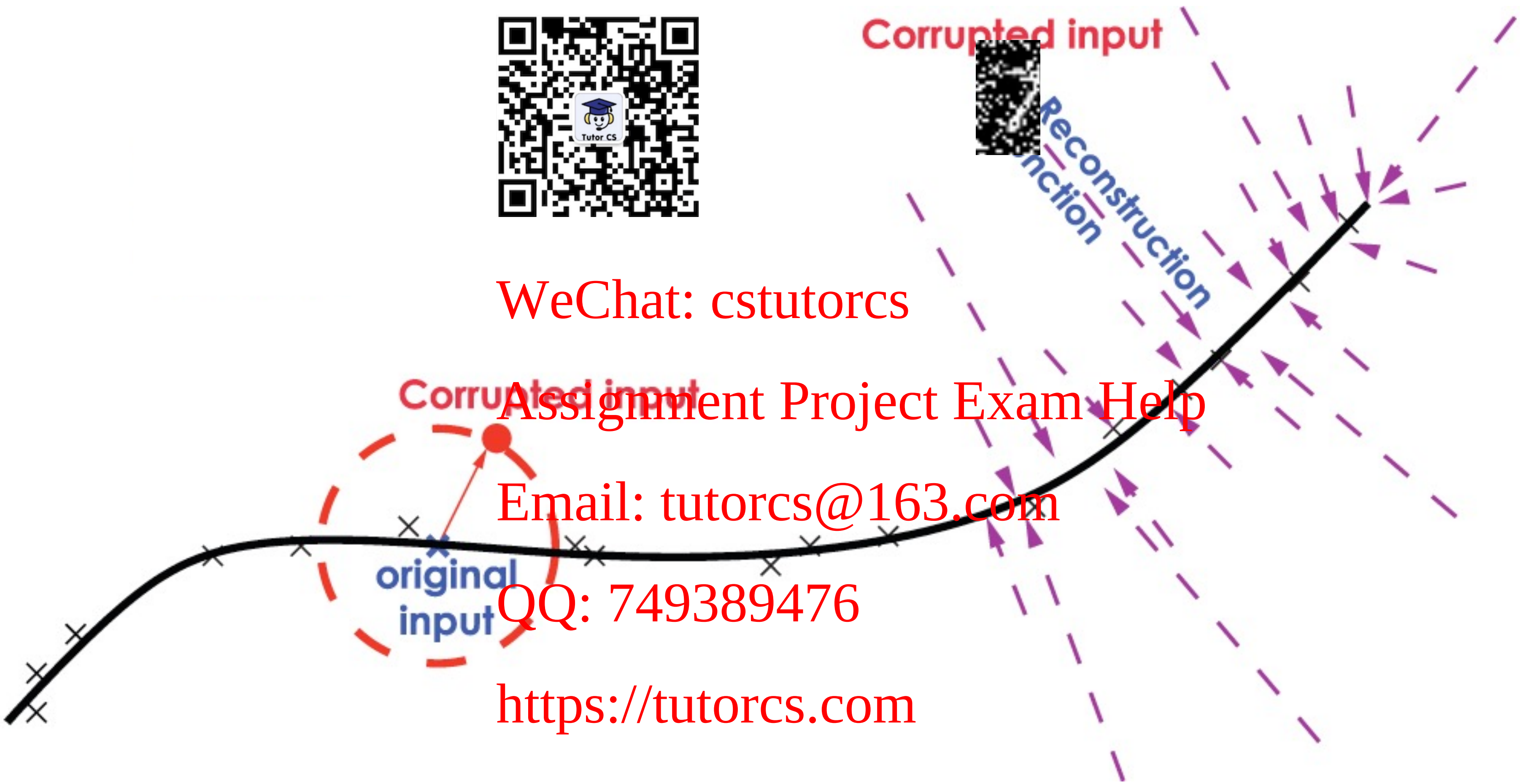
Intuition of Denoising Autoencoder



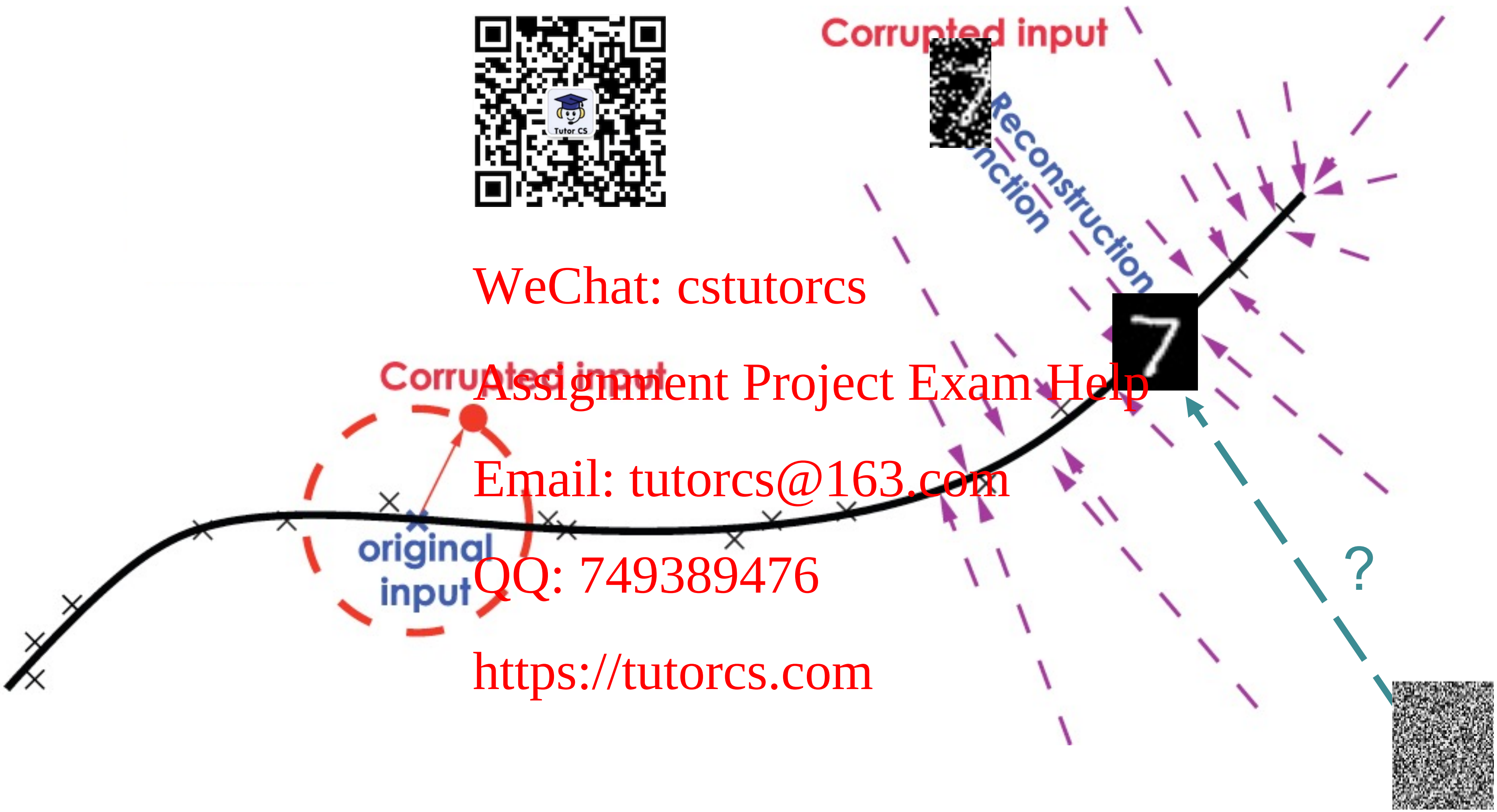
Intuition of Denoising Autoencoder



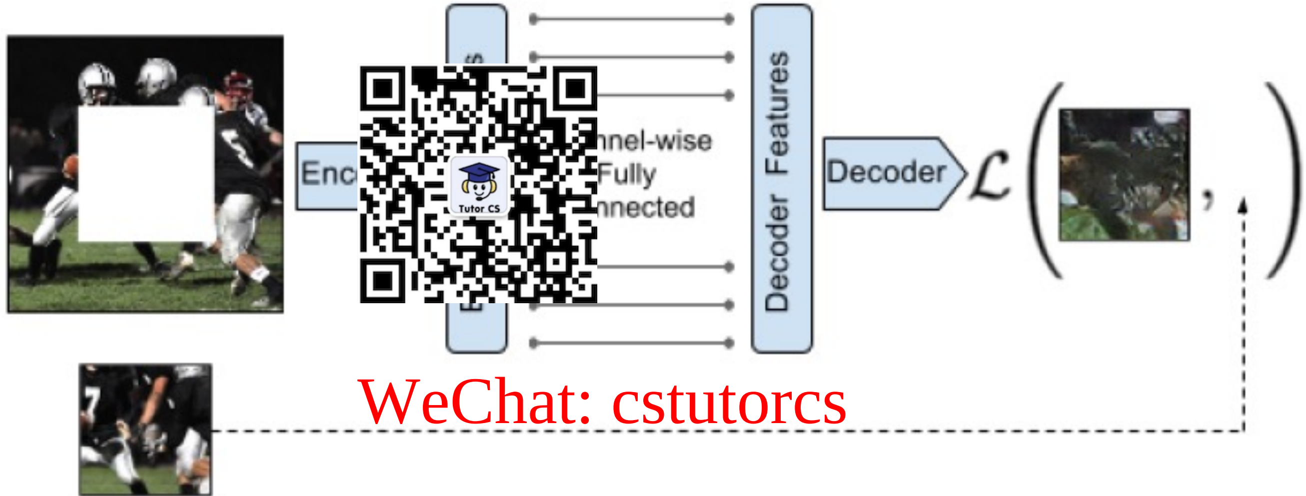
Intuition of Denoising Autoencoder



Intuition of Denoising Autoencoder



Application: Image Patching (Neural Inpainting) [9]



Problem with AEs

- Their latent space and their encoded vectors, may not be continuous.

Variational Autoencoder (VAE) [6]

- VAEs are in a class of models called generative models, they can be used to generate examples of in[illegible]y learning their statistics (e.g., mean and variance).

- Instead of learning $f_\phi(x)$ [illegible]), VAEs learn distributions of the features given the input, and the input given the activations, i.e., probabilistic versions of f_ϕ and g_θ. The VAE will learn:

  – $q_\phi(h|x)$: the distribution of the features given the input.

  – $p_\theta(x|h)$: the distribution of the input given the features.

- **Objective:** Find a distribution $q_\phi(h|x)$ of some latent variables h, which we can sample from $h \sim q_\phi(h|x)$, to generate new samples $\hat{x} \sim p_\theta(x|h)$.

Encoding of AE vs. VAE

Standard Autoencoder
(direct encoding coordinates)

Variational Autoencoder
(μ and σ initialize a probability distribution)

Why VAE Learns Data Distribution?

- Often times the data is noisy, and a model of the distribution of the data is more useful for a given applicat[illegible]

- The relationship between [illegible]ved variables and the latent variables can be nonlinear, in which cas[illegible] provides a way to do inference.

- The VAE is a generative model; by learning $p_\theta(x|h)$, it is possible to sample h and then sample x. This enables the generation of data that has similar statistics to the input.

VAE Architecture

VAE's Loss Function

$$\mathcal{L}(\theta, \phi) = \underbrace{\mathbb{E}_{q_\phi(h|x)}[\log p_\theta(x|h)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}(q_\phi(h|x) \| p_\theta(h))}_{\text{Regulariser (KL divergence)}}$$

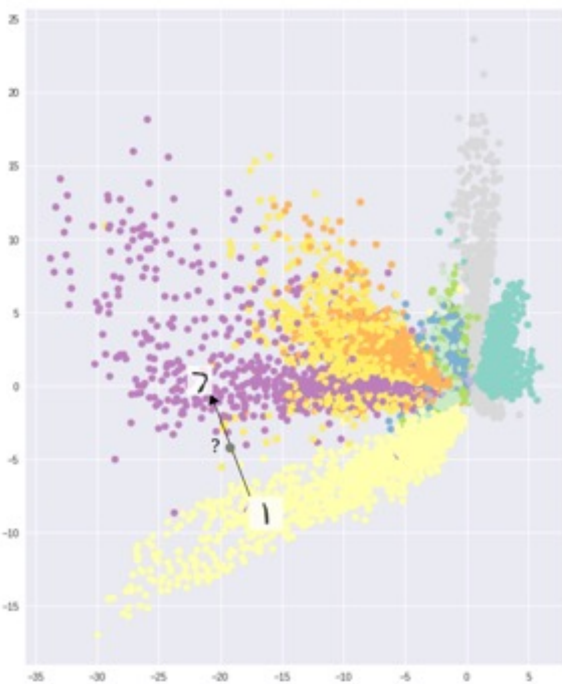
- **Reconstruction Loss:** The expected log-likelihood measures how well samples from $q_\phi(h|x)$ are able to explain the data x.

- **Regulariser:** Ensures that the explanation of the data $q_\phi(h|x)$ doesn't deviate too far from the prior distribution $p_\theta(h)$.

- **Kullback–Leibler (KL)-divergence:** Measure of difference between two distributions (the approximate posterior and the prior for h)

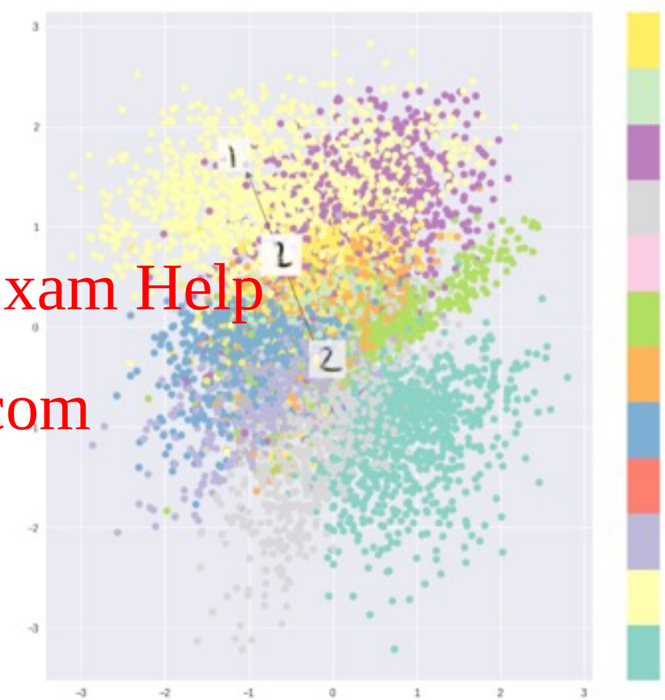
Intuition of VAE’s Loss Function [10]

Only reconstruction loss



divergence

Combination



Example: Two-dimensional Latent Space for MNIST

Demo: https://www.siarez.com/projects/variational-autoencoder

Application: Generating celebrity-lookalike photos

Application: Forecasting [7]

Disentangled VAE (β-VAE)

- Disentangle Representation: Each variable in the inferred latent representation h is only se[illegible] one single generative factor and relatively invariant to other factors.

- Extract very useful feature[illegible]y high dimensional space and use them to a task it wants to learn.

- Those features *generalise* to domains outside the training data, and enhance *interpretability*.

Non examinable

Disentangled VAE (β-VAE)

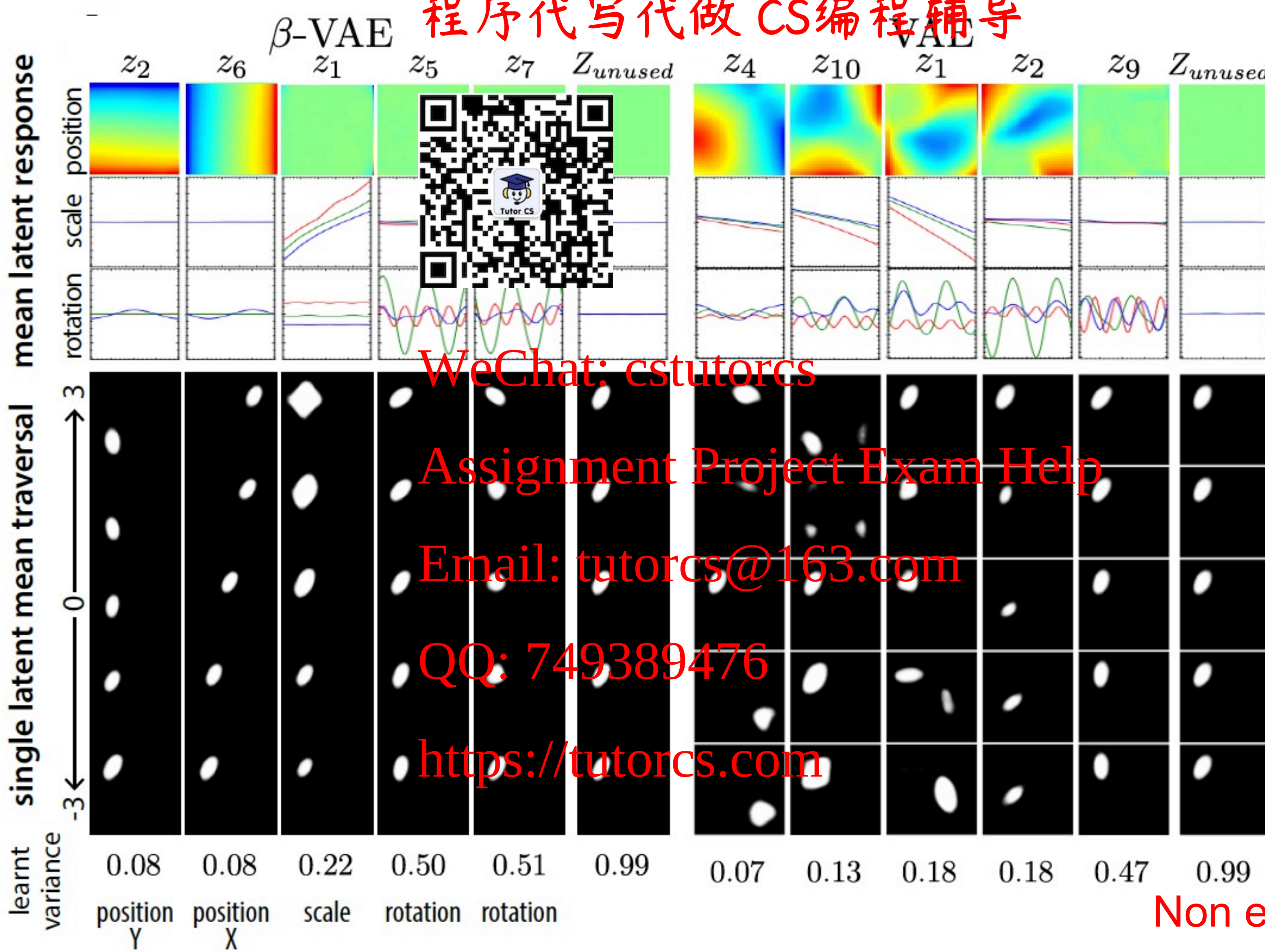
$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(h|x)}[\log p_\theta(x|h)] - \boxed{\beta} D_{KL}(q_\phi(h|x) \| p_\theta(h))$$

- For $\beta > 1$, it applies a stronger constraint on the latent bottleneck and limits the representation capacity of h.

- **Advantages:** Extremely flexible, even if each conditional is simple (e.g. conditional Gaussian), the marginal likelihood can be arbitrarily complex

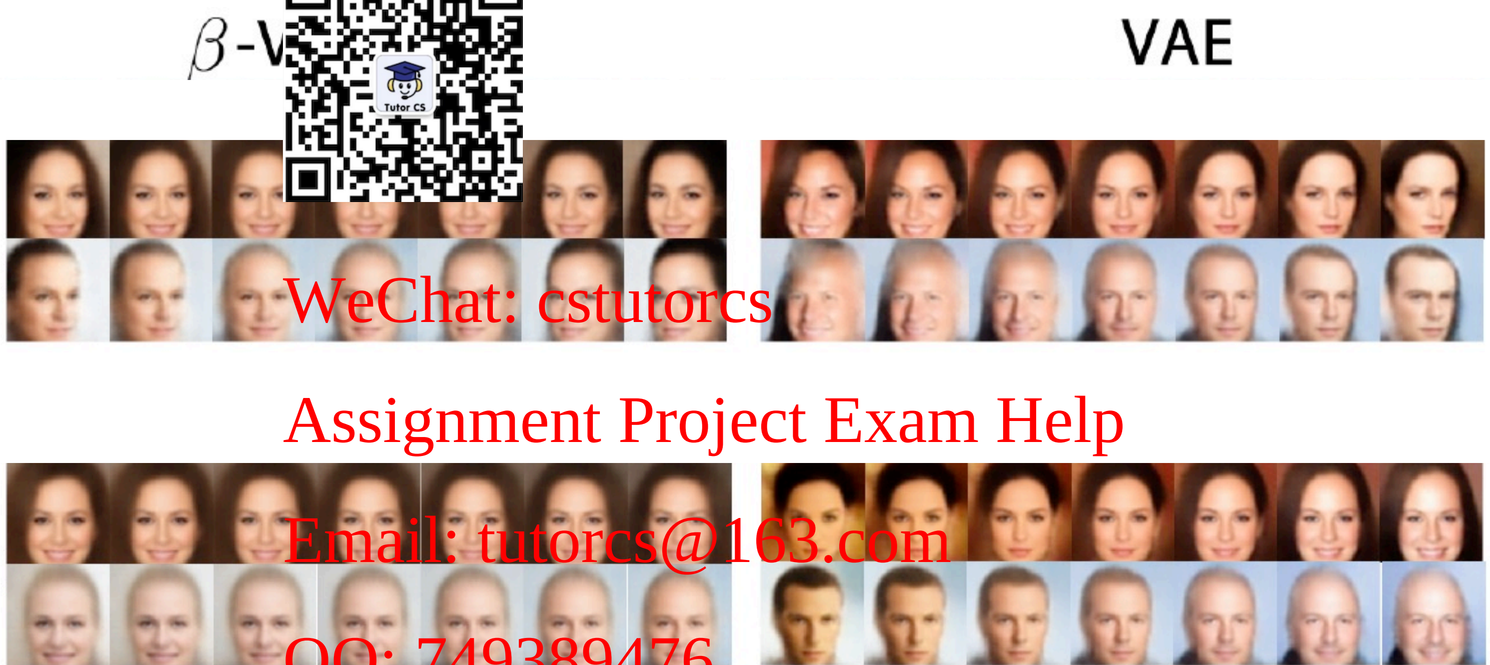
Non examinable

Representation Learning by β-VAE and VAE



Non examinable

Manipulating Latent Variables



Azimuth (Rotation)

Emotion (Smile)

Non examinable

Using VAE for Anomaly Detection [11]

- Training VAE: using only data of normal instances to learn $q_\phi(h|x)$ and $p_\theta(x|h)$
- For a test instance z
  - Evaluate the mean and standard deviation vectors with the probabilistic encoder $(\mu_z, \sigma_z) = q_\phi(h|z)$
  - Draw L samples from $h \sim N(\mu_z, \sigma_z)$
  - Compute the reconstruction probability

$$P_{recons}(z) = \frac{1}{L}\sum_{l=1}^{L} p_\theta(z|\mu_{\hat{z}_l}, \sigma_{\hat{z}_l})$$

$$(\mu_{\hat{z}_l}, \sigma_{\hat{z}_l}) = q_{\phi(x|h_l)}$$

  - z is anomaly if $P_{recons}(z) > \alpha$

Advantages of VAE for Anomaly Detection

- Latent variables are stochastic variables
  - The probabilistic enc[illegible]AE models the distribution of the latent variables (rather than [illegible] variable itself).
  - It can capture normal [illegible]alies which share the same mean, but different variance.

- Reconstructions are stochastic variables

  - Reconstruction probability considers the reconstruction error, as well as the variability of the reconstruction (by considering the variance parameter of the distribution function).

  - This property enables selective sensitivity to reconstruction according to variable variance.

Summary

- How to train a neural network?
- What is an autoencoder and what are its applications?
- How we can apply autoencoders to noisy data?
- What is a generative autoencoder and how we can use it for anomaly detection?

Next: Graph anomaly detection

References

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, “Deep Learning Book”, MIT Press, Chapter 14
2. Hugo Larochelle, Yoshu[illegible] Jérôme Louradour, Pascal Lamblin, “Exploring Strategies fo[illegible] Deep Neural Networks”, JMLR, 2009
3. Sarah Erfani, Sutharsha[illegible]jarar, Shanika Karunasekera, Christopher Leckie, “High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning”, Pattern Recognition, 2016.
4. Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. “Multimodal deep learning”, International Conference on Machine Learning, 2011
5. Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”, IEEE transactions on pattern analysis and machine intelligence, 2017.
6. Diederik P. Kingma, Max Welling, “Auto-Encoding Variational Bayes”, Neural Information Processing Systems (NeurIPS), 2014.

References

7. Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. "An uncertain future: Forecasting from static images using variational autoencoders." In *European Conference on Computer Vision,* 2016.

8. Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner, “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”, *ICLR,* 2017.

9. Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, Alexei A. Efros, “Context encoders: Feature learning by inpainting”, IEEE Conference on Computer Vision and Pattern Recognition, 2016.

10. Irhum Shafkat, “Intuitively Understanding Variational Autoencoders”,2018. https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

11. Jinwon An, Sungzoon Cho, “Variational autoencoder based anomaly detection using reconstruction probability”, *Special Lecture on IE* 2, (2015).

Other References

- Raghavendra Chalapathy, Sanjay Chawla, “Deep Learning for Anomaly Detection: A Survey”, 201 … rxiv.org/pdf/1901.03407.pdf
- Jinwon An, Sungzoon Ch … onal Autoencoder based Anomaly Detection using Reconstruction Prob … 015. http://dm.snu.ac.kr/static/d … NUDM-TR-2015-03.pdf

References

- http://dm.snu.ac.kr/static/d NUDM-TR-2015-03.pdf
- https://datascience.stacke com/questions/48533/why-use-variational-autoencoders-vae-instead ncoders-ae-in-anomaly-detec
- https://thingsolver.com/time-series-anomaly-detection-using-a-variational-autoencoder-vae/
- https://www.semanticscholar.org/paper/Variational-Autoencoder-based-Anomaly-Detection-An-Ch 1a00fceb3c78e8
- https://rstudio-pubs-static.s3.amazonaws.com/308801_ca2c3b7a649b4fd1838402ac0cb921e0.html#/39
- https://arxiv.org/pdf/1606.05908.pdf
- http://dm.snu.ac.kr/static/docs/TR/SNUDM-TR-2015-03.pdf
- https://arxiv.org/pdf/1903.00904.pdf