Please take several minutes to complete the Unit of Study Survey (USS)

https://student-surveys.sydney.edu.au/students/complete/form.cfm?key=uss221828

or through a shorter url: https://bit.ly/2WQNcAM

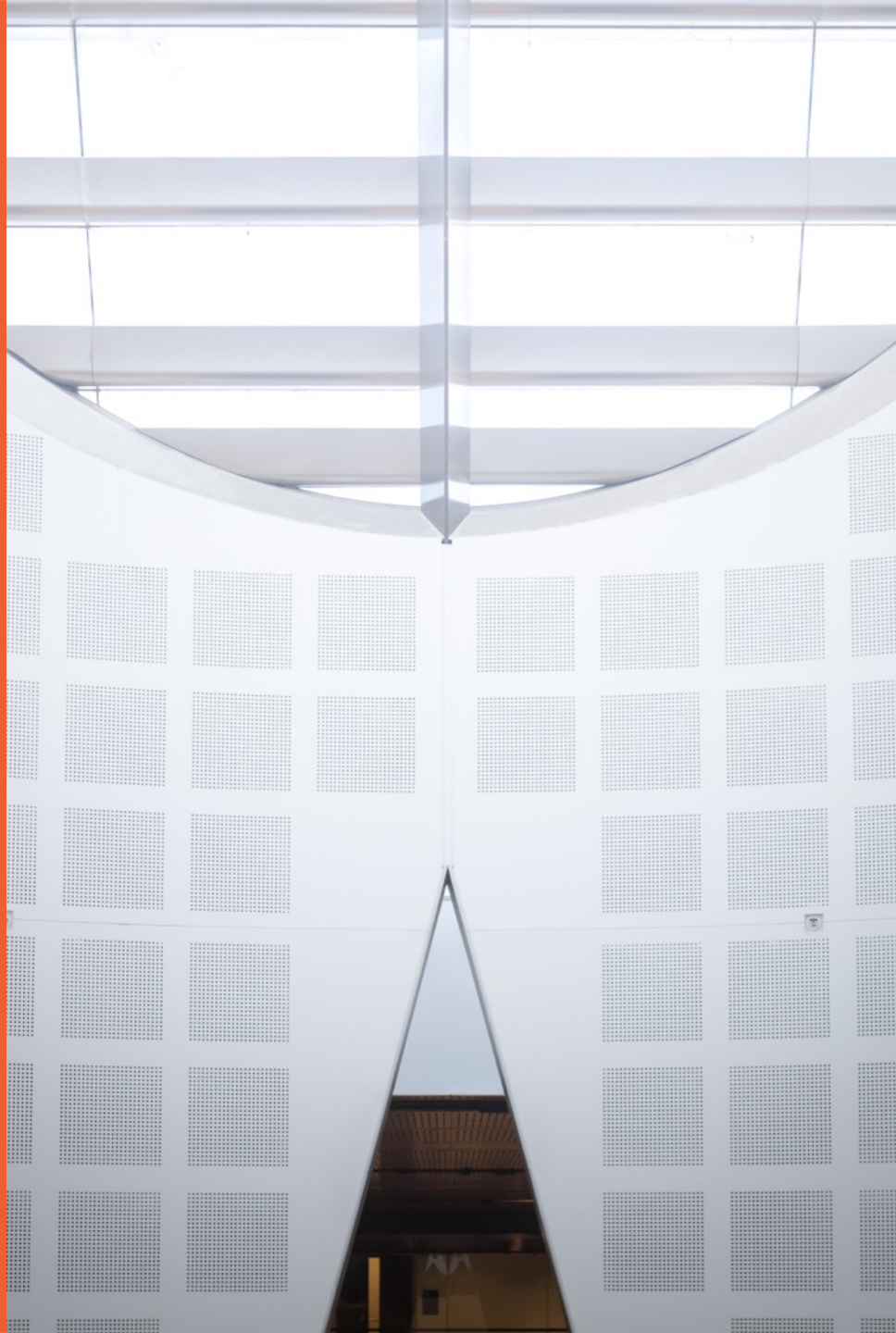
or by scanning a QR code

Deep Learning
COMP 5329

Dr Chang Xu

School of Computer Science

THE UNIVERSITY OF SYDNEY



What to Learn?

Week	Topic
1	Introduction to Deep Learning
2	Multilayer Neural Network
3	Optimization for Deep Models
4	Regularization for Deep Models
5	Quiz
6	Convolutional Neural Networks
7	Holiday
8	Neural Network Architectures
9	Recurrent Neural Network and LSTM
10	Graph Neural Networks
11	Deep Learning Applications
12	Deep Generative Models
13	Review

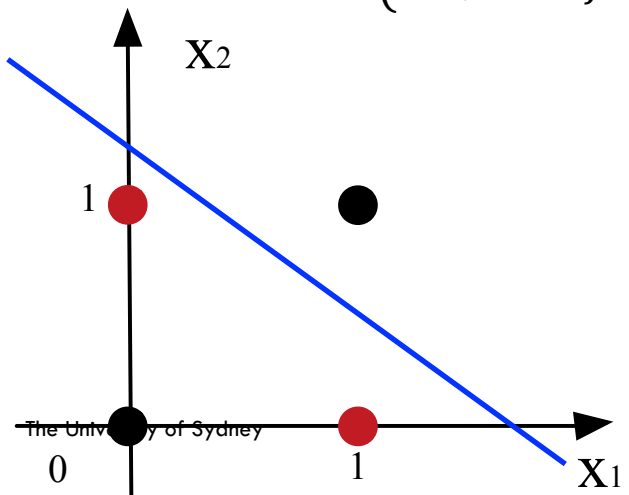
Multilayer Neural Network

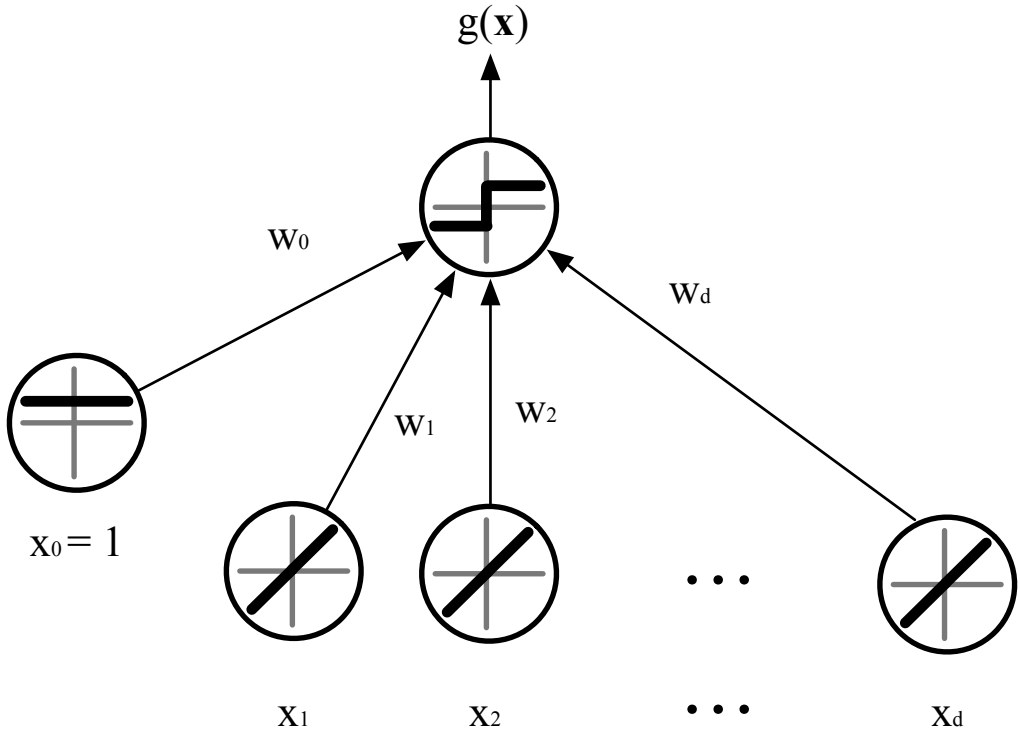
What is Perceptron? And How does it Work?

A Perceptron is a linear model used for binary classification. It models a neuron which has a set of inputs, each of which is given a specific weight. The neuron computes some function on these weighted inputs and gives the output.

$$g(\boldsymbol{x}) = f\left(\sum_{i=1}^{d} x_i w_i + w_0\right) = f(\boldsymbol{w}^T \boldsymbol{x})$$

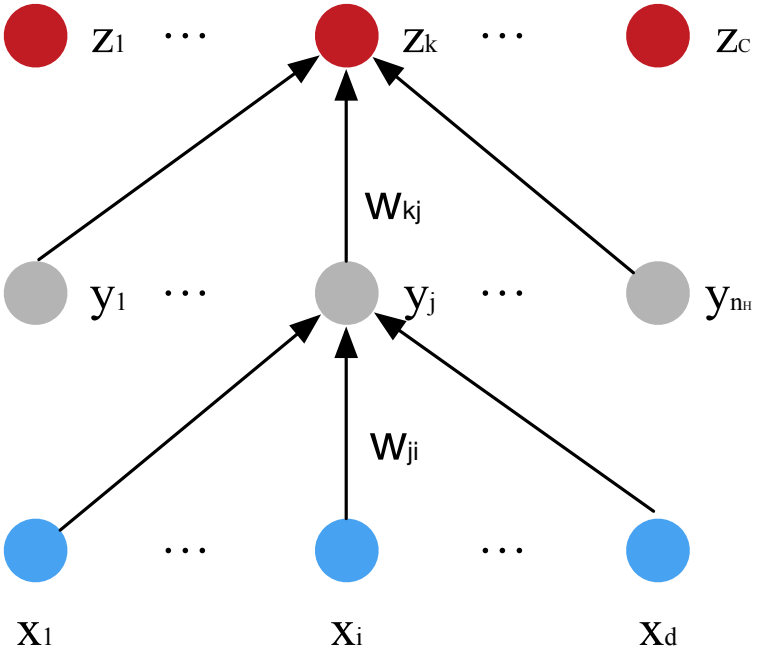
$$f(s) = \begin{cases} 1, & if\ s \geq 0 \\ -1, & if\ s < 0 \end{cases}$$





What is a Multi-Layer-Perceptron?

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP.

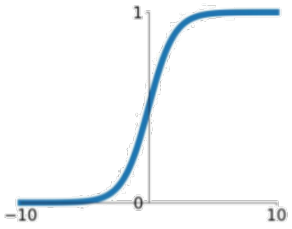


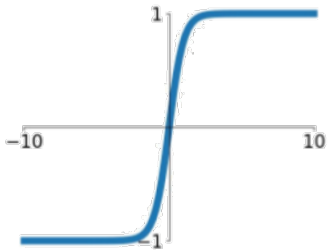
$$y_j = f(net_j)$$

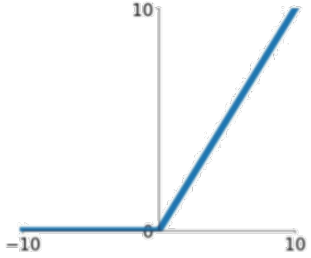
$$net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} = \boldsymbol{w}_j^T \boldsymbol{x}$$

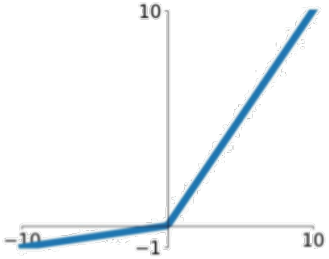
What are the activation functions?

Activation function translates the inputs into outputs. Activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.







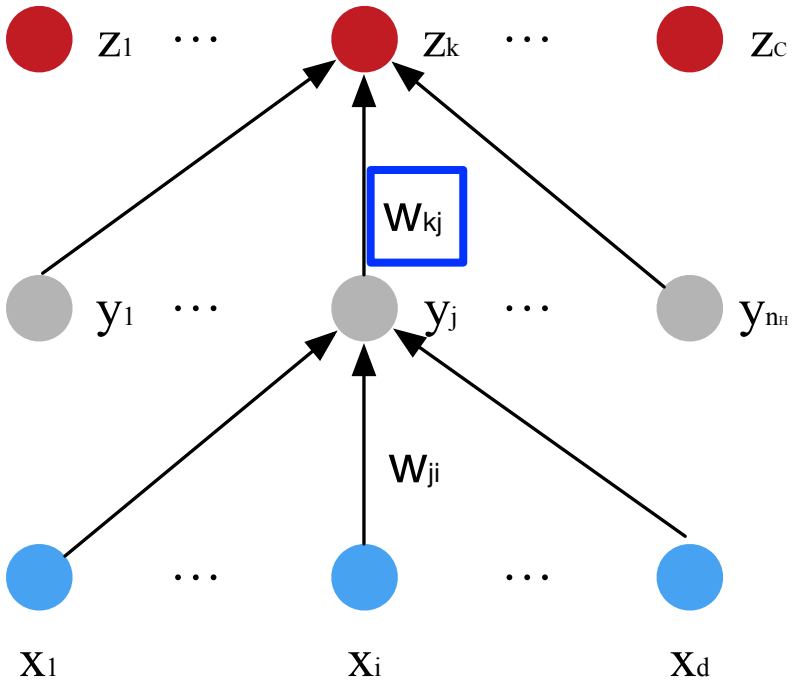


How to conduct backpropagation?

- Chain rule

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} y_j$$

$$\begin{cases} J(\boldsymbol{w}) = \dfrac{1}{2}\|\boldsymbol{t} - \boldsymbol{z}\|^2 \\ z_k = f(net_k) \\ net_k = \displaystyle\sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} \end{cases}$$



Analysis

- One-example based SGD
  - Estimation of the gradient is noisy, and the weights may not move precisely down the gradient at each iteration
  - Faster than batch learning, especially when training data has redundancy
  - Noise often results in better solutions
  - The weights fluctuate and it may not fully converge to a local minimum
- Mini-batch based SGD
  - Conditions of convergence are well understood
  - Some acceleration techniques only operate in batch learning
  - Theoretical analysis of the weight dynamics and convergence rates are simpler

What is the significance of a Cost/Loss function?

A cost function is a measure of the accuracy of the neural network with respect to a given training sample and expected output. It provides the performance of a neural network as a whole. In deep learning, the goal is to minimize the cost function. For that, we use the concept of gradient descent.

$$J(t,z) = \frac{1}{2}\sum_{k=1}^{C}(t_k - z_k)^2 = \frac{1}{2}\|\boldsymbol{t} - \boldsymbol{z}\|^2 \qquad J(t,z) = -\sum_{k=1}^{C} t_k \log z_k$$

In multi-class classification, softmax function before the output layer is to assign conditional probabilities (given x) to each one of the K classes.

$$\hat{P}(class_k|x) = z_k = \frac{e^{net_k}}{\sum_{i=1}^{K} e^{net_i}}$$

Optimization for Training Deep Models

A common training process for neural networks

1. Initialize the parameters

2. Choose an optimization algorithm

3. Repeat these steps:
    1. Forward propagate an input
    2. Compute the cost function
    3. Compute the gradients of the cost with respect to parameters using backpropagation
    4. Update each parameter using the gradients, according to the optimization algorithm

Gradient Descent Optimization

Methods	Adaptive lnearning rate	Consistent units	Easily handle Saddle points
SGD	No	No	No
Momentum	No	No	No
Nesterov	No	No	No
Adagrad	Yes	No	Yes
Adadelta	Yes	Yes	Yes
Rmsprop	Yes	No	Yes
Adam	Yes	No	Yes

Initialization

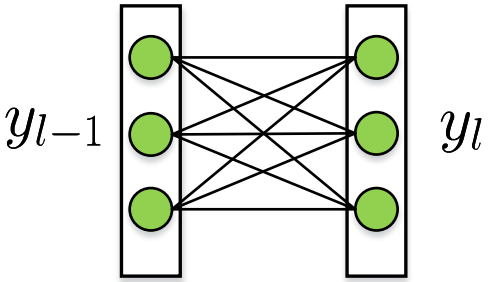
All zero initialization:

Every neuron computes the same output.

They will also compute the same gradients.

They will undergo the exact same parameter updates.

There will be no difference between all the neurons.



$$y_l = W y_{l-1} + b, \quad \frac{\partial \mathcal{L}}{\partial y_{l-1}} = W^\top \frac{\partial \mathcal{L}}{\partial y_l}$$
$$if \quad W = 0, \quad then \quad \frac{\partial \mathcal{L}}{\partial y_{l-1}} = 0$$
$$then, \quad \frac{\partial \mathcal{L}}{\partial y_m} = 0, \quad m = 0, ...l-1.$$

How to find appropriate initialization values

1. The *mean* of the activations should be zero.
2. The *variance* of the activations should stay the same across every layer.

Under these two assumptions, the backpropagated gradient signal should not be multiplied by values too small or too large in any layer. It should travel to the input layer without exploding or vanishing.

Initialization (Xavier) -- [Xavier Glorot, Yoshua.Bengio 2010]

We worked on activations computed during the forward propagation, and get

$$Var(W^{[l]}) = \frac{1}{n^{[l-1]}}$$

The same result can be derived for the backpropagated gradients:

$$Var(W^{[l]}) = \frac{1}{n^{[l]}}$$

Xavier initialization would either initialize the weights as

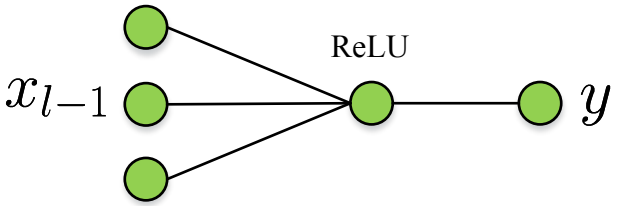
xavier_normal $\quad \mathcal{N}(0, \frac{2}{n^{[l-1]} + n^{[l]}})$

xavier_uniform $\quad \mathcal{U}(-\frac{\sqrt{6}}{\sqrt{n^{[l-1]}+n^{[l]}}}, \frac{\sqrt{6}}{\sqrt{n^{[l-1]}+n^{[l]}}})$

Initialization（He） -- [Kaiming He, Xiangyu Zhang, et al. 2015]

If we use ReLU as the activation function, n is the number of inputs, then for layer l we have:

$$\mathrm{Var}(y_l) = (n\mathrm{Var}(w))\, E[x_{l-1}^2]$$
$$x_{l-1} = max(0, y_{l-1})$$



If y_{l-1} has zero mean, and has a symmetric distribution around zero:

$$E[x_{l-1}^2] = E[(max(0, y_{l-1}))^2] = \frac{1}{2}\mathrm{Var}[y_{l-1}]$$

So we have: $\mathrm{Var}(w) = \frac{2}{n}$

$$Var(x) = E(x^2) - \big(E(x)\big)^2$$

kaiming_normal $\mathcal{N}(0, \frac{2}{n})$

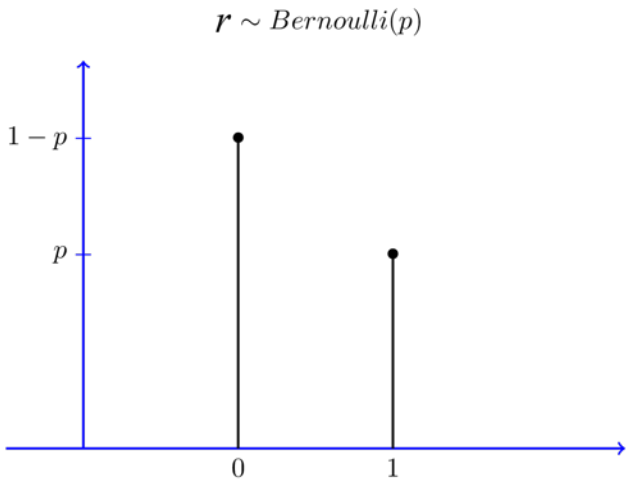
kaiming_uniform $\mathcal{U}(-\sqrt{\frac{6}{n}}, \sqrt{\frac{6}{n}})$

Regularizations for Deep Models

- Weight decay
- Robustness to noise
  - Noise to the input
  - Noise to the weights
- Data augmentation
- Early stopping

Dropout

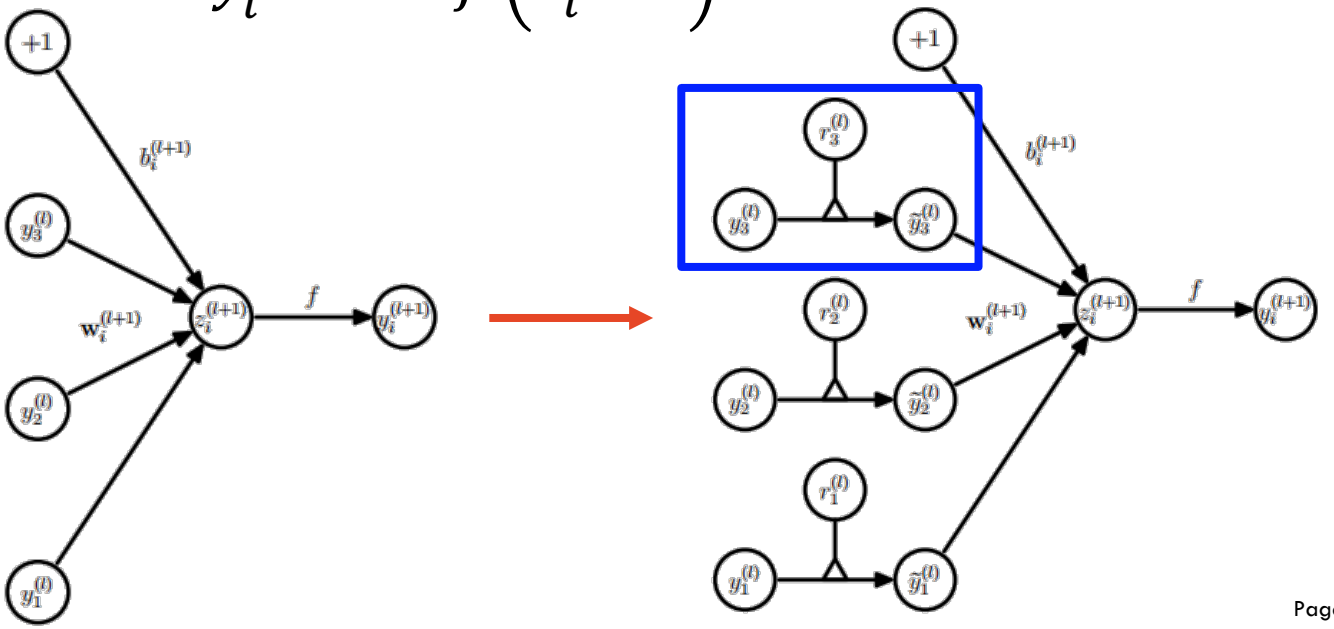
With dropout: $r_i^{(l)} \sim \text{Bernoulli}(p)$



$$\tilde{\boldsymbol{y}}^{(l)} = \boldsymbol{r}^{(l)} * \boldsymbol{y}^{(l)}$$

$$z_i^{(l+1)} = \boldsymbol{w}_i^{(l+1)} \tilde{\boldsymbol{y}}^{(l)} + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right)$$



Batch Normalization

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

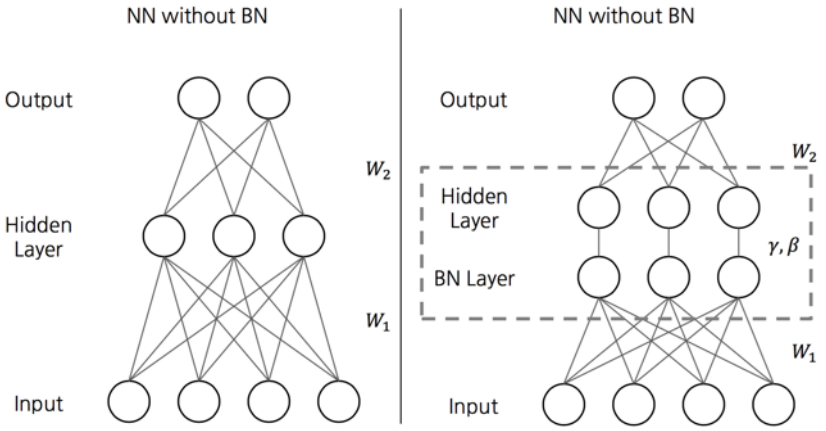
$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$
$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

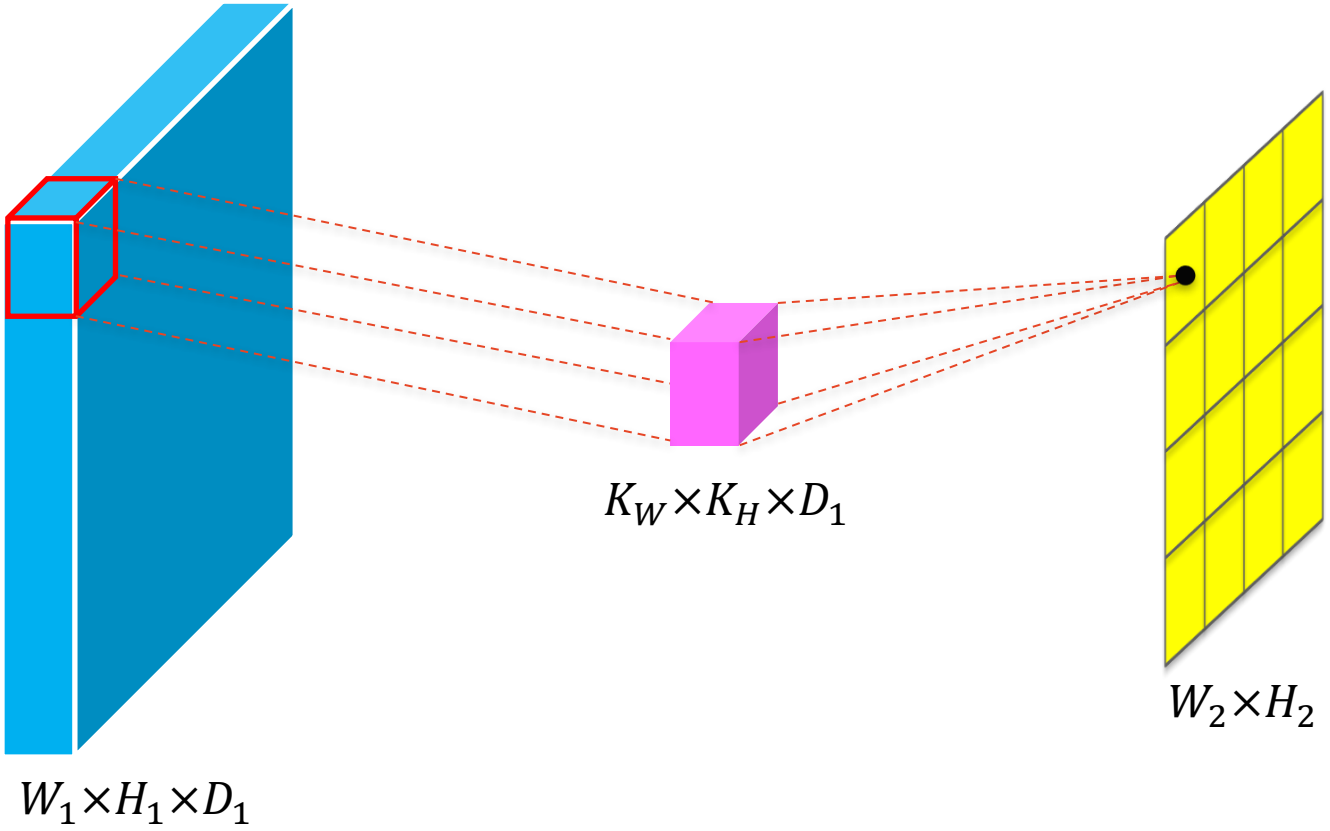


Convolutional Neural Networks

Convolutional Layer

- Suppose stride is (S_W, S_H) and pad is (P_W, P_H)

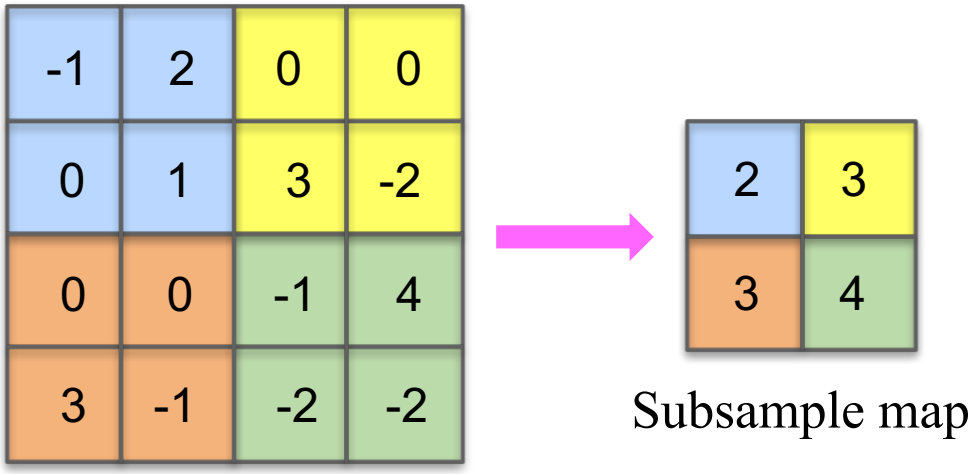
$$W_2 = \frac{W_1 + 2P_W - K_W}{S_W} + 1 \text{ and } H_2 = \frac{H_1 + 2P_W - K_H}{S_H} + 1$$



Pooling

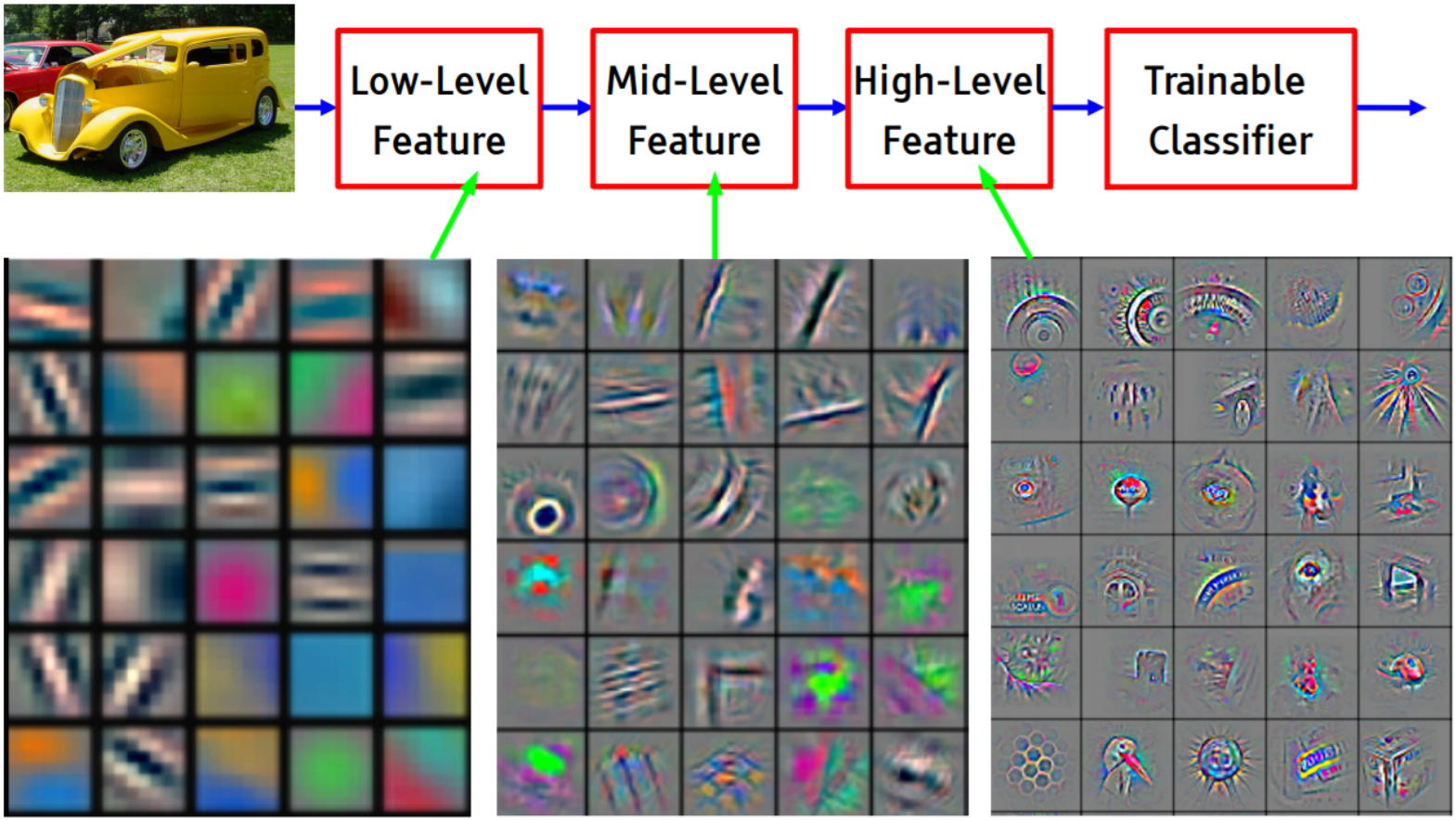
Max pooling

- Filter size: (2,2)
- Stride: (2,2)
- Pooling ops: max(·)



Visualize features

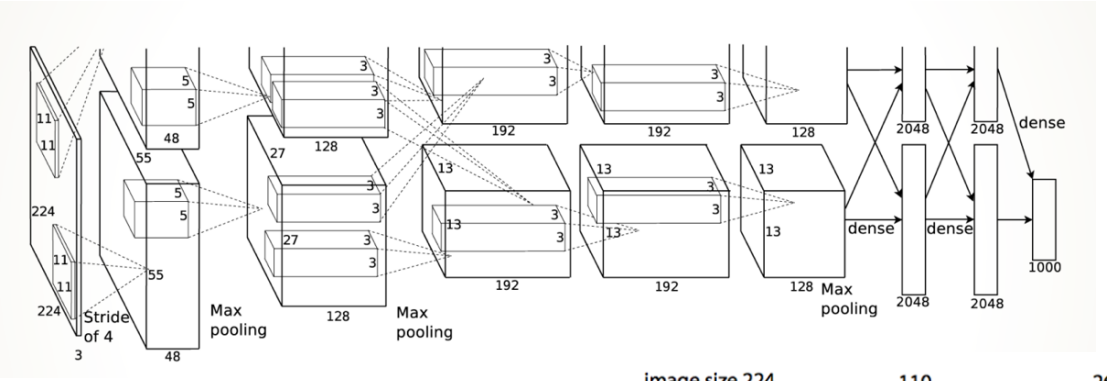
- Give insight into the function of intermediate feature layers and the operation of the classifier

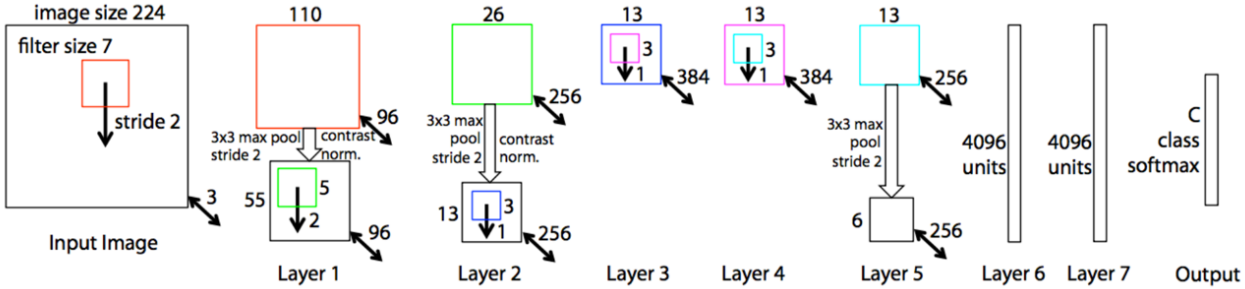


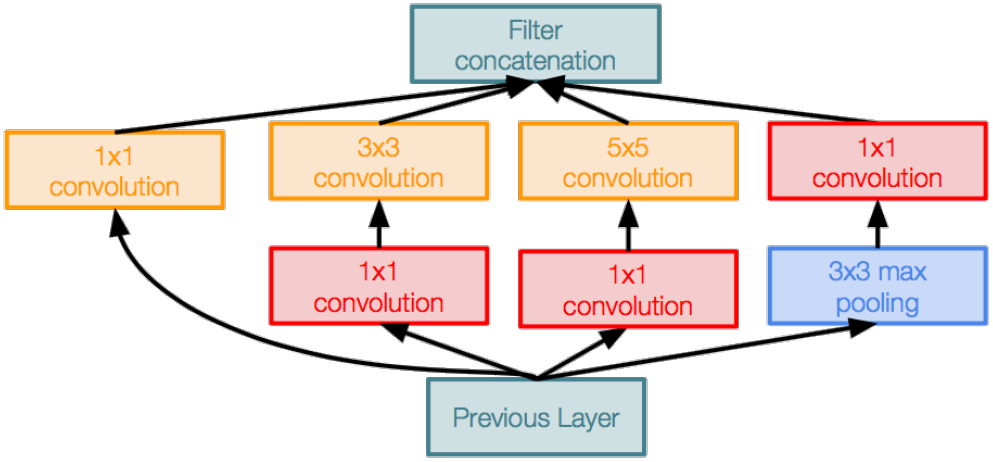
(Zeiler and Fergus, 2014)

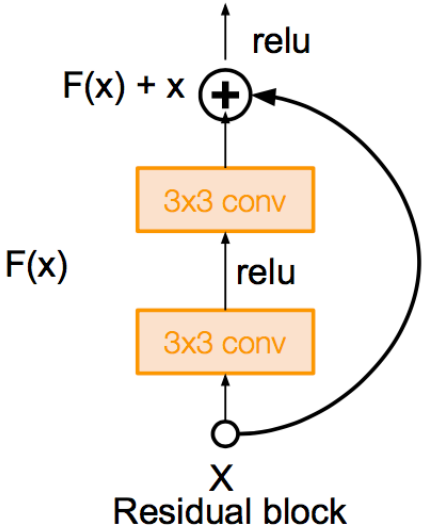
Network Structures

Popular CNN Architectures





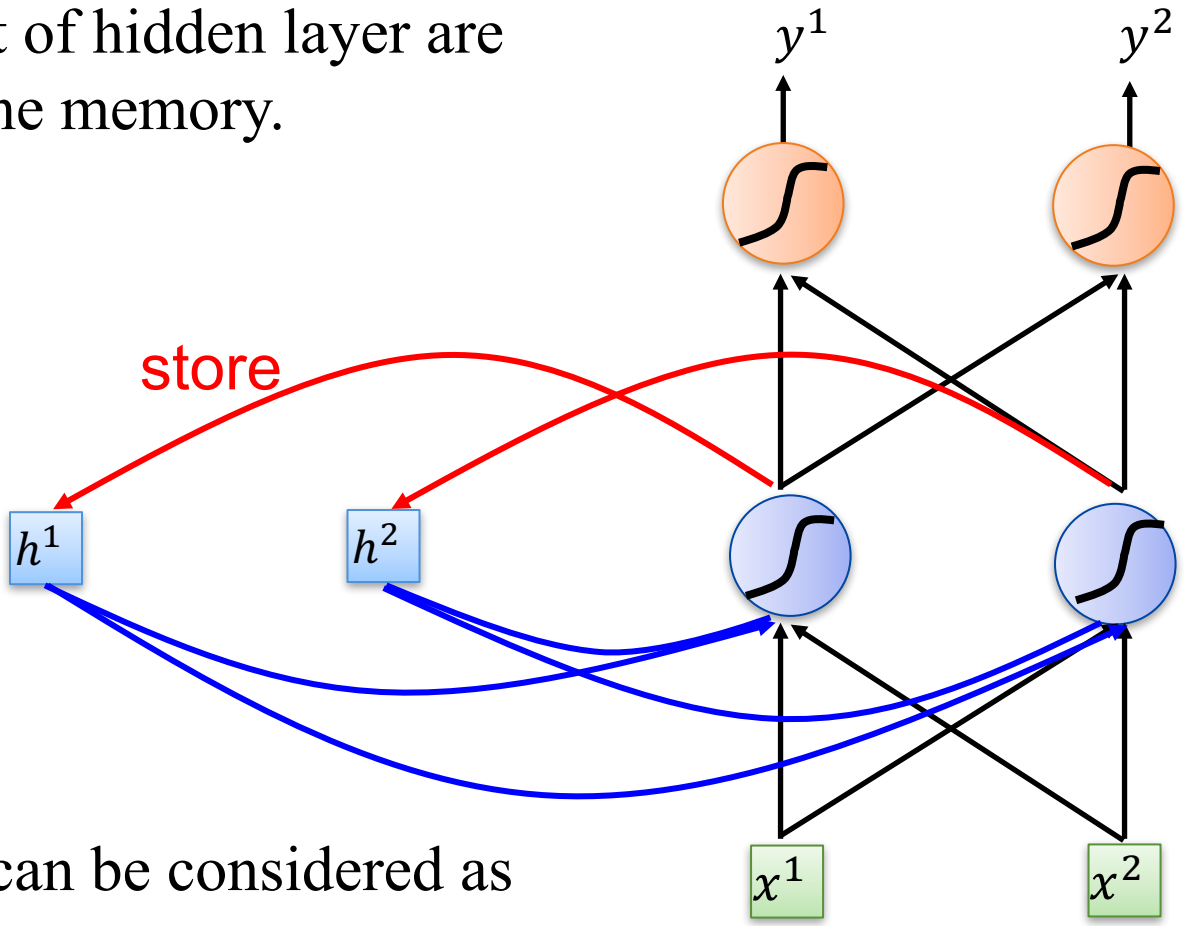




Recurrent Neural Network and LSTM

Recurrent Neural Network

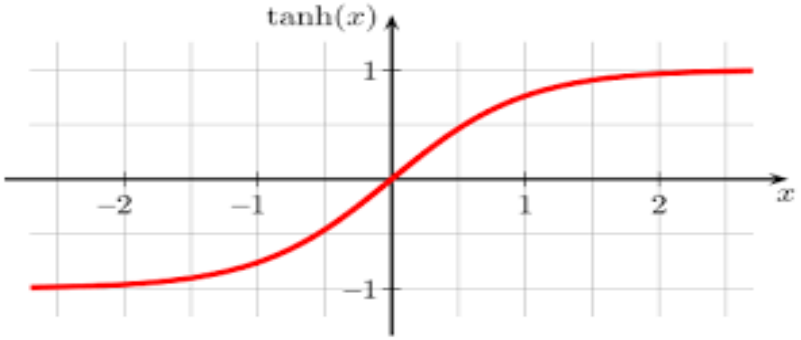
The output of hidden layer are stored in the memory.

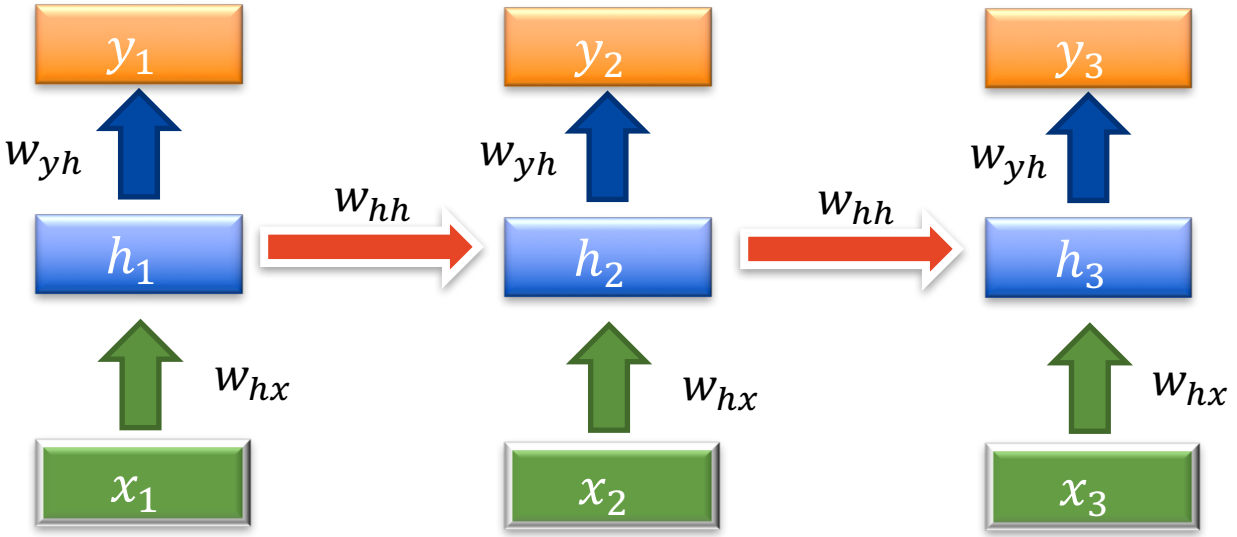


Memory can be considered as another input.

Recurrent Neural Network

- Formally
  - x_t is the input
  - h_t is the hidden state
  - y_t is the output
  - $h_0 = \vec{0}$
  - $h_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
  - $y_t = w_{yh}h_t$





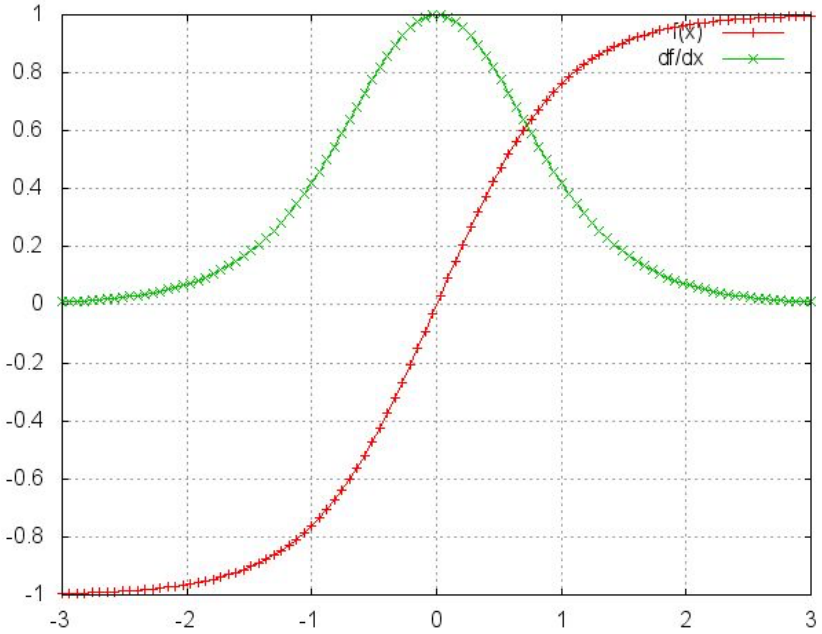
Recurrent Neural Network

- Vanishing gradient problem
  - $h_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
  - $y_t = w_{yh}h_t$
  - For every step, its loss is E_n

$$0.9^{1000} \approx 0$$
$$1.01^{1000} \approx 21000$$

$$\frac{\partial E_3}{\partial w_{hh}} = \frac{\partial E_3}{\partial y_3}\frac{\partial y_3}{\partial h_3}\frac{\partial h_3}{\partial w_{hh}}$$

$$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots$$



$(\tanh)' \leq 1$

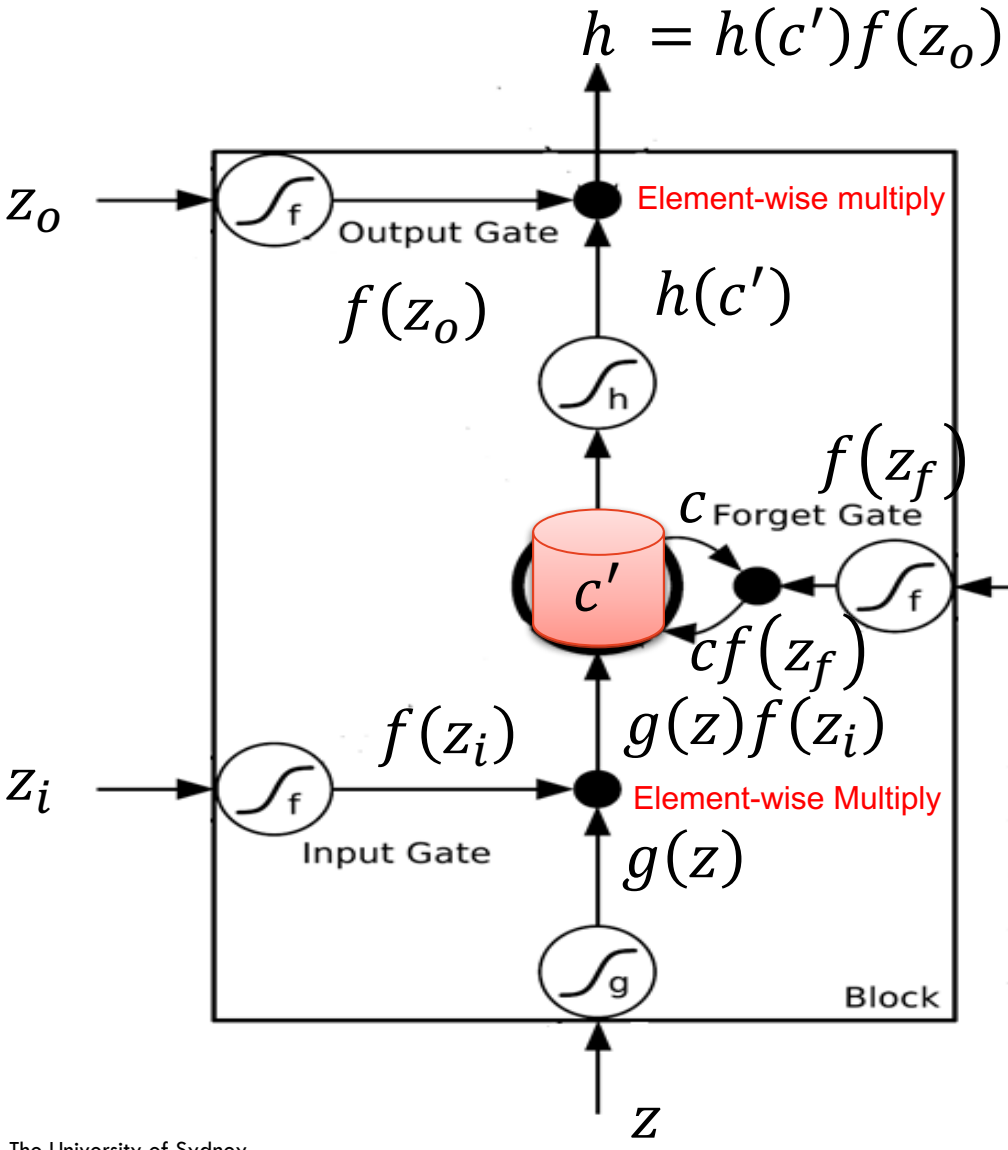
If $0 < w_{hh} < 1$, Vanishing Gradients

$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots \to 0$

If w_{hh} is larger, Exploding Gradients

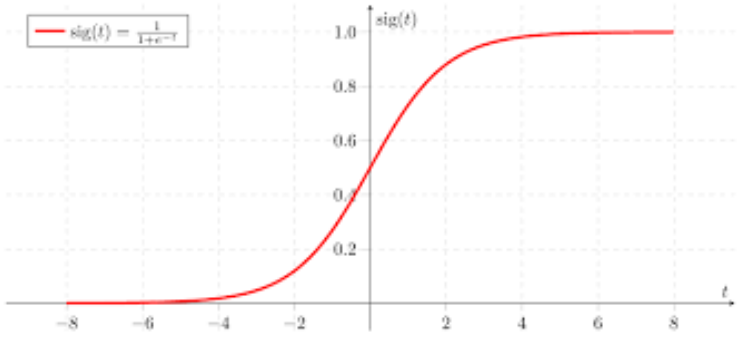
$(\tanh)' w_{hh} (\tanh)' w_{hh} (\tanh)' w_{hh} \cdots \to \infty$

Long Short-Term Memory



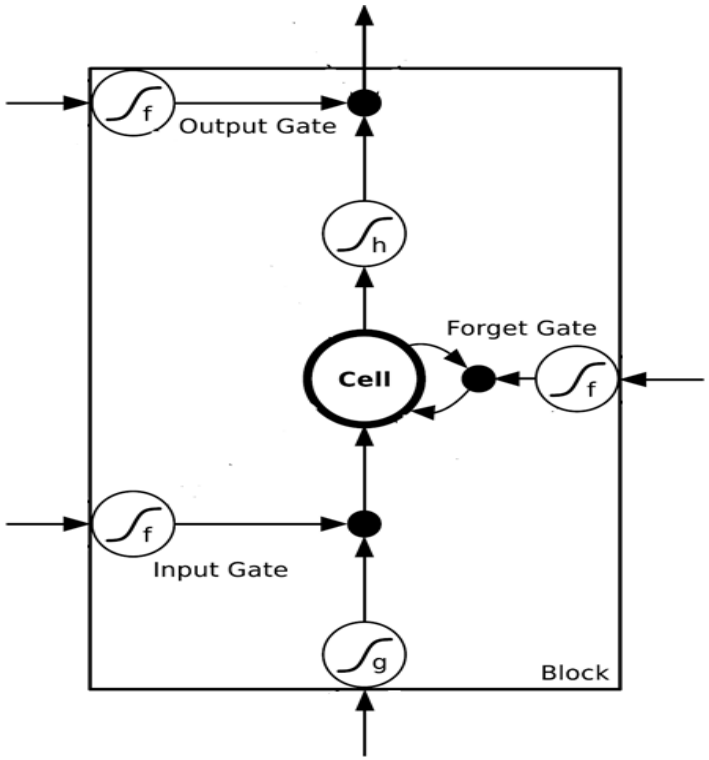
f is usually a sigmoid function
Value From 0 to 1
Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$



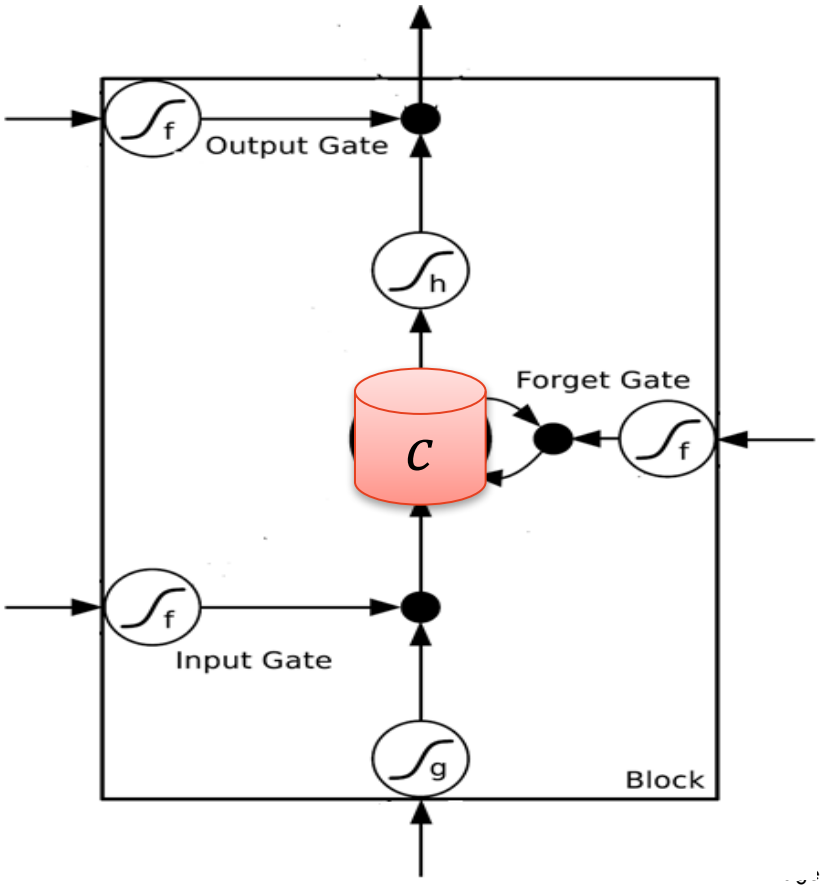
Long Short-Term Memory

- Forget gate
  - $f_t = \sigma(w_{fh}h_{t-1} + w_{fx}x_t + b_f)$
- Input gate
  - $i_t = \sigma(w_{ih}h_{t-1} + w_{ix}x_t + b_i)$
- Output gate
  - $o_t = \sigma(w_{oh}h_{t-1} + w_{ox}x_t + b_o)$



Long Short-Term Memory

- Candidate cell state
  - $\tilde{c}_t = \tanh(w_{hh}h_{t-1} + w_{hx}x_t + b_h)$
- Update cell state
  - $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$
- Output state
  - $h_t = o_t * \tanh(c_t)$

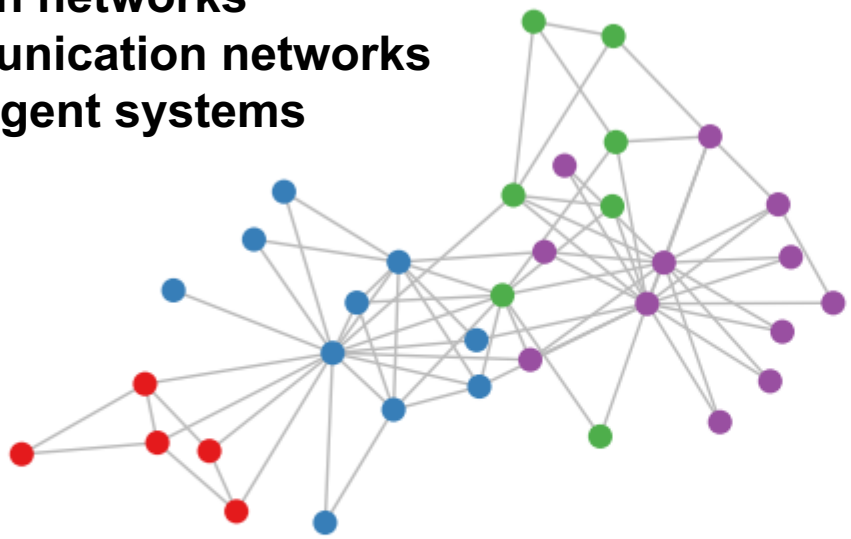


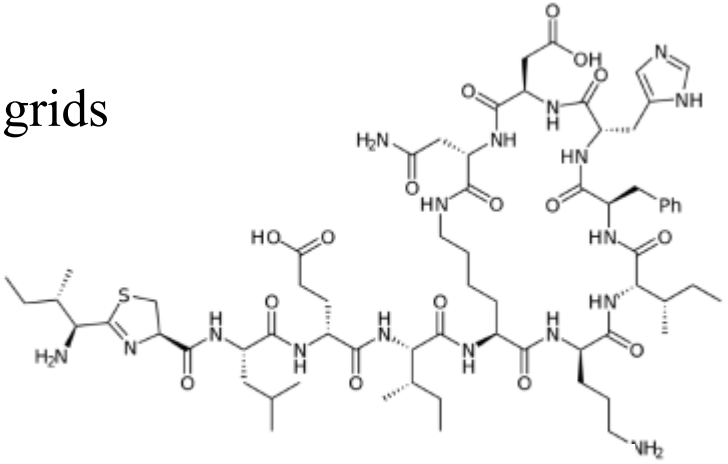
Graph Neural Networks

Background: Graph data

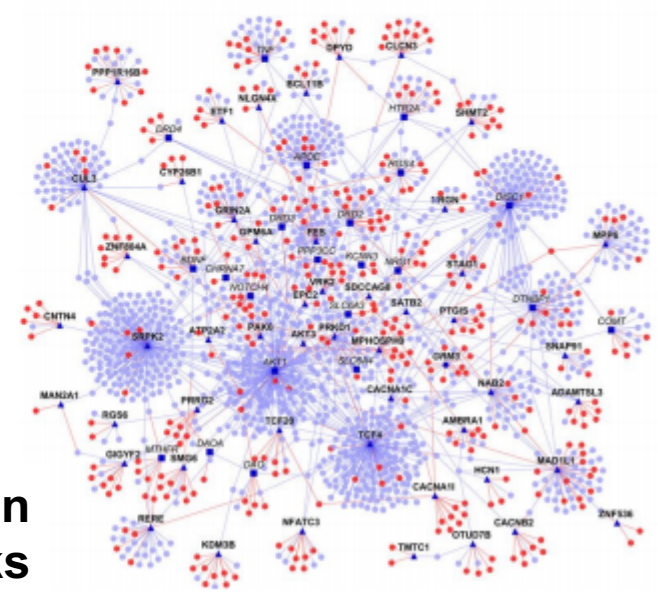
A lot of real-world data does not 'live' on grids

Social networks
Citation networks
Communication networks
Multi-agent systems





Molecules

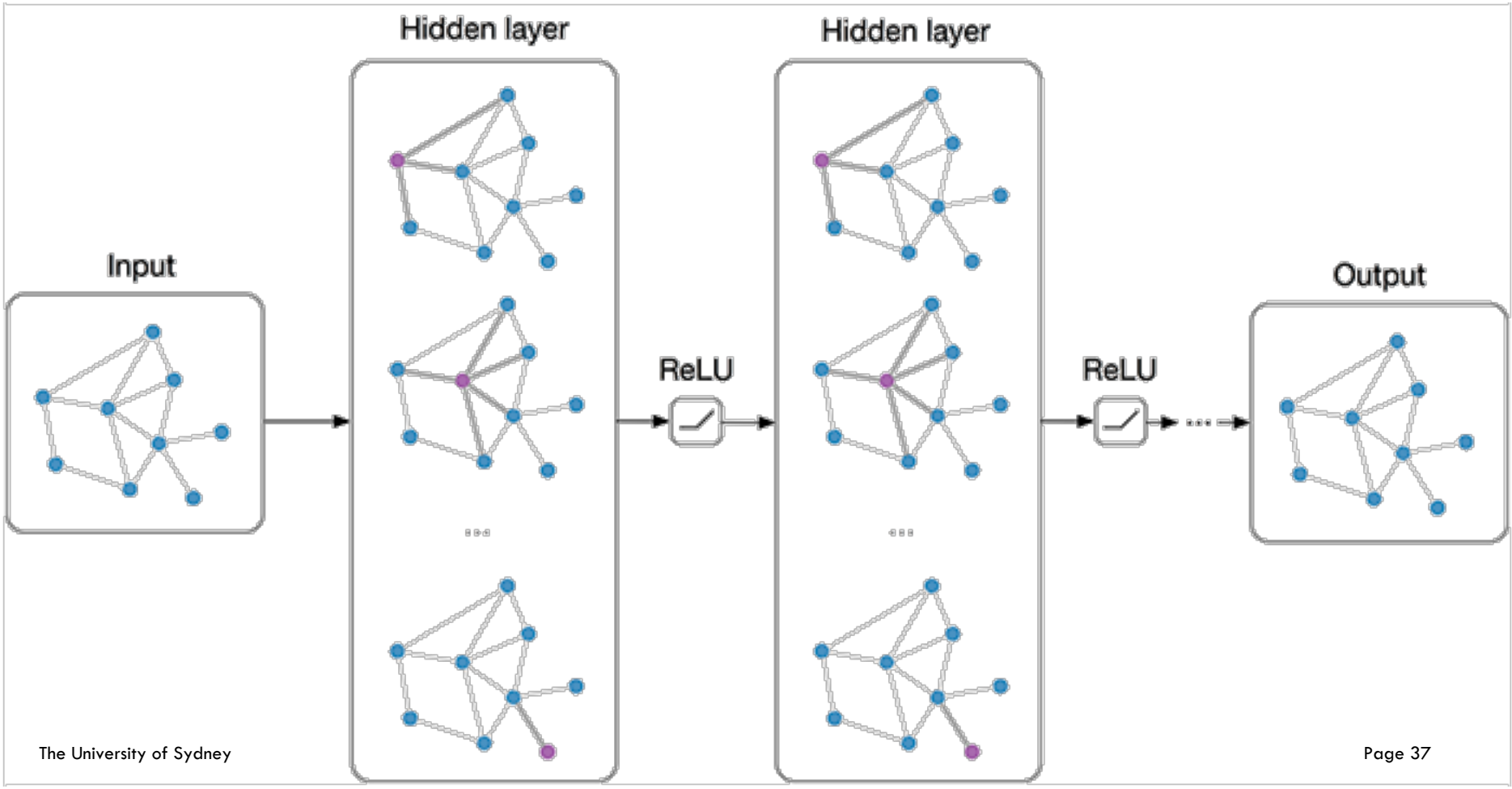


Protein interaction networks

Standard deep learning architectures like CNNs and RNNs don't work here!

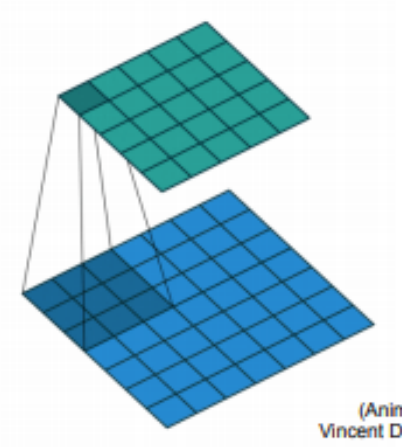
Graph Convolutional Networks (GCN)

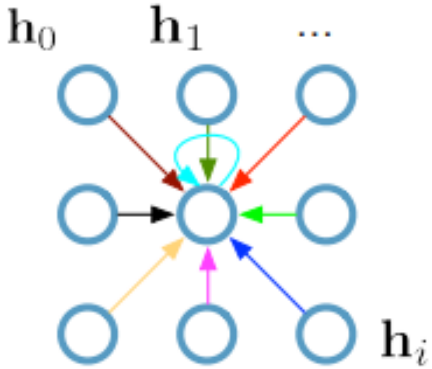
Extract features from the graph data.



Convolution in CNN

Single CNN layer with 3x3 filter:





Update for a single pixel:

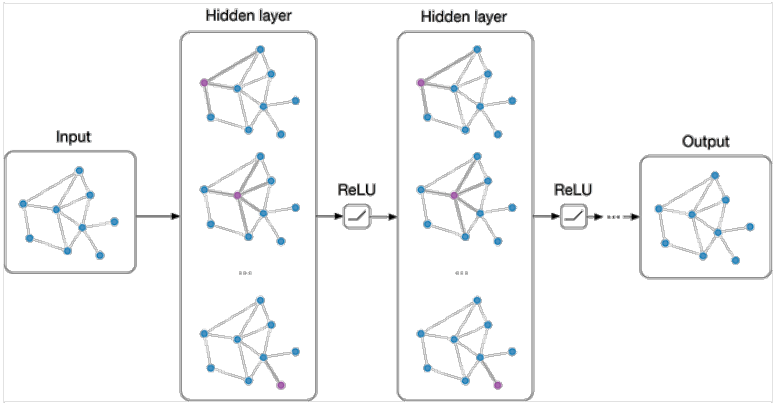
- Transform messages individually $W_i h_i$
- Add everything up $\sum_i W_i h_i$

$h_i \in R^F$ are (hidden layer) activations of a pixel/node

Full update:

$$h_4^{(l+1)} = \sigma(W_0^{(l)} h_0^{(l)} + W_1^{(l)} h_1^{(l)} + \cdots + W_8^{(l)} h_8^{(l)})$$

Convolution in GCN



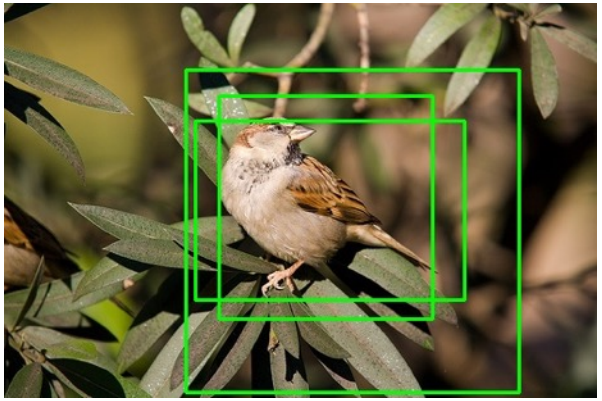
$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{l}W^{l})$$

- Adjacency matrix with self-connections: $\hat{A} = A + I_N$
- Degree: number of edges connected to a node
- Degree matrix: $\hat{D} \in R^{N \times N}$ (computed from $\hat{A}$)

Deep Learning Applications

Typical architecture

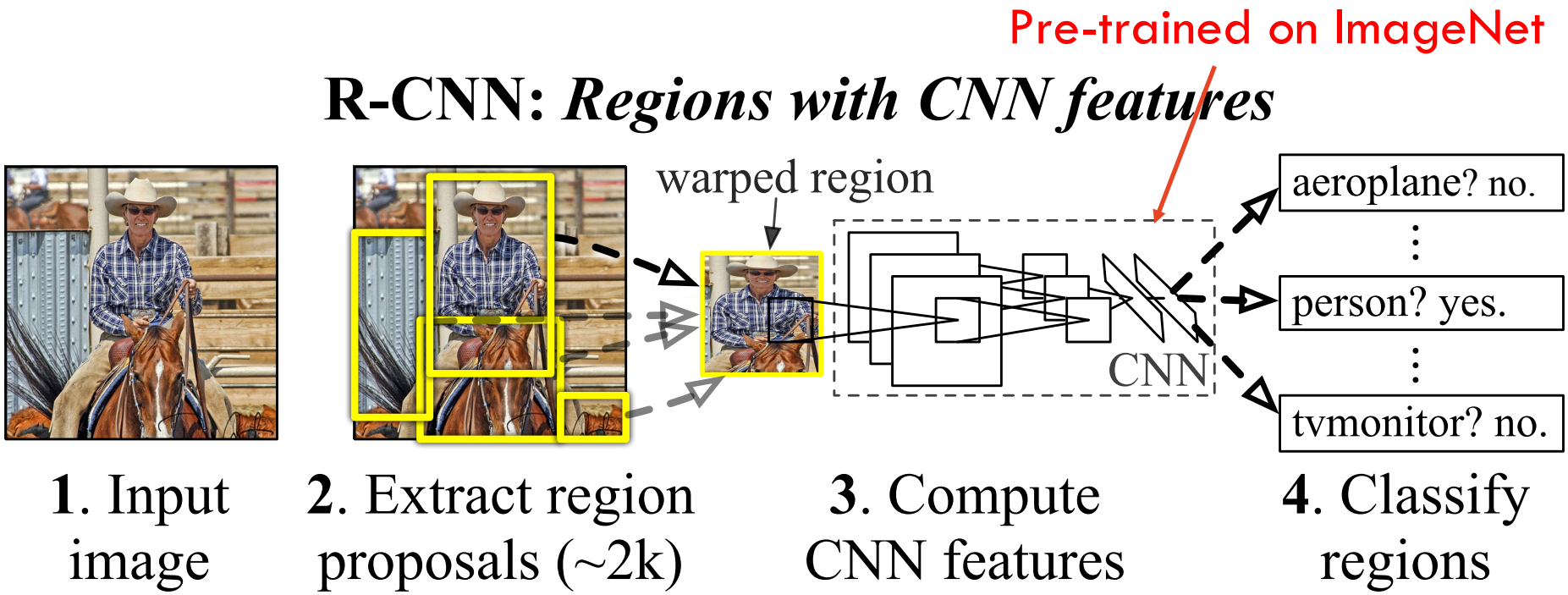
Proposals



1.Region proposal: Given an input image find all possible places where objects can be located. The output of this stage should be a list of bounding boxes of likely positions of objects. These are often called region proposals or regions of interest.

2.Final classification: for every region proposal from the previous stage, decide whether it belongs to one of the target classes or to the background. Here we could use a deep convolutional network.

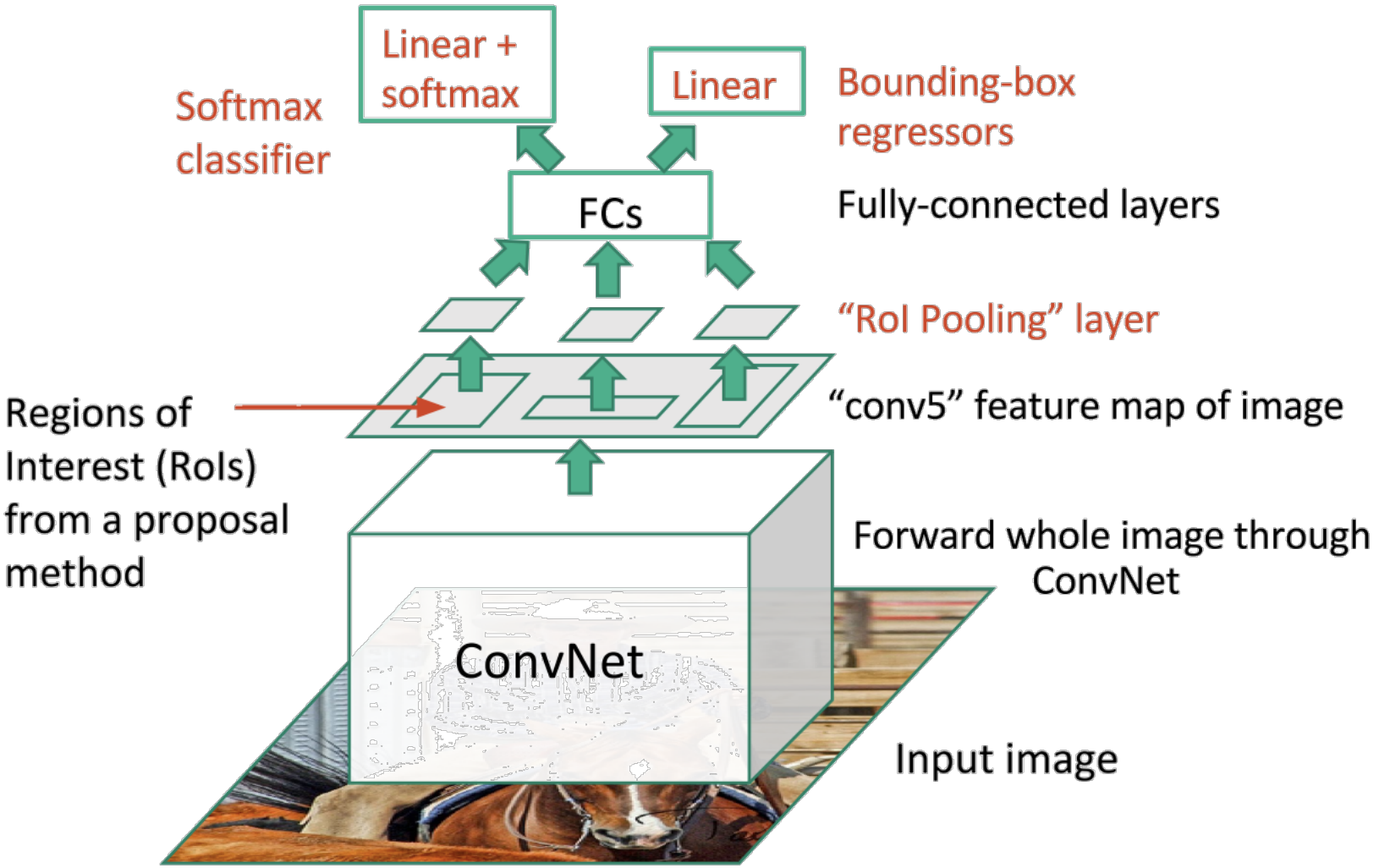
Region CNN (R-CNN)



Object Detection to Image Classification

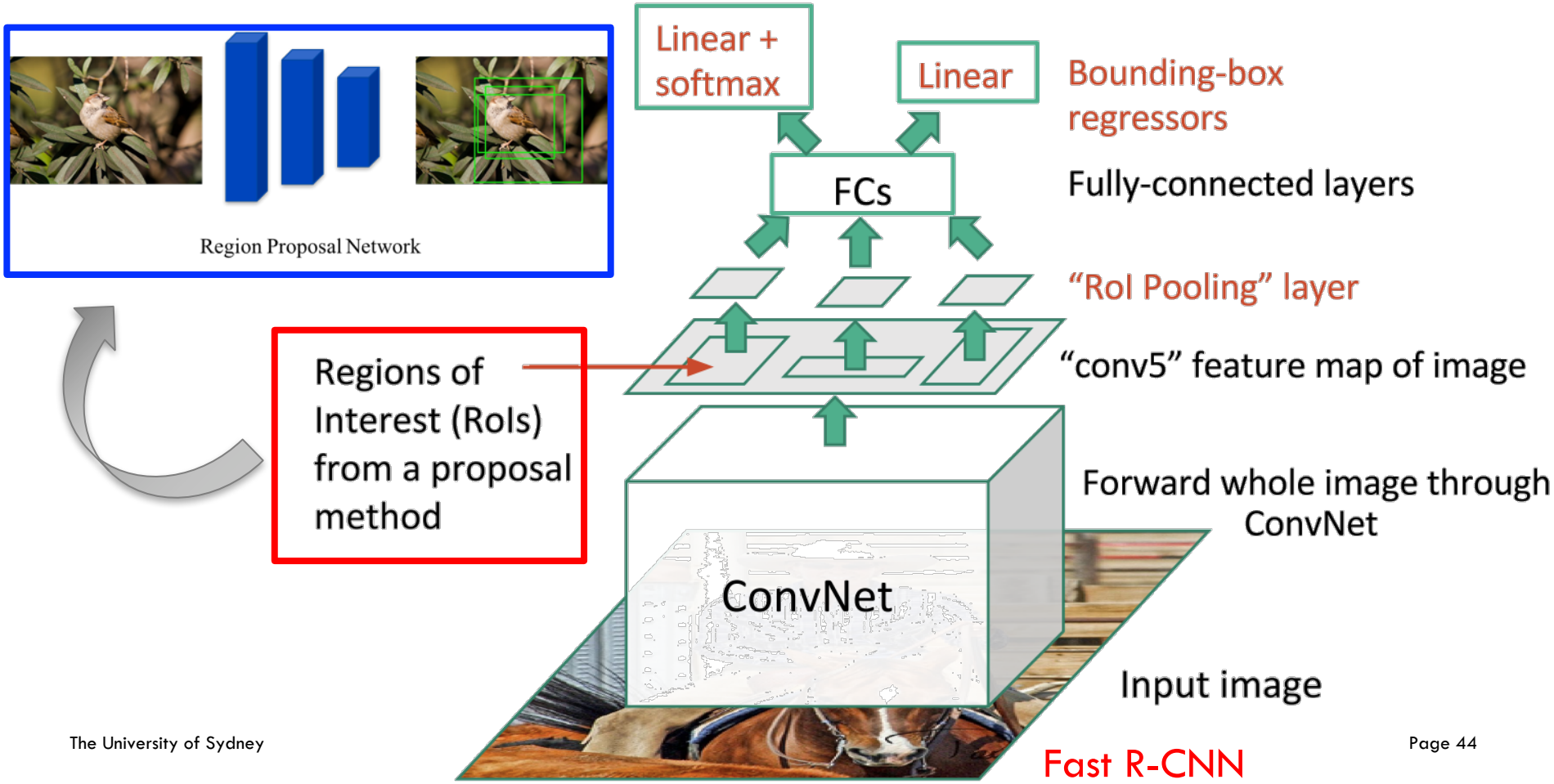
[Girshick14] R. Girshick, J. Donahue, S. Guadarrama, T. Darrell, J. Malik: **Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation**, CVPR 2014

Fast R-CNN


Linear + softmax
Linear
Softmax classifier
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Faster R-CNN

Replace the slow selective search algorithm with a fast neural net - region proposal network (RPN).



RoIAlign

Realigning RoI Pooling to be More Accurate.

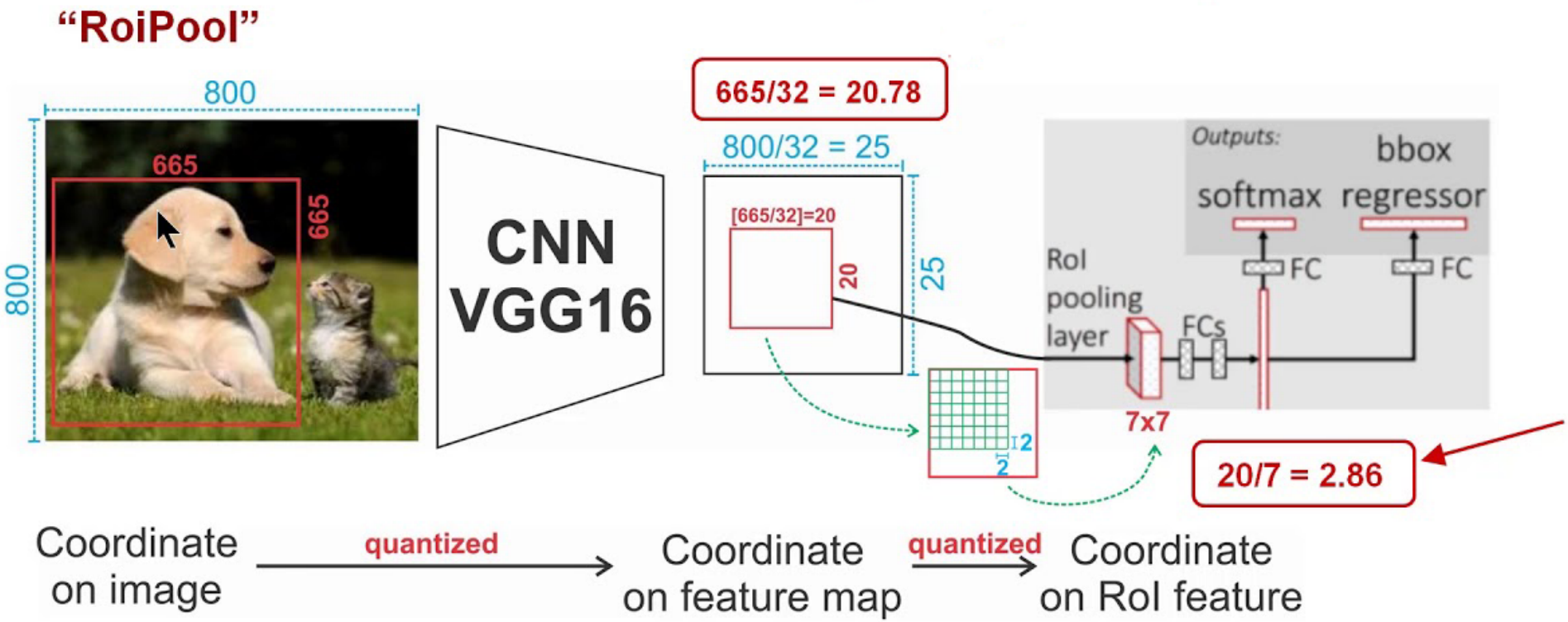


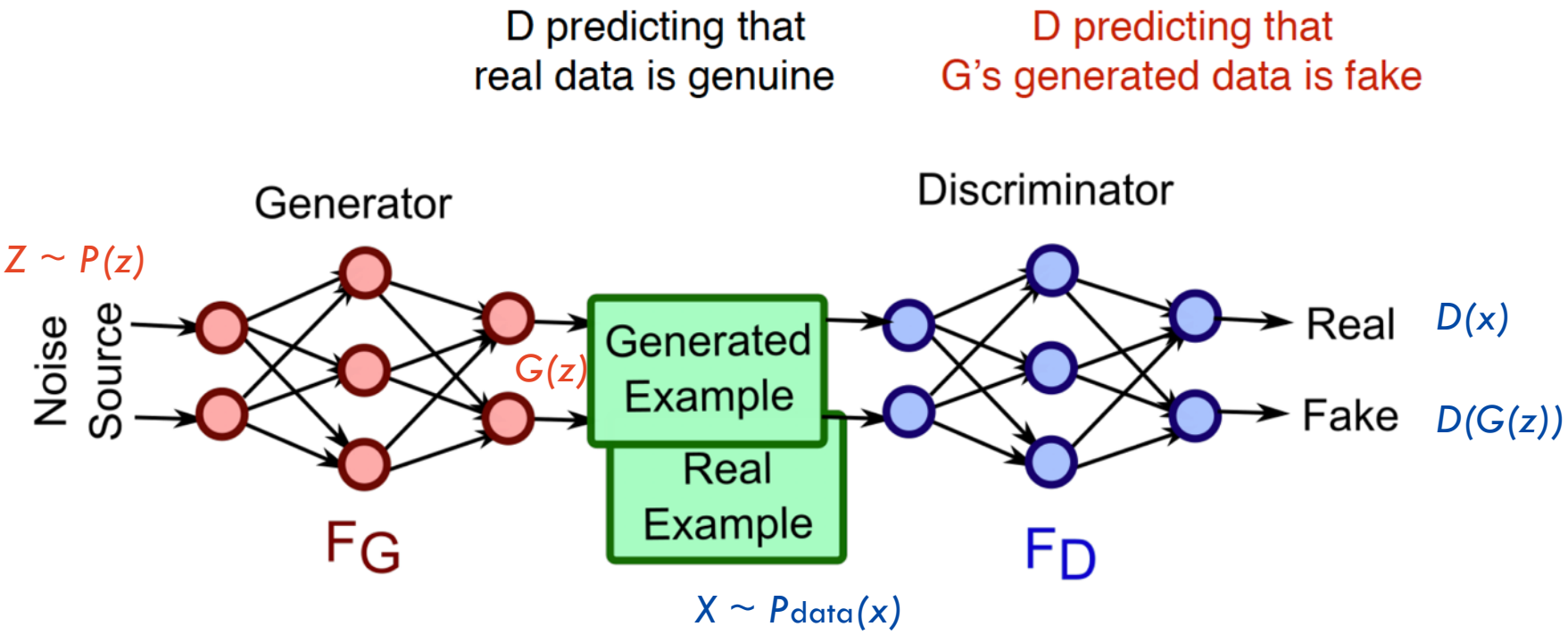
Image credit to Ardian Umam

Deep Generative Models

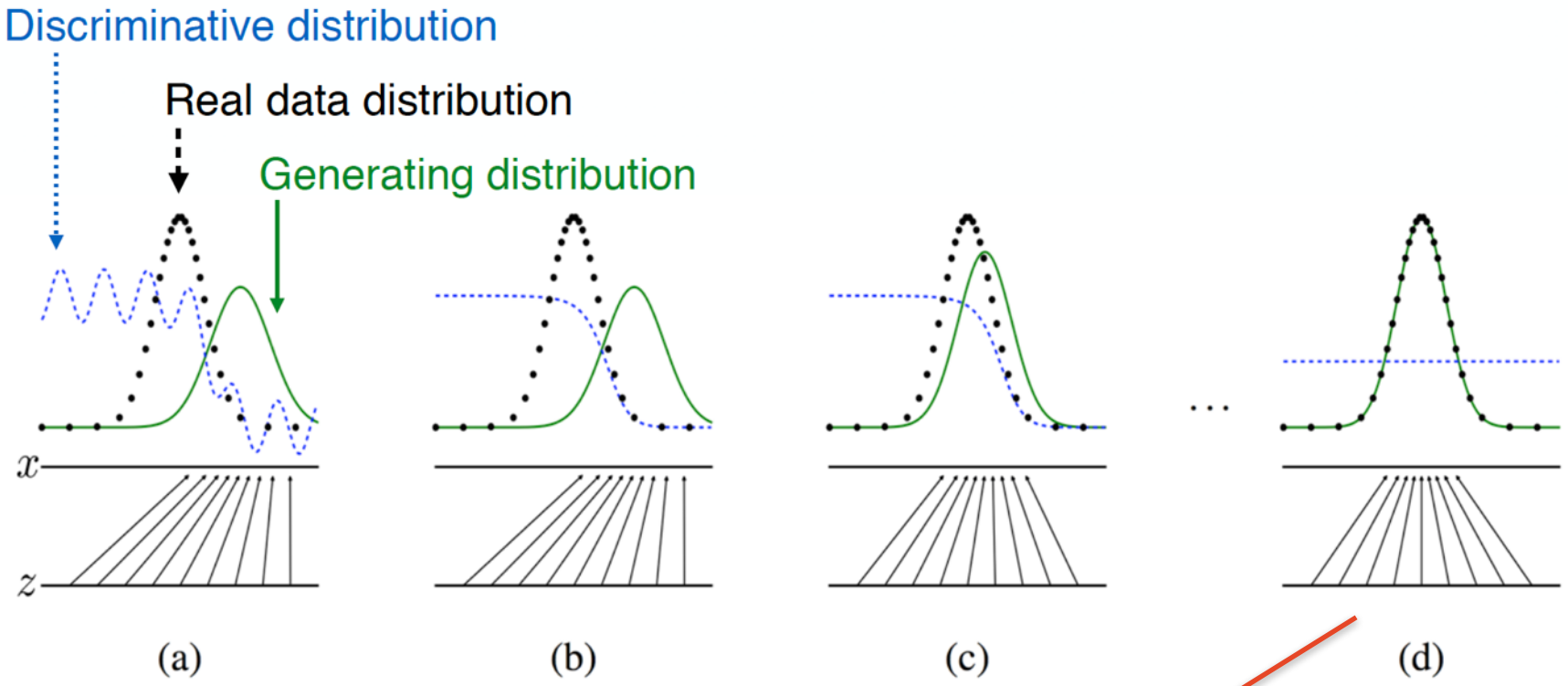
Generative Adversarial Networks

- A two-player game between two components: a generator **G** and a discriminator **D**

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$



A simple example



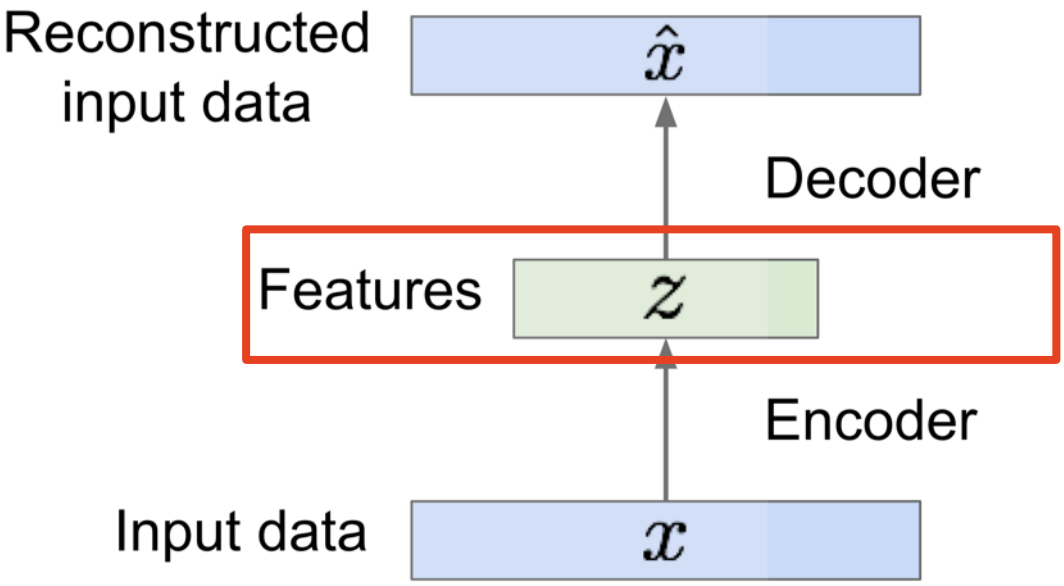
After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{data}$. The discriminator is unable to differentiate between the two distributions, i.e. **D(x) = 1/2** .

Image credit to [Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2016]

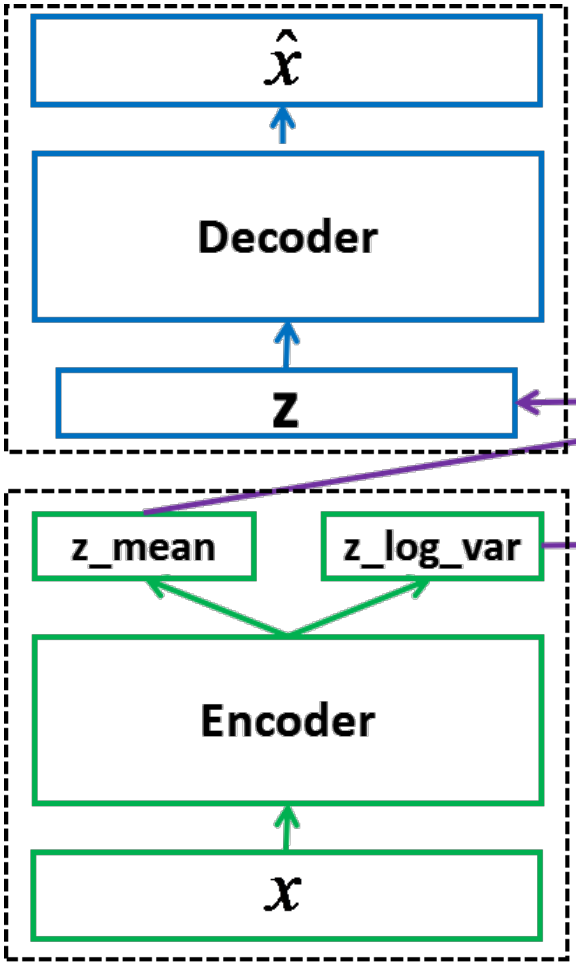
Recap: Autoencoder

Autoencoders can reconstruct data, and can learn features to initialize a supervised model. Features capture factors of variation in training data.

Can we generate new images from an autoencoder?



Variational Autoencoders



Cross Entropy

$$\sum_{i}^{n} -[x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \hat{x}_i)]$$

MSE $$\sum_{i}^{n} (x_i - \hat{x}_i)^2$$

KL Divergence

$$-0.5(1 + \log \sigma^2 - \mu^2 - \sigma^2)$$

Min CrossEntropy + KLDivergence

Final exam for COMP5329

Exam date: 15/06/20

Start time: 13:00 (Sydney time)

EXAM WRITING TIME: 2 hours

READING TIME: 10 minutes. During reading time - writing is not permitted at all

EXAM CONDITIONS: This is an OPEN book examination

MATERIALS PERMITTED IN THE EXAM VENUE:

- Calculator - non-programmable
- Hard copies of reference material
- Blank scratch paper

MATERIALS NOT PERMITTED: No mobile phones and software that are unrelated to the exam.

- This exam has 11 questions:
  - 4 choice questions (see midterm quiz)
  - 7 essay questions (see samples on Ed)

Essay questions may involve programing.

Important advice from the Faculty:

- In the interests of equity there will be **no communications between students and coordinators or other teaching staff while an exam of any type** is going on.

- If a student is uncertain about an exam question they are instructed to answer the question to the best of their ability and email you after the exam with relevant information.

Contact TA if you really really need help during the exam …
Shumin Kong <shumin.kong@sydney.edu.au> 0420 598 237

No zoom and proctorU during the exam.

But you need to be aware of the academic honesty.

Integrity Contract

Online exams/tests (supervised/unsupervised)

Academic honesty

By enrolling at the University of Sydney, you have committed to acting honestly and ethically in all your dealings with the University and its community in accordance with the Academic Honesty in Coursework Policy 2015. The University is strongly opposed to and will not tolerate academic dishonesty or plagiarism and will treat all allegations of dishonesty seriously.

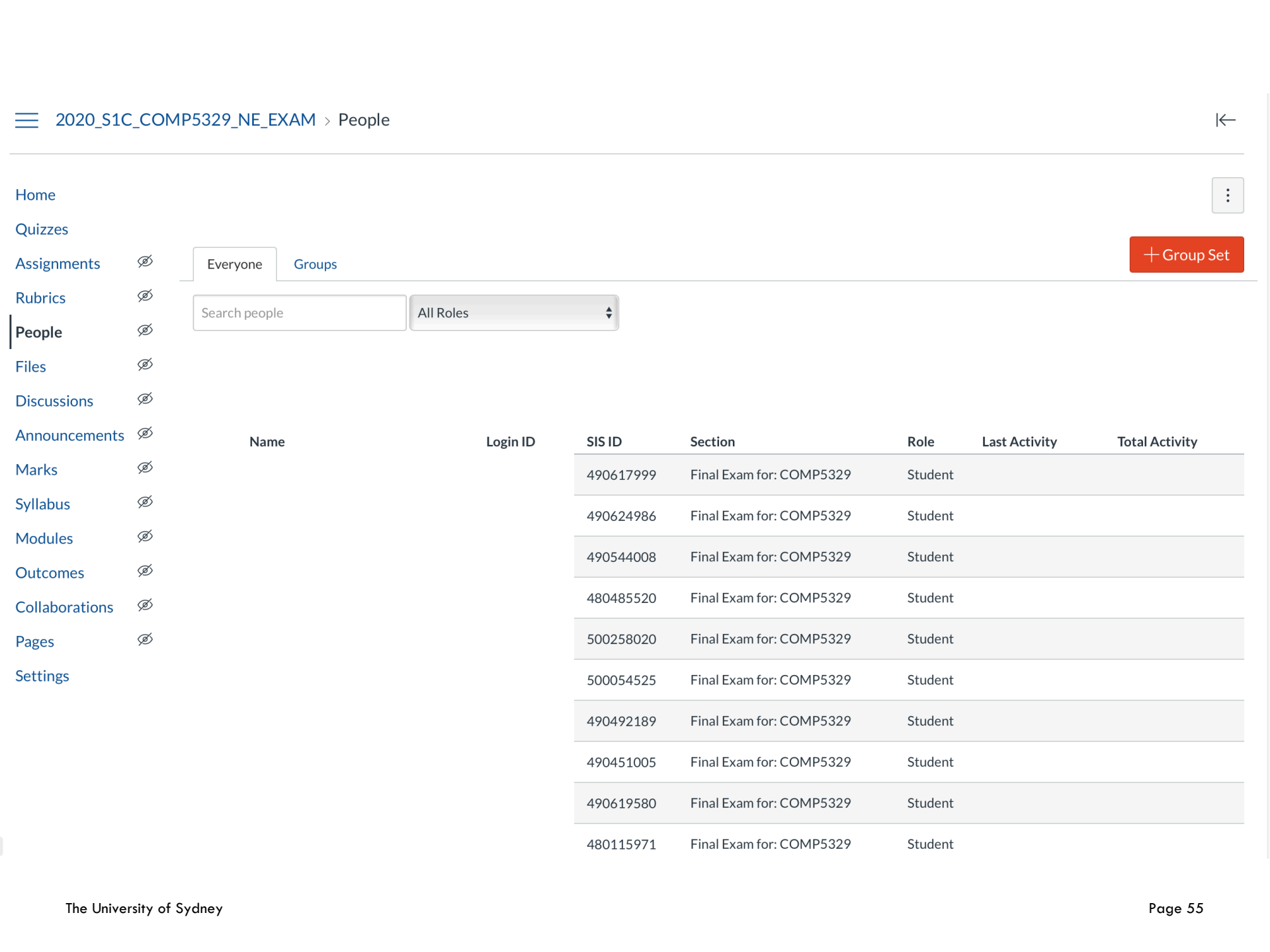
Compliance statement

In submitting this work, I acknowledge the following:

- I will act honestly and ethically as required under the Academic Honesty in Coursework Policy 2015.
- I have carefully read and understand the conditions under which this examination is to be completed, including the specified time limit, prohibited materials and the individual nature of the assessment.
- I am aware that I am not permitted to use any personal communication devices, which includes wired and wireless headphones with or without an integrated microphone.
- I am aware that I am prohibited from communicating with another person by any means during the examination, including other students or persons sharing my accommodation.
- Any responses I submit for short or extended response questions are substantially my own work, and where any parts of these responses are not my own, I have indicated this by enclosing any quoted text in quotation marks and acknowledging the source of the text.
- Any responses I submit may be submitted to similarity detection software (Turnitin) and a copy of the work will be retained in Turnitin's paper repository for future similarity checking.
- I am aware that engaging in academic dishonesty or plagiarism will, if detected, lead to the University commencing proceedings against me under the Academic Honesty in Coursework Policy 2015 and the Academic Honesty Procedures 2016.
- I am aware that having another person complete any part of this examination on my behalf will, if detected, lead to the University commencing proceedings against me for potentially serious student misconduct under the University of Sydney (Student Discipline) Rule 2016.

This exam is still timed, and you need to treat it as if it is a normal exam, and follow these steps to prepare:

1. **Read through the exam instructions on Canvas.**
   Exams will be hosted in separate Canvas sites identified by unit code. Please read through this site for detailed information about your exam once you have been added in Canvas (no later than **29th May**).
2. **Prepare your computer.**
   Restart your computer and close down any unnecessary programs.
3. **Prepare your exam space.**
   The room you take the exam in should be like an exam room. Clear your desk, and make sure the area is as tidy as possible.
   You should also prepare any extra materials which are allowed in your exam, such as a textbook or calculator. The usual rules apply here, you can only use approved calculators and dictionaries . If you are registered with Disability Services and have an academic plan, please carefully read through the advice on what additional resources you can have with you during your exam. If you are not yet registered with Disability Services and require an academic plan or adjustments for your online exam due to a pre-existing condition, please contact Disability Services ASAP.
4. **Remind your housemates or family not to interrupt you.**
   You should also remind your friends, family, or flatmates that you will need to have good bandwidth while you take your exam. They should stay off the internet and avoid streaming movies or tv or playing games.


2020_S1C_COMP5329_NE_EXAM
People
Home
Quizzes
Assignments
Rubrics
People
Files
Discussions
Announcements
Marks
Syllabus
Modules
Outcomes
Collaborations
Pages
Settings
Everyone
Groups
+ Group Set
Search people
All Roles
Name
Login ID
SIS ID
Section
Role
Last Activity
Total Activity
490617999 Final Exam for: COMP5329 Student
490624986 Final Exam for: COMP5329 Student
490544008 Final Exam for: COMP5329 Student
480485520 Final Exam for: COMP5329 Student
500258020 Final Exam for: COMP5329 Student
500054525 Final Exam for: COMP5329 Student
490492189 Final Exam for: COMP5329 Student
490451005 Final Exam for: COMP5329 Student
490619580 Final Exam for: COMP5329 Student
480115971 Final Exam for: COMP5329 Student

Thanks!