Deep Learning (CS324)

11. Beyond vectors: DL on graphs*

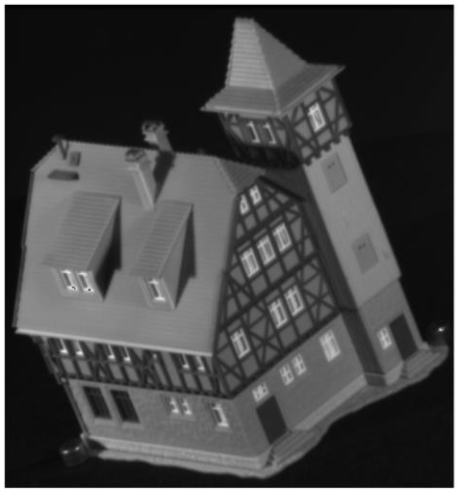
Jianguo Zhang

SUSTech

*This lecture is mostly based on this article and the references at the end of the slides
Kipf & Welling (ICLR 2017), Semi-Supervised Classification with Graph Convolutional Networks

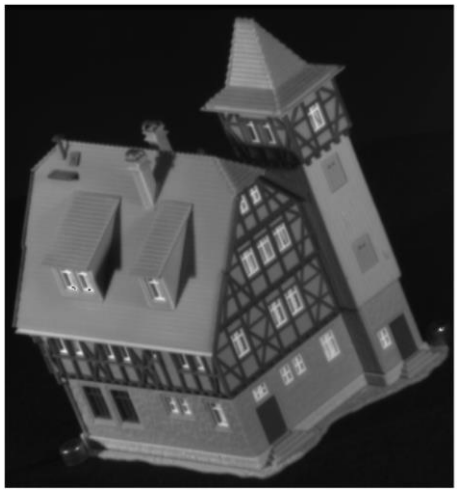
Features vs Structure

- Object as set of extracted **features**
- Object as components in **relation** with each other



Features vs Structure

- Object as set of extracted **features**
- Object as components in **relation** with each other

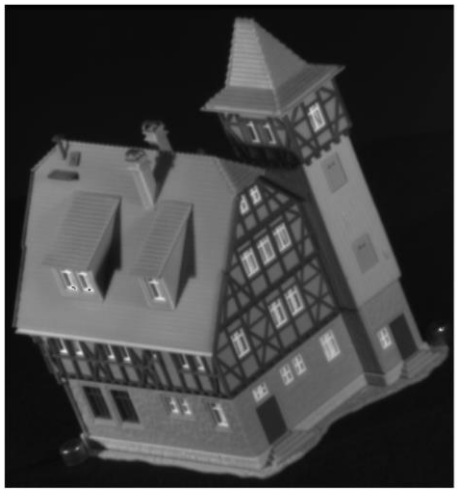


$[f_1, f_2, f_3, \ldots, f_n]$

Features vs Structure

- Object as set of extracted **features**

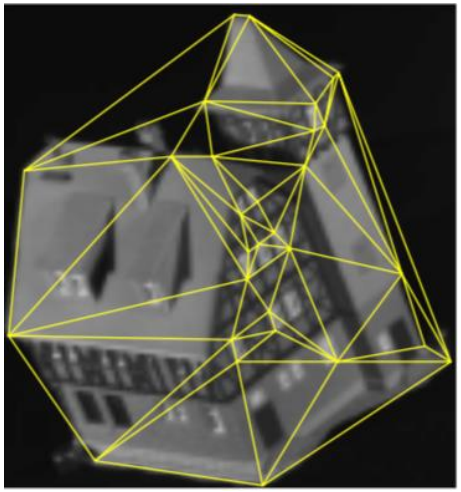
- Object as components in **relation** with each other



Features vs Structure

- Object as set of extracted **features**

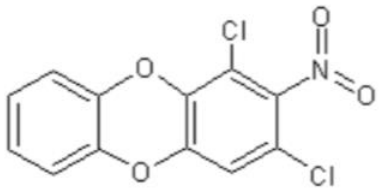
- Object as components in **relation** with each other



Features vs Structure

- Object as set of extracted **features**

- Object as components in **relation** with each other



Toxic or not toxic?

Features vs Structure

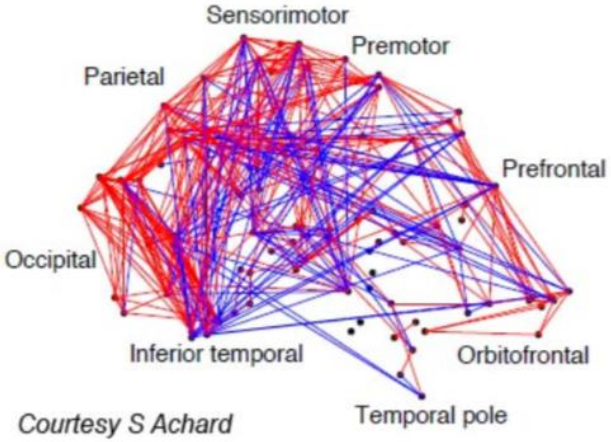
- Object as set of extracted **features**

- Object as components in **relation** with each other

Features vs Structure

- Object as set of extracted **features**

- Object as components in **relation** with each other

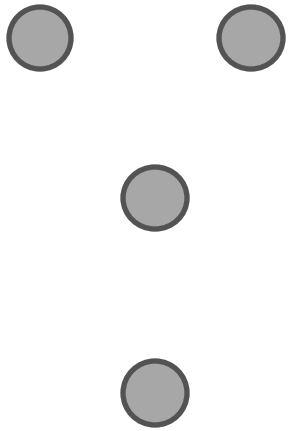


Graphs

- A graph consists of

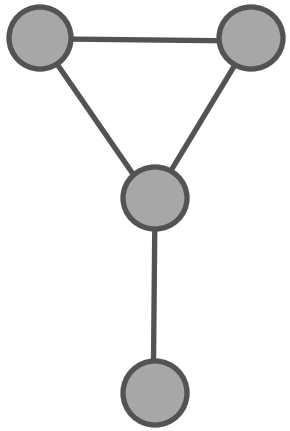
Graphs

- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)



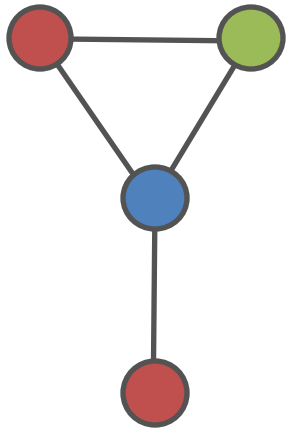
Graphs

- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)



Graphs

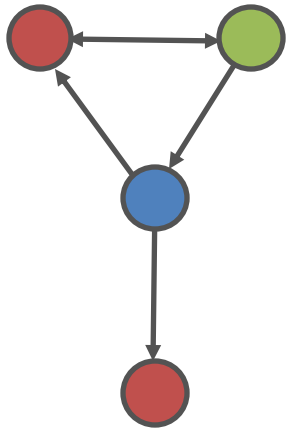
- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)



Vertices can be labelled

Graphs

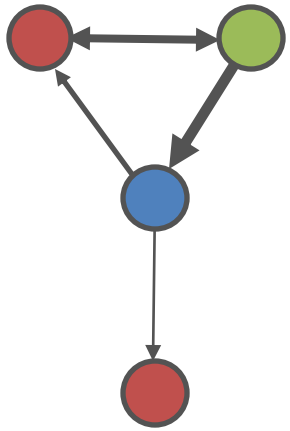
- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)



Edges can be directed

Graphs

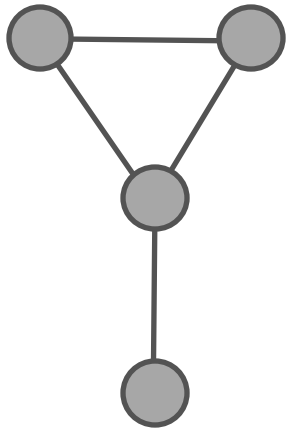
- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)



Edges can be weighted

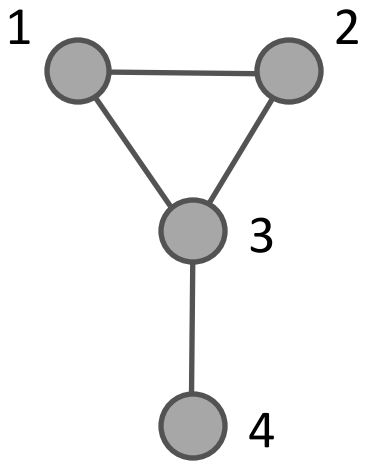
Graphs

- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)



Graphs

- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)

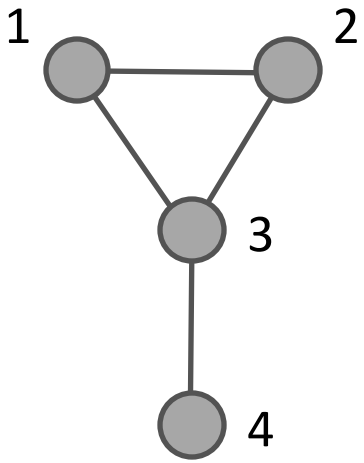


Graphs

- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Degree matrix



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

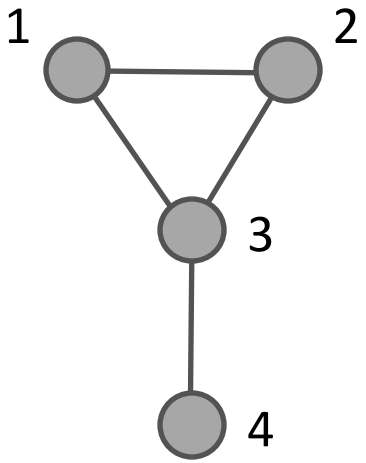
Adjacency matrix

Graphs

- A graph consists of
  - A set of vertices **V** (e.g., parts of an object)
  - A set of edges **E** (e.g., relations between parts)

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Degree matrix



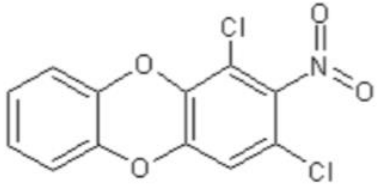
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency matrix

Laplacian matrix $L = D - A$

Learning on graphs

- How to apply deep learning (or even just standard machine learning) to graph data?
- You have a dataset of graphs representing chemical compounds, with label toxic/not-toxic. How to train an algorithm to distinguish them?



Toxic or not toxic?

Learning on graphs

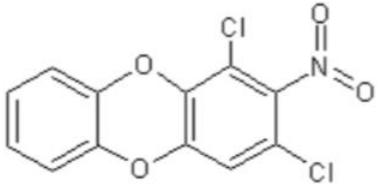
- How to apply deep learning (or even just standard machine learning) to graph data?
- You have a dataset of graphs representing chemical compounds, with label toxic/not-toxic. How to train an algorithm to distinguish them?
- ML/DL algorithms work on vectorial spaces

Learning on graphs

- How to apply deep learning (or even just standard machine learning) to graph data?
- You have a dataset of graphs representing chemical compounds, with label toxic/not-toxic. How to train an algorithm to distinguish them?
- ML/DL algorithms work on vectorial spaces
  - i.e., the input is a matrix of data points

Learning on graphs

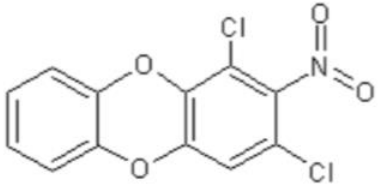
- How to apply deep learning (or even just standard machine learning) to graph data?
- You have a dataset of graphs representing chemical compounds, with label toxic/not-toxic. How to train an algorithm to distinguish them?
- ML/DL algorithms work on vectorial spaces
  - i.e., the input is a matrix of data points

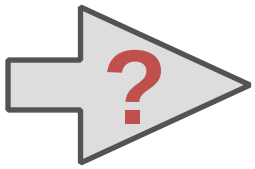


Toxic or not toxic?

Learning on graphs

- How to apply deep learning (or even just standard machine learning) to graph data?
- You have a dataset of graphs representing chemical compounds, with label toxic/not-toxic. How to train an algorithm to distinguish them?
- ML/DL algorithms work on vectorial spaces
  - i.e., the input is a matrix of data points

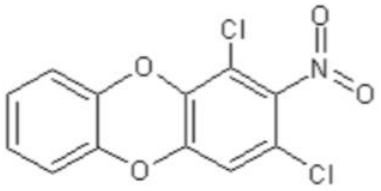




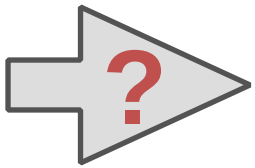
Toxic or not toxic?

Learning on graphs

- How to apply deep learning (or even just standard machine learning) to graph data?
- You have a dataset of graphs representing chemical compounds, with label toxic/not-toxic. How to train an algorithm to distinguish them?
- ML/DL algorithms work on vectorial spaces
  - i.e., the input is a matrix of data points



Toxic or not toxic?



…
[-2, 0, 0, 1.2, …]
[0, 1, -2, 0.5, …]
[-1.2, 1, 0, 0, …]
…

Learning on graphs

- Turning a graph into a vector (i.e., embedding a graph into a vectorial space) it's not an easy task
  - What's the *right* order of the nodes? If I change the order I change the point in the embedding space, e.g., [1 0 2] is different from [2 1 0]
  - Graphs describing objects from the same class can vary in size, so the dimension of the corresponding vectors would change as well. How do I compare [1 0 2] to [1 0 1 1]?

Learning on graphs

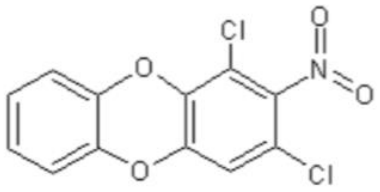
- Luckily researchers in the field of machine learning and deep learning have come up with a variety of ways to apply machine/deep learning to graph by either computing the embedding directly, or indirectly
- In this lecture, we only focus on how CNNs can be generalised to work on graphs

Learning on graphs

- Luckily researchers in the field of machine learning and deep learning have come up with a variety of ways to apply machine/deep learning to graph by either computing the embedding directly, or indirectly
- In this lecture, we only focus on how CNNs can be generalised to work on graphs
- **Note: is a very new and rapidly evolving field, my intention is just to offer you a glimpse of it**

Learning on graphs

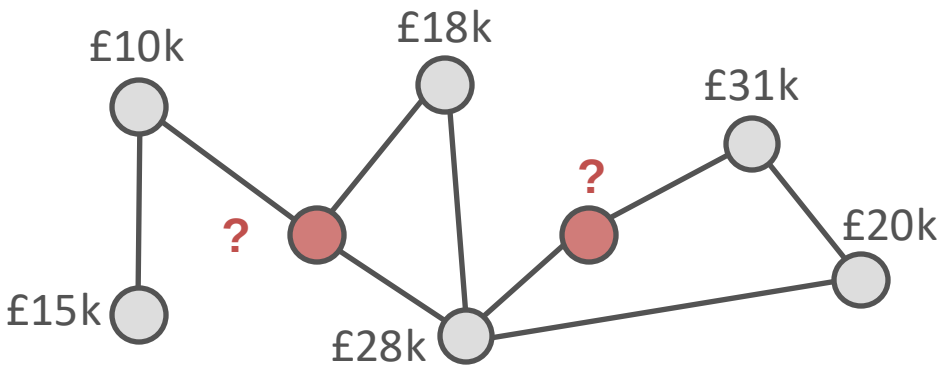
- The type of tasks we're interested in solving when dealing with graphs are typically:
  - Graph classification
    - Just like what you're used to, but instead of assigning a label to a point you assign it to a graph



Toxic or not toxic?

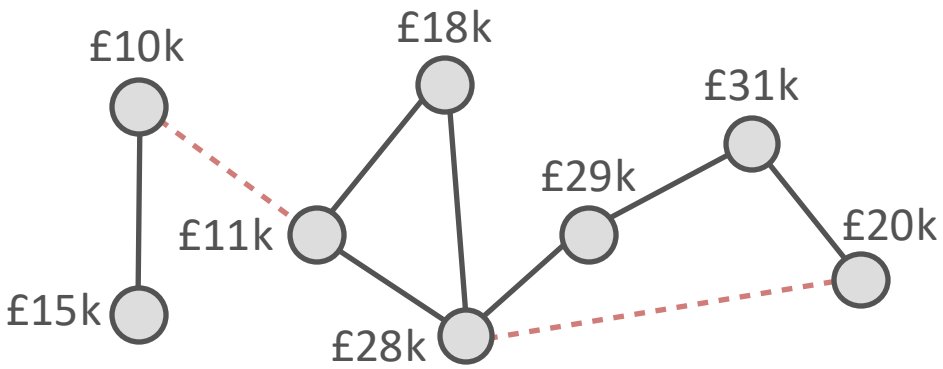
Learning on graphs

- The type of tasks we're interested in solving when dealing with graphs are typically:
  - Node classification
    - You assume some nodes have labels (e.g., age, sex, income, etc.) other don't, and you want to find out the missing information

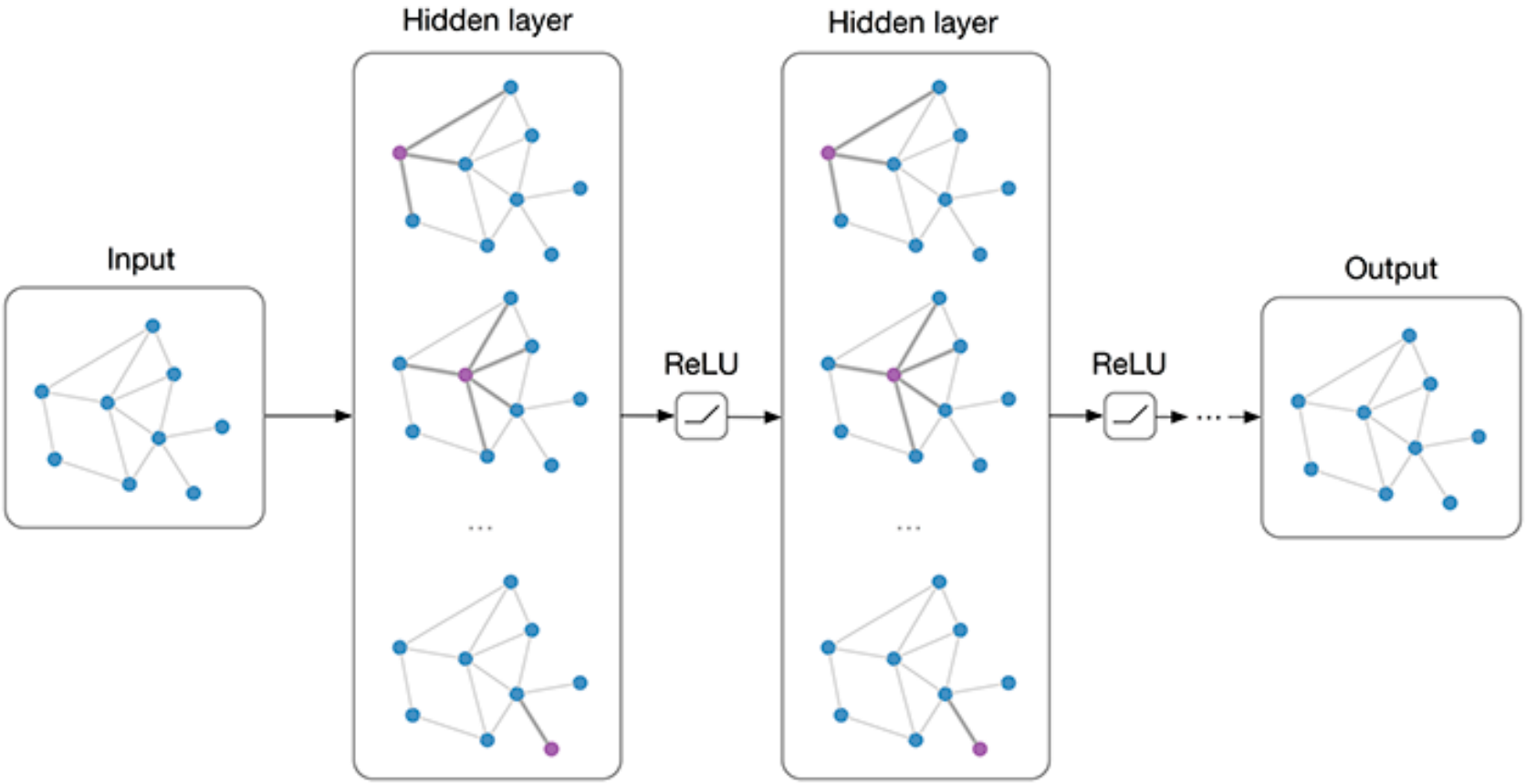


Learning on graphs

- The type of tasks we're interested in solving when dealing with graphs are typically:
  - Link prediction
    - Some links in your graphs are missing and your task is to find what they are



Graph Convolutional Networks



Source: https://tkipf.github.io/graph-convolutional-networks/

Graph Convolutional Networks

- Assume each node *i* of the graph G with adjacency matrix A has an associated feature X_i
- The **input** of the GCN is the matrix of node features X and the adjacency matrix A
- The **output** is either a matrix Z associating a scalar/vector to each node of G or combine them in some way* to obtain a single label for G

*See for example this paper

Graph Convolutional Networks

- Assume each node *i* of the graph G with adjacency matrix A has an associated feature x_i
- The **input** of the GCN is the matrix of node features X and the adjacency matrix A
- The **output** is either a matrix Z associating a scalar/vector to each node of G or combine them in some way* to obtain a single label for G

Graph Convolutional Networks

- The layers of the network transform their input according to a non-linear transformation

$$H^{(l+1)} = f(H^{(l)}, A)$$

Graph Convolutional Networks

- The layers of the network transform their input according to a non-linear transformation

$$H^{(l+1)} = f(H^{(l)}, A)$$

$H^{(0)}$ is the feature matrix X

Graph Convolutional Networks

- The layers of the network transform their input according to a non-linear transformation

$$H^{(l+1)} = f(H^{(l)}, A)$$

$H^{(L)}$ is the output matrix Z (the last layer)

Graph Convolutional Networks

- The layers of the network transform their input according to a non-linear transformation

$$H^{(l+1)} = f(H^{(l)}, A)$$

Different GCN models vary in the choice and parametrisation of *f*

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

Parameters of the l-th layer

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

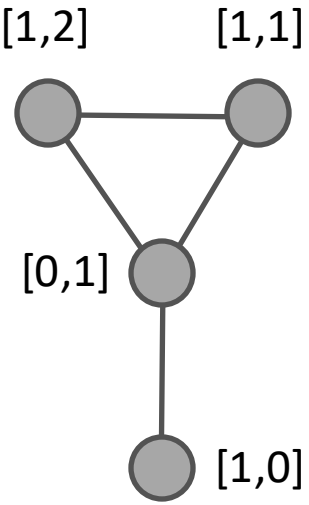
What's happening here? Remember, for l=0 that's a multiplication between the adjacency matrix and the node-feature matrix

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

What's happening here? Remember, for l=0 that's a multiplication between the adjacency matrix and the node-feature matrix



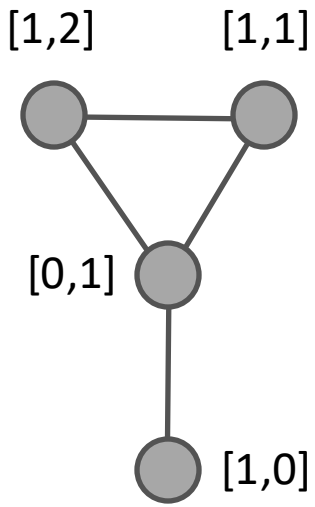
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

Multiplying A and H computes a new feature for the node *i* which is the sum of all the features of its neighbours



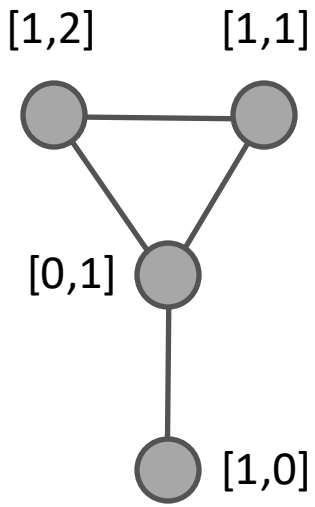
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

Multiplying A and H computes a new feature for the node *i* which is the sum of all the features of its neighbours



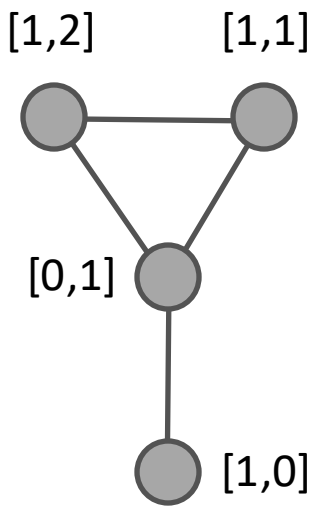
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

So at each layer for each node we compute a new feature vector which is a non-linear function of its neighbours' features, and so on. The features are propagated.



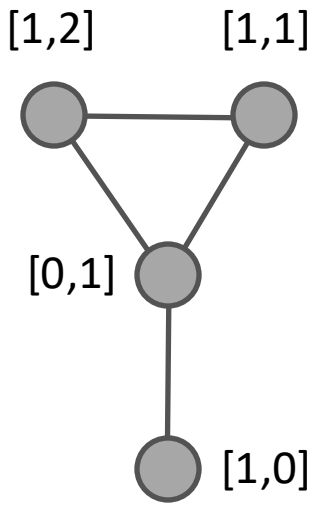
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

- The problems of the above formulation:
  - The feature at node i is NOT included in the computation of the new aggregated feature at node i.



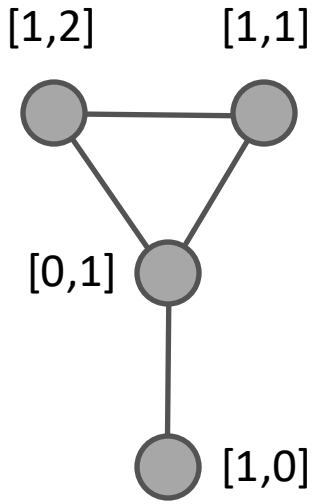
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

- The problem of the above formulation:
  - Different node has different degrees, i.e, has different number of edges; node with large degrees might have large representations, and vice versa. This might cause explosion or vanishing problem.



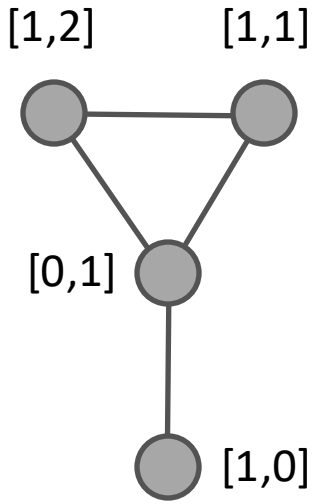
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

- We modify this in two ways:
  - Add a loop to each node, so the old feature is included in the computation of the new feature



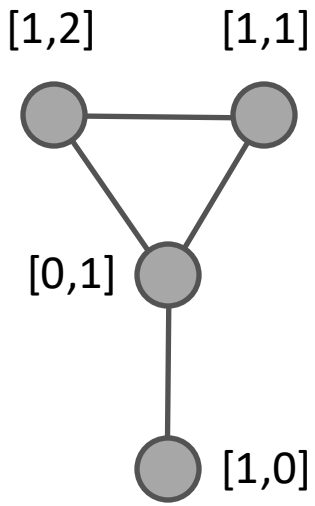
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

- We modify this in two ways:
  - Normalise A so the scale of the feature vectors doesn't change, i.e.,



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$H^{(l+1)} = \mathrm{Re\,LU}\left(\hat{D}^{-1}\hat{A}H^{l}W^{l}\right)$$

GCN: a simple example

- A simple network could be defined as follows

$$H^{(l+1)} = \mathrm{ReLU}(AH^{(l)}W^{(l)})$$

- We modify this:
  - Normalise A so the scale of the feature vectors doesn't change.

$$H^{(l+1)} = \mathrm{Re}\,LU\left(\hat{D}^{-1}\hat{A}H^{l}W^{l}\right)$$

  - When computing the aggregate feature representation of the ith node, we not only take into consideration the degree of the ith node, but also the degree of the jth node. We could have the following formula

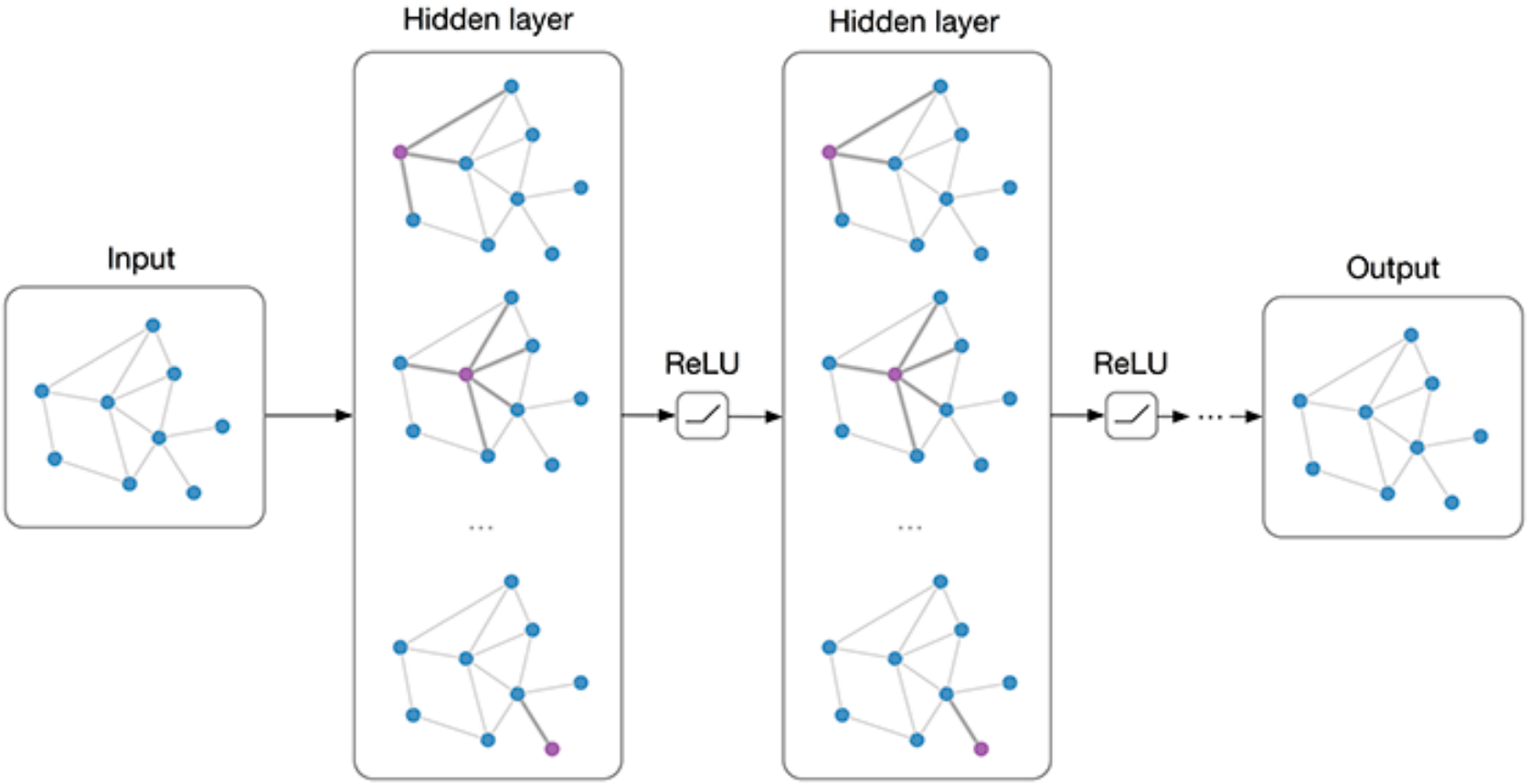
$$H^{(l+1)} = \mathrm{ReLU}(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)})$$

Source: https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-62acf5b143d0

Source: https://arxiv.org/pdf/1609.02907.pdf

$$H^{(l+1)} = \mathrm{ReLU}(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)})$$

Graph Convolutional Networks
Receptive Field and Feature Depth



Source: https://tkipf.github.io/graph-convolutional-networks/

GCN for node classification

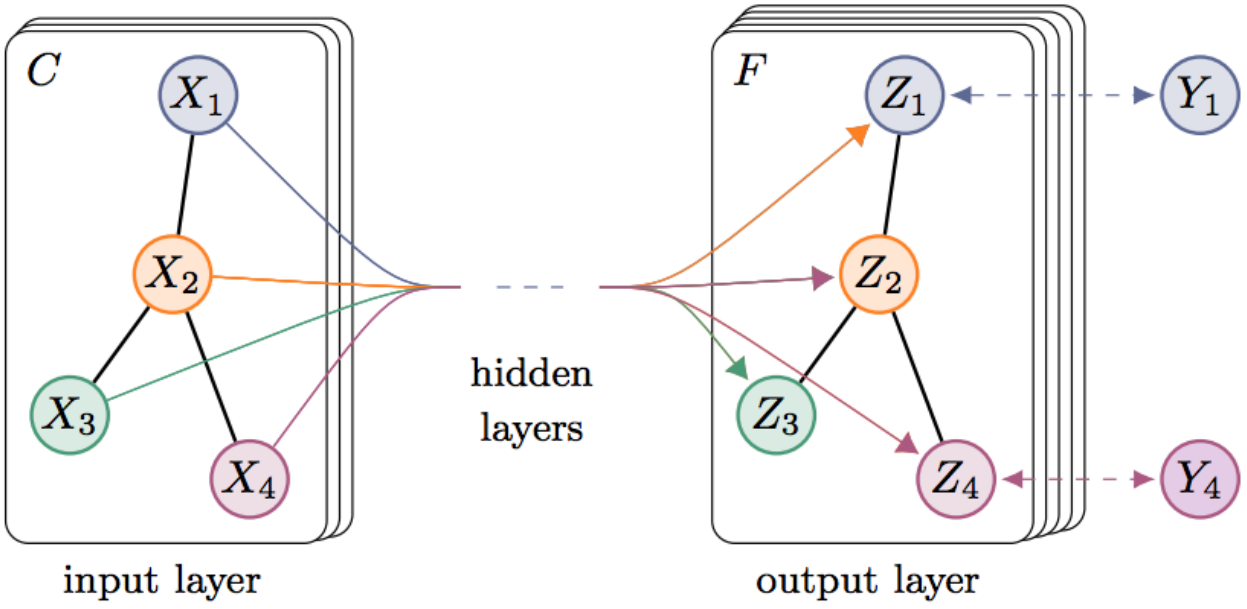
- We can add a softmax layer at the end and apply the GCN to predict the labels of the nodes of a graph, where some of the nodes already have labels (training set)

$$Z = f(X, A) = \mathrm{softmax}\left(\hat{A}\ \mathrm{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right)$$

- If we add a cross-entropy loss function we get the 2-layers GCN of Kipf et al.

$$\mathcal{L} = -\sum_{l\in\mathcal{Y}}\sum_{f=1}^{F} Y_{lf}\ln Z_{lf}$$

GCN for node classification

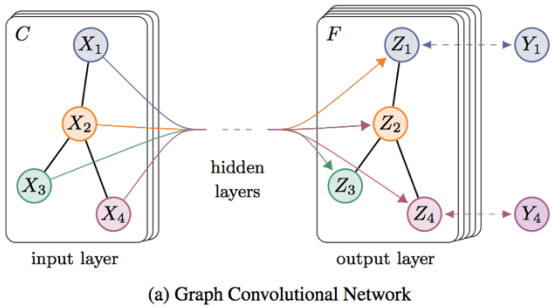


(a) Graph Convolutional Network

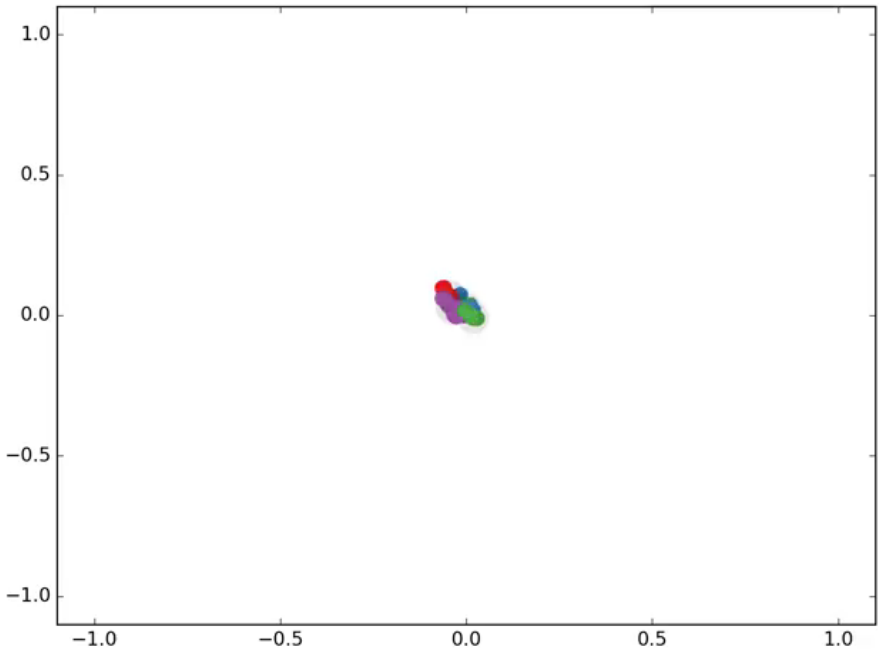
We can also explore the latent space to plot the network during the training: see this video

For an explanation of the video see Appendix A.1 and A.2 in the paper

GCN for node classification



(a) Graph Convolutional Network



We can also explore the latent space to plot the network during the training: see this video

For an explanation of the video see Appendix A.1 and A.2 in the paper

GCN for graph classification

- The previous architecture can be extended to deal with graph classifications (i.e., one label per graph) by combining the node-level features in a single graph-level feature
  - E.g., following the pooling-like approach of this paper

References and further reading

- *Duvenavud et al., NIPS 2015*
  http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints.pdf
- *Defferrard et al., NIPS 2016*
  https://arxiv.org/pdf/1606.09375.pdf
- *Kipf et al., ICLR 2017*
  https://arxiv.org/pdf/1609.02907.pdf
- *Gilmer et al., ICML 2017*
  https://arxiv.org/pdf/1704.01212.pdf

A very good tutorial with examples

How to do Deep Learning on Graphs with Graph Convolutional Networks

- https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780

Strongly recommend to read and excecise.

Summary

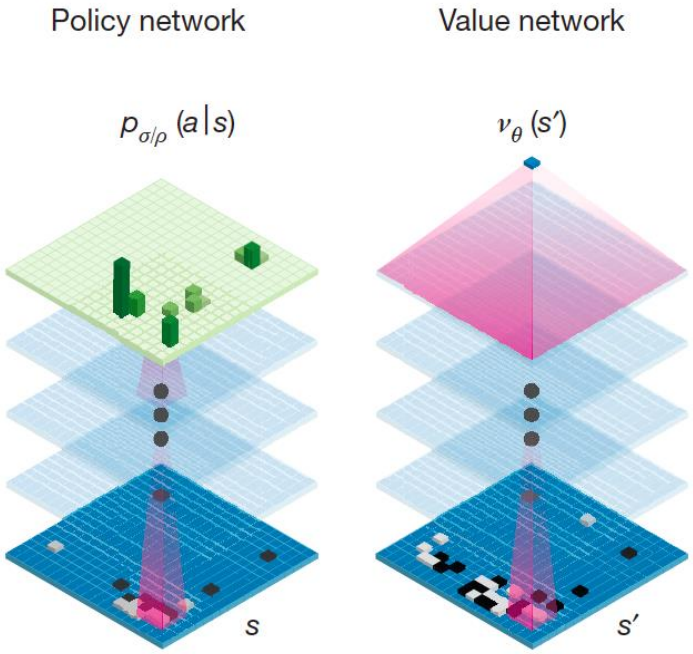
- Motivation of designing a deep learning model on Graphs
- What are the challenges
- Adjacency matrix, node features matrix, degree
- Normalization of the node matrix and adding the self-loop.
- Node classification and whole graph classification

One other area in DL not covered in this module

- Deep Reinforcement learning (Deep Q-Learning)
- Self-reading materials:
  - *A good introductory book for reinforcement learning is Reinforcement Learning: An Introduction
  - https://web.mst.edu/~gosavia/tutorial.pdf
  - V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Human-level control through deep reinforcement learning, *Nature* 2015
  - : https://adeshpande3.github.io/Deep-Learning-Research-Review-Week-2-Reinforcement-Learning

Applications of deep RL

- AlphaGo and AlphaZero



https://deepmind.com/research/alphago/

Deep Learning
and
Learning Deeper