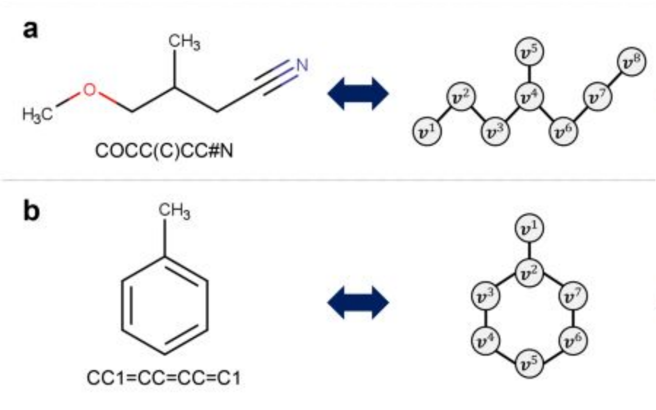
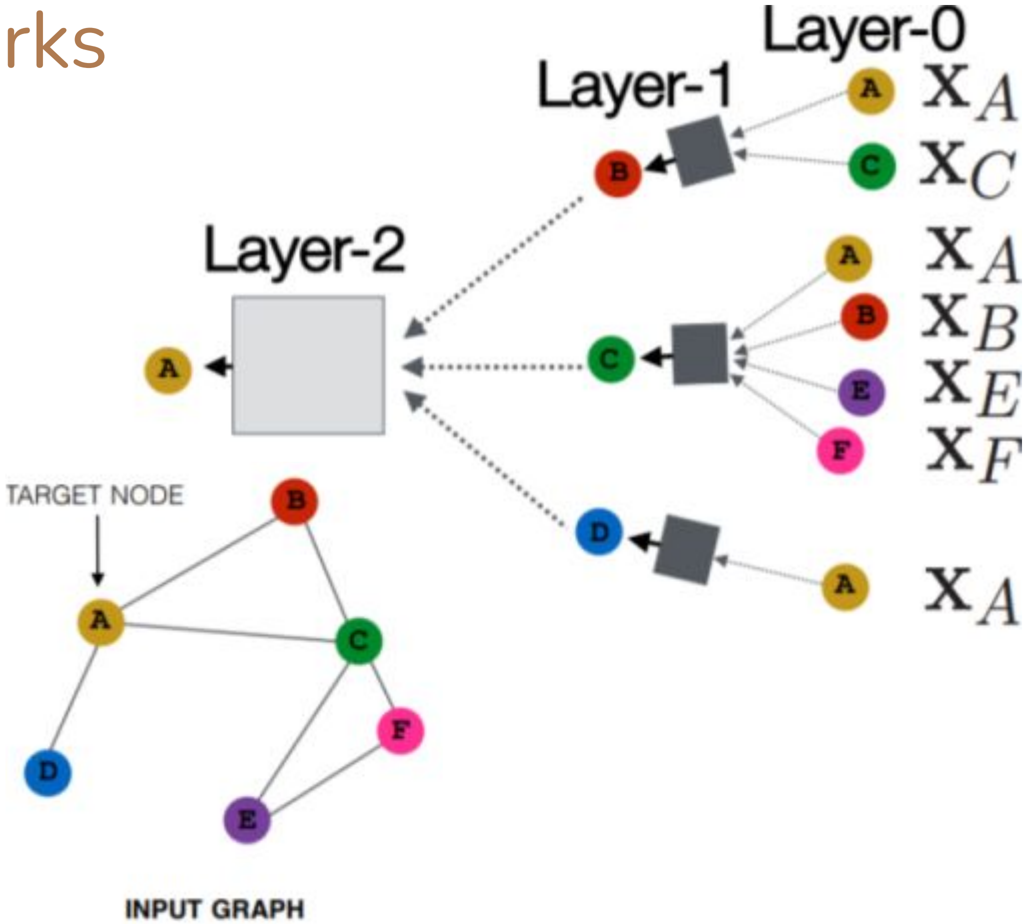
Training of Graph Neural Networks using Coarsening and Dynamic Mode Decomposition

William Hawkins, working under **Christopher Brissette**
Advisor: **George Slota**

Graph Neural Networks

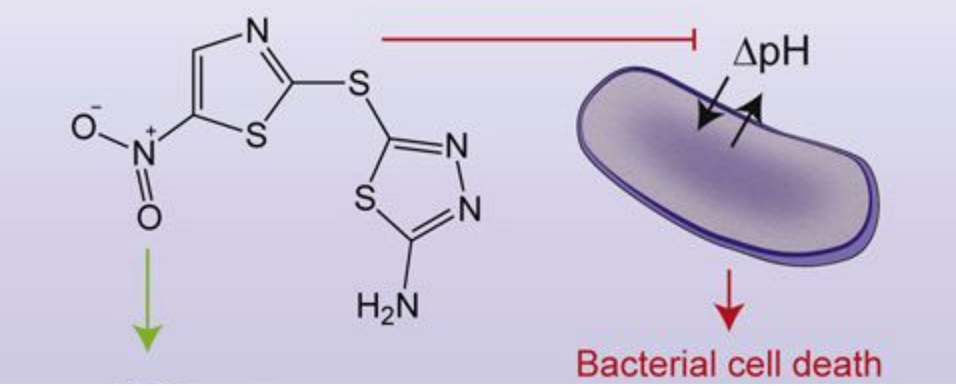
- Embeddings
  - Vertex Level
  - Edge Level
  - Graph Level

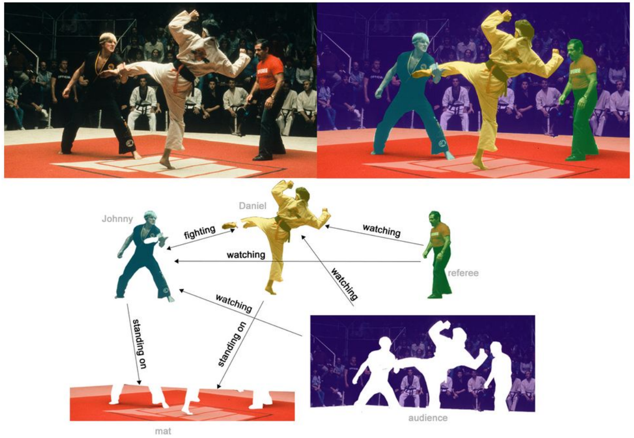


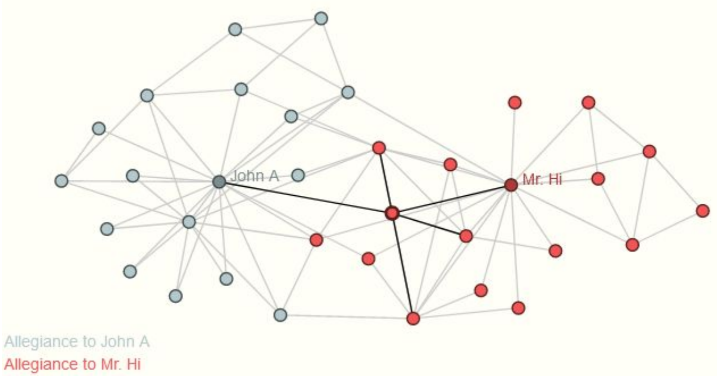


Graph Neural Network Tasks

- Graph Classification
- Vertex Classification
- Edge prediction

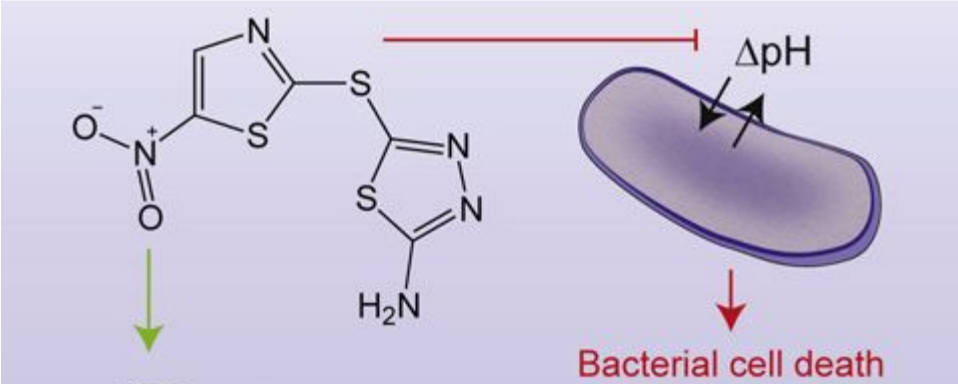


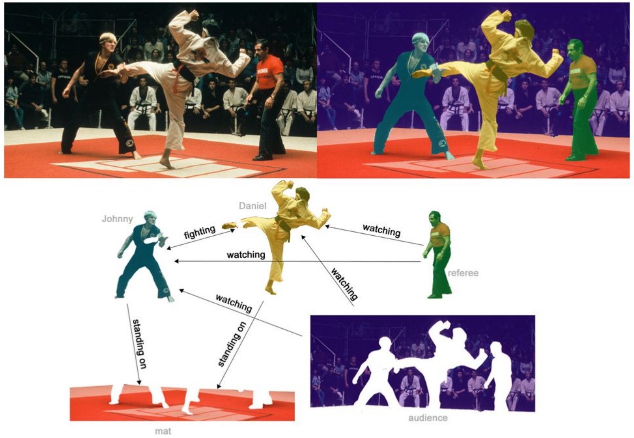


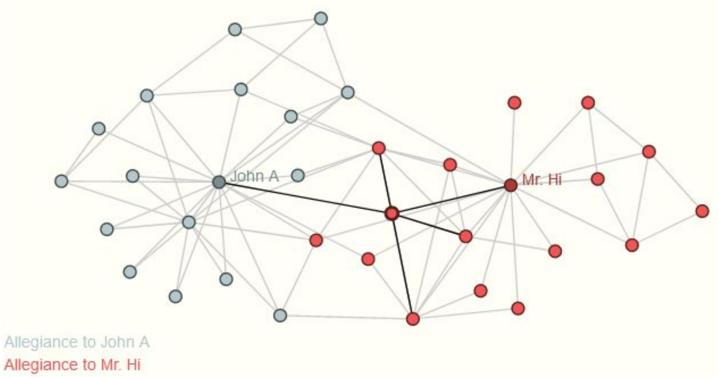


Graph Neural Network Tasks

- Graph Classification
- Vertex Classification
- Edge prediction

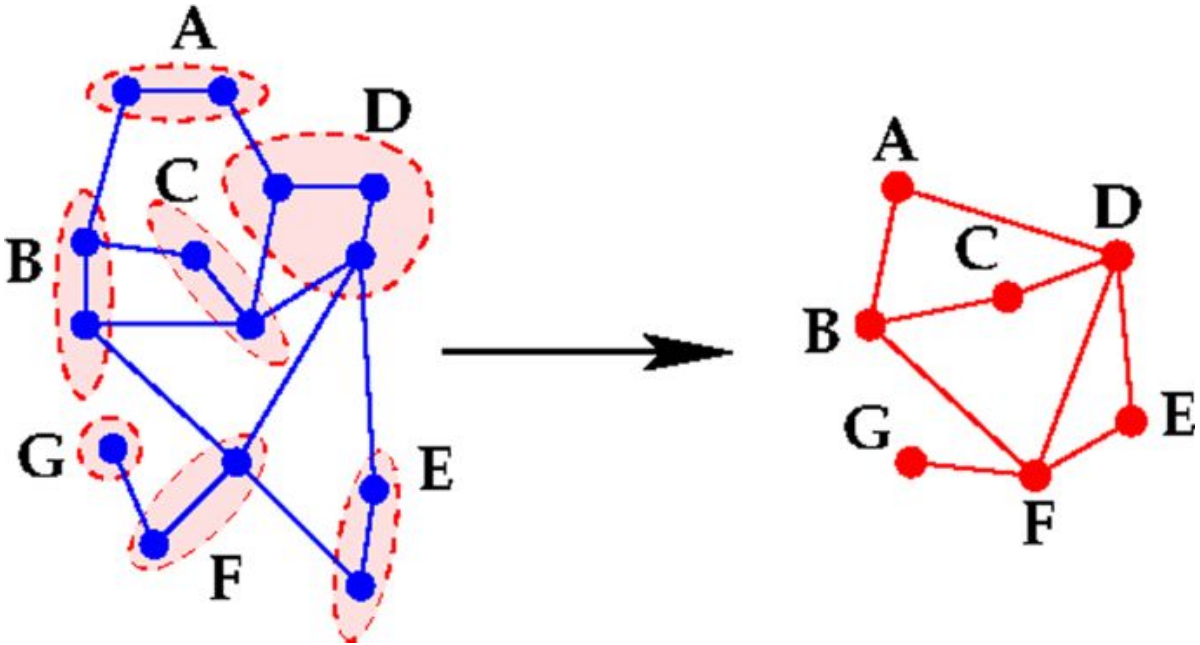






Coarsening

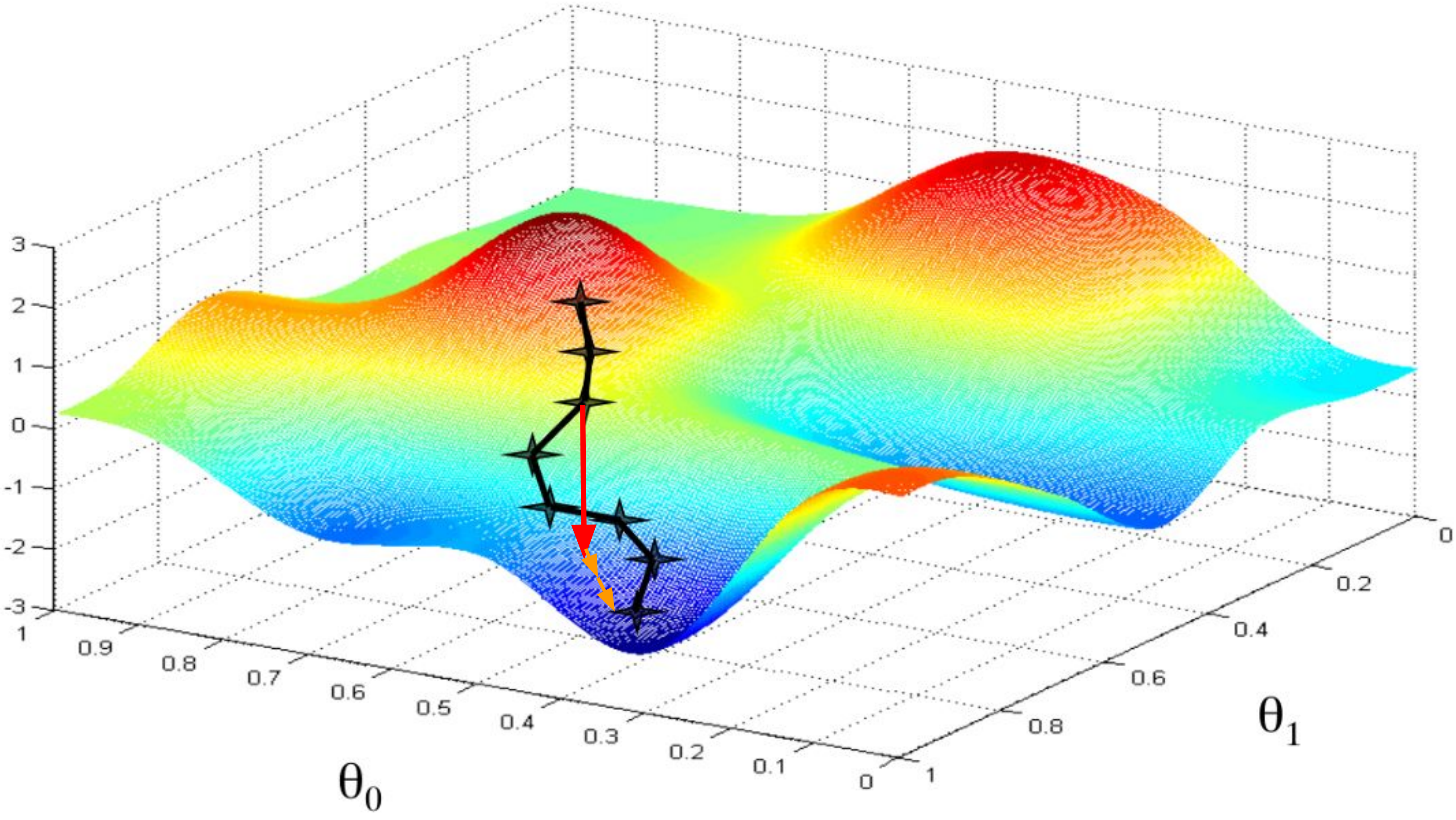
- Spectral Coarsening
- Multilevel coarsening



Dynamic Mode Decomposition

- SCHMID, P. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics, 656*, 5-28. doi:10.1017/S0022112010001217
- Used to model the evolution of a system.
- Matix ***W,*** where rows represent the state at each timestep

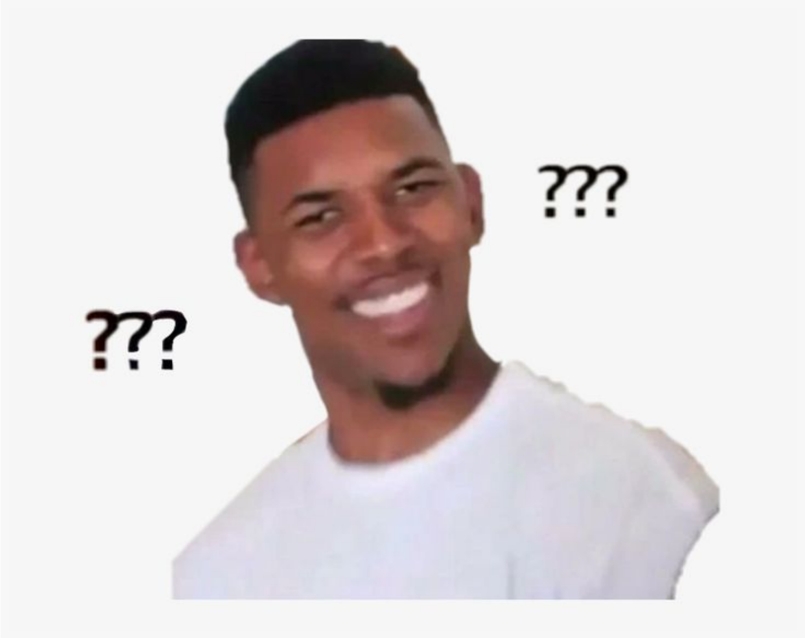
Compute low-cost SVD decomposition: $\boldsymbol{W}^{\ell,m^-} = \boldsymbol{U}_r^{\ell,m}\boldsymbol{\Sigma}_r^{\ell,m}\boldsymbol{V}_r^{\ell,m^T}$;
Select r modes such that $\boldsymbol{\Sigma}_r^{\ell,m}[r,r]/\boldsymbol{\Sigma}_r^{\ell,m}[0,0] >$ DMD filter tolerance;
Build reduced Koopman operator with $\boldsymbol{A}_r^{\ell,m} = \boldsymbol{U}_r^{\ell,m^T}\boldsymbol{W}^{\ell,m^+}\boldsymbol{V}_r^{\ell,m}\boldsymbol{\Sigma}_r^{\ell,m^{-1}}$.
Perform eigendecomposition of the reduced Koopman operator with $\boldsymbol{A}_r^{\ell,m}\boldsymbol{Y}_r^{\ell,m} = \boldsymbol{\Lambda}_r^{\ell,m}\boldsymbol{Y}_r^{\ell,m}$
Compute the matrix of weights modes: $\boldsymbol{\Phi}_r^{\ell,m} = \boldsymbol{U}_r^{\ell,m^+}\boldsymbol{Y}^{\ell,m}$ Compute initial
DMD condition: $\boldsymbol{b}_r^{\ell,m} = \boldsymbol{\Phi}_r^{\ell,m^T}\boldsymbol{w}^{\ell,m}$;
Evolve weights with DMD using $\boldsymbol{w}^{\ell,s} = \boldsymbol{\Phi}_r^{\ell,m}\left[\boldsymbol{\Lambda}_r^{\ell,m}\right]^{s-m}\boldsymbol{b}_r^{\ell,m}$


3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Algorithm 1: Acceleration of backpropagation with Dynamic Mode Decomposition

Input: m, DMD filter tolerance, s, Backpropagation Parameters, Total Epochs
Result: Trained weights $\boldsymbol{w}^{\ell}, \forall \ell \in \mathcal{H}_{\ell}$
$bp_{iter} = 0$,;
while *epoch* $\leq$ *Total Epochs* **do**
  Do backpropagation step;
  Extract weights $\boldsymbol{w}^{\ell, bp_{iter}}, \forall \ell \in \mathcal{H}_{\ell}$;
  Store weights: $\boldsymbol{W}^{\ell} \leftarrow [\boldsymbol{W}^{\ell}\, \boldsymbol{w}^{\ell, bp_{iter}}], \forall \ell \in \mathcal{H}_{\ell}$;
  $bp_{iter} + = 1$;
  if $bp_{iter} == m$ **then**
    for $\ell \in \mathcal{H}_{\ell}$ **do**
      Build training matrices: $\boldsymbol{W}^{\ell,m^-}$ and $\boldsymbol{W}^{\ell,m^+}$ with (1) and (2);
      Compute low-cost SVD decomposition: $\boldsymbol{W}^{\ell,m^-} = \boldsymbol{U}_r^{\ell,m} \boldsymbol{\Sigma}_r^{\ell,m} \boldsymbol{V}_r^{\ell,m^T}$;
      Select r modes such that $\boldsymbol{\Sigma}_r^{\ell,m}[r,r] / \boldsymbol{\Sigma}_r^{\ell,m}[0,0] >$ DMD filter tolerance;
      Build reduced Koopman operator with (3);
      Perform eigendecomposition of the reduced Koopman operator with (4) ;
      Compute the matrix of weights modes: $\boldsymbol{\Phi}_r^{\ell,m} = \boldsymbol{U}_r^{\ell,m^+} \boldsymbol{Y}^{\ell,m}$ Compute initial DMD condition: $\boldsymbol{b}_r^{\ell,m} = \boldsymbol{\Phi}_r^{\ell,m^T} \boldsymbol{w}^{\ell,m}$;
      Evolve weights with DMD using (5) ;
      Assign updated weights to layer ℓ in the neural network;
    end
    $bp_{iter} = 0$;
  end
end

Results



Data: r/Place

- A collaborative project hosted on reddit.
- Registered users could place a single pixel on the canvas from a set of colors.
- Had to wait 5 to 10 minutes before placing another pixel.
- Alone, one user could have little impact on the canvas. Communities would need to work together to construct designs.

r/Place as a Graph

- CSV file with 1 line per placed tile
  - User ID
  - Timestamp
  - x coordinate
  - y coordinate
  - Color
- Can consider the canvas as Graph and the pixels as vertices.

Next Steps and Ideas

- Interweaving with Coarsening
- Depth vs Height

- Implement in parallel C

References

- https://distill.pub/2021/gnn-intro/
- https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications
- DeepMind TensorFlow ML Tech Talks Petar Veličković