

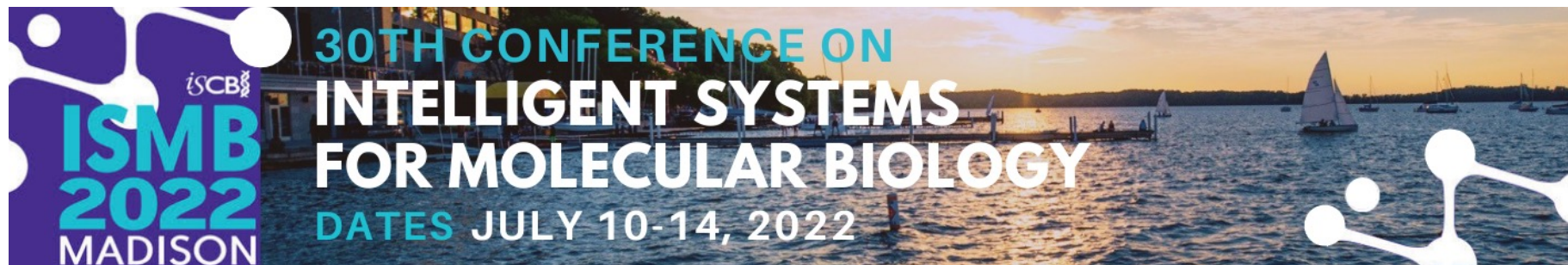
# Towards Precision Medicine with Graph Representation Learning

Michelle M. Li & Marinka Zitnik

Department of Biomedical Informatics  
Broad Institute of Harvard and MIT  
Harvard Data Science

[zitniklab.hms.harvard.edu/biomedgraphml](http://zitniklab.hms.harvard.edu/biomedgraphml)





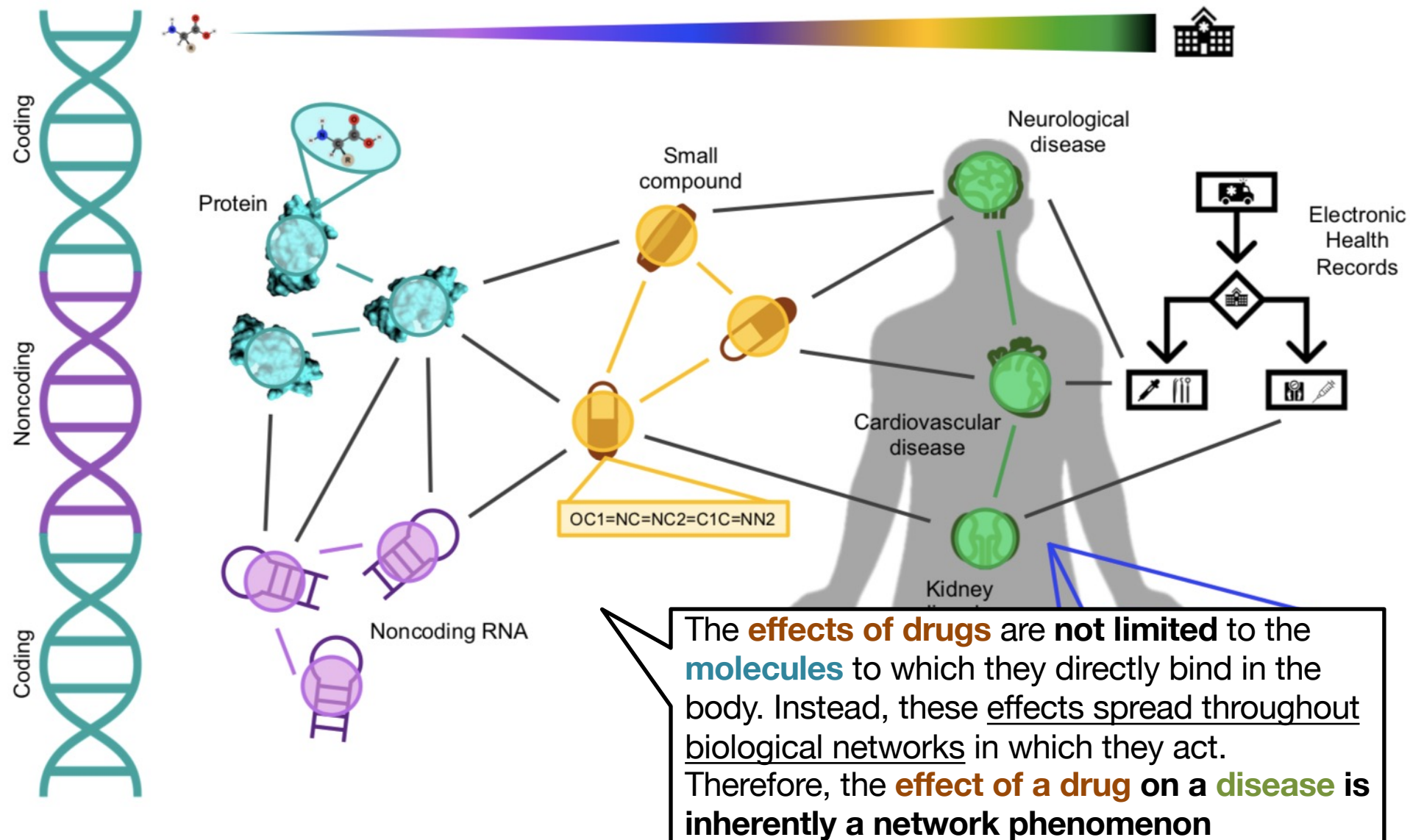
## Tutorial VT4

July 7, 2022 at 9am – 1pm CDT

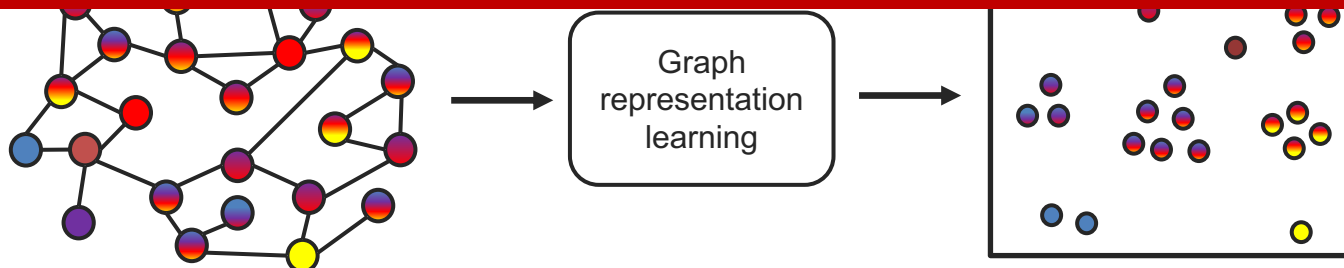
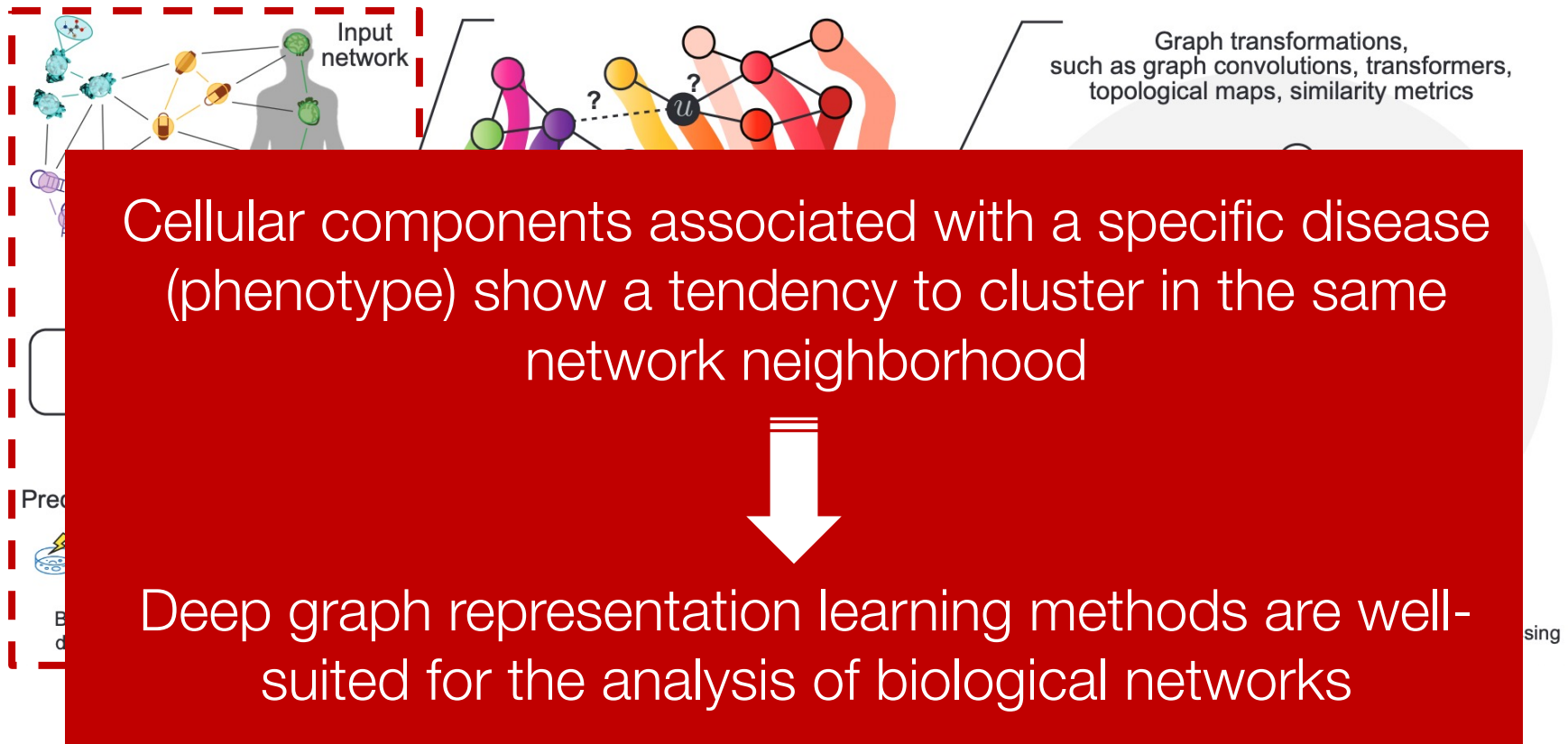


All tutorial materials are available at  
[zitniklab.hms.harvard.edu/biomedgraphml](https://zitniklab.hms.harvard.edu/biomedgraphml)


# Biology is interconnected



# Graph representation learning realizes key network principles for data-rich biomedicine



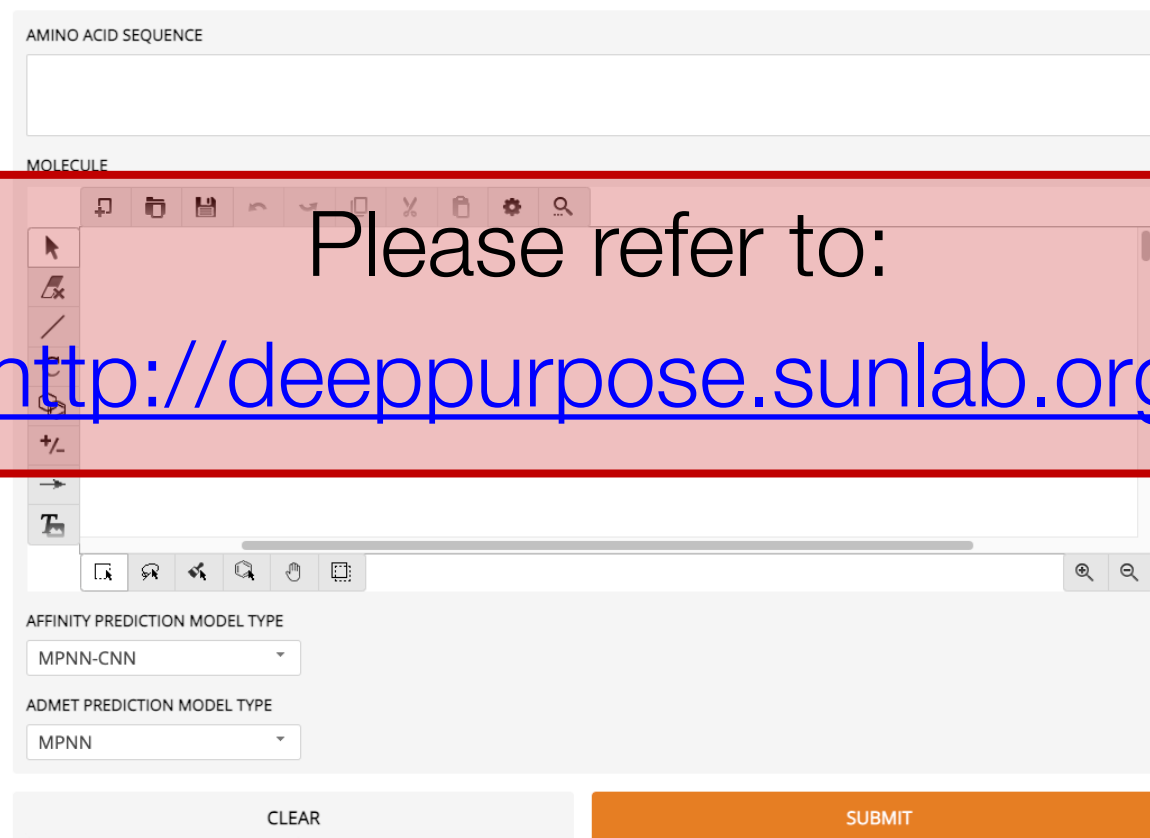
# This Tutorial

- ✓ 1. Methods: Network diffusion, shallow network embeddings, graph neural networks, equivariant neural networks
- ✓ 2. Applications: Fundamental biological discoveries and precision medicine
-  3. Hands-on exercises: Demos, implementation details, tools, and tips

# Hands-on Exercises

## [NeurIPS 2020 Demo] MolDesigner: Interactive Design of Efficacious Drugs with Deep Learning

|| Kexin Huang, Tianfan Fu, Vivian Hu, Dawood Khan, Ali Abid, Ali Abdalla, Abubakar Abid, Lucas M. Glass, Marinka Zitnik, Cao Xiao, Jimeng Sun || Demo: [youtube.com/watch?v=PbyHP0iRoP8](https://youtube.com/watch?v=PbyHP0iRoP8) || DeepPurpose: [github.com/kexinhuang12345/DeepPurpose](https://github.com/kexinhuang12345/DeepPurpose) || TDC: [github.com/mims-harvard/TDC](https://github.com/mims-harvard/TDC) || Gradio: [github.com/gradio-app/gradio](https://github.com/gradio-app/gradio) ||



The screenshot shows the MolDesigner web interface. At the top is a text input field labeled "AMINO ACID SEQUENCE". Below it is a section labeled "MOLECULE" which contains a 3D molecular model viewer. A large red rectangular box is overlaid on the "MOLECULE" section, containing the text "Please refer to:" followed by the URL <http://deeppurpose.sunlab.org/> in blue. Below the molecular viewer is a toolbar with various icons for manipulating the molecule. At the bottom of the interface, there are two dropdown menus: "AFFINITY PREDICTION MODEL TYPE" with "MPNN-CNN" selected, and "ADMET PREDICTION MODEL TYPE" with "MPNN" selected. At the very bottom are two buttons: "CLEAR" and "SUBMIT".

# Practical Advice & Tips

## Network & machine learning packages

- **Data science & machine learning basics:** Numpy, pytorch, pandas, scipy, scikit-learn
- **Visualization basics:** Matplotlib, plotly, UMAP, seaborn
- **Network construction & analysis:** NetworkX, iGraph, Stanford Network Analysis Platform (SNAP)
- **Graph machine learning:** Pytorch Geometric (+ tutorials), Deep Graph Library (+ public slack channel), Pytorch Lightning

## Developer tools for graphs & machine learning

- **Network visualization:** Gephi, Cytospace, GraphVis, Neo4j
- **Model visualization:** Weights & Biases, Neptune, TensorBoard, Comet, HiPlot
- **Code profilers:** cProfile/profile for Python3
- **Making paper-ready figures:** draw.io, Adobe color wheel

*Quick tip: New packages and software are released all the time, but there's no need to be overwhelmed! Stick with a few that you like and be proficient in them!*

# Practical Advice & Tips

## Helpful tools for programming

- **Notebooks:** Google Colab (free GPUs), Jupyter Notebook
- **Version control:** GitHub
- **Environment management system:** Conda
- **Integrated Development Environment (IDE):** Visual Studio Code, PyCharm, Spyder
- **Terminal multiplexers:** Screen, tmux
- **Runtime:** tqdm
- Use debuggers and breakpoints (e.g., pdb package, built-in debuggers in IDEs)

## Implementation tips

- Normalize by degree (e.g., node features, node weights)
- Apply batch and/or layer normalization
- Use minibatching (especially on large graphs)
- Experiment with the number of layers (deeper GNNs do not necessarily yield better performance)



# Practical Advice & Tips

## Training tips: Data & metrics

- Split dataset into train, validation, and test sets (avoid data leakage!)
- For self-supervised learning, carefully define negative samples
- Define metrics carefully (e.g., Is the data is balanced or imbalanced?)
  - Balanced: Receiver operating characteristic (ROC) curve
  - Imbalanced: Average precision (AP) score
- Start small (e.g., toy dataset) and then scale up
- Log everything (e.g., print statements, progress bars, model visualization software)

## Training tips: Model evaluation

- Ensure that the model can overfit on training data (e.g., >90% ROC)
- Avoid over- or under-training the model (e.g., specify number of epochs, define early-stopping criteria)
- Evaluate hyperparameters on left-out validation set (e.g., grid search, random search) – *Don't be overwhelmed by all the possible parameters; start with learning rate! Consider using a learning rate scheduler.*
- Establish strong baseline models (e.g., consider domain-specific as well as classic machine learning models)
- Set a seed for code to ensure reproducibility
- Save models at critical points (e.g., new minimum loss, new maximum ROC)
- Make sure code *actually* works before requesting tons of resources (e.g., cluster)

# Practical Advice & Tips

## Common bugs in implementation

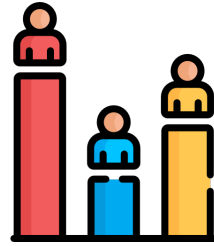
- Performance is random / opposite to what is expected
  - Edges are unintentionally shuffled during training
  - An imported function is re-indexing, and it is unaccounted for in our code
- Out-of-memory error on GPU
  - Implement minibatching or decrease batch size
  - Decrease model size (e.g., number of layers, number of attention heads)

## Common model training issues

- Model has near-perfect performance on the validation/test set
  - Data leakage between train and validation/test set
- Model takes forever to start running
  - Minimize preprocessing steps done at launch
  - Consider loading only the necessary data (e.g., columns/rows in dataframe)
- Model takes a long time to converge
  - Increase learning rate
  - Apply normalization (e.g., batch norm, which allows us to increase learning rate too!)
- Model is overfitting
  - Apply dropout layer
  - Simplify model (e.g., remove layers)
  - Use early stopping

# This Tutorial

- ✓ 1. Methods: Network diffusion, shallow network embeddings, graph neural networks, equivariant neural networks
- ✓ 2. Applications: Fundamental biological discoveries and precision medicine
- ✓ 3. Hands-on exercises: Demos, implementation details, tools, and tips



Time for a poll question about...

# OUR TUTORIAL

1. What is the most memorable thing about graph representation learning that you learned today?  
*Fill in the blank*
2. What topic would you like us to cover in future iterations of this tutorial? *Fill in the blank*

# ISMB 2022 Tutorial Feedback

Which tutorial did you attend? (If you attended more than one tutorial, you will <sup>\*</sup> have the opportunity to submit a second survey)

- ☐ Tutorial IP1: Gene regulatory network inference from single-cell transcriptomics data
- ☐ Tutorial IP2: A practical introduction to the design, quantification, and analysis of

Please refer to:

[docs.google.com/forms/d/e/1FAIpQLSftVbl5O-P6EidL-PBgmqjdVE9QX3SfsgGKqkX6DDxJzGvrfQ/viewform](https://docs.google.com/forms/d/e/1FAIpQLSftVbl5O-P6EidL-PBgmqjdVE9QX3SfsgGKqkX6DDxJzGvrfQ/viewform)

- ☐ Tutorial VT1: Introduction to Python programming for bioscientists
- ☐ Tutorial VT2: Building Interactive Visualizations of Genomics Data with Gosling
- ☐ Tutorial VT3: Federated Learning in Biomedicine
- ☒ Tutorial VT4: Towards Precision Medicine with Graph Representation Learning
- ☐ Tutorial VT5: Computational analysis of antibody repertoires, with applications for therapeutic discovery
- ☐ Tutorial VT6: Online tools for visualizing RNA structure

# Resources

- Books & survey papers

- William Hamilton, *Graph Representation Learning* ([morganclaypool.com/doi/abs/10.2200/S01045ED1V01Y202009AIM046](https://morganclaypool.com/doi/abs/10.2200/S01045ED1V01Y202009AIM046))
- Li et al., Graph Representation Learning for Biomedicine ([arxiv.org/abs/2104.04883](https://arxiv.org/abs/2104.04883))

- Keynotes & seminars

- Michael Bronstein, “Geometric Deep Learning: The Erlangen Programme of ML” (ICLR 2021 keynote) ([youtube.com/watch?v=w6Pw4MOzMuo](https://youtube.com/watch?v=w6Pw4MOzMuo))
- Broad Institute Models, Inference & Algorithms: Actionable machine learning for drug discovery; Primer on graph representation learning ([youtube.com/watch?v=9YpTYdru0Rg](https://youtube.com/watch?v=9YpTYdru0Rg))
- Stanford University (CS224W Lecture): Graph neural networks in computational biology ([youtube.com/watch?v=\\_hy9AgZXhbQ](https://youtube.com/watch?v=_hy9AgZXhbQ))
- AI Cures Drug Discovery Conference ([youtube.com/watch?v=wNXSkISMTw8](https://youtube.com/watch?v=wNXSkISMTw8))

- Conferences & summer schools

- London Geometry and Machine Learning Summer School ([logml.ai](https://logml.ai))
- Learning on Graphs Conference ([logconference.github.io](https://logconference.github.io))

# Resources

- **Software & packages**

- PyTorch Geometric
- NetworkX
- Stanford Network Analysis Platform (SNAP)

- **Tutorials & code bases**

- Pytorch Geometric Colab Notebooks ([pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html](https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html))
- Zitnik Lab Graph ML Tutorials ([github.com/mims-harvard/graphml-tutorials](https://github.com/mims-harvard/graphml-tutorials))
- Stanford University's CS224 ([web.stanford.edu/class/cs224w](https://web.stanford.edu/class/cs224w))

- **Datasets**

- Precision Medicine Oriented Knowledge Graph (PrimeKG) ([zitniklab.hms.harvard.edu/projects/PrimeKG](https://zitniklab.hms.harvard.edu/projects/PrimeKG))
- Therapeutic Data Commons (TDC) ([tdcommons.ai](https://tdcommons.ai))
- BioSNAP ([snap.stanford.edu/biodata/](https://snap.stanford.edu/biodata/))
- Open Graph Benchmark (OGB) ([ogb.stanford.edu](https://ogb.stanford.edu))