Visualization

AI & Visualization

Overview

- Applying AI techniques in Information Visualization
- Visual Analytics In Deep Learning
- Applying Deep Learning techniques in Scientific Visualization

Overview

- Applying AI techniques in Information Visualization
- Visual Analytics In Deep Learning
- Applying Deep Learning techniques in Scientific Visualization

Introduction

- There has been a growing research interest to apply AI techniques to assist visualization-related activities
- Similar to common data formats like text and images, visualizations are incingreasingly created, analyzed and reused with the power of AI.
- Some areas where AI techniques have been applied successfully are:
  - Visualization Generation
  - Graph Layout
  - Visualization Understanding

Visualization Feature Representation

- Features of visualizations are the measurable properties serving as the input to machine learning models.
- Features are extracted by feature engineering or feature learning.
- Feature engineering is the process of using domain knowledge to extract features from raw data
- Feature learning replaces this manual process by developing automated approaches that automatically discover useful representations.
- The features of visualization data are multimodal:
  - Graphics
  - Program
  - Text
  - Data
- Some papers use multiple features for different tasks
- Some use hybrids by feature fusion for improving performances

Graphical Features

- The most common features of visualizations.
- The overarching goal is for predictive tasks, e.g., to predict the chart type, or detection tasks.
- Early work uses general image descriptors
  - These descriptors are designed for general visual content, containing only low-level information such as shapes and regions.
- To raise the level of abstraction, researchers have proposed special domain descriptors for visualization-specific information
  - In many cases outperform general descriptors
  - Examples include regions of text elements (e.g., titles and labels), positions of text and mark elements, relative positions between text and marks, and visual styles of elements.
  - Despite promising results, such feature engineering process remains labor-intensive and requires expertise.
  - Besides, it remains unclear whether human-crafted features are informative and discriminating for accomplishing machine learning tasks.
- Recent work has adopted feature learning more than feature engineering

Program Features

- Program features are extracted from the programs such as specifications.
- The most straightforward representation is the parameters.
  - For example some models learn to operate on the chart parameters, e.g., positioning the element.
- However few systems uses programs as features which might be due to several reasons, including
  - the lack of training data
  - the overheads of reserve engineering to extract the program from visualization graphics.
- Still, programs are a promising representation as they contain high-level visualization specific information.

Text Features

- Text features refer to the text content in visualizations such as titles.
- They are considered to improve the feature informativeness by incorporating semantic information.
- Moreover, text features can capture the subject matter of visualizations.
- However, it still remains unclear how text features could be effectively fused with graphical features to more comprehensively represent a visualization

Data Features

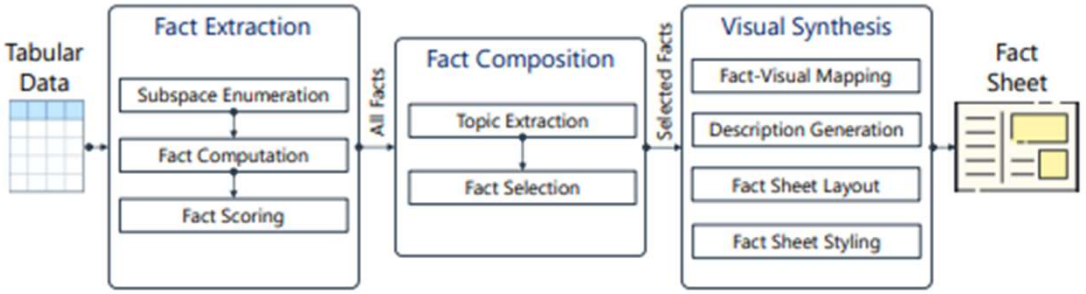
- The underlying data that the visualization encodes can also be used as a feature.
- The data can be used to describe a visualization
- Systems also perform feature engineering to consider data statistics such as the number of unique values in a column.
- Future research should propose effective feature selection approaches or complementary feature learning methods

Visualization Generation

- Authoring effective visualizations is often time-consuming and challenging
- One of the central research problems in the visualization community is to ease the creation of visualizations
- AI techniques can be used to automatically generate visualizations.
- There are four subcategories that distinguish visualization generation approaches by user input:
  - Data-based generation: Outputs visualizations given a database or a data-table.
  - Anchor-based generation: Recommend a visualization given an anchor visualization
  - Design-based generation: Generating visualizations by injecting the target data into a reference design. This is referred to as style imitation or visualization-by-example.
  - Insight-based generation: Learn the mapping from insights to suitable visual representations. To facilitate this mapping, the systems usually describe visual representations using specifications in a predefined design space (such as natural language description) to ensure a well-defined output space

DataShot

- DataShot automatically identifies insights from tabular data and presents these identified insights to the users through infographics
- DataShot system overview and processing pipeline

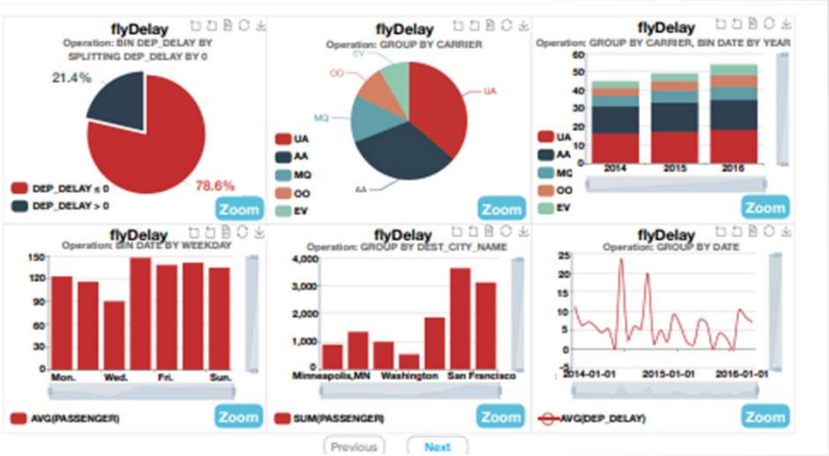


- A decision tree model is used with the 793 exemplary infographic elements, taking data types as the input of the model, and retrieve the suitable infographic or visualization options as the output.

- Y. Wang, Z. Sun, H. Zhang, W. Cui, K. Xu, X. Ma, and D. Zhang, "Datashot: Automatic generation of fact sheets from tabular data," IEEE Transactions on Visualization and Computer Graphics, 2019.

DeepEye

- Combines supervised ML techniques with expert rules to automatically recommend good visualizations for users.
- Given a dataset and a keyword query, DeepEye understands the query intent.
- All the relevant visualizations are enumerated and classified as "good" or "bad" by a binary decision tree classifier.
- All the "good" visualizations are then ranked and provided to users
- Employs declarative visualization languages similar to Vega-Lite to enable the creation of common visualizations (e.g., bar, pie charts).
- A decision tree model learns the mapping from insights and data characteristics to visualization specifications.

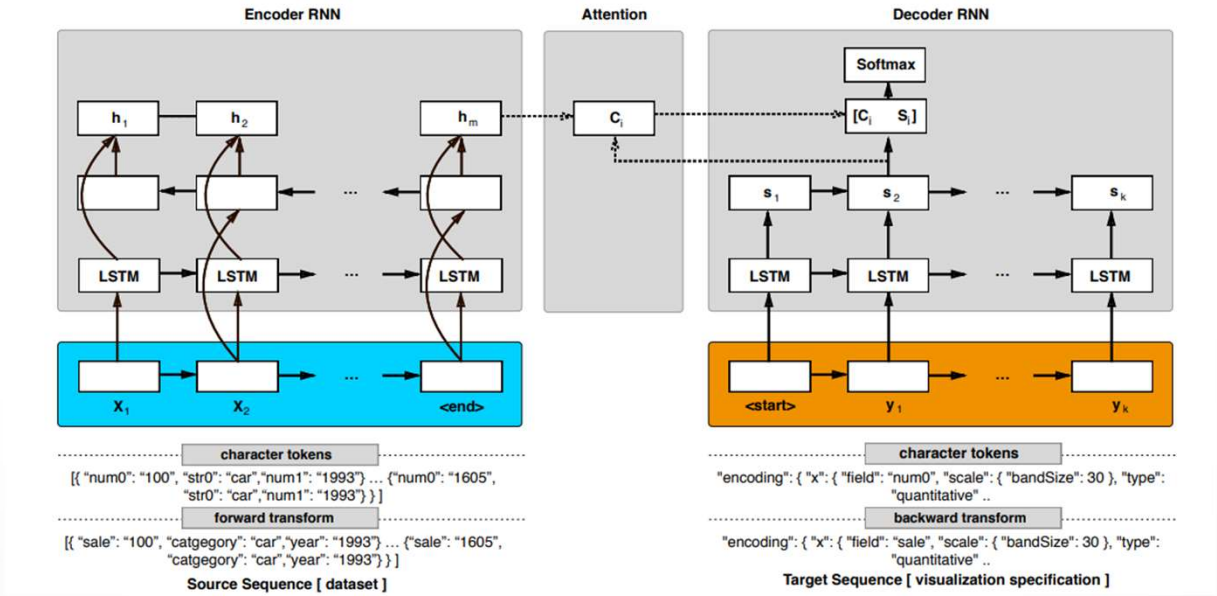


Screenshot for Running DEEPEYE for Dataset of Flight Statistics

- Y. Luo, X. Qin, N. Tang, G. Li, and X. Wang, "Deepeye: Creating good data visualizations by keyword search," in Proceedings of the ACM International Conference on Management of Data, 2018.
- Y. Luo, X. Qin, N. Tang, and G. Li, "Deepeye: towards automatic data visualization," in Proceedings of the IEEE International Conference on Data Engineering. IEEE, 2018

Data2Vis

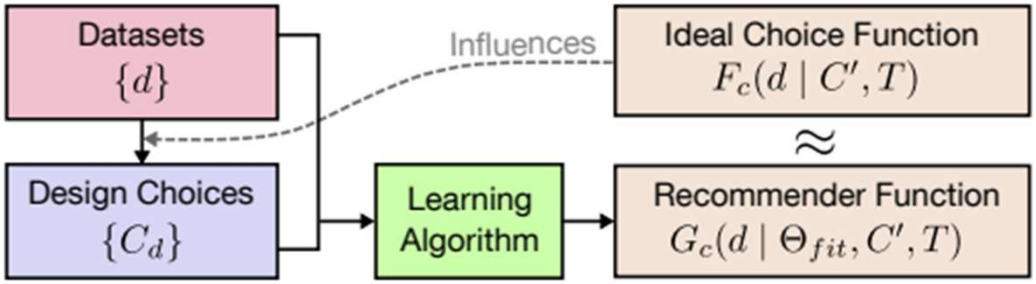
- Formalizes the process of Visualization Generation as a sequence to sequence translation from the original data to the visualization specifications
- Data2Vis is a sequence to sequence model with encoder-decoder architecture and attention module.
- To simplify learning, the system performs a simple forward and backward transformations on the source (dataset in JSON format) and target sequence (Vega-Lite visualization specification) which are then converted to character tokens
- ML automatically learns design choices from the training dataset and directly translates data to suitable visualizations.
- As a result, the employment of ML can effectively reduce the manual efforts in selecting data attributes, designing visual representations, and specifying the mapping from data values to graphic marks.



- V. Dibia and C. Demiralp, "Data2vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks," IEEE Computer Graphics and Applications, 2019.

VizML

- Collects a large visualization dataset
- Trains a fully-connected neural network
- Predicts the top five design choices when creating visualizations for a specific dataset.
- Decomposes the creation of a visualization as a series of classification tasks, including classifying the type of marks, the type of shared axis, and the type of visualization.
- VizML utilizes 841 features including single- and pairwise-column features of the input dataset and claims to significantly outperforms Data2Vis and DeepEye.

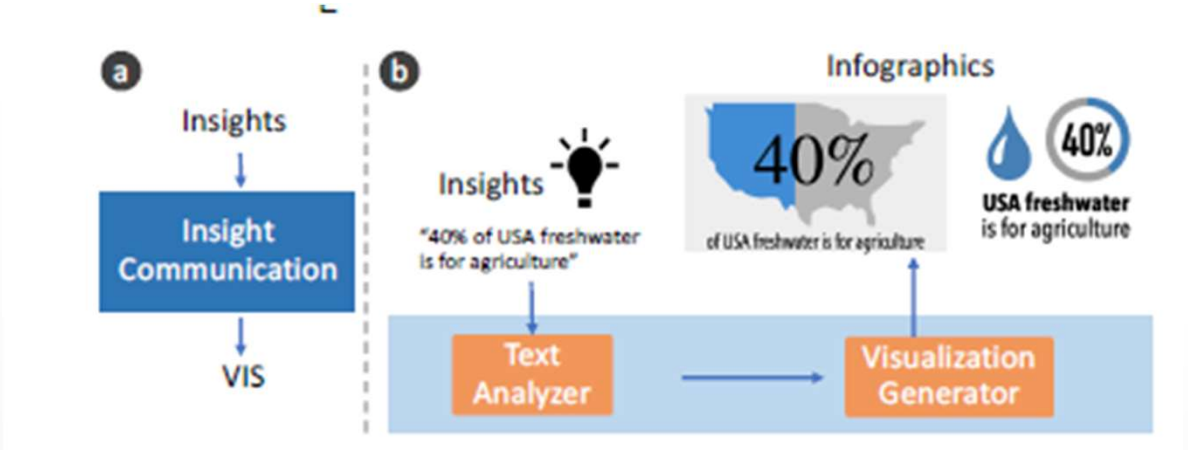


- Basic setup of learning models to recommend design choices with a corpus of datasets and corresponding design choices

- K. Hu, M. A. Bakker, S. Li, T. Kraska, and C. Hidalgo, "VizML: A machine learning approach to visualization recommendation," in CHI 2019.

Text-to-Viz

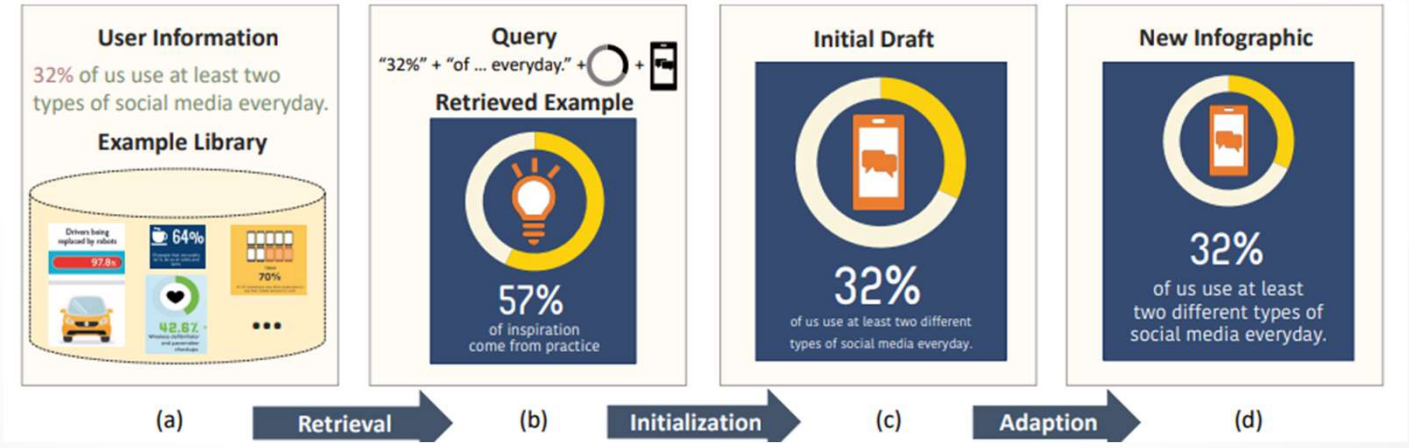
- Summarizes four design-space dimensions of infographics from discussion with design experts.
  - layout
  - description
  - graphic
  - color
- A visualization generator synthesizes infographics by generating a tuple of values on these four dimensions.
- Moreover allows users to provide insights through natural language statements such as "40% of USA freshwater is for agriculture"
- A text analyzer (a supervised CNN+CRF model) is trained to extract entities from the sentence and understand the provided insights.



- W. Cui, X. Zhang, Y. Wang, H. Huang, B. Chen, L. Fang, H. Zhang, J.-G. Lou, and D. Zhang, "Text-to-Viz: Automatic generation of infographics from proportion-related natural language statements," IEEE Transactions on Visualization and Computer graphics, 2019

Retrieve-then-Extract

- The system proposes to generate infographics by automatically imitating examples.
- It presents a two-stage approach, namely retrieve-then-adapt.
- In the retrieval stage, online examples are indexed by their visual elements.
- A given user information is transformed to a concrete query by sampling from a learned distribution about visual elements
- Retrieve appropriate examples in the example library based on the similarity between example indexes and the query.
- For a retrieved example, an initial drafts are generated by replacing its content with user information.
- However, in many cases, user information cannot be perfectly fitted to retrieved examples.
- Therefore there is an adaption stage.
- Leverage recursive neural networks to help adjust the initial draft and improve its visual appearance iteratively, until a satisfactory result is obtained.



C. Qian, S. Sun, W. Cui, J.-G. Lou, H. Zhang, and D. Zhang, "Retrieve-then-adapt: Example-based automatic generation for proportion related infographics," IEEE Transactions on Visualization and Computer Graphics, 2020.

Graph Layout

- Machine Learning techniques have been extensively employed to generate graph drawings that mimic the layout styles of given graph drawing examples.
- DeepDrawing [Y. Wang, et al, "Deepdrawing: A deep learning approach to graph drawing," IEEE TVCG, 2019]
  - Trains a graph-LSTM to learn one specific layout style from graph drawing examples.
  - Instead of drawing a single graph in diverse layouts, the trained ML model in DeepDrawing directly maps new input data into graph drawings that share similar layout styles with the training examples

Graph Layout

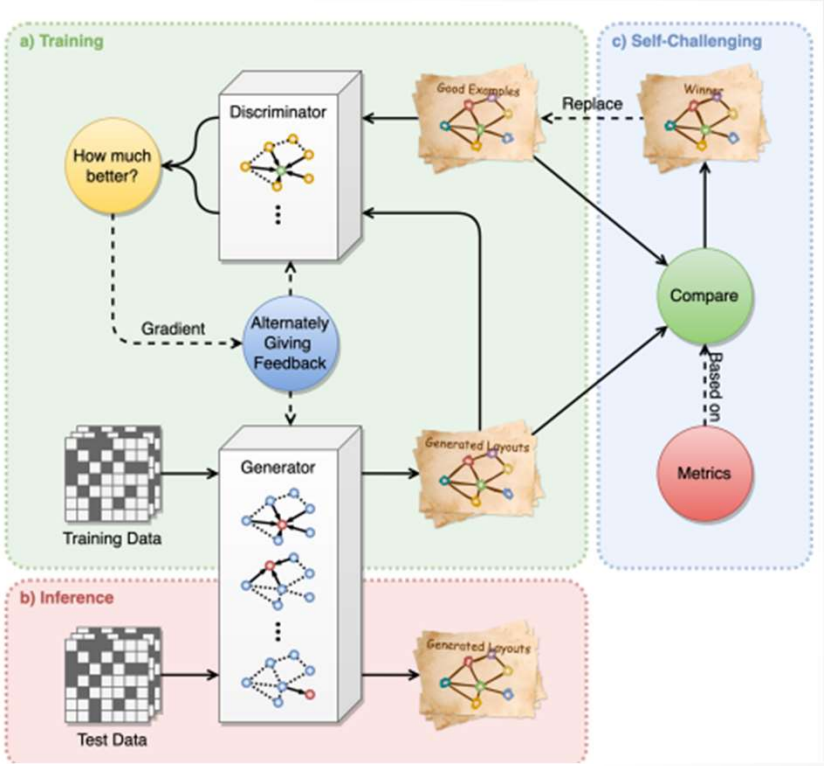
- O. Kwon et al., "What would a graph look like in this layout? a machine learning approach to large graph visualization," IEEE TVCG, 2018
  - Presented an ML approach to facilitate large graph visualization by learning the topological similarity between large graphs.
  - Specifically, the method provides a quick overview of visualizations for an input large graph data by learning from the drawings with similar topology structures.
  - This approach is further extended in [O. Kwon and K. Ma, "A deep generative model for graph layout," IEEE TVCG, 2020] which learns the layout from examples using a deep generative model.
  - The model is then used to provide users with an intuitive way to explore the layout design space of the input data.

Graph Layout

- DeepGD can generates graph layouts complying with multiple aesthetic metrics simultaneously
- It can be applied easily to optimize most of the commonly agreed aesthetic metrics.
- Also, DeepGD only needs to be trained once
- A Convolutional Graph Neural Network (ConvGNN) is used to produce the position of nodes such that the desired aesthetic metrics are optimized in the resulting layout.
- To accomplish this goal, a multi-objective loss function is designed in a way that each of the aesthetic aspects is represented by an individual component in the composite loss function.
- Additionally, two adaptive training strategies are proposed to automatically adjust the weight factors corresponding to each loss component such that the human preference on aesthetics trade-off is reflected in the weight factors.
- Xiaoqi Wang, Kevin Yen, Yifan Hu and Han-Wei Shen. DeepGD: A Deep Learning Framework for Graph Drawing Using GNN. IEEE Computer Graphics and Applications, 2021

Graph Layout

- SmartGD is a novel Generative Adversarial Network (GAN) based deep learning framework for graph drawing,
- The system can optimize different quantitative aesthetic goals like minimizing stress, minimizing edge crossing, maximizing crossing angle, as well a combination of multiple aesthetics.
- Xiaoqi Wang, Kevin Yen, Yifan Hu and Han-Wei Shen. SmartGD: A GAN-Based Graph Drawing Framework for Diverse Aesthetic Goals. IEEE transactions on visualization and computer graphics, 2022



The high-level overview of SmartGD:
(a) The training procedure of the GAN-based model.
(b) Describes the inference procedure for drawing unseen graph.
(c) The self-challenging mechanism is applied only when the optimization criterion is given to learn better quality layouts than the collected examples.

Visualization Understanding

- AI techniques have been used to automatically understand visualizations.
- For this purpose extracting the content of the visualization is needed. Various techniques can be utilized
- Once the visualizations are understood they can be utilized for various purposes like:
  - Chart Classification
  - Chart Summarization
  - Visual Question Answering

Extracting Visualization Content

- Poco & Heer (EuroVis 2017) automatically recovers visual encodings from a chart image, primarily using inferred text elements.
  - Utilizes an end-to-end pipeline which takes a bitmap image as input and returns a visual encoding specification as output.
  - The system detects text elements in a chart, classifies their role (e.g., chart title, x-axis label, y-axis title, etc.), and recovers the text content using optical character recognition.
  - A Convolutional Neural Network is trained for mark type classification.
  - Using the identified text elements and graphical mark type, we can then infer the encoding specification of an input chart image.
- Since contents of bitmap visualizations are usually hard to be automatically extracted, some studies have proposed methods to embed the required information (e.g., meta data, color schema) into bitmap visualizations.
  - The embedded information will not influence human perception of the visualization and can then be easily extracted from the visualization.
- VisCode trains an encoder-decoder network
  - The encoder embeds a QR code to the background of a bitmap visualization that could conceal additional information
  - The decoder extracts the QR code from the visualization.
  - IEEE Transactions on Visualization and Computer Graphics, 2020

Chart Classification

- Utilizing machine learning techniques for Chart Classification in a well studied problem
- Initial research utilized Classical classifiers (e.g., support vector machine)
- However classical classifiers require hand-crafted image features such as histograms of the image gradients (HOG) and dense sampling
- They have been gradually superseded by deep learning classifiers (e.g., convolutional neural network (CNN)).
- This is mainly because CNNs can effectively learn abstract features from raw visualization images,
- Several approaches seek to improve the representativeness of features by incorporating element-level features such as text and shape style features.

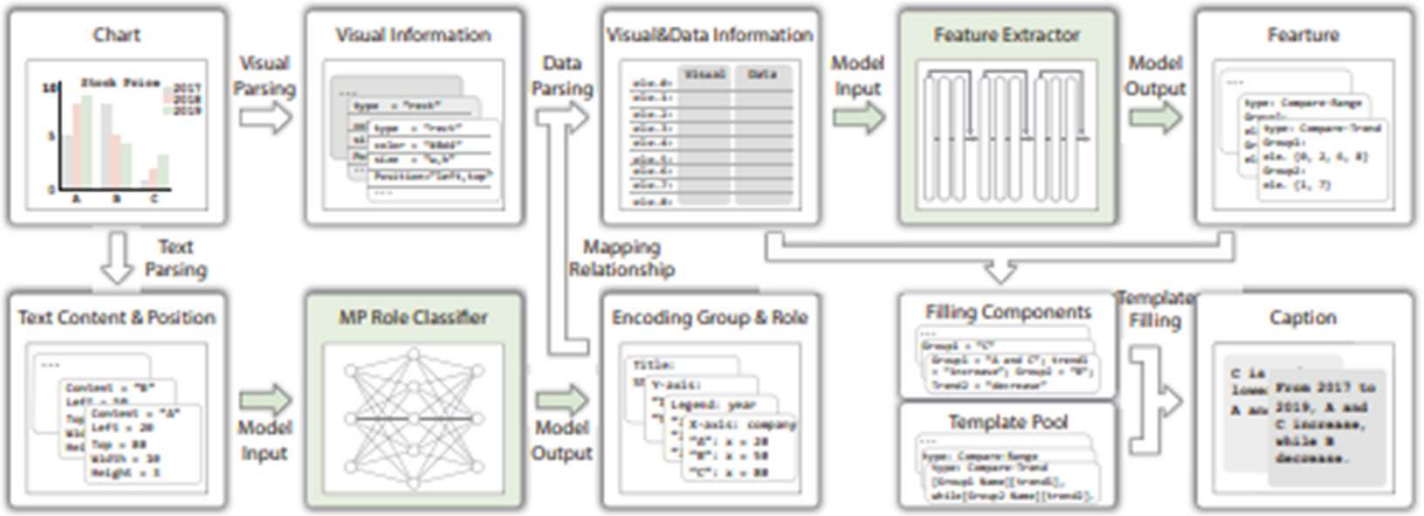
Chart Summarization

- Chart summarization becomes increasingly important with the rapid popularization of visualizations.
- Most existing approaches generate text summaries such as natural language description or captions
- The simplest approach is to provide a short description of how to interpret the chart
- More advanced approaches focus on explaining and communicating high-level insights conveyed by charts.
- Those approaches extract the data patterns and subsequently convert patterns to natural language according to pre-defined templates
- A common limitation of the above work is that natural language summaries are generated via pre-defined templates
- Therefore confined to few variations and generality.
- Recent research proposes several end-to-end deep-learning solutions for generating chart captions or text summarization.
- However, summarization still remains highly under-explored since the algorithm performances have much space for improvements.

AutoCaption

- AutoCaption generates a text caption to describe the insights of a visualization based on four types of chart features detected (i.e., aggregation, comparison, trend, distribution) automatically.
- Visual marks and visual channels, together with the associated text information in the original charts, are first extracted and identified with a multilayer perceptron classifier. Data information can abe retrieved by parsing visual marks with extracted mapping relationships.
- Then a 1-D convolutional residual network is employed to analyze the relationship between visual elements, and recognize significant features of the visualization charts, with both data and visual information as input.
- In the final step, the full description of the visual charts can be generated through a template-based approach.
- The generated captions can effectively cover the main visual features of the charts and support major feature types in commons charts

AutoCaption - Workflow

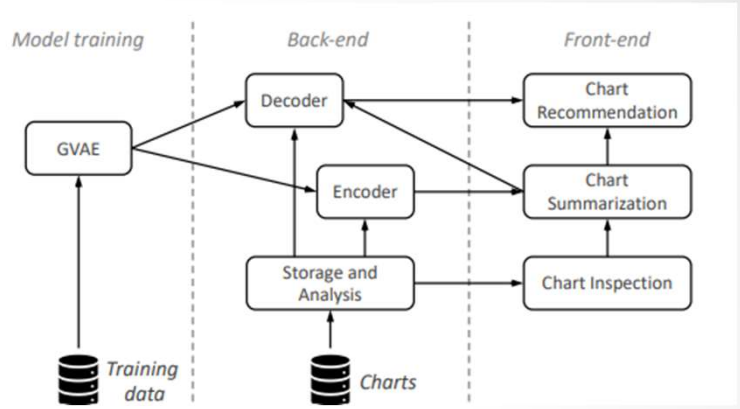


1. A feature extraction model takes the visual and data information as input and outputs the chart feature with type and grouped elements.
2. The chart feature type decides which summary template to use, while grouped elements decide the content of the filling components needed by the templates.
3. The caption with several descriptions is generated.

C. Liu, L. Xie, Y. Han, X. Yuan et al., "AutoCaption: An approach to generate natural language description from visualization automatically," in Proceedings of the IEEE Pacific Visualization Symposium (PacificVis). IEEE, 2020.

Chart Seer

- ChartSeer, a system that uses machine intelligence to enable analysts to visually monitor the current state of an Exploratory Visual Analysis and effectively identify future activities to perform.
- ChartSeer utilizes deep learning techniques to characterize analyst-created data charts to generate visual summaries and recommend appropriate charts for further exploration based on user interactions.
- ChartSeer leverages grammar variational autoencoders (GVAE) to obtain a mapping between charts and vectors in a semantic space and uses this mapping to create chart meta-visualizations and enable interactive recommendations.
  - It converts Vega-Lite specifications of charts into visualization embeddings by autoencoders.
  - The embeddings are used to measure similarities between charts to assist in analyzing chart ensembles



ChartSeer consists of a back-end that includes a pre-trained encoder and decoder and a front-end that has three coordinated views

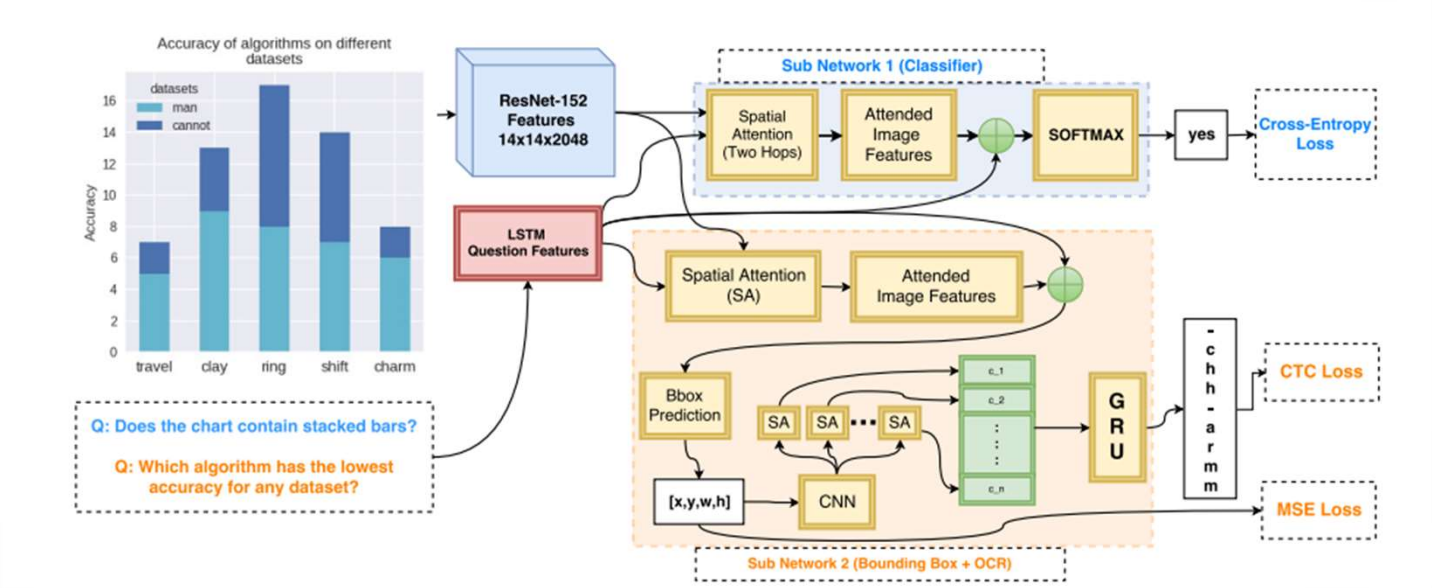
J. Zhao, M. Fan, and M. Feng, "Chartseer: Interactive steering exploratory visual analysis with machine intelligence," IEEE TVCG, 2020

Visual Qs Answering

- Visual question answering is an emerging research area that aims to answer a natural language question given a visualization image.
- Traditional methods utilize two steps:
  - decode visualizations into data tables
  - parse template-based questions into queries over the data table to generate answers
- End-to-end deep learning approaches have also been proposed.
- Some of the key challenges in Visual QA are:
  - Answering questions for visualizations requires high-level reasoning of which existing visual question answering models are not capable
  - Visualization images are sensitive to small local changes, i.e., shuffling the color in legends greatly alters the charts' information.
  - How to learn the features from visualization images and fuse them with features from natural language questions.
-

DVQA

- DVQA learns and fuses features via a sophisticated model containing multiple sub-networks, each responsible for different components such as spatial attention.
- DVQA combines a CNN-based and an LSTM-based network
- Given a visualization, the model reads the visualization content and answers reasoning questions such as "which item sold the most units?"



- Multi-Output Model (MOM) for DVQA: MOM uses two sub-networks: 1) classification sub-network that is responsible for generic answers, and 2) OCR sub-network that is responsible for chart-specific answers
- K. Kafle, B. Price, S. Cohen, and C. Kanan, "DVQA: Understanding data visualizations via question answering," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

6.2 Interpreting content

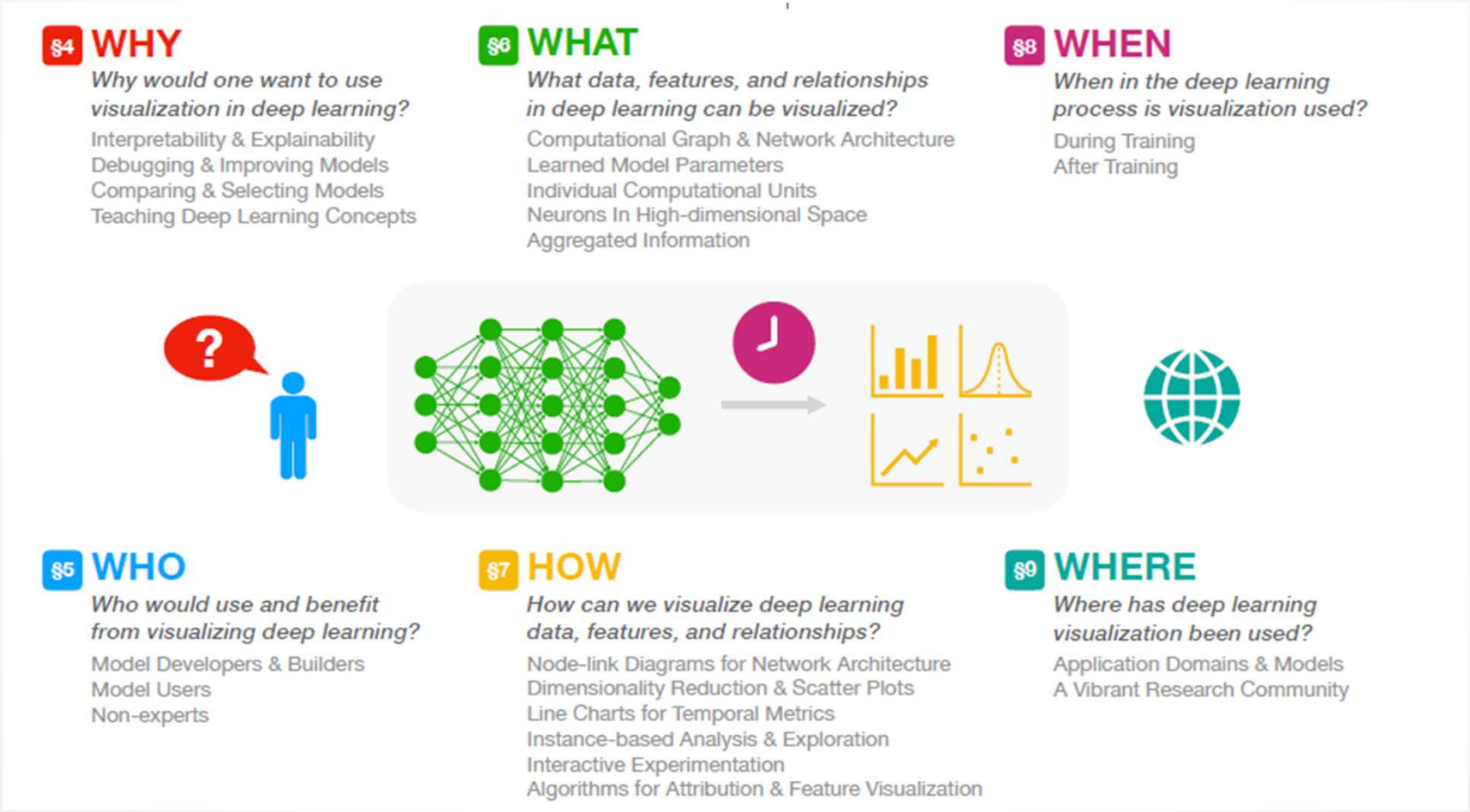
Overview

- Applying AI techniques in Information Visualization
- Visual Analytics In Deep Learning
- Applying Deep Learning techniques in Scientific Visualization

Introduction

- Deep learning has recently seen rapid development and received significant attention due to its state-of-the-art performance on previously-thought hard problems.
- However, because of the internal complexity and nonlinear structure of deep neural networks, the underlying decision making processes for why these models are achieving such performance are challenging and sometimes mystifying to interpret.
- As deep learning spreads across domains, it is of paramount importance that we equip users of deep learning with tools for understanding when a model works correctly, when it fails, and ultimately how to improve its performance.
- Standardized toolkits for building neural networks have helped democratize deep learning
- Visual analytics systems have now been developed to support model explanation, interpretation, debugging, and improvement

Visual Analytics in Deep Learning



What to visualize in Deep Learning

- Computational Graph & Network Architecture
- Learned Model Parameters
- Individual Computational Units
- Aggregated Information

Computational Graph & Network Architecture

- The first thing that can be visualized in a deep learning model is the model architecture.
- This includes the computational graph that defines how a neural network model would train, test, save data to disk, and checkpoint after epoch iterations.
- Also called the Dataflow graph,
- The computational graph can be visualized to potentially inform model developers of the types of computations occurring within their model

Learned Model Parameters: Neural Network Edge Weights

- Each node has an outgoing edge with an accompanying weight that sends signal from one neuron in a layer to potentially thousands of neurons in an adjacent layer
- These are the parameters that are tweaked during the backpropagation phase of training a deep model
- Visualizing the weights can help understanding what the model has learned

Learned Model Parameters: Convolutional Filters

- CNN convolutional layers apply filters over the input data, often times images represented as a two-dimensional matrix of values, to generate smaller representations of the data to pass to later layers in the network.
- These filters are then updated throughout the training process
- Therefore, visualizing the learned filters could be useful as an alternate explanation for what a model has learned

Individual Computational Units: Activations

- Throughout the network, the neurons compute activations using activation functions to combine the signal from the previous layer into a new node
- During inference, we can recover the activations produced at each layer.
- Although these feature representations are typically high dimensional vectors of the input data at a certain stage within the network , it could be valuable in helping people visualize how input data is transformed into features,.
- Feature representations may also shed light upon how the network and its components respond to particular data instances
- Moreover we can think of the feature vectors recovered as vectors in a high-dimensional space.
- Each neuron in a layer then becomes a dimension
- We can now take advantage of high dimensional visualization techniques to visualize extracted activations

Individual Computational Units: Gradients for Error Measurement

- Backpropagation, is a method to calculate the gradient of a specified loss function.
- When used in combination with an optimization algorithm, we can compute the error at the output layer of a neural network and redistribute the error by updating the model weights using the computed gradient.
- it could be useful to visualize the gradients of a network to see how much error is produced at certain outputs and where it is distributed

Aggregated Information: Groups of Instances

- Instance-level activations allow one to recover the mapping from data input to a feature vector output.
- This can also be done on collections of instances.
- Instance groups provide some unique advantages like computing all the activations simultaneously.
- Using visualization, one can compare these individual activations
- With instance groups, we can now take multiple groups, and compare how the distribution of activations from one group compares or differs from another.
- This aggregation of known instances into higher-level groups could be useful for uncovering the learned decision boundary in classification tasks

Aggregated Information: Model Metrics

- Model metrics, including loss, accuracy, and other measures of error are typically computed every epoch and represented as a time series over the course of a model's training phase.
- These metrics are key indicators for communicating how the network is progressing during the training phase.
- For example, is the network "learning" anything at all or is it learning "too much" and is simply memorizing data causing it to overfit?
- In the case of model comparison, these metrics become more important

How to Visualize Deep Learning

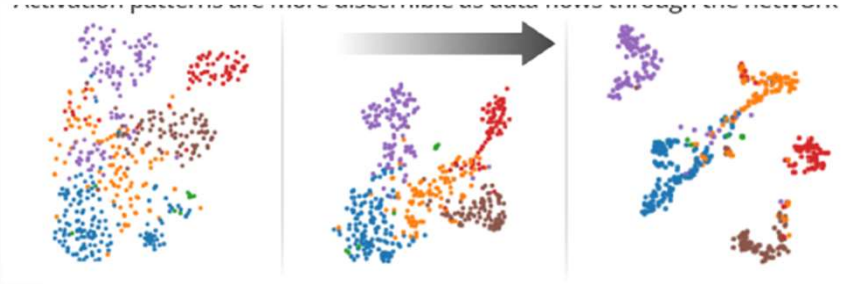
- Node-link Diagrams for Network Architectures
- Dimensionality Reduction & Scatter Plots
- Line Charts for Temporal Metrics
- Instance-based Analysis & Exploration
  - Identifying & Analysing Misclassified Instances
  - Analysing Groups of Instances
- Interactive Experimentation
  - Models Responding to User-provided Input Data
  - How Hyperparameters Affect Results
- Algorithms for Attribution & Feature Visualization
  - Heatmaps for Attribution, Attention, & Saliency
  - Feature Visualization

Node-link Diagrams for Network Architectures

- Neural network's dataflow graph can be visualized as a node-link diagram.
- Neurons are shown as nodes, and edge weights as links
- Magnitude and sign of the edge can then be encoded using color or link thickness.
- Example: K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Man´e, D. Fritz, D. Krishnan, F. B. Vi´egas, and M. Wattenberg, "Visualizing dataflow graphs of deep learning models in TensorFlow," IEEE TVCG, 2018.
- However displaying large numbers of links from complex models can generate cluttered visualizations where many edge crossings impede pattern discovery.
- Various solutions have been proposed:
  - Extracts high-degree nodes (responsible for many of the edge crossings), visualizes them separately from the main graph
  - Provide a quick snapshot of the dataflow in and out of a node by visualizing its activations within each node
  - Edge bundling

Dimensionality Reduction & Scatter Plots

- Neural Networks utilize different types of high-dimensional embeddings:
  - Text can be represented as vectors in word embeddings
  - Images can be represented as feature vectors
  - Mathematically represented as large tensors, or sometimes as 2D matrices, where each row may correspond to an instance and each column a feature.
- To visualize these embeddings dimensionality reduction techniques like PCA and tSNE is used o reduce the number of columns (e.g., features) to two or three.
- We can plot all data instances as points in a 2D or 3D scatter plot



Each point is a data instance's high dimensional activations at a particular layer inside of a neural network, dimensionally reduced, and plotted in 2D. Notice as the data flows through the network the activation patterns become more discernible.

M. Kahng, P. Andrews, A. Kalro, and D. H. Chau, "ActiVis: Visual exploration of industry-scale deep neural network models," IEEE TVCG, 2018.

Line Charts for Temporal Metrics

- Model developers track the progression of their deep learning models by monitoring
- Observe a number of different metrics computed after each epoch, including the loss, accuracy, and different measure of errors.
- This can be useful for diagnosing the long training process of deep learning models.,
- The most common visualization technique for visualizing this data is by considering the metrics as time series and plotting them in line charts. (Example: TensorFlow)

Instance-based Analysis: Identifying & Analyzing Misclassified Instances

- Another technique to help interpret and debug deep learning models is testing specific data instances to understand how they progress throughout a model.
- It is important to understand when a specific instance can fail and how it fails.
- By visualizing feature spaces of misclassified instances, we can gain a more intuitive understanding of recognition systems.
- Example:
  - HOGgles [C. Vondrick, et al ICCV, 2013], uses an algorithm to visualize feature spaces by using object detectors while inverting visual features back to natural images.
  - For textual data, a popular technique for analyzing particular data instances is to use color as the primary encoding.
  - For example, the background of particular characters in a phrase of words in a sentence would be colored using a divergent color scheme
  - This helps identify particular data instances that may warrant deeper inspection

Instance-based Analysis: Analyzing Groups of Instances

- Sometimes testing and debugging a model is done using groups of instances
- While some detail may be lost when performing grouplevel analysis it allows experts to further test the model by evaluating its average and aggregate performance across different groups
- Much of the work using this technique is done on text data using LSTM models
- Some approaches compute the saliency for groups of words across the model and visualize the values as a matrix
- Others use matrix visualizations to show the activations of word groups when represented as feature vectors in word embeddings
- ActiVis (M. Kahng, P. Andrews, A. Kalro, and D. H. Chau, "ActiVis: Visual exploration of industry-scale deep neural network models," IEEE TVCG, 2018) places instance group analysis at the focus of its interactive interface, allowing users to compare preset and user-defined groups of activations.
- However, sometimes it can be challenging to define groups for images or text.

Interactive Experimentation: Models Responding to User-provided Input Data

- To engage the user with the desired concepts to be taught, many systems require the user to provide some kind of input data into the system to obtain results.
- Some visual analytics systems use a webcam to capture live videos, and visualize how the internals of neural network models respond to these dynamic inputs [J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in ICML Deep Learning Workshop, 2015.].
- Another example is a 3D visualization of a CNN trained on the classic MNIST dataset that shows the convolution windows and activations on images that the user draws by hand [A. W. Harley, "An interactive node-link visualization of convolutionalneural networks," in ISVC, 2015].
- [Zhu, Krahenbuhl, Shechtman, and Efros, "Generative visual manipulation on the natural image manifold," in ECCV, 2016]uses GANs to interactively generate images based off of user's sketches. By sketch ing a few colored lines, the system presents the user with multiple synthetic images using the sketch as a guideline for what to generate..

Interactive Experimentation: Models Responding to User-provided Input Data

- ShapeShop [F. Hohman, N. Hodas, and D. H. Chau in CHI, Extended Abstracts, 2017]
  - Allows a user to select data from a bank of simple shapes to be classified.
  - The system then trains a neural network and using the class activation maximization technique to generate visualizations of the learned features of the model.
  - This can be done in realtime; therefore a user can quickly train multiple models with different shapes to observe the effect of adding more diverse data to improve the internal model representation
- Adversarial Playground [A. P. Norton and Y. Qi, "Adversarial-Playground: A visualization suite showing how adversarial examples fool deep learning," in VizSec. IEEE, 2017]
  - A visual analytics system that enables users to compare adversarially-perturbed images, to help users understand why an adversarial example can fool a CNN image classifier.
  - The user can select from one of the MNIST digits and adjust the strength of adversarial attack.
  - The system then compares the classifications scores in a bar chart to observe how simple perturbations can greatly impact classification accuracy.

Interactive Experimentation: How Hyperparameters Affect Results

- In deep learning models hyperparameters require finetuning.
- These hyperparameters can have major impact on model performance and robustness.
- Some visual analytics systems expose model hyperparameters to the user for interactive experimentation.
- One example is TensorFlow Playground [D. Smilkov, S. Carter, D. Sculley, F. B. Viegas, and M. Wattenberg, "Direct-manipulation visualization of deep networks," in ICML Workshop on Vis for Deep Learning, 2016] where users can use direct manipulation to adjust the architecture of a simple, fully connected neural network, as well as the hyperparameters associated with its training.
- Another example is a [M. Wattenberg, F. Vigas, and I. Johnson, "How to use t-SNE effectively," Distill, 2016.] that meticulously explores the hyper parameters of the t-SNE dimensionality reduction method.

Heatmaps for Attribution, Attention, & Saliency

- Some systems generate translucent heatmaps that overlay images to highlight important regions that contribute towards classification and their sensitivity [Example: B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in CVPR, 2016].
- A technique called visual backpropagation attempts to visualize which parts of an image have contributed to the classification, and can do so in real-time in a model debugging tool for self-driving vehicles [M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, and K. Zieba, "Visualbackprop: visualizing cnns for autonomous driving," arXiv:1611.05418, 2016]
- Another technique is to invert representations, i.e., attempt to reconstruct an image using a feature vector to understand the what a CNN has learned [A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in CVPR, 2016]
- Visualizing CNN filters is also popular, and has famously shown to generate dream-like images, becoming popular in artistic tasks [A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," Google Research Blog, 2015]

Feature Visualization

- For feature visualization one of the most studied techniques, class activation maximization, maximizes the activation of a chosen, specific neuron using an optimization scheme, such as gradient ascent, and generates synthetic images that are representative of what the model has learned about the chosen class [D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," University of Montreal, 2009.].
- This led to a number of works improving the quality of the generated images.
- Ngyuen et al. ["Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in NIPS, 2016] present hundreds of high-quality images using a deep generator network to improve upon the state-of-the-art
- A recent comparison of feature visualization techniques highlights their usefulness [C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," Distill, 2017]

Application Domains

- Autonomous Vehicles [M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, and K. Zieba, "Visualbackprop: visualizing cnns for autonomous driving," arXiv:1611.05418, 2016]
- Social Good [C. Robinson, F. Hohman, and B. Dilkina, "A deep learning approach for population estimation from satellite imagery," in SIGSPATIAL Workshop on Geospatial Humanities, 2017]
- Reinforcement Learning T. Zahavy, N. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding DQNs," in ICML, 2016]
- Medical Imaging Diagnostics [L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," arXiv:1702.04595, 2017.]
- Urban Planning [L. Li, J. Tompkin, P. Michalatos, and H. Pfister, "Hierarchical visual feature analysis for city street view datasets," in Workshop on Visual Analytics for Deep Learning, 2017]

Neural Network Models studies for Visual Analytics

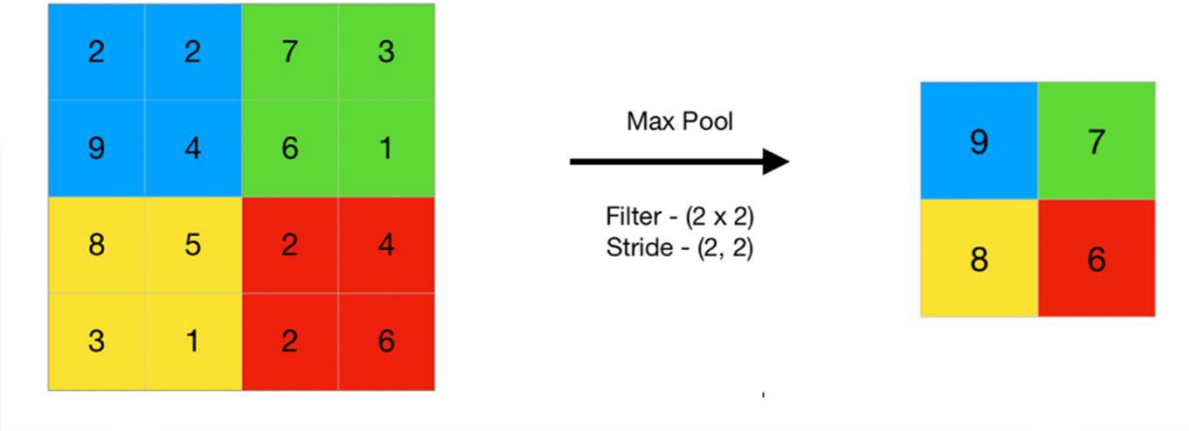
- Convolutional Neural Network (CNN):
  - Much of the existing work has used image-based data and models, namely CNNs, to generate attribution and feature visualization explanations for what a model has learned from an image dataset.
  - Example: M. D. Zeiler and R. Fergus, "Visualizing and understandingconvolutional networks," in ECCV, 2014
- Generative Adversarial Network (GAN):
  - Several systems focus on understanding the training dynamics of GANs to help model developers better train these complex models, often consisting of multiple dueling neural networks.
  - Example: J. Wang, L. Gou, H. Yang, and H.-W. Shen, "GANViz: A visual analytics approach to understand the adversarial game," IEEE TVCG., 2018
- Graph Neural Network (GNN):
  - The most prevalent current approach for GNN model evaluation is to compute quantitative metrics based on the downstream ML which are used for cross-validation during training. Some Visual Analytics systems are also being developed to aid in model evaluation.
  - Example: Zipeng Liu, Yang Wang, Jurgen Bernard, Tamara Munzner. "Visualizing Graph Neural Networks with CorGIE: Corresponding a Graph to Its Embedding", IEEE TVCG, 2022
- NLP Models:
  - For learning representations of large text corpora interactive tools that support dimensionality reduction techniques to solve problems such as sequence-to-sequence conversion, translation, and audio recognition have been developed.
  - Example: Li, Wang, Yang, Wu, Zhang, Liu, Sun, Zhang, Liu. "A Unified Understanding of Deep NLP Models for Text Classification", IEEE Transactions on Visualization and Computer Graphics, 2022

Visualizing and Understanding Convolutional Networks

- This seminal work is often credited for popularizing visualization in the machine learning and computer vision communities, putting a spotlight on it as a powerful tool that helps people understand and improvedeep learning models.

Convent

- The system uses standard fully supervised CNN (convnet) models
- These models map a color 2D input image via a series of layers, to a probability over different classes.
- Each layer consists of:
  - Convolution of the previous layer output (or, in the case of the 1st layer, the input image) with a set of learned filters
  - Passing the responses through a rectified linear function
  - [optionally] Max pooling over local neighborhoods
  - [optionally] A local contrast operation that normalizes the responses across feature maps.



Visualizing and Understanding Convolutional Networks

Deconvnet

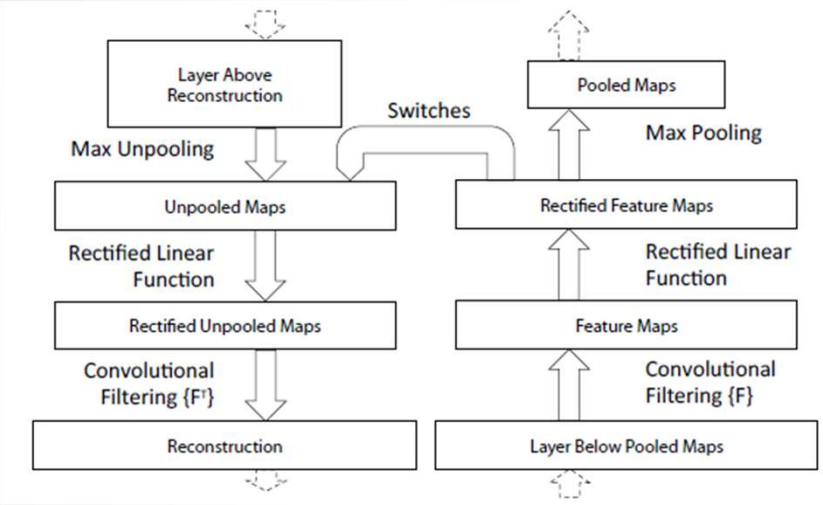
- Understanding the operation of a convnet requires interpreting the feature activity in intermediate layers.
- This system maps these activities back to the input pixel space, showing what input pattern originally caused a given activation in the feature maps.
- This mapping is done with a Deconvolutional Network (deconvnet)
  - A deconvnet can be thought of as a convnet model that uses the same components but in reverse, so instead of mapping pixels to features does the opposite.
- To examine a convnet, a deconvnet is attached to each of its layers providing a continuous path back to image pixels.
- To start, an input image is presented to the convnet and features computed throughout the layers.
- To examine a given convnet activation, all other activations in the layer are set to zero and pass the feature maps as input to the attached deconvnet layer.
- Then successively (i) unpool, (ii) rectify and (iii) filter to reconstruct the activity in the layer beneath that gave rise to the chosen activation.
- This is then repeated until input pixel space is reached.

Visualizing and Understanding Convolutional Networks

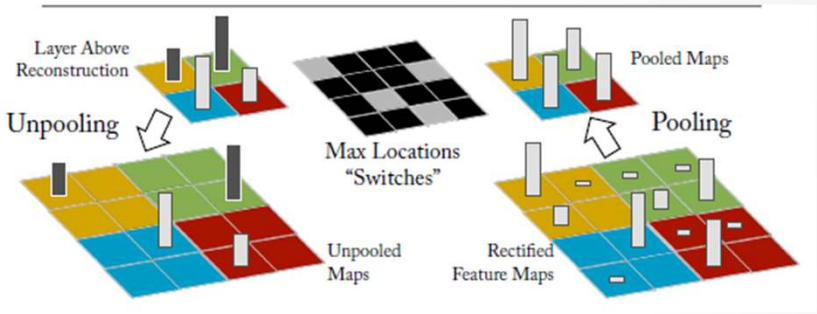
Unpooling

- In the convnet, the max pooling operation is non-invertible
- However we can obtain an approximate inverse by recording the locations of the maxima within each pooling region in a set of switch variables.
- In the deconvnet, the unpooling operation uses these switches to place the reconstructions from the layer above into appropriate locations, preserving the structure of the stimulus.
- Projecting down from higher layers uses the switch settings generated by the max pooling in the convnet on the way up.
- As these switch settings are peculiar to a given input image, the reconstruction obtained from a single activation thus resembles a small piece of the original input image, with structures weighted according to their contribution toward to the feature activation.
- Since the model is trained discriminatively, they implicitly show which parts of the input image are discriminative.

Visualizing and Understanding Convolutional Networks



A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath.

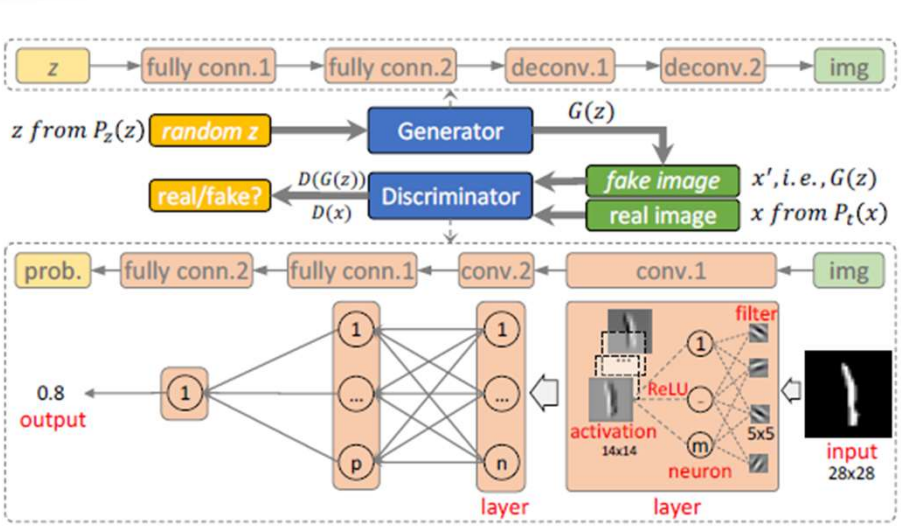


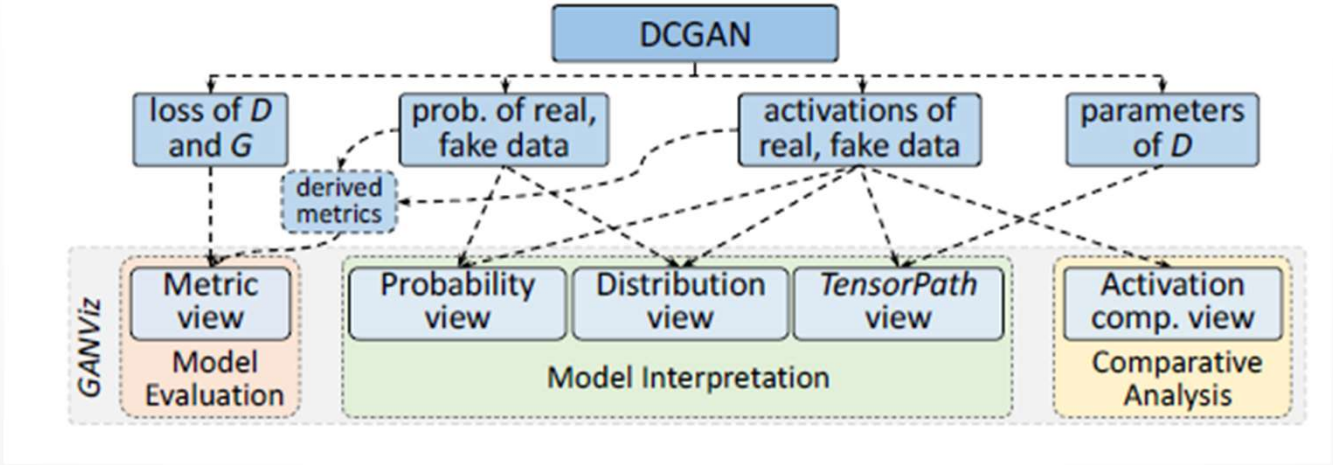
An illustration of the unpooling operation in the deconvnet, using switches which record the location of the local max in each pooling region (colored zones) during pooling in the convnet.

GANViz

- Generative models bear promising implications to learn data representations in an unsupervised fashion with deeplearning.
- Generative Adversarial Nets (GAN) is one of the most popular frameworks in this arena.
- Despite the promising results from different types of GANs, in-depth understanding on the adversarial training process of the models remains a challenge to domain experts.
- The complexity and the potential long-time training process of the models make it hard to evaluate, interpret, and optimize them.
- GANViz is a visual analytics system utilizing the popular GAN model, DCGAN
- The system helps domain experts understand the adversarial process of GANs in-depth.
- Specifically, GANViz evaluates the model performance of two subnetworks of GANs, provides evidence and interpretations of the models' performance, and empowers comparative analysis with the evidence.

DCGAN & GANViz Architectures





GANViz – Modules and Components

- The components of GANViz are organized into three modules
- In the three modules, five views are designed to analyze different data in a top-down style.

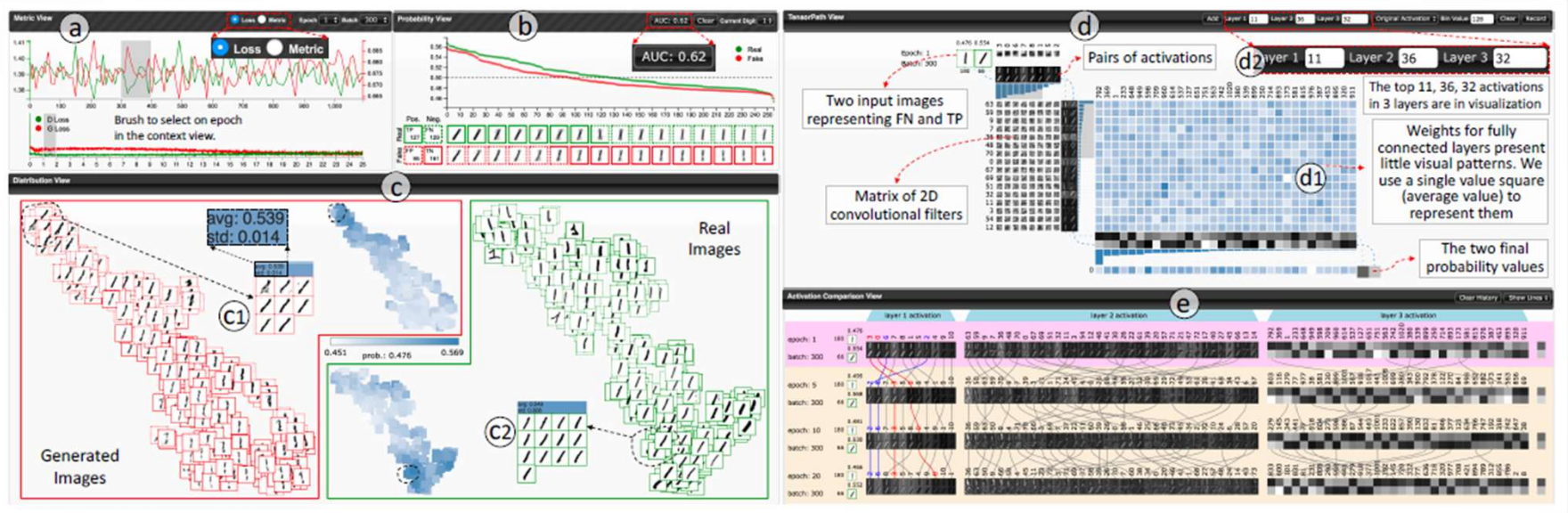
I. Model Evaluation Module
  - The metric view provides an overview of model dynamics in the training process
  - It helps users understand the overall quality by visualizing various model metrics

II. Model Interpretation Module
  - In this module probability view, distribution view, and TensorPath view enable users to examine and interpret the model at a specific training stage of their interest
  - These components offer rich evidence and interpretations of the model performance, such as the decision features of D, feature patterns of both real and fake data, neural network structures, and associated decision critical data in the structures (e.g., neurons, activations, parameters, etc.).

III. Comparative analysis Modules
  - Over the journey of exploration, the activation comparison view supports users to collect relevant model information and conduct comparative analysis
  - It allows users to understand how the decision-critical data are evolved and how they work for the data from different sources (e.g., fake/real, true/false predictions).

- The five views are coordinated with each other, and rich interactions (e.g., brushing, filtering, selection, etc.) are provided to link the views for flexible exploration.

GANViz



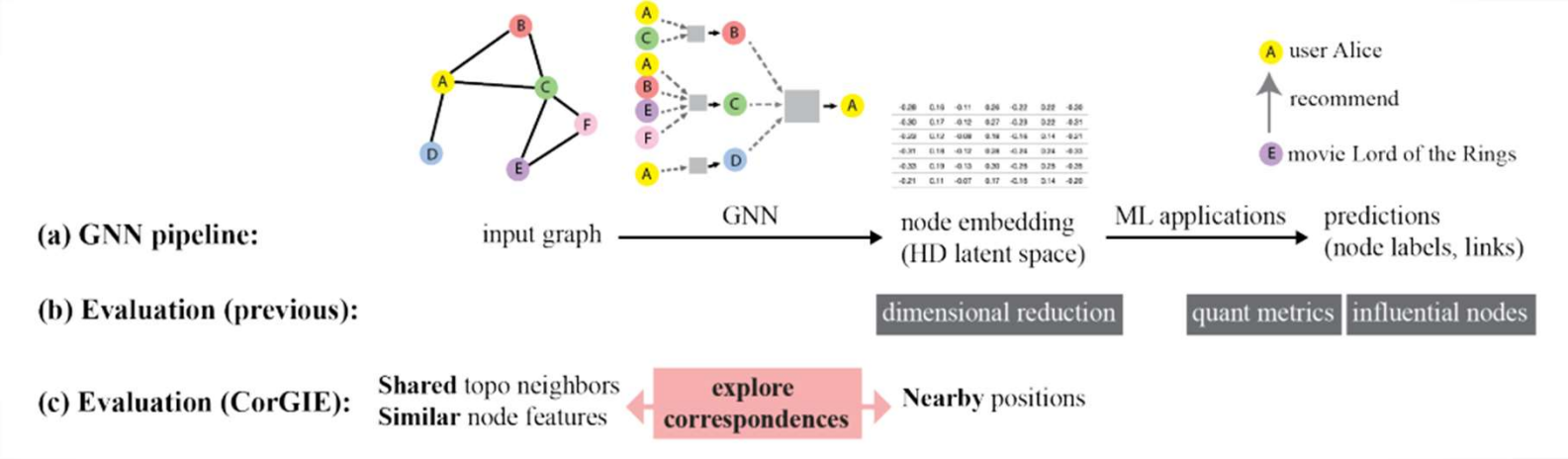
a) The metric view with focus+context support
b) The probability view shows the probability of real/fake images in a decreasing order
c) The distribution view reveals the feature distributions of real/fake images
d) The TensorPath view highlights important activations, filtersand weights of D
e) The activation comparison view tracks the changes among multiple pairs of images from different sources or timestamps

CorGIE

- During training, a GNN model takes in a node-link graph as input and as output generates a node embedding; that is, a representation of all discrete nodes in the graph as fixed dimensional vectors in a continuous latent space.
- Proximities between embedded nodes in the latent space represent meaningful similarities between nodes in the input graph learned during the training.
- The node embedding is then available for feeding into downstream ML applications, for example to make predictions about nodes and links
- The most prevalent current approach for model evaluation is to compute quantitative metrics based on the downstream ML applications
- However, these existing evaluation approaches fall short because they do not sufficiently support developers in directly understanding the correspondences between input graphs and their node embeddings.

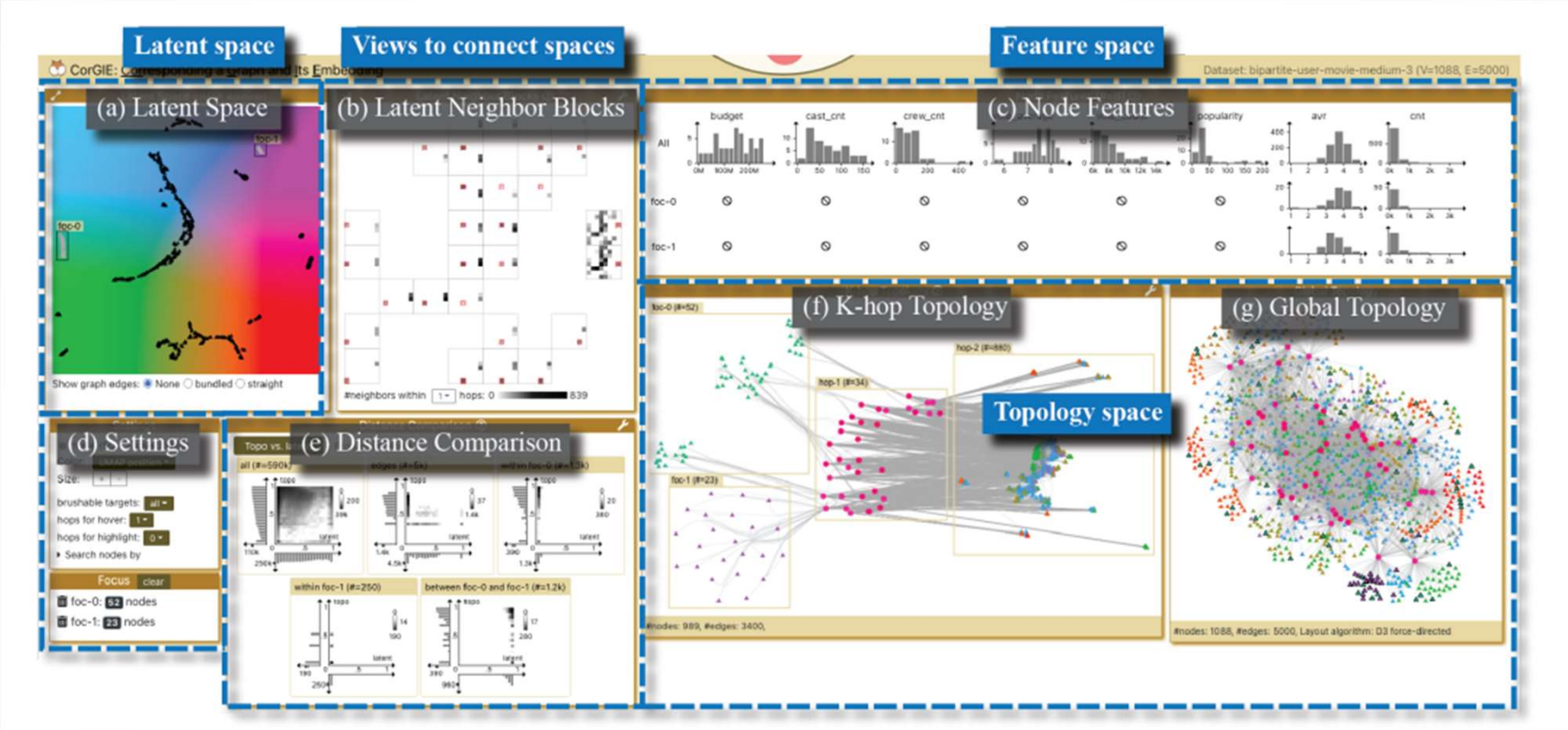
CorGIE

- CorGIE (Corresponding a Graph to Its Embedding) is an interactive multi-view interface to support exploration of correspondences between an input graph and its node embedding



(a) Standard pipeline for graph neural network (GNN)
(b) Previous approaches to evaluate GNNs mostly focus on the predictions or to separately inspect the node embedding
(c) Approach with CorGIE is to directly support exploration of correspondences between an input graph and its node embedding.

CorGIE



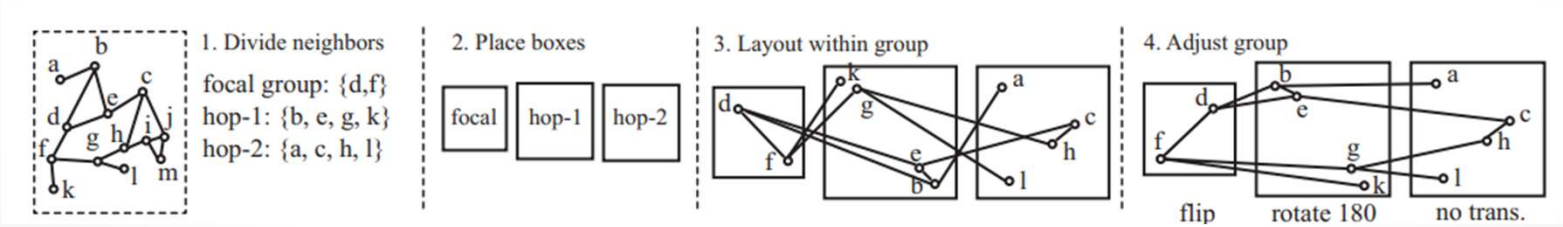
Screenshot of CorGIE interface

(a) The LATENT SPACE VIEW is for 2D node positions in latent space
(b) The LATENT NEIGHBOR BLOCKS VIEW
(c) The NODE FEATURES VIEW is for feature distribution of all and focal nodes
(d) The SETTINGS VIEW: For toggles and menus
(e) DISTANCE COMPARISON VIEW connect different spaces
(f) The TOPOLOGY VIEW showing local topological neighbors
(g) The TOPOLOGY VIEW showing global topological neighbors

K-Hop Layout

- CorGIE uses a new visual encoding technique, the K-hop graph layout, which is used in the TOPOLOGY VIEW.
- The K-hop layout aims to organize topological neighbors of user-specified nodes by the number of hops and their clustering structure.
- It is computed in four main steps:

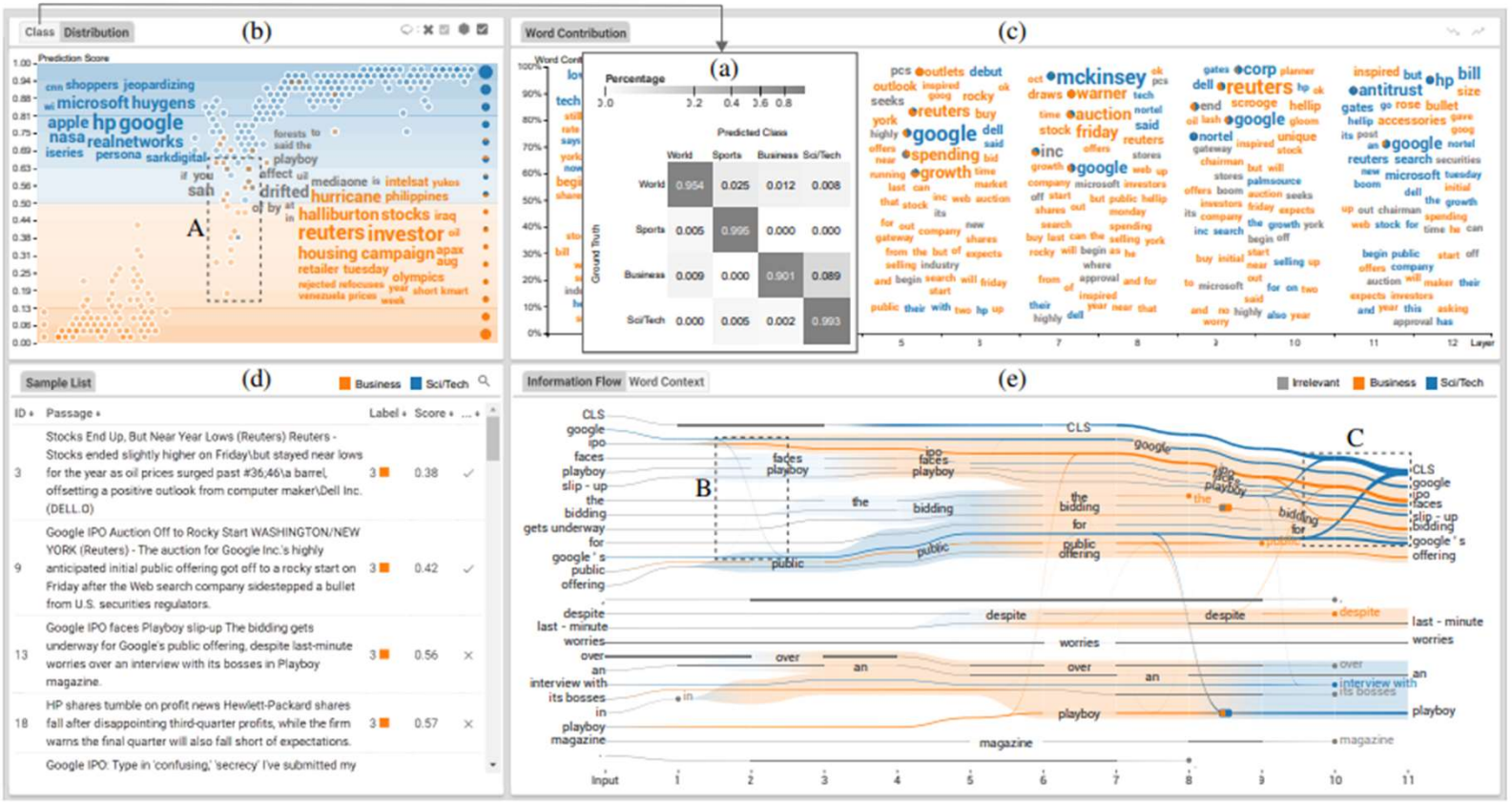
1. divide relevant nodes into groups
2. bound nodes within boxes and lay out the boxes
3. lay out nodes within each box independently
4. perform transformations to optimize global
5. edge bundling (optional)



DeepNLPVis

- DeepNLPVis is an interactive visual analysis tool to help model developers gain a unified understanding of different NLP models for text classification, quickly identify problems, and make informed improvements.
- DeepNLPVis adopts a coordinated multiple-level visualization connecting the analysis from the overall training corpus to individual samples and words.
- At the corpus level, a class view shows the overall model performance on all classes, from which the user can select two classes to explore the corpus against predictions on the two classes in the distribution view.
- The sample-level visualization supports analyzing a sample in terms of both intra-word and inter-word information.
- The word-level visualization supports examining words in terms of word contribution and word meaning.
- The interactive analysis enabled by the coordinated visualization helps users explore model deficiencies and identify the root cause of low performance

DeepNLPVis



DeepNLPVis for analyzing the BERT model on news classification:

(a) Class view for showing the overall model performance;

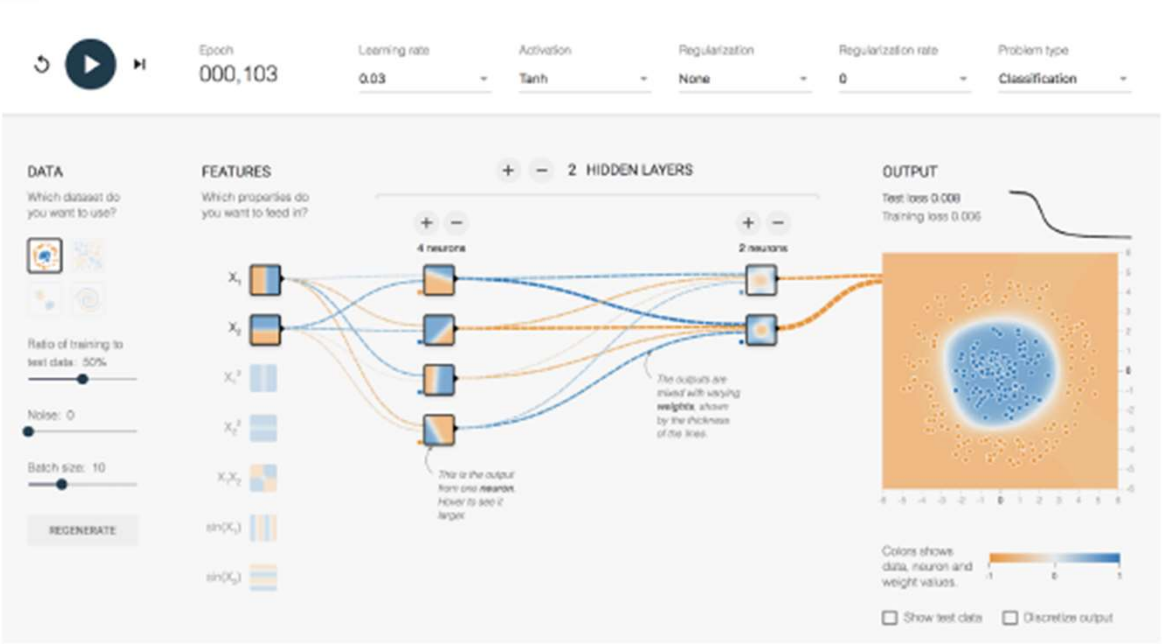
(b) Distribution view for identifying samples and words of interest;

(c) Word contribution of selected samples;

(d) Sample list

(e) Information flow for analyzing a sample by its intra- and inter-word information

TensorFlow Playground



A web-based visual analytics tool for exploring simple neural networks that uses direct manipulation rather than programming to teach deep learning concepts and develop an intuition about how neural networks behave.

https://playground.tensorflow.org/

TensorBoard

- TensorBoard provides the visualization and tooling needed for machine learning experimentation:
  - Tracking and visualizing metrics such as loss and accuracy
  - Visualizing the model graph (ops and layers)
  - Viewing histograms of weights, biases, or other tensors as they change over time
  - Projecting embeddings to a lower dimensional space
  - Displaying images, text, and audio data
  - Profiling TensorFlow programs
- https://www.tensorflow.org/tensorboard

Overview

- Applying AI techniques in Information Visualization
- Visual Analytics In Deep Learning
- Applying Deep Learning techniques in Scientific Visualization

Research Tasks

- Deep Learning techniques have been applied to various aspects of Scientific Visualization
  I. Data Generation
  II. Visualization Generation
  III. Prediction: Data-relevant & Visualization-relevant
  IV. Object Detection and Segmentation
  V. Feature Learning and Extraction

I. Data Generation

- Includes the following tasks:
  i. Super-resolution
  ii. Compression and reconstruction
  iii. Translation
  iv. Extrapolation

i. Super-resolution

- A class of techniques that aim to enhance or increase the image, video, or volumetric data resolution.
- In SciVis, the resolution encompasses the three spatial dimensions and the temporal dimension.
- Due to the limited storage space, generating super-resolution data from their low-resolution counterparts brings the immediate benefit of storage-saving via data reduction.
- This is because only the low-resolution data and the trained network model need to be stored to recover the super-resolution data.
- DL-based super-resolution techniques thus provide domain scientists an alternative to manage their simulation data cost-effectively
- Examples:
  - Scalar field data: Z. Zhou, Y. Hou, Q. Wang, G. Chen, J. Lu, et al. Volume upscaling with convolutional neural networks. In Proceedings of Computer Graphics International, 2017
  - Vector field data: L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, et al. SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In Proceedings of IEEE Pacific Visualization Symposium, 2020. [48]
  - Fluid simulation: Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. ACM Transactions on Graphics, 2018.

ii. Compression & Reconstruction

- Compression Example:
  - [Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. Computer Graphics Forum, 2021] Introduced neurcomp, a coordinate-based MLP for compressive neural representations of scalar field volume data.
  - Once learned, the network itself becomes the compressed representation of the underlying data.
  - By quantizing network weights, neurcomp could achieve an impressive CR over 1,000× while preserving important volumetric feature
- Reconstruction Example:
  - [Y. Wang, Z. Zhong, and J. Hua. DeepOrganNet: On-the-fly reconstruction and visualization of 3D/4D lung models from single-view projections by deep deformation network. IEEE Transactions on Visualization and Computer Graphics, 2020] designed DeepOrganNet that reconstructs 3D/4D lung models from single-view CT projections or X-ray images.
  - DeepOrganNet can reconstruct manifold meshes of lung models in high quality and high fidelity

iii. Translation

- DL-based data translation has been used for various translation tasks like translating from one variable sequence to another or from an input field to another that satisfies specific properties.
- Example:
  - [P. Gu, J. Han, D. Z. Chen, and C. Wang. Scalar2Vec: Translating scalar fields to vector fields via deep learning. In Proceedings of IEEE Pacific Visualization Symposium, 2022] presented Scalar2Vec that translates scalar fields to vector fields via DL
  - They picked suitable scalar variables for the translation.
  - A CNN-based network takes a set of sampled scalar field volumes as input and extracts their multi-scale information to synthesize the corresponding vector field volumes

iv. Extrapolation

- Extrapolation aims to generate historical or future data values based on the current values.
- Example:
  - For fluid simulation, [S. Wiewel, M. Becher, and N. Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. Computer Graphics Forum, 2019] presented latent space physics (LSP), an LSTM-CNN hybrid approach to predict the changes of pressure fields over time for fluid flow simulation.
  - LSP can achieve 150× speedups compared with a regular pressure solver, a significant boost in simulation performance.

II. Visualization Generation

- Deep Learning has been used in visualization generation for direct volume rendering or isosurface rendering.
- Example: M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. IEEE Transactions on Visualization and Computer Graphics, 2019.
  - Presented a generative model for volume rendering where a GAN was trained on a large collection of volume rendering images under different viewpoints and transfer functions.
  - Once trained, the model can infer novel rendering conditioned on new viewpoints and transfer functions without following the traditional rendering pipeline.
- Example: W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, et al. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. IEEE Transactions on Visualization and Computer Graphics, 2020
  - Developed InSituNet for parameter-space exploration of ensemble simulations. The training data (i.e., visualization images conditioned on visual mappings and view parameters) were collected in situ.
  - Then, they trained a convolutional regression model offline that learns the mapping from simulation parameters to visualization outputs.
  - The trained model supports interactive post hoc exploration and analysis by synthesizing images from novel parameter settings.

III. Data-relevant Prediction

- Example: W. He, J. Wang, H. Guo, H.-W. Shen, and T. Peterka. CECAVDNN: Collective ensemble comparison and visualization using deep neural networks. Visual Informatics, 2020.
  - For scalar field data
  - Designed CECAV-DNN that predicts ensemble similarity for collective ensemble comparison and visualization (CECAV).
  - Given a sequence of ensemble pairs (each ensemble is a collection of scalar fields), they trained the DNN to assign a likelihood score to each scalar field, indicating the probability that the field is from one ensemble rather than the other.
  - After training, three levels of comparison: dimensionality comparison, member comparison, and region comparison, are provided for ensemble comparison and visualization.
- Example: F. Hong, J. Zhang, and X. Yuan. Access pattern learning with long short-term memory for parallel particle tracing. In Proceedings of IEEE Pacific Visualization Symposium, 2018
  - For vector field data,
  - Aimed to predict the access pattern for parallel particle tracing.
  - Their LSTM based model learns the access pattern from a small set of pathline samples.
  - Such prediction results can assist workload balancing by prefetching data blocks to reduce I/O costs and improve time efficiency.

IV. Visualization-relevant Prediction

- Estimating visualization-related quality or parameters
- Example: C. Yang, Y. Li, C. Liu, and X. Yuan. Deep learning-based viewpoint recommendation in volume visualization. Journal of Visualization, 2019.
  - Designed a CNN-based model to estimate the viewpoint quality given a volume rendering image.
  - The aim is to mimic the traditional scoring method and user preference and predict viewpoint quality close to human judgment.

V. Object Detection & Segmentation

- For object detection and segmentation systems extensively utilize U-Net [O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015], initially designed for biomedical image segmentation
- Beyond U-Net, researchers also explored the use of GNN for volume classification.
- [X. He, S. Yang, Y. Tao, H. Dai, and H. Lin. Graph convolutional network-based semi-supervised feature classification of volumes. Journal of Visualization, 2022]
  - Generated a super voxel graph from a scalar volumetric dataset where a node represents a super-voxel (i.e., a group of voxels with similar spatial locations and properties), and an edge represents the neighborhood relation between the corresponding super voxels.
  - They then utilized a GCN to learn node embedding.
  - Finally, the output of the GCN goes through an MLP to predict the label of each node for volume classification.

VI. Feature Learning & Extraction

- For scalar field data, many solutions use CNN-based neural networks for feature learning and extraction
- [H.-C. Cheng, A. Cardone, S. Jain, E. Krokos, K. Narayan, et al. Deep-learning-assisted volume visualization. IEEE Transactions on Visualization and Computer Graphics, 2019]
  - Applied a pre-trained CNN to learn voxel neighborhood information.
  - They then employed vector quantization to the high-level features extracted from volume patches to generate the characteristic feature vector to support the hierarchical exploration of complex volumetric structures
- Beyond CNNs, researchers have also investigated GNN based solutions for feature learning and extraction.
- X. He, Y. Tao, S. Yang, C. Chen, and H. Lin. ScalarGCN: Scalarvalue association analysis of volumes based on graph convolutional network. Journal of Visualization, 2022
  - Designed ScalarGCN, a GNN-based solution for scalar value association analysis of volumes.
  - ScalarGCN aims to learn the high-order topological structural relationships of multiple variables using a multilayer GCN with the self-attention mechanism.
  - The input to ScalarGCN is ScalarGraph, where nodes represent sampled scalar values from multivariate data, and edges encode local (within the same variable) and global (across different variables) connections.
  - Node features consider context, spatial, and gradient distributions.
  - ScalarGCN performs local learning for node embedding and global learning for variable embedding

Readings

- Aoyu Wu, Yun Wang, Xinhuan Shu, Dominik Moritz, Weiwei Cui, Haidong Zhang, Dongmei Zhang, Huamin Qu. AI4VIS: Survey on Artificial Intelligence Approaches for Data Visualization. InfoVis 2022.
- Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. A Survey on ML4VIS: Applying Machine Learning Advances to Data Visualization. InfoVis 2022.
- F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," IEEE Transactions on Visualization and Computer Graphics (TVCG), 2019
- Chaoli Wang and Jun Han DL4SciVis: A State-of-the-Art Survey on Deep Learning for Scientific Visualization. InfoVis 2022.