Interpretable components and graph neural networks

Arkadiusz Tomczyk
arkadiusz.tomczyk@p.lodz.pl

Institute of Information Technology
Lodz University of Technology
Poland

01.2024

Graph neural networks

Machine learning

- application of machine laerning should allow computers to operate on real objects $\mathcal{O}$ and lead to answers, which are easily understandable by humans

Machine learning

- application of machine laerning should allow computers to operate on real objects $\mathcal{O}$ and lead to answers, which are easily understandable by humans
- differnt tasks can be solved in this way

Prediction

$$f \in \Lambda(\mathcal{O}, \{1, \ldots, L\})$$

Metric learning

$$f \in \Lambda(\mathcal{O} \times \mathcal{O}, \mathbb{R})$$

Machine learning

- application of machine laerning should allow computers to operate on real objects $\mathcal{O}$ and lead to answers, which are easily understandable by humans
- differnt tasks can be solved in this way

Prediction

$$f \in \Lambda(\mathcal{O}, \mathbb{R})$$

Metric learning

$$f \in \Lambda(\mathcal{O} \times \mathcal{O}, \mathbb{R})$$

Machine learning

- application of machine laerning should allow computers to operate on real objects $\mathcal{O}$ and lead to answers, which are easily understandable by humans
- differnt tasks can be solved in this way

Prediction

$$f \in \Lambda\left(\mathbb{R}^N, \mathbb{R}^M\right)$$

Metric learning

$$f \in \Lambda\left(\mathbb{R}^N \times \mathbb{R}^N, \mathbb{R}^M\right)$$

- naturally computers cannot operate on real objects and concepts, so they need to be properly encoded $\Lambda\left(\mathcal{O}, \mathbb{R}^N\right)$ (feature extraction) and model outputs need to be decoded $\Lambda\left(\mathbb{R}^M, \{1, \ldots, L\}\right)$ or $\Lambda\left(\mathbb{R}^M, \mathbb{R}\right)$

Representation

- there are many models (trainable or not) that solve typical tasks, but manually extracted features $\mathbb{R}^N$ usually do not satisfy their requirements

Representation

- there are many models (trainable or not) that solve typical tasks, but manually extracted features $\mathbb{R}^N$ usually do not satisfy their requirements

Prediction

Linear classifiers can be applied in $\mathbb{R}^N$ only if feature vectors are in fact linearly separable in that feature space.

Metric learning

Having features in $\mathbb{R}^N$ one can use Euclidean metric only if feature vectors reflect that kind of distance between objects.

Representation

- there are many models (trainable or not) that solve typical tasks, but manually extracted features $\mathbb{R}^N$ usually do not satisfy their requirements

Prediction

Linear regression can be applied in $\mathbb{R}^N$ only if there is linear relationship between feature vectors and predicted variable.

Metric learning

Having features in $\mathbb{R}^N$ one can use Euclidean metric only if feature vectors reflect that kind of distance between objects.

Representation

- there are many models (trainable or not) that solve typical tasks, but manually extracted features $\mathbb{R}^N$ usually do not satisfy their requirements

Prediction

Linear regression can be applied in $\mathbb{R}^N$ only if there is linear relationship between feature vectors and predicted variable.

Metric learning

Having features in $\mathbb{R}^N$ one can use Euclidean metric only if feature vectors reflect that kind of distance between objects.

- that is why additional, representation (embedding) module $f_1 \in \Lambda\left(\mathbb{R}^N, \mathbb{R}^K\right)$ is required to adjust the extracted features

Representation

- there are many models (trainable or not) that solve typical tasks, but manually extracted features $\mathbb{R}^N$ usually do not satisfy their requirements

Prediction

Linear regression can be applied in $\mathbb{R}^N$ only if there is linear relationship between feature vectors and predicted variable.

Metric learning

Having features in $\mathbb{R}^N$ one can use Euclidean metric only if feature vectors reflect that kind of distance between objects.

- that is why additional, representation (embedding) module $f_1 \in \Lambda\left(\mathbb{R}^N, \mathbb{R}^K\right)$ is required to adjust the extracted features
- consequently mentioned above models operate in embedding space $f_2 \in \Lambda\left(\mathbb{R}^K, \mathbb{R}^M\right)$

Representation

- there are many models (trainable or not) that solve typical tasks, but manually extracted features $\mathbb{R}^N$ usually do not satisfy their requirements

Prediction

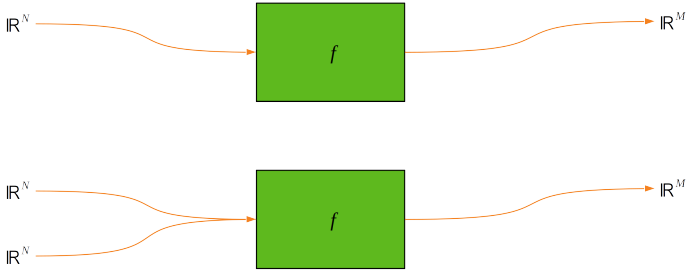
Linear regression can be applied in $\mathbb{R}^N$ only if there is linear relationship between feature vectors and predicted variable.

Metric learning

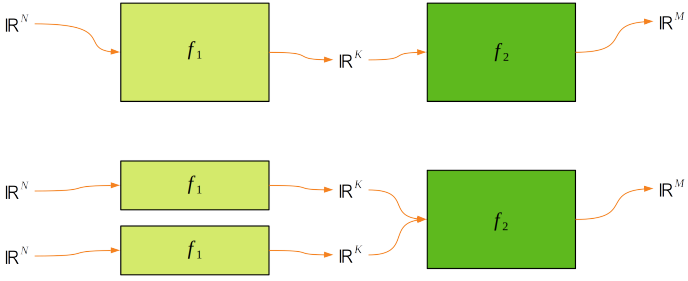
Having features in $\mathbb{R}^N$ one can use Euclidean metric only if feature vectors reflect that kind of distance between objects.

- that is why additional, representation (embedding) module $f_1 \in \Lambda\left(\mathbb{R}^N, \mathbb{R}^K\right)$ is required to adjust the extracted features
- consequently mentioned above models operate in embedding space $f_2 \in \Lambda\left(\mathbb{R}^K, \mathbb{R}^M\right)$
- reresentation (embedding) module is usually trainable

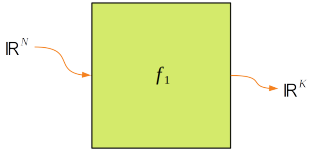
Representation



Representation

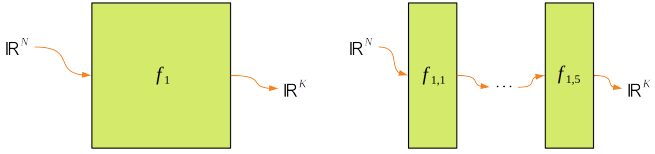


Representation



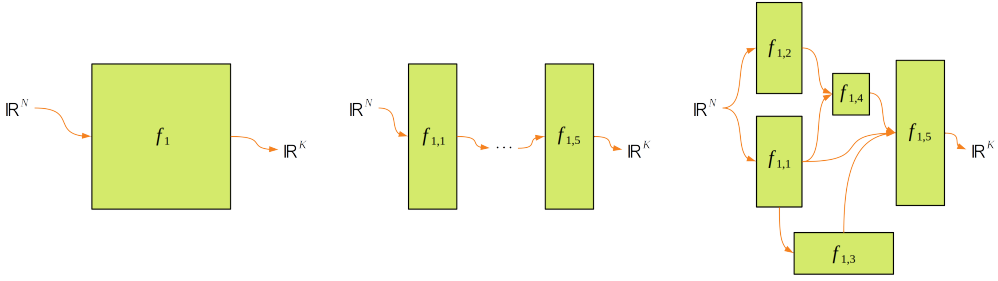
- since models solving typical tasks are known (predictors, metrics), in practice we usually focus only on representation learning

Representation



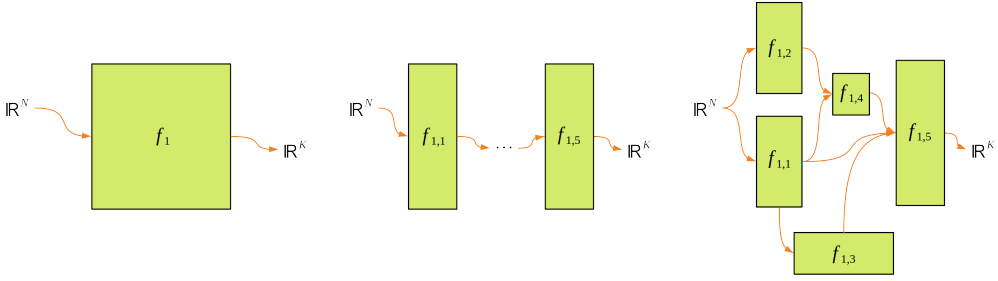
- since models solving typical tasks are known (predictors, metrics), in practice we usually focus only on representation learning

Representation



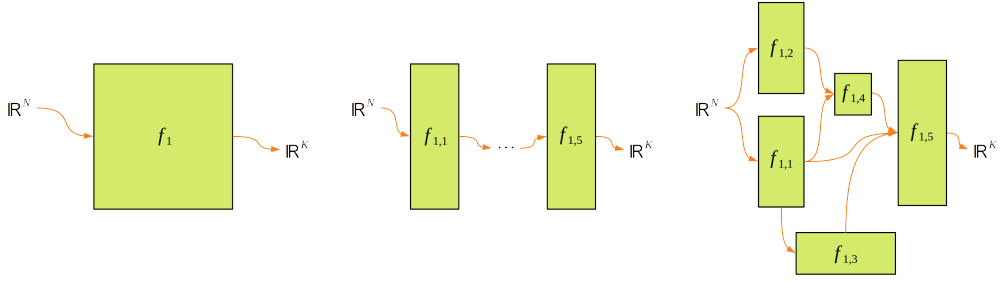
- since models solving typical tasks are known (predictors, metrics), in practice we usually focus only on representation learning

Representation



- since models solving typical tasks are known (predictors, metrics), in practice we usually focus only on representation learning
- mainly we are interested in feedforward architectures (no cycles and loops)

Representation



- since models solving typical tasks are known (predictors, metrics), in practice we usually focus only on representation learning
- mainly we are interested in feedforward architectures (no cycles and loops)
- all blocks can be trainable or not, but all of them should be differentiable with respect to their inputs and parameters (possible end-to-end training)

Explainability

- explainability is required if trained models have a crucial influence on people or environment

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global
- methods can explain model (internal, surrogate)

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global
- methods can explain model (internal, surrogate)
- methods can explain phenomenon (attributions, counterfacts)

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global
- methods can explain model (internal, surrogate)
- methods can explain phenomenon (attributions, counterfacts)
- explanations can have different form (text, image, rules)

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global
- methods can explain model (internal, surrogate)
- methods can explain phenomenon (attributions, counterfacts)
- explanations can have different form (text, image, rules)
- explanations should use interpretable components

Explainability

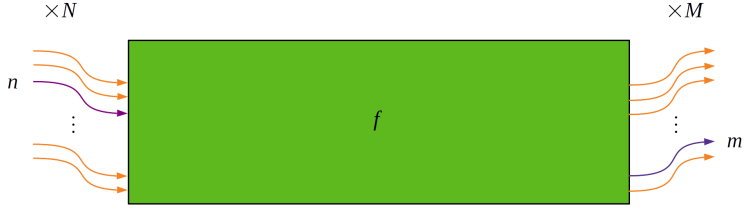
- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global
- methods can explain model (internal, surrogate)
- methods can explain phenomenon (attributions, counterfacts)
- explanations can have different form (text, image, rules)
- explanations should use interpretable components
- it is hard to assess quality of explanations

Explainability

- explainability is required if trained models have a crucial influence on people or environment
- explainability should provide model insights to satisfy needs of some targeted audience
- explainability is necessary since system learns undesirable tricks
- methods can be model specfic or model agnostic
- methods can be local or global
- methods can explain model (internal, surrogate)
- methods can explain phenomenon (attributions, counterfacts)
- explanations can have different form (text, image, rules)
- explanations should use interpretable components
- it is hard to assess quality of explanations
- explanations can lead to knowldge discovery

Attributions

- attribution method assigns scores to inputs based on their contribution to model output



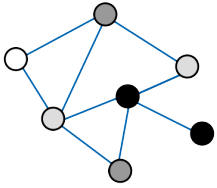
Attributions

- attibutions can be assigned to interpretable components and not to specific inputs

LIME ([10])

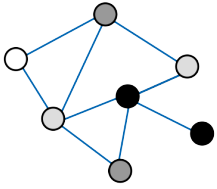
Graphs

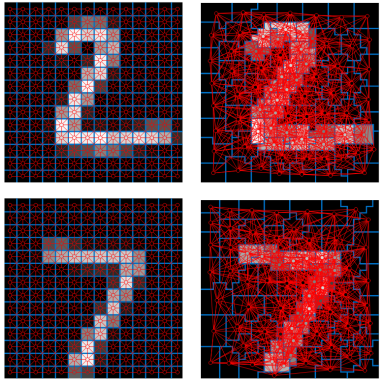
- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$



Graphs

- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$

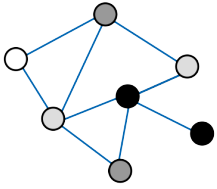


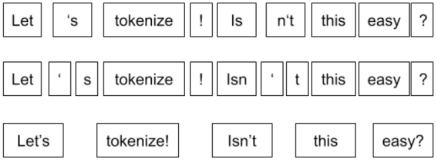


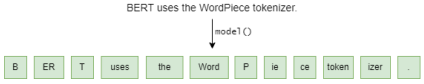
Images ([4])

Graphs

- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$



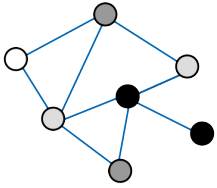


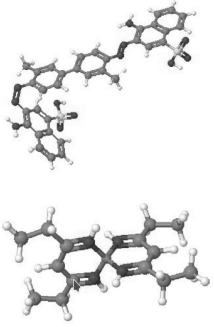


Sentences ([8, 2])

Graphs

- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$

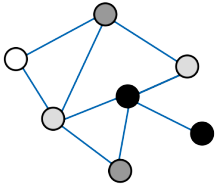


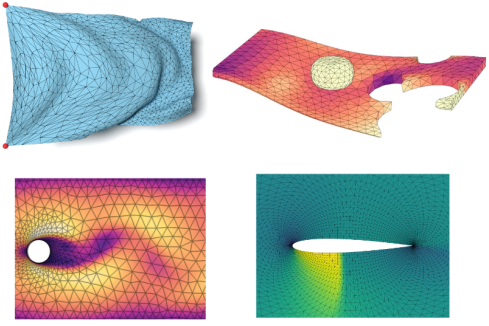


Chemical molecules ([11])

Graphs

- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$

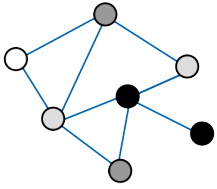




Meshes ([7])

Graphs

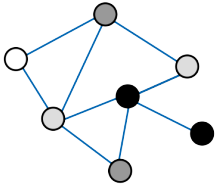
- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$

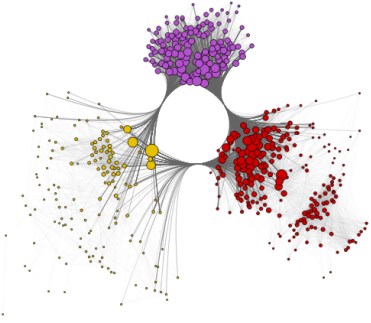


Point clouds ([12])

Graphs

- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$

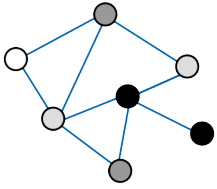


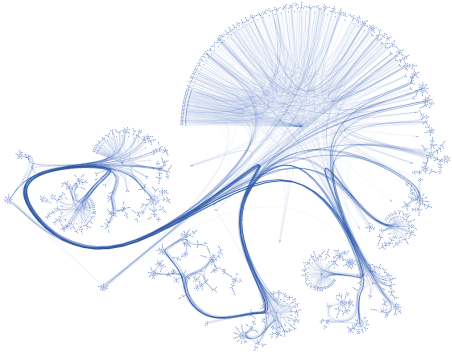


Social networks ([6])

Graphs

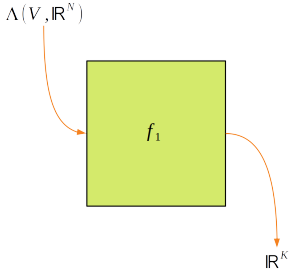
- there are many problems where input objects $\mathcal{O}$ are graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and connecting them edges $\mathcal{E}$





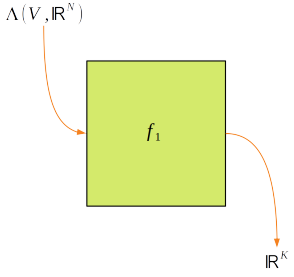
Citation networks ([1])

Representation



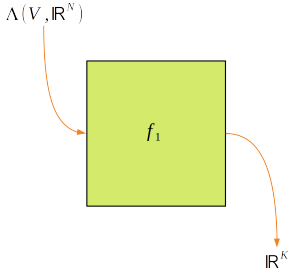
- before graph representation learning some manual feature extraction for graph elements (nodes and edges) must be done

Representation



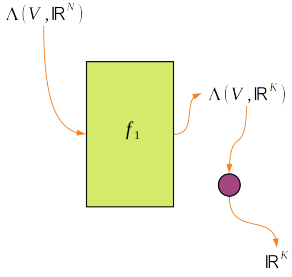
- before graph representation learning some manual feature extraction for graph elements (nodes and edges) must be done
- although it is not a rule we are interested usually in node features $\Lambda\left(\mathcal{V}, \mathbb{R}^{\mathsf{N}}\right)$

Representation



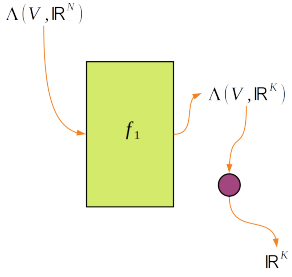
- before graph representation learning some manual feature extraction for graph elements (nodes and edges) must be done
- although it is not a rule we are interested usually in node features $\Lambda\left(\mathcal{V}, \mathbb{R}^{\mathsf{N}}\right)$
- in some tasks edge features can also be considered but it is out of scope of this presentation

Representation



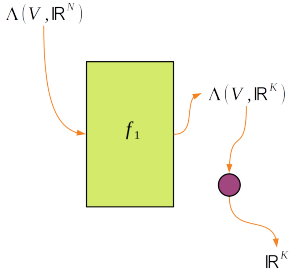
- in practice representation (embedding) learning is usually performed for graph elements

Representation



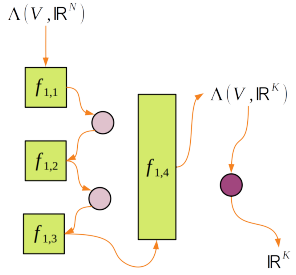
- in practice representation (embedding) learning is usually performed for graph elements
- in particular embeddings are sought for nodes $\Lambda\left(\mathcal{V}, \mathbb{R}^{\mathsf{K}}\right)$

Representation



- in practice representation (embedding) learning is usually performed for graph elements
- in particular embeddings are sought for nodes $\Lambda\left(\mathcal{V}, \mathbb{R}^{\mathsf{K}}\right)$
- this requires additional aggregation (global pooling) finding whole graph embedding in $\mathbb{R}^{\mathsf{K}}$

Representation



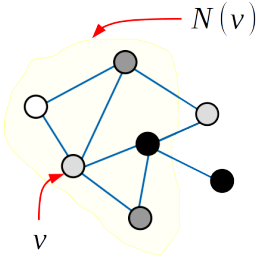
- in practice representation (embedding) learning is usually performed for graph elements
- in particular embeddings are sought for nodes $\Lambda\left(\mathcal{V}, \mathbb{R}^{\mathsf{K}}\right)$
- this requires additional aggregation (global pooling) finding whole graph embedding in $\mathbb{R}^{\mathsf{K}}$
- modification of graph structure (local pooling) can be an element of processing

Structure

- calculating node embedding we cannot take into account its features only since nodes are elements of a structure

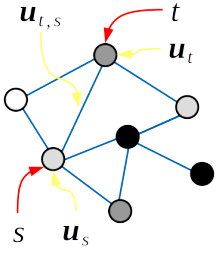
Structure

- calculating node embedding we cannot take into account its features only since nodes are elements of a structure
- that is why we usually define some local neighbourhood $\mathcal{N}(v) \subseteq \mathcal{V}$ for given node $v \in \mathcal{V}$



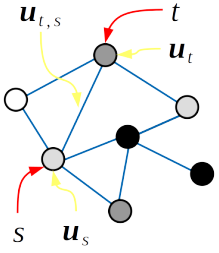
Structure

- calculating node embedding we cannot take into account its features only since nodes are elements of a structure
- that is why we usually define some local neighbourhood $\mathcal{N}(v) \subseteq \mathcal{V}$ for given node $v \in \mathcal{V}$
- for some graphs there is natural spatial ordering of nodes defined by a position $\mathbf{u}_v \in \mathbb{R}^d$ assigned to every node $v \in \mathcal{V}$



Structure

- calculating node embedding we cannot take into account its features only since nodes are elements of a structure
- that is why we usually define some local neighbourhood $\mathcal{N}(v) \subseteq \mathcal{V}$ for given node $v \in \mathcal{V}$
- for some graphs there is natural spatial ordering of nodes defined by a position $\mathbf{u}_v \in \mathbb{R}^d$ assigned to every node $v \in \mathcal{V}$
- node position may be taken into account as an additional information assigned either to nodes (global) $\mathbf{u}_s \in \mathbb{R}^d$ or edges (relative) $\mathbf{u}_{t,s} = \mathbf{u}_t - \mathbf{u}_s \in \mathbb{R}^d$



Tasks

- typical tasks can be considered not only for a graph as a whole but also for its elements

Metric learning

Measuring similarity between nodes may lead to link prediction.

Tasks

- typical tasks can be considered not only for a graph as a whole but also for its elements
- here representation learning for graph elements becomes an obvious choice

Metric learning

Measuring similarity between nodes may lead to link prediction.

Tasks

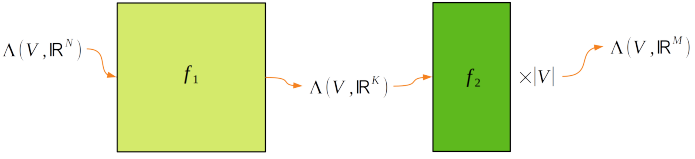
- typical tasks can be considered not only for a graph as a whole but also for its elements
- here representation learning for graph elements becomes an obvious choice

Structured prediction

Classification of nodes leads to segmentation.

Metric learning

Measuring similarity between nodes may lead to link prediction.



Tasks

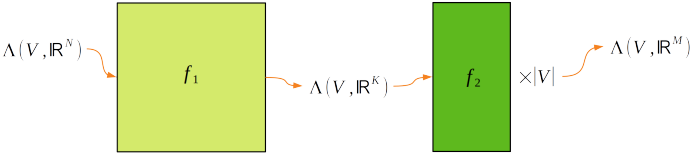
- typical tasks can be considered not only for a graph as a whole but also for its elements
- here representation learning for graph elements becomes an obvious choice

Structured prediction

Regression analysis applied for nodes can be used in recommender systems.

Metric learning

Measuring similarity between nodes may lead to link prediction.



Tasks

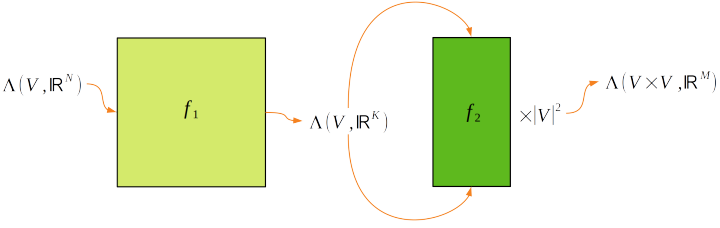
- typical tasks can be considered not only for a graph as a whole but also for its elements
- here representation learning for graph elements becomes an obvious choice

Structured prediction

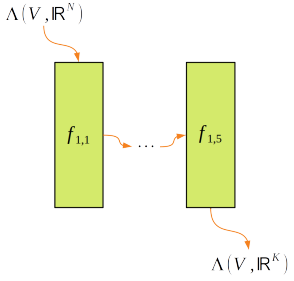
Regression analysis applied for nodes can be used in recommender systems.

Metric learning

Measuring similarity between nodes may lead to link prediction.

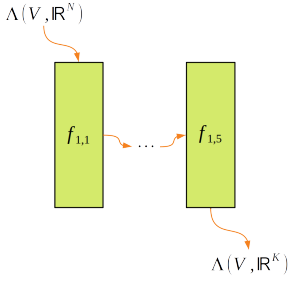


Graph neural networks



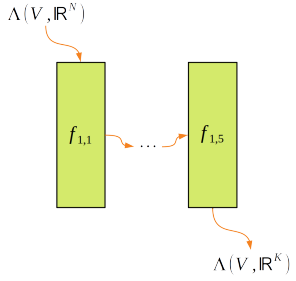
- graph neural networks use graph operators to find successive embeddings of graph nodes

Graph neural networks



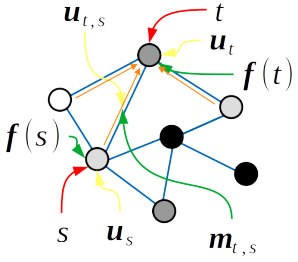
- graph neural networks use graph operators to find successive embeddings of graph nodes
- graph operators are not the only components of those networks (local pooling, non-linear activation functions, etc.)

Graph neural networks



- graph neural networks use graph operators to find successive embeddings of graph nodes
- graph operators are not the only components of those networks (local pooling, non-linear activation functions, etc.)
- the general scheme of graph operator working can be expressed using so called message passing mechanism

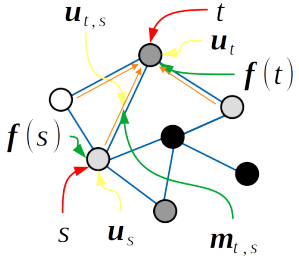
Message passing



Message

$$\mathbf{m}_{t,s} = \mathrm{MSG}(\mathbf{f}(t), \mathbf{f}(s), \mathbf{u}_t, \mathbf{u}_s)$$

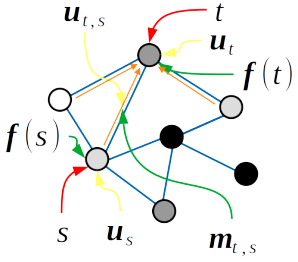
Message passing



Aggregation

$$\mathbf{m}_t = \mathsf{AGG}(\mathbf{m}_{t,s} : s \in \mathcal{N}(t))$$

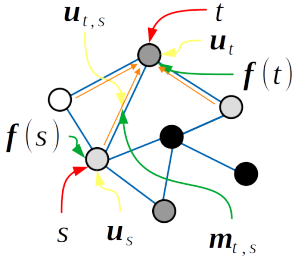
Message passing



Update

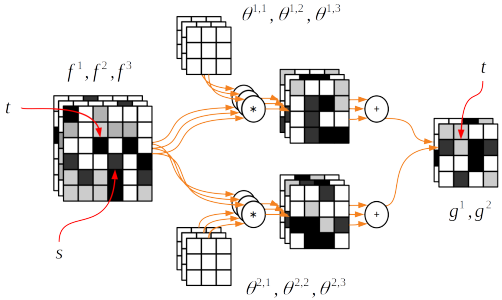
$$\mathbf{g}(t) = \text{UPDATE}(\mathbf{f}(t), \mathbf{m}_t)$$

Message passing

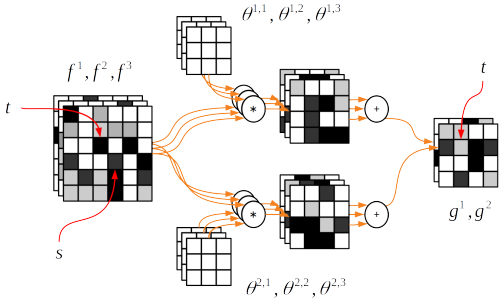


- commonly used graph operators, taking into account spatial ordering of nodes, base on concepts taken from computer vision (convolutional neural networks) and natural language processing domain (transformers)
- naturally, there are also operators designed for specific graph-based tasks (e.g. GCN, GarpgSAGE)

Convolutional neural networks

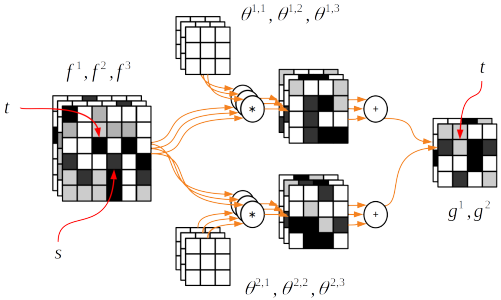


Convolutional neural networks



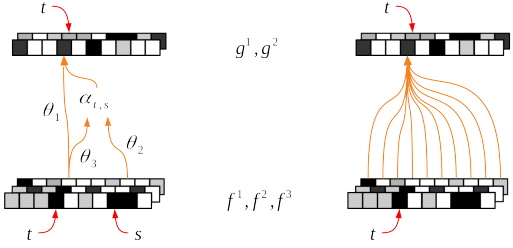
- Mixture Model Network (MoNet) defines continous filters using a trainable gaussian mixture model, taking into account relative coordinates $\mathbf{u}_{t,s} \in \mathbb{R}^d$ which can be defined almost arbitrarily

Convolutional neural networks

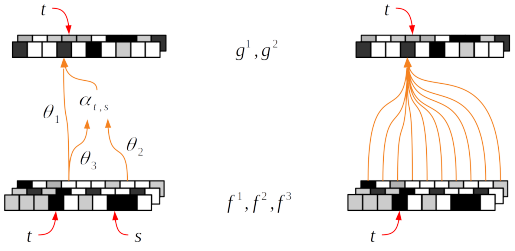


- Mixture Model Network (MoNet) defines continous filters using a trainable gaussian mixture model, taking into account relative coordinates $\mathbf{u}_{t,s} \in \mathbb{R}^d$ which can be defined almost arbitrarily
- Filter weights do not depend on features of currently processed pixels

Transformer

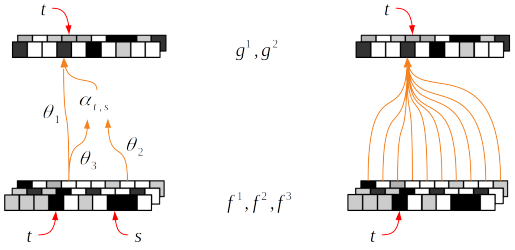


Transformer



- Graph Transformer (GT) or Graph Attention Network (GAT, GATv2) need not to consider the fully connected graph ($\mathcal{N}$ is used) and uses either global $\mathbf{u}_s \in \mathbb{R}^d$ or relative node positions $\mathbf{u}_{t,s} \in \mathbb{R}^d$ to calculate attention weights

Transformer



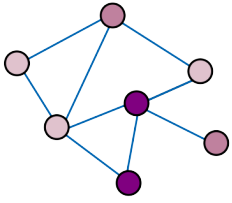
- Graph Transformer (GT) or Graph Attention Network (GAT, GATv2) need not to consider the fully connected graph ($\mathcal{N}$ is used) and uses either global $\mathbf{u}_s \in \mathbb{R}^d$ or relative node positions $\mathbf{u}_{t,s} \in \mathbb{R}^d$ to calculate attention weights
- attention weights depend on features of processed graph elements

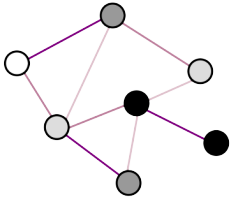
Explainability

- although many classic attibution methods can be used for models operating on graphs, there are also some dedicated techniques

Explainability

- although many classic attibution methods can be used for models operating on graphs, there are also some dedicated techniques
- since graphs have internal structure (nodes and edges), new interpretable components can be considered (separate or common features across all elements, whole elements)

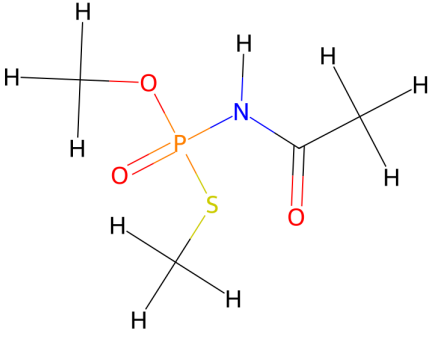




Chemistry

Chemical compounds

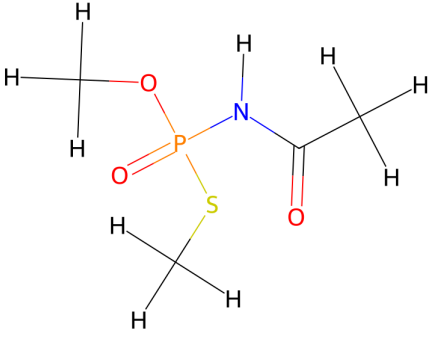
- we consider problems where the set of analysed objects $\mathcal{O}$ are chemical compounds



Acephate

Chemical compounds

- we consider problems where the set of analysed objects $\mathcal{O}$ are chemical compounds
- this work was conducted with Bartosz Durys



Acephate

Prediction

Regression loss

In regression problems $\mathrm{M} = 1$ and loss function can be defined as:

$$\mathrm{L}^{prediction}(\mathbf{z}^j, y^j) = (\mathbf{z}^j - y^j)^2$$

where $\mathbf{z}^j = \mathrm{f}(G^j, \theta) \in \mathbb{R}^{\mathrm{M}}$

- selected MoleculeNet [14] datasets: ESOL, FreeSolv, and Lipophilicity
- feature vectors contained 9 numerical features describing atoms
- considered tasks: water solubility prediction, hydration free energy estimation, and finding octanol/water distribution coefficient

Prediction

Classification loss

In classification problems $\mathrm{M} = \mathrm{L}$ and loss function can be defined as:

$$\mathrm{L}^{prediction}(\mathbf{z}^j, y^j) = -\ln \frac{\exp \mathbf{z}^j_{y^j}}{\sum_{l=1}^{L} \exp \mathbf{z}^j_l}$$

where $\mathbf{z}^j = \mathrm{f}(G^j, \theta)\mathbb{R}^{\mathrm{M}}$

- selected TUDataset [5] datasets: AIDS, ENZYMES, and PROTEINS
- feature vectors were one-hot encoded representation of node class: chcemical element for AIDS, secondary structure element for ENZYMES and PROTEINS
- considered tasks: identification of molecule activity against HIV, assigning a molecule to one of the six Enzyme Commission top-level classes, and predicting if a protein is an enzyme

Explainability

- in GT, GAT and GATv2 models attention coefficients $\alpha_{t,s}$ can be used as attributions of graph edges

Explainability

- in GT, GAT and GATv2 models attention coefficients $\alpha_{t,s}$ can be used as attributions of graph edges
- to improve interpretability of attention coefficients $\alpha_{t,s}$ a new regularization component of loss function was proposed

Regularization

$$\mathrm{L}^{explain}(\mathbf{z}^j, y^j) = \sum_{t \in \mathcal{V}} \left(1 - \max_{s \in \mathcal{N}(t)} \alpha_{t,s}(\theta)\right)$$

Explainability

- in GT, GAT and GATv2 models attention coefficients $\alpha_{t,s}$ can be used as attributions of graph edges
- to improve interpretability of attention coefficients $\alpha_{t,s}$ a new regularization component of loss function was proposed

Regularization

$$\mathrm{L}^{explain}(\mathbf{z}^j, y^j) = \sum_{t \in \mathcal{V}} \left(1 - \max_{s \in \mathcal{N}(t)} \alpha_{t,s}(\theta)\right)$$

- it utilizes the fact that for a given node t those coefficients are normalized with softmax function, which means that $\alpha_{t,s} \in [0, 1]$ for $s \in \mathcal{N}(t)$ and their sum is equal to 1

Explainability

- in GT, GAT and GATv2 models attention coefficients $\alpha_{t,s}$ can be used as attributions of graph edges
- to improve interpretability of attention coefficients $\alpha_{t,s}$ a new regularization component of loss function was proposed

Regularization

$$\mathrm{L}^{explain}(\mathbf{z}^j, y^j) = \sum_{t \in \mathcal{V}} \left(1 - \max_{s \in \mathcal{N}(t)} \alpha_{t,s}(\theta)\right)$$

- it utilizes the fact that for a given node t those coefficients are normalized with softmax function, which means that $\alpha_{t,s} \in [0, 1]$ for $s \in \mathcal{N}(t)$ and their sum is equal to 1
- the final loss function used during training is $\mathrm{L} = \mathrm{L}^{prediction} + \lambda \cdot \mathrm{L}^{explain}$, where λ controls the trade-off between the two components

Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion

Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion
- only single-headed attention mechanism was used for a fair comparison and easier interpretability

Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion
- only single-headed attention mechanism was used for a fair comparison and easier interpretability
- for each operator, we used two layers with batch normalization, dropout, and ReLU activation function

Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion
- only single-headed attention mechanism was used for a fair comparison and easier interpretability
- for each operator, we used two layers with batch normalization, dropout, and ReLU activation function
- hidden node embeddings were aggregated using global average pooling

Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion
- only single-headed attention mechanism was used for a fair comparison and easier interpretability
- for each operator, we used two layers with batch normalization, dropout, and ReLU activation function
- hidden node embeddings were aggregated using global average pooling
- a multi-layer perceptron was used to generate the final predictions

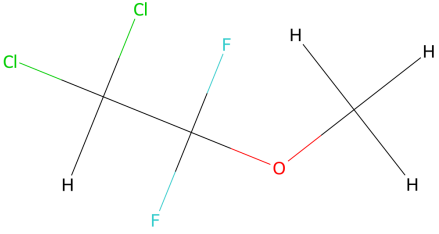
Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion
- only single-headed attention mechanism was used for a fair comparison and easier interpretability
- for each operator, we used two layers with batch normalization, dropout, and ReLU activation function
- hidden node embeddings were aggregated using global average pooling
- a multi-layer perceptron was used to generate the final predictions
- λ was set to 0.1 experimentally.

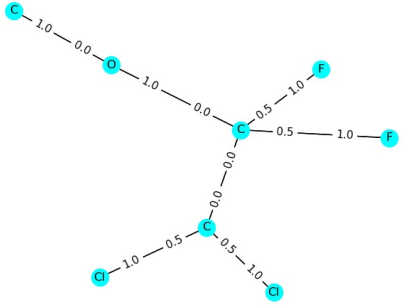
Methodology

- every dataset was split into training, validation, and test sets using an 80/10/10 proportion
- only single-headed attention mechanism was used for a fair comparison and easier interpretability
- for each operator, we used two layers with batch normalization, dropout, and ReLU activation function
- hidden node embeddings were aggregated using global average pooling
- a multi-layer perceptron was used to generate the final predictions
- λ was set to 0.1 experimentally.
- every experiment was repeated 50 times with 500 epochs per repeat, and the results were averaged using the best epoch on the validation set.

Explanations



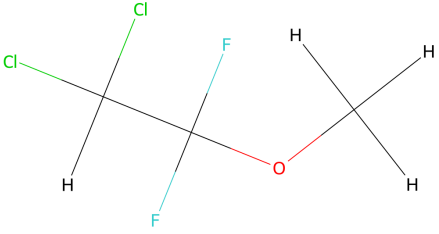
(a)



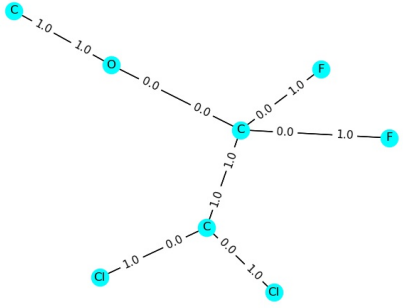
(b) normal

Methoxyflurane from the FreeSolv dataset

Explanations



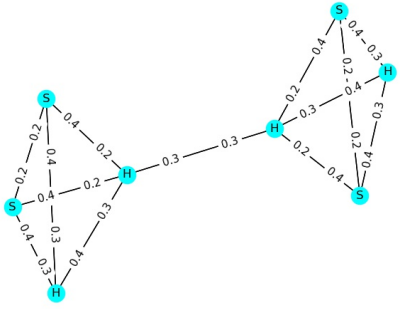
(a)



(b) with $\mathrm{L}^{explain}$

Methoxyflurane from the FreeSolv dataset

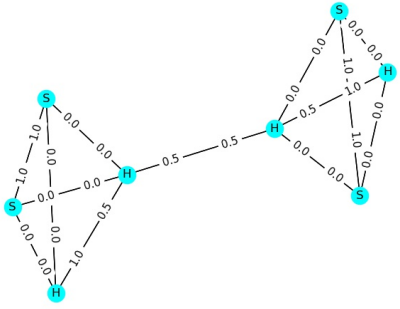
Explanations



(a) normal

Protein from the PROTEINS set

Explanations



(a) with $\mathrm{L}^{explain}$

Protein from the PROTEINS set

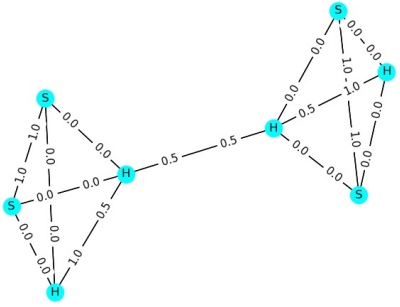
Regression

Dataset	Operator	Validation set			Test set		
		MSE	Standard deviation	Best	MSE	Standard deviation	Best
ESOL	GCN	35.24	9.11	8.15	35.24	8.80	8.11
	GAT	**11.76**	**5.09**	4.62	**11.01**	**5.04**	**4.46**
	GT	32.07	18.40	5.61	31.58	17.80	5.41
	GT with $L^{explain}$	17.87	11.29	**4.53**	17.25	11.11	**4.46**
FreeSolv	GCN	78.64	38.21	17.42	75.11	34.37	17.01
	GAT	36.02	16.94	14.42	**33.95**	**15.02**	14.17
	GT	44.30	26.35	**13.91**	41.87	25.38	14.01
	GT with $L^{explain}$	**34.71**	**11.35**	14.14	34.11	17.25	**13.86**
Lipophilicity	GCN	12.65	6.75	2.34	12.69	6.84	2.35
	GAT	10.67	4.54	2.12	10.64	4.58	2.16
	GT	12.51	9.19	2.44	12.44	9.15	2.46
	GT with $L^{explain}$	**2.62**	**0.80**	**1.62**	**2.56**	**0.77**	**1.64**

Classification

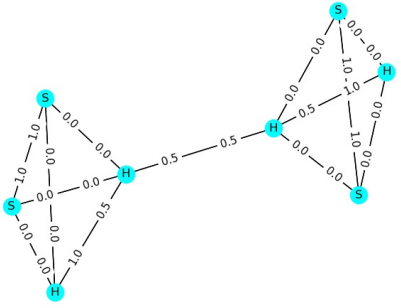
Dataset	Operator	Validation set			Test set		
		Accuracy	Standard deviation	Best	Accuracy	Standard deviation	Best
AIDS	GCN	80.22	2.81	**88.50**	79.84	**2.27** width=6cm	84.00
	GAT	79.84	**2.74**	85.50	79.91	2.55	86.00
	GT	**81.05**	2.80	86.00	**80.49**	2.46	**87.00**
	GT with $L^{explain}$	79.75	2.76	86.00	80.03	2.42	86.00
ENZYMES	GCN	30.90	4.14	43.33	20.87	5.69	35.00
	GAT	32.03	4.79	**46.67**	21.43	**5.56**	35.00
	GT	35.17	**3.45**	43.33	24.03	5.80	**40.00**
	GT with $L^{explain}$	**36.77**	3.81	45.00	**24.33**	5.74	35.00
PROTEINS	GCN	73.69	3.99	81.08	68.88	5.27	78.57
	GAT	73.39	**3.43**	81.08	68.55	4.63	76.79
	GT	**74.29**	3.97	**81.98**	**69.20**	5.04	**79.46**
	GT with $L^{explain}$	74.22	3.64	**81.98**	68.66	**4.50**	78.57

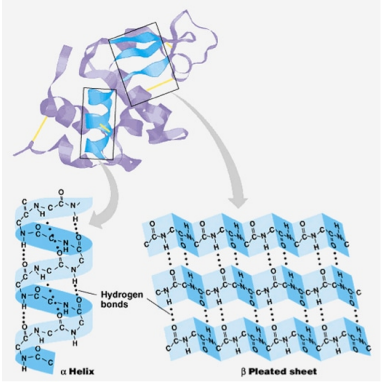
Interpretable components



Protein secondary structure ([9])

Interpretable components





Protein secondary structure ([9])

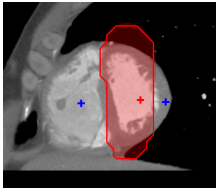
Image analysis

Research problem

How to use domain knowledge in image analysis?

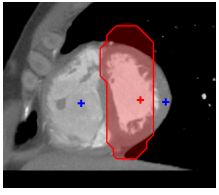
Research problem

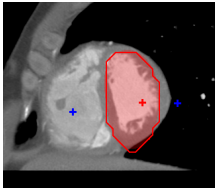
How to use domain knowledge in image analysis?



Research problem

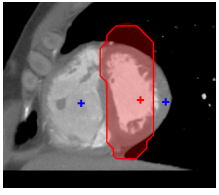
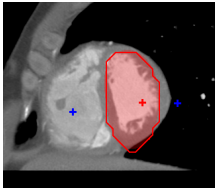
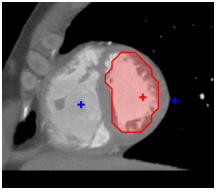
How to use domain knowledge in image analysis?





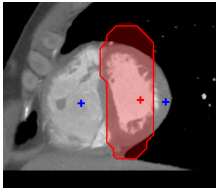
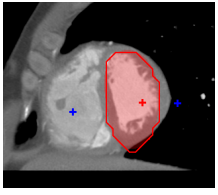
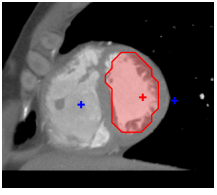
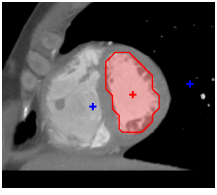
Research problem

How to use domain knowledge in image analysis?

Research problem

How to use domain knowledge in image analysis?

Research problem

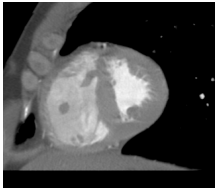
- we consider problems where the set of analysed objects $\mathcal{O}$ contains images

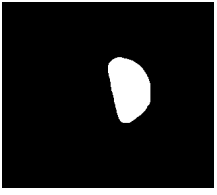
Research problem

- we consider problems where the set of analysed objects $\mathcal{O}$ contains images
- natutally tasks like prediction (classification, regression) or metric learning can be considered here

Research problem

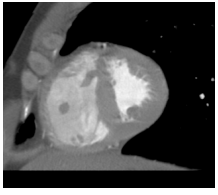
- we consider problems where the set of analysed objects $\mathcal{O}$ contains images
- natutally tasks like prediction (classification, regression) or metric learning can be considered here
- above tasks can be considered not only for whole images, but for pixels as well

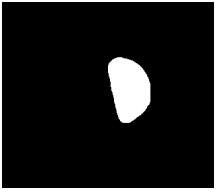




Research problem

- we consider problems where the set of analysed objects $\mathcal{O}$ contains images
- natutally tasks like prediction (classification, regression) or metric learning can be considered here
- above tasks can be considered not only for whole images, but for pixels as well
- images have their internal structure since they are composed of pixels $\mathcal{V}$ organized in a grid structure





Research hypothesis

Use interpretable components to describe image content.

Research hypothesis

Use interpretable components to describe image content.

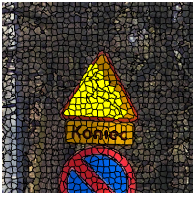
Research hypothesis

Use interpretable components to describe image content.

Research hypothesis

Use interpretable components to describe image content.

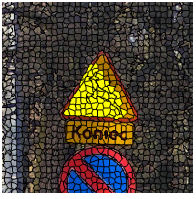
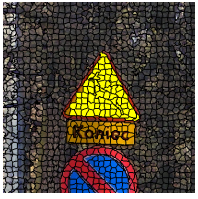




Research hypothesis

Use interpretable components to describe image content.



Research hypothesis

- convolutional neural networks revolutionized image analysis

Research hypothesis

- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge

Research hypothesis

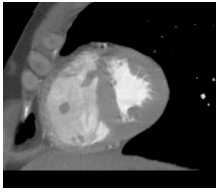
- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge
- interpretable components enable better communication with domain experts

Research hypothesis

- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge
- interpretable components enable better communication with domain experts
- additional knowledge may reduce the required number of training data

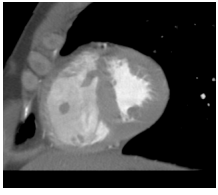
Research hypothesis

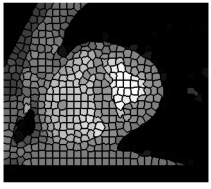
- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge
- interpretable components enable better communication with domain experts
- additional knowledge may reduce the required number of training data
- alternative image representations are not entirely new



Research hypothesis

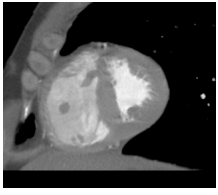
- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge
- interpretable components enable better communication with domain experts
- additional knowledge may reduce the required number of training data
- alternative image representations are not entirely new

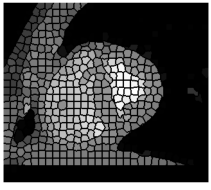


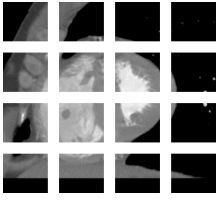


Research hypothesis

- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge
- interpretable components enable better communication with domain experts
- additional knowledge may reduce the required number of training data
- alternative image representations are not entirely new

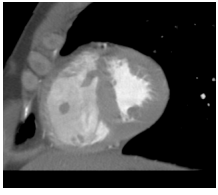


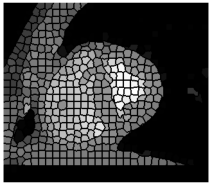


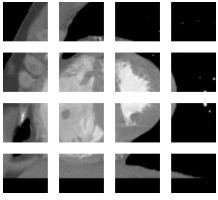


Research hypothesis

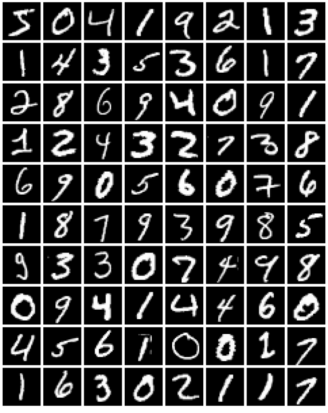
- convolutional neural networks revolutionized image analysis
- massive number of annotated training data is considered as the only domain knowledge
- interpretable components enable better communication with domain experts
- additional knowledge may reduce the required number of training data
- alternative image representations are not entirely new
- graph neual networks are a tool allowing to process that kind of data





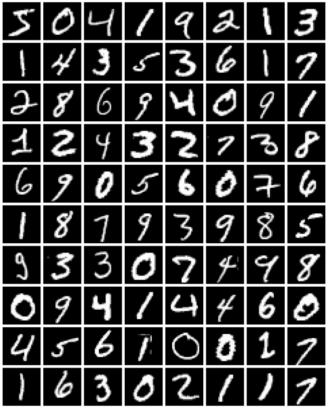


Dataset



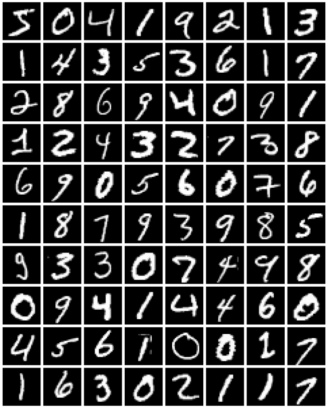
- MNIST [3] dataset with handwitten digits

Dataset



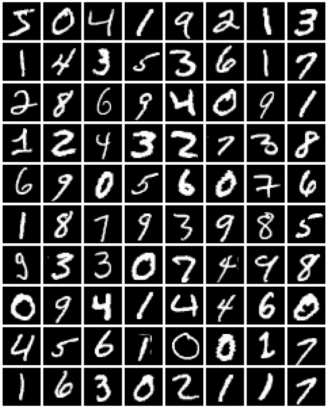
- MNIST [3] dataset with handwitten digits
- it contains 60000 train and 10000 test images

Dataset



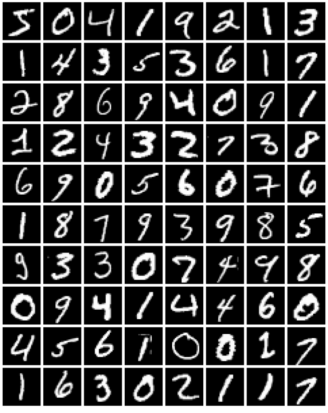
- MNIST [3] dataset with handwitten digits
- it contains 60000 train and 10000 test images
- classic methods operating on pixels were able to gain 99.87% of accuracy

Dataset



- MNIST [3] dataset with handwitten digits
- it contains 60000 train and 10000 test images
- classic methods operating on pixels were able to gain 99.87% of accuracy
- authors of MoNet for full pixel grid reported 99.19%

Dataset



- MNIST [3] dataset with handwitten digits
- it contains 60000 train and 10000 test images
- classic methods operating on pixels were able to gain 99.87% of accuracy
- authors of MoNet for full pixel grid reported 99.19%
- they achieved also 97.30% for 300 superpixels and 91.11% for 75 superpixels

Sparse coding

Convolutional sparse coding

$$\arg\min_{\{\mathbf{c}_i\}} \| \sum_i \mathbf{d}_i * \mathbf{c}_i - \mathbf{o} \|_2^2 + \lambda \sum_i \|\mathbf{c}_i\|_1$$

Sparse coding

Convolutional sparse coding

$$\arg\min_{\{\mathbf{c}_i\}} \| \sum_i \mathbf{d}_i * \mathbf{c}_i - \mathbf{o} \|_2^2 + \lambda \sum_i \|\mathbf{c}_i\|_1$$

Convolutional dictionary learning

$$\arg\min_{\{\mathbf{d}_i, \mathbf{c}_i^j\}} \sum_j \| \sum_i \mathbf{d}_i * \mathbf{c}_i^j - \mathbf{o}^j \|_2^2 + \lambda \sum_j \sum_i \|\mathbf{c}_i^j\|_1$$

Sparse coding

Convolutional sparse coding

$$\arg\min_{\{\mathbf{c}_i\}} \| \sum_i \mathbf{d}_i * \mathbf{c}_i - \mathbf{o} \|_2^2 + \lambda \sum_i \|\mathbf{c}_i\|_1$$

Convolutional dictionary learning

$$\arg\min_{\{\mathbf{d}_i, \mathbf{c}_i^j\}} \sum_j \| \sum_i \mathbf{d}_i * \mathbf{c}_i^j - \mathbf{o}^j \|_2^2 + \lambda \sum_j \sum_i \|\mathbf{c}_i^j\|_1$$

- to generate alternative representation of image $\mathbf{o}$ convolutional sparse coding from SPORCO [13] library was used

Sparse coding

Convolutional sparse coding

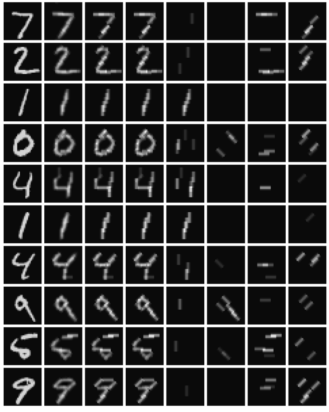
$$\arg\min_{\{\mathbf{c}_i\}} \| \sum_i \mathbf{d}_i * \mathbf{c}_i - \mathbf{o} \|_2^2 + \lambda \sum_i \|\mathbf{c}_i\|_1$$

Convolutional dictionary learning

$$\arg\min_{\{\mathbf{d}_i, \mathbf{c}_i^j\}} \sum_j \| \sum_i \mathbf{d}_i * \mathbf{c}_i^j - \mathbf{o}^j \|_2^2 + \lambda \sum_j \sum_i \|\mathbf{c}_i^j\|_1$$

- to generate alternative representation of image $\mathbf{o}$ convolutional sparse coding from SPORCO [13] library was used
- to increase sparsity thresholding and non-maximum suppression were applied

Interpretable components

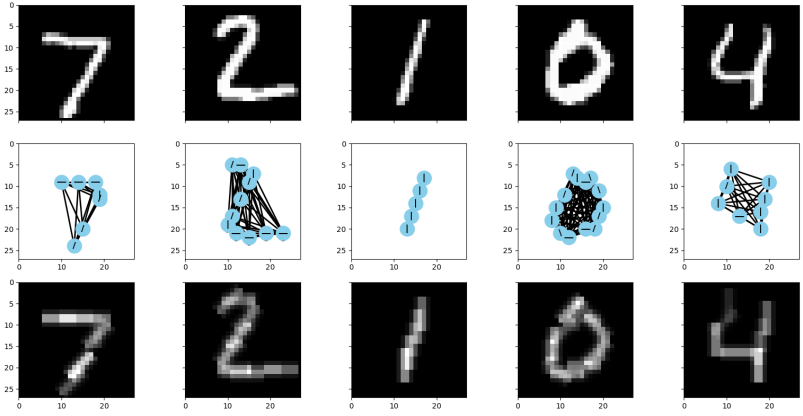


fixed dictionary

Interpretable components

trained dictionary

Graphs



fully connected graph was considered where $\mathcal{N}(v) = \mathcal{V}$ for every $v \in \mathcal{V}$

Reference

- in the experiments the reduced number (50, 100, 200, 1000) of training samples was considered

Reference

- in the experiments the reduced number (50, 100, 200, 1000) of training samples was considered
- 1000 samples were kept in validation set for early stopping

Reference

- in the experiments the reduced number (50, 100, 200, 1000) of training samples was considered
- 1000 samples were kept in validation set for early stopping
- all architectures were similar (number of layers, embedding sizes, etc.)

Reference

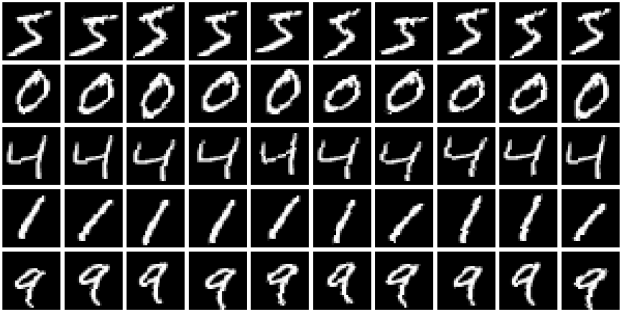
- in the experiments the reduced number (50, 100, 200, 1000) of training samples was considered
- 1000 samples were kept in validation set for early stopping
- all architectures were similar (number of layers, embedding sizes, etc.)
- presented results are maximum scores from 3 runs with random initial weights

Reference

- in the experiments the reduced number (50, 100, 200, 1000) of training samples was considered
- 1000 samples were kept in validation set for early stopping
- all architectures were similar (number of layers, embedding sizes, etc.)
- presented results are maximum scores from 3 runs with random initial weights

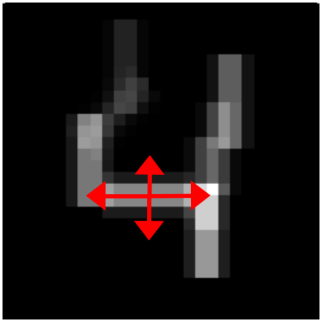
method	50	100	200	1000	ALL
CNN	33.51	55.77	71.62	**90.67**	**99.21**
MoNet	16.41	58.79	71.77	85.11	97.38
GT	**47.94**	**66.52**	**77.02**	88.87	96.52

Augmentation



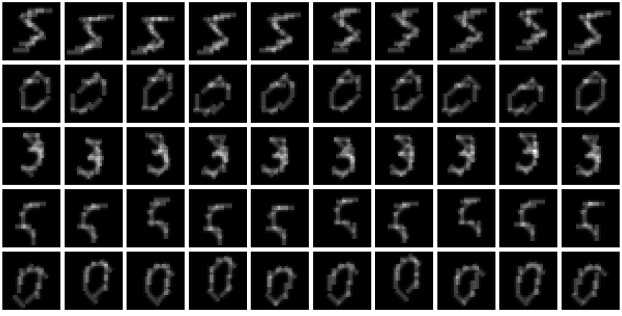
- augmentation, for example affine transformations, can artificially increase the number of training data

Augmentation



- augmentation, for example affine transformations, can artificially increase the number of training data
- if components are interpretable, augmentation can use domain specific knowledge

Augmentation



- augmentation, for example affine transformations, can artificially increase the number of training data
- if components are interpretable, augmentation can use domain specific knowledge

Augmentation

method	50	100	200	1000	ALL
CNN	33.51	55.77	71.62	90.67	99.21
MoNet	16.41	58.79	71.77	85.11	97.38
GT	**47.94**	66.52	77.02	88.87	96.52
CNN with augmentation	39.61	61.97	76.94	**93.75**	**99.31**
MoNet with augmentation	16.14	59.37	74.44	88.11	97.39
GT with augmentation	40.87	**72.16**	**80.94**	88.78	96.88

- augmentation, for example affine transformations, can artificially increase the number of training data
- if components are interpretable, augmentation can use domain specific knowledge

Knowledge

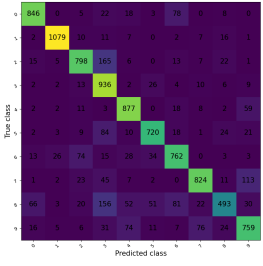
Cross-entropy loss

$$\mathrm{L}^{ce}(\mathbf{z}^j, l^j) = -\ln \frac{\exp \mathbf{z}^j_{lj}}{\sum_{l=1}^{L} \exp \mathbf{z}^j_l}$$

Knowledge

Cross-entropy loss

$$\mathrm{L}^{ce}(\mathbf{z}^j, l^j) = -\ln \frac{\exp \mathbf{z}^j_{lj}}{\sum_{l=1}^{L} \exp \mathbf{z}^j_l}$$

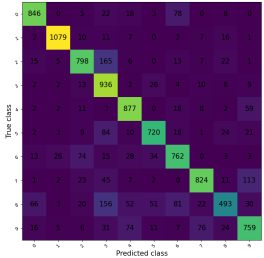


confusion matrix

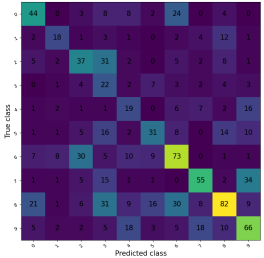
Knowledge

Cross-entropy loss

$$\mathrm{L}^{ce}(\mathbf{z}^j, l^j) = -\ln \frac{\exp \mathbf{z}^j_{lj}}{\sum_{l=1}^{L} \exp \mathbf{z}^j_l}$$



confusion matrix



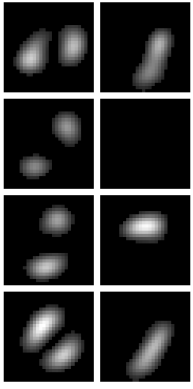
uncertainity

Knowledge

- interpretable components may allow domain experts to easier express expectations about components distribution

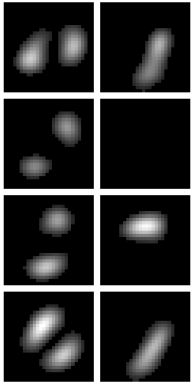
Knowledge

- interpretable components may allow domain experts to easier express expectations about components distribution
- those expectations can have form of rules or some visual guidelines



Knowledge

- interpretable components may allow domain experts to easier express expectations about components distribution
- those expectations can have form of rules or some visual guidelines
- they can be used to measure the coarse similarity w_l^j of given training sample to considered concepts



Knowledge

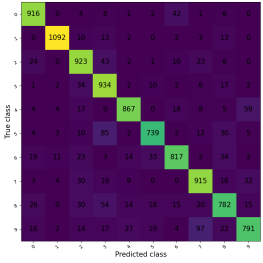
Knowledge loss

$$\mathrm{L}^{knowledge}(\mathbf{z}^j, l^j) = (1-\alpha)\cdot \mathrm{L}^{ce}(\mathbf{z}^j, l^j) + \alpha \cdot \sum_{l=1}^{L} w_l^j \cdot \mathrm{L}^{ce}(\mathbf{z}^j, l)$$

Knowledge

Knowledge loss

$$\mathrm{L}^{knowledge}(\mathbf{z}^j, l^j) = (1-\alpha)\cdot \mathrm{L}^{ce}(\mathbf{z}^j, l^j) + \alpha \cdot \sum_{l=1}^{L} w_l^j \cdot \mathrm{L}^{ce}(\mathbf{z}^j, l)$$

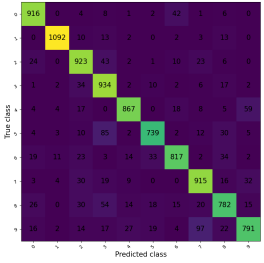


confusion matrix

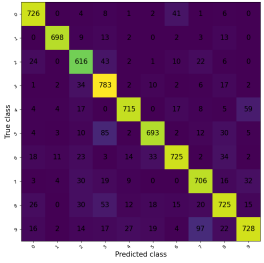
Knowledge

Knowledge loss

$$\mathrm{L}^{knowledge}(\mathbf{z}^j, l^j) = (1-\alpha)\cdot \mathrm{L}^{ce}(\mathbf{z}^j, l^j) + \alpha \cdot \sum_{l=1}^{L} w_l^j \cdot \mathrm{L}^{ce}(\mathbf{z}^j, l)$$



confusion matrix

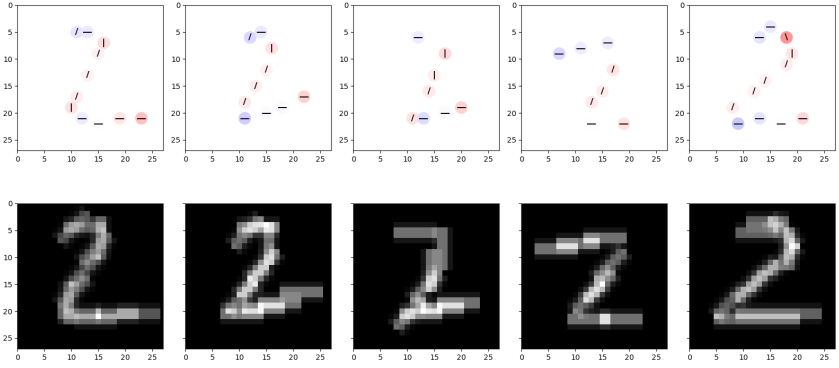


uncertainity

Knowledge

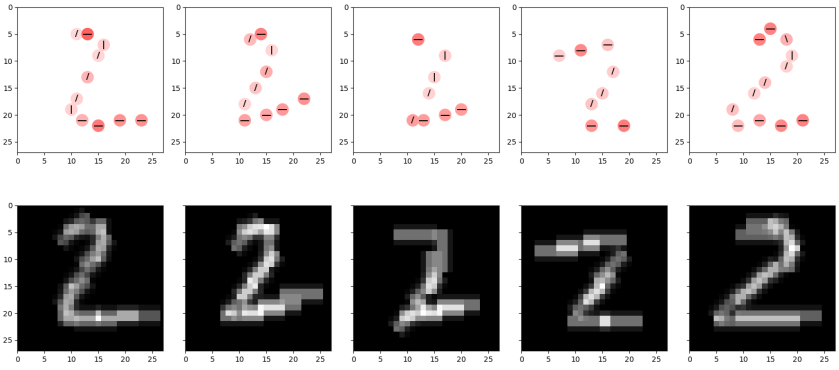
method	50	100	200	1000	ALL
CNN	33.51	55.77	71.62	90.67	99.21
MoNet	16.41	58.79	71.77	85.11	97.38
GT	47.94	66.52	77.02	88.87	96.52
CNN with augmentation	39.61	61.97	76.94	**93.75**	**99.31**
MoNet ($\alpha = 0.5$)	57.21	67.92	81.60	91.09	97.75
MoNet with augmentation ($\alpha = 0.5$)	56.25	71.21	82.29	92.34	97.56
GT ($\alpha = 0.5$)	60.30	75.21	81.84	91.61	96.87
GT with augmentation ($\alpha = 0.5$)	**67.43**	**78.00**	**87.76**	93.26	97.12
MoNet ($\alpha = 1.0$)	44.67	56.98	63.87	75.12	76.87
MoNet with augmentation ($\alpha = 1.0$)	46.88	59.21	66.14	75.93	77.91
GT ($\alpha = 1.0$)	50.30	59.57	68.54	74.12	77.15
GT with augmentation ($\alpha = 1.0$)	58.64	67.11	70.23	77.06	77.81

Explanations



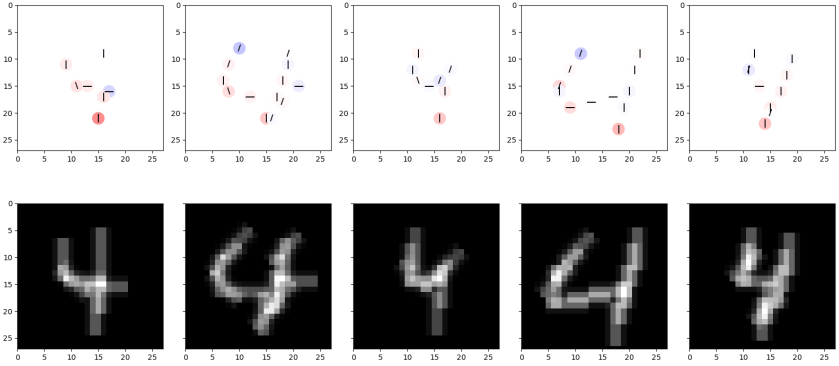
integrated gradients

Explanations



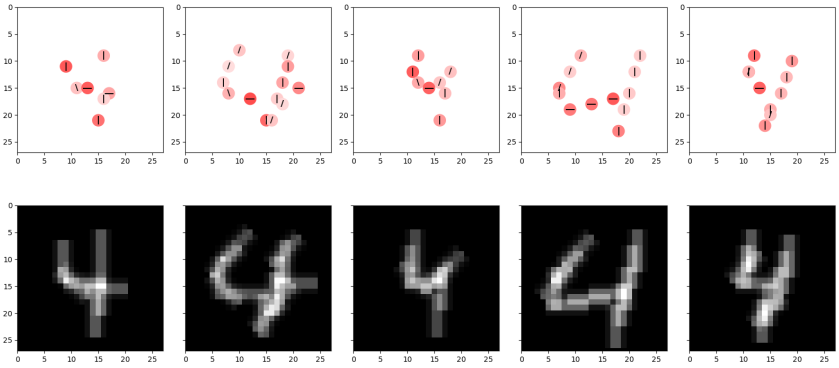
gnn explainer

Explanations



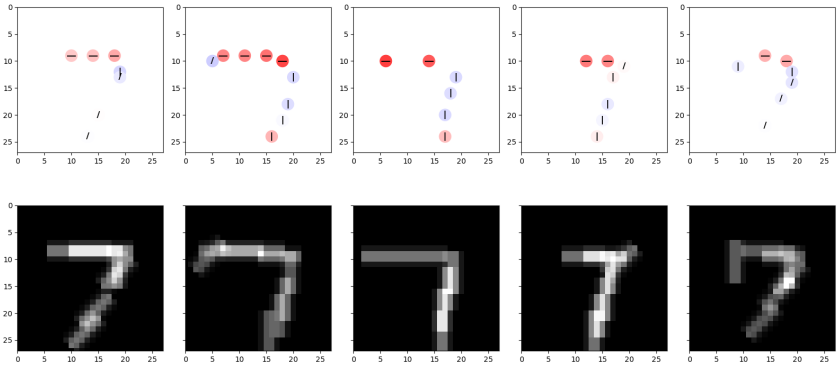
integrated gradients

Explanations



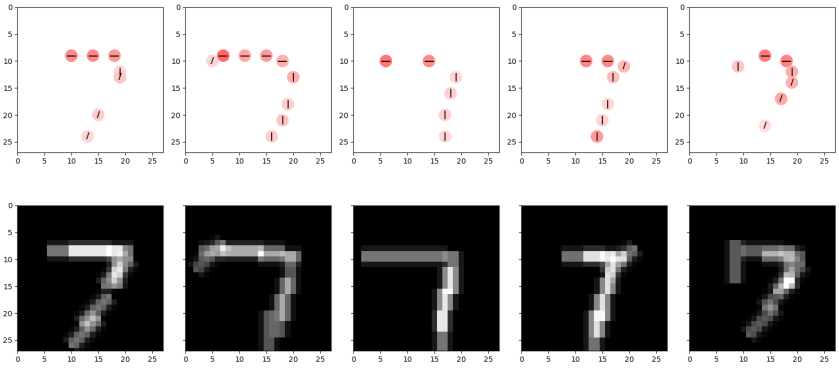
gnn explainer

Explanations



integrated gradients

Explanations



gnn explainer

Summary

Interpretable components allow to:

- use domain knowledge other then annotated set of training data
- explain results using domain specific concepts, which may lead to knowledge discovery

Summary

Open questions:

- should interpretable components be universal or task specific?
- should interpretable components be carefully designed or trainable?
- are interpretable components only useful in image analysis?

References I

[1] Orbifold Consulting.
The cora dataset.
https://graphsandnetworks.com/the-cora-dataset.
Accessed: August 18, 2023.

[2] Cathal Horan.
Tokenizers: How machines read.
https://blog.floydhub.com/tokenization-nlp/.
Accessed: January 7, 2024.

[3] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges.
The MNIST database of handwritten digits.
http://yann.lecun.com/exdb/mnist/.
Accessed: September 3, 2023.

References II

[4] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein.
Geometric deep learning on graphs and manifolds using mixture model CNNs.
In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5425–5434. IEEE Computer Society, 2017.

[5] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann.
Tudataset: A collection of benchmark datasets for learning with graphs.
CoRR, abs/2007.08663, 2020.

[6] Neptune.ai.
Graph neural network and some of gnn applications: Everything you need to know.
`https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications`.
Accessed: August 18, 2023.

References III

[7] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia.
Learning mesh-based simulation with graph networks.
CoRR, abs/2010.03409, 2020.

[8] Dimitris Poulopoulos.
The ultimate guide to training bert from scratch: The tokenizer.
https://towardsdatascience.com/the-ultimate-guide-to-training-bert-from-scratch-the-tokenizer-ddf30f124822.
Accessed: January 7, 2024.

[9] Creative Proteomics.
Protein secondary structure prediction service.
https://www.creative-proteomics.com/services/protein-secondary-structure-prediction-service.htm.
Accessed: January 10, 2024.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin.
"Why Should I Trust You?": Explaining the predictions of any classifier, 2016.

References IV

[11] Kaspar Riesen and Horst Bunke.
IAM graph database repository for graph based pattern recognition and machine learning.
In Niels da Vitoria Lobo, Takis Kasparis, Fabio Roli, James T. Kwok, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Marco Loog, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–297, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[12] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon.
Dynamic graph CNN for learning on point clouds.
CoRR, abs/1801.07829, 2018.

[13] Brendt Wohlberg.
SPORCO: A Python package for standard and convolutional sparse representations.
In *Proceedings of the 15th Python in Science Conference*, pages 1–8, Austin, TX, USA, July 2017.

References V

[14] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande.
Moleculenet: A benchmark for molecular machine learning.
CoRR, abs/1703.00564, 2017.