TreeCaps : Tree-Structured Capsule Networks for Program Source Code Processing

Presented by Vinoj Jayasundara

Software Intelligence Group

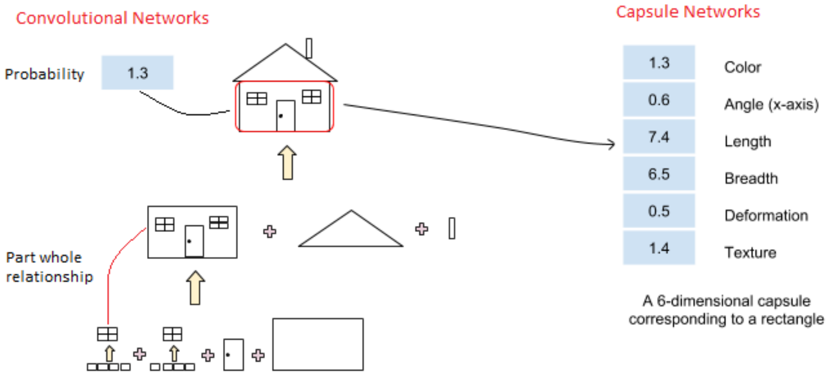
December 1, 2019

Presentation Outline

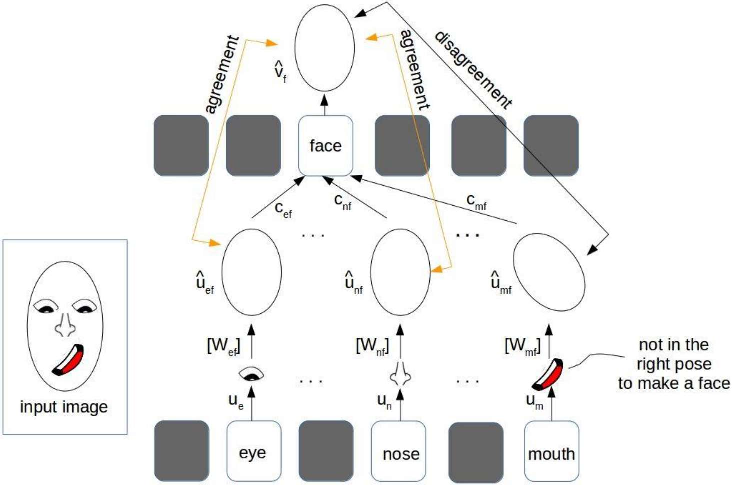
1. Introduction to Capsule Networks
2. Methodology
3. Task : Program Classification
4. Limitations and Future Work

Capsule Network : Instantiation parameters

Capsule Networks can encode any entity in instantiation parameters.



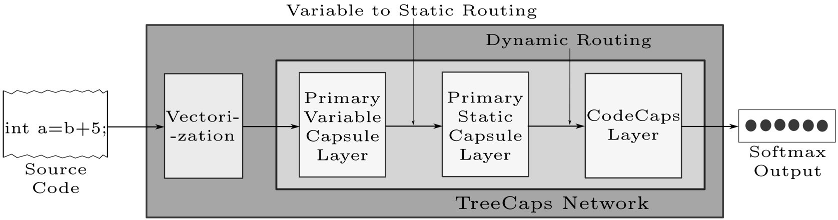
Capsule Networks propose a novel routing by agreement algorithm.



Presentation Outline

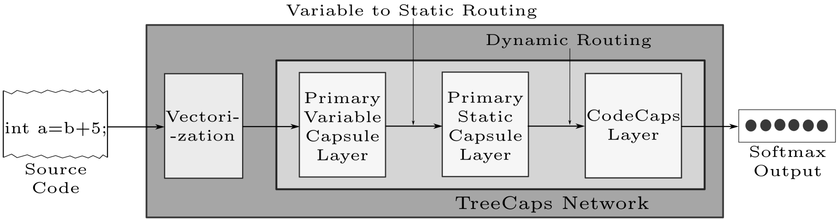
1. Introduction to Capsule Networks
2. Methodology
3. Task : Program Classification
4. Limitations and Future Work

TreeCaps Overview



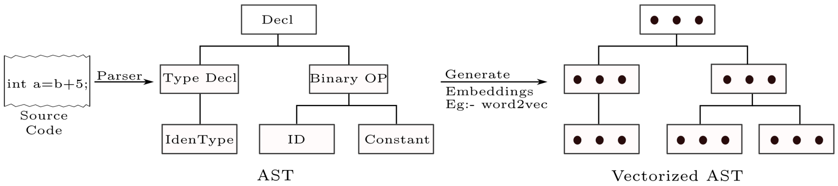
- The source code of the training sample program is parsed into an AST and vectorized with the aid of a suitable technique. (Eg :- Word2Vec)

TreeCaps Overview



- The source code of the training sample program is parsed into an AST and vectorized with the aid of a suitable technique. (Eg :- Word2Vec)
- The vectorized AST is then fed to the proposed TreeCaps network, which consists of :
  - ✓ Primary Variable Capsule layer
    - ↕ Variable to Static Routing algorithm
  - ✓ Primary Static Capsule layer
    - ↕ Dynamic Routing algorithm
  - ✓ Code Capsule layer

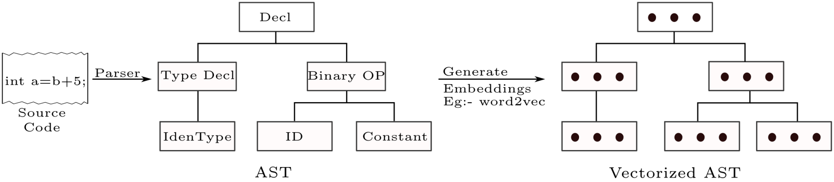
Abstract Syntax Tree Vectorization



- Every raw source code is parsed with an appropriate parser corresponding to the programming language to generate the AST[1].

[1] Python - Python AST parser, C & Java - srcML

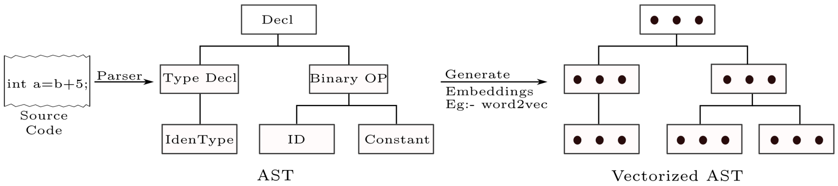
Abstract Syntax Tree Vectorization



- Every raw source code is parsed with an appropriate parser corresponding to the programming language to generate the AST[1].
- We use ASTs to train the embeddings by using techniques similar to Penget al. (2015), which learns a vectorized vocabulary of node types.

[1] Python - Python AST parser, C & Java - srcML

Abstract Syntax Tree Vectorization



- Every raw source code is parsed with an appropriate parser corresponding to the programming language to generate the AST[1].
- We use ASTs to train the embeddings by using techniques similar to Penget al. (2015), which learns a vectorized vocabulary of node types.
- The learned vocabulary can subsequently be used to vectorize each individual nodeof the ASTs, generating the vectorized ASTs.

[1] Python - Python AST parser, C & Java - srcML

- One of the main challenges in creating a tree-based capsule network is that the input of the network is tree-structured.
  - ✓ Image data $\in \mathbb{R}^{H \times W \times C}$, where H, W, C are fixed.
  - ✓ Natural language data $\in \mathbb{R}^{L \times E}$, where L, E are the fixed.
  - × Tree-structured data $\in \mathbb{R}^{T \times V}$, where T is **dynamic**.

TreeCaps : Tree Structured Convolution

- One of the main challenges in creating a tree-based capsule network is that the input of the network is tree-structured.
  - ✓ Image data $\in \mathbb{R}^{H \times W \times C}$, where H, W, C are fixed.
  - ✓ Natural language data $\in \mathbb{R}^{L \times E}$, where L, E are the fixed.
  - × Tree-structured data $\in \mathbb{R}^{T \times V}$, where T is **dynamic**.

- A further challenge is that the # of children varies from node to node.

- One of the main challenges in creating a tree-based capsule network is that the input of the network is tree-structured.
  - ✓ Image data $\in \mathbb{R}^{H \times W \times C}$, where H, W, C are fixed.
  - ✓ Natural language data $\in \mathbb{R}^{L \times E}$, where L, E are the fixed.
  - × Tree-structured data $\in \mathbb{R}^{T \times V}$, where T is **dynamic**.

- A further challenge is that the # of children varies from node to node.

  Solutions :

  - Zero padding ?

TreeCaps : Tree Structured Convolution

- One of the main challenges in creating a tree-based capsule network is that the input of the network is tree-structured.
  - ✓ Image data $\in \mathbb{R}^{H \times W \times C}$, where H, W, C are fixed.
  - ✓ Natural language data $\in \mathbb{R}^{L \times E}$, where L, E are the fixed.
  - × Tree-structured data $\in \mathbb{R}^{T \times V}$, where T is **dynamic**.

- A further challenge is that the # of children varies from node to node.

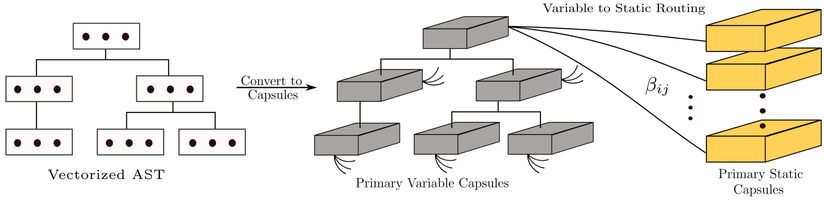
  Solutions :

  - Zero padding ?
  - Tree-based Convolution better

$$\mathbf{y} = tanh(\sum_{i=1}^{K+1} [\eta_i^t \mathbf{W}^t + \eta_i^l \mathbf{W}^l + \eta_i^r \mathbf{W}^r]\mathbf{x}_i + \mathbf{b}) \tag{1}$$

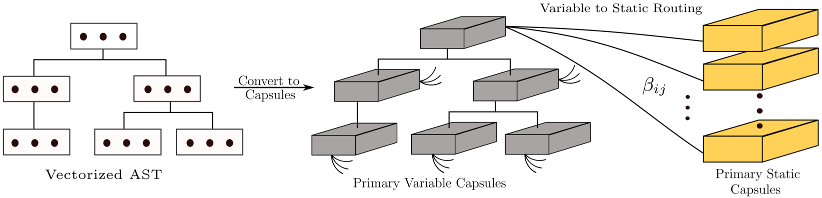
$\eta_i^t, \eta_i^l, \eta_i^r$ are weights defined corresponding to the depth and the position of the children nodes, and $\mathbf{Y}_{conv} \in \mathbb{R}^{T \times V'}$.

TreeCaps : Primary Variable TreeCaps Layer



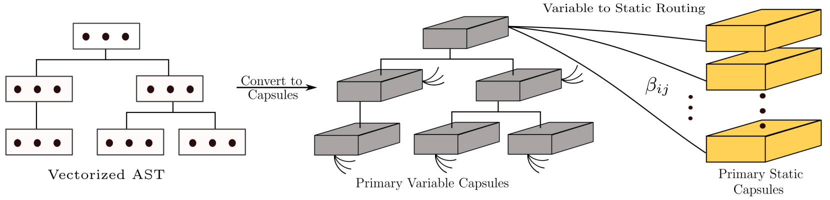
- **y** obtained from Eq 1 corresponds to the output of one convolutional slice. We use ε such slices with different initializations for $\mathbf{W}, \mathbf{b}$.

TreeCaps : Primary Variable TreeCaps Layer



- $\mathbf{y}$ obtained from Eq 1 corresponds to the output of one convolutional slice. We use ε such slices with different initializations for $\mathbf{W}, \mathbf{b}$.
- We group the convolutional slices together to form $N_{pvc} = \frac{T \times V' \times \varepsilon}{D_{pvc}}$ sets of capsules with outputs $\mathbf{u}_i \in \mathbb{R}^{D_{pvc}}$, $i \in [1, N_{pvc}]$, where D_{pvc} is the dimensions of the capsules in the PVC layer.

TreeCaps : Primary Variable TreeCaps Layer



- $\mathbf{y}$ obtained from Eq 1 corresponds to the output of one convolutional slice. We use ε such slices with different initializations for $\mathbf{W}, \mathbf{b}$.
- We group the convolutional slices together to form $N_{pvc} = \frac{T \times V' \times \varepsilon}{D_{pvc}}$ sets of capsules with outputs $\mathbf{u}_i \in \mathbb{R}^{D_{pvc}}$, $i \in [1, N_{pvc}]$, where D_{pvc} is the dimensions of the capsules in the PVC layer.
- To vectorize each capsule output, we subsequently apply a non-linear squash function, producing the output of the PVC layer $\in \mathbb{R}^{N_{pvc} \times D_{pvc}}$.

TreeCaps : Primary Static TreeCaps Layer

- The key issue with passing the outputs of the PVC layer to the Code Capsule layer is that N_{pvc} is variable with the training sample.

TreeCaps : Primary Static TreeCaps Layer

- The key issue with passing the outputs of the PVC layer to the Code Capsule layer is that N_{pvc} is variable with the training sample.
- Prior to routing the lower level capsules to a set of higher level capsules, the lower dimensional capsule outputs need to be projected to the higher dimensionality, with a transformation matrix which learns the part-whole relationship between the lower and the higher level capsules.

- The key issue with passing the outputs of the PVC layer to the Code Capsule layer is that N_{pvc} is variable with the training sample.

- Prior to routing the lower level capsules to a set of higher level capsules, the lower dimensional capsule outputs need to be projected to the higher dimensionality, with a transformation matrix which learns the part-whole relationship between the lower and the higher level capsules.

- However, a trainable transformation matrix cannot be defined in practice with variable dimensions. Thus, the dynamic routing in cannot be applied here.

- The key issue with passing the outputs of the PVC layer to the Code Capsule layer is that N_{pvc} is variable with the training sample.
- Prior to routing the lower level capsules to a set of higher level capsules, the lower dimensional capsule outputs need to be projected to the higher dimensionality, with a transformation matrix which learns the part-whole relationship between the lower and the higher level capsules.
- However, a trainable transformation matrix cannot be defined in practice with variable dimensions. Thus, the dynamic routing in cannot be applied here.

  Solution : Proposed Variable to Static Routing Algorithm

TreeCaps : Variable to Static Routing Algorithm

Algorithm 1 Variable-to-Static Capsule Routing

procedure ROUTING($\hat{\mathbf{u}}_i, r, a, b$)

  $\hat{\mathbf{U}}_{\mathbf{sorted}} \leftarrow sort([\hat{\mathbf{u}}_1, ..., \hat{\mathbf{u}}_{N_{pvc}}])$

  Initialize $\mathbf{v}_j : \forall i, j \leq a, \mathbf{v_j} \leftarrow \hat{\mathbf{U}}_{\mathbf{sorted}}[i]$

  Initialize $\alpha_{ij} : \forall j \in [1, a], \forall i \in [1, b], \alpha_{ij} \leftarrow 0$

  for r iterations **do**

    $\forall j \in [1, a], \forall i \in [1, b], f_{ij} \leftarrow \hat{\mathbf{u}}_i \cdot \mathbf{v}_j$

    $\forall j \in [1, a], \forall i \in [1, b], \alpha_{ij} \leftarrow \alpha_{ij} + f_{ij}$

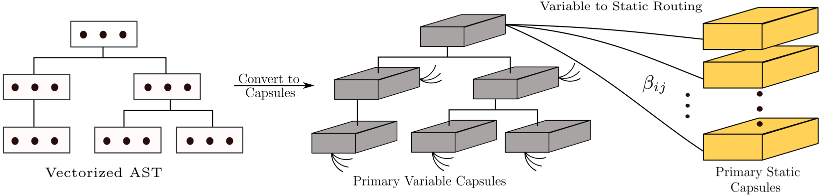
    $\forall i \in [1, b], \boldsymbol{\beta}_i \leftarrow Softmax(\boldsymbol{\alpha}_i)$

    $\forall j \in [1, a], \mathbf{s}_j \leftarrow \sum_i \beta_{ij} \hat{\mathbf{u}}_i$

    $\forall j \in [1, a], \mathbf{v}_j \leftarrow Squash(\mathbf{s}_j)$

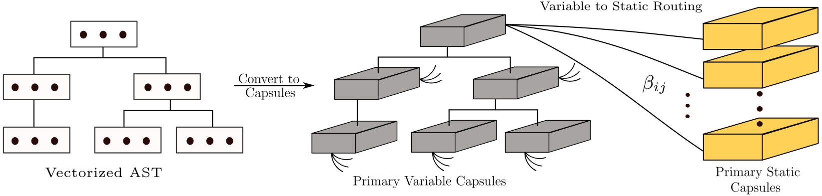
  return $\mathbf{v}_j$

TreeCaps : Variable to Static Routing Algorithm



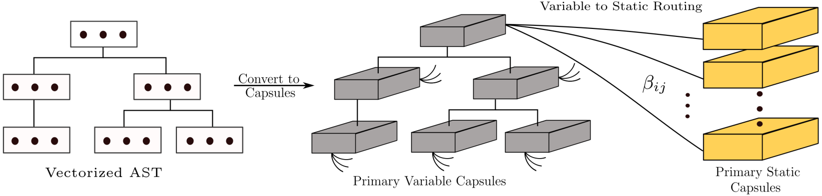
- Often, source code consists of non-essential entities, and only a portion of all entities determine the code class.

TreeCaps : Variable to Static Routing Algorithm



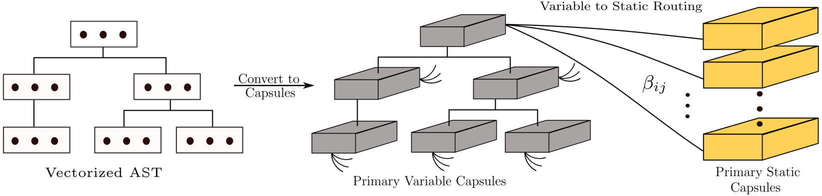
- Often, source code consists of non-essential entities, and only a portion of all entities determine the code class.
- $||\mathit{Capsule\ output}||_2 \propto \mathit{Prob.\ of\ existence}$.

TreeCaps : Variable to Static Routing Algorithm



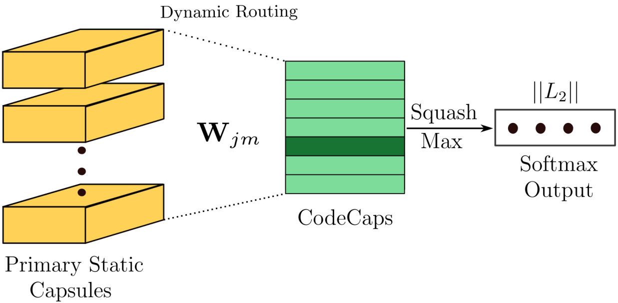
- Often, source code consists of non-essential entities, and only a portion of all entities determine the code class.
- $||Capsule\ output||_2 \propto Prob.\ of\ existence.$
- Dependency relationships may exist among entities that are not spatially co-located.

TreeCaps : Variable to Static Routing Algorithm



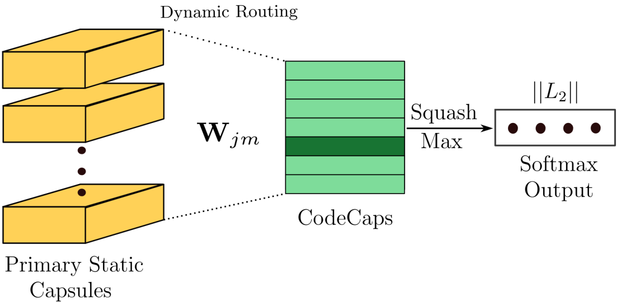
- Often, source code consists of non-essential entities, and only a portion of all entities determine the code class.
- $||\mathit{Capsule\ output}||_2 \propto \mathit{Prob.\ of\ existence}$.
- Dependency relationships may exist among entities that are not spatially co-located.
- Routing by agreement $\uparrow \cdot \uparrow = (+)$ $\quad \uparrow \cdot \rightarrow = (0)$ $\quad \uparrow \cdot \downarrow = (-)$.

TreeCaps : Code Capsule Layer



- Code Capsule layer is the final layer of the TreeCaps network, which acts as the classification capsule layer.

TreeCaps : Code Capsule Layer



- Code Capsule layer is the final layer of the TreeCaps network, which acts as the classification capsule layer.
- Since the output of the PSC layer is a fixed set of capsules, it can be routed to the CC layer with dynamic routing.

TreeCaps : Margin Loss

- For every code capsule μ, the margin loss L_μ is defined as follows,

$$L_\mu = T_\mu \max(0, m^+ - \|v_\mu\|)^2 + \lambda(1 - T_\mu) \max(0, \|v_\mu\| - m^-)^2 \quad (2)$$

- T_μ is 1 if the correct class is μ and zero otherwise.
- λ is set to 0.5 to control the initial learning from shrinking the length of the output vectors of all the code capsules.
- m^+, m^- are set to $0.9, 0.1$ as the lower bound for the correct class and the upper bound for the incorrect class respectively.

Presentation Outline

Datasets

- **Dataset A :** Python 6 classes of sorting algorithms, with 346 training programs on average per class.
- **Dataset B :** Java 10 classes of sorting algorithms, with 64 training programs on average per class.
- **Dataset C :** C 104 classes, with 375 training programs on average per class.

Quantitative Results

- The means and the standard deviations from 3 trials are shown.

Model	**Dataset A**	**Dataset B**	**Dataset C**
GGNN	-	85.00%	86.52%
TBCNN	99.30%	75.00%	79.40%
TreeCaps	$100.00 \pm 0.00\%$	$92.11 \pm 0.90\%$	$87.95 \pm 0.23\%$
TreeCaps (3-ens.)	**100.00%**	**94.08%**	**89.41%**

Quantitative Results

- The means and the standard deviations from 3 trials are shown.

Model	**Dataset A**	**Dataset B**	**Dataset C**
GGNN	-	85.00%	86.52%
TBCNN	99.30%	75.00%	79.40%
TreeCaps	$100.00 \pm 0.00\%$	$92.11 \pm 0.90\%$	$87.95 \pm 0.23\%$
TreeCaps (3-ens.)	**100.00%**	**94.08%**	**89.41%**

- We followed the techniques proposed in Allamanis et al. and BUI et al. to re-generate the results for GGNN and the techniques proposed in Mou et al. to re-generate the results for TBCNN.

Quantitative Results

- The means and the standard deviations from 3 trials are shown.

Model	Dataset A	Dataset B	Dataset C
GGNN	-	85.00%	86.52%
TBCNN	99.30%	75.00%	79.40%
TreeCaps	$100.00 \pm 0.00\%$	$92.11 \pm 0.90\%$	$87.95 \pm 0.23\%$
TreeCaps (3-ens.)	**100.00%**	**94.08%**	**89.41%**

- We followed the techniques proposed in Allamanis et al. and BUI et al. to re-generate the results for GGNN and the techniques proposed in Mou et al. to re-generate the results for TBCNN.
- Why Mou et al. reports a higher performance for Dataset C than us?
  - ✓ Custom-trained initial embeddings
  - ✓ A small set of AST node types defined specifically for C language only

Quantitative Results

- The means and the standard deviations from 3 trials are shown.

Model	Dataset A	Dataset B	Dataset C
GGNN	-	85.00%	86.52%
TBCNN	99.30%	75.00%	79.40%
TreeCaps	$100.00 \pm 0.00\%$	$92.11 \pm 0.90\%$	$87.95 \pm 0.23\%$
TreeCaps (3-ens.)	**100.00%**	**94.08%**	**89.41%**

- We followed the techniques proposed in Allamanis et al. and BUI et al. to re-generate the results for GGNN and the techniques proposed in Mou et al. to re-generate the results for TBCNN.
- Why Mou et al. reports a higher performance for Dataset C than us?
  - ✓ Custom-trained initial embeddings
  - ✓ A small set of AST node types defined specifically for C language only
- For a fairer comparison (B & C), we used general embeddings based on srcML node vocabulary as the initial embeddings across all models.

Model Analysis

Model Variant	Accuracy
Variable-to-Static Routing Algorithm $\rightarrow$ Dynamic Pooling	83.43%
Instantiation parameters $\rightarrow D_{cc} = 4$	90.90%
$D_{cc} = 8$	92.10%
$D_{cc} = 12$	90.33%
$D_{cc} = 16$	91.51%
TreeCaps $\rightarrow$ TreeCaps + Secondary Capsule Layer	92.31%
TreeCaps with Variable-to-Static Routing and $D_{cc} = 8$	92.11%

- Dynamic max pooling is bad for capsule networks, as it destroys spatial/dependency relationships.

Model Analysis

Model Variant	Accuracy
Variable-to-Static Routing Algorithm $\rightarrow$ Dynamic Pooling	83.43%
Instantiation parameters $\rightarrow D_{cc} = 4$	90.90%
$D_{cc} = 8$	92.10%
$D_{cc} = 12$	90.33%
$D_{cc} = 16$	91.51%
TreeCaps $\rightarrow$ TreeCaps + Secondary Capsule Layer	92.31%
TreeCaps with Variable-to-Static Routing and $D_{cc} = 8$	92.11%

- Dynamic max pooling is bad for capsule networks, as it destroys spatial/dependency relationships.
- The instantiation parameters D_{cc} of the CC layer acts as the dimensionality of the latent representation of source code.
  $D_{cc} \uparrow\uparrow$ - Sparsity and/or correlated instantiation parameters
  $D_{cc} \downarrow\downarrow$ - Under-representation

Presentation Outline

Limitations

- Limitations inherited from capsule networks
  - × High computational complexity in comparison to CNNs.
  - × Relative performance reduction with the increasing number of classes.

Limitations

- Limitations inherited from capsule networks
  - × High computational complexity in comparison to CNNs.
  - × Relative performance reduction with the increasing number of classes.
- TreeCaps lacks a decoder network, due to which
  - × We loose a lot of interpretability.
  - × We cannot study the relationship between the learnt instantiation parameters and the physical attributes of data.

Future Work

- Investigate the extent to which TreeCaps can actually capture the dependency relationships of ASTs. How?

Future Work

- Investigate the extent to which TreeCaps can actually capture the dependency relationships of ASTs. How?
  - ✓ Integrate a back-tracking mechanism after a forward pass with the test case.

Future Work

- Investigate the extent to which TreeCaps can actually capture the dependency relationships of ASTs. How?
  - ✓ Integrate a back-tracking mechanism after a forward pass with the test case.
  - ✓ For a given primary static capsule, the primary variable capsules connected to it with ϕ-highest coupling coefficients are considered to have dependency relationships.

Future Work

- Investigate the extent to which TreeCaps can actually capture the dependency relationships of ASTs. How?
  - ✓ Integrate a back-tracking mechanism after a forward pass with the test case.
  - ✓ For a given primary static capsule, the primary variable capsules connected to it with ϕ-highest coupling coefficients are considered to have dependency relationships.
  - ✓ We subsequently compare related pieces of code identified by TreeCaps to program dependencies identified by program analysis techniques.

Future Work

- Investigate the extent to which TreeCaps can actually capture the dependency relationships of ASTs. How?
  - ✓ Integrate a back-tracking mechanism after a forward pass with the test case.
  - ✓ For a given primary static capsule, the primary variable capsules connected to it with ϕ-highest coupling coefficients are considered to have dependency relationships.
  - ✓ We subsequently compare related pieces of code identified by TreeCaps to program dependencies identified by program analysis techniques.
- Evaluate the effectiveness of TreeCaps as an embedding generating technique.

Future Work

- Investigate the extent to which TreeCaps can actually capture the dependency relationships of ASTs. How?
  - ✓ Integrate a back-tracking mechanism after a forward pass with the test case.
  - ✓ For a given primary static capsule, the primary variable capsules connected to it with ϕ-highest coupling coefficients are considered to have dependency relationships.
  - ✓ We subsequently compare related pieces of code identified by TreeCaps to program dependencies identified by program analysis techniques.
- Evaluate the effectiveness of TreeCaps as an embedding generating technique.
- Extend TreeCaps to other related tasks such as bug detection and localization.

Supplementary

This work was accepted to be presented at NeurIPS workshops this year!

Thank you!