

ESEC/FSE '18, November 4–9, 2018, Lake Buena Vista, FL, USA

---

# DeepSim : Deep Learning Code Functional Similarity

Gang Zhao  
Jeff Huang

---

発表者：作島大智

---

# 背景

---

- ❖ コードの類似性を測定することは多くのソフトウェアエンジニアリングの活動に欠かせない
- ❖ 構造的類似性のみ焦点がしぼられており、**機能的類似性**を測定することはいまだに困難

---

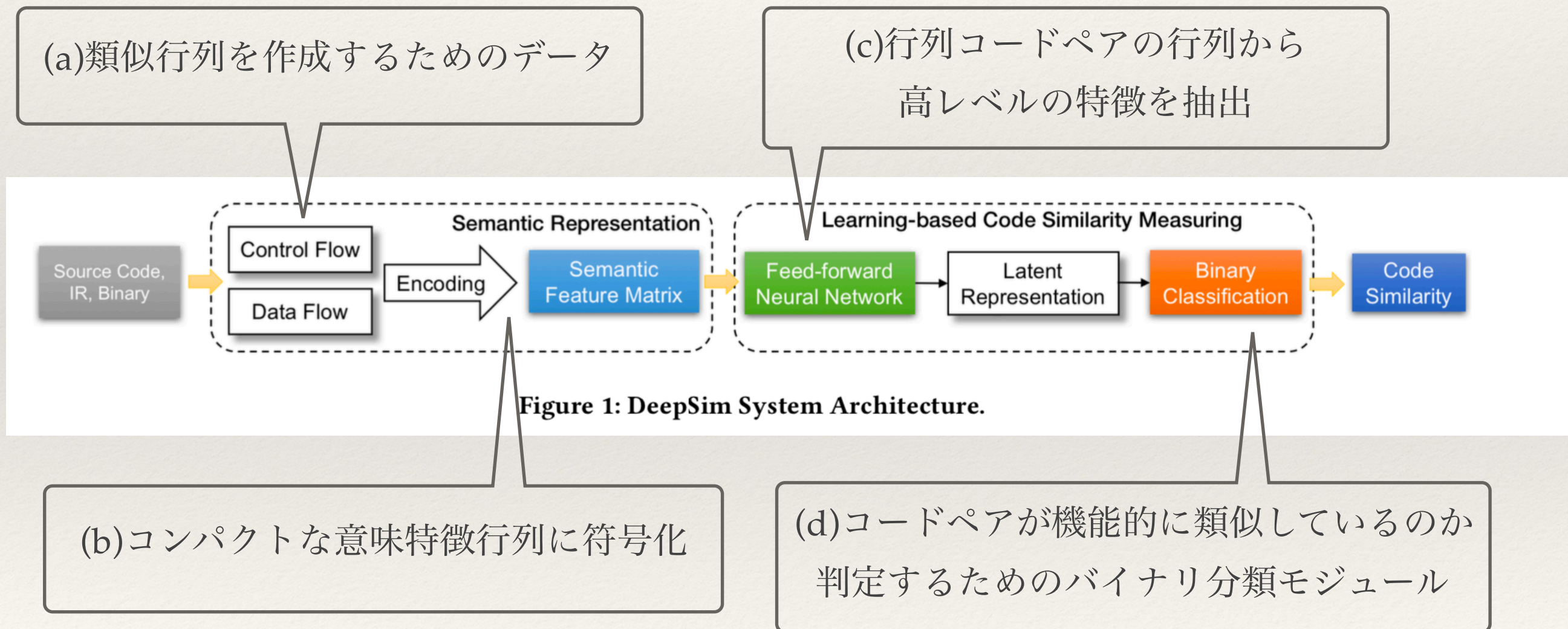
# 提案手法

---

- ❖ 類似性を測定精度を向上させるDeepSim for Javaを実装
- ❖ 機能的に類似している2つのデータセットで評価し、既存の最先端技術と比較する

# DeepSim

## ❖ 類似性を測定精度を向上させるためのアプローチ



# (a)類似行列の元データ

---

- ❖ 制御フロー解析とデータフロー解析を行う
- ❖ 前者はコードの基本ブロックとプロシージャー間の依存関係を，後者はデータ値の流れを捉える
- ❖ 3つの機能に着目  
変数機能，基本ブロック機能，2者間の関係機能

# (b)意味行列エンコード

## ❖ 11個の変数と6つの基本ブロックを符号化

データフローと制御フロー

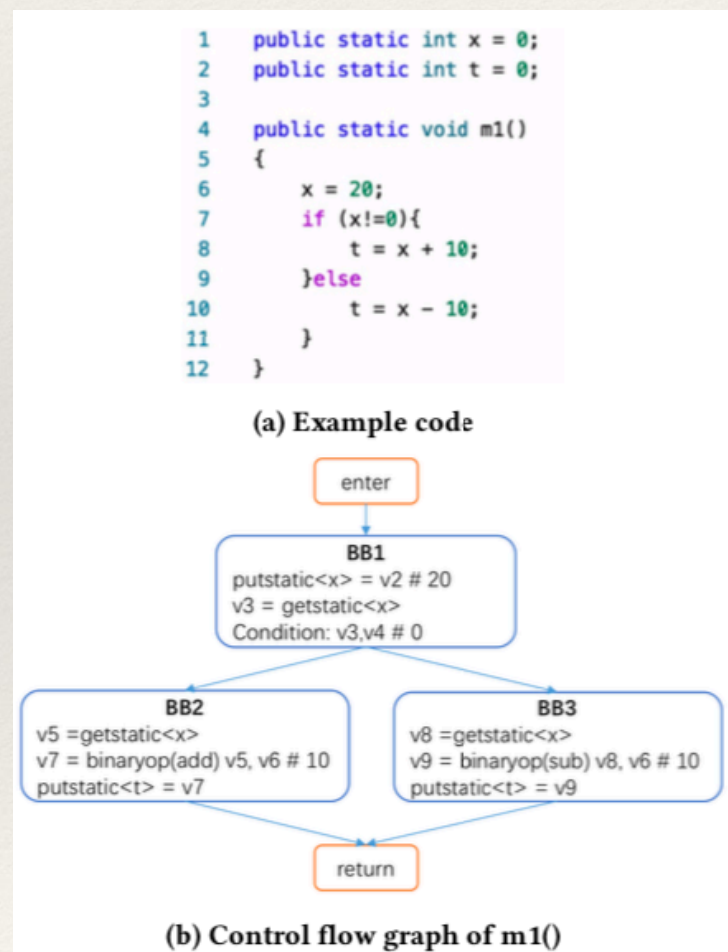


Figure 2: Control flow and data flow example.

特徴意味行列

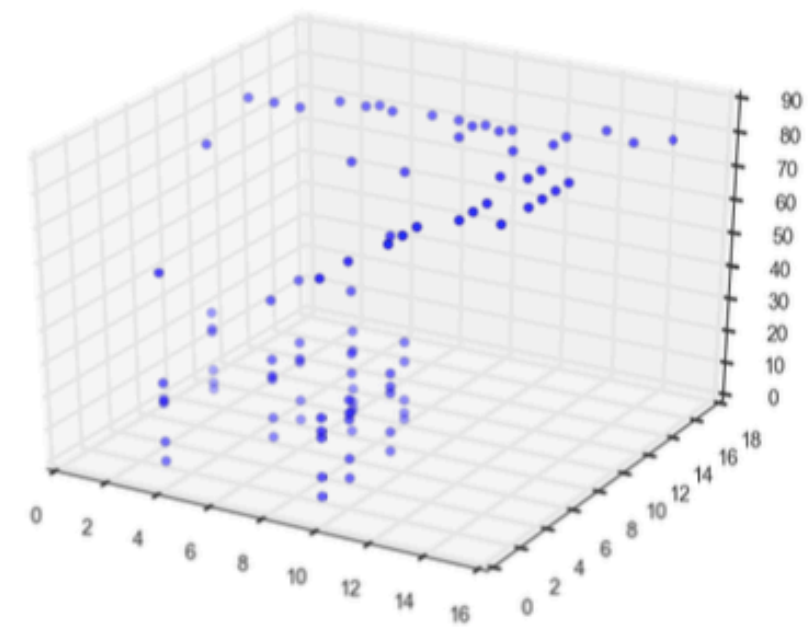


Figure 3: Semantic features matrix (17×17) generated from method *m1* in Figure 2. Along *z* axis are the 88d binary feature vectors. The value 1 is represented by a blue dot, and 0 represented by empty.

---

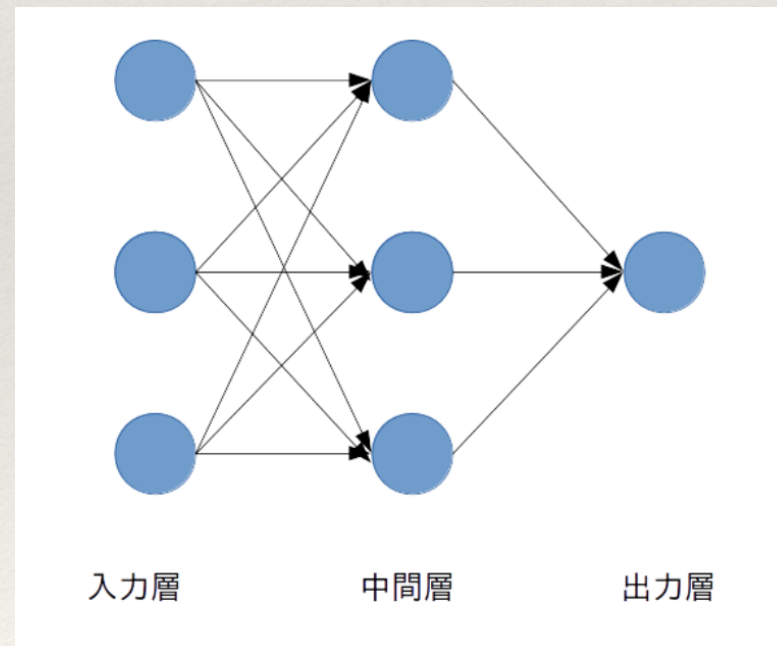
# エンコードの意義

---

- ❖ コード類似性測定問題を行列内の類似パターンを識別することに縮小し，スケーラブルにする
- ❖ 単一行列に変換することで後のプロセス(クラスタリングやニューラルネットワーク)で使いやすくなる

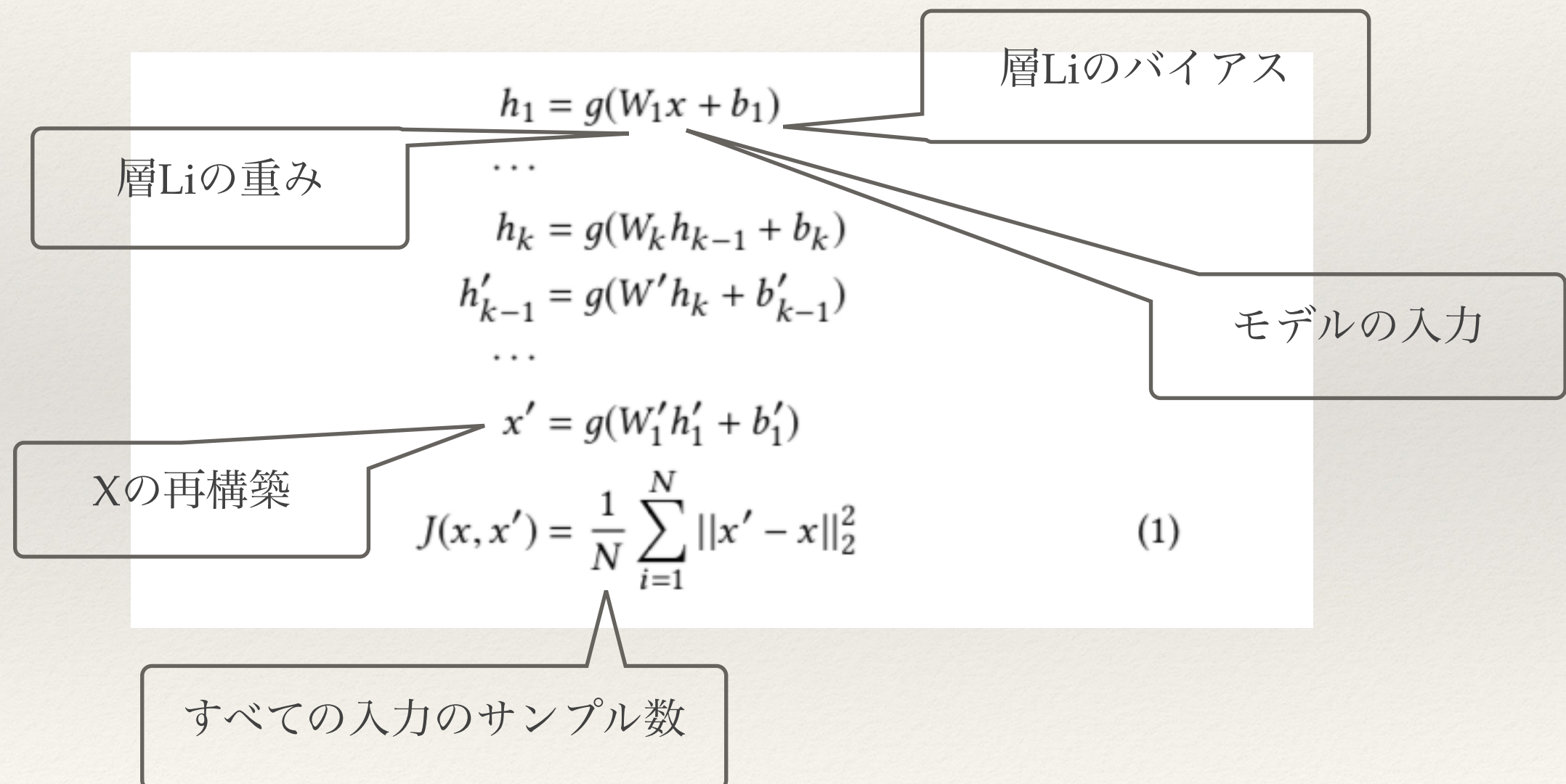
# (c) Feed-Forward Neural Network

- ❖ 普遍的で強力な近似手法
- ❖ データの流れが一方方向  
→ データが行ったり来たり，ループしたりしない



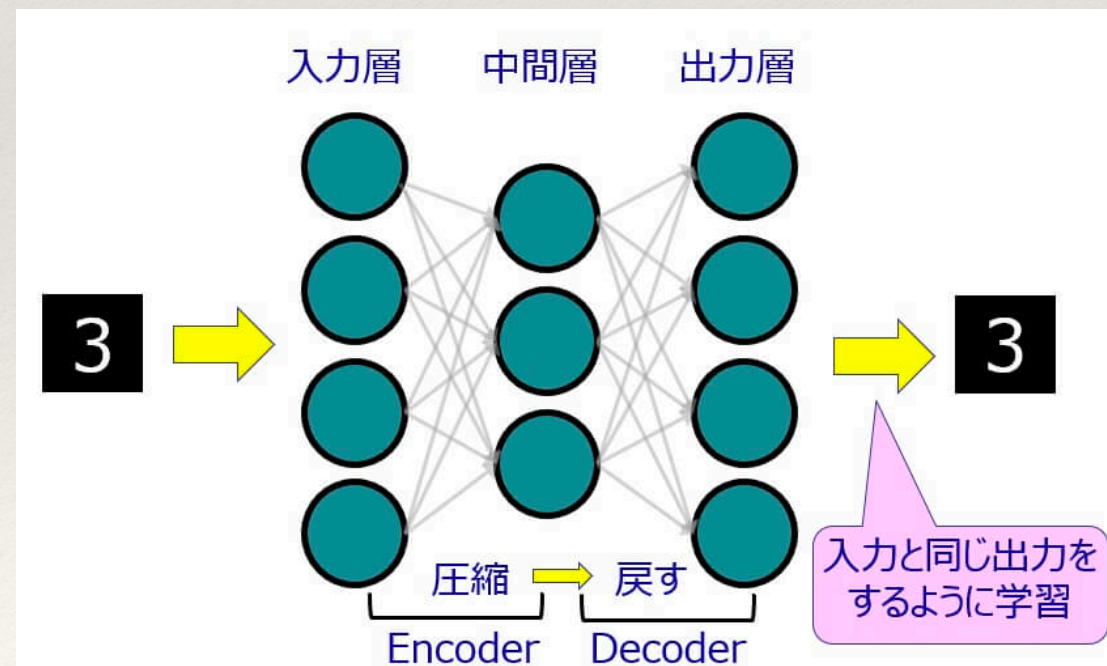
# Deep AutoEncoder

## ❖ Feed-Forward Neural Networkの代表的な手法



# AutoEncoder

- ❖ 機械学習においてニューラルネットワークを使用した次元圧縮のためのアルゴリズム
- ❖ 正解データだけを用いて正解データを出力する学習モデル  
→ 正常とは違うパターンは復元できない



# Deep AutoEncoder

SdA

$$h' = g(W \cdot \varepsilon(x) + b)$$

$$x'' = g(W'h' + b')$$

$$J(x'', x) = \frac{1}{N} \sum_{i=1}^N \|x'' - x\|_2^2 \quad (2)$$

❖ Stacked denoising AutoEncoder (SdA)

→再構成誤差を最小にするため

---

# Model Training

---

- ❖ (d)検証データと出力結果を比較し，重みやバイアスを修正する誤差逆伝播法でモデルトレーニングを行う
- 1, 現在の重みを用いて出力を生成
- 2, 検証データと1で各ニューロンの誤差を計算
- 3, 誤差を用いて勾配を計算し，学習率で重みを更新

# Model Training

教師データ

予測ラベル

$$J(\theta) = -\frac{1}{N} \sum_{i=1}^N S^{(i)} \log \tilde{S}^{(i)} + (1 - S^{(i)}) \log(1 - \tilde{S}^{(i)}) \quad (3)$$

$J(\theta)$ を最適化するために勾配降下を適用してWとbを更新できる

各更新のバッチ処理サンプルのサイズ

$$\theta := \theta - \alpha \cdot \frac{1}{n} \sum \frac{\partial J(\theta)}{\partial \theta}$$

学習率

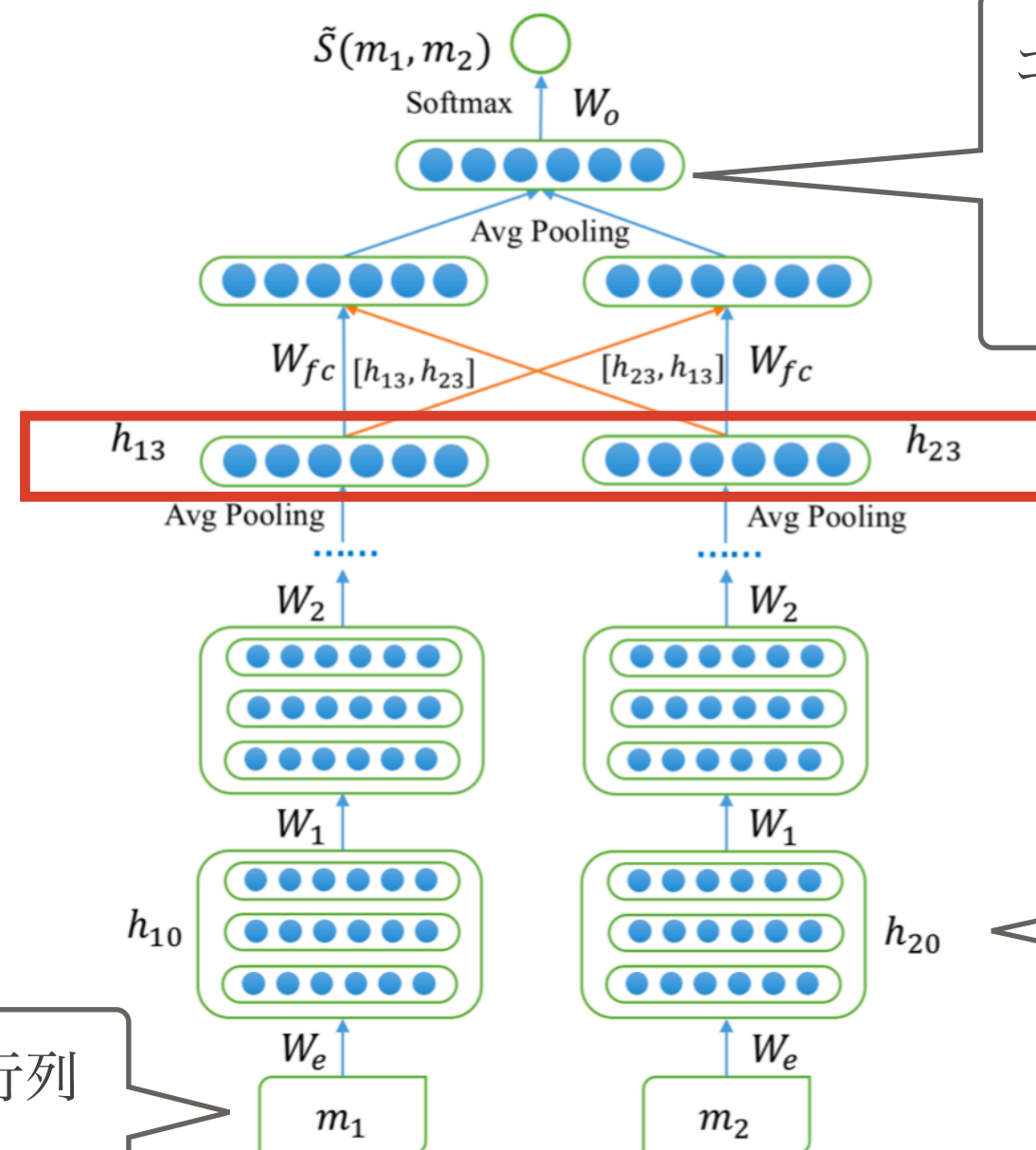
---

# 最適化

---

- ❖ 構造的差異があり機能的に類似するコードをエンコードした際、異なる意味行列が生成される場合がある
- ❖ 類似パターンを効果的に判別することが可能な  
深層モデル(Feature Learning Model)の開発を行う

# Feature Learning Model



コードステートメントの並び替え  
による影響を軽減して  
特徴表現の統合

特徴表現

2つの関係特徴ベクトル間の  
類似性を学習することが可能

固定サイズの意味特徴行列

Figure 4: A schematic of our feed-forward model for measuring code functional similarity.

# 並び替え

```
1      int c = 2;  
2      int x = 2*c;  
3      int y = 3*c;  
4      x = 2*y;  
5      y = x*y;
```

```
1      int c = 2;  
2      int y = 3*c;  
3      int x = 2*c;  
4      y = x*y;  
5      x = 2*y;
```

- ❖ 2, 3行は並び替えても結果は同じ
- ❖ 4, 5行を並び替えると違う結果になる
- ❖ 依存関係を持つ変数, 基本ブロックの順序のみ保持  
→全ての行に対して平均プーリングを行い, 残りの作業はモデルにて近似

# DeepSimの評価

❖ 2つの評価実験を以下のデータセットを用いて行う.

実験 1 : Google Code Jam (GCJ) [2]

実験 2 : BigCloneBench [53]

**Table 1: The Google Code Jam dataset.**

| Dataset statistics      | Value     |
|-------------------------|-----------|
| Projects                | 1,669     |
| Total lines of code     | 98,117    |
| Tokens                  | 445,371   |
| Vocabulary size         | 4,803     |
| Similar method pairs    | 275,570   |
| Dissimilar method pairs | 1,116,376 |

25,000のシステムから収集された  
6,000,000以上のタグ付きクローンペアと  
260,000の偽クローンペアを含む  
大規模コードクローンベンチマーク

---

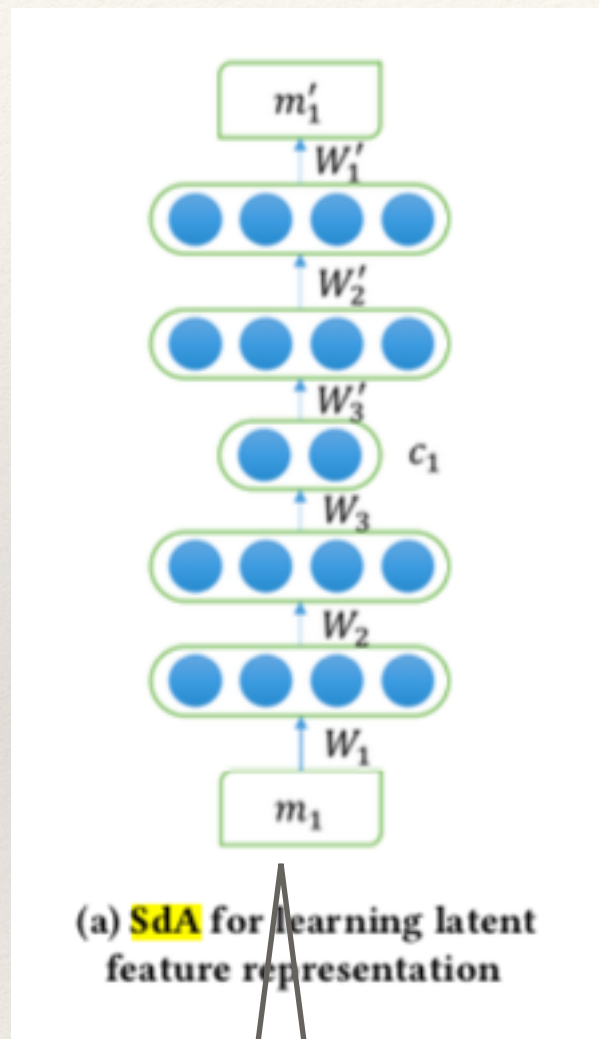
# 比較対象

---

- ❖ DECKARD : 古典的なASTベースの手法
- ❖ RtvNN : ユークリッド距離を使用してコードの類似性を測定する手法
- ❖ CDLH : ASTにLSTMを適用する最近の機械学習ベースの機能的クローン検出器

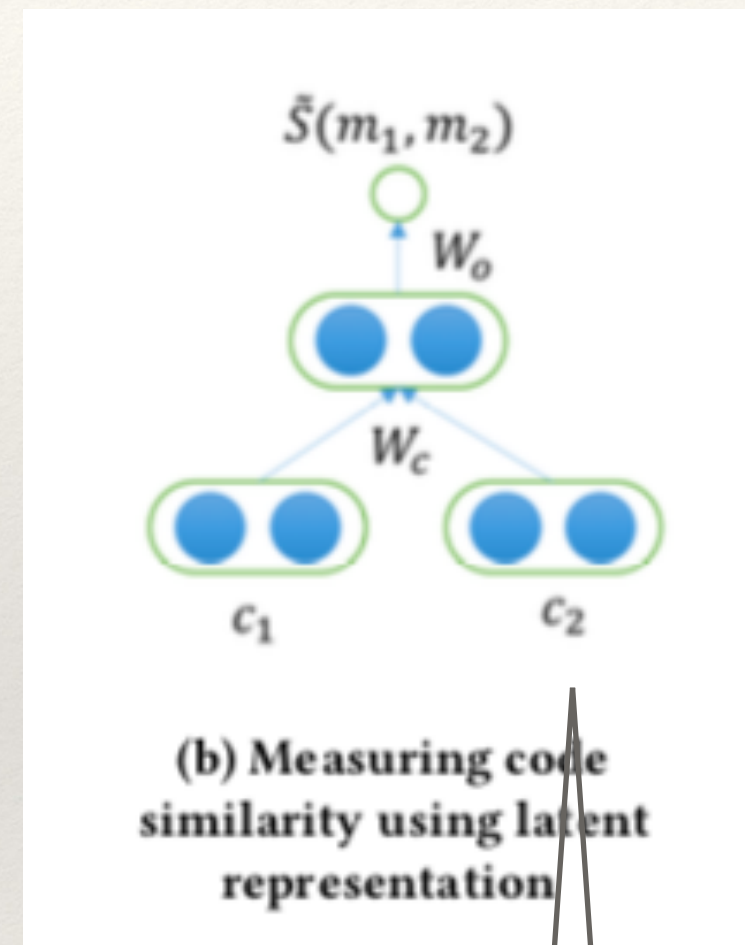
# 比較対象

SdA-base (a+b)



メソッドから生成された  
意味特徴行列

SdA-unsup(b)



メソッドペアからSdAによって  
生成された特徴ベクトル

# 実験 1

## ❖ DeepSimの有用性が示された

Table 3: Results on the GCJ dataset.

| Tool      | Recall      | Precision   | F1 score    |
|-----------|-------------|-------------|-------------|
| DECKARD   | 0.44        | 0.45        | 0.44        |
| SdA-base  | 0.51        | 0.50        | 0.50        |
| SdA-unsup | 0.56        | 0.26        | 0.35        |
| RtvNN     | <b>0.90</b> | 0.20        | 0.33        |
| DeepSim   | 0.82        | <b>0.71</b> | <b>0.76</b> |

全体の何割を検出することが  
できたのかを示す割合

検出された結果の中で  
実際に類似しているメソッドの割合

# 実験 2

## ❖ DeepSimの有用性が示された

Table 5: Results on BigCloneBench.

| Tools   | Recall      | Precision   | F1 Score    |
|---------|-------------|-------------|-------------|
| DECKARD | 0.02        | 0.93        | 0.03        |
| RtvNN   | 0.01        | 0.95        | 0.01        |
| CDLH    | 0.74        | 0.92        | 0.82        |
| DeepSim | <b>0.98</b> | <b>0.97</b> | <b>0.98</b> |

有用性あり

Table 6: F1 score for each clone type.

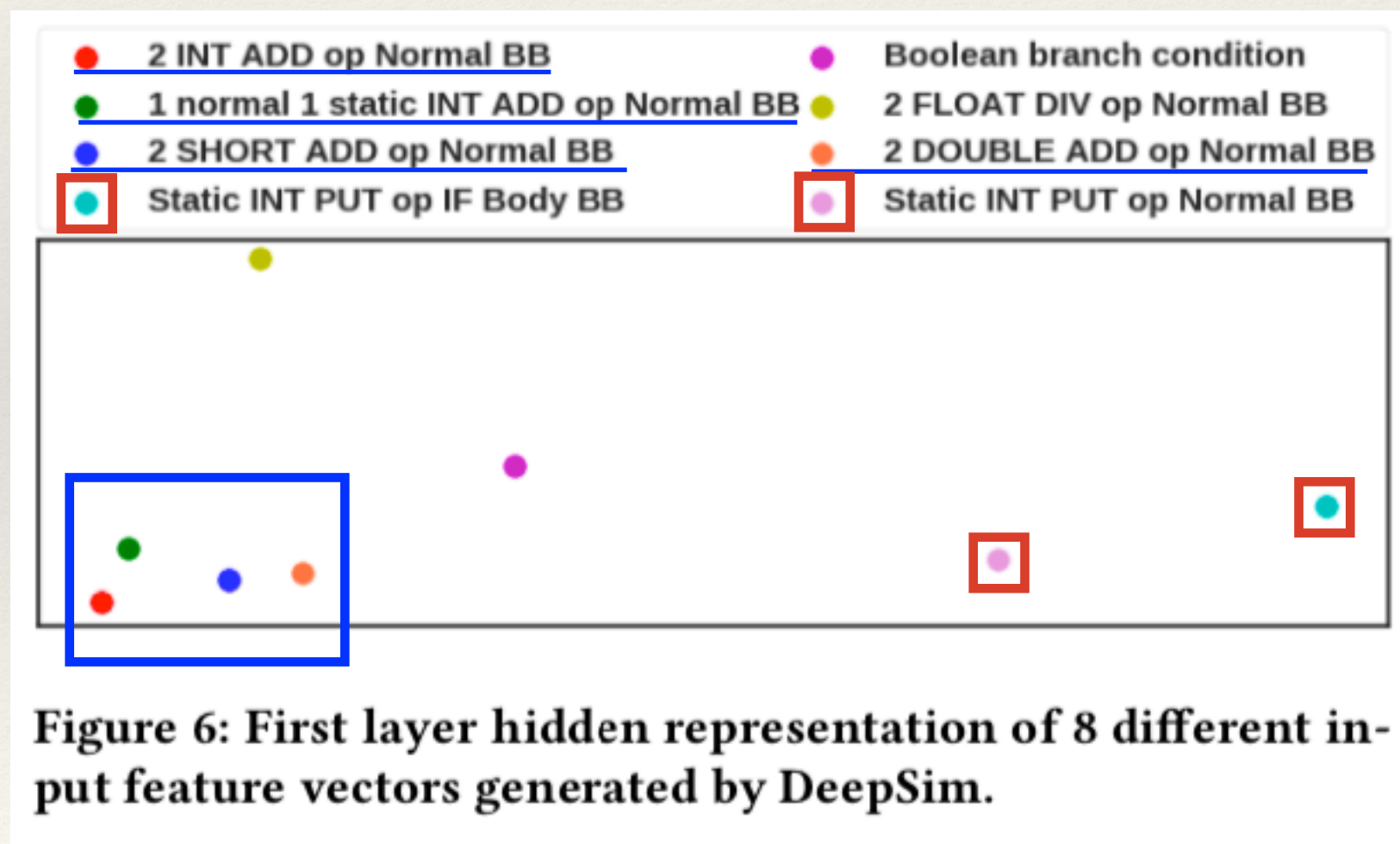
| Clone Type | DECKARD | RtvNN | CDLH        | DeepSim     |
|------------|---------|-------|-------------|-------------|
| T1         | 0.73    | 1.00  | <b>1.00</b> | 0.99        |
| T2         | 0.71    | 0.97  | <b>1.00</b> | 0.99        |
| ST3        | 0.54    | 0.60  | 0.94        | <b>0.99</b> |
| MT3        | 0.21    | 0.03  | 0.88        | <b>0.99</b> |
| WT3/T4     | 0.02    | 0.00  | 0.81        | <b>0.97</b> |

1にはならない

→エンコードの際、3つの機能に絞って近似しているから

# エンコードの有効性

- ❖ 機能的に類似しているコードが近い関係にあり、異なるブロックでの同一操作を区別できている。



---

# まとめ

---

- ❖ DeepSimは再現率と精度の点で既存のアプローチよりもはるかに優れていることを示した.
- ❖ 提唱したエンコード手法が有効である.

---

# 所感

---

- ❖ 知らない知識が多すぎて整理するのに時間を要した
- ❖ 次元の多い行列の話なので計算が多く，難しい
- ❖ とにかく難しくて時間がかかった．
- ❖ 難しかった．

---

# 参考文献

---

- ❖ ディープラーニングで簡単な時系列データの異常検知

[https://sinyblog.com/deaplearning/abnorm\\_detect\\_001/](https://sinyblog.com/deaplearning/abnorm_detect_001/)

- ❖ 定番のConvolutional Neural Networkをゼロから理解する

[https://deepage.net/deep\\_learning/2016/11/07/convolutional\\_neural\\_network.html](https://deepage.net/deep_learning/2016/11/07/convolutional_neural_network.html)

- ❖ これから始める人の為のディープラーニング講座

<https://www.slideshare.net/NVIDIAJapan/ss-75109746>