Seminar Software Analytics

Review of

Automatically Learning Semantic Features for Defect Prediction

Presentation Meeting

20.01.2017

Felix Schober

Seminar Software Analytics

Review of

Why Dreaming Networks Can Help a Company to Save Costs

Presentation Meeting

20.01.2017

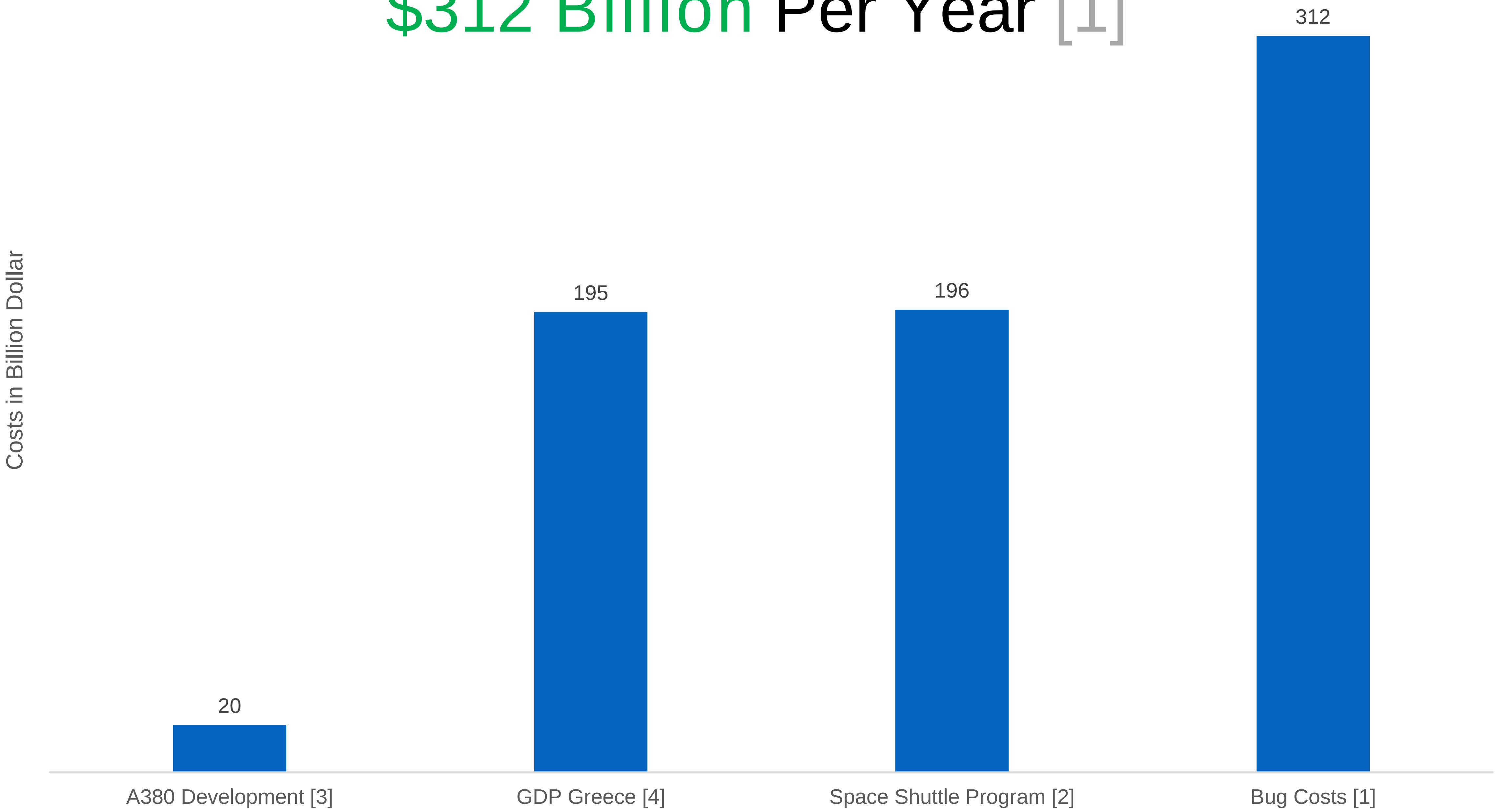
Felix Schober

Agenda

- Defect Prediction - Introduction
- Technical Background
- Discussion of Results
- Demo
- Conclusion

Cost of Defects

Global economy cost for bugs / software defects:
$312 Billion Per Year [1]



[1] F. Brady "Cambridge University Study States Software Bugs Cost Economy $312 Billion Per Year" prweb (2013) accessed 12.01.2017 http://www.prweb.com/releases/2013/1/prweb10298185.htm
[2] R. Pielke, R. Byerly "Shuttle Programme Lifetime Cost." In Nature Vol. 472 (2011)
[3] "Milliardengrab A380" Handelsblatt (2012) http://www.genios.de/presse-archiv/artikel/HB/20121108/milliardengrab-a380/2B70894B-6C35-4D7B-A956-B0B5783C418D.html
[4] World Bank – Last accessed 12.01.2017 http://data.worldbank.org/indicator/NY.GDP.MKTP.CD?locations=GR&name_desc=true

Defects in Software

"The application division at Microsoft experiences about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per 1000 lines of code in released product [...]." [5]

Windows Vista contains 50 Million Lines of code [6]

~25,000 Defects
in the released product

[5] S. McConnell "Code Complete 2nd edition." Microsoft Press (2004) page 521
[6] S. Lohr, J. Markoff "Windows Is So Slow, but Why?" New York Times (2006) Last accessed 12.01.2017 http://www.nytimes.com/2006/03/27/technology/27soft.html?adxnnl=1&pagewanted=all&adxnnlx=1382805118-0jnNRGXEVPip3xoW+BDp8Q

Defect Prediction

25,000 Defects
in the released product

50x – 200x
more expensive to fix a bug at a later stage [7]

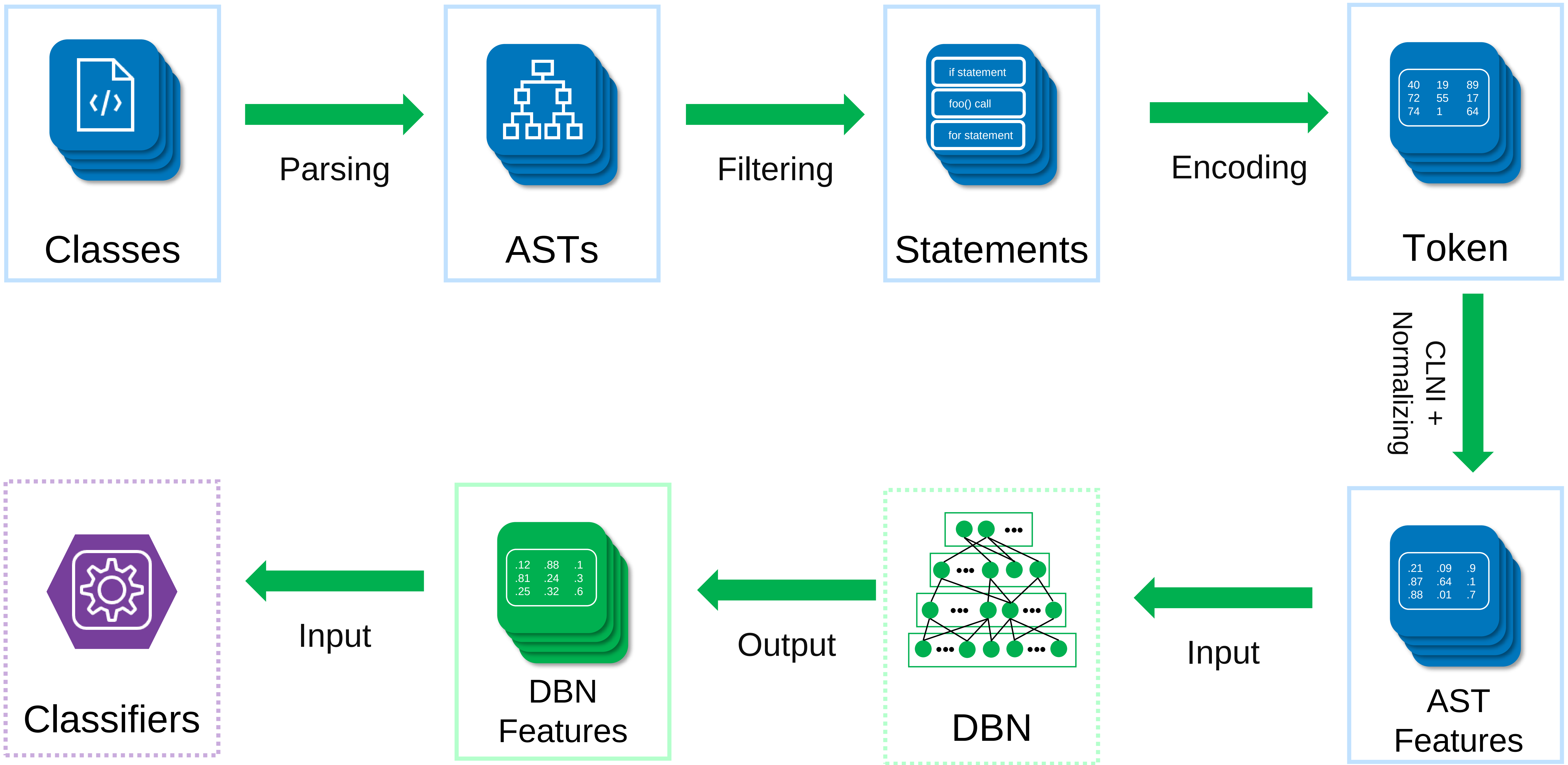
Automatically detect defects as quick as possible

[7] B. Boehm, P. Papaccio "Understanding and Controlling Software Costs." In: IEEE Transactions on Software Engineering (1988) page 1466

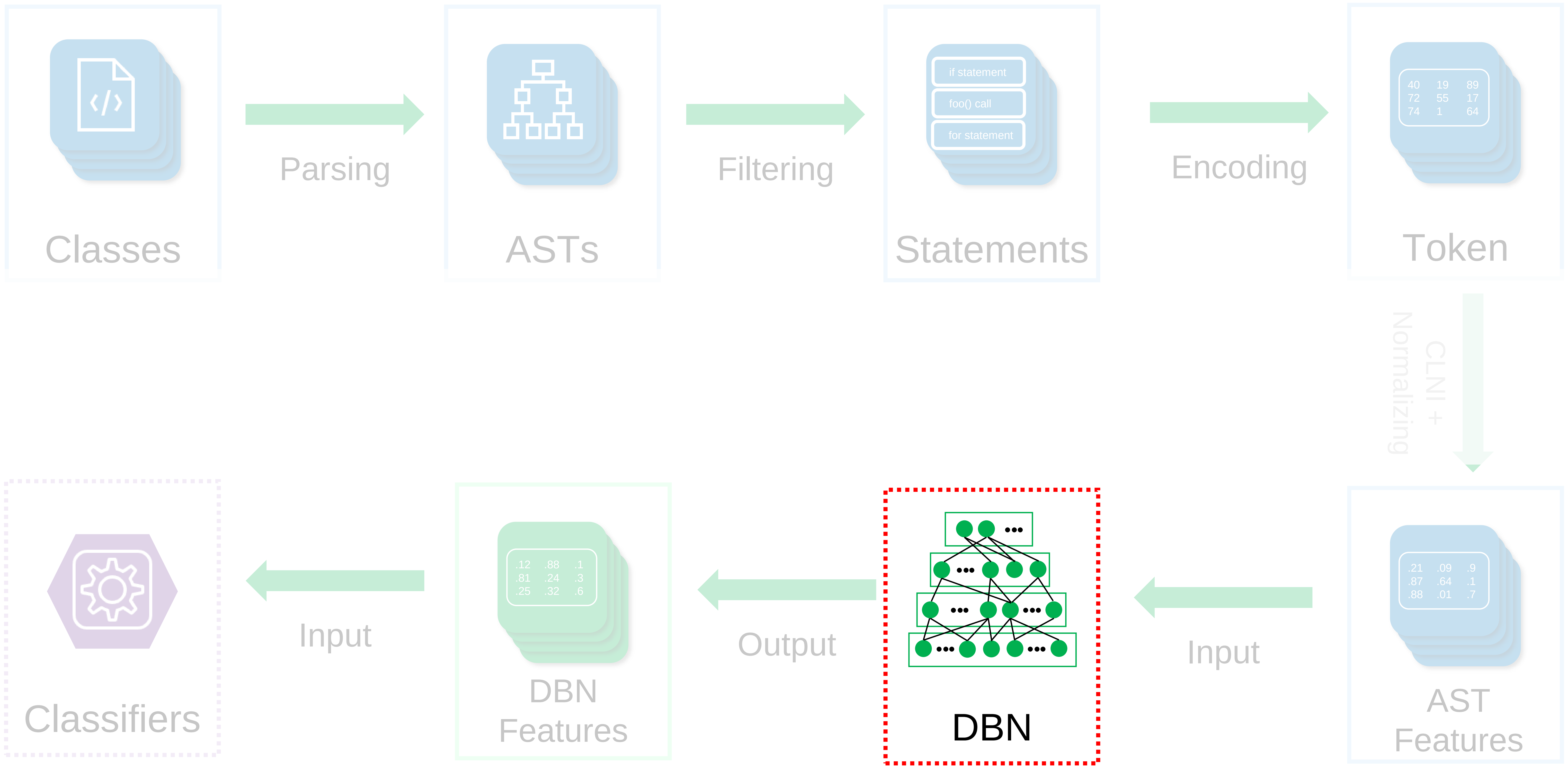
Agenda

- Defect Prediction
- Technical Background
- Discussion of Results
- Demo
- Conclusion

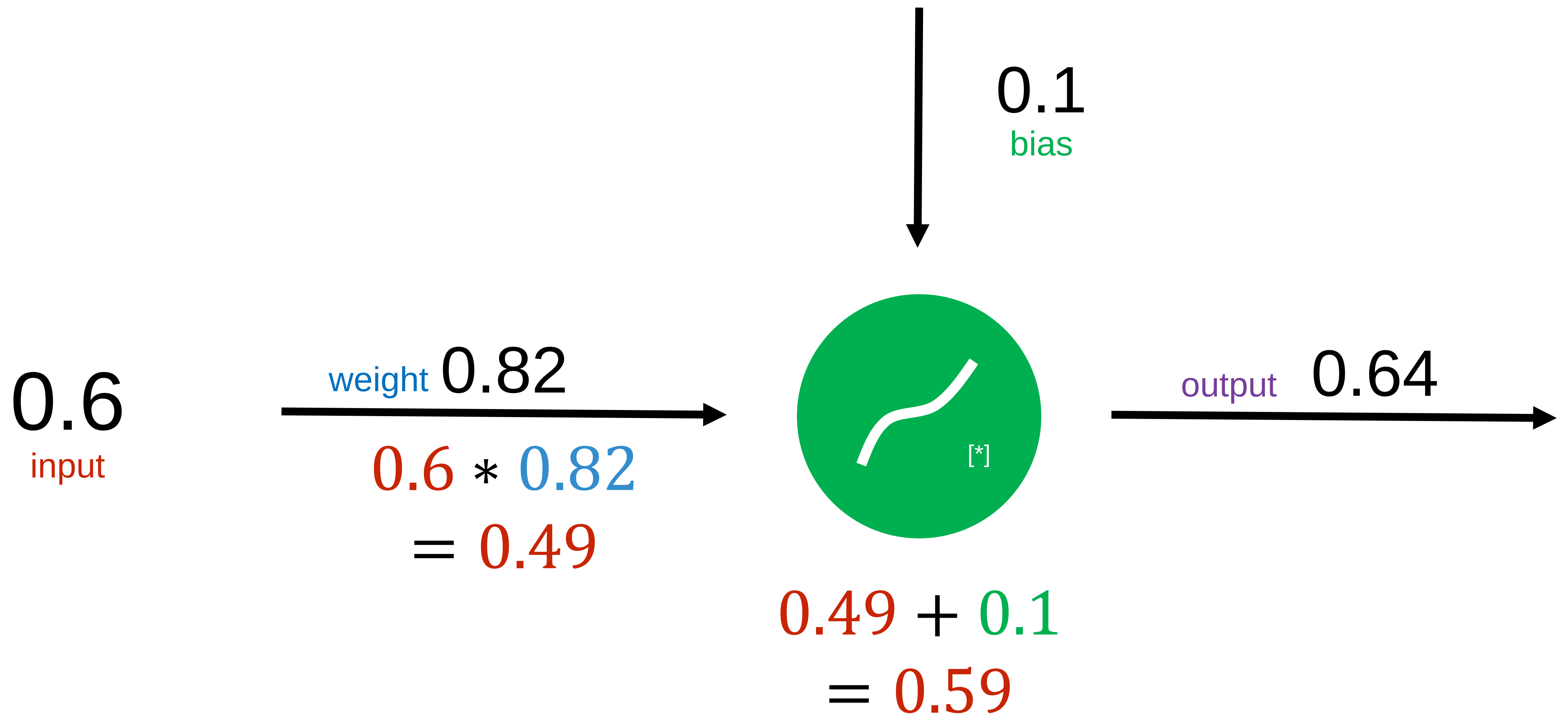
Prediction Process



Prediction Process

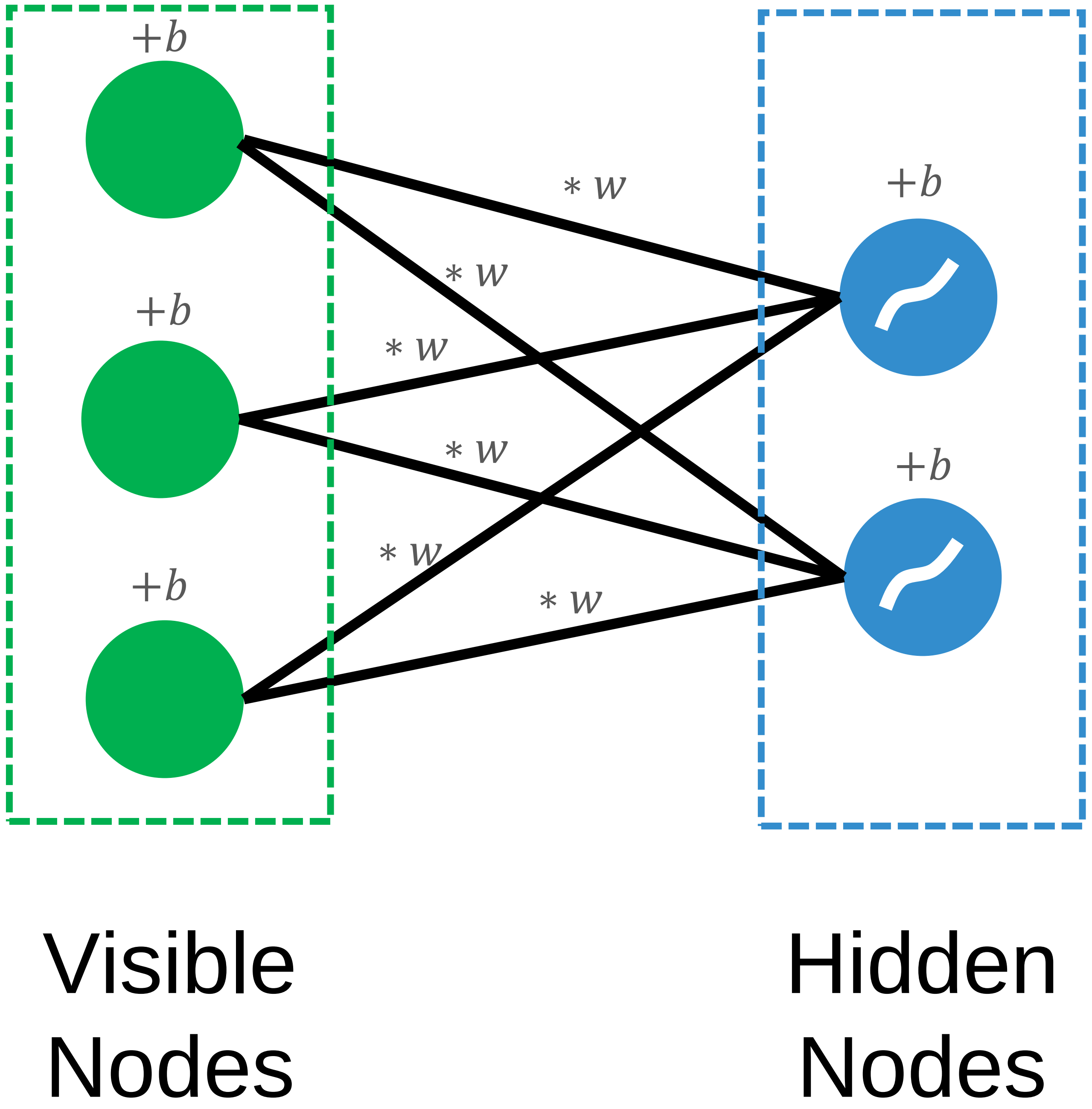


Recap: Perceptron



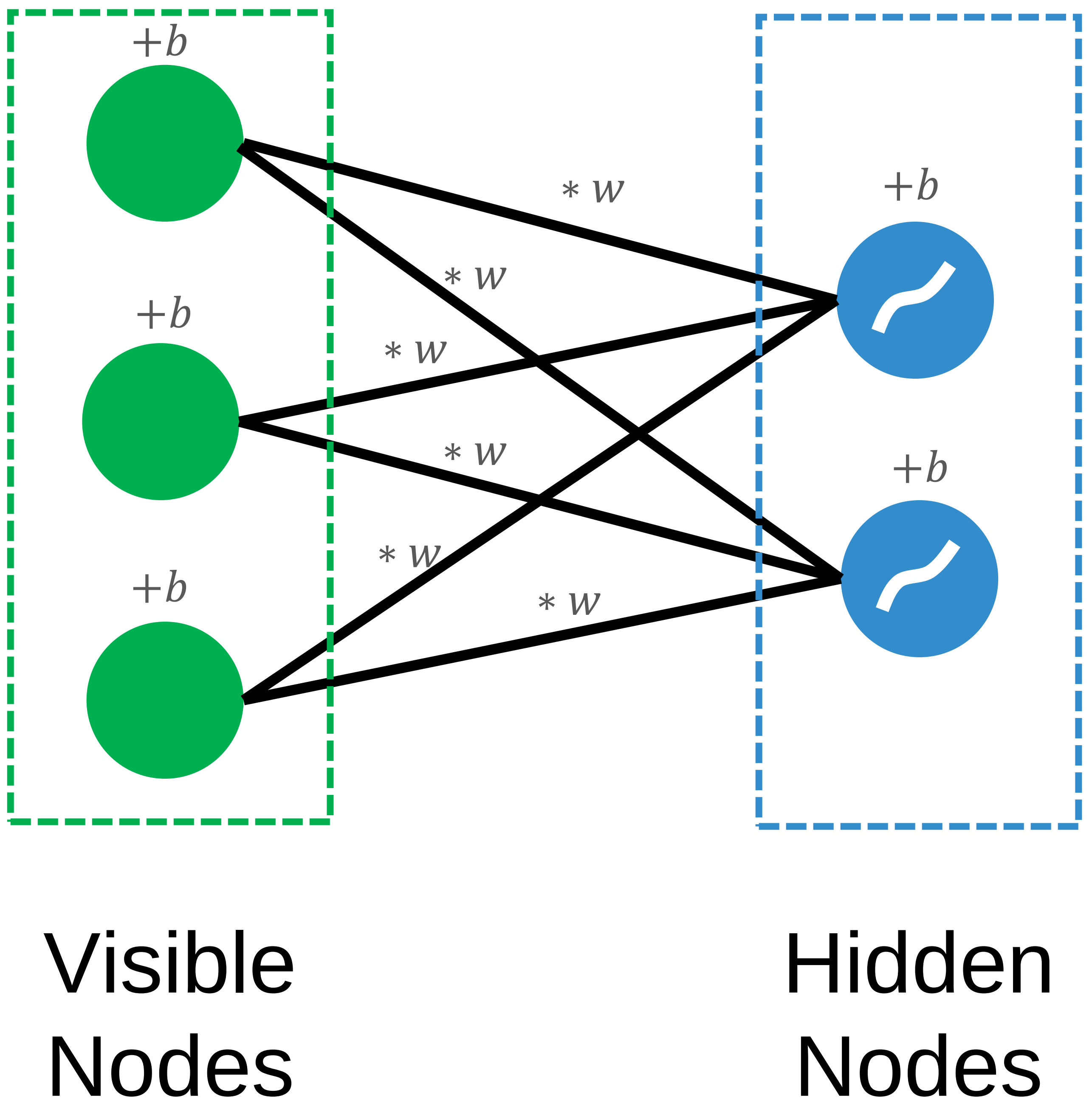
[*] Activation function. In this case:
Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$
In [8] this is just a „binary“ threshold.

Restricted Boltzmann Machine



[8] D. Ackley, G. Hinton, T. Sejnowski "A learning algorithm for boltzmann machines." In: Cognitive Science (1985) Volume 9

Restricted Boltzmann Machine



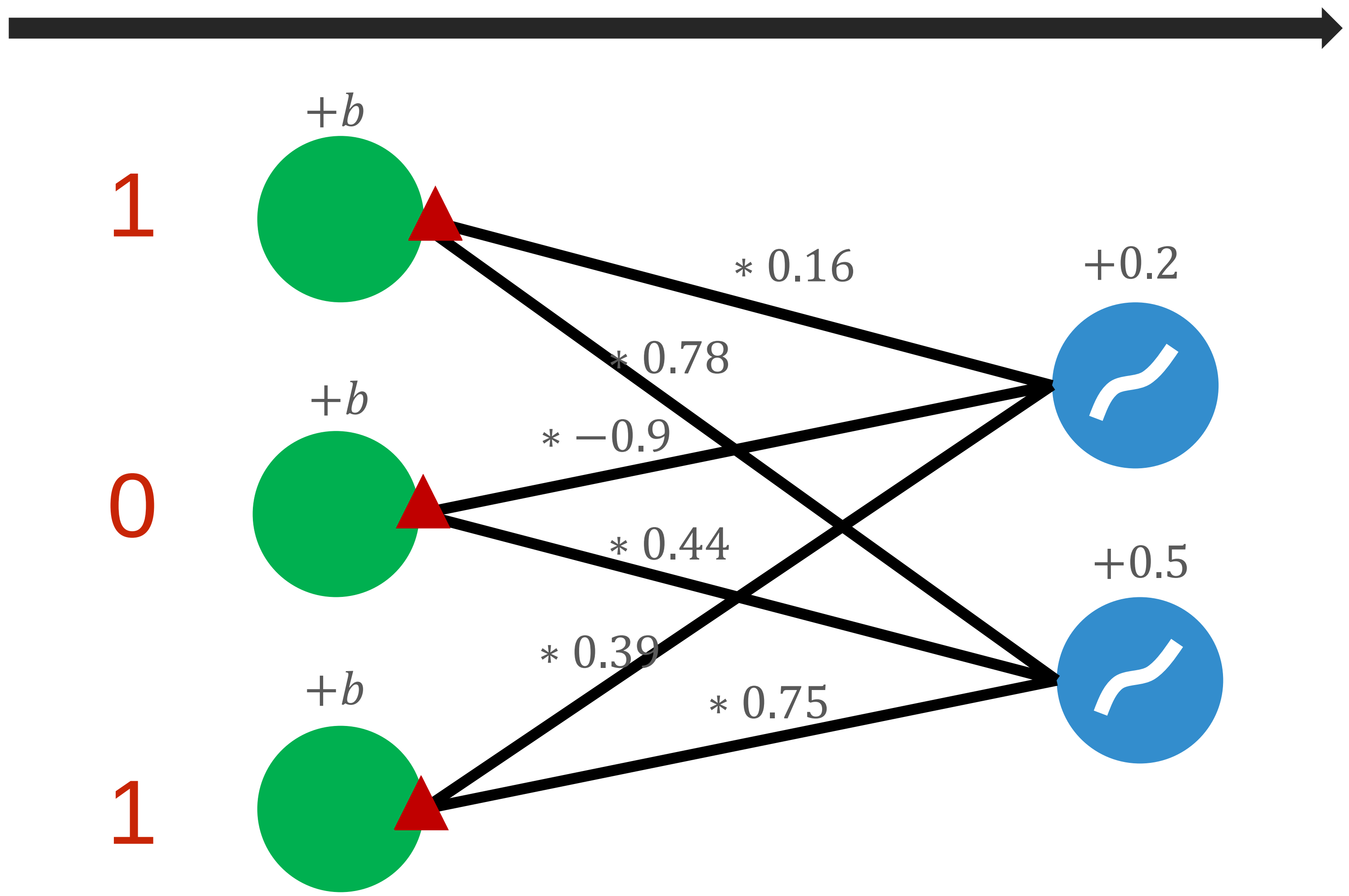
- Shallow two layer network
- No intra layer connections (Restriced)
- Fully connected
- Can be trained unsupervised
- ‘Contrastive Divergence‘ [9] / ‘Gibbs Sampling‘ is used to train the network

[9] G. Hinton, S. Osindero, Y. Teh “A Fast Learning Algorithm for Deep Belief Nets.” In: Neural Computation (2006) Volume 18

Restricted Boltzmann Machine - Training

input

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

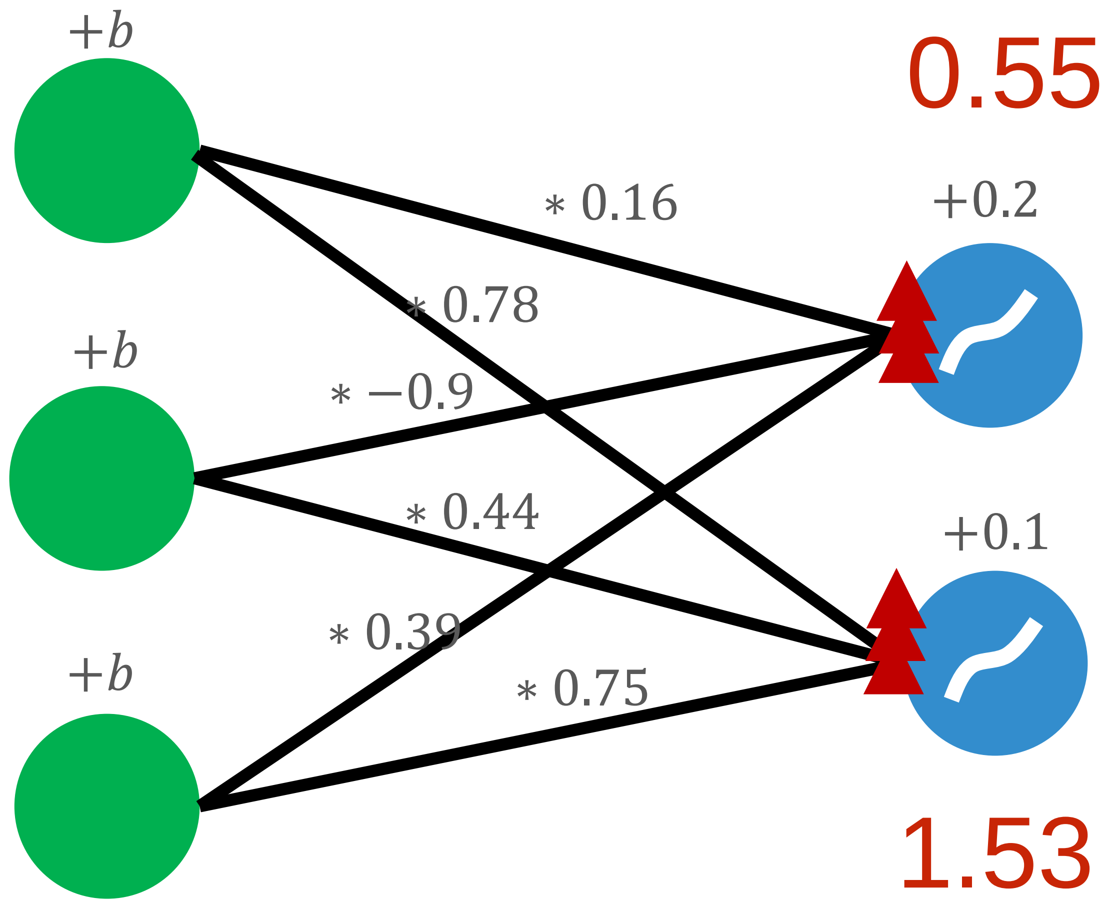


Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

input

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



input weights

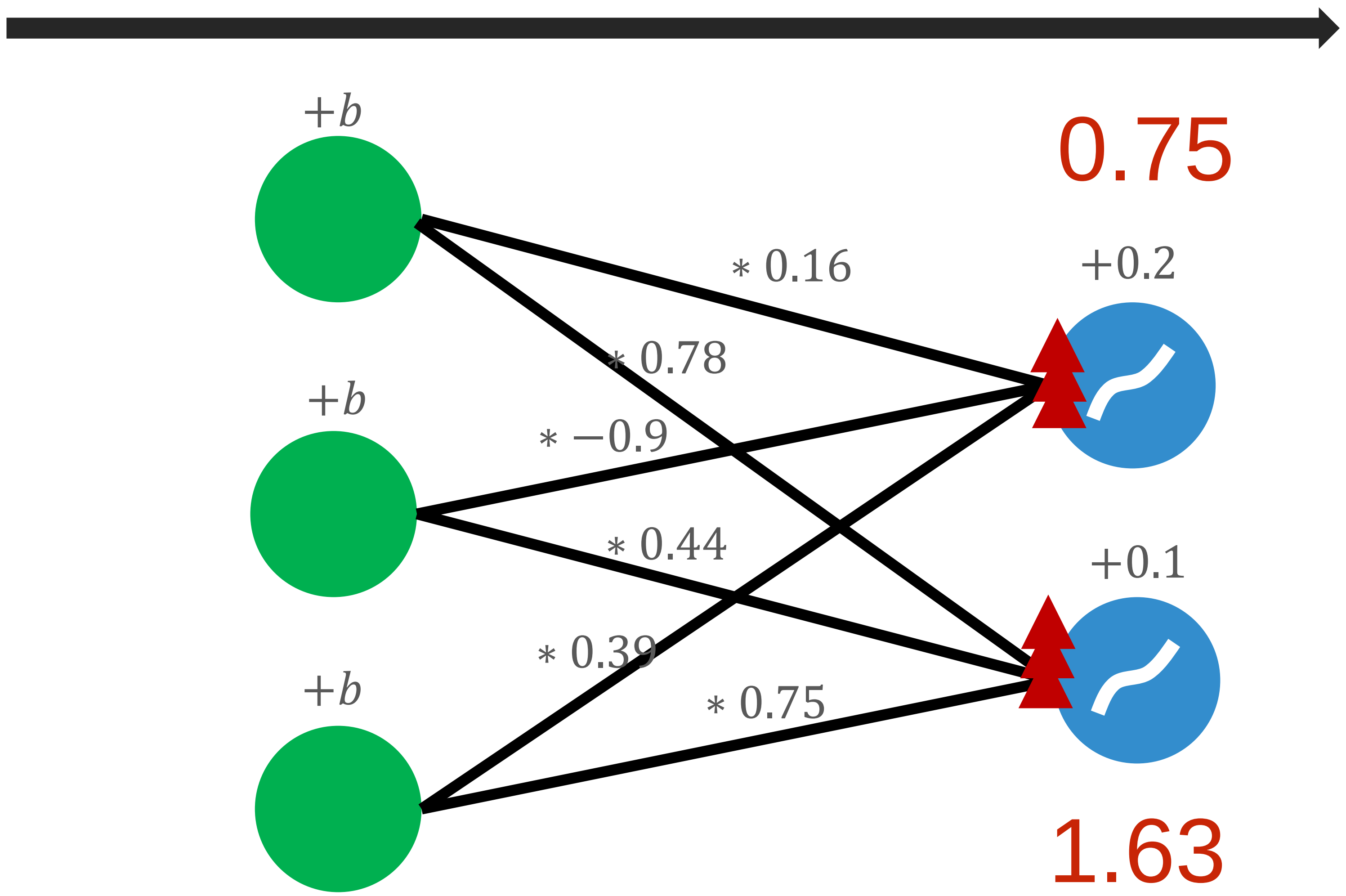
$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0.16 & 0.78 \\ -0.9 & 0.44 \\ 0.39 & 0.75 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

input

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



input weights

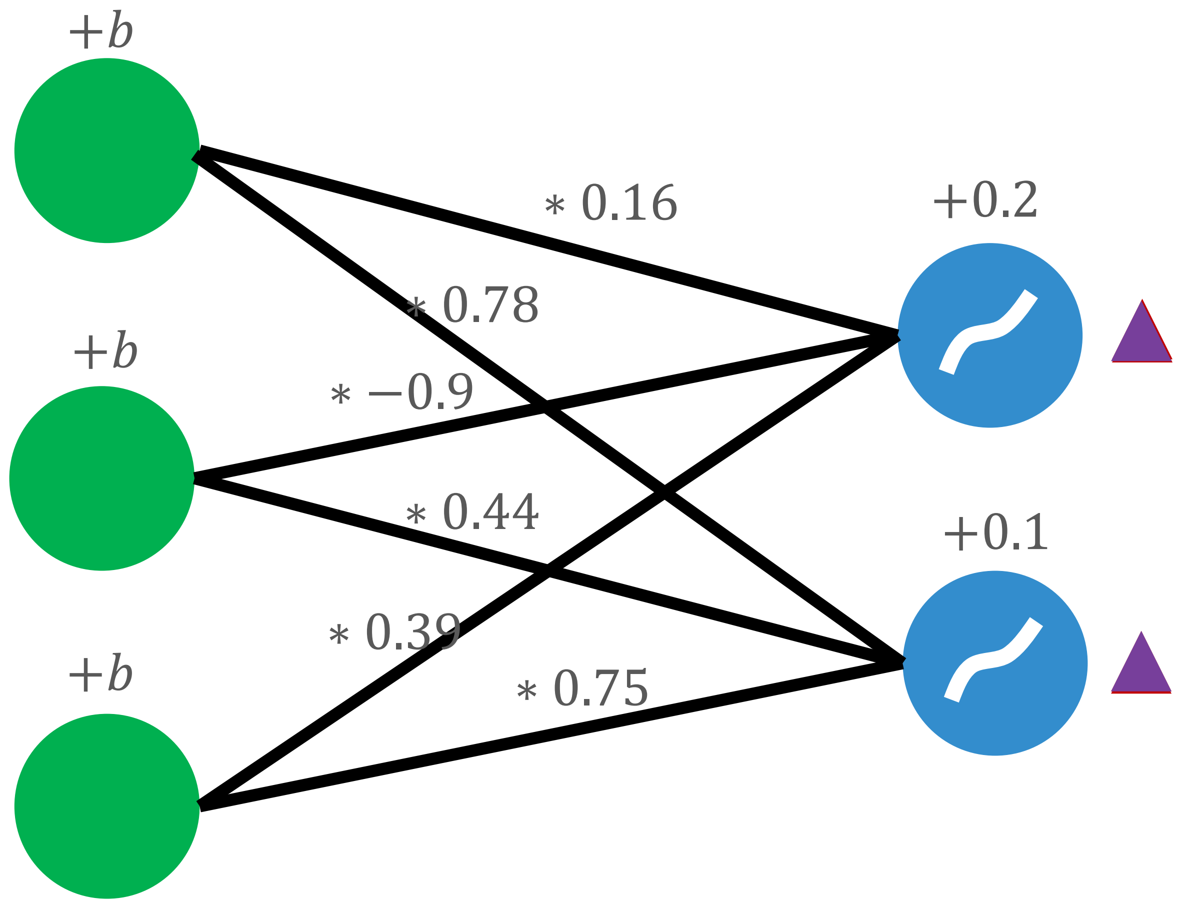
$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0.16 & 0.78 \\ -0.9 & 0.44 \\ 0.39 & 0.75 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

input

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



0.68

0.84

$$h = \begin{bmatrix} 0.68 \\ 0.84 \end{bmatrix}$$

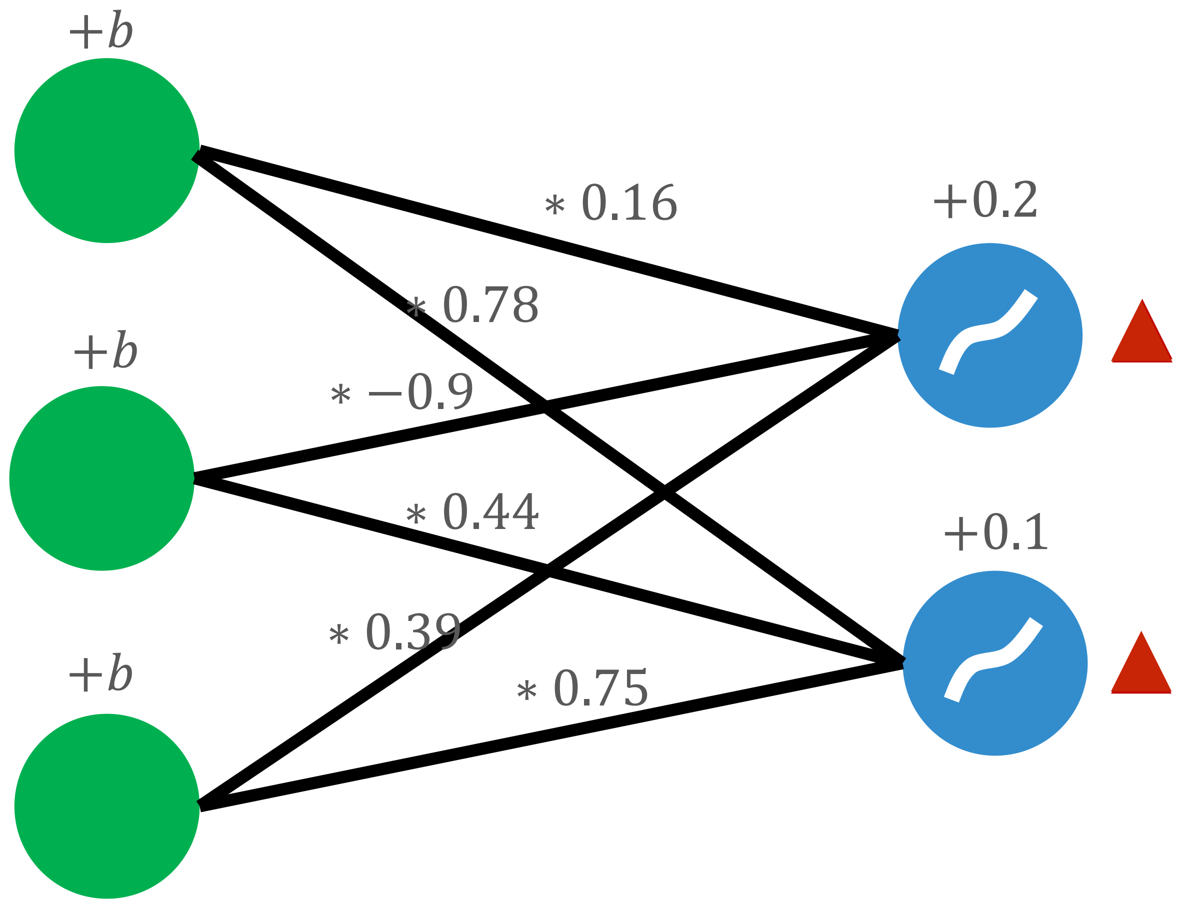
input weights

$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0.16 & 0.78 \\ -0.9 & 0.44 \\ 0.39 & 0.75 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



input

1

1

$$h = \begin{bmatrix} 0.68 \\ 0.84 \end{bmatrix}$$

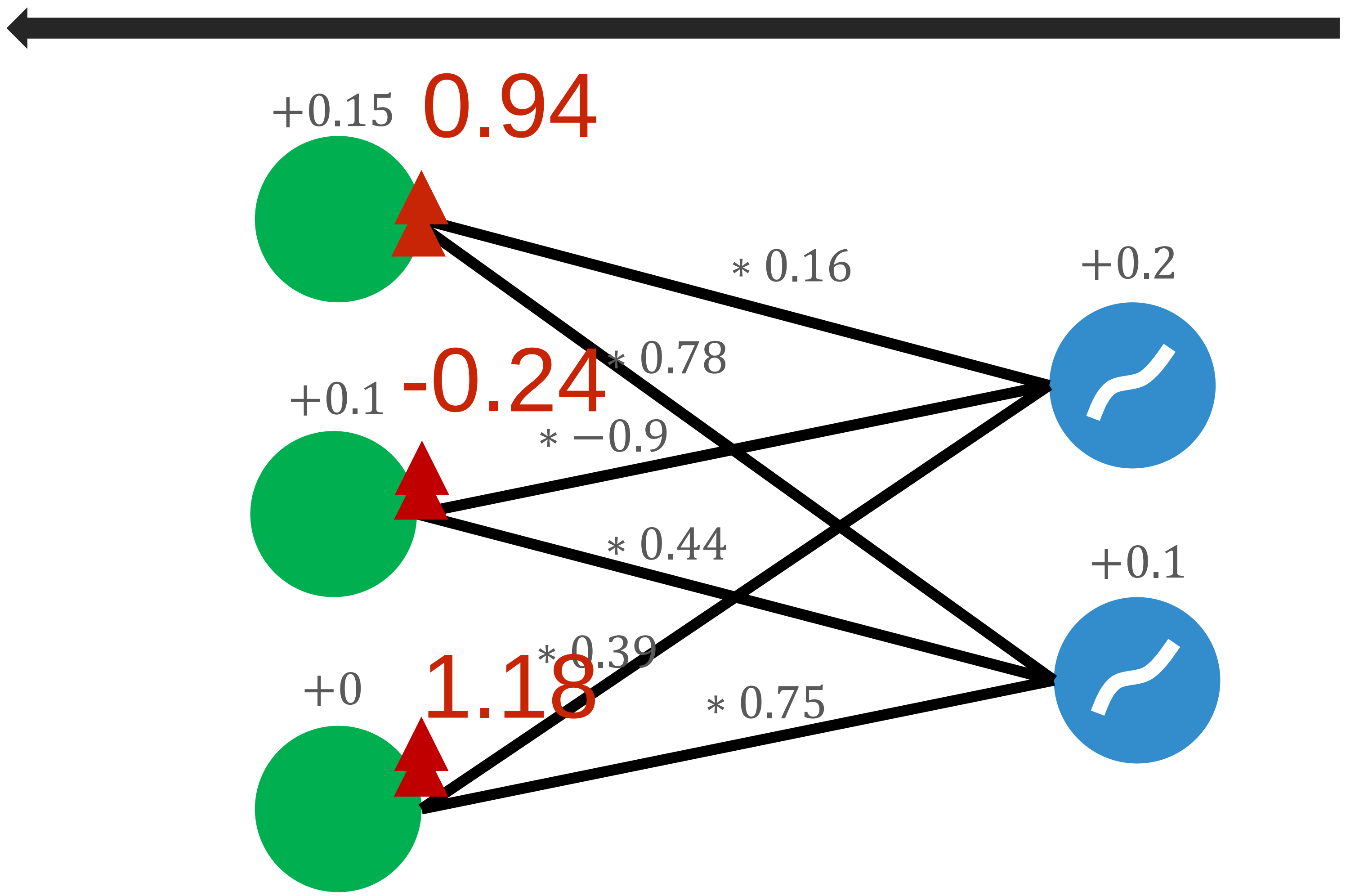
weights input

$$\begin{bmatrix} 0.16 & -0.9 & 0.39 \\ 0.78 & 0.44 & 0.79 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



$$h = \begin{bmatrix} 0.68 \\ 0.84 \end{bmatrix}$$

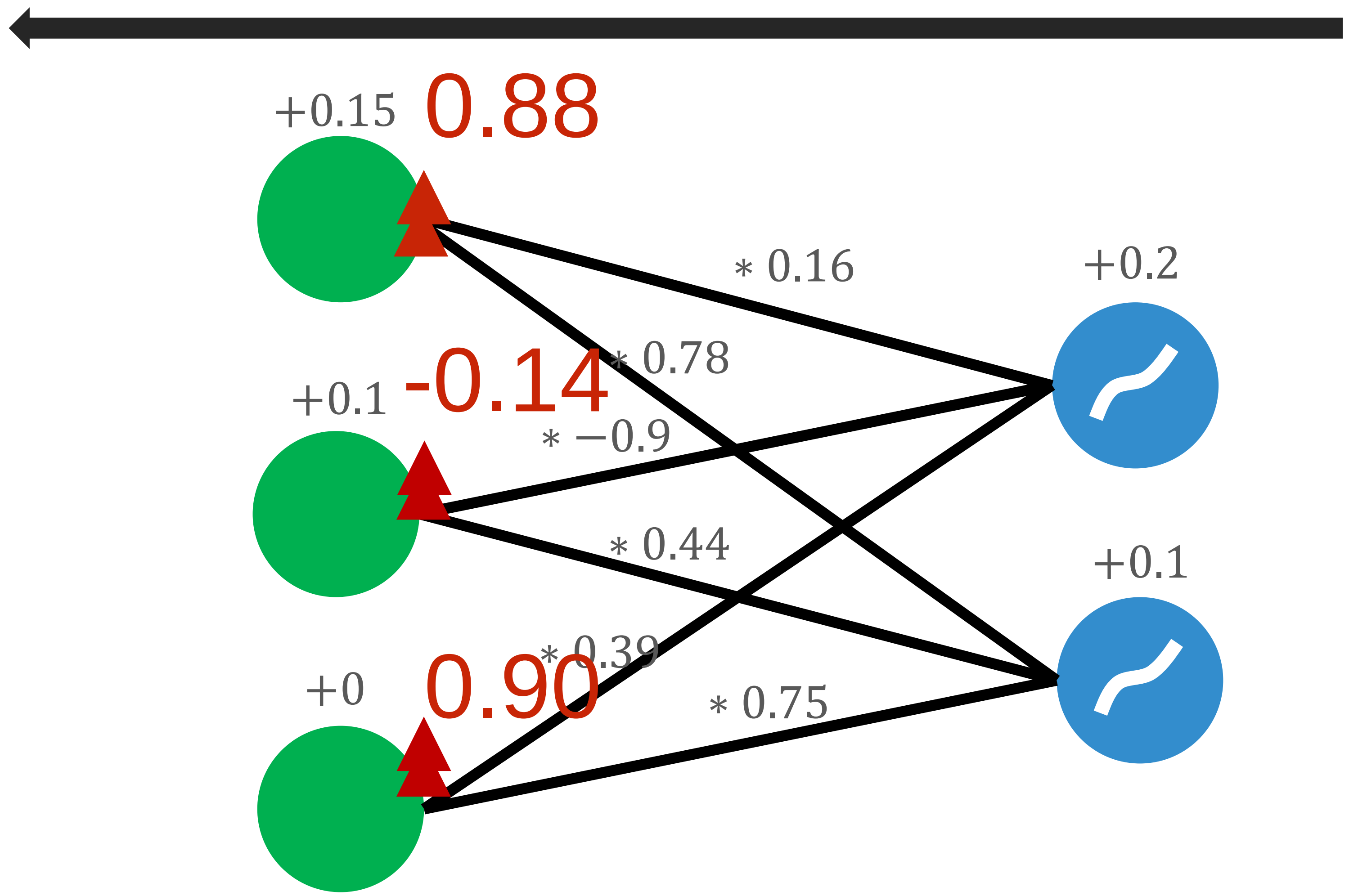
weights input

$$\begin{bmatrix} 0.16 & -0.9 & 0.39 \\ 0.78 & 0.44 & 0.79 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$



$$h = \begin{bmatrix} 0.68 \\ 0.84 \end{bmatrix}$$

weights input

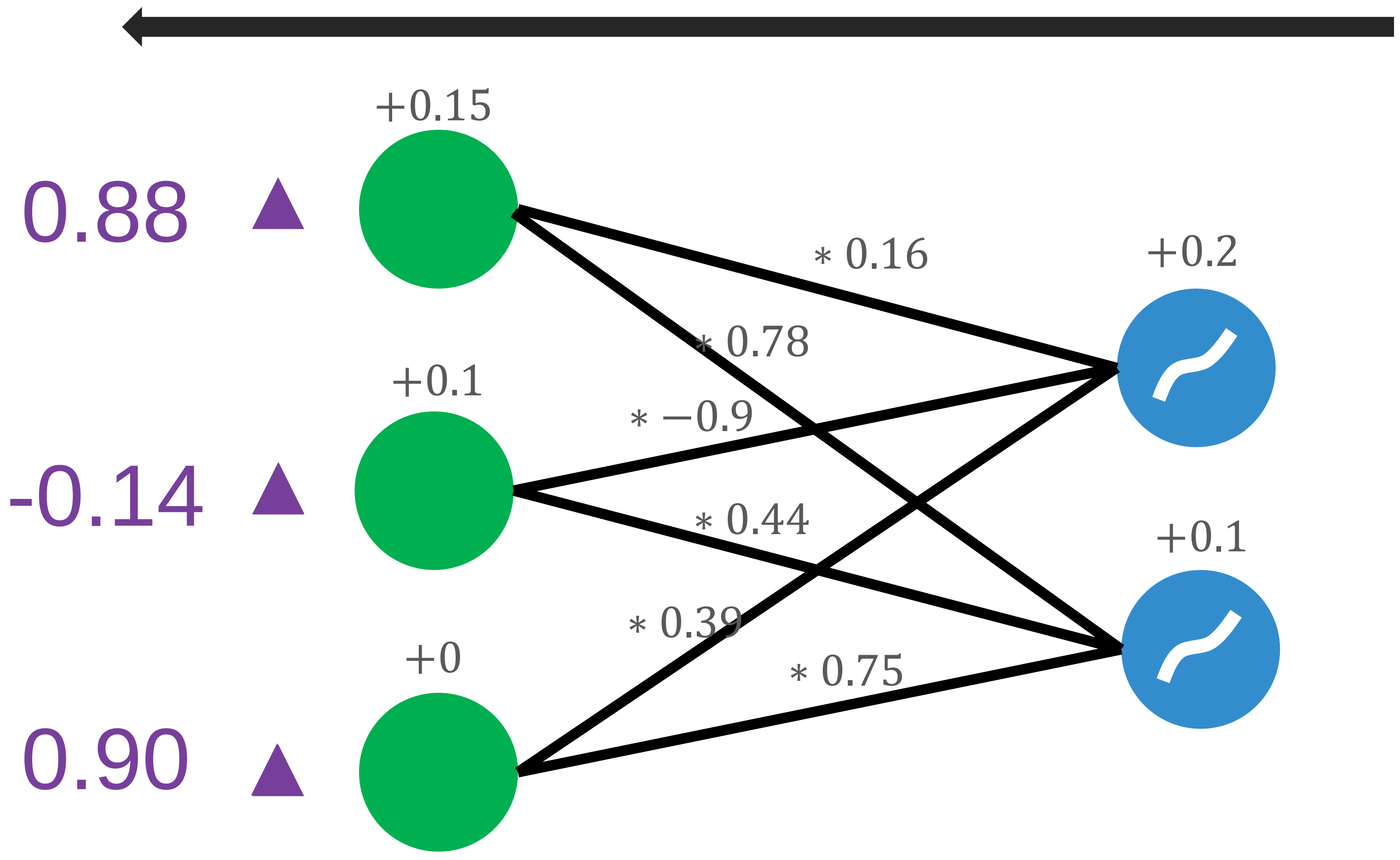
$$\begin{bmatrix} 0.16 & -0.9 & 0.39 \\ 0.78 & 0.44 & 0.79 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$v' = \begin{bmatrix} 0.88 \\ -0.14 \\ 0.90 \end{bmatrix}$$



$$h = \begin{bmatrix} 0.68 \\ 0.84 \end{bmatrix}$$

weights input

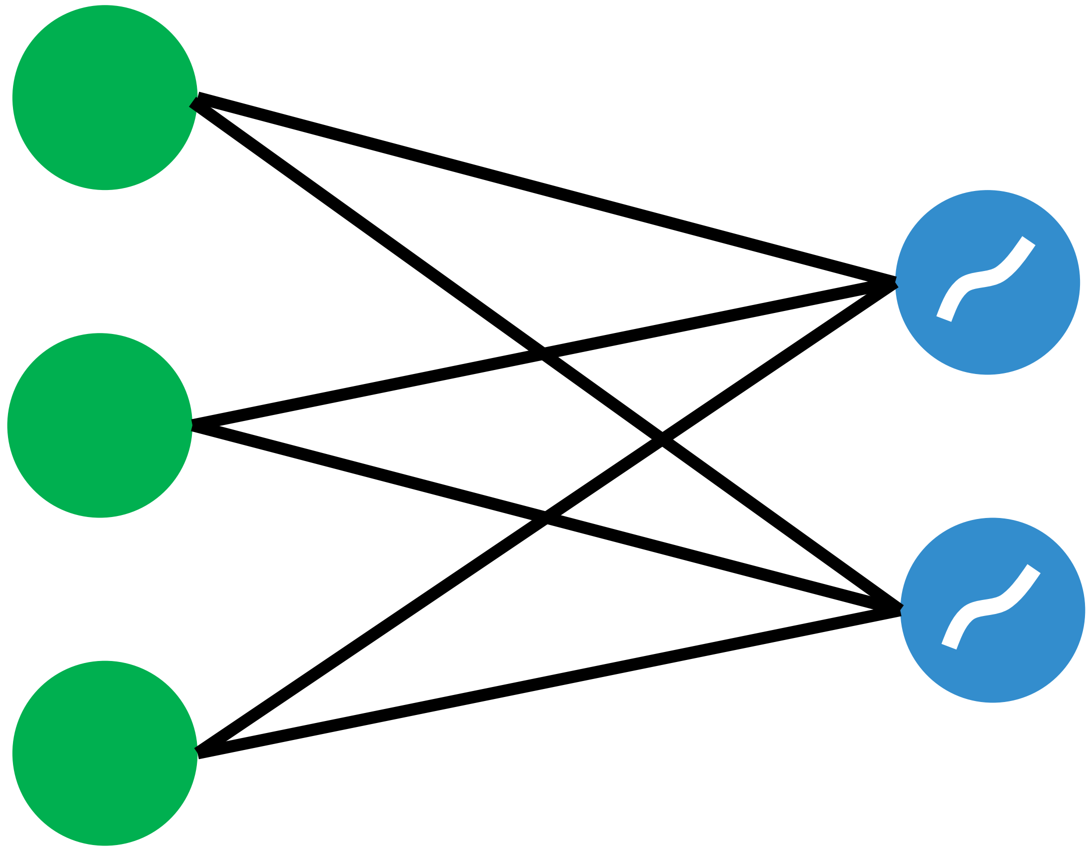
$$\begin{bmatrix} 0.16 & -0.9 & 0.39 \\ 0.78 & 0.44 & 0.79 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$v' = \begin{bmatrix} 0.88 \\ -0.14 \\ 0.90 \end{bmatrix}$$



Calculate error and adjust weights so that $v \approx v'$

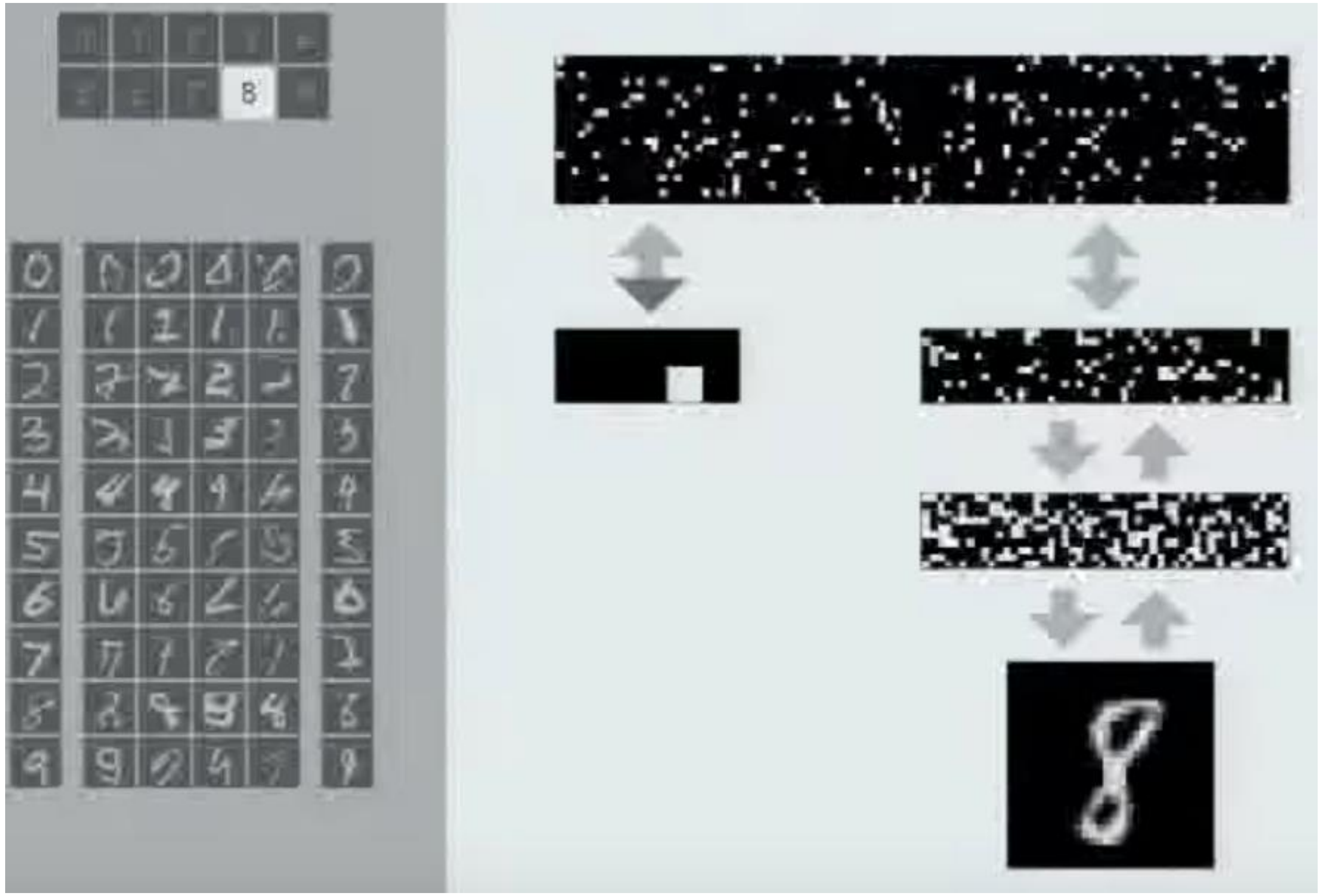
Sigmoid function: $S(t) = \frac{1}{1+e^{-t}}$

Restricted Boltzmann Machine - Training

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

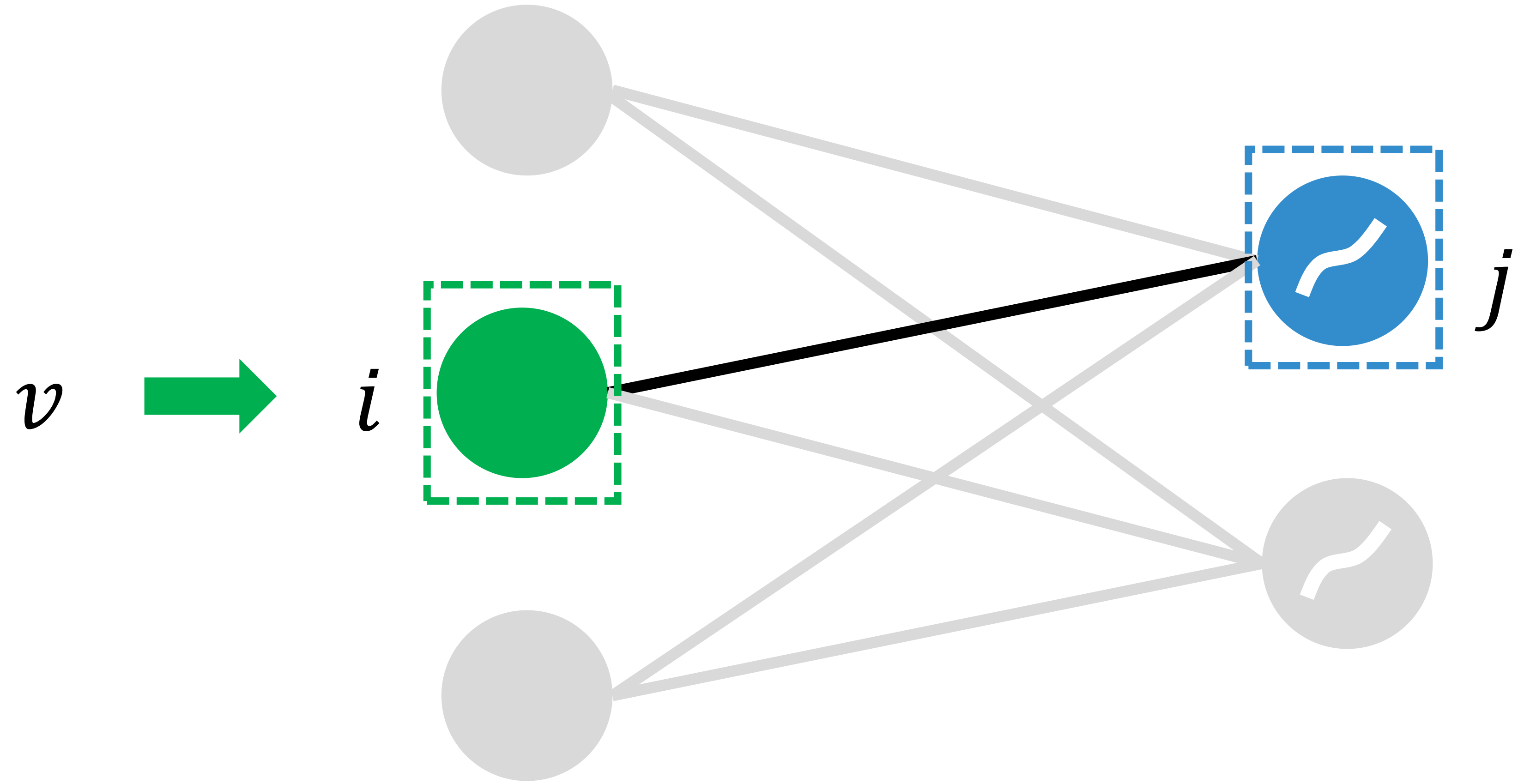
$$v' = \begin{bmatrix} 0.88 \\ -0.14 \\ 0.90 \end{bmatrix}$$

Calculate error and adjust weights so that $v \approx v'$



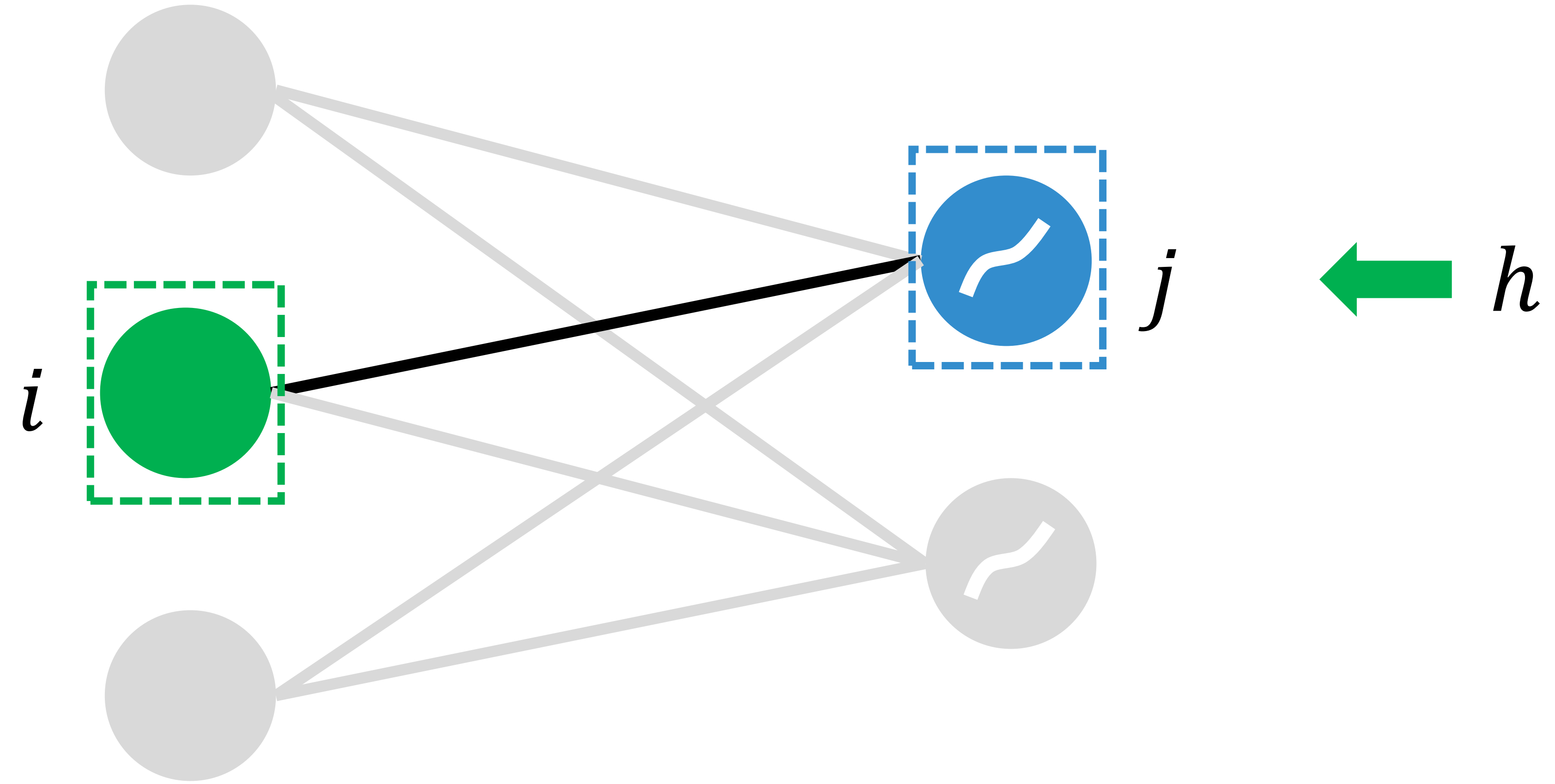
[10] G. Hinton. "The Next Generation of Neural Networks." Google TechTalks (2007) https://www.youtube.com/watch?v=AyzOUbkUf3M Last accessed: 12.01.2017

Restricted Boltzmann Machine - Training



How often are they on together with input v?

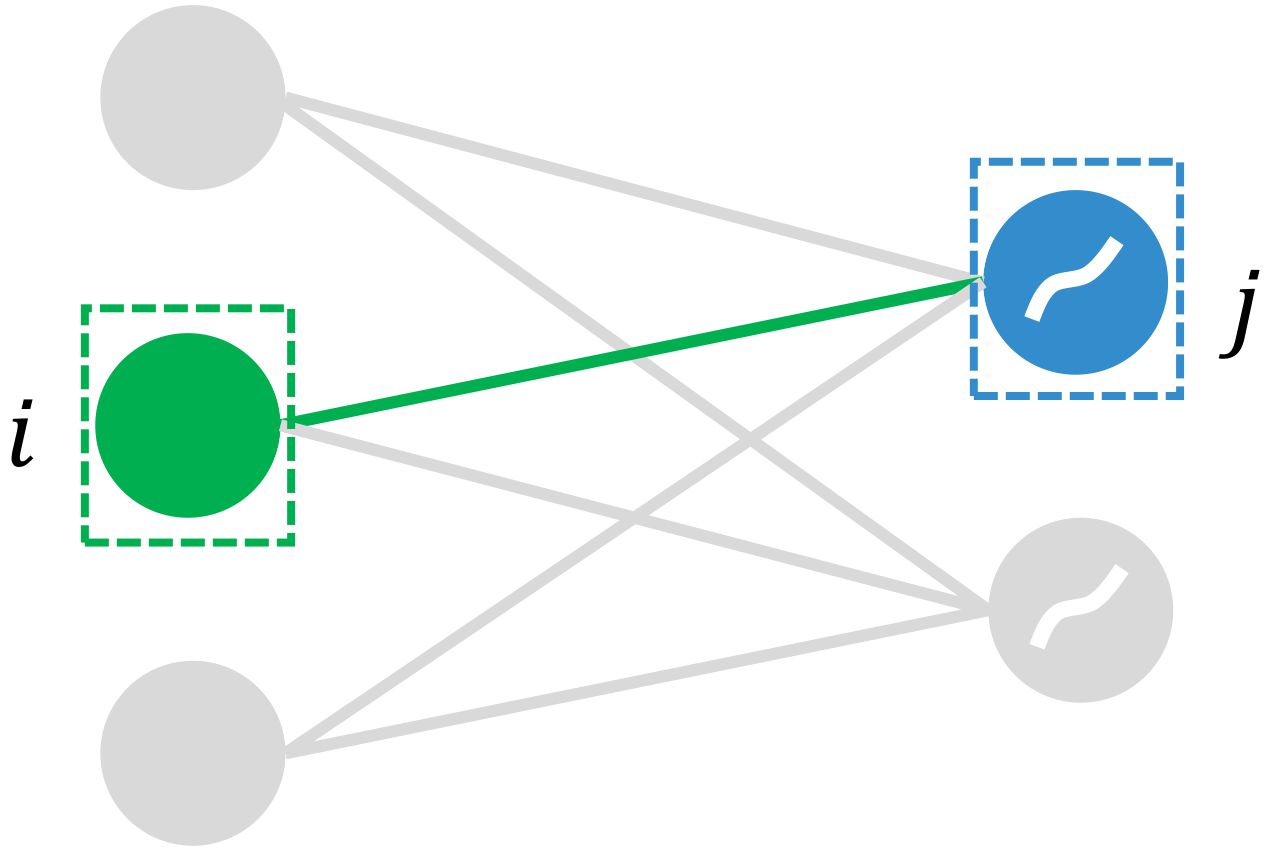
Restricted Boltzmann Machine - Training



How often are they on together when the model is dreaming?

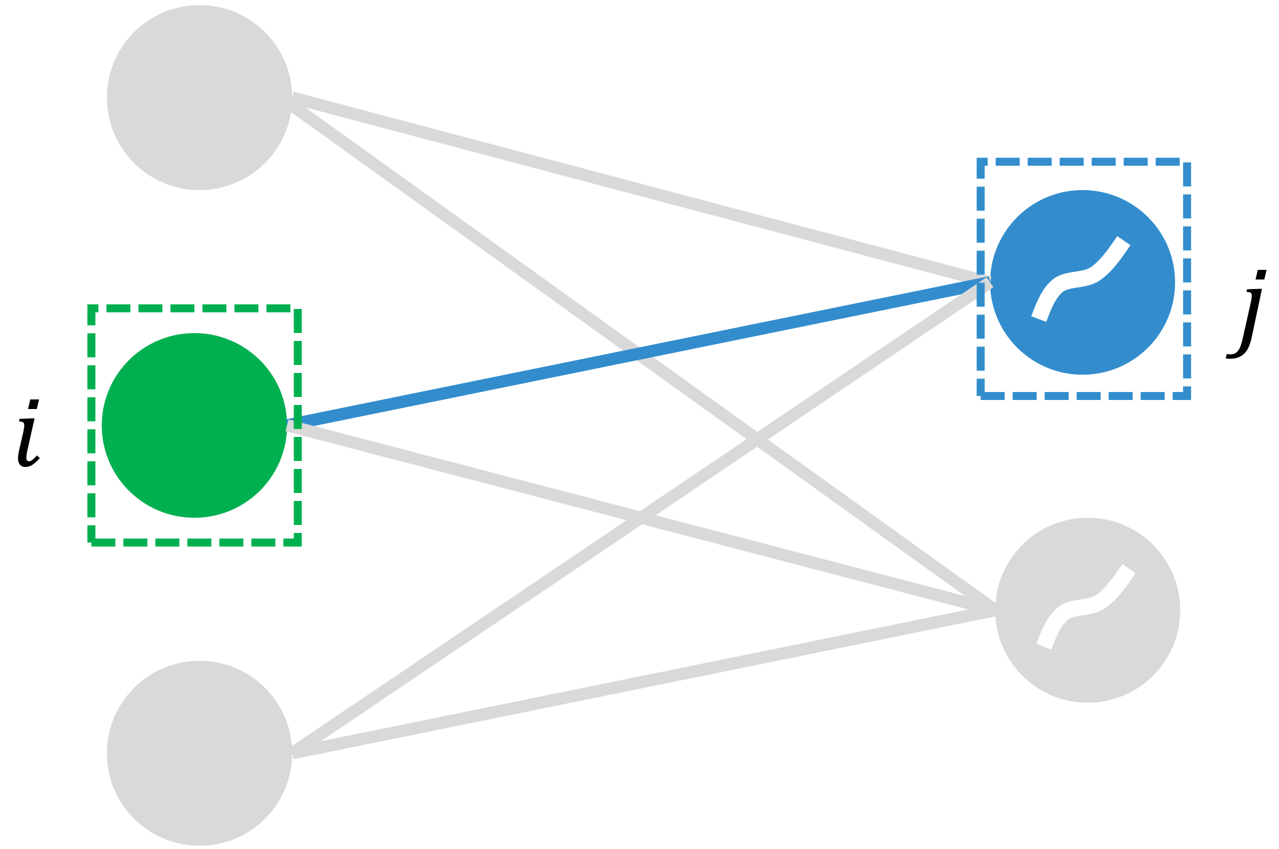
Restricted Boltzmann Machine - Training

Raise weights when i and j are on together



Restricted Boltzmann Machine - Training

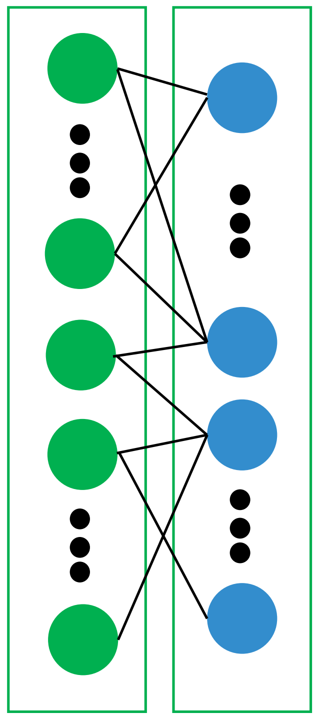
Raise weights when i and j are on together with input



Lower weights when i and j are on together when dreaming

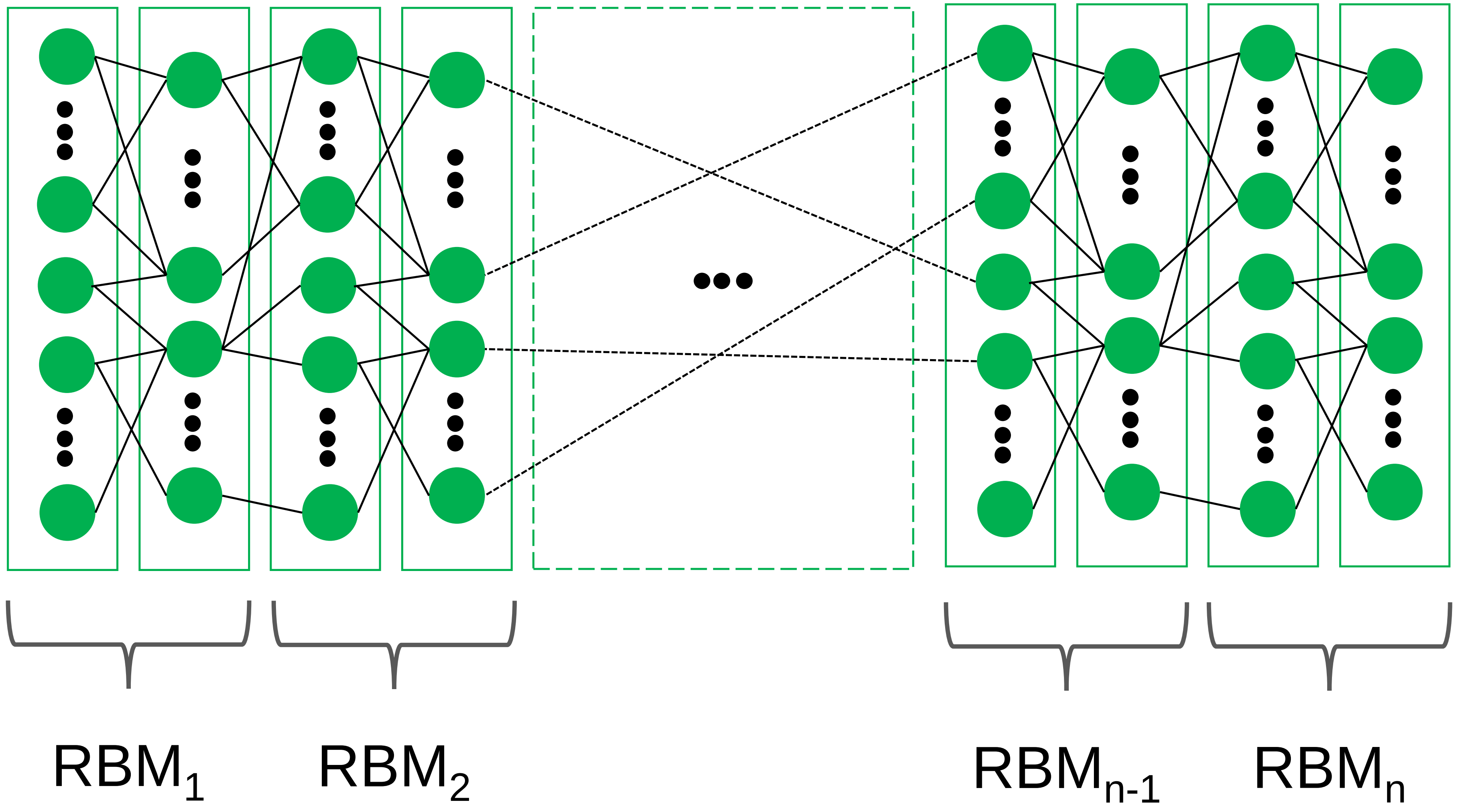
This will force towards „dreaming“ the actual data

Deep Belief Network



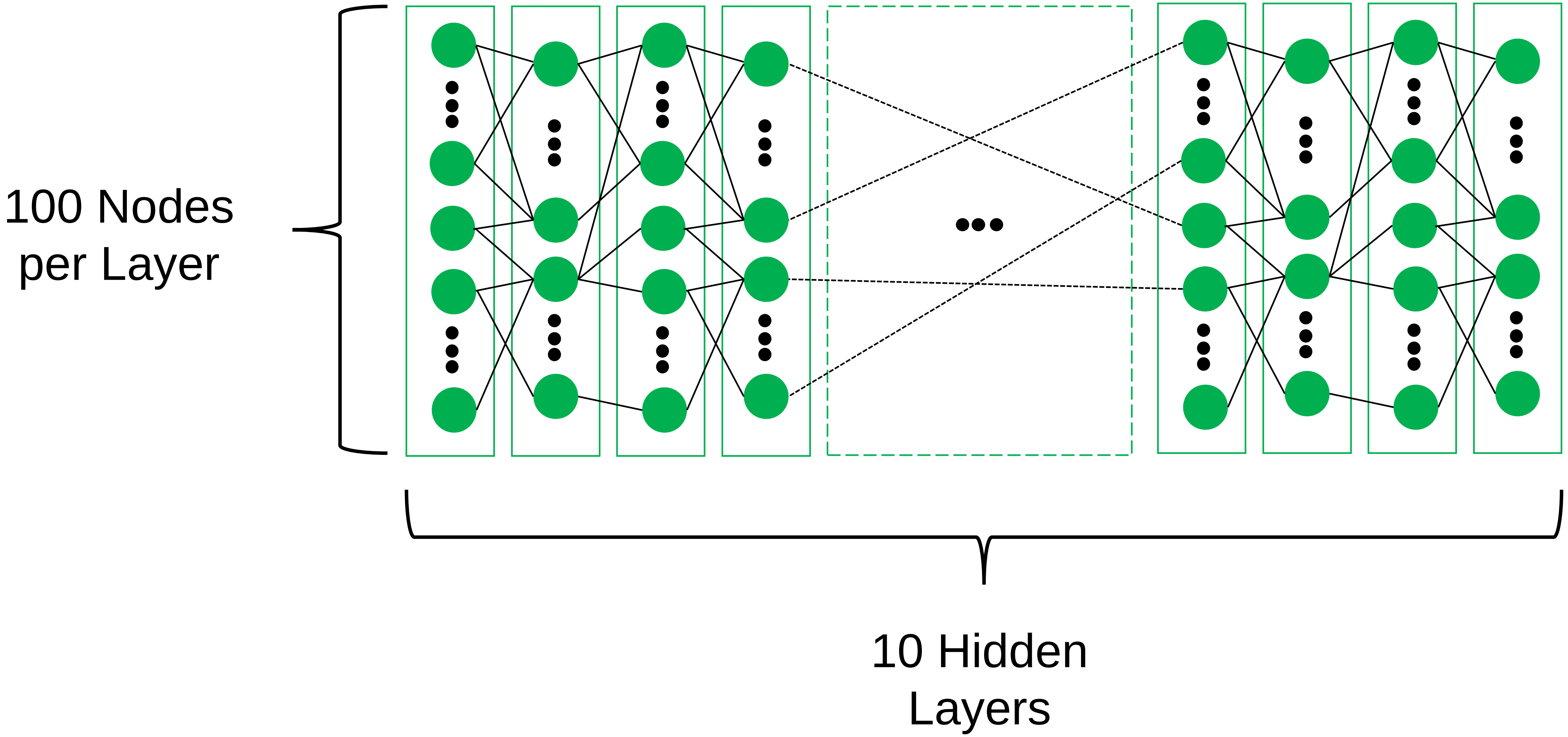
[9] G. Hinton, S. Osindero, Y. Teh "A Fast Learning Algorithm for Deep Belief Nets." In: Neural Computation (2006) Volume 18

Deep Belief Network



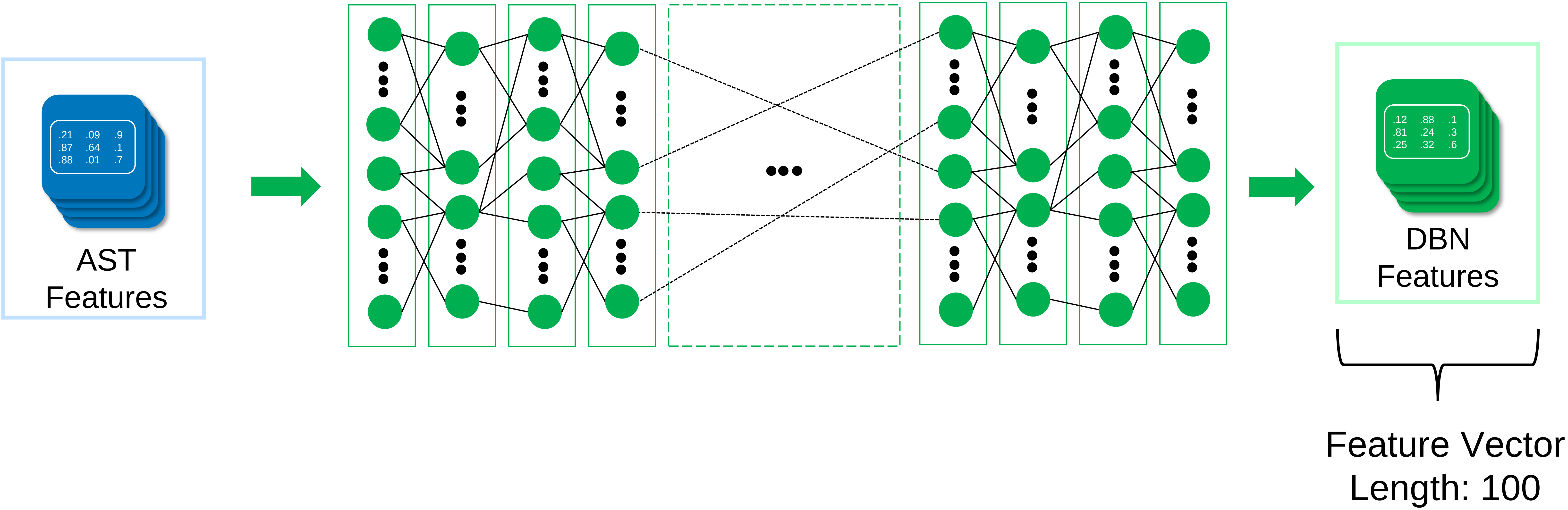
[9] G. Hinton, S. Osindero, Y. Teh "A Fast Learning Algorithm for Deep Belief Nets." In: Neural Computation (2006) Volume 18

Deep Belief Network

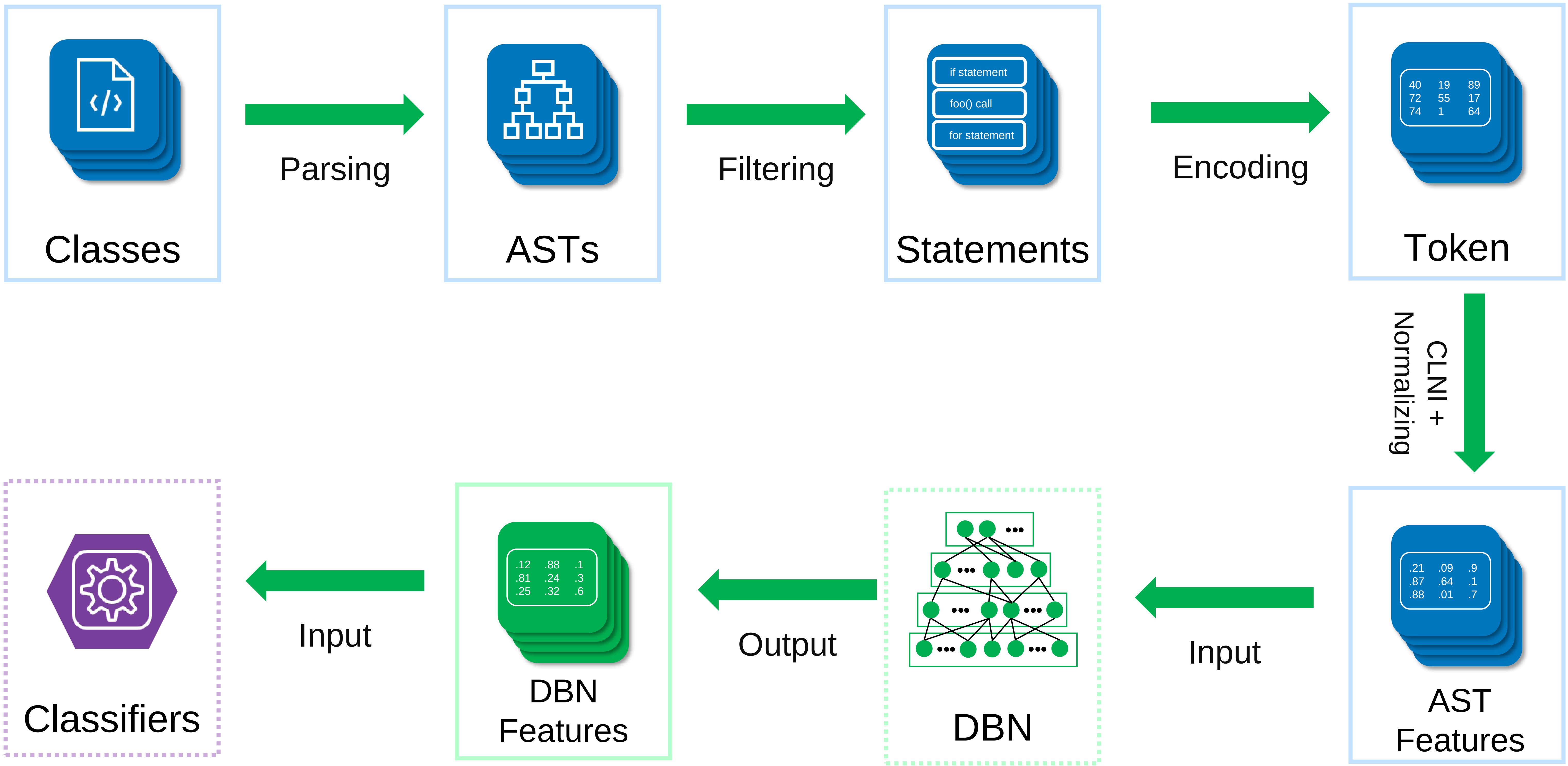


[9] G. Hinton, S. Osindero, Y. Teh "A Fast Learning Algorithm for Deep Belief Nets." In: Neural Computation (2006) Volume 18

Deep Belief Network



Prediction Process



Agenda

Defect Prediction

Technical Background

Discussion of Results

Demo

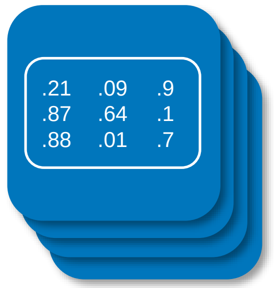
Conclusion

Results – Within Project Defect Prediction

- Success is measured with F1-Score
- Baselines:

Code metrices

20 traditional features:
- Lines of code
- Operand count
- McCabe complexity



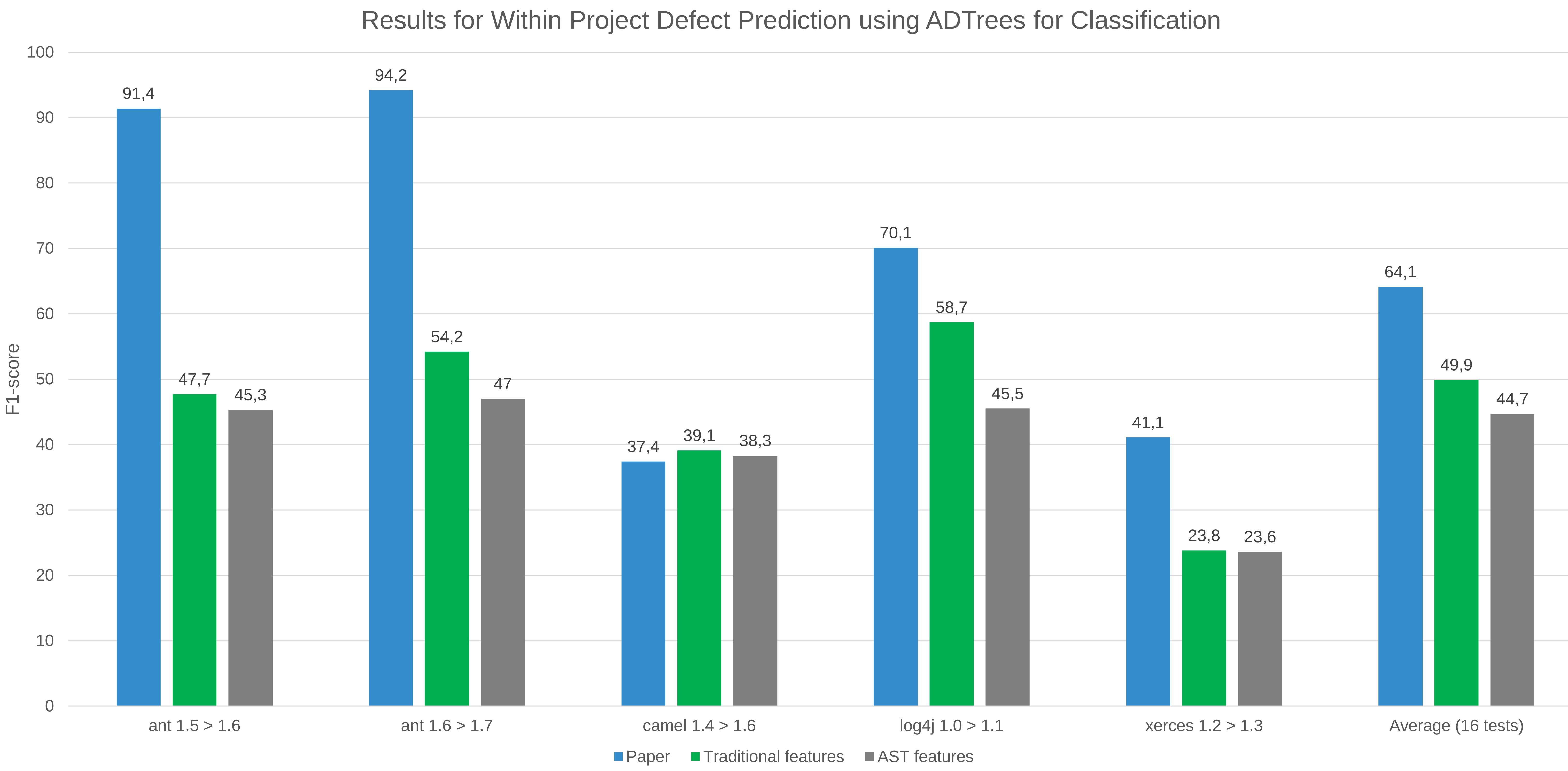
AST Features

Filtered AST features

Results – Within Project Defect Prediction

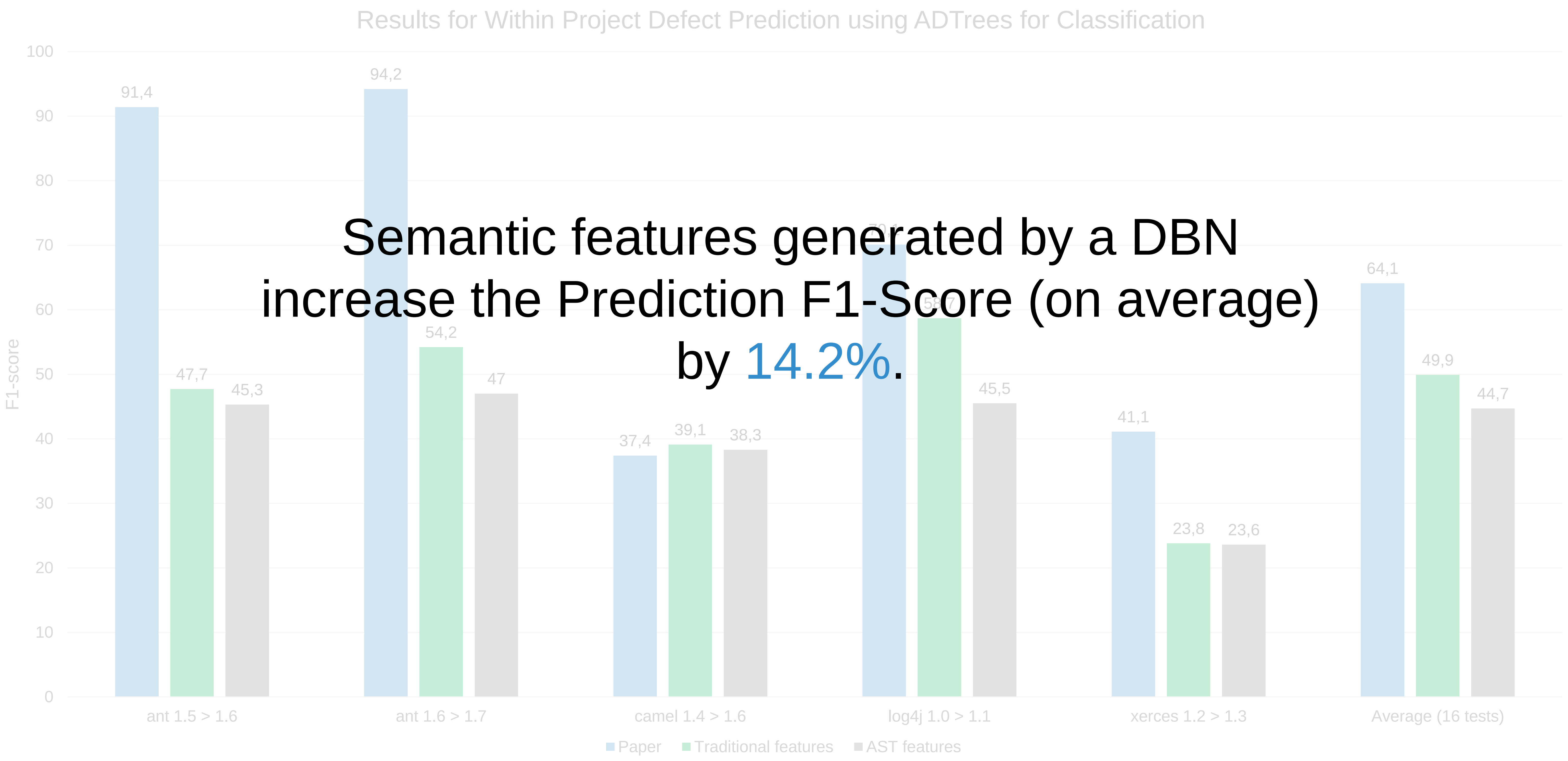
- Success is measured with F1-Score
- Baselines:
  - 20 traditional features
  - AST features
- Classifiers:
  - ADTree
  - Naïve Bayes
  - Logistic Regression

Results – Within Project Defect Prediction



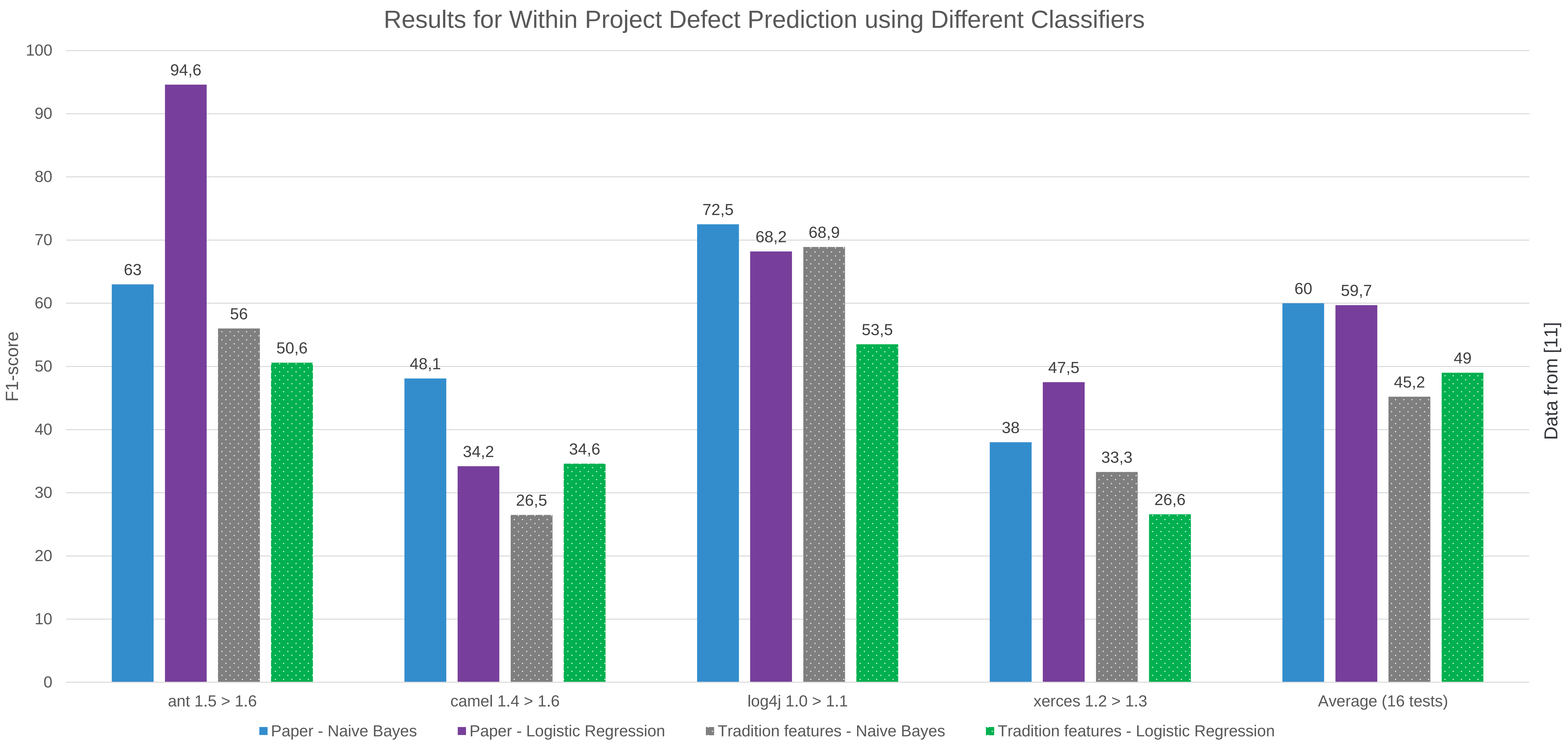
[11] S. Wang, T. Liu, L. Tan. “Automatically learning semantic features for defect prediction.” In: Proceedings - International Conference on Software Engineering (2016)

Results – Within Project Defect Prediction

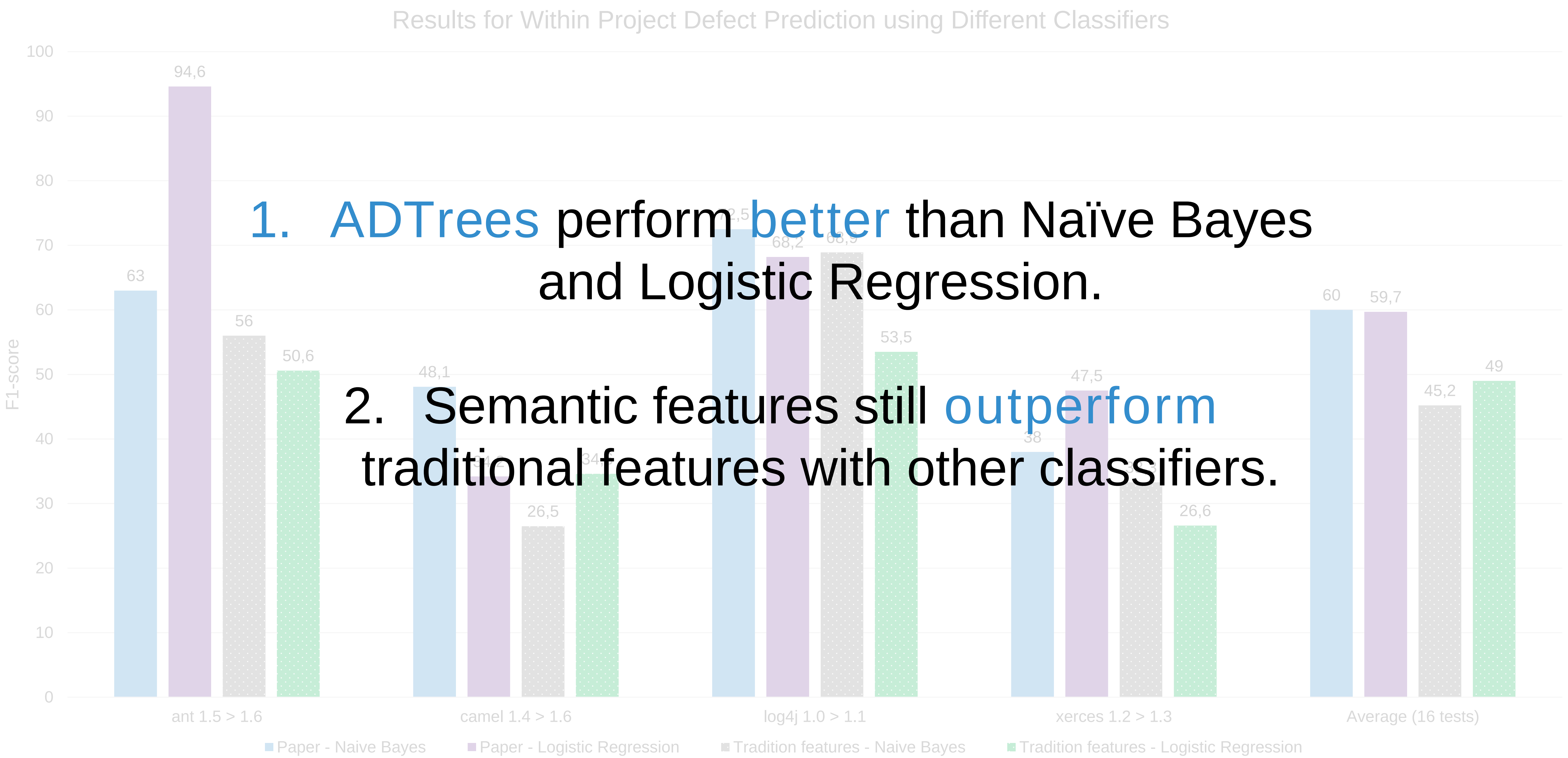


Semantic features generated by a DBN increase the Prediction F1-Score (on average) by 14.2%.

Results – Within Project Defect Prediction



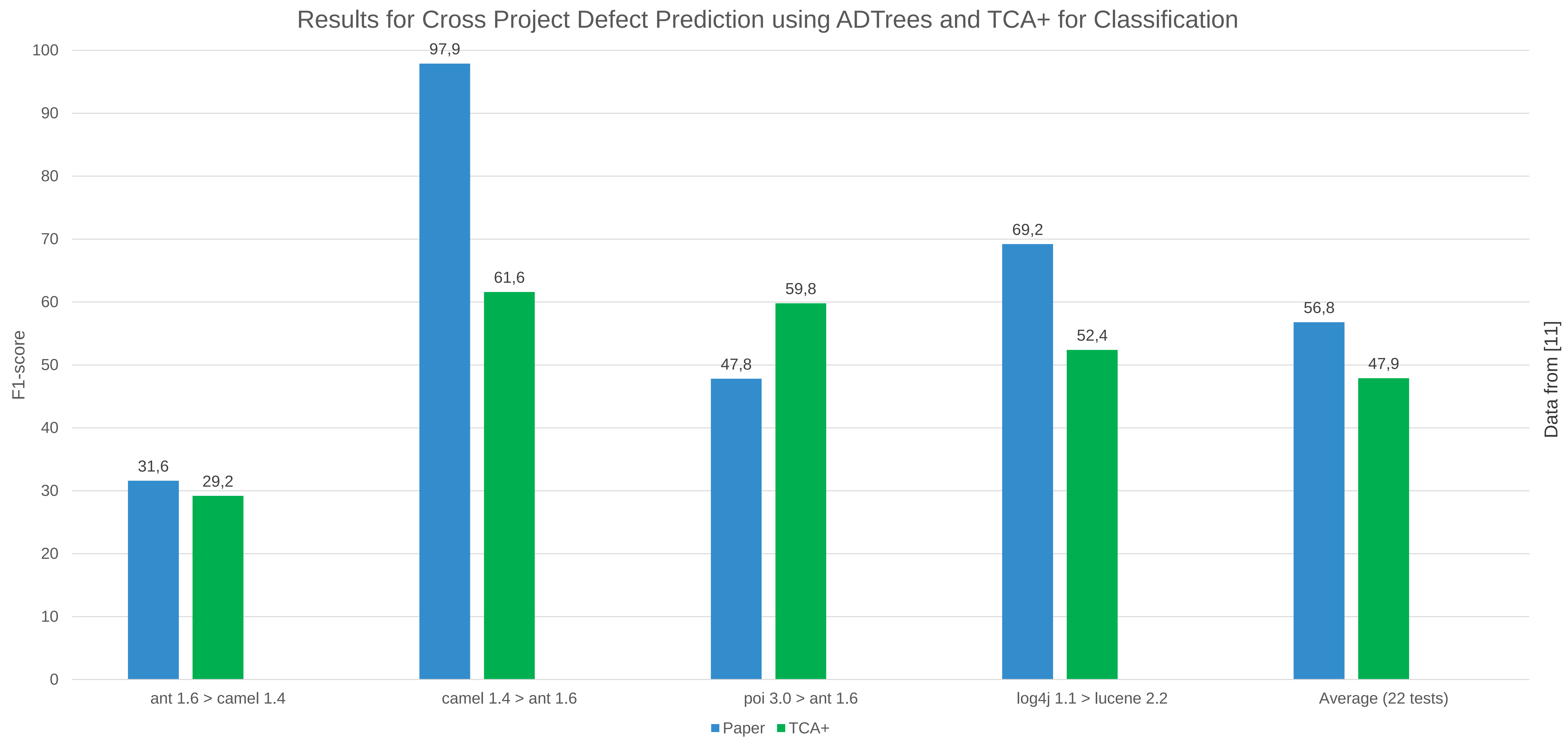
Results – Within Project Defect Prediction



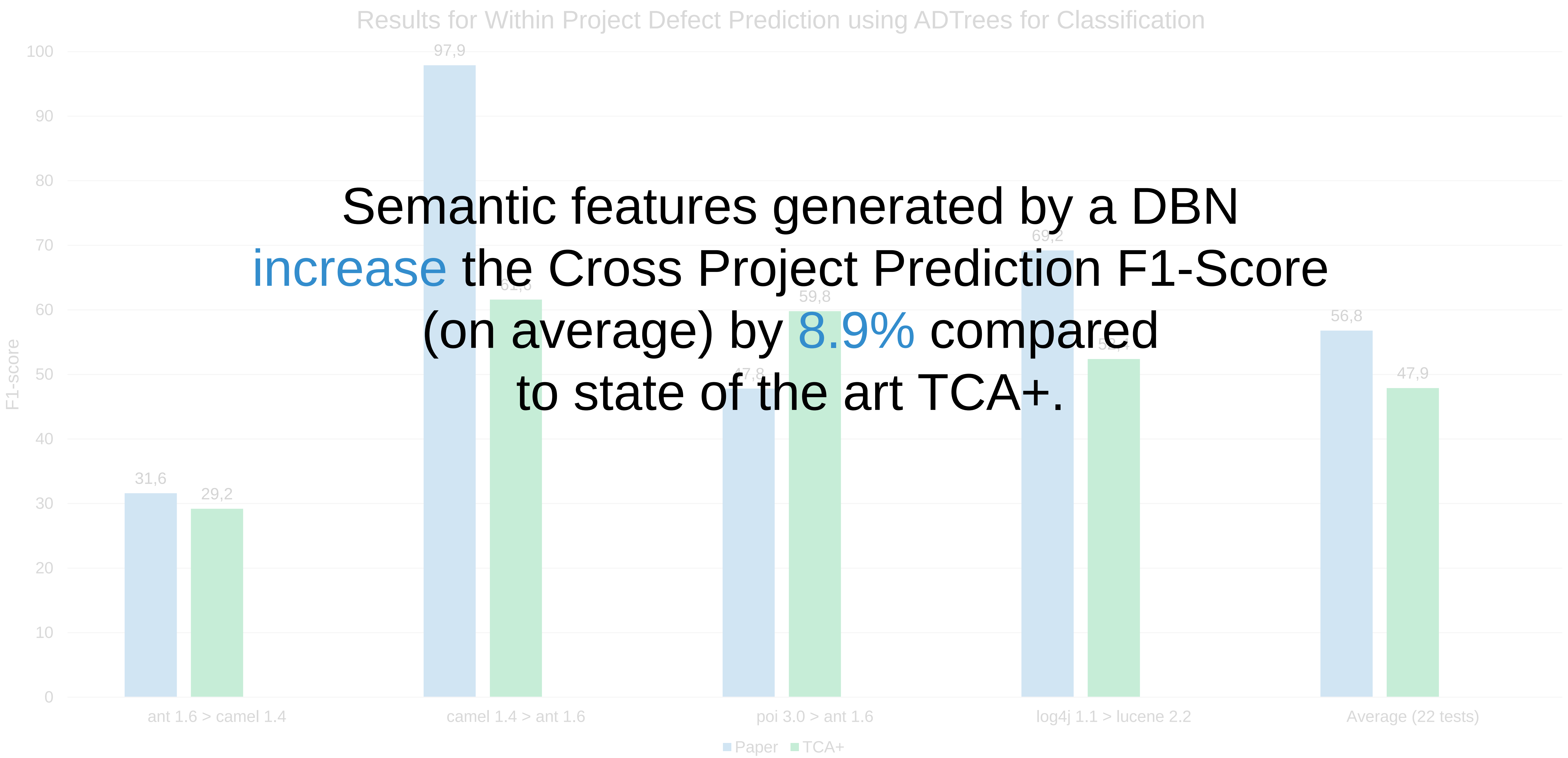
1. ADTrees perform better than Naïve Bayes and Logistic Regression.

2. Semantic features still outperform traditional features with other classifiers.

Results – Cross Project Defect Prediction

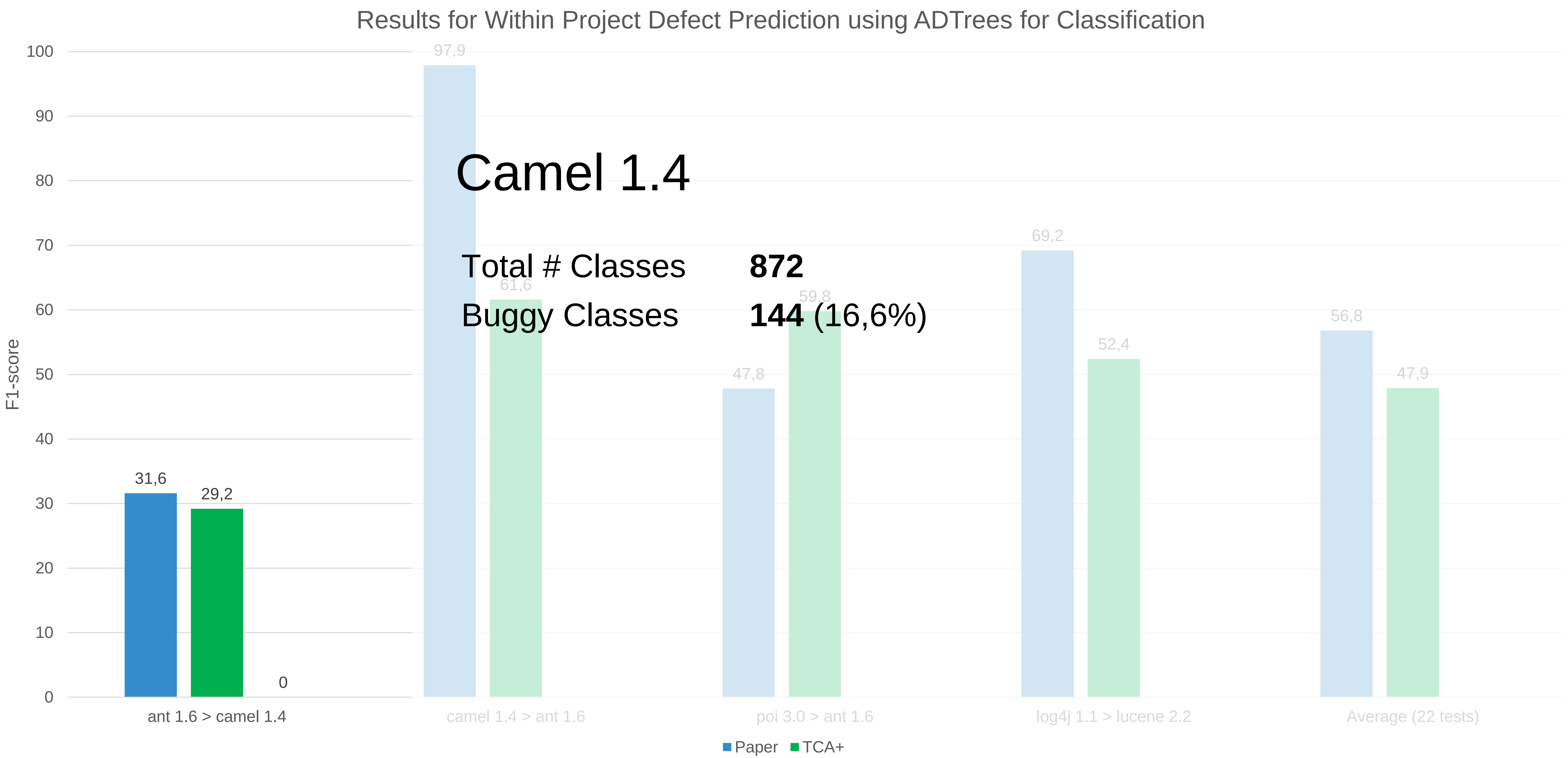


Results – Cross Project Defect Prediction



Semantic features generated by a DBN increase the Cross Project Prediction F1-Score (on average) by 8.9% compared to state of the art TCA+.

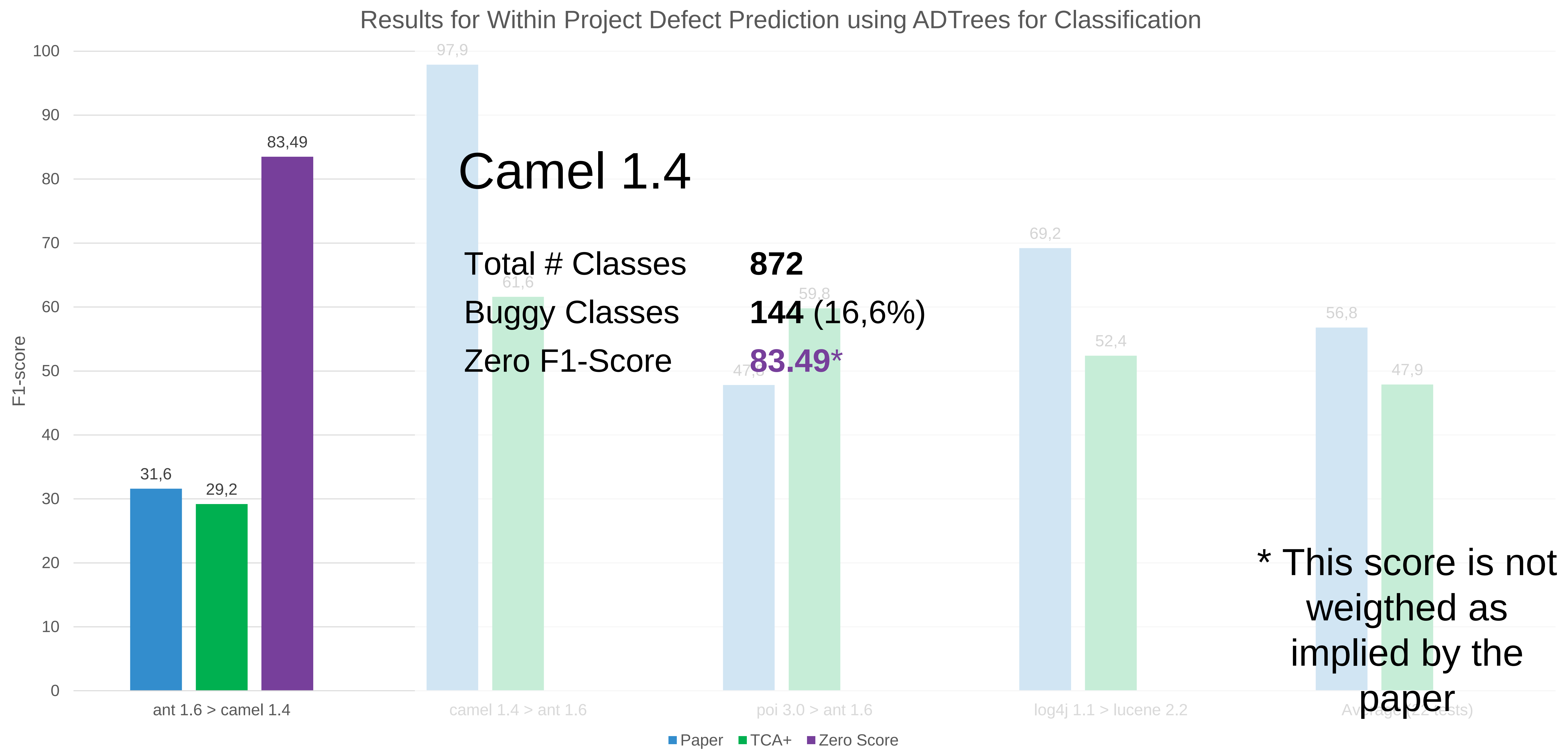
Results – Cross Project Defect Prediction



Camel 1.4

Total # Classes	**872**
Buggy Classes	**144** (16,6%)

Results – Cross Project Defect Prediction

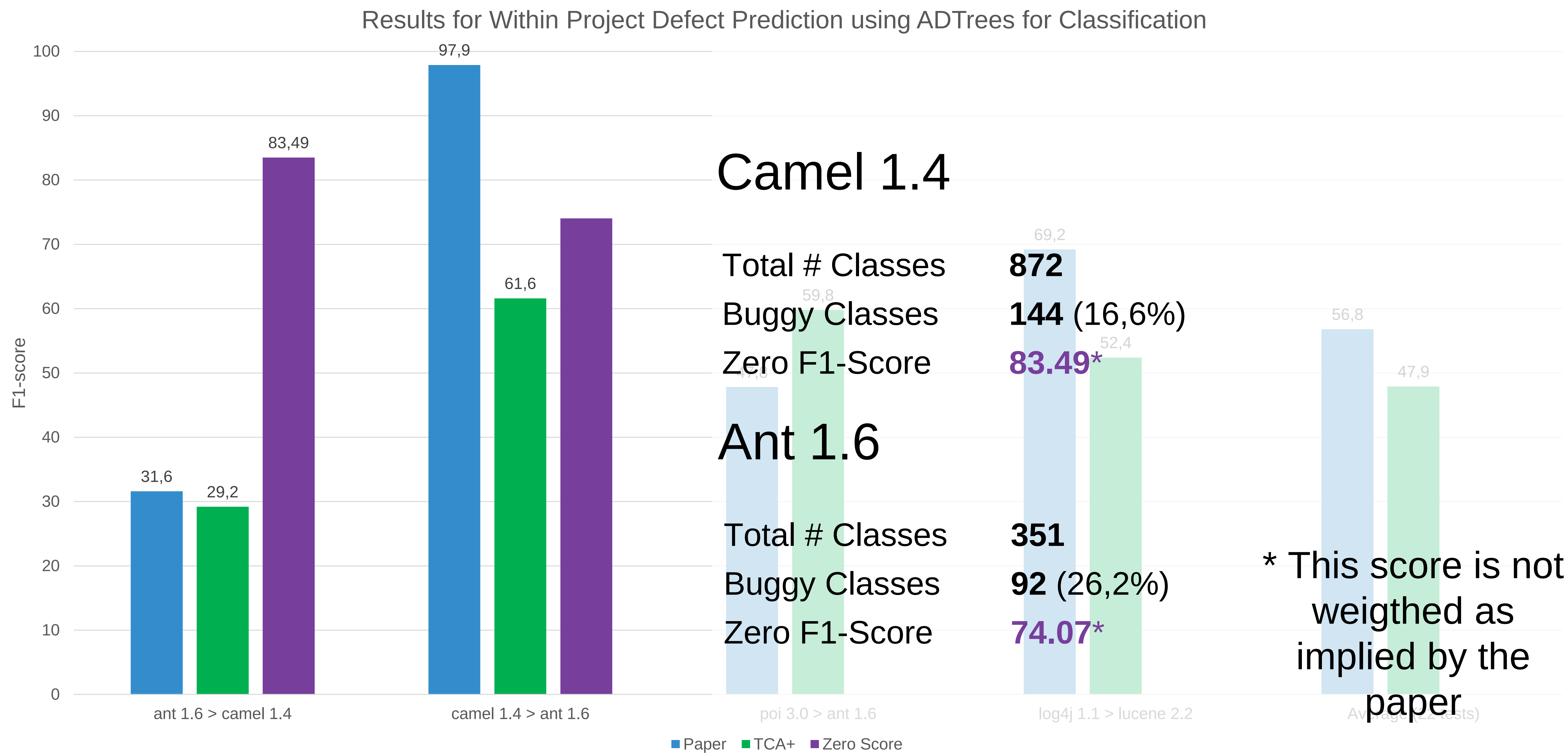


Camel 1.4

Total # Classes	**872**
Buggy Classes	**144** (16,6%)
Zero F1-Score	**83.49***

* This score is not weighted as implied by the paper

Results – Cross Project Defect Prediction



Camel 1.4

Total # Classes	**872**
Buggy Classes	**144** (16,6%)
Zero F1-Score	**83.49***

Ant 1.6

Total # Classes	**351**
Buggy Classes	**92** (26,2%)
Zero F1-Score	**74.07***

* This score is not weighted as implied by the paper

Results – How could the results be improved?

- Short Training (no overfitting yet)

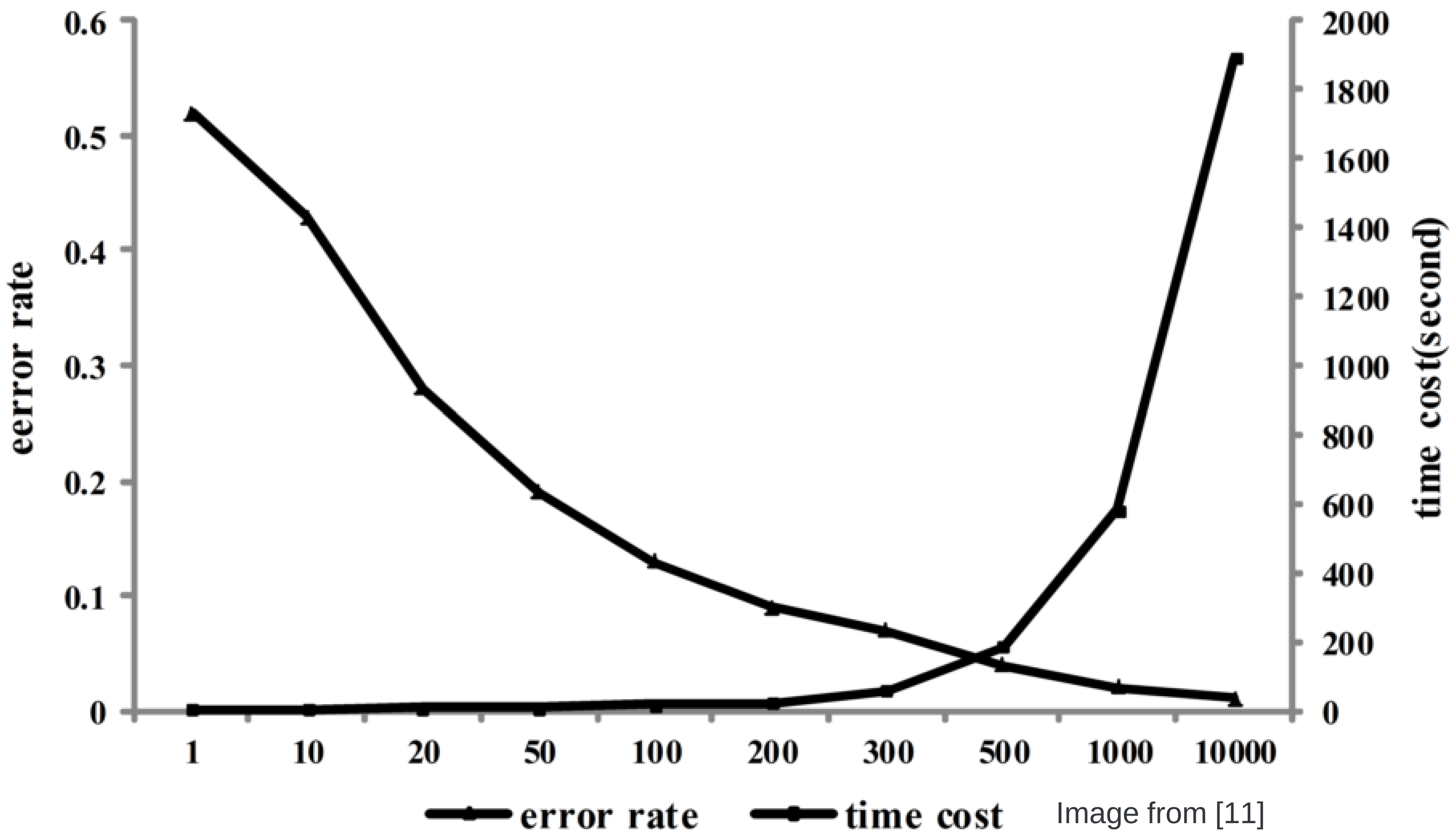
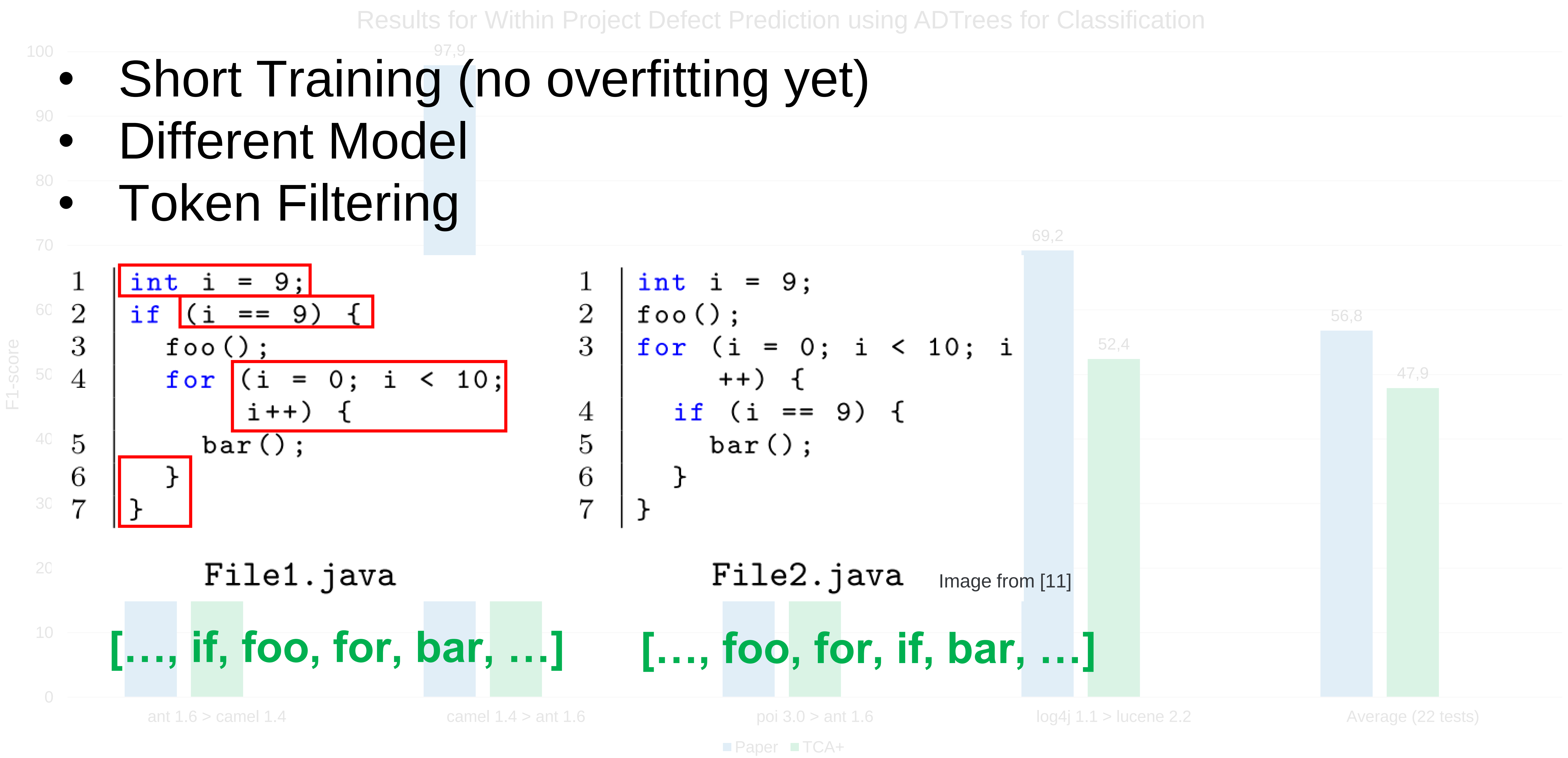


Image from [11]

Results – How could the results be improved?



- Short Training (no overfitting yet)
- Different Model
- Token Filtering

```java
int i = 9;
if (i == 9) {
   foo();
   for (i = 0; i < 10;
         i++) {
      bar();
   }
}
```

File1.java

```java
int i = 9;
foo();
for (i = 0; i < 10; i
      ++) {
   if (i == 9) {
      bar();
   }
}
```

File2.java

Image from [11]

[…, if, foo, for, bar, …] **[…, foo, for, if, bar, …]**

Agenda

- Defect Prediction
- Technical Background
- Discussion of Results
- Demo
- Conclusion

Defect Prediction - Demo

Can normal Neural Networks with GPU-Training improve the F1-Score?

Does the F1-Score increase if information about the control flow of the program is included?

Defect Prediction - Demo

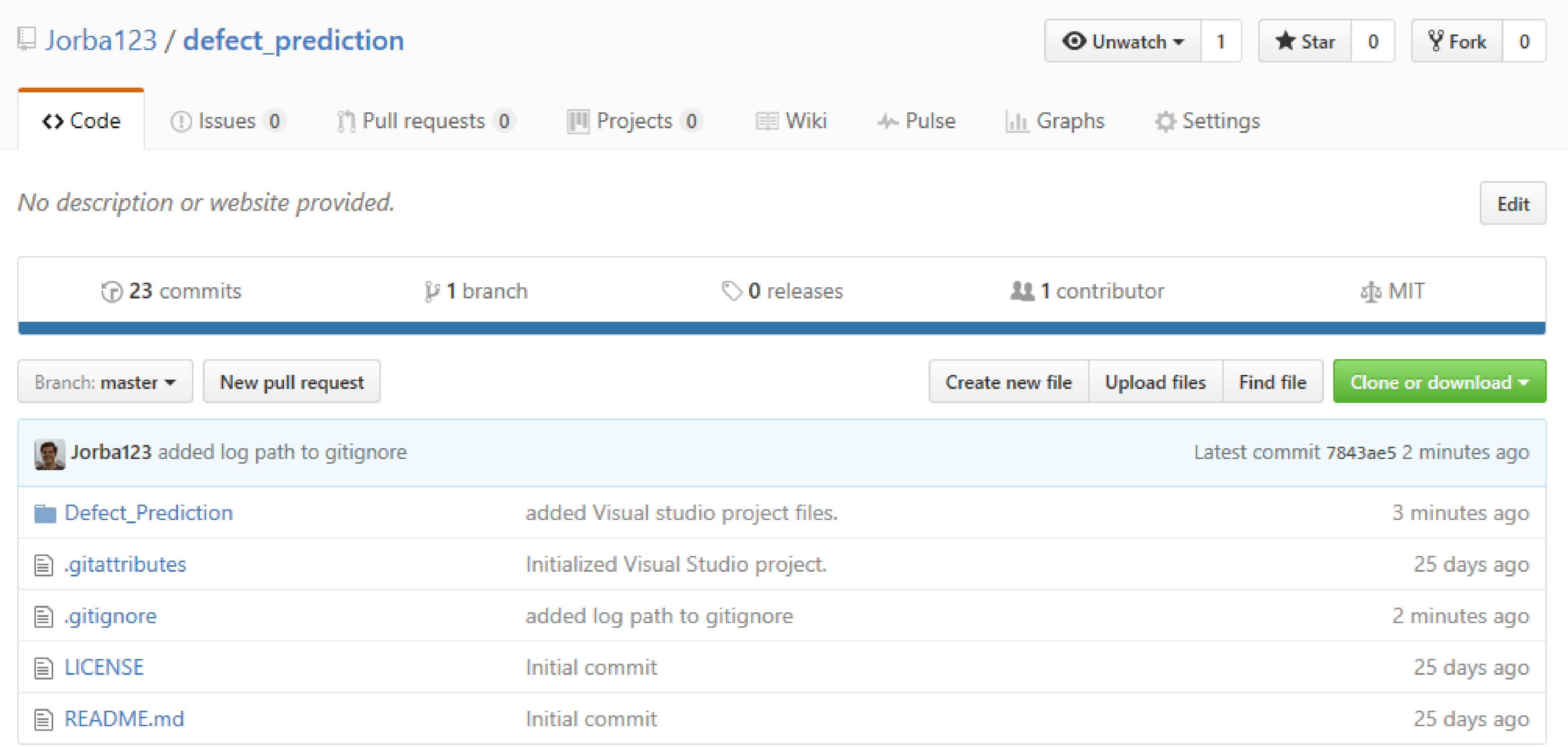
Can normal Neural Networks with GPU-Training improve the F1-Score?

Does the F1-Score increase if information about the control flow of the program is included?

NO ☹

(but I will show you the program nonetheless)

Defect Prediction - Demo



Download or clone at

https://github.com/Jorba123/defect_prediction

Discussion of Demo Result

- Number of training samples is very small
- Data is imbalanced (subsampling not possible)
- Sparse input features due to padding
- Possible labeling errors in data set
- Class-wise bug labeling

Agenda

Defect Prediction

Technical Background

Discussion of Results

Demo

Conclusion

Conclusion

- Defect Prediction can lower cost and save time during and after development
- Deep Belief Networks (DBNs) significantly outperform traditional features and classifiers
- Defect prediction rates differ a lot across projects
- Prediction rates for cross project prediction are lower than for within project prediction

Thank you!

References

- [1] F. Brady “Cambridge University Study States Software Bugs Cost Economy $312 Billion Per Year” prweb (2013) accessed 12.01.2017 http://www.prweb.com/releases/2013/1/prweb10298185.htm
- [2] R. Pielke, R. Byerly “Shuttle Programme Lifetime Cost.” In Nature Vol. 472 (2011)
- [3] “Milliardengrab A380” Handelsblatt (2012) http://www.genios.de/presse-archiv/artikel/HB/20121108/milliardengrab-a380/2B70894B-6C35-4D7B-A956-B0B5783C418D.html
- [4] World Bank – Last accessed 12.01.2017 http://data.worldbank.org/indicator/NY.GDP.MKTP.CD?locations=GR&name_desc=true
- [5] S. McConnell “Code Complete 2nd edition.” Microsoft Press (2004) page 521
- [6] S. Lohr, J. Markoff “Windows Is So Slow, but Why?” New York Times (2006) Last accessed 12.01.2017 http://www.nytimes.com/2006/03/27/technology/27soft.html?adxnnl=1&pagewanted=all&adxnnlx=1382805118-0jnNRGXEVPip3xoW+BDp8Q
- [7] B. Boehm, P. Papaccio “Understanding and Controlling Software Costs.” In: IEEE Transactions on Software Engineering (1988) page 1466
- [8] D. Ackley, G. Hinton, T. Sejnowski “A learning algorithm for boltzmann machines.” In: Cognitive Science (1985) Volume 9

References

- [9] G. Hinton, S. Osindero, Y. Teh “A Fast Learning Algorithm for Deep Belief Nets.” In: Neural Computation (2006) Volume 18

- [10] G. Hinton. “The Next Generation of Neural Networks.” Google TechTalks (2007) https://www.youtube.com/watch?v=AyzOUbkUf3M Last accessed: 12.01.2017

- [11] S. Wang, T. Liu, L. Tan. “Automatically learning semantic features for defect prediction.” In: Proceedings - International Conference on Software Engineering (2016)

- All Icons from Icons8 https://icons8.com/