Predicting Satisfiability in SAT3 & 3-Colorable Problems
with Graph Transformers and Long Short-Term Memory Networks

Tatiana Boura

NCSR Demokritos & University of Piraeus
MSc in Artificial Intelligence

Supervisor: Theodoros Giannakopoulos

Deep Learning Assignment
June, 2023

Table of Contents

The SAT3 problem

- A *satisfiability problem (SAT problem)* is a family of problems, where we examine if there is some truth-assignment to the variables that will make the entire expression true
- A **3SAT** problem is a special case of SAT problems, where the boolean expression has the conjunctive normal form (CNF). A CNF is a conjunction of one or more clauses, where a clause is a disjunction of literals
- In the 3SAT problem, every disjunction consists of 3 literals
- An example of a 3SAT problem instance is the following:

$$(x_1 \vee x_2 \vee \neg x_4) \wedge (\neg x_3 \vee x_4 \vee x_2)$$

NP-Complete problems

- The 3SAT problem is NP-complete problem and is used as the starting point to prove that the other problems (graph coloring, clique detection, dominating set, and vertex cover problems etc.) are also NP-Complete.
- These problems can be converted to one another in polynomial time

SAT3 solvers

- Many algorithms have been developed in order to prove the (un)satisfiability of the 3SAT problems and point a truth assignment for their variables
- All of them, being deterministic, have an *exponential worst time complexity*
- The state-of-the-art solvers can solve many large instances using heuristics, but still **very large instances cannot be solved**
- Deep Learning methods generalize and can **perform well on unseen data**, so researchers thought of using these methods to solve satisfiability problems
- In this assignment we use DL methods in order **predict the satisfiability** of an instance

Dataset

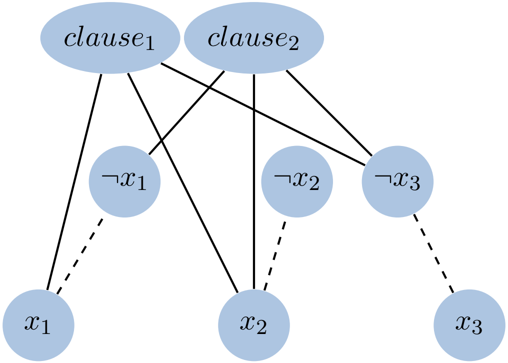
- For this classification task, the acquired dataset is Benchmark dataset for classic solvers : SATLIB-Benchmark Problems
- Dataset's name is the *"Uniform Random-3SAT"*

SAT3 as a Graph

- Given a problem instance with k clauses and n variables, we construct the undirected graph $G = (V, E)$ where the vertices V are the literals and the clauses of the CNF
- There are two types of edges E:
    1. One that connects each variable with its negation
    2. One connects each literal with every clause it appears in

SAT3 as a Graph

$$\overbrace{(x_1 \lor x_2 \lor \neg x_3)}^{clause_1} \land \overbrace{(\neg x_3 \lor \neg x_1 \lor x_2)}^{clause_2}$$



GNNs

- The GNN formalism is a general framework for defining deep neural networks on graph data
- Its main idea is that it generates representations of nodes that depend on the structure of the graph and on any feature information of the problem
- Its defining characteristic is that it uses a form of neural message passing in which vector messages are exchanged between neighboring nodes and updated using neural networks

Graph Transformers

- A Transformer model is a neural network that learns context by tracking relationships in sequential data
- They apply a technique, called attention, to detect subtle ways even distant data elements in a series influence and depend on each other
- In 2021 *Yunsheng Shi, Zhengjie Huang and others* infuse the vanilla multi-head attention into GNNs, considering the case of edge features
- This layer is called *Graph Transformer*. Neural networks that use this layer are called **Graph Transformer Networks (GTNs)**

Since we formulated the problem as a graph, we can now use a GTN architecture to solve our SAT3 problems

Graph Transformers

Why GTNs and not GNNs? ...

Graph Transformers

... because GTNs support *edge features*

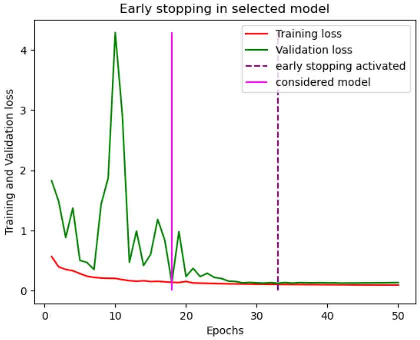
Model architecture

- Firstly, there is a **repeated block** of 3 layers:
  1. A *graph transformer* (TransformerConv) $\rightarrow$
  2. A *linear* layer that goes through a *ReLu* activation function $\rightarrow$
  3. A *batch normalization* layer $\rightarrow$
- After these blocks are *2 linear layers* each one followed by a *ReLu* activation function and a *final linear layer* that produces the output.

The number of repeated blocks, the input and output size of each layer (except for the input and output layer) are decided after **hyperparameter tuning**.

Training process

- The dataset was randomly split into *training* (64%), *validation* (16%) and *evaluation* sets (20%)
- The avoidance of overfitting is achieved using the regularization mechanism of *early stopping*



Training process

The rest of the training requirements are :

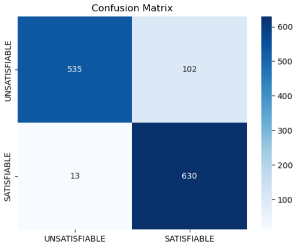
Loss Function	`torch.nn.BCEWithLogitsLoss`
Optimizer	`torch.optim.Adam`
Scheduler	`torch.optim.lr_scheduler.ExponentialLR`

Hyperparameter tuning

Model parameters			
Parameter	Possible Values	Selected Value	Explanation
embedding size	$\{64, 128\}$	64	Number of I/O samples passed through the Graph Transform network
attention heads	$\{1\}$	1	Number of multi-head-attentions in Graph Transform
number of layers	$\{2, 3, 4\}$	2	Number of repeated blocks
dropout rate	$\{0.1, 0.3, 0.5\}$	0.1	Dropout probability of the normalized attention coefficients in Graph Transform
dense neurons	$\{128, 256\}$	128	Number of I/O samples passed through the final linear layers
Training parameters			
Parameter	Possible Values	Selected Value	Explanation
batch size	$\{32, 64\}$	64	The number of training examples in one forward/backward pass
learning rate	$\{0.001, 0.01, 0.1\}$	0.01	Step size at each iteration
weight decay	$\{0.00001, 0.0001, 0.001\}$	$1e-05$	Weight decay (L2 penalty)

Table: Parameters to be tuned (GTN - same distribution)

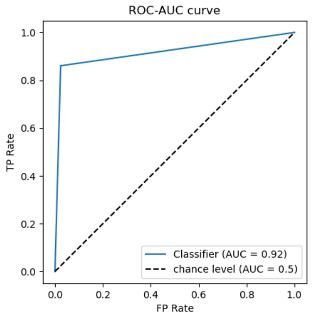
Evaluation (test set from the same distribution)



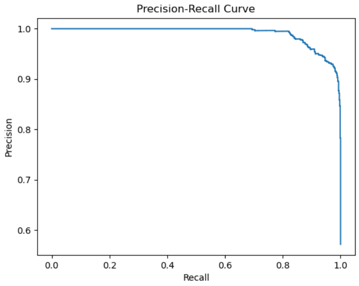
Performance	
Metric	**Value**
f1-score	0.9164
accuracy	0.9102
precision	0.9798
recall	0.8607
roc-auc	0.9185

Baseline is : 0.58

Evaluation (test set from the same distribution)



(a) ROC curve



(b) PR curve

Evaluation (test set from a different distribution)

- Recall that large SAT3 problem instances cannot be solved by classic solvers
- It makes sense to train neural networks to solve this problem as these models are capable of generalization and, maybe, can solve problem instances larger than those solved by standard algorithms
- In order to explore this possibility, **from the initial dataset I excluded the largest problems** and trained the model with the rest data (6200 train & valid - 200 test)
- The training and tuning process was the same one showcased earlier

Evaluation (test set from a different distribution)

Performance	
Metric	**Value**
f1-score	0.5525
accuracy	0.5950
precision	0.6173
recall	0.5000
roc-auc	0.5950

SAT3 as a sequence

- For all problem instances there is a clause number-threshold after which the problem is highly unlikely to be satisfiable
- Before this threshold, the problem is most likely to be satisfiable
- So, for any problem instance with k-clauses one can create a k-sized timeseries corresponding to that instance, by starting from one clause (3 literals) and labelling it as satisfiable and gradually adding clauses until the threshold is reached and the label is changed to unsatisfiable
- This data augmentation process is not infallible, but it is an accepted one

SAT3 as a sequence

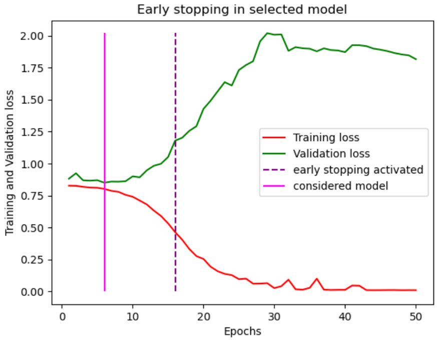
- As it is computationally very costly to perform the aforementioned sequencing of the data, here I created 2-sequence-length data as follows :
  1. For each instance is provided a satisfiable smaller version of it (the instance below the threshold) and
  2. For each unsatisfiable instance is provided another unsatisfiable instance (the instance just above the threshold)
- The dataset is split randomly at the preprocessing stage into training (80 %), validation (10 %) and evaluation (10 %) sets.

Model architecture

- Since we have sequential data we can use an **LSTM** architecture
- 2 architectures were tried
- The selected model has the following structure:
  1. One or more stacked LSTM layers $\rightarrow$
  2. A *linear* layer to produce the output
- The number of stacked LSTMs, the input and output size of each layer (except for the input and output) are tuned

Training Process

- Here, $max\_epochs = 50$ and $stopping\_counter = 10$
- We accepted the models that have been trained for at least 4 epochs



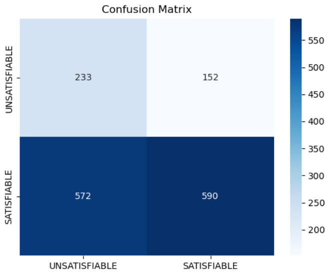
Hyperparameter tuning

Model parameters			
Parameter	Possible Values	Selected Value	Explanation
hidden units	$\{8, 16, 32, 64\}$	32	Number of features in the hidden state of the LSTM layers
number of layers	$\{1, 3, 5\}$	1	Number of recurrent layers - stacked LSTMs
dropout rate	$\{0.0, 0.3, 0.5, 0.8\}$	0.0	Introduces a Dropout layer on the outputs of each LSTM layer except the last one
Training parameters			
Parameter	Possible Values	Selected Value	Explanation
batch size	$\{16, 32, 64\}$	16	The number of training examples in one forward/backward pass
learning rate	$\{0.001, 0.01, 0.05, 0.1\}$	0.05	Step size at each iteration
weight decay	$\{0.00001, 0.0001, 0.001\}$	0.001	Weight decay (L2 penalty)

Table: Parameters to be tuned (LSTM)

Evaluation

Performance	
Metric	**Value**
f1-score	0.6197
accuracy	0.5320
precision	0.5077
recall	0.7951
roc-auc	0.5423



Comparison

Why are GTNs better that LSTMs for the SAT3?

Comparison

1. The LSTM needed to be trained with **more data** given the data's high dimensionality
2. To achieve better results we could have followed the computationally costly process of **data augmentation**
3. GTN architecture enforces enforces both **permutation invariance** and **negation invariance**

3-colorable problem

- In the beginning of this presentation we mentioned that there are other problems that belong to the same class as the SAT3 ones
- Each problem is transformed to the other efficiently
- One of those problems, is the **graph coloring problem**
- We use this model for predicting if a graph can be colored using *at most 3* colors by applying **transfer learning**

Dataset

- Again, we use a benchmark dataset for the *3-colorable graph* problem
- This dataset contains only 1700 satisfiable instances and no unsatisfiable ones
- Thus, we augment the dataset and create unsatisfiable problem instances from the existing satisfiable ones simply by *adding cliques of size 4* to each instance

Using the pre-trained model directly

Performance	
Metric	**Value**
f1-score	0.0531
accuracy	0.5279
precision	0.3750
recall	0.0286
roc-auc	0.4937

Re-training the model

Performance	
Metric	**Value**
f1-score	0.5115
accuracy	0.5309
precision	0.4941
recall	0.5302
roc-auc	0.5308

Evaluation

Why transfer learning did not work?

Evaluation

Many *possible* reasons:

1. The **amount of data** provided for re-training is not sufficient
2. The **data augmentation** that created unsatisfiable instances may not be of good quality
3. In the SAT3 dataset, all clauses have **no tautologies**, but in the graph coloring satisfiability problem tautologies are added into the clauses
4. Each dataset was created using a different distribution of variables and clauses $\rightarrow$ **graphs with different topologies**

Thank you for your attention!

A short demo follows