[[1]](#footnote-1)

Assignment 1 - Report

Jose Zavala, *Student, University of Essex*

*Abstract*— This report defines initial proposal of the creation of a Decision Tree Classifier model that can learn to *play* Tic-Tac-Toe by initially using Supervised Learning with a dataset generated by a Monte Carlo Tree Search with UCT that can successfully play the game, to the use a type of Reinforcement Learning to further tune the model to become a strong adversary in the game. The aim is to create a Decision Tree Classifier capable of recognizing the best possible move for a given game-state, without considering any other factors and without assuming any action on his adversary. This greedy approach has been successful in previous Tic-Tac-Toe playing agents, as well as the reinforcement learning techniques used have been used in the past with more complex models to successfully play more difficult games such as Go. It is expected that the proposed model will be able to recognize the strongest moves in the initial dataset since they should statistically be played more, as well as the reinforcement learning approach taken will accentuate the successful moves over the not-so-successful ones. A timetable of how the project will be taken into action is presented.

# INTRODUCTION

T

HIS project aims to demonstrate the use of supervised learning classifiers as tools powerful enough to imitate the best behavior of more complicated reinforcement learning models such as those used by the *Alpha Go* team to defeat the human world champion in a game of Go [Go].

The project will explore the use of a Decision Tree Classifier to imitate the behavior of an already existing Monte Carlo Tree Search solution, to then use simple reinforcement learning techniques to strengthen the model’s performance [Go].

DTs are to be used since they’ve been demonstrated to imitate MCTSs in the past [MCTS to Learn DT], as well as they offer a more memory and time efficient method to MCTS in practice [Alpaydin].

If performance results are satisfactory after experiments are conducted, this will show that at least for Full-Information games with small state, simpler models can learn to predict moves given enough data about the current state of the game.

The paper first shows previous research on MCTS and Decision Trees on the topic. Then, it further explains the model that we are used for training, as well as the MCTS that we will use to generate the initial training data. After this, the dataset recollection methodology is explained. Ultimately, the experiments to evaluate and further improve the performance of the DT are described.

# Background

The famous Tic-Tac-Toe, or OXO, is a “Full Information” game, which refers to the fact that all players have complete knowledge of the state of the match at any given point in time, unlike games like “Poker”, in which certain information (i.e. the hand of a given player) is unavailable [Implementing No-Loss, In Search of No-Loss]. Other examples of this type of games are Chess, Go, and Checkers.

## Tic-Tac-Toe Knowledge

The size of the board for a game of Tic-Tac-Toe is 3x3 and each slot can have any of three symbols (‘X’, ‘O’, *empty*) at any given time, so intuition indicates that there are or 19,683 possible game states [In Search of No-Loss]. However, these are limited by both conditions that restrict the feasibility of a game state of occurring (Figure 1) and the fact that by rotating the board some game states are equivalent (Figure 2) [In Search of No-Loss].

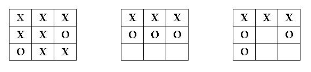


Figure 1 - Infeasible game-states [In Search of No-Loss]

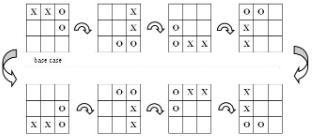


Figure 2 - Equivalent game-states [In Search of No-Loss]

These conditions reduce the intuitive number to 765 unique game-states.

## Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a commonly used technique for the creation of agents capable of playing these types of games [Go, Adversarial]. For instance, the recent success of the *AlphaGo*, which used supervised learning from real-game examples followed by reinforcement learning, team at defeating the human world champion of Go motivated the development of *AlphaGo Zero*, which implements a form of MCTS for unsupervised learning with a policy network to search high-probability moves [Go]. The results of the MCTS approach have been surprising, since *AlphaGo Zero* has defeated its predecessor with a score of 100-0 [Go].

MCTS uses a variation of the Upper Confidence Bounds (UBC) algorithm, called UCB applied to trees (UTC) [Bandit]. This multi-armed bandit approach takes the action that takes to a node which maximizes Equation 1 for the current state [Bandit]. Where is the number of wins after visiting the ith node, is the number of simulations after the ith node and is the total number of simulations ever considered [Bandit].

Equation 1 - UCT Node Selection [Bandit]

This allows the algorithm to explore the possible scenarios of different games just by correctly encoding its mechanics into the algorithm, which demonstrates the effectiveness of MCTS for tracking future game states and predicting the optimal move based on that information [Adversarial, Go]. However, its implementation is inherently time and resource expensive since it needs to recreate a full tree in each turn since the game state changes with every move. Also, the random nature of the algorithm allows to overlook specific game paths that may result on a loss, since these may be the consequence of “weird” plays.

## Alternatives to MCTS

The power of MCTS but the disadvantages it carries have motivated the development of models that imitate its behavior but are more time and resource efficient [MCTS to Learn DT, Thinking Fast]. These implementations aim to train models based on remembering the decisions taken by MCTS for quicker decision making [MCTS to Learn DT], or by observing multiple runs and training a model with the examples gathered and the decisions taken by the MCTS for a given state [Thinking Fast].

The objective of these implementations is to create a model that is capable of recognizing which actions are taken more frequently over different games, reducing time needed for decision making and removing the random factor of MCTS.

These models show that it is feasible to imitate the behavior of a MCTS with another, more time efficient, Machine Learning model, such as a Deep Neural network with promising results [Thinking Fast].

## Decision Trees

Decision Trees (DTs) have been proven to work against *Minimax,* another powerful algorithm that provides a No-Loss solution to the game [Implementing a No-Loss]. This works since *Minimax* has been proven to not perform optimal moves when the opponent makes a sub optimal choice [Implementing a No-Loss]; however, it maintains a No-Loss Strategy in a non-optimal number of moves. DTs have been trained to perform a best optimal action despite the opponent’s actions [Implementing a No-Loss]. However, this DT was manually created, which is non-optimal for our purposes.

Following the research mentioned on II.C, it is possible to implement a DT trained on different games performed by a MCTS, to learn to imitate the behavior of the algorithm without the need of creating a full tree each time it need to perform a move.

This provides enough theoretical ground to explore the performance of training a Decision Tree that imitates the behavior a MCTS to play Tic-Tac-Toe [Thinking Fast, MCTS to Learn DT, Implementing a No-Loss].

# Methodology

The goal of the analysis is to determine if a Decision Tree algorithm may be used to imitate the MCTS behavior for a game of Tic-Tac-Toe with successful results. Unlike [MCTS to Learn DT], the aim of this is not to train a DT with the UCT algorithm (Equation 1). Instead, the approach is to observe the decisions taken by an MCTS and train a DT with that information.

## Model

To achieve the goal, a DT with information grain (ID3) will be used [Alpaydin]. The model will receive the state of the board as an input of 9 features, each one symbolizing a position, and target those results to the cell that was selected by the MCTS algorithm.

The DT solution was selected for two major reasons. First, it will be able to identify the most selected plays by the MCTS for a given game state. Since the game has a very limited number of states [In Search of No-Loss], the DT should be able to determine which choices were most played overall, removing the *random* factor of the MCTS algorithm [Bandit]. The second reason is that, since the DT is going to be trained to model the MCTS agent, and not merely to learn “how to play”, the resulting tree will provide more insight about the relevance of certain cells over others at given game states.

To determine the decisions of the tree, the entropy function will be used to determine information gain [Alpaydin]. The Entropy Function (Equation 2) allows us to know the homogeneity of a sample. The more homogeneous it is, the more relevant it becomes to know which action to take [Alpaydin]. The information gain takes the Entropy of the current state and the Entropy of the state if a given feature (or cell in the board) was to be selected. The feature that gives the most Information Gain is selected to split the node at the given game state.

Equation 2 - Entropy Function

Equation 3 - Information Gain

This allows the DT to understand which cells are more important for the MCTS algorithm in a given situation having enough training examples to choose from. To “understand” the importance of a cell is to learn which cell’s states possess less entropy as to correctly discriminate the next action taken. At a leaf node, the action taken is the one with the most representation available, which statistically should be the most effective move performed by the target MCTS [Bandit].

To implement the DT, the DecisionTreeClassifier package from scikit-learn (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>) is going to be used. A game state will be encoded in a vector of data, in which each slot is a cell in the game board, and the target will be one number from 0 to 8, indicating the slot in which the next move will be performed.

## Dataset Collection

To gather data to train the proposed DT model, the code provided by the Monte Carlo Tree Search Research Hub (<http://mcts.ai/code/python.html>) was used as a basis. The original code provides an implementation of the UCT algorithm for three games: Nim’s Game, OXO (Tic-Tac-Toe), and Othello. The Code was modified to save the game states during a match from the perspective of the player whose current turn is being played.

Previous research mentioned in II.A indicates that there are 765 unique game states in the board at any possible time [In Search of N|o-Loss]. However, since neither MCTS nor DT can determine when a state is a rotation of another game state, we undo this knowledge to have a number of 6,120 game states. To be sure to capture all this, an arbitrary number of 10,000 games will be stored as to account for the validation set. No repeated game states will be removed since they reflect the frequency the MCTS makes those choices.

To allow our implementation to play either as “X” or “O”, we encoded the board of the game depending on the current player, and not as “X”’s and “O”’s, , being “X” if it indicates a piece placed by the current player, “Y” if it was placed by the rival, and “\_” represents an empty slot (Figure 5).

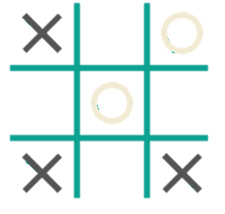


Figure 3 - Game State

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 2 | 0 | 2 | 0 | 1 | 0 | 1 |

Figure 4 - Original Encoding by MCTS Research Hub

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Y | \_ | X | \_ | X | \_ | Y | \_ | Y |

Figure 5 - Encoding Used if Second Player's Turn (O)

Upon inspection of data, it was decided that every move should be stored, even from the perspective of a losing agent. This was determined since MCTS will always perform the best possible move available for a given scenario [Bandit]. In an optimistic approach, even if the agent loses, it means that no better move was possible for a scenario [Go]. Also, it is assumed that in the long run, the MCTS algorithm will chose the best moves more frequently over the non-optimal ones [Bandit][Go], which the DT implementation should capture.

## Training

Training will ensure that the DT reflects the behavior of the MCTS algorithm, and not the performance of the DT as a playing agent. To evaluate the model’s performance over the original dataset, 10-Fold Cross Validation (CV) will be used. The CV is needed for exploring the values for the minimum impurity decrease for pre-pruning, diminishing the possibility of overfitting.

When an optimal parameter is found, the classifier will be trained over 70% of the dataset, using the remaining 30% as a testing set to measure performance with accuracy, precision, recall, and the F1-Score [Aplaydin].

# Experiments

All the processes described in Section III explain how the initial training and classification performance of the model will be realized. However, we now need to test the ability of the model as a candidate to compete against other agents in matches of Tic-Tac-Toe.

## Initial Performance Testing

To test the abilities of the DT, we will compare its gameplay against the original MCTS agent used to train it. We will observe its win\_rate score, which is only the percentage of the games that resulted in draw or a win, in order to aim for a No-Loss strategy performance for the DT.

100 games will be played against the original MCTS to calculate an initial win\_rate score. The games will be stored as explained in section III.B, but instead of storing all games, the model will be fine tuned with the movements of the winning agent. This was not done before since we assumed an optimal strategy from the MCTS in all scenarios, but with enough examples for basic play, we are now focusing on learning the best strategies. This method of further learning was also performed by [Go], in which they first explored Go games between expert human players and then let their own models to further play against previous iterations of themselves for faster learning with successful results.

## Further Training

After the initial plays with the MCTS agent, a second version will be developed with the new training examples. 10 games will be played against the first version, and the new win\_rate score will be saved. Again, new training examples will be saved from the winning agent’s moves [Go]. These moves will be used to train a new agent each time. To avoid overfitting on the same decisions each time, a random factor will be added, which will consist on selecting a random available move with a probability of 10% each turn.

This process will allow us to understand if the DT is able to learn from successful strategies if enough training examples are provided. The success of the model will be determined with the change of the win\_rate score over time.

# Discussion

As mentioned in Section IV, the performance of the DT as an agent capable of playing Tic-Tac-Toe will be measured with the win\_rate score over time.

Even though we are implementing the basic process used to train *Alpha Go* for the far more complex game of Go [Go], the team behind that agent was using a Reinforcement-Learning technique, which allowed the final model to find optimal strategies over time in a more dynamic environment.

There are several limitations to our approach, the first and most notorious one is that the model we are training is a supervised learning classifier and not a reinforcement learning agent [Alpaydin]. This means that it will only go as far as imitating the data it was trained with, and not to actually learn successful strategies. This is why we expect small performance improvement after competing against past versions of itself, since the only change is that it will add more weight to successful movements in a given game-state without knowledge on the opponent’s strategies nor previous time steps.

Another limitation is the way data will be stored in the Experiments section. Since only the actions already recorded by the classifier are going to be played, this may cause the model to overfit on a set of plays, which then may lose against the original MCTS model.

# Conclusion

This project will aim to implement a DT that at first will imitate the behavior of a MTCS playing a game of Tic-Tac-Toe, to then use reinforcement learning techniques [Go] to try to obtain more information about game-winning plays.

It is expected that the DT will learn form the MCTS with a high performance, but with a considerable classification error, since MCTS has a random factor which will cause it to perform different moves at the same state for different games, which will impact the classifier performance on initial testing.

For the reinforcement learning part, it is expected that the DT will improve its win\_rate score after some iterations against past versions of itself, but since it will only consist of adding new training examples of more efficient plays this increase will be limited.

# Plan

A Gantt Chart below show how the development of the project is going to be performed:

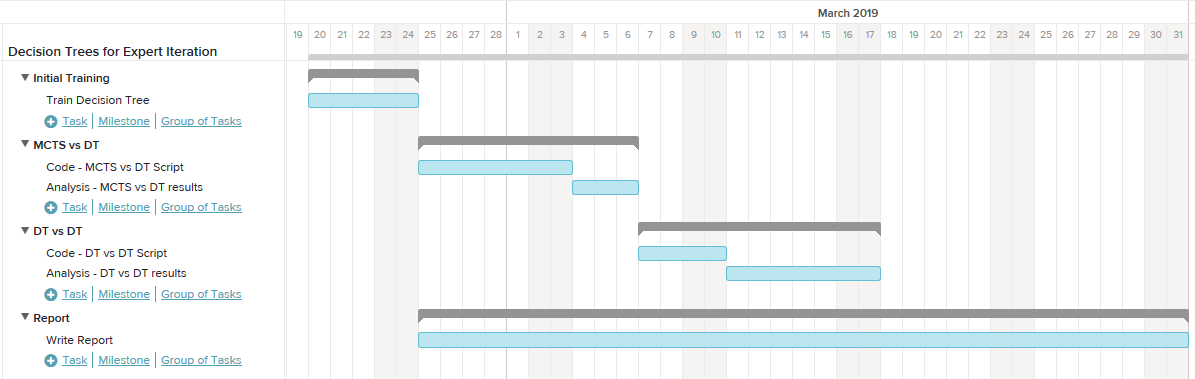


Figure - Gantt Chart of the Project

# References

*Basic format for books:*

J. K. Author, “Title of chapter in the book,” in *Title of His Published Book, x*th ed. City of Publisher, (only U.S. State), Country: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx.*

*Examples:*

1. G. O. Young, “Synthetic structure of industrial plastics,” in *Plastics,* 2nd ed., vol. 3, J. Peters, Ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15–64.
2. W.-K. Chen, *Linear Networks and Systems.* Belmont, CA, USA: Wadsworth, 1993, pp. 123–135.

*Basic format for periodicals:*

J. K. Author, “Name of paper,” *Abbrev. Title of Periodical*, vol. *x, no*. *x,* pp*. xxx-xxx,* Abbrev. Month, year, DOI. 10.1109.*XXX*.123456.

*Examples:*

1. J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility,” *IEEE Trans. Electron Devices*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959, 10.1109/TED.2016.2628402.
2. E. P. Wigner, “Theory of traveling-wave optical laser,”   
   *Phys. Rev*.,   
   vol. 134, pp. A635–A646, Dec. 1965.
3. E. H. Miller, “A note on reflector arrays,” *IEEE Trans. Antennas Propagat*., to be published.

*Basic format for reports:*

J. K. Author, “Title of report,” Abbrev. Name of Co., City of Co., Abbrev. State, Country, Rep. *xxx*, year.

*Examples:*

1. E. E. Reber, R. L. Michell, and C. J. Carter, “Oxygen absorption in the earth’s atmosphere,” Aerospace Corp., Los Angeles, CA, USA, Tech. Rep. TR-0200 (4230-46)-3, Nov. 1988.
2. J. H. Davis and J. R. Cogdell, “Calibration program for the 16-foot antenna,” Elect. Eng. Res. Lab., Univ. Texas, Austin, TX, USA, Tech. Memo. NGL-006-69-3, Nov. 15, 1987.

*Basic format for handbooks:*

*Name of Manual/Handbook, x* ed., Abbrev. Name of Co., City of Co., Abbrev. State, Country, year, pp. *xxx-xxx.*

*Examples:*

1. *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, USA, 1985, pp. 44–60.
2. *Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, USA, 1989.

*Basic format for books (when available online):*

J. K. Author, “Title of chapter in the book,” in *Title of Published Book*, *x*th ed. City of Publisher, State, Country: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx*. [Online]. Available: http://www.web.com

*Examples:*

1. G. O. Young, “Synthetic structure of industrial plastics,” in Plastics, vol. 3, Polymers of Hexadromicon, J. Peters, Ed., 2nd ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15-64. [Online]. Available: http://www.bookref.com.
2. *The Founders’ Constitution*, Philip B. Kurland and Ralph Lerner, eds., Chicago, IL, USA: Univ. Chicago Press, 1987. [Online]. Available: http://press-pubs.uchicago.edu/founders/
3. The Terahertz Wave eBook. ZOmega Terahertz Corp., 2014. [Online]. Available: http://dl.z-thz.com/eBook/zomega\_ebook\_pdf\_1206\_sr.pdf. Accessed on: May 19, 2014.
4. Philip B. Kurland and Ralph Lerner, eds., *The Founders’ Constitution.* Chicago, IL, USA: Univ. of Chicago Press, 1987, Accessed on: Feb. 28, 2010, [Online] Available: http://press-pubs.uchicago.edu/founders/

*Basic format for journals (when available online):*

J. K. Author, “Name of paper,” *Abbrev. Title of Periodical*, vol. *x*, no. *x*, pp. *xxx-xxx*, Abbrev. Month, year. Accessed on: Month, Day, year, DOI: 10.1109.*XXX*.123456, [Online].

*Examples:*

1. J. S. Turner, “New directions in communications,” *IEEE J. Sel. Areas Commun*., vol. 13, no. 1, pp. 11-23, Jan. 1995.
2. W. P. Risk, G. S. Kino, and H. J. Shaw, “Fiber-optic frequency shifter using a surface acoustic wave incident at an oblique angle,” *Opt. Lett.*, vol. 11, no. 2, pp. 115–117, Feb. 1986.
3. P. Kopyt *et al., “*Electric properties of graphene-based conductive layers from DC up to terahertz range,” *IEEE THz Sci. Technol.,* to be published. DOI: 10.1109/TTHZ.2016.2544142.

*Basic format for papers presented at conferences (when available online):*

J.K. Author. (year, month). Title. presented at abbrev. conference title. [Type of Medium]. Available: site/path/file

*Example:*

1. PROCESS Corporation, Boston, MA, USA. Intranets: Internet technologies deployed behind the firewall for corporate productivity. Presented at INET96 Annual Meeting. [Online]. Available: http://home.process.com/Intranets/wp2.htp

*Basic format for reports and handbooks (when available online):*

J. K. Author. “Title of report,” Company. City, State, Country. Rep. no., (optional: vol./issue), Date. [Online] Available: site/path/file

*Examples:*

1. R. J. Hijmans and J. van Etten, “Raster: Geographic analysis and modeling with raster data,” R Package Version 2.0-12, Jan. 12, 2012. [Online]. Available: http://CRAN.R-project.org/package=raster
2. Teralyzer. Lytera UG, Kirchhain, Germany [Online]. Available: http://www.lytera.de/Terahertz\_THz\_Spectroscopy.php?id=home, Accessed on: Jun. 5, 2014

*Basic format for computer programs and electronic documents (when available online):*

Legislative body. Number of Congress, Session. (year, month day). *Number of bill or resolution*, *Title*. [Type of medium]. Available: site/path/file

***NOTE:*** ISO recommends that capitalization follow the accepted practice for the language or script in which the information is given.

*Example:*

1. U.S. House. 102nd Congress, 1st Session. (1991, Jan. 11). *H. Con. Res. 1, Sense of the Congress on Approval of Military Action*. [Online]. Available: LEXIS Library: GENFED File: BILLS

*Basic format for patents (when available online):*

Name of the invention, by inventor’s name. (year, month day). Patent Number[Type of medium]. Available: site/path/file

*Example:*

1. Musical toothbrush with mirror, by L.M.R. Brooks. (1992, May 19). Patent D 326 189

[Online]. Available: NEXIS Library: LEXPAT File: DES

*Basic format for conference proceedings (published):*

J. K. Author, “Title of paper,” in *Abbreviated Name of Conf.*, City of Conf., Abbrev. State (if given), Country, year, pp. *xxxxxx.*

*Example:*

1. D. B. Payne and J. R. Stern, “Wavelength-switched pas- sively coupled single-mode optical network,” in *Proc. IOOC-ECOC,* Boston, MA, USA,1985,   
   pp. 585–590.

*Example for papers presented at conferences (unpublished):*

1. D. Ebehard and E. Voges, “Digital single sideband detection for interferometric sensors,” presented at the *2nd Int. Conf. Optical Fiber Sensors,* Stuttgart, Germany, Jan. 2-5, 1984.

*Basic format for patents:*

J. K. Author, “Title of patent,” U.S. Patent *x xxx xxx*, Abbrev. Month, day, year.

*Example:*

1. G. Brandli and M. Dick, “Alternating current fed power supply,” U.S. Patent 4 084 217, Nov. 4, 1978.

*Basic format**for theses (M.S.) and dissertations (Ph.D.):*

a) J. K. Author, “Title of thesis,” M.S. thesis, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

b) J. K. Author, “Title of dissertation,” Ph.D. dissertation, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

*Examples:*

1. J. O. Williams, “Narrow-band analyzer,” Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, USA, 1993.
2. N. Kawasaki, “Parametric study of thermal and chemical nonequilibrium nozzle flow,” M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

*Basic format for the most common types of unpublished references:*

a) J. K. Author, private communication, Abbrev. Month, year.

b) J. K. Author, “Title of paper,” unpublished.

c) J. K. Author, “Title of paper,” to be published.

*Examples:*

1. A. Harrison, private communication, May 1995.
2. B. Smith, “An approach to graphs of linear forms,” unpublished.
3. A. Brahms, “Representation error for real numbers in binary computer arithmetic,” IEEE Computer Group Repository, Paper R-67-85.

*Basic formats for standards:*

a) *Title of Standard*, Standard number, date.

b) *Title of Standard*, Standard number, Corporate author, location, date.

*Examples:*

1. IEEE Criteria for Class IE Electric Systems, IEEE Standard 308, 1969.
2. Letter Symbols for Quantities, ANSI Standard Y10.5-1968.

*Article number in reference examples:*

1. R. Fardel, M. Nagel, F. Nuesch, T. Lippert, and A. Wokaun, “Fabrication of organic light emitting diode pixels by laser-assisted forward transfer,” *Appl. Phys. Lett.*, vol. 91, no. 6, Aug. 2007, Art. no. 061103.
2. J. Zhang and N. Tansu, “Optical gain and laser characteristics of InGaN quantum wells on ternary InGaN substrates,” *IEEE Photon. J.*, vol. 5, no. 2, Apr. 2013, Art. no. 2600111

*Example when using et al.:*

1. S. Azodolmolky *et al.*, Experimental demonstration of an impairment aware network planning and operation tool for transparent/translucent optical networks,” *J. Lightw. Technol.*, vol. 29, no. 4, pp. 439–448, Sep. 2011.

1. [↑](#footnote-ref-1)