Reninforcement Learning Application

Requirements Specification and Analysis

1.0

21.02.2019

Ersin ÇEBİ

Prepared for

COMP4902 Graduation Design Project



**Table of Contents**

[1.](#_1fob9te) Introduction 1

[1.1.](#_3znysh7) Purpose of the System 1

[1.2.](#_2et92p0) Scope of the System 1

[1.3.](#_tyjcwt) Objectives and Success Criteria of the Project 1

[1.4.](#_3dy6vkm) Definitions, Acronyms, and Abbreviations 1

[1.5.](#_1t3h5sf) Overview 1

[2.](#_4d34og8) Current System 1

[3.](#_2s8eyo1) Proposed System 1

[3.1.](#_17dp8vu) Overview 1

[3.2.](#_3rdcrjn) Functional Requirements 2

[3.3.](#_26in1rg) Nonfunctional Requirements 2

[Usability 2](#_lnxbz9)

[Reliability 2](#_35nkun2)

[Performance 2](#_1ksv4uv)

[Supportability 2](#_44sinio)

[Implementation 2](#_2jxsxqh)

[Interface 2](#_z337ya)

[Packaging 2](#_3j2qqm3)

[Legal 2](#_1y810tw)

[3.4.](#_4i7ojhp) System Models 2

[Scenarios 2](#_2xcytpi)

[Use case model 2](#_1ci93xb)

[Object model 2](#_3whwml4)

[Dynamic model 2](#_2bn6wsx)

[User interface—navigational paths and screen mock-ups 3](#_qsh70q)

[3.5.](#_3as4poj) Project Schedule 3

[4.](#_1pxezwc) Glossary 3

[5.](#_49x2ik5) References 3

**REQUIREMENTS ANALYSIS DOCUMENT**

**1. Introduction**

There is supervised and unsupervised learning techniques, Reinforcement Learning is under unsupervised learning technique. In unsupervised learning, the training data is unlabeled. The system tries to learn without a teacher. So basically the agent observerses the environment and learns how to reach the end point or the goal that defined.

When we think about the concept of learning, one of the first things that come to our mind is interaction-based learning. While babies are playing games, waving their arms or looking left and right, there is no teacher telling them what to do with their heads.

Reinforcement Learning agents have clear goals, they can feel the characteristics of their environment and choose the actions that will be effective in their environment. In the basic structure, the agent shows an action according to the environment, it is called policy and expects a response from the environment. The resulting reactions are subject to a predefined reward system. In line with the award won, the agent is trained and understands how wrong or right he is doing. The agent should try various actions and gradually choose the ones that look best.

Application areas; production planning, classification of blood analysis, brain modeling studies, quality control, fingerprint recognition, automatic vehicle inspection, determining credit card tricks, optimal route determination for intelligent vehicles and robots, predicting the lifespan of mechanical parts, voice recognition, inspection, meteorological interpretation, Examples include electrical sign recognition, handwriting recognition, identification and treatment of diseases, classification of radar and sonar signals, filtering spam mails.

* 1. **Purpose of the System**

The aim of the system is designing an accurate and efficient reinforcement learning model, than simple DENSE(fully connected) model from keras.

* 1. **Scope of the System**

The project is based on a new approach on reinforcement learning that can learn its receptive field and so its local connections in a topological structure. In Keras, you assemble layers to build models. A model is a graph of layers. The most common type of model is a stack of layers the sequential model. By using Keras’s Dense model I will implement and develop a structure to the project.

* 1. **Objectives and Success Criteria of the Project**

**Following objectives are planing to achieve in this project:**

1. Development of the new model
2. Testing of a new model.
3. Development of a simple user interface to demonstrate model dynamics.
4. Application and comparison of the model in 3 gym environment, including the environment that prepared by me.

Success criteria of the project is to get more accurate results than a Simple Dense model.

**1**- More Effective than Simple Dense Model

**2**- Easy to implement

**3**- Time Efficiency

**1.4. Definitions, Acronyms, and Abbreviations**

RAD: Requirements Analysis Document

**1.5. Overview**

As a summary there is plenty of techniques in reinforcement learning such as Q-Learning, Q-Learning is a basic form of Reinforcement Learning which uses Q-values (also called action values) to iteratively improve the behavior of the learning agent. The object diagram, class diagram and a gantt chart to understand the project. I try to tell you how this project will end. Also what we have and what we will develop. We will discuss them in the following chapters.

**2. Current System**

### There is plenty of types and algorithms of reinforcement learning, such as, deep reinforcement learning approach extends reinforcement learning by using a deep neural network and without explicitly designing the state space. Inverse reinforcement learning (IRL), no reward function is given. Instead, the reward function is inferred given an observed behavior from an expert. The idea is to mimic observed behavior, which is often optimal or close to optimal. Apprenticeship learning, an expert demonstrates the target behavior. The system tries to recover the policy via observation. In this project I will follow deep reinforcement learning for my model. Also, implement FocusedNN and compare their success according to their score table and graphs. Focused RNN is a neural network system that have different parameter and structure from feedforward networks.

**3. Proposed System**

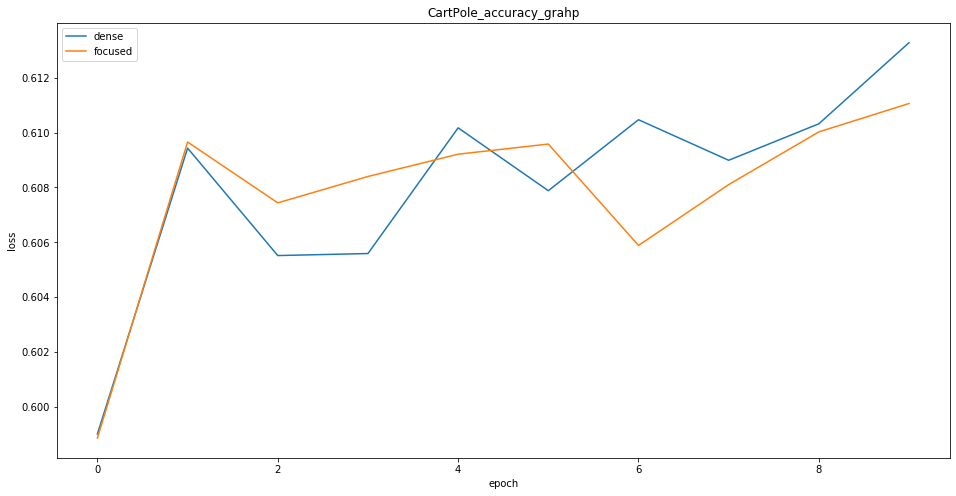
**3.1. Overview**

The FocusedRNN implementation allows people to test the datasets by trained FocusedRNN. It trained faster than Dense layer and comparable accuracy rate to Dense.

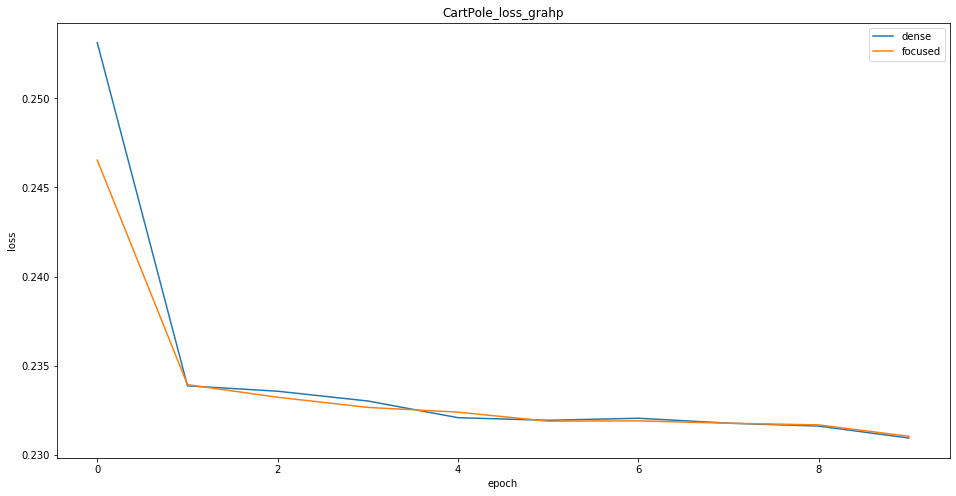
In my tests, I use three different gym-environment to create train-test datas. I run both FocusedRNN and Dense layers and take several graphs, such as loss and accuracy, also create score graphs for each FocusedRNN, Dense runs and search for a trend. While training the environments on building model block environments creates 10K to 130K dimensional data and I run 5-10 iterations for each, not all agents show ascending trend, their scores was good and it was higher the score\_requerinment that, in the other hand their loss and accuracy was quite good. The question is how can I make ascending trend, lower loss and higher accuracy at the same time?

One way is playing with the functions defined(loss, activation functions) and changing neuron count for each layer, but it might drive you to a endless road because it’s like a blackbox, you can not be sure the result and values you given is right for each environment and agent. I will try to find out which parameters are best. Also, I will make FocusRNN stable for most situation.

Here graphs of CardPole-env, for showing FocusedNN and Dense performance;



In accuracy graph after 5th iteration Focused acc drops, this means it diverges from the solution at that point.



In loss graph after 1st iteration both Focused and dense loss drops dramatically, but after there is no explicit change on loss.

**3.2. Functional Requirements**

|  |
| --- |
| **Scenario name:** Taking Training Data |
| **Participant actor instances:** User |
| 1. User opens the colab page 2. User chooses the required environment 3. Defines the optimizer, episodes, layer option(FocusedNN or Dense), test iteration 4. Runs the required environments training\_data method |

|  |
| --- |
| **Scenario name:** Training Model |
| **Participant actor instances:** User |
| 1. User runs the required environments training\_data method 2. Users runs train\_model method and method returns ‘trained\_model, model\_history’ datas |

|  |
| --- |
| **Scenario name:** Testing Model |
| **Participant actor instances:** User |
| 1. User runs the required environments ‘Run’ method 2. This method returns scores, and choices that agent made |

|  |
| --- |
| **Scenario name:** Plotting Results |
| **Participant actor instances:** User |
| 1. User sends score and model\_history datas into plotTheValues() method 2. the method gives gives trend line according to score value, shows loss and accuracy graphs to user, also gives the percentages of agent that, which action does it take. |

**3.3. Nonfunctional Requirements**

**Usability:** A user who wants to build a Reinforcement Learning, can build a network with setting a few parameters in a web or Google Colab interface.

**Reliability:** The system must be stable to many cases .

**Performance:** The system must faster than Simple Dense.

**Supportability:** The system must be able to be maintained and changeable easily. Implementing with Tensorflow, Keras and also it can be run with keras functions.

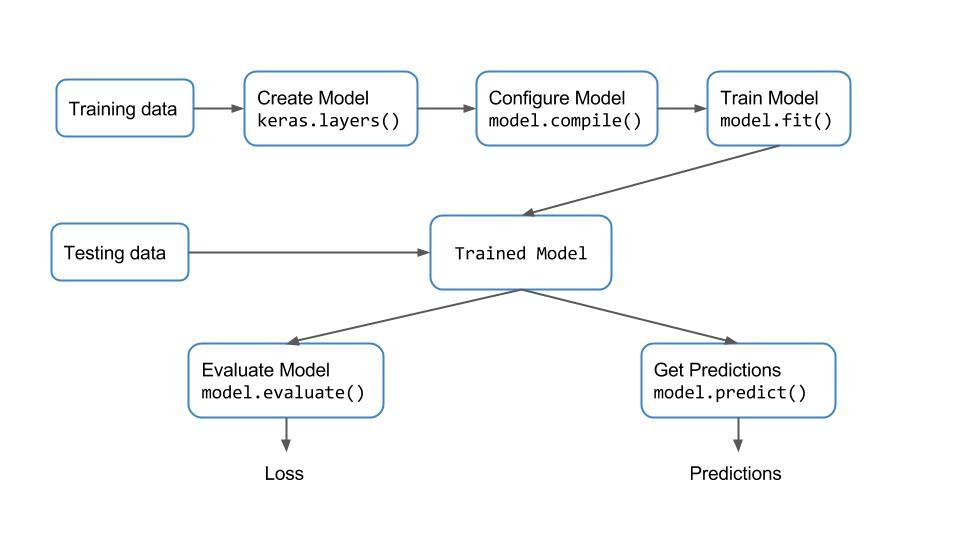
**Implementation:** The system will be implemented on .py and .ipynb platform. In addition, Python 3.7 will be used as the programming language in the process.

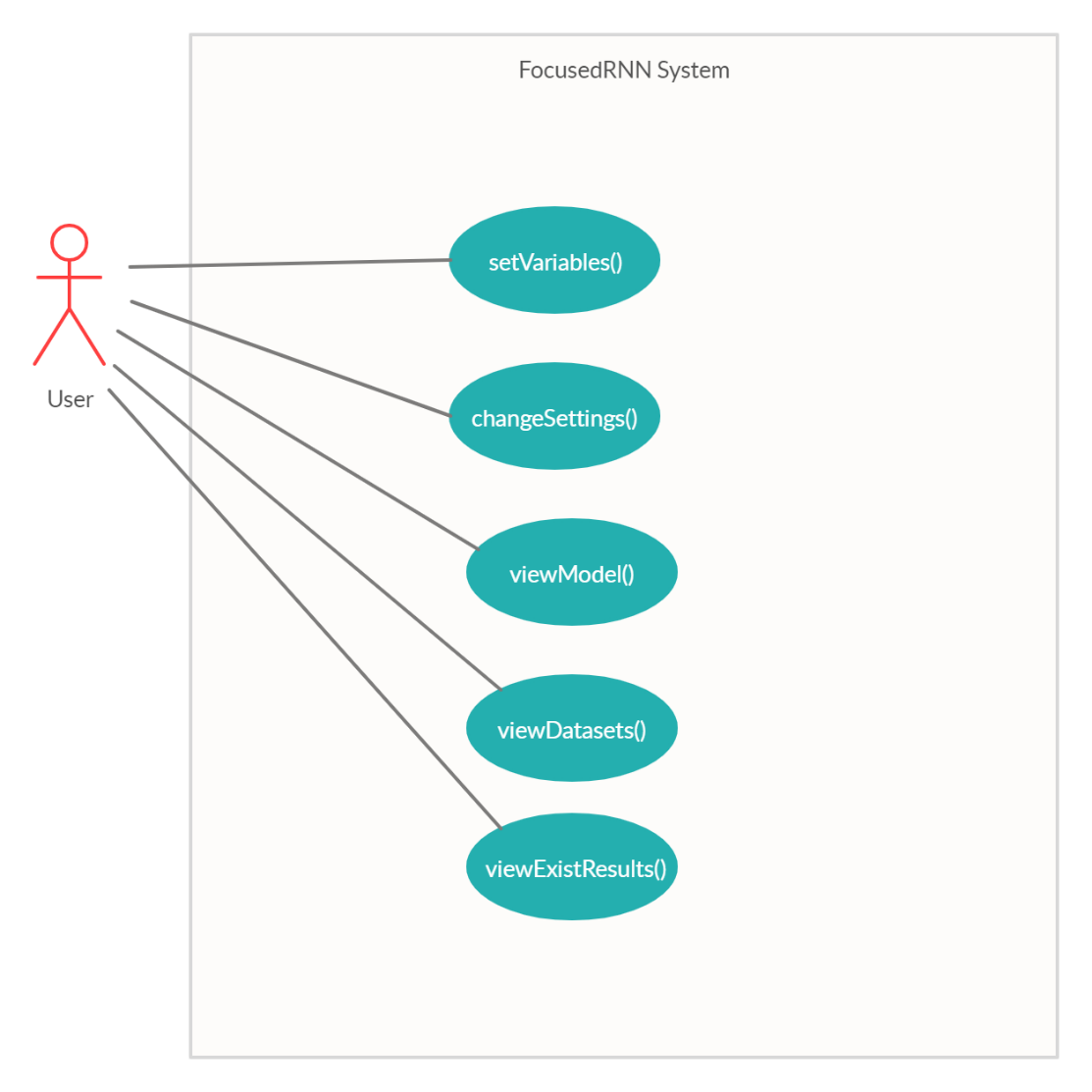
**Interface:** There is a web, Google Colab interface or terminal to set or change some variables and settings.

**Packaging:** No constraints on the actual delivery of the system are determined. Thus, packaging requirements will be decided in the future.

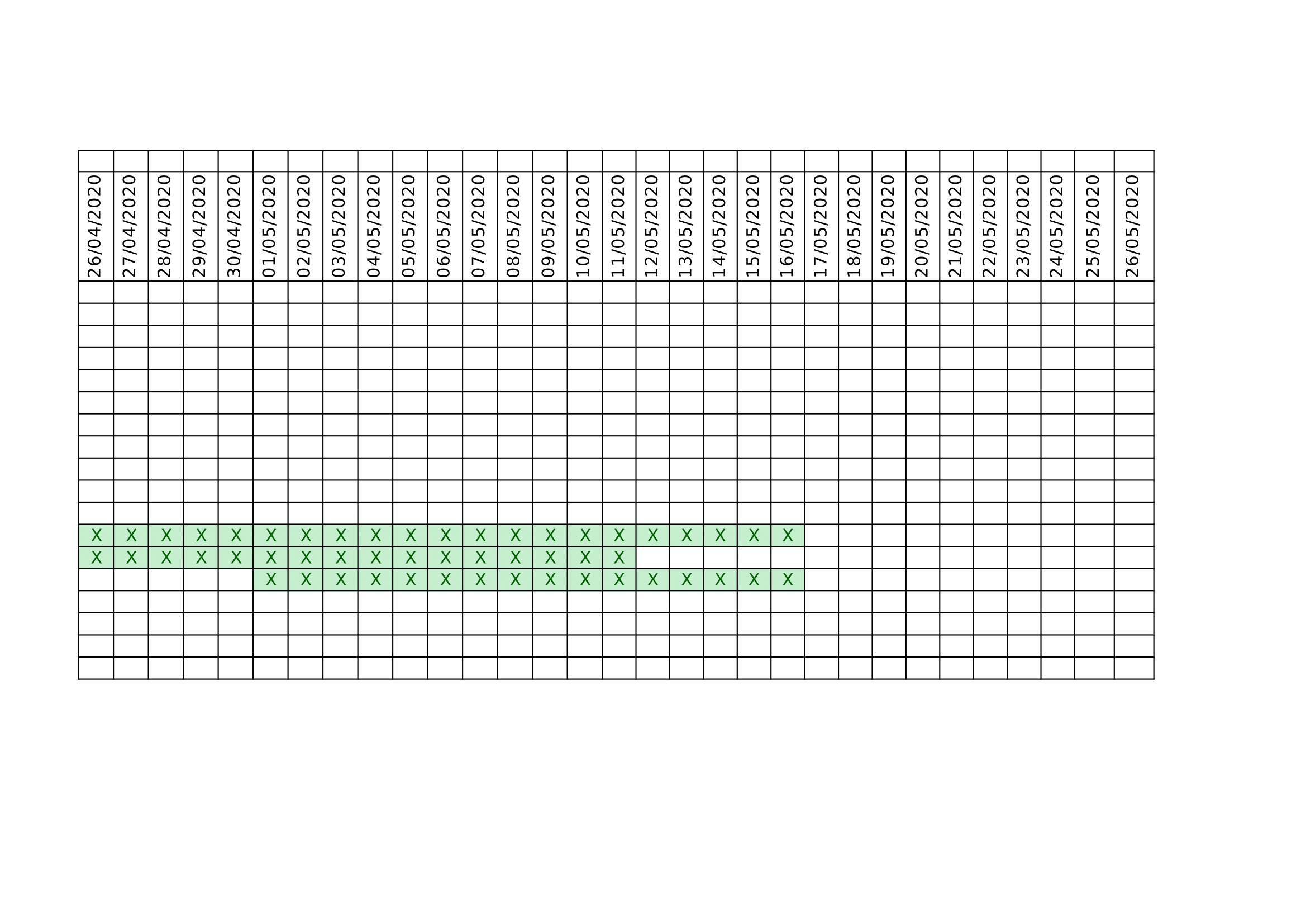
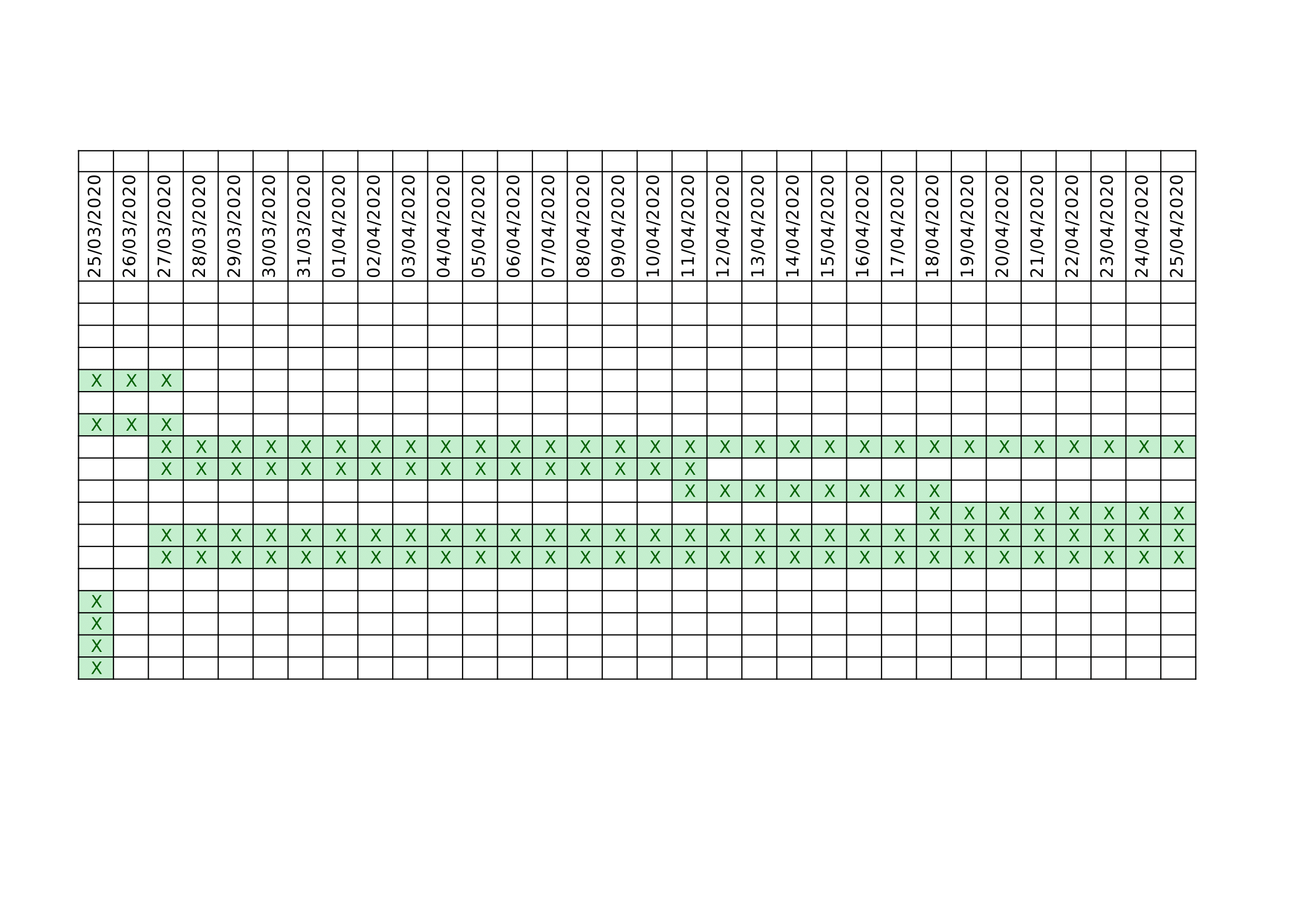
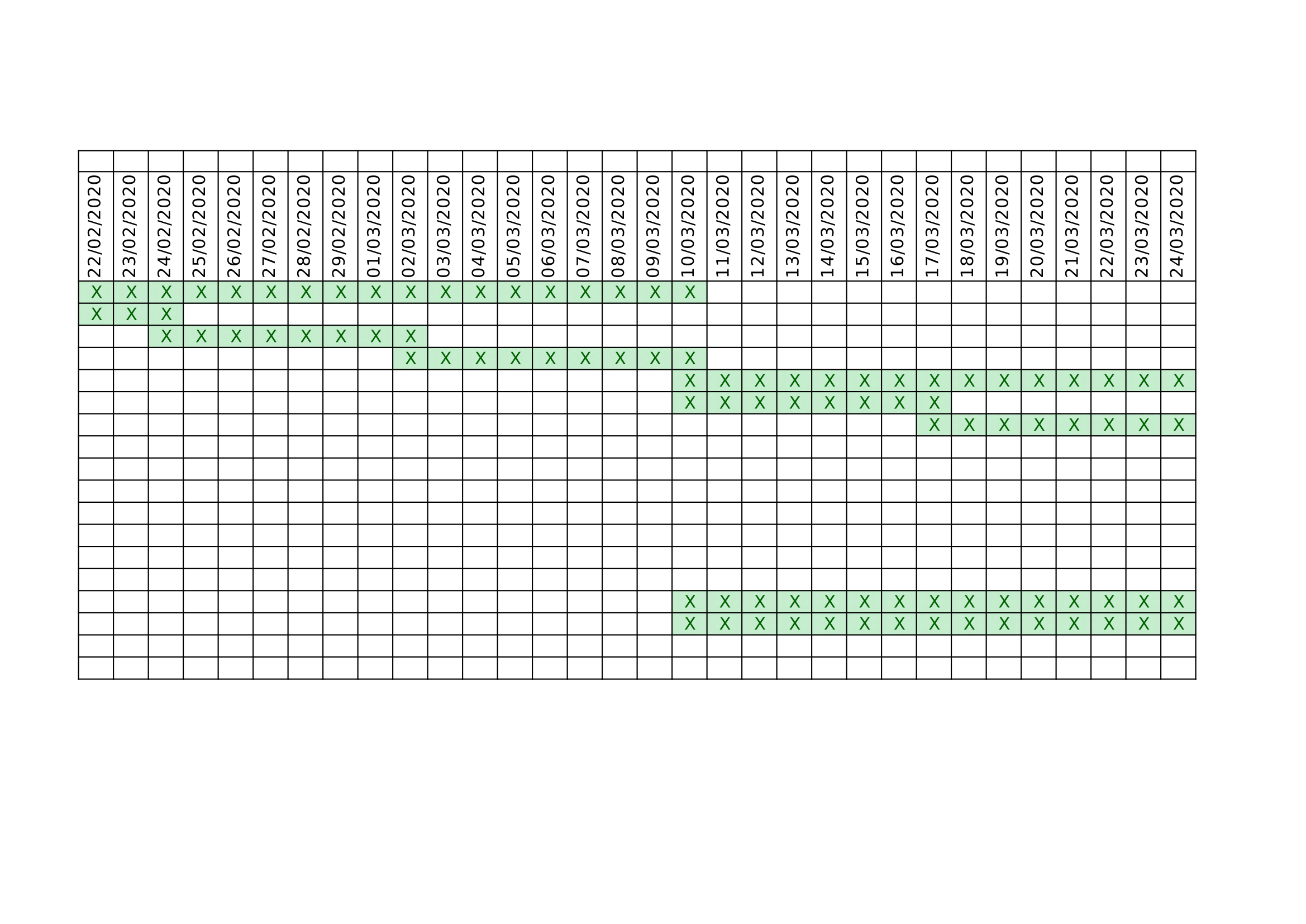
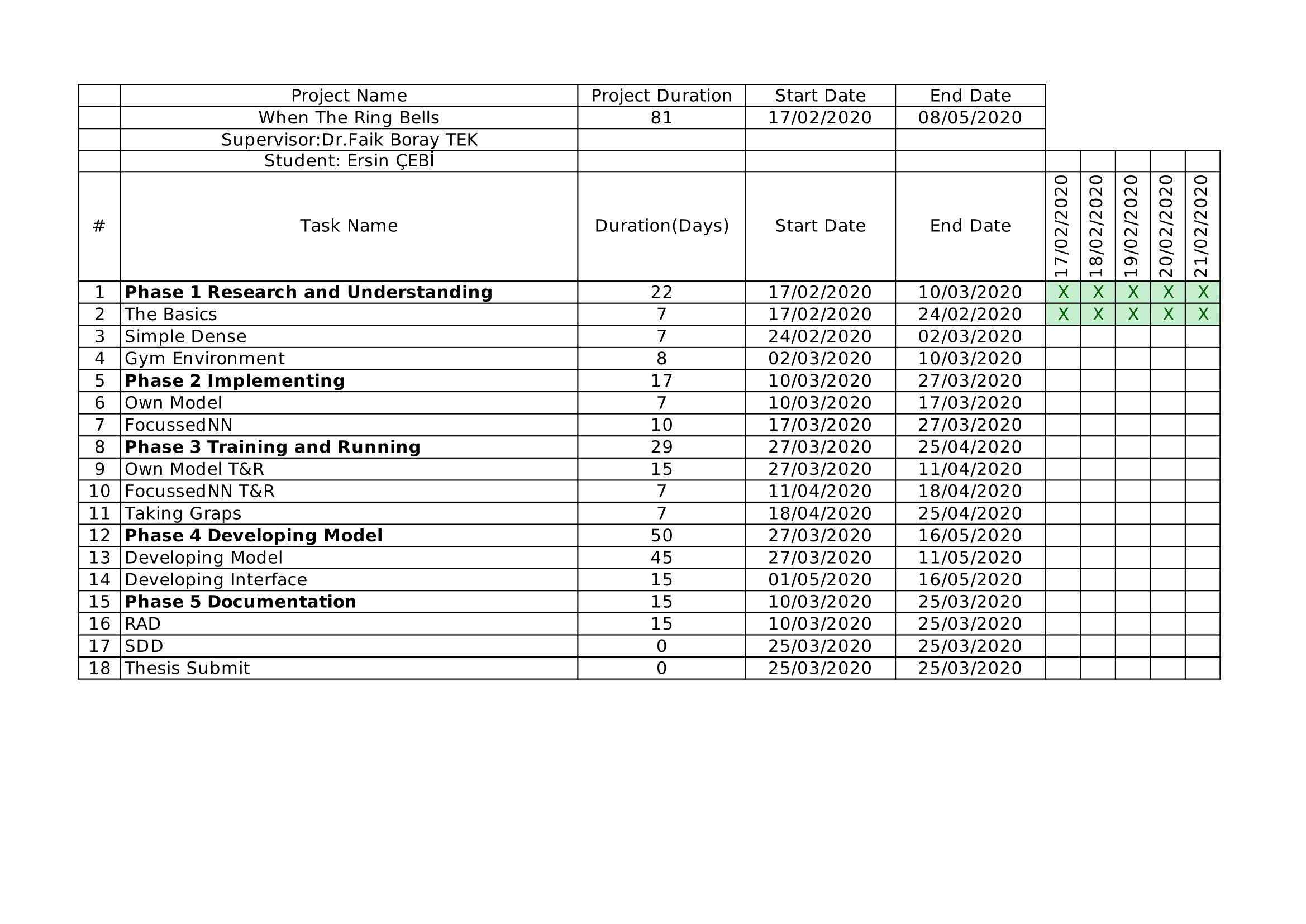
**Legal:** The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and no infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

**Keras Workflow:**





**Gantt Chart**

****

**4. Glossary**

**Simple DENSE**: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected.

**Colab**: Google Colab is a free cloud service and now it supports free GPU! You can; improve your Python programming language coding skills. develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

**Python**: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

**Tensorflow**: TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning faster and easier

**Keras**: Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](https://github.com/tensorflow/tensorflow), [CNTK](https://github.com/Microsoft/cntk), or [Theano](https://github.com/Theano/Theano). It was developed with a focus on enabling fast experimentation.

**Gym Environment:** The gym library is a collection of test problems (environ- ments) that you can use to work out your reinforcement learning algorithms.

**5. References**

1. Boray, F. “An Adaptive Locally Connected Neuron Model: Focusing Neuron.” ArXiv.org, 31 July 2019, <https://arxiv.org/abs/1809.09533>.
2. Géron, Aurélien. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.", 2017.
3. Online documentations of open gym: <https://gym.openai.com/docs/>
4. Online documentations of keras: <https://keras.io>
5. Online documentations of tensorflow: <https://www.tensorflow.org/>
6. Online (2019). [link] URL <https://github.com/btekgit/FocusingNeuron>