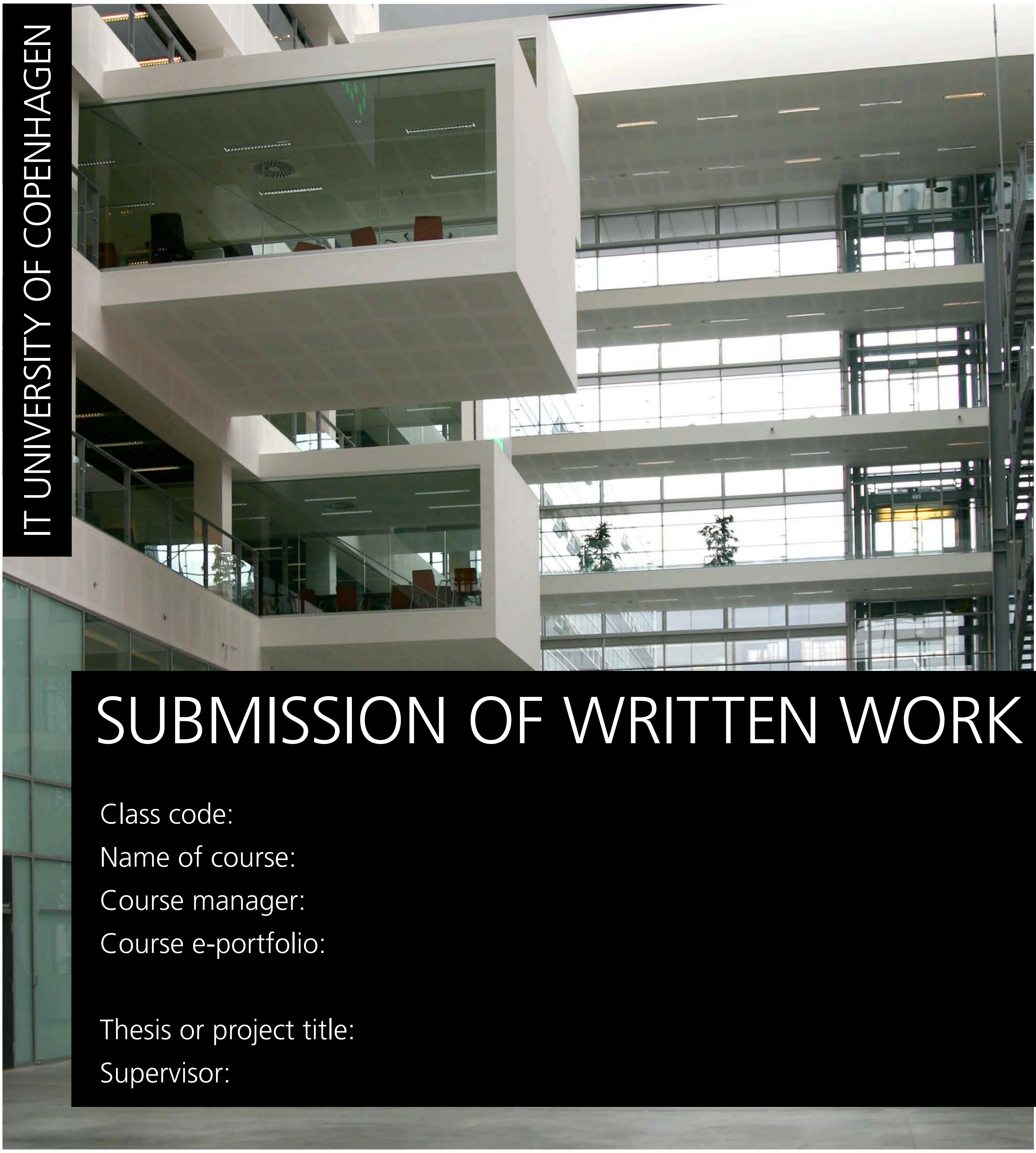

IT UNIVERSITY OF COPENHAGEN

SUBMISSION OF WRITTEN WORK

Class code:
Name of course:
Course manager:
Course e-portfolio:

Thesis or project title:
Supervisor:

Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.		@itu.dk
2.		@itu.dk
3.		@itu.dk
4.		@itu.dk
5.		@itu.dk
6.		@itu.dk
7.		@itu.dk

Multitasking Game Agents using Single-Unit Pattern Generators

Mads Anthony

December 17, 2015

1 Problem statement

1.1 Research Question

What are the results of using single-unit pattern generators for multitasking agents in a video game? And will the results be better than what is already achieved by other modular approaches?

1.2 Description

Most learning based methods tend to focus on optimizing one specific task. The found solution often works great for the given problem, but it might not always generalize very well, when introduced to a new task. I'm interested to work with problems that solve multiple tasks and make something that is good and fast at switching between those tasks. More specifically I want to use evolutionary algorithms to evolve neural networks that are good at multitasking.

1.3 Method

To evolve the multitasking agents I want to to use a similar approach as done in the paper about SUPG [1]. But instead of using it to evolve gaits, I want to use the SUPG as a controller for an agent in a game. The game I'm using will be Ms. Pac-Man, since it is non-deterministic and its framework is used in other research [2]. The idea is to let each frequency be assigned to a certain task, and when the current task changes, the network could then adapt by changing the frequency. These frequencies will have to be controlled by certain triggers, which will be related to its environment (e.g. when a distance to an object reaches a certain threshold). How these triggers is chosen and how exactly the structure is put together, is something I will have to tweak and re-iterate on. Aside from the SUPG approach I will implement some of the same methods in the paper about "Discovering Multimodal Behavior" [2] and compare it with the results from the SUPG approach.

1.4 Project plan

I will be having weekly meetings with my supervisor, Sebastian Risi. And in the figure below is a rough plan for my project.

	February	Marts	April	May
Implement SUPG approach	x			
Implement Modular approach	x			
Re-iterate on approaches		x	x	
Report Writing		x	x	x

2 Literature Review

In this section I will give a review of papers there are relevant for my project.

2.1 Single-Unit Pattern Generators for Quadruped Locomotion [1]

Gregory Morse, Sebastian Risi, Charles R. Snyder & Kenneth O. Stanley

This paper is relevant because I will be using a modified version of their approach to evolve multitasking agents.

In this paper they evolve a controller for a four-legged virtual robot using HYPER-NEAT on a type of neuron called a Single-Unit Pattern Generators (SUPG). The inspiration for the SUPG is build upon the Compositional Pattern Producing Network (CPPN) used within HYPER-NEAT. Since CPPN is good at encoding spatial pattern, the idea for the SUPG is to use time instead of space. This is done using a timer that increases linearly as input to the CPPN along with a trigger that tells when to start the timer. For the 4-legged robot the trigger is getting reset every time a leg touches the ground.

The SUPG is then placed on a plane (called a substrate) and the position for each SUPG is given as input to the CPPN within the SUPG. Since the robot has three motors for each leg (knee, hip in/out, hip forward/back), the different motor types are then spread across the substrate and each motor type is then grouped together with other legs that have the same type. This is done to ensure that each motor type produce similar motion trajectories. The results of using SUPG on the robot are then measured in the distance traveled and compared to two other methods. The comparison shows that SUPG has the longest distance traveled and outperforms the two other methods.

2.2 Discovering Multimodal Behavior in Ms. Pac-Man through Evolution of Modular Neural Networks [2]

Jacob Schrum & Risto Miikkulainen

This paper is relevant because I will be using similar approaches and use those results to hold against my results using SUPG.

In this paper they train nine different modular networks to play Ms. Pac-Man. The nine methods can be grouped into three main approaches.

The first one is called a multitasking network. The idea is to choose how many modules the network should have and how it should switch between the modules. Each module has its own policy output neurons, and it's up to the user to tell the network when it should switch between the policies. This approach is biased, since it's up to the user to choose the number of modules and how to split the tasks. So to avoid how to split the task, another network is used which introduces an extra neuron for each module called a preference neuron.

The idea is to let the preference neuron decide which module to choose. This is done by choosing the module with the preference neuron of highest value. Although this approach removes the human-bias on how to switch between tasks, the user still has to decide on how many modules should be used. So to account for this, a third approach called module mutation is used.

Module mutation starts with a single module and new modules with policy and preference neurons are then later added. There are different ways to add a new module. MM(P) lets the module be connected to a previous module outputs. MM(R) randomly chooses input links used for the policy and preference neurons. MM(D) duplicates the same links used for previously policy neurons, and choose a random link for the new preference neuron.

These nine methods are then evaluated on the game score and their results are compared to previously used methods. The comparison shows that the nine methods all scored better than the previously used methods.

2.3 Dynamic reconfiguration of human brain networks during learning [3]

Danielle S. Bassetta, Nicholas F. Wymbsb, Mason A. Porter, Peter J. Muchae, Jean M. Carlson & Scott T. Grafton

This article can help put things into perspective when studying multitasking agents, since research might indicate that our brain does something similar when learning tasks.

In this article they take a group of people and let them do a learning task, where they have to react swiftly and accurately to visual cues presented as musical notes. While the participants are playing, their brain is then being measured using a MRI scanner. The data from the MRI scanner is then used to construct a functional network that represents each participant's brain.

These functional networks are then evaluated by how modular they are, by calculating a modularity factor Q. They then do the same for randomly generated networks and compare the results. The comparison shows that functional networks have higher values of Q and changes smoothly over smaller scales. The result of this is thereby suggesting that during learning our brain works in a significant modular way.

2.4 Compositional Pattern Producing Networks: A Novel Abstraction of Development [4]

Kenneth O. Stanley

This is the original paper for Compositional Pattern Producing Networks (CPPN) and since I will be using CPPN as part of my method with SUPG, it is relevant to know more about what CPPN is and how it works.

CPPN uses an abstraction that is based on patterns that are typical seen in nature. This include: repetition, repetition with variation, symmetry, imperfect symmetry, elaborated regularity and preservation of regularity. As an example the human body is symmetric along the center of the body and has repetition with variation for the fingers on hands and feet.

With development encoding you use local interaction and temporal unfolding in simulations to mimic the development in nature. However with CPPN the development encoding is abstracted away and CPPN are instead using functions to achieve the patterns mentioned above. The idea is that the phenotype can be viewed as a distribution of points in a multidimensional Cartesian space and that any phenotype produced through a temporal progression (development encoding) is possible to be represented using a functional description. To get symmetry, a Gaussian function or $f(x) = abs(x)$ can be used (since they are mirrored around the y-axis). For repetition, functions like sin(x) or modulus can be used.

These functions can then be put together in composition to get more advanced phenotypes. This is done using CPPN-NEAT which is a modified version of the NEAT algorithm. The reason that NEAT can be modified is that artificial neural networks (ANN) and CPPN structurally are the same.

2.5 References

[1] Gregory Morse, Sebastian Risi, Charles R. Snyder & Kenneth O. Stanley, Single-Unit Pattern Generators for Quadruped Locomotion

http://eplex.cs.ucf.edu/papers/morse_gecco13.pdf

[2] Danielle S. Bassetta, Nicholas F. Wymbsb, Mason A. Porter, Peter J. Muchae, Jean M. Carlson & Scott T. Grafton, Dynamic reconfiguration of human brain networks during learning

http://www.cs.utexas.edu/users/ai-lab/downloadPublication.php?filename=
http://nn.cs.utexas.edu/downloads/papers/schrum.tciaig15.pdf&pubid=127492

[3] Danielle S. Bassetta, Nicholas F. Wymbsb, Mason A. Porter, Peter J. Muchae, Jean M. Carlson & Scott T. Grafton Dynamic reconfiguration of human brain networks during learning

http://www.pnas.org/content/108/18/7641.full

[4] Kenneth O. Stanley, Compositional Pattern Producing Networks: A Novel Abstraction of Development

http://eplex.cs.ucf.edu/papers/stanley_gpem07.pdf