### Task title: Utilization of a Tensorflow neural network python model inside the existing agent based modelling Repast framework

### Definition of the task

The aim of the task is to enable the utilization of a neural network model inside the existing agent based modelling framework (Repast) used by several members of the agricultural policy chair of the institute for food and resource economics. Specifically, a neural network was designed and trained with data generated from the single farm model "Farmdyn" with the aim to substantially increase the computational efficiency of running farm planning scenarios. The trained neural network was also created by a member of the chair. Regarding this contract, the primary use case for the neural network model is to be available inside several technology adoptions focused agent based models created under the PhenoRob excellence cluster. The task is highly technical and specific as it involves deep knowledge in both computer languages and their libraries (Java and Phyton) and artificial intelligence, machine learning and discrete event simulation. The resulting Java API shall be optimized for computational efficiency. The task is fulfilled with the creation of a Java executable jar enabling Java programmers, when implemented, to retrieve predictions by feeding any input parameter specified in the original Phyton model.

### Solution alternative (a): implementing a socket/port connection between Applications

We define sockets, which uniquely identify the endpoint of a communication link between two application ports (Python & Java). A port represents an application process on a TCP/IP host. Hereby, by means of an example (documents attached to this report: Agent java class.text & Iterate java main.text) we have clarified a simple guideline to reveal the path of implementing this solution. We have defined a class Agent. Each agent has two attributes, attr\_1 from type integer and attr\_2 from type double. By means of a public void iteration method beside a public void evolve, in each iteration of the simulation, the attr\_1 will add itself just to one unit. Simultaneously, the attr\_2 should be updated in each iteration of the simulation subject to a defined ANN model embedded in the python application. By defining the additional attribute port for each agent e.g. port "3333" for agent 1 and using the process builder mechanism inside the java program, we target the path way of running our python application execution e.g. "C:/Users/Hamed/anaconda3/python.exe", where the ANN model is saved e.g. "C:/Users/Hamed/Desktop/saved\_model.py".

As in the original agent based model we are confronted with thousand number of agents, it seems to be essential to make use of the Multithreading feature of the Java feature that allows concurrent execution of two or more parts (here ports) of a program for maximum utilization of CPU. We additionally make use of the sleep method, which suspends the running threads for the specified amount of time(milliseconds). This mechanism avoids to allow one agent run the command ahead of the predefined footsteps in application. For example, running Thread.sleep(1000) in the Iterate java main, renders the program to allow first, all agents being connected to the python application and first when all connected, start to update their attr\_2 through the ANN defined in python. As the execution of command for all agents in Iterate java main gets done in a parallel way (multi-threading), the delaying effect of the defined millisecond sleeps is expected to be negligible.

In the python side of our port/socket connection (saved\_model.py), to variables attr\_1 and attr\_2 get received as input and are added according the function addition (num1, num2). A neural network keras model, gets trained in each iteration through inputting attr\_1 and attr\_2 and outputting addition of attr\_1 and attr\_2. After fitting the model, a prediction as the output named ans in python gets generated by the ANN model and will be sent through the sock.send command again to the Java application. How to bind the socket with port 127.0.0.1 within both programs is easily to imitate from the attached files. In the file Socket port java & python.mp4, a video snapshot of implementing the socket/port method between two applications is again illustrated.

### Solution alternative (b): implementing a dockerized post request between Applications

Dockerizing a program enables developers to build executable applications which entail all application source codes with the required operating system libraries and dependencies together to enable users to run a code in any environment. Once you dockerized e.g. the Python code, it could be utilized in any type of environment is needed by means of an API accessible with a url http call in a Docker container. Hereby, for solving the concrete issue of ANN in our project we can utilize the existing image Tesorflow/serving, from docker hub (<https://hub.docker.com/r/tensorflow/serving>) by running the corresponded pulling command. Tensorflow serving hosts our ANN model to make their functions available via API so that the Java application can incorporate AI into its systems. The main docker file docker-compose .yml is attached. The main body of the docker compose command can be seen also in the attached Docker compose run command.JPEG. Alternatively, one can just run the following code in command line:

Docker run -p 8501:8501 --name tfserving\_prediction --mount type=bind,source=C:\Users\Hamed\Desktop\,target=/models/saved\_model -e MODEL\_NAME=saved\_model -t tensorflow/serving

The above code mounts the file in directory C:\Users\Hamed\Desktop\ to /models/saved\_model in the in the container named tfserving\_prediction. As a result, the local port 8501 becomes available to send post requests from the java, python or any other applications (a code example of the application of the generated url is illustrated for python through the attached file Python example sending post request.txt as well as for java through the attached file Java example sending post request.txt). How to implement that in the program is revealed in the attached Docker tensorflow prediction.mp4 file.

The saved ANN model (saved\_model.pb) can be swapped any time by new one as long as the updated versions are put along the same directory path C:\Users\Hamed\Desktop\. Hereby, it is crucial to specifying the correct number for the leaf directory. The serving program, then automatically identifies the updates along the path and uses the new model. A video illustration is demonstrated in Docker uploading the served model by new saved\_model.mp4.