N2Sky - A Neural Network Problem Solving Environment Fostering Virtual Resources

Andrii Fedorenko
University of Vienna
Vienna, Austria
andriifedorenko@gmail.com

Aliaksandr Adamenko
University of Vienna
Vienna, Austria
alexadamenko@gmail.com

Erich Schikuta
University of Vienna
Vienna, Austria
erich.schikuta@univie.ac.at

***Index Terms*—Problem Solving Environment, Cloudcomputing, Containers, Microservices, Neural Networks**

***Abstract*—This paper presents the novel N2Sky system which provides a virtual collaboration platform to the computational intelligence community. It realizes the Neural Networks as a Service paradigm allowing to share and exchange neural network knowledge resources on a worldwide basis by a transparent environment fostering state-of-art hardware and software infrastructures. The system is motivated by the goal to deliver an intuitive tool for different stakeholders of the neural network community, as arbitrary users, looking for a packaged neural network solution to a given problem, neural network engineers, creating and training their own neural network object based on available paradigms, and experienced neural network contributors, developing and implementing their own neural network type and sharing it with the community.**

To meet these targets N2Sky provides an intuitive user interface which embodies latest web technology to make the user's interaction as simple and efficient as possible. Fostering cloud resources N2Sky is based on container-based virtualization technology to provide higher flexibility, portability, dynamic orchestration, and performance. We give the motivation and design goals for the new system, describe its architectural layout and technical specifics, and present use cases for the different user types.

I. Introduction

A virtual organization/community platform [3] is a medium for human beings, organizations or enterprises to share different types of geographically distributed information (domain specific knowledge) or computational infrastructure (computing nodes, storage systems, databases, libraries and special purpose scientific instruments). This enables more effective and seamless collaboration of scattered communities, both commercial and scientific, enabling large-scale applications and transparent access to high-end resources from the desktop.

The driving stimulus for the development of science is the exchange of information and resources between researchers. As the fields of computational intelligence and machine learning mature, there is a growing need to provide researchers with the ability to exchange information, share resources, discuss problems and new directions, and learn about other's work. The limitations of traditional scientific communication inspired to create virtual communities to gather research, education, and application-oriented resources. The goal of the community is to create a place where scientists, students, and the general public can work together despite any of their geographic limitations. Anyone who is interested can share research, obtain resources, or simply learn more about computational intelligence.

However, in the domain of neural network research, we see this target only insufficiently met until now. In our preceding research, we designed and developed N2Sky [15], a virtual platform aiming for the computational intelligence (CI) community, which enables access to neural network knowledge and fosters virtualized computing resources. Though, due to design decisions then, this original system did not allow to follow the ongoing shift in cloud computing easily. Also, the focus on one classical programming paradigm made it difficult to provide the experience of today's user interface technology.

Therefore, in this paper, we present the new N2Sky system, which realizes the *Neural Networks as a Service* paradigm allowing to share and exchange neural network knowledge resources on a worldwide basis by a transparent environment fostering state-of-art hardware and software infrastructures. Hereby provides N2Sky a neural network solution stack with specific capabilities to its different stakeholders. We focus on three main groups of users with different motivations using N2Sky:

- The arbitrary *Neural Network User* is searching for packaged neural network solutions for given problem domains and an execution framework providing computational resources for analysis of the user's provided data representing the problem.
- The *Neural Network Engineer* shows profound neural network knowledge and is looking for a comfortable simulation environment for artificial neural networks. Here a development stack is needed which allows creating new neural network objects based on available paradigms and gives access to easy to use powerful computational resources for network training. Further capable data management mechanism has to be available for network and training data storage.
- The *Neural Network Contributor* represents the neural network scientist who envisions new neural network paradigms and aims for sharing findings with the community. This goal asks for mechanisms, which allow to code, deploy and analyse the new network paradigm and to describe its semantics by a domain specific standardized language for easy distribution in the community.

The structure of the paper, is as follows: In section II the state of the art of neural network simulation environments and the baseline research is presented. In section III we identify the shortcomings of the original N2Sky system, derive our new technical design decisions and present the new micro-service and container-based architecture. The new user interface is introduced in section IV, which follows the responsive design paradigm and fosters latest web technologies. Use cases for the different user types are presented in section V, which give an impression of the user's experience working with the new system. Finally, the paper concludes our findings and presents our plans for future work.

II. Related Work and Baseline Research

The driving stimulus for development in the computational science domain is the exchange of knowledge and resources between researchers. This principle is just as valid for any other research community too.

The UK e-Science initiative [2] describes several goals to be reached by fostering new stimulating techniques:

- Enabling more effective and seamless collaboration of dispersed communities, both scientific and commercial.
- enable large-scale applications comprising of thousands of computers, large-scale pipelines etc.
- Transparent access to high-end resources from the desktop.
- Provide a uniform look & feel to a wide range of resources
- Location independence of computational resources as well as data.

However, these targets are not reached in the computational intelligence community until now. As an example, we examine the situation of neural network simulation. Over the last decades, a very large number of artificial neural network simulation environments has been developed which aim to mimic the behaviour of biological neural networks [11]. It started with systems which were developed for specific network families, as Aspirin/MIGRAINES [7], SOM-PAK [6]. Some systems aimed for a more comprehensive environment, as SNNS [18]. New technology shifts enabled new concepts, as distributed cooperative environments over the Internet, as NeuroWeb [12]. With the advance of virtual resources by Grid and Cloud computing new collaborative environments motivated the authors of this paper to aim for an "everything about sharing" approach leading the way towards virtual collaborative organisations, as N2Grid [16] and N2Cloud [5].

However, all these systems, reaching from programming language extensions over proprietary stand alone systems to distributed platforms, share the same common problems:

- *Complex tool*, which mostly do not present an intuitive interface to the user.
- *Proprietary system* with missing interconnection and data exchange to other software systems.
- *Lacking* provisioning of arbitrary *computing resources*, as CPUs, disks, network, on demand.

These problems lead to the situation that quite a number of simulation systems exist, but which are rarely used. Hence, scientists invent the wheel over and over again and develop their own neural network systems for their specific needs. We believe that this situation is one of the reasons for an obstructed open information and data exchange within the scientific community.

A promising project, totally in line with our motivation, was the CIML (Computational Intelligence and Machine Learning) community [19]. The goal of CIML was to create an online virtual scientific community wherein anyone interested in computational intelligence and machine learning can share research, obtain resources, or simply learn more. Sorry to say, but CIML failed. One reason was that CIML targeted a too huge and dispersed community and offered too many and different resources. Due to lack of automated guidance of the system it was difficult for the user "to find his specific needle in the haystack".

Having this situation in mind we realized N2Sky [15], with a clear focus on neural networks aiming for intuitive user guidance and transparent resource access. N2Sky was designed as an artificial neural network provisioning environment facilitating the users to create, train, evaluate neural networks fostering different types of computing resources. The system is cloud based in order to allow for a growing virtual user community. N2Sky supports experienced users to easily run their simulations by accessing data related neural network objects. Moreover, N2Sky provides a facility to end users to solve their problems by using predefined objects and paradigms. For the purpose of thin clients a simple Web browser, which can execute on a PC or a smart phone, can be used to access the front-end, the N2Sky (Mobile) Web Portal. N2Sky aroused strong interest even beyond the computational intelligence community[1].

The pillar of our envisioned system is ViNNSL, the Vienna Neural Network Specification Language [1]. It is key for easy sharing of resources between the paradigm provider and the customers. ViNNSL is an XML-based domain specific language providing mechanisms to specify neural network objects in a standardized way by attributing them with semantic information. Originally it was developed as communication framework to support service-oriented architecture based neural network environments. Even more, ViNNSL is capable of describing the static structure, the training and execution phase of neural network objects in a distributed infrastructure, as grids and clouds. Its last extension [13] supports semantic information too, describing the usage scenario of network objects for a given problem domain.

The following short example illustrates the use of ViNNSL. The listing 1 defines a 2-5-1 backpropagation network for the well-known XOR problem. Besides the description of the structure of the network, also the application domain for using this network is specified. This is the basis for smart searching of feasible neural networks for given problems.

[1]http://cacm.acm.org/news/171642-neural-nets-now-available-in-the-Cloud/

```
<definition xmlns="http://www.pri.univie.ac.at/../vinnsl">
  <identifier>7</identifier>
  <problemdomain>
    <backpropagation/>
    <classifier/>
      <XOR-net/>
    </classifier/>
  </problemdomain>
  <executionenvironment>
  ...
  </executionenvironment>
  <structure>
    <input>
      <id>Input1</id>
      <dimension>2</dimension>
      <size>1</size>
    </input>
    <hidden>
      <id>Hidden1</id>
      <dimension>5</dimension>
      <size>1</size>
    </hidden>
    <output>
      <id>Output1</id>
      <dimension>1</dimension>
      <size>1</size>
    </output>
    <connections>
      ...
    </connections>
  </structure>
  <parameters>
    <valueparameter>
      <name>lrate</name>
      <label>Learning Rate</label>
      <value>0.3</value>
    </valueparameter>
    ...
  </parameters>
</definition>
```

Listing 1. ViNNSL: XOR-Backpropagation Neural Network Definition

III. A Micro-Kernel Container-based System Architecture

Service migration to the cloud in the recent years is a clear trend in the IT-industry, and it is not surprising that some of the companies are trying to leverage existing cloud infrastructure to bring deep and machine learning solutions to the market. IBM SPSS, Amazon AWS, Google Cloud AI services are meant to provide software tools to the different stakeholders, but they are primarily focusing on the business users, which primary goal is to use applied statistics techniques to fulfill existing business needs. Services are split into components, each of which hides internal implementation of the solution and provides an interface for a specific task: speech recognition, text analysis, etc. Variety of pricing models, a necessity to use other software products of the company, proprietary code and low level of scalability and extensibility are making these products less attractive for the scientific and academic community. Vendor lock-in and inability to control execution and development process are making such software systems and services non-flexible and nontransparent, leading to the scenario where each service provider tends to aggregate and try to sell as many proprietary services as possible.

We consider these aspects as limiting factors for the development of the framework. We believe that such properties as an ability to share knowledge and resources, re-usage of the existing solutions and collaborative work are key features which should be put as a foundation of distributed neural network execution platform.

Keeping that in mind, we tried to re-design the existing solution N2Sky, where new architecture and usability solutions are reflecting core functional ideas of cloud-based execution environments.

N2Sky was designed to provide natural support for cloud deployment with distributed computational resources. However, N2Sky was intended as a single monolithic application written in Java, which is not well aligned with cloud infrastructure and virtual organization concepts – high redeployment costs, low level of extensibility, the necessity to understand the application code to experiment with neural networks objects and poor computational distribution capacities.

That is why we have decided to put microservices [10] approach as a foundation of the new architectural design. The main idea of the microservices approach is to decompose the software system functionality into separate components, which are responsible for specific tasks. Each component runs in its own isolated environment which is achieved by using Linux containers (LXC) [8] and Docker [9]. All the components are interacting with each other through the API, so they are not aware of any internal implementation details. It allows to have a scalable architecture which can be easily extended or modified by introducing new modules or changing the existing ones – the only thing a developer should care about is proper communication between components.

As the previous system was developed as a Java application, it was entirely language dependent – to apply any changes to the neural network execution engine or introduce a new neural network object or paradigm, a developer was restricted to the Java language. The new design allows developers and researchers to introduce both system modules and neural networks implementations in any language of their choice: the only restriction is to provide API to communicate with. If a user makes small adjustments to the existing Docker image, adds train and test endpoints, the neural network is deployed and is ready to be used.

The microservice architecture brings additional advantage – load balancing, which comes from the symbiosis of shared cloud resources and microservices. If there is a (high) computational demand, scalability is achieved by merely running new a container – which is a secure and robust solution that was not possible in the monolithic approach as spawning a new instance with a whole application is a noticeable overhead.

Container quantity can grow very fast; it becomes clear that manual maintenance of dozens, or even millions of Docker containers can be a tough task, especially considering a cloud environment. For that reason, we have decided to use container orchestration software as Cloudify [17]. This tool is providing high-level interfaces to communicate with cloud-based platforms and control deployment and execution of containers.

Putting all these technologies and design approaches together, we receive a robust and efficient cloud-based environment, which can be easily scaled both horizontally and vertically.

The current working version of the new architecture is presented in figure 1.

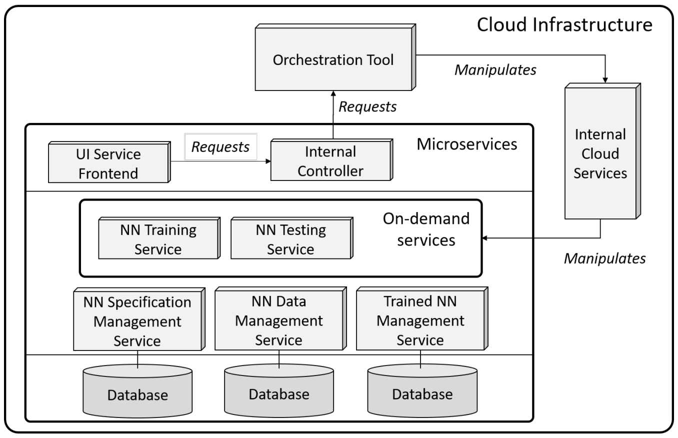


Fig. 1. New N2Sky architechture

Communication between components of the system is performed as follows:

- The internal controller is responsible for handling user requests, for controlling them and either submitting them to the orchestration tool to spawn new container or transfer it to an the existing instance of the requested resource,
- Orchestration tool is responsible for interacting with cloud infrastructure and spawning new instances of neural network objects and internal services,
- The on-demand services are a pool of neural network object instances, which can be used for training and testing using the provided data,
- Neural network data archive is a separate service on top of the database. It will provide a specific API to add any external data source,
- Neural specification management service is a component which is responsible for managing existing neural network paradigms, used for instantiation of neural network objects, and
- Trained NN Management Service is responsible for saving and retrieving trained neural network object, in case a user wants to use pre-trained neural networks.

IV. Novel User Centered Interface

The user-centered design is a fundamental requirement for N2Sky. Looking back on past experiences with the application, the real capabilities and needs of users were identified. N2Sky was moved from a complex expert system to an easily understandable application. Every interested user without deep knowledge in the neural network field can freely use N2Sky. The goal was to save and gain the current functionality of the application and decrease the visual complexity of it.

A. Frontend and services

N2Sky today is a cross-platform handy application with responsive design. The frontend is written on the ReactJS framework and it is convertible to the React-Native framework. The application is accessible from desktop computers, as well as from mobile devices or other devices with any operational system. Furthermore, the backend has the microservices architecture to support scalability. Each of the microservices is developed on NodeJS server, which implies efficiency and lightweight. This architecture enables its users to freely and easily work with the application without interruptions or waiting until it is completely loaded.

B. Modular design

The central concept of the application is to support the Software as a Service (SaaS) and Platform as a Service (PaaS) distributions. N2Sky consists of two modules: administration module and main application module.

a) Administration module: The administration module allows the system administrator to control the environment. The module supports OpenStack and Cloudify monitoring. Managing is possible through the application dashboard. It also contains custom monitoring and an alerting management system, which can be installed on any server within the N2Sky user interface. The administration module implements PaaS. It is fully configurable and wrapped into the open source project to make the module accessible to third-party applications.

b) Main application module: The main application module is the central neural network user module of N2Sky. Within this module, users can use, train and test existing neural networks. It is possible to reuse the neural network paradigms and create own neural networks. N2Sky allows to deploy own networks and to store data in the cloud. Module services are supporting the SaaS distribution. Experts can use an application directly through the N2Sky API, or they can integrate N2Sky services into their own application.

C. User roles

In order to make the N2Sky user interface intuitive for arbitrary users as well as advanced professional users, it was decided to separate the user roles. Every user role has its own way of interaction with the application:

- The arbitrary neural network user. Such a user has no necessity to have deep knowledge of the neural network field or know any programming language. The main goal of the arbitrary user is to find neural network solvers to given problems and to study found neural networks within N2Sky. The contributor has access only to his own dashboard and public available resources on the main application module. He can perform semantic search for available neural network paradigms and use them.
- The neural network engineer. He has the same options as the arbitrary user but he can also create and train new running neural network instances and test them. This user can share his trained neural network by making it public.

- The neural network developer is an expert user, which has enough knowledge and experience to create his neural network type. This user can create neural network paradigms using the ViNNSL schema and publish them on N2Sky. He can deploy neural networks on the N2Sky environment as well as on his environment by providing training and testing endpoints. The goal of the developer is the study how his networks will behave with different network structures, input parameters and training data that is provided by other users.
- The system administrator is a user who has full access to all application including environment management, monitoring and alerting features. The administrator can manage OpenStack and Cloudify instances. He can also monitor any N2Sky user to observe the application from a shadowed user perspective.The administrator has access to all dashboards in every module.

V. Use Cases

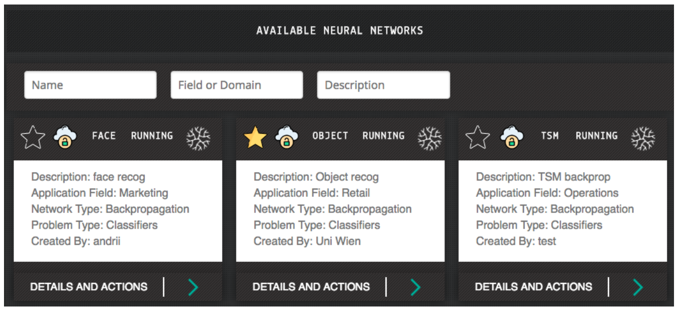
Workflows of solving problems as a user, engineer or contributor differ. A user needs an easy step-by-step workflow, the engineer just wants to perform procedures quickly without any complex distractions. There are only a few pre-requirements, which will remain the same: authentification within N2Sky and the creation of a first project from the N2Sky dashboard, which describes the problem field.

A. *Use Case 1: The Arbitrary Neural Network User*

After creating the project, users have two possibilities to find already available neural networks which correspond to their needs: on the one hand they can search for a neural network using their identifiers or user-defined tags, and on the other hand, they can perform a semantic search by using a natural language approach. The latter method, which we termed as N2Query, provides the semantic discovery of N2Sky services through a human-centered querying mechanism [14]. N2Query allows N2Sky users to specify their problem statement as natural language queries. In response to the natural language queries, it delivers a list of ranked neural network services to the user as a solution to their stated problem. The search algorithm of N2Query is based on the semantic description of neural network objects by ViNNSL and the mapping of ontologies referring to problem and solution domains.

The following simple workflow corresponds to the arbitrary user:

1) **Create a new project.** After the first login into the N2Sky platform, the system will propose the user to create a new project. The project on the N2Sky platform is a collection of neural networks and training models, which help solve the user's problem.
   N2Sky contains multiple solutions to common problems. During the creation of a new project, a user will fill up fields, which will be used for semantic search among available neural networks and models repository. As soon as the project is created, the platform will suggest some existing neural networks and trained models corresponding to the users needs. The user can choose some of the proposed solutions as well as add existing neural networks and trained models manually.



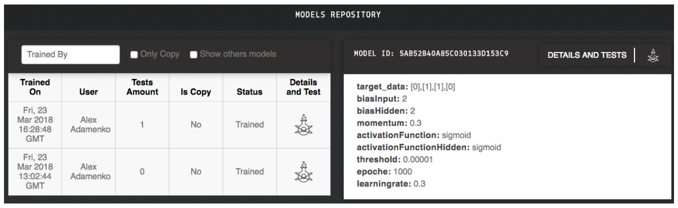
Description: face recog
Application Field: Marketing
Network Type: Backpropagation
Problem Type: Classifiers
Created By: andrii

Description: Object recog
Application Field: Retail
Network Type: Backpropagation
Problem Type: Classifiers
Created By: Uni Wien

Description: TSM backprop
Application Field: Operations
Network Type: Backpropagation
Problem Type: Classifiers
Created By: test

Fig. 2. N2Sky neural network repository

2) **Add neural networks into the project.** N2Sky has a neural network repository, which stores different solutions for typical problems, which can be reused easily. Users navigate to the neural network repository view, as shown in figure 2, and copy the requested neural networks to their project space. In order to find a suitable neural network for his needs, the arbitrary user can perform the semantic search, which is based on the N2Query method.
   Before adding the advised neural networks proposed by the semantic search, the user can inspect the neural network first. The neural network owner can publish some training data in order to demonstrate how his neural network behaves. The arbitrary user can also observe the popularity of the neural network, namely how many users performed training of the neural network and if any significant training and evaluation results were published.
   As soon as the user decides that he will add the particular neural network to his project, he can click on copy indicator, which has a star icon form. If the star indicator is grey, users can copy it to their project and perform training and testing on the copied networks.
   Directly after copying the neural network to their own projects, it will be available to the arbitrary user. If the neural network owner modifies the neural network which was added to the project of the arbitrary user, the performed training can be repeated. In case that the neural network owner makes the network private, this network will be hidden from arbitrary user projects. The arbitrary user will always be notified about any occurred changes.
3) **Perform the neural network training.** After adding the neural network into its own project, the user can perform training operations. The arbitrary user does not have to know the technical jargon in order to do it. When the user will be on the training screen, the input parameters will be pre-filled with default values as well as default training data nearby attached. With this approach, a user does not need deep knowledge in the neural network



MODEL ID: 5AB52B40A85C030133D153C9

Trained On	User	Tests Amount	Is Copy	Status	Details and Test
Fri, 23 Mar 2018 16:28:48 GMT	Alex Adamenko	1	No	Trained	
Fri, 23 Mar 2018 13:02:44 GMT	Alex Adamenko	0	No	Trained	

target_data: [0],[1],[1],[0]
biasInput: 2
biasHidden: 2
momentum: 0.3
activationFunction: sigmoid
activationFunctionHidden: sigmoid
threshold: 0.00001
epoche: 1000
learningrate: 0.3

Fig. 3. Trained Neural Network models repository

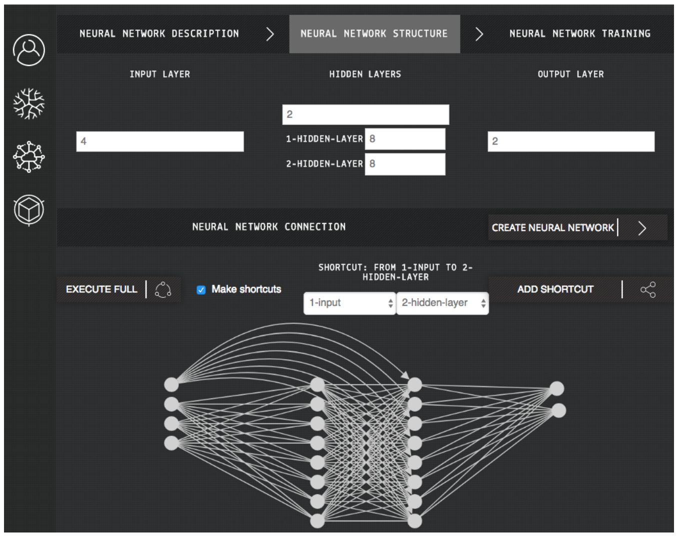
field. The N2Sky platform guides the user in order to teach him how the training process works.

4) **Add trained model into the project.** The model repository is available to the arbitrary user. Also,a semantic search for trained models can be performed the same way as previously done for the neural network repository.The user can preview the trained models and observe the previous testing results. If the model corresponds to the users needs, he can add the model to his own project.
5) **Evaluate the trained models.** The user can perform testing against models trained by himself, as well as copied ones from model repositories. The default testing data will be proposed in order to teach the user how the evaluation process works. The user can modify testing data on demand.

B. *Use Case 2: The Neural Network Engineer*

To create a neural network from the paradigm, engineer users have to choose a paradigm and add it to their project. Following steps have to be completed:

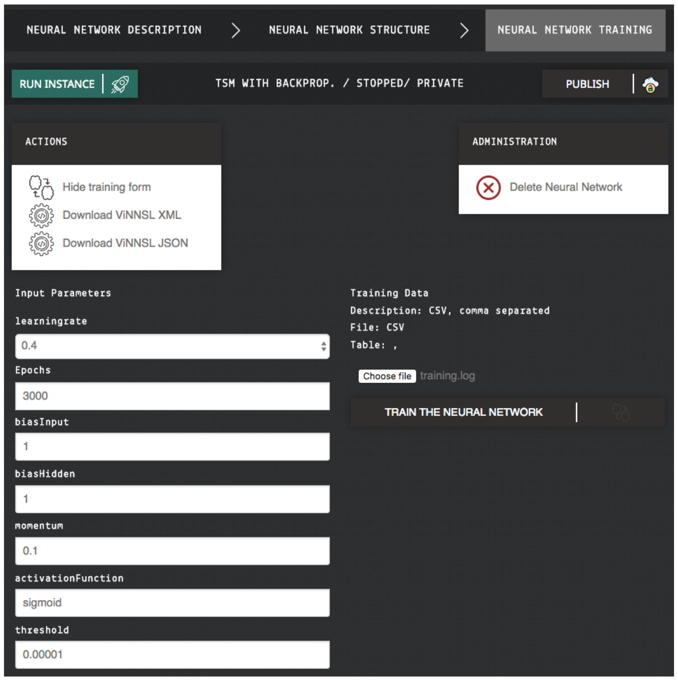
1) **Fill out the Neural Network Description form.** Among the available paradigms, the user needs to select one. Every neural network paradigm is defined by the ViNNSL schema. The mandatory fields are name, description, propagation type, learning type, application field and problem type. All this information will be used for semantic searching by other users.
2) **Define the network structure.** After submitting the Neural Network Description form, the user will be redirected to neural network structure view as shown in figure 4. The structure definition can be customized. It is obligatory to specify the input, hidden and output layers with at least one node in each layer. The number of hidden layers as well as the number of nodes in each layer is unlimited. After defining the network structure, the engineer user has to set connections between nodes. The user can execute full connections as well as add shortcut connections to any node. The users do not need to have any concerns about the correctness of the neural network structure. The application will take care of any errors and mistakes, which users can make.



INPUT LAYER: 4
HIDDEN LAYERS: 2
1-HIDDEN-LAYER: 8
2-HIDDEN-LAYER: 8
OUTPUT LAYER: 2
Make shortcuts
SHORTCUT: FROM 1-INPUT TO 2-HIDDEN-LAYER
1-input
2-hidden-layer

Fig. 4. Neural network structure definition

3) **Train the newly created neural network and evaluate its model.** After creating a neural network, the users will be redirected to the training view which is shown in figure 5. From here, they can run the neural network instance on N2Sky and publish it to make this neural network available for the rest of the users. In case users want to study the neural network in detail, they can also download it in ViNNSL format. Most importantly, users can perform training. Since the neural network was created from the paradigm, default parameters for training are set by default, which guides users through further steps.



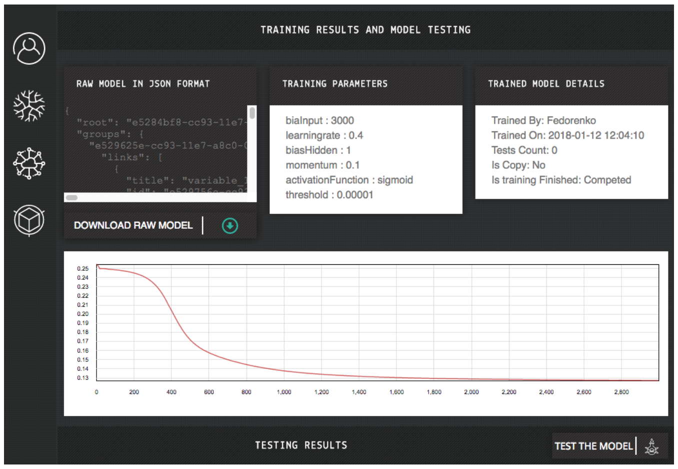
TSM WITH BACKPROP. / STOPPED/ PRIVATE

Input Parameters
learningrate: 0.4
Epochs: 3000
biasInput: 1
biasHidden: 1
momentum: 0.1
activationFunction: sigmoid
threshold: 0.00001

Training Data
Description: CSV, comma separated
File: CSV
Table: ,
training.log

Fig. 5. Neural network training view

After training is completed, users can perform testing, see figure 6. The visual representation of evaluation will be shown. The learning curve is constantly updated during neural network training. The x-Axis shows epochs, and the y-Axis shows costs.



```
{
  "root": "e5284bf8-cc93-11e7-
  "groups": {
    "e529625e-cc93-11e7-a8c0-
      "links": [
        {
          "title": "variable_
```

bialnput : 3000
learningrate : 0.4
biasHidden : 1
momentum : 0.1
activationFunction : sigmoid
threshold : 0.00001

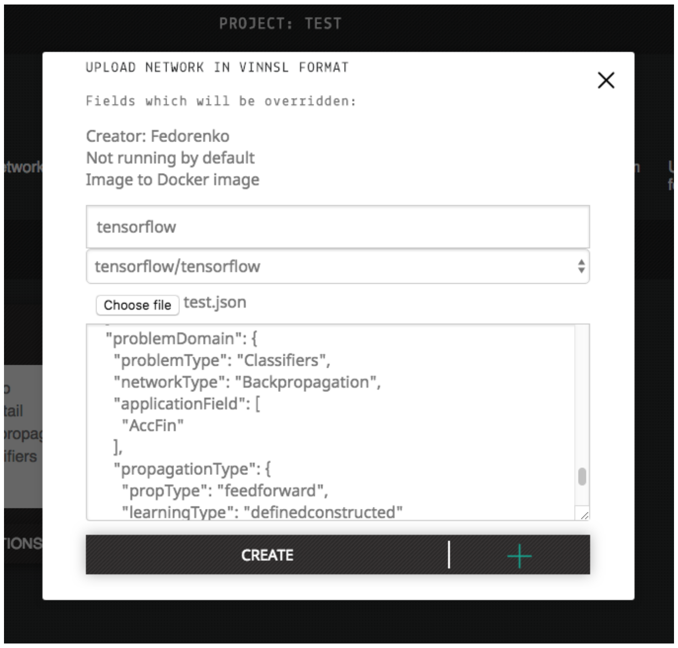
Trained By: Fedorenko
Trained On: 2018-01-12 12:04:10
Tests Count: 0
Is Copy: No
Is training Finished: Competed

Fig. 6. Neural network evaluation process

4) **Publish and share the results .** Finally, it is possible to publish and share the trained model so that other users can use it with their own data.

C. Use Case 3: The Neural Network Contributor

Neural network contributors are expert users. They can download the ViNNSL schema template from N2Sky platform, fill it up and customize their envisioned neural network as shown in figure 7. After creating a project, users can just upload their ViNNSL formatted paradigm and deploy it.



Fields which will be overridden:

Creator: Fedorenko
Not running by default
Image to Docker image

tensorflow
tensorflow/tensorflow
test.json

```
"problemDomain": {
  "problemType": "Classifiers",
  "networkType": "Backpropagation",
  "applicationField": [
    "AccFin"
  ],
  "propagationType": {
    "propType": "feedforward",
    "learningType": "definedconstructed"
```

Fig. 7. Deploying own neural network on N2Sky

The following steps have to be completed in order to publish an own neural network paradigm in the N2Sky platform:

1) **Contribute own neural network paradigm.** Since the contributor user is an expert, he will work with a specialised user interface which incorporates technical jargon. Predefined requirements for publishing a users paradigm are the neural network image and the description of this network in ViNNSL format.
The neural network image has to be a Docker image, which is publicly available in DockerHub repository. N2Sky uses provided Docker images for deploying it into its own cloud. Every image requires the following endpoints:
   - "/train" - endpoint for training a neural network, which provides callback function in order to get the progress results of the training.
   - "/test" - endpoint for evaluation the network, which accepts trained neural network model with requested testing data and responses with evaluating results

   The description has to be either in ViNNSL XML or ViNNSL JSON format and contain the following predefined components:
   - General metadata about neural network paradigm.
   - The default structure of neural network and its connections
   - Possible input parameters with default values
   - Type and format of accepted training data

   ViNNSL template schema allows users to specify which parameters can be edited as well as possible values of particular parameters. The contributor user is free to decide how other people could use his paradigm.
   As soon as all pre-requirements are fulfilled, the neural network is available and visible only in the private dashboard of contributor user.
2) **Deploy and publish own neural network paradigm.** N2Sky platform gives to contributor users more flexibility. There are two ways how the instance can be run with his own neural network paradigm:
   - Deploy in the N2Sky cloud. For every image with the neural network paradigm, the independent instance in N2Sky cloud will be created. The user provides the publicly available Docker image which will be pulled and deployed in the cloud.
     Since the contributor user wants to know how his neural network will behave during training and evaluation, special monitoring information will be provided. The monitoring charts are available on N2Sky portal as well as via API service. After gathering the monitoring data, the user can adjust the virtual machine environment configuration. Considering that the instance of our neural network is located in the cloud, any adjustment in configuration will be automatically applied without restarting of the instance.

- Deploy in an external cloud. If the contributor user decides to keep his own neural network paradigm private, he can publish it on any third party cloud like Amazon Web Services (AWS) or Google Cloud Platform. The user has to provide correct endpoints for training the neural network and testing the trained model. After N2Sky platform validates the reachability of the neural network instance, the contributor user can operate the instance directly from the user interface.
  All functionality, except the instance monitoring, is available. It is also possible to adjust environment configuration because N2Sky can not operate instances in the external cloud. The neural network training and evaluation on trained models can take longer because of the latency between the N2Sky platform and the external cloud.

3) **Analyse the training and evaluating results of the other users.** One of the main purposes of a contributor user is to study the correctness and the behaviour of the provided neural network paradigm. In order to gather this information, the neural network paradigm has to be published on N2Sky platform. As soon as the paradigm becomes available, other users like consumer and neural network engineer ones can start working on it.
   The contributor user is the owner of the neural network instance. Besides monitoring data of the instance environment, the user can observe following information:
   - Observe the training models of the other users against his neural network. The instance owner can audit the training processes of other users, asset some deviations and examine the behaviour of the neural network.
     After gathering the training data of other users, the contributor user can modify his neural network on demand. In this case, the new instance of the neural network will be created and all users who participated in the training of previous versions of this neural network will be notified. The instance owner can retrain his modified neural network with previous users train data. The contributor can perform modification of the neural network and retrain operations multiple times until the training results are satisfying.
   - Observe the evaluation process of the trained models from other users. The contributor user can study the testing results of other users trained models, which were trained against his neural network. The user can distinguish the trained models that offer him the best results.

VI. Summary and Future Research

We presented the novel virtual community platform N2Sky, which allows to share and exchange neural network knowledge and computing resources on a worldwide basis by a transparent user-friendly environment. To cope with the ongoing technology shift N2Sky is fostering cloud container technology, which aims for increased extensibility, portability, dynamic orchestration and performance .

In the future, we will extend the focus of N2Sky from the neural network domain to arbitrary machine learning. Hereby a new specification of ViNNSL is under development which aims for compatibility with PMML (Predictive Model Markup Language) [4].

References

[1] Peter Paul Beran, Elisabeth Vinek, Erich Schikuta, and Thomas Weishaupl. Vinnsl-the vienna neural network specification language. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1872–1879. IEEE, 2008.

[2] UK e Science. Uk e-science programme. [online], http://www.escience-grid.org.uk, last visited January 2018, 2016.

[3] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[4] Alex Guazzelli, Michael Zeller, Wen-Ching Lin, Graham Williams, et al. Pmml: An open standard for sharing models. *The R Journal*, 1(1):60–65, 2009.

[5] Altaf Ahmad Huqqani, Xin Li, Peter Paul Beran, and Erich Schikuta. N2cloud: Cloud based neural network simulation application. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–5. IEEE, 2010.

[6] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, and Jorma Laaksonen. Som pak: The self-organizing map program package. *Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science*, 1996.

[7] Russell R Leighton and A Wieland. The aspirin/migraines software tools, user's manual. *Technical Report MP-91W00050*, 1991.

[8] Canonical Ltd. Infrastructure for container projects. [online], https://linuxcontainers.org/, last visited January 2018, 2016.

[9] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

[10] Sam Newman. *Building microservices: designing fine-grained systems.* " O'Reilly Media, Inc.", 2015.

[11] Alberto Prieto, Beatriz Prieto, Eva Martinez Ortigosa, Eduardo Ros, Francisco Pelayo, Julio Ortega, and Ignacio Rojas. Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, 214:242–268, 2016.

[12] Erich Schikuta. Neuroweb: an internet-based neural network simulator. In *Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. 14th IEEE International Conference on*, pages 407–412. IEEE, 2002.

[13] Erich Schikuta, Altaf Huqqani, and Thomas Kopica. Semantic extensions to the vienna neural network specification language. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[14] Erich Schikuta, Abdelkader Magdy, and A Baith Mohamed. A framework for ontology based management of neural network as a service. In *International Conference on Neural Information Processing*, pages 236–243. Springer, 2016.

[15] Erich Schikuta and Erwin Mann. N2skyneural networks as services in the clouds. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[16] Erich Schikuta and Thomas Weishaupl. N2grid: neural networks in the grid. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 1409–1414. IEEE, 2004.

[17] GigaSpaces Technologies. Clodify. [online], http://docs.getcloudify.org/3.4.1/intro/what-is-cloudify/, last visited January 2018, 2017.

[18] Andreas Zell, Niels Mache, Ralf Huebner, Günter Mamier, Michael Vogt, Michael Schmalzl, and Kai-Uwe Herrmann. Snns (stuttgart neural network simulator). In *Neural Network Simulation Environments*, pages 165–186. Springer, 1994.

[19] Jacek M Zurada, Maciej A Mazurowski, Rommohan Ragade, Artur Abdullin, Janusz Wojtudiak, and James Gentle. Building virtual community in computational intelligence and machine learning [research frontier]. *IEEE Computational Intelligence Magazine*, 4(1):43–54, 2009.