Decision Support System for Neural Network R&D

Rok Tavčar
and Jože Dedič
Cosylab, d.d.
Ljubljana, Slovenia
Email: rok.tavcar@cosylab.com,
joze.dedic@cosylab.com

Andrej Žemva
Faculty of Electrical Engineering
University of Ljubljana
Ljubljana, Slovenia.
Email: a.zemva@ieee.org

Drago Bokal
Faculty of Natural Sciences and Mathematics
University of Maribor
Maribor, Slovenia
Email: drago.bokal@uni-mb.si

***Abstract*— One of the reasons that keep Neural Networks (NNs), which are advanced computational methods (ACM) of great potential, from coming into broader practical use, is the lack of systematic method in finding the optimal match between NN architecture and target application. If this match is performed erratically, practical solutions often yield unimpressive results. It is the a) validation of the problem's fitness for a NN-based solution and b) matching of an optimal NN implementation to the given problem that is crucial. This paper presents a theoretical foundation for an inference engine decision space and a taxonomic framework for a knowledge base, which are part of our proposed knowledge-driven decision support system (DSS). Furthermore, this paper provides details of our inference engine, namely a) an algorithm for optimal matching of a NN setup against the given learning task, b) the application of this same algorithm for interactive exploration of the decision space and c) an algorithm for automatic inference of potential NN research synergies based on existing successful NN solutions. Finally, we propose a process for establishing and maintaining the growth of the DSS knowledge database.**

Keywords—Neural Networks; DSS; Knowledge Base; Taxonomy; Inference Engine.

I. Introduction

This paper presents a theoretical foundation for an inference engine decision space and a taxonomic framework for a knowledge base, which are part of our proposed knowledge-driven DSS. Furthermore, it introduces new additions to the suite of related algorithms, previously presented in [1].

The compelling notion, that NNs are universal approximators [2], leads quickly to believe, that any NN will do well on any presented machine learning task. However, the universal approximation theorem only guarantees the existence of an approximation, but not that it can be learned, nor that it would be efficient. Practice shows that every given problem requires a carefully crafted NN design and that advanced NN concepts, tailored to specific types of tasks are necessary to attain best results. This factor, among others, has led to the existence of a large number of conceptually varying NN architectures and learning algorithms [3].

For best results, any researcher or practitioner of today needs to understand a vast domain of knowledge in order to find a NN solution most suitable to their task. Due to domain vastness, researchers often limit themselves to NN domains they are familiar with, preventing new knowledge from propagating efficiently among all who would benefit from it. Specifically, we thus face a twofold handicap for progress of NN research: a) practitioners use suboptimal NN setups for real-world applications [3], inhibiting broader NN acceptance in the industry and b) researchers delve into local extrema of research (e.g., through jumping on the bandwagon of imminent peers [4]), pushing frontiers of NN research in suboptimal directions.

Figure 1 shows a simple flowchart view of the current typical approach to selection of NNs for chosen learning task. It illustrates that the lack of systematic approach to NN selection often yields suboptimal results. This has a negative effect on a wider acceptance of NNs in the industry. A key prerequisite in current NN design is expert intuition, which can be attained either through significant experience with NN implementation and applications in practice, or through access to expert intuition in an environment of experienced NN users. When expert intuition is present in early stages of design, the subsequent efforts give promising results (Figure 1, left); and the opposite, when not (Figure 1, right). The NN community needs a streamlined way of enabling existing and potential NN users to make optimal technological choices efficiently and systematically. Having today's foremost NN research applied in the industry can foster wider acceptance of NNs into practice and improve NN research.

In 2006, Taylor and Smith [5] created an important taxonomy-based evaluation of NNs, which aids in validating whether a given problem is solvable with a NN at all. The next concern, which they point out and we hereby address, is to choose the right NN architecture and its concrete implementation for the problem. Our goal is to provide a DSS for industry practitioners and researchers to systematically find the right NN for their application or research interest. This paper proposes a solution that enables (1) a systematical overview of the complete NN knowledge domain to (2) compare NN instances through their capabilities in a (3) quickly interpretative way using a framework that is (4) adaptable in terms of NN properties, even classification dimensions.

This paper is structured as follows. Section II briefly outlines the state of the art, Section III explains our approach to DSS design. Sections IV and V present the DSS' inference engine decision space and the taxonomy for DSS' knowledge

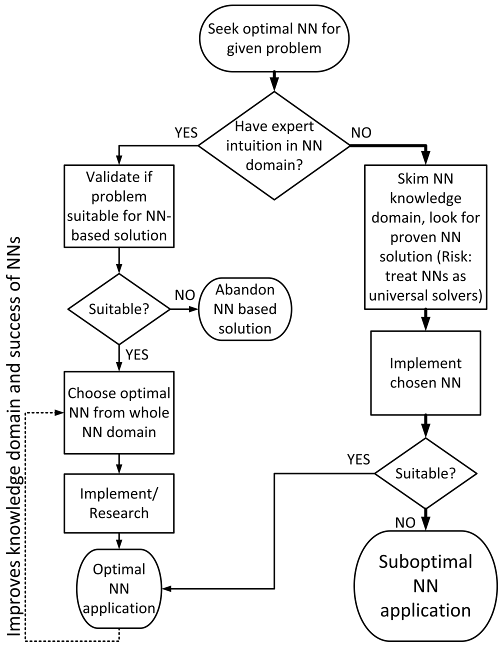

Seek optimal NN for given problem
Have expert intuition in NN domain?
YES
NO
Validate if problem suitable for NN-based solution
Suitable?
NO
Abandon NN based solution
YES
Choose optimal NN from whole NN domain
Implement/ Research
Optimal NN application
Skim NN knowledge domain, look for proven NN solution (Risk: treat NNs as universal solvers)
Implement chosen NN
Suitable?
YES
NO
Suboptimal NN application
Improves knowledge domain and success of NNs

Figure 1. Typical design flow of selection and implementation of NNs for a given learning task.

base, respectively. Sections VI and VII present the DSS and its main use cases, respectively. Section VIII proposes strategies for maintenance of the knowledge database, and finally, Section IX concludes this paper and outlines the future work.

II. State of the Art

The state of the art in NN design methodology can be split into two groups. The first group focuses on choosing the optimal NN *macrostructure* (e.g., Support Vector Machine versus Recurrent Neural Network), while the second group focuses on guidelines and (semi)automated methods, that help find the optimal *microstructure* (e.g., number of hidden layers and neurons) of a selected macrostructure.

The first group of approaches consists of guidelines and overview literature [6][7][8]. The problem (and virtue) with this set of methodologies is that they require understanding of a vast set of NN concepts, before the designer is able to make an optimal choice. Dreyfus [6] states, for example: "No recipes will be provided here. It is our firm belief that no significant application can be developed without a basic understanding of the principles and methodology of model design and training." Of course, we agree with this position. However, it can be observed in practice, that there is a lack of systematic approach in choosing the macrostructure. As a consequence, Feedforward NN (FFNN) [2], learned with Backpropagation (BP) [9], is still chosen in the majority of applications, which we consider a negative trend [10].

The second group of approaches is necessary for the fine microstructural tuning of a chosen macrostructure (also usually demonstrated on FFNN with BP). These approaches are either given as a set of rules and recipes, or as an automated optimization tool. The most systematic approaches rely on the Design of Experiments (DoE) method, involving Taguchi principles [11][12][13]. Such methods systemize and automate the selection of, e.g., number of hidden layers or neurons, through experimenting with different setups. Similar methods are constructive and pruning algorithms, that add or remove neurons from an initial architecture [14][15]. Also, related are evolutionary strategies, which employ genetic operators for similar purposes [16][17][18].

We turn to the state of the art in search of a formal taxonomy of the whole NN knowledge domain. For a taxonomy to serve as the foundation of our DSS, it must facilitate a qualitative measure between its elements. However, it turns out, that directly comparing NN instances in detail is prohibitively problematic due to bias or lack of method in the description process [19]. The Andrews-Diederich-Tickle (ADT) taxonomy [20] enables two NNs to be compared pairwise through ADT^5 criteria (defined by Andrews et al. [20] and refined by Tickle et al. [21]), but this taxonomy lacks orthogonality since some of its taxonomical categories (dimensions) are interdependent. Other taxonomies classify NNs purely through topology [22] or realization method [9]. And more recently, researchers create taxonomies that assist in choosing the best solution for the task [4][23] within a limited application area and solve locally what our work attempts to solve globally.

III. Design of the Decision Support System

Our DSS-based approach proposed fits between the two groups of NN design methodology, presented in Section II, and improves the results of both group's goals. It exhibits the main qualities of the second group (ability to automate the decision process) and applies them to the problematic of the first group (i.e., choosing the macrostructure), which is a crucial step in NN design, because the effect of any design actions depends greatly on early decisions. The aim of our proposed DSS is to improve the performance of NN-based based applications on a large scale, through enabling designers to perform optimal early design decisions. Figure 2 illustrates how our proposed DSS improves the NN design process by enabling users to systematically find optimal NN instances for their application.

The concept of a DSS is extremely broad and scientists have been researching DSSs for more than 40 years. Used in a broad range of applications, a DSS supports operations decision making, financial management, strategic decision-making in business organizations, military, logistics, etc., at different levels of an organization. In this work, we apply DSS principles to an engineering decision process.

A historical overview of DSSs is given in [24], along with a classification of DSSs into five distinct categories, summarized as follows:

- A **model-driven DSS** emphasizes access to and manipulation of financial, optimization and/or simulation models;
- A **data-driven DSS** emphasizes access to and manipulation of data, provides query tools, includes ad-hoc interpretation and visualization tools, data warehousing, etc.;

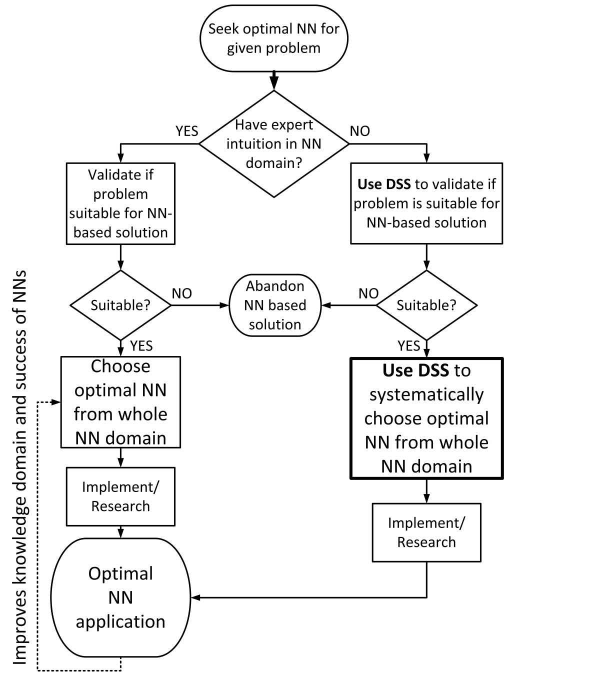

Seek optimal NN for given problem
Have expert intuition in NN domain?
YES
NO
Validate if problem suitable for NN-based solution
Use DSS to validate if problem is suitable for NN-based solution
Suitable?
NO
Abandon NN based solution
NO
Suitable?
YES
Choose optimal NN from whole NN domain
YES
Use DSS to systematically choose optimal NN from whole NN domain
Implement/ Research
Implement/ Research
Optimal NN application
Improves knowledge domain and success of NNs

Figure 2. Our proposed DSS improves the NN design process by enabling users to systematically find optimal NN instances for their application.

- A **communication-driven DSS** uses network and communications technologies to facilitate decision-relevant collaboration and communication between decision makers and consider communication technologies as the main architectural components;
- A **document-driven DSS** uses computer storage and processing technologies to provide document retrieval and analysis, with a search engine being the core architectural component and decision-making tool;
- A **knowledge-driven DSS** can suggest or recommend optimal actions to users, based on specialized problem-solving expertise, incorporated into the DSS.

After review of DSS theory in existing literature, we decide to design our proposed DSS as a Knowledge-Driven DSS, which comprises the following components:

1. Knowledge base
2. User interface
3. Inference engine model
4. Communications component

Corresponding to the above, also shown in Figure 3, our proposed system comprises the following: NN Knowledge base (Section V), 3D visualization of data and qualitative relations (Section VI-A), inference engine and qualitative relations in data (Section VI-B), interaction with 3D environment and entry of objective parameters (Section VII), respectively. Section VII demonstrates the use of inference engine in two major use cases and presents visually the inference results.

IV. Inference engine decision space

The main and fundamental result of our work is the conceptualization and theoretical foundation for the inference engine

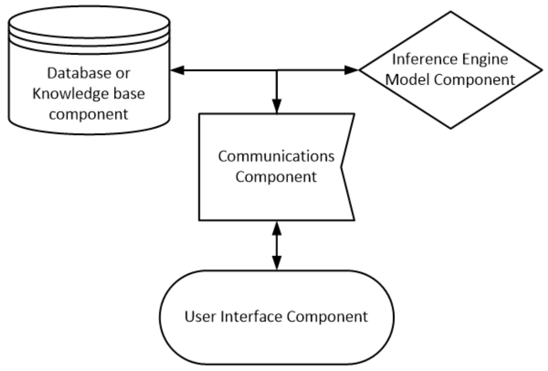

Database or Knowledge base component
Inference Engine Model Component
Communications Component
User Interface Component

Figure 3. Components of a knowledge-driven DSS.

decision space (this section) and knowledge base framework (Section V), that enables a global overview of the complete NN domain. The decision space, presented hereby, serves also as a coherent terminology and context for our knowledge base framework. Mathematical structure of interrelations must be well defined, to facilitate an effective inference engine, used in solving multiple-objective optimization problems [25]. The decision space is defined via the following descriptors of the NN knowledge domain:

- Set of NN instances $\mathcal{I}$: contains the subset of elements of the NN knowledge domain, which are neural networks. From the whole neural network knowledge domain (NNs, research initiatives, research groups, research goals, application areas, etc.), we gather concrete NN implementations and form the set of NN instances.
- NN classifier $\zeta : \mathcal{I} \to \mathcal{P}$: provides a classification of each member of $\mathcal{I}$ into a particular set of groups $\mathcal{P}$.
- Property $\mathcal{P}$: the co-domain of a classifier ζ, with the latter considered as a function.
- Property value $p_i \in \mathcal{P}$: a specific group of some classifier. It is given a name, which is then identified as this property value.
- NN framework $\mathcal{F}$: ordered list of classifiers relevant for a given user's interest.
- NN universe $\mathcal{U}$: defined by a framework $\mathcal{F}$, it is an $|\mathcal{F}|$-dimensional space, which is Cartesian product of the properties defined by the classifiers in $\mathcal{F}$.
- NN instance $\mathcal{I}_i \in \mathcal{I}$: $\mathcal{I}_i = (p_1, \ldots, p_f)$; an f-tuple of property values, each coming from its corresponding NN property.
- NN category $\mathcal{C}_p \subseteq \mathcal{P}$: subset of a specific property, containing a set of values (classifier groups) of this property. Possibly a singleton.
- NN landscape $\mathcal{L} = C_1 \times C_2 \times C_3 \times \ldots \times C_f$ with at least one C_i being equal to the whole property P_i. Subspace of a NN universe.
- NN type $\mathcal{T} = C_1 \times C_2 \times C_3 \times \ldots \times C_f$: Cartesian product of categories. If all categories in the cartesian product are singletons, the NN type is also a NN instance.
- NN comparator δ: innate comparative quality, defining a corresponding partial order, denoted as $>_\delta$, on the set of

NN instances $\mathcal{I}$, by which some pairs of NN instances can be compared. In our proposed DSS, NN comparators are chosen by defining the NN selection criteria (see Section V). NN comparators are represented as colored arrows between NN types, with the color specifying different NN instance selection criteria and the thickness of the arrow proportional to number of evidence papers supporting the comparison.

- NN selection criteria ∇: a set of possibly competing NN comparators used for comparison of NN instances.
- Pareto front $\mathcal{R}$ of given NN selection criteria ∇: a set of (discrete) NN instances $j \in \mathcal{R}$ such that whenever some NN instance $i \in \mathcal{I}$ is better than j with respect to some NN comparator $\delta \in \nabla$, i.e., $j >_\delta i$, then there is some other comparator $\delta' \in \nabla$, such that $i >_{\delta'} j$, i.e., i is better than j w.r.t. δ'. In other words, a NN instance belongs to the Pareto front of ∇, if it cannot be improved over without harming at least one of the NN selection criteria in ∇.

What signifies our approach is the decision to abandon the aim for back-to-back comparison of specific NN implementations via rigid criteria (which would limit us to NN research subdomains) and employ a flexible DSS, enabling self-organization of data and allowing the evolution of the framework, together with the evolution of knowledge base contents.

V. Taxonomy for Knowledge Base

In contrast to related taxonomic efforts, presented in Section II, our taxonomy must provide a significant level of abstraction, allowing both a complete field overview and sufficient depth to aid qualitative comparison, while providing the flexibility for future adaptations of the proposed classification. Our generically specified ranking between feasible solutions permits us to deliver rule-of-thumb guidance that provides an excellent starting point for further in-depth analysis based on, e.g., ADT^5 criteria.

With the inference engine decision space theoretically defined in Section IV, we proceed to determine the principal dimensions for classification of NN instances. As no single source provides a definitive field overview, we as first step systematically create a taxonomic blueprint for our knowledge base. We define the NN classifiers ζ as operators for sorting of NN instances into main taxonomic branches:

ζ_1 Implementation Platform
ζ_2 NN Architecture
ζ_3 Learning Paradigm
ζ_4 Learning Algorithm
ζ_5 Learning Task

Using our defined NN classifiers, we proceed to build the taxonomy. For its core, we extract the classification used in the book Neural Networks: A Comprehensive Foundation [5], which offers a wide overview of main concepts in NN domain. To build upon this core, we add the overviews of evolutionary methods [26], Spiking Neural Networks [27] and a recent 20-years overview of hardware-friendly neural networks [28]. A principal quality of our system lies in our choice of high abstraction when defining the taxonomy; e.g., while there exist numerous flavors of the BP algorithm, our taxonomy does not differentiate between them. Only by obscuring a such detail, we can achieve a domain-wide overview. Still, as the field of NNs is very diverse, an ultimate taxonomy requires broader community collaboration and finally, consensus; both of which exceed the scope of this work.

Through processing the selected literature using our defined NN classifiers, we find that our chosen NN classifiers map NN instances into NN properties (i.e., sets of NN categories $\mathcal{C}$, possibly singletons) $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4$ and $\mathcal{P}_5$, respectively:

$\mathcal{P}_1$ (ζ_1: **Implementation Platform**) takes values from:

- General Purpose ($\mathcal{C}^1_1$): Software Simulation on general purpose computer of Von Neumann Architecture (CPU), Digital Signal Processor (DSP) Graphical Processing Unit (GPU), Supercomputer (SCP)
- Dedicated Hardware ($\mathcal{C}^1_2$): Field Programmable Gate Array, Neural Hardware / Neural Processing Unit (NPU), Analog Implementation (ANLG), Application Specific Integrated Circuit (ASIC)

$\mathcal{P}_2$ (ζ_2: **NN Architecture**) takes values from:

- Feedforward Neural Network (FFNN)
- Second Generation NNs ($\mathcal{C}^2_2$): Recurrent Neural Network (RNN),Long Short-Term Memory (LSTM)
- Spiking Neural Network (SNN)
- Cellular Neural Network (CNN)
- Self-organizing Map (SOM)
- Reservoir Networks (RSV) ($\mathcal{C}^2_6$): Echo-state Network (ESN), Liquid-state Machine (LSM)
- Convolutional NN (CONN)
- Deep Belief Network (DBN)
- Hybrid (HYB)

$\mathcal{P}_3$ (ζ_3: **Learning Paradigm**) takes values from:

- Supervised Learning (SUP)
- Reinforcement Learning (REINF)
- Unsupervised Learning (UNSUP)
- Genetic Learning (GENL)

$\mathcal{P}_4$ (ζ_4: **Learning Algorithm**) takes values from:

- Error Correction ($\mathcal{C}^4_1$, ECR): Backpropagation (BP), Extended Kalman Filter (EKF), Direct Stochastic Error Descent (DSED),
- Hebbian Learning (HBL)
- Competitive Learning (CPL)
- Evolutionary ($\mathcal{C}^4_4$, EVOL): Evolution of Architecture (EVLARCH), Evolution of Weights (EVLWT), Evolution of Learning Algorithm (EVLALG)
- Hybrid (HYB)

$\mathcal{P}_5$ (ζ_5: **Learning Task**) takes values from:

- Pattern Association ($\mathcal{C}^5_1$): Autoassociation (PASCAUT), Heteroassociation (PASCHET)
- Pattern Recognition ($\mathcal{C}^5_2$, PREC): Natural Language Processing (NLP), Principal Component Analysis (PCA), Speech (SPC), Dimensionality Reduction (DRED), Spatio-temporal (SPT)

- Control ($\mathcal{C}_3^5$, CTL): Indirect (CTLIND), Direct (CTLDIR)
- Function Approximation ($\mathcal{C}_5^5$, FAPPROX): System Identification (SYSID), Inverse System (INVSYS)
- Classification (CSF)
- Regression (RGR)

Property $\mathcal{P}_2$ thus comprises 11 NN property values, gathered in 9 categories, of which $\mathcal{C}_2^2$ and $\mathcal{C}_6^2$ each contain two property values; $\mathcal{C}_2^2$ contains p_2^2 and p_3^2 and $\mathcal{C}_6^2$ contains p_7^2 and p_8^2. Property value indices run free from category indices.

A. Qualitative comparison through NN selection criteria ∇

With the taxonomic backbone defined, we can proceed with classification of NN instances from processed literature through property values $\mathcal{P}$, using our set of NN classifiers ζ. This comparative dimension, well-defined but very permitting, is a core facility of our knowledge base and the heart of our DSS' inference engine. Therefore, we also extract from literature sources the qualitative comparison information between NN instances w.r.t. the following set of chosen NN selection criteria ∇:

δ_1 **Low cost of ownership** (feasibility, practicality, low hardware cost, low development complexity, presence of user community)

δ_2 **Capability** (effectiveness, convergence speed, generalization performance, benchmark success, high learning rate, low error)

δ_3 **Real-time requirement** (speed of execution, on-line vs. off-line learning, pre-learned vs. adaptive learning)

δ_4 **Design maturity** (proven solution vs. emerging technology)

While estimates for all NN criteria can be extracted from literature or provided by a domain expert, design maturity could also be automatically calculated as a measure of occurrence frequency in literature.

B. 5-letter notation and knowledge base formation

In the 5-dimensional NN universe, defined by our NN framework $\mathcal{F}$, that we define in Section V through selecting our set of NN classifiers $\zeta_{1,\ldots5}$, each NN instance is described via five NN properties $\mathcal{P}_{1,\ldots5}$. Therefore, each element in the database compares two NN instances or NN landscapes in terms of five parameters. To construct our formal notation, we build upon the idea of 3-letter notation used in the theory of scheduling problems [29] and adapt it to a 5-letter notation for describing NN instances. Our resulting formal representation of relation(s) between two NN instances is as follows:

$$(\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_3,\mathcal{P}_4,\mathcal{P}_5) >_{\delta_{i\ldots n}} (\mathcal{P}'_1,\mathcal{P}'_2,\mathcal{P}'_3,\mathcal{P}'_4,\mathcal{P}'_5), \quad (1)$$

where each NN property $\mathcal{P}_{1\ldots5}$ can be a comma-separated list of elements (NN property values), n is the total number of selection criteria and where $i, i \in \{1\ldots n\}$ denotes the group of indices of NN qualifiers, by which the 'greater' NN instance is superior to the 'lesser' NN instance.

In our knowledge database, the following example statement extracted from a scientific source [28]: "FPGA is a superior implementation platform to ASIC in terms of flexibility and cost for implementations of FFNN or RNN with supervised or reinforcement learning, using perturbation-based error descent learning algorithms." is formally denoted as follows:

$$\begin{array}{l}(\mathcal{P}_1[4],\mathcal{P}_2[3,4],\mathcal{P}_3[1,2],\mathcal{P}_4[3],\mathcal{P}_5[x]) >_{\delta_1} \\ (\mathcal{P}_1[6],\mathcal{P}_2[3,4],\mathcal{P}_3[1,2],\mathcal{P}_4[3],\mathcal{P}_5[x]),\end{array} \quad (2)$$

Or:

$$\begin{array}{l}(FPGA,\{FFNN,RNN\},\{SUP,REINF\},\\ DSED,x)\\ >_{\delta_{cost,flexibility}}\\ (ASIC,\{FFNN,RNN\},\{SUP,REINF\},\\ DSED,x)\end{array} \quad (3)$$

This example also illustrates the case where the paper does not specify all property values (in this example, the learning task $\mathcal{P}_5[x]$), the statement is incomplete and it may mean either that the relation is indifferent to that property, or that there is no information present about that property's role in the relationship. After reviewing selected literature (e.g., [28][30][26][31]–[37]), we get a number of such specific statements that comprise our knowledge base seed information, which serves as basis for development of our inference engine and visualization scheme.

VI. Result: knowledge-driven DSS with inference engine and visualization tool

The proposed inference engine, together with knowledge base visualization, are the final results of our efforts presented in this paper. Both modules operate on the data in the knowledge database in a read-only fashion. In the following subsections, we present our scheme for exploratory visualization of our multidimensional knowledge database and describe our interactive inference engine.

A. Visualization scheme

Every point in the NN universe's graphical representation corresponds to one NN instance. The most valuable information in our knowledge database is the qualitative comparison between NN instances. This is shown in Figure 4, illustrating the graphical representation of Statement (3) from Section V-B. We have found that using three dimensions for the visualization is optimal, because it allows users to navigate the environment interactively and to recognize interdependencies, even after switching between the chosen set of three dimensions. The 3D visualization can only represent three dimensions at a time and the user can explore the NN knowledge domain using any dimension set.

Figure 5 shows the 3D representation our NN universe $\mathcal{U}$, containing points from our prototype knowledge base. This view allows users to examine the NN knowledge domain in a full 3D environment, visually exploring (through zoom and rotation of view around any axis) the comparative relations between NN instances. Axes correspond to NN properties $\mathcal{P}$; each dot corresponds to a single NN instance $\mathcal{I}_i$; arrows represent qualitative comparators $\delta_{1\ldots4}$ between two NN instances; arrow thickness and dot size indicate the quantity of

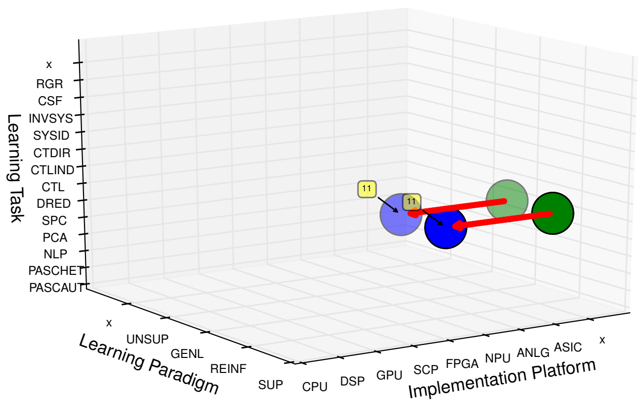


Figure 4. Example graphical representation of qualitative relation between two NN instances. The figure represents Statement (3) from Section V-B.

source papers (database entries) for the shown information; call-out-type labels are references to source literature. Each of the selection criteria is assigned its own arrow color (red, magenta, blue and black for δ_1, δ_2, δ_3 and δ_4, respectively). Coloring of NN instances aids in visual comparison (blue is better, green is worse). Using the inference engine (Section VI-B), the visualization can be actively augmented according to the user's decision input.

B. *Inference Engine*

Our inference engine approaches the NN instance selection process as a multiple-objective optimization problem [25] and applies a Pareto front method [38], using our disrete-equivalent Pareto front $\mathcal{R}$, as defined in Section IV, to find the suitable, multiple, non-dominant solutions. After the user specifies their boundary conditions and sets weights of the NN selection criteria ∇ through the graphical user interface, the DSS automatically identifies the discrete-equivalent of Pareto front $\mathcal{R}$ and the user can directly locate and examine the source literature, relating the NN instances in $\mathcal{R}$. Rating of alternatives is based on a weighted pairwise comparison matrix [39], resulting in levels within a discrete-space equivalent of Pareto front, which guide the user towards NN instances, specified as superior with respect to their criteria. The user can iteratively and interactively further fine-tune the selection of best candidates via weights of their criteria ∇, to determine the optimal NN instance for their problem, until the final choice is made. Information, inferred by the inference engine, is also used as input into the visualization tool, to augment the database visualization by superimposing relationships, marking Pareto points and their scores, hiding a subset of NN instances, etc. (see Figure 6). Both the visualization tool and the inference engine can be extended with additional inference and visualization functions. Section VII gives further insight into the typical application of the inference engine, through step-by-step explanation.

VII. Practical examples of DSS use

The two major user groups that can gain remarkable benefits from using our proposed DSS, are **Industry Practitioner** and **Academic Researcher**. Both user groups share the main interest of finding the optimal NN instance for their scenario, but have a different angle: a) the industry practitioner's goal is to find the **best fitting, well proven** NN implementation for their **application** (with set boundary conditions on task type, implementation platform, etc.), and b) the academic researcher's aim is to find an active research area or synergies between domains, to systematically selects the most meaningful **research direction**.

A. *DSS Use Case I: Industry Practitioner*

In this section, we illustrate step-by-step a typical use case for the industry practitioner. Let our example demand a **highly accurate** and **real-time capable** NN instance for image-based **object recognition** using **supervised learning**. The following steps illustrate how this ground truth is used with our DSS as decision input and how the inference engine results are interpreted and used:

1) Enter task requirements into DSS: the practitioner enters their set of boundary conditions by selecting the NN instance properties, that are defined by the application. In our example, these are the learning task and learning paradigm. The DSS considers these two properties as pivots, therefore, only the **remaining** three properties (dimensions) are shown in the 3D visualization tool (Figure 6a). After the axes are determined, the user specifies pivot axis values (learning task = classification, CSF and learning paradigm = supervised, SUP). The effect of this is shown in Figure 6b, where only those NN instances are shown, whose pivot property values are as specified by the user. Thus, this step narrows down the search down to three dimensions and defines the NN landscape, which optimal solutions can be chosen from. If more than two pivot axes are specified by the use case, the NN landscape is 2- or 1-dimensional, further focusing the search.

2) Set weights for selection criteria ∇: once the 3D NN landscape is defined in the previous step, the user specifies weights for each of ∇ within the range from -5 to 5. In this example, the selected weights for $\delta_{1...4}$ are (see Section V-A for list of criteria):

δ_1	**Low cost of ownership**: 2
δ_2	**Capability**: 5
δ_3	**Real-time requirement**: 5
δ_4	**Design maturity**: 3

3) Examine Pareto front $\mathcal{R}$: based on weighted criteria, the inference engine extracts NN instances, that belong to the discrete Pareto front $\mathcal{R}$. These are NN instances, for which there is no NN instance superior w.r.t. any of the selection criteria (no arrows leaving the NN instance). These points are highlighted by the DSS via black squares. For better viewing of the points in $\mathcal{R}$, the user can interact with the 3D view by rotation around any axis. This is seen in 6c (left), showing the $\mathcal{R}$ points in an updated view, obtained by rotating 6b

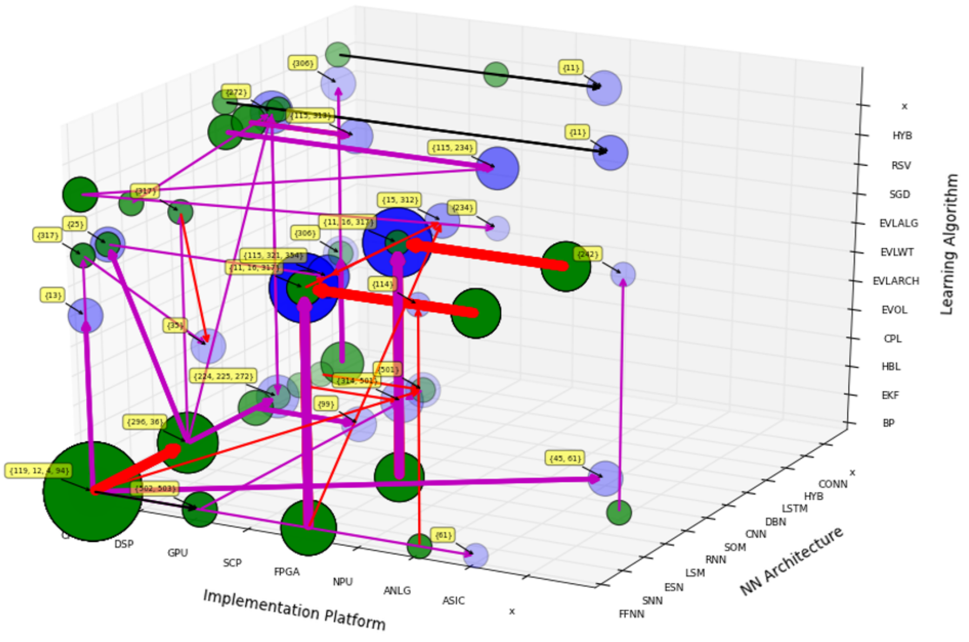


Figure 5. The 3D representation our NN universe $\mathcal{U}$, containing points from our prototype knowledge base. See Section III for axis label definitions.

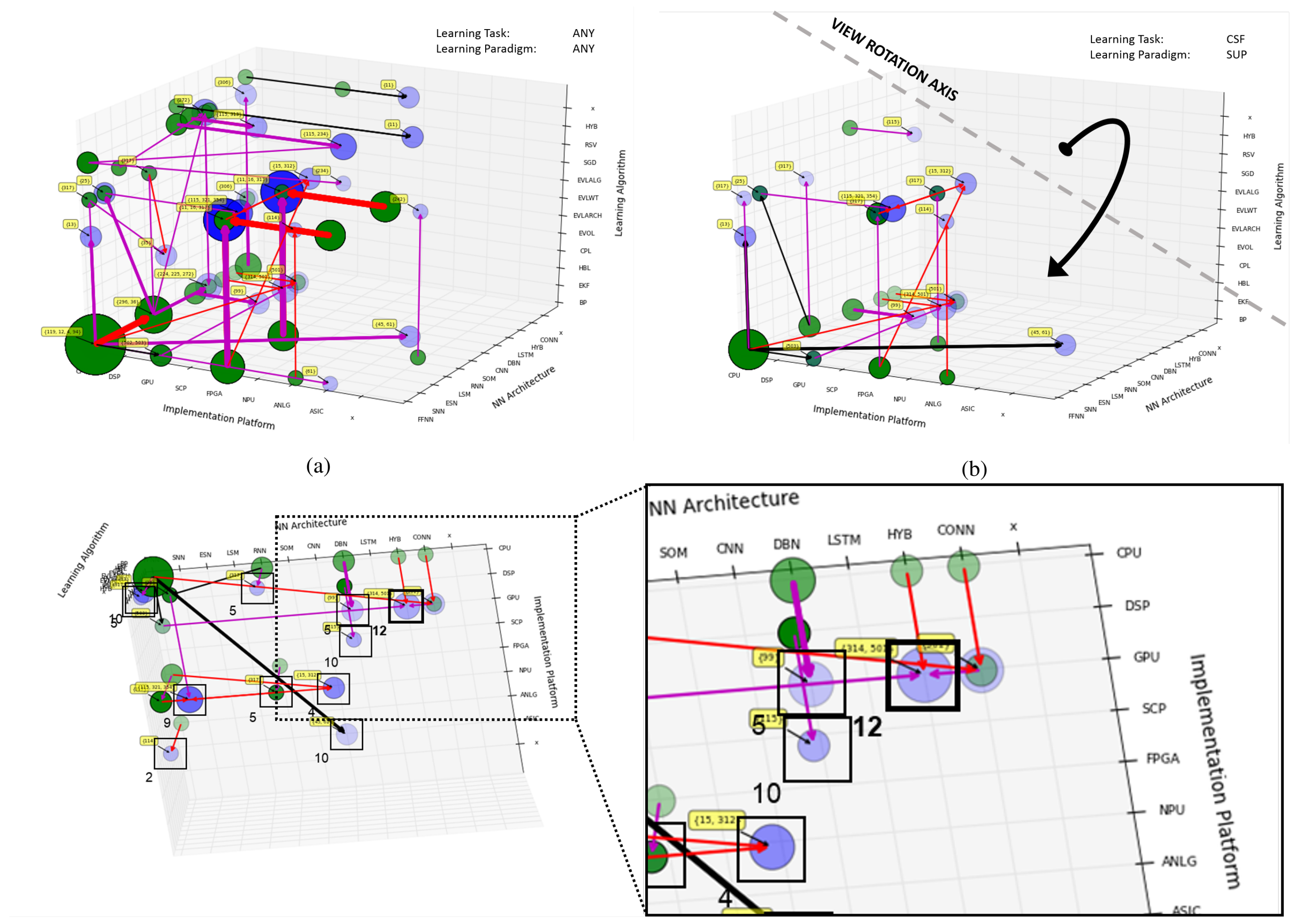


Figure 6. Rapid assessment of complete knowledge domain. Typical step-by-step application of the inference engine, together with the visualization scheme, used for finding an optimal NN instance, based on user input parameters.

around the 'view rotation axis' in the indicated direction. In the lower left corner of each $\mathcal{R}$-marking is the NN instance's score (closeup view in Figure 6c, right), calculated by the inference engine using the weighted pairwise comparison matrix.

4) Analyze top alternative in Pareto front: the user chooses the highest-ranking NN instances in the Pareto frontier and analyzes their corresponding source literature, indicated by call-outs (see Figure 4). Our example gives the highest score of 12 to NN instance, described in database entries 314 and 502 (Figure 6c). From corresponding source papers [33] and [34], the practitioner learns, that a) FFNNs can be used as convolution NNs, b) GPU implementation in [34] has better flexibility than previously known implementations, c) GPU implementation of CONN has better real-time capabilities than CPU implementation, d) Hybrid between pure CONN and FFNN has better recognition performance than any of these two used alone, e) hybrid implementation in [34] has won an impressive series of image classification competitions, etc.

In conclusion, based on the industry practitioner's input criteria, the DSS recommends, that a GPU-based hybrid CONV-FFNN NN should be investigated as best choice for the given use case. This simple case illustrates how a user can, using our DSS in a few simple steps, rapidly traverse an immensely diverse knowledge base, in order to choose an optimal direction for further investigation, and finally, concrete implementation.

5) Iterate until optimal choice is found: the user can tune the criteria to observe how the levels in the Pareto front change. Through such exploration of the NN landscape, the practitioner learns whether there exist satisfactory NN instances for their problem, given their chosen criteria, and if it exists, they systematically finds the most suitable NN instance of current state of the art for their application. Furthermore, through this exploration, the practitioner is exposed to ever new concepts and types of NNs, and at the same time also observes their relevance with respect to their own selection criteria. The database is equipped with direct references to literature sources, so the practitioner can learn about new NN concepts that directly pertain to their problem. Thus, even if there is no NN instance that fits directly with their selection criteria, the practitioner can systematically find possibilities for hybridization of seemingly unrelated approaches. This requires understanding of a broad area of NN technology and our knowledge database has a crucial role in the process. Finally, once the optimal NN instance is found or a fruitful new hybrid approach selected, the practitioner can attempt to implement the solution or conduct novel research on their own, or use our knowledge base directly to find the most relevant research groups to form collaboration. Thus, researchers can, through exploration, learn about relevant use cases for their NN types, which steers their research towards compelling yet-unknown real-world scenarios for NN use.

B. DSS Use Case II: Academic Researcher

Even though the knowledge base is constructed on existing science and lets us explore within its own, confined boundaries, it can also lead us to the discovery of new synergies, expanding the domain knowledge. It is able to do this because it can be used as a broad overview to understand the limitations of existing practices and give an indication where more research is required. This leads us to the second use case for an Academic Researcher.

For clarity of explanation of the second use case, we take the same decision input as in the case for Industry Practitioner: a highly accurate and real-time NN instance for image-based object recognition using supervised learning. However, we slightly modify the choice of weights for qualitative comparators, because an Academic Researcher is typically less interested in design maturity. Therefore, the weights used for the qualitative comparators $\delta_{1...4}$ in the selection criteria ∇ are as follows:

δ_1 **Low cost of ownership**: 2
δ_2 **Capability**: 5
δ_3 **Real-time requirement**: 5
δ_4 **Design maturity**: 1

The main significance of this use case is the notion of *pending instances $\mathcal{I}^*$*. These are *inferred* by the inference engine to direct the researcher to viable research directions, based on the highest scoring properties of existing NN instances.

The academic researcher further examines the pending NN instances $\mathcal{I}^*$ by reviewing related literature and experimenting. This is where the exploratory visualization plays a crucial role, because *through guided navigation through the NN knowledge database, the academic researcher can discover combinations of NN properties that have not yet been researched, but have high research potential.*

This gives the researcher the opportunity to critically assess the reasons why certain combinations of NN properties have not yet been tested. Furthermore, it enables the researcher to identify voids in current research and prompts them to direct their research into undiscovered domains.

The algorithm for rating and highlighting *previously-not-researched NN instances with high research potential* is:

Step I: **Enter task requirements into DSS:** The decision input is the same as for the Industry Practitioner use case.

Step II: **Set weights for selection criteria ∇**: The decision input is the same as for the Industry Practitioner use case.

Step III: **Examine Pareto front $\mathcal{R}$**: In this step, points in the Pareto front are computed by the DSS to be used as input in the subsequent steps.

Step IV: **Infer pending NN instances $^*\mathcal{I}$ with high research potential:** Using Algorithm 1, the inference engine determines the NN instances, that are not yet present in the knowledge database, but - based on the scores of NN property values of NN instances included in the Pareto front - have high research potential.

Step V: **Visualize pending NN instances $^*\mathcal{I}_i$**: Superimpose pending NN instances into the 3D visualization (Figure 7a).

Algorithm 1: Infer and rank pending NN instances based on Pareto front members

```
Step 1: Calculate score of individual NN property values of Pareto front members:
Input: List of rated NN instances $\mathcal{I}_R \in \mathcal{R}$ according to
       selection criteria $\nabla$ (based on decision input from
       Section VII-B)
foreach NN property j that is not a pivot do
    foreach NN property value $p_j$ do
        foreach NN instance $\mathcal{I}_R \in \mathcal{R}$ do
            add score of $\mathcal{I}_R$ to score of $p_j$
Step 2: Create NN instances from all possible combinations of $p_j$ with non-zero scores.
Step 3: From set of NN instances created in Step 2, remove all that already exist in the knowledge database. The remaining NN instances make up the set of pending NN instances $^*\mathcal{I}$.
Step 4: For each NN instance $^*\mathcal{I}_i \in {}^*\mathcal{I}$, compute its score through the sum of scores of its property values $p$ (calculated in Step 1).
```

Step VI: **Per user specification input, limit number of displayed** $^*\mathcal{I}$: For better visibility, the user specifies the upper threshold for the number of pending NN instances to be displayed. In Figure 7b, this threshold is set to 30%, meaning that only the top 30% of the high-scoring pending NN instances are shown in the 3D view.

Following the steps I to VI, the researcher is drawn to areas of research, which do not yet exist in the knowledge database (see Figure 8). The researcher thus learns from the visualization, that (see Figure 9):

- in terms of NN Architecture, the top-scoring unresearched NN instances are SNN, RNN, CNN, DBN and hybrid architectures (Figure 9c)
- in terms of implementation platform, most pending NN instances are on GPU, FPGA and ASIC implementation platforms (Figure 9d)
- in terms of learning algorithm, most pending NN instances use backpropagation (BP), direct stochastic error descent (DSED) and Evolutionary learning (the variant based on evolving weights, EVLWT). (Figure 9e).
- overall, top-five best scoring pending NN instances are (Shown in Figure 8 and, with context, in Figure 9f):
  - (GPU, CNN, SUP, BP, CSF); score: 165
  - (GPU, SNN, SUP, BP, CSF); score: 164
  - (GPU, RNN, SUP, BP, CSF); score: 142
  - (FPGA, CNN, SUP, BP, CSF); score: 139
  - (FPGA, SNN, SUP, BP, CSF); score: 138

A choice of a different set of input parameters yields a completely different set of interest points and viability scores. Using different settings, the academic researcher without a narrowly-defined problem area can explore different problem areas and select their research direction based on DSS-determined viability score and their own interests, e.g., preferences regarding implementation platform, depending on previous knowledge.

VIII. Process for establishing and maintaining the knowledge database

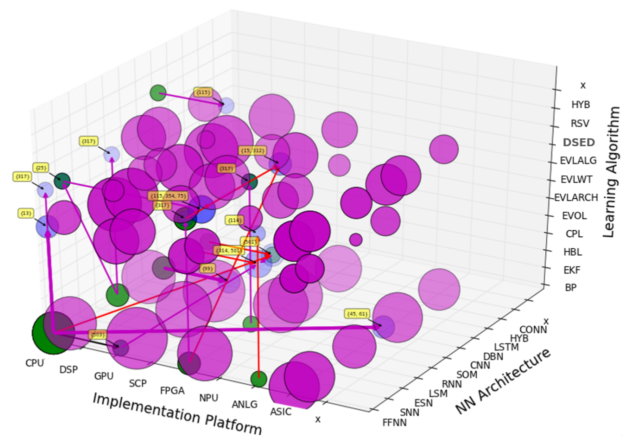
In this section, we propose concrete steps to fully exploit the potential of our results and develop our initiative further. The presented foundation, laid out as result of our research, if adopted and its potential fully exploited, can bring industrial applications of NNs closer to the forefront of today's NN research.

A central NN authority, ideally the International Neural Network Society (INNS) in coordination with IEEE Computational Intelligence Society (IEEE CIS) could consider endorsing this effort and form a working group to refine our taxonomy for use as a definitive basis for the proposed knowledge base. Once the knowledge base reaches critical mass, researchers could be motivated to contribute their own work or populate it with papers from where they notice a lack of coverage. For initial population of the knowledge base, a special issue journal could attract survey papers of NN instances and relations, represented fully in the 5-letter notation. The editorial board could, per need basis, revise the taxonomy when novel NN properties or categories emerge. When authors embrace the 5-letter notation in their papers, this information could, after the review process, be parsed and input into the knowledge base automatically.

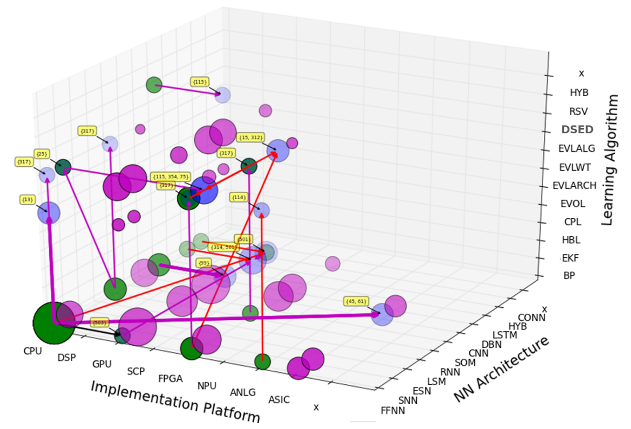
An online resource could be provided where the knowledge base, enabling navigation through the visual representation, would be freely accessible. A moderated collaborative editing among researchers could also be considered. Thus, also the expert system users themselves could submit data gathered from studying their chosen domain. An automatically-generated dynamic survey paper could be always kept up-to-date and available in printed form for quick overview of recent developments. For the database to reach critical mass, initial effort may be supported by a central NN authority. The NN community could thus set a precedence and the conceptual solution could be transferred also to other research fields.

IX. Conclusion and Future Work

In this work, we have identified the need for an abstract-level overview of the NN knowledge domain and alleviate the barriers, which an industry practitioner or researcher meet, when selecting the right NN instance or research direction for their specific scenario. We devised a theoretical foundation for a DSS, comprising a knowledge database and inference engine, that can automate the decision process of choosing the best NN architecture for the task at hand. We also presented a prototype implementation and a proof-of-concept through step-by-step use of our DSS. Next, we demonstrated an extension to the inference engine, which support automatic inference of



(a) The DSS inference engine infers pending NN instances $^*\mathcal{I}$, using NN property values of Pareto points inferred based on user input. Pending NN instances are marked as magenta circles, superimposed onto the 3D database view. Marker size corresponds to score.



(b) Updated view of Figure 7a, after user specifies that only the top 30% of the high-scoring pending NN instances be shown. Pending NN instances are marked as magenta circles, added to the 3D database view. Marker size corresponds to score.

Figure 7. Academic researcher use case: examination of pending NN instances.

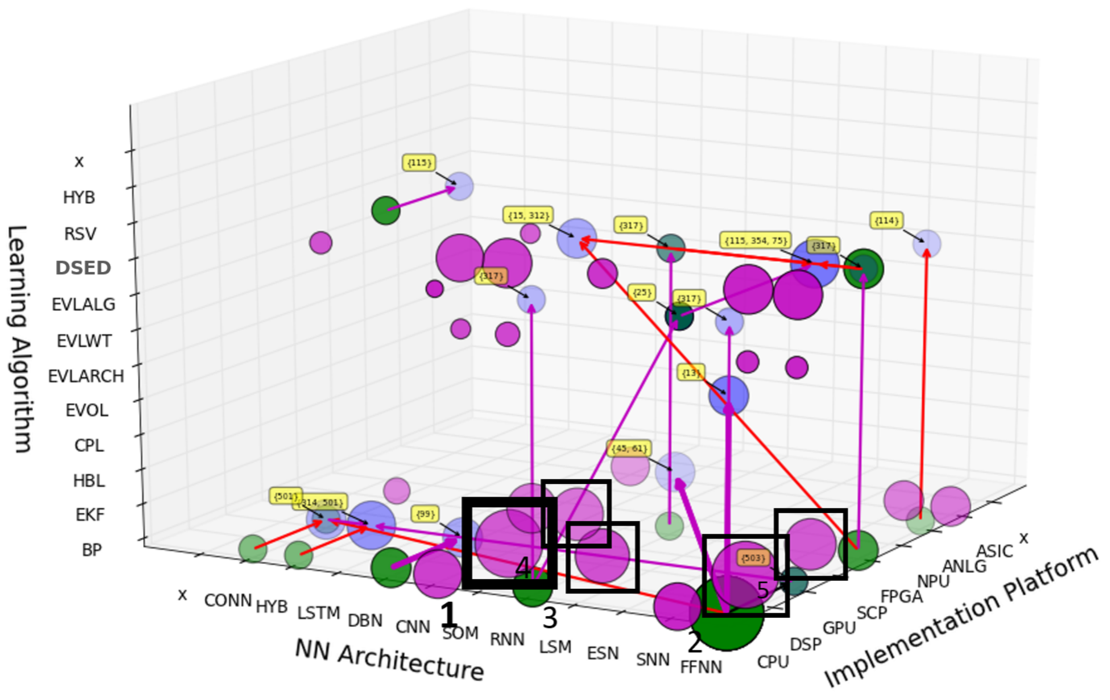
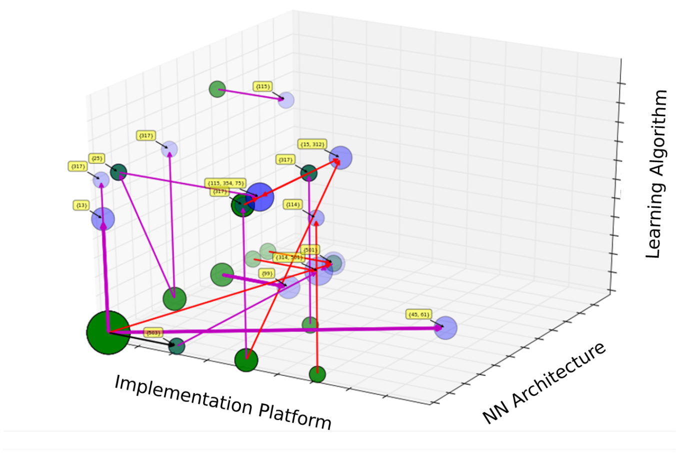
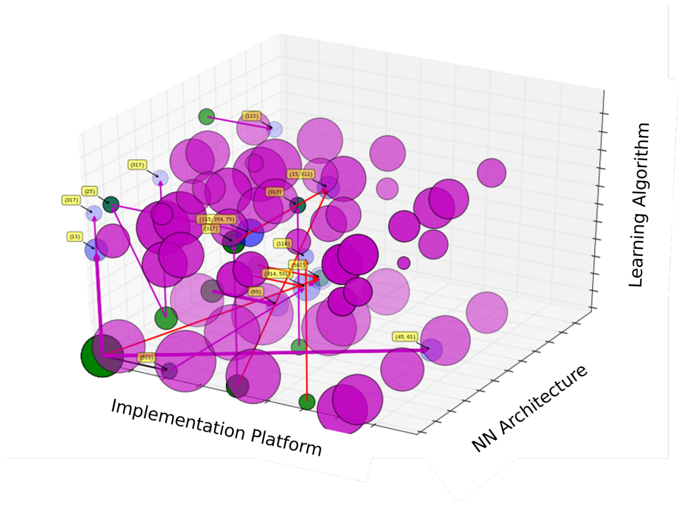


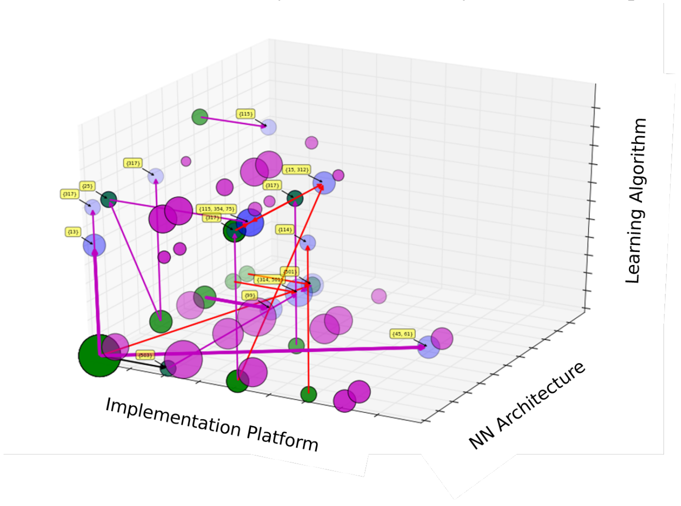
Figure 8. Best view of top-five ranking pending NN instances, marked with black squares, ranked in lower left corner.



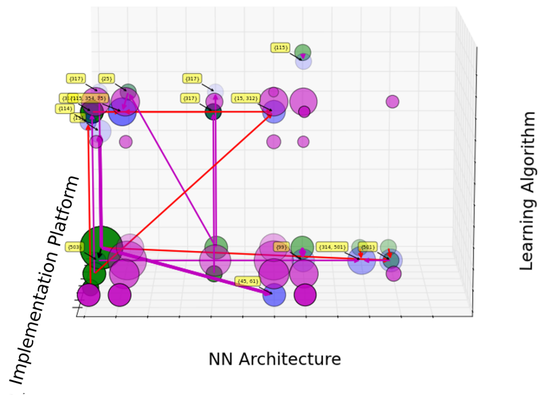
(a) NN instances that satisfy the user boundary conditions (Step I).



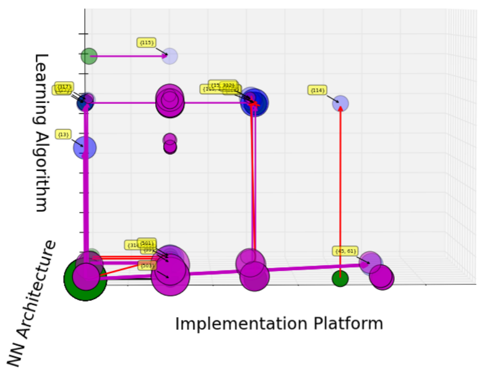
(b) Superimposed pending NN instances (Step V).



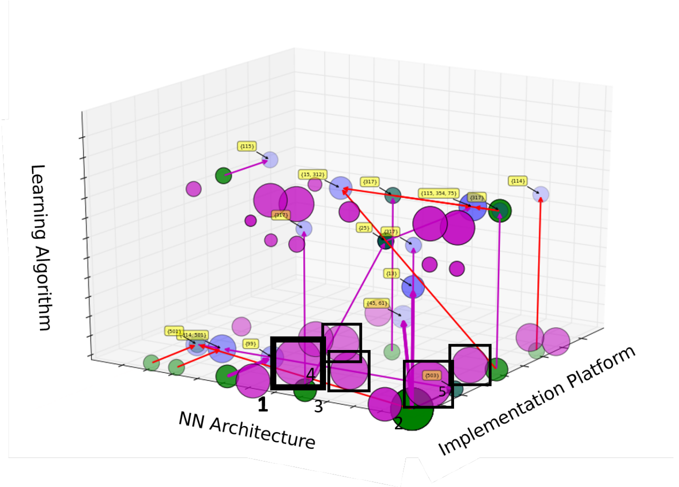
(c) Top 30% pending NN instances (Step VI).



(d) Figure 9c after clockwise rotation around Learning Algorithm axis by cca 45°.



(e) Figure 9d after clockwise rotation around Learning Algorithm axis by cca 90°.



(f) Best view of top-five ranking pending NN instances, marked with black squares, ranked in lower left corner (same as Figure 8).

Figure 9. Academic researcher use case: the DSS infers and visualizes highest-scoring pending NN instances.

promising combinations of NN properties, based on current highest-scoring NN instances within the database. This will enable our system to automatically highlight synergies between existing NN design approaches.

Future work aims towards moderated, collaborative editing of the knowledge base among researchers. The proposed 5-letter notation enables automatic parsing of the literature, keeping the knowledge database up-to-date at all times and solving this problem once and for all.

Acknowledgment

Operation part financed by the European Union, European Social Fund. The authors would like to thank the reviewers for their valuable comments that contributed greatly in improving this paper.

References

[1] R. Tavčar, J. Dedič, D. Bokal, and A. Žemva, "Towards a decision support system for automated selection of optimal neural network instance for research and engineering," in Proceedings of Advanced Engineering Computing and Applications in Sciences (ADVCOMP), The Eighth International Conference on. Rome, Italy: ADVCOMP, 2014, pp. 78–85.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 5, 1989, pp. 359–366.

[3] B. M. Wilamowski, "Neural network architectures and learning algorithms," Industrial Electronics Magazine, IEEE, vol. 3, no. 4, 2009, pp. 56–63.

[4] H. Jacobsson, "Rule extraction from recurrent neural networks: A taxonomy and review," Neural Computation, vol. 17, no. 6, 2005, pp. 1223–1263.

[5] B. Taylor and J. Smith, "Validation of neural networks via taxonomic evaluation," in Methods and Procedures for the Verification and Validation of Artificial Neural Networks. Springer US, 2006, pp. 51–95.

[6] G. Dreyfus, "Modeling with neural networks: Principles and model design methodology," in Neural Networks. Springer Berlin Heidelberg, 2005, pp. 85–201.

[7] A. Omondi and J. C. Rajapakse, FPGA Implementations of Neural Networks. Springer Netherlands, 2006.

[8] Y. Huang, "Advances in artificial neural networks–methodological development and application," Algorithms, vol. 2, no. 3, 2009, pp. 973–1007.

[9] R. Rojas, "Neutral networks: A systematic introduction," Springer, 1996.

[10] C. Moraga, "Design of neural networks," in Knowledge-Based Intelligent Information and Engineering Systems. Springer, 2007, pp. 26–33.

[11] J. Ortiz-Rodríguez, M. Martínez-Blanco, and H. Vega-Carrillo, "Robust design of artificial neural networks applying the taguchi methodology and doe," in Electronics, Robotics and Automotive Mechanics Conference, 2006, vol. 2. IEEE, 2006, pp. 131–136.

[12] E. Inohira and H. Yokoi, "An optimal design method for artificial neural networks by using the design of experiments." JACIII, vol. 11, no. 6, 2007, pp. 593–599.

[13] J. F. Khaw, B. Lim, and L. E. Lim, "Optimal design of neural networks using the taguchi method," Neurocomputing, vol. 7, no. 3, 1995, pp. 225 – 245.

[14] J.-F. Qiao, Y. Zhang, and H.-g. Han, "Fast unit pruning algorithm for feedforward neural network design," Applied Mathematics and Computation, vol. 205, no. 2, 2008, pp. 622–627.

[15] H. Han and J. Qiao, "A self-organizing fuzzy neural network based on a growing-and-pruning algorithm," Fuzzy Systems, IEEE Transactions on, vol. 18, no. 6, 2010, pp. 1129–1143.

[16] S. G. Mendivil, O. Castillo, and P. Melin, "Optimization of artificial neural network architectures for time series prediction using parallel genetic algorithms," in Soft Computing for Hybrid Intelligent Systems. Springer, 2008, pp. 387–399.

[17] Z.-J. Zheng and S.-Q. Zheng, "Study on a mutation operator in evolving neural networks," Journal of Software, vol. 13, no. 4, 2002, pp. 726–731.

[18] G. G. Yen, "Multi-objective evolutionary algorithm for radial basis function neural network design," in Multi-Objective Machine Learning. Springer, 2006, pp. 221–239.

[19] M.-T. Vakil-Baghmisheh and N. Pavesic, "A fast simplified fuzzy artmap network," Neural Processing Letters, vol. 17, no. 3, 2003/06/01 2003, pp. 273–316.

[20] R. Andrews, J. Diederich, and A. B. Tickle, "Survey and critique of techniques for extracting rules from trained artificial neural networks," Knowledge-Based Systems, vol. 8, no. 6, 1995, pp. 373–389.

[21] A. Tickle, R. Andrews, M. Golea, and J. Diederich, "The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks," Neural Networks, IEEE Transactions on, vol. 9, no. 6, 1998, pp. 1057–1068.

[22] E. Fiesler, "Neural network classification and formalization," Computer Standards & Interfaces, vol. 16, no. 3, 1994, pp. 231–239.

[23] H. R. Maier, A. Jain, G. C. Dandy, and K. P. Sudheer, "Methods used for the development of neural networks for the prediction of water resource variables in river systems: Current status and future directions," Environmental Modelling & Software, vol. 25, no. 8, 2010, pp. 891–909.

[24] D. J. Power, Decision support systems: concepts and resources for managers. Greenwood Publishing Group, 2002.

[25] N. Xiong and M. L. Ortiz, "Principles and state-of-the-art of engineering optimization techniques," in ADVCOMP 2013, The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences, 2013, pp. 36–42.

[26] S. Ding, H. Li, C. Su, J. Yu, and F. Jin, "Evolutionary artificial neural networks: a review," Artificial Intelligence Review, 2013, pp. 1–10.

[27] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," International Journal of Neural Systems, vol. 19, no. 4, 2009, pp. 295–308.

[28] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," Neurocomputing, vol. 74, no. 1-3, 2010, pp. 239–255.

[29] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," Annals of Discrete Mathematics, vol. 5, 1977, pp. 287–326.

[30] L. Fortuna, P. Arena, D. Balya, and A. Zarandy, "Cellular neural networks: a paradigm for nonlinear spatio-temporal processing," Circuits and Systems Magazine, IEEE, vol. 1, no. 4, 2001, pp. 6–21.

[31] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez, "Training recurrent networks by evolino," Neural Computation, vol. 19, no. 3, 2007, pp. 757–779.

[32] G. Andrienko et al., "Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns," Computer Graphics Forum, vol. 29, no. 3, 2010, pp. 913–922.

[33] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, ""the german traffic sign recognition benchmark: a multi-class classification competition"," in Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011, pp. 1453–1460.

[34] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," Neural Networks, vol. 32, 2012, pp. 333–338.

[35] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in ICML, vol. 9, 2009, pp. 873–880.

[36] D. Coyle, "Neural network based auto association and time-series prediction for biosignal processing in brain-computer interfaces," Computational Intelligence Magazine, IEEE, vol. 4, no. 4, 2009, pp. 47–59.

[37] R. K. Al Seyab and Y. Cao, "Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation," Journal of Process Control, vol. 18, no. 6, 2008, pp. 568–581.

[38] M. Farshbaf and M.-R. Feizi-Derakhshi, "Multi-objective optimization of graph partitioning using genetic algorithms," in Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09. Third International Conference on, Oct. 2009, pp. 1–6.

[39] S. Ghodsypour and C. O'Brien, "A decision support system for supplier selection using an integrated analytic hierarchy process and linear programming," International Journal of Production Economics, vol. 56–57, 1998, pp. 199 – 212, production Economics: The Link Between Technology And Management.