
SIMPLEMIND ADDS THINKING TO DEEP NEURAL NETWORKS

TECHNICAL REPORT

Youngwon Choi, M. Wasil Wahi-Anwar, Matthew S. Brown*

Center for Computer Vision and Imaging Biomarkers
 Department of Radiological Sciences
 David Geffen School of Medicine at UCLA
 University of California, Los Angeles

December 1, 2022

ABSTRACT

Deep neural networks (DNNs) detect patterns in data and have shown versatility and strong performance in many computer vision applications. However, DNNs alone are susceptible to obvious mistakes that violate simple, common sense concepts and are limited in their ability to use explicit knowledge to guide their search and decision making. While overall DNN performance metrics may be good, these obvious errors, coupled with a lack of explainability, have prevented widespread adoption for crucial tasks such as medical image analysis. The purpose of this paper is to introduce *SimpleMind*, an open-source software framework for Cognitive AI focused on medical image understanding. It allows creation of a knowledge base that describes expected characteristics and relationships between image objects in an intuitive human-readable form. The SimpleMind framework brings thinking to DNNs by: (1) providing methods for reasoning with the knowledge base about image content, such as spatial inferencing and conditional reasoning to check DNN outputs; (2) applying process knowledge, in the form of general-purpose software agents, that are chained together to accomplish image preprocessing, DNN prediction, and result post-processing, and (3) performing automatic co-optimization of all knowledge base parameters to adapt agents to specific problems. SimpleMind enables reasoning on multiple detected objects to ensure consistency, providing cross checking between DNN outputs. This machine reasoning improves the reliability and trustworthiness of DNNs through an interpretable model and explainable decisions. Example applications are provided that demonstrate how SimpleMind supports and improves deep neural networks by embedding them within a Cognitive AI framework.

Keywords Cognitive AI · neuro-symbolic AI · trustworthy AI · deep neural networks · knowledge base · medical image analysis

1 Introduction

Deep neural networks (DNNs) detect patterns in data and have shown versatility and strong performance in many computer vision applications. However, despite a large number of research publications and promising headlines there has not been broad adoption of artificial intelligence (AI) in crucial tasks such as medical imaging. Medical researchers have pointed to potential overestimation of DNN performance (Aggarwal et al., 2021), diminished DNN performance on external datasets (Yu et al., 2022), lack of consistent performance across cohorts (Panch et al., 2019), and the need to build trustworthy AI (Hasani et al., 2022).

DNNs can be considered a "stimulus-response" function without the thinking, using knowledge and reasoning, that makes human vision superior. DNNs are susceptible to obvious ("dumb") mistakes that violate simple common sense concepts and are limited in their ability to use explicit knowledge to guide their search and decision making. While overall DNN performance metrics may be good, these obvious errors, coupled with a lack of explainability, have prevented widespread adoption for crucial tasks such as medical image analysis.

*Corresponding author: mbrown@mednet.ucla.edu

To improve computer vision accuracy and reliability we embed deep neural networks within a Cognitive AI framework to meld pattern recognition with conceptual knowledge and reasoning. Cognitive AI includes not only the learning of patterns in data, but also learning through teaching and concepts (declared knowledge) as well as reasoning to apply this knowledge to guide the interpretation of a specific image. We provide an implementation of “thinking” that encompasses both learning and reasoning.

The purpose of this paper is to introduce *SimpleMind*, an open-source software framework for image understanding, i.e., segmenting and recognizing image elements to form a coherent high-level model of a scene where reasoning can be performed. The SimpleMind framework brings thinking to DNNs by:

1. allowing creation of a knowledge base for a type of image,
2. providing methods for reasoning with the knowledge base about image content such as spatial inferencing and conditional reasoning to check neural network outputs,
3. applying process knowledge, in the form of general-purpose software agents, that can be chained together to accomplish image preprocessing, DNN prediction, and result post-processing, and
4. performing automatic co-optimization of all knowledge base parameters to adapt agents to specific problems.

The framework uses a multi-agent architecture where the knowledge base and processing are decoupled. It uses a semantic network to represent domain knowledge intuitively and to guide processing agents — for example, segmentation agents that include DNNs and machine reasoning agents that evaluate the results. We will describe the SimpleMind framework for Cognitive AI and demonstrate its flexible application to medical image segmentation on chest x-ray (Brown et al., 2022), kidney CT (Melendez-Corres et al., 2021), and prostate MRI (Choi et al., 2022).

2 Knowledge Representation

SimpleMind adds reasoning to deep neural networks using a knowledge base that is explicit, human-readable, and decoupled from processing algorithms. This stands in contrast to implicit (“black box”) knowledge encoded within DNN weights or in pre/post processing code. In a SimpleMind application for a particular type of image, the knowledge base defines what is known and can be considered “long term memory”. The representation is based on semantic networks, originally developed and applied in language processing (Quillian, 1968), and is intuitive and allows easy creation, extension, and re-use of knowledge.

The knowledge base for a SimpleMind application is created as a semantic network (SN) where each node represents a concept — typically an object, object component, or object state. Each node contains attributes that describe expected object characteristics relating to size, shape, pixel intensity, and relative position. Spatial relationships that can be described between objects, include part of, right of, left of, above, below, inside, etc. Attributes are derived from a vocabulary that defines the name of the attribute and its associated parameters. For example, the vocabulary defines “PartOf” as physical composition with a single parameter, the name of the related parent node. When used to create an attribute in a semantic network, “PartOf” forms a relational link between two nodes. The vocabulary also defines attributes with numerical parameters, such as “RightOf”, which includes two parameters: (1) the related node, and (2) the expected distance to the right. Fuzzy sets are currently used to represent prior expectations for object characteristics using a confidence function over the range of possible parameter values (Zadeh, 1965). The fuzzy functions can be set initially by a human expert and refined as the system learns from data (see Section 5).

In addition to content knowledge (e.g., anatomical knowledge in medical images), the semantic network attributes can represent procedural knowledge used by processing agents, including deep neural network architectures, learning hyper parameters, and image pre and post processing parameters. Crucially, all attribute parameters in the semantic network are exposed (separate from the processing code) and human readable, so they can be both specified by a human and auto optimized by SimpleMind (see Section 5). A SimpleMind DNN agent trains DNN weights using the above attribute parameters from a given SN node. The DNN weights are then stored with the node, embedding the DNN within the semantic network. Thus a SimpleMind knowledge base can include both declared knowledge (that it is “taught”) and learned knowledge from examples (acquired through machine learning), i.e., we can actively teach the system as well as have it learn passively from data.

The framework provides tools for a user to create a knowledge base for their application by adding semantic network nodes and attributes. The nodes of a semantic network are stored in human readable text files. The network can be scaled using the existing vocabulary by adding nodes and attributes that expand the knowledge base. The SimpleMind vocabulary can be also extended, expanding the attributes that can be used when creating a knowledge base.

The semantic network provides a direct approach to capturing knowledge, as opposed to indirectly represented and entangled within processing code. With this abstraction, developing a SimpleMind application is like teaching or

instructing a human at a cognitive level without writing code for each processing step. The SimpleMind framework handles the aggregation and chaining of many processing agents, enabling a large scale Cognitive AI system. In the next section we will describe how processing agents use the knowledge base.

3 Multi-Agent Vision System

In SimpleMind, computer vision involves recognizing objects (nodes) from the knowledge base in a given image. Multiple software agents work together to segment the image into candidate regions, then select the best candidate based on object attributes described in the knowledge base. Software agents collaborate to solve the vision problem by reading from, and writing to, a global Blackboard data structure (Corkill, 2003). The analogy is a team of specialists observing and writing on a blackboard to contribute their specific expertise to solving a problem jointly. The Blackboard is the working space, or “short term memory”, of SimpleMind during the “thinking” process, i.e., during comparison and matching of the image to a knowledge base for image understanding. An agent can read information from the Blackboard generated by other agents and add or update information. Agents operate independently and collaborate only via the Blackboard, giving them a degree of autonomy and making the system more flexible and scalable.

For each node in the semantic network, a data structure called a Solution Element is created on the Blackboard, corresponding to an object to be recognized in the image. The Solution Element stores all agent contributions while recognizing the object. Knowledge base attributes and candidate image regions are transformed into a common feature space for selection of the best candidate. A Knowledge Agent accesses the knowledge base and creates each Solution Element, initializing it with prior expectations for object feature values. Knowledge base attributes reflect commonly understood concepts, such as left, right, above, below; these spatial attributes are transformed to a “relative centroid position” feature with x, y, and z axis parameters that are set according to the particular attribute. Knowledge base object attributes can be transformed to unary features (e.g., size, shape, intensity) or binary relational features (e.g., spatial relationships). Thus, the objects and their relationships represented in the semantic network are transformed into a directed graph of Solution Elements on the Blackboard, with the direction of the link reflecting the dependency of an object’s attribute upon another object.

Solution Elements are processed sequentially by agents. Objects are recognized in order based on the directed links between their Solution Elements. A Scheduling Agent determines the order dynamically - after one Solution Element is processed, the next is then determined. The Scheduling Agent selects the Solution Element with the highest percentage of features that can be computed, i.e., where prerequisite Solution Elements have already been processed. This approach is flexible and opportunistic, it can automatically handle any set of Solution Elements; and if a particular object is not present or not found in a particular image, the ordering can change without requiring hard-coded exception rules.

When a particular Solution Element is scheduled for processing, a Reasoning Agent computes an image search area using spatial inferencing from the relationships to previously recognized objects. This search area is provided as a mask to guide a Segmentation Agent that generates candidate image regions for the object. Segmentation is typically performed by a DNN agent that generates multiple connected components as candidate regions. Feature values are computed for each candidate region and compared against the expected values by a Reasoning Agent. Thus, by pattern classification, the candidate that best matches these expectations from the knowledge base is selected.

Agent types currently provided in SimpleMind and shown in Figure 1 include:

- Knowledge Agents: to transform the knowledge in the semantic network into Solution Elements on the Blackboard and attributes to feature space
- Scheduling Agents: to determine which Solution Element should be processed next, based on graph dependencies as described above
- Segmentation Agents: including intensity thresholding, mathematical morphology, and deep convolutional neural networks
 - The DNN agent will perform training to learn the weights if they do not already exist
- Reasoning Agents: to (1) derive search areas for objects (to guide Segmentation Agents), and (2) evaluate candidate image regions and select the best match to an object based on features.

Agents are activated iteratively, one at a time. At each iteration the system computes an activation score for each registered agent. The agent with the highest score is activated and can contribute to the solution on the Blackboard. Each agent provides a function to compute its activation score based on the contents of the Blackboard, in particular whether the Solution Element being processed has the relevant attributes and necessary data required by the agent. The process repeats until all activation scores are zero and no further agents activate. The system control is simple, yet highly flexible, with agent priorities determined through their activation functions.

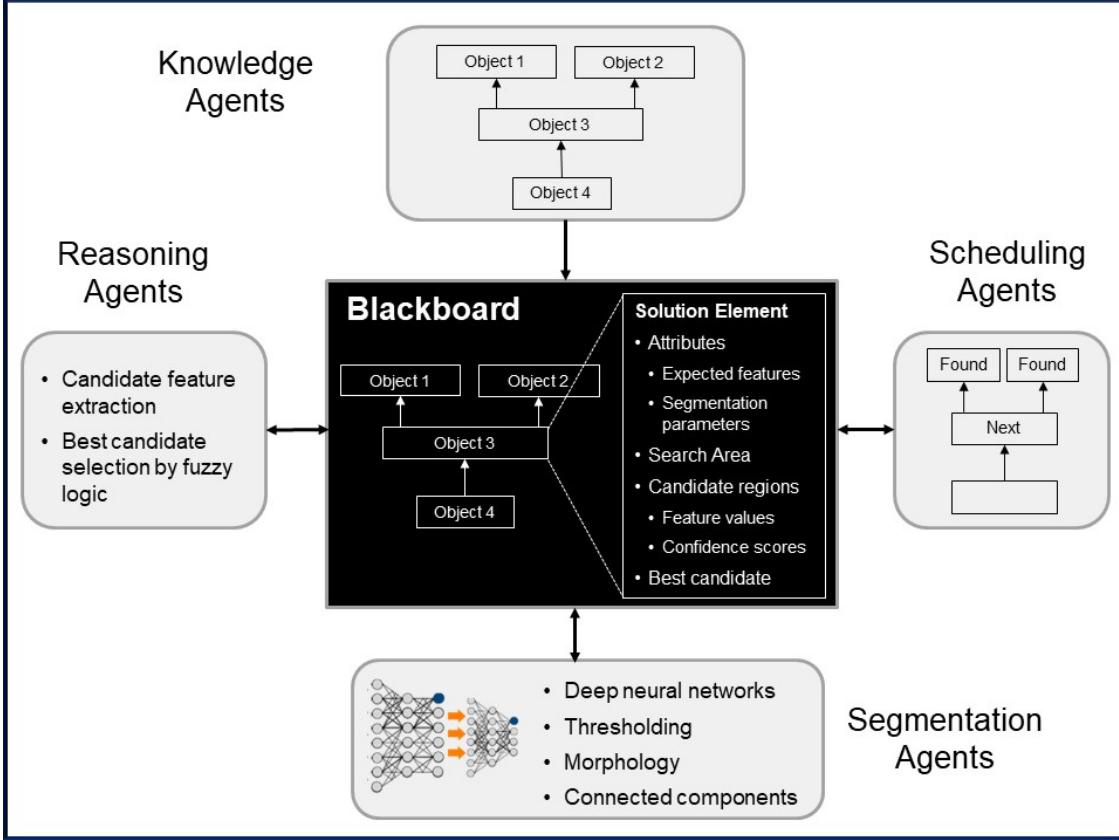


Figure 1: SimpleMind multi-agent vision architecture showing agent types and the Blackboard for sharing of results during image understanding.

The contents of the Blackboard reflect what SimpleMind is thinking at any point in time and its current understanding of the image. Once all Solution Elements have been processed, the Blackboard contains an instantiation of the general knowledge base to a particular image. The object attributes from the general knowledge base are now instantiated with actual numerical feature values from the image, from the particular patient anatomy in the case of medical imaging, enabling further high-level reasoning.

4 Application Examples

We provide three example SimpleMind applications that demonstrate the knowledge base and how the agents make use of it. In particular, we describe how thinking using the knowledge base improves deep neural networks.

4.1 Kidney Segmentation on CT Images

State of the art kidney segmentation on CT uses DNNs with ad hoc connected component post-processing. Although performance is good, there is an opportunity for improvement with increased reliability by applying anatomical knowledge using SimpleMind. This knowledge is used to: (1) improve discrimination of adjacent abdominal organs with similar image intensities, (2) select the best connected component based on size and relative location.

The knowledge base provides the following basic relevant information about the kidneys:

- The kidneys have a soft tissue image intensity range of 300 - 2000 Hounsfield Units (HU) on contrast-enhanced CT scans
- The kidneys do not overlap spatially with bone (they are adjacent and of similar intensity in contrast-enhanced CT scans)
- The right kidney is 5 - 10 cm to the right of the spine

- The left kidney is 5 - 10 cm to the left of the spine
- The right and left kidneys are at the same craniocaudal level in the abdomen

This kidney knowledge is reflected as attributes in the knowledge base; it includes relationships to the abdomen, bone, and spine. As such, these objects are also described as nodes in the semantic network (SN) shown in Figure 2.

The SimpleMind SN is stored as human readable text files, with a “Node List” file being the entry point that lists the nodes in the SN. The attributes of each node are then stored in a text file. Figure 2 shows the SimpleMind SN for kidney CT segmentation. This kidney SN Node List file is shown in Figure 2 (a). Figure 2 (b) - (d) are example node files that contain the attributes describing general knowledge about an object. For relational attributes, the node name of the related object is given. A visualization of the relational links between nodes in the kidney SN is shown in Figure 2 (e).

The abdomen node in (b) describes the attributes of the abdomen as imaged on CT:

- Line 2 gives the CT HU intensity range of [-400, 2000] (which can guide intensity thresholding)
- Line 5 gives the average cross-sectional area of the abdomen tissue within the HU threshold range; since this varies between patients, it is represented using a fuzzy membership function that assigns a confidence [0.0, 1.0] to each area value; the membership function is piecewise linear with vertices specified. In this case, the confidence is 0 at 100 cm² or less, with a linear progression towards a full confidence of 1.0 at 500 cm² or more.

Dense_bone and *spine* nodes have similar attributes describing their image intensity range and cross sectional area. The *kidney_cnn* node invokes a deep convolutional network to provide an initial segmentation that generates candidates for both the left and right kidneys; some of these candidates are expected to be the actual kidneys but may also include detections of incorrect stray regions.

The *kidney_left_init* node demonstrates the use of relational attributes to reliably identify the left kidney among the DNN connected components:

- Line 3 indicates that it is part of the *kidney_cnn* output
- Line 4 that it is not part of bone (the DNN occasionally includes bone within its initial segmentation since they have similar intensity in contrast-enhanced CT scans)
- Line 5 that it is left of the spine (typically 50 - 100 mm)
- Line 6 that it is at the same level anatomically as the right kidney in the longitudinal or z direction (typically within 3 cm of each other)

Machine reasoning is applied using these attributes to ensure that a candidate is selected that is in the appropriate location relative to the spine and right kidney. Following segmentation of *left_kidney_init*, there is a dependent node, *kidney_left*, that includes attributes relating to contiguity and smoothness that trigger hole filling and morphological smoothing operations to achieve the final result.

Figures 3 shows results from the SimpleMind kidney application. The output from the DNN alone (the *kidney_cnn* node) in Figure 3c shows an obvious (dumb) mistake with the DNN including an additional region that is disconnected from the kidneys. By applying the anatomical knowledge described above with machine reasoning, SimpleMind is able to separately identify the right and left kidneys in the *kidney_right* (Figure 3d) and *kidney_left* (Figure 3e) nodes, respectively, while excluding the extraneous region initially segmented by the DNN. This demonstrates the benefit of adding reasoning to improve DNN reliability.

(a) Node List file of the semantic network for kidney (kidney SN)

```
Model: kidney_model
1
kidney_right
kidney_right_init
kidney_left
kidney_left_init
kidney_cnn
spine
spine_init
dense_bone
abdomen
End: kidney_model;
```

(b) "abdomen" node file

```
AnatPathEntity: abdomen;
HRange -400 to 2000;
UseSubsampledImage 4 4 2;
DoNotFormCandsWith_NumVoxelsLessThan 1000;
Area_PerXYplane_cm2 [(100, 0) (500, 1)];
End: abdomen;
```

(c) "kidney_left_init" node file

```
AnatPathEntity: kidney_left_init;
DoNotFormCandsWith_NumVoxelsLessThan 500;
PartOf_E kidney_cnn;
NotPartOf dense_bone;
LeftOf_PlanarCentroid_mm spine [(0,0) (50,1) (100,1) (150,0)];
NearZ_Centroid_mm kidney_right_init [(-100,0) (-30,1) (30,1) (100,0)];
Volume_cc [(0,0) (150,1)];
End: kidney_left_init;
```

(d) "kidney_right_init" node file

```
AnatPathEntity: kidney_right_init;
DoNotFormCandsWith_NumVoxelsLessThan 500;
PartOf_E kidney_cnn;
NotPartOf dense_bone;
RightOf_PlanarCentroid_mm spine [(0,0) (50,1) (100,1) (150,0)];
NearZ_Centroid_mm kidney_left_init [(-100,0) (-30,1) (30,1) (100,0)];
Volume_cc [(0,0) (150,1)];
End: kidney_right_init;
```

(e) The relational links between nodes in the kidney SN

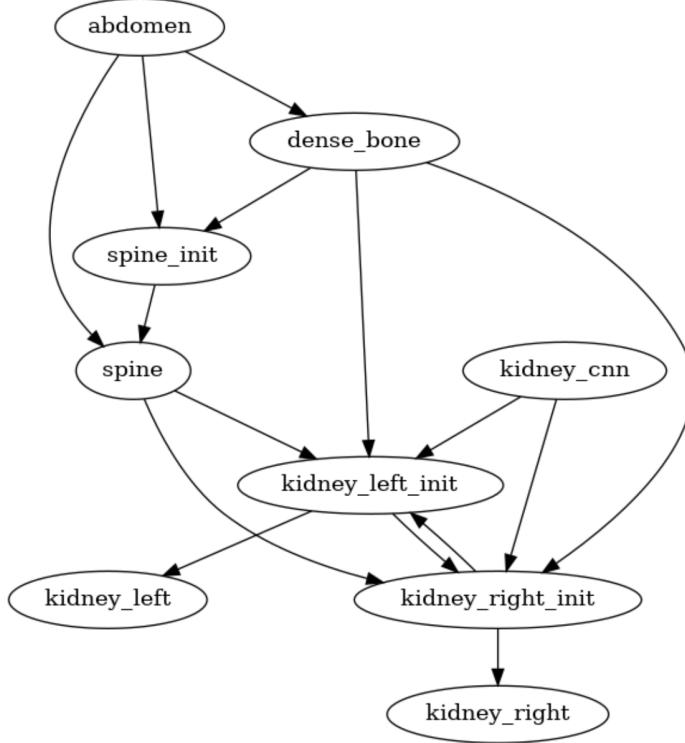


Figure 2: The human readable text files of the SimpleMind semantic network (SN) for kidney segmentation. (a) The “Node List” text file of the SN where each line corresponds to a node in the SN. (b) The text file for the thorax node, containing the attributes for recognizing the thorax object. (c) The text file for the *kidney_left_init* node. (d) The text file for the *kidney_right_init*. (e) Graph visualization of the kidney SN.

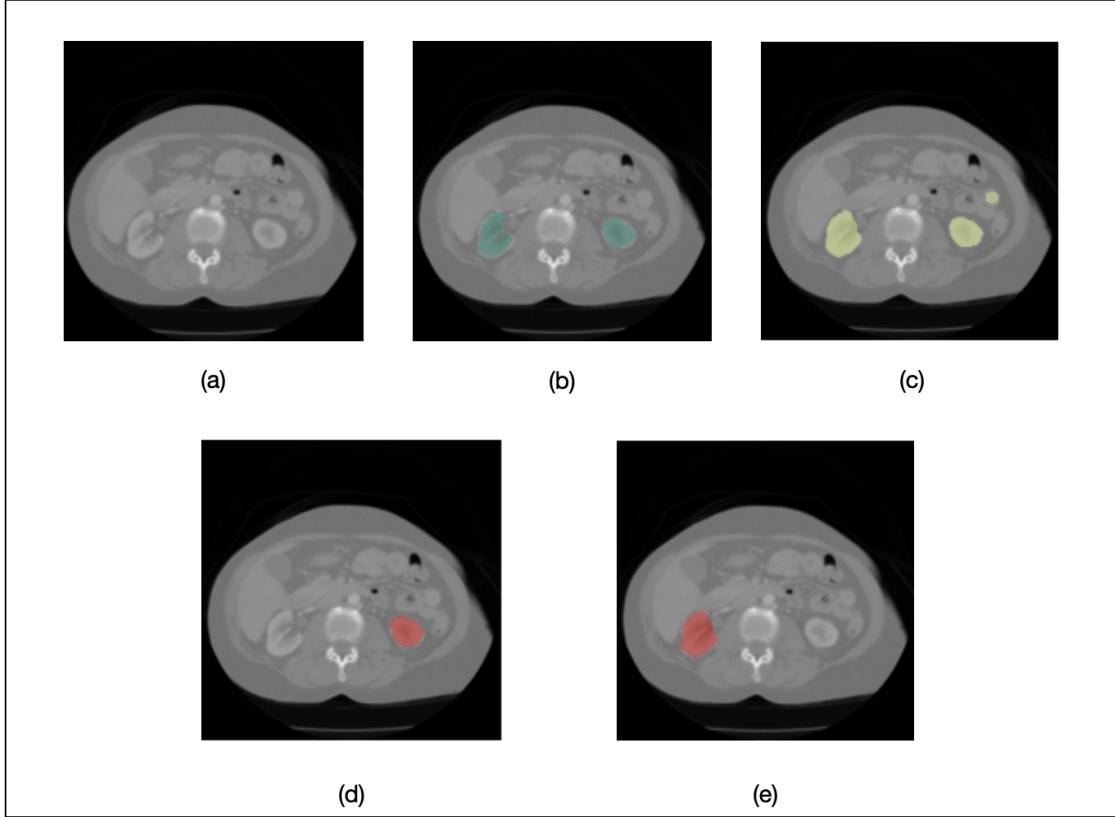


Figure 3: Kidney segmentation on CT: (a) original image; (b) reference kidney segmentation (green overlay); (c) segmentation result for the *kidney_cnn* node using a DNN only; (d) SimpleMind segmentation of the *right_kidney* node (red overlay); (e) SimpleMind segmentation of the *left_kidney* node (red overlay)

4.2 Prostate Segmentation in 3D MRI

Prostate segmentation in 3D T2-weighted MRI can be accomplished using a DNN, with performance being significantly influenced by application-specific preprocessing algorithms with data scientist hand-tuning. There is room for improvement by incorporating human heuristic knowledge using SimpleMind to overcome: 1) the relatively low contrast in the apex/base region of the prostate gland compared to the central region, 2) the presence of an endorectal coil in some patients that causes shape distortions in the prostate base and significant intensity histogram changes.

We built the SimpleMind prostate knowledge base to combine multiple DNNs for segmenting the center, apex, and base regions of the prostate and adding thinking to each DNN. The knowledge base mimics the following human expert behavior to compensate for the segmentation challenges described above:

- Experts adjust the window/level settings (intensity remapping) differently when viewing the apex and base regions to improve the perceived contrast based between prostate and background.
- Experts infer the shape of unclear or incomplete boundaries by considering adjacent slices and knowledge of the overall shape of the prostate.

This processing based on human expert knowledge is reflected in attributes in the prostate semantic network (prostate SN) shown in Figure 4. The prostate SN comprises of the Node List file in Figure 4 (a), and Figure 4 (b) - (d) are example node files for the image objects for related to the prostate apex region. The *prostate_apex_focused_cnn* node in Figure 4 (c) includes attributes representing the learning hyper parameters of a DNN. Figures 4 (b) and (d) show nodes that are used to apply thinking to the input and output of the DNN node *prostate_apex_focused_cnn*. *Prostate_apex_box* specifies the image subregion that provides the histogram for normalization of the DNN input image. *Prostate_apex_attention* refines and cross-checks the DNN output. A visualization of the relational links between nodes in the prostate SN is shown in Figure 4 (e). The knowledge base employs multiple DNN nodes to mimic the human behaviors in applying specific processing to the apex and base regions of the prostate gland.

The *prostate_apex_box* node in Figure 4 (b) describes a box-shaped region that contains the prostate apex:

- Line 3 describes a box that is 100 x 100 mm in plane and 5 to 25 mm above the center slice of the prostate, which is in turn derived from the initial approximate segmentation provided by the *prostate_whole_cnn* node

The *prostate_apex_focused_cnn* node trains a DNN specifically focused on the prostate apex using the following attributes:

- Line 2 bases input image normalizations on the local histogram of the *prostate_apex_box*, mimicking the region-specific window/leveling (contrast enhancement) observed with human experts
- Lines 4 - 11 specify input image normalization methods with tunable parameters that make use of the local histogram
- Line 12 specifies the tunable learning rate parameter for training DNN weights.

The *prostate_apex_focused_cnn* node has additional attributes not shown in the figure for DNN learning schemes, such as batch size, DNN weight optimizer, etc. These parameters are refined and optimized for the apex region as the system learns from data (see Section 5).

The *prostate_apex_attention* node describes the attributes of the prostate apex region:

- Line 3 indicates that it is within the *prostate_apex_box*.
- Line 4 indicates that it is further part of the *prostate_apex_focused*, which is the smoothed output of the *prostate_apex_focused_cnn*.

These attributes demonstrate spatial reasoning between different objects to refine and cross-check regions.

Similar nodes were created for the prostate base region to add similar thinking to the *prostate_base_focused_cnn* node. Then, the prostate SN combines regions from the apex/base- and center-focused predictions just as the human expert seamlessly integrates their visual processing of each region.

Figures 5 - 6 show results from the SimpleMind prostate application. Figure 5 demonstrates SimpleMind adding thinking to the preprocessing of DNN inputs. Figure 5a shows an original axial MR image slice, 5b shows the segmentation of the *prostate_apex_box* node as a red overlay on a coronal view, and 5c shows the preprocessed DNN input image used by the *prostate_apex_focused_cnn* node, i.e., normalized based on the local histogram of the *prostate_apex_box*. Normalizing based on the local histogram of the *prostate_apex_box* node enhances contrast between the prostate apex and background. For comparison, Figure 5d shows the normalized input image for the *prostate_whole_cnn* node having much less contrast. Thus SimpleMind is able to describe and make use of expert human knowledge akin to choosing a window/level setting specific to the prostate apex that is different than for the central prostate region.

Figure 6 shows an example of prostate segmentation by SimpleMind. Figure 6a shows the original axial MR image slice and 6b the reference prostate segmentation overlay with a green contour. Figure 6c shows the *prostate_whole_cnn* segmentation that includes a dumb mistake by the DNN in including a discontiguous region as part of the prostate. When recognizing the *prostate_whole* node, a SimpleMind reasoning agent uses the *prostate_whole_cnn* segmentation results and selects a candidate region that best matches the expected size and location attributes, yielding the final prostate segmentation result shown in Figure 6d and demonstrating the ability of SimpleMind to think about DNN outputs.

(a) The text file entry point of the semantic network for prostate segmentaion.

```
Model: prostate_model
1
prostate_final
prostate_combine
prostate_base_attention
prostate_apex_attention
prostate_center
prostate_base_focused
prostate_base_focused_cnn
prostate_apex_focused
prostate_apex_focused_cnn
prostate_center_box
prostate_base_box
prostate_apex_box
prostate_center_slice
prostate_whole
prostate_whole_cnn
End: prostate_model
```

(b) "prostate_apex_box" node text file

```
AnatPathEntity: prostate_apex_box;
IncludeAllVoxels;
NearCentroid_Range_mm prostate_center_slice -50 50 -50 50 -25 -5;
End: prostate_apex_box;
```

(c) Some part of "prostate_apex_focused_cnn" node text file

```
AnatPathEntity: prostate_apex_focused_cnn;
IntensitySubregion prostate_apex_box;
NeuralNetKeras custom_architecture.py/unet_3d_prostate_custom 320 320 24;
NeuralNet_Normalization 0 2 1 {28, 28, 0, 1} none n4biascorr;
NeuralNet_Normalization 0 2 1 {29, 29, 0, 1} mean0_std1 minmax;
NeuralNet_Normalization 1 3 2 {30, 31, 0, 3} no_channel none n4biascorr;
NeuralNet_Normalization 1 3 1 {32, 32, 0, 1} none centile 20 {33, 34, 10, 40} 80 {35, 36, 60, 90};
NeuralNet_Normalization 1 2 1 {37, 37, 0, 1} mean0_std1 minmax;
NeuralNet_Normalization 2 3 2 {38, 39, 0, 3} no_channel none n4biascorr;
NeuralNet_Normalization 2 2 1 {40, 40, 0, 1} none clahe 0.06 {41, 42, 0.01, 0.07} 64 {43, 43, 64, 128};
NeuralNet_Normalization 2 2 1 {44, 44, 0, 1} mean0_std1 minmax;
NeuralNet_LearningRate 1.0 -2 {45, 46, -5, -2};
```

(d) "prostate_apex_attention" node text file

```
AnatPathEntity: prostate_apex_attention;
PartOf_E prostate_apex_focused;
PartOf_E prostate_apex_box;
End: prostate_apex_attention;
```

(e) The visualization of the semantic network for prostate segmentation

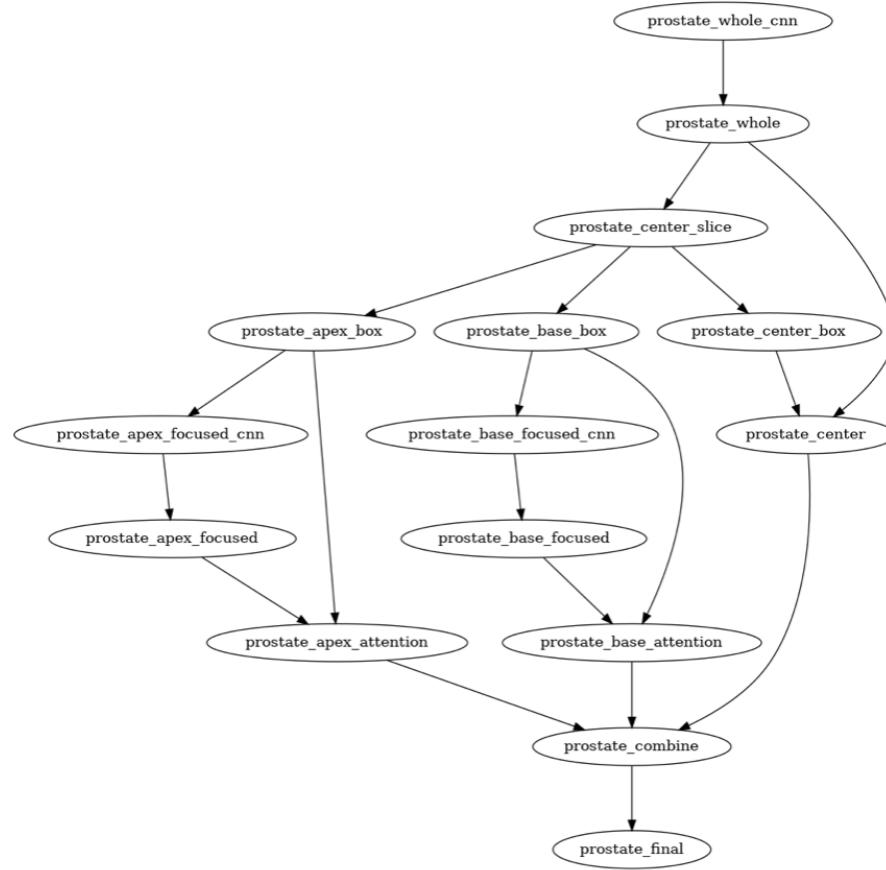


Figure 4: The SimpleMind semantic network (SN) for prostate segmentation. The SN of a Node List file listing the nodes in the SN and the corresponding set of human readable node files. (a) The prostate SN Node List file; (b) The *prostate_apex_box* node file; The node file is derived with the attributes for recognizing the apex area slices from the initial prediction from the *prostate_whole_cnn* node. (c) The *prostate_apex_focused_cnn* node file. (d) The *prostate_apex_attention* node file. (e) Graph visualization of the prostate SN.

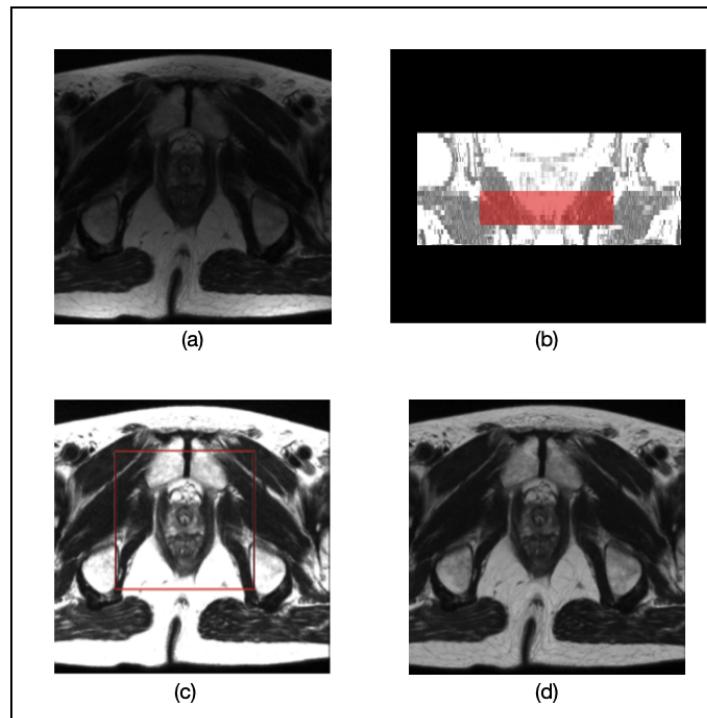


Figure 5: (a) original image; (b) The segmented *prostate_apex_box_node*; (c) preprocessed image for input to the DNN in the *prostate_apex_focused_cnn* node (red box indicates the guided area for the local histogram generated from the *prostate_apex_box_node*); (d) preprocessed image used for the DNN in the *prostate_whole_cnn* node

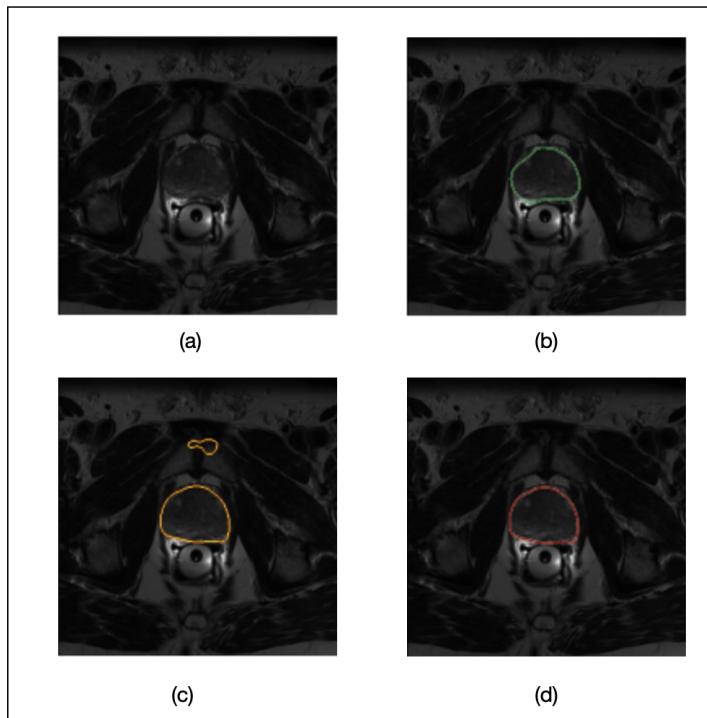


Figure 6: (a) original image; (b) prostate reference segmentation (green contour); (c) segmentation of the *prostate_whole_cnn* node (yellow contour); (d) SimpleMind segmentation of the *prostate_whole_node*.

4.3 Endotracheal Tube Assessment on Chest X-Ray

Endotracheal tubes (ETTs) are used to maintain airway patency and lung ventilation. They must be placed correctly to avoid severe complications or even death. Chest x-rays (CXRs) are taken frequently in the intensive care unit to check tube positioning. The medical literature states that the tip of the endotracheal tube should be 5 ± 2 cm above the carina, where the trachea bifurcates into the two main stem bronchi.

Accurate ETT segmentation is challenging due to its narrow diameter, low contrast, and multitude of overlying edges in the CXR image. While a DNN alone can achieve promising results, there are challenges in: (1) acquiring sufficient examples of misplaced tubes to train and test the system (it occurs infrequently in practice), (2) lack of explainability of how the DNN determines the tube is misplaced, (3) the high level of performance and reliability required in detecting not only the ETT but also the anatomic landmarks for checking its position, given that an error can be fatal. SimpleMind tackles these issues using a knowledge base that describes not only the ETT but also the anatomic landmarks and includes relational attributes to cross check multiple DNNs and ensure consistency and overall reliability of the system. Rather than attempting to learn misplacement of the tip indirectly from examples, the SimpleMind knowledge base can directly describe when an alert should be given based on the tip location relative to the carina.

The semantic network shown in Figure 7 includes DNNs for the trachea (*trachea_cnn*), carina (*carina_cnn*), and ETT (*et_tube_1_cnn* and *et_tube_2_cnn*). It defines a “safe zone” for the ETT tip using spatial concepts:

- part of the trachea: Line 3 of the *et_zone_1* node (Figure 7b)
- 3 - 7 cm above the carina: Line 4 of the *et_zone_1* node - based on the y-coordinate of the centroid (Figure 7b)
- ETT tip must be inside the safe zone: *et_tip_correct* node describes this relative to the *et_zone* node and represents the state of the ETT tip (Figure 7c)
- the ETT path must be within the trachea (and thus not going into the esophagus): *et_path_incorrect* node describes this relative to the *trachea* node and represents the state of the ETT path
- for the ETT position to be correct the two criteria above must be met: this requirement is defined in the *et_tube_correct* node which represents the final decision of the system based on its machine reasoning (Figure 7d)

The model also demonstrates checking of DNN outputs for consistency. The *carina_cnn* outputs a coordinate for the position of the carina, represented by the *carina_1* node. The carina location can also be derived from the inferior portion of the trachea where it branches into the two main stem bronchi, represented by the *carina_2* node. The *carina_3* node indicates that these two should correspond and refines the final result. Accurate detection of the carina is necessary for ETT position checking. Crucially, if the alternate carina locations do not correspond, then the system will report that it is unable to reliably identify the carina rather than outputting an incorrect result. Using knowledge to identify interpretation errors is an important benefit of machine reasoning that allows the system to determine when it is likely to be wrong rather than failing silently.

Figures 8 - 10 show results from the SimpleMind chest x-ray ETT application. 8a shows an original chest x-ray image, 8b shows the *trachea* region overlaid on an enhanced image, 8c shows the ETT safe zone (*et_zone*) and tube tip location (*et_tip*), and 8d is the final output of the system as presented to the ICU physician. The enhanced image is preprocessed specifically for the DNNs; however, human observers also indicated that they find the image useful and hence the AI output is presented as an overlay on this image. Figure 8c shows that the system identified the ETT tip as within the safe zone, explaining why it thinks the position is correct, and the green color coding of the overlay in Figure 8d indicates to the physician that the AI has identified the tube as correctly placed. Figure 9 shows an example of incorrect tube placement with the red overlay output indicating an alert. Figure 9c shows the ETT tube tip outside the safe zone (tip too low relative to the carina). Figure 10 also shows incorrect tube placement with the tube tip too high relative to the carina (Figure 10c).

(a) The text file entry point of the semantic network for ET tip placement.

```

Model: cxr_model
10
et_tube_correct
et_path_incorrect
et_tip_correct_1
et_tip_correct
et_tip
et_tube
et_tube_2
et_tube_2_cnn
et_tube_2_input
et_tube_1
et_tube_1_cnn
et_zone
et_zone_1
carina
carina_3
carina_2
carina_1
carina_cnn
carina_input
trachea_top
trachea_dilated
trachea
trachea_cnn
End: cxr_model

```

(b) "et_zone_1" node text file

```

AnatPathEntity: et_zone_1;
IncludeAllVoxels;
PartOf_E trachea;
NearY_RangePlanar_mm_E carina -70 -30;
RetainCandidates;
End: et_zone_1;

```

(c) "et_tip_correct" node text file

```

AnatPathEntity: et_tip_correct;
PartOf_E et_tip;
Inside_2D_E et_zone;
Area_AllXYplanes_cm2 [(0, 0) (1, 1)];
RetainCandidates;
End: et_tip_correct;

```

(d) "et_tube_correct" node text file

```

AnatPathEntity: et_tube_correct;
IncludeAllVoxels;
PartOf_E et_tip_correct_1;
NotFound et_path_incorrect;
RetainCandidates;
End: et_tube_correct;

```

(e) The visualization of the semantic network for ET tip placement

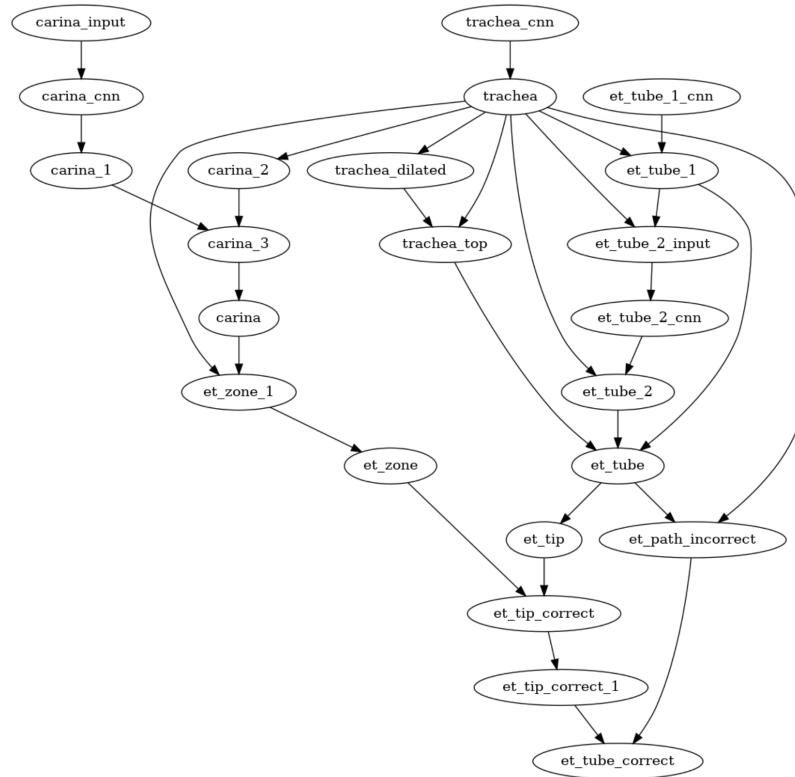


Figure 7: The SimpleMind semantic network (SN) for the ETT. The SN of a Node List file listing the nodes in the SN and the corresponding set of human readable node files. (a) The endotracheal tube SN Node List file. (b) The *et_zone_1* node file. (c) The *et_tip_correct* node file. (d) The *et_tube_correct* node file. (e) Graph visualization of the endotracheal tube SN.

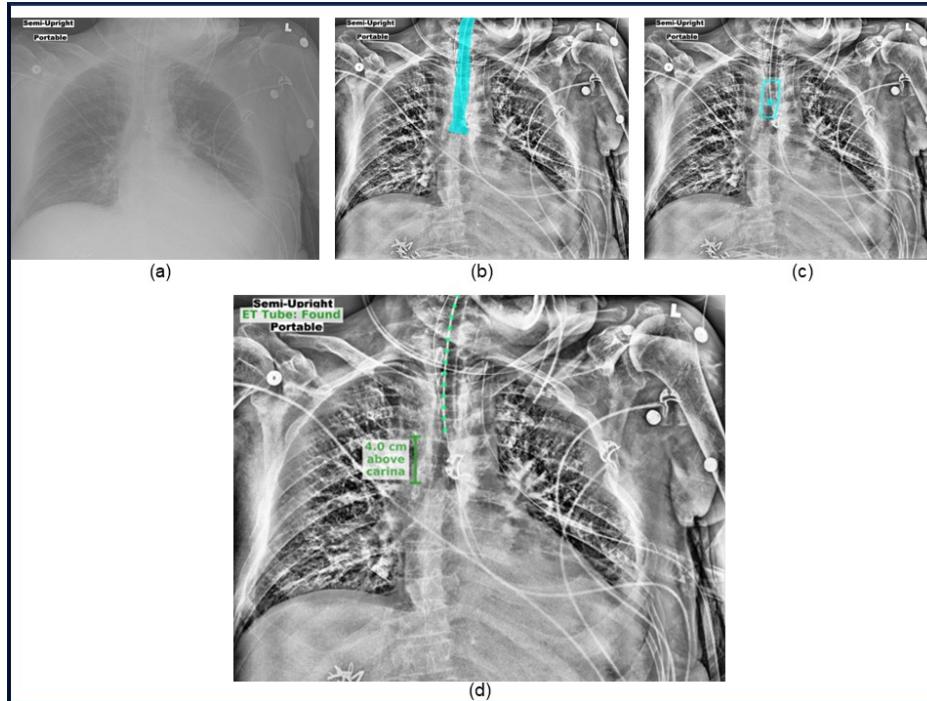


Figure 8: (a) Original chest x-ray image; (b) trachea region overlaid on an enhanced image; (c) ETT safe zone (*et_zone*) and tube tip location (*et_tip*) showing the tip within the safe zone; (d) the final output of the system as presented to the physician with the green overlay indicating the system determination of correct tube placement.

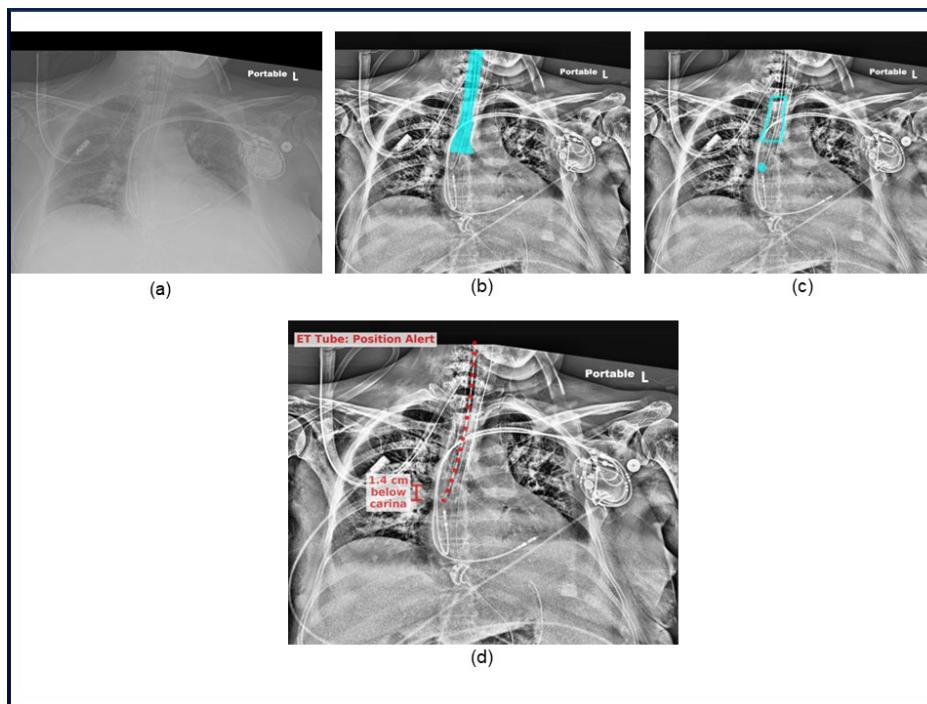


Figure 9: (a) Original chest x-ray image; (b) trachea region overlaid on an enhanced image; (c) ETT safe zone (*et_zone*) and tube tip location (*et_tip*) showing the tip outside the safe zone; (d) the final output of the system as presented to the physician with the red overlay indicating the system determination of incorrect tube placement (tip too low relative to the carina).

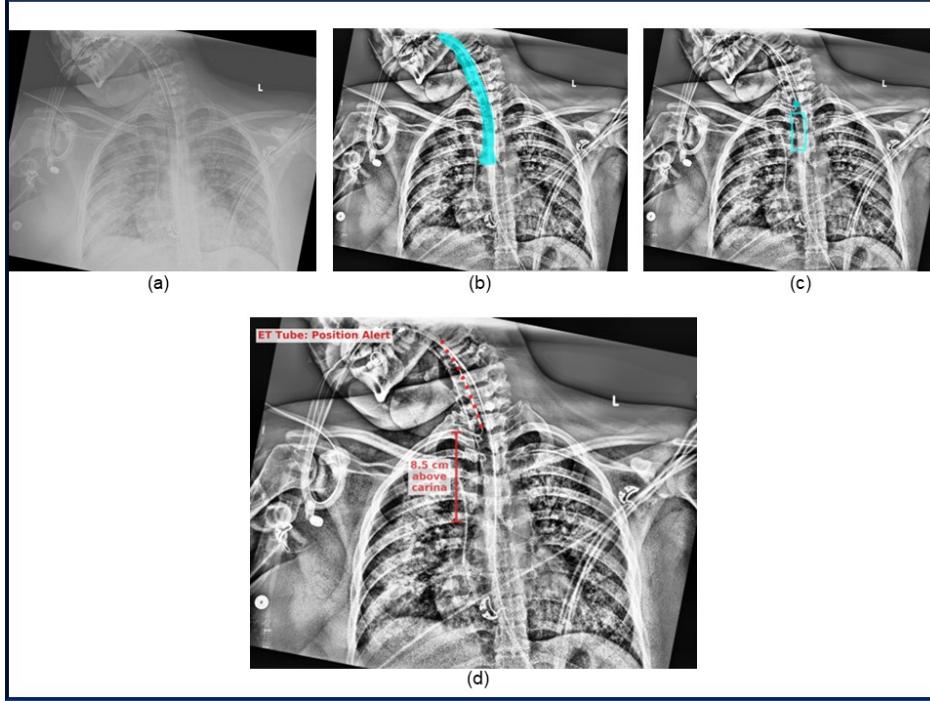


Figure 10: (a) Original chest x-ray image; (b) trachea region overlaid on an enhanced image; (c) ETT safe zone (*ett_zone*) and tube tip location (*ett_tip*) showing the tip outside the safe zone; (d) the final output of the system as presented to the physician with the red overlay indicating the system determination of incorrect tube placement (tip too high relative to the carina).

5 Learning and Optimization

A SimpleMind knowledge base can have a number of attribute parameters that determine the behavior of multiple processing agents. Intelligent behavior emerges from the interaction between agents, and parameters must be co-optimized for them to work in concert. A change in one parameter will alter the optimal setting for others within a node. Also, a SimpleMind application usually links multiple DNNs and their attribute parameters should be co-optimized since the output of one often drives the input of another. An exhaustive search for the optimal parameter combination is beyond the capacity of hand-tuning by a data scientist. In deep learning, hyper parameter tuning is usually performed manually, guided by prior experience, limited and isolated experimentation, and traditional best practices. Using hand tuning, a very small subset of the possible parameter combinations are evaluated. Even using existing AutoML tools, more emphasis is usually placed on learning hyperparameters than parameters implicit in the pre/post processing code that may nonetheless significantly influence performance. Also, existing AutoML implementations typically optimize a single DNN in isolation rather than co-optimizing multiple DNNs (Liaw et al., 2018). Allowing the co-optimization of all nodes in the knowledge (semantic) network is necessary since nodes are inter-dependent by virtue of search area dependencies and relational attributes that ensure consistency and allow cross-checking between nodes. For example, if new parameter settings in one node increase or decrease segmentation accuracy, then the parameter describing its spatial relationship to another node may need to be correspondingly updated to a more narrow or larger possible range, respectively.

The SimpleMind framework includes a Knowledge Network Learning and Optimization (KNoLO) module to assess and optimize all attribute parameters in the knowledge base and can encompass AutoML functions. It comprehensively co-optimizes all attribute parameters from all nodes simultaneously, including:

- Object expected characteristics (anatomical knowledge),
- DNN input channels and image preprocessing options,
- DNN learning hyper parameters,
- Post-processing parameters (operating on DNN outputs).

In SimpleMind optimization, a parameter set is a specific combination of values of the tunable attribute parameters in the knowledge base. With a parameter set, the knowledge base becomes fully specified and can be applied to a KNoLO tuning set, yielding a performance for the parameter set. Optimization involves evaluating multiple parameter sets and selecting the best to achieve a data-optimized (tuned) knowledge base.

In the SimpleMind framework, KNoLO is comprised of two primary modules, an “optimizer” and a “distributor”. The optimizer iteratively determines parameter sets to test, which are spawned into jobs by the distributor to compute their performance over the KNoLO tuning set. The distributor can be configured to process jobs sequentially or in parallel. The results are relayed back to the optimizer that then decides which parameter sets to evaluate next. The SimpleMind optimizer and distributor modules can be customized or replaced specific to user needs, preferences, and computing environment. If a parameter set includes tunable DNN learning parameters, SimpleMind will trigger learning of DNN weights. For large-scale optimization involving many parameter sets, parallelization is necessary and available in SimpleMind.

SimpleMind currently provides the genetic algorithm (GA) as a KNoLO optimizer. The GA is an optimization approach inspired by evolution in nature (Holland, 1992). It encodes all tunable attribute parameters into a “chromosome” where each parameter can be considered a gene that changes system behavior and performance. Each unique chromosome encodes a different parameter set. A group of chromosomes (a “population”) represents a group of parameter sets, and each individual chromosome will have a performance value (“fitness score”) when evaluated on the KNoLO tuning set. Example fitness functions include dice coefficient (in segmentation tasks), confusion matrix metrics (sensitivity, specificity, precision, recall, etc. - for classification and detection tasks), and distance metrics (e.g. mean squared error — for point detection tasks). Also fitness functions may be derived as a weighted combination of these or other implemented metrics. Based on their fitness, high-performing chromosomes (with the highest fitness scores) are selected to persist to the next iteration (“generation”). Genetic operations, like mutation of chromosome values and crossover between pairs of chromosomes, introduce variation in the population and thus exploration of parameter sets during optimization. As combinations of strong genes propagate and synergize with other complementary genes, the genetic pool evolves to improve performance as defined by the fitness function. The GA offers an explainable, intuitive approach towards optimization, and has been shown to derive innovative solutions in complex systems in nature, and can handle non-smooth functions better than gradient-based methods (Holland, 1992). The GA also allows for easy user intervention to guide evolution.

5.1 KNoLO in MR Prostate Segmentation

As described in Section 4.2, prostate segmentation can be improved by splitting into three sub-tasks: segmentation of the whole-prostate, the base of the prostate, and the apex of the prostate. The attribute parameters of each DNN, including the number of input channels, the preprocessing method and parameters, and the DNN learning hyper parameters (e.g. learning rate), can be exposed for tuning by SimpleMind’s KNoLO module. This is demonstrated in the SN node for *prostate_apex.Focused_cnn* as shown in Figure 11.

```
AnatPathEntity: prostate_apex.Focused_cnn;
IntensitySubregion prostate_apex_box;
NeuralNetKeras custom_architecture.py/unet_3d_prostate_custom 320 320 24;
NeuralNet_Normalization 0 2 1 {28, 28, 0, 1} none n4biascorr;
NeuralNet_Normalization 0 2 1 {29, 29, 0, 1} mean0_std1 minmax;
NeuralNet_Normalization 1 3 2 {30, 31, 0, 3} no_channel none n4biascorr;
NeuralNet_Normalization 1 3 1 {32, 32, 0, 1} none centile 20 {33, 34, 10, 40} 80 {35, 36, 60, 90};
NeuralNet_Normalization 1 2 1 {37, 37, 0, 1} mean0_std1 minmax;
NeuralNet_Normalization 2 3 2 {38, 39, 0, 3} no_channel none n4biascorr;
NeuralNet_Normalization 2 2 1 {40, 40, 0, 1} none clahe 0.06 {41, 42, 0.01, 0.07} 64 {43, 43, 64, 128};
NeuralNet_Normalization 2 2 1 {44, 44, 0, 1} mean0_std1 minmax;
NeuralNet_LearningRate 1.0 -2 {45, 46, -5, -2};
```

Figure 11: *prostate_apex.Focused_cnn* semantic network node showing parameters exposed for optimization relating to DNN input normalization and learning.

Examining the *Learning Rate* DNN hyper parameter, the default value is explicitly defined as:

```
NeuralNet_LearningRate 1.0 -2
```

which translates to the default value being set as $1.0 * 10^{-2}$. We can expose this parameter to be tuned by appending a “value-encoding” operator following the default value, for example,

```
NeuralNet_LearningRate 1.0 -2 {45, 46, -5, -2}
```

This value-encoding operator follows the format,

```
{encoded_bit_start_position, encoded_bit_end_position, lower_bound_value, upper_bound_value}
```

This operator encodes the possible values to be evaluated by the optimizer as alternatives to the default value of “-2”. The value is encoded in the 45th and 46th bits in the binary chromosome of the genetic algorithm, with 2 bits encoding 4 possible values. The lower bound for the 4 possible values is -5, and the upper bound is -2. The rest of the values are equidistant between the lower and upper bounds, resulting in the set of possible learning rate exponents to be $\{-5, -4, -3, -2\}$, corresponding to the chromosome bits, $\{00, 01, 10, 11\}$, respectively.

The *NeuralNet_Normalization* attributes include similar parameter value encoding operators as shown in Figure 11. These parameters control the number of input channels for the DNN model, as well as the preprocessing methods applied to those activated channels and their associated parameters. For instance, bias field correction is a preprocessing step that can be optionally applied to each channel during KNoLO. In this example, the DNN can have up to 3 input channels, with bias field correction, normalization, centile-based preprocessing, and clipping and histogram equalization, as the potential preprocessing steps.

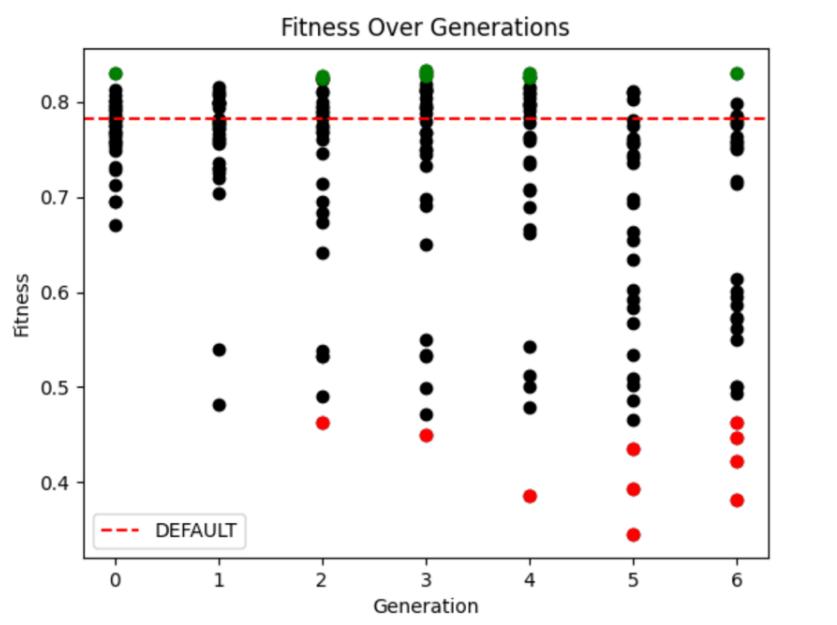


Figure 12: Evolution of fitness performance of chromosomes (i.e., different parameter sets) over time within the GA optimization in prostate segmentation. The y-axis represents the fitness score for each chromosome in a population, representing the weighted average of the prostate segmentation performance of the apex, base, and whole prostate regions, averaged across the cases of the KNoLO tuning set. For comparison, the red dotted line demarcates the performance of the prostate segmentation SN using only hand-tuned parameters by a data scientist. Each generation (iteration) has 30 chromosomes, with the green points representing the highest fitness chromosomes and the red points representing the lower fitness chromosomes. The highest fitness score was 0.833.

For each of the three neural network nodes (whole, base, apex) in the prostate SN, this set of tunable parameters (preprocessing method, number of channels, and learning rate) were exposed for tuning. Through 10 iterations (generations with population size 30) within KNoLO, approximately 300 different parameter sets were tested on a tuning set of 10 cases (with the DNNs trained on a separate 30 cases). Figure 12 visualizes the optimization by the

	Channel 1	Channel 2
<i>prostate_whole_cnn</i>	Bias Field Correction Min-Max Normalization	Bias Field Correction, Clipping & Histogram Equalization
<i>prostate_apex_focused_cnn</i>	Bias Field Correction Min-Max Normalization	Bias Field Correction Min-Max Normalization
<i>prostate_base_focused_cnn</i>	Bias Field Correction Min-Max Normalization	-

Table 1: Image preprocessing steps selected by the KNoLO optimizer for each DNN input channel for the three prostate regions.

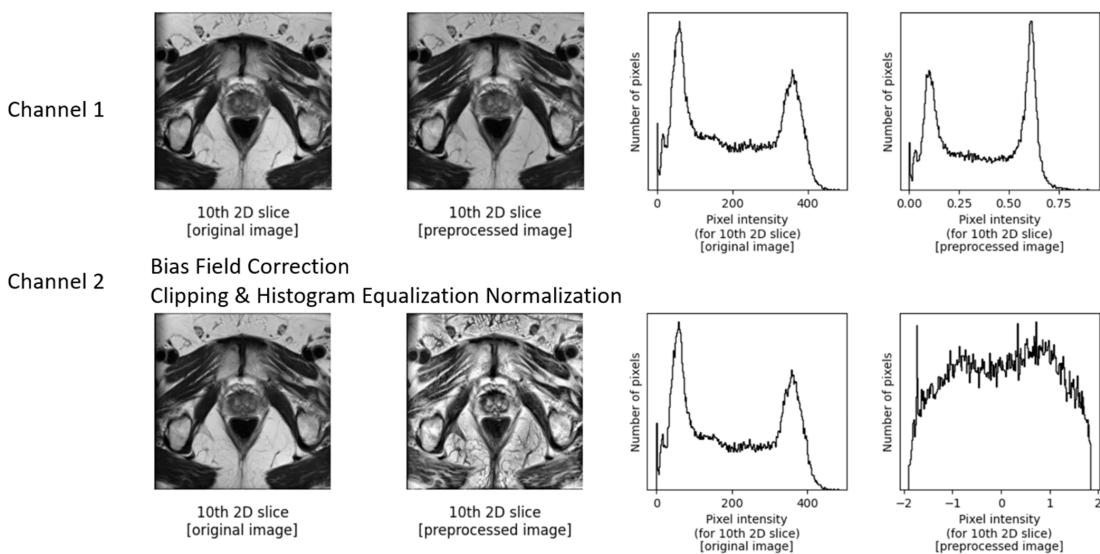
genetic algorithm with fitness scores for generations 3 through 10. For comparison, the red dotted line demarcates the performance of the prostate segmentation SN using only hand-tuned parameters by a data scientist, which was exceeded by the auto KNoLO optimization.

It was found that within the first 10 generations, the best performing parameter set included parameters that set the various preprocessing steps with multiple input channels across the DNN nodes to be a combination of bias-field correction, min-max normalization, and clipping and histogram equalization. (see Table 1). Figure 13 displays the images and pixel intensity histograms before and after these preprocessing steps. After GA optimization, the *prostate_whole_cnn* had two channels, one using bias field correction and min-max normalization; the *prostate_apex_focused_cnn* used two channels with bias field correction filters followed by min-max normalization; and the *prostate_base_focused_cnn* used one channel with min-max normalization applied as preprocessing. For both the apex and base DNNs, preprocessing was done based on a focused, localized region immediately surrounding the prostate region (derived from the *prostate_whole_cnn*), to provide normalization and preprocessing more specific to the local area of the prostate. The optimal learning rates for these input channels were, 0.01 (*prostate_whole_cnn*), 0.0001 (*prostate_apex_focused_cnn*), and 0.001 (*prostate_base_focused_cnn*). Together, this led to a fitness of 0.833, performing with an average dice coefficient score of 0.842 in segmenting the overall prostate across the KNoLO tuning set — and surpassing the hand-tuned fitness score of 0.78. It is significant to note that the learning rates were co-optimized with the image preprocessing; different input images may require different learning hyperparameters for optimal performance.

Whole Prostate CNN

Bias Field Correction

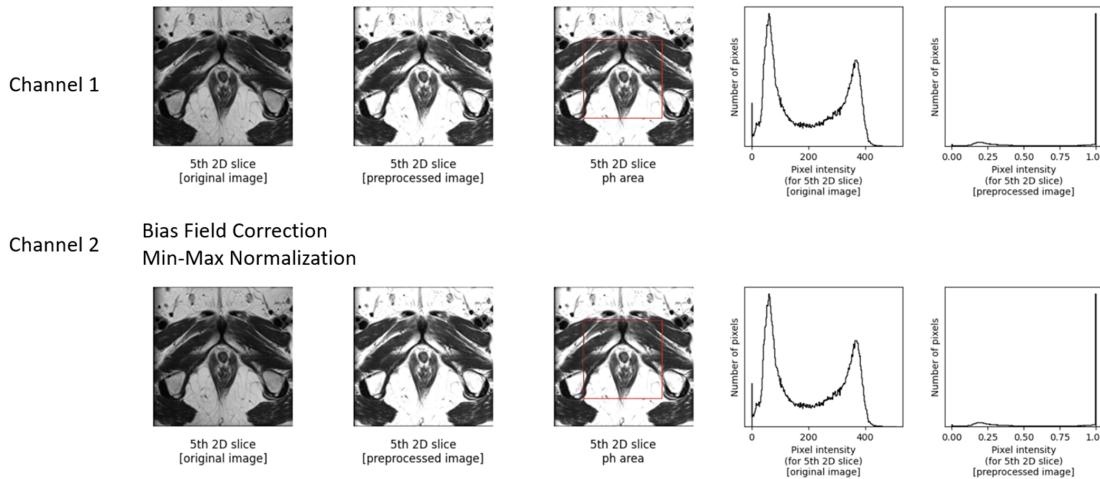
Min-Max Normalization



Apex CNN

Bias Field Correction

Min-Max Normalization



Base CNN

Min-Max Normalization

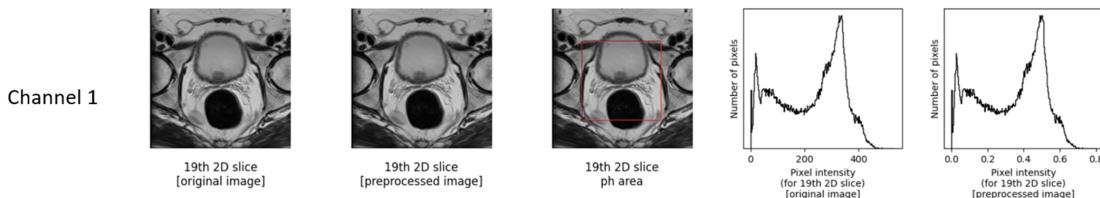


Figure 13: Example 2D image slices and their associated pixel intensity histograms before and after preprocessing.

6 Open Source Implementation

The open-source SimpleMind framework is hosted on the public GitLab repository (<https://gitlab.com/sm-ai-team/simplemind>). It supports Docker installation with the official Docker image from the public DockerHub repository. The SimpleMind framework provides SimpleMind applications for several public datasets and a software development toolkit (SDK) to build a new SimpleMind applications. It integrates existing open-source frameworks within agents (e.g., the DNN agent uses Tensorflow) and is open for users to incorporate other open-source frameworks (e.g. PyTorch, NVIDIA Clara, nnUNet) or custom algorithms.

SimpleMind applications can be trained and run through an easy-to-use python API. A Think Module runs a SimpleMind application with an input image and knowledge base. A Learn Module includes the KNoLO and manages the training process. It provides optimization of SimpleMind SN parameters and supports parallelized GPU/CPU computing. An analysis toolkit is also provided to easily track intermediate results of SimpleMind SN optimization (parameter search progression, optimal parameters, recognized image objects, computational resource management, etc). Altogether, the SimpleMind framework enables a comprehensive Cognitive AI approach to better address complex problems requiring both data-driven pattern-recognition and knowledge-driven conceptual rules, with knowledge network parameter co-optimization.

6.1 Think Module

The SimpleMind framework includes a Think Module that executes a SimpleMind application through a Python API with arguments:

- a SimpleMind knowledge base (semantic network)
- an input image
- (optionally) a specific set of knowledge base attribute parameters (chromosome)

Executing a SimpleMind application yields segmented objects from the input image, corresponding to nodes in the SN, and Blackboard contents that describe the thinking as SimpleMind processed the image. The Think Module will read inputs, build a Blackboard from the SN (using a Knowledge Mapping agent), iteratively run Scheduling, Segmentation, and Reasoning agents, and write the Blackboard and segmented image regions to output files.

SimpleMind provides a **Blackboard Viewer** to visualize the results contained in each Solution Element as overlays on the input image. It can show all candidate regions and the best candidate selected for the node. A **Blackboard Summary Tool** generates an html report of the multiple blackboards from a set of input images, including key images with segmentation results overlaid.

6.2 Learn Module

The Learn Module encompasses tools for creating semantic networks as well as the KNoLO functionality, including tuning of all SN parameters and training of DNN weights. When creating a semantic network, **SN Summary Tools** provide 1) a visualization of the relational links between nodes. Figure 2 (e), Figure 4 (e), and Figure 7 (e) were generated by this tool; 2) a text summary of the chromosomes and exposed parameters tunable with KNoLO.

As seen in the prostate KNoLO example, KNoLO can involve the computation of many hundreds (or sometimes thousands) of parameter sets, and processing each parameter set may involve executing or training multiple SN nodes (potentially including multiple DNNs). For optimization of a knowledge base to be feasible, parallelization of processing is necessary. This is enabled via the KNoLO Distributor, with current support for HTCondor (Erickson et al., 2018) and further support for other job distribution systems (Airflow, Kubernetes) in development. When automatically tuning parameters, a **KNoLO Status Summary Tool** provides an analysis of performances for chromosomes processed with KNoLO in csv and figures and HTML report format. The KNoLO Status Summary tool help users track the overall process of automatic parameter tuning within the KNoLO process.

7 Discussion

Cognitive AI is broadly defined as enabling human level reasoning and intelligence and more recently there is an emerging interest in the area of “neuro-symbolic” AI that seeks to bring together neural and symbolic approaches in a best-of-both-worlds scenario. A recent survey article identified a small but increasing number of publications at top tier AI conferences (Sarker et al., 2021). However, there have been few general-purpose implementations and computing architectures for neuro-symbolic AI and for computer vision specifically.

Neural-symbolic research starts from two observations: (1) the physical implementation of our mind is based on the neural system; (2) abstract thinking is based on symbol manipulation and complex symbolic data structures (like graphs, trees, shapes, and grammars) (Besold et al., 2017). However, it is unclear at present how symbolic processing emerges from neural activity, and at present there exists no viable alternative to symbolic approaches in order to encode complex tasks. A possible response is to break the modelling problem into two steps: the first, a symbolic modeling of cognitive behavior and the second, a mapping from symbolic to artificial neural network (Besold et al., 2017). SimpleMind achieves this by embedding deep neural networks within a symbolic knowledge base. It can be considered a “hybrid learning system” that brings together features from connectionism (e.g., DNNs) and symbolic AI. Four key advantages have been suggested as arising from this combined approach (Sarker et al., 2021): (1) interpretability, (2) error recovery, (3) out of distribution (OOD) handling, and (4) learning from small data. We will describe the how SimpleMind supports and improves DNNs in terms of these four advantages of neuro-symbolic AI.

7.1 Supporting and Improving Deep Neural Networks with SimpleMind Reasoning

While there have been encouraging results from deep neural networks in controlled experiments, there is a notable lack of translation into routine use in radiology clinical practice. SimpleMind supports DNNs with machine reasoning to increase real-world reliability and explainability of decisions. The neural and semantic networks store, represent, and use knowledge in different ways and are complementary. The embedding of DNNs in a SimpleMind semantic network confers three primary benefits:

1. It allows explicit knowledge to be applied systematically during pre and post processing, including to improve performance and particularly reliability:
 - Computing a search area in which to apply DNN segmentation using spatial relationships in the semantic network.
 - This has potential utility in approaches based on guiding attention (Zagoruyko and Komodakis, 2016; Li et al., 2018).
 - Invoking additional segmentation agents to refine the DNN segmentation result, e.g., using morphological operators etc.
 - Selection of the best candidate image region outputted by the DNN based on expected characteristics defined in the knowledge base, or conversely, rejection of the output if it does not meet expectations.
 - Thus SimpleMind enables reasoning on multiple detected objects to ensure consistency, providing cross checking between DNN outputs. The more of a scaffold of knowledge that exists, the more robust the overall image interpretation can be. Furthermore, it has been recognized that central to construction of a “mental model” is the formation of a combined whole representation from sets of part representations and that establishing global coherency is a key aspect of human reasoning (Besold et al., 2017).
 - The ability of SimpleMind to reject candidates for an object, and output nothing, is in contrast to DNNs that always output a result and can fail silently. It does not preclude recognition of subsequent objects based on other knowledge and avoids propagating errors. This gives SimpleMind applications more resilience in handling OOD cases and error recovery.
2. It provides a high degree of interpretability and explainability, deriving from the knowledge base and the Blackboard.
 - The knowledge base makes explicit the knowledge that was previously implicit in pre and post processing code and makes it easier to apply more knowledge intuitively.
 - The knowledge base and the initialized Blackboard make the object recognition model highly interpretable. Also, by exposing parameters in the knowledge base the result of optimization is transparent and a human can oversee the process and discover which parameters have the greatest impact on performance.
 - The human also has the ability to override the optimization of any parameter, for example, if it appears to be overfit to a data set or if they have an intuition that there is a more optimal value (which is readily evaluated within the SimpleMind framework and further optimized by KNoLO).
 - The thinking of SimpleMind as it processes an image is captured in the Blackboard. A human can know what it was thinking by reviewing the Blackboard contents.
 - The instantiated Blackboard provides a high degree of explainability as to how SimpleMind reached its decisions about the image.
3. Using a human-provided knowledge base, SimpleMind can perform object recognition with little or no training data.

- When there is insufficient data to train a DNN, other segmentation agents (e.g., intensity thresholding or edge detection) can use the knowledge base to generate initial segmentation results. Little or no training data is needed since the initial semantic network can be constructed using declarative knowledge engineering rather than machine learning. These initial results can be used with manual editing to generate the necessary training sets for DNN learning.
- Thus SimpleMind strongly supports learning from small data. When an OOD situation arises it can be added to the knowledge base and handled without training data being initially available.

These benefits are also consistent with goals of trustworthy AI according to the High-Level Expert Group on AI from the European Commission (Ala-Pietilä et al., 2020), in particular the following guidelines:

- Transparency: AI systems and their decisions should be explained.
- Technical Robustness and safety: AI systems need to be resilient with a fall back plan in case something goes wrong.
- Human agency and oversight: AI systems should empower human beings, allowing them to make informed decisions with proper oversight mechanisms.

A driver for neural-symbolic research has been stated as the “aim to create a new, neurally-inspired computational platform for symbolic computation which would, among others, be massively distributed with no central control, robust to unit failures, and self-improving with time” (Besold et al., 2017). SimpleMind can be seen as a step in this direction as it allows knowledge and agents to be added without modification of its simple control system, its semantic network adds robustness to failures in the recognition of any single node, and it provides the KNoLO module for self-improvement.

7.2 SimpleMind as a Scalable Virtual Data Scientist

The Cognitive AI implemented in SimpleMind uses a knowledge base to chain together multiple DNNs and pre/post processing algorithms and perform machine reasoning with their outputs. It also includes knowledge base learning and optimization, encompassing AutoML. A vision system typically involves the following modules: (1) image preprocessing, (2) DNN segmentation, (3) segmentation post-processing. SimpleMind allows for a knowledge-based description to drive (1) and (3), rather than ad hoc, implicit use of knowledge in computer code, and provides a rich and expandable knowledge base of methods with parameter optimization. The result is a more flexible and scalable framework.

The knowledge base is independent of image processing algorithms, with a descriptive form that is intuitive, understandable, and able to be created, expanded, and maintained, by domain experts (rather than necessarily requiring a data scientist). This allows the expert knowledge to be reused and applied to other problems, for example, analysis of images acquired by different imaging modalities. In contrast, many applications incorporate domain knowledge by embedding it implicitly within segmentation algorithms. SimpleMind allows a set of general algorithms to be applied to a wide range of tasks by simply extending a high-level descriptive model.

In the SimpleMind framework, agent inputs and outputs must respect Blackboard data structures, but the processing they perform is modular, encapsulated, and highly flexible. Agents maintain a degree of autonomy with their interactions regulated via the Blackboard and its control system. For example, a new segmentation algorithm can be developed and added to SimpleMind without any direct awareness of other agents, how knowledge is represented, or even the problem of medical imaging at all. Parameters used by agents (meta level knowledge) are derived from the knowledge base and transformed to the Blackboard (rather than hard coded implicitly). This means that they are not fixed but can be adapted to a particular problem by the KNoLO module. This is also liberating for the algorithm developer in that SimpleMind takes care of integrating, tailoring, and optimizing that algorithm for the specific task. In effect, it is able to generalize the agent skill and apply it as needed to any vision problem, a characteristic of intelligence.

Currently, the work of a data scientist training deep neural networks involves hand tuning of parameters and applying knowledge ad hoc in heuristic pre/post processing algorithms. The result is application-specific Narrow AI, that is typically suboptimal in terms of parameter search and limited as to the level of knowledge and reasoning applied. A computer programmer or data scientist uses domain knowledge when developing an image processing algorithm. However, their expectations for the image content are often reflected implicitly in the processing steps of their software. SimpleMind supports two forms of knowledge for developing AI:

1. Concept knowledge: knowledge of the scene and reasoning about the content (derived from an application expert);
2. Processing knowledge: meta knowledge about learning and image processing (normally provided by a data scientist).

SimpleMind serves as a virtual data scientist in two key ways:

1. It applies domain knowledge during image analysis - to guide segmentation and check results (typically a data scientist does this ad hoc within pre- and post- processing code)
2. It automatically tunes parameters - including deep learning hyper parameters and parameters associated with pre/post processing
 - Hand-tuning of a large number of inter-dependent parameters is infeasible, especially on an ongoing basis as more data is made available through real-world use.
 - SimpleMind co-optimization of all agent parameters provides a more extensive, unbiased search than is possible by a human data scientist.

By incorporating these activities that are typically done by a human data scientist, we achieve a virtual data scientist that is more scalable and better able to optimize than a human. Rather than tacitly applying knowledge ad hoc within coded algorithms, SimpleMind accepts high level descriptions of scene content and concepts and automatically chains together agents for image pre/post processing, DNNs, and machine reasoning. Thus SimpleMind brings transparency, efficiency, scalability, and reliability to AI development and automates tasks of the data scientist.

The SimpleMind framework has improved the performance of a number of medical imaging applications (Brown et al., 2022; Melendez-Corres et al., 2021; Choi et al., 2022), and we believe that there is strong potential utility for broader research and commercial applications in building trustworthy AI. The open source software allows for knowledge base expansion and agent aggregation by a community of developers. SimpleMind supports and improves deep neural networks by embedding them within a Cognitive AI framework.

Acknowledgement

The authors wish to thank their friends and colleagues at the UCLA Center for Computer Vision and Imaging Biomarkers.

References

- Ravi Aggarwal, Viknesh Sounderajah, Guy Martin, Daniel SW Ting, Alan Karthikesalingam, Dominic King, Hutan Ashrafian, and Ara Darzi. Diagnostic accuracy of deep learning in medical imaging: A systematic review and meta-analysis. *NPJ digital medicine*, 4(1):1–23, 2021.
- Alice C Yu, Bahram Mohajer, and John Eng. External validation of deep learning algorithms for radiologic diagnosis: a systematic review. *Radiology: Artificial Intelligence*, 4(3):e210064, 2022.
- Trishan Panch, Heather Mattie, and Leo Anthony Celi. The “inconvenient truth” about ai in healthcare. *NPJ digital medicine*, 2(1):1–3, 2019.
- Navid Hasani, Michael A Morris, Arman Rahmim, Ronald M Summers, Elizabeth Jones, Eliot Siegel, and Babak Saboury. Trustworthy artificial intelligence in medical imaging. *PET clinics*, 17(1):1–12, 2022.
- Matthew S Brown, Koon-Pong Wong, Liza Shrestha, Muhammad Wahi-Anwar, Morgan Daly, George Foster, Fereidoun Abtin, Kathleen L Ruchalski, Jonathan G Goldin, and Dieter Enzmann. Automated endotracheal tube placement check using semantically embedded deep neural networks. *Academic Radiology*, 2022.
- Gabriel Melendez-Corres, Muhammad W. Wahi-Anwar, Youngwon Choi, Heidi Coy, Steven S. Raman, and Matthew S. Brown. Accelerating training data annotation via a continuous ai-assisted, human-supervised feedback loop in kidney segmentation in ct., 2021. URL <http://archive.rsna.org/2021/704158.html>.
- Youngwon Choi, Marlena Garcia, Steven S Raman, Dieter R Enzmann, and Matthew S Brown. Ai-human interactive pipeline with feedback to accelerate medical image annotation. In *Medical Imaging 2022: Computer-Aided Diagnosis*, volume 12033, pages 741–747. SPIE, 2022.
- M. Ross Quillian. Semantic networks. In Marvin L. Minsky, editor, *Semantic Information Processing*. MIT Press, 1968.
- L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965. ISSN 0019-9958. doi:[https://doi.org/10.1016/S0019-9958\(65\)90241-X](https://doi.org/10.1016/S0019-9958(65)90241-X). URL <https://www.sciencedirect.com/science/article/pii/S001999586590241X>.
- D Corkill. Collaborating software. In *International Lisp Conference, New York*, volume 44, 2003.
- Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.
- John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
- Richard A Erickson, Michael N Fienan, S Grace McCalla, Emily L Weiser, Melvin L Bower, Jonathan M Knudson, and Greg Thain. Wrangling distributed computing for high-throughput environmental science: An introduction to htcondor. *PLoS computational biology*, 14(10):e1006468, 2018.
- Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence: Current trends, 2021. URL <https://arxiv.org/abs/2105.05330>.
- Tarek R. Besold, Artur d’Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kuehnberger, Luis C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation, 2017. URL <https://arxiv.org/abs/1711.03902>.
- Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.
- Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Tell me where to look: Guided attention inference network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9215–9223, 2018.
- Pekka Ala-Pietilä, Yann Bonnet, Urs Bergmann, Maria Bielikova, Cecilia Bonefeld-Dahl, Wilhelm Bauer, Loubna Bouarfa, Raja Chatila, Mark Coeckelbergh, Virginia Dignum, et al. *The assessment list for trustworthy artificial intelligence (ALTAI)*. European Commission, 2020.