Neurosymbolic Learning on Activity Summarization of Video Data

Steve Kommrusch SK@LEELA-AI.COM
Leela AI, Cambridge, Massachusetts, USA

Sanket Bhave SANKET.BHAVE@COLOSTATE.EDU
Mridul Banik MRIDUL.BANIK@COLOSTATE.EDU
Colorado State University

Henry Minsky HQM@LEELA-AI.COM
Leela AI, Cambridge, Massachusetts, USA

Editor: Kristinn R. Thórisson

Abstract

Neurosymbolic learning systems have been shown to quickly discover how to interact with discrete representations of the world. Symbolic learners often allow for a higher level of understandability than neural networks which learn feature vectors for actions being taken, as seen in modern reinforcement learning systems. Symbolic learners excel at learning higher-level concepts, but struggle with certain types of generalization. Symbolic learners might benefit in such situations from a learned representation of the world. This paper discusses a pipeline that uses state-of-the-art object and pose detection neural networks as input to a symbolic learning system. We show how the knowledge from the symbolic system can automatically correct object and pose data from the neural network and hence provide corrected samples that can be used to incrementally train and improve the neural network. We show how symbolic learning techniques can improve action detection when given example ground truths by humans. We also demonstrate how novel actions that are not recognized by humans might be recognized by a learning engine capable of recognizing results and preconditions for an action to be valid.

Keywords: symbolic learning, neurosymbolic, machine learning, artificial intelligence

1. Introduction

Machine learning with neural networks has seen enormous success in the computer vision community (Krizhevsky et al., 2012). Convolutional Neural Networks (ConvNets[1]) excel at recognizing patterns in images and are used to identify object classes and segment image pixels into separate objects. Yet for many applications, the accuracy of neural networks is still inadequate. For example, video applications that require accurate per-frame information often have unacceptably high frame-to-frame detection errors. In contrast, symbolic learning has had success discovering and asserting concepts such as object permanence which infer that objects observed at one moment persist consistently into the future.

1. https://en.wikipedia.org/wiki/MNIST_database#Classifiers — accessed Nov. 20th, 2022.

LeelaCore Leela Video Intelligence Framework aims to solve this problem by adding common-sense reasoning on video data streams. Leela Framework does so by interpreting convolutional neural network classification results from images, repairing common failure modes, and improving the quality of the neural network. LeelaCore uses simple physics models of motion and can be customized to recognize common object use cases for a given task. Leveraging its learned patterns expected for object and person activity, Leela can smooth over classification errors in object recognition and human pose predictions from ConvNets. The frames that are repaired can be used to key retraining of the network so that it improves recognition on similar frames of data in the future. Additionally, LeelaCore incorporates a schema-inspired symbolic learning engine (Kommrusch, 2020) which can form and test hypotheses about the world in a self-supervised way. Confirmed hypotheses can be used to enhance neural network performance.

Our key contributions in this paper are:

1. Neural network models have known error categories when doing object classification (such as false negatives where no object is predicted or misses where the wrong category is predicted) as well as for human pose prediction (such as keypoint position error or left/right swapping). We present LeelaCore which is capable of detecting and correcting such errors.

2. Given LeelaCore error detection we show how automatic generation of samples can be done to incrementally train a neural network model with self-supervised samples.

3. We summarize the Leela symbolic learning system and show how the symbolic learning engine can create self-supervised samples based on reliable activity recognition.

2. Background

Neurosymbolic learning systems aim to combine historical symbolic AI techniques with modern neural network approaches. The approach fits well with the 'fast and slow' systems of thinking initially defined by (Kahneman, 2011). In his work Kahneman described system-1 thinking as fast, automatic, and unconscious whereas system-2 thinking is slow, effortful, and conscious. Current neural network architectures, including convolutional neural nets (ConvNets) for image classification and transformers for natural language processing, enable system-1 thinking. We propose that system-2 thinking can be advanced with more careful consideration of how brains reason about facts in the world. By combining system-1 and system-2 techniques, we can create artificial intelligence systems that can solve a wider range of real-world challenges.

2.1. Convolutional neural networks

Convolutional neural networks, also referred to as CNNs or ConvNets, provide an efficient way of learning to recognize features in digital images. However, significant amounts of training data involving hundreds of thousands of samples are needed to reach acceptable performance levels (Wu et al., 2019). For a given application, pretrained models may yield poor performance and incrementally training such models for a specific need often requires thousands of human-labeled images. Nonetheless, using such models can provide us with an artificial approximation of 'system-1' thinking.

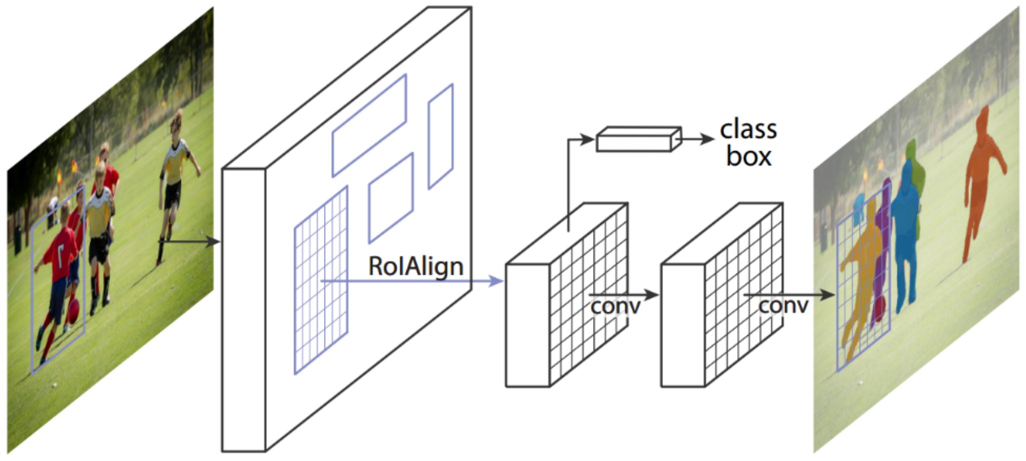

Figure 1: Mask R-CNN framework for instance segmentation (He et al., 2017).

The Mask R-CNN framework (He et al., 2017) shown in Figure 1 will identify classes of objects as well as the pixel mask associated with each object so that the final output includes a list of the objects in a scene as well as their location details in the scene. This is accomplished using learnable weights for multiple layers of convolution filters. Early layers tend to find low-level image components such as edges, curves, and color gradients. Later layers may combine these prior layer results to identify curves and shapes such as hands and feet. As thousands of images are fed into the network, images for which the network output does not match the provided label are corrected by back-propagating the errors through the layers and adjusting the weights. Ultimately the weights converge to produce usable accuracy for object identification and pixel segmentation.

Training such a network can be rather time-consuming and require lots of labeled data. There are, however, techniques for incrementally training the network to recognize new object types. For example, if the initial network was trained on 100 different classes, a new network could reuse only the initial convolution filters but could be applied to recognize a new set of only 10 different object categories. In this way we can customize the recognition to the needs of a given task. This paper focuses on creating incremental training samples for improving the network automatically without requiring human labeling.

A network similar to the one shown in Figure 1 can also be used to generate pose predictions for humans in a scene as shown in Figure 2. In this figure, we see 17 key points identified to predict the position for each person (eyes, ears, nose, shoulders, elbows, wrists, hips, knees, and ankles). In general, the pose detection network does not need to be retrained for new tasks, but the accuracy of the network may be diminished with certain backgrounds or when people are wearing certain clothing (including masks).

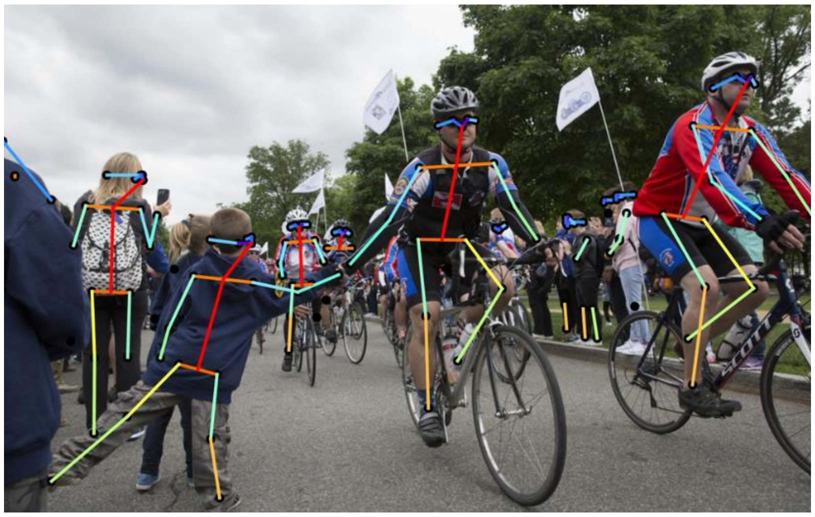

Figure 2: Keypoint locations identified for pose detection (Wu et al., 2019).

2.2. Symbolic understanding and learning

In order to add some system-2 thinking to our system, the first layers of symbolic understanding which we build upon are object permanence and simple physics. For example, we know from experience that physical objects don't disappear from a scene without a cause and we are aware of the physical movement limits of the human body. Despite known challenges in integrating laws of physics with machine learning (Karniadakis et al., 2021), we can make use of this knowledge to correct common errors made by machine learning models when analyzing images (Ronchi and Perona, 2017; Bolya et al., 2020). We cover our approach to these physics issues in section 3.1.

Jean Piaget proposed a theory of childhood cognitive development in which a child learns about objects in the world through sensorimotor experience related to moving its hands and visually perceiving the world (Piaget, 1964). As the child grows it learns more complex concepts such as object permanence. Piaget proposed the concept of schemas to decide when an action (like picking up an object) is applicable and guess what the result of the action will be on the world. He organized a child's learning process into four main stages that exemplify how knowledge gained in previous stages is used to build more complex concepts in later stages.

Gary Drescher proposed a system to bring Piaget's concepts into the field of computing and artificial intelligence (Drescher, 1991). The basis for Drescher's approach is a software model for the schema concept introduced by Piaget, shown in Figure 3. Initially, the learning system has no understanding of how actions affect its sensory input, but repeated actions can produce data usable for the generation of hypotheses about the world. For example, a schema might be learned that when the world context includes the hand on the left side of the visual field, then the action of moving the hand to the right has the result that the hand is seen in the middle of the visual field. As schemas are learned and used, their reliability is tracked and may increase if the schema represents a valid hypothesis about how the world works.

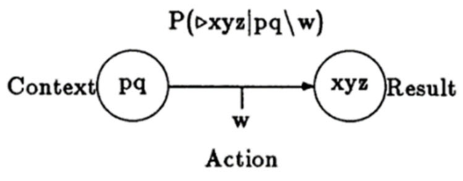

Figure 3: Schema implemented as a data structure including results occurring when a given action is applied to a given world context. Reliability of the schema is also tracked (Drescher, 1986).

3. Methodology

Our aim in this paper is to demonstrate ways in which symbolic understandings of video data information can be used to improve neural network models which are generating the symbols on which reasoning is being done. Figure 4 introduces LeelaCore as the symbolic engine which receives and processes symbolic information from a convolutional neural network. LeelaCore is the core software used by Leela AI, Inc. For our current pipeline, the ConvNet is creating object and pose information for each image in a video stream independently. LeelaCore then processes that information with knowledge of the image sequence as well as the video frame rate which can be used to track velocity and acceleration of items in the video.

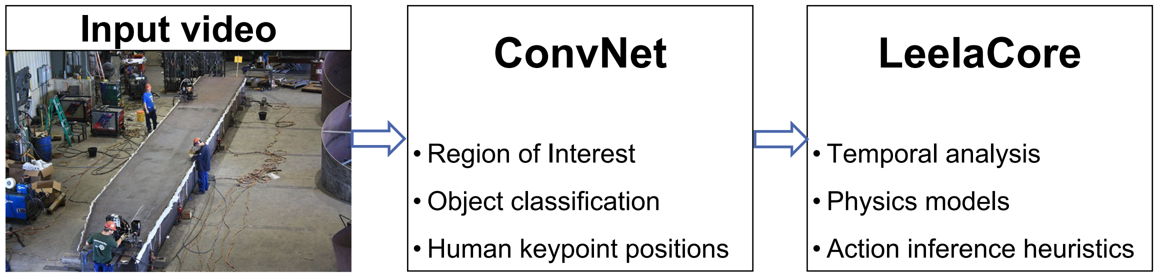

Figure 4: LeelaCore adds symbolic reasoning to results of convolutional networks

We use pre-trained ConvNets for object recognition but add custom objects for specific use cases. In this section we discuss how LeelaCore can analyze results from the ConvNet to discover new training samples which can be used to further perform incremental training on the ConvNet and help it perform at a higher accuracy for a given use case.

3.1. Common pose detection errors and repair

Let us consider a neural network that recognizes keypoint position data for humans within a scene. When a given image is processed by the neural network there are a variety of errors that may occur as itemized in Figure 5 (Ronchi and Perona, 2017). LeelaCore can recognize and recover from many of these error types.

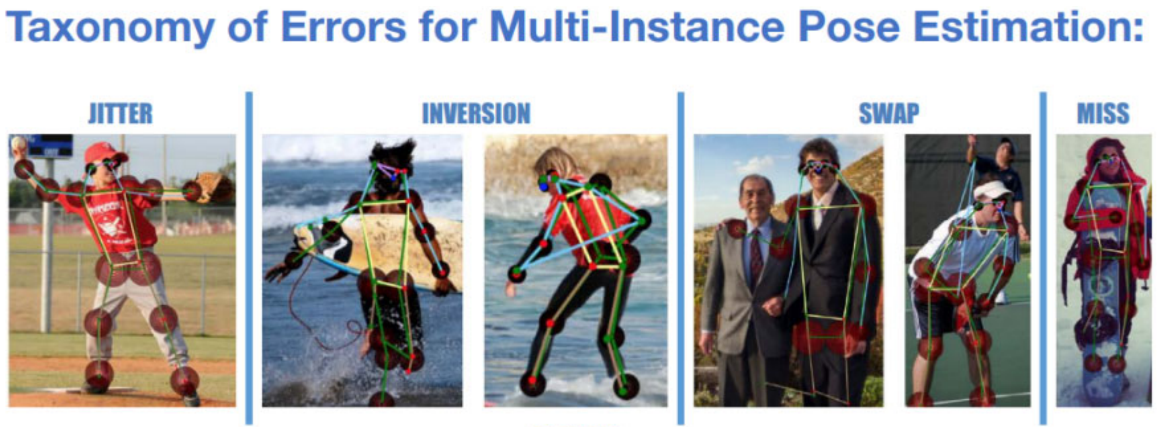

Figure 5: Taxonomy of pose errors (Ronchi and Perona, 2017).

In Listing 1 we show the code skeleton for how all of these error types can be addressed. For example, in order to handle jitter in the position predicted for hips and shoulders, we time-average the hipToShoulder pixel distance computation over multiple frames. We do this by averaging in the newly computed distance as 20% of the value, thereby limiting how much a 'jitter' type error will affect this value. This hipToShoulder value becomes useful for distance measurements since the average hip-to-shoulder distance for an adult is about 60cm (Gordon et al., 2012). In a similar way (not shown) we keep a running average of the velocity and acceleration of each keypoint to help predict where the keypoint is likely to be in the next frame. An athletic human can create a jerk (change in acceleration per second) of about 10,000m/s^3 (i.e. a 50m/s baseball pitch completed in 100ms), but in typical environments a limit of about 2,000m/s^3 is more appropriate (i.e. when using a hammer). For a video at 30fps (frames per second), if the prior velocity and acceleration of a keypoint are known, 2,000m/s^3 can result in a variation of 7cm from the predicted location; at 15fps, the variation can be about 59cm. We allow for some additional variation to account for jitter. Using this error limit we can recover and repair inversion, swap, and miss type errors.

Listing 1: Limit movement with jerk computation.
```
# Update position each frame
# Time-average a standard distance measure for this person
hipToShoulder =
  hipToShoulder * 0.8 + 0.2 * calcHipToShoulder(hips,shoulders)
maxDist = (maxJerk / fps / fps / fps + 0.1) * hipToShoulder * 100/60
for keypoint:
```

```
if new keypoint position further than maxDist from expected:
  use expected position
update keypoint velocity and acceleration estimates
```

Figure 6 shows an example of mispredicted arm keypoints and the repaired position from LeelaCore. Using the repaired position data, new training samples for the neural network can be created. While the neural network is still only inferring pose data given a single frame at a time, by LeelaCore creating new pose data we can improve the accuracy of the prediction for similar images in the future. In cases where the keypoint detection before and after the repair was consistent and high confidence, the new sample could be used directly without human verification of the image, providing for self-supervised samples to be generated by the system; but some images may warrant human review before inclusion. Given a specific use case, updating the model to be more accurate in the given case is quite valuable.

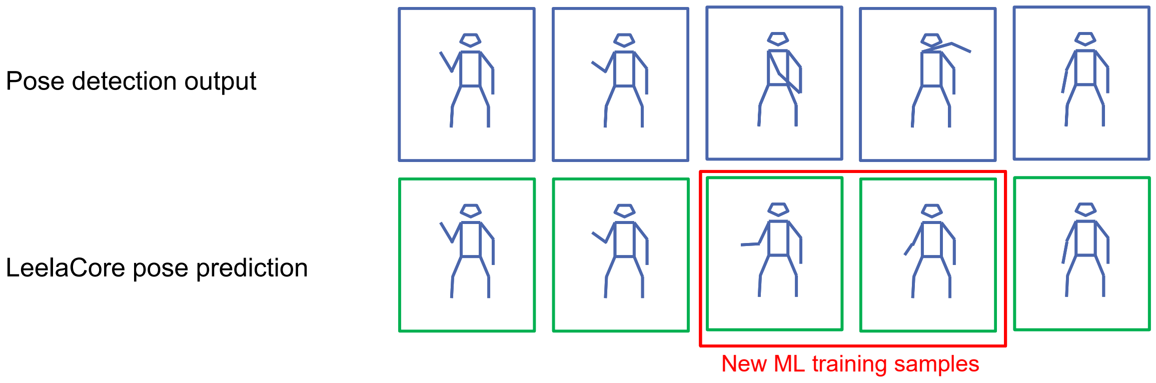

Figure 6: Legal motion clamping to create new pose samples.

3.2. Common object detection errors and repair

Bolya, *et al.* define six types of errors for object detection, as shown in Figure 7: classification error, location error, class+location, duplicate object, background (false positive), and missed (false negative).

Figure 8 shows an example of recovering from a missed 'Table' detection. In this case, object persistence code in LeelaCore recognizes that the object is still in the same location (object velocity is zero). Because the object reappears in a later frame, we presume that the object was indeed in the frame that had dropped it. Note that, if there is no clear symbolic reasoning to ensure the object is visible in the image it is more robust to create a training sample from the 'Object detection output' data just before and after the missing image. For example, if an object of some kind obscured the view of the table, then it may not be visible in the frame for recognition. However, by adding the frames which did recognize the object just before and after it was not seen to the training, the neural network will adjust to recognize similar conditions in the future and improve performance.

Figure 9 shows an example of recovering from a missed 'Cup' detection. In this case, we rely on symbolic common sense rules in which LeelaCore infers that when the wrist is

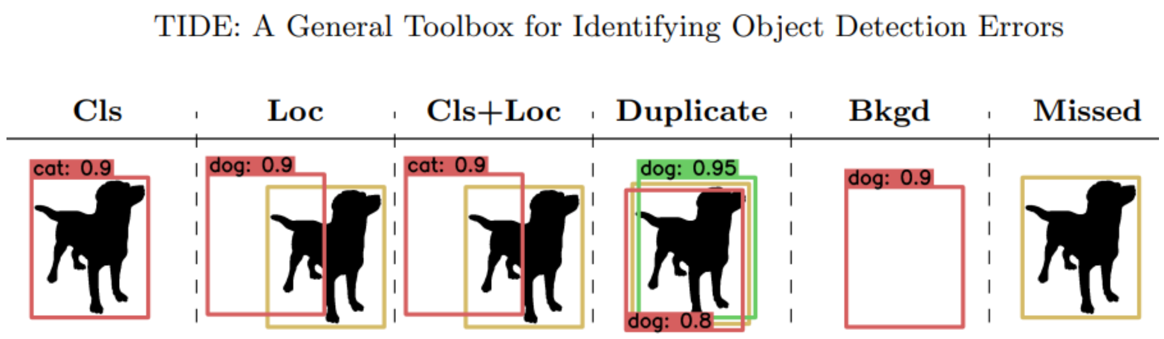

Figure 7: Bolya et al. (2020) define 6 error types: Classification error, location error, class+location, duplicate object, background (false positive), and missed (false negative).

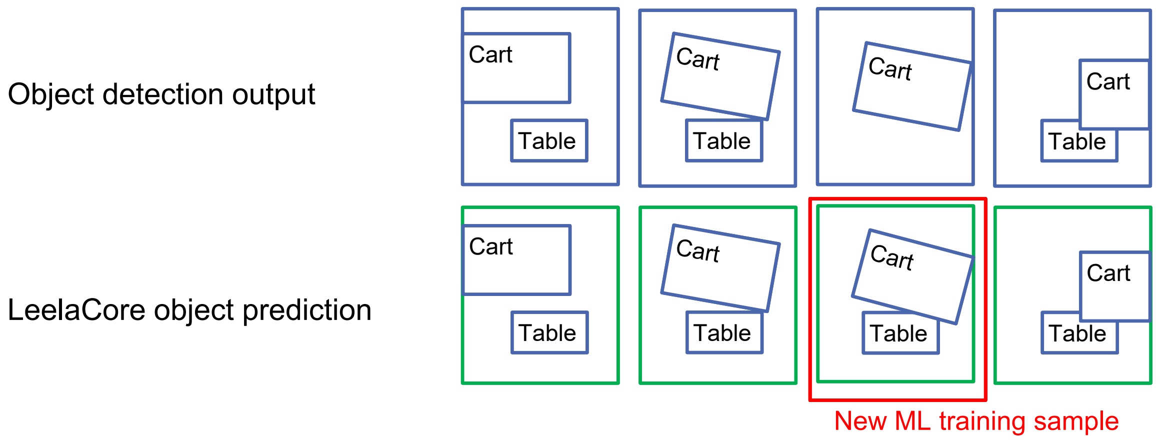

Figure 8: LeelaCore persists objects from frame to frame and can identify new training samples for incremental improvement.

near an object then the object may be picked up and carried. Since in this case the object ‘disappears’ from the table, LeelaCore can assume the object is now in the hand of the person. Again, if there symbolic reasoning to infer the cup should be visible in the image (based on the orientation of the person and their hand, for example) then we may add training examples using the images mispredicted by the ‘Object Detection output’. But if we are unsure that the cup is visible in the frame, we could at least automatically add the frame in which the cup is being touched by the person since that frame has been inferred as correct by LeelaCore.

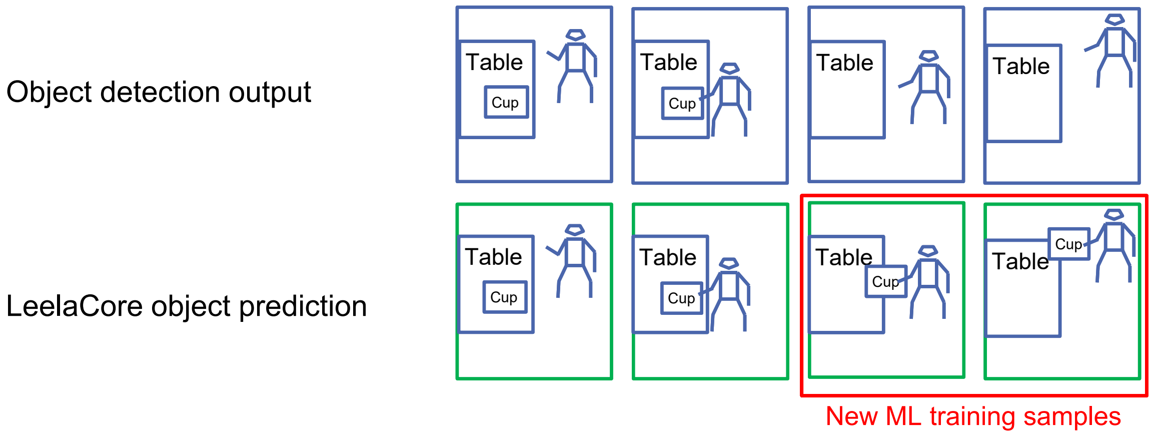

Figure 9: LeelaCore can infer objects based on actions (such as ’object picked up persists in the hand’).

Figure 10 shows an example of recovering from a ‘Cup’ being incorrectly identified as a ‘Can’. In this case, we make use of the ‘confidence’ score from the neural network to observe that the can is not predicted with high confidence. Then we rely on a LeelaCore rule that infers that objects reached at with a hand may be picked up and a hypothetical LeelaCore rule which infers that objects taken to a water cooler and then brought near the mouth are probably cups. Given this inference path and the low confidence of the ‘Can’ prediction, LeelaCore can correct the frame predictions to include the ‘Cup’ and the corrected samples can be provided to train the neural network so that it may improve performance on similar cases in the future.

The ability of object detection samples for retraining a neural network model to be self-supervised depends on the confidence of the LeelaCore system in identifying samples. At first, one might alternate between an automatically trained model and a model with incremental human improvements. For example, given a small set of labeled images, a model is produced which then gets used to find a set of self-supervised samples which improve the model, then a round of clean-up labeled samples by humans improves the dataset further. In this way, less effort is required by humans to fully label all samples, but some oversight is maintained. In cases where the LeelaCore inference is highly confident in its prediction, a label could be added to the sample such that it is truly self-supervised and human oversight is not needed.

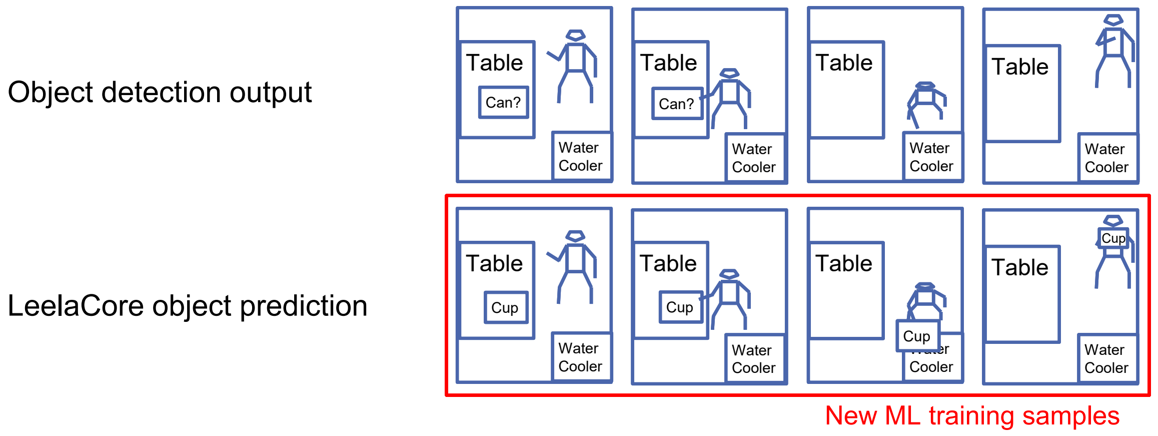

Figure 10: LeelaCore can correct object classification based on observed usage.

4. Experiments In Progress

Table 1 shows frames detected by LeelaCore which can be used to improve model training. The first category identifies human keypoints being detected as incorrect. These keypoints may be missing, duplicates, or mispredicted. The human pose direction is estimated over multiple frames and if the left and right predictions swap suddenly, LeelaCore can correct this by swapping the predicted locations. If the joints accelerate in a way that violates the limitations of human jerk levels this also results in corrected joint positions. In order to improve the quality of the training data, the LeelaCore prediction limit is set to 2 frames for sample generation. That is, the keypoints may be incorrectly predicted by the ConvNet model and repaired by LeelaCore for 1 or 2 video frames but then return to being correctly predicted by the model. This limit helps produce self-supervised samples of problematic images but which are confirmed as correct.

The second category shown in Table 1 is object use confirming an object permanence assumption. When an object is interacted with, such as a ladder being used by a human, the object may have poor quality recognition by the classifier. LeelaCore can detect when a ladder is in use on a given frame but disappears from the neural network prediction in the next frame. In this case, we currently add the last frame that the neural network saw the ladder into the training data. This last frame is confirmed correct but is also on the edge of what the neural network can recognize. In contrast, the first frame in which the ladder is not recognized but that LeelaCore asserts is real may be a case in which the ladder is obscured and, hence, that frame would be a poor training example.

We are currently working to complete the retraining and validation process. We have early results showing that with automatic object retraining our object recognition is better for some categories but don't have sufficient sample sets to quantify the benefit. For early experiments on the pose detection, it appears that the baseline model we are using is difficult to improve - our automatically generated examples of pose error did not appear to improve the results. However, for both the object detector ConvNet and the pose detector ConvNet the methodology for incremental training may need to be tuned.

Table 1: Identified training sample frames given LeelaCore processing of video data.

LeelaCore Correction	Total Video Frames	Identified Sample Frames
Human keypoints corrected to match motion limits	7,135	87
Ladder object persisted during use by human	1,550	67

5. Future Work

In the future we aim to integrate the ConvNet object and pose datapaths with the Leela schema learning mechanism introduced in Section 2.2 which provides a way to make and test hypotheses about the world. The schema code has already been tested in block world settings, as detailed in our previous work (Kommrusch, 2020). Since schema predictions include reliability estimates, when we have confidence that the conditions for a reliable schema are met then we can infer that the results of the schema occur. This can provide for yet another way to create new neural network training samples from the symbolic domain - in this case completing the neurosymbolic learning system by including learning agents in both the neural and symbolic domains. Indeed, the symbolic learner is capable of inferring and creating representations for new objects altogether, which extends the self-supervised domain to not only generate new samples with currently recognized objects, but to add newly learned categories to the neural network classifier.

A broader theme which could be explored based on our current work is the proper form of the interface layer between 'system-1' neural networks and 'system-2' symbolic reasoning. One could imagine finding that it is best for the neural network to identify components of items instead of entire objects (such as wheels and doors instead of cars and trucks) and the symbolic reasoner to receive this component data for reasoning on (in which case the symbolic reasoner would learn to create the concept of a car based on wheels and doors being recognized by the neural network). As the symbolic reasoner interacts with the neural network in early self-supervised ways, it may be possible to use accuracy results to learn how to best structure this interface.

6. Conclusion

In the next decade AI will need to begin learning with fewer labeled examples and reasoning about the world. We've presented techniques developed at Leela AI, which use symbolic reasoning on the output of neural network models to create training examples that can improve the neural network model. In this way, the true spirit of neuro-symbolic reasoning is realized. By combining neural networks to address 'system-1' automatic thinking with symbolic reasoning to address 'system-2' effortful reasoning, we seek to improve AI quality for video reasoning in general.

References

Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. Tide: A general toolbox for identifying object detection errors. In *Computer Vision – ECCV 2020: 16th European*

Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III, page 558–573, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58579-2. doi: 10.1007/978-3-030-58580-8_33. URL https://doi.org/10.1007/978-3-030-58580-8_33.

Gary L. Drescher. Genetic AI: Translating piaget into LISP. Technical report, MIT AI Lab, USA, 1986.

Gary L. Drescher. *Made-up Minds: A Constructivist Approach to Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1991. ISBN 0262041200.

Claire C. Gordon, Cynthia L Blackwell, Bruce Bradtmiller, Joseph L. Parham, Patricia Barrientos, Steven Paquette, Brian D. Corner, Jeremy Carson, Joseph Venezia, Belva M Rockwell, Michael Mucher, and Shirley Kristensen. 2012 anthropometric survey of u.s. army personnel: Methods and summary statistics. 2012.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322.

Daniel Kahneman. *Thinking, Fast and Slow.* Farrar, Straus and Giroux, New York, 2011. ISBN 978-0-374-27563-1.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. 2021.

Steve Kommrusch. Self-supervised learning for multi-goal grid world: Comparing leela and deep q network. In Henry Minsky, Paul Robertson, Olivier L. Georgeon, Milan Minsky, and Cyrus Shaoul, editors, *Proceedings of the First International Workshop on Self-Supervised Learning*, volume 131 of *Proceedings of Machine Learning Research*, pages 72–88. PMLR, 27–28 Feb 2020. URL https://proceedings.mlr.press/v131/kommrusch20a.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Jean Piaget. Part i: Cognitive development in children: Piaget development and learning. *Journal of Research in Science Teaching*, 2(3):176–186, 1964. doi: 10.1002/tea.3660020306. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/tea.3660020306.

Matteo Ruggero Ronchi and Pietro Perona. Benchmarking and error diagnosis in multi-instance pose estimation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 369–378, 2017. doi: 10.1109/ICCV.2017.48.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.