Knowledge-augmented Deep Learning and Its Applications: A Survey

Zijun Cui, Tian Gao, Kartik Talamadupula, and Qiang Ji

Abstract—**Deep learning models, though having achieved great success in many different fields over the past years, are usually data hungry, fail to perform well on unseen samples, and lack of interpretability. Various prior knowledge often exists in the target domain and their use can alleviate the deficiencies with deep learning. To better mimic the behavior of human brains, different advanced methods have been proposed to identify domain knowledge and integrate it into deep models for data-efficient, generalizable, and interpretable deep learning, which we refer to as *knowledge-augmented deep learning* (KADL). In this survey, we define the concept of KADL, and introduce its three major tasks, i.e., knowledge identification, knowledge representation, and knowledge integration. Different from existing surveys that are focused on a specific type of knowledge, we provide a broad and complete taxonomy of domain knowledge and its representations. Based on our taxonomy, we provide a systematic review of existing techniques, different from existing works that survey integration approaches agnostic to taxonomy of knowledge. This survey subsumes existing works and offers a bird's-eye view of research in the general area of knowledge-augmented deep learning. The thorough and critical reviews of numerous papers help not only understand current progresses but also identify future directions for the research on knowledge-augmented deep learning.**

Index Terms—**Domain Knowledge, Deep Learning, Neural-symbolic Models, Physics-informed Deep Learning.**

I. INTRODUCTION

DESPITE the impressive performance that existing deep models have achieved in various fields, they suffer from several serious deficiencies, including high data dependency and poor generalization [1]. These deficiencies originate primarily from the models' data-driven nature and their inability to effectively exploit the domain knowledge. To address these limitations, a knowledge-augmented deep learning paradigm begins to attract researchers' attention, whereby domain knowledge and observable data work together synergistically to produce data-efficient, generalizable, and interpretable deep learning algorithms.

Real-world domain knowledge is rich. In the context of deep learning, the domain knowledge mainly originates from two sources: target knowledge and measurement knowledge. Target knowledge governs the behaviors and properties of the target variables we intend to predict, while measurement knowledge controls the underlying mechanism that produces the observed data of the target variables. Based on its representation, we propose to divide the domain knowledge that has

been explored in deep learning into two categories: *scientific knowledge* and *experiential knowledge*. Scientific knowledge represents the well-established laws or theories in a domain that govern the properties or behaviors of target variables. In contrast, experiential knowledge refers to well-known facts or rules extracted from longtime observations and can also be inferred through humans' reasoning. Knowledge can be represented and organized in various formats. Scientific knowledge is usually well represented rigorously with mathematical equations. Experiential knowledge, on the other hand, is usually represented less formally, such as through logic rules, knowledge graphs or probabilistic dependencies. Knowledge with different representations is integrated with data in a deep learning framework through different integration approaches.

Recognizing the deficiencies with current deep learning, there is a growing interest recently in capturing and encoding prior knowledge into the deep learning. Two mainstream techniques are neural-symbolic integration and physics informed deep learning. Neural-symbolic integration models focus on encoding experiential knowledge into the traditional symbolic AI models and integrating the symbolic models with deep learning models. The physics informed deep learning focuses on encoding various theoretical physical knowledge into different stages of deep learning. Current survey papers in this area are limited in scope as they focus on reviewing either neural-symbolic models or physics-informed machine learning methods, while ignoring many other related works. Specifically, the existing surveys on neural-symbolic models mainly consist of the discussions on logic rules or knowledge graphs, and their integration into deep models [2], [3]. Existing surveys on physics-informed machine learning, however, are limited to a specific scientific discipline, and the integration methodologies are usually task-specific, e.g., physics [4], [5], cyber-physical systems [6], geometry [7], and chemistry [8]. These surveys are hence focused on the methodologies for solving scientific problems under a lab environment, lack of discussions on real-world applications. To address this limitation, we present a comprehensive yet systematic review of existing works on knowledge-augmented deep learning. The contributions of our survey are three folds:

- This survey creates a novel taxonomy on domain knowledge, including both scientific knowledge and experiential knowledge. Our work subsumes existing works that focus on a subset of domain knowledge on specific disciplines [4]–[8].
- This survey covers a wide array of methodologies for knowledge representation and integration with a system-

Z. Cui and Q. Ji are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA (e-mail: cuiz3@rpi.edu; jiq@rpi.edu).

T. Gao and K.Talamadupula are with IBM Research AI, USA.

TABLE I
OVERVIEW OF KNOWLEDGE-AUGMENTED DEEP LEARNING

Knowledge Identification	Knowledge Representation	Knowledge Integration			
		Data-level	Architecture-level	Training-level	Decision-level
Scientific Knowledge	Mathematical Equations	[14]	[15] [16] [17] [18] [23] [24] [25] [30] [31] [32] [33] [38] [39] [40] [41] [46] [47] [48] [49]	[19] [20] [21] [22] [26] [27] [28] [29] [34] [35] [36] [37] [42] [43] [44] [45] [50]–[52] [53] [54] [55]	
	Simulation Engines	[56] [57] [58] [59] [60] [61] [62] [63]			
Experiential Knowledge	Probabilistic Relationships	[14] [64] [65]	[66] [67] [68] [69] [74]	[70] [71] [72] [73]	[64] [14]
	Logic Rules	[22]	[75]	[76] [77]	
	Knowledge Graphs		[78] [79] [80]	[81] [82]	[83] [84]

atic categorization. It differs from existing surveys on general integration techniques, that are agnostic to the taxonomy of domain knowledge [9]–[13].

- This survey covers the methodologies that are not only for solving scientific problems under a lab environment, but, more importantly, for real-world application tasks. Not being limited to a specific application task, this survey involves the tasks ranging from computer vision to natural language processing. Our survey is hence of interest not only to deep learning researchers but also to deep learning practitioners in different fields.

We organize this survey as follows. We first introduce the concept of KADL in Section II, whereby we define the three basic tasks (i.e., knowledge identification, knowledge representation, and knowledge integration). We then review KADL methodologies, where we categorize different techniques based on their domain knowledge of focus: 1) deep learning with scientific knowledge in Section III and 2) deep learning with experiential knowledge in Section IV. In each category, we identify the domain knowledge, its representation formats, and the existing methods proposed for the integration of knowledge with data. An overview of existing methodologies in knowledge-augmented deep learning is included in Table I.

II. KNOWLEDGE-AUGMENTED DEEP LEARNING

Major tasks of knowledge-augmented deep learning include knowledge identification, knowledge representation, and knowledge integration into deep models. In the following sections, we introduce each of the major tasks in detail.

A. Knowledge Identification

Knowledge identification involves identifying the domain knowledge for a specific task. For some tasks, domain knowledge is readily available and hence easy to identify, while in other tasks, the knowledge is less straightforward and requires domain experts' efforts to identify. In this survey, we divide the domain knowledge into *scientific knowledge* and *experiential knowledge*. Scientific knowledge is prescriptive and mainly refers to well-formulated mathematical theories or physics laws. These laws are validated extensively through scientific experiments and are true in a universal setting (e.g., Newton's laws). These laws explicitly define the deterministic relationships among different variables that are involved in the system. Recent work in physics-informed machine learning seeks to exploit various scientific knowledge from different disciplines for enhancing deep learning. Experiential knowledge is descriptive and mainly refers to well-known facts from daily life, indicating semantic properties of an entity or semantic relationships among multiple entities. Experiential knowledge usually is extracted from longtime observations but can also be derived from well-established studies or theories. The latter type of experiential knowledge is science-grounded and is focused on a semantic and abstract level of description. Experiential knowledge usually contains lots of fragmented information, and could be uncertain, imprecise, or ambiguous. Recent work in neural-symbolic models focuses on embedding experiential knowledge into deep learning.

B. Knowledge Representation

Knowledge representation involves representing the identified domain knowledge in a well-organized and structured format. The appropriate representation depends on the type of domain knowledge. Scientific knowledge is usually expressed using equations. Besides, a simulation engine is also considered as an alternative representation of the scientific knowledge. Experiential knowledge is less formal compared to scientific knowledge. Experiential knowledge can be represented through probabilistic relationships, logic rules or knowledge graphs.

C. Knowledge Integration

Knowledge integration entails integrating domain knowledge into deep models. Through the integration, a deep model can leverage both existing datasets and domain knowledge for certain tasks. Different integration methods can be employed depending on the types of knowledge, and can be divided into

four levels: data-level, architecture-level, training-level, and decision-level, as shown in Figure 1. Data-level integration
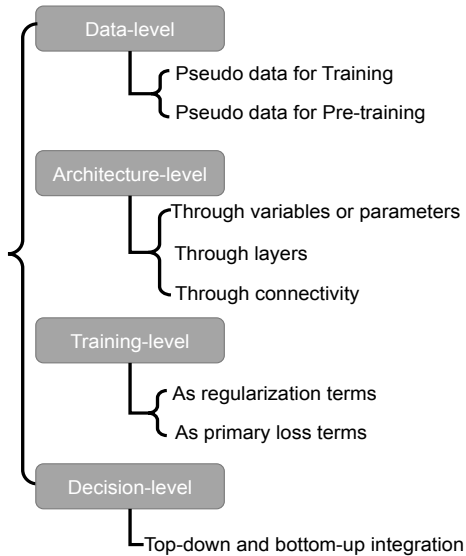

Fig. 1. Four levels of integration: data-level, architecture-level, training-level, and decision-level.

focuses on integrating the knowledge by augmenting the original training data with pseudo data, which is generated based on the knowledge. Architecture-level integration embeds knowledge by modifying neural network architecture. Training-level integration regularizes the training of a deep model via regularization terms or primary loss terms that are derived from the knowledge. Finally, decision-level integration combines a top-down prediction from a prior model with a bottom-up prediction from a deep model, whereby knowledge encoded in the prior model help refine the predictions of a deep learning pipeline.

Each type of integration could be beneficial from different aspects. Data-level integration can help alleviate data paucity issue that faced by many deep models [56]–[59]. Additionally, data generation via automatic simulation is usually less expensive compared to human annotation. Architecture-level integration brings the bonus of making deep models interpretable and explainable, which are two crucial factors for trustworthy artificial intelligence [85]. Training-level integration can be regarded as the most common approach due to its straightforwardness. It can be flexibly applied to different deep model frameworks. Flexibility of training-level integration also promotes the quantification of knowledge uncertainties [45], [53]–[55]. Decision-level integration, different from previous three approaches, employs knowledge independent of the training of deep models, and is relatively less explored by existing works.

Selection of integration approach could be both task-dependent and knowledge-dependent. Architecture-level integration would be preferred if the knowledge is expected to be integrated in a hard way. The rest of three integration approaches (i.e., data-level, training-level, or decision-level) introduce knowledge into a deep learning pipeline in a soft way. Secondly, if the knowledge only involves target variables

(i.e., predictions of a neural network), training-level integration would be preferred. To perform the other three types of integration, the knowledge is expected to involve measurements (e.g., intermediate variables or observations) and target variables. Lastly, if the identified knowledge consists of highly nonlinear and complex relationships, leveraging well-established engines or simulators for data-level integration would be the primary choice.

III. DEEP LEARNING WITH SCIENTIFIC KNOWLEDGE

Deep learning models are gaining importance in advanced science and engineering areas that have traditionally been dominated by mechanistic (e.g., first principle) models. Such models produce particularly promising performances for scientific problems whose undergoing mechanisms are not well understood by experts or those problems for which exact solutions are computationally infeasible. Existing deep learning, however, requires a significant amount of annotated data, and it generalizes poorly to novel data or settings.

There is a growing consensus in the research community regarding combining conventional methodologies in science and engineering with existing data-driven deep models. Deep learning with scientific knowledge explores the continuum between classical mechanistic models and modern deep ones. There have been growing efforts in the machine learning community to incorporate scientific knowledge into deep learning (also referred to as physics-informed machine learning) to generate physically consistent and interpretable predictions and to reduce data dependency.

In the following, we first identify the types of scientific knowledge and their representations. We then introduce different methodologies on integrating scientific knowledge with deep models.

A. Scientific Knowledge Identification

Scientific knowledge refers to well-formulated mathematical or physics equations that have been validated through extensive scientific experiments and are true in a universal setting (e.g., Newton's laws). These laws explicitly define a deterministic and precise relationship among different objects that are involved in a system.

Current physics-informed deep learning seeks to explore the usage of classical mechanics models. For a dynamic system, the most widely considered scientific knowledge is Newtonian mechanics, which includes kinematics and dynamics. The former refers to observable motion (such as motion trajectories), which is typically represented with polynomial equations involving measurable properties (e.g., velocity, acceleration, or positions). Kinematics studies motion without regard for the cause. In contrast, dynamics studies the causes of a motion, whereby partial differential equations (PDEs) are used to capture the relationships between forces and measurable properties. Existing works explore the usage of dynamics in various physics systems (e.g., gas and fluid dynamics [86] and molecular dynamics of protein [87], [88]). With the understanding of dynamics, kinematics can be better predicted. Newtonian mechanics has hence been leveraged in real-world

applications, such as human body behavior analysis [29], [89], [90]. Newtonian mechanics, unfortunately, can result in equations of motion that are intractable to solve, even for a seemingly simple system (e.g., double pendulum system). *Lagrangian mechanics* or *Hamiltonian mechanics* can instead be considered. As re-formulations to Newtonian mechanics, both Lagrangian mechanics and Hamiltonian mechanics leverage generalized coordinates, making them flexible with respect to which coordinates to use to understand a system. In Lagrangian mechanics, L is defined as the difference between the kinetic energy, T, and potential energy, U, of a system (i.e., $L = T - U$). The Hamiltonian H resembles the Lagrangian L, and is defined as the summation of the kinetic energy, T, and potential energy, U, of a system (i.e., $H = T + U$). In Lagrangian mechanics, the time derivative of position $\dot{x}$ is considered as generalized momentum, whereas in Hamiltonian mechanics, momentum p is considered. For simple particle systems, such differences are trivial, while in more complicated systems (e.g., magnetic fields), momentum can no longer be computed as simple a product of mass and velocity. The dynamic equations for both a Lagrangian system and a Hamiltonian system conserve energy over time with conservative forces.

Symmetries have also been widely explored in physics. Philip Anderson famously argued, "It is only slightly overstating the case to say that physics is the study of symmetry" [91]. Discovering symmetries has been proven to be important in both deepening the understanding of physics and enhancing machine learning algorithms. Equivariant or invariant functions preserve symmetries and have often been exploited for incorporating these symmetries into deep learning algorithms.

Optics, another type of physics knowledge, has also been considered. Optics studies the behavior and properties of light. Fermat's principle [92] is the basic law in optics. In addition, the illumination models [93] and the rendering equation [94] capture the 3D object appearances with their image appearances. Existing works explore the usage of various illumination models for different computer vision tasks [95], [96] and computer graphics tasks [97]–[99].

Besides physics knowledge, mathematical theories such as theorems on existing algorithms (e.g., sorting or ranking), as well as continuous relaxation [100], have also been considered. Projective geometry theories [101] are widely applied to various computer vision tasks.

B. Representation of Scientific Knowledge

Scientific knowledge is usually represented as equations, such as differential equations. Another important representation tool of scientific knowledge is simulation. Simulation mimics a real-world physics system and is treated as a surrogate representation of the physics principles that govern the real-world physics system. We visualize the taxonomy of scientific knowledge and its representations in Figure 2.

1) Mathematical equation: Equations can include polynomial equations, differential equations, and integral equations, among which differential equations are widely explored by existing works. Dynamic laws are usually represented by PDEs,

which indicate a deterministic relationship among different variables. In general, the PDEs are of the form

$$\mathcal{N}_{\boldsymbol{x}}(u(\boldsymbol{x}); \boldsymbol{\lambda}(\boldsymbol{x})) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in D$$
$$\mathcal{B}_{\boldsymbol{x}}(u(\boldsymbol{x}); \boldsymbol{\lambda}(\boldsymbol{x})) = b(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma \tag{1}$$

where $\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$ are N variables involved in the system. $\mathcal{N}_{\boldsymbol{x}}$ is a general differential operator, and $\mathcal{B}_{\boldsymbol{x}}$ is the boundary condition operator. D represents a physical domain, and $\Gamma = \partial D$ represents the boundary of the domain. $\boldsymbol{\lambda}(\boldsymbol{x})$ represents the physics parameters involved in the PDEs. For simple systems, physical parameters are constants (i.e., $\boldsymbol{\lambda}(\boldsymbol{x}) = \boldsymbol{\lambda}$). $f(\boldsymbol{x})$ is a forcing term, and $b(\boldsymbol{x})$ specifies the boundary condition, e.g., Dirichlet boundary conditions for the Darcy flow problem [54]. $u(\boldsymbol{x})$ is the solution of the differential equation given the specified boundary conditions. The equations become ordinary differential equations (ODEs) when only one variable is present. When terms (e.g., physics parameters $\boldsymbol{\lambda}$) exist in differential equations that undergo stochastic processes, the equations become stochastic differential equations (SDEs). The general form of SDEs resembles standard differential equations, except for the random event ω:

$$\mathcal{N}_{\boldsymbol{x}}(u(\boldsymbol{x}; \omega); \boldsymbol{\lambda}(\boldsymbol{x}; \omega)) = f(\boldsymbol{x}; \omega), \quad \boldsymbol{x} \in \mathcal{D}, \quad \omega \in \Omega$$
$$\mathcal{B}_{\boldsymbol{x}}(u(\boldsymbol{x}; \omega); \boldsymbol{\lambda}(\boldsymbol{x}; \omega)) = b(\boldsymbol{x}; \omega), \quad \boldsymbol{x} \in \Gamma \tag{2}$$

The physics parameters $\boldsymbol{\lambda}(\boldsymbol{x}; \omega)$ and forcing terms $f(\boldsymbol{x}; \omega)$ are modeled as random processes, and thus the solution $u(\boldsymbol{x}; \omega)$ follows stochastic processes specified by f and $\boldsymbol{\lambda}$.

Differential equations describe a system's evolution over time when we explicitly set one of the variables to correspond to time t, which commonly occurs in different dynamic systems. For example, the Euler-Lagrange equation defines the dynamics of a Lagrangian system,

$$\frac{\partial L}{\partial x} - \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = 0 \tag{3}$$

which connects the derivatives of the Lagrangian L with respect to location in generalized coordinates x, time t, and generalized momentum $\dot{x}$. Differential equations have been widely explored by existing works, such as Newton's second law [16], [34], Burgers' equation occurring in gas and fluid dynamics [86], Hamilton's equations in Hamiltonian dynamics [25], Euler-Lagrange equations for Lagrangian dynamics [102], and the Lorenz equations, which describe a nonlinear chaotic system used for atmospheric convection [103].

Laws of light are also expressed as equations. Fermat's principle considers the integration over a light path. In [97], the transient I in a transient imaging system is characterized through an integration as

$$I(\tau; \boldsymbol{v}) = \int_{\boldsymbol{x} \in \chi} f(\boldsymbol{x}; \boldsymbol{v}) \delta(\tau - \tau(\boldsymbol{x}; \boldsymbol{v})) dA(p, q) \tag{4}$$

where τ is the traveled pathlength and $\boldsymbol{v}$ is the visible point. $A(p, q)$ measures the unit area of a surface, and the function f absorbs reflectance and shading. The rendering equation represented as an integral equation has also been considered [104]. Equality algebraic equation has also been considered. According to the law of reflection, an image I
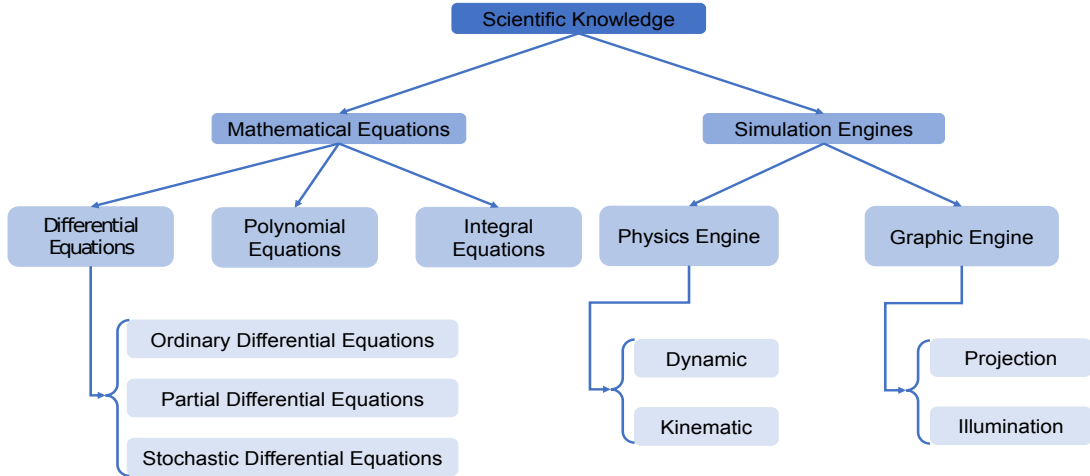
Fig. 2. Taxonomy of scientific knowledge and its representations.

with reflections is a sum of the glass reflected back scene $\hat{R}$ and the glass transmitted front scene $\hat{T}$, i.e.,

$$I(x, y) = \hat{T}(x, y) + \hat{R}(x, y) \tag{5}$$

Such equality algebraic equation is demonstrated to be helpful in reflection removal tasks [98]. Another example is Malus' law, expressed as an algebraic equation, which defines the effect of polarization [99].

2) Simulation engines: Besides explicitly representing physics laws with equations, simulation through engines is another method of representation. Simulation mimics a real physical system governed by physics laws and is hence considered a surrogate representation of the knowledge. A physics engine mainly encodes the governing dynamic laws of a physics system, such as rigid-body, soft-body, and fluid. It computes the accelerations, velocity, and displacement of an object from forces, by solving the equation of motion. Specifically, a physics engine simulates observable kinematics given certain causes following governing dynamic rules. To simulate a sequence of motion of rigid-body objects given specific forces, a physics engine is considered [36], [61], [62], [105]–[107]. Most of these simulators are non-differentiable, making them prohibited to be employed in an end-to-end deep learning framework. There also exist simulation engines that encode kinematic laws for robotic manipulation. These inverse engines, however, estimate control actions using kinematic equations such that a desired position can be reached, agnostic to the underlying dynamic laws [35]. Graphic engines have also been explored. A graphic engine encodes the principled projection and illumination models, and renders realistic 2D observations by following the governing principles. For example, an engine, governed by laws of reflection, was proposed for the generation of faithful image rendering [98].

C. Integration into Deep Models

To integrate domain-specific scientific knowledge into deep models, existing methodologies can be classified into three categories: data-level, architecture-level, and training-level integration as shown in Figure 1. Decision-level integration is rarely considered for scientific knowledge. Below we review methods to integrate scientific knowledge into deep models using these methods.

1) Data-level Integration: One way to harness domain knowledge is to train deep models with data synthetized from conventional mechanistic models. Physics-based mechanistic models capturing the domain knowledge serve as simulators, and are used for generating synthetic data. The simulated data can either be combined with real data to jointly train the model or be used independently to pre-train the model through self-supervised learning.

Simulated data has been widely employed for entire training, where no additional real training data is required. In computer vision, Mottaghi et al. [62] proposed understanding the force acting on a query object by predicting its long-term motion in 3D space as the response to the force, given a static 2D image. An overview of its procedure appears in Figure 3. The Blender[1] game engine, consisting of a physics and a graphic engine, is employed for the simulation. Specifically, the physics engine takes a scene configuration as input (e.g., a ball sliding on a slide) and simulates the scene forward in time according to motion equations derived from Newton's second law to produce different Newtonian scenarios. Physics parameters (e.g., force magnitude) are randomly sampled. For each Newtonian scenarios, the graphic engine renders 2D videos from the synthetic 3D world under different viewpoints in perspective projection. In total, 66 synthetic videos corresponding to 12 possible physical generated scenarios are used for training. Similarly, to better understand the human
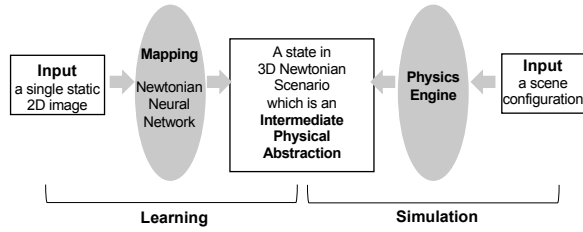
---
[1]http://www.blender.org/

Fig. 3. Understanding physics from 2D images through simulation [62]. The game engine takes scene configurations as input and generates in total 66 Newtonian scenarios as an abstraction of the real physical world. The Newtonian neural network learns to understand the physics by classifying a static image into one of the 66 possible Newtonian scenarios.

interaction with objects from 2D videos, Ehsani et al. [61] proposed a physical understanding of actions by inferring contact points together with forces from videos. A forward physics simulation is applied to supervise the force estimation, given observed videos, without requiring GT labels for forces. In particular, the 3D contact points of a moving object are estimated over a period given estimated forces, through physics simulation which is governed by Newton's second law. A projection operator is then applied transforming the estimated 3D keypoints into a 2D space. By minimizing the difference between the estimated and observed 2D contact points over time, physically consistent forces are obtained. To perform differentiable physics simulations, a finite difference method is applied for gradient computation using the PyBullet[2] simulator, which is focused on rigid-body simulation following Newton's second law. In a similar vein, Tobin et al. [63] showed the effectiveness of synthetic samples generated by robotic simulation for training a deep model for object localization tasks, which is instrumental in robotic manipulation. Simulation is performed by employing the MuJoCo physics engine,[3] whereby simulated 2D images are generated based on its built-in graphic engine [108]. In particular, the MuJoCo physics engine is built based on Newtonian mechanics. The built-in graphic engine renders a 2D image given a selected camera in the 3D virtual environment through a perspective projection. To ensure sufficient simulation variabilities, a domain randomization strategy is proposed, where simulation parameters, such as the positions and orientation of objects, are all randomly specified during the simulation.

Deep model training often starts with a pre-training stage, followed by a fine-tuning. Existing studies also show that pre-training affects the final performance of deep models, mainly because poor pre-training can lead models to anchor in a local optimum. Pre-training through simulated data has shown to help improve parameters' initialization. Jia et al. [56], [57] introduced a physics-guided recurrent neural network (PGRNN) to model lake temperature dynamics. PGRNN is pre-trained on synthetic data generated from a physics-based mechanistic simulator and is then fine-tuned with some observation data. The simulator models the lake temperature dynamics as a function of physics parameters (e.g., water

[2]https://pybullet.org/wordpress/
[3]https://mujoco.org/

clarity and wind sheltering) through PDEs. It was shown that, even with synthetic data generated with an imperfect set of physics parameters, PGRNN still achieves competitive performance. Such an idea has also been explored in engineering disciplines. In robotics, Bousmalis et al. [58] showed that the observation data required for accurate object grasping is significantly reduced (by a factor of 50) through physics-guided initialization. In autonomous driving, Shah et al. [59] pre-trained a driving algorithm with synthetic samples generated by a proposed simulator built on a game engine with physics laws embedded. In particular, the simulator includes a vehicle model and a physics engine. To describe a virtual 3D environment, physics parameters, such as gravity, air density, air pressure, and magnetic field, are specified manually. With these specified parameters, the physics engine predicts the kinematic states, given forces and torques estimated from a vehicle model by following motion equations derived from Newton's second law. This work showed that the data needs of driving algorithms can be drastically lessened through pre-training with simulated samples. In addition to augmenting data through simulation with physics engines, synthetic data can be generated from mathematical equations [14].

2) Architecture-level Integration: Domain knowledge can also be integrated through a customized design of neural network architectures. Architecture-level integration can be accomplished by 1) introducing specific physically meaningful variables or neural network parameters, 2) introducing layers derived from domain knowledge, and 3) introducing physics-inspired connectivity among neurons. We introduce each type of approaches at below.

a) Integrating through variables or parameters: One way to embed physical principles into the architecture of neural networks is to introduce physically meaningful variables in neural networks. The variable can be the output node of a neural network. Hamiltonian functionality that enforces energy conservation has attracted much attention [24], [25], [30], [31]. The Hamiltonian operator in physics is the primary tool for modeling dynamic systems with conserved quantities. In Hamiltonian mechanics, a classical physical system is described by a generalized coordinate, q, and a conjugate momenta, p. Hamiltonian H then calculates the total energy of the system. The Hamiltonian equations defining the dynamics of a system are as follows:

$$\dot{q} = \frac{\partial H}{\partial p}; \quad \dot{p} = -\frac{\partial H}{\partial q} \tag{6}$$

Inspired by the Hamiltonian mechanics, a Hamiltonian neural network (HNN) has been proposed, where the output represents the Hamiltonian dynamics, through which energy conservation is explicitly enforced [24]. The differences between traditional NNs and HNNs can be understood easily, as shown in Figure 4, which shows that traditional NNs learn to predict particle trajectories, while HNNs learn a particle's Hamiltonian, upon which trajectories can be predicted. Choudhary et al. [25] later showed that an HNN can better handle highly nonlinear dynamics in a chaos system by enforcing the conservation of the total energy. To demonstrate the practical importance of Hamiltonian formalism, Hamilto-
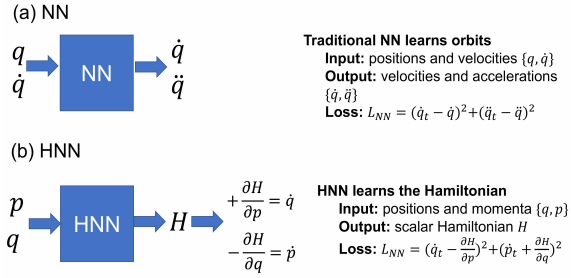
Fig. 4. Comparison between a Hamiltonian neural network (HNN) and conventional neural network (NN). For dynamic modeling, instead of directly predicting trajectories, the HNN predicts the Hamiltonian H of a system, through which the trajectory can be obtained under the energy conservation constraint.

nian functionality was incorporated into a generative network, where Hamiltonian dynamics is learned from 2D observations without domain coordinate assumptions, such as images. The proposed Hamiltonian generative network has been applied to density estimation, leading to a neural Hamiltonian flow [24]. By leveraging Hamiltonian formalism, the total probability can be easily conserved, while the density modeling remains expressive. Sharing a similar idea with HNNs, a generalized energy conservation employing Lagrangian mechanics was explored in the Lagrangian neural network (LNN) [32], [33], where the output of the LNN is the Lagrangian dynamics. Nevertheless, evaluations of all these existing proposed models remain conceptual, without real applications in practice [38].

Variables can also be intermediate variables in a neural network. Jaques et al. [16] proposed a latent dynamics learning framework called the Newtonian Variational Autoencoder (NewtonianVAE). Inspired by Newton's second law, a linear dynamic system in hidden space is defined, specifically by considering a rigid-body system with D degrees of freedom and modeling the configuration of this system by a set of coordinates, $\boldsymbol{x} \in R^D$. Its dynamics are defined as

$$
\begin{aligned}
\frac{d\boldsymbol{x}}{dt} &= \boldsymbol{v} \\
\frac{d\boldsymbol{v}}{dt} &= A(\boldsymbol{x}, \boldsymbol{v}) \cdot \boldsymbol{x} + B(\boldsymbol{x}, \boldsymbol{v}) \cdot \boldsymbol{v} + C(\boldsymbol{x}, \boldsymbol{v}) \cdot \boldsymbol{u}
\end{aligned}
\tag{7}
$$

where $\boldsymbol{u}$ is the given actuation. To incorporate the above dynamic equations into a VAE, positions $\boldsymbol{x}$ are treated as random variables with velocities $\boldsymbol{v}$ being intermediate variables. Specifically, velocities are computed as $\boldsymbol{v}_t = \frac{\boldsymbol{x} - \boldsymbol{x}_{t-1}}{\Delta t}$ with time interval Δt. The conditional distribution of $\boldsymbol{x}_t$ given $\boldsymbol{u}_t$ now becomes

$$
\begin{aligned}
p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}; \boldsymbol{v}_t) &\sim \mathcal{N}(\boldsymbol{x}_{t-1} + \Delta t \cdot \boldsymbol{v}_t, \sigma^2) \\
\text{with} \quad \boldsymbol{v}_t &= \boldsymbol{v}_{t-1} + \Delta t \cdot (A\boldsymbol{x}_{t-1} + B\boldsymbol{v}_{t-1} + C\boldsymbol{u}_{t-1})
\end{aligned}
\tag{8}
$$

where A, B, and C are estimated through a neural network, f, whose input consists of current system states (i.e., $\{\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{u}_t\}$). Newtonian VAE then outputs 2D images given the estimated $\boldsymbol{x}$. This is in contrast with the existing approaches that simply assume a Gaussian distribution over $p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})$ without considering the underlying deterministic relationships among positions, forces, and velocities, as shown in Eq. (8). Similarly, to incorporate the physics

behind the transport dynamics governed by advection-diffusion PDEs, Liu et al. [15] proposed a learning framework based on an autoencoder with the advection-diffusion equation explicitly incorporated. Two hidden variable outputs from the encoder are physically meaningful, representing velocity field and diffusion field, respectively, within the advection-diffusion equation. To model lake temperature, Daw et al. [17] introduced a physically meaningful intermediate variable for the proposed monotonicity-preserving long short term memory (LSTM) architecture. Specifically, the density value, as the intermediate variable of the LSTM, is enforced to monotonically increase as the depth increases, which is a crucial characteristic of lake temperature. A similar idea is applied in modeling drag forces acting on each particle in moving fluids [18]. Muralidhar et al. [18] proposed a PhyNet, where physics-constrained intermediate variables are introduced into a convolutional neural network (CNN) architecture. Specifically, two intermediate variables, characterizing velocity field and pressure field, respectively, are introduced into CNN for drag force prediction.

Besides introducing physically meaningful variables, another approach is to directly map some of the neural network parameters to physically meaningful parameters. These physics parameters can either be non-modifiable during training or be fine-tuned through learning from observations. In geophysics, neural networks have been considered for modeling the dynamic process of seismic waveform inversion [23]. To mimic seismic wave propagation, a theory-guided recurrent neural network (RNN) is proposed; an RNN is specially designed for solving the governing differential equations with some of the parameters assigned as the physics parameters in governing physics equations. In particular, given the wave equation discretized in the time domain, the wave field at the next time step (i.e., $u(t + \Delta t)$) is calculated in terms of two previous time steps (i.e., $u(t)$ and $u(t - \Delta t)$) as

$$
\begin{aligned}
u(r, t + \Delta t) = v^2(t)\Delta t^2 [\nabla^2 u(r, t) \\
- s(r, t)\delta(r - r_s)] + 2u(r, t) - u(r, t - \Delta t)
\end{aligned}
\tag{9}
$$

where ∇^2 is the spatial Laplacian operator, r stands for position, and s is the source function. The symbolic computation of $u(r, t + \Delta t)$ given $u(r, t)$ and $u(r, t - \Delta t)$ is directly implemented via a neural network, whose trainable parameters correspond to physics parameters $v^2\Delta t^2$ in the wave equation Eq. (9).

b) Integrating through layers: The most representative type of knowledge that is usually integrated through neural network layers involves symmetries. Symmetry usually refers to a set of invertible transformations g, such as translation, rotation, or scaling.

Equivariance and invariance as representative types of symmetry have been widely considered. An invariant function is a mapping such that the output space is not affected by the symmetry transformations in the input space, and an equivariant function relaxes the invariant function. It states a mapping such that symmetries in the input space can be preserved in the output space. Mathematically, assume a

symmetry transformation, g, and a function, F, mapping from X to Y. F is then said to be equivariant to g if

$$F(g \cdot x) = g \cdot F(x) \tag{10}$$

The symmetry transformation g on the input space X is preserved on the output space Y. F is said to be invariant if

$$F(g \cdot x) = F(x) \tag{11}$$

In other words, the output $y = F(x)$ is not affected by the symmetry transformation, g, acting upon the input space, X. Invariance can be a special case of equivariance, and vice versa. For example, transformation g is an identify transformation with $g \cdot F(x) = F(x)$.

Equivariant or invariant neural network is designed to preserve symmetries. Illumination invariance features were explored [47] in computer vision, where a knowledge-guided convolutional layer is incorporated into existing deep models. Consider a day-night domain adaptation problem, and illumination changes from source domain to target domain cause a distribution shift. To tackle the distribution shift issue, features that are invariant to illumination are desired, which can be derived from Kubelka-Munk theory. The Kubelka-Munk theory [109] models material reflections by defining the spectrum of light reflected from an object in the viewing direction. The calculation of illumination invariant features defined by Kubelka-Munk theory is directly implemented through the proposed Color Invariant Convolution (CIConv) layer, as shown in Figure 5. Similarly, in turbulence mod-
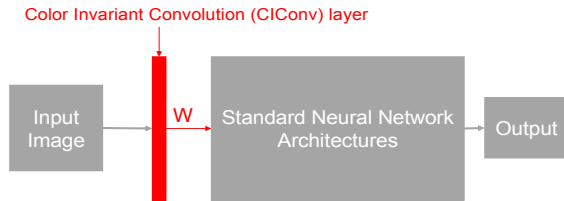

Fig. 5. A physics layer (marked as red) for illumination invariant features [47]. The new layer extracts illumination invariant features given input images and is customized based on Kubelka-Munk theory, termed as Color Invariant Convolution (CIConv) layer.

eling, rotational invariance states that the physics of a fluid flow does not depend on the orientation of the coordinate of an observer and is a fundamental physical principle. To embed the rotational invariance into a neural network, Ling et al. [41] defined a tensor basis neural network (TBNN), where the NN architecture is modified by adding a higher order multiplicative layer. Particularly, TBNN has one additional input layer accepting the tensor basis, and its last hidden layer performs a pair-wise multiplication using this tensor basis input layer to provide the output. The modified architecture ensures the prediction lie on a rotationally invariant tensor basis. By incorporating rotational invariance, TBNN achieves improved accuracy in predicting the normalized Reynolds stress anisotropy tensor. In the application to molecular dynamics, Anderson et al. [46] proposed a rotation invariant neural network, named Cormorant, whereby the behavior

and properties of complex many-bodied physics systems are learned. Each neuron in Cormorant explicitly corresponds to a subset of atoms. Given specified neurons, the activation layer is ensured to be covariant to rotations, making the proposed Cormorant is guaranteed to be rotationally invariant.

Equivariance has also been explored. Wang et al. [39] showed that existing spatiotemporal deep models can achieve improved generalization ability by incorporating symmetries through equivariant functionality. More specifically, they consider four types of equivariance: time and space translation equivariance, rotation equivariance, uniform motion equivariance, and scale equivariance. These symmetries are incorporated into neural networks using customized equivariant layers. Through the composition of equivariant functions of layers, the network becomes equivariant.

In the real physical world, however, symmetries could be brittle. For example, a small perturbation can easily cause the discontinuous transition of a dynamic system or break the rotational symmetry of a pendulum system. Small perturbations often occur, and these can cause significant difference by accumulating over time. As a result, enforcing equivariance through customized layers as a hard inductive bias, could be problematic. To address this problem, Finzi et al. [40] recently proposed a soft way to impose equivariance constraints, whereby the proposed neural network architecture consists of a hybrid of restrictive layers and flexible layers. Restrictive layers are strictly constrained while flexible layers are unconstrained. Through a mixture of two types of layers, the equivariance is introduced as a flexible inductive bias.

c) Integrating through connectivity: Given physical dependencies among objects, connectivity among neural network neurons can be manually specified. To model dynamics with multiple objects involved, physically plausible interactions are employed to design NN connectivity. Neural Physics Engine (NPE), a differentiable physics simulator that combines symbolic structure with gradient-based learning, has been proposed [48]. Different from conventional physics engines based on mechanistic models, NPE is realized as a learning-based neural network while remaining generalizable across different scenarios. The development of NPE considers the fact that physics is Markovian, both temporally and spatially. Temporal Markovian allows the NPE to predict system states by only considering the states at the current step. Spatial Markovian allows the NPE to factorize the interaction dynamics into pairwise interactions. The NPE consists of a symbolic model structure mimicking pairwise interactions among objects, realized as a neural network. The NPE takes 2D observations as input and performs forward dynamics to predict the motions of objects in the future. A very similar work was independently developed for n-body interaction systems [49].

3) Training-level Integration: One of the most common techniques to integrate scientific knowledge into deep models is through the training of deep models. In particular, constraints over the outputs of deep models are derived from scientific knowledge and are used as regularization terms for training a deep model. An augmented training objective is

generally expressed as

$$\mathcal{L} = \mathcal{L}_{TRN}(y^{GT}, y^{Pred}) + \lambda \mathcal{L}_{PHY}(y^{Pred}; x) \quad (12)$$

$\mathcal{L}_{TRN}(y^{GT}, y^{Pred})$ represent standard training loss given predictions y^{Pred} and ground truth labels y^{GT}. For classification task, $\mathcal{L}_{TRN}(y^{GT}, y^{Pred})$ is usually defined based on the cross-entropy loss. The physics-based regularization term $\mathcal{L}_{PHY}(y^{Pred}; x)$ corresponds to physical constraints, with an adjustable importance coefficient, λ. When the physics-based constraints are independent of inputs x, the regularization term is reduced to $\mathcal{L}_{PHY}(y^{Pred})$. Through $\mathcal{L}_{PHY}(y^{Pred}; x)$, training is guided towards producing models with physically consistent outputs. The computation of physics-based regularization, $\mathcal{L}_{PHY}(y^{Pred}; x)$, does not require annotations of observations and hence allows unlabeled data to be included in training, reducing dependencies on data. Physics-based regularization, $\mathcal{L}_{PHY}(y^{Pred}; x)$, can also be employed directly for training deep models in a label-free manner.

$\mathcal{L}_{PHY}(y^{Pred}; x)$ can be explicit or implicit and can be flexibly employed in different deep learning frameworks. Explicit regularization is directly defined over the output of deep models based on domain knowledge, while implicit regularization is induced by the physics-based models that are embedded into a deep learning pipeline. In the following, we introduce the knowledge-guided model regularization under two distinguishable deep model frameworks: discriminative deep models and generative deep models.

a) Regularization with discriminative deep models: Model regularization with scientific knowledge is widely seen in discriminative deep models. In the context of climate modeling, constraints derived from the conservation laws that a physical system should satisfy are encoded as a regularization term. In particular, a NN, f maps input x to output y (i.e., $y = f(x)$). Conservation constraints $\mathcal{C}$ are summarized as a linear system (i.e., $\mathcal{C} = \{C \begin{bmatrix} x \\ y \end{bmatrix} = 0\}$), where C is a given constraint matrix. These physical constraints are then encoded as a regularization term on the NN outputs:

$$\mathcal{L}_{PHY}(x, y) = ||C \begin{bmatrix} x \\ y \end{bmatrix} || \quad (13)$$

Evaluation results show that by adding this physics-guided regularization, prediction performance is improved for emulating cloud processes [19], [20]. Similarly, Zhang et al. [21] proposed parameterizing atomic energy for molecular dynamics through a NN, whose loss function considers the conservation of kinetic and potential energy. Regularization for a physics system with non-conservative forces has also been considered [22]. In a double pendulum system with friction, the total energy of a system is decreasing due to the existence of friction. Decreasing of energy is formulated as a constraint $E(\boldsymbol{x}_{t+1}) < E(\boldsymbol{x}_t)$, where $E(\boldsymbol{x}_t)$ and $E(\boldsymbol{x}_{t+1})$ denote the total energy of the system at current time step and a future time step, respectively. Considering a neural network which takes the state at current time step (i.e., $\boldsymbol{x}_t$) as input and outputs the estimated state at the next time step (i.e., $\boldsymbol{y} := \hat{\boldsymbol{x}}_{t+1}$), the

constraint can be integrated into the neural network through the regularization term as

$$\mathcal{L}_{PHY}(\boldsymbol{x}_t, \boldsymbol{y}) = \text{ReLU}(E(\boldsymbol{y}) - E(\boldsymbol{x}_t)) \quad (14)$$

where $E(\boldsymbol{x}_t)$ and $E(\boldsymbol{y})$ computes the energy of the system at the current and next time step, respectively. According to the decreasing of energy, $E(\boldsymbol{y})$ is expected to be smaller than $E(\boldsymbol{x}_t)$, leading to the constraint above. Similarly, a set of common physical properties of a dynamic system is considered in [26], where each of these physical properties is represented as an equality or inequality constraint. These physics-informed constraints are then incorporated into deep models as regularization via an augmented Lagrangian method.

PDEs (as described in Eq. (1)) have been widely considered as constraints and are integrated into deep models as regularization terms. A physics-informed neural network (PINN) [27] is proposed for solving PDEs by leveraging NNs. A PINN learns solutions, u, by using both the observed data and PDEs whereby PDEs serve as inductive bias. Consider the viscous Burgers' equation as an example [28]:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = v \frac{\partial^2 u}{\partial x^2} \quad (15)$$

In a PINN, a feed forward NN predicts a PDE solution, $\tilde{u}$, by taking positions x and time t as input. The objective function consists of a data loss term and an PDE residual,

$$\mathcal{L} = \mathcal{L}_{data} + \lambda \mathcal{L}_{PDE} \quad (16)$$

where $\mathcal{L}_{data}$ measures the difference between predicted PDE solution $\tilde{u}$ and given u at certain positions and time step (i.e., $\mathcal{L}_{data} = \sum_{i=1}^{N_{data}} (\tilde{u}(x_i, t_i) - u_i)^2$). $\mathcal{L}_{PDE}$ measures the PDE residual of predicted solution $\tilde{u}$ at position and time step:

$$\mathcal{L}_{PDE} = \sum_{j=1}^{N_{PDE}} (\frac{\partial \tilde{u}}{\partial t} + u \frac{\partial \tilde{u}}{\partial x} - v \frac{\partial^2 \tilde{u}}{\partial x^2})^2 |_{(x_j, t_j)} \quad (17)$$

Partial derivatives are calculated through numerical estimator given predicted solution $\tilde{u}$. Data points $\{(x_i, t_i)\}_{i=1}^{N_{data}}$ for data loss and $\{(x_j, t_j)\}_{j=1}^{N_{PDE}}$ for PDE residual are collected separately. λ is the coefficient for the regularization term. In PINN, PDEs are directly encoded as a regularization term by measuring the solution residuals to constrain the model parameters. In the context of human body pose estimation, physics mechanics is employed to ensure physically plausible estimations, where the Euler-Lagrange equation represented as an ODE is derived, and is encoded as a soft constraint for model regularization. By integrating the Euler-Lagrange equation into data-driven deep models, the estimated 3D body pose is ensured to be physically plausible [29].

Physics-guided functions derived from domain knowledge have been used for training deep models in a label-free manner. Stewart and Ermon [34] proposed a label-free supervision of NNs with physics equations. The goal of this paper was to supervise NNs by specifying constraints g that should hold over the output space f instead of using labels. The loss function then becomes

$$\hat{f}^* = \arg \min_{f \in F} \sum_{i=1}^{n} g(x_i, f(x_i)) + R(f) \quad (18)$$

where $R(f)$ refers to an additional regularization term penalizing model complexity. One example provided in the paper is to track an object performing free fall. The training of a regression network is formulated as a structured prediction problem operating on a sequence of N images (i.e., $(\mathcal{R}^{H \times W \times 3})^N \to \mathcal{R}^N$). Newton's second law of gravity (free fall motion) is represented as the algebraic equation and is directly incorporated into the loss function for training. In particular, for an object in free fall, its height at i-th time step with time interval Δt is computed as $y_i = y_0 + v_0(i\Delta t) + a(i\Delta t)^2$, with y_0 and v_0 being initial height and velocity. $a = -9.8m \backslash s^2$ is the fixed acceleration for an object performing free fall. Any predicted trajectory $f(\boldsymbol{x})$ hence should fit such parabola with fixed curvature. A loss is then defined measuring the fitting residual,

$$
\begin{aligned}
g(\boldsymbol{x}, f(\mathbf{x})) = g(f(\boldsymbol{x})) &= ||\hat{\boldsymbol{y}} - f(\boldsymbol{x})||_1 \\
&= ||\mathbf{a} + A(A^T A)^{-1} A^T (f(\mathbf{x}) - \mathbf{a}) - f(\boldsymbol{x})||_1
\end{aligned}
\tag{19}
$$

where

$$
A = \begin{bmatrix}
\triangle t & 1 \\
2\triangle t & 1 \\
3\triangle t & 1 \\
... & \\
N\triangle t & 1
\end{bmatrix}
\tag{20}
$$

and $\mathbf{a} = [a\triangle t^2, a(2\triangle t)^2, ..., a(N\triangle t)^2]$ with $a = -9.8m \backslash s^2$. Besides, the algorithmic supervisions derived based on well-established algorithms have been used for training a neural network such that direct supervisions from ground truth annotations are no longer required [100].

For all the related works discussed above, the relative importance between generic knowledge and data information is pre-defined by the design of training objective, and is not adjustable after training. However, the relative importance could vary with different inputs. For example, given an unseen input, data-based prediction is less reliable and knowledge is playing a more important role in the final prediction. Seo et al. [22] proposed a framework where the relative importance of generic knowledge compared to data information is adjusted by a controlling parameter α. The controlling parameter α is assumed to be a random variable following a pre-defined distribution $p(\alpha)$. Two sets of latent representations $\boldsymbol{z}_r$ and $\boldsymbol{z}_d$ extracted from input data $\boldsymbol{x}$ correspond to generic knowledge and data information, respectively. The final latent features are obtained as $\boldsymbol{z} = \alpha \boldsymbol{z}_r \oplus (1 - \alpha) \boldsymbol{z}_d$ and are used for generating final predictions $\hat{\boldsymbol{y}}$. Two sets of losses $\mathcal{L}_{rules}$ and $\mathcal{L}_{data}$ are defined based on generic knowledge and annotations from downstream tasks, respectively. $\mathcal{L}_{rules}$, being a function of inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$, measures the violation of the rules derived based on prior knowledge specific to a target downstream task (e.g., Eq. 14). The final training loss is computed as the expected loss over $p(\alpha)$, i.e.,

$$
\mathcal{L} = \mathbb{E}_{\alpha \sim p(\alpha)}[\alpha \mathcal{L}_{rule} + \rho(1 - \alpha)\mathcal{L}_{data}]
\tag{21}
$$

where ρ is a scale parameter to balance the units of two loss terms. Through the proposed framework, α denoting the relative importance becomes a variable during testing.

The physics-guided regularization terms discussed above are all explicitly defined over the output space of deep models. Physics-guided regularization terms can also be implicit (i.e., induced by a physics-based model embedded as one intermediate primitive of a neural network pipeline) [35]–[37]. For example, Wu et al. [36] constructed a system for understanding physical scenes without human annotations. The core of the system is a physical world representation that is first recovered by a perception module and then utilized by a simulation engine. The perception module is a deep neural network that is self-supervised without annotation. The simulation engine, consisting of a physics engine and graphics engine, is aimed at generating physics predictions. The proposed pipeline is
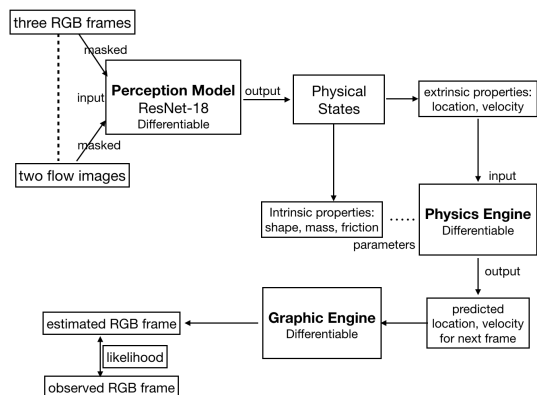

Fig. 6. Physical scene understanding without human annotations [36]. A physics engine and graphic engine are concatenated to a perception model. By aligning the rendered RGB images with observed ones, the perception model captures meaningful object features without annotations. The knowledge represented by the physics and graphic engines is encoded into the loss function implicitly.

evaluated on a synthetic billiard table experiment, as shown in Figure 6. Given the features extracted by the perception model, the physics engine predicts the future physics states of a system by following motion equations, and the graphic engine renders RGB images given the predicted physics states. The perception model is learned by maximizing the likelihood of estimated RGB images given observed sequences.

b) Regularization with generative deep models: Besides discriminative models, knowledge has also been applied to deep generative models. To generate realistic results, different generative models have explored the usage of domain knowledge. For example, to efficiently employ a generative adversarial network (GAN) for generating solutions of PDE-governed complex systems, Wu et al. [44] proposed a statistically constrained GAN, where a statistical regularization term is derived measuring the distance between covariance structures of training samples and generated samples, respectively,

$$
\mathcal{L}_c(D, G) = \mathcal{L}(D, G) + \lambda d(\Sigma(p_{data}), \Sigma(p_G))
\tag{22}
$$

where D and G represent standard discriminator and generator, respectively, and $L(D, G)$ represent the standard training loss for GAN. $\Sigma(p_{data})$ and $\Sigma(p_G)$ represents the covariance structures of the distributions of training data and generated samples, respectively. $d(\cdot)$ represents a distance measurement, such as the Frobenius norm. λ denotes the coefficient of the

regularization term. Introducing the statistical constraints on the lower dimensional manifold (i.e., covariance structure) helps reduce the searching space for finding desired solutions in high dimension. As a result, not only is the amount of data reduced, but the training time needed to converge on solutions is shorten as well, as demonstrated in solving turbulence PDEs. In the application to computational materials science, Shah et al. [43] proposed a deep generative model named InvNet, through which synthetic structured samples satisfying desired physical properties are generated. InvNet is an extension of conventional GAN, whereby an additional invariance checker is introduced, along with a traditional generator and discriminator. The invariance checker is introduced as an intermediate primitive, based on which an implicit knowledge-guided regularization term is defined. An invariance loss is defined based on the proposed invariance checker, measuring the violation of invariance. Through the invariance loss, the generated samples are encouraged to satisfy certain invariances (such as motif invariance, planting a predefined motif in all synthetic images at a fixed location). Besides knowledge represented as PDEs, physical connectivity and stability have been considered for realistic 3D shape generation. Mezghanni et al. [42] proposed incorporating physical constraints into a deep generative model, whereby the physical constraints capture both the connectivity of 3D components and the physical stability of the 3D shape. Fully differentiable physical loss terms are then defined for integrating physics constraints into neural networks. Specifically, a neural stability predictor, implemented as a neural network classifier and pre-trained with simulated data, is proposed for enforcing physical stability constraints. For each synthetic 3D shape in the simulated data, its stability is labeled by the Bullet physics engine. The stability constraint is encoded via the pre-trained stability classifier and is integrated into the deep model via the stability loss.

Variational AutoEncoder (VAE) has also been explored with physics integrated for robust and interpretable generative modeling [51], [52]. In particular, physics knowledge represented as PDEs is integrated into VAEs. Latent variables of VAEs are subject to the constraints defined by PDEs. Furthermore, in [50], instead of assuming a full access to the complete expression of PDEs, only part of the PDE is assumed to be known with latent variables of VAE being partially grounded with physics meaning for the known part of PDEs. The rest of unknown PDEs are modelled in a data-driven manner.

Leveraging probabilistic framework, uncertainty quantification has been considered in physics-informed deep models [45], [53]–[55]. Zhu et al. [54] considered a physics-informed CNN solving PDEs with uncertainty quantification. The uncertainty is originated from the randomness of the physics parameters $\boldsymbol{\lambda}$, which is denoted as random vector $\boldsymbol{x} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_{n_s}\}$, where n_s is the total number of possible physics parameter settings and can be very high. Correspondingly, the solution of PDE u with respect to each possible physics parameter setting becomes $\boldsymbol{y} = \{u_1, u_2, ..., u_{n_s}\}$. The task is to model $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ given a set of observations $\mathcal{D}_{input} = \{\boldsymbol{x}^{(i)}\}_{i=1}^N$ with $\boldsymbol{x}^{(i)} \sim p(\boldsymbol{x})$. $\boldsymbol{\theta}$ denote the neural network parameters to be learned. In the meantime, uncertainty

of PDE solutions is modeled via the variance $Var[\boldsymbol{y}]$. To train the NN, instead of using labeled data, an energy-based model is defined based solely on the PDE and its boundary conditions, from which we obtain the reference density. In particular, the reference density $p_\beta(\boldsymbol{y}|\boldsymbol{x})$ follows a Boltzmann-Gibbs distribution:

$$p_\beta(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(-\beta E(\boldsymbol{y}, \boldsymbol{x}))}{Z_\beta(\boldsymbol{x})} \quad (23)$$

with β being a learnable parameter of the constructed energy-based model. Energy function $E(\boldsymbol{y}, \boldsymbol{x}) = V_{PDE}(\boldsymbol{y}, \boldsymbol{x}) + \lambda V_{boundary}(\boldsymbol{y})$ measures the violation of the PDE and boundary conditions. λ is a tunable hyper-parameter. Physics equations are encoded into the energy-based probabilistic model. In the end, the NN is trained by minimizing the KL divergence between the estimated distribution and the reference distribution:

$$\min_{\beta, \boldsymbol{\theta}} D_{KL}(p(\boldsymbol{x})p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})||p(\boldsymbol{x})p_\beta(\boldsymbol{y}|\boldsymbol{x})) \quad (24)$$

Through the training process, physics equations are integrated into the NN. Along the same line, Yang and Perdikaris [53] simplified the objective function by considering the lower bound of Eq. (24). Following a similar idea, Karumuri et al. [55] employed a deep residual network (ResNet) for solving elliptic stochastic PDEs in a label-free manner. In particular, the physics-informed loss function is defined as the expectation of PDE residuals over the probability distribution of stochastic variables. All three works discussed above focus on time-independent physics systems without evolution over time. Geneva and Zabaras [45] extended the idea to a dynamic system, where an auto regressive network is employed for predicting future physics states given a history of system states.

IV. DEEP LEARNING WITH EXPERIENTIAL KNOWLEDGE

Besides scientific knowledge, experiential knowledge has been widely considered, as the major source of knowledge for neural-symbolic models. Experiential knowledge refers to well-known facts from everyday life, describing semantic properties of an object or semantic relationships among multiple objects. It is generally intuitive and is derived through long time observations or well-established studies. Unlike the scientific knowledge, experiential knowledge, though widely available, is descriptive and imprecise. Experiential knowledge containing semantic information can serve as a strong prior knowledge for predictive tasks in deep learning (e.g., regression or classification tasks), especially in the small-data regime where training data alone is insufficient in capturing relationships among variables [65].

A. Experiential Knowledge Identification

Depending on the application domain, experiential knowledge may manifest in two types: entity properties and entity relationships. Entity relationships reveal semantic relationships among entities. They may be inferred from daily facts on relationships between named entities or be derived from well-established studies or theories, e.g., anatomy. For example,
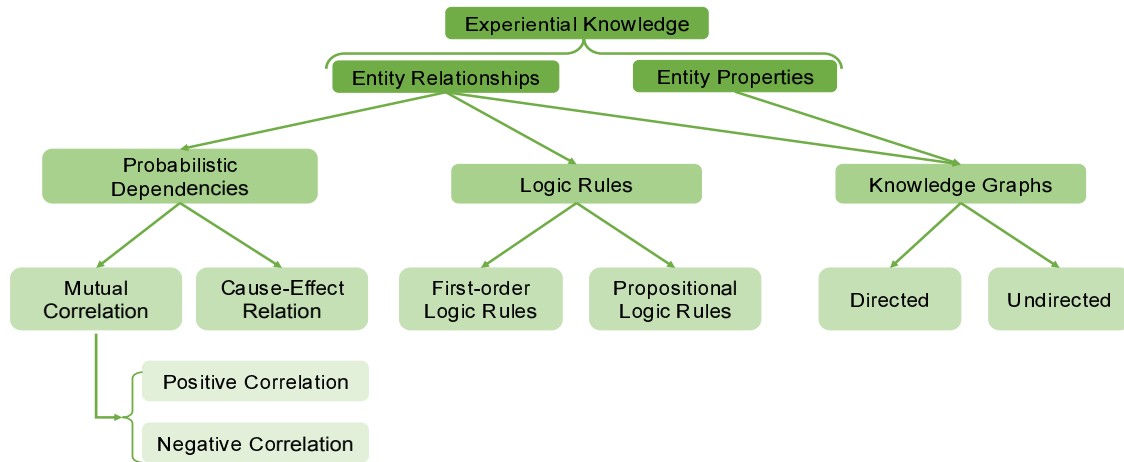
Fig. 7. Taxonomy of experiential knowledge and its representations.

human anatomy has been considered widely in computer vision for human body and facial behavior analysis [14], [110], [111]. For facial behavior analysis, facial anatomy knowledge may provide information on relationships among the facial muscles to produce natural facial expression. Similarly, body anatomy may provide information on the relationships among the body joints to produce stable and physically plausible body pose and movement. The semantic relationships can be directly given or indirectly inferred from the existing ones. For example, from the facts that `Helen Mirren acted in the Ink Heart` and `Helen Mirren wins the Best Actress award`, one can infer that `The Ink Heart is nominated for an award`. Inferred facts, however, are susceptible to errors. Entity properties capture the knowledge about properties of entities. They can refer to ontological information describing hierarchical relations of concepts in the world perceived by humans [112]. For example, `Rain is made of water`, and `Sea is a synonym for ocean`. Linguistic knowledge, a major source of experiential knowledge, is analyzed in [113], [114]. A large language model has been considered as inductive bias for an abstract textual reasoning task [115]. Linguistic knowledge such as textual explanations has been explored for language model refinement [116].

B. Representation of Experiential Knowledge

Representations of experiential knowledge vary with domains. In general, the representations for experiential knowledge include probabilistic dependencies, logic rules and knowledge graphs. These representations capture both relationships and properties of entities revealed by the experiential knowledge. We illustrate the taxonomy of experiential knowledge and its representations in Figure 7.

1) Probabilistic Dependencies: Due to inherent uncertainty, semantic relationships among objects are widely represented through probabilistic dependencies. States of objects are modelled in a probabilistic way, whereby relationships among objects are captured via probabilistic dependencies. Relationships can be further divided into positive correlations and negative

correlations. Let's take facial action units (AUs) as an example. According to FACS [117], AUs represent facial muscles and one facial muscle can control one or multiple AUs. A binary AU can be on if the corresponding muscle is activated. AU1 (inner brow raiser) and AU2 (outer brow raiser) usually occur together because they are controlled by the same muscle *frontalis*. The pair AU15 (lip corner depressor) and AU24 (lip pressor) is another example of positive correlation, which is due to the fact that their underlying controlling muscles (i.e., *depressor anguli oris* and *orbicularis oris*, respectively) always move together. If two variables X and Y are positively correlated (e.g., X="AU1" and Y="AU2") with $X = \{0, 1\}$ and $Y = \{0, 1\}$, then we have

$$\begin{aligned} p(X = 1, Y = 1) &> p(X = 1, Y = 0) \\ p(X = 1, Y = 1) &> p(X = 0, Y = 1) \end{aligned} \tag{25}$$

Considering negative correlation, AU12 (lip corner puller) and AU15 (lip corner depressor) cannot show up together as their corresponding muscles (i.e., *Zgomaticus major* and *depressor anguli oris*, respectively) are unlikely to be activated simultaneously. Negative correlation can be represented in a similar way. If two variables X and Y are negatively correlated (e.g., X="AU12" and Y="AU15") with $X = \{0, 1\}$ and $Y = \{0, 1\}$, then we have

$$\begin{aligned} p(X = 1, Y = 1) &< p(X = 1, Y = 0) \\ p(X = 1, Y = 1) &< p(X = 0, Y = 1) \end{aligned} \tag{26}$$

2) First-order and Propositional Logic: Logic can be categorized into first-order logic and propositional logic. First-order logic (FOL) [118] employs logic rules to infer new experiential knowledge from existing experiential knowledge; it has been employed as an inference method to derive different types of knowledge, e.g., textual explanations [116]. The formula of FOL is as follows:

$$R_1(\boldsymbol{x}) \wedge R_2(\boldsymbol{x}) \wedge \cdots \wedge R_n(\boldsymbol{x}) \Rightarrow H(\boldsymbol{x}) \tag{27}$$

where $R_1(\boldsymbol{x}), R_2(\boldsymbol{x}), \cdots R_n(\boldsymbol{x})$ represent logic atoms. Each atom captures a known object property or relationship through a predicate. The atoms are combined through connective (e.g.

conjunctive) operators to form the condition part of the logic rule. $H(\boldsymbol{x})$ is the implied result or conclusion of the logic rule. It represents the new experiential knowledge derived from the logic rule. The conditional part and the conclusion part of the rule are connected through the implication operator. For example, we have

$$\text{Smokes}(x) \Rightarrow \text{Cough}(x), \quad \forall x \qquad (28)$$

$\text{Smokes}(x)$ is an atom, with Smokes being a predicate and x as a logic variable. It captures the entity property knowledge that the person represented by variable x smokes or not. $\text{Cough}(x)$ is the implied result or derived knowledge, which captures the knowledge that the person coughs or not. The rule states that if the conditional part, $\text{Smokes}(x)$ is true, then $\text{Cough}(x)$ is also true.

3) Knowledge Graph: The knowledge graph is another symbolic representation of experiential knowledge, that is used primarily to capture the semantic relationships among objects, whereby semantic knowledge is expressed in triple format: (subject, predicate, object). The number of such triples is usually huge. In a knowledge graph, these triples are organized as a graph containing nodes and edges. Nodes represent subjects or objects, such as animals or places, as well as named entities, such as a person named Mary Kelley. Edges represent the predicate and connect pairs of nodes and describe the relationship between them. Besides, edges can also represent properties of an entity with nodes representing the attributes. Taking the triplet (cat, attribute, paw) as an example, nodes are cat and claw, and the relationship is attribute. This triple states a fact: "The attribute of a cat is paw". Edges can be directed or undirected, for example, the food chain relationship between animals or the social relationship between people. Knowledge graphs can encode a large amount of commonsense, rules, and domain knowledge that capture semantic relationships and properties about entities. A knowledge graph hence is an important basic resource for obtaining experiential knowledge. For example, experiential knowledge of the semantic meanings of objects can be organized in a knowledge graph for image classification [119]. The Miscrosoft Concept Graph [120] is another example of a knowledge graph, where vertices in the Miscrosoft Concept Graph [120] could represent food such as fruit, mammals such as dog and cat, or facilities such as bus and gas station. The edges indicate the relationships between concepts based on daily facts, such as a cat is a mammal, where is reflects relationship between cat and mammal.

C. Integration into Deep Models

To integrate experiential knowledge into deep models, existing methodologies cover four types of integration: data-level, architecture-level, training-level, and decision-level as shown in Figure 1. We introduce each group of the approaches in the following subsections.

1) Data-level Integration: Pseudo training data is usually considered to incorporate experiential knowledge represented as probabilistic dependencies, and is employed to augment existing training data. For example, Teshima and Sugiyama [65]

proposed to incorporate the conditional independence relationships among variables into predictive modeling. A set of conditional independence relationships among variables are firstly extracted from prior knowledge. Training data is then augmented by generating synthetic data that satisfies the extracted conditional independence relationships. Besides augmenting training data, pseudo data can also be employed for constructing a prior knowledge model. For facial AU recognition, Li et al. [64] proposed to leverage pseudo data generated based on the knowledge for constructing a data-free prior model that captures the prior distribution of the target variables for downstream tasks. Constraints on parameters and variables are firstly derived from generic AU knowledge. Effective sampling methods are then proposed for generating the pseudo data satisfying the variable and parameter constraints. A Bayesian network is then learned from pseudo data, serving as the data-free prior model. Similar idea has been exploited in upper body pose estimation task [14]. Four types of constraints (i.e., connectivity constraint, body length constraint, kinesiology constraint and symmetry constraint) are firstly derived from human anatomic knowledge. Synthetic data is then generated given these constraints, based on which a prior probabilistic model is learned.

Besides, synthetic data can be generated from generic knowledge represented as Boolean rules [22]. For instance, the probability of j-th class (i.e., $\hat{\boldsymbol{y}}_j$) is higher when k-th input feature is bigger than a constant a (i.e., $\boldsymbol{x}_k > a$). To incorporate this Boolean rule into a deep model, Seo et al. [22] proposed to augment each of the training data point $\boldsymbol{x}$ with a paired perturbed one $\boldsymbol{x}_p = \boldsymbol{x} + \delta\boldsymbol{x}$. $\delta\boldsymbol{x}$ is a small positive value for perturbation. The regularization is then be defined as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}_p, \hat{\boldsymbol{y}}_j, \hat{\boldsymbol{y}}_{p,j}) = \text{ReLU}(\hat{\boldsymbol{y}}_j - \hat{\boldsymbol{y}}_{p,j}) \cdot \mathbb{I}(\boldsymbol{x}_k < a) \cdot \mathbb{I}(\boldsymbol{x}_{p,k} > a) \quad (29)$$

If $\boldsymbol{x}_k < a$ and $\boldsymbol{x}_{p,k} > a$, according to the prior knowledge, we should have $\hat{\boldsymbol{y}}_j - \hat{\boldsymbol{y}}_{p,j} < 0$. If the constraint is satisfied, we have $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}_p, \hat{\boldsymbol{y}}_j, \hat{\boldsymbol{y}}_{p,j}) = 0$. Otherwise, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}_p, \hat{\boldsymbol{y}}_j, \hat{\boldsymbol{y}}_{p,j}) > 0$.

2) Architecture-level Integraion: Domain knowledge describing relationships between variables can be integrated into deep models through architecture design. We discuss the architecture-level integration methods for experiential knowledge represented as probabilistic dependencies, logic rules, and knowledge graphs, respectively.

a) Architecture design to incorporate probabilistic dependencies: One representative line of architecture-level integration is focused on the experiential knowledge represented as probabilistic dependencies, whereby a probabilistic model, constructed from the knowledge, serves as a prior model and is embedded as one layer of a neural network. Semantic relationships among variables can hence be incorporated into neural network in a probabilistic way. Usually, a probabilistic model is concatenated to the last layer of a neural network. A conditional random field (CRF) is usually employed, which takes hidden features from neural networks as input and outputs final predictions that satisfy the knowledge encoded in the CRF (e.g., among AUs [66]). In [67], a fully-connected CRF is concatenated to the last layer of a CNN to jointly perform facial landmark detection. By leveraging the fully-connected

CNN-CRF, probabilistic predictions of facial landmark locations are obtained capturing the structural dependencies among landmark points. For scene graph generation [68], structured relationships among entities and relations are firstly captured via an energy-based probabilistic model. The energy-based probabilistic model takes the output of a typical scene graph generation model as input and refines it by minimizing the energy. Prior knowledge captured via a probabilistic model can also be leveraged to define the adjacency matrix of a graph convolutional network [69]. For facial action unit (AU) density estimation task, a Bayesian network is employed to capture the inherent dependencies among AUs. A probabilistic graph convolution is then proposed whereby its adjacency matrix is defined by the structure of the Bayesian network. In addition, a probabilistic model can also be introduced as a learnable intermediate layer of a neural network. A CausalVAE [74] was proposed whereby a causal layer is introduced to the latent space of a variational auto-encoder (VAE). The causal layer essentially describes a structural causal model (SCM). Through the causal layer, independent exogenous factors are transformed to causal endogenous factors for causal representation learning.

b) Architecture design to incorporate logic rules: Integration through the architecture of neural network is a conventional neural-symbolic approach to integrate symbolic logic rules into deep models. Logic rules are integrated into neural network architectures by introducing logic variables or parameters. Such an approach can be traced back to the 1990s, when knowledge-based artificial neural network (KBANN) [121] and connectionist inductive learning and logic programming (CILP) [122] methods were introduced. More recently, logical neural network (LNN) [123] is proposed where each neuron represents an element in a logic formula, which can either be a concept (e.g., `cat`) or a logical connective (e.g., `AND`, `OR`). These works, however, are focused on leveraging neural networks for differentiable and scalable logic reasoning.

Few works are proposed to improve deep models via customizing their architectures via logic rules. To leverage logic rules for improved deep model performance, logic rules are encoded into a Markov Logic Network (MLN), and the constructed MLN, serving as a prior model, is embedded into a neural network as an output layer for improved knowledge graph completion task [75]. Particularly, four types of logic rules are firstly identified for capturing the knowledge in a Knowledge graph: (1) composition rules: a predicate R_k is composed of two predicates R_i and R_j if for any three variables Y_1, Y_2, Y_3 we have $R_i(Y_1, Y_2) \wedge R_j(Y_2, Y_2) \Rightarrow R_k(Y_1, Y_3)$; (2) Inverse rules: a predicate R_i is an inverse of R_j if for any two variables Y_1 and Y_2 we have $R_i(Y_1, Y_2) \Rightarrow R_j(Y_2, Y_1)$; (3) Symmetric rules: a predicate R is symmetric if for any two variables Y_1 and Y_2 we have $R(Y_1, Y_2) \Rightarrow R(Y_2, Y_1)$; (4) Subset rules: a predicate R_j is a subset of R_i if for any two variables Y_1 and Y_2 we have $R_j(Y_1, Y_2) \Rightarrow R_i(Y_2, Y_1)$. Given a set of identified logic rules $l \in \mathcal{L}$, a Markov logic network (MLN) defines a joint distribution of target variables y of a triple as

$$p(y) \propto \exp(\sum_{l \in \mathcal{L}} w_l \sum_y \phi_l(y_l)) \qquad (30)$$

where ϕ_l is the potential function and is computed given the observed triples. Such an MLN is then concatenated to the last layer of a deep model whereby the confidence scores w_l are learned via the deep model given the observed triples. By introducing the MLN, the task of predicting the missing triplets is re-formulated as inferring the posterior distribution of unseen configurations following the encoded logic rules.

c) Architecture design to incorporate knowledge graphs: The knowledge graph can also be integrated into the architecture of neural networks as one layer. Liang et al. [78] proposed a graph convolution with symbolic reasoning. Prior knowledge from the knowledge graph is specifically integrated into the neural network through a proposed symbolic graph reasoning (SGR) layer, as shown in Figure 8. To make the proposed SGR
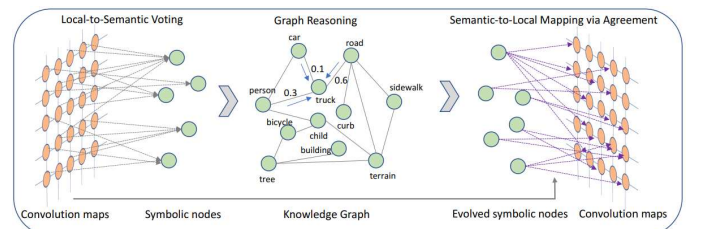

Fig. 8. Image is adopted from [78]. Nodes in the current convolutional layer are first mapped to symbolic nodes. Connectivities among the symbolic nodes are defined based on a knowledge graph. Through a graph reasoning based on the knowledge graph, evolved symbolic nodes are finally mapped to the nodes in the next convolutional layer.

cooperate with a convolutional layer, local hidden features from the current convolutional layer are firstly transferred to features of corresponding symbolic nodes in SGR. Connectivities among the symbolic nodes in SGR are defined based on prior knowledge from the knowledge graph. Guided by prior knowledge, SGR then performs graph reasoning and updates features. Finally, the updated features are mapped onto local features in the next convolutional layer. To generate a medical report in an unsupervised manner in a cross visual domain and textual domain, a knowledge-driven encoder-decoder model is proposed to leverage knowledge graphs [79]. The knowledge graph is encoded into the knowledge-driven attention module within the encoder. Taking an image and a medical report as inputs, the encoder first obtains the image embedding and the report embedding through standard deep models, respectively. An attention mechanism is then introduced whereby the embeddings are the queries and a knowledge graph is used to define the lookup matrix. The learned representations from the attention module bridge the vision and the textual domain by leveraging the knowledge graph. During training, the proposed model is learned by minimizing reconstruction error between generated and observed medical reports in the textual domain. During testing, the knowledge-driven encoder-decoder model can generate medical reports from medical images by leveraging experiential knowledge in knowledge graphs that is applicable to both visual and textual domains.

CRF has been leveraged to capture the experiential knowledge in a knowledge graph and is integrated as one layer of a neural network. Luo et al. [80] proposed a context-aware zero-shot recognition (CA-ZSL) method. Prior inter-object relation

is extracted from knowledge graphs and encoded using a conditional random field (CRF). For an image containing N objects, an image region and a class assignment of each object are denoted as B_i and c_i, respectively, with $i = 1, 2, ..., N$. The CRF model is then defined as

$$P(c_1, ..., c_N | B_1, ..., B_N)$$
$$\propto \exp(\sum_i \theta(c_i|B_i) + \gamma \sum_{i|\neq j} \phi(c_i, c_j|B_i, B_j)) \quad (31)$$

where the unary potential $\theta(c_i|B_i)$ is estimated given extracted features of each object correspondingly. Pairwise potential $\phi(c_i, c_j|B_i, B_j)$ is estimated using both the extracted features and the knowledge graph. Semantic relations extracted from the knowledge graph are encoded in the pairwise potential functions, where γ is a tunable hyper-parameter. Neural networks are trained by maximizing the log-likelihood. During testing, labels of unseen objects are inferred in a context-aware manner through a maximum-a-posteriori (MAP) inference in the learned CRF model.

3) Training-level Integration: Experiential knowledge is treated as prior bias that guides the training of a deep model. Constraints are obtained based on knowledge and are integrated as regularization terms into a deep model [77], [112], [124], [125]. Regularization can be derived from probabilistic dependencies, logic rules, or knowledge graphs, and we discuss further in the paragraphs below.

a) Regularization with probabilistic dependencies: Probabilistic dependencies derived from knowledge about semantic relationships are commonly integrated into deep models through regularization. Srinivas Kancheti et al. [70] considered causal domain priors for regularizing neural networks during training, whereby the learned causal effects in NNs are enforced to match the prior knowledge on causal relationships through regularization. Consider a neural network f with d inputs and C outputs, and for jth input, δG^j is a $C \times d$ matrix containing prior causal knowledge (in terms of gradients). To enforce f to be consistent with the prior knowledge, a regularization is defined

$$R = \frac{1}{N} \sum_{j=1}^{N} \max\{0, ||\nabla_j f \odot M - \delta G^j||_1 - \epsilon\} \quad (32)$$

where M is a $C \times d$ binary matrix indicating the availability of prior knowledge and N is the total number of training samples. $\nabla_j f$ is the $C \times d$ Jacobian of f w.r.t. jth input. ϵ indicates an acceptable error margin, and $\odot$ is the element-wise product. Similarly, Rieger et al. [71] proposed to penalize model explanations that did not align with prior knowledge through an explanation loss. For AU detection task, probabilistic relationships among AUs are derived from facial anatomic knowledge. Each of these probabilistic relationships is formulated as a constraint. In [72], a loss function measuring the satisfaction of each of these constraints is correspondingly defined and is employed for learning an AU detector. Differently, Cui et al. [73] proposed to learn a Bayesian network (BN) to compactly capture a large set of constraints on AU relationships. The BN is then used to construct the expected cross-entropy loss to train a deep neural network for AU detection.

b) Regularization with logic rules: Logical knowledge is encoded as constraints for model regularization. Through regularization, a deep model is penalized if its output violates the constraints derived from logical rules. Xu et al. [76] proposed to combine the automatic reasoning technology of propositional logic with existing deep learning models. Propositional logic was encoded in the loss function through the proposed semantic loss. A sentence α in propositional logic is defined over variables $\boldsymbol{X} = \{X_1, ..., X_n\}$. The sentence is the semantic constraint to be imposed on the outputs of a neural network. Suppose that p is a vector of probabilities, where each element p_i denotes the predicted probability of variable X_i and corresponds to a single output of the neural net. The semantic loss $L^s(\alpha, p)$ measures the violation of α given p as

$$L^s(\alpha, p) \propto -\log \sum_{\boldsymbol{x} \models \alpha} \prod_{i:\boldsymbol{x} \models X_i} p_i \prod_{i:\boldsymbol{x} \models \neg X_i} (1 - p_i) \quad (33)$$

$\boldsymbol{x} \models \alpha$ represents that the state x satisfies the sentence α. The larger the probability of the state satisfying the sentence, the smaller the semantic loss. The proposed semantic loss bridges neural network regularization with logic reasoning. It is effective for different applications, such as classification and preference ranking.

For relation prediction task, a logic embedding network with semantic regularization (LENSR) was proposed [77], where a propositional logic is integrated into a relation detection model. For a given image, the probability distribution of the relation predicate is first estimated using a standard visual relation detection model. Another probability distribution of the relation predicate is then proposed based on the propositional logic formula pre-defined given the input image. Finally, a semantic regularization is defined to align these two probability distributions by minimizing their difference.

c) Regularization with knowledge graph: The knowledge graph, a graphical representation of experiential knowledge, has also been employed for model regularization. Fang et al. [81] proposed to extract semantic consistency constraints from knowledge graph which are then used as regularization terms. Particularly, a consistency score between a pair of object O and subject S is calculated through random walk with restart

$$\mathcal{C}_{O,S} = \lim_{T \to \infty} p(Y_T = Y_S | Y_0 = Y_O; \alpha) \quad (34)$$

where $\mathcal{C} \in R^{|O| \times |S|}$. $|O|$ and $|S|$ represent the total number of target objects and target subjects. T is the total moving steps and α is the restart probability, meaning at each moving step, there is a probability of α to restart from the starting node instead of moving to one of the neighbors of node. The calculated matrix $\mathcal{C}$ is employed as constraints on semantic consistency and is used for regularizing the neural network for an object detection task. Similarly, Gu et al. [82] proposed extracting external knowledge from KGs and applying image reconstruction to improve scene graph generation, especially when the dataset is biased, or annotations are noisy or missing. Object relationships are retrieved from ConceptNet as the external domain knowledge and are applied to refine object features through an object-to-image generation branch. The

object-to-image generation branch reconstructs images based on object features and prior relational knowledge about the objects. Semantically meaningful object features can be learned by minimizing reconstruction error.

4) Decision-level Integration: Predictions from deep models and from the prior knowledge can be directly combined through the joint top-down and bottom-up prediction strategy. Through integrating two sets of predictions, the final prediction can be more accurate and robust.

For open-domain knowledge-based visual question answering, Marino et al. [83] combined implicit and symbolic knowledge. Implicit knowledge refers to knowledge learned from data (e.g., raw texts). Symbolic knowledge refers to graph-based knowledge encoded in existing knowledge graphs (e.g., DBpedia [126] and ConceptNet [127]). The proposed KRISP model contains two sub-modules: implicit knowledge reasoning and explicit knowledge reasoning. Two sources of knowledge are then combined to generate the final output by using a late-fusion strategy. Through the late-fusion strategy, predictions from data and symbolic knowledge are directly combined, independent of the training of deep models.

Two sets of predictions can be combined in a probabilistic way by following Bayes' rule. To obtain probabilistic knowledge-based predictions, a PGM model capturing prior knowledge is employed as a prior model and its prediction is obtained via a probabilistic inference. For AU recognition task, Li et al. [64] considered a top-down and bottom-up integration, where a Bayesian network learned from generic knowledge serves as the top-down model and a data-driven model serves as the bottom-up model. Predictions from two models are then combined using Bayes' rule to produce the final predictions. Probabilistic knowledge-based predictions can also be directly defined based on the knowledge. For a knowledge graph completion task, Cui et al. [84] derived a prior distribution on relations based on type information. The prior distribution is then combined with bottom-up predictions from existing embedding-based models through Bayes' rule for final prediction.

V. Discussion and Future Direction

In this survey, we reviewed traditional and prevailing technologies for knowledge-augmented deep learning, including knowledge identification, knowledge representation, and integration. We divided knowledge into two categories: scientific knowledge and experiential knowledge. Within each category, we introduced knowledge identification, representations, and integration into deep learning. As we have discussed, much work has been done to improve deep learning with related prior knowledge to yield data-efficient, generalizable, and interpretable deep learning models. To help readers better understand and apply the KADL to their work, we provide a prescriptive tree based on the summary of existing works (Table I). The prescriptive tree as shown in Figure 9 serves as a recipe that includes different pathways to incorporate a particular type of prior knowledge into deep models. Each pathway consists of a specific knowledge type, a knowledge representation format, and a knowledge integration method, accompanied with the references to related works.
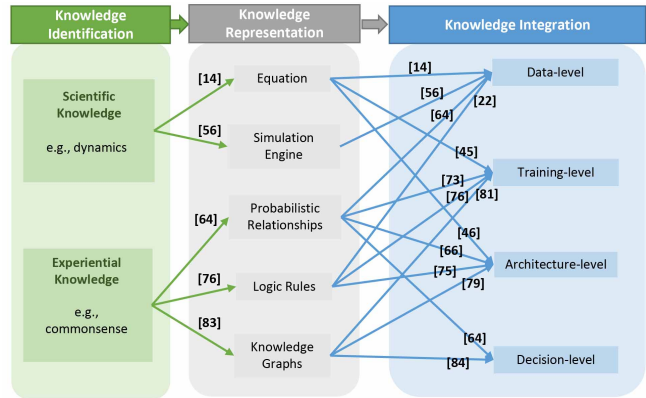

Fig. 9. Prescriptive tree on knowledge-augmented deep learning

Despite these efforts, existing methods suffer from several shortcomings. In the following paragraphs, we discuss existing techniques and highlight promising directions to pursue in the future.

a) Diverse types of knowledge: Existing knowledge-augmented deep models explore different types of domain knowledge, including scientific knowledge and experiential knowledge. However, most of the knowledge explored are scientific knowledge in physics and symbolically represented experiential knowledge. Injecting well-established algorithmic knowledge into deep models has already began to attract researchers' attention whereby deep models are trained through algorithmic supervisions instead of ground truth annotations [100]. Furthermore, existing approaches are usually limited to one specific type of knowledge. For a certain application task, both scientific knowledge and experiential knowledge could exists from multiple sources. Hence, diverse types of knowledge can be combined jointly for improved deep model performance.

b) Effective knowledge integration: Existing integration methods utilize synthetic data, model architecture design, regularization function, or prediction refinement. Among them, the majority of the integration methods are performed during training. Existing integration techniques hence heavily depend on the specific training procedure, through which deep models are trained by considering two sources of information jointly, without explicitly differentiating data from knowledge. The problem can be addressed through decision-level fusion. Decision-level knowledge integration scheme using a prior model to capture the domain knowledge attracts relatively less attention. Knowledge integration with a prior model to combine top-down prediction from the knowledge and bottom-up estimation from the data could be beneficial from several aspects. Firstly, the construction of a prior model is independent of deep models whereby deep models are initialized through observable data. Since the prior model and the deep model are built independent of the integration process, top-down and bottom-up integration process can be flexibly applied to any deep model and prior model. Secondly, knowledge integration is performed by following Bayes' rule in a principled manner. Refinement of data-based prediction based on knowledge becomes tractable and interpretable.

c) Hybrid integration methods: Existing methods tend to integrate scientific knowledge and experiential knowledge separately. In addition, they tend to employ one specific approach to perform the knowledge integration. For certain application domains, both types of knowledge may exist at the same time. Hence, they should be integrated jointly to further improve the performance of the deep models. In addition, users always need to choose an integration approach. There is no universal integration scheme that applies to all types of knowledge, and it remains an open question on how to automatically integrate knowledge with data in an optimal way. Hence, given the complementary nature of different integration methods, it may be beneficial to simultaneously employ different integration methods to exploit their respective strengths.

d) Knowledge integration with uncertainty: Existing works have explored the encoding of experiential knowledge in a probabilistic way such as capturing the uncertain relationships using a probabilistic graphical model. But in general, existing knowledge integration methods are deterministic methods, ignoring the underlying knowledge uncertainty and their impacts on deep model learning and inference. Uncertainty not only exists with experiential knowledge but also with scientific knowledge. For example, in physics, uncertainty arises from random physics parameters or unknown physics parameters or incomplete observations. Existing works along this line aim to measure the quality of PDE solutions governing a physics system and thus are subject to specific domain assumptions. Probabilistic tools, such as probabilistic graphical models (PGMs), are powerful in capturing uncertainties for experiential knowledge. However, few works explored the usage of PGMs in modeling uncertainties in scientific laws [128], [129]. It remains an open question to deep learning community on how to effectively and systematically model uncertainties in scientific knowledge for real-world application tasks.

REFERENCES

[1] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[2] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima *et al.*, "Neural-symbolic learning and reasoning: A survey and interpretation," *arXiv preprint arXiv:1711.03902*, 2017.

[3] D. Yu, B. Yang, D. Liu, and H. Wang, "A survey on neural-symbolic systems," *arXiv preprint arXiv:2111.08164*, 2021.

[4] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv:2003.04919*, vol. 1, no. 1, pp. 1–34, 2020.

[5] J. Han, L. Zhang *et al.*, "Integrating machine learning with physics-based modeling," *arXiv preprint arXiv:2006.02619*, 2020.

[6] R. Rai and C. K. Sahu, "Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus," *IEEE Access*, vol. 8, pp. 71 050–71 073, 2020.

[7] M. Rath and A. P. Condurache, "Boosting deep neural networks with geometrical prior knowledge: A survey," *arXiv preprint arXiv:2006.16867*, 2020.

[8] P. Nowack, P. Braesicke, J. Haigh, N. L. Abraham, J. Pyle, and A. Voulgarakis, "Using machine learning to build temperature-based ozone parameterizations for climate sensitivity simulations," *Environmental Research Letters*, vol. 13, no. 10, p. 104016, 2018.

[9] C. Deng, X. Ji, C. Rainey, J. Zhang, and W. Lu, "Integrating machine learning with human knowledge," *Iscience*, vol. 23, no. 11, p. 101656, 2020.

[10] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy *et al.*, "Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems," *arXiv preprint arXiv:1903.12394*, 2019.

[11] S. W. Kim, I. Kim, J. Lee, and S. Lee, "Knowledge integration into deep learning in dynamical systems: An overview and taxonomy," *Journal of Mechanical Science and Technology*, vol. 35, no. 4, pp. 1331–1342, 2021.

[12] L. von Rueden, S. Mayer, R. Sifa, C. Bauckhage, and J. Garcke, "Combining machine learning and simulation to a hybrid modelling approach: Current and future directions," in *International Symposium on Intelligent Data Analysis*. Springer, 2020, pp. 548–560.

[13] A. Sagel, A. Sahu, S. Matthes, H. Pfeifer, T. Qiu, H. Rueß, H. Shen, and J. Wörmann, "Knowledge as invariance–history and perspectives of knowledge-augmented machine learning," *arXiv preprint arXiv:2012.11406*, 2020.

[14] J. Chen, S. Nie, and Q. Ji, "Data-free prior model for upper body pose estimation and tracking," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4627–4639, 2013.

[15] P. Liu, L. Tian, Y. Zhang, S. Aylward, Y. Lee, and M. Niethammer, "Discovering hidden physics behind transport dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 082–10 092.

[16] M. Jaques, M. Burke, and T. M. Hospedales, "Newtonianvae: Proportional control and goal identification from pixels via physical latent spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4454–4463.

[17] A. Daw, R. Q. Thomas, C. C. Carey, J. S. Read, A. P. Appling, and A. Karpatne, "Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling," in *Proceedings of the 2020 siam international conference on data mining*. SIAM, 2020, pp. 532–540.

[18] N. Muralidhar, J. Bu, Z. Cao, L. He, N. Ramakrishnan, D. Tafti, and A. Karpatne, "Phynet: Physics guided neural networks for particle drag force prediction in assembly," in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 559–567.

[19] T. Beucler, S. Rasp, M. Pritchard, and P. Gentine, "Achieving conservation of energy in neural network emulators for climate modeling," *arXiv preprint arXiv:1906.06622*, 2019.

[20] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine, "Enforcing analytic constraints in neural networks emulating physical systems," *Physical Review Letters*, vol. 126, no. 9, p. 098302, 2021.

[21] L. Zhang, J. Han, H. Wang, R. Car, and E. Weinan, "Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics," *Physical review letters*, vol. 120, no. 14, p. 143001, 2018.

[22] S. Seo, S. O. Arik, J. Yoon, X. Zhang, K. Sohn, and T. Pfister, "Controlling neural networks with rule representations," *arXiv preprint arXiv:2106.07804*, 2021.

[23] J. Sun, Z. Niu, K. A. Innanen, J. Li, and D. O. Trad, "A theory-guided deep-learning formulation and optimization of seismic waveform inversion," *Geophysics*, vol. 85, no. 2, pp. R87–R99, 2020.

[24] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins, "Hamiltonian generative networks," *arXiv preprint arXiv:1909.13789*, 2019.

[25] A. Choudhary, J. F. Lindner, E. G. Holliday, S. T. Miller, S. Sinha, and W. L. Ditto, "Physics-enhanced neural networks learn order and chaos," *Physical Review E*, vol. 101, no. 6, p. 062207, 2020.

[26] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Neural networks with physics-informed architectures and constraints for dynamical systems modeling," *arXiv preprint arXiv:2109.06407*, 2021.

[27] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[28] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[29] Z. Li, J. Sedlar, J. Carpentier, I. Laptev, N. Mansard, and J. Sivic, "Estimating 3d motion and forces of person-object interactions from monocular video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8640–8649.

[30] S. J. Greydanus, M. Dzumba, and J. Yosinski, "Hamiltonian neural networks," 2019.

[31] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," *arXiv preprint arXiv:1909.12077*, 2019.

[32] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," *arXiv preprint arXiv:2003.04630*, 2020.

[33] C. Allen-Blanchette, S. Veer, A. Majumdar, and N. E. Leonard, "Lagnetvip: A lagrangian neural network for video prediction," *arXiv preprint arXiv:2010.12932*, 2020.

[34] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[35] F. Xie, A. Chowdhury, M. Kaluza, L. Zhao, L. L. Wong, and R. Yu, "Deep imitation learning for bimanual robotic manipulation," *arXiv preprint arXiv:2010.05134*, 2020.

[36] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum, "Learning to see physics via visual de-animation," *Advances in Neural Information Processing Systems*, vol. 30, pp. 153–164, 2017.

[37] Y.-L. Qiao, J. Liang, V. Koltun, and M. Lin, "Differentiable simulation of soft multi-body systems," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[38] A. Botev, A. Jaegle, P. Wirnsberger, D. Hennes, and I. Higgins, "Which priors matter? benchmarking models for learning latent dynamics," 2021.

[39] R. Wang, R. Walters, and R. Yu, "Incorporating symmetry into deep dynamics models for improved generalization," *arXiv preprint arXiv:2002.03061*, 2020.

[40] M. Finzi, G. Benton, and A. G. Wilson, "Residual pathway priors for soft equivariance constraints," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[41] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, 2016.

[42] M. Mezghanni, M. Boulkenafed, A. Lieutier, and M. Ovsjanikov, "Physically-aware generative network for 3d shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9330–9341.

[43] V. Shah, A. Joshi, S. Ghosal, B. Pokuri, S. Sarkar, B. Ganapathysubramanian, and C. Hegde, "Encoding invariances in deep generative models," *arXiv preprint arXiv:1906.01626*, 2019.

[44] J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, H. Xiao *et al.*, "Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems," *Journal of Computational Physics*, vol. 406, p. 109209, 2020.

[45] N. Geneva and N. Zabaras, "Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks," *Journal of Computational Physics*, vol. 403, p. 109056, 2020.

[46] B. Anderson, T.-S. Hy, and R. Kondor, "Cormorant: Covariant molecular neural networks," *arXiv preprint arXiv:1906.04015*, 2019.

[47] A. Lengyel, S. Garg, M. Milford, and J. C. van Gemert, "Zero-shot day-night domain adaptation with a physics prior," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4399–4409.

[48] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics," *arXiv preprint arXiv:1612.00341*, 2016.

[49] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," *arXiv preprint arXiv:1612.00222*, 2016.

[50] N. Takeishi and A. Kalousis, "Physics-integrated variational autoencoders for robust and interpretable generative modeling," *arXiv preprint arXiv:2102.13156*, 2021.

[51] Y. Yin, V. Le Guen, J. Dona, E. de Bézenac, I. Ayed, N. Thome, and P. Gallinari, "Augmenting physical models with deep networks for complex dynamics forecasting," *Journal of Statistical Mechanics: Theory and Experiment*, 2021.

[52] O. Linial, N. Ravid, D. Eytan, and U. Shalit, "Generative ode modeling with known unknowns," in *Proceedings of the Conference on Health, Inference, and Learning*, 2021, pp. 79–94.

[53] Y. Yang and P. Perdikaris, "Adversarial uncertainty quantification in physics-informed neural networks," *Journal of Computational Physics*, vol. 394, pp. 136–152, 2019.

[54] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *Journal of Computational Physics*, vol. 394, pp. 56–81, 2019.

[55] S. Karumuri, R. Tripathy, I. Bilionis, and J. Panchal, "Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks," *Journal of Computational Physics*, vol. 404, p. 109120, 2020.

[56] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar, "Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles," in *Proceedings of the 2019 SIAM International Conference on Data Mining*.   SIAM, 2019, pp. 558–566.

[57] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, and V. Kumar, "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles," *ACM/IMS Transactions on Data Science*, vol. 2, no. 3, pp. 1–26, 2021.

[58] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE international conference on robotics and automation (ICRA)*.   IEEE, 2018, pp. 4243–4250.

[59] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*.   Springer, 2018, pp. 621–635.

[60] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[61] K. Ehsani, S. Tulsiani, S. Gupta, A. Farhadi, and A. Gupta, "Use the force, luke! learning to predict physical forces by simulating effects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 224–233.

[62] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi, "Newtonian scene understanding: Unfolding the dynamics of objects in static images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3521–3529.

[63] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*.   IEEE, 2017, pp. 23–30.

[64] Y. Li, J. Chen, Y. Zhao, and Q. Ji, "Data-free prior model for facial action unit recognition," *IEEE Transactions on affective computing*, vol. 4, no. 2, pp. 127–141, 2013.

[65] T. Teshima and M. Sugiyama, "Incorporating causal graphical prior knowledge into predictive modeling via simple data augmentation," in *Uncertainty in Artificial Intelligence*.   PMLR, 2021, pp. 86–96.

[66] C. Corneanu, M. Madadi, and S. Escalera, "Deep structure inference network for facial action unit recognition," in *Proceedings of European Conference on Computer Vision*, 2019.

[67] L. Chen, H. Su, and Q. Ji, "Deep structured prediction for facial landmark detection," *Advances in neural information processing systems*, vol. 32, 2019.

[68] M. Suhail, A. Mittal, B. Siddiquie, C. Broaddus, J. Eledath, G. Medioni, and L. Sigal, "Energy-based learning for scene graph generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 936–13 945.

[69] T. Song, Z. Cui, W. Zheng, and Q. Ji, "Hybrid message passing with performance-driven structures for facial action unit detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6267–6276.

[70] S. Srinivas Kancheti, A. Gowtham Reddy, V. N. Balasubramanian, and A. Sharma, "Matching learned causal effects of neural networks with domain priors," *arXiv e-prints*, pp. arXiv–2111, 2021.

[71] L. Rieger, C. Singh, W. Murdoch, and B. Yu, "Interpretations are useful: penalizing explanations to align neural networks with prior knowledge," in *International conference on machine learning*.   PMLR, 2020, pp. 8116–8126.

[72] Y. Zhang, W. Dong, B.-G. Hu, and Q. Ji, "Classifier learning with prior probabilities for facial action unit recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5108–5116.

[73] Z. Cui, T. Song, Y. Wang, and Q. Ji, "Knowledge augmented deep neural networks for joint facial expression and action unit recognition," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[74] M. Yang, F. Liu, Z. Chen, X. Shen, J. Hao, and J. Wang, "Causalvae: Disentangled representation learning via neural structural causal models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9593–9602.

[75] M. Qu and J. Tang, "Probabilistic logic neural networks for reasoning," *arXiv preprint arXiv:1906.08495*, 2019.

[76] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck, "A semantic loss function for deep learning with symbolic knowledge," in *International conference on machine learning*.   PMLR, 2018, pp. 5502–5511.

[77] Y. Xie, Z. Xu, M. S. Kankanhalli, K. S. Meel, and H. Soh, "Embedding symbolic knowledge into deep networks," *arXiv preprint arXiv:1909.01161*, 2019.

[78] X. Liang, Z. Hu, H. Zhang, L. Lin, and E. P. Xing, "Symbolic graph reasoning meets convolutions," *Advances in Neural Information Processing Systems*, vol. 31, pp. 1853–1863, 2018.

[79] F. Liu, C. You, X. Wu, S. Ge, X. Sun *et al.*, "Auto-encoding knowledge graph for unsupervised medical report generation," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[80] R. Luo, N. Zhang, B. Han, and L. Yang, "Context-aware zero-shot recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 11 709–11 716.

[81] Y. Fang, K. Kuan, J. Lin, C. Tan, and V. Chandrasekhar, "Object detection meets knowledge graphs." International Joint Conferences on Artificial Intelligence, 2017.

[82] J. Gu, H. Zhao, Z. Lin, S. Li, J. Cai, and M. Ling, "Scene graph generation with external knowledge and image reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1969–1978.

[83] K. Marino, X. Chen, D. Parikh, A. Gupta, and M. Rohrbach, "Krisp: Integrating implicit and symbolic knowledge for open-domain knowledge-based vqa," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 111–14 121.

[84] Z. Cui, P. Kapanipathi, K. Talamadupula, T. Gao, and Q. Ji, "Type-augmented relation prediction in knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 7151–7159.

[85] D. Kaur, S. Uslu, K. J. Rittichier, and A. Durresi, "Trustworthy artificial intelligence: a review," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–38, 2022.

[86] Z. Wang, W. Xing, R. Kirby, and S. Zhe, "Physics informed deep kernel learning," 2020.

[87] M. M. Sultan, H. K. Wayment-Steele, and V. S. Pande, "Transferable neural networks for enhanced sampling of protein dynamics," *Journal of chemical theory and computation*, vol. 14, no. 4, pp. 1887–1894, 2018.

[88] O. T. Unke, M. Bogojeski, M. Gastegger, M. Geiger, T. Smidt, and K.-R. Müller, "Se (3)-equivariant prediction of molecular wavefunctions and electronic densities," *arXiv preprint arXiv:2106.02347*, 2021.

[89] Y. Yuan, S.-E. Wei, T. Simon, K. Kitani, and J. Saragih, "Simpoe: Simulated character control for 3d human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7159–7169.

[90] S. Shimada, V. Golyanik, W. Xu, P. Pérez, and C. Theobalt, "Neural monocular 3d human motion capture with physical awareness," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–15, 2021.

[91] P. W. Anderson, "More is different," *Science*, vol. 177, no. 4047, pp. 393–396, 1972.

[92] M. Born and E. Wolf, *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013.

[93] M. Oren and S. K. Nayar, "Generalization of the lambertian model and implications for machine vision," *International Journal of Computer Vision*, vol. 14, no. 3, pp. 227–251, 1995.

[94] J. T. Kajiya, "The rendering equation," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 143–150.

[95] X. Cao, Z. Chen, A. Chen, X. Chen, S. Li, and J. Yu, "Sparse photometric 3d face reconstruction guided by morphable models," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4635–4644.

[96] H. Wang, S. Z. Li, Y. Wang, and W. Zhang, "Illumination modeling and normalization for face recognition," in *2003 IEEE International SOI Conference. Proceedings (Cat. No. 03CH37443)*. IEEE, 2003, pp. 104–111.

[97] S. Xin, S. Nousias, K. N. Kutulakos, A. C. Sankaranarayanan, S. G. Narasimhan, and I. Gkioulekas, "A theory of fermat paths for non-line-of-sight shape reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6800–6809.

[98] S. Kim, Y. Huo, and S.-E. Yoon, "Single image reflection removal with physically-based training images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5164–5173.

[99] C. Zhou, M. Teng, Y. Han, C. Xu, and B. Shi, "Learning to dehaze with polarization," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[100] F. Petersen, C. Borgelt, H. Kuehne, and O. Deussen, "Learning with algorithmic supervision via continuous relaxations," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[101] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[102] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," *arXiv preprint arXiv:1907.04490*, 2019.

[103] V. G. Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," 2019.

[104] Y. Zhang, J. Sun, X. He, H. Fu, R. Jia, and X. Zhou, "Modeling indirect illumination for inverse rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 18 643–18 652.

[105] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, 2020.

[106] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi, ""what happens if..." learning to predict the effect of forces in images," in *European conference on computer vision*. Springer, 2016, pp. 269–285.

[107] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," *Advances in neural information processing systems*, vol. 28, pp. 127–135, 2015.

[108] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[109] P. Kubelka and F. Munk, "An article on optics of paint layers," *Z. Tech. Phys*, vol. 12, no. 593-601, pp. 259–274, 1931.

[110] M. Haker, M. Böhme, T. Martinetz, and E. Barth, "Self-organizing maps for pose estimation with a time-of-flight camera," in *Workshop on Dynamic 3D Imaging*. Springer, 2009, pp. 142–153.

[111] T. Chen, C. Fang, X. Shen, Y. Zhu, Z. Chen, and J. Luo, "Anatomy-aware 3d human pose estimation with bone-based pose decomposition," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[112] A. Li, T. Luo, Z. Lu, T. Xiang, and L. Wang, "Large-scale few-shot learning: Knowledge transfer with class hierarchy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7212–7220.

[113] R. Yu, A. Li, V. I. Morariu, and L. S. Davis, "Visual relationship detection with internal and external linguistic knowledge distillation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1974–1982.

[114] A. Akula, V. Jampani, S. Changpinyo, and S.-C. Zhu, "Robust visual reasoning via language guided neural module networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[115] C. Rytting and D. Wingate, "Leveraging the inductive bias of large language models for abstract textual reasoning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[116] H. Yao, Y. Chen, Q. Ye, X. Jin, and X. Ren, "Refining language models with compositional explanations," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[117] P. Ekman and W. V. Friesen, "Facial action coding system," *Environmental Psychology & Nonverbal Behavior*, 1978.

[118] H. B. Enderton, *A mathematical introduction to logic*. Elsevier, 2001.

[119] C. Menglong, J. Detao, Z. Ting, Z. Dehai, X. Cheng, C. Zhibo, and X. Xiaoqiang, "Image classification based on image knowledge graph and semantics," in *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2019, pp. 81–86.

[120] L. Ji, Y. Wang, B. Shi, D. Zhang, Z. Wang, and J. Yan, "Microsoft concept graph: Mining semantic concepts for short text understanding," *Data Intelligence*, vol. 1, no. 3, pp. 238–270, 2019.

[121] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial intelligence*, vol. 70, no. 1-2, pp. 119–165, 1994.

[122] A. S. A. Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Applied Intelligence*, vol. 11, no. 1, pp. 59–77, 1999.

[123] R. Riegel, A. Gray, F. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma *et al.*, "Logical neural networks," *arXiv preprint arXiv:2006.13155*, 2020.

[124] I. Donadello, L. Serafini, and A. D. Garcez, "Logic tensor networks for semantic image interpretation," *arXiv preprint arXiv:1705.08968*, 2017.

[125] M. Diligenti, M. Gori, and C. Sacca, "Semantic-based regularization for learning and inference," *Artificial Intelligence*, vol. 244, pp. 143–165, 2017.

[126] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[127] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[128] C. G. Enright, M. G. Madden, and N. Madden, "Bayesian networks for mathematical models: techniques for automatic construction and efficient inference," *International Journal of Approximate Reasoning*, vol. 54, no. 2, pp. 323–342, 2013.

[129] S. Evers and P. J. Lucas, "Constructing bayesian networks for linear dynamic systems," *BMAW-11 Preface*, 2011.

Zijun Cui received the B.S. degree from the Department of Physics, University of Science and Technology of China, Hefei, China, in 2015, and received the M.S. degree from the School of Engineering, Brown University, Rhode Island, USA, in 2017. She is currently pursuing the Ph.D. degree with the Rensselaer Polytechnic Institute, NY, USA. She has broad experiences with deep learning and probabilistic graphical models. Her current research interests include knowledge-augmented deep learning, learning and inference on probabilistic graphical models, and their applications to computer vision and natural language processing.

Tian Gao received the Ph.D and B.S degrees both from the Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA. He is currently a research staff member of IBM Research AI at T. J. Watson Research Center. His research focuses on machine learning and its application in computer vision and natural language processing. He has worked on probabilistic graphical models, knowledge extraction, causal discovery, and other aspects of machine learning. From 2010 to 2012, he was a National Science Foundation Triple Helix Program Fellow.

Kartik Talamadupula is a Senior Research Scientist and Research Manager at IBM Research AI. His background is in automated planning and sequential decision making. He has applied decision-making techniques to human-in-the-loop AI, data-driven dialog, and human-agent collaboration.

Qiang Ji received his Ph.D degree in Electrical Engineering from the University of Washington. He is currently a Professor with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute (RPI). From 2009 to 2010, he served as a program director at the National Science Foundation (NSF), Arlington, VA, USA, where he managed NSF's computer vision and machine learning programs. He also held teaching and research positions with the Beckman Institute at University of Illinois at Urbana-Champaign, Urbana, IL, USA; the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, USA; the Dept. of Computer Science at University of Nevada, Reno, Nevada, USA; and the Air Force Research Laboratory, Rome, NY, USA. Prof. Ji currently serves as the director of the Intelligent Systems Laboratory (ISL) at RPI. Prof. Ji's research interests are in human-centered computer vision, probabilistic graphical models, probabilistic deep learning, and their applications in various fields. He has published over 300 papers in peer-reviewed journals and conferences, and has received multiple awards for his work. Prof. Ji is has served as an editor on several related IEEE and international journals and as a general chair, program chair, technical area chair, and program committee member for numerous international conferences/workshops. Prof. Ji is a fellow of the IEEE and the IAPR.