Chapter 2

Preliminaries and Background

This is an excerpt of Chapter 2 from https://github.com/bamos/thesis

This section provides a broad overview of foundational ideas and background material relevant to this thesis. In most chapters of this thesis, we include a deeper discussion of the related literature relevant to that material.

2.1 Preliminaries

The content in this thesis builds on the following topics. We assume preliminary knowledge of these topics and give a limited set of key references here. The reader should have an understanding of statistical and machine learning modeling paradigms as described in Wasserman [Was13], Bishop [Bis07], and Friedman, Hastie, and Tibshirani [FHT01]. Our contributions mostly focus on end-to-end modeling with deep architectures as described in Schmidhuber [Sch15] and Goodfellow, Bengio, Courville, and Bengio [Goo+16] with applications in computer vision as described in Forsyth and Ponce [FP03], Bishop [Bis07], and Szeliski [Sze10]. Our contributions also involve optimization theory and applications as described in Bertsekas [Ber99], Boyd and Vandenberghe [BV04], Bonnans and Shapiro [BS13], Griewank and Walther [GW08], Nocedal and Wright [NW06], Sra, Nowozin, and Wright [SNW12], and Wright [Wri97]. One application area of this thesis work focuses on control and reinforcement learning. Control is one kind of optimization-based modeling and is further described in Bertsekas, Bertsekas, Bertsekas, and Bertsekas [Ber+05], Sastry and Bodson [SB11], and Levine [Lev17b]. Reinforcement learning methods are summarized in Sutton, Barto, et al. [SB+98] and Levine [Lev17a].

2.2 Energy-based Learning

Energy-based learning is a machine learning method typically used in supervised settings that explicitly adds relationships and dependencies to the model's output space. This is in contrast to purely feed-forward models that typically cannot explicitly capture dependencies in the output space. At the core of energy-based learning methods is a scalar-valued *energy* function $E_\theta(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ parameterized by θ that measures the fit between some input x and output y. Inference in energy-based models is done by solving the optimization problem

$$\hat{y} = \operatorname*{argmin}_y E_\theta(x, y). \tag{2.1}$$

We note that this is a powerful formulation for modeling and learning and subsumes the representational capacity of standard deep feedforward models, which we show how to do in Section 2.2.1. The energy function can also be interpreted from a probabilistic lens as the negated unnormalized joint distribution over the input and output spaces.

Energy-based methods have been in use for over a decade and the tutorial LeCun, Chopra, Hadsell, Ranzato, and Huang [LeC+06] overviews many of the foundational methods and challenges in energy-based learning. The two main challenges for energy-based learning are 1) learning the parameters θ of the energy function E_θ and 2) efficiently solving the inference procedure in Equation (2.1). These challenges have historically been tamed by using simpler energy functions consisting of hand-engineered feature extractors for the inputs x and linear functions of y. This captures models such as Markov random fields [Li94] and conditional random fields [LMP01; SM+12]. Standard gradient-based methods are difficult to use for parameter learning because $\hat{y}$ depends on θ through the argmin operator, which is not always differentiable. Historically, a common approach to doing parameter learning in energy-based models has been to directly shape the energy function with a max-margin approach Taskar, Guestrin, and Koller [TGK04] and Taskar, Chatalbashev, Koller, and Guestrin [Tas+05].

More recently, there has been a strong push to further incorporate structured prediction methods like conditional random fields as the "last layer" of a deep network architecture [PBX09; Zhe+15; Che+15] as well as in deeper energy-based architectures [BM16; BYM17; Bel17; WFU16]. We further discuss Structured Prediction Energy Networks (SPENs) in Section 2.2.2.

An ongoing discussion in the community argues whether adding the dependencies explicitly in an energy-based is useful or not. Feedforward models have a remarkable representational capacity that can implicitly learn the dependencies and relationships from data without needing to impose additional structure or modeling assumptions and without making the model more computationally expensive with an optimization-based inference procedure. One argument against this viewpoint that supports energy-based modeling is that explicitly including modeling information improves the data efficiency and requires less samples to learn because some structure and knowledge is already present in the model and does not have to be learned from scratch.

2.2.1 Energy-based Models Subsume Feedforward Models

We highlight the power of energy-based modeling for supervised learning by noting how they subsume deep feedforward models. Let $\hat{y} = f_\theta(x)$ be a deep feedforward model. The energy-based representation of this model is $E(x, y) = ||y - f_\theta(x)||_2^2$ and inference becomes the convex optimization problem $\hat{y} = \operatorname{argmin}_y E(x, y)$, which has the exact solution $\hat{y} = f_\theta(x)$. An energy function that has more structure over the output space adds representational capacity that a feedforward model wouldn't be able to capture explicitly.

2.2.2 Structured Prediction Energy Networks

Structured Prediction Energy Networks (SPENs) [BM16; BYM17; Bel17] are a way of bridging the gap between modern deep learning methods and classical energy-based learning methods. SPENs provide a deep structure over input and output spaces by representing the energy function $E_\theta(x, y)$ with a standard feed-forward neural network. This expressive formulation comes at the cost of making the inference procedure in Equation (2.1) difficult and non-convex. SPENs typically use an approximate inference procedure by taking a fixed-number of gradient descent steps for inference. For learning, SPENs typically replace the inference with an *unrolled gradient-based optimizer* that starts with some prediction y_0 and takes a fixed number of gradient steps to minimize the energy function

$$y_{i+1} = y_i - \alpha \nabla_y E_\theta(x, y_i).$$

The final iterate as then taken as the prediction $\hat{y} \triangleq y_N$. Gradient-based parameter learning can be done by differentiating the prediction $\hat{y}$ with respect to θ by unrolling the inference procedure. Unrolling the inference procedure can be done in most autodiff frameworks such as PyTorch [Pas+17b] or TensorFlow [Aba+16]. activation functions with smooth first derivatives such as the sigmoid or softplus [GBB11] should be used to avoid discontinuities because unrolling the inference procedure involves computing $\nabla_\theta \nabla_y E_\theta(x, y)$.

2.3 Modeling with Domain-Specific Knowledge

The role of domain-specific knowledge in the machine learning and computer vision fields has been an active discussion topic over the past decade and beyond. Historically, domain knowledge such as fixed hand-crafted feature and edge detectors were rigidly part of the computer vision pipeline and have been overtaken by learnable convolutional models LeCun, Cortes, and Burges [LCB98] and Krizhevsky, Sutskever, and Hinton [KSH12]. To highlight the power of convolutional architectures, they provide a reasonable prior for vision tasks even without learning [UVL18]. Machine learning models extend far beyond the reach of vision tasks and the community has a growing interest on domain-specific priors rather than just using fully-connected architectures. These priors ideally can be integrated as end-to-end learnable modules into a larger system that are learned as a whole with gradient-based information. In contrast to pure fully-connected architectures, specialized submodules ideally improve the data efficiency of the model, add interpretability, and enable grey-box verification.

Recent work has gone far beyond the classic examples of adding modeling priors by using convolutional or sequential models. A full discussion of all of the recent improvements is beyond the scope of this thesis, and here we highlight a few key recent developments.

- Differentiable beam search [Goy+18] and differentiable dynamic programming [MB18]

- Differentiable protein simulator [Ing+18]
- Differentiable particle filters [JRB18]
- Neural ordinary differential equations [Che+18] and applications to reversible generative models [Gra+18]
- Relational reasoning on sets, graphs, and trees [Bat+18; Zah+17; KW16; Gil+17; San+17; HYL17; Bat+16; Xu+18; Far+17; She+18]
- Geometry-based priors [Bro+17; Gul+18; Mon+17; TT18; Li+18]
- Memory [SWF+15; GWD14; Gra+16; XMS16; Hil+15; PS17]
- Attention [BCB14; Vas+17; Wan+18]
- Capsule networks [SFH17; HSF18; XC18]
- Program synthesis [RD15; NLS15; Bal+16; Dev+17; Par+16]

2.4 Optimization-based Modeling

Optimization can be used for modeling in machine learning. Among many other applications, these architectures are well-studied for generic classification and structured prediction tasks [Goo+13; SRE11; BSS13; LeC+06; BM16; BYM17]; in vision for tasks such as denoising [Tap+07; SR14] or edge-aware smoothing [BP16]. Diamond, Sitzmann, Heide, and Wetzstein [Dia+17] presents unrolled optimization with deep priors. Metz, Poole, Pfau, and Sohl-Dickstein [Met+16] uses unrolled optimization within a network to stabilize the convergence of generative adversarial networks [Goo+14]. Indeed, the general idea of solving restricted classes of optimization problem using neural networks goes back many decades [KC88; Lil+93], but has seen a number of advances in recent years. These models are often trained by one of the following four methods.

2.4.1 Explicit Differentiation

If an analytic solution to the argmin can be found, such as in an unconstrained quadratic minimization, the gradients can often also be computed analytically. This is done in Tappen, Liu, Adelson, and Freeman [Tap+07] and Schmidt and Roth [SR14]. We cannot use these methods for the constrained optimization problems we consider in this thesis because there are no known analytic solutions.

2.4.2 Unrolled Differentiation

The argmin operation over an unconstrained objective can be approximated by a first-order gradient-based method and unrolled. These architectures typically introduce an optimization procedure such as gradient descent into the inference procedure. This is done in Domke [Dom12], Belanger, Yang, and McCallum [BYM17], Metz, Poole, Pfau, and Sohl-Dickstein [Met+16], Goodfellow, Mirza, Courville, and Bengio [Goo+13], Stoyanov, Ropson, and Eisner [SRE11], Brakel, Stroobandt, and Schrauwen [BSS13], and Finn, Abbeel, and Levine

[FAL17]. The optimization procedure is unrolled automatically or manually [Dom12] to obtain derivatives during training that incorporate the effects of these in-the-loop optimization procedures.

Given an unconstrained optimization problem with a parameterized objective

$$\operatorname*{argmin}_x f_\theta(x),$$

gradient descent starts at an initial value x_0 and takes steps

$$x_{i+1} = x_i - \alpha \nabla_x f_\theta(x).$$

For learning, the final iterate of this procedure x_N can be taken as the output and $\partial x_N / \partial \theta$ can be computed with automatic differentiation.

In all of these cases, the optimization problem is unconstrained and unrolling gradient descent is often easy to do. When constraints are added to the optimization problem, iterative algorithms often use a projection operator that may be difficult to unroll through and storing all of the intermediate iterates may become infeasible.

2.4.3 Implicit argmin differentiation

Most closely related to this thesis work, there have been several applications of the implicit function theorem to differentiating through constrained convex argmin operations. These methods typically parameterize an optimization problem's objective or constraints and then applies the *implicit function theorem* (Theorem 1) to optimality conditions of the optimization problem that implicitly define the solution, such as the *KKT conditions* [BV04, Section 5.5.3]. We will first review the implicit function theorem and KKT conditions and then discuss related work in this space.

Implicit function analysis [DR09] typically focuses on solving an equation $f(p, x) = 0$ for x as a function s of p, *i.e.* $x = s(p)$. *Implicit differentiation* considers how to differentiate the solution mapping with respect to the parameters, *i.e.* $\nabla_p s(p)$. The *implicit function theorem* used in standard calculus textbooks can be traced back to the lecture notes from 1877-1878 of Dini [Din77] and is presented in Dontchev and Rockafellar [DR09, Theorem 1.B.1] as follows.

Theorem 1 (Implicit function theorem)**.** *Let $f : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable in a neighborhood of $(\bar{p}, \bar{x})$ and such that $f(\bar{p}, \bar{x}) = 0$, and let the partial Jacobian of f with respect to x at $(\bar{p}, \bar{x})$, namely $\nabla_x f(\bar{p}, \bar{x})$, be nonsingular. Then the solution mapping $S(p) = \left\{ x \in \mathbb{R}^n \;\middle|\; f(p, x) = 0 \right\}$ has a single-valued localization s around $\bar{p}$ for $\bar{x}$ which is continuously differentiable in a neighborhood Q of $\bar{p}$ with Jacobian satisfying $\nabla s(p) = -\nabla_x f(p, s(p))^{-1} \nabla_p f(p, s(p))$ for every $p \in Q$.*

In addition to the content in this thesis, several other papers apply the implicit function theorem to differentiate through the argmin operators. This approach frequently comes up in bilevel optimization [Gou+16; KP13] and sensitivity analysis [Ber99; FI90; BB08; BS13]. [Bar18] is a note on applying the implicit function theorem to the KKT conditions of convex optimization problems and highlights assumptions behind the derivative being well-defined. Gould, Fernando, Cherian, Anderson, Santa Cruz, and Guo [Gou+16] describes general techniques for differentiation through optimization problems, but only describe the case of exact equality constraints rather than both equality and inequality constraints (they add inequality constraints via a barrier function). Johnson, Duvenaud, Wiltschko, Adams, and Datta [Joh+16] performs implicit differentiation on (multi-)convex objectives with coordinate subspace constraints. The older work of Mairal, Bach, and Ponce [MBP12] considers argmin differentiation for a LASSO problem, derives specific rules for this case, and presents an efficient algorithm based upon our ability to solve the LASSO problem efficiently. Jordan-Squire [Jor15] studies convex optimization over probability measures and implicit differentiation in this context. Bell and Burke [BB08] adapts automatic differentiation to obtain derivatives of implicitly defined functions.

2.4.4 An optimization view of the ReLU, sigmoid, and softmax

In this section we note how the commonly used ReLU, sigmoid, and softmax functions can be interpreted as explicit closed-form solutions to constrained convex optimization (argmin) problems. Bibi, Ghanem, Koltun, and Ranftl [Bib+18] presents another view that interprets other layers as proximal operators and stochastic solvers. We use these as examples to further highlight the power of optimization-based inference, not to provide a new analysis of these layers. The main focus of this thesis is *not* on learning and re-discovering existing activation functions. In this thesis, we rather propose new optimization-based inference layers that do *not* have explicit closed-form solutions like these examples and show that they can still be efficiently turned into differentiable building blocks for end-to-end architectures.

Theorem 2. *The ReLU, defined by $f(x) = \max\{0, x\}$, can be interpreted as projecting a point $x \in \mathbb{R}^n$ onto the non-negative orthant as*

$$f(x) = \underset{y}{\operatorname{argmin}} \;\; \frac{1}{2}||x-y||_2^2 \;\; \text{s.t.} \;\; y \geq 0. \tag{2.2}$$

Proof. The usual solution can be obtained by looking at the KKT conditions of Equation (2.2). Introducing a dual variable $\lambda \geq 0$ for the inequality constraint, the Lagrangian of Equation (2.2) is

$$L(y, \lambda) = \frac{1}{2}||x-y||_2^2 - \lambda^\top y. \tag{2.3}$$

The stationarity condition $\nabla_y L(y^\star, \lambda^\star) = 0$ gives a way of expressing the primal optimal variable $y^\star$ in terms of the dual optimal variable $\lambda^\star$ as $y^\star = x + \lambda^\star$. Complementary slackness $\lambda_i^\star(x_i + \lambda_i^\star) = 0$ shows that $\lambda_i^\star \in \{0, -x_i\}$. Consider two cases:

- **Case 1:** $x_i \geq 0$. Then $\lambda_i^\star$ must be 0 since we require $\lambda^\star \geq 0$. Thus $y_i^\star = x_i + \lambda_i^\star = x_i$.
- **Case 2:** $x_i < 0$. Then $\lambda_i^\star$ must be $-x_i$ since we require $y \geq 0$. Thus $y_i^\star = x_i + \lambda_i^\star = 0$.

Combining these cases gives the usual solution of $y^\star = \max\{0, x\}$. □

Theorem 3. *The sigmoid or logistic function, defined by $f(x) = (1 + e^{-x})^{-1}$, can be interpreted as projecting a point $x \in \mathbb{R}^n$ onto the interior of the unit hypercube as*

$$f(x) = \operatorname*{argmin}_{0<y<1} \; -x^\top y - H_b(y), \tag{2.4}$$

where $H_b(y) = -\left(\sum_i y_i \log y_i + (1 - y_i)\log(1 - y_i)\right)$ is the binary entropy function.

Proof. The usual solution can be obtained by looking at the first-order optimality condition of Equation (2.4). The domain of the binary entropy function H_b restricts us to $0 < y < 1$ without needing to explicitly represent this as a constraint in the optimization problem. Let $g(y; x) = -x^\top y - H_b(y)$ be the objective. The first-order optimality condition $\nabla_y g(y^\star; x) = 0$ gives us $-x_i + \log y_i^\star - \log(1 - y_i^\star) = 0$ and thus $y^\star = (1 + e^{-x})^{-1}$. □

Theorem 4. *The softmax, defined by $f(x)_j = e^{x_j} / \sum_i e^{x_i}$, can be interpreted as projecting a point $x \in \mathbb{R}^n$ onto the interior of the $(n - 1)$-simplex*

$$\Delta_{n-1} = \{p \in \mathbb{R}^n \mid 1^\top p = 1 \;\text{ and }\; p \geq 0\}$$

as

$$f(x) = \operatorname*{argmin}_{0<y<1} \; -x^\top y - H(y) \;\; \text{s.t.} \;\; 1^\top y = 1 \tag{2.5}$$

where $H(y) = -\sum_i y_i \log y_i$ is the entropy function.

Proof. The usual solution can be obtained by looking at the KKT conditions of Equation (2.5). Introducing a scalar-valued dual variable ν for the equality constraint, the Lagrangian is

$$L(y, \nu) = -x^\top y - H(y) + \nu(1^\top y - 1) \tag{2.6}$$

The stationarity condition $\nabla_y L(y^\star, \nu^\star) = 0$ gives a way of expressing the primal optimal variable $y^\star$ in terms of the dual optimal variable $\nu^\star$ as

$$y_j^\star = \exp\{x_j - 1 - \nu^\star\}. \tag{2.7}$$

Putting this back into the equality constraint $1^\top y^\star = 1$ gives us $\sum_i \exp\{x_i - 1 - \nu^\star\} = 1$ and thus $\nu^\star = \log \sum_i \exp\{x_i - 1\}$. Substituting this back into Equation (2.7) gives us the usual definition of $y_j = e^{x_j} / \sum_i e^{x_i}$. □

Corollary 1. *A temperature-scaled softmax scales the entropy term in the objective and the sparsemax [MA16] replaces the objective's entropy penalty with a ridge section.*

2.5 Reinforcement Learning and Control

The fields of reinforcement learning (RL) and optimal control typically involve creating agents that act optimally in an environment. These environments can typically be represented as a Markov decision process (MDP) with a continuous or discrete state space and a continuous or discrete action space. The environment often has some oracle-given reward associated with each state and the goal of RL and control is to find a policy that maximizes the cumulative reward achieved.

Using the notation from [Lev17a], *policy search* methods learn a policy $\pi_\theta(u_t|x_t)$ parameterized by θ that predicts a distribution over next action to take given the current state x_t. The goal of policy search is to find a policy that maximizes the expected return

$$\underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t \gamma^t r(x_t, u_t)\right], \tag{2.8}$$

where $p_\theta(\tau) = p(x_1)\prod \pi_\theta(u_t|x_t)p(x_{t+1}|x_t, u_t)$ is the distribution over trajectories, $\gamma \in (0, 1]$ is a discount factor, $r(x_t, u_t)$ is the state-action reward at time t, and $p(x_{t+1}|x_t, u_t)$ is the state-transition probability. In many scenarios, the reward r is assumed to be a black-box function that derivative information cannot be obtained from. *Model-free* techniques for policy search typically do not attempt to model the state-transition probability while *model-based* and *control* approaches do.

Control approaches typically provide a policy by planning based on known state transitions. For example, in continuous state-action spaces with deterministic state transitions, the finite-horizon model predictive control problem is

$$\underset{x_{1:T}\in\mathcal{X}, u_{1:T}\in\mathcal{U}}{\operatorname{argmin}}\ \sum_{t=1}^{T} C_t(x_t, u_t) \quad \text{subject to} \quad x_{t+1} = f(x_t, u_t), \quad x_1 = x_{\text{init}}, \tag{2.9}$$

where x_{init} is the current system state, the cost C_t is typically hand-engineered and differentiable, and $x_{t+1} = f(x_t, u_t)$ is the deterministic next-state transition, *i.e.* the point-mass given by $p(x_{t+1}|x_t, u_t)$. While this thesis focuses on the continuous and deterministic setting, control approaches can also be applied in discrete and stochastic settings.

Pure model-free techniques for policy search have demonstrated promising results in many domains by learning *reactive polices* which directly map observations to actions [Mni+13; Oh+16; Gu+16b; Lil+15; Sch+15; Sch+16; Gu+16a]. Despite their success, model-free methods have many drawbacks and limitations, including a lack of interpretability, poor generalization, and a high sample complexity. **Model-based methods** are known to be more sample-efficient than their model-free counterparts. These methods generally rely on learning a dynamics model directly from interactions with the real system and then integrate the learned model into the control policy [Sch97; AQN06; DR11; Hee+15; Boe+14]. More recent approaches use a deep network to learn low-dimensional latent state representations and associated dynamics models in this learned representation. They then apply standard trajectory optimization methods on these learned embeddings [LKS15; Wat+15; Lev+16]. However, these methods still require a manually specified and hand-tuned cost function, which can become even more difficult in a latent representation. Moreover, there is no guarantee that the learned dynamics model can accurately capture portions of the state space relevant for the task at hand.

To leverage the benefits of both approaches, there has been significant interest in **combining the model-based and model-free paradigms.** In particular, much attention has been dedicated to utilizing model-based priors to accelerate the model-free learning process. For instance, synthetic training data can be generated by model-based control algorithms to guide the policy search or prime a model-free policy [Sut90; TBS10; LA14; Gu+16b; Ven+16; Lev+16; Che+17; Nag+17; Sun+17]. [Ban+17] learns a controller and then distills it to a neural network policy which is then fine-tuned with model-free policy learning. However, this line of work usually keeps the model separate from the learned policy.

Alternatively, the policy can include an **explicit planning module** which *leverages learned models* of the system or environment, both of which are learned through model-free techniques. For example, the classic Dyna-Q algorithm [Sut90] simultaneously learns a model of the environment and uses it to plan. More recent work has explored incorporating such structure into deep networks and learning the policies in an end-to-end fashion. Tamar, Wu, Thomas, Levine, and Abbeel [Tam+16] uses a recurrent network to predict the value function by approximating the value iteration algorithm with convolutional layers. Karkus, Hsu, and Lee [KHL17] connects a dynamics model to a planning algorithm and formulates the policy as a structured recurrent network. Silver, Hasselt, Hessel, Schaul, Guez, Harley, Dulac-Arnold, Reichert, Rabinowitz, Barreto, et al. [Sil+16] and Oh, Singh, and Lee [OSL17] perform multiple rollouts using an abstract dynamics model to predict the value function. A similar approach is taken by Weber, Racanière, Reichert, Buesing, Guez, Rezende, Badia, Vinyals, Heess, Li, et al. [Web+17] but directly predicts the next action and reward from rollouts of an explicit environment model. Farquhar, Rocktäschel, Igl, and Whiteson [Far+17] extends model-free approaches, such as DQN [Mni+15] and A3C [Mni+16], by planning with a tree-structured neural network to predict the cost-to-go. While these approaches have demonstrated impressive results in discrete state and action spaces, they are not applicable to continuous control problems.

To tackle continuous state and action spaces, Pascanu, Li, Vinyals, Heess, Buesing, Racanière, Reichert, Weber, Wierstra, and Battaglia [Pas+17a] propose a neural architecture which uses an abstract environmental model to plan and is trained directly from an external task loss. Pong, Gu, Dalal, and Levine [Pon+18] learn goal-conditioned value functions and use them to plan single or multiple steps of actions in an MPC fashion. Similarly, Pathak, Mahmoudieh, Luo, Agrawal, Chen, Shentu, Shelhamer, Malik, Efros, and Darrell [Pat+18] train a goal-conditioned policy to perform rollouts in an abstract feature space but ground the policy with a loss term which corresponds to true dynamics data. The aforementioned approaches can be interpreted as a distilled optimal controller which does not separate components for the cost and dynamics. Taking this analogy further, another strategy is to differentiate through an optimal control algorithm itself. Okada, Rigazio, and Aoshima [ORA17] and Pereira, Fan, An, and Theodorou [Per+18] present a way to differentiate through path integral optimal control [Wil+16; WAT17] and learn a planning policy end-to-end. Srinivas, Jabri, Abbeel, Levine, and Finn [Sri+18] shows how to embed differentiable planning (unrolled gradient descent over actions) within a goal-directed policy. In a similar vein, Tamar, Thomas, Zhang, Levine, and Abbeel [Tam+17] differentiates through an iterative LQR (iLQR) solver [LT04; XLH17; TMT14] to learn a cost-shaping term offline. This shaping term enables a shorter horizon controller to approximate the behavior of a solver with a longer horizon to save computation during runtime.

Bibliography

This bibliography contains 143 references.

[Aba+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[AQN06] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. "Using inaccurate models in reinforcement learning". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 1–8.

[Bal+16] Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. "Deepcoder: Learning to write programs". In: *arXiv preprint arXiv:1611.01989* (2016).

[Ban+17] Somil Bansal, Roberto Calandra, Sergey Levine, and Claire Tomlin. "MBMF: Model-Based Priors for Model-Free Reinforcement Learning". In: *arXiv preprint arXiv:1709.03153* (2017).

[Bar18] Shane Barratt. "On the differentiability of the solution to convex optimization problems". In: *arXiv preprint arXiv:1804.05098* (2018).

[Bat+16] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. "Interaction networks for learning about objects, relations and physics". In: *Advances in neural information processing systems*. 2016, pp. 4502–4510.

[Bat+18] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[BB08] Bradley M Bell and James V Burke. "Algorithmic differentiation of implicit functions and optimal values". In: *Advances in Automatic Differentiation*. Springer, 2008, pp. 67–77.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[Bel17] David Belanger. "Deep Energy-Based Models for Structured Prediction". PhD thesis. University of Massachusetts Amherst, 2017.

[Ber+05] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*. Vol. 1. 3. Athena scientific Belmont, MA, 2005.

[Ber99] Dimitri P Bertsekas. *Nonlinear programming.* Athena scientific Belmont, 1999.

[Bib+18] Adel Bibi, Bernard Ghanem, Vladlen Koltun, and René Ranftl. “Deep Layers as Stochastic Solvers”. In: (2018).

[Bis07] Christopher Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn.* Springer, New York, 2007.

[BM16] David Belanger and Andrew McCallum. “Structured prediction energy networks”. In: *Proceedings of the International Conference on Machine Learning.* 2016.

[Boe+14] Joschika Boedecker, Jost Tobias Springenberg, Jan Wulfing, and Martin Riedmiller. “Approximate Real-Time Optimal Control Based on Sparse Gaussian Process Models”. In: *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL).* 2014.

[BP16] Jonathan T Barron and Ben Poole. “The fast bilateral solver”. In: *European Conference on Computer Vision.* Springer. 2016, pp. 617–632.

[Bro+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[BS13] J Frédéric Bonnans and Alexander Shapiro. *Perturbation analysis of optimization problems.* Springer Science & Business Media, 2013.

[BSS13] Philémon Brakel, Dirk Stroobandt, and Benjamin Schrauwen. “Training energy-based models for time-series imputation.” In: *Journal of Machine Learning Research* 14.1 (2013), pp. 2771–2797.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[BYM17] David Belanger, Bishan Yang, and Andrew McCallum. “End-to-End Learning for Structured Prediction Energy Networks”. In: *Proceedings of the International Conference on Machine Learning.* 2017.

[Che+15] Liang-Chieh Chen, Alexander G Schwing, Alan L Yuille, and Raquel Urtasun. “Learning deep structured models”. In: *Proceedings of the International Conference on Machine Learning.* 2015.

[Che+17] Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. “Combining Model-Based and Model-Free Updates for Trajectory-Centric Reinforcement Learning”. In: *arXiv preprint arXiv:1703.03078* (2017).

[Che+18] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. “Neural ordinary differential equations”. In: *Advances in Neural Information Processing Systems.* 2018, pp. 6572–6583.

[Dev+17] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdelrahman Mohamed, and Pushmeet Kohli. “Robustfill: Neural program learning under noisy I/O”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org. 2017, pp. 990–998.

[Dia+17] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. "Unrolled optimization with deep priors". In: *arXiv preprint arXiv:1705.08041* (2017).

[Din77] U Dini. "Analisi infinitesimale, Lezioni dettate nella R". In: *Universitá di Pisa (1877/78)* (1877).

[Dom12] Justin Domke. "Generic Methods for Optimization-Based Modeling." In: *AISTATS*. Vol. 22. 2012, pp. 318–326.

[DR09] Asen L Dontchev and R Tyrrell Rockafellar. "Implicit functions and solution mappings". In: *Springer Monogr. Math.* (2009).

[DR11] Marc Deisenroth and Carl E Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1126–1135.

[Far+17] Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. "TreeQN and ATreeC: Differentiable Tree Planning for Deep Reinforcement Learning". In: *arXiv preprint arXiv:1710.11417* (2017).

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, NY, USA: 2001.

[FI90] Anthony V Fiacco and Yo Ishizuka. "Sensitivity and stability analysis for nonlinear programming". In: *Annals of Operations Research* 27.1 (1990), pp. 215–235.

[FP03] David A Forsyth and Jean Ponce. "A modern approach". In: *Computer vision: a modern approach* (2003), pp. 88–101.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323.

[Gil+17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural message passing for quantum chemistry". In: *arXiv preprint arXiv:1704.01212* (2017).

[Goo+13] Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. "Multi-prediction deep Boltzmann machines". In: *Advances in Neural Information Processing Systems*. 2013, pp. 548–556.

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[Goo+16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[Gou+16] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. “On Differentiating Parameterized Argmin and Argmax Problems with Application to Bi-level Optimization”. In: *arXiv preprint arXiv:1607.05447* (2016).

[Goy+18] Kartik Goyal, Graham Neubig, Chris Dyer, and Taylor Berg-Kirkpatrick. “A continuous relaxation of beam search for end-to-end training of neural sequence models”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[Gra+16] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. “Hybrid computing using a neural network with dynamic external memory”. In: *Nature* 538.7626 (2016), p. 471.

[Gra+18] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. “Ffjord: Free-form continuous dynamics for scalable reversible generative models”. In: *arXiv preprint arXiv:1810.01367* (2018).

[Gu+16a] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. “Q-prop: Sample-efficient policy gradient with an off-policy critic”. In: *arXiv preprint arXiv:1611.02247* (2016).

[Gu+16b] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. “Continuous Deep Q-Learning with Model-based Acceleration”. In: *Proceedings of the International Conference on Machine Learning*. 2016.

[Gul+18] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. “Hyperbolic attention networks”. In: *arXiv preprint arXiv:1805.09786* (2018).

[GW08] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[GWD14] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[Hee+15] Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. “Learning continuous control policies by stochastic value gradients”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 2944–2952.

[Hil+15] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. “The goldilocks principle: Reading children’s books with explicit memory representations”. In: *arXiv preprint arXiv:1511.02301* (2015).

[HSF18] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. “Matrix capsules with EM routing”. In: (2018).

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 1024–1034.

[Ing+18] John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. “Learning Protein Structure with a Differentiable Simulator”. In: (2018).

[Joh+16] Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. “Composing graphical models with neural networks for structured representations and fast inference”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2946–2954.

[Jor15] Christopher Jordan-Squire. “Convex Optimization over Probability Measures”. PhD thesis. 2015.

[JRB18] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. “Differentiable particle filters: End-to-end learning with algorithmic priors”. In: *arXiv preprint arXiv:1805.11122* (2018).

[KC88] Michael Peter Kennedy and Leon O Chua. “Neural networks for nonlinear programming”. In: *IEEE Transactions on Circuits and Systems* 35.5 (1988), pp. 554–562.

[KHL17] Peter Karkus, David Hsu, and Wee Sun Lee. “Qmdp-net: Deep learning for planning under partial observability”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4697–4707.

[KP13] Karl Kunisch and Thomas Pock. “A bilevel optimization approach for parameter learning in variational models”. In: *SIAM Journal on Imaging Sciences* 6.2 (2013), pp. 938–983.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[KW16] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[LA14] Sergey Levine and Pieter Abbeel. “Learning neural network policies with guided policy search under unknown dynamics”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 1071–1079.

[LCB98] Yann LeCun, Corinna Cortes, and Christopher JC Burges. *The MNIST database of handwritten digits*. 1998.

[LeC+06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. “A tutorial on energy-based learning”. In: *Predicting structured data* 1 (2006).

[Lev+16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. “End-to-end training of deep visuomotor policies”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[Lev17a] Sergey Levine. *Introduction to Reinforcement Learning*. Berkeley CS 294-112: Deep Reinforcement Learning. 2017.

[Lev17b] Sergey Levine. *Optimal Control and Planning*. Berkeley CS 294-112: Deep Reinforcement Learning. 2017.

[Li+18] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. “Smoothing the Geometry of Probabilistic Box Embeddings”. In: (2018).

[Li94] Stan Z Li. “Markov random field models in computer vision”. In: *European conference on computer vision*. Springer. 1994, pp. 361–370.

[Lil+15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[Lil+93] Walter E Lillo, Mei Heng Loh, Stefen Hui, and Stanislaw H Zak. “On solving constrained optimization problems with neural networks: A penalty method approach”. In: *IEEE Transactions on neural networks* 4.6 (1993), pp. 931–940.

[LKS15] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. “DeepMPC: Learning Deep Latent Features for Model Predictive Control.” In: *Robotics: Science and Systems*. 2015.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: (2001).

[LT04] Weiwei Li and Emanuel Todorov. “Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems”. In: 2004.

[MA16] Andre Martins and Ramon Astudillo. “From softmax to sparsemax: A sparse model of attention and multi-label classification”. In: *International Conference on Machine Learning*. 2016, pp. 1614–1623.

[MB18] Arthur Mensch and Mathieu Blondel. “Differentiable dynamic programming for structured prediction and attention”. In: *arXiv preprint arXiv:1802.03676* (2018).

[MBP12] Julien Mairal, Francis Bach, and Jean Ponce. “Task-driven dictionary learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2012), pp. 791–804.

[Met+16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. “Unrolled Generative Adversarial Networks”. In: *arXiv preprint arXiv:1611.02163* (2016).

[Mni+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.

[Mni+16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. In: *International Conference on Machine Learning*. 2016, pp. 1928–1937.

[Mon+17] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. “Geometric deep learning on graphs and manifolds using mixture model cnns”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5115–5124.

[Nag+17] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. “Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning”. In: *arXiv preprint arXiv:1708.02596*. 2017.

[NLS15] Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. “Neural programmer: Inducing latent programs with gradient descent”. In: *arXiv preprint arXiv:1511.04834* (2015).

[NW06] Jorge Nocedal and Stephen J Wright. *Sequential quadratic programming*. Springer, 2006.

[Oh+16] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. “Control of Memory, Active Perception, and Action in Minecraft”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (2016).

[ORA17] Masashi Okada, Luca Rigazio, and Takenobu Aoshima. “Path Integral Networks: End-to-End Differentiable Optimal Control”. In: *arXiv preprint arXiv:1706.09597* (2017).

[OSL17] Junhyuk Oh, Satinder Singh, and Honglak Lee. “Value prediction network”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6120–6130.

[Par+16] Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. “Neuro-symbolic program synthesis”. In: *arXiv preprint arXiv:1611.01855* (2016).

[Pas+17a] Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia. “Learning model-based planning from scratch”. In: *arXiv preprint arXiv:1707.06170* (2017).

[Pas+17b] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in PyTorch”. In: *NIPS Autodiff Workshop* (2017).

[Pat+18] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. “Zero-shot visual imitation”. In: *arXiv preprint arXiv:1804.08606* (2018).

[PBX09] Jian Peng, Liefeng Bo, and Jinbo Xu. “Conditional neural fields”. In: *Advances in neural information processing systems*. 2009, pp. 1419–1427.

[Per+18] Marcus Pereira, David D. Fan, Gabriel Nakajima An, and Evangelos Theodorou. “MPC-Inspired Neural Network Policies for Sequential Decision Making”. In: *arXiv preprint arXiv:1802.05803* (2018).

[Pon+18] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. “Temporal Difference Models: Model-Free Deep RL for Model-Based Control”. In: *arXiv preprint arXiv:1802.09081* (2018).

[PS17] Emilio Parisotto and Ruslan Salakhutdinov. “Neural map: Structured memory for deep reinforcement learning”. In: *arXiv preprint arXiv:1702.08360* (2017).

[RD15] Scott Reed and Nando De Freitas. “Neural programmer-interpreters”. In: *arXiv preprint arXiv:1511.06279* (2015).

[San+17] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. “A simple neural network module for relational reasoning". In: *arXiv preprint arXiv:1706.01427* (2017).

[SB+98] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction.* MIT press, 1998.

[SB11] Shankar Sastry and Marc Bodson. *Adaptive control: stability, convergence and robustness.* Courier Corporation, 2011.

[Sch+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15).* 2015, pp. 1889–1897.

[Sch+16] John Schulman, Philpp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *International Conference on Learning Representations* (2016).

[Sch15] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[Sch97] Jeff G Schneider. "Exploiting model uncertainty estimates for safe dynamic control learning". In: *Advances in neural information processing systems.* 1997, pp. 1047–1053.

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules". In: *Advances in neural information processing systems.* 2017, pp. 3856–3866.

[She+18] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. "Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks". In: *arXiv preprint arXiv:1810.09536* (2018).

[Sil+16] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. "The predictron: End-to-end learning and planning". In: *arXiv preprint arXiv:1612.08810* (2016).

[SM+12] Charles Sutton, Andrew McCallum, et al. "An introduction to conditional random fields". In: *Foundations and Trends® in Machine Learning* 4.4 (2012), pp. 267–373.

[SNW12] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning.* Mit Press, 2012.

[SR14] Uwe Schmidt and Stefan Roth. "Shrinkage fields for effective image restoration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2014, pp. 2774–2781.

[SRE11] Veselin Stoyanov, Alexander Ropson, and Jason Eisner. "Empirical Risk Minimization of Graphical Model Parameters Given Approximate Inference, Decoding, and Model Structure." In: *AISTATS.* 2011, pp. 725–733.

[Sri+18] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. "Universal Planning Networks". In: *arXiv preprint arXiv:1804.00645* (2018).

[Sun+17] Liting Sun, Cheng Peng, Wei Zhan, and Masayoshi Tomizuka. "A Fast Integrated Planning and Control Framework for Autonomous Driving via Imitation Learning". In: *arXiv preprint arXiv:1707.02515*. 2017.

[Sut90] Richard S Sutton. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming". In: *Proceedings of the seventh international conference on machine learning*. 1990, pp. 216–224.

[SWF+15] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. "End-to-end memory networks". In: *Advances in neural information processing systems*. 2015, pp. 2440–2448.

[Sze10] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[Tam+16] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. "Value iteration networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2154–2162.

[Tam+17] Aviv Tamar, Garrett Thomas, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. "Learning from the hindsight plan—Episodic MPC improvement". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 336–343.

[Tap+07] Marshall F Tappen, Ce Liu, Edward H Adelson, and William T Freeman. "Learning gaussian conditional random fields for low-level vision". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[Tas+05] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. "Learning structured prediction models: A large margin approach". In: *Proceedings of the 22nd International Conference on Machine Learning*. ACM. 2005, pp. 896–903.

[TBS10] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. "A generalized path integral control approach to reinforcement learning". In: *Journal of Machine Learning Research* 11.Nov (2010), pp. 3137–3181.

[TGK04] Ben Taskar, Carlos Guestrin, and Daphne Koller. "Max-margin Markov networks". In: *Advances in neural information processing systems*. 2004, pp. 25–32.

[TMT14] Yuval Tassa, Nicolas Mansard, and Emo Todorov. "Control-limited differential dynamic programming". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1168–1175.

[TT18] Chengzhou Tang and Ping Tan. "Ba-net: Dense bundle adjustment network". In: *arXiv preprint arXiv:1806.04807* (2018).

[UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9446–9454.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[Ven+16] Arun Venkatraman, Roberto Capobianco, Lerrel Pinto, Martial Hebert, Daniele Nardi, and J Andrew Bagnell. "Improved learning of dynamics models for control". In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 703–713.

[Wan+18] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. "Non-local neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7794–7803.

[Was13] Larry Wasserman. *All of statistics: a concise course in statistical inference.* Springer Science & Business Media, 2013.

[Wat+15] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in neural information processing systems*. 2015, pp. 2746–2754.

[WAT17] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. "Model predictive path integral control: From theory to parallel computation". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 344–357.

[Web+17] Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. "Imagination-Augmented Agents for Deep Reinforcement Learning". In: *arXiv preprint arXiv:1707.06203* (2017).

[WFU16] Shenlong Wang, Sanja Fidler, and Raquel Urtasun. "Proximal deep structured models". In: *Advances in Neural Information Processing Systems*. 2016, pp. 865–873.

[Wil+16] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. "Aggressive driving with model predictive path integral control". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1433–1440.

[Wri97] Stephen J Wright. *Primal-dual interior-point methods*. Siam, 1997.

[XC18] Zhang Xinyi and Lihui Chen. "Capsule Graph Neural Network". In: (2018).

[XLH17] Zhaoming Xie, C. Karen Liu, and Kris Hauser. "Differential Dynamic Programming with Nonlinear Constraints". In: *International Conference on Robotics and Automation (ICRA)*. 2017.

[XMS16] Caiming Xiong, Stephen Merity, and Richard Socher. "Dynamic memory networks for visual and textual question answering". In: *International conference on machine learning*. 2016, pp. 2397–2406.

[Xu+18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[Zah+17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. "Deep sets". In: *Advances in neural information processing systems*. 2017, pp. 3391–3401.

[Zhe+15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. “Conditional random fields as recurrent neural networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1529–1537.