Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time

Ferran Alet, Kenji Kawaguchi, Maria Bauza,
Nurullah Giray Kuru, Tomás Lozano-Pérez, Leslie Pack Kaelbling
MIT - CSAIL
`{alet,kawaguch,bauza,ngkuru,tlp,lpk}@mit.edu`

Abstract

From CNNs to attention mechanisms, encoding inductive biases into neural networks has been a fruitful source of improvement in machine learning. Auxiliary losses are a general way of encoding biases in order to help networks learn better representations by adding extra terms to the loss function. However, since they are minimized on the training data, they suffer from the same generalization gap as regular task losses. Moreover, by changing the loss function, the network is optimizing a different objective than the one we care about. In this work we solve both problems: first, we take inspiration from *transductive learning* and note that, after receiving an input but before making a prediction, we can fine-tune our models on any unsupervised objective. We call this process tailoring, because we customize the model to each input. Second, we formulate a nested optimization (similar to those in meta-learning) and train our models to perform well on the task loss after adapting to the tailoring loss. The advantages of tailoring and meta-tailoring are discussed theoretically and demonstrated empirically on several diverse examples: encoding inductive conservation laws from physics, increasing robustness to adversarial examples, meta-tailoring with contrastive losses to improve theoretical generalization guarantees, and increasing performance in model-based RL.

1 Introduction

The key to successful generalization in machine learning is the encoding of useful inductive biases. A variety of mechanisms, from parameter tying to data augmentation, have proven useful but there is no systematic strategy for designing and implementing these biases.

Auxiliary losses are a paradigm for encoding a wide variety of biases, constraints and objectives, helping networks learn better representations and generalize more broadly. They add an extra term to the task loss and minimize it over the training data or, in semi-supervised learning, on an extra set of unlabeled data. However, they have two major difficulties:

1. Auxiliary losses are only minimized at training time, but not for the query points. This causes a generalization gap between training and testing, in addition to that of the task loss.
2. By minimizing the sum of the task loss plus the auxiliary loss, we are optimizing a different objective than the one we care about (only the task loss).

In this work we propose a solution to each problem:

1. We use ideas from *transductive learning* to minimize the auxiliary loss at the query by running an optimization at prediction time, eliminating the generalization gap for the auxiliary loss. We call this process *tailoring*, because we customize the model to each query.

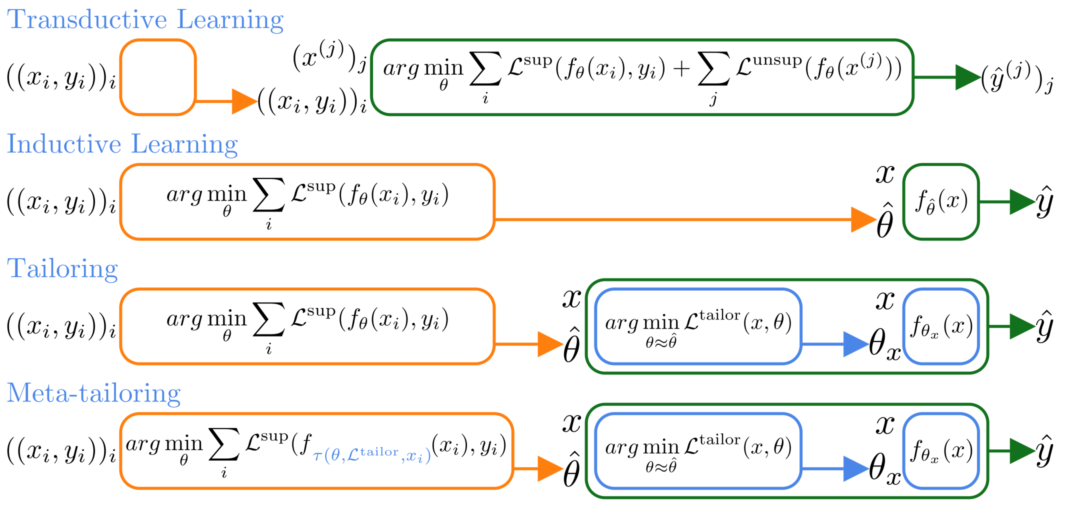
Transductive Learning
$((x_i, y_i))_i$
$(x^{(j)})_j$
$((x_i, y_i))_i$
$arg\min_\theta \sum_i \mathcal{L}^{\text{sup}}(f_\theta(x_i), y_i) + \sum_j \mathcal{L}^{\text{unsup}}(f_\theta(x^{(j)}))$
$(\hat{y}^{(j)})_j$
Inductive Learning
$((x_i, y_i))_i$
$arg\min_\theta \sum_i \mathcal{L}^{\text{sup}}(f_\theta(x_i), y_i)$
x
$\hat{\theta}$
$f_{\hat{\theta}}(x)$
$\hat{y}$
Tailoring
$((x_i, y_i))_i$
$arg\min_\theta \sum_i \mathcal{L}^{\text{sup}}(f_\theta(x_i), y_i)$
x
$\hat{\theta}$
$arg\min_{\theta\approx\hat{\theta}} \mathcal{L}^{\text{tailor}}(x, \theta)$
x
θ_x
$f_{\theta_x}(x)$
$\hat{y}$
Meta-tailoring
$((x_i, y_i))_i$
$arg\min_\theta \sum_i \mathcal{L}^{\text{sup}}(f_{\tau(\theta,\mathcal{L}^{\text{tailor}},x_i)}(x_i), y_i)$
x
$\hat{\theta}$
$arg\min_{\theta\approx\hat{\theta}} \mathcal{L}^{\text{tailor}}(x, \theta)$
x
θ_x
$f_{\theta_x}(x)$
$\hat{y}$

Figure 1: Comparison of several learning settings with *offline* computation in the orange boxes and *online* computation in the green boxes, with tailoring in blue. For meta-tailoring training, $\tau(\theta, \mathcal{L}^{\text{tailor}}, x) = \arg\min_{\theta' \approx \theta} \mathcal{L}^{\text{tailor}}(x, \theta')$ represents the tailoring process resulting in θ_x. Although tailoring and meta-tailoring are best understood in supervised learning, they can also be applied in reinforcement learning, as shown in section 4.3.

2. We use ideas from *meta-learning* to learn a model that performs well on the task loss assuming that we will be optimizing the auxiliary loss. This *meta-tailoring* effectively trains the model to leverage the unsupervised tailoring loss to minimize the task loss.

Tailoring a predictor In classical inductive supervised learning, an algorithm consumes a training dataset of input-output pairs, $((x_i, y_i))_{i=1}^n$, and produces a set of parameters $\hat{\theta}$ by minimizing a supervised loss $\sum_{i=1}^n \mathcal{L}^{\text{sup}}(f_\theta(x_i), y_i)$ and, optionally, an unsupervised auxiliary loss $\sum_{i=1}^n \mathcal{L}^{\text{unsup}}(\theta, x_i)$. These parameters specify a hypothesis $f_{\hat{\theta}}(\cdot)$ that, given a new input x, generates an output $\hat{y} = f_{\hat{\theta}}(x)$. This problem setting misses a substantial opportunity: before the learning algorithm sees the query point x, it has distilled the data down to a set of parameters, which are frozen during inference, and so it cannot use new information about the *particular* x that it will be asked to make a prediction for.

Vapnik recognized an opportunity to make more accurate predictions when the query point is known, in a framework that is now known as *transductive learning* [Vapnik, 1995, Chapelle et al., 2000]. In transductive learning, a single algorithm consumes both labeled data, $((x_i, y_i))_{i=1}^n$, and a set of input points for which predictions are desired, $(x^{(j)})_j$, and produces predicted outputs $(\hat{y}^{(j)})_j$ for each of the queries, as illustrated in the top row of figure 1. In general, however, we do not know queries *a priori*, and instead we want an inductive rule that makes predictions on-line, as queries arrive. To obtain a prediction function from a transductive system, we would need to encapsulate the entire learning procedure inside the prediction function. This strategy would achieve our objective of taking x into account at prediction time, but would be computationally much too slow.

We observe that this strategy for combining induction and transduction would perform very similar computations for each prediction, sharing the same training data and objective. We can use ideas from meta-learning to find a shared "meta-hypothesis" that can then be efficiently adapted to each query (treating each prediction as a task). As shown in the third row of figure 1, we first run regular supervised learning to obtain parameters $\hat{\theta}$; then, given a query input x, we fine-tune $\hat{\theta}$ on an unsupervised loss ($\mathcal{L}^{\text{tailor}}$) to obtain customized parameters θ_x and use them to make the final prediction: $f_{\theta_x}(x)$. We call this process *tailoring*, because we adapt the model to each particular input for a customized fit. Notice that tailoring optimizes the loss at the query point, eliminating the generalization gap on the auxiliary loss.

Meta-tailoring Since we will be applying tailoring at prediction time, it is natural to anticipate this adaptation during training, resulting in a two-layer optimization similar to those used for meta-learning. Because of this similarity, we call this process, illustrated in the bottom row of figure 1, *meta-tailoring*. Now, rather than letting $\hat{\theta}$ be the direct minimizer of the supervised loss, we set it to

$$\hat{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} \mathcal{L}^{\text{sup}}(f_{\tau(\theta, \mathcal{L}^{\text{tailor}}, x_i)}(x_i), y_i).$$

Notice that by optimizing this nested objective, the outer process is now optimizing the only objective we care about, $\mathcal{L}^{\text{sup}}$, instead of a proxy combination of $\mathcal{L}^{\text{sup}}$ and $\mathcal{L}^{\text{unsup}}$. At the same time, we are learning to leverage the unsupervised tailoring losses in the inner optimization to affect the model before making the final prediction, both during training and at prediction time.

In many settings, we wish to make predictions for a large number of query points in a (mini-)batch, but because tailoring adapts to every *point*, we must take extra care to be sure it can run efficiently in parallel. Inspired by conditional normalization [Dumoulin et al., 2016] we propose adding element-wise affine transformations and only adapting these parameters in the inner optimization. This allows us to*tailor* outputs for multiple inputs in parallel, without inputs affecting each other's computations. We prove theoretically, in section 3, and provide experimental evidence, in section 4.1, that optimizing these parameters alone has enough capacity to minimize a large class of tailoring losses.

Losses for tailoring Tailoring can be used any time we are making a prediction and have an unsupervised loss that can be minimized on the current input; although we mostly focus on supervised learning, tailoring can also be applied to reinforcement learning, as shown in section 4.3. There are many types of *tailoring* losses that may be useful; here we enumerate three broad classes.

Losses imposing priors and constraints satisfied by the correct predictions, such as conservation of momentum and energy when learning a physics model, symmetry under specific transformations, or cycle-consistency when learning to translate between two languages. We show that tailoring a loss of this type improves predictive losses in section 4.1, where we model a planetary system. We can also leverage soft priors; for instance, in section 4.3 we show how encouraging action-conditional predictions to be likely under an action-unconditional model can improve model-based RL.

Losses that help learn better representations, such as the contrastive losses [Hadsell et al., 2006] explored in semi-supervised learning, or learning to predict one part of the input from another part,such as depth from RGB [Mirowski et al., 2016]. In section A.1 we show how tailoring with a contrastive loss improves supervised prediction performance.

Losses informed by theoretical guarantees. The guarantees of many theorems depend on unsupervised quantities, such as smoothness or distance to the prediction boundary. By optimizing such quantities on the query, or on the surrounding area, we can get better guaranteed performance. In section 4.2, we show this on adversarial examples, where smooth predictions around the test point are critical.

Contributions In summary, our contributions are the following:

1. Introducing *tailoring*, a new framework for encoding inductive biases by minimizing unsupervised losses at prediction time, with theoretical guarantees and broad potential applications.
2. Formulating *meta-tailoring*, which adjusts the outer objective to optimize only the task loss, and developing a new algorithm, CNGRAD, for efficient meta-tailoring.
3. Demonstrating *tailoring* in four domains: encoding conservation laws in a physics prediction problem, increasing resistance to adversarial examples by increasing local smoothness at prediction time, making model-based reinforcement learning more data-efficient, and improving theoretical guarantees of prediction quality by tailoring with a contrastive loss.

For conciseness, the theoretical guarantees behind meta-tailoring with contrastive losses and with general losses have been moved to appendix A.

2 Related work

There are other learning frameworks that perform optimization at prediction time, such as energy-based models [Ackley et al., 1985, Hinton, 2002, LeCun et al., 2006] or models that embed optimization layers in neural networks, whose outputs are the solution of an optimization problem defined by the previous layer [Amos and Kolter, 2017, Tschiatschek et al., 2018]. In contrast to these lines of work, we optimize the parameters of the model, not the hidden activations or the output.

Meta-learning [Schmidhuber, 1987, Bengio et al., 1995, Thrun and Pratt, 1998] has the same two-level optimization structure as our work, but focuses on multiple prediction tasks, each of which has its own separate training data. Most optimization-based meta-learning algorithms can be converted to the meta-tailoring setting. There is a particularly clear connection to MAML [Finn et al., 2017], when we let the tailoring method be a step of gradient descent: $\tau(\hat{\theta}, \mathcal{L}^{\text{tailor}}, x) = \hat{\theta} - \lambda\nabla_{\hat{\theta}}\mathcal{L}^{\text{tailor}}(x, \hat{\theta})$. There are other optimization-based approaches to meta-learning whose adaptations can be batched [Zintgraf et al., 2018, Rakelly et al., 2019, Alet et al., 2019]. In particular, FiLM networks [Perez et al., 2018], which predict customized conditional normalization layers, have been used in meta-learning [Zintgraf et al., 2018, Requeima et al., 2019]. By optimizing the conditional normalization layers themselves, our method CNGrad is simpler, while remaining provably sufficiently expressive. More importantly, we can add to a trained model CNGrad layers with weights initialized to the identity and adapt them to perform tailoring or fine-tune the model with a meta-tailoring objective.

Tailoring is inspired by transductive learning. However, transductive methods, because they operate on a batch of unlabeled points, are able to make use of the underlying distributional properties of those points. On the other hand, tailoring does not need to receive the queries before doing the bulk of the computation.Within transductive learning, local learning [Bottou and Vapnik, 1992] has input-dependent parameters, but it uses similarity in raw input space to select a few data-points instead of reusing the prior learned across the whole data. Some methods [Garcia and Bruna, 2017, Liu et al., 2018] in meta-learning propagate predictions along the test samples in a classic transductive fashion.

Optimization processes similar to tailoring and meta-tailoring have been proposed before, to adapt to different types of variations between training and testing. [Sun et al., 2019] propose to adapt to a change of distribution with few samples by unsupervised fine-tuning at test-time, applying it with a loss of predicting whether the input has been rotated. Other methods in the meta-learning setting exploit test samples of a new task by minimizing either entropy [Dhillon et al., 2020] or a learned loss [Antoniou and Storkey, 2019] in the inner optimization. Finally, [Wang et al., 2019] uses entropy in the inner optimization to adapt to large scale variations in image segmentation. In contrast, we propose (meta-)tailoring as a general effective way to impose inductive biases in the classic machine learning setting. We also unify tailoring and meta-tailoring with arbitrary losses under one paradigm, theoretically and empirically showing the advantage of each modification to classic inductive learning. Appendix G shows experimental results analyzing why using prior work on adapting to the test distribution performs worse than tailoring (which, in turn, performs worse than meta-tailoring) in the classic ML setting where test and training come from the same distribution.

3 CNGrad: a simple algorithm for expressive, efficient tailoring

In the previous sections, we have discussed the principal motivation and theoretical advantages of (meta-)tailoring. However, there is a remaining issue for efficient GPU computations: whereas one can efficiently parallelize the evaluation of a single model over a batch across inputs, it is challenging to efficiently parallelize the evaluation of multiple *tailored* models over a batch. To overcome this issue, by building on CAVIA [Zintgraf et al., 2018] and WarpGrad [Flennerhag et al., 2019], we propose CNGrad which adapts only *conditional normalization* parameters and enables efficient GPU computations for (meta-)tailoring. CNGrad can also be used in regular meta-learning; details and pseudo-codes of both versions can be found in appendix D.

As is done in batch-norm [Ioffe and Szegedy, 2015] after element-wise normalization, we can implement an element-wise affine transformation with parameters (γ, β), scaling and shifting the output $h_k^{(l)}(x)$ of each k-th node at l-th hidden layer independently: $\gamma_k^{(l)} h_k^{(l)}(x) + \beta_k^{(l)}$. In conditional normalization, [Dumoulin et al., 2016] train a collection of (γ, β) in a multi-task fashion to learn different tasks with a single network. We propose to bring this concept to the meta-learning and (meta-)tailoring settings and adapt the affine parameters (γ, β) to each query . For meta-tailoring, the inner optimization minimizes the tailoring loss at an input x by adjusting the affine parameters and the outer optimization adapts the rest of the network. Similar to MAML [Finn et al., 2017], we implement a first-order version, which does not backpropagate through the optimization, and a second-order version, which does. We can efficiently parallelize computations of multiple tailored models over a (mini-)batch in a GPU in the same way as that in a classic induction model, because the adapted parameters only require element-wise multiplications and additions.

CNGRAD is widely applicable, since we can add these adaptable affine parameters to any hidden layer.While we are only changing a tiny portion of the network, we prove below that, under realistic assumptions, we can minimize the inner tailoring loss using only the affine parameters. However, it is worth noting that we still need the entire network to minimize the outer meta-objective.

To analyze properties of the adaptable affine parameters, let us decompose θ into $\theta = (w, \gamma, \beta)$, where the w contains all the weight parameters (including bias terms), and the (γ, β) contains all the affine parameters. Given arbitrary coefficients $\eta_1, \ldots, \eta_{n_g} \in \mathbb{R}$ and an arbitrary function $(f_\theta(x), x) \mapsto \ell_{\text{tailor}}(f_\theta(x), x)$, let $\mathcal{L}^{\text{tailor}}(x, \theta) = \sum_{i=1}^{n_g} \eta_i \ell_{\text{tailor}}(f_\theta(g^{(i)}(x)), x)$, where $g^{(1)}, \ldots, g^{(n_g)}$ are arbitrary input augmentation functions at prediction time. Note that n_g is typically small ($n_g \ll n$) in meta-tailoring; e.g., $n_g = 1$ in the method in Section 4.1 regardless of the size of the dataset n.

Corollary 1 states that for any given $\hat{w}$, if we add any non-degenerate Gaussian noise δ as $\hat{w} + \delta$ with zero mean and any variance on δ, the global minimum value with respect to all parameters (w, γ, β) can be achieved by optimizing only the affine parameters (γ, β), with probability one.

Assumption 1. *(Common activation)* The activation function $\sigma(x)$ is real analytic, monotonically increasing, and the limits exist as: $\lim_{x \to -\infty} \sigma(x) = \sigma_- > -\infty$ and $\lim_{x \to +\infty} \sigma(x) = \sigma_+ \leq +\infty$.

Theorem 1. *For any $x \in \mathcal{X}$ that satisfies $\|g^{(i)}(x)\|_2^2 - g^{(i)}(x)^\top g^{(j)}(x) > 0$ (for all $i \neq j$), and for any fully-connected neural network with a single output unit, at least n_g neurons per hidden layer, and activation functions that satisfy Assumption 1, the following holds: $\inf_{w,\gamma,\beta} \mathcal{L}^{\text{tailor}}(x, w, \gamma, \beta) = \inf_{\gamma,\beta} \mathcal{L}^{\text{tailor}}(x, \bar{w}, \gamma, \beta)$ for any $\bar{w} \notin \mathcal{W}$ where Lebesgue measure of $\mathcal{W} \subset \mathbb{R}^d$ is zero.*

Corollary 1. *Under the assumptions of Theorem 1, for any $\hat{w} \in \mathbb{R}^d$, with probability one over randomly sampled $\delta \in \mathbb{R}^d$ accordingly to any non-degenerate Gaussian distribution, the following holds: $\inf_{w,\gamma,\beta} \mathcal{L}^{\text{tailor}}(x, w, \gamma, \beta) = \inf_{\gamma,\beta} \mathcal{L}^{\text{tailor}}(x, \hat{w} + \delta, \gamma, \beta)$ for any $x \in \mathcal{X}$.*

The assumption and condition in theorem 1 are satisfied in practice(see Appendix B for more details). Therefore, CNGRAD is a practical and computationally efficient method to implement (meta-)tailoring.

4 Experiments

4.1 Tailoring to impose symmetries and constraints at prediction time

Constructing inductive biases that exploit invariances and symmetries is an established strategy for boosting performance in machine learning. During training, we often regularize our networks to satisfy certain criteria; however, this does not guarantee that these criteria will be satisfied outside the training dataset [Suh and Tedrake, 2020]. Another option is to construct *ad hoc* neural network architectures to exploit constraints for important problems, such as using Hamiltonian neural networks [Greydanus et al., 2019] to impose energy (but not momentum) conservation. Tailoring provides a general solution to this problem by adapting the model at prediction time to satisfy the criteria. In meta-tailoring, we also train the system to make good predictions after satisfying the constraint.

We demonstrate this use of tailoring by enforcing physical conservation laws to more accurately predict the evolution of a 5-body planetary system governed by gravitational forces. This prediction problem is challenging, as m-body systems become chaotic for $m > 2$. We generate a supervised-learning dataset with positions, velocities and masses of all 5 bodies as inputs and the changes in position and velocity at the next time-step as targets. Appendix F describes the dataset in greater detail.

To predict the data, we use a 3-layer feed-forward network and apply a tailoring loss based on the laws of conservation of energy and momentum. More concretely, we take the original predictions and adapt our model using the L_1 loss between the initial and final energy and momentum of the whole system. Ensuring this conservation can improve predictions, but notice that minimizing the tailoring loss alone does not guarantee good predictions: predicting that the output equals the input conserves energy and momentum perfectly, but is not correct.

Tailoring adapts some parameters in the network in order to improve the tailoring loss. An alternative for enforcing conservation would be to adapt the output y value directly. Table 1 compares the

Method	loss	relative
Inductive learning	.041	-
Opt. output(50 st.)	.041	$(0.7 \pm 0.1)\%$
6400-spl. TTT(50st.)	.040	$(3.6 \pm 0.2)\%$
Tailoring(1 step)	.040	$(1.9 \pm 0.2)\%$
Tailoring(5 st.)	.039	$(6.3 \pm 0.3)\%$
Tailoring(10 st.)	.038	$(7.5 \pm 0.1)\%$
Meta-tailoring(0 st.)	.030	$(26.3 \pm 3.3)\%$
Meta-tailoring(1 st.)	.029	$(29.9 \pm 3.0)\%$
Meta-tailoring(5 st.)	.027	$(35.3 \pm 2.6)\%$
Meta-tailoring(10 st.)	.026	$(36.0 \pm 2.6)\%$

Table 1: Test MSE loss for different methods; the second column shows the relative improvement over basic inductive supervised learning. The test-time training (TTT) baseline uses a full batch of 6400 test samples to adapt, not allowed in regular SL. With a few gradient steps, tailoring significantly over-performs all baselines. Meta-tailoring improves even further, with 35% improvement.

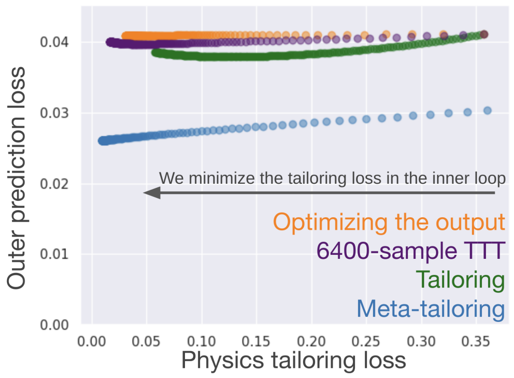

Figure 2: Optimization at prediction time on the planet data; each path going from right to left as we minimize the physics tailoring loss. We use a small step size to illustrate the path. Tailoring and the two baselines share only differ in their test-time computations, thus sharing their starts. Meta-tailoring has a lower starting loss, faster optimization, and no overfitting to the tailoring loss.

predictive accuracy of inductive learning to direct output optimization as well as tailoring and meta-tailoring, using varying numbers of gradient steps. We observe that tailoring is more effective than adapting the output, as the parameters provide a prior on what changes are more natural. For meta-tailoring, we try both first-order and second-order versions of CNGRAD: the first-order version gave slightly better results, possibly because it was trained with a higher tailor learning rate (10^{-3}) with which the second-order version meta-training was unstable (we thus used 10^{-4}). More details can be found in Appendix F. Interestingly, meta-tailoring without any query-time tailoring steps already performs much better than the original model, even though both models have almost the same number of parameters and can overfit the dataset.We conjecture meta-tailoring training is adding an inductive bias that guides optimization towards learning a more generalizable model, even without tailoring at test time. Finally, plot 2 shows the prediction-time optimization paths for different methods.

4.2 Tailoring for robustness against adversarial examples

Despite their successes, neural networks remain susceptible to the problem of adversarial examples [Biggio et al., 2013, Szegedy et al., 2013]: targeted small perturbations of an input can cause the network to mis-classify it. One approach is to make the prediction function smooth via adversarial training [Madry et al., 2017]; however, this only ensures smoothness in the training points and constraining our model to be smooth everywhere makes it lose capacity. This is a perfect opportunity for applying (meta-)tailoring, since we can ask for smoothness *a posteriori*, only on the specific query.

We apply meta-tailoring to robustly classifying CIFAR-10 [Krizhevsky et al., 2009] and ImageNet [Deng et al., 2009] images, tailoring predictions so that they are locally smooth. We meta-tailor our classifier using (the first-order version of) CNGRAD and a tailoring loss that enforces smoothness on the entire vector of features of the penultimate layer in the neural network (denoted $g_\theta(x)$):

$$\mathcal{L}^{\text{tailor}}(x, \theta) = \mathbb{E}[\text{cos\_dist}(g_\theta(x), g_\theta(x+\delta))], \quad \delta \sim N(0, \nu^2), \tag{1}$$

where $\text{cos\_dist}(v_1, v_2)$ is the cosine distance between vectors v_1 and v_2. This loss is inspired by [Ilyas et al., 2019], who argue that adversarial examples are caused by features which are predictive, but non-robust to perturbations. In that way, our loss adjusts the model to ensure features are locally robust at the input query before making a prediction. To keep inference fast, we approximate this loss with a single sample, but it could be improved with more samples or more sophisticated methods.

We build on the work of [Cohen et al., 2019], who developed a method for certifying the robustness of a model via randomized smoothing. It samples several points from a Gaussian $N(x, \sigma^2)$ around

σ	Method	0.0	0.5	1.0	1.5	2.0	2.5	3.0	ACR
0.25	(Inductive) RandSmooth	0.67	0.49	0.00	0.00	0.00	0.00	0.00	0.470
	Meta-tailored	**0.72**	**0.55**	0.00	0.00	0.00	0.00	0.00	**0.494**
0.50	(Inductive) RandSmooth	0.57	0.46	0.37	0.29	0.00	0.00	0.00	0.720
	Meta-tailored	0.66	0.54	**0.42**	**0.31**	0.00	0.00	0.00	**0.819**
1.00	(Inductive) RandSmooth	0.44	0.38	0.33	0.26	0.19	0.15	0.12	0.863
	Meta-tailored	0.52	0.45	0.36	**0.31**	**0.24**	**0.20**	**0.15**	**1.032**

Figure 3: Percentage of points with certificate above different radii, and average certified radius (ACR) for on the ImageNet dataset. Meta-tailoring improves the Average Certification Radius by $5.1\%, 13.8\%, 19.6\%$ respectively. Results for [Cohen et al., 2019] are taken from [Zhai et al., 2020].

the query and, if there is enough agreement in classification, it provides a certificate that the example cannot be adversarially modified by a small perturbation to have a different class. We show that meta-tailoring improves the original randomized smoothing method, testing for $\sigma = 0.25, 0.5, 1.0$. For simplicity, we use $\nu = 0.1$ for all experiments. We initialized with the weights of [Cohen et al., 2019] to speed up training in all ImageNet experiments and to avoid training divergence for CIFAR-10, $\sigma = 1$ (this divergence had already been noted by [Zhai et al., 2020]). Note that we could leverage these pre-trained weights because of CNGRADcan start from a pre-trained model by initializing the extra affine layers to the identity. Finally, we use $\sigma' = \sqrt{\sigma^2 - \nu^2} \approx 0.23, 0.49, 0.995$ so that the points used in our tailoring loss come from $N(x, \sigma^2)$.

In table 3, we show results on Imagenet where we improve the average certification radius by $5.1\%, 13.8\%, 19.6\%$ respectively. In table 6, in the appendix, we show results on CIFAR-10 where we improve the average certification radius by $8.6\%, 10.4\%, 19.2\%$. We chose to meta-tailor this randomized smoothing method because it represents a strong standard in certified adversarial defenses, but it is important to note that there have been advances on this method that sometimes achieve better results than those we present here [Zhai et al., 2020, Salman et al., 2019], see appendix I. However, it is likely that these methods could also be improved through meta-tailoring.

These experiments only scratch the surface of what tailoring allows for adversarial defenses: usually, the adversary looks at the model and gets to pick a particularly bad perturbation $x + \delta$. With tailoring, the model responds, by changing to weights $\theta_{x+\delta}$. This leads to a game, where both weights and inputs are perturbed, similar to: $\max_{|\delta|<\epsilon_x} \min_{|\Delta|<\epsilon_\theta} \mathcal{L}^{\text{sup}}\left(f_{\theta+\Delta}(x+\delta), y\right)$. However, since we don't get to observe y; we optimize the weight perturbation by minimizing $\mathcal{L}^{\text{tailor}}$ instead.

4.3 Tailoring to improve the performance of model-based RL

In model-based reinforcement learning (MBRL) we consider a Markov decision process (MDP) with state-space $\mathcal{S}$, action-space $\mathcal{A}$ and *unknown* transition distribution $p(s'|s, a)$. Here, we assume access to a *known* reward function $r(s, a)$, although this assumption could be relaxed. In MBRL we learn a deterministic transition model $T_\theta(s, a) \to s'$ from past experience and use it to either learn a policy or (as done here) to plan good actions by maximizing the reward over some planning horizon H:

$$a^*_{t\ldots t+H} = \underset{a_{t\ldots t+H-1}}{\arg\max} \sum_{h=1}^{H} r\left(\widehat{s_{t+h}}\right) = \underset{a_{t\ldots t+H-1}}{\arg\max} \sum_{h=1}^{H} r\left(T_\theta\left(s_t, a_{t\ldots t+h-1}\right)\right)$$

where $T_\theta\left(s_t, a_{t\ldots t+h-1}\right) = T_\theta\left(\ldots T_\theta\left(T_\theta\left(s_t, a_t\right), a_{t+1}\right), \ldots, a_{t+h-1}\right)$, i.e. applying T_θ h times. We also often apply receding-horizon control, where we plan with horizon H, but then only apply the first action a_t, observe the next state a_{t+1} and re-plan again. However, in this setting we are optimizing the output of the transition model with respect to (part of) the input: the actions. Similar to adversarial examples in the previous section, this often results in the action optimisation finding regions of the input-space where the model makes overly-optimistic predictions. In order to improve performance it has been previously proposed to regularize the confidence of the models [Ha and Schmidhuber, 2018], to make them uncertainty-aware by learning an ensemble of models [Nagabandi et al., 2020], or to make pessimistic predictions [Kidambi et al., 2020], among many other approaches. Similar to the adversarial examples experiments, meta-tailoring opens new simple ways of making transition models more robust.

In this section, we explain a simple way of using meta-tailoring to improve transition models in MBRL. In particular, we can learn an action-independent model $M(s_t) \to p\,(s_{t+1})$ that outputs a probability distribution for the next state given the current state, independently of the action a_t. This model presents a trade-off with respect to the original model $T(s, a)$: on the one hand, it is intrinsically uncertain since it lacks part of the information (the actions); we thus output a probability distribution in the form of a mixture of Gaussians to model this uncertainty. On the other hand, it has the advantage of not being "hackable" by the action-selection process, making the planner unable to find overly confident regions.

More concretely, we first train an action-independent prediction model to maximize the log-likelihood $M(s_t)\,(s_{t+1})$. Then, when making predictions with $T_\theta(s_t, a_t)$ we use the action-independent log-likelihood of the action-dependent predictions as the tailoring loss: $-M(s_t)\,(T_\theta(s_t, a_t))$, with a minus sign because we want to maximize the log-likelihood. From this we obtain tailored parameters $\theta_{(s_t,a_t)}$ and use them to make the final prediction. In meta-tailoring fashion we train these final predictions to be close to the truth $T_{\theta_{(s_t,a_t)}}(s_t, a_t) \approx s_{t+1}$. In practice, we use the first-order version of CNGRAD.

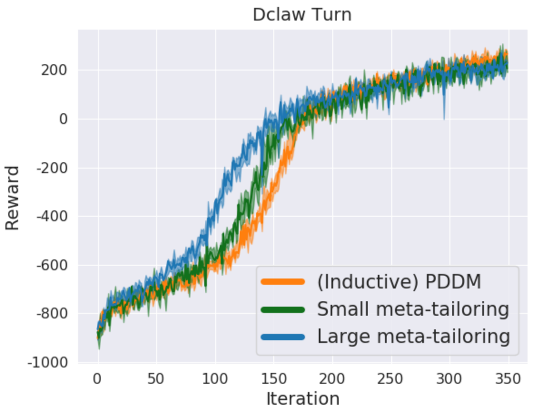

Figure 4: Meta-tailoring improves the performance of MBRL in dclaw, a complex manipulation problem, especially for low-data regimes where the transition model is imperfect and the prior is thus useful. We plot one confidence interval for the median reward bootstrapped from 16 different runs per configuration. Further increasing the tailoring learning rate results in (meta-)training instabilities because the inner loop is highly variable.

We use this idea and build on PDDM [Nagabandi et al., 2020], which recently showed great results in model-based reinforcement learning for robotic manipulation problems. Since they use an ensemble of deep networks $T_i(s_t, a_t)$ (to make robust predictions), we independently meta-tailor each network to leverage a single action-independent learned prior $M(s_t)$, leaving the rest of PDDM the same. In figure 4, we observe that meta-tailoring improves the performance of the overall predictive model, especially during early parts of training when the action-conditioned model has not become accurate. As expected, once the action-conditioned model is good enough (to get positive reward) meta-tailoring is not as useful as we do not need the action-independent prior anymore. For more experimental details, see appendix J.

5 Conclusion

We have presented *tailoring*, a simple way of embedding a powerful class of inductive biases into models, by minimizing unsupervised objectives at prediction time. We have leveraged the generality of auxiliary losses and improved them in two key ways: first, we eliminate the generalization gap on the auxiliary loss by optimizing it on the query point instead of on a training set; second, we change the optimization to minimize only the task loss in the outer optimization, and the tailoring loss in the inner optimization. This results in the whole network optimizing the only objective we care about, instead of a proxy loss. Finally, we have formalized these intuitions by proving the benefits of meta-tailoring under mild assumptions.

The framework is broadly applicable, as one can vary the model, the unsupervised loss and the task loss. We have shown its applicability in three diverse domains: physics prediction time-series, contrastive learning, and adversarial robustness. We also provide a simple algorithm,CNGRAD, to make meta-tailoring practical with little additional code. Currently, most unsupervised or self-supervised objectives are optimized in task-agnostic ways; instead, meta-tailoring provides a generic way to make them especially useful for particular applications. It does so by learning how to best leverage the unsupervised loss to perform well on the final task we care about.