Data-Efficient Temporal Regression with Multi-Task Recurrent Neural Networks

Sigurd Spieckermann[*]
Siemens AG
Corporate Technology
81739 Munich, Germany

Steffen Udluft
Siemens AG
Corporate Technology
81739 Munich, Germany

Thomas Runkler[*]
Siemens AG
Corporate Technology
81739 Munich, Germany

Abstract

We demonstrate the utility of multi-task learning with recurrent neural networks in the context of a temporal regression task using real world gas turbine data. Our goal is to learn the input-output relationship of a task with temporal dependencies given only few data by exploiting additional data of related tasks. Therefore, we propose a multi-task learning approach with many inter-task and few task-specific parameters making it data-efficient with respect to each individual task and, in particular, with respect to the target task. We motivate the relevance of our problem setting by industrial use cases, formalize it, present and discuss the models, and empirically assess their effectiveness. As a result, learning a model solely on the target task data is futile due to insufficient samples. However, our multi-task learning approach successfully exploits auxiliary data from related source tasks and dramatically boosts the predictive performance.

1 Introduction

Consider a complex technical system, e.g. a gas or wind turbine, for which a model of one or multiple sensor(s) is needed to perform tasks such as condition monitoring or model based control [7]. Analytical models explicitly incorporate the technical understanding of domain experts about the system of interest, but they may be expensive to build and may not be able to capture the full complexity of a real world task. Data driven methods, such as recurrent neural networks [1], alleviate some of the difficulties with analytical models by learning the actual input-output relationship from observations [6], but this approach often requires large amounts of data which is a scarce resource in many real world applications. For instance, the dynamics of an industrial plant may change due to maintenance or upgrades invalidating the previously optimized model. However, an accurate model is desired as soon as possible after recommissioning the plant and only few data are available after the modifications. Hence, data efficient procedures utilizing all available data are crucial.

In this paper we demonstrate the utility of multi-task learning with recurrent neural networks to successfully obtain a model of a NO_x sensor despite only relatively few data. Therefore, we introduce additional related learning tasks, i.e. learning NO_x sensor models of turbines with altered configurations. In a series of publications the Factored Tensor Recurrent Neural Network (FTRNN) was presented as a model that learns the state transition function of a dynamical system given few observations through multi-task and transfer learning [8, 10, 9]. The FTRNN consists of many inter-task and few task-specific parameters. Hence, this composite model shares knowledge among the tasks while yet being able to encode task-specific properties which makes it a data-efficient approach in particular with respect to the target task. While learning a target task model using only the available data yields poor predictive performance, multi-task learning—and especially the FTRNN—turns out to be a successful approach in our problem setting.

[*]Technical University of Munich, Department of Informatics, 85748 Garching, Germany

2 Problem Definition

Let $I := \{1, 2, 3, ...\}$ denote the set of task identifiers of related temporal regression tasks, which are observed in fixed time intervals, defined by the inputs $x_t \in X$ and the targets $y_t \in Y$ where X is the input space Y is the target space.

A data set $D := \{(i^{(j)}, x_t^{(j)}, y_t^{(j)}) \,|\, j \in \mathbb{N}, t \in \mathbb{N}\}$ of size $|D|$ is drawn i.i.d. from a probability distribution $\mathcal{D}$. The triple (i, x_t, y_t) represents an observation of task $i \in I$ made at single point in time.

Let $H \subseteq \{h \,|\, h\colon I \times X^T \to Y\}$, $T \in \mathbb{N}$, denote a hypothesis space. Further, let $h^* \in H$ be the optimal hypothesis within H. Let $\mathcal{L}\colon Y \times Y \to \mathbb{R}_{\geq 0}$ be an error metric between a predicted target $\hat{y}_t = h(i, x_{t-T+1}, ..., x_t)$ and the true target y_t. The optimal hypothesis h^* minimizes the expected error $\varepsilon(h) := \mathbb{E}_{(i,x_{t-T+1},...,x_t,y_t)\sim\mathcal{D}}[\mathcal{L}(h(i, x_{t-T+1}, ..., x_t), y_t)]$ where $\mathbb{E}$ denotes the expectation operator, hence, $h^* = \arg\min_h \varepsilon(h)$. Since $\mathcal{D}$ is generally unknown, an approximately optimal hypothesis $\hat{h}$ is determined by minimizing the empirical error $\hat{\varepsilon}_D(h) := \frac{1}{|D|} \sum_{(i,x_{t-T+1},...,x_t,y_t)\in D} \mathcal{L}(h(i, x_{t-T+1}, ..., x_t), y_t)$ induced by a hypothesis h on a data set D.

Let $I_{\text{src}} \cup \{i_{\text{tar}}\} = I$, $i_{\text{tar}} \notin I_{\text{src}}$, $|I_{\text{src}}| \neq \emptyset$, denote the set of source task identifiers and the target task identifier with $|D_{i_{\text{tar}}}| \ll |D_{i_{\text{src}}}| \; \forall i_{\text{src}} \in I_{\text{src}}$. Further, let $h^*_{i_{\text{tar}}}$ be the optimal hypothesis within H of the task i_{tar}. Assuming the amount of data from task i_{tar} is insufficient, $\hat{h}_{i_{\text{tar}}}$ is expected to differ significantly from $h^*_{i_{\text{tar}}}$. The problem addressed in this paper is to develop and assess methods that yield a better hypothesis $\hat{h}_{i_{\text{tar}}}$, learned from the data set $D_{i_{\text{tar}}}$, by exploiting auxiliary information from $\bigcup_{i\in I_{\text{src}}} D_i$, based on which a general model is learned, so that it suffices to have a small data set $D_{i_{\text{tar}}}$ in order to adjust the general model to the peculiarities of the target task i_{tar}.

3 Temporal Regression with Recurrent Neural Networks

Recurrent neural networks (RNN) are powerful models for sequence modeling tasks. In contrast to feedforward neural networks, RNNs process their input vectors $x_1, ..., x_T$, $x_t \in \mathbb{R}^{n_x}$ (n_l denotes the dimensionality of layer l) sequentially along the time axis thereby taking its sequential structure directly into account. The input sequence is mapped to a hidden state sequence $h_1, ..., h_T$, $h_t \in \mathbb{R}^{n_h}$, from which the output sequence $\hat{y}_1, ..., \hat{y}_T$, $\hat{y}_t \in \mathbb{R}^{n_y}$, is computed. Notation is slightly abused by overloading the variable h to describe the hidden state of an RNN as well as a hypothesis. In temporal regression tasks an output y_t typically depends on inputs $x_{t-d}, ..., x_t$, $d \geq 0$. Thus, the RNN needs to assemble information from d preceding inputs, where d is task-specific, in order to be able to compute the output. A simple RNN is defined in the following recursive manner

$$h_0 = 0 \tag{1a}$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{hx} x_t + b_h) \tag{1b}$$

$$\forall t \geq d\colon\; \hat{y}_t = W_{yh} h_t + b_y \tag{1c}$$

where $W_{vu} \in \mathbb{R}^{n_v \times n_u}$ is the weight matrix from layer u to layer v, $b_v \in \mathbb{R}^{n_v}$ is the bias vector of layer v and $\tanh$ is a nonlinear activation function. The loss function is typically chosen as the mean squared error (MSE) between the predicted and the true target sequence. If outliers are a concern, the $\ln\cosh$ loss function is an alternative choice.

3.1 Naïve RNN

The most naïve approach to learn a joint model of multiple tasks is to learn from the concatenated data of all tasks. This way, the model is forced to generalize over their different properties. However, since the training examples of the tasks are not distinguished, the model can only learn the average task which may be vastly suboptimal for rather different tasks. In particular, it is impossible for this type of model to disentangle the inter-task properties from their individual characteristics.

3.2 RNN+ID

One way to provide information that allows the model to distinguish between the tasks is to tag each example with an identifier $i \in I$ corresponding to the task which generated the data. Encoded as a

one-hot vector e_i, i.e. the i-th Euclidean basis vector, this tag is provided as an additional input to the model at each time step and contributes to the hidden state through $W_{hi}e_i$. In fact, $W_{hi}e_i$ yields the i-th column of W_{hi} which is equivalent to a task-specific bias vector. The task-independent and task-specific biases can be combined yielding the RNN+ID model through substitution of (1b) with (2).

$$h_t^{(i)} = \tanh(W_{hh}h_{t-1}^{(i)} + W_{hx}x_t + b_h^{(i)}) \tag{2}$$

3.3 Factored Tensor RNN

The temporal memory of an RNN is determined by the matrix W_{hh}. Thus, learning task-specific matrices $W_{hh}^{(i)}$ instead of a single task-independent one allows to learn the temporal structure of the different tasks while sharing the input-to-hidden and hidden-to-output parameters, i.e. the feature extraction and regression sub-models. However, the fact that the $W_{hh}^{(i)}$ are learned independently prevents any information sharing among the tasks with respect to the common structure among these transformations. Viewing $W_{hh}^{(i)}$ as the i-th slice of a third-order tensor we can reduce the number of task-specific parameters and increase parameter sharing by learning the factored tensor representation $W_{hh}^{(i)} \approx W_{hf}\,\mathrm{diag}(W_f^{(i)})W_{fh}$ such that only the diagonal matrix $\mathrm{diag}(W_f^{(i)})$ is task-specific. The FTRNN model is obtained by substituting (1b) with (3).

$$h_t^{(i)} = \tanh(W_{hf}\,\mathrm{diag}(W_f^{(i)})W_{fh}h_{t-1}^{(i)} + W_{hx}x_t + b_h) \tag{3}$$

4 Experiment: Gas Turbine NO_x Emissions

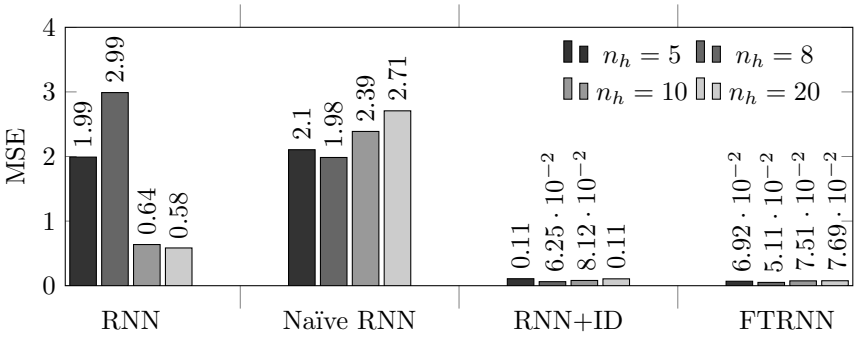
We evaluated the utility of the models presented in section 3 on real world gas turbine data comprising various hardware configurations[1]. In particular, various burners were installed and tested over the course of several weeks during which sensor data were collected. Our goal was to obtain a model of the NO_x emissions given ambient conditions (e.g. temperature, pressure, humidity) and control parameters (e.g. angle of the inlet guide vane, position of fuel valves), in total 23 values per time step. As a result of the different configurations, the NO_x production changed. Despite only relatively few samples of one of the configurations we required an accurate NO_x model of this instance.

We created six learning tasks $I = \{1, ..., 6\}$ each with a different hardware setup. Out of the six tasks, we chose five source tasks $I_{\text{src}} = I \setminus \{6\}$. The examples used to train and evaluate the models were generated by extracting T-step windows of the sequences of observations. In order to decorrelate examples of the training, validation and test data sets, the block validation method discussed in [3] was used. The source tasks comprised 6072, 7126, 5448, 8335, 1361 training examples and there were 3182 examples available of the target task i_{tar}. The training examples of the target task were upsampled to match the average number of samples of the source tasks. In order to evaluate the model error we set aside 8895 examples. We compared a simple RNN model learned only from the target task data with the Naïve RNN, RNN+ID and FTRNN. The models were configured as follows: $n_h \in \{5, 8, 10, 20\}$, $n_{f_h} = n_h$, $T = 10$, $d = 6$. For each model we trained 10 instances with randomly initialized parameters and formed an ensemble using the mean prediction. The experiments were implemented using Theano [2]. The model parameters were optimized using Hessian-Free optimization with structural damping [4, 5].

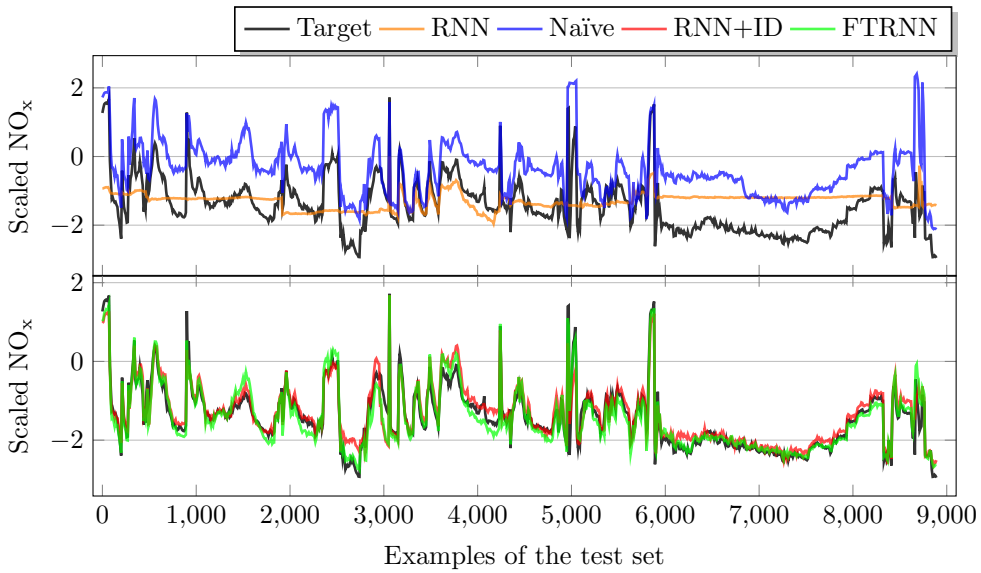
Fig. 1 compares the results of the explored models using the mean squared error per time step of the zero mean unit variance NO_x targets according to the training set standardization. As shown in Fig. 1a the FTRNN performs best across the range of considered hidden layer sizes. The RNN+ID achieves a comparable but consistently larger error. Compared to the RNN trained only on the target task, the two models yield an improvement on the order of a factor 10. The Naïve RNN, which is the simplest multi-task learning model in our comparison, suffers from the auxiliary learning tasks. In order to better understand the implications of these results we depict the prediction quality of the models in Fig. 1b by comparing their predicted NO_x values with the ground truth. Although the target task RNN outperformed the Naïve RNN in terms of the MSE we can observe that it was in fact unable to learn the input-output relationship but rather predicted a nearly constant value on the test set. The Naïve RNN appeared to have captured some of the structure of the input-output

[1] Please note that we cannot make the data publicly available due to confidentiality reasons.

relationship, but could not adjust properly for each individual task. In contrast, the RNN+ID and FTRNN could accurately predict the regression targets despite only few data of the target task.



(a) Comparison of the model errors



(b) Comparison of the model predictions

Figure 1: Comparison of model performances

5 Conclusion

We demonstrated the utility of multi-task learning in the context of real world gas turbine NO_x emission modeling. We showed that a recurrent neural network trained on the target task alone achieves poor predictive performance due to insufficient data and suggested three variants of RNNs to utilize auxiliary data. As a result, the simple concatenation of the source and target task data turned out unsuccessful. The RNN+ID, which receives a one-hot encoded task identifier as an additional input, yielded a significantly better NO_x model. The FTRNN, which encodes task-specific parameters into a diagonal third-order core tensor, consistently outperformed all other models. Despite this successful use case of multi-task learning there are many open questions yet to be answered. For instance, given only few data of the target task, it is unclear how to best employ an early stopping procedure to avoid overfitting because the validation data are not necessarily trustworthy due to their small sample size. Using a concatenated validation set of all source and target tasks has served as a proxy, but there may be alternative approaches. Further, although the FTRNN has proven to be a viable method in this and previous work, a more comprehensive series of studies is required to solidify its potential usefulness across a wider range of tasks and to systematically identify its strengths and weaknesses.

References

[1] Coryn A. L. Bailer-Jones, David J. C. MacKay, and Philip J. Withers. A recurrent neural network for modelling dynamical systems. *Network: Computation in Neural Systems*, 9(4):531–547, 1998.

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[3] Siegmund Düll, Steffen Udluft, and Volkmar Sterzing. Solving partially observable reinforcement learning problems with recurrent neural networks. In *Neural Networks: Tricks of the Trade*, pages 709–733. Springer, 2012.

[4] James Martens and Ilya Sutskever. Learning recurrent neural networks with Hessian-Free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1033–1040, 2011.

[5] James Martens and Ilya Sutskever. Training deep and recurrent networks with Hessian-Free optimization. In *Neural Networks: Tricks of the Trade*, pages 479–535. Springer, 2012.

[6] Anton M. Schäfer, Daniel Schneegass, Volkmar Sterzing, and Steffen Udluft. A neural reinforcement learning approach to gas turbine control. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1691–1696, 2007.

[7] Anton Maximilian Schäfer, Steffen Udluft, and Hans-Georg Zimmermann. The recurrent control neural network. In *Proceedings of the 15th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 319–324, 2007.

[8] Sigurd Spieckermann, Siegmund Düll, Steffen Udluft, Alexander Hentschel, and Thomas Runkler. Exploiting similarity in system identification tasks with recurrent neural networks. In *Proceedings of the 22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2014.

[9] Sigurd Spieckermann, Siegmund Düll, Steffen Udluft, and Thomas Runkler. Multi-system identification for efficient knowledge transfer with factored tensor recurrent neural networks. In *Proceedings of the European Conference on Machine Learning (ECML), Workshop on Generalization and Reuse of Machine Learning Models over Multiple Contexts*, 2014.

[10] Sigurd Spieckermann, Siegmund Düll, Steffen Udluft, and Thomas Runkler. Regularized recurrent neural networks for data efficient dual-task learning. In *Proceedings of the 24th International Conference on Artificial Neural Networks (ICANN)*, 2014.