A Framework for Improving Offline Learning Models with Online Data

Sabrina Luftensteiner
Data Science
Software Competence Center Hagenberg GmbH
Hagenberg im Muehlkreis, Austria
email: sabrina.luftensteiner@scch.at

Michael Zwick
Data Science
Software Competence Center Hagenberg GmbH
Hagenberg im Muehlkreis, Austria
email: michael.zwick@scch.at

***Abstract*—The usage of available online data is rising as machines get equipped with more sensors to control and monitor processes. The produced data can be used to directly fit existing prediction models to enhance their accuracy, adapt to alterations within the environment and avoid the training of new models. During the online learning step, which is used for the adaptation of the models using online data, catastrophic forgetting of already learned tasks may occur. We propose a new framework that utilizes several state-of-the-art methods in deep learning, as well as machine learning to minimize catastrophic forgetting. The methods range from memory-based approaches to methods for loss calculation and different optimizers, whereat the framework also provides possibilities to compare the methods and their impact with each other. The proposed framework is specifically tailored for regression problems, focusing on industrial settings in the experiments section. It is able to cope with single and multi-task models, is expandable and enables a high variety of configuration possibilities for adaptation to a given problem.**

***Index Terms*— Online Learning; Catastrophic Forgetting; Regression; Domain Adaption.**

I. Introduction

In manufacturing, the usage of sensors and microprocessors on machines and their produced data is continuously increasing. This trend is part of Industry 4.0 and enables a huge source of structured and unstructured streaming data [1]. The produced data stream is used to achieve a higher level of operational efficiency, as well as productivity and furthermore enables a higher level of automatization and flexibility [2] [3]. Another important aspect regarding Industry 4.0 is the customization of applications with small batch sizes enabling flexible adaptions and optimizations [2].

At the moment, machine learning, especially deep neural networks, are very effective in solving various tasks, including classification and regression problems [4]. Industry 4.0 is able to utilize such machine learning techniques to create self-learning and adaptive systems for predictions, predictive maintenance, outlier detection and various other evaluations [2] [3]. The used techniques have to support domain adaptation to provide models with the needed adaptability for expanding and alternating environments, which is especially useful in custom process industry.

Many of the current learning approaches are based on batch settings, meaning the complete training data set has to be available prior to the learning task [5]. As sensors in Industry 4.0 settings continuously produce new data [1] [2], which are used to optimize models, the so-called offline learning is often not sufficient enough. Online learning, on the other hand, is able to deal with such continuous data streams and dynamically changing environments and additionally enables domain adaptation [6]. Online models have the ability to gain new knowledge over time while retaining previously gained knowledge to a certain extent [7]. The main issue of online learning models is catastrophic forgetting, meaning the forgetting or fading of previously learned knowledge due to the stability-plasticity-dilemma [7] [8]. Researchers have worked on solving this dilemma and came up with various solutions, ranging from memory-based approaches [9] to parameter-specific approaches, like *Learning without Forgetting* [10] or *Natural Gradient Descent* [11]. As different scenarios need different approaches, the need for a general framework covering various approaches to minimize catastrophic forgetting arises.

The motivation for the creation of an online learning framework is the easing of the path for the development of models, especially in Industry 4.0 applications. Our approach offers a high variety of configuration possibilities for various online learning scenarios, whereat it is possible to select from different models and solution approaches to identify the most fitting approach. As a base for each online learning cycle, an already pre-trained offline model is used, which is then further trained using online data. The availability of various configurations and models allows users to experiment with similar and unresembling methods for easier comparisons regarding which configuration is more suitable for a scenario. Data can be added in mini-batches or per sample, enabling an industry-like usage as clients often have varying requirements. A main advantage of the framework is the visualization of diverse metrics, e.g., mean-squared error, for each adaptation step over time, that is further referred to as time-step. The amount of time-steps can be dynamically adapted over time and represents the next adaptation of a model with online data. Using such a representation of the model development, it is straightforward to estimate if a model adapts well to new data.

The paper covers the following points: Section 2 addresses related work and highlights missings in current literature. In Section 3, the problem setup and the basic idea of our

 ISBN: 978-1-61208-857-0

framework are discussed. A more detailed description of the framework structure is discussed in Section 4, followed by experiments and results in Section 5. Closing, Section 6 covers a conclusion and a prospective future work.

II. Related Work

In this section, we briefly recap existing frameworks and relevant topics for online and offline learning. The frameworks' advantages and drawbacks are highlighted regarding their usage in industrial environments and applicability. The second part of this section focuses on deep learning using state-of-the-art methods for decreasing catastrophic forgetting.

A. Existing Frameworks

Currently, only few frameworks with the support for online learning are mentioned in literature. An unsupervised online learning framework for moving object detection was presented 2004 by Nair et al. [12], focusing on the adaptation of the classifier. They start similar to our approach with an offline trained model and fit it continuously with new data. In comparison to their framework our framework has a broader field of application, as it can deal with various regression problems. An online multi-task learning framework for ensemble learning was developed by Xu et al. [13], focusing on time series data and insensitive loss functions. Both of the mentioned frameworks have a fixed model structure whereat Xu et al. also enable the selection of the loss function. Our framework supports the usage of various models to select from and provides additional configuration possibilities, e.g., loss function and optimizer for neural networks. These features enable the training of a broad variety of different models using the same setup and provide straightforward comparisons as the visualization of results is incorporated into the framework.

B. Relevant Deep Learning Topics

1) Multi-Task Learning: Multi-task learning uses the common knowledge of tasks to improve all tasks, whereat the bottom layers of the model are usually shared and the top layers are task-specific [14]. Such models are often used in the industrial field, as they enable the training of similar tasks in one model to save time and even enhance results. Continuously adding new tasks or data to an already trained multi-task model could lead to unwanted side-effects such as catastrophic forgetting [10].

The proposed framework supports the use for (online) domain adaptation [15], i.e., learning a model based on a source domain that performs sufficiently well on different but related target domains [16]. This scenario frequently arises in industrial settings, e.g., when a previously learned model needs to be applied in a different machine/tool setting or with different materials. This is often achieved by finding a common feature representation where source and target distributions become as similar as possible [17] [18].

2) Catastrophic Forgetting: Catastrophic forgetting is one of the main constraints in online learning and is caused by the stability-plasticity dilemma. This dilemma states that a model requires a certain plasticity for the integration of new knowledge, but also stability to prevent the fading of previously gained knowledge [19]. Researchers engaged themselves in finding a solution for this problem and came up with various approaches. Li et al. [10] propose a method which preserves the original capabilities of a multi-task deep learning model by taking the response of old tasks for the new data into account for the loss calculation. Kirkpatrick et al. [4] developed an algorithm analogous to synaptic consolidation in brains, which decelerates learning on certain weights in deep learning models depending on how important they seem to previously seen tasks. Zhang et al. [20] created a new optimizer where they exploit a connection between natural gradient descents and variational inference to enable further adaptive training. Rusu et al. [21] propose an approach where the network can utilize preliminary knowledge via lateral connections to previously learned features. Castro et al. [9] store the most representative samples of a task in the representative memory of the model and later reuse them for the fitting process.

III. Problem Setup

Consider an unknown target function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ with N input features, e.g., a virtual sensor in an industrial manufacturing setting predicting an unobserved process quality measure based on N physical sensor measurements. Typically, the physical sensors are cost efficient, whereas the quality measure can only be recorded with significant effort with respect to hardware cost and setup time.

In a first step (offline step), a regression model $h : \mathbb{R}^N \rightarrow \mathbb{R}$ is trained using a homogeneous set $(X_S, Y_S = f(X_S))$ of all (source) data available at this point in time. The trained model is then used in production to cyclically predict the quality measure in order to control the production process.

At some point (online step), the model is required to adapt to additional operating environments, e.g., different machine/tool settings or additional materials. At this point, new (target) data $(X_T, Y_T = f(X_T))$ is gathered under the new operating environment that is different but related to the original data set (X_S, Y_S). The framework then has to learn an updated model $h' : \mathbb{R}^N \rightarrow \mathbb{R}$, which is able to predict the new data (X_T, Y_T) while still retaining the performance of the original source data (X_S, Y_S).

As during the production process data can be generated continuously or in bulk, the framework needs to be able to train the updated model incrementally, i.e., using only the next available sample or alternatively all the available data from the new operating environment at once.

IV. Framework

In this section, we describe the main parts of our framework, focusing on the central configuration file, as well as the learning algorithm. The framework was implemented in Python 3, using the PyTorch deep learning framework.

 ISBN: 978-1-61208-857-0

```
CONFIG = {
  'App_Scen1': {
    'TASK_DICT': {
      'task_1': [(1, 50, False),
        (2, 20, True, True),
        (3, 10, False, True)]
      'task_2': [(2, 40, False),
        (3, 30, True, True)]
    }
     # additional dictionary entries
  }
  'App_Scen2': {
    # other scenario entries
  }
}
```

Fig. 1. General Configuration Dictionary.

A. Configuration

The configuration file is the core of our framework and is represented as a dictionary.

1) Application Scenario Configuration: The dictionary, see Fig. 1, is able to store configurations for more than one application scenario in sub-dictionaries, making it possible to access a specific scenario setup by using its key. The scenario dictionary contains, next to information about loading and saving paths for models and (training) data, information about the training process represented in a dictionary containing the different tasks of a scenario.

2) Task Training Configuration: The task dictionary consists of n entries, each containing a task name and an array of tuples, which represent the processing of the available data. One tuple contains four pieces of information. First of all, the time-step is specifying at which time-slot the task data is used in the model. Time-slots represent the the course of adding new online data in our mock-up scenario and are defined by the data assigned to the single time-steps. As the results and intermediate models are stored, it is straightforward to add new time-steps and start the model at specific steps, which enables a dynamic training. The second tuple value defines the used percentage of the available task data. It has a range between 1-100 whereat the sum of percentages for one task should not exceed 100. If the overall percentage of a task is below 100, the remaining data is used for testing. The third element of the tuple is a boolean flag indicating if the data should be trained elementwise or batchwise.

3) Optimizer and Loss Configuration: The framework can be used with arbitrary regression models as long as they are supported by the frameworks wrapper class (see Section IV-B). In case neural networks are used as training models, the framework enables the selection of an optimzer and a loss function. Currently, the framework supports the following optimizers and loss functions:

- Stochastic Gradient Descent (Optimizer)
- Noisy Natural Gradient Descent (Optimizer), as described by Zhang et al. [20]
- Mean Squared Error (Loss)
- Learning without Forgetting (Loss), as described by Li et al. [10]
- Elastic Weight Consolidation (Loss), as described by Kirkpatric et al. [4]

We adapted the Learning without Forgetting approach [10] by using mean squared error instead of multinomial logistic loss used for the loss calculation of the new task in order to support regression tasks. We also had to adapt the elastic weight consolidation approach [4] by using mean squared error instead of cross entropy loss.

4) Other Configurations: Furthermore, the framework enables the selection of source and target column(s) as maybe not all features are used during training in order to allow different experiments with the available data for new models. Additionally, it is possible to define at which steps to start and end a learning run. This is especially useful for stopping training at a specific step and continuing later, as the model can be reloaded and trained further at a later time.

B. Wrapper

A wrapper stores the model and enables equal treatment of models. It is used as intersection point between the model and the learning algorithm of the framework and stores additional information, e.g., prediction results. The wrapper class also contains the calculation methods for the metrics, of which Root Mean Squared Error (RMSE), Maximum Absolute Error (MaxAE), sigma, sigma^2 and R^2 are available. Currently, the wrapper supports neural networks, linear regression models, elastic net models and random forest models.

C. Algorithm

For an outline of our main learning loop see Fig. 2. At first, the data is loaded and stored in memory according to the definition in the configuration, see Fig. 1. Afterwards, an offline model is either loaded or trained and placed into a wrapper, which is described in section IV-B. After this step, the main part of the algorithm starts. As long as the last time-step is not reached, the data belonging to the time-step is fetched from the dictionary and used to fit the model. Depending on the configuration of the task at the time-step, the model is either fitted batchwise or elementwise with the available task data. Adding new tasks is performed in the wrapper and, therefore, not incorporated in the algorithm, see Fig. 2. After fitting of the model, it is evaluated with the test data. After the last time-step is completed, the model can be saved depending on the configuration. Finally, the selected metrics are calculated and visualized.

D. Visualization

The framework enables visualizations of the training and testing results as Portable Document Format (PDF) plots, e.g., see Fig. 3 and Fig. 4. Numerical results are additionally stored in Excel sheets. In the configuration files, a user can specify

 ISBN: 978-1-61208-857-0

Step 1 : Load data and partion data according to configuration file into train/test, batchwise/elementwise and online/offline
Step 2 : Load or train offline model
while time-steps available **do**
  Step 3 : Get data for time-step
  Step 4 : Fit model with according data
  Step 5 : Evaluate model with test data
  Step 6 : Get next time-step
end
Step 7 : If required, save model
Step 8 : Calculate and visualize metrics

Fig. 2. Learning Algorithm. The algorithm can be started multiple times with varying time-steps and, therefore, enables a dynamic type of online learning.

Fig. 3. Visualization example with black mean-line. Dots represent training results, half-filled squares testing results.

which metrics are visualized (multiple metrics are possible) and whether training and testing results should be visualized separately. Additionally, it is possible to anonymize the results in case of sensitive information has to be visualized.

V. Experiment

A. Dataset

We show the usability of our proposed framework using a resin production dataset provided by the Austrian company Metadynea. The dataset consists of three different recipes, each containing 5639 samples. Each sample of the dataset consists of 2692 features, which are composed of the following values: sample id, sample time, date, batch, spectrum light intensity, process pressure, process temperature, condensation time and various values representing the spectrum trend. The target is represented by a reference value measured in C.

B. Setting

1) Dataset Partition: For the experiment, we use five time-steps to simulate online learning whereat the first one is used to train an offline model. To simulate a real world application, with regards to adding new tasks and enhancing existing tasks, we partitioned the dataset the following way:

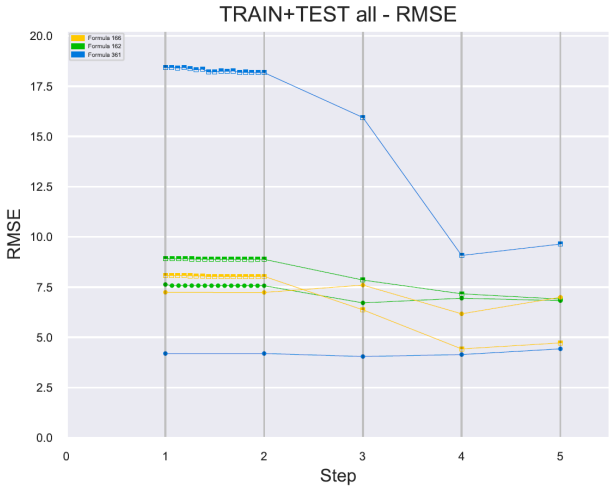

Fig. 4. Visualization example including elementwise adding of data.

- **Recipe 166**: 30% used in offline training at step 1; 40% added at step 4; 25% added at step 5; 5% used for testing
- **Recipe 162**: 25% used in offline training at step 1; 25% added at step 2; 25% added at step 3; 25% used for testing
- **Recipe 361**: 80% added at step 3; 19% added at step 4; 1% used for testing

2) Deep Learning Models: We use the following network architectures for our learning scenarios:

- **Feed-Forward Network** with 5 hidden layers, containing 35/30/25/15/7 hidden nodes
- **Feed-Forward Multi-Task Network** with 4 hidden layers, containing 300/50/30/15 hidden nodes

The models use learning rate schedulers and early stopping, as well as warm starts in the online learning part.

3) Model Configurations: To demonstrate the framework's possibilities, we chose the following model configurations for our experiments:

- Feed-Forward (Single and Multi-Task) with Stochastic Gradient Descent and MSE
- Feed-Forward Multi-Task with Stochastic Gradient Descent and Learning without Forgetting (LwF)
- Feed-Forward (Single and Multi-Task) with Natural Gradient Descent and MSE
- Feed-Forward Multi-Task with Natural Gradient Descent and LwF
- Feed-Forward Multi-Task with Stochastic Gradient Descent and Elastic Weight Consolidation (EWC)
- Linear Regression
- Random Forest
- Elastic Net

C. Results

The results of the experiments are presented in Table I, which contains the RMSE results of the training and testing environments. It is possible to see an improvement in nearly every model configuration regarding the first and last step of

 ISBN: 978-1-61208-857-0

TABLE I
RMSE TRAIN AND TEST VALIDATION RESULTS.

	Train/Test	Step 1 (Offline)	Step 2 (Online)	Step 3 (Online)	Step 4 (Online)	Step 5 (Online)
FF S with SGD/MSE	Train	10.73	13.49	10.15	7.80	9.11
FF S with SGD/MSE	Test	8.85	9.93	9.97	6.65	7.26
FF S with NGD/MSE	Train	12.87	12.7	9.81	8.32	8.28
FF S with NGD/MSE	Test	8.35	11.34	7.36	7.15	6.34
FF M with SGD/MSE	Train	13.24	11.76	10.30	10.38	9.19
FF M with SGD/MSE	Test	12.59	13.32	10.18	10.31	8.99
FF M with SGD/LwF	Train	10.72	13.71	10.15	7.08	9.11
FF M with SGD/LwF	Test	8.85	13.24	9.97	6.65	6.27
FF M with SGD/EWC	Train	19.13	18.47	20.01	18.78	21.46
FF M with SGD/EWC	Test	17.19	17.46	18.67	18.13	20.33
FF M with NGD/MSE	Train	10.91	12.42	9.80	10.02	10.50
FF M with NGD/MSE	Test	10.64	11.43	9.48	9.71	9.26
FF M with NGD/LwF	Train	12.87	12.70	9.81	8.32	**8.20**
FF M with NGD/LwF	Test	9.33	11.34	7.36	7.16	**6.29**
Linear Regression	Train	2.06E-11	1.89	5.98	5.45	**5.8**
Linear Regression	Test	41.92	21.34	9.54	7.32	**7.47**
Random Forest	Train	6.44	2.30	4.67	5.43	9.78
Random Forest	Test	12.14	10.28	10.12	10.10	10.13
Elastic Net	Train	14.12	12.69	11.50	10.32	10.75
Elastic Net	Test	10.32	10.37	10.10	10.05	10.08

both training and testing results. The first time-step, Step 1 (Offline), represents the base model, which is trained offline from scratch. Step 2 to Step 5 represent the adaptation of models with online data over time. The models should not have a decreasing performance due to the integration of online data, as a worse RMSE would indicate a less fitting model.

In general, the models performed quite differently at the same steps as for some models the RMSE is increasing whereat the RMSE is decreasing for others. The best performing machine learning model in our experiment is the linear regression model, although it is stagnating during the training process. The best performing neural network model is the Feed-Forward Multi-Task model with Natural Gradient Descent using Learning without Forgetting (FF M with NGD/LwF). It is one of the few models continually improving during the online training process.

The insertion of a new task at the third step enhances all of the neural network models, except the one model using the Elastic Weight Consolidation. The linear regression model and the random forest model are not able to deal with the new task as well as other models, which decreases their improvement although they still have good results. According to our experimental setting, a feed-forward multi-task model using LwF to minimize catastrophic forgetting fits best to our scenario and should be considered for further usage.

VI. CONCLUSION AND FUTURE WORK

Concluding, we present a framework which is able to improve offline learning models with online data. The need for such a framework increases as continuously more data is produced, especially in Industry 4.0 environments, and the training of a new model is either computationally expensive or not possible, e.g., old data may not be available anymore. The online learning setting in our framework enables the adaption of models using new data, either batchwise or elementwise, and additionally allows the inclusion of new tasks. The learning process of a model can be visualized to enable the evaluation of its development using different metrics. The framework enables various configuration possibilities regarding the training process and the models itself. The main focus of the model configuration is on the avoidance of catastrophic forgetting and, therefore, includes different state-of-the-art approaches for decreasing this problem. The amount of possible configurations make the framework flexible and adaptive regarding new regression problems The framework can easily be extended regarding supported models as currently only neural network and some selected machine learning models are available.

As the framework is still under development and we presented its current state in this paper, we will briefly outline further steps. Currently, we are working on the handling of censored online data, which are highly likely to cause negatively biased models, to reach a broader application area. Additionally, we want to enable more flexible neural network architectures and the parallel training of selected models resulting in the automatic selection of the best to continue working with.

ACKNOWLEDGMENT

The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry for Digital and Economic Aairs, and the Province of Upper Austria in the frame of the COMET center SCCH.

REFERENCES

[1] K. Zhou, T. Liu, and L. Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE, 2015, pp. 2147–2152.

[2] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *Journal of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.
[3] A. Schütze, N. Helwig, and T. Schneider, "Sensors 4.0–smart sensors and measurement technology enable industry 4.0," *Journal of Sensors and Sensor Systems*, vol. 7, no. 1, pp. 359–371, 2018.
[4] Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
[5] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: learning deep neural networks on the fly," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 2660–2666.
[6] L. C. Jain, M. Seera, C. P. Lim, and P. Balasubramaniam, "A review of online learning in supervised neural networks," *Neural Computing and Applications*, vol. 25, no. 3-4, pp. 491–509, 2014.
[7] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.
[8] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European symposium on artificial neural networks (esann)*, 2016.
[9] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 233–248.
[10] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.
[11] M. Rattray, D. Saad, and S.-i. Amari, "Natural gradient descent for online learning," *Physical review letters*, vol. 81, no. 24, p. 5461, 1998.
[12] V. Nair and J. J. Clark, "An unsupervised, online learning framework for moving object detection," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. IEEE, 2004, pp. II–II.
[13] J. Xu, P.-N. Tan, J. Zhou, and L. Luo, "Online multi-task learning framework for ensemble forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1268–1280, 2017.
[14] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.
[15] B.-D. Shai *et al.*, "A theory of learning from different domains," *Machine Learning*, vol. 79, pp. 151–175, 2010.
[16] K. Crammer, M. Kearns, and J. Wortman, "Learning from multiple sources," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 321–328.
[17] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.
[18] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (CMD) for domain-invariant representation learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
[19] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers in psychology*, vol. 4, p. 504, 2013.
[20] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse, "Noisy natural gradient as variational inference," in *International Conference on Machine Learning*, 2018, pp. 5847–5856.
[21] A. Rusu *et al.*, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

 ISBN: 978-1-61208-857-0