# Reducing catastrophic forgetting with learning on synthetic data

Wojciech Masarczyk
Institute of Theoretical and Applied Informatics,
Polish Academy of Sciences
Tooploox
wojciech.masarczyk@gmail.com

Ivona Tautkute
Polish-Japanese Academy of Information Technology
Tooploox
ivona.tautkute@tooploox.com

## Abstract

*Catastrophic forgetting is a problem caused by neural networks' inability to learn data in sequence. After learning two tasks in sequence, performance on the first one drops significantly. This is a serious disadvantage that prevents many deep learning applications to real-life problems where not all object classes are known beforehand; or change in data requires adjustments to the model. To reduce this problem we investigate the use of synthetic data, namely we answer a question: Is it possible to generate such data synthetically which learned in sequence does not result in catastrophic forgetting? We propose a method to generate such data in two-step optimisation process via meta-gradients. Our experimental results on Split-MNIST dataset show that training a model on such synthetic data in sequence does not result in catastrophic forgetting. We also show that our method of generating data is robust to different learning scenarios.*

## 1. Introduction

Deep learning methods have succeeded in many different domains such as: scene understanding, image generation, natural language processing [2, 28, 26, 20]. While deep learning methods differ in architecture choice, objective function or optimization strategy, they all assume that the training data is independent and identically distributed (i.i.d). Methods built on this assumption are effective for fixed environments with stationary data distributions – where tasks to be solved do not change over time or classes present in the dataset are known from the beginning. However, in most real-life scenarios this assumption is violated and there is a need for methods that are able to handle such cases. Among many examples of such scenarios, a few can be highlighted: new object class is introduced, however the dataset used to train the baseline model is no longer available; the data characteristics seem to change seasonally and model needs to change its predictions accordingly to these trends. Continual learning [22] is a paradigm where data is presented sequentially to the algorithm without the ability to manipulate this sequence. Additionally, there is no assumption about the structure of the sequence. A successful continual learning algorithm needs to be able to learn a growing number of tasks, be resistant to catastrophic forgetting [18] and be able to adapt do distribution shifts. The memory and computational requirements of such algorithm should scale reasonably with the incoming data.

Although the problem of continual learning is known for many years [22, 18], only recently has the field gained significant traction and many interesting ideas have been proposed. Most of continual learning contributions can be divided into three categories [13, 21]: optimization, architecture and rehersal. Methods based on optimization modifications usually add additional regularization terms to objective function to dampen catastrophic forgetting [9, 12]. Second category gathers methods that propose various architectural modifications e.g. Progressive Net [23] where increasing capacity is obtained by initialising new network for each task. The last category – rehersal based methods – consists of methods that assume life-long presence of a subset of historical data that can be re-used to retain knowledge about past tasks [14, 5].

This work proposes a new data-driven path that is orthogonal to existing approaches. Specifically, we would like to explore the possibility of creating input data artificially in a coordinated manner in such a way that it reduces

the catastrophic forgetting phenomena. We achieve this by combining two separate neural networks connected by two-step optimisation. We use generative model to create synthetic dataset and form a sequence of tasks to evaluate learner model in continual learning scenario. The sequence of synthetic tasks is used to train the learner network. Then, the learner network is evaluated on real data. The loss obtained on real data is used to tune the parameters of the generative network. In the following step, the learning network is replaced with a new one.

Differently from existing approaches, our method is independent of training method and task and it can be easily incorporated to above-mentioned strategies providing additional gains.

## 2. Related Work

One line of research for continual learning focuses on optimization process. It draws inspiration from the biological phenomena known as synaptic plasticity [1]. It assumes that weights (connections) that are important for particular task become less plastic in order to retain the desired performance on previous tasks. An example of such approach is Elastic Weight Consolidation (EWC) [9], where regularisation term based on Fisher Information matrix is used to slow down the change of important weights. However accumulation of these constrains prevents network from learning longer sequences of tasks. Another optimization based method is Learning without Forgetting (LwF) [12]. It tries to retain the knowledge of previous tasks by optimizing linear combination of current task loss and knowledge distillation loss. LwF is conceptually simple method that benefits from knowledge distillation phenomenon [6]. The downside of such approach is that applying LwF requires additional memory and computation resources for each optimization step.

Methods based on architectural modifications allow to dynamically expand/shrink networks, select sub-networks, freeze weights or create additional networks to preserve knowledge. Authors of [23] propose algorithm that for each new task creates a separate network (a column) that is trained to solve particular task. Additionally, connections between previous columns and the current column are learned to enable forward transfer of knowledge. This algorithm avoids catastrophic forgetting completely and enables effective transfer learning. However the computational cost of this approach is prohibitive for longer sequences of tasks. Other methods [10, 30] address the problem of computational cost by expanding single layers/neurons instead of whole networks, however these methods has less capacity to solve upcoming tasks. Different approaches that modify architectures are based on selecting sub-networks used for solving current task in such a way that only a fraction of network's parameters relevant to current task is changed [17, 16, 3]. The challenge here is to balance the number of frozen and active weights in such way that network is still able to learn new tasks and preserve current knowledge.

Rehearsal methods are based on the concept of memory replay. It is assumed that subset of previously processed data is stored in memory bank and interleaved with upcoming data in such a way that neural network learns to solve current task in addition to preserving current knowledge [14, 29, 5]. A naive rehearsal method would be to save random data samples that were present during training. However such approach is inefficient, since samples are not equally informative, hence the challenge of rehearsal methods is to choose the most representative samples for a given dataset, such that minimum storage is occupied. In [29], authors apply method of dataset distillation based on meta-gradient optimization to reduce the size of memory bank. It is possible to represent whole class of examples just by storing one carefully-optimized example. Unfortunately, applying this meta-optimization method is computationally exhaustive. The biggest downside of using rehearsal based methods is the need to store the actual data which in some cases can violate data privacy rules or can be computationally prohibitive. To mitigate this issue solution based on Generative Networks was proposed [31, 24]. Namely, they use dual model architecture composed of learner network and generative network. Role of the generative network is to model data previously experienced by the learner network. Data sampled from the generator network is used as a rehersal data for learner network to reduce the effect of catastrophic forgetting.

Our method is also dual architecture model based on generative network, however the aim of generative network is radically different. In contrast to authors [31, 24] we do not aim to capture the statistics of real data, instead we try to generate entirely synthetic data such that when learner does learn on a sequence of such data it does not suffer from catastrophic forgetting.

## 3. Method

The main idea of our approach is to generate data samples such that network trained on them in sequence would not suffer from catastrophic forgetting. One of many ways to generate artificial data is to use meta-optimization strategy introduced in [15]. It is shown that by applying meta-learning it is possible to use gradient optimization both to hyperparameters and to input data. However, this approach is limited to small problems, since each data point must be optimised separately. To overcome this bottleneck, authors of Generative Teaching Networks (GTNs) [27] use generative network to create artificial data samples instead of directly optimizing the data input. We adopt similar approach in our method, namely, we use generative network – green rectangle "Generator" in Fig. 2 – to produce synthetic

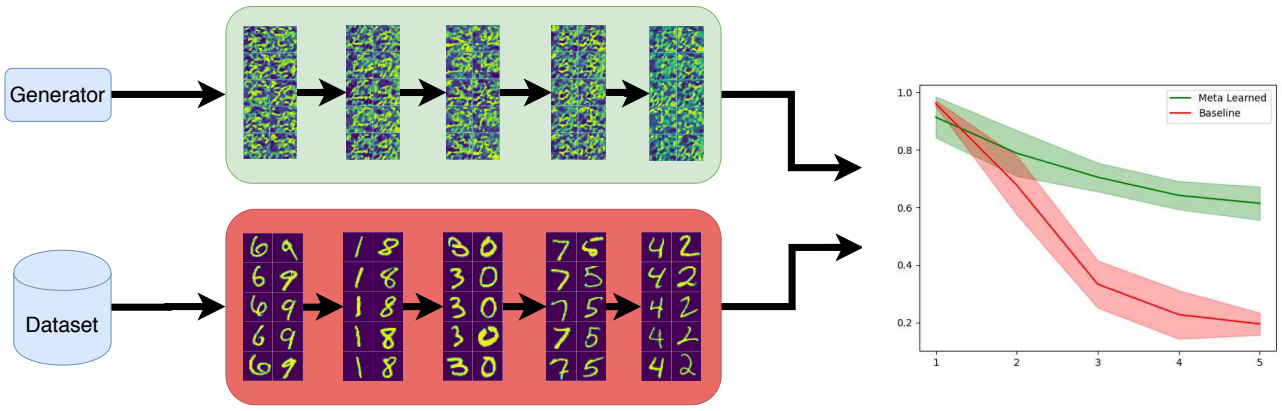


Figure 1. Synthetic data created from generator is divided into five tasks according to classes and learner (green) learns tasks sequentially. The same procedure is applied to learner with real data (red). The right plot shows that accuracy at the end of each task does not decrease on learned data in contrast to real data where it deteriorates sharply.

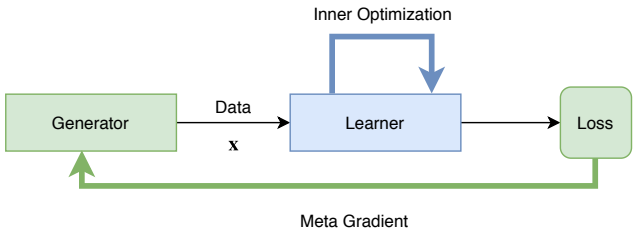


Figure 2. Synthetic data from generator is passed to learner where the inner optimization is performed and meta-loss is backpropagated to $\mathcal{G}$.

data from noise vectors sampled from a random distribution. Next, we split the data into separate tasks according to classes and form a continual learning task for the learner network – blue rectangle in Fig. 2. Learner network after completing whole sequence of tasks in evaluated on real training data. The loss from real data classification after learning all tasks in sequence is then backpropagated to generator network to tune the parameters as shown in Fig. 2.

Our approach is similar to one proposed in work [7]. Using two step meta-learning optimization they try to learn best representation of input data such that the model learned in with standard optimization does not suffer from catastrophic forgetting.

Differently from [27], we do not use curriculum based learning as our goal is to have a realistic continual learning scenario where the order of data sequence is not known beforehand. To ensure that the Generator network does not generate data suitable for particular sequence of tasks at each meta-optimization we shuffle order of tasks. Precisely, at each step we generate $p$ samples for each class and then randomly create a sequence of binary classification tasks with particular data.

Precisely, let $\mathcal{G}$ be a generative neural network, $\mathcal{S}$ a standard convolutional network for classification, $\mathbf{t} = (t_1, t_2, \dots t_n)$ a sequence of tasks, where each tasks is binary classification task and classes in each task form mutually disjoint sets.

The inner training loop consists of sequence of tasks, where generated samples from previous tasks are not replayed once the task is finished. To achieve this, the sequence of tasks $\mathbf{t} = (t_1, t_2, \dots t_n)$ must be defined a priori and samples generated by network $\mathcal{G}$ are conditioned on the information of particular task. For each task $t_i$ the network $\mathcal{G}$ generates two batches of samples $\mathbf{x} = \mathcal{G}(\mathbf{z}, \mathbf{y_{i_j}})$ for $j = 1, 2$, where $\mathbf{z}$ is a batch of noise vectors sampled from Normal distribution and $\mathbf{y_{ij}}$ is a class indicator for task $\mathbf{t_i}$. Note that generator networks has access to class indicators since we aim to learn in continual learning scenario only the learner network.

Neural network $\mathcal{S}$ learns sequentially on following tasks using standard SGD optimizer with learning rate and momentum optimized through meta-gradients. At the end of the sequence $\mathbf{t}$ network $\mathcal{S}$ is evaluated on real dataset $(\mathbf{x_r}, \mathbf{y_r})$ obtaining meta-loss as shown in Fig. 2. This meta-loss is backpropagated through all training inner-loops of model $\mathcal{S}$ to optimize network $\mathcal{G}$. Parameters $\theta$ of network $\mathcal{G}$ are updated according to the equation:

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\mathcal{S}(\mathbf{x_r}; \mathbf{w_m}), \mathbf{y_r}), \tag{1}$$

where $\mathbf{w_m}$ are parameters of the network $\mathcal{S}$ after $m$ optimization steps, $\eta$ is fixed learning rate, $\mathcal{L}$ is a cross entropy loss function, $\mathbf{x_r}, \mathbf{y_r}$ are real data samples and labels respectively.

## 4. Experiments

To test our hypothesis we use popular continual learning benchmark Split-MNIST [11, 25]. In first experiment, we use 5-fold split with two classes for each task to create a moderately difficult sequence of tasks. Network $\mathcal{G}$

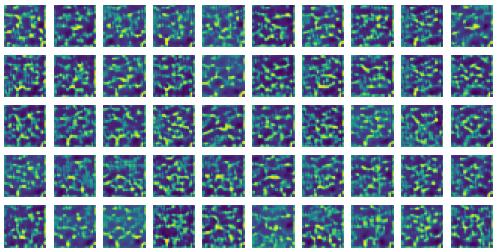

Figure 3. Samples generated by network $\mathcal{G}$ at the end of meta-optimisation. Starting from zero (leftmost), each sample to the right represents the following class.

generates 250 samples per each class. During inner optimisation learner network is optimized on batch size formed with 40 generated images (20 samples per class drawn randomly from the pool of 250 samples per class). We train the learner network on each task for 5 inner steps with batch size 40. Once the task is over, samples from this task are not shown to the network to the end of training. At test time, after learning on each task the network is evaluated on part of a test set composed of classes seen in previous taks. Both networks are simple convolutional neural networks with two convolutional layers with addition of one and two fully connected layers for classification and generative network respectively. Each layer is followed by a batch normalisation layer.

As a baseline to compare with, we use simple fully connected network proposed in [8] ('MLP' – red – in Fig. 4). To further investigate the impact of generated data we use the same network architectures and optimizer settings with learning rate and momentum optimized with by a meta learning process as described in Section 3 but for optimizing the learner network we use real data ('Real Data' – yellow – in Fig. 4). We also compare our results with GAN-based data samples. In this scenario we follow the setting of 'Real Data' scenario except for the source of data. We use Conditional-GAN [19] to model the original data distribution and then sample 250 samples per each class ('GAN based' – blue – in Fig. 4).

We implement experiments in PyTorch library, which is well suited for computing higher-order gradients [4].

**Results** – obtained results support our hypothesis, that it is possible to generate synthetic data such that, even if networks learns this data in sequence (one time per sample), the learning process does not result in castastrophic forgetting.

Figure 4 shows how learning on synthetic data in sequence results in less catastrophic forgetting compared to learning on a sequence of real data samples. Note that additional performance could be gained with careful hyperparameter tuning, however we did not want to compete for best performance and rather show the potential of this approach. Higher accuracy of 'Real data' scenario over 'MLP' can be attributed to the effectiveness of optimised learning rate and momentum parameters, however the main advantage comes from using meta learned data samples. Results obtained with data generated with GAN are almost identical to ones obtained with real data. This result is expected as the data modeled by a GAN resembles original data closely.

An example batch of generated samples is shown in Figure 3. The samples are ordered according to classes (starting from 0). In contrast to [27] the data samples are abstract blobs, rather than interpretable images. We verify experimentally that the reason for the lack of structure in generated samples is the lack of curriculum learning in our scenario. We skip it intentionally to provide more realistic continual learning scenario for the learner network.

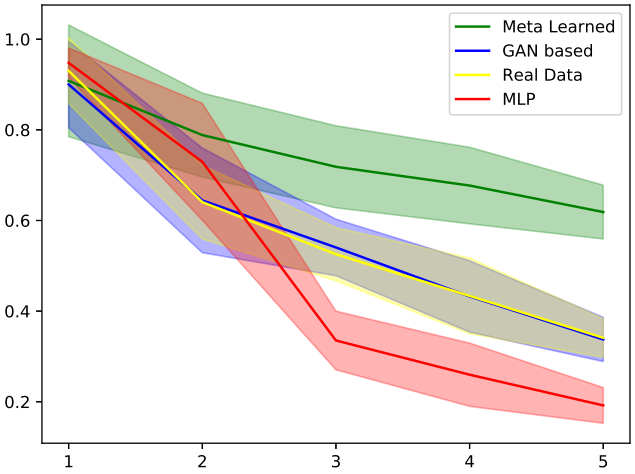


Figure 4. Overall accuracy measured on test data subset. After learning each task, test data subset is made of samples only from classes seen during recent and previous tasks.

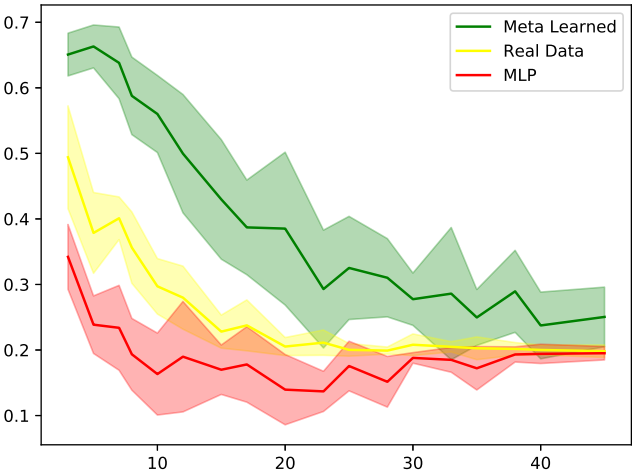


Figure 5. Overall accuracy measured on test set after learning network $\mathcal{S}$ with synthetic data for $x$ inner steps on each task.

Fig. 5 shows the impact of change of learning scenario of network $\mathcal{S}$ *after* network $\mathcal{G}$ is trained. In this experiment data generated by a network $\mathcal{G}$ in first experiment is used. Here, we investigate how the final accuracy after learning five consecutive tasks changes with the number of inner optimization steps. Note that $\mathcal{G}$ was optimised to create samples that are robust to catastrophic forgetting with inner optimization loop of 5 steps. As we can see, in case of longer learning horizon, network learned on synthetic (green plot Fig. 5) data suffers significantly less than the same network learned on real data (yellow plot Fig. 5). Even though accuracy of the networks drops with increasing number of inner steps, the drop is smoother in case of synthetic data.

## 5. Conclusions

The aim of this work was to answer a question, whether it is possible to create data that would dampen the effect of catastrophic forgetting. Experiments show that this hypothesis is true – it is possible to generate such samples, however usually they do not visually resemble real data. Surprisingly, even applying the method alone can result in high performing network. Additional interesting advantage of this synthetic data is the robustness to changes of inner optimisation parameters – increasing 15-fold size of a batch and length on training still results in compelling performance. We believe that our experiments open a new and exciting path in continual learning research. As a future work we plan to adjust current method to datasets of higher complexity and test its effectiveness in online learning scenario.

## 6. Acknowledgements

Authors would like to thank Petr Hlubuček and GoodAI for publishing the code at `https://github.com/GoodAI/GTN`.

## References

[1] Joseph Cichon and Wen-Biao Gan. Branch-specific dendritic ca2+ spikes cause persistent synaptic plasticity. *Nature*, 520, 03 2015.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *CoRR*, abs/1903.04476, 2019.

[4] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[5] Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. *CoRR*, abs/1809.05922, 2018.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[7] Khurram Javed and Martha White. Meta-learning representations for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1820–1830. Curran Associates, Inc., 2019.

[8] Ronald Kemker, Angelina Abitino, Marc McClure, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *CoRR*, abs/1708.02072, 2017.

[9] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.

[10] Jeongtae Lee, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *CoRR*, abs/1708.01547, 2017.

[11] Sang-Woo Lee, Jin-Hwa Kim, JungWoo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *CoRR*, abs/1703.08475, 2017.

[12] Zhizhong Li and Derek Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016.

[13] Vincenzo Lomonaco. *Continual Learning with Deep Architectures*. PhD thesis, 2018.

[14] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continuum learning. *CoRR*, abs/1706.08840, 2017.

[15] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2113–2122. JMLR.org, 2015.

[16] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. *CoRR*, abs/1711.05769, 2017.

[17] Arun Mallya and Svetlana Lazebnik. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. *CoRR*, abs/1801.06519, 2018.

[18] Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

[19] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[20] Witold Oleszkiewicz, Tomasz Wlodarczyk, Karol J. Piczak, Tomasz Trzcinski, Peter Kairouz, and Ram Rajagopal. Siamese generative adversarial privatizer for biometric data. *CoRR*, abs/1804.08757, 2018.

[21] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with

neural networks: A review. *Neural Networks*, 113:54 – 71, 2019.

[22] Mark Bishop Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, USA, 1994.

[23] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016.

[24] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *CoRR*, abs/1705.08690, 2017.

[25] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2310–2318. Curran Associates, Inc., 2013.

[26] Wojciech Stokowiec, Tomasz Trzcinski, Krzysztof Wołk, Krzysztof Marasek, and Przemyslaw Rokita. Shallow reading with deep learning: Predicting popularity of online content using only its title. pages 136–145, 07 2017.

[27] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeff Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data, 2020.

[28] Ivona Tautkute, Tomasz Trzcinski, Aleksander Skorupa, Lukasz Brocki, and Krzysztof Marasek. Deepstyle: Multimodal search engine for fashion and interior design. *CoRR*, abs/1801.03002, 2018.

[29] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation. *CoRR*, abs/1811.10959, 2018.

[30] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. *CoRR*, abs/1907.07844, 2019.

[31] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: learning to generate images from new categories without forgetting. *CoRR*, abs/1809.02058, 2018.