Pay attention to the Task: Avoid forgetting in sequential learning through dynamic weight allocation

Felix Geilert
felixnext@outlook.com

Abstract

While learning multiple tasks, especially if trained in sequential order, classic learning techniques for DNNs tend to leverage the weights of the network in a suboptimal manner. This results in the overwriting of relevant information and in turn to catastrophic forgetting. Such information loss limits the capabilities for real-world learning of these network architectures and remains one of the hurdles towards artifical general intelligence. In this paper, we introduce a novel attention based technique that learns a semantic distribution of information across network weights. Our approach generates weight masks during inference, allowing the network to choose optimal weight configurations based on input type classification. We show that our approach drastically reduces forgetting ratios on sequentially learned datasets and provides further insight into the weight allocation of neural networks. Furthermore, we show that it learns a semantic understanding of weight relevance w.r.t. task information, which we believe can also aid generalization and one-shot learning.

1 Introduction

Current DNNs still struggle with the retaining and reusing of information presented to them sequentially across multiple tasks (*continual learning*). This problem is known as *catastrophic forgetting* (CF) [1, 2] and provides a major roadblock towards more complex systems. CF describes the process of overwritting previously learned information after a task switch due to sequential learning or learning without a rehearsal curriculum. In comparision, the human brain is able to extract generalized information from few training examples and even refine general contexts given information from different domains [3, 4]. The problem of CF is especially pronounced in real-world situations, where task switches might occur at any moment. In such scenarios, there might not even be a clear distinction between tasks, but a gradual shift from one task to the next.
CF originates from a list of settings that are inherent in the common way of training DNNs. First of all, gradient descent forces a global optimization goal, which is replaced with every new task. This leads to a "**recency bias**" in the network, meaning that recent tasks and information are deemed more relevant and worthy to preserve. In relation to that, DNNs tend to have no meta-specialization in their neurons, using all neurons for the storage of goal related information (although there are exceptions, e.g. [5]). Therefore such networks have a "**relevance blindness**" in a sense that they are unable to dedicate portion of the network to relevant subtasks in a non-linear manner. In contrast, the human brain contains a variety of brain regions specialized in different tasks and can distribute relevant information through pathways to the respective areas. This in term results in a "**context invariance**", which makes it more difficult for DNNs to put information into a broader context and therefore efficiently reuse and sharpen previous knowledge even across task boundaries.
In this paper we will introduce a novel approach that learns a semantic task-embedding and is able to efficiently distribute and reuse features across the network based on gradual complexity and overlap of the current objective to previously learned features. In detail, the system learns a task embedding

either from a distinct input signal or derived from the main input. This embedding is fine tuned not only to differences in the task, but also to differences in inner-task complexity, allowing for more nuanced understanding of the task context. The system then uses this embedding to generate a context-based and layer specific attention mask applied to the weights to distribute concepts throughout the networks. We also introduce an auxillary loss function to make the attention mask sensitive to weight relevance to the current task as well as to previous objectives. This allows us to assess the relevance of information and train specialized network regions, while still efficiently reusing previously trained information. Furthermore, as irrelevant information from previous tasks remain largely untouched it allows us to bypass "recency bias" during learning. We believe that our solution provides a way to create task specialized neurons inside the network and leverages dynamic pathways to efficiently reuse information across fuzzy task boundaries. The code used for this research is available on github[1].

2 Related Work

Recent approaches toward CF tried to tackle these shortcomings. Regularization focused approaches [6, 7, 8] add additional constraints to the global optimization function to preserve relevant information. These constraints measure the relevance of weights w.r.t. previous tasks and limits the modification of previously relevant weights, thereby limit the information degradation. However, since the network still acts as a monolithic function, sequential learning adds noise to the system and degrades the recall of older tasks. Especially orthogonal tasks lead to contradicting neuron outputs.
Sparse-coding approaches [9, 10, 11] on the other hand allocate network capacity to build task-specific sub-networks, which are selected through task context. This allows for greater specialization in the network regions and less confusion due to competing neuron outputs. In the extreme case of non-overlapping sparse networks, these approaches generate n independently trained, sparse networks for each task. While some of the approaches allow for feature reuse between the sub-networks, this reuse is limited to a task level (as compared to a sample level). This can lead to sub-optimal performance in real-world scenarios where boundaries between tasks are often fuzzy and inputs within tasks might vary in complexity (requiring different feature overlaps). Our general concept of attention masks is similar to the one proposed by J. Serrà et al. [11], however the task embedding training and semantic meaning of the mask differs significantly as outlined in chapter 3.
Rehearsal and bayesian based approaches [12, 13] replay the information and objectives from previous tasks, thereby eliminating the "recency bias". Such approaches require to keep track of training information from previously learned tasks and inject this information into the training pipeline, which is not always applicable in real-world situations. Furthermore, the results show that these approaches still suffer from the lack of specialization inside the network for orthogonal tasks, leading to a diminished performance for all tasks [2]. Overall, recent studies have shown that these approaches only solve the problem in specific domains and do not provide a real-world applicable solution [14, 2]. Specifically, efficient reuse of previous information on large datasets and through many sequentially learned tasks still proves to be difficult.

3 Dynamic Weight Allocation

The human brain shows an incredible flexibility when it comes to learning diverse tasks. Dependent on the context and environment it activates different pathways and brain-areas to accomplish diverse tasks and leverage previously learned information even for unseen tasks. We implemented a DNN that is inspired by these biological processes and performs a context analysis during the sequential learning of tasks. In the following section we explain how this approach works and why we believe it will overcome the aforementioned problems of CF.
As in a default network, our system receives an input signal (e.g. image) and pushes that information through the first $j-1$ layers, generating an output tensor X_{j-1} (see figure 1). However, in order to evaluate the context of the input signal our network also requires a context signal. This signal can be given by the user directly (e.g. one-hot encoding of the task, *path A*) or computed directly from the input signal or from the output of the first k layers (i.e. stem layers, *path B*) through a context processor (see section 3.1). The context signal is then converted into task embedding through one layer. We introduce a variant of the triplet loss function [15] to introduce semantics to the

[1] https://github.com/felixnext/dwa

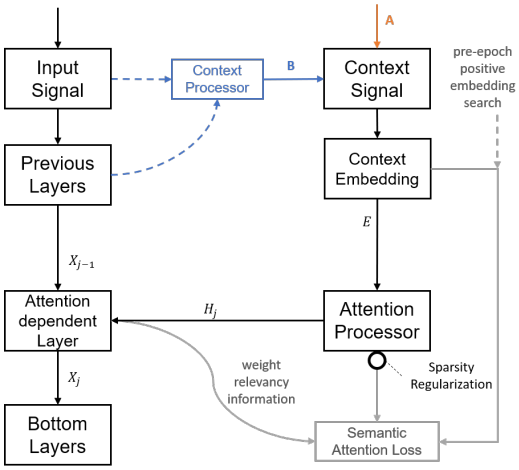

Figure 1: Overview of the general approach in this paper. The context signal in generated either from input or previous layer (a) or given as supervised value (b). The attention layers use an additional auxillary loss function to add semantics to the embeddings and attention masks. The gray paths are only relevant during training.

embedding space and allow for layer-wise continuous overlap of feature use across task boundaries (see section 3.3). From this embedding an attention mask is computed independently for every attention dependent layer in the network. This mask is then used to modulate the weight tensor in the original network layer (i.e. layer j) and thereby creating pathways through the network (see section 3.2). During the training a sparsity regulation is applied to the attention mask, to ensure an efficient use of network capacity (see section 3.3.3). Furthermore, we added a separate loss function for the attention mask based on the fisher information matrix diagonal [16, 6] to make the mask sensitive to "weight relevance" w.r.t. previous tasks.

3.1 Context Processor

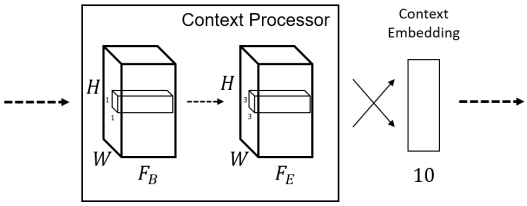

Figure 2: Layers in context processor. The signal is processed through a 1x1 bottleneck layer and expanded by a 3x3 convolution. Note that $F_B \ll F_E$. The actual embedding is then calculated through a fully connected layer.

The context signal should contain information relevant for the distinction of tasks. In case of an external calculated signal (i.e. path A in figure 1), this could be the average color of the input image, a one-hot vector of the current task or extraction of a dominant feature. However, systems in real-world applications might not have this information available in a supervised fashion (or it might be hard to estimate the correct feature to use).
We therefore created a second approach that allows extraction of the context information from the input signal (i.e. path B in figure 1). The context processor consists of a bottleneck layer, minimizing the number of feature-maps followed by an expansion layer (see figure 2). For our image related experiments we usually used $F_B = 32$ and $F_E = 128$. The bottleneck layer is used to map to a lower dimensional space in order to generalize information, making the resulting signal better suited to model the context.
In cases, where the context processor receives information from multiple stem layers with varying images sizes additional max-pooling layer are used and the feature maps are concatenated before insertion into the context processor.

3.2 Attention Mask Generation

After the context signal is retrieved, the system passes it to a layer-specific processor that computes the actual attention mask H_j. The attention output is used for weight modulation and has the same shape as the weight matrix in layer j.
Inside the attention processor the context embedding is converted into the attention mask through a fully connected layer as shown in figure 3. In our experiments, we applied this approach to two attention-dependent layer types: dense and convolutional.

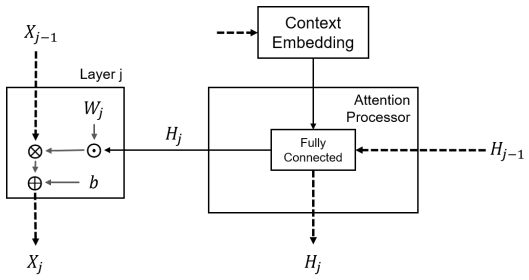

Figure 3: Attention modulation inside a specific layer. Later attention layers might use previous layers as input to stabilize the attention training.

Dense Layer
In terms of the dense layer, the weights are dropped out through an element-wise multiplication with the weight matrix W_j.

$$X_j = (W_j \cdot H_j) \times X_{j-1} + b_j \tag{1}$$

Where b_j is the bias and X_{j-1} the output from the previous layers. As $W_j \cdot H_j$ is simply an element-wise multiplication, the backward pass is straight forward.

Convolutional Layer
In the case of a convolutional layer the weight matrix is convolved over the input tensor.

$$X_j = g((W_j \cdot H_j), X^*_{j-1}, b) \tag{2}$$

Where X^*_{j-1} is the padded input tensor, W_j is the weight matrix of the kernel (with dimensions $d_h \times d_w$), H_j is the output of the attention mask for the current layer with the same dimensions and g the convolution operator.

During training both of these attention layer types are guided by specific loss function to embed additional semantics into the mask.

3.3 Semantic Attention Loss

During back-propagation through the main network (left side in figure 1) the attention acts as modulation, meaning it will limit the loss to the weights relevant to the attention mask. Therefore, the learning objective for the attention mask should be the selection of weights relevant to the current task, while avoiding weights that do not contribute or are relevant to previously learned tasks.
Shifts in attention can come from two different modulations in the input signal. On the one hand, differences between tasks (*inter-task semantics*), in cases where learning switches to a different task. As shown in figure 4, these differences can be overlapping (task A and B) or completely orthogonal (task A and C). In our approach we add an auxiliary triplet loss modulated by curriculum complexity at the level of the context embedding. On the other hand, attention can be modulated by the position of the attention-dependent layer inside the network (*inter-layer semantics*). Studies have shown that earlier layers in DNNs tend to learn more general features, which are also easier transferable to new tasks, while higher layers tend to specialize [17]. Consequently, we assumed that attention masks in earlier layers should have a higher overlap in relevant neurons even between more dissimilar tasks

than higher layers. To account for this insight, we introduce a semantic loss function that searches for weight relevance and then forces the attention. In particular the goal of this loss is to focus the attention mask to on weights that are relevant to the current task. Furthermore, we also account for relevance of the weights to previous tasks, in that we try to focus on weights that are irrelevant to previous tasks to avoid perturbation of learned information.

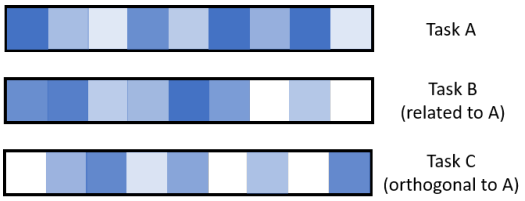

Figure 4: Example of the attention space. Strength of color indicates the relevance of a weight for the given task. Tasks that are closer related should have a stronger overlap in the weights, as weights from Task A might be useful for Task B as well. However, orthogonal tasks (i.e. A and C) should have minimal overlap in relevant weights to avoid forgetting.

3.3.1 Inter-Task Semantics

The objective of the inter-task semantic loss is the introduction of meaningful semantics into the context embedding (see figure 4). As the embedding is a vector, we adapted the triplet loss function [15] for context signals (see equation 4). The challenge hereby lies in the search of meaningful positive and negative anchors w.r.t. the context. In particular, we assumed that there can be a high variance in intra-task complexity, resulting in some tasks deviating from the core semantic meaning of the task. This in turn could lead to a fuzzy definition of task semantics. To solve this challenge, we introduce a variant of curriculum learning [18] into the training process, which assigns a complexity ζ to all training samples. This complexity can either be given by human label or (as we did) computed through a set of rules. We approximate the complexity of an image through mean entropy, but other approaches can be used as well (e.g. selective search [19] or color-variance). Furthermore, we normalize the complexity between 0 and 1 for each dataset. We then use the centroid embedding of the n training samples with the lowest complexity as positive anchor for the triplet loss. The negative anchor is generated by finding the orthogonal vector to the positive anchor with the highest loss for the main network. After the training of each task the positive vector is stored. In consecutive tasks the positive vector from a previous task with the highest loss (in the main loss function) is used as additional negative sample to incentivize spreading of vectors. Furthermore, to allow for the variance in complexity within a task, we relax the influence of the loss function by the converse complexity ζ.

$$L_{triplet}(E, P, N) = \Big[\|E - P\|_2^2 - \|E - N\|_2^2 + \alpha\Big]_+ \tag{3}$$

$$\mathcal{L}_{emb} = (1 - \zeta) * (L_{triplet}(E, P, N) + \delta * L_{triplet}(E, P, N_T)) \tag{4}$$

Where E is the output of the context embedding layer. P and N are the positive and negative anchors respectively. N_T is the positive anchor from a previous task with the highest loss. Finally, δ defines the impact of overlap penalty with previous vectors. In order to keep the reduce the computational cost, we compute the positive and negative anchors only at the beginning of each epoch.

3.3.2 Inter-Layer Semantics

We define the optimal attention mask for a specific task (or input for that matter) to optimize two attributes: relevance and plasticity. Relevance describes the contribution that each weight that is used by the attention mask provides to the correct output for the task at hand. Plasticity describes the contribution of each weight to previous tasks (i.e. tasks with lower contributions have a higher plasticity). In general, we want to include weights in the attention mask that have a high relevance as well as high plasticity.
In general, we assume that the Loss for the current Task is sufficient for the network to detect the relevance of weights. To aid this assumption we experimented with different warm-up techniques

during training to aid the learning process (see section 4.3). We needed, however, an additional loss function to make the attention processor sensitive to the plasticity of weights w.r.t. previously learned information. Following work done by [16, 6] we use the diagonal of the Fisher Information Matrix F as a measure for the plasticity of weights inside the network. The plasticity loss for the attention mask in layer j is thereby defined as:

$$\mathcal{L}_{pl,j}(\theta) = \sum_i F_i * [H_{j,i} * \eta]_+ \tag{5}$$

Where i is the index for the weights inside layer j. F_i is therefore the plasticity of the weight in the original network and $H_{j,i}$ is the value of the attention mask at the regarding position scaled by parameter η. The scaling allows us to influence the relevance of plasticity adjustments and we experimented with fixed values (e.g. 10) as well as dependencies on curriculum complexity.

3.3.3 Sparsity Regularization

To introduce sparsity into the network and create a consistent size of the sub-networks, we added an additional regularization constraint that controls the desired percentage of maximal active activations. We explicitly choose not to define a fixed sparsity through a selection function (e.g. top 10% of attention values) to allow the network to learn the capacity of the sub-networks used based on task complexity. The calculation of the regularization is derived from L1 regularization.

$$\mathcal{R}_{sp,j} = \left[\frac{1}{|H_j|} \sum_{i=0} (1|H_j(i) > 0) - \mu \right]_+ \tag{6}$$

Where H_j is the attention matrix generated and μ is a hyperparameter that defines the threshold for sparsity. In effect this punishes matrices that have more than μ non-zero elements.

3.3.4 Total Loss

Given these components, the total loss of the network is defined as:

$$\mathcal{L}(\theta) = \mathcal{L}_T(\theta) + \lambda_1 \mathcal{L}_{emb}(\theta) + \lambda_2 \sum_j \mathcal{L}_{pl,j}(\theta) + \lambda_3 \mathcal{R}_{sp,j} \tag{7}$$

Where T is the index for the current task and j is the index that iterates through all attention dependent layers. λ_{1-3} control the intesity with which the losses are applied (see section 4.3).

4 Experiments

As [2, 7] showed, reliable validation of continual learning approaches requires more elaborate scenarios than permuted MNIST datasets. Therefore, we used a setup of 8 common datasets that will be learned in a sequential fashion. We ran multiple experiments, testing various hyper-parameters including different topologies, sparsity and loss distributions. The results are compared using the forgetting ratio (FR) introduced in [11] as well as through total accuracy.
All tests were run on machines in the google cloud, using a Nvidia V100 GPU, 8 vCPUs and 52GB of RAM. The code is implemented in PyTorch 1.4 and based on the implementation from Serrà et al. [11]. Not mentioned hyper-parameters are set to PyTorch defaults.

4.1 Datasets

We used an ensemble of the following eight datasets, which were trained in sequence using different classification heads. The filtering started at a complexity of 0.2 and was ramped up through the number of epochs. All input data was scaled to a size of 32×32 with 3 channels, images with unequal sides are stretched. The datasets we used here are: MNIST [20], Fashion-MNIST [21], Traffic-Signs [22], facescrub [23], cifar 10 & 100 [24], NotMNIST [25] and SVHN [26]. If the original dataset has a train-test-split in place this is kept, otherwise the data is randomly split to use 15% for validation.

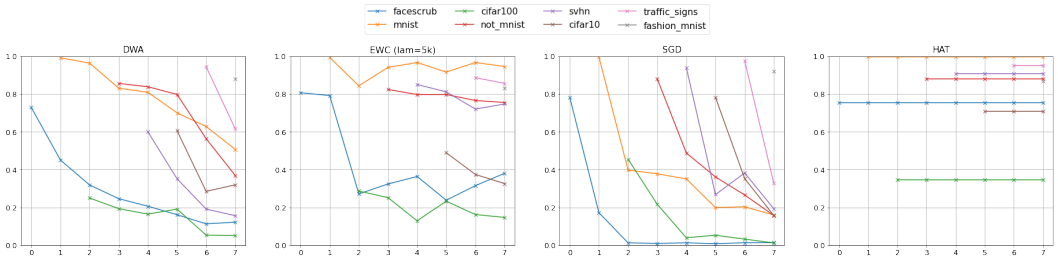

Figure 5: Comparison of accuracy after each learned task for different approaches

For all datasets, we apply a linear curriculum, where the complexity is increased from 0.2 to 1.0 in epoch 100.

4.2 Network Topologies

For the experiments we use a modified version of AlexNet [27]. This network contains a regular 4×4 convolution with 64 feature maps followed by 2×2 max-pooling. The output from this layer is then send to the context processor, where an embedding of size 100 is calculated. The remaining network consists of two attention-dependent convolution layers with 3×3 & 2×2 kernels and 128 & 256 feature maps respectively. Each is followed by a 2×2 max-pooling. Note that we do not use dropout here, as it would influence the attention mask. Next are two attention-dependent dense layers with 2048 outputs each. For the dense layers, we decided to compute the attention mask (with output size s) through a combination of two dense layers with size $\sqrt{s}$, to reduce parameters and performance costs. Classification is done through a regular dense layer with outputs each to the number of classes and a softmax activation. Due to the attention masks the network has a total of 28.8M parameters.
We use ReLU activations and initialize all convolution layers with the He normal initialization [28] and all dense layers with the Xavier uniform initialization [29],
We are currently running further experiments with other topologies (e.g. DenseNet [30]) and Task-Ensembles with higher complexity (e.g. CUB-200 [31]) and will make the results available once completed.

4.3 Training

For each experiment the training runs through 200 epochs with a batch-size of 32, a learning rate starting at 0.075 and an SGD optimizer. The learning rate is divided by 3 if the validation loss does not improve for 5 consecutive epochs. The training is stopped early if the learning rate falls below 0.0001.
During the training phase we ran a grid-search for diffferent hyper-parameters with reduced epoch counts (from 25 to 100). We found that a *tanh* activation gate for the attention processor shows better results compared to a *sigmoid* gate. We assume that this gate allows the network to better control positive or negative effects of the concepts from individual neurons (as described in [32]). For the parameters λ_{1-3} we choose $\lambda_1 = 20$ (to enforce embedding differentiation), $\lambda_2 = 0.1$ (to allow for plasticty) and $\lambda_3 = 500$ (to provide a strong incentive for sparsity). The sparsity μ is set to 0.2, α to 1.5 and δ to 1.
To provide time for the network to adjust the embeddings to the new task, we begin each task with a warmup period of 10 epochs, where the learning rate is divided by 750. We recompute the anchor vectors after every epoch.

4.4 Results

Figure 5 shows a direct comparison between our approach, vanilla SGD and the current state-of-the-art on accuracy after each learned task. The effect of CF on the baseline (*SGD*) is clearly visible. While our approach (*DWA*) provides a clear improvement over the baseline, it lacks in overall accuracy retention behind approaches

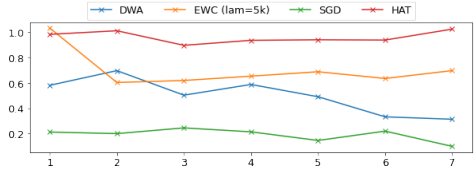

Figure 6: FR for tested approaches

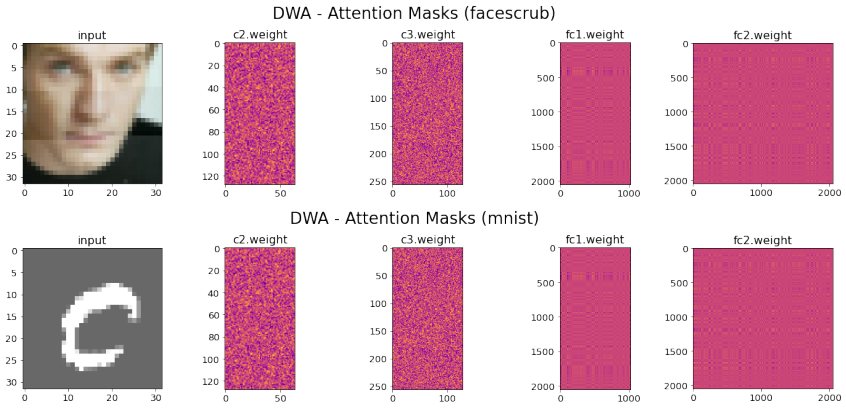

Figure 7: Example of attention masks for different tasks.

like *HAT* [11] and *EWC*. This is also confirmed by the forgetting ratio in Figure 6. When we look at the starting accuracy for each task, we notice a slight increase compared to *EWC* in tasks 5 and 6, indicating a slightly higher plasticity.
Our goal is to transfer these findings to more complex real-world datasets in the future, that have a higher semantic overlap. We believe that in these scenarios the dynamic allocation approach can provide better utilization of the network capacity.

4.5 Further Findings

To visualize the effects of attention masks, we generated visualizations (see Figure 7) that display the amount of attention generated. In this we can see a sparse pattern generated from the masks. While differences in the attention masks are visible, we can see that task overlapping is sub-optimal.
We assume that this results from the approximation of image complexity through entropy, which might be inferior to human labeling. The random generation of negative tasks also appears to not generate enough diversity in the attention masks to ensure proper task related learning. Therefore, we encourage further research in the automatic calculation of complexities to build curriculums, as this might not only aid this approach, but provide a more robust approach to learning in general.

5 Conclusion

In this paper we introduced DWA, a novel attention based mechanism for dynamic weight allocation in sequential learning environments. It leverages context features and curriculum learning to identify relevant weights to introduce sparsity to networks on the fly. For that we outlined a way to generate context embeddings and dynamically build per-layer attention masks that can be applied to the weight matrices. To aid the training, we provided multiple auxillary losses that differentiate the embeddings and masks as well as ensure sparsity of the attention masks.
While the results show that CF can be significantly reduced by our approach, it is also outperformed through existing solutions such as HAT or EWC. Based on the visualization of the weight-masks, we believe that the differentiation in embeddings through the auxillary loss in its current form is insufficient. Furthermore, we see potential for this approach in real-world settings and for larger network architectures (which were outside the scope of this work) to leverage the advantages of dynamic weight allocation. Future research will have to validate these claims.
Finally, we believe this line of research can lead to more general approach to DNNs.

Broader Impact

While our approach did not beat the current state-of-the-art on the provided research datasets, we found the current state still worthy sharing with the community. Our assumption is that attention masks can be used subsequently to drive more generalization in specific areas of a network, similar to regions in the human brain. In that sense, we can see how attention masks for networks can be used

to suppress biases and aim for better generalization as well as strengthen existing tendencies. For this work we did not explicitly account for biases or test them (especially as we leveraged standarized datasets with limited real-world impact).
To encourage reproducability and further research in this area, we will provide our trained models as well as all the code through github.

Acknowledgments and Disclosure of Funding

We would like to thank Niklas Kiehne and Prof. Wolf-Tilo Balke from the University of Braunschweig for their feedback through out our research.

References

[1] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[2] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[3] Earl K Miller. The prefontral cortex and cognitive control. *Nature reviews neuroscience*, 1(1):59, 2000.

[4] Joël Fagot and Robert G Cook. Evidence for large long-term memory capacities in baboons and pigeons and its implications for learning and the evolution of cognition. *Proceedings of the National Academy of Sciences*, 103(46):17564–17567, 2006.

[5] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

[6] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[7] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in neural information processing systems*, pages 4652–4662, 2017.

[8] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[9] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[10] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

[11] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

[12] Alexander Gepperth and Cem Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, 2016.

[13] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[14] Benedikt Pfülb, Alexander Gepperth, S Abdullah, and A Kilian. Catastrophic forgetting: still a problem for dnns. In *International Conference on Artificial Neural Networks*, pages 487–497. Springer, 2018.

[15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[16] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[17] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[18] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[19] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[20] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[21] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[22] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[23] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*, pages 343–347. IEEE, 2014.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[25] Yaroslav Bulatov. Notmnist dataset. Technical report, 2011.

[26] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

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

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[29] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[30] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[31] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[32] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.