The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Measuring Catastrophic Forgetting in Neural Networks

Ronald Kemker,[1] Marc McClure,[1] Angelina Abitino,[2] Tyler L. Hayes,[1] Christopher Kanan[1]
[1]Rochester Institute of Technology [2]Swarthmore College
{rmk6217, mcm5756}@rit.edu , aabitin1@swarthmore.edu, {tlh6792, kanan}@rit.edu

Abstract

Deep neural networks are used in many state-of-the-art systems for machine perception. Once a network is trained to do a specific task, e.g., bird classification, it cannot easily be trained to do new tasks, e.g., incrementally learning to recognize additional bird species or learning an entirely different task such as flower recognition. When new tasks are added, typical deep neural networks are prone to catastrophically forgetting previous tasks. Networks that are capable of assimilating new information incrementally, much like how humans form new memories over time, will be more efficient than retraining the model from scratch each time a new task needs to be learned. There have been multiple attempts to develop schemes that mitigate catastrophic forgetting, but these methods have not been directly compared, the tests used to evaluate them vary considerably, and these methods have only been evaluated on small-scale problems (e.g., MNIST). In this paper, we introduce new metrics and benchmarks for directly comparing five different mechanisms designed to mitigate catastrophic forgetting in neural networks: regularization, ensembling, rehearsal, dual-memory, and sparse-coding. Our experiments on real-world images and sounds show that the mechanism(s) that are critical for optimal performance vary based on the incremental training paradigm and type of data being used, but they all demonstrate that the catastrophic forgetting problem is not yet solved.

Introduction

While the basic architecture and training algorithms behind deep neural networks (DNNs) are over 30 years old, interest in them has never been greater in both industry and the artificial intelligence research community. Owing to far larger datasets, increases in computational power, and innovations in activation functions, DNNs have achieved near-human or super-human abilities on a number of problems, including image classification (He et al. 2016), speech-to-text (Khilari and Bhope 2015), and face identification (Schroff, Kalenichenko, and Philbin 2015). These algorithms power most of the recent advances in semantic segmentation (Long, Shelhamer, and Darrell 2015), visual question answering (Kafle and Kanan 2017), and reinforcement learning (Mnih et al. 2013). While these systems have become more capable, the standard multi-layer perceptron (MLP) architecture and typical training algorithms cannot handle incrementally learning new tasks or categories without catastrophically forgetting previously learned training data. Fixing this problem is critical to making agents that incrementally improve after deployment. For non-embedded or personalized systems, catastrophic forgetting is often overcome simply by storing new training examples and then re-training either the entire network from scratch or possibly only the last few layers. In both cases, retraining uses *both* the previously learned examples and the new examples, randomly shuffling them so that they are independent and identically distributed (iid). Retraining can be slow, especially if a dataset has millions or billions of instances.

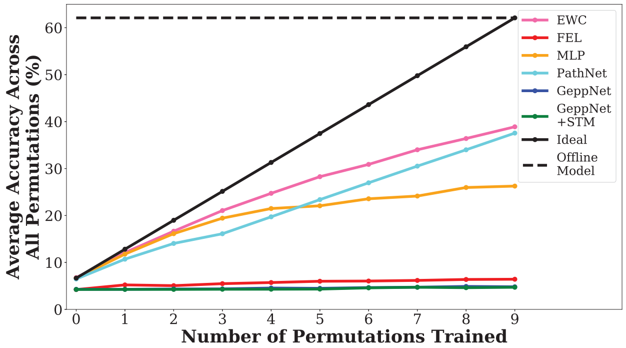


Figure 1: Catastrophic forgetting impairs incremental learning in neural networks. As a network is incrementally trained (solid lines), ideally its performance would match that of a model trained offline with all of the data upfront (dashed line). In this paper, we develop methods and benchmarks for measuring catastrophic forgetting. Our experiments show that even methods designed to prevent catastrophic forgetting perform significantly worse than an offline model. Incremental learning is key to many real-world applications because it allows the model to adapt after being deployed.

Catastrophic forgetting was first recognized in MLPs almost 30 years ago (McCloskey and Cohen 1989). Since then, there have been multiple attempts to mitigate this phenomenon (Hinton and Plaut 1987; Robins 1995; Goodrich and Arel 2014; Draelos et al. 2016; Ren et al. 2017; Fernando et al. 2017; Kirkpatrick et al. 2017). However, these methods have not been systematically compared, and most have only been evaluated on easy datasets, e.g., MNIST. It is not clear if these methods will scale to larger datasets containing hundreds of categories. In this paper, we provide a comprehensive empirical review of methods to mitigate catastrophic forgetting using a variety of new metrics. While catastrophic forgetting occurs in unsupervised frameworks (Draelos et al. 2016; Goodrich and Arel 2014; Triki et al. 2017), we focus on supervised classification.

Our major contributions are:

- We demonstrate that despite popular claims (Kirkpatrick et al. 2017), catastrophic forgetting is not solved.
- We establish new benchmarks with novel metrics for measuring catastrophic forgetting. Previous work has focused on MNIST, which contains low-resolution images and only 10 classes. Instead, we use real-world image/audio classification datasets containing 100-200 classes. We show that, although existing models perform well on MNIST for a variety of different incremental learning problems, performance drops significantly with more challenging datasets.
- We identify five common mechanisms for mitigating catastrophic forgetting: 1) regularization, 2) ensembling, 3) rehearsal, 4) dual-memory models, and 5) sparse-coding. We directly compare these distinct approaches.

Problem Formulation

In this paper, we study catastrophic forgetting in MLP-based neural networks that are incrementally trained for classification tasks. In our setup, the labeled training dataset D is organized into T study sessions (batches), i.e., $D = \{B_t\}_{t=1}^{T}$. Each study session B_t consists of N_t labeled training data points, i.e., $B_t = \left\{(\mathbf{x}_j, y_j)\right\}_{j=1}^{N_t}$, where $\mathbf{x}_j \in \mathbb{R}^d$ and y_j is a discrete label. N_t is variable across sessions. The model is only permitted to learn sessions sequentially, in order. At time t the network can only learn from study session B_t; however, models may use auxiliary memory to store previously observed sessions, but this memory use must be reported. We do not assume sessions are iid, e.g., some sessions may contain data from only a single category. In between sessions, the model may be evaluated on test data. Because this paper's focus is catastrophic forgetting, we focus less on representation learning and obtain feature vectors using embeddings from pre-trained networks. Note that in some other papers, new sessions are called new 'tasks.' We refer to the first study session as the model's 'base set knowledge.'

Why Does Catastrophic Forgetting Occur?

Catastrophic forgetting in neural networks occurs because of the stability-plasticity dilemma (Abraham and Robins 2005). The network requires sufficient plasticity to acquire new tasks, but large weight changes will cause forgetting by disrupting previously learned representations. Keeping the network's weights stable prevents previously learned tasks from being forgotten, but too much stability prevents the model from learning new tasks. Prior research has tried to solve this problem using two broad approaches. The first is to try to keep new and old representations separate, which can be done using distributed models, regularization, and ensembling. The second is to prevent the forgetting of prior knowledge simply by training on the old tasks (or some facsimile of them) as well as new tasks, thereby preventing the old tasks from being forgotten. Besides requiring costly relearning of previous examples and additional storage, this scheme is still not as effective as simply combining the new data with the old data and completely re-training the model from scratch. This solution is inefficient as it prevents the development of deployable systems that are capable of learning new tasks over the course of their lifetime.

Previous Surveys

French (1999) exhaustively reviewed mechanisms for preventing catastrophic forgetting that were explored in the 1980s and 1990s. Goodfellow et al. (2013) compared different activation functions and learning algorithms to see how they affected catastrophic forgetting, but these methods were not explicitly designed to mitigate catastrophic forgetting. The authors concluded that the learning algorithms have a larger impact, which is what we focus on in this paper. They sequentially trained a network on two separate tasks using three different scenarios: 1) identical tasks with different forms of input, 2) similar tasks, and 3) dissimilar tasks. We adopt a similar paradigm, but our experiments involve a much larger number of tasks. We also focus on methods explicitly designed to mitigate catastrophic forgetting.

Soltoggio, Stanley, and Risi (2017) reviewed neural networks that can adapt their plasticity over time, which they called Evolved Plastic Artificial Neural Networks. Their review covered a wide range of brain-inspired algorithms and also noted that the field lacks appropriate benchmarks. However, they did not conduct any experiments or establish benchmarks for measuring catastrophic forgetting. We remedy this gap in the literature by establishing metrics and large-scale benchmarks for measuring catastrophic forgetting in neural networks.

Mitigating Catastrophic Forgetting

While not exhaustive, we have identified five main approaches that have been pursued for mitigating catastrophic forgetting in MLP-like architectures, which we describe in the next subsections. We describe the models we will compare in greater detail in the Experimental Setup section.

Regularization Methods

Regularization methods add constraints to the network's weight updates, so that when a new session is learned, it is less likely to interfere with previously learned knowledge. Hinton and Plaut (1987) implemented a network that had both 'fast' and 'slow' training weights. The fast weights had high plasticity and were easily affected by changes to the network, and the 'slow' weights had high stability and were harder to adapt. This kind of dual-weight architecture is similar in idea to dual-network models, but has not been shown to be sufficiently powerful to learn a large number of new tasks. Elastic weight consolidation (EWC) (Kirkpatrick et al. 2017) adds a constraint to the loss function that directs plasticity away from weights that contribute the most to previous tasks. We use EWC to evaluate the regularization mechanism.

Ensemble Methods

Ensemble methods attempt to mitigate catastrophic forgetting either by explicitly or implicitly training multiple classifiers together and then combining them to generate the final prediction. For the explicit methods, such as Learn++ and TradaBoost, this prevents forgetting because an entirely new sub-network is trained for a new session (Polikar et al. 2001; Dai et al. 2007). However, memory usage will scale with the number of sessions, which is highly undesirable. Moreover, this prevents portions of the network from being re-used for the new session. Two methods that try to alleviate the memory usage problem are Accuracy Weighted Ensembles and Life-long Machine Learning (Wang et al. 2003; Ren et al. 2017). These methods automatically decide whether a sub-network should be removed or added to the ensemble.

PathNet can be considered as an implicit ensemble method (Fernando et al. 2017). It uses a genetic algorithm to find an optimal path through a fixed-size neural network for each study session. The weights in this path are then frozen, so that when new sessions are learned, earlier sessions are not forgotten. In contrast to the explicit ensembles, the base network's size is fixed and it is possible for learned representations to be re-used which allows for smaller, more deployable models. The authors showed that PathNet learned subsequent tasks more quickly, but not how well earlier tasks were retained. We have selected PathNet to evaluate the ensembling mechanism, and we measure its retention of earlier tasks.

Rehearsal Methods

Rehearsal methods try to mitigate catastrophic forgetting by mixing data from earlier sessions with the current session being learned (Robins 1995). The cost is that this requires storing past data, which is not resource efficient. Pseudorehearsal methods use the network to generate pseudopatterns (Robins 1995) that are combined with the session currently being learned. Pseudopatterns allow the network to stabilize older memories without storing all previously observed training data points. Draelos et al. (2016) used this approach to incrementally train an autoencoder, where each session contained images from a specific category. After the autoencoder learned a particular session, they passed the session's data through the encoder and stored the output statistics. During replay, they used these statistics and the decoder network to generate the appropriate pseudopatterns for each class.

The model proposed by Gepperth and Karaoguz (2016) reserves its training data to replay after each new class was trained. We refer to this model as GeppNet. GeppNet used a self-organizing map (SOM) as a hidden layer to topologically reorganize the data from the input layer, i.e., clustering the input onto a 2-D lattice. We use GeppNet to explore the value of rehearsal.

Dual-Memory Models

Dual-memory models are inspired by memory consolidation in the mammalian brain, which is thought to store memories in two distinct neural networks. Newly formed memories are stored in a brain region known as the hippocampus. These memories are then slowly transferred/consolidated to the pre-frontal cortex during sleep. Several algorithms based on these ideas have been created. Early work used fast (hippocampal) and slow (cortical) training networks to separate pattern-processing areas, and they passed pseudopatterns back and forth to consolidate recent and remote memories (French 1997). In general, dual-memory models incorporate rehearsal, but not all rehearsal-based models are dual-memory models.

Another model proposed by Gepperth and Karaoguz (2016), which we denote GeppNet+STM, stores new inputs that yield a highly uncertain prediction into a short-term memory (STM) buffer. This model then seeks to consolidate the new memories into the entire network during a separate sleep phase. They showed that GeppNet+STM could incrementally learn MNIST classes without forgetting previously trained ones. We use GeppNet and GeppNet+STM to evaluate the dual-memory approach.

Sparse-Coding Methods

Catastrophic forgetting occurs when new internal representations interfere with previously learned ones (French 1999). Sparse representations can reduce the chance of this interference; however, sparsity can impair generalization and ability to learn new tasks (Sharkey and Sharkey 1995).

Two models that implicitly use sparsity are CALM and ALCOVE. To learn new data, CALM searches among competing nodes to see which nodes have not been committed to another representation (Murre 2014). ALCOVE is a shallow neural network that uses a sparse distance-based representation, which allows the weights assigned to older tasks to be largely unchanged when the network is presented with new data (Kruschke 1992). The Sparse Distributed Memory (SDM) is a convolution-correlation model that uses sparsity to reduce the overlap between internal representations (Kanerva 1988). CHARM and TODAM are also convolution-correlation models that use internal codings to ensure that new input representations remain orthogonal to one another (Murdock 1983; Eich 1982).

The Fixed Expansion Layer (FEL) model creates sparse representations by fixing the network's weights and specifying neuron triggering conditions (Coop, Mishtal, and Arel 2013). FEL uses excitatory and inhibitory fixed weights to sparsify the input, which gates the weight updates throughout the network. This enables the network to retain prior learned mappings and reduce representational overlap. We use FEL to evaluate the sparsity mechanism.

Experimental Setup

We explore how well methods to mitigate catastrophic forgetting scale on hard datasets involving fine-grained image and audio classification. These datasets were chosen because they contain 1) different data modalities (image and

	MNIST	CUB-200	AudioSet
Classification Task	Gray Image	RGB Image	Audio
Classes	10	200	100
Feature Shape	784	2,048	1,280
Train Samples	50,000	5,994	28,779
Test Samples	10,000	5,794	5,523
Train Samples/Class	5,421-6,742	29-30	250-300
Test Samples/Class	892-1,135	11-30	43-62

Table 1: Dataset Specifications

audio), 2) a large number of classes, and 3) a small number of samples per class. These datasets are more meaningful (real-world problems) and more practical than MNIST. We also use MNIST to showcase the value of these real-world datasets. See Table 1 for dataset statistics.

Dataset Description

MNIST MNIST is a classic dataset in machine learning containing 10 digit classes. Its grayscale images are 28×28.

CUB-200 Caltech-UCSD Birds-200 (CUB-200) is an image classification dataset containing 200 different bird species (Wah et al. 2011). We use the 2011 version. Each high-resolution image is turned into a 2048-dimensional vector with ResNet-50 (He et al. 2016), which is a deep convolutional neural network (DCNN) pre-trained on ImageNet (Russakovsky et al. 2015). Extracting image features from the last hidden (fully-connected) layer of pre-trained DCNNs is a common practice in computer vision. We report mean-per-class accuracy, which is the CUB-200 standard.

AudioSet AudioSet (Gemmeke et al. 2017) is a hierarchically organized audio classification dataset built from YouTube videos. It has over 2 million human-labeled, 10 second sound bytes drawn from one or more of 632 classes. We used the pre-extracted frame-wise features from AudioSet concatenated in order. These features were extracted using a variant ResNet-50 for audio data (Hershey et al. 2017), which was pre-trained on an early version of the YouTube-8m dataset (Abu-El-Haija et al. 2016). We used 100 classes from AudioSet, none of which were super- or sub-classes of each other. The classes did not have any restrictions based on the AudioSet ontology, and all of them had a quality estimation of over 70%. Each audio sample can have multiple labels, so we chose training and testing samples that were labeled with only 1 of the 100 classes.

Models Evaluated

We evaluated five models that correspond to each of the five mechanisms described in the previous section: 1) EWC, 2) PathNet, 3) GeppNet, 4) GeppNet+STM, and 5) FEL. To choose the number of parameters to use across models, we established a baseline MLP architecture that performed well for CUB-200 and AudioSet when trained offline. The goal is to determine which mechanism(s) work best for various incremental learning paradigms. To provide a fair comparison, the number of parameters in each model were chosen to be as close as possible to the number of parameters in the baseline MLP. We optimized each model's hyperparameters to work well for our benchmarks, which are given in supplemental materials[1]. The supplemental materials provides the stopping criteria for each model as defined by their creators, which involved 1) training for a fixed period of time or 2) using test accuracy to stop training early.

Standard Multi-Layer Perceptron For our baseline, we use a standard MLP. Its architecture was chosen by optimizing performance using the entire training set for both CUB-200 and AudioSet, i.e., it was trained offline. The offline MLP achieves 62.1% accuracy on the CUB-200 test set and 46.1% on the AudioSet test set. We did a hyperparameter search for the number of units per hidden layer (32-4,096), number of hidden layers (2-3), and weight decay parameter (0, 10^{-4}, $5 \cdot 10^{-4}$). The MLP model was also trained incrementally to measure the severity of catastrophic forgetting.

Elastic Weight Consolidation EWC adds an additional constraint to the loss function $L(\theta)$, i.e.,

$$L(\theta) = L_t(\theta) + \sum_i \frac{\lambda}{2} F_i \left(\theta_i - \theta^*_{A,i}\right)^2, \tag{1}$$

where $L(\theta)$ is the combined loss function, θ is the network's parameters, $L_t(\theta)$ is the loss for session B_t, λ is a hyperparameter that indicates how important the old task(s) are compared to the new task, F is the Fisher information matrix, and θ^*_A are the trainable parameters (weights and biases) important to previously trained tasks. The Fisher matrix is used to constrain the weights important to previously learned tasks to their original value; that is, plasticity is directed to the trainable parameters that contribute the least to performing previously trained tasks. The size of the hidden layer was chosen to match the baseline MLP's capacity.

PathNet PathNet is a fixed size neural network that uses a genetic algorithm to find the optimal path through the network. Only this path is trainable when learning a particular session, which is why the authors described their model as an evolutionary dropout network. PathNet creates an independent output layer for each task in order to preserve previously trained tasks, and it cannot be used without modifications for incremental class learning. Since entire portions of the network are sequentially frozen as new tasks are learned, there is a risk of PathNet losing its ability to learn once the maximum capacity is reached. PathNet's capacity was chosen to match the capacity of the MLP baseline.

GeppNet GeppNet and GeppNet+STM are biologically-inspired approaches that use rehearsal to mitigate forgetting. In these models, training the initial task starts by initializing the SOM-layer, which is used to project the probability density of the input to a higher two-dimensional lattice. The SOM-layer features are passed to a linear regression classification layer to make a prediction. During training, the SOM-layer is initialized with the first session for a fixed-period of time, and then the SOM- and classification-layers are trained jointly. The SOM-layer is only updated when a

[1] Supplemental materials are provided at the end of our arXiv submission: https://arxiv.org/abs/1708.02072

training example is determined by the model to be novel, i.e., using the prediction probabilities to generate a confidence measure. After GeppNet has been trained on the initial session for a fixed period of time, it incrementally learns subsequent sessions. GeppNet performs updates to the SOM-layer and classification-layer when a training example is considered novel. When GeppNet+STM detects novelty, it instead uses a fixed-size short-term memory buffer to store that training example, which then replays it during a sleep phase. The sleep phase repeats after a fixed number of training iterations. Since the replay queue has a fixed-size (i.e., older examples are replaced), the GeppNet+STM model will train more efficiently than GeppNet. GeppNet stores all previous training data and replays it along with the previous data during a portion of its incremental learning step. GeppNet+STM also stores all previous and new training data; however, each training example is only replayed if the model is uncertain on the prediction. In addition, GeppNet+STM is capable of making real-time predictions by determining if the desired memory is in short-term memory (the memory buffer) or in long-term storage (the SOM- and classification-layers).

Fixed Expansion Layer FEL uses sparsity to mitigate catastrophic forgetting (Coop, Mishtal, and Arel 2013). FEL is a two hidden layer MLP where the second hidden layer (FEL-layer) has a higher capacity than the first fully-connected layer, but the weights are sparse and remain fixed through training. Each FEL-layer unit is only connected to a subset of the units in the first hidden layer, and these connections are split between excitatory and inhibitory weights. Only a subset of the FEL-layer units are allowed to have non-zero output to the final classification layer, which causes only some of the units in the first hidden layer to be updated.

Experiments and Results

We have established three benchmark experiments for measuring catastrophic forgetting:

1. **Data Permutation Experiment** - The elements of every feature vector are randomly permuted, with the permutation held constant within a session, but varying across sessions. The model is evaluated on its ability to recall data learned in prior study sessions. Each session contains the same number of examples.
2. **Incremental Class Learning** - After learning the base set, each new session learned contains only a single class.
3. **Multi-Modal Learning** - The model incrementally learns different datasets, e.g., learn image classification and then audio classification.

For the data permutation and incremental class learning experiments, each model was also evaluated on MNIST. The goal is to examine whether results on MNIST generalize to the real-world datasets. More results, including detailed plots, can be found in the supplementary materials.

Evaluation Metrics

We propose three new metrics to evaluate a model's ability to retain prior sessions while still learning new knowledge,

$$\Omega_{base} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{base,i}}{\alpha_{ideal}} \tag{2}$$

$$\Omega_{new} = \frac{1}{T-1}\sum_{i=2}^{T}\alpha_{new,i} \tag{3}$$

$$\Omega_{all} = \frac{1}{T-1}\sum_{i=2}^{T}\frac{\alpha_{all,i}}{\alpha_{ideal}} \tag{4}$$

where T is the total number of sessions, $\alpha_{new,i}$ is the test accuracy for session i immediately after it is learned, $\alpha_{base,i}$ is the test accuracy on the first session (base set) after i new sessions have been learned, $\alpha_{all,i}$ is the test accuracy of all of the test data for the classes seen to this point, and α_{ideal} is the offline MLP accuracy on the base set, which we assume is the ideal performance. Ω_{base}, Ω_{new}, and Ω_{all} are normalized area under the curve metrics. Ω_{base} measures a model's retention of the first session, after learning in later study sessions. Ω_{new} measures the model's ability to immediately recall new tasks. Ω_{all} computes how well a model both retains prior knowledge and acquires new information. By normalizing Ω_{base} and Ω_{all} by α_{ideal}, the results will be easier to compare between datasets. Unless a model exceeds α_{ideal}, results will be between $[0,1]$, which enables comparison between datasets.

Experimental Results

Data Permutation Experiment This experiment evaluates a model's ability to retain multiple representations of the dataset, with each representation learned sequentially. These representations are created by randomly permuting the elements of the input feature vectors, with the random permutation changing between sessions. An identically permuted test set is used along with each session. This paradigm provides overlapping tasks in which each session contains the same information and categories, so each session is of equal complexity. This paradigm is identical to that used by Goodfellow et al. (2013) and Kirkpatrick et al. (2017).

Results are given in Table 2. In nearly every case, Ω_{all} is greater for MNIST than on CUB-200 or AudioSet, demonstrating the need for alternative incremental learning benchmarks. To some extent, EWC, PathNet, GeppNet, and GeppNet+STM retain prior knowledge without forgetting; however, GeppNet and GeppNet+STM fail to learn new sessions. PathNet and EWC seem to retain base knowledge while still learning new information; however, PathNet performs better on AudioSet and MNIST, while EWC performs better on CUB-200 (see discussion).

Incremental Class Learning In the incremental class learning experiment, a model's ability to sequentially learn new classes is tested. The first session learned contains training data from half of the classes in each dataset: 5 for MNIST, 100 for CUB-200, and 50 for AudioSet. Once this base set was learned, each subsequent session contained training data from a single new class. We measure mean-per-class test accuracy on the base set after each new class is learned to assess a model's long-term memory. We also calculate the accuracy of each class after it is trained to ensure

Model	Dataset	Data Permutation			Incremental Class			Multi-Modal			Memory Constraints	Model Size (MB)
		Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}		
MLP	**MNIST**	0.434	0.996	0.702	0.060	1.000	0.181	N/A	N/A	N/A	Fixed-size	1.91
	CUB	0.488	0.917	0.635	0.020	1.000	0.031	0.327	0.412	0.610		4.24
	AS	0.186	0.957	0.446	0.016	1.000	0.044	0.197	0.609	0.589		2.85
EWC	**MNIST**	0.437	0.992	0.746	0.001	1.000	0.133	N/A	N/A	N/A	Fixed-size	3.83
	CUB	0.765	0.869	0.762	0.105	0.000	0.094	0.944	0.369	0.872		8.48
	AS	0.129	0.687	0.251	0.021	0.580	0.034	1.000	0.588	0.984		5.70
PathNet	**MNIST**	0.687	0.887	0.848	N/A	N/A	N/A	N/A	N/A	N/A	New output layer for each task	2.80
	CUB	0.538	0.701	0.655	N/A	N/A	N/A	0.908	0.376	0.862		7.46
	AS	0.414	0.750	0.615	N/A	N/A	N/A	0.069	0.540	0.469		4.68
GeppNet	**MNIST**	0.912	0.242	0.364	0.960	0.824	0.922	N/A	N/A	N/A	Stores all training data	190.08
	CUB	0.606	0.029	0.145	0.628	0.640	0.585	0.156	0.010	0.089		53.48
	AS	0.897	0.170	0.343	0.984	0.458	0.947	0.913	0.005	0.461		150.38
GeppNet+STM	**MNIST**	0.892	0.212	0.326	0.919	0.599	0.824	N/A	N/A	N/A	Stores all training data	191.02
	CUB	0.615	0.020	0.142	0.727	0.232	0.626	0.031	0.329	0.026		55.94
	AS	0.820	0.041	0.219	1.007	0.355	0.920	0.829	0.005	0.418		151.92
FEL	**MNIST**	0.117	0.990	0.279	0.451	1.000	0.439	N/A	N/A	N/A	Fixed-size	4.54
	CUB	0.043	0.764	0.184	0.316	1.000	0.361	0.110	0.329	0.412		6.16
	AS	0.081	0.848	0.239	0.283	1.000	0.240	0.473	0.320	0.494		6.06

Table 2: Results on MNIST, CUB-200 (CUB), and AudioSet (AS) for our evaluation metrics as well as model size (in MB) for each model/dataset combination.

the model is still learning, and we calculate the performance of all previously learned classes.

PathNet is incapable of learning new classes incrementally because it creates a separate output layer for each additional session. Accessing that output layer during prediction time requires a priori information on which session the model needs to access. This means PathNet requires the testing label to make the appropriate prediction, which would result in a misleading high test accuracy. For this reason, we omitted PathNet from this experiment.

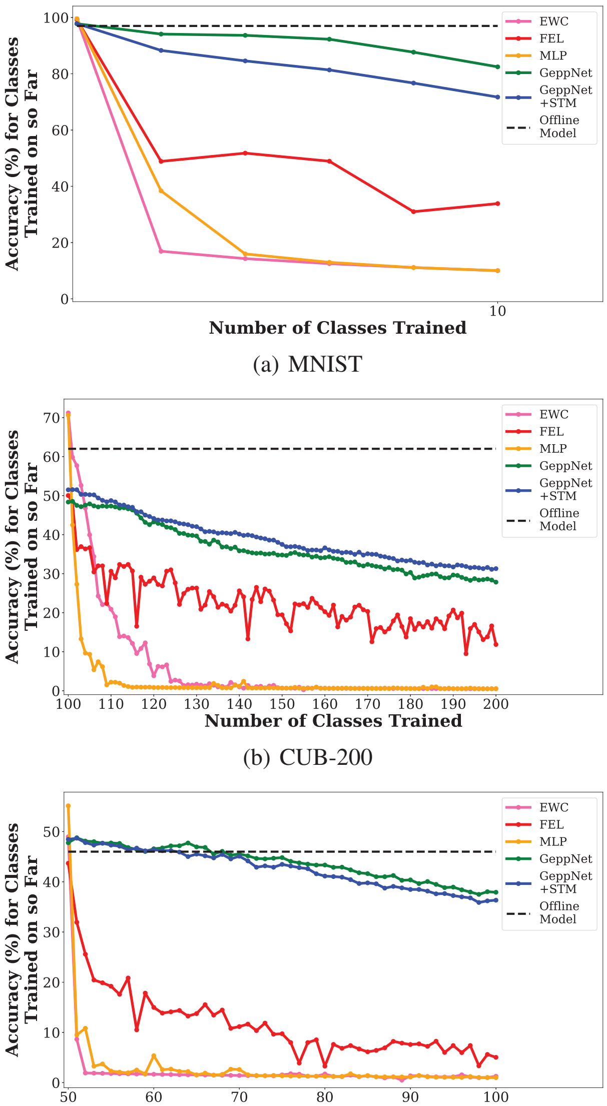
Results are summarized in Table 2 and Fig. 2 contains plots for the mean-per-class test accuracy for all classes seen so far. The only models that were able to both retain the base knowledge and learn new classes were GeppNet, GeppNet+STM, and FEL, with the clear winner being the two GeppNet variants. Much like the data permutation experiment, the CUB-200 and AudioSet results were noticeably lower than the MNIST results. GeppNet+STM did well at retaining the base set, but it struggled to learn new classes on CUB-200 and AudioSet. This could be because the model only trains during sleep for efficiency reasons. Additionally, the short-term memory buffer is emptied after training each study session, which is when the model is evaluated. This type of model could work better in a real-time environment. FEL learned new classes well, but suffered from forgetting of the base set. FEL may benefit from larger model capacity, but this would require more memory/processing power.

Multi-Modal Experiment The goal of the multi-modal experiment is to determine if a network can learn and retain multiple dissimilar tasks that have 1) inputs with different dimensionality and feature distributions and 2) a different number of classes. A system like this could be useful for learning tasks that have multi-modal data using a single network and could be more efficient than building a separate neural network for each modality (e.g., video has visual and audio information). In this experiment, we evaluated each model's ability to perform image and audio classification with CUB-200 and AudioSet respectively. In this experiment, there are only two incrementally learned sessions, where each session contains AudioSet or CUB-200. We compare learning CUB-200 first then AudioSet as well as learning AudioSet followed by learning CUB-200. The ResNet features obtained from CUB-200 have a higher dimensionality than the AudioSet features, so we zero-padded the AudioSet input to match the dimensionality of CUB-200. This experiment is done by training one dataset to completion followed by training the other dataset to completion (and vice-versa). Once both modalities have been trained, we evaluate the first modality that was trained in order to measure how well the model was able to retain what it learned about the first task.

Table 2 shows summary results for the multi-modal experiment, where the corresponding row is the modality that was trained first, i.e., the row for CUB-200 is where CUB-200 is learned first followed by AudioSet. Additional results are in supplementary materials. Although several models perform well at one of the two experiments, EWC is the only model capable of preserving the first modality while also learning the second modality for both cases, which we explore further in the discussion.

Discussion

In our paper we introduced new metrics and benchmarks for measuring catastrophic forgetting. Our results reveal that none of the methods we tested solve catastrophic forgetting,



(a) MNIST

(b) CUB-200

(c) AudioSet

Figure 2: Results for the incremental class learning experiment. The offline model line denotes the performance of the offline model on the entire dataset. This shows the mean-class test accuracy for all classes seen so far.

while also enabling the learning of new information. Table 3 summarizes these results for each of our experiments by averaging Ω_{all} over all datasets. While no method excels at incremental learning, some perform better than others.

PathNet performed best overall on the data permutation experiments, with the exception of CUB-200. However, PathNet requires being told which session each test instance is from, whereas the other models do not use this information. This may give it an unfair advantage. PathNet works by locking the optimal path for a given session. Because permuting the data does not reduce feature overlap, the model requires more trainable weights (less feature sharing) to build a discriminative model, causing PathNet to saturate (freeze all weights) more quickly. When PathNet reaches the saturation point, the only trainable parameters are in the output layer. While EWC was the second best performing method in the permutation experiments, it only redirects plasticity instead of freezing trainable weights.

Model	Data Permutation	Incremental Class	Multi-Modal
MLP	0.594	0.085	0.600
EWC	0.586	0.087	**0.913**
PathNet	**0.706**	N/A	0.666
GeppNet	0.284	**0.818**	0.275
GeppNet+STM	0.229	0.790	0.222
FEL	0.234	0.347	0.453

Table 3: Summary of Experimental Results. Average of Ω_{all} over MNIST, CUB-200, and AudioSet results.

Both GeppNet variants performed best at incremental class learning. These models make slow, gradual changes to the network that are inspired by memory consolidation during sleep. For these models, the SOM-layer was fixed to 23×23 to have the same number of trainable parameters as the other models. With 100-200 classes, this corresponds to 2-5 hidden layer neurons per class respectively. The experiments on MNIST in Gepperth and Karaoguz (2016) used 90 hidden layer neurons per class, so their performance may improve if their model capacity was significantly increased, but this would demand more memory and computation.

EWC performed best on the multi-modal experiment. This may be because features between the two modalities are non-redundant. We hypothesize that EWC is a better choice for separating non-redundant data and PathNet may work well when working with data that has different, but not entirely dissimilar, representations. To explore this, we used the Fast Correlation Based Filter proposed by Yu and Liu (2003) to show the features in MNIST and AudioSet are more redundant than those in CUB-200 (see supplemental material). The performance of EWC and PathNet for both the data permutation and multi-modal experiments are consistent with this hypothesis.

Table 2 shows the memory constraints and usage of each model. While we kept the number of trainable parameters roughly the same across all models in their hidden layers, some require additional memory resources. PathNet generates a new output layer for each session. Both GeppNet variants store all training data and rehearse over it during their incremental learning stage. The creators of EWC stored validation data from all previous sessions and used it to minimize forgetting when learning a new session. This was not done in our experiments to fairly compare it to the other models, which only had access to validation data for the current session.

Table 4 shows the total time to train each model for the data permutation and incremental class learning experiments using CUB-200. Both variants of GeppNet are orders of magnitude slower because they train the model one sample at a time. PathNet is also very slow at the data permutation

Model	Data Permutation	Incremental Class
MLP	16	15
EWC	16	13
PathNet	1,385	N/A
GeppNet	507	1,123
GeppNet+STM	179	410
FEL	53	8

Table 4: Each model's training time (minutes) on CUB-200.

task because the optimal path through a large DCNN needs to be found for each permutation. The fixed-size models are noticeably faster; however, only EWC was effective at mitigating catastrophic forgetting (in the data permutation and multi-modal experiments).

In general, models that expand as a function of the number of sessions and those that are allowed to store data from prior sessions may have limited real-world application. In our opinion, methods for mitigating catastrophic forgetting should have the amount of total memory they use constrained. While our summary statistics did not take this into account, it is an important factor in deploying a method that learns incrementally. This is the reason we chose to keep the number of trainable parameters fixed across all models.

An alternative would have been to tune the number of trainable parameters in each model for each experiment, which is what we did for the data permutation and incremental class learning experiments as well (see Supplemental Materials for details). Although in most cases the base performance increased, there were no changes to any of our conclusions on which model/mechanism yielded superior performance. The one interesting thing we did observe is that the sparsity model (i.e., FEL) can sometimes improve significantly; however, the cost is a 40x increase in the models memory footprint. This reinforces our claim that a model that only uses the sparsity mechanism to mitigate catastrophic forgetting may not be ideal in a deployed environment. We urge future incremental learning algorithm creators to take memory footprint into account, especially when comparing to other models.

Our metrics could be expanded to other training paradigms such as reinforcement learning, unsupervised learning, etc. In reinforcement learning, the agent learns an initial study-session (e.g., an ATARI game), which represents the base knowledge. We would track the performance of the base-knowledge as the model learns additional games and ensure that the model is learning new games as well. The main difference is that the performance metrics would be normalized by the maximum performance for each study-session when the model only has to learn that single study session. In unsupervised learning, we could follow the experiments performed by (Draelos et al. 2016) where the metrics would be the same, but we would train the models using a different loss function (e.g., reconstruction error).

Model	Incremental Class	Similar Data	Dissimilar Data	Memory Efficient	Trains Quickly
MLP	✗	✗	✗	✓	✓
EWC	✗	✗	✓	✓	✓
PathNet	✗	✓	✗	✗	✗
GeppNet	✓	✗	✗	✗	✗
GeppNet+STM	✓	✗	✗	✗	✗
FEL	✗	✗	✗	✗	✓

Table 5: Summary of the optimal performer on the incremental class learning, data permutation (Similar Data), and multi-modal (Dissimilar Data) experiments, as well as the memory/computational efficiency of each model.

Conclusion

In this paper, we developed new metrics for evaluating catastrophic forgetting. We identified five families of mechanisms for mitigating catastrophic forgetting in DNNs. We found that performance on MNIST was significantly better than on the larger datasets we used. Using our new metrics, experimental results (summarized in Table 5) show that 1) a combination of rehearsal/pseudo-rehearsal and dual-memory systems are optimal for learning new classes incrementally, and 2) regularization and ensembling are best at separating multiple dissimilar sessions in a common DNN framework. Although the rehearsal system performed reasonably well, it required retaining all training data for replay. This type of system may not be scalable for a real-world lifelong learning system; however, it does indicate that models that use pseudorehearsal could be a viable option for real-time incremental learning systems. Future work on lifelong learning frameworks should involve combinations of these mechanisms. While some models perform better than others in different scenarios, our work shows that catastrophic forgetting is not solved by any single method. We urge the community to use larger and more difficult datasets in future work.

Acknowledgements

Angelina Abitino was supported by NSF Research Experiences for Undergraduates (REU) award #1359361 to Roger Dube. We thank NVIDIA for the generous donation of a Titan X GPU.

References

Abraham, W. C., and Robins, A. 2005. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in Neurosciences* 28(2):73–78.

Abu-El-Haija, S.; Kothari, N.; Lee, J.; et al. 2016. Youtube-8m: A large-scale video classification benchmark. *arXiv:1609.08675*.

Coop, R.; Mishtal, A.; and Arel, I. 2013. Ensemble learning in fixed expansion layer networks for mitigating catastrophic forgetting. *IEEE Trans. on Neural Networks and Learning Systems* 24(10):1623–1634.

Dai, W.; Yang, Q.; Xue, G.-R.; and Yu, Y. 2007. Boosting for transfer learning. In *ICML*, 193–200. ACM.

Draelos, T. J.; Miner, N. E.; Lamb, C. C.; Vineyard, C. M.; Carlson, K. D.; James, C. D.; and Aimone, J. B. 2016. Neurogenesis deep learning. *arXiv:1612.03770*.

Eich, J. M. 1982. A composite holographic associative recall model. *Psych. Review* 89(6):627.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A.; and Wierstra, D. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv:1701.08734*.

French, R. M. 1997. Pseudo-recurrent connectionist networks: An approach to the 'sensitivity-stability' dilemma. *Connection Science* 9(4):353–380.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3(4):128–135.

Gemmeke, J. F.; Ellis, D. P. W.; Freedman, D.; Jansen, A.; Lawrence, W.; Moore, R. C.; Plakal, M.; and Ritter, M. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*.

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

Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv:1312.6211*.

Goodrich, B., and Arel, I. 2014. Unsupervised neuron selection for mitigating catastrophic forgetting in neural networks. In *IEEE 57th Int. Midwest Symposium on Circuits and Systems (MWSCAS), 2014*, 997–1000. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hershey, S.; Chaudhuri, S.; Ellis, D. P.; et al. 2017. Cnn architectures for large-scale audio classification. In *ICASSP*.

Hinton, G. E., and Plaut, D. C. 1987. Using fast weights to deblur old memories. In *Proc. of the Ninth Annual Conference of the Cognitive Science Society*, 177–186.

Kafle, K., and Kanan, C. 2017. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*.

Kanerva, P. 1988. *Sparse distributed memory*. MIT press.

Khilari, P., and Bhope, V. 2015. A review on speech to text conversion methods. *Int. J. of Advanced Research in Computer Engineering and Technology* 4:3067–3072.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proc. of the National Academy of Sciences* 201611835.

Kruschke, J. K. 1992. Alcove: An exemplar-based connectionist model of category learning. *Psych. review* 99(1):22.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 3431–3440.

McCloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psych. of Learning & Motivation* 24:109–165.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Murdock, B. B. 1983. A distributed memory model for serial-order information. *Psych. Review* 90(4):316.

Murre, J. M. 2014. *Learning and categorization in modular neural networks*. Psych. Press.

Polikar, R.; Upda, L.; Upda, S. S.; and Honavar, V. 2001. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 31(4):497–508.

Ren, B.; Wang, H.; Li, J.; and Gao, H. 2017. Life-long learning based on dynamic combination model. *Applied Soft Computing* 56:398–404.

Robins, A. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science* 7(2):123–146.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 815–823.

Sharkey, N. E., and Sharkey, A. J. 1995. An analysis of catastrophic interference. *Connection Science* 7:301–329.

Soltoggio, A.; Stanley, K. O.; and Risi, S. 2017. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *arXiv:1703.10371*.

Triki, A. R.; Aljundi, R.; Blaschko, M. B.; and Tuytelaars, T. 2017. Encoder based lifelong learning. *arXiv:1704.01920*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. Tech Report: CNS-TR-2011-001.

Wang, H.; Fan, W.; Yu, P. S.; and Han, J. 2003. Mining concept-drifting data streams using ensemble classifiers. 226–235. ACM.

Yu, L., and Liu, H. 2003. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML*, 856–863.