Project - Continual learning

Option: MNIST

METHOD: GRADIENT EPISODIC MEMORY(GEM)

## Group 8:

## Shivakalyan Soundarathiagarajan,

## Kowsick Venkatachalapathi,

## Adish Vakharia

(Experimented based on the paper - <http://papers.nips.cc/paper/7225-gradient-episodic-memory-for-continual-learning.pdf>)

# Gradient Episodic Memory

* Continual learning, where the model observes, once and one by one, examples concerning a sequence of tasks.
* Set of metrics to evaluate models learning over a continuum of data.
* These metrics characterize models not only by their test accuracy, but also in terms of their ability to transfer knowledge across tasks the continual learning, called Gradient Episodic Memory (GEM) alleviates forgetting, while allowing beneficial transfer of knowledge to previous tasks. The experiments on variants of the MNIST and CIFAR-100 datasets demonstrate the strong performance of GEM when compared to the state-of-the-art.

## Data Sets

We consider the following datasets:

* **MNIST Permutation** where each task is transformed by a fixed permutation of pixels. In this dataset, the input distribution for each task is unrelated.
* **MNIST Rotations**, a variant of MNIST where each task contains digits rotated by a fixed angle between 0 and 180 degrees.
* **Incremental CIFAR100** a variant of the CIFAR object recognition dataset with 100 classes, where each task introduces a new set of classes.

For all the datasets, we considered T = **20 tasks**.

On the MNIST datasets, each task has **1000 examples from 10 different classes**.

On the CIFAR100 dataset each task has **2500 examples from 5 different classes**. The model observes the tasks in sequence, and each example once.

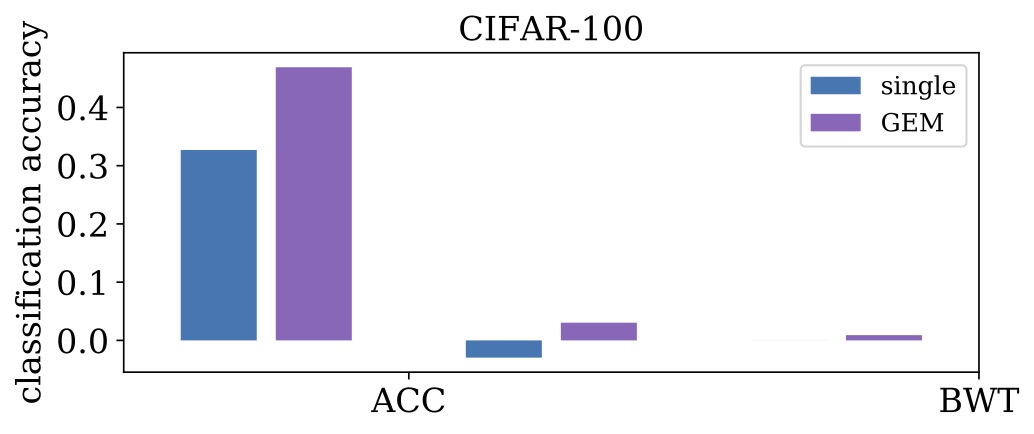
The evaluation for each task is performed on the test partition of each dataset.

## 

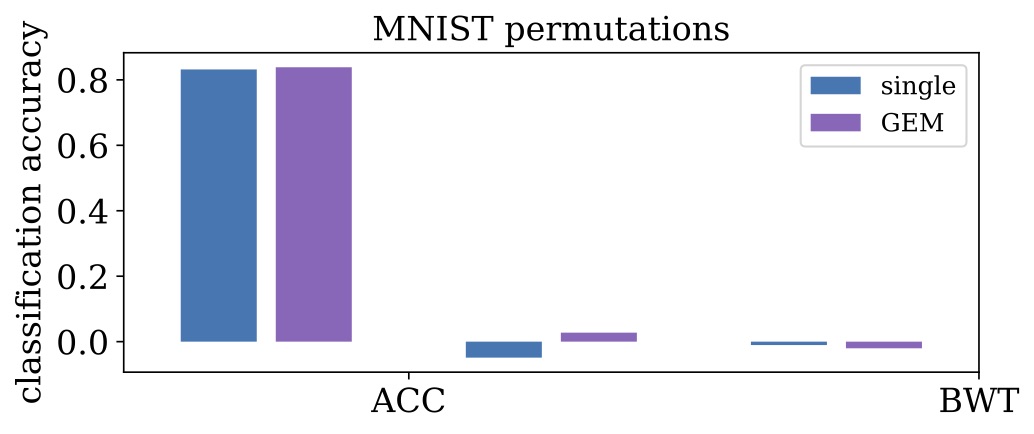
## Architectures:

* On the MNIST tasks, we use fully connected neural networks with two hidden layers of 100 ReLU units.
* On the CIFAR100 tasks, we use a smaller version of ResNet18 with three times less feature maps across all layers.
* Also, on CIFAR100, the network has a final linear classifier per task. We train all the networks and baselines using plain SGD on mini batches of 10 samples.

**Figure 1**: ACC and BWT for all CIFAR-100 datasets and methods.



**Figure 2**: ACC and BWT for all MNIST datasets and methods.



**Figure 3**: ACC and BWT for all MNIST rotations datasets and methods.

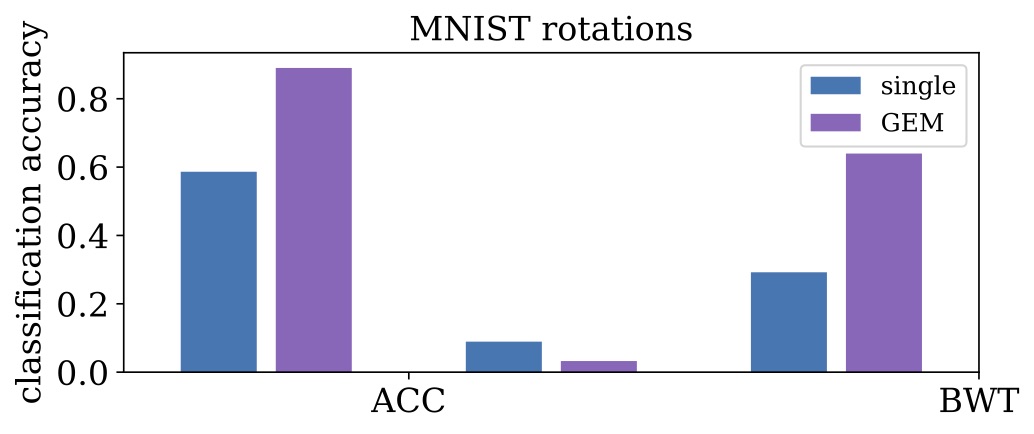
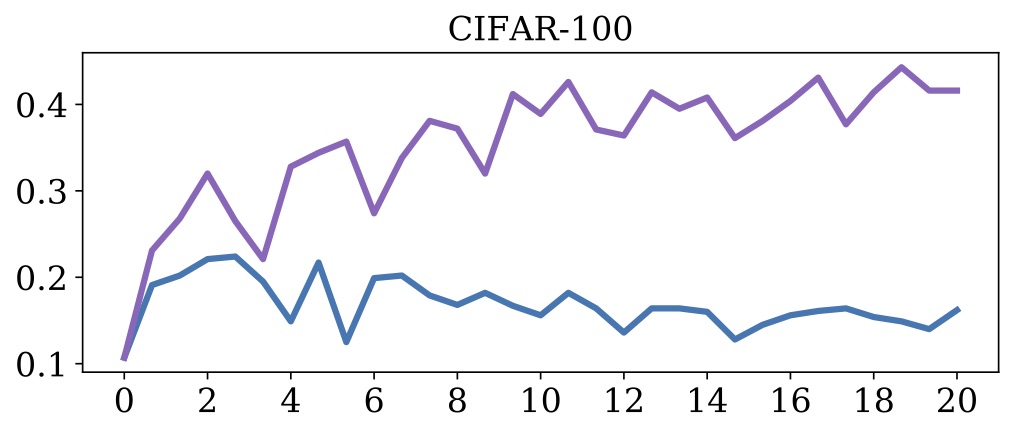
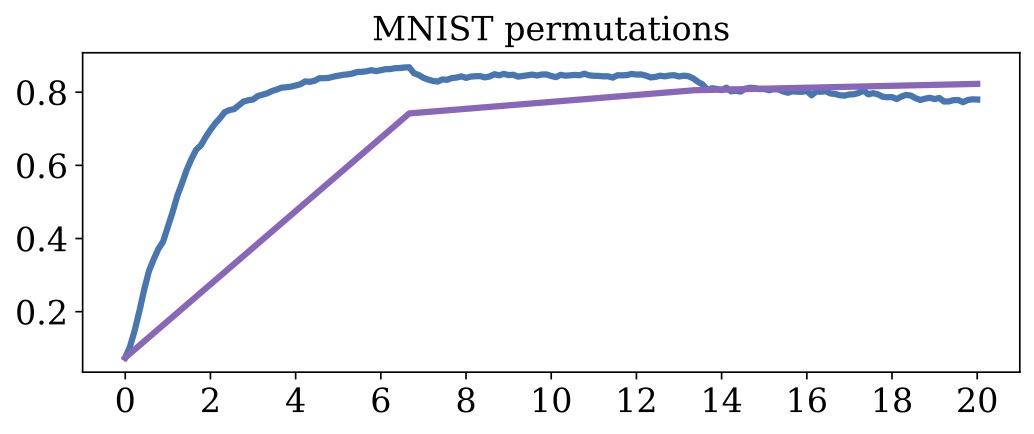


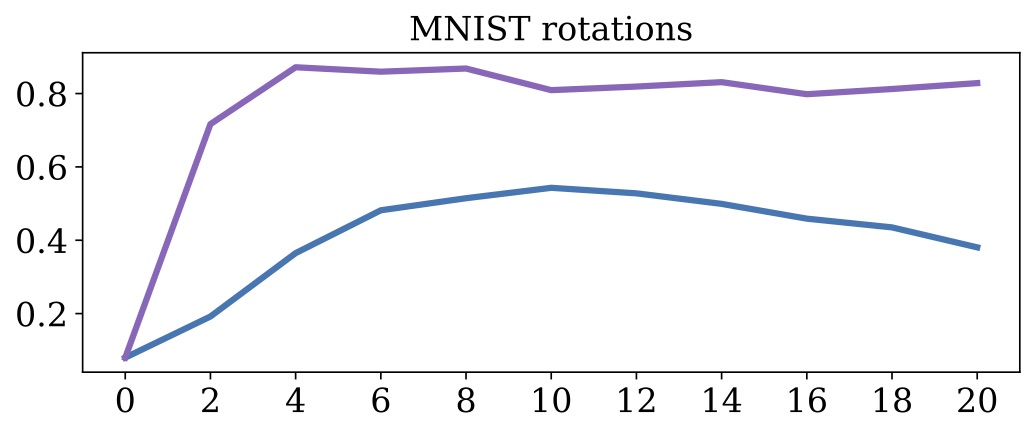
Figure 4, 5 and 6 shows the evolution of the test accuracy, as more tasks are learned.

**Figure 4:**



**Figure 5:**

**Figure 6:**



## Methods:

We compare GEM to five alternatives:

1. a single predictor trained across all tasks.
2. one independent predictor per task. Each independent predictor has the same architecture as “single” but with T times less hidden units than “single”. Each new independent predictor can be initialized at random or be a clone of the last trained predictor (decided by grid-search).
3. a multimodal predictor, which has the same architecture of “single”, but with a dedicated input layer per task (only for MNIST datasets).
4. EWC, where the loss is regularized to avoid catastrophic forgetting.
5. GEM and EWC have the same architecture as “single”, plus episodic memory.

## Evaluation Metrics

1. Backward transfer (BWT), which is the influence that learning a task t has on the performance on a previous task k ≺ t. On the one hand, there exists positive backward transfer when learning about some task t increases the performance on some preceding task k. On the other hand, there exists negative backward transfer when learning about some task t decreases the performance on some preceding task k.
2. Large negative backward transfer is also known as (catastrophic) forgetting.
3. Forward transfer (FWT), which is the influence that learning task t has on the performance on a future task k ≻ t. In particular, positive forward transfer is possible when the model is able to perform “zero-shot” learning, perhaps by exploiting the structure available in the task descriptors.

## Results

Refer the results folder in the GitHub repo.

Overall, GEM performs significantly better than other continual learning methods like EWC, while spending less computation (Table 1). GEM’s efficiency comes from optimizing over a number of variables equal to the number of tasks (T = 20 in our experiments), instead of optimizing over a number of variables equal to the number of parameters (p = 1109240 for CIFAR100 for instance). GEM’s bottleneck is the necessity of computing previous task gradients at each learning iteration.

## Conclusion

* We formalized the scenario of continual learning.
* First, we defined training and evaluation protocols to assess the quality of models in terms of their accuracy, as well as their ability to transfer knowledge forward and backward between tasks.
* Second, we introduced GEM, a simple model that leverages an episodic memory to avoid forgetting and favor positive backward transfer. Our experiments demonstrate the competitive performance of GEM against the state-of-the-art.
* GEM has three points for improvement.
  + First, GEM does not leverage structured task descriptors, which may be exploited to obtain positive forward transfer (zero-shot learning).
  + Second, we did not investigate advanced memory management.
  + Third, each GEM iteration requires one backward pass per task, increasing computation time.