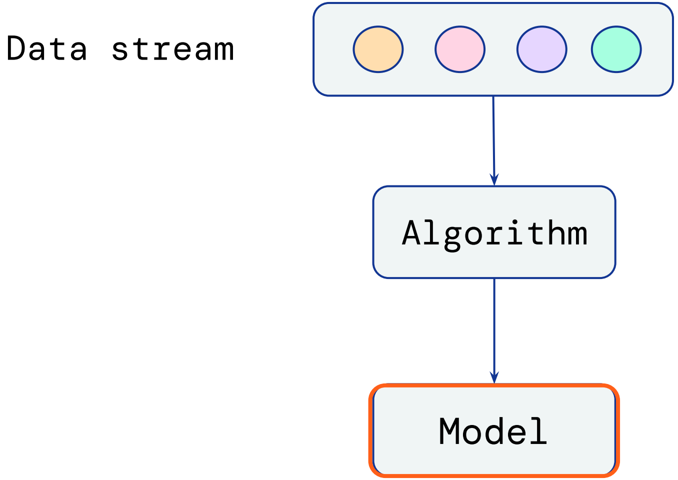
Wide Neural Networks Forget Less Catastrophically

Seyed Iman Mirzadeh*[1], Arslan Chaudhry†[2], Dong Yin[2],
Huiyi Hu[2], Razvan Pascanu[2], Dilan Gorur[2], Mehrdad Farajtabar†[2]

*Work done during an internship at DeepMind. †Equal Advising
[1]Washington State University, [2]DeepMind

Motivation

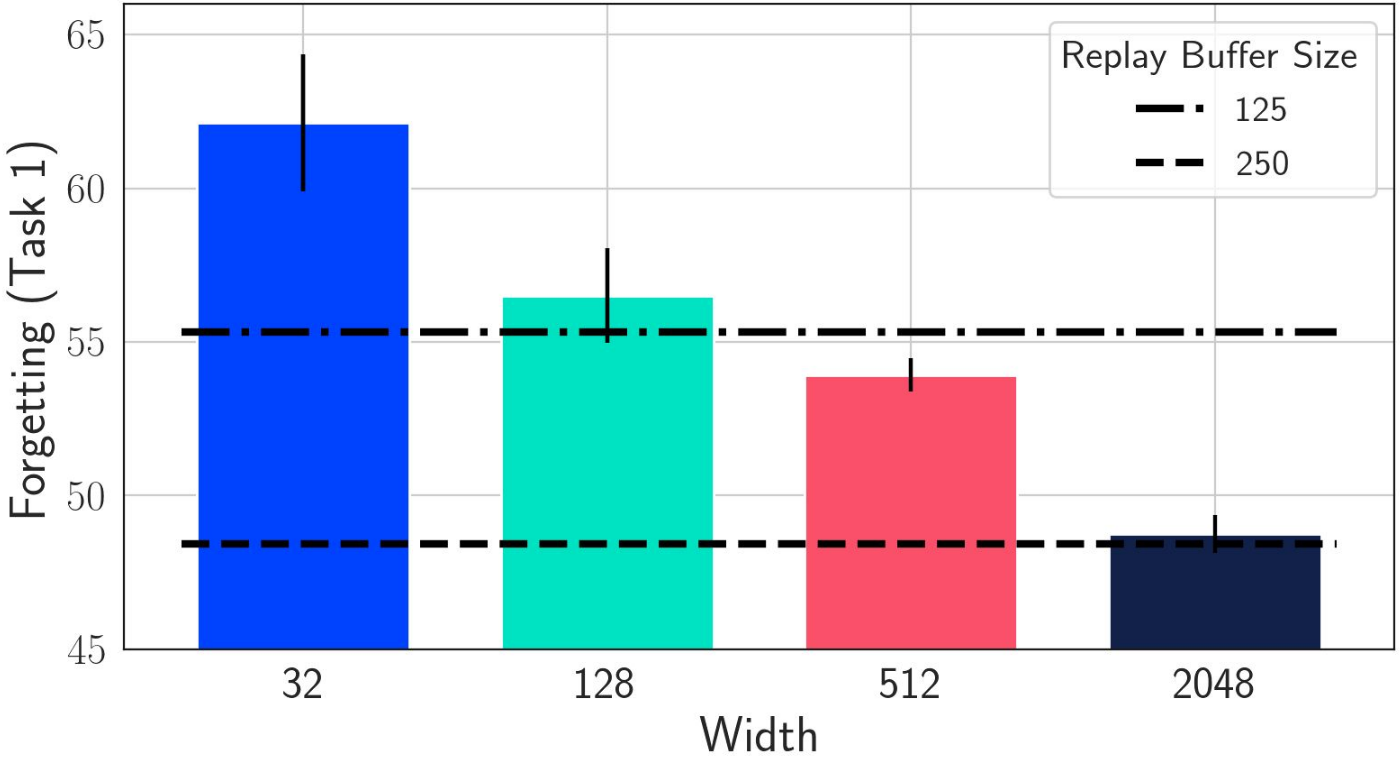
- The focus of the CL literature is mainly on **algorithms** rather than the **model/architecture**
- A typical Setup in Continual Learning:



The algorithm controls the train-loop:
* regularization
* replay
* parameter-isolation

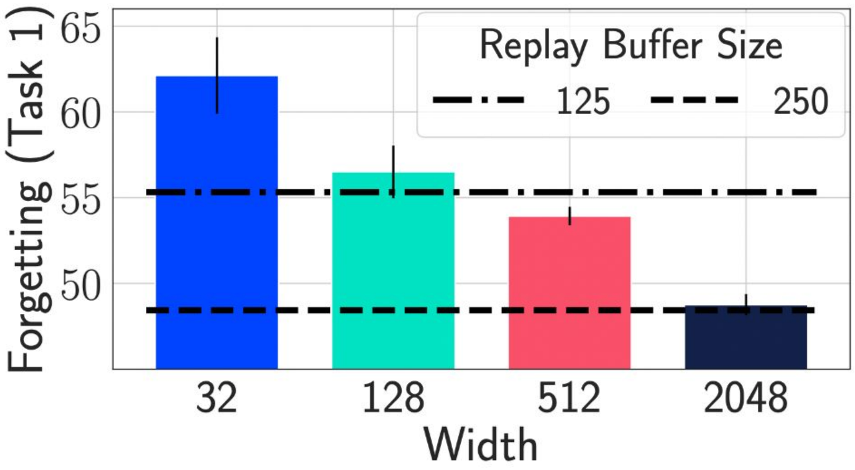
Often, the model is assumed to be fixed

What Will Happen If We Use Wider Networks?

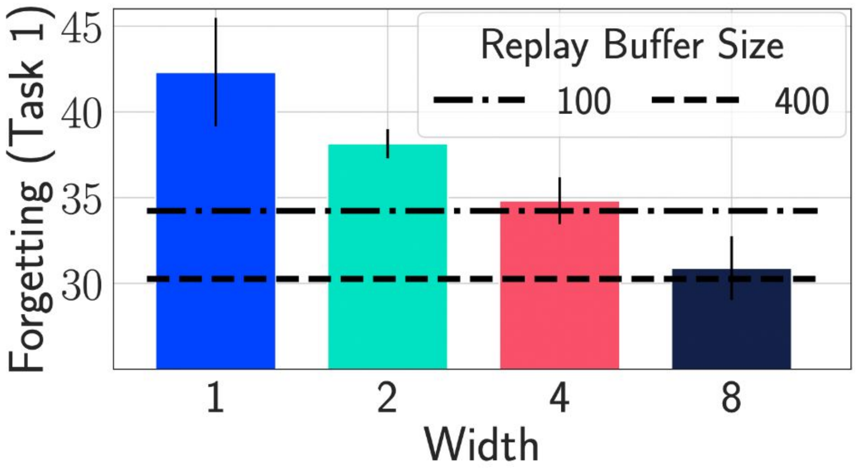


MLP with 2 layers, Rotated MNIST (5 tasks)

What Will Happen If We Use Wider Networks?



MLP on Rotated MNIST

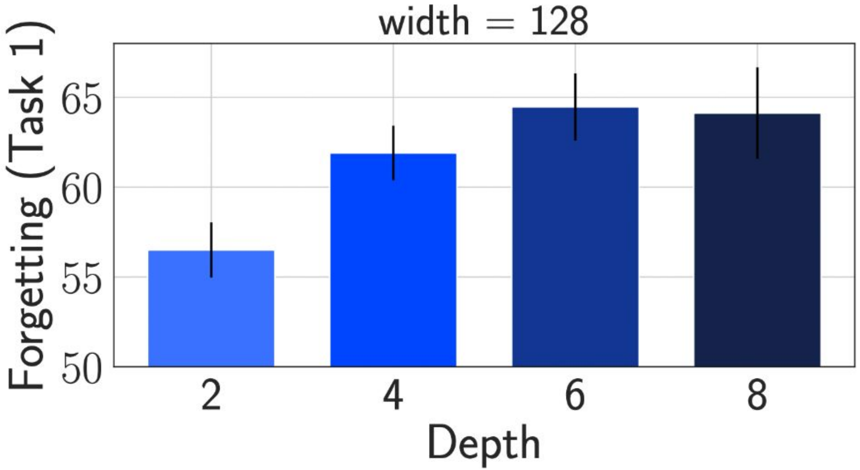


WRN-`10-W` on Split CIFAR-100

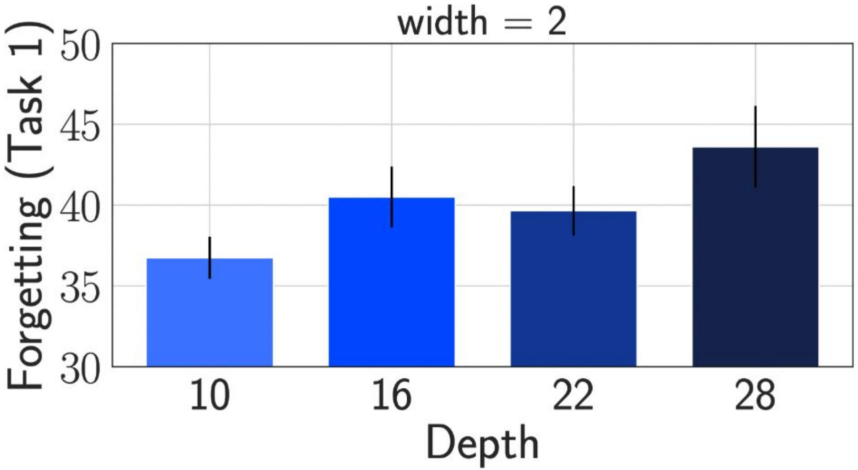
Why?

- Is it because of the parameters?
  - Wider model → more parameters → larger capacity → less forgetting?
  - Increasing the depth can also be helpful (?)

What Will Happen If We Use Deeper Networks?



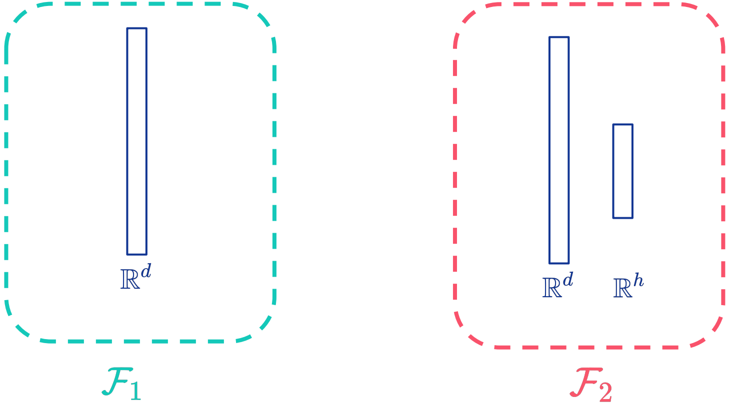
MLP on Rotated MNIST



WRN-**D**-**2** on Split CIFAR-100

Theoretical Explanation

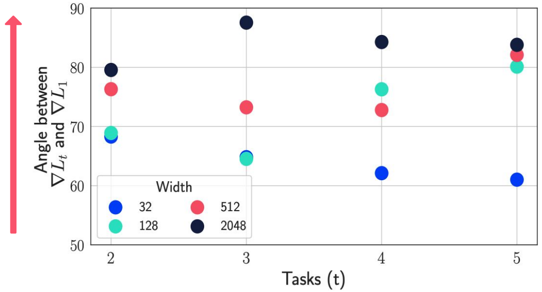
- A very simple theoretical analysis:



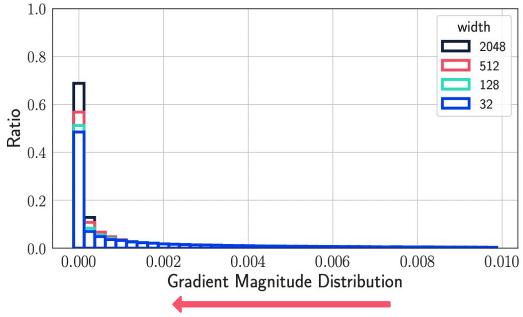
Claim 4.1 (informal). *Consider learning problems with input space $\mathbb{R}^d$, output space $\mathbb{R}$, and squared loss. Let $\mathcal{F}_1$ be the class of linear models that maps the input to the output and $\mathcal{F}_2$ be the class of two-layer **linear** networks (i.e., no nonlinear activation). Then, there exist two tasks such that when we train task 2 using gradient descent, if we use model class $\mathcal{F}_1$, the amount of forgetting for task 1 is strictly zero; whereas if we use model class $\mathcal{F}_2$, the amount of forgetting can be positive.*

Empirical Explanations

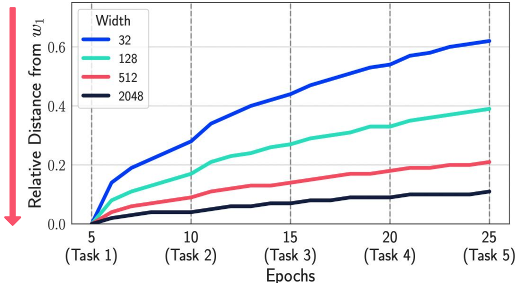
Gradient orthogonalization



Gradient sparsity



Lazy training regime



Experiment: Width vs Depth

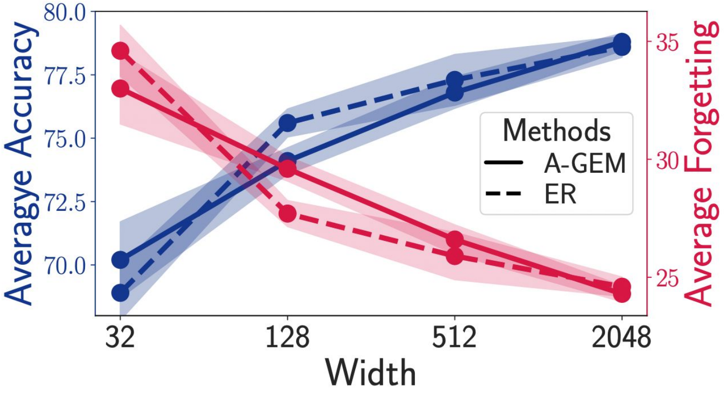
MLP on Rotated MNIST

Width	Depth	Parameters	Average Accuracy	Average Forgetting	Joint Accuracy
128	8	217.35 K	68.9 $\pm$1.07	35.4 $\pm$1.34	94.1 $\pm$0.73
256	2	269.32 K	**71.1 $\pm$ 0.43**	**31.4 $\pm$ 0.48**	93.9 $\pm$0.65
256	8	664.08 K	70.4 $\pm$0.61	32.1 $\pm$0.75	94.78 $\pm$0.67
512	2	669.70 K	**72.6 $\pm$ 0.27**	**29.6 $\pm$ 0.36**	94.08 $\pm$0.77

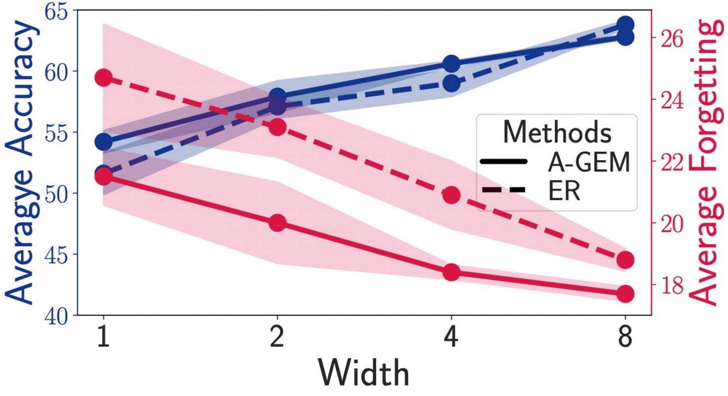
WRN on Split CIFAR-100

Depth	Width	Params	Average Accuracy	Average Forgetting	Joint Accuracy
10	4	1.69 M	**53.8 $\pm$ 2.74**	**33.8 $\pm$ 2.16**	83.4 $\pm$0.64
28	2	1.61 M	46.6 $\pm$2.56	37.1 $\pm$2.47	83.6 $\pm$0.54
10	8	3.72 M	**59.7 $\pm$ 2.33**	**29.4 $\pm$ 2.52**	84.8 $\pm$0.49
16	4	3.24 M	50.1 $\pm$2.59	37.0 $\pm$2.77	85.1 $\pm$0.45
28	3	3.58 M	49.4 $\pm$1.82	36.2 $\pm$1.98	84.7 $\pm$0.92

Experiment: Interacting with Other Algorithms



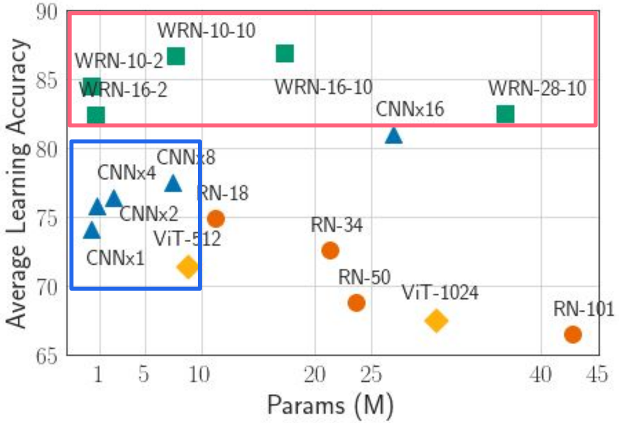
MLP on Rotated MNIST



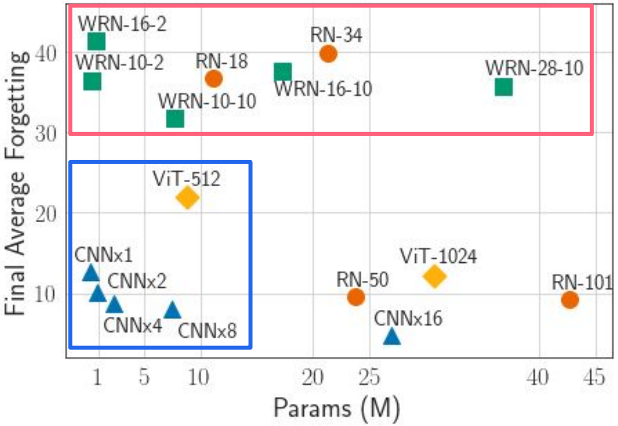
WRN on Split CIFAR-100

Follow-up Work : Beyond Width & Depth

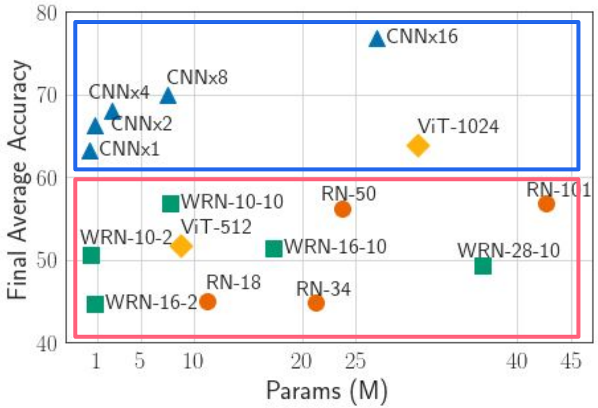
- What about other architectures & other Benchmarks?



Learning Accuracy: The accuracy for each task directly after it is learned.



Forgetting: the difference between the peak accuracy and the final accuracy of each task.



Average Accuracy: the average of validation accuracies, after learning all tasks.

"Architecture Matters in Continual Learning", https://arxiv.org/abs/2202.00275

Conclusion

- Instead of focusing on algorithms, let's focus on models & architectures

- Increasing the width can reduce the forgetting in continual learning by:
    - Increasing the gradient orthogonality
    - Increasing the gradient sparsity
    - Having a lazier training regime

- The findings hold for other CL algorithms, architectures, and benchmarks.

- We hope our work draws more attention to the research at the intersection of continual learning and neural network architectures.

Thank You!

Poster: Hall E #628