Momentum Residual Neural Networks

M. E. Sander*, P. Ablin*, M. Blondel†, G. Peyré*

*ENS, CNRS †Google Research, Brain team

Residual Neural Networks and their invertible versions

- **Residual blocks:** $x_{n+1} = x_n + f(x_n, \theta_n)$, a scalar loss $L(x_N, \theta)$ to minimize
- **Backpropagation**: $\nabla_{x_n} L = \nabla_{x_{n+1}} L + \partial_x f(x_n, \theta)^T \nabla_{x_{n+1}} L$
- **Memory issue in increasingly deep architectures:** requirement to store the x_n's
- **Solution**: **re**-compute iteratively the x_n's during the backpropagation iterations
- **Invertible** models rely on constrained architectures so far

Simple modification of the ResNet's forward rule

ResNet

$$x_{n+1} = x_n + f(x_n, \theta_n)$$

Momentum ResNet

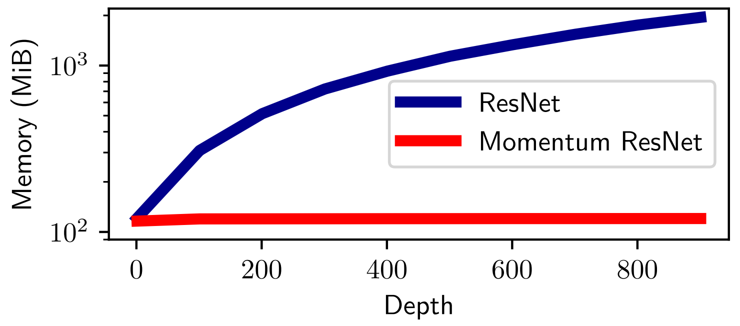
$$\begin{cases} v_{n+1} &= \gamma v_n + (1-\gamma) f(x_n, \theta_n) \\ x_{n+1} &= x_n + v_{n+1}, \end{cases}$$

- **Drop-in replacemen**t: ResNet turned into its Momentum counterpart only by changing the forward equations (same parameters as inputs)

Invertibility and memory savings

- **Exactly** inverted by $\begin{cases} x_n &= x_{n+1} - v_{n+1}, \\ v_n &= \frac{1}{\gamma}(v_{n+1} - (1-\gamma) f(x_n, \theta_n)) \end{cases}$

- Avoids the **memory bottleneck**

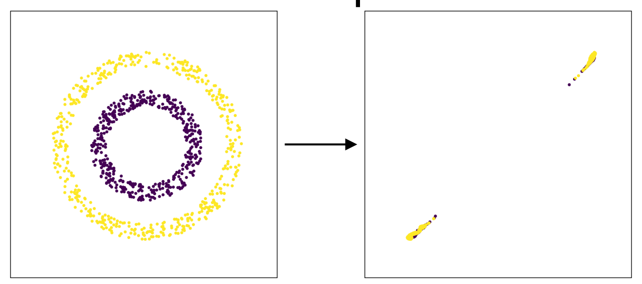


Theoretical properties of Momentum ResNets

ResNet

$$\dot{x} = f(x, \theta)$$

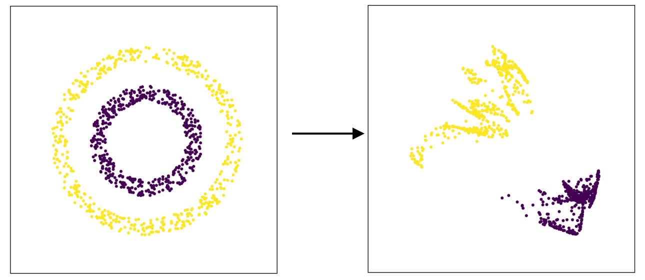
Homeomorphisms



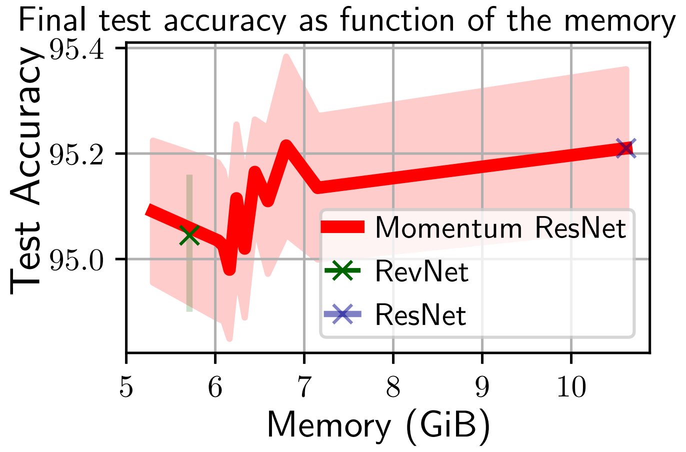
Momentum ResNet

$$\varepsilon \ddot{x} + \dot{x} = f(x, \theta)$$

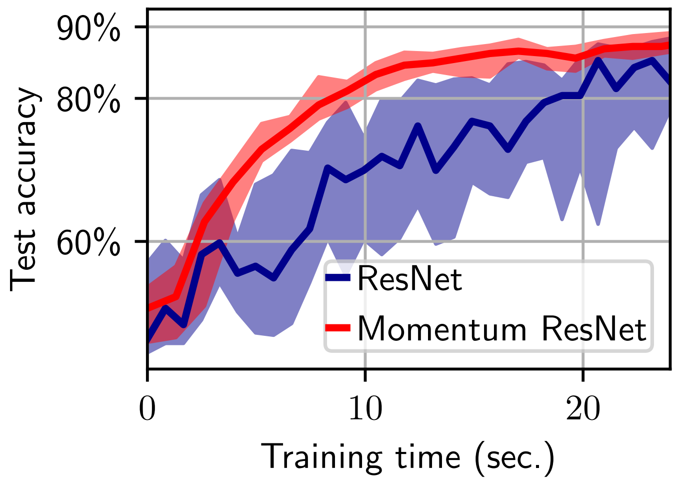
Larger class of mappings



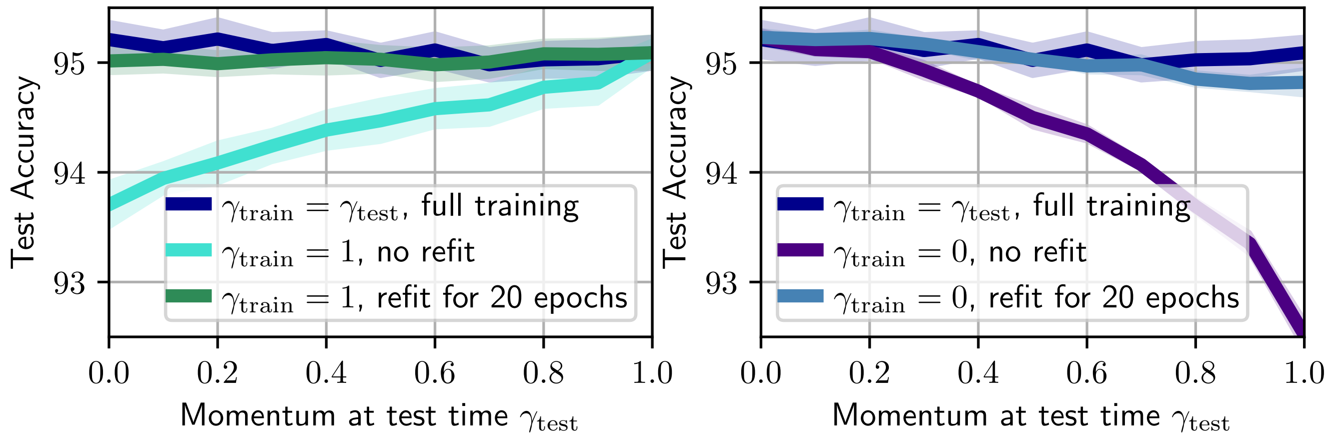
Memory Aspect



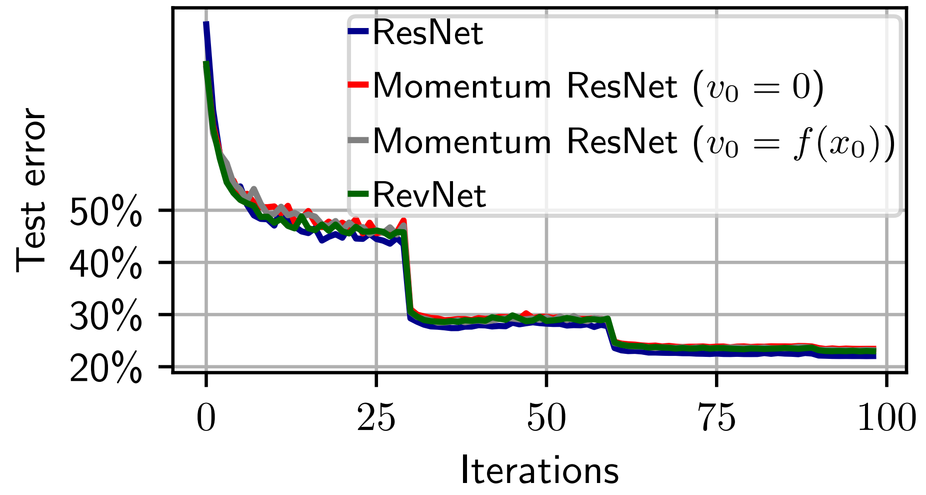
Pre-training and fine-tuning



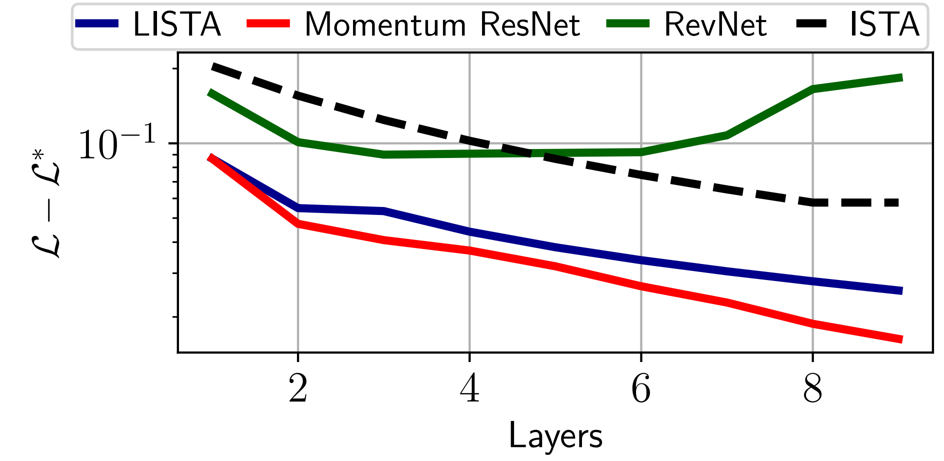
Influence of the momentum term



ImageNet learning curves



Learning to optimize setting



Open-source python package

Website: https://michaelsdr.github.io/momentumnet/

Github: https://github.com/michaelsdr/momentumnet

```
$ pip install momentumnet
```

```
>>> import torch
>>> from momentumnet import transform_to_momentumnet
>>> from torchvision.models import resnet101
>>> resnet = resnet101(pretrained=True)
>>> mresnet101 = transform_to_momentumnet(resnet, gamma=0.99, use_backprop=False)
```

```
>>> import torch
>>> from momentumnet import transform_to_momentumnet
>>> transformer = torch.nn.Transformer(num_encoder_layers=6, num_decoder_layers=6)
>>> mtransformer = transform_to_momentumnet(transformer,
>>>                     residual_layers=["encoder.layers", "decoder.layers"])
```