Hyperparameters of the experiments

- Optimizer is RMSprop unless specified otherwise.
- Batch size is consistently 128 for each task and each model. PTB-c especially use eval batch size of 10 (this decides the data being used for validation).

Data	Many-to-many	Batch first	Epoch/Iteration
MNIST	False	True	70
Copying Memory Task	True	True	4000
PTB-c	True	False	100

expRNN

Data	Hidden size	Skew init	Recurrent learning rate	Learning rate	α
MNIST	170	Cayley	$7 * 10^{-5}$	$7 * 10^{-4}$	0.99
MNIST	360	Cayley	$5 * 10^{-5}$	$5 * 10^{-4}$	0.99
MNIST	512	Cayley	$3 * 10^{-5}$	$3 * 10^{-4}$	0.99
P-MNIST	170	Cayley	10^{-4}	10^{-3}	0.99
P-MNIST	360	Cayley	$7 * 10^{-5}$	$7 * 10^{-4}$	0.99
P-MNIST	512	Cayley	$5 * 10^{-5}$	$5 * 10^{-4}$	0.99
Copying Problem $L = 1000$	190	Henaff	$2 * 10^{-5}$	$2 * 10^{-4}$	0.99
Copying Problem $L = 2000$	190	Henaff	$2 * 10^{-5}$	$2 * 10^{-4}$	0.99
TIMIT	224	To Be Tuned	10^{-4}	Adam 10^{-3}	0.99
TIMIT	322	To Be Tuned	$7 * 10^{-5}$	Adam $7 * 10^{-4}$	0.99
TIMIT	425	To Be Tuned	$7 * 10^{-5}$	Adam $7 * 10^{-4}$	0.99
PTB-c $T = 150$	1024	Cayley	0.0001	0.005	0.9
PTB-c $T = 150$	1386	Cayley	0.0001	0.005	0.9
PTB-c $T = 300$	1024	Cayley	0.0001	0.005	0.9
PTB-c $T = 300$	1386	Cayley	0.0001	0.005	0.9

scoRNN

- Always init as Cayley
- D is init as:

```
D = np.diag(np.concatenate([np.ones(n_hidden - n_neg_ones), \
    -np.ones(n_neg_ones)]))
```

Data	Hidden size	Recurrent learning rate	Other parameters learning rate	ρ
MNIST	170	10^{-4}	10^{-3}	$\frac{1}{10}$
MNIST	360	10^{-5}	10^{-4}	$\frac{1}{10}$
MNIST	512	10^{-5}	10^{-4}	$\frac{1}{10}$
P-MNIST	170	10^{-4}	10^{-3}	$\frac{1}{2}$
P-MNIST	360	10^{-5}	10^{-4}	$\frac{1}{2}$
P-MNIST	512	10^{-5}	10^{-4}	$\frac{1}{2}$
Copying Problem $L = 1000$	190	10^{-4}	10^{-3}	$\frac{1}{2}$
Copying Problem $L = 2000$	190	10^{-4}	10^{-3}	$\frac{1}{2}$
TIMIT	224	$1e-3$	Adam $1e-3$	$\frac{1}{10}$
TIMIT	322	$1e-3$	Adam $1e-3$	$\frac{1}{10}$
TIMIT	425	$1e-3$	Adam $1e-3$	$\frac{1}{10}$

LSTM

- Learning rate 10^{-3} and decay rate 0.9 in all experiment.
- Gradient clipping norm 1.

Data	Hidden size	Forget bias init
MNIST	128	1
MNIST	256	1
MNIST	512	1
P-MNIST	128	1
P-MNIST	256	1
P-MNIST	512	1
Copying Problem $L = 1000$	68	1
Copying Problem $L = 2000$	68	1
TIMIT	84	-4

Data	Hidden size	Forget bias init
TIMIT	120	-4
TIMIT	158	-4

- URNN, EURNN and RGD has complicated implementation given out-dated or no code at all. I'll do it if we're certain to use it, or maybe at some point in the future.
- EURNN has multiple variants, and it isn't clear which was used in Lezcano, et al. Even though RGD only has one implementation, the reported #PARAMS is confusing. I'm not really sure which setting was used in RGD.
- → Leave for future dissertation.

Partial Space Unitary RNN (Arjovsky et al., 2016)

- We initialize $\mathbf{V}$ and $\mathbf{U}$ (the input and output matrices) as in (Glorot & Bengio, 2010), with weights sampled independently from uniforms, $\mathcal{U}\left[-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}\right]$.
- The biases, b and b_o are initialized to 0. This implies that at initialization, the network is linear with unitary weights, which seems to help early optimization (Saxe et al., 2014).
- The reflection vectors for $\mathbf{R}_1$ and $\mathbf{R}_2$ are initialized coordinate-wise from a uniform $\mathcal{U}[-1, 1]$. Note that the reflection matrices are invariant to scalar multiplication of the parameter vector, hence the width of the uniform initialization is unimportant.
- The diagonal weights for $\mathbf{D}_1, \mathbf{D}_2$ and $\mathbf{D}_3$ are sampled from a uniform, $\mathcal{U}[-\pi, \pi]$. This ensures that the diagonal entries $\mathbf{D}_{j,j}$ are sampled uniformly over the complex unit circle.
- We initialize h_0 with a uniform, $\mathcal{U}\left[-\sqrt{\frac{3}{2n_h}}, \sqrt{\frac{3}{2n_h}}\right]$, which results in $\mathbb{E}\left[\|h_0\|^2\right] = 1$. Since the norm of the hidden units are roughly preserved through unitary evolution and inputs are typically whitened to have norm 1, we have hidden states, inputs and linear outputs of the same order of magnitude, which seems to help optimization.

- Default to a learning rate of 10^{-3} and a decay rate of 0.9.
- Because the implementation of this architecture is complicated, I'll do it when we're really need it or at a later time.

Data	Hidden size

Data	Hidden size
MNIST	512
MNIST	2170
P-MNIST	512
P-MNIST	2170

EURNN

- Unclear whether the model used in Cheap Orthogonal is Tunable or FFT, also which capacity was used.
- Test accuracy of 0.937 is the reported accuracy of EURNN with hidden size 1024 Jing et. al, and with hidden size 512 in Lezcano, et al. → Either the hidden size or the architecture is reported wrongly in Lezcano, et al.

Data	Hidden size
MNIST	512
P-MNIST	512
TIMIT	158
TIMIT	256
TIMIT	378

RGD(Full-capacity Unitary RNN)

MODEL	N	# PARAMS
EXPRNN	170	$\approx 16K$
EXPRNN	360	$\approx 69K$
EXPRNN	512	$\approx 137K$
SCORNN	170	$\approx 16K$
SCORNN	360	$\approx 69K$
SCORNN	512	$\approx 137K$
LSTM	128	$\approx 68K$
LSTM	256	$\approx 270K$
LSTM	512	$\approx 1058K$
RGD	116	$\approx 9K$
RGD	512	$\approx 137K$

MODEL	N	# PARAMS
EXPRNN	224	$\approx 83K$
EXPRNN	322	$\approx 135K$
EXPRNN	425	$\approx 200K$
SCORNN	224	$\approx 83K$
SCORNN	322	$\approx 135K$
SCORNN	425	$\approx 200K$
LSTM	84	$\approx 83K$
LSTM	120	$\approx 135K$
LSTM	158	$\approx 200K$
EURNN	158	$\approx 83K$
EURNN	256	$\approx 135K$
EURNN	378	$\approx 200K$
RGD	128	$\approx 83K$
RGD	192	$\approx 135K$
RGD	256	$\approx 200K$

- Either there is a mistake in the reported #PARAMS, or RGD is not really Full-capacity Unitary RNN.