# Neurogenic Deep Learning

Timothy Draelos[1], Nadine Miner[1], Jonathan Cox[1], Christopher Lamb[1], William Severa[1], Conrad James[1], James B. Aimone[1] | *Sandia National Laboratories*[1]

Sandia National Laboratories

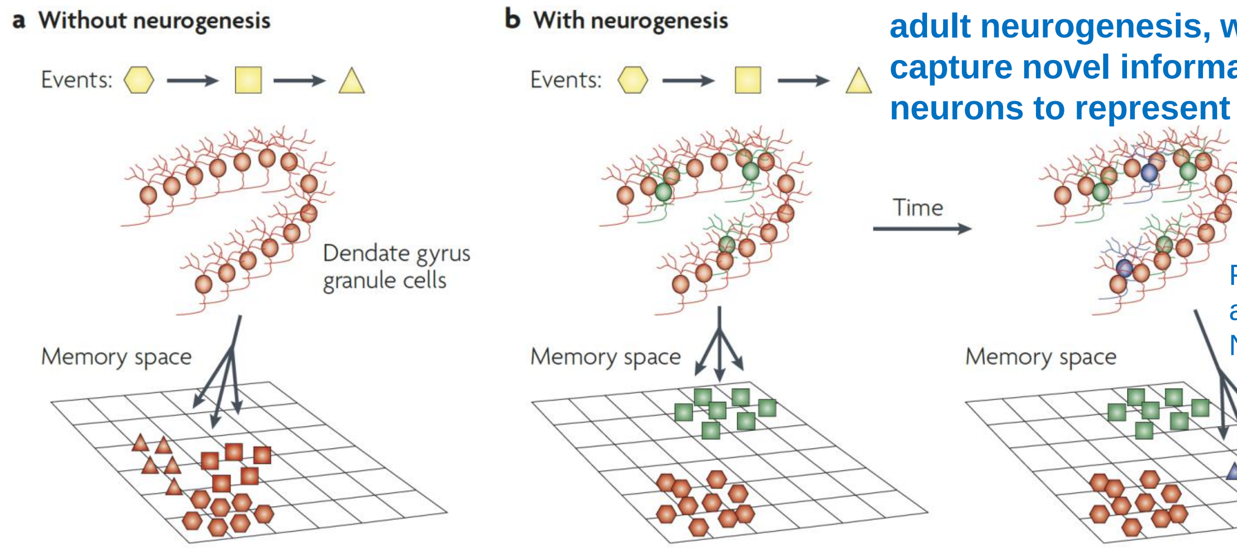
SAND2016-4054C

## PROBLEM: How to expand trained deep nets to accommodate new data classes?

In contrast to biological neural systems, capable of continuous learning, DNNs have a limited ability to incorporate new information in a trained network, so methods for continuous learning may be highly impactful in enabling the application of DNNs to dynamic data sets. Inspired by adult neurogenesis in the hippocampus, we explore the potential for adding new nodes to layers of DNNs to facilitate their acquisition of novel information while preserving previously trained representations. Results demonstrate that neurogenesis is well suited for addressing the stability-plasticity dilemma that has long challenged adaptive machine learning algorithms.

*Acronyms*
- AE: Autoencoder
- DL: Deep Learning
- DNN: Deep Neural Network
- IR: Intrinsic Replay
- NDL: Neurogenic DL
- OL: Online Learning
- RE: Reconstruction Error

### *Neuroscience Inspiration*




**New memories are encoded with the help of adult neurogenesis, whereby new neurons capture novel information allowing mature neurons to represent familiar features.**

Figure from Deng, Aimone, and Gage, Nature Reviews Neuroscience 2010.

### *Neurogenesis algorithm as an efficient method for adapting DNNs*

The value of a model to continuously adapt to changing data is challenging to quantify. Here, we quantify the value of a machine learning algorithm at a given time as follows.

Utility = Benefit – (Cost of Model / Lifetime) – Cost of runtime

**Extending the lifetime of a model by adapting in response to real-world data changes (e.g., via neurogenesis) mitigates the high initial training costs of DNNs.**

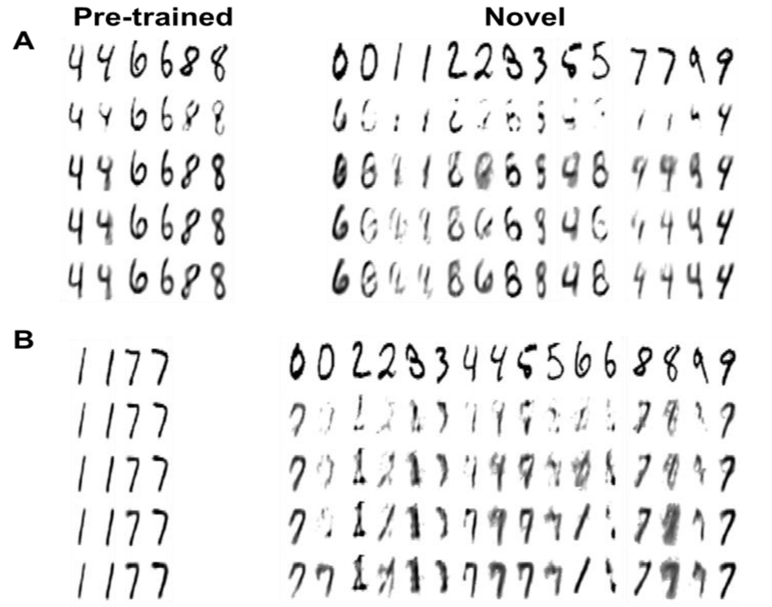
### *Layer-wise Reconstruction Error as a measure of representation capability*

RE is computed at internal layer $L$ within an AE by encoding an input sample through $L$ encode layers, then propagating through the corresponding $L$ decode layers to the output. An AE parameterized with weights $W$, biases $b$, and activation function $s$ is described from input, $x$, to output as $N$ encode layers followed by $N$ decode layers.

Encoder: $f_{\theta_N} \circ f_{\theta_{N-1}} \cdots f_{\theta_2} \circ f_{\theta_1}(x)$ where $y = f_\theta(x) = s(Wx + b)$

Decoder: $g_{\theta'_N} \circ g_{\theta'_{N-1}} \cdots g_{\theta'_2} \circ g_{\theta'_1}(y)$ where $g_{\theta'(y)} = s(W'y + b')$

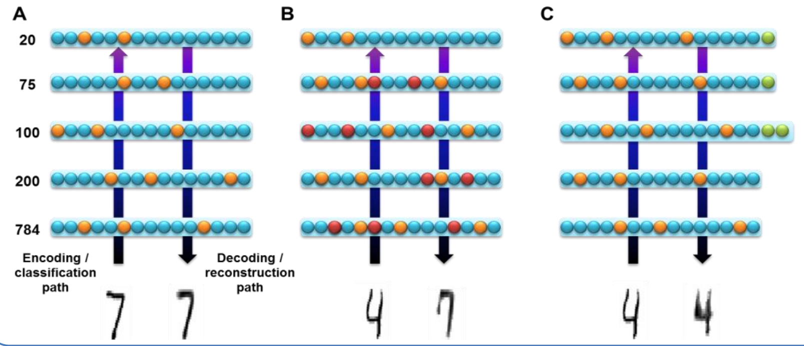
Then, the global RE at layer $L$ is $RE_{Global,L}(x) = (x - g_{\theta'_N} \circ \cdots g_{\theta'_{N-L}} \circ f_{\theta_L} \circ \cdots f_{\theta_1}(x))^2$




**PROBLEM**

**Existing AE can't represent novel data**

Networks initially trained on (**A**) 4, 6, and 8's and (**B**) 1 and 7's and not yet trained on any of the other MNIST digits reconstruct those novel digits using features biased by their original training data. The first row of each panel is the original data, with the 2nd - 5th rows the reconstruction through the 1st - 4th (full) encode and corresponding decode layers of the autoencoder, respectively.




**SOLUTION**

**Add new node(s) to each layer of an AE as needed to represent novel data**

(**A**) Deep AE can faithfully reconstruct originally trained digit 7.
(**B**) AE fails at reconstructing novel digit 4.
(**C**) New nodes added to layers 2, 3, and 4 enable AE to be trained to reconstruct the 4.

## Neurogenic Deep Learning (NDL) Algorithm

The NDL algorithm adds and trains new nodes to a layer of an AE similar to layerwise pretraining when a critical number of input samples fail to achieve adequate representation.

- ***Plasticity*** occurs as new nodes are added to represent novel data, then network
- ***Stability*** occurs by leveraging both new data and replayed samples from previously seen classes. Samples from old classes are created using the representation capability of the AE in a process we call "intrinsic replay" (see below).

**Input**: 2N-layer autoencoder *AE* trained on data classes $\{D_1, D_2, \ldots, D_{U-1}\}$, new class of data $D_U$, vector of per-layer RE thresholds *Th*, vector of per-layer maximum nodes allowed to add *MaxNodes*, maximum number of samples allowed to have $RE_L > Th_L$, *MaxOutliers*, Learning Rate *LR*

**Output**: Autoencoder *AE* capable of representing data classes $\{D_1, D_2, \ldots, D_U\}$

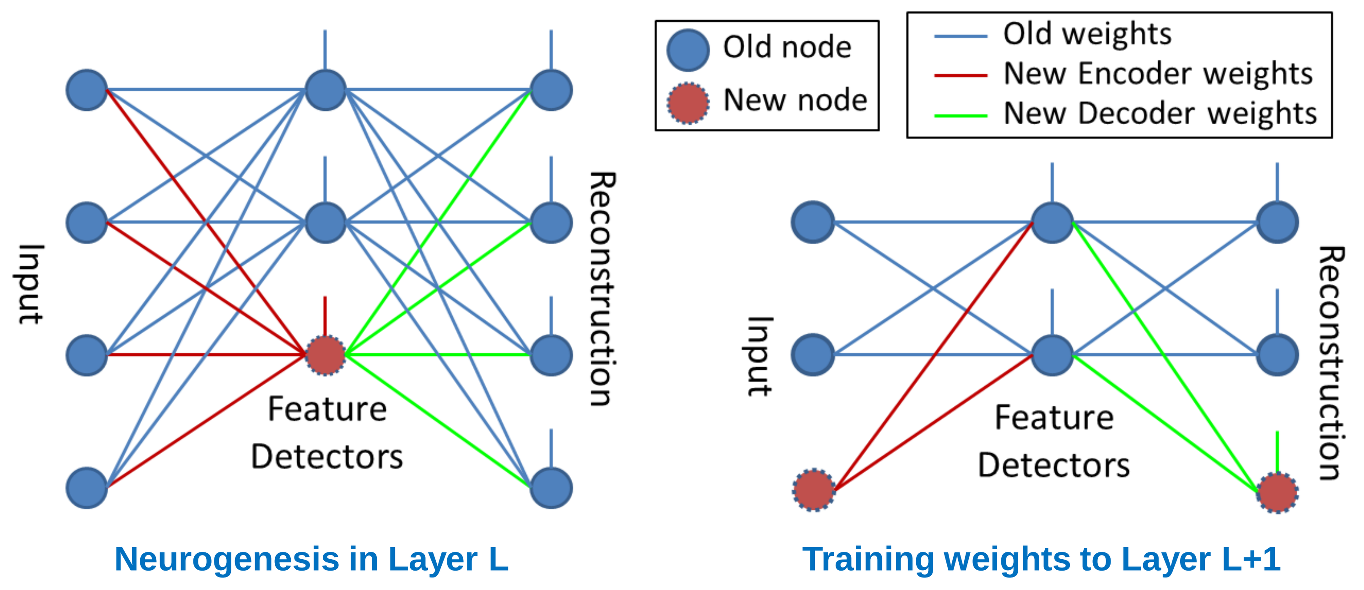
// Combine samples from the new class of data with replayed samples of old data
*AE_TrainingSamples* ← $\{D_U$ U IntrinsicReplay$(D_1, D_2, \ldots, D_{U-1})\}$

// Perform neurogenesis layer by layer
*numOutliers* ← $|D_U|$
for *Layer L* ← 1 to *N*
  *NewNodes* ← 0
  *Outliers* ← $\{d \ni D_U \mid RE_{Global,L}(d) > Th_L\}$
  $N_{Out}$ ← |*Outliers*|
  $N_{Out}^{Prev}$ ← $N_{Out}$+1

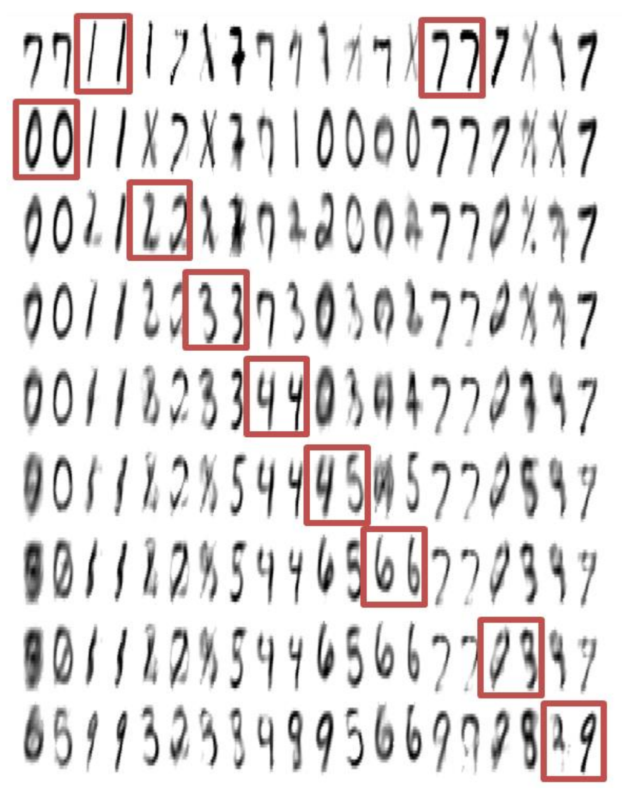
  // Add and train new nodes to layer *L*
  while $N_{Out}$ > *MaxOutliers* and *NewNodes* < $MaxNodes_L$ and $N_{Out} < N_{Out}^{Prev}$
    $AE_L$ ← $(W_L, b_L ; W'_{N+1-L}, b'_{N+1-L})$ from *AE*
    *Plasticity*: Add a node with random weights to $AE_L$ and train on *Outliers*
      Use *LR* to update encoder weights connected into new node only
      Use *LR*/100 to update decoder weights
    *Stability*: Train $AE_L$ on *AE_TrainingSamples*
      Using *LR* to update all weights
    $(W_L, b_L ; W'_{N+1-L}, b'_{N+1-L})$ ← $AE_L$
    *Outliers* ← $\{d \ni D_U \mid RE_{Global,L}(d) > Th_L\}$
    $N_{Out}^{Prev}$ ← $N_{Out}$
    $N_{Out}$ ← |*Outliers*|
    *NewNodes* ← *NewNodes*+1

  // Add connections from new nodes in layer *L* to existing nodes in layer *L*+1 and train
  If *NewNodes* > 0 & *L* < *N*
    *Plasticity*: Add weights initialized to zero to $AE_{L+1}$ connected to new nodes from layer *L*
    *Stability*: Train $AE_{L+1}$ on *AE_TrainingSamples*
    $(W_L, b_L ; W'_{N+1-L}, b'_{N+1-L})$ ← $AE_L$

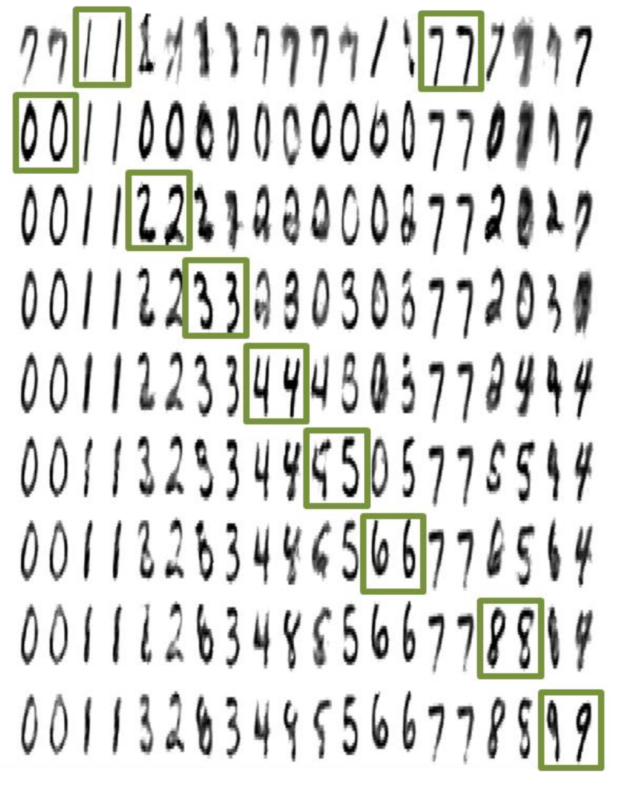
**Greedy Layerwise Neurogenesis on an Autoencoder. The goal is to learn new feature detectors for novel data.**




**Online Learning with Intrinsic Replay**



**NDL with Intrinsic Replay**



Reconstructions of all digits by AE pre-trained on digits 1 and 7' networks and progressive learning of new classes.

## Intrinsic Replay (IR) Algorithm

**The hippocampus region of the brain is known to "replay" experienced activity to aid the consolidation of newly acquired information into memory.**

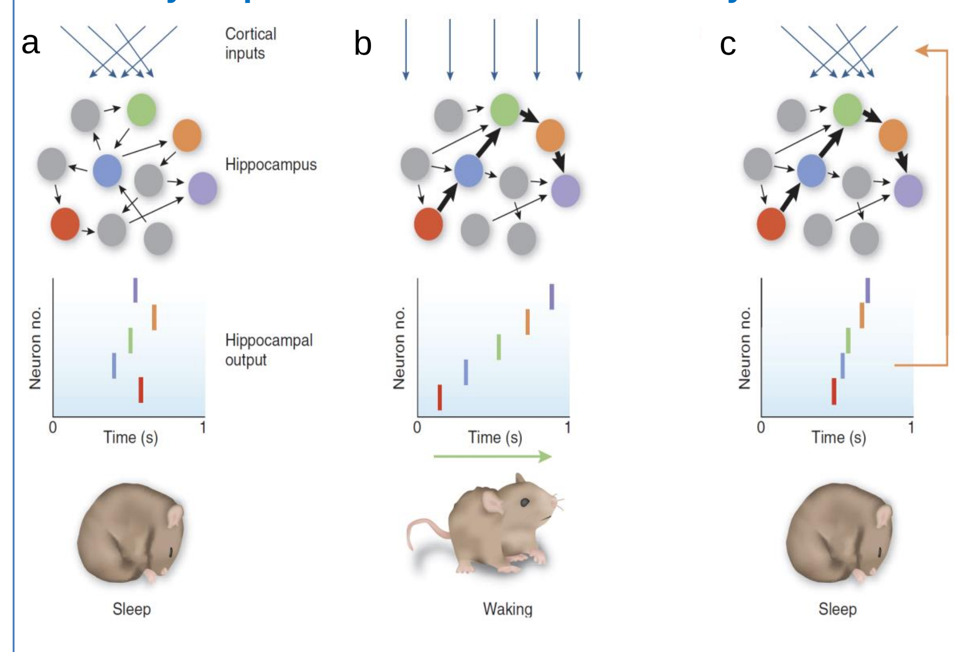
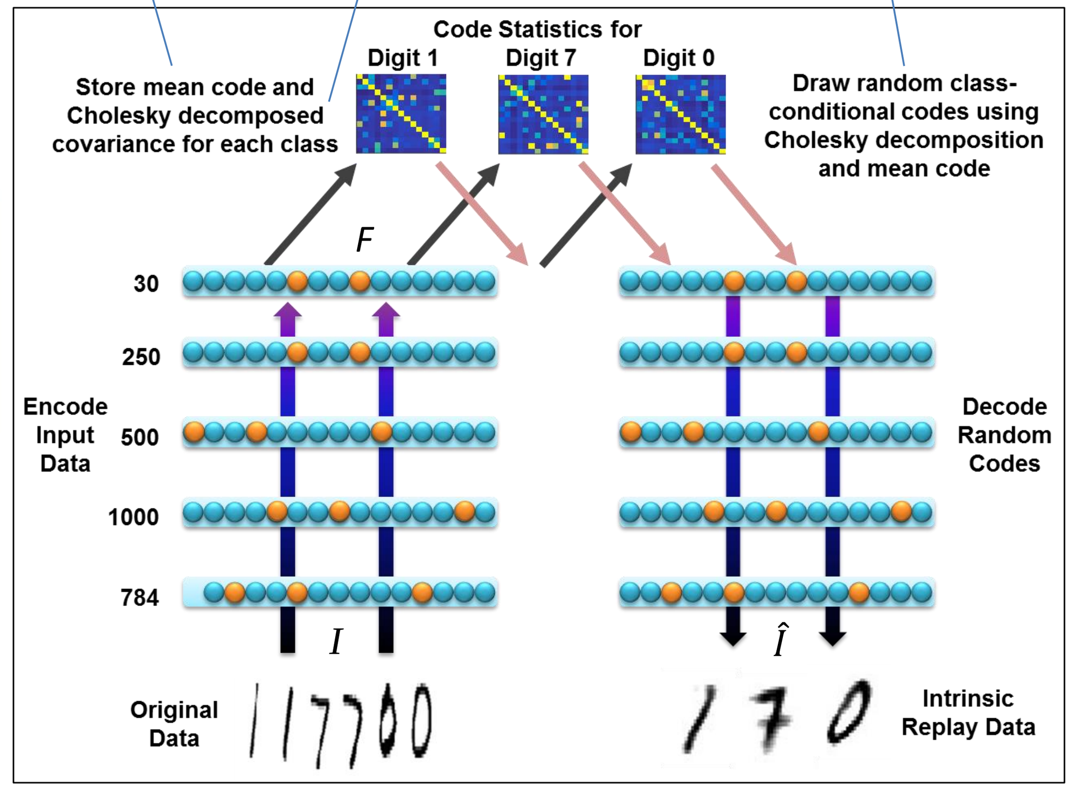



Figure from Mehta et al., Nature Neuroscience 2007

$\mu_F$ = Mean($F$), $Ch_F$ = Chol(Cov($F$))   $\hat{I}$ = Decode($\mu_F$ + N(0,1) * $Ch_F$)




Digits sampled via Intrinsic Replay from a trained 784-1000-500-250-30-250-500-1000-784 autoencoder.

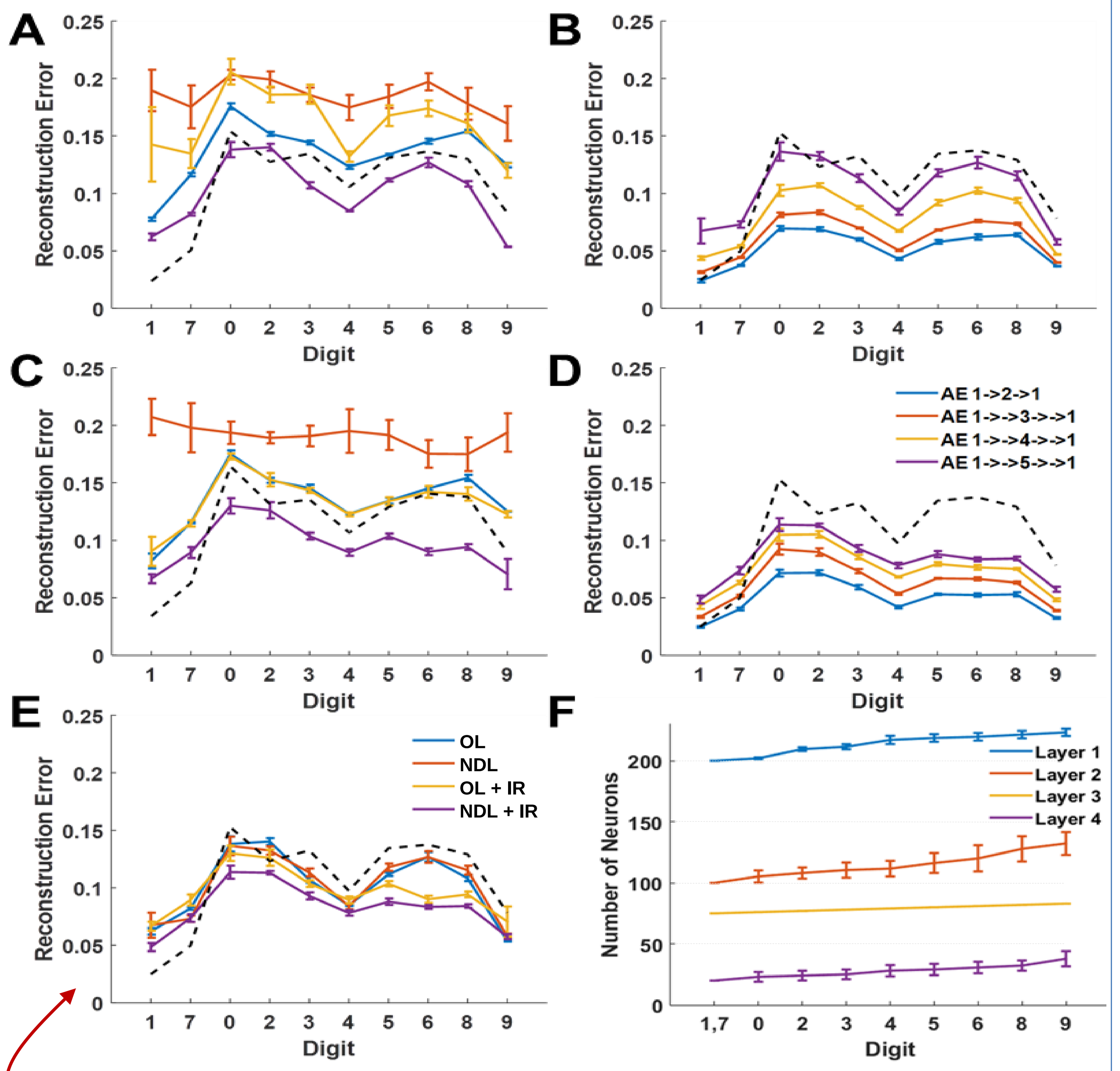
## RESULTS

We evaluate NDL on MNIST data, where a Deep AE is initially trained with two digits (1, 7), then learning a new task is simulated by progressively expanding the number of encountered classes (0, 2, 3, 4, 5, 6, 8, 9), one at a time. For each experiment, all training samples in a class are presented at once. For OL networks (A & C), the entire AE is retrained as each new class of data is presented.

- NDL with IR (NDL+IR) - Figure **D**: Starting network size of 784-200-100-75-20-75-100-200-784
- OL with IR (OL+IR) - Figure **C** below: Starting network size = NDL+IR generated network
- NDL without IR (NDL) - Figure **B** below: Starting network size of 784-200-100-75-20-75-100-200-784
- OL without IR (OL) - Figure **A** below: Starting network size = NDL generated network

Our results who that NDL with IR enables training of new digits while minimally impairing original representations.

- NG+IR outperforms OL not only overall, but in both the ability to represent the new data as well as preserving the ability to represent previously trained digits.
- OL+IR performs well on new digits, but poorly on retaining original old digits, whereas the NG+IR process does well on all digits.




*A-E*: Average REs of trained AEs after exposure to all 10 digits (legend in Plot D applies to Plots A, B, & C; dotted line shows REs of the original AE trained just on 1 and 7).
*E*: RE Comparison of the output of each full AE showing the superior performance of NDL+IR.
*F*: Neurogenesis contribution to network size in NDL+IR networks

## FUTURE WORK

- Apply Neurogenic Deep Learning to Convolutional Neural Nets
  - Add new filters via neurogenesis
  - Test classification performance
- Apply to streaming data to address concept drift

This work was supported by Sandia National Laboratories' Laboratory Directed Research and Development (LDRD) Program under the Hardware Acceleration of Adaptive Neural Algorithms Grand Challenge. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000



* tjdrael@sandia.gov // Dynamic Monitoring Software Department