Dolphin: A Programmable Framework for Scalable Neurosymbolic Learning

Aaditya Naik, Jason Liu, Claire Wang, Amish Sethi, Saikat Dutta, Mayur Naik, Eric Wong

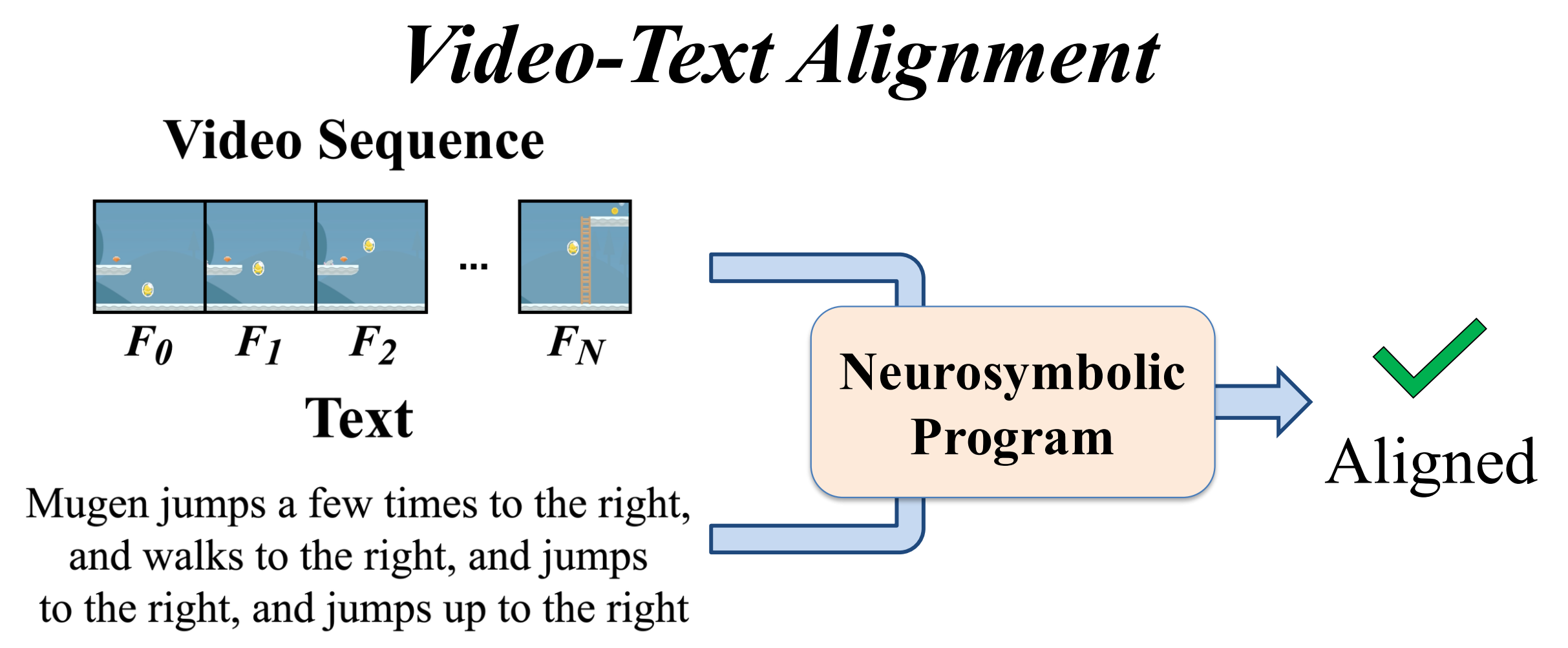
Motivation: Scaling Neurosymbolic Programming

Task

Training Time

with Scallop

Video-Text Alignment


Video Sequence

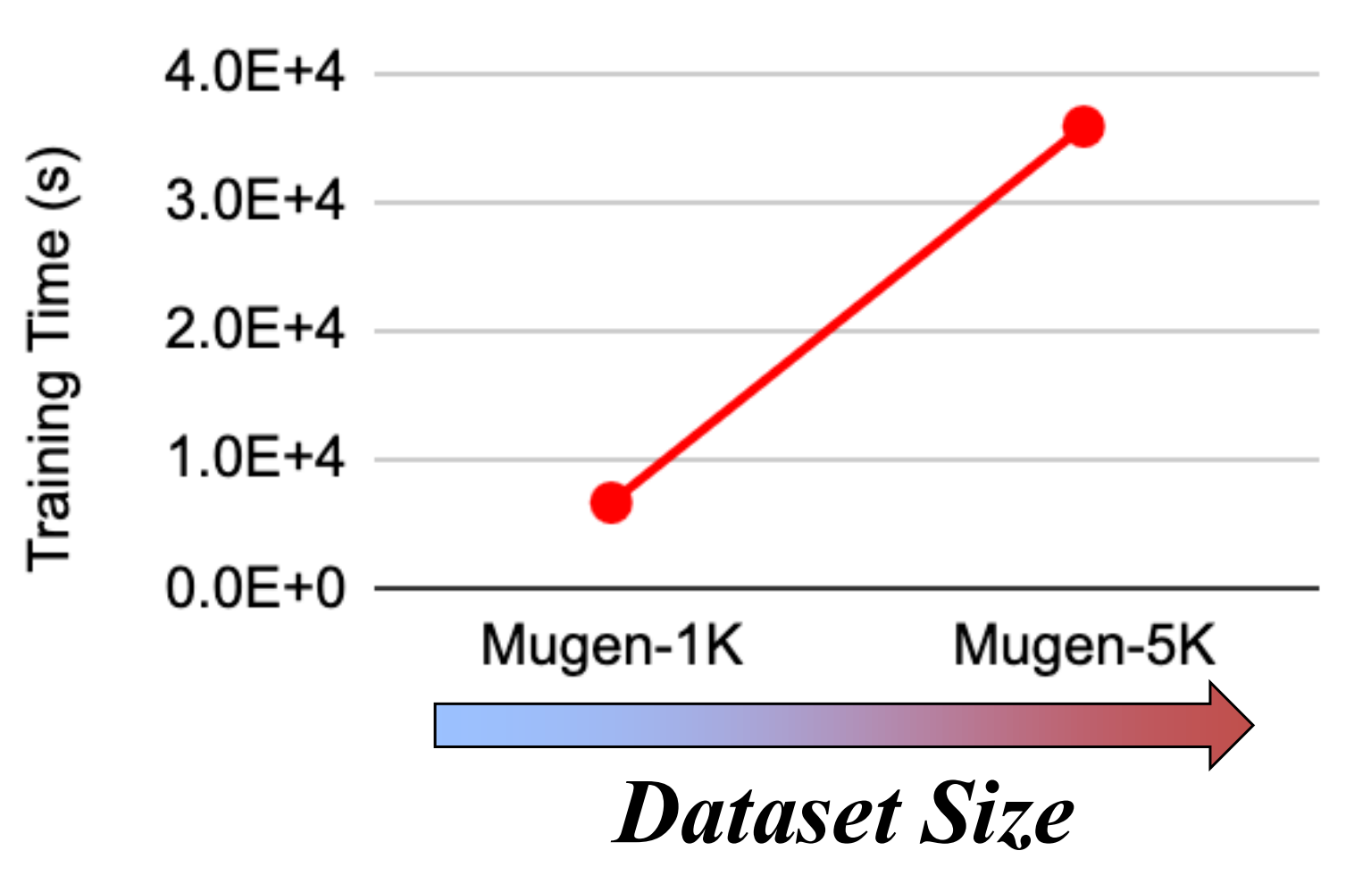
F_0 F_1 F_2 ... F_N

Text

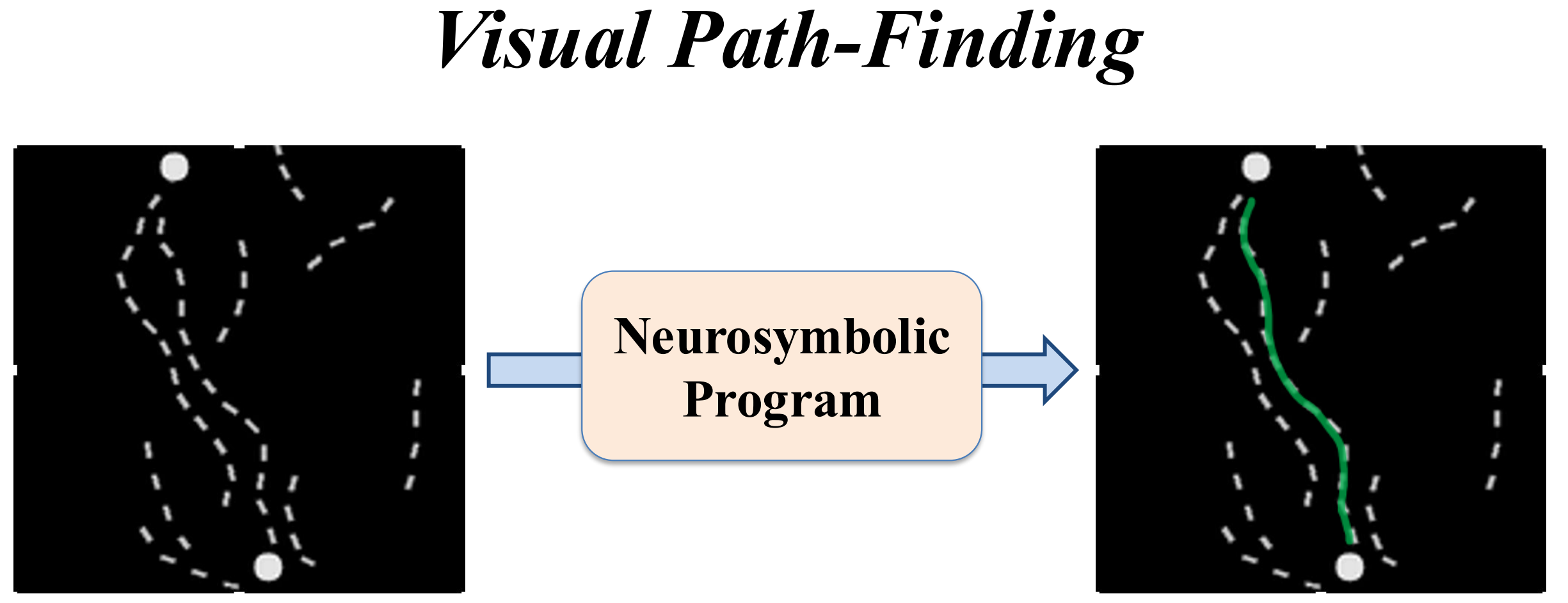
Mugen jumps a few times to the right, and walks to the right, and jumps to the right, and jumps up to the right

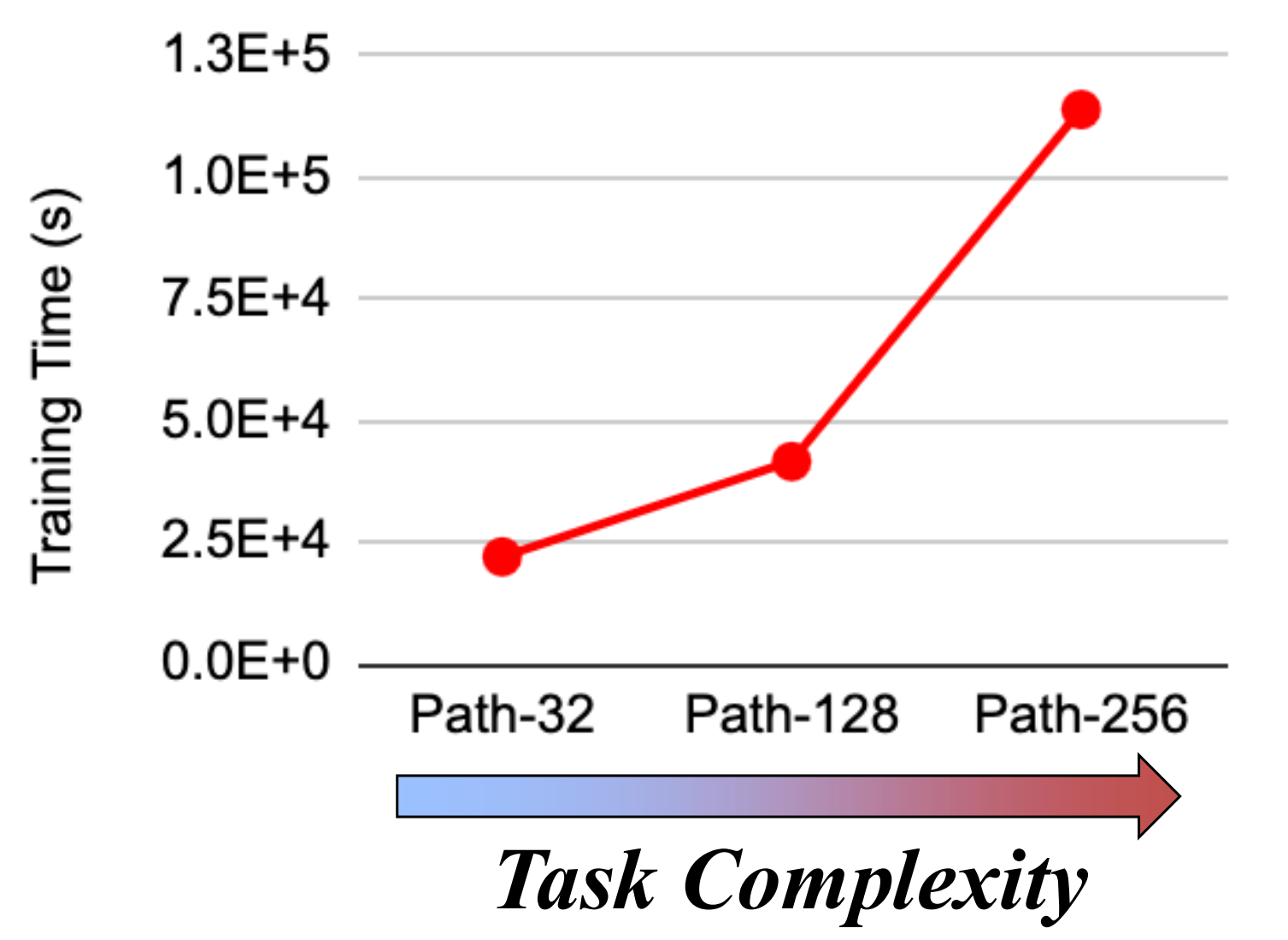
Neurosymbolic Program

Aligned



Visual Path-Finding


Neurosymbolic Program



An Example Dolphin Program

Task: Add N MNIST Digits

```
class SumNNet(torch.nn.Module):
  def __init__(self):
    super(SumNNet, self).__init__()
    self.CNN = MNISTNet()

  def forward(self, imgs):
    d = range(10)
    D_res = Distribution(self.CNN(imgs[0]), d)
    for i in range(1, len(imgs)):
      D_i = Distribution(self.CNN(imgs[i]), d)
      D_res = apply(D_res, D_i, lambda x,y: x + y)
    return get_logits(D_res)
```

Tight integration with PyTorch

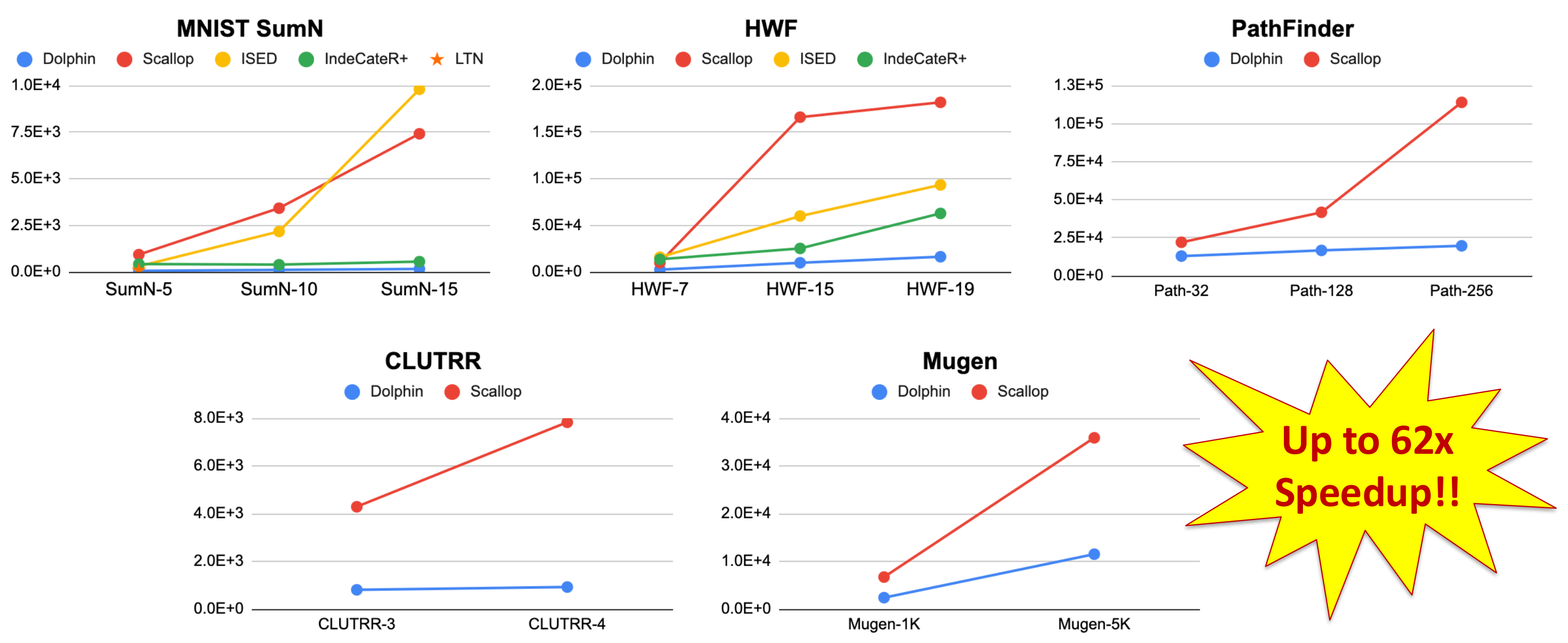
Intuitive Pythonic Primitives

Recursion, Control-Flow, Blackbox Functions

GPU-driven Vectorized Probablistic Computations

Experimental Results

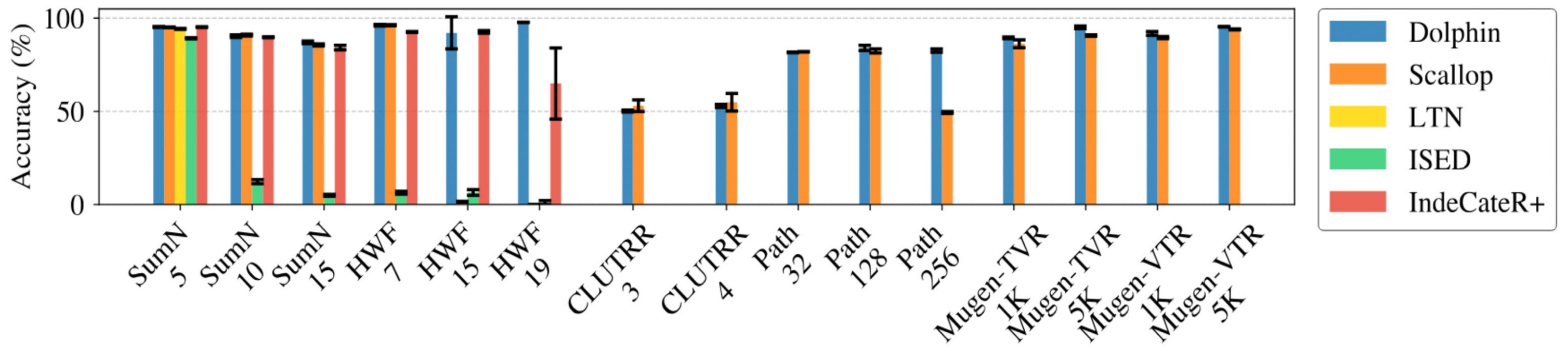
Dolphin Speeds up Training



Up to 62x Speedup!!

Dolphin Training Scales Better as Complexity Grows

Dolphin achieves SOTA Accuracies



Dolphin Splits Execution over CPU and GPU to allow Scaling

Backpropagation

Video Sequence

F_0 F_1 F_2 ... F_N

Text

Mugen jumps a few times to the right, and walks to the right, and jumps to the right, and jumps up to the right

Frame Action Classifier

LLM for Sequential Parsing

X_0	crouch	stand	...	jump
P_0	0.10	0.07	...	0.39
...				
X_N	crouch	stand	...	jump
P_N	0.00	0.03	...	0.64

jump	walk	...	jump
A_0	A_1		A_M

$\mathcal{L}$

$$\mathtt{m}(A_i, j, j) := (x_j = A_i)$$
$$\mathtt{m}(A_i, j, k) := \mathtt{m}(A_i, j+1, k) \wedge \mathtt{m}(A_i, j, j)$$
$$\ldots$$
$$\mathtt{Y}_{al}(\overline{A}, 1, N) := \mathtt{m}(A_0, 1, j_1) \wedge \mathtt{m}(A_1, j_1+1, j_2) \wedge \ldots \mathtt{m}(A_M, j_k+1, N)$$

CPU

$\mathcal{P}$

$$P(\mathtt{m}(A_i, j, j)) = P_j(x_j = A_i)$$
$$P(\mathtt{m}(A_i, j, k)) = P(\mathtt{m}(A_i, j+1, k)) \times P(\mathtt{m}(A_i, j, j))$$
$$P(\mathtt{Y}_{al}(\overline{A}, 1, N)) = \sum_{\forall j_1 \cdots j_k} P(\mathtt{m}(A_0, 1, j_1)) \times \ldots \times P(\mathtt{m}(A_M, j_k, N))$$

GPU

Symbolic Program

f_{loss}

Y_{al}	True	False
P	0.82	0.18

Output

Target

Aligned ✓

Label

Paper

Code