Robust Online Deep Learning

Antonios Valkanas, Boris Oreshkin, Mark Coates

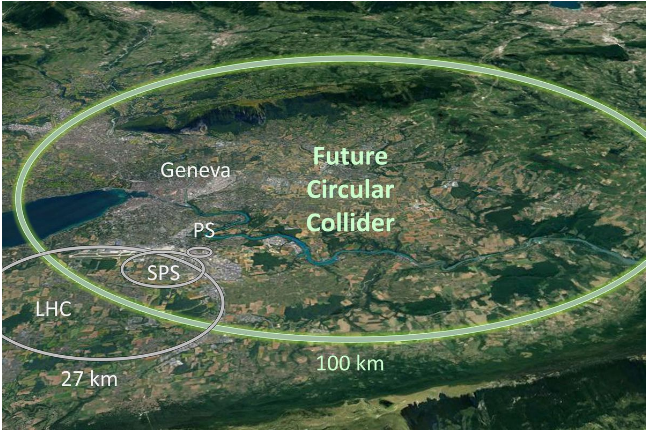
Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada

Bellairs Woskhop
January 25, 2024

Motivation

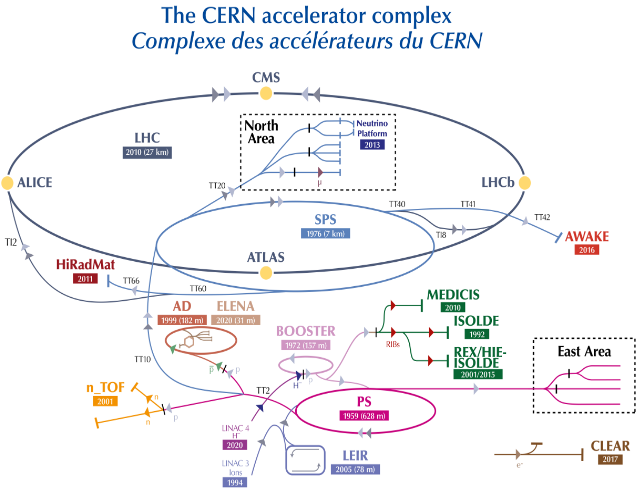
Online Learning (OL): When / Why?

- Data generating process is sequential
- Learning from massive data streams
- Edge device lifelong learning



Real world motivating example: CERN

- CERN collects $\geqslant$ 30 petabytes of data per year from LHC experiments
- Enough to fill about 1.2 million Blu-ray discs, 250 years of HD video.
- Only $\sim$100 petabytes of data permanently archived.



Example Application Areas

- Recommendation systems[1]
- Bandit Algorithms[2], Reinforcement learning
- Online Clustering (Human-robot interaction)[3]

[1] Y. Zheng, L. Siyi, Z. Li, S. Wu. "Cold-start sequential recommendation via meta learner", in Proc. AAAI, p. 4706-4713, 2021.

[2] Q. Zhang, Z. Deng, "Online Learning for Non-monotone DR-Submodular Maximization: From Full Information to Bandit Feedback", in Proc. AI-STATS, p. 3515-3537, 2023

[3] Y. Wang, J. Shen, S. Petridis, M. Pantic, "A real-time and unsupervised face re-identification system for human-robot interaction", Pattern Recognition, p. 559-568, 2019.

Problem Setup

A typical Online Learning process resembles this procedure:

Algorithm 1: Online Binary Classification process.

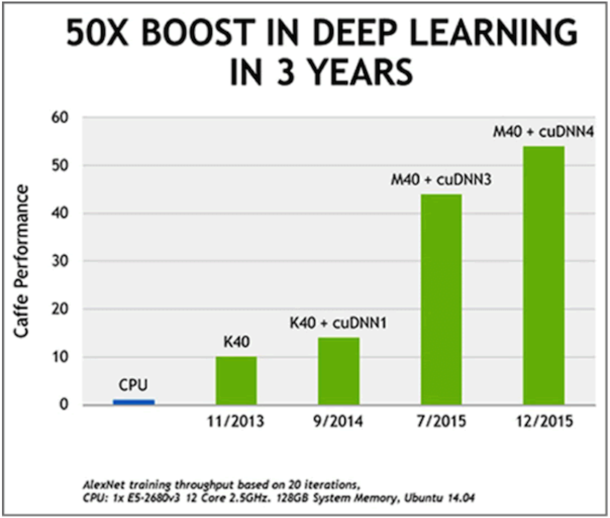
Initialize the prediction function as $\mathbf{w}_1$;
for $t = 1, 2, \ldots, T$ **do**
  Receive instance: $\mathbf{x}_t \in \mathbb{R}^d$;
  Predict $\widehat{y}_t = \mathrm{sign}(\mathbf{w}_t^\top \mathbf{x}_t)$ as the label of $\mathbf{x}_t$;
  Receive the true class label: $y_t \in \{-1, +1\}$;
  Suffer loss: $\ell_t(\mathbf{w}_t)$ which is a convex loss function on both $\mathbf{w}_t^\top \mathbf{x}_t$ and y_t;
  Update the prediction model $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$;
end for

- A metric is cumulative prediction error: $\epsilon_T = \sum_{t=1}^{T} c_t$, for some $c(\cdot)$.
- Example 0-1 loss: $\epsilon_T = \sum_{t=1}^{T} \mathbb{I}_{[\widehat{y}_t \neq y_t]}$.

Why is this even a problem?

- For many tasks such as (Bayesian) linear regression full batch vs. recursive solutions are equivalent.
- But in general batch size matters.
- Small batches provide better accuracy, but can be time consuming.
- Large batches lead to faster training.

Why is this even a problem?



Source: https://developer.nvidia.com/blog/production-deep-learning-nvidia-gpu-inference-engine/

A basic approach

Online Gradient Descent[4] treats the problem as regular learning via backpropagation with batch size 1.
Given input $\mathbf{x}_t \in \mathbb{R}$, L hidden layers $\mathbf{h}^{(i)}$ the output is defined recursively:

$$
\begin{aligned}
F(\mathbf{x}_t) &= \textsc{softmax}(\mathbf{W}^{(L+1)}\mathbf{h}^{(L)}) \quad \text{where} \\
h^{(l)} &= \sigma(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}) \quad \forall l \in \{1, \ldots, L\} \\
h^{(0)} &= \mathbf{x}_t
\end{aligned}
$$

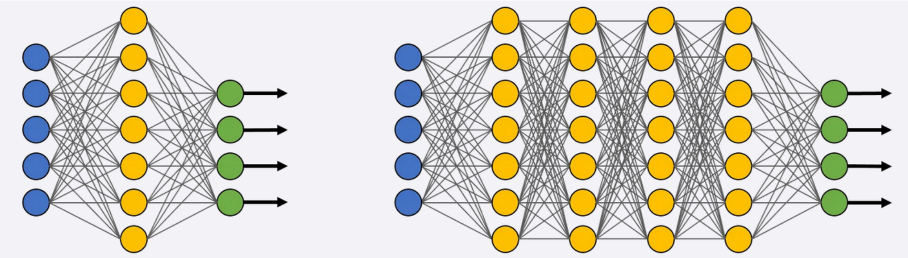
and the weights are updated by standard backpropagation:

$$\mathbf{W}^{(l)}_{t+1} = \mathbf{W}^{(l)}_t - \eta \nabla_{W^{(l)}_t} \mathcal{L}(F(\mathbf{x}_t), y_t)$$

[4] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent", in Proc. ICML p. 928-936, 2003.

Problems with Online Gradient Descent (OGD)

It is unclear which network will do better in an Online task.



On small datasets, deep net overfits, on large ones shallow net underfits.

Goals

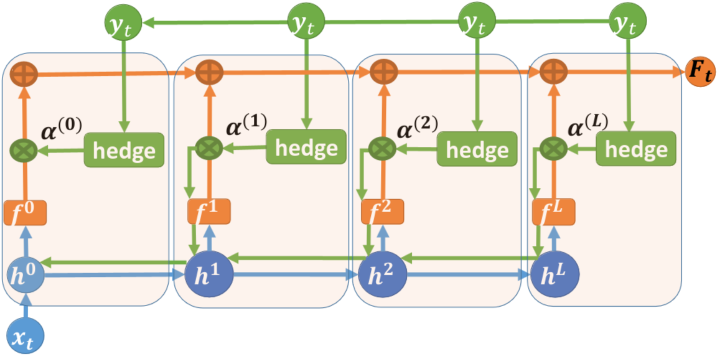
1. Architectural robustness

   $\longrightarrow$ to number of data points

   $\longrightarrow$ to missing features
2. Make GPUs useful again!
3. Improve performance metrics

Online Deep Learning (ODL)

ODL[5] learns the network topology jointly with $p(Y|X)$.



[5] D. Sahoo, Q. Pham, J. Lu, S. Hoi, "Online deep learning: learning deep neural networks on the fly", in Proc. IJCAI, p. 2660–2666, 2018.

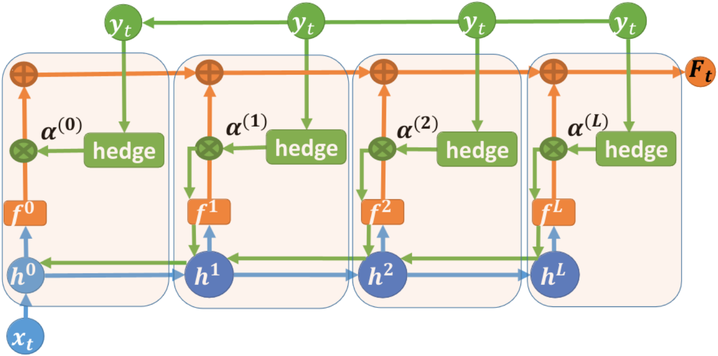
Online Deep Learning (ODL)

$$F(\mathbf{x}_t) = \sum_{l=0}^{L} \alpha^{(l)} f^{(l)}$$

$$f^{(l)} = \textsc{softmax}(\mathbf{h}^{(l)}\mathbf{\Theta}^{(l)}) \quad \forall l \in \{1, \ldots, L\}$$

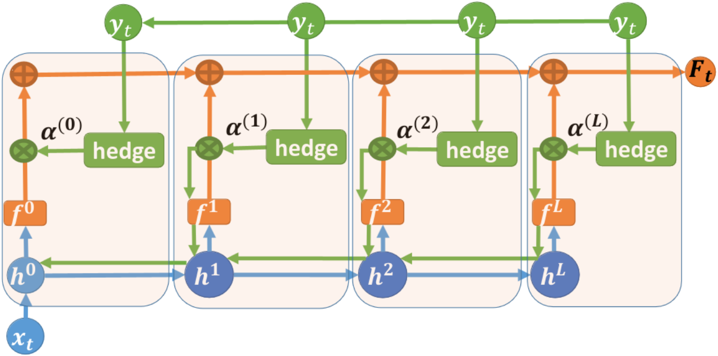
$$h^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}) \quad \forall l \in \{1, \ldots, L\}$$

$$h^{(0)} = \mathbf{x}_t$$



Online Deep Learning (ODL)

Gradient updates for weights, hedge[6] update for α.

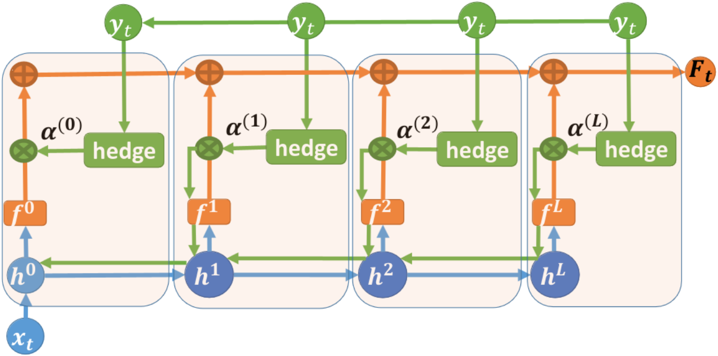


$$\alpha_{t+1}^{(l)} = \alpha_t^{(l)} \beta^{\mathcal{L}(\mathbf{x}_t, \mathbf{y}_t)}$$

[6] Y. Freund, R. Schapire, "A decision-theoretic generalization of on-line learn- ing and an application to boosting", J. Comp. Sys. sciences, 1997

Online Deep Learning (ODL)

Gradient updates for weights, hedge[7] update for α.

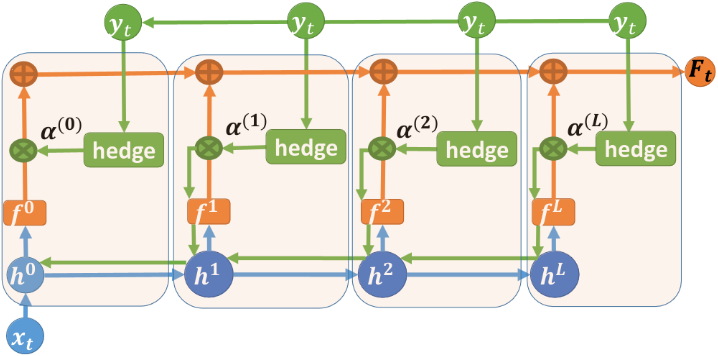


$$\alpha_{t+1}^{(l)} = \alpha_t^{(l)} \beta^{\mathcal{L}(\mathbf{x}_t, \mathbf{y}_t)}$$

[7] Y. Freund, R. Schapire, "A decision-theoretic generalization of on-line learn- ing and an application to boosting", J. Comp. Sys. sciences, 1997

Online Deep Learning (ODL)

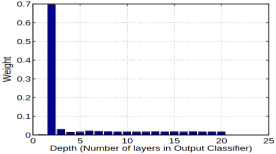
Gradient updates for weights, hedge update for α.



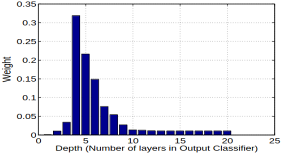
$$\mathbf{\Theta}_{t+1}^{(l)} = \mathbf{\Theta}_{t}^{(l)} - \eta \nabla_{\mathbf{\Theta}_{t}^{(l)}} \mathcal{L}(\mathbf{x}_t, \mathbf{y}_t)$$

$$\mathbf{W}_{t+1}^{(l)} = \mathbf{W}_{t}^{(l)} - \eta \sum_{j=l}^{L} \alpha^{(j)} \nabla_{\mathbf{W}_{t}^{(l)}} \mathcal{L}(\mathbf{x}_t, \mathbf{y}_t)$$

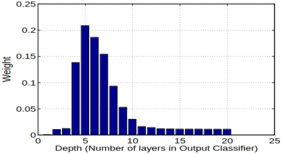
Learned Architectures



(a) First 0.5% of Data

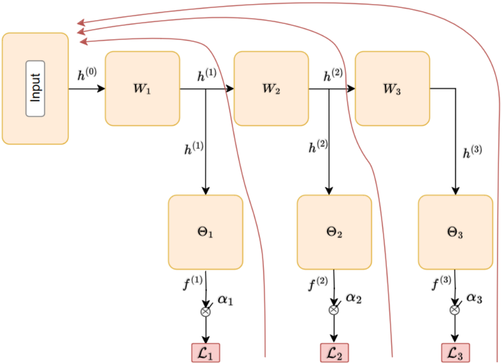


(b) 10-15% of Data

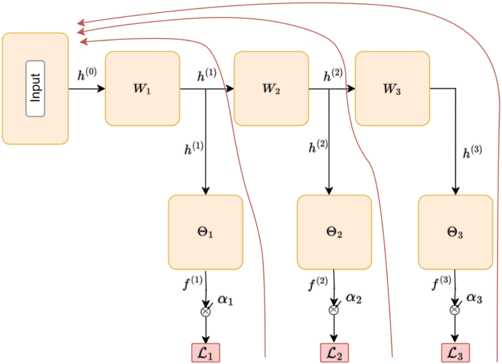


(c) 60-80% of Data

Computational issues



Computational issues

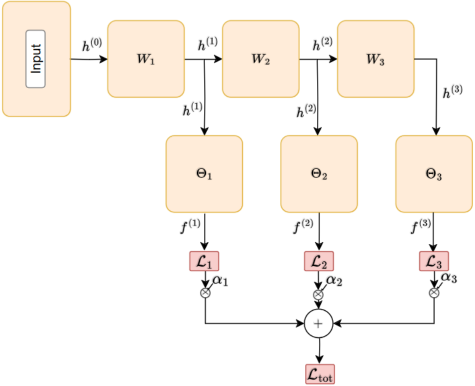


$$\mathbf{W}_{t+1}^{(3)} = \mathbf{W}_t^{(3)} - \eta\alpha_3 \frac{\partial \mathcal{L}_3}{\partial \mathbf{f}^{(3)}} \frac{\partial \mathbf{f}^{(3)}}{\partial \mathbf{h}^{(3)}} \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{W}_t^{(3)}}$$

$$\mathbf{W}_{t+1}^{(2)} = \mathbf{W}_t^{(2)} - \eta \left(\alpha_2 \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}^{(2)}} \frac{\partial \mathbf{f}^{(2)}}{\partial \mathbf{h}^{(2)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{W}_t^{(2)}} + \alpha_3 \frac{\partial \mathcal{L}_3}{\partial \mathbf{f}^{(3)}} \frac{\partial \mathbf{f}^{(3)}}{\partial \mathbf{h}^{(3)}} \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{W}_t^{(2)}} \right)$$

$$\mathbf{W}_{t+1}^{(1)} = \mathbf{W}_t^{(1)} - \eta \left(\alpha_1 \frac{\partial \mathcal{L}_1}{\partial \mathbf{f}^{(1)}} \frac{\partial \mathbf{f}^{(1)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{W}_t^{(1)}} + \alpha_2 \frac{\partial \mathcal{L}_2}{\partial \mathbf{f}^{(2)}} \frac{\partial \mathbf{f}^{(2)}}{\partial \mathbf{h}^{(2)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{W}_t^{(1)}} \right.$$
$$\left. + \alpha_3 \frac{\partial \mathcal{L}_3}{\partial \mathbf{f}^{(3)}} \frac{\partial \mathbf{f}^{(3)}}{\partial \mathbf{h}^{(3)}} \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \mathbf{W}_t^{(1)}} \right)$$

Computational issues



$$\mathbf{W}^{(3)}_{t+1} = \mathbf{W}^{(3)}_t - \eta\alpha_3 \frac{\partial\mathcal{L}_3}{\partial\mathbf{f}^{(3)}}\frac{\partial\mathbf{f}^{(3)}}{\partial\mathbf{h}^{(3)}}\frac{\partial\mathbf{h}^{(3)}}{\partial\mathbf{W}^{(3)}_t}$$

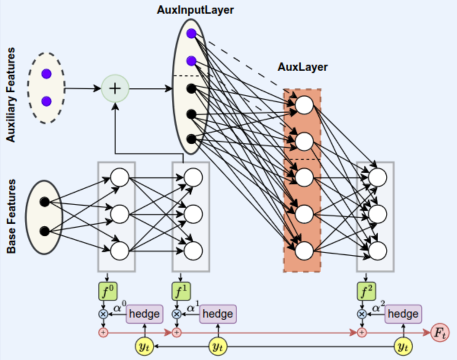
$$\mathbf{W}^{(2)}_{t+1} = \mathbf{W}^{(2)}_t - \eta\left(\alpha_2 \frac{\partial\mathcal{L}_2}{\partial\mathbf{f}^{(2)}}\frac{\partial\mathbf{f}^{(2)}}{\partial\mathbf{h}^{(2)}}\frac{\partial\mathbf{h}^{(2)}}{\partial\mathbf{W}^{(2)}_t} + \alpha_3 \frac{\partial\mathcal{L}_3}{\partial\mathbf{f}^{(3)}}\frac{\partial\mathbf{f}^{(3)}}{\partial\mathbf{h}^{(3)}}\frac{\partial\mathbf{h}^{(3)}}{\partial\mathbf{h}^{(2)}}\frac{\partial\mathbf{h}^{(2)}}{\partial\mathbf{W}^{(2)}_t}\right)$$

$$\mathbf{W}^{(1)}_{t+1} = \mathbf{W}^{(1)}_t - \eta\left(\alpha_1 \frac{\partial\mathcal{L}_1}{\partial\mathbf{f}^{(1)}}\frac{\partial\mathbf{f}^{(1)}}{\partial\mathbf{h}^{(1)}}\frac{\partial\mathbf{h}^{(1)}}{\partial\mathbf{W}^{(1)}_t} + \alpha_2 \frac{\partial\mathcal{L}_2}{\partial\mathbf{f}^{(2)}}\frac{\partial\mathbf{f}^{(2)}}{\partial\mathbf{h}^{(2)}}\frac{\partial\mathbf{h}^{(2)}}{\partial\mathbf{h}^{(1)}}\frac{\partial\mathbf{h}^{(1)}}{\partial\mathbf{W}^{(1)}_t} + \alpha_3 \frac{\partial\mathcal{L}_3}{\partial\mathbf{f}^{(3)}}\frac{\partial\mathbf{f}^{(3)}}{\partial\mathbf{h}^{(3)}}\frac{\partial\mathbf{h}^{(3)}}{\partial\mathbf{h}^{(2)}}\frac{\partial\mathbf{h}^{(2)}}{\partial\mathbf{h}^{(1)}}\frac{\partial\mathbf{h}^{(1)}}{\partial\mathbf{W}^{(1)}_t}\right)$$

Robustness to missing inputs

Missing inputs are prevalent in data streams. Existing solutions:

1. Wait and re-acquire
2. Deterministic Dropout (AuxDrop[8])
3. We propose treating input as a set.

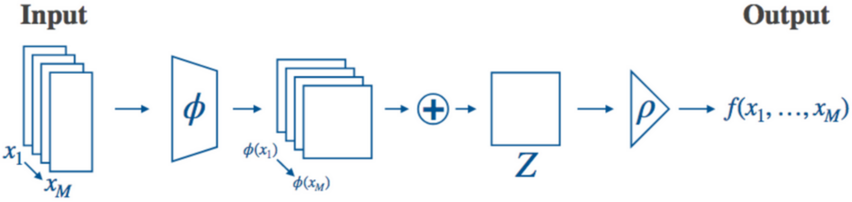


(b) Top: ODL, Down: Aux-Drop(ODL)

[8] R. Agarwal, D. Gupta, A. Horsch, D. Prasad, "Aux-Drop: Handling Haphazard Inputs in Online Learning Using Auxiliary Dropouts", Trans. Machine Learning Research, 2023

Input features as a set

- Consider treating $\mathbf{x}_t$ as a set $\mathcal{X}_t$ of its features.
- Suppose we have access to set of indices of $\mathbf{x}_t$, $\mathcal{I}_t = \{i_j, i_k, \ldots, i_z\}$
- Define inputs as $\mathbf{x}_{i_j} \leftarrow \text{CONCAT}[\mathbf{x}_{i_j}, \text{EMB}(i_j)]$
- Input dimension is $(B, M, F + E)$, output is (B, D)
- $f(\mathbf{x}_{i_j}, \mathbf{x}_{i_k}, \ldots, \mathbf{x}_{i_z}) = \rho\left(\sum_{i \in \mathcal{I}_t} \phi(\mathbf{x}_i)\right)$



Some Early results

Table: Comparison between AuxDrop+ODL and our (set + fast backprop method)

Dataset	Aux-Drop(ODL)	Ours	(HH:min:sec)
german	306.6 ± 9.1	305.9 ± 8.8	0:00:16 vs. 0:00:10
svmguide3	296.8 ± 1.3	296.8 ± 1.3	0:00:20 vs. 0:00:12
magic0	5571.9 ± 249.7	5578.7 ± 253.4	0:05:52 vs. 0:03:42
a8a	6914.2 ± 138.0	6914.35 ± 137.8	0:25:22 vs. 0:20:55
HIGGS $p = 0.2$	438442.2 ± 324.7	438434.8 ± 140.7	22:29:37 vs. 6:54:04
HIGGS $p = 0.5$	427484.6 ± 505.6	427443.2 ± 703.5	16:04:59 vs. 4:29:05
HIGGS $p = 0.8$	412504.2 ± 891.0	411891.2 ± 416.5	15:44:14 vs. 4:16:19
SUSY $p = 0.2$	274974.4 ± 936.1	274865.6 ± 960.2	12:57:26 vs. 6:50:16
SUSY $p = 0.5$	256994.8 ± 1220	256684.6 ± 1035.4	13:04:11 vs. 6:54:21
SUSY $p = 0.8$	237066.6 ± 742	237024.8 ± 736.8	22:28:09 vs. 6:53:59

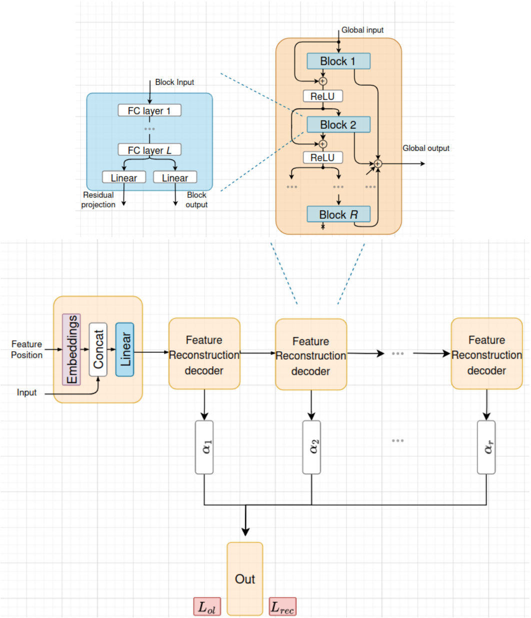
Current work: Making the GPU useful again

- By design we are limited to one training datum at a time.
- However, we still enable batch training by considering augmentations.
- For input $\mathbf{x}_t$, $\mathcal{I}_t = \{i_j, i_k, \ldots, i_z\}$ consider augmentations by sampling indices from power set of $\mathcal{I}_t$.
- Deterministic dropout approaches cannot do this.

Current work: A better base architecture

- Simple dense feed forward layers in ODL are not maximally expressive.
- Residual architectures with potential self attention mechanisms improve gradient flow and expressivity.

Current work: A better base architecture



Conclusion

- Architectural robustness ✓
  - $\longrightarrow$ to number of data points ✓
  - $\longrightarrow$ to missing features ✓
- Make GPUs useful again! ✓
- Improve performance metrics ✗(in progress)

Advertisement

- Motion in-filling via transformer models
- Presented in this workshop last year (recently accepted in IEEE Trans. Vis. Comp. Graphics)

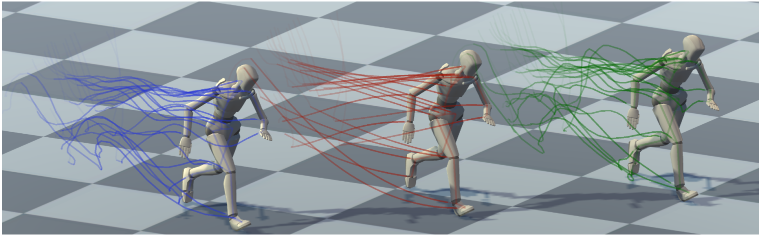


Figure: Animation generated from three different models, lines trace the position of skeletal joints.

Thank you :)