Capturing Emerging Complexity in Lenia

Sanyam Jain;
Aarati Shrestha;
Stefano Nichele.

Dept of Computer Science and Communication

Østfold University College & OsloMet University

Table of Contents

Complexity and Open-Endedness

Complexification

- In ALife, End Goal: How to solve a task? Solving a specific task is suboptimal. [Clune, 2019];
- Where Manual AI fails? Extremely difficult to evolve further to reuse parameters. [Stanley et al., 2017];
- Novelty Search, better than Task-based and Hybrid approaches [Stanley et al., 2017];
- Open-endedness and Complexification goes together [Randazzo and Mordvintsev, 2023];

Why open-endedness?

- Sometimes no task helps to evolve smart and interesting metrics over a task landascape.
- Endless variation implies complexification, any fixed size of complexity would exhasut existing interestingness or variation eventually;
- Fundamental questions: initial conditions, selection pressure, etc;
- Endless variation + Sensitivity to initial conditions + Selection pressure = Complexification
  [Randazzo and Mordvintsev, 2023]
  OR
  Open-endedness allows AI systems to continue to learn and improve over time, adapting to changing environments and evolving to meet new challenges.

Why Evolvability? [Randazzo and Mordvintsev, 2023]

- Heritable Genetics and Selectable Phenotype with variation;
- Without Evolvability there would be no discovery, no new behaviour;
- Dynamical task landscapes (autotelic or co-evolved), adaptive mutations (epigenetic autonomy and adaptations), novelty search (can be stochastic);

Lenia [Chan, 2018]

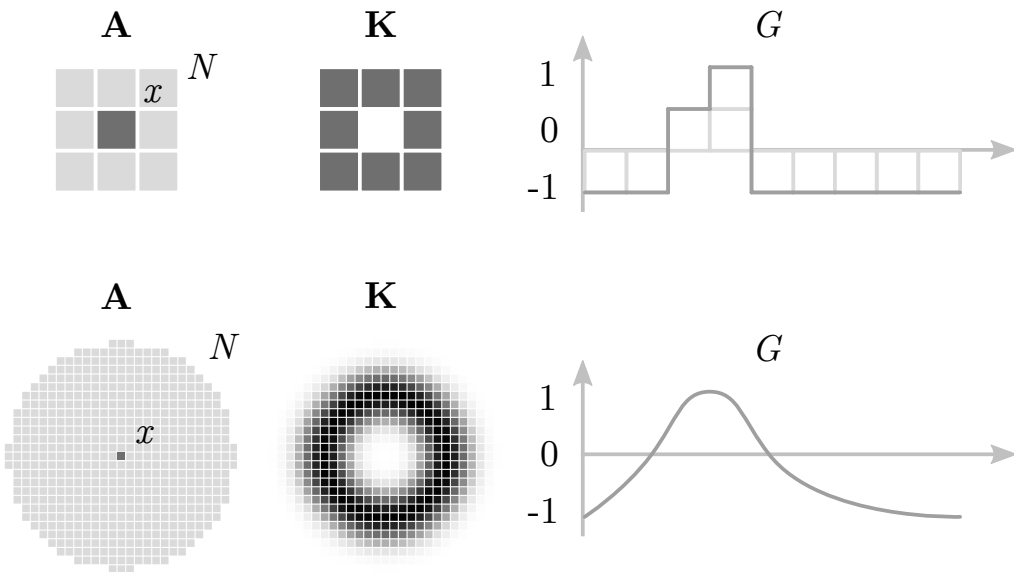


Figure: Discrete CA vs Continuous CA

Lenia Update Rule

Update function

The Lenia update rule is given by:

$$A_{t+1} = [A_t + \Delta t\, G(K * A_t)]$$

- A_t: Current State at t
- Δt: Step size
- G: Growth Function (for eg. Gaussian)
- K: Neighborhood Kernel
- $*$: Convolution operation

Kernel and Growth Function

Kernel Function

- Weighted importance to neighboring pixels and gradually reduced importance to distant pixels.
- Calculate the distances of each coordinate from the center and apply a mask to filter out values outside the desired radius.

Growth Function

- By adjusting the μ and σ parameters of the G, each cell's growth or decay can be controlled by taking input as Nh. sum array.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Emerging Complexity and Behaviour in Lenia using standard Genetic Algorithm

Fitness AE

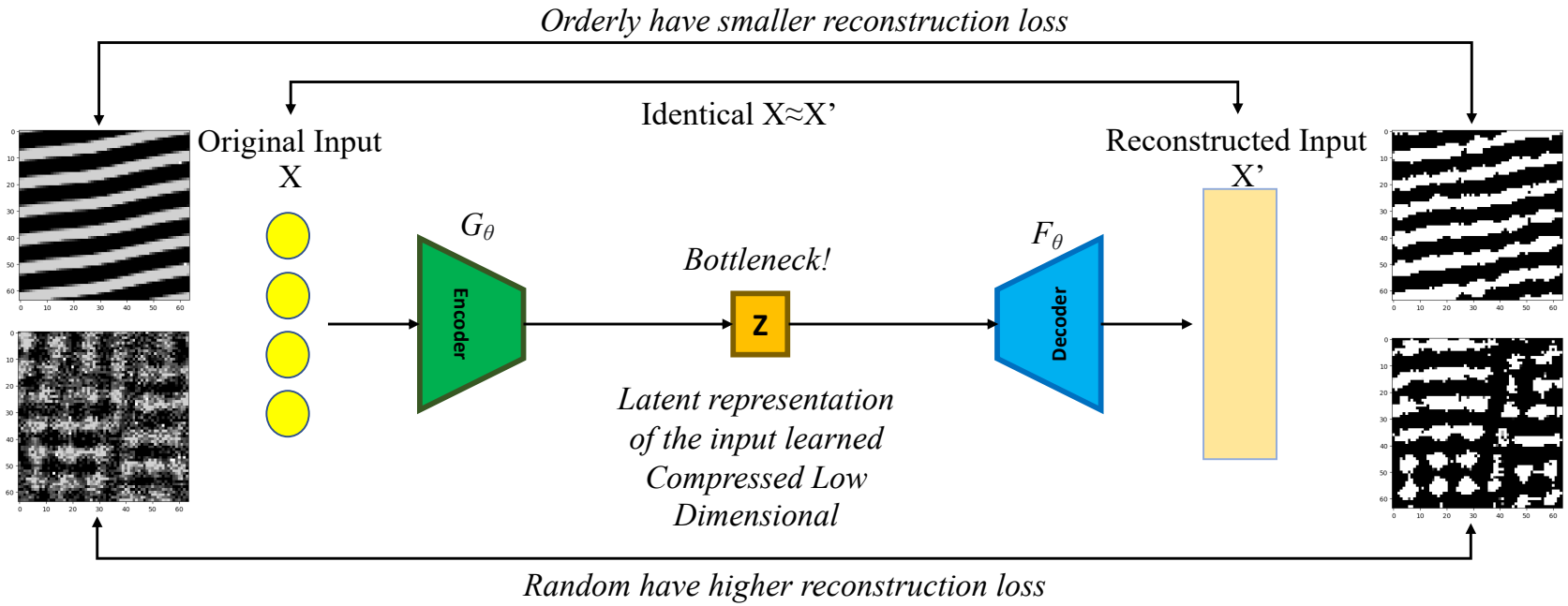


Figure: Compression based: AutoEncoder [Cisneros et al., 2019]

Fitness VoT

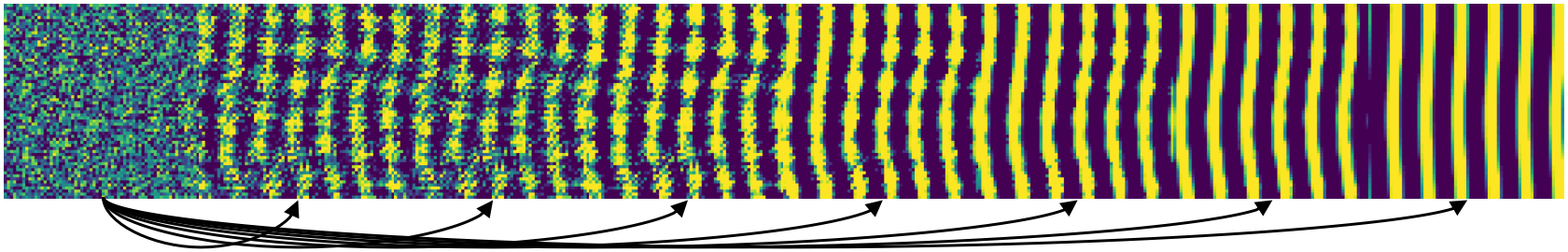


Figure: Variation based: Variation over Time

Fitness AEVoT

Algorithm 3. Auto-Encoder based Variation over Time (AEVoT)

Input : Input frames of Lenia patterns
Output: Population standard deviation (pstd) over list of alive cells count of each frame

Begin
  Reconstruct the original frames using an auto-encoder (AE):; **For** *each input frame* f **do**
    $f_{AE} = \mathrm{AE}(f)$; Calculate the number of alive cells in the reconstructed frame f_{AE} using a threshold:;
    $alive_f = count(p \geq threshold)$; Store the number of alive cells $alivef$ in a list;
  Calculate the population standard deviation (pstd) of the list of alive cells counts:; $mean = \frac{1}{n} \sum i = 1^n alivei$;
  $variance = \frac{1}{n} \sum i = 1^n (alive_i - mean)^2$; $pstd = \sqrt{variance}$;
  return *Population standard deviation (pstd) over list of alive cells count of each frame*;

Figure: AEVoT Based: Combined Approach

Selection, Crossover, and Mutation

- Roulette Wheel Selection
- No Crossover
- Mutation by perturbation

Results from different experiments for AE, VoT and AEVoT

VoT based Experiments

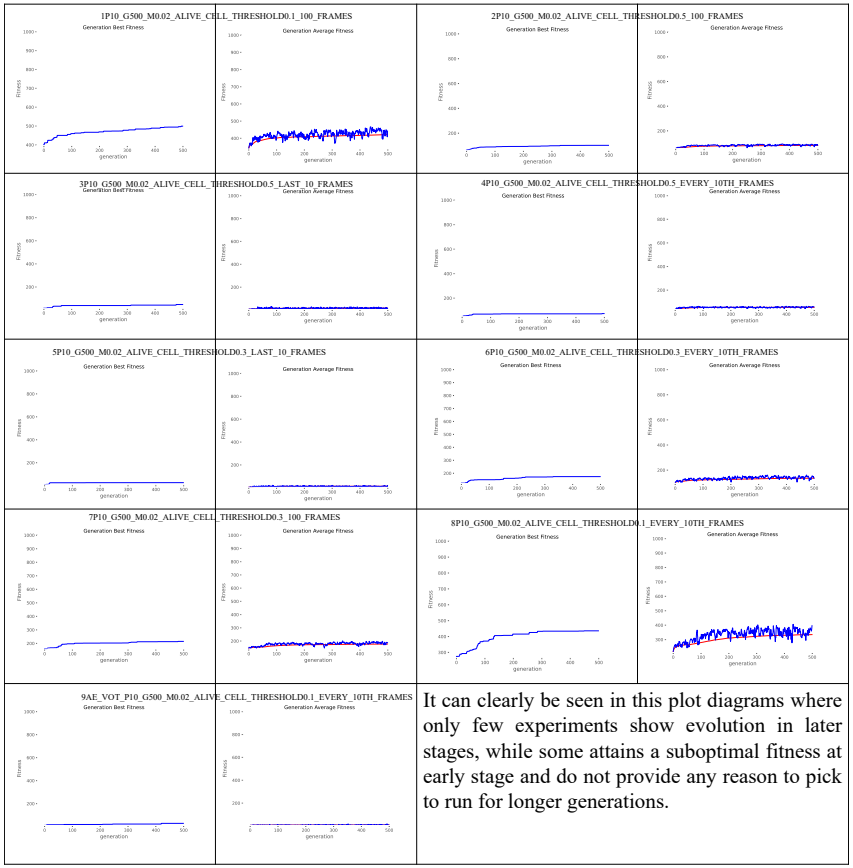


Figure: VoT experiments

AE based Experiments

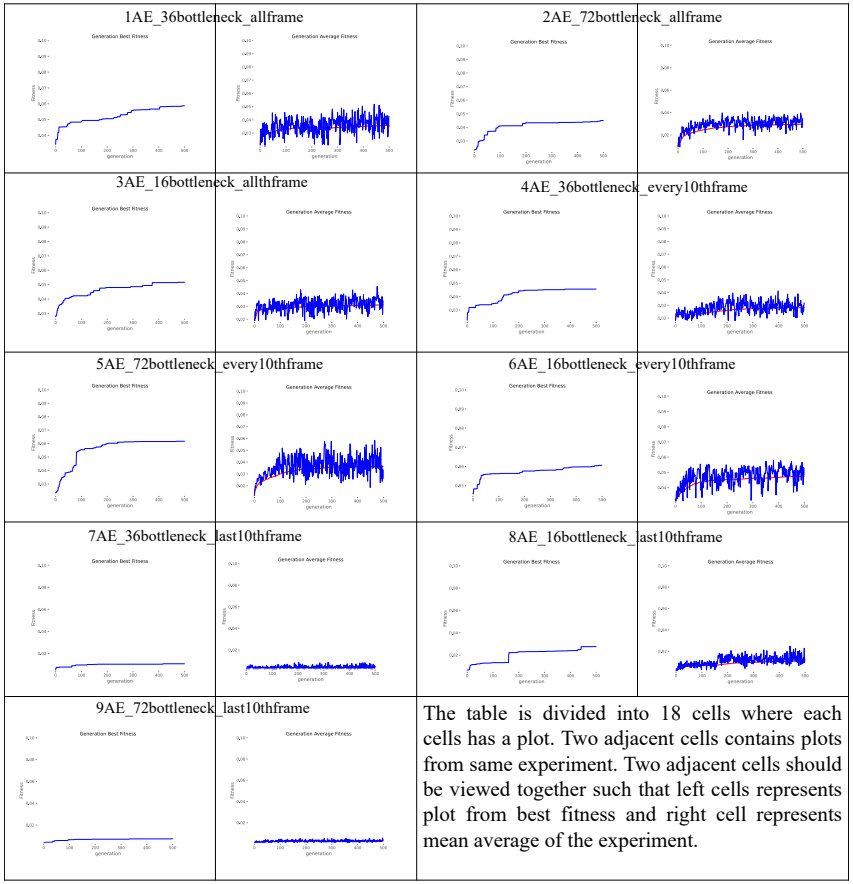


Figure: AE experiments

AEVoT based Experiments

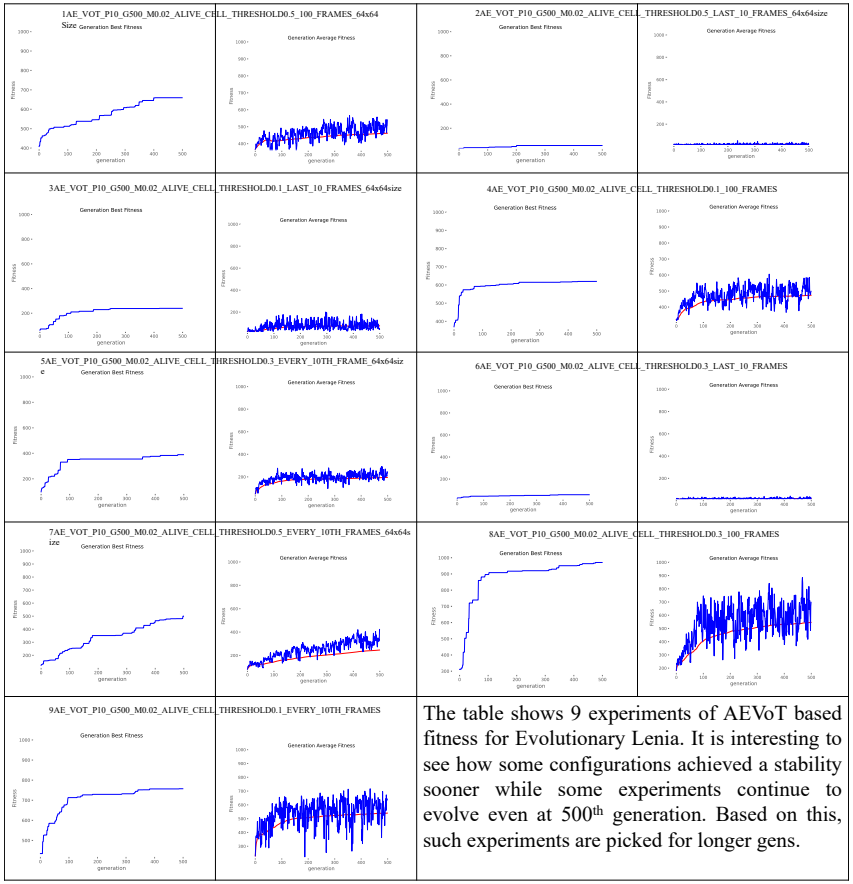


Figure: AEVoT experiments

AEVoT based Experiments: Known Kernel

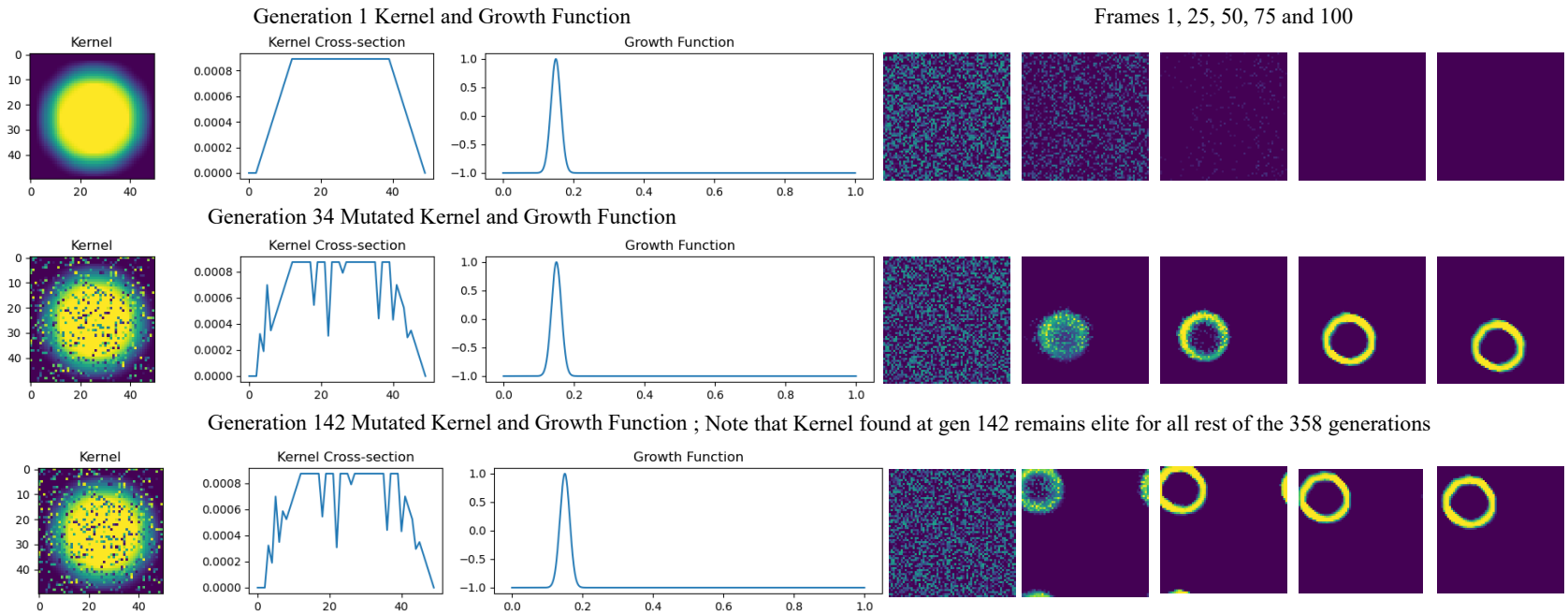


Figure: AEVoT experiment for known kernel: Adaptable mutations

Challenges, Learnings and Conclusion

- Automated Discovery, Emergent Agency, Open-Endedness [Chan, 2018];
- A strong hypothesis and supporting proofs for chosen parameters, yields good results (We discovered a ring forming bacteria);
- Evolution is time-taking!

Future Scope of Improvement

- Mutating Known Kernels.
- Particle Lenia, Flow Lenia, Sensorimotor Lenia
- Using JAX.

Major References

Chan, B. W.-C. (2018).
Lenia-biology of artificial life.
arXiv preprint arXiv:1812.05433.

Cisneros, H., Sivic, J., and Mikolov, T. (2019).
Evolving structures in complex systems.
In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 230–237. IEEE.

Clune, J. (2019).
Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence.
arXiv preprint arXiv:1905.10985.

Randazzo, E. and Mordvintsev, A. (2023).
Biomaker ca: a biome maker project using cellular automata.
arXiv preprint arXiv:2307.09320.

Stanley, K. O., Lehman, J., and Soros, L. (2017).
Open-endedness: The last grand challenge you've never heard of.
While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself.

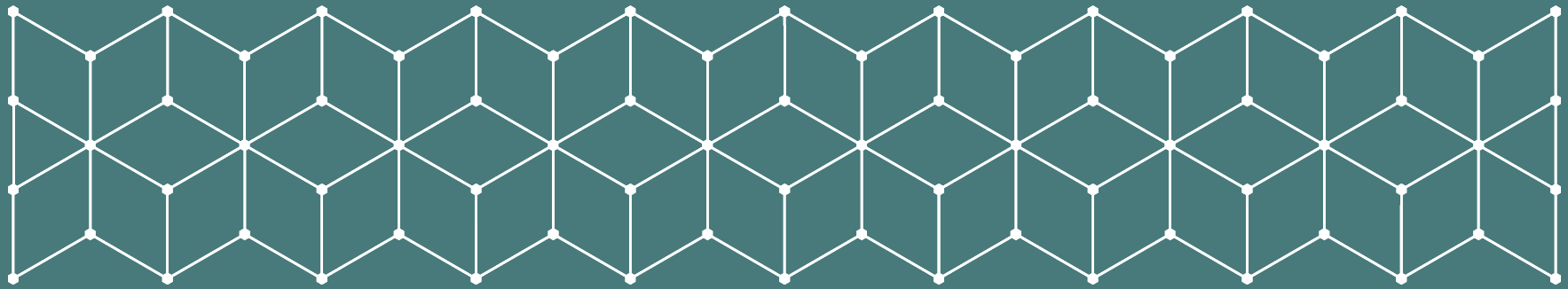
Play with it:

https://s4nyam.github.io/evolenia/

Østfold University College

ICAPAI
Human-Centric AI for the Digital Society

SVM at Edge: Low Cost Caching Prediction for Connected Edge Intelligence in Federated Machine Learning



About me

- Bachelor's (2015-19) @ UPES Dehradun, India
- Junior Research Fellowship (2020-22) @ IIT Jodhpur, India
- Masters Applied Computer Science (2022-current) @ ØUC

Contents

- Federated Machine Learning
- Caching
- SVM at Edge
- Results

Recap of Traditional Deep Learning

- Training objective: Minimize the Loss function! $\mathcal{L}(y, t)$

$$\min_{\theta} \mathcal{L}\,(y, t) = \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, t_i)$$

- Training Optimization: Relies on SGD or its variants, with mini-batches

$$W^{(l+1)} \leftarrow W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}}$$

Federated Learning – Training Objective

- Suppose "n" training samples are distributed to "k" clients,

$$Fed\,\mathcal{L}\,(y,t) = \sum_{k=1}^{K} \frac{n^k}{n} \mathcal{L}_k(y_k, t_k)$$

Where $\mathcal{L}_k(y_i, t_i) = \frac{1}{n^k} \sum_{i \in pool}^{K} l_k(y_i, t_i)$

sampling pool of Federated cycle is number of local devices

Federated Learning – Training Optimization using FedSGD

- One cycle of FedML has "*C/K*" fractions of client selected out of Total K clients.
  - *C/K = 1* implies Full batch GD
  - *C/K < 1* implies Stochastic GD

θ_{t+1} represents the global model parameters at time $t+1$.
$\theta_{k,t+1}$ represents the local model update for client k at time $t+1$.
n_k represents the number of training samples held by client k.
N represents the total number of training samples across all clients.

- FedSGD:

SGD Update: $\boldsymbol{\theta_k, t+1 \leftarrow \theta_k, t - \eta g_k, t}$

Central Aggregation: $\theta_{t+1} = \sum_{k=1}^{K} \theta_{k,t+1} \times \frac{n_k}{N}$

Caching

- Caching is an essential component of edge devices, as it enables these devices to store frequently used data and access it quickly, reducing latency and improving efficiency.
- However, caching on edge devices is a challenging task. These devices have limited storage capacity and are often connected to networks with limited bandwidth, making it difficult to predict which data should be stored in cache

SVM at Edge and Mimicking ANN

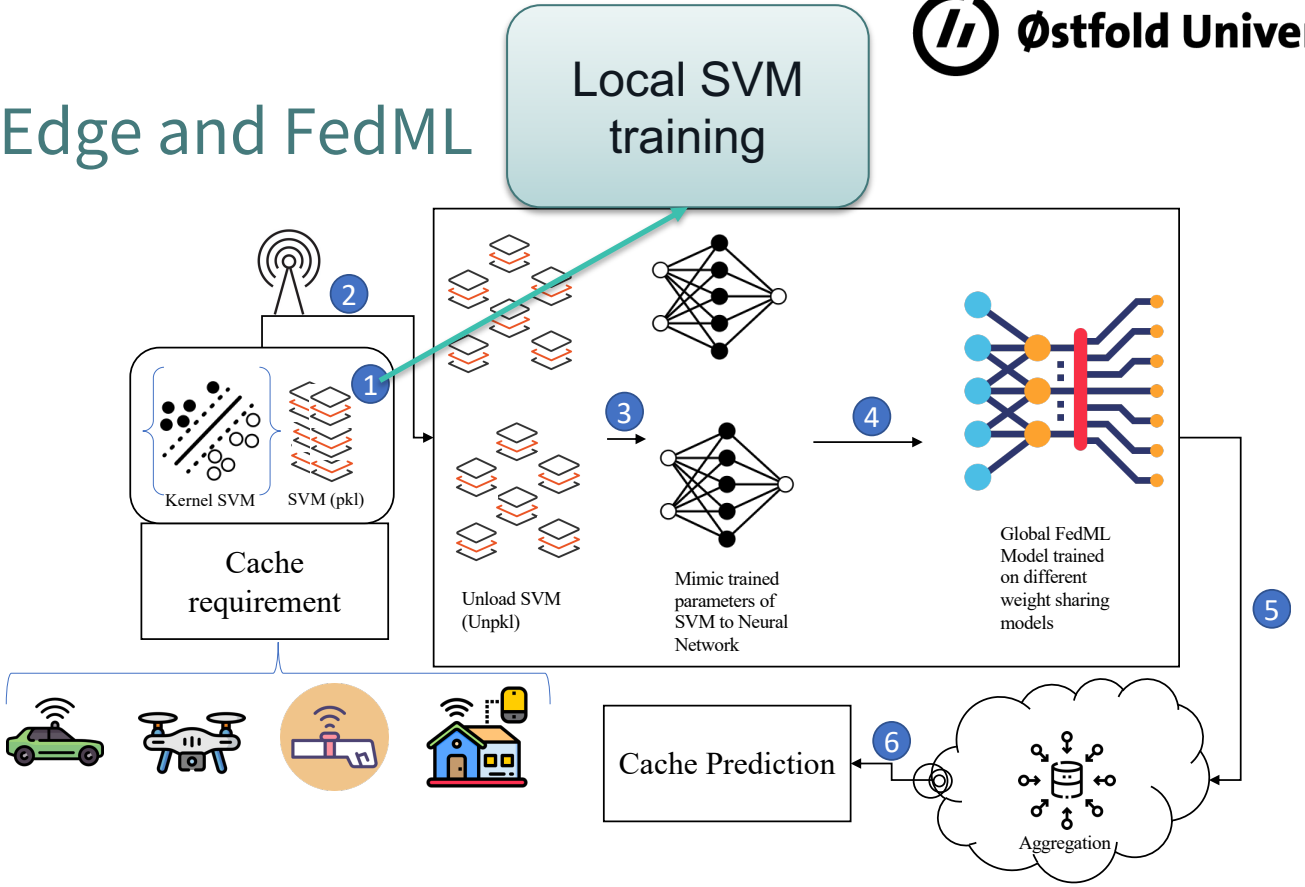
- SVM: Margin Error + Classification Error

$$\min_{w} |w|_2^2 + C \sum_{i=1}^{n} y_i (w^\intercal x_i + b) \geq 1$$

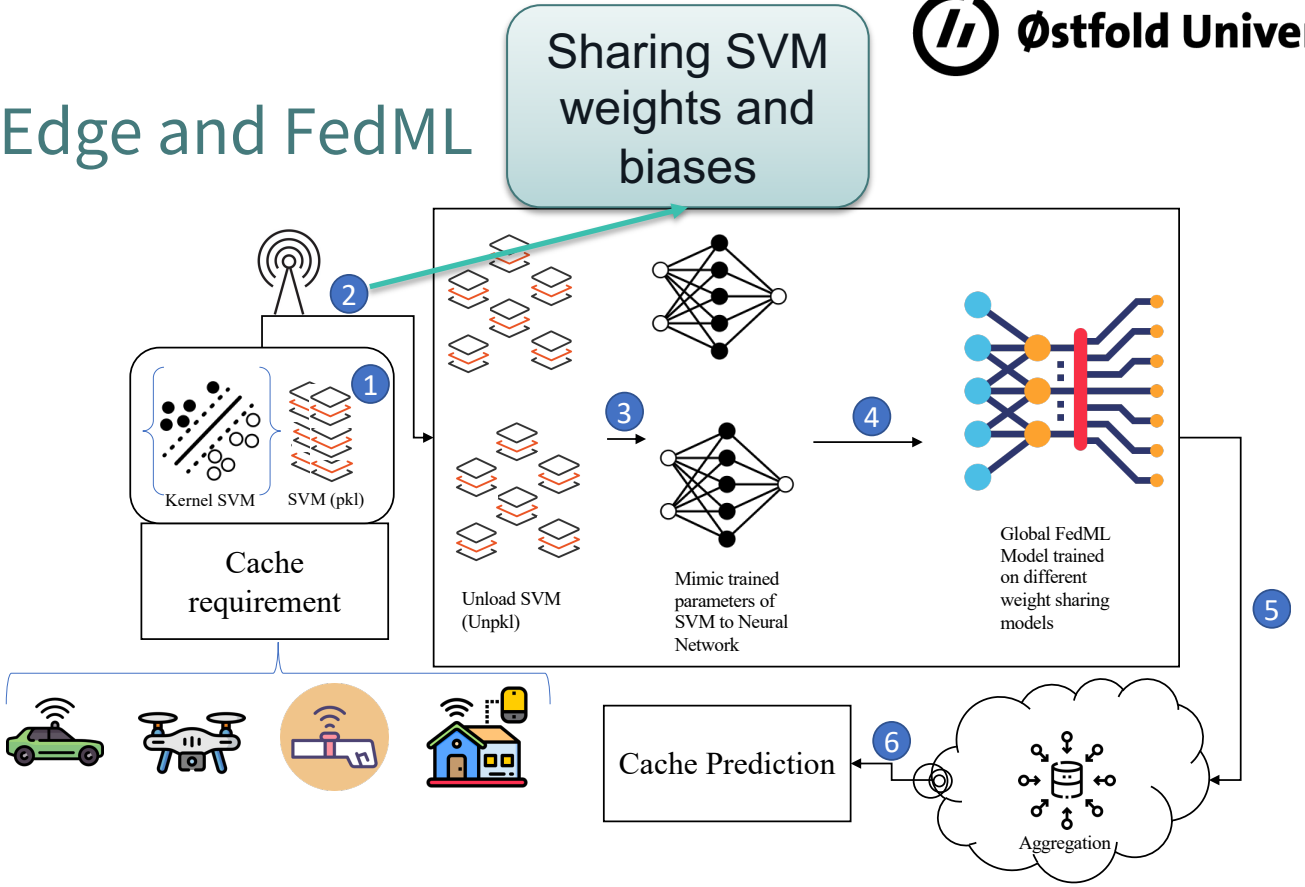
- Mimicking Neural Network: The predictions of trained SVM model works as training examples for the ANN. The idea is train ANN such that it starts giving same predictions as that of SVM.

- In Other words, weights and biases of trained SVM are assigned to weights and biases of randomly initialized ANN.

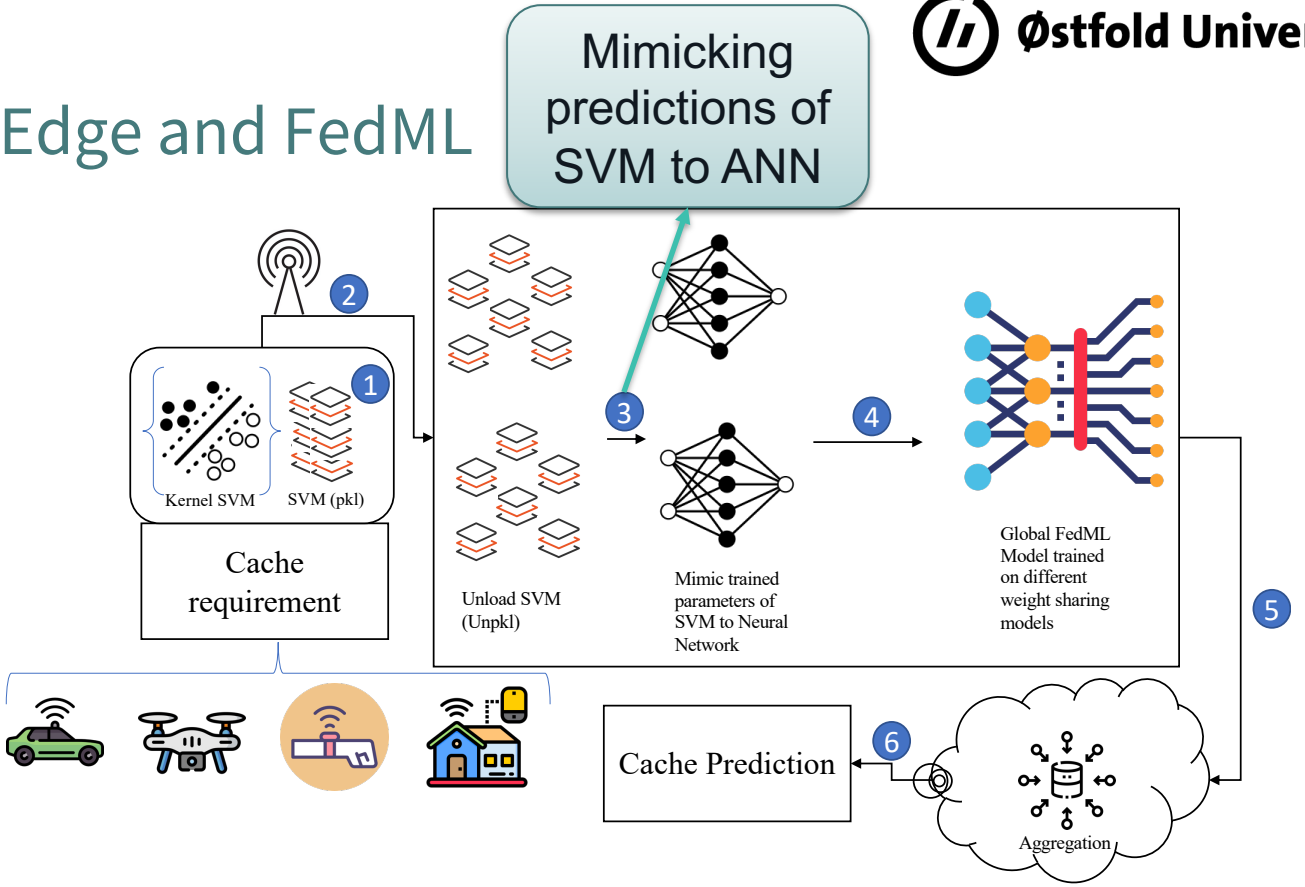
SVM at Edge and FedML



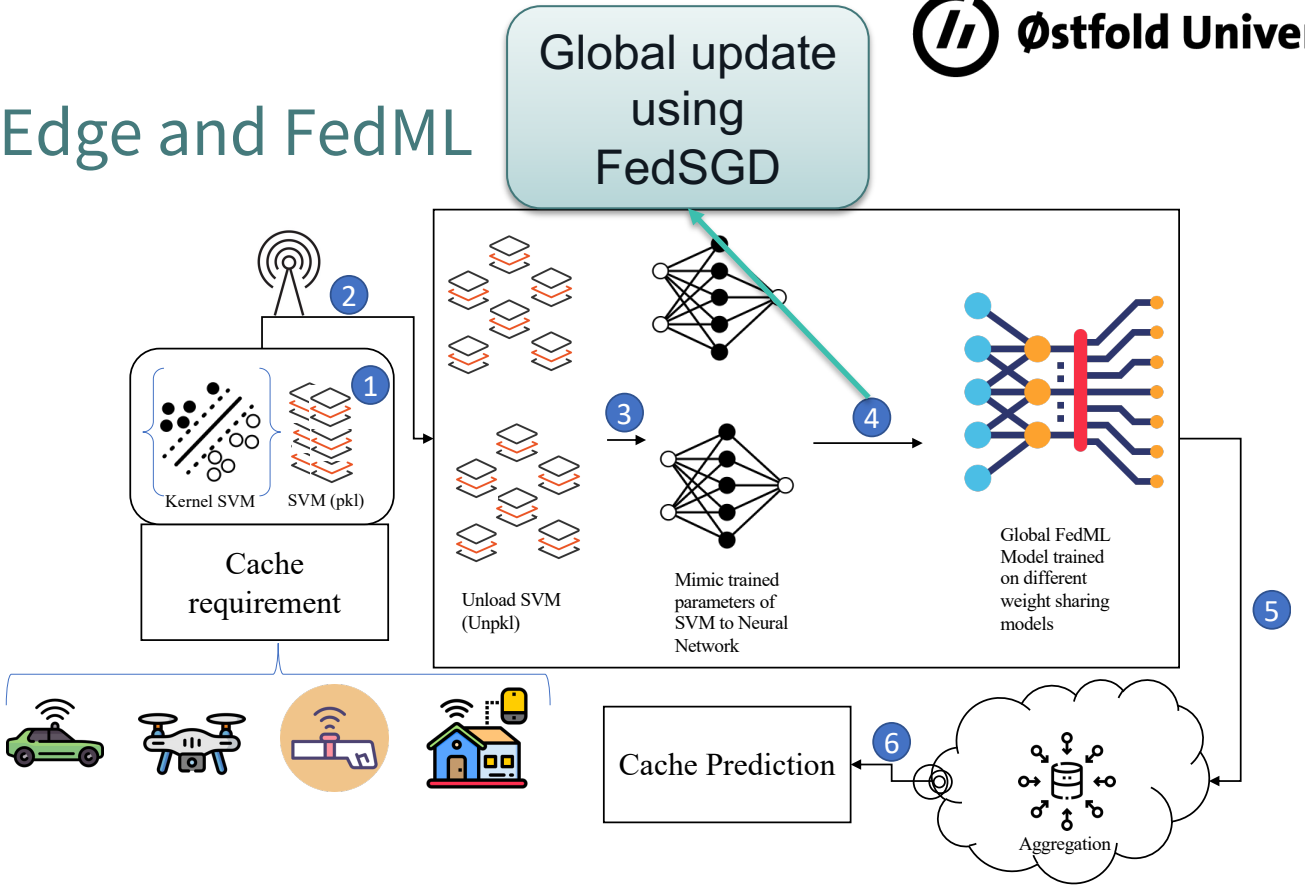
SVM at Edge and FedML



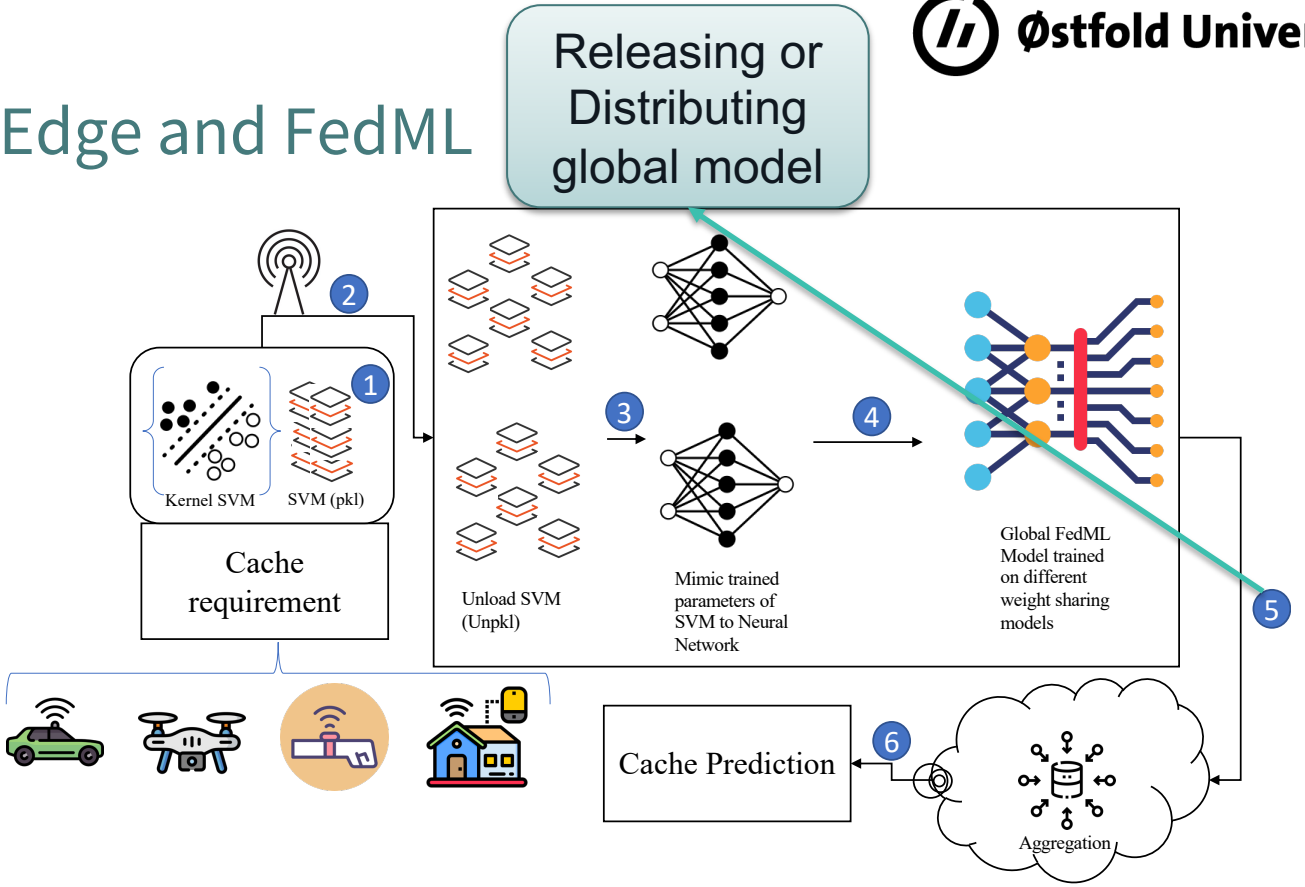
SVM at Edge and FedML



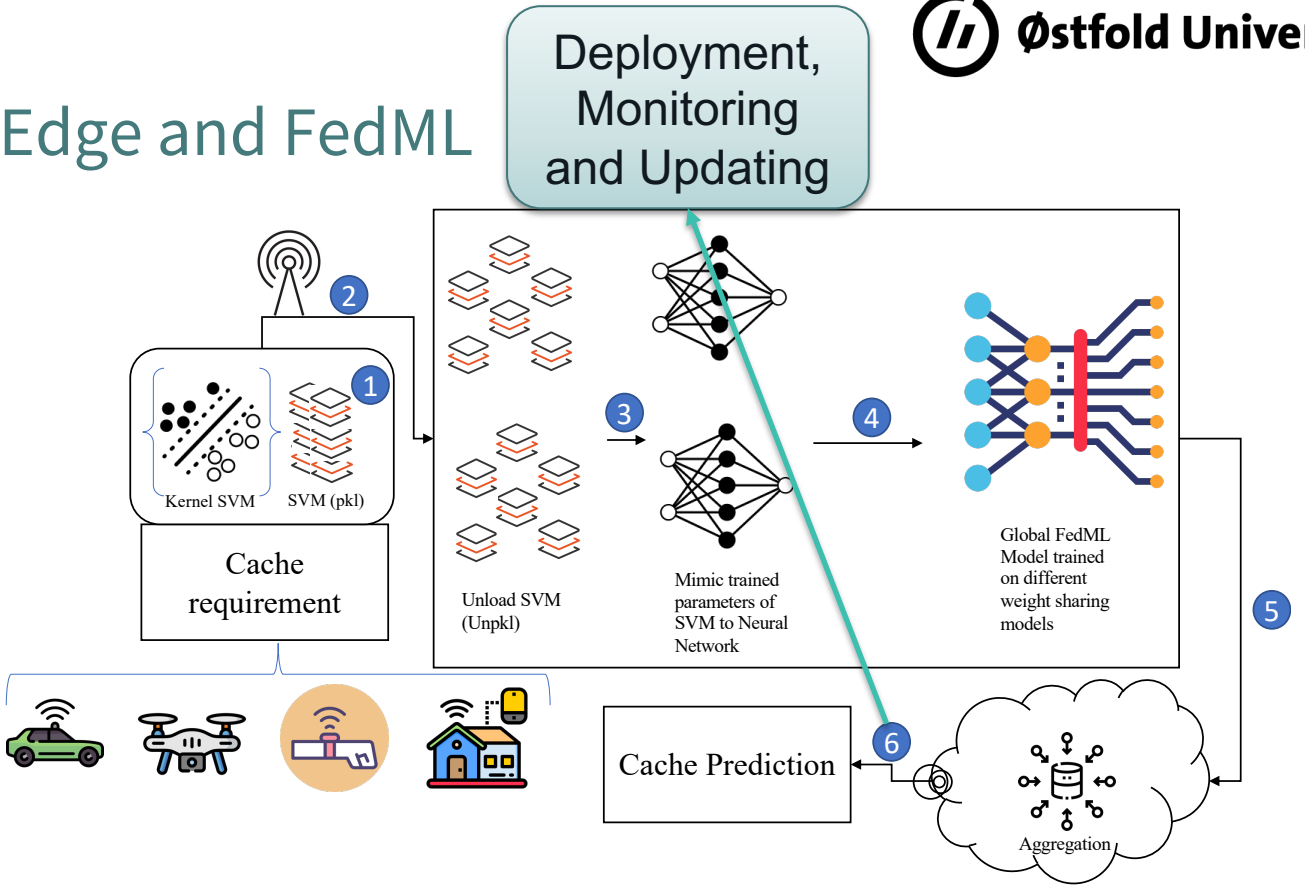
SVM at Edge and FedML



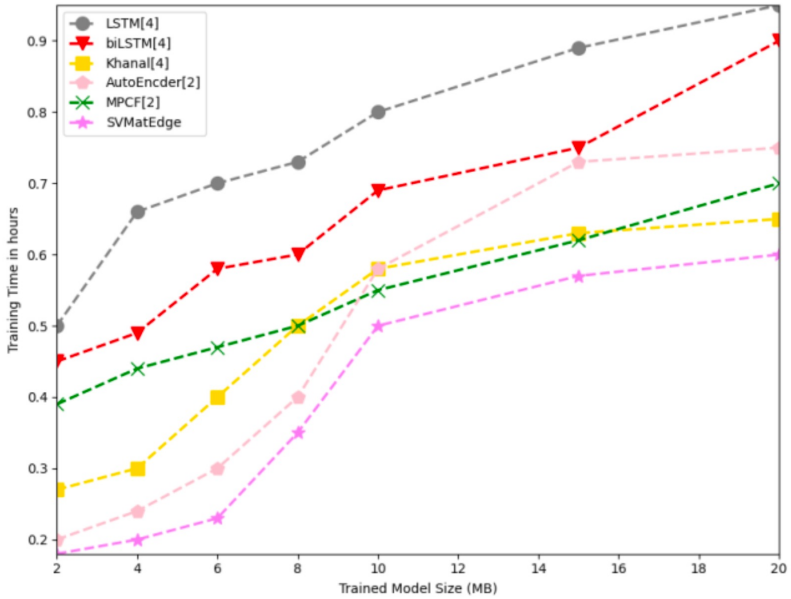
SVM at Edge and FedML

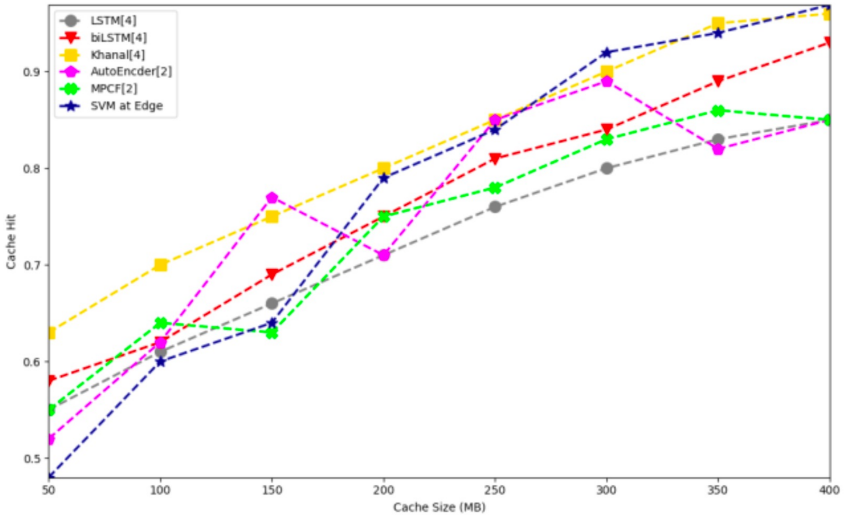


SVM at Edge and FedML



Results


LSTM[4]
biLSTM[4]
Khanal[4]
AutoEncder[2]
MPCF[2]
SVMatEdge
Training Time in hours
Trained Model Size (MB)


LSTM[4]
biLSTM[4]
Khanal[4]
AutoEncder[2]
MPCF[2]
SVM at Edge
Cache Hit
Cache Size (MB)

Implementation

- https://github.com/s4nyam/EdgeSVM

Østfold University College

Thank You