Adaptive FEC for Cloud Gaming

Abdulmueez Eniola, Ozioma Okonicha, Pavel Tishkin

Introduction

Cloud gaming is an emerging technology that allows users to play high-quality video games without the need for powerful gaming hardware.

Introduction

Cloud gaming is a rapidly growing industry, with more and more people turning to streaming services to play games.

Problem Statement

However, one of the biggest challenges is ensuring a smooth and seamless experience for gamers, regardless of their internet connection quality.

Problem Statement

More specifically, one of which is the issue of network latency and packet loss. These issues can lead to a poor gaming experience, including visual artifacts, input lag, and dropped frames.

Problem Statement

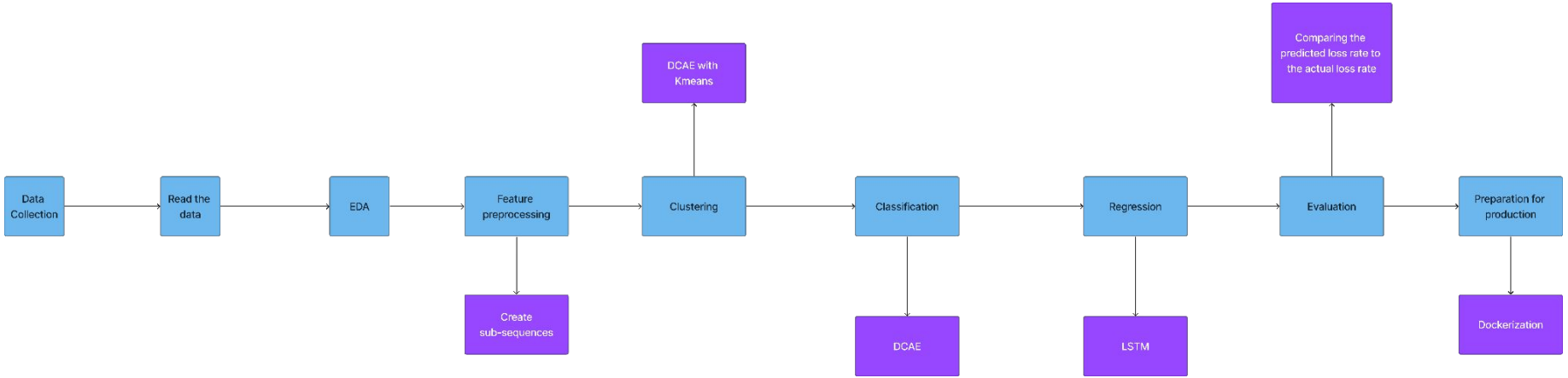
One solution to these problems is the use of Forward Error Correction (FEC), which adds redundant data to the packets to allow for error correction.

Problem Statement

Forward Error Correction (FEC) is a technique commonly used in data transmission to ensure data integrity by adding redundant information to the data packets. This technique can also be applied to cloud gaming to reduce the negative impact of packet loss on the player's experience.

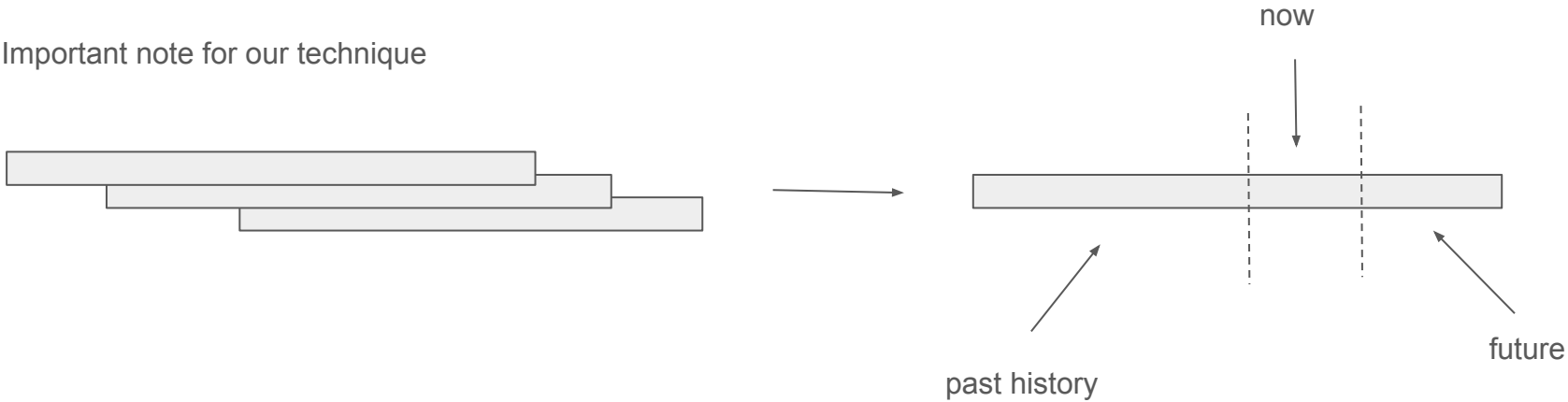
Proposed Solution

For our task, we will leverage neural networks to predict the loss rate in some steps in the future, hence we can send additional data corresponding to the amount of lost packets.



Proposed Solution

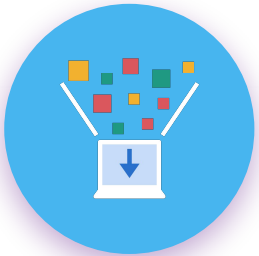
Important note for our technique



Proposed Solution

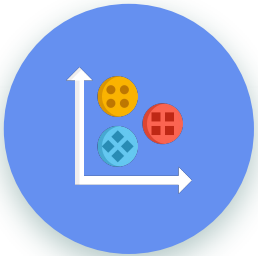
Below is the core ML part of our initial solution

Data Collection



Collecting raw data, dropping outliers, duplicated data, and splitting into subsequences

Clustering



Using kmeans to group the subsequences into clusters

Classification

Categorizing the subsequences into different classes.

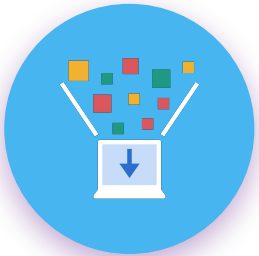
Regression

Predicting the loss rate based on the relationship between the other features in the data

Proposed Solution

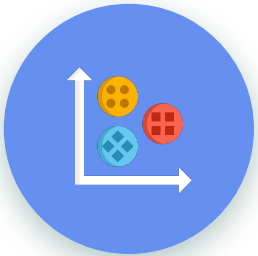
Below is the core ML part of our initial solution

Data Collection



Collecting raw data, dropping outliers, duplicated data, and splitting into **subsequences**

Clustering



Using a **DCAE model with kmeans** to group the subsequences into clusters and the classification is the cluster label

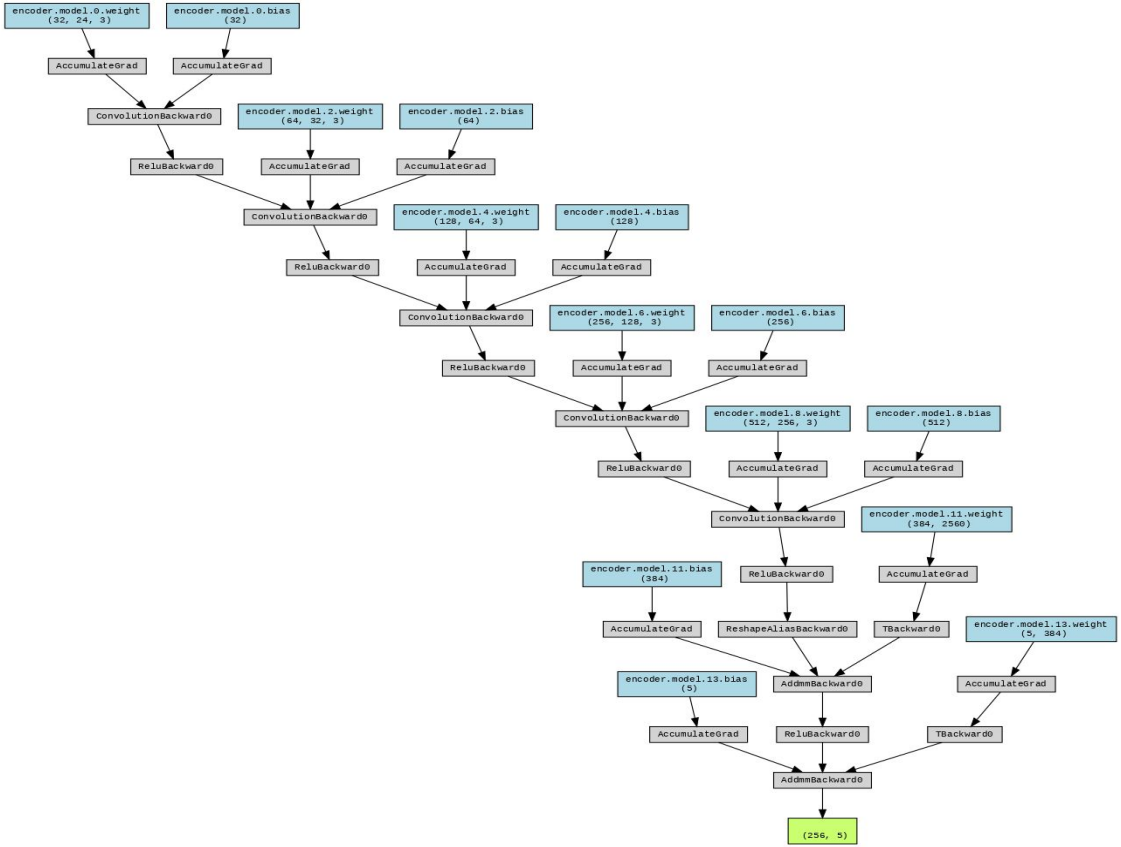
Classification

Categorizing the subsequences into different classes.

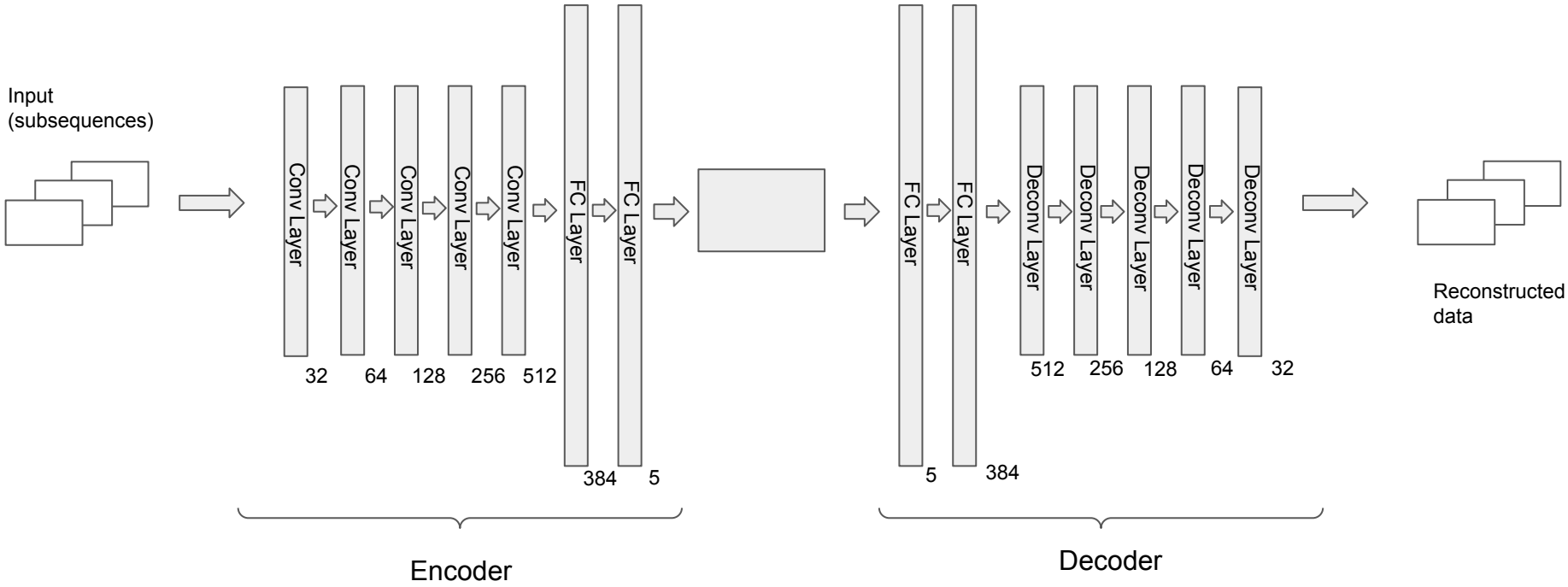
Regression

Using **lstm** to predict the loss rate based on the relationship between the other features in the data

Architecture- Autoencoder

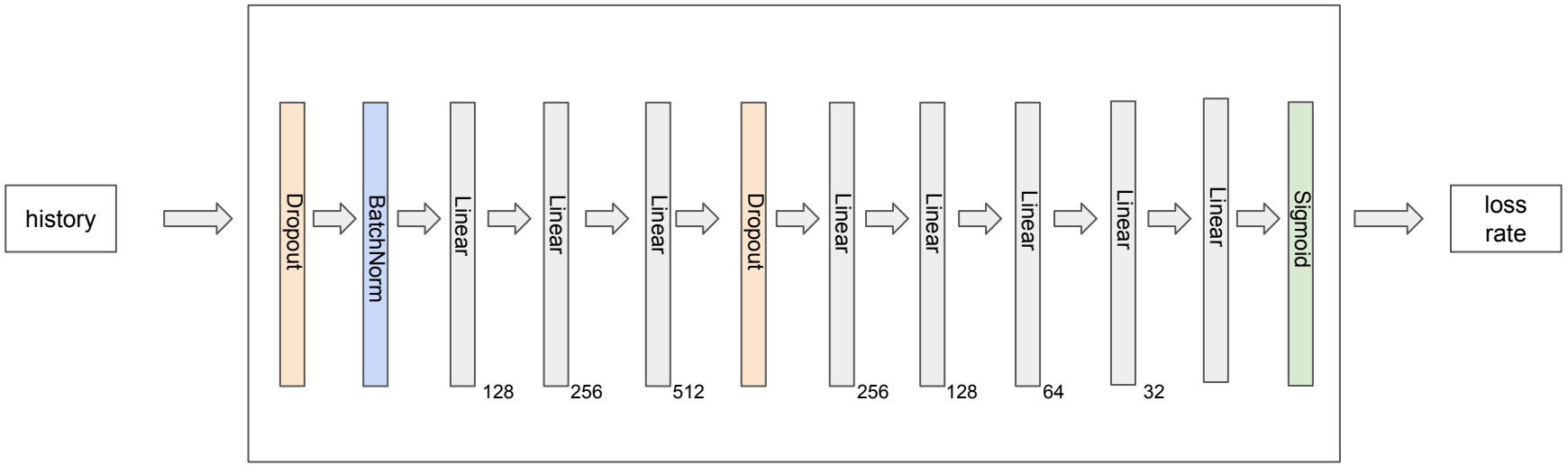


Architecture- DCAE

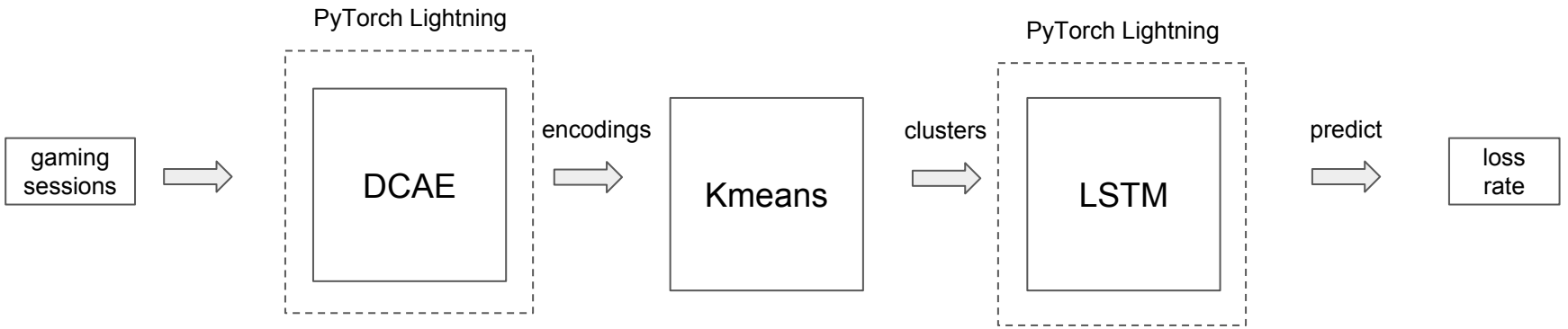


Architecture- LossRatePredictor

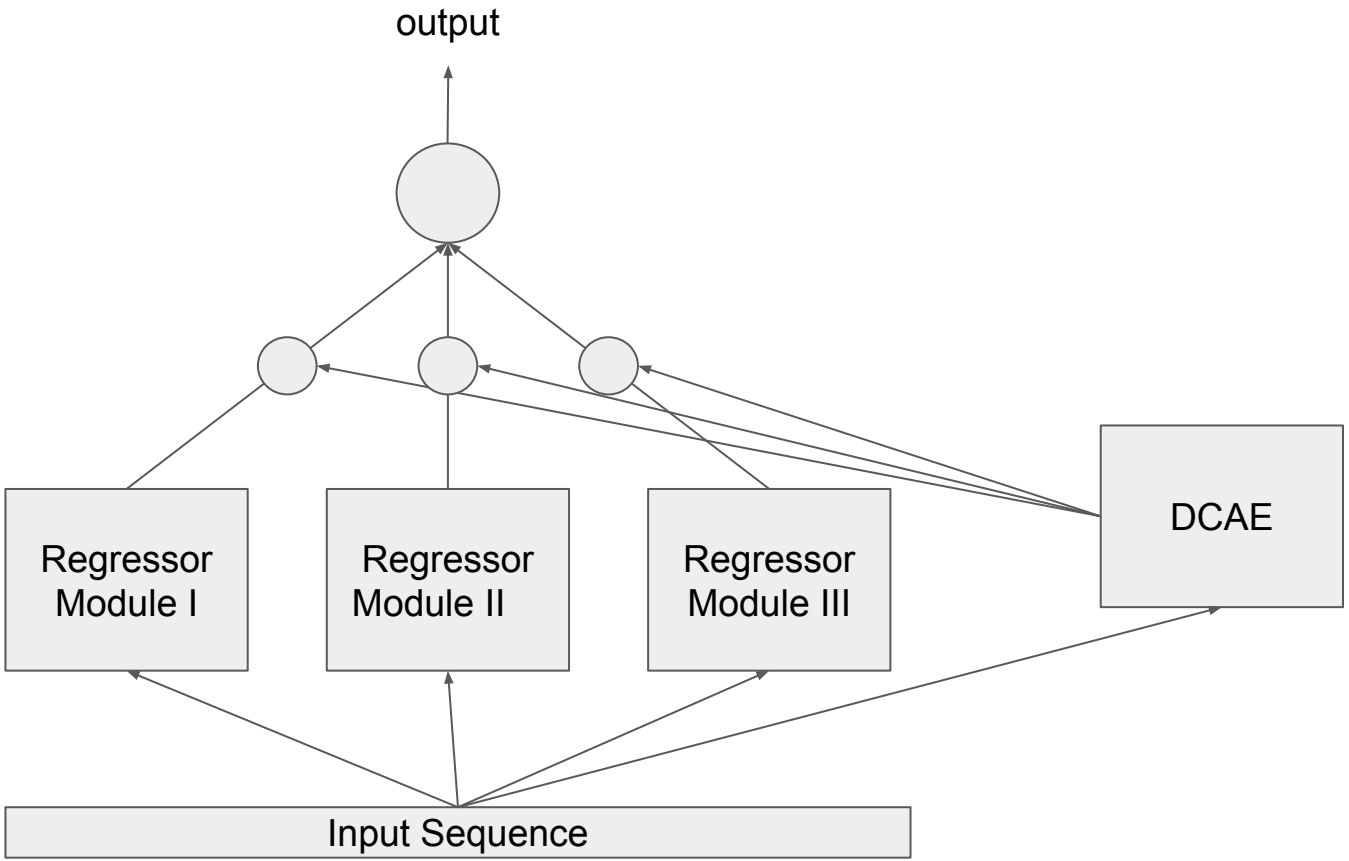
Architecture- LSTM



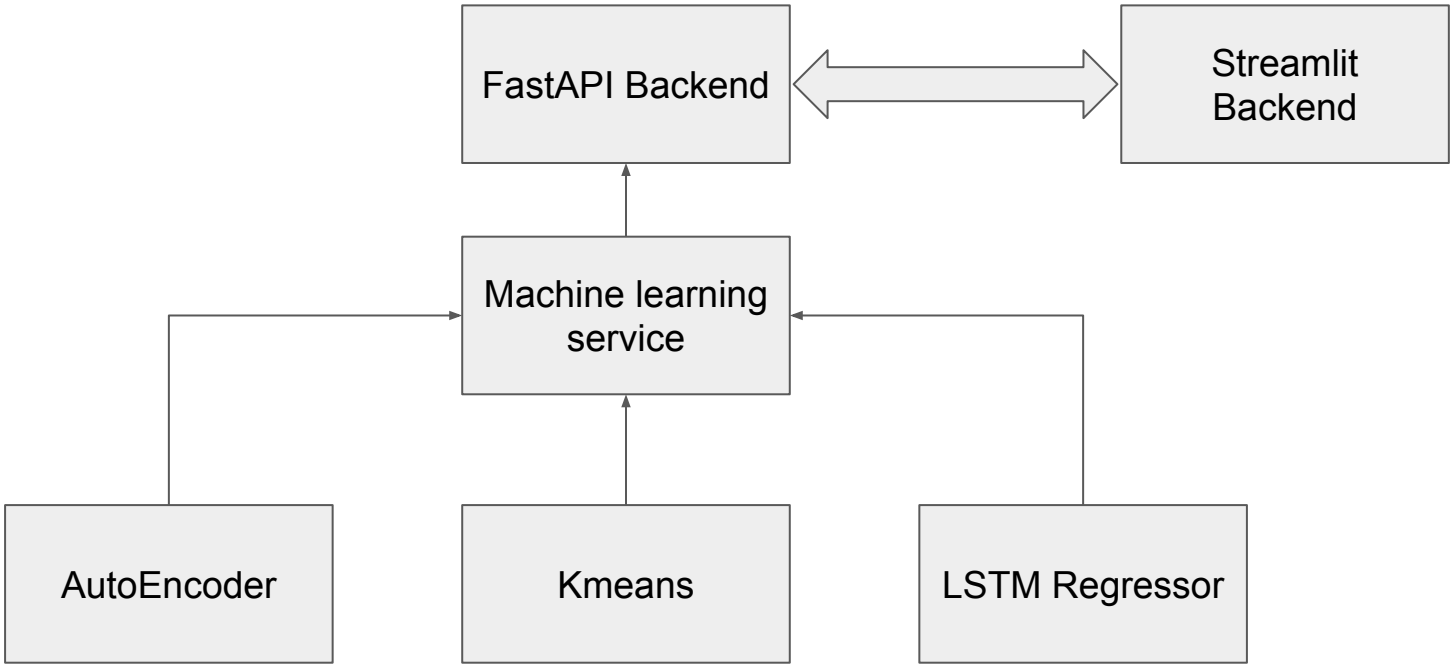
Architecture- Full view



Overview


output
Regressor Module I
Regressor Module II
Regressor Module III
DCAE
Input Sequence

Backend Architecture



Challenges and Changes

- So, we ended up **removing the classification section** as our clustering model automatically does that
- Also, our dataset was quite erroneous so we will need to **collect more adequate data**
- While we aimed to achieve no loss but we couldn't, this can be solved by **increasing model size** and getting more accurate data
- We eventually made use of **PyTorch Lightning** in both clustering and regression parts due to resources limitation

Challenges and Changes

- Since our technique involved **past, present and future**, we tried various combinations of subsequences and gaps to find the optimal
- Our **loss function was modified** in order to ensure that the predicted loss rate is never lesser than the actual loss rate.
- We needed to fine tune several deep learning models and then wire it all together
- In general, there is **not much research done in the field** compared to other ml fields like llm

Metrics & Duration

Below is a table to show our scores and time taken

	Clustering	Regressor 1	Regressor 2	Regressor 3
Training loss	0.797			
Training inertia	3.3e-07			
Training mse loss		8.37	0.14	29.24
Custom training loss		0.067	0.14	0.093
Duration		0.25 (without gpu)		

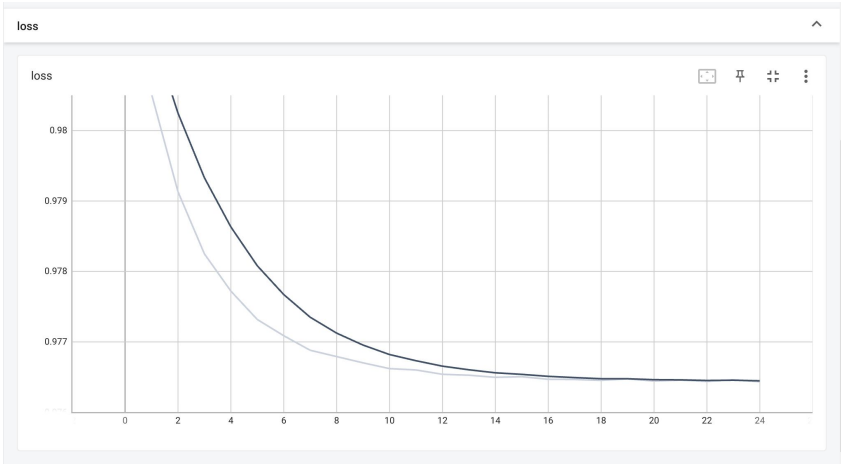
Metrics & Duration

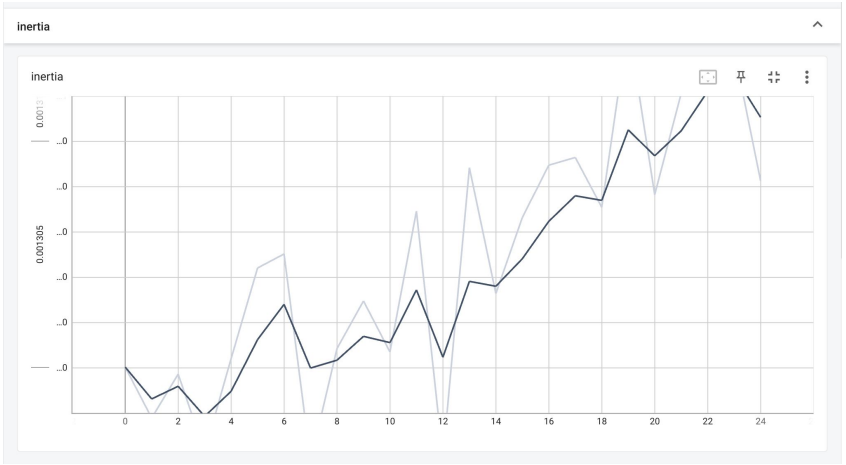
Below are the specifics for autoencoder training

	Validation MSE loss	Validation kmeans loss	epoch	step	Training MSE loss	Training kmeans loss
2	0.8006845712661743	5.639385562972166e-06	0	349		
3			0	349	0.7968215346336365	0.0002783223753795028
4	0.8006669282913208	1.272338408853102e-06	1	699		
5			1	699	0.7966016530990601	3.2577727324678563e-06
6	0.800666093826294	1.4935386616343749e-06	2	1049		
7			2	1049	0.7965887784957886	2.002028168135439e-06
8	0.8006616234779358	3.4284275329810043e-07	3	1399		
9			3	1399	0.7965836524963379	9.142794397121179e-07
10	0.8006616830825806	2.1211153011790884e-07	4	1749		
11			4	1749	0.7965797185897827	3.3043750136130257e-07

Metrics & Duration

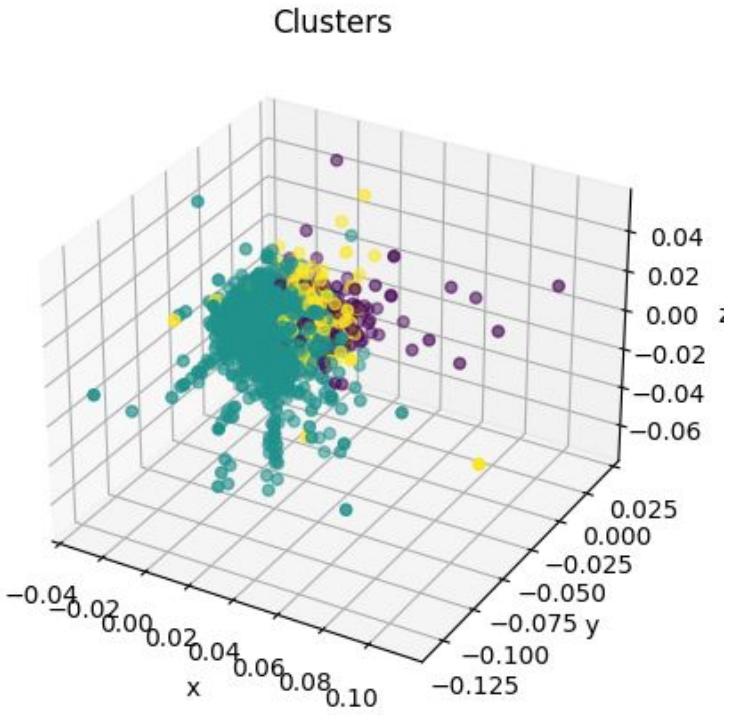
Below are the specifics for clustering training





Metrics & Duration

Below are the specifics for clustering training



Metrics & Duration

Below are the specifics for regression training

Regressor 1

Validation MSE loss	Validation Penalty loss	epoch	step	Training MSE loss	Training Penalty loss
8.826621055603027	0.06687212735414505	0	301		
		0	301	23.009737014770508	0.06667641550302505
8.826621055603027	0.06687212735414505	1	603		
		1	603	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	2	905		
		2	905	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	3	1207		
		3	1207	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	4	1509		
		4	1509	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	5	1811		
		5	1811	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	6	2113		
		6	2113	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	7	2415		
		7	2415	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	8	2717		
		8	2717	8.368857383728027	0.06716329604387283
8.826621055603027	0.06687212735414505	9	3019		
		9	3019	8.368857383728027	0.06716329604387283

Regressor 2

Validation MSE loss	Validation Penalty loss	epoch	step	Training MSE loss	Training Penalty loss
2191.06884765625	0.0	0	0		
		0	0	2426.64892578125	0.0
240.92459106445312	0.0	1	1		
		1	1	2102.323486328125	0.0
2.115949415681939e-09	0.0	2	2		
		2	2	153.32388305664062	0.0
0.0	0.0	3	3		
		3	3	0.1428571492433548	0.1428571492433548
0.0	0.0	4	4		
		4	4	0.1428571492433548	0.1428571492433548
0.0	0.0	5	5		
		5	5	0.1428571492433548	0.1428571492433548
0.0	0.0	6	6		
		6	6	0.1428571492433548	0.1428571492433548
0.0	0.0	7	7		
		7	7	0.1428571492433548	0.1428571492433548
0.0	0.0	8	8		
		8	8	0.1428571492433548	0.1428571492433548
0.0	0.0	9	9		
		9	9	0.1428571492433548	0.1428571492433548

Regressor 3

Validation MSE loss	Validation Penalty loss	epoch	step	Training MSE loss	Training Penalty loss
24.107194900512695	0.10066349059343338	0	48		
		0	48	99.53902435302734	0.09032749384641647
24.107194900512695	0.10066349059343338	1	97		
		1	97	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	2	146		
		2	146	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	3	195		
		3	195	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	4	244		
		4	244	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	5	293		
		5	293	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	6	342		
		6	342	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	7	391		
		7	391	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	8	440		
		8	440	29.237407684326172	0.09397009760141373
24.107194900512695	0.10066349059343338	9	489		
		9	489	29.237407684326172	0.09397009760141373

Metrics & Duration

Below are the specifics for regression training (CONTD.)

Regressor 1

22	8.826621055603027	0.06687212735414505	10	3321			
23			10	3321	8.368857383728027	0.06716329604387283	
24	8.826621055603027	0.06687212735414505	11	3623			
25			11	3623	8.368857383728027	0.06716329604387283	
26	8.826621055603027	0.06687212735414505	12	3925			
27			12	3925	8.368857383728027	0.06716329604387283	
28	8.826621055603027	0.06687212735414505	13	4227			
29			13	4227	8.368857383728027	0.06716329604387283	
30	8.826621055603027	0.06687212735414505	14	4529			
31			14	4529	8.368857383728027	0.06716329604387283	
32	8.826621055603027	0.06687212735414505	15	4831			
33			15	4831	8.368857383728027	0.06716329604387283	
34	8.826621055603027	0.06687212735414505	16	5133			
35			16	5133	8.368857383728027	0.06716329604387283	
36	8.826621055603027	0.06687212735414505	17	5435			
37			17	5435	8.368857383728027	0.06716329604387283	
38	8.826621055603027	0.06687212735414505	18	5737			
39			18	5737	8.368857383728027	0.06716329604387283	
40	8.826621055603027	0.06687212735414505	19	6039			
41			19	6039	8.368857383728027	0.06716329604387283	

Regressor 2

	0.0	0.0	10	10			
			10	10	0.1428571492433548	0.1428571492433548	
	0.0	0.0	11	11			
			11	11	0.1428571492433548	0.1428571492433548	
	0.0	0.0	12	12			
			12	12	0.1428571492433548	0.1428571492433548	
	0.0	0.0	13	13			
			13	13	0.1428571492433548	0.1428571492433548	
	0.0	0.0	14	14			
			14	14	0.1428571492433548	0.1428571492433548	
	0.0	0.0	15	15			
			15	15	0.1428571492433548	0.1428571492433548	
	0.0	0.0	16	16			
			16	16	0.1428571492433548	0.1428571492433548	
	0.0	0.0	17	17			
			17	17	0.1428571492433548	0.1428571492433548	
	0.0	0.0	18	18			
			18	18	0.1428571492433548	0.1428571492433548	
	0.0	0.0	19	19			
			19	19	0.1428571492433548	0.1428571492433548	

Regressor 3

	24.107194900512695	0.10066349059343338	10	538			
			10	538	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	11	587			
			11	587	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	12	636			
			12	636	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	13	685			
			13	685	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	14	734			
			14	734	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	15	783			
			15	783	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	16	832			
			16	832	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	17	881			
			17	881	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	18	930			
			18	930	29.237407684326172	0.09397009760141373	
	24.107194900512695	0.10066349059343338	19	979			
			19	979	29.237407684326172	0.09397009760141373	

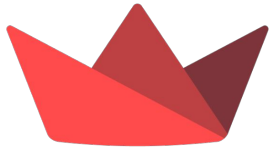
Future Improvements

- We need to improve on the clustering algorithm as ideal clusters for our task couldn’t be found
- Fine tune the models better, and try out other architectures to see how the performance differs
- Oversee the data collection process in order to get more accurate clusters. Data collection should focus on getting bad network.

Deploying

We made use of streamlit and FastAPI to prepare an interface that interacts with our models and is able to train and predict.

- Streamlit allows the end user to easily upload new datasets to finetune our preliminary model.
- While the fast api allows the end user to easily make predictions for new subsequences.



Streamlit

FastAPI

Demo