Advanced Machine Learning Project Report

Abdulmueez Emiola, Ozioma Okonicha, Pavel Tishkin
a.emiola@innopolis.university,o.okonicha@innopolis.university,p.tishkin@innopolis.university

1 Motivation

Cloud gaming is now a fairly popular form of online gaming. The way it works is that it renders the data of the game using a high-speed network on cloud-based and high-speed servers instead of on the player's system. Realistically, it's not all the time that the user have stable network connection and this is a requirement for a good streaming quality. As a result of this, issues with delay, distortion and data packets loss arise. To mitigate this and improve the quality of the experience of the players, adaptive algorithms have been created in order to enhance the streaming quality on this cloud based gaming and minimize the loss of data packets. By utilizing Machine Learning algorithms, 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. The rest of the report is structured as: Data where we specify the content of our dataset. Exploratory Data Analysis where we explain the structure and trends in the dataset. Task where we go into the details, architecture and training process. Results where we describe the outcome of our steps and last Conclusion where we summarize our project with the main takeaways.

2 Data

The original dataset consists of 8 columns: client_user_id, session_id, dropped_frames, FPS, bitrate, RTT, timestamp, loss_rate. It has 3351430 rows. Datasets for training and clustering are its derivatives. We group the data by tuple (client_user_id, session_id), and index the resulting dataframes by timestamp. From it, we derive subsequences of length 24. These subsequences serve as a modelling of the real world process of video streaming. The first 15 observations are the past data. They are used in clustering to advocate for varying network topologies. The next 2 are the gap that represents the process of transmitting the data. And the last 7 values represent the future we are trying to predict. Past data (all 5 columns in each timestamp) combined with only the loss_rate of the future signal (The target variable) comprise the training data for regression.

3 Exploratory Data Analysis

The original dataset is comprised of points and there are no NaNs amongst them. Unfortunately this is the only virtue we have been provided with regarding the data quality. Analyzing the behaviour, we found out several instances of samples that could not possibly happen in real life, or are questionable. We had to analyze three combinations of the parameters RTT, dropped_frames and bitrate:

- 0 RTT, 0 dropped frames, non-zero bitrate (0.16% of data) - this is an impossible situation, as 0 RTT means that packets are not coming back, therefore, dropped frames should be non-zero
- non-zero RTT, 0 dropped frames, zero bitrate (0.0014 % of data) - This is impossible due to the fact that no packets are sent
- 0 RTT, non-zero dropped frames, zero bitrate (negligibly small amount) - clear mistakes, as there is nothing to drop
- 0 RTT, non-zero dropped frames, non-zero bitrate (0.003% of data) - RTT information simply does not come due to network error. Not a clear mistake, nonetheless an unintended behaviour
- non-zero RTT, zero dropped frames, non-zero bitrate (95.74 % of data) - Actually, those are healthy examples we can use in our ML pipeline
- non-zero RTT, non-zero dropped frames, zero bitrate (none of them) - There can be no RTT or no frames lost if we don't send frames

For the feature selection, we decided to have all of them in the training, because we do not have lot.

4 Task

We seek to be a predict the loss rate of a particular session for a length of time in the future. To this end, we built a mixture of experts using neural networks. The experts are gated using a trained clustering neural network which assigns each input to different experts. The experts are regressor models trained to predict the future loss rate. This approach is motivated by the work done in this paper on **Deep-Learning Based Network-Adaptive FEC for Real-Time Video Communications**[2].

Before training, our input sequence needed to be preprocessed to remove anomalies. The sequence is also scaled down to obtain better predictions.

Finally, we had to preprocess the input and obtain a lower dimensional representation using an deep convolutional auto encoder.

4.1 Preprocessing

Preprocessing consisted of two separate stages. In the first one, we removed the data with the aforementioned problems (discussed in EDA). Second stage consisted of removing outliers. The outliers mostly consisted of unrealistic measurements (1835009 dropped frames, for instance), that likely indicate buffer overflows, or default values on the data collection side.

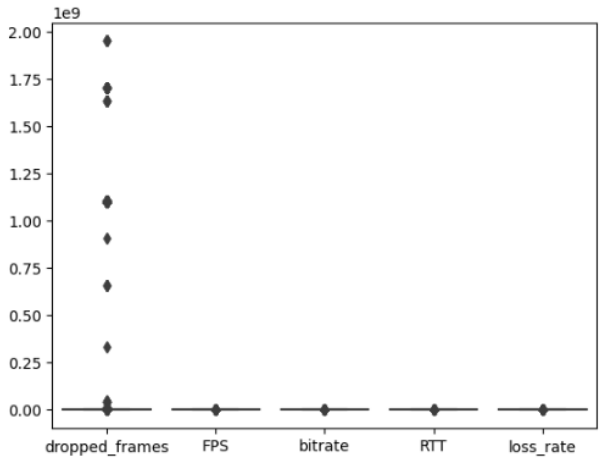

Figure 1. Outliers in the original data

Finally, we divide the data to the slides of length 24, and take appropriate parts of it for training and classification.

4.2 Clustering

To cluster our subsequences, we first transform them into a lower dimensional manifold. This is due to the fact that traditional clustering algorithms suffer from the curse of dimensionality. Also, traditional clustering algorithms are not able to capture the intricacies of time series data like feature representation at different time scales, distortion by high frequency pertubations etc.

To transform the subsequences into a lower dimensional view, we employ a deep convolutional auto encoder architecture [1]. We then find clusters from the learned latent features.

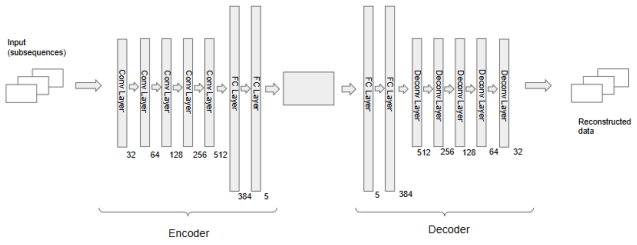

Figure 2. Architecture of our DCAE model

We train the **DCAE** model on the past history segment of each subsequence, as during the inferencing stage, we will have access to only that part of the subsequence. We are able to transform a single segment from a two dimensional space (24*5) to a one dimensional space (1*5).

Also, in order to improve the usability of our encodings, we introduced a new objective to our encodings. We tried to generate encodings that resulted in the lowest clustering inertia.

Consequently, we are able to remove the problems caused by the curse of dimensionality and apply a clustering technique on the generated embeddings. We use the kmeans algorithm for clustering because they are fast, scalable and efficient. We find three different clusters in our data and use them in the remaining procedure.

4.3 Regression

In this part of the work, our aim is to accurately predict the loss rate of the future segment of a subsequence. To do this, we train three different regression model corresponding to the three different clusters we got from our clustering module. Each of our regression model have the same structure.

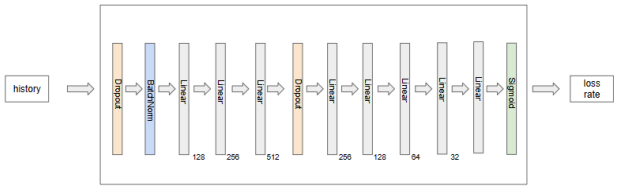

Figure 3. Architecture of our Regressor model

A regressor model contains a LSTM model that helps us to capture the intricacies of our dataset. The output of the LSTM is then passed to a feed-forward network which produces a sequence of values which are future predictions across a length of time. In our work, we predict 7 steps into the future corresponding to 35 seconds.

In other to further improve the performance of our model, we introduce 2 new restrictions to our task.

- The predicted loss rate value should never be lesser than the actual loss rate. To do this, we introduce a new loss function which penalizes the model when it predicts a value lesser than the actual loss rate.
- The predicted loss rate should always be between 0 and 100. To do this, we use a sigmoid layer in the final layer.

4.4 Inference

As stated earlier, our aim is to build a mixture of experts capable of making predictions. In inference mode, when a new past data subsequence of length 24 * 5 corresponding to 2 minutes of network data is passed into the whole system. Our trained clustering module serves as the gates of our architecture. It helps to select the right regressor model that should be used in predicting the loss rate.

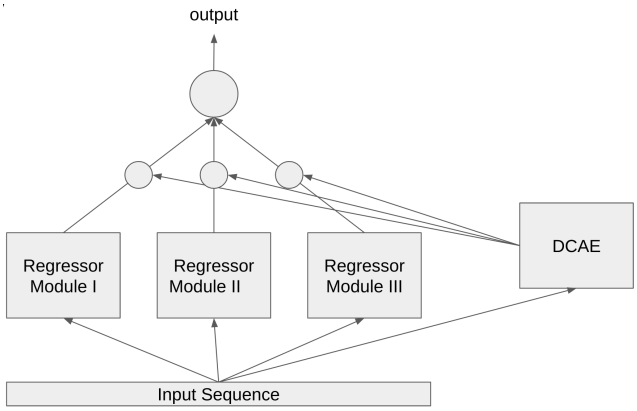

Figure 4. Overview of our mixture of experts architecture

5 Results

In this section, we evaluate the efficiency of each of our modules.

5.1 Clustering

To evaluate the clustering module, we need to evaluate the accuracy of our encodings and also measure the goodness of the generated clusters. We measure the accuracy of our encodings by finding the mean squared error between the original subsequencies.

Figure 5. Training and validation loss of the DCAE

Next, we measure the goodness of the clusters by computing the within cluster sum of squares across different epochs.

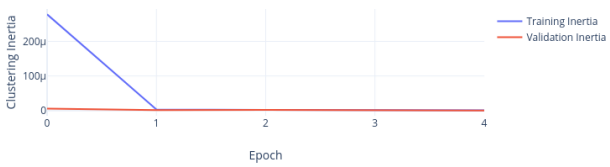

Figure 6. Inertia of the clusters

Table 1. Clustering metrics

Model	Training Inertia	Validation Inertia
Kmeans	3.3043750136130257e-07	2.1211153011790884e-07

We can see from the above that our model doesn't overfit and tends to produce clusters of very low inertia which makes them suitable for our task.

5.2 Regression

To evaluate the regrssion module, firstly, we need to compute the mean squared error between the predictions and the actual values for each of the regressor models. We also need to compute the ability of each of the module to make predictions not lesser than the actual loss rate.

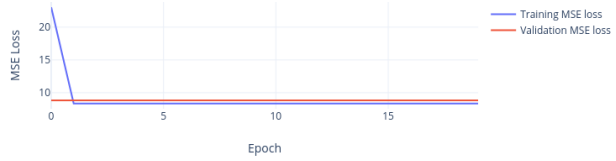

Figure 7. MSE loss of the regression model for the first cluster

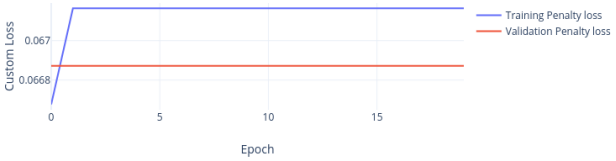

Figure 8. Custom loss of the regression model for the first cluster

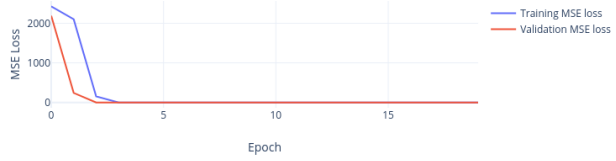

Figure 9. MSE loss of the regression model for the second cluster

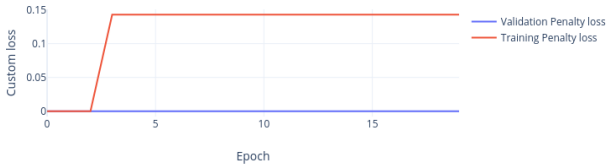

Figure 10. Custom loss of the regression model for the second cluster

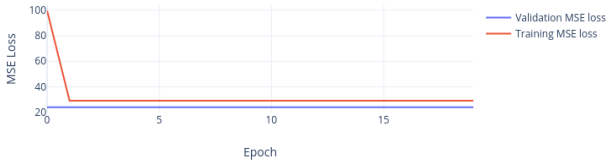

Figure 11. MSE loss of the regression model for the third cluster

Figure 12. Custom loss of the regression model for the third cluster

Table 2. Regression metrics

Model	MSE	Custom Loss
Regressor 1	8.368857383728027	0.06716329604387283
Regressor 2	0.1428571492433548	0.1428571492433548
Regressor 3	29.237407684326172	0.09397009760141373

We can observe two things from the values, there's an indirect relationship between the MSE loss and the custom loss. Hence, improving the MSE loss leads to a reduction in the custom loss. Also, we can see that although we do not achieve a very low MSE loss in some of our regressor loss. The models still fulfills our objective by providing a very low percentage of it's predictions below the actual value. This makes it's predictions usable to predict the loss rate.

6 Challenges and Future Improvements

While implementing the project, we faced some challenges and noticed areas for future improvements. This section highlights these challenges and mentions things we can improve upon.

6.1 Challenges

To begin with, our dataset was found to be erroneous so we will need to collect more adequate data. Next, regardless of our efforts, we were unable to achieve zero loss. This could be because of the model size, limitations in the available resources, or the quality of our data.

Additionally, the clustering algorithm employed in our project struggled to identify ideal clusters for our task. Lastly, compared to other machine learning fields, there is relatively limited research conducted in this field.

6.2 Future Improvements

In the future we could aim to refine and fine-tune the models in our project as it could result in improved performance. We can do that by trying out different architectures, hyperparameters, and optimization techniques. Other than fine-tuning the models, exploring alternative architectures and approaches could provide better insights. Finally, in the data collection process we could focus on obtaining more accurate and representative samples of network conditions, especially in scenarios with poor network performance, would be beneficial.

In conclusion, our project faced challenges related to data quality, achieving zero loss, clustering algorithm performance, and limited research in the field. Overcoming these challenges and going for future improvements, such as fine-tuning models, exploring alternative architectures and better data collection could contribute to more accurate and reliable results and in turn cloud gaming experiences.

7 Conclusion

To conclude, we started out by exploring the given dataset to understand the underlying structure and distributions. Then we performed preprocessing and generated subsequences. After that we used a DCAE model, wrapped in PyTorch lightning to get encodings from the subsequences. Using the encodings, we made use of kmeans to extract three clusters. For each cluster, we then trained a Regressor using LSTM and wrapped with PyTorch Lightning to predict the loss rate. We logged the training and validation metrics for each model to evaluate their performance. Finally, the results can be tested out using our Streamlit application as can be seen along with every other specific in our Github repository.

References

[1] Ali Alqahtani et al. “Deep Time-Series Clustering: A Review”. In: *Electronics* 10.23 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10233001. URL: https://www.mdpi.com/2079-9292/10/23/3001.

[2] Sheng Cheng et al. “DeepRS: Deep-Learning Based Network-Adaptive FEC for Real-Time Video Communications”. In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180974.