Multimodal Deep Regression on TikTok Content Success

Louis Wong
College of ComputingGeorgia
Institute of Technology
lwong64@gatech.edu

Ahmed Salih
College of Computing Georgia
Institute of Technology
asalih6@gatech.edu

Mingyao Song
DeGroote School of Business
McMaster University
songm45@mcmaster.ca

Jason Xu
College of Computing Georgia
Institute of Technology
jxu623@gatech.edu

***Abstract*—Content creators grapple with the challenge of predicting if their investments will lead to increased viewership and audience growth on social media platforms. By employing advanced techniques in video encoding and natural language processing, we construct a powerful multimodal ensemble model for accurately predicting video success. Our preliminary results demonstrate the model's effectiveness in predicting video virality**

Keywords—virality, multimodal, predicting.

I. Introduction

The digital content creation landscape keeps evolving, challenging creators to predict video success and audience growth. Opaque algorithms and unpredictable interactions on platforms like TikTok complicate matters. Diverse approaches and models have been explored to predict video success and understand content virality.

To address YouTube viewership challenges, Liu et al. [15] proposed a Precise Wide-and-Deep Learning model, accurately predicting viewership while interpreting feature effects. In the context of TikTok, researchers [7] introduced a deep learning model to predict user participation in challenges by learning latent user and challenge representations. Salvador et al. [3] used attention mechanisms to improve video recognition models for human action recognition. A multi-modal fusion framework [5] integrated different modalities for effective short video understanding and recommendation. To describe videos, a deep neural network with multi-modal fusion [6] learned joint representations for video description tasks. Predicting Instagram popularity involved convolutional neural networks and long short-term memory networks [14], exploring spatial and temporal information. Another work used neural networks and regression analysis [9] to predict popularity by comparing aesthetics and social metadata. While these contributions are significant, the challenge of effectively integrating diverse modalities, like visual and audio data, remains for multi-modal fusion research. Striking the right balance and seamless integration between modalities are crucial for successful fusion techniques.

Two primary fusion approaches in multimodal learning are early fusion, which concatenates features at the input level, and late fusion, which aggregates predictions at the decision level [12]. Performance comparison between these approaches depends on several factors such as multimodal data characteristics, task complexity, interdependence between modalities, feature quality, network architecture, dataset size, and labeled data availability [12, 4, 2, 13]. As there is no universally superior fusion approach, each method can be more effective in specific scenarios. This study adopts the late fusion approach, considering unique project factors, as it effectively leverages modality strengths, capturing complementary information and improving performance.

Our study aims to predict video virality using a multimodal ensemble model. Beyond predicting video content creators' success, this project validates the use of multimodal approaches, surpassing traditional unimodal methods. Validating this approach emphasizes the potential for enhancing predictive ability by analyzing multiple streaming modalities. Our work showcases the promise of the multimodal era, demonstrating how leveraging diverse data modalities can lead to innovative solutions and deeper insights..

II. Approach

2.1 Data collection

The data collection process involved scraping video data from TikTok using a custom-built Selenium scraper running on a Chromium browser. The scraper was designed to randomly collect videos across various hashtag topics, ensuring a diverse representation of content. The selected hashtag topics included Sports, Dance, Entertainment, Comedy and Drama, Autos, Fashion, Lifestyle, Pets and Nature, Relationships, Society, Informative, and Music. This approach aimed to capture a wide range of video content to ensure the model's generalizability. In total, the data collection effort yielded approximately 1,100 videos, each in .mp4 format, resulting in a substantial dataset with a total size of 6.8 GB. To facilitate further analysis and model training, each video was assigned a unique video ID tag for reference and data mapping. Alongside the video files, the dataset also includes JSON data containing relevant metadata for each video, such as the video ID tag and its corresponding TikTok URL. Additionally, the video view count on TikTok was recorded as an essential metric for evaluating video creator success

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the A4 paper size. If you are using US letter-sized paper, please close this file and download the Microsoft Word, Letter file.

2.2 Data Preprocessing

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

2.2.1 Visual Preprocessing

The visuals for the video are extracted using TorchVision, converting the raw .mp4 format into tensors. The tensors are sized as (Channels, Frames, Height, Width). Here, 'Channels' represents RGB (3), 'Frames' varies greatly depending on the length of the video; some videos are as short as a few seconds, some are as long as 10 minutes. However, with a frame rate of 60fps, that can be

$$60\ FPS \cdot 60\ Seconds \cdot 10\ Minutes = 3{,}600 frames$$

For 'Height' and 'Width', most videos are 1024 by 576 pixels respectively, according to mobile application standards. Nevertheless, there are also various different heights and widths as well.

$$3(C) \cdot 3600(D) \cdot 1080(H) \cdot 576(W) = 6{,}718{,}464{,}000$$

The tensor size shown is a single input tensor. With the inclusion of batches leading to further increases in the tensor size. These tensors were too large to train on our accessible hardware. Therefore, we strategized to use multiple techniques to reduce the computational cost by decreasing the amount of information. We explored several downsampling techniques such as trimming the video length, averaging the tensors, skipping frames, and rescaling the video height and width. After several attempts, we chose to skip frames and rescale the pixels ad it made it feasible to run on our available hardware. For other size format videos, we adjusted the height and width to accommodate the mobile aspect ratio to a standard format of 1024 by 574, we first rescaled to this size. Then, filled the remaining values with 0. This approach created more uniformity for the model, as most models only accept input with a fixed window size.

2.2.2 Audio Preprocessing

We extracted audio from the video datasets using FFmpeg. The extracted audio is then converted to .wav format, which is solely for audio, and saved to an audio directory with an ID tag as a naming convention. This setup was designed for the audio embedding portion of this project. We leverage the Whisper framework to transcribe audio into sentences and obtain the corresponding word embeddings.

2.2.3 Target Values

The target labels are scraped metadata representing the counts of video views. However, these labels are rounded to '30.58M' or '75.8K', etc. They are not perfectly decimal metrics, but they should be sufficient for our needs. We clean up these metrics to a floating point format in thousands (K), for example '30.3', to unify the target format before we feed it into our model. The targets can range from 0 to 10,000s. However, in later experiments, we decided to standard-normalize these metrics, discussed in Section 3.2.2.

2.3 Multimodal Architecture

Our approach involves data preparation and multimodal deep regression modeling. The data preparation process includes scraping video data from TikTok, audio extraction, and meticulous organization of visual tensors for efficient integration into the model. For audio analysis, we leverage the open-source Whisper model to transcribe audio content and obtain audio embeddings that capture the semantic meaning of the audio. The heart of our model lies in the visual embeddings generated through an unsupervised pretraining process using a ConvLSTM Autoencoder. This process encodes the context of the video into compact and informative embeddings that retain essential spatial and temporal features. Subsequently, the visual and audio embeddings are concatenated and fed into a Transformer-based regression model for multimodal analysis. The late fusion technique combines the visual and audio data, enabling the model to learn the semantic and nonlinear relationships between the two modalities. These relationships are important for understanding what makes a video successful. The Transformer model, with its self-attention layers and feedforward neural networks, captures complex patterns and relationships within the data.

The video is first broken down into visual and audio tracks, with our primary focus on the video visual. The video visual will undergo an autoencoder process, utilizing a convolution-based network architecture, ConvLSTM Autoencoder, to unsupervised pre-training from scratch. This process encodes the context of the video into embedding vectors. Subsequently for the audio, we leverage a pretrained model, the open-source Whisper, to create a transcript for the audio, supplementing the project. The visual embeddings and audio embeddings are then extracted from the visual and audio branches, respectively. The embeddings are then concatenated and input into a Transformer-based regression model.

2.3.1 Visual Embedding

To generate compact and informative visual embeddings, we employ an unsupervised pretraining method using a ConvLSTM Autoencoder. The ConvLSTM Autoencoder is introduced by Shi et al.[11] and Nielsen [8]. This architecture comprises a series of ConvLSTM Cells, which are a combination of Convolutional and Long Short-Term Memory (LSTM) layers. The ConvLSTM Cells acts both as an encoder and a decoder, capturing the spatial features via the convolutional layers and temporal dependencies through the LSTM layers. Finally, the decoder passes through 3D convolutional layers to reconstruct the sequences of visuals (video tensors). This enables the Autoencoder to retain important contextual information from the video data, while also reducing dimensionality to create compact embeddings at the encoder output. During training, the Autoencoder takes video frames as input and tries to reconstruct them at the output. The difference between the original and reconstructed frames is quantified using the Mean Squared Error (MSE) loss function. Through backpropagation, the Autoencoder adjusts its weights and biases to minimize this reconstruction error, thus learning to capture meaningful patterns in the video data. The trained Autoencoder is evaluated thoroughly over multiple epochs to ensure that it learns robust and meaningful embeddings. The embeddings generated by the Autoencoder represent the visual context of the video. These embeddings condense the raw visual data into a more informative and compact representation, which is crucial for downstream modality fusion.

2.3.2 Audio Speech Embedding

Audio speech embeddings are obtained using the opensource Whisper model through unsupervised pretraining. Whisper is a powerful automatic speech recognition (ASR) system developed by OpenAI to convert spoken language into text. First, the audio is extracted from the video dataset, and

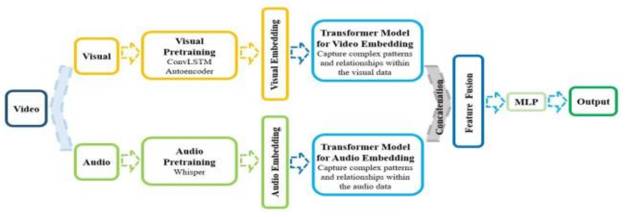
Video
Visual
Visual Pretraining
ConvLSTM Autoencoder
Visual Embedding
Transformer Model for Video Embedding
Capture complex patterns and relationships within the visual data
Audio
Audio Pretraining
Whisper
Audio Embedding
Transformer Model for Audio Embedding
Capture complex patterns and relationships within the audio data
Concatenation
Feature Fusion
MLP
Output

Fig.1. The multimodal model architecture

then Whisper transcribes the audio dialog, producing textual transcripts that capture essential information from the spoken content. These textual outputs serve as audio embeddings, effectively encapsulating the semantic meaning and characteristics of the audio. Utilizing the preexisting Whisper model for audio embedding generation offers several advantages. The Whisper model is already trained on extensive speech data, making it effective in understanding diverse spoken language patterns. This saves the effort and time required to train an ASR system from scratch, making the process more efficient [10]. Subsequently, the resulting audio embeddings play a pivotal role in audio speech analysis. The audio speech embeddings can enrich context comprehension in downstream modals.

2.3.3 Dual Transformer-based Regression

A Transformer-based ensemble regression is employed to perform multimodal deep regression by combining visual and audio data using a late fusion technique. The ensemble model consists of two primary components: a Transformer model for visual embedding and another Transformer model for audio embedding. Both models exploit the Transformer architecture, a series of Transformer encoder blocks, which heavily relies on the self-attention mechanism [1]. These models accept visual and audio embeddings as inputs respectively. They process these inputs using the Transformer to capture complex patterns and relationships within the data. Given the unique shapes and sizes of the visual and audio embeddings, it's challenging to consolidate the information into a single transformer network. Therefore, we introduce a dual transformer network. This configuration accommodates each modality, allowing the network to learn each output separately, and subsequently discover the relationship with the regression problem.

2.4 Loss Function

The regression models and auto-encoder are both trained using the MSE loss function, which serves as a measure of the discrepancy between the model's predicted values and the actual ground truth value. The primary objective is to minimize this while the auto-encoder will reduce the reconstructed pixel metric loss, the Tranformer-based regression will reduce loss from prediction to truth success metrics. The MSE loss function calculates the average squared difference between the predicted values and the actual values. By squaring the differences, it penalizes larger errors more severely, emphasizing the importance of accurate predictions across all data points. MSE loss functions are defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Beside MSE, we later also introduce Mean Absolute Error (MAE) to our model, since it does not heavily penalize large prediction errors, which is beneficial in cases where outliers in the data may cause excessive influence on the model training, we later discuss this further in Section 3.2.3 . MAE loss functions are defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|.$$

Where n is the total number of samples in the dataset. yi represents the actual value of the success metric for the ith batch of videos. yˆi represents the predicted value of the success metric or pixel values for the ith batch of videos.

2.5 Implementation and Hardware

The models primarily utilized the PyTorch library, with Torch-vision and Scikit-learn employed for preprocessing. We also forked the Whisper project for audio transcript processing and used a forked version of Swin Transformer 3D from haofanwang's repository. Training implementations were executed on local machines equipped with NVIDIA RTX 2070 Super/3080 GPUs, using CUDA. Initially, we faced some challenges due to the size of the tensors, as the GPUs had limited VRAM capacities. Consequently, some early training attempts failed due to exceeding VRAM memory. In the later phases, we primarily used CPUs in conjunction with physical RAM, and we also allocated additional virtual memory to accommodate the size of the visual tensors. The code for this project can be found in the Multimodal-Deep-Regression GitHub repository. The model and functional modules were developed using Python, and the various experiments were set up using Jupyter Lab/Notebook. Please refer to 'The_Multimodal_Final.ipynb' for the setup guide and detailed installation instructions to rerun the experiments. The required dependencies for this project can be found in the requirements.txt file of the GitHub repository.

III. TRAINING AND EXPERIMENTS

In the experiment, we first pretrained the Convolutional LSTM Autoencoder to an optimal hyperparameter setting and stored the training weight. Then we applied the Transformer ensemble model along with Whisper embedding input for regression prediction. As this is a regression prediction task, we evaluated the result using the common MSE metric. Since this is a new dataset, there isn't a preestablished performance benchmark to reference. Therefore, we decided to compare with two baseline models. The first is a vanilla 3D convolution model, and the second is a state-ofthe-art pretrained SWIN-Transformer 3D, as referenced in Swin Transformer: Hierarchical Vision Transformer using

Shifted Windows. The SWIN-Transformer is developed by Microsoft. The Swin Transformer utilizes shifted windows to adapt the transformer architecture for computer vision use cases [9]. The key design element for the Swin Transformer is the shifting windows between consecutive self-attention layers bridging between the layers which connects them [9]. Furthermore, we explored an alternative approach by narrowing down the regression task to classification using quantile ranges. The video data was split into a training set and a validation set, with 800 videos in the training set and 200 videos in the validation set. Lastly, we held out 100 videos in a separate set for final testing.

3.1 Convolutional LSTM Autoencoder

The initial phase of the experiments focuses on training the Convolutional LSTM Autoencoder, a crucial component responsible for learning a compact and meaningful representation of the input video data. The Autoencoder follows an unsupervised learning approach, with the primary objective of reconstructing the original input from the learned latent space representation. The following key parameters and settings are employed during this training phase: Before training, the video data undergoes preprocessing steps to prepare it for the model. The ’Frame Skip’ parameter is utilized to determine how many frames to skip during the preprocessing stage, effectively reducing the temporal depth of the video. Additionally, the ’Shrink’ parameter is applied to scale down the resolution of each frame, resulting in a reduced (Height x Width) dimension. Frame skipping and shrinking were two techniques we chose to reduce the computational and storage requirements while maintaining essential information. We were careful about choosing the degree of shrinkage and number of frames to skip given that too high of either might lead to a loss of important visual details and temporal information. We decided on shrinking by a factor of 8. Although this resulted in significantly reduced frame clarity, subjects within the frames remained identifiable. The number of frames to skip was set to 200, which was feasible to run experiments without each epoch taking a significant amount of time and computational memory resources.

3.1.1 Handling Sequential Visual Tensors

The input tensor size fed to the model after processing is (Batch, Channels, Frames, Height, Width), where ’Batch’ refers to the size of the training batches, ’Channels’ represents RGB (3), ’Frames’ varies greatly depending on the length of the video, and ’Height’ and ’Width’, after preprocessing, are 1024 by 576 respectively. Inserting video sequences of various depths into the model presents a challenge because most CNN models are not designed to account for varying depths. To address this, we have implemented several techniques, including fixed size padding, max sequence batch, and average pooling techniques. First, the most straightforward method is to pad all tensors to a fixed depth size, using the maximum depth from the dataset and padding the shorter ones with zeros. Second, we can handle this at the batch level, retaining the varying depths as long as the model design can manage variable sequences, such as with LSTM layers. Third, we can use average pooling techniques to average out to a fixed size output. However, this is less than ideal as it may lead to loss of some temporal information. In our ConvLSTM model, we primarily use the max sequence batch technique, although we also utilize other methods depending on the architecture design throughout the project.

3.1.2 Convolutional Normalization

In this experiment, we utilized normalization, which reduced the original pixel value range from 0 to 255 down to a range of 0 to 1 using the Min-Max scaler. However, after a few trials, we observed a noticeable reduction in the speed of loss reduction from each epoch. We found that normalization in the ConvLSTM Autoencoder resulted in higher overall loss compared to trials that did not use normalization. We suspect that the cause of this phenomenon might be due to the sensitivity of the values during training introduced by normalization, which may require further adjustments in learning rate. After comparing results, we decided to use non-normalized visual tensors for the ConvAutoencoder since it yielded significantly better results in fewer epochs.

3.2 Multimodal Ensemble Model

The second part of the experiments involves training the Ensemble Model, a more complex architecture that combines the outputs from the Visual Transformer and the Audio Transformer with the learned visual and audio embeddings from the pre-trained Convolutional LSTM Autoencoder. The Ensemble Model is designed to effectively fuse information from both visual and audio modalities and predict the output values. The parameters and settings applied during this training phase can be found in Table 1.

In addition to the visuals, the Ensemble Model requires input from the audio modality. The ’extract audio’ function is called to extract audio from the video dataset and saves it in .wav format. Subsequently, the ’extract embeddings’ function is used to transcribe the audio dialogue and extract Low-Level Modulation Spectrogram (LLMs) embeddings from the audio files. These LLMs embeddings are crucial for training the Audio Transformer within the Ensemble Model. The sizes of the audio tensors are output as [1, 7, S, 512], where ”S” represents a variable-length dimension that can range from 1 to several thousands. In order to reduce the computational intensity of the model, an average pooling technique was deployed to reduce the dimension of S to 1, while keeping the other dimensions as originally specified before input to the transformer. While this might result in some loss of temporal information regarding sub-parts in speech, we are more interested in the global wording context to aid our model prediction.

3.2.1 Dual Transformer Late Fusion

The Ensemble Model consists of two transformer-based sub-models: the Visual Transformer and the Audio Transformer. Both sub-models share common hyper-parameters, including the number of attention heads, hidden dimension, and the number of transformer layers. Specifically, the ’Number of Attention Heads’ is set to 4, providing the models with multiple attention mechanisms to focus on relevant visual and audio features. The ’Hidden Dimension’ is set to 32, representing the size of the hidden layer in each transformer. Finally, the ’Number of Transformer Layers’ is set to 2, determining the depth and number Transformer encoder blocks [1] of the transformer-based architectures.

Throughout training, the losses on both the training and validation sets are recorded to assess the Ensemble Model’s performance. The ensemble transformer regressor model has a total of 836,484,738 trainable parameters.

3.2.2 Standard Normal Scalar

The nature of this problem made it difficult to generate accurate predictions. Initially, the regression model seemed to predict a constant result around the training mean, leading us to suspect that the model was tending to underfit the problem space. Later, we introduced standard normalization to the target values. Originally, these target values could range between 0 to 10,000s. The standard normalization not only shortened our training time of the Transformer regression model, but also produced results with better variations, closely resembling the high and low values in the validation set. Consequently, we observed improved performance in terms of reduced MSE loss.

3.2.3 Impact of Outliers on MSE and MAE

As mentioned above, we used MAE as part of our loss function. Initially, we only deployed the common regression metric MSE as our main loss function. After a few trials, we observed a pattern where MSE was predicting higher values than most of the targets, seen in Figure 2.

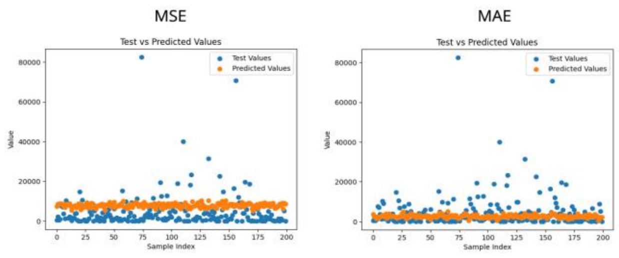

Fig. 2. MSE loss function versus MAE loss function

Since there were a few outliers with high amounts of video views, the norm of the other errors were made insignificant when MSE squared the loss. However, these outliers didn't appear often. When we switched to MAE, we saw significant improvement in generalization, surprisingly even when we evaluated using MSE for both the validation set and the holdout set. Thus, we implemented MAE loss as part of our strategy.

IV. Results

4.1 Convolutional LSTM Autoencoder Results

Initially, we set our Convolutional LSTM Autoencoder with increasingly higher hyperparameters, hoping that it could produce better results. After some trials, we realized that using a smaller model by reducing the hidden size over a high number of epochs was extremely effective in training our ConvLSTM Autoencoder. The total trainable parameters of this model is 1,041,859. Notably, the model showed a considerable decrease in loss after each epoch. During training, we also inspected the visual samples. Noticeably, the reconstructed images continued to improve as the number of epochs increased and the loss decreased, seen in Figure 3.

Samples of various frames predicted by the ConvLSTM Autoencoder model can be seen in Figure 4.

After training the Convolutional LSTM Autoencoder, we analyzed and visualized the losses on the training and validation sets on a graph. The graph shows a trend of continually decreasing losses on both the training and validation sets over the epochs. This indicates that the Autoencoder is effectively learning to reconstruct the input video frames and captures essential visual features in the learned latent space representation. The decreasing loss values affirm that the Autoencoder has learned to produce accurate reconstructions of the original video frames.

To further evaluate the quality of the Autoencoder's reconstructions, a random sample inspection is performed on the validation set. The actual video frames and their corresponding reconstructed versions are visually compared. The results illustrate that the Autoencoder successfully preserves critical details and spatial structures in the frames, indicating its proficiency in reconstructing visual information.

A series of additional experiments revealed that the number of frames to skip has a significant impact on the performance of the autoencoder. When the number of frames to skip was reduced from 200 to 100, the validation loss of the autoencoder decreased by approximately 300%, as seen in Figure 5.

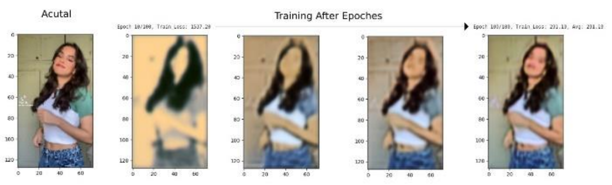
Acutal
Training After Epoches
Epoch 10/100, Train Loss: 1537.29
Epoch 100/100, Train Loss: 231.13, Avg: 231.13

Fig. 3. Training After Epochs.

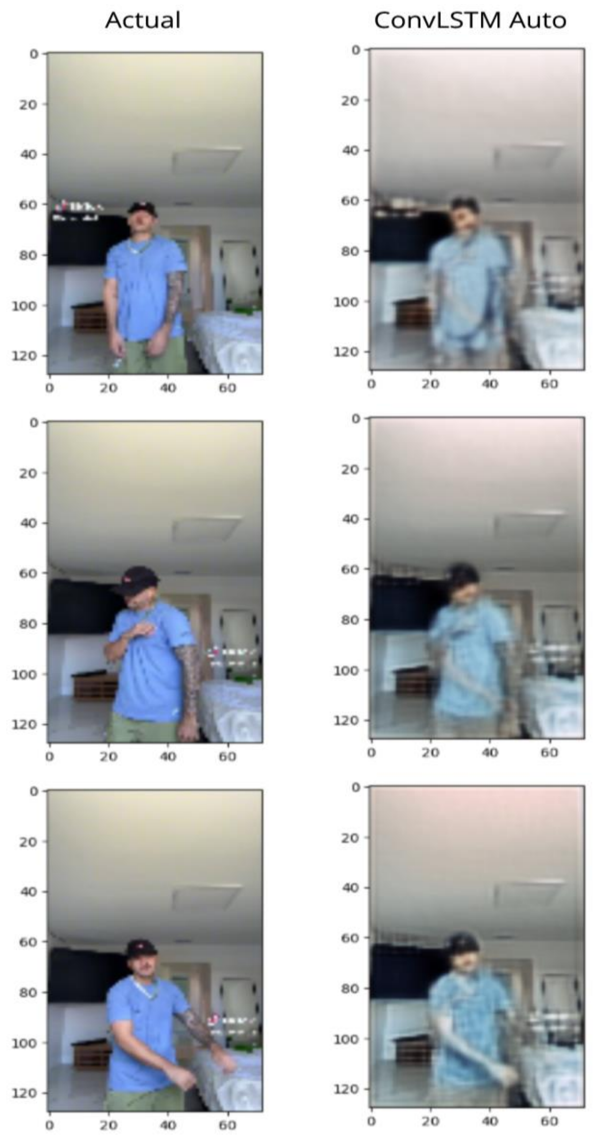

Fig. 4. Video reconstruction samples from ConvLTSM Autoencoder

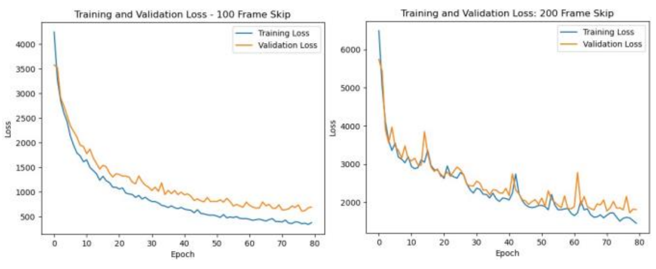

Fig. 5. Losses in comparison for Autoencoder visual depth

Figure 6 demonstrates that the autoencoder was able to reconstruct the original image much better with the lower frame skip.

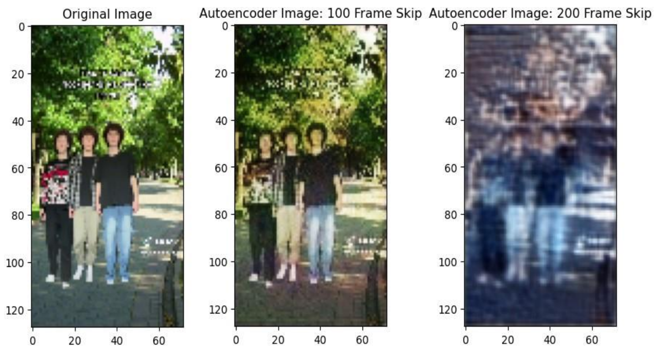
Original Image
Autoencoder Image: 100 Frame Skip
Autoencoder Image: 200 Frame Skip
0
20
40
60
80
100
120
0
20
40
60

Fig. 6. Autoencoder reconstruction comparison.

However, this experiment was computationally expensive, requiring 80 epochs, each of which took an average of 40 minutes to complete. This resulted in a total runtime of over 2 days. Therefore the rest of experiments are run using 200 frame skips.

4.2 Multimodal Ensemble Regression Results

With our test set constructed of 100 randomly sampled videos the Multimodal Ensemble Model was able to outperform all models except for the Swin Transformer (w/Scaled Target). From the MAE we are able to gain further insights from the model's predictive ability and we observed the Multimodal Ensemble Model was able to better handle non-outlier predictions compared to the Swin Transformer (w/Scaled Target). The results showcased the architecture which features a combination of an Auto-encoder and Transformer was able to begin to learn a function that is not solely over-biased in over estimating the view count due to the nature of the dataset that includes a small percentage of video gaining immense amount of views. The table below shows the breakdown between the different models and performance metrics.

TABLE I. MODEL TEST PERFORMANCE

Model	MSE	MAE
SwinTransformer	83.96M	4.73k
SwinTransformer(ScaledTarget)	70.2M	5.39k
CNN3D	82.97M	4.87k
MultimodalEnsembleModel	72.96M	4.89k

Given the nature of the dataset, which is a sample set of videos, there's a probability that any dataset used will exhibit some heteroscedasticity. This is because TikTok, as a platform, has a diverse range of videos with varying levels of popularity. Videos can range from being part of viral trends to more specialized content. To handle heteroscedasticity, MAE might be a better metric. From this, we can observe that the Swin Transformer performs well with nonscaled values, while it suffers when using scaled values. The Multimodal Ensemble Model was trained using normalized scaled values and tends to achieve balanced results in both MSE and MAE.

4.3 Multimodal Classifier Variation Results

These classes serve as proxies for the expected success of videos. For instance, videos in class 4 are expected to generate more views than 75% of all videos. In simpler terms, we experimented with predicting the quartile of viewership in which a video might land. This mapping could also be useful for content creators trying to predict whether their video is likely to go viral or fall into the least-viewed quartile. The quartile thresholds were derived from the training set of videos and applied to both the validation and test sets. To further illustrate the complexity of this problem, even when the problem is modeled as classification, all the different algorithms recorded an accuracy of under 30%. The algorithms tend to overfit the noise rather than making meaningful predictions. The simplest model, CNN 3D, performed the best, followed by the Swin Transformer with an accuracy of 27%. Due to time limitations, we didn't explore and train the ensemble model classification as extensively, which resulted in a score of 23%. However, it's worth noting that the parameters used in this experiment differed from those in the regression experiment. Overall, the classification approach may suffer from the imbalance of the classes. As a solution, we could introduce further balancing techniques, or even better, ensure more balanced data collection, which might yield better results in future experiments.

V. CONCLUSION

In conclusion, the results have paved the way for a successful series of experiments in deep learning regression for predicting video success metric. The Multimodal Ensemble Architecture proposed in this paper demonstrates its ability to outperform both a Vanilla CNN 3D and a Swin Transformer Model in MSE evaluation results. Notably, the Swin Transformer was trained on a diverse visual dataset, while the Multimodal Ensemble Model was trained solely on a limited video set. Despite these constraints such as limited computational resources, frame skipping, and video scales, our model still demonstrates competitive results comparable to state-of-the-art models, which is noteworthy. However, due to the vastness of the problem space, the high-accuracy prediction capabilities of all explored models were not fully demonstrated. Nonetheless, our research provides a proof of concept for the multimodal approach, laying the groundwork and establishing a milestone towards the development of general multimodal models.

In future work, more extensive and diverse datasets could be explored to enhance the models' ability to generalize across various scenarios. Additionally, fine-tuning the hyper-parameters and experimenting with different diverse datasets could lead to substantial improvements in predictive demonstration. Despite the current limitations, the experiments lay a solid foundation for future advancements in multimodal deep regression tasks. As the field of deep learning continues to evolve, these preliminary results provide valuable insights and directions for further research and development.

It is important to note that the focus of this work was to introduce and experiment with the multimodal deep regression framework. With further refinements and enhancements, such a framework holds great potential for diverse applications, including audio-visual recognition, video detection analysis, and multi-source data integration. Our experiments showcased the feasibility of the multimodal deep regression approach in handling complex tasks involving visual and audio data. The work sets the stage for future investigations and improvements in this exciting and challenging area of research. By continuing to explore advanced techniques and methodologies, the potential for more accurate and robust predictions in multimodal data analysis can be realized.

TABLE II

ConvLSTMAutoencoder	**Ensemble Transformer Regression**	**Swin Transformer**
Learning Rate: 1e-4	Learning Rate: 1e-9	Epochs: 20
Epochs: 200	Epochs: 7	Dropout: 0.1
Hidden Size: 64	Hidden Size: 32	Embed Dim: 96
Frame Skip: 200	Number of Attention Heads: 4	Patch Size: (4, 4, 4)
Batch Size: 4	Dropout: 0.1	Window Size: (2, 7, 7)
Shrink: 8	Number of Layers: 4	Depths: [2, 2, 6, 2]
Padding: False	Late Fusion: True	Number of Attention Heads: [3, 6, 12, 24]
Normalize: False	Audio Transformer: True	Patch Normalization: True
Adam Optimizer Weight Decay: 0	Adam Optimizer Weight Decay: 1e-3	Adam Optimizer Weight Decay: 1e-4
Adam Optimizer Learning Rate: 1e-4	Adam Optimizer Learning Rate: 1e-9	Adam Optimizer Learning Rate: 1e-4

References

[1] Ashish Vaswani and Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *In NeurIPS*, pages 5998—6008, 2017.

[2] Said Yacine Boulahia, Abdenour Amamra, Mohamed Ridha Madi, and Said Daikh. Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Machine Vision and Applications*, 32(121), 2021.

[3] Qiliang Chen, Hasiqidalatu Tang, and Jiaxin Cai. Human action recognition based on vision transformer and l2 regularization. *In Proceedings of the 2022 11th International Conference on Computing and Pattern Recognition (ICCPR '22)*, pages 224–228, 2022.

[4] Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. Early vs late fusion in multimodal convolutional neural networks. *In Proceedings of the 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Rustenburg, South Africa*, pages 1–6, 2020.

[5] Daya Guo, Jiangshui Hong, Binli Luo, Qirui Yan, and Zhangming Niu. Multi-modal representation learning for short video understanding and recommendation. *In 2019 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 687–690, 2019.

[6] Qin Jin, Jia Chen, Shizhe Chen, Yifan Xiong, and Alexander Hauptmann. Describing videos using multi-modal fusion. *Proceedings of the 24th ACM international conference on Multimedia*, pages 1087—-1091, 2016.

[7] Lynnette Hui Xian Ng, John Yeh Han Tan, Darryl Jing Heng Tan, and Roy Ka-Wei Lee. Will you dance to the challenge? predicting user participation of tiktok challenges. *In Proceedings of the 2021 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM'21)*, pages 356–360, 2021.

[8] Andreas Holm Nielsen. Video prediction using convlstm autoencoder (pytorch). 2020.

[9] Crystal J. Qian, Jonathan D. Tang, Matthew A. Penza, and Christopher M. Ferri. Instagram popularity prediction via neural networks and regression analysis. *In IEEE Transactions on Multimedia*, pages 2561–2570, 2017.

[10] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2209.01109*, 2022.

[11] Xingjian Shi, Zhourong Chen, Hao Wang, and Dit-Yan Yeung. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[12] Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early versus late fusion in semantic video analysis. *In Proceedings of the Annual ACM International Conference on Multimedia, Singapore*, pages 399–402, 2005.

[13] Georgios Tziafas and Hamidreza Kasaei. Early or late fusion matters: Efficient rgb-d fusion in vision transformers for 3d object recognition. *arXiv preprint arXiv:2210.00843*, 2023.

[14] Massimiliano Viola, Luca Brunelli, and Gian Antonio Susto. Instagram images and videos popularity prediction: a deep learning-based approach. *Universita degli Studi di Padova,` Padova, IT*, 2021.

[15] Jiaheng Xie, Yidong Chai, and Xiao Liu. Unbox the blackbox: Predict and interpret youtube viewership using deep learning. *Journal of Management Information Systems*, pages 541–579, 2023.