ResQwen: A Multimodal Model for Predicting YouTube Video Engagement

Arnaudo Baptiste
École polytechnique
baptiste.arnaudo@polytechnique.edu

Chikhi Salah
École polytechnique
salah.chikhi@polytechnique.edu

Abstract

We present ResQwen, a multimodal model trained to predict YouTube video engagement. The model leverages visual, textual and numerical signals from the video to estimate its number of views. The relationship between the data and the target views being extremely erratic, multiple key ideas are necessary to build a satisfying and robust model. In particular, to better handle videos with abnormally high or low view counts, we introduce a data augmentation pipeline that increases the robustness of our model by perturbing titles, thumbnails and timestamps. Coupled with a specific training scheme on approximately 15,000 real-world videos and appropriate dataset design choices, ResQwen achieves a final MSLE loss of 3.4.

1. Introduction

The goal of this project was to consistently predict the number of views a given YouTube video would get. To accomplish this goal, a dataset of approximately 15,000 real-world YouTube videos were at our disposal. More precisely, the given dataset was curated from a limited number of channels with data spanning from 2011 to 2023. Our objective was then to forecast view counts for videos released in 2024 and early 2025.

In order to achieve this, a certain number of features were at our disposal, namely:

- The video's thumbnail
- The video's title
- The video's description
- The channel's name
- The date of publication of the video

This task is rather complex due to the multimodal nature of the data that is given to us. Indeed, in order to satisfyingly predict the number of views, one must make use of textual data as well as visual signals.

Furthermore, the distribution of the number of views is complicated. Indeed, a certain number of videos act as outliers as they have an abnormally high or low amount of views. Those are going to be particularly hard to predict due to the lack of key signals and since they are underrepresented in the dataset. This observation proves the need to conduct a proper exploratory data analysis.

2. Exploratory data analysis and feature engineering

The dataset given to our disposal showcased videos whose views vary from 0 to almost 200 million. Its distribution is very skewed; when plotting the views' logarithm, we notice a "normal distribution" (Figure 1)

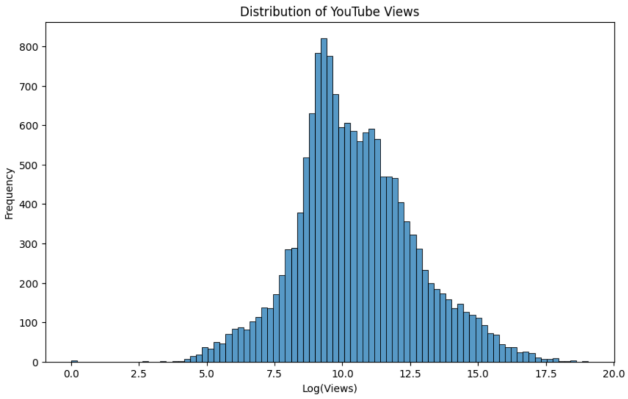

Figure 1. Distribution of the views' logarithm

We decided to use all of the features that were given to us in the dataset. However, we used the video's description in a particular way as we only extracted the number of hashtags and the number of links. Hashtags allow to categorize the video and present it to specific audiences while the number of links in the description allows us to know if the channel is already well established (links to merch, website, other social media...) or not. Let us now focus on the specifics of some of these features.

2.1. Temporal features (Salah)

The dataset provides the publication date in ISO 8601 format, from which valuable temporal features can be extracted. The context in which a video is published can significantly influence its performance. For instance, videos published on a Saturday may receive more views early on, increasing their chances of being flagged as trending, and thereby gaining additional visibility.
Based on this intuition, we extracted features such as the month of publication (relative to the year), the hour of publication (relative to the day), and the day of publication (both within the week and within the month). We then cyclically encoded these features with sine and cosine. For example, the month of publication (relative to the year) was replaced with the couple $\left(\cos\left(\frac{2\pi\times\text{month}}{12}\right), \sin\left(\frac{2\pi\times\text{month}}{12}\right)\right)$ reflecting the fact that, contextually, publishing a video at the end of a year is similar to publishing it at the beginning of the next.

We applied min-max normalization to the year of publication, based on the assumption that videos published in consecutive years are more similar in context than those separated by several years. As a result, the earliest videos in the dataset (published in 2011) were assigned a normalized year value of 0, the most recent ones (published in 2025) were assigned a value of 1, and all others received a value between 0 and 1, scaled linearly according to their publication year.

2.2. Channels (Baptiste)

The channel that publishes a video can be a key factor influencing its view count, as more popular channels tend to consistently produce high-performing content. To capture this effect, we one-hot encoded the 46 unique channels present in the dataset. Furthermore, when training a small Random Forest model to regress the logarithm of the view count, we observed that 7 channels stood out in terms of predictive power (see Figure 4). Consequently, it is important to ensure that the proportions of these influential channels are consistent across the training, validation, and test sets to avoid distributional bias.

2.3. Dataset creation (Salah)

We aimed to construct a validation set composed of recent videos, in order to better reflect the temporal distribution of the test set, which contains only very recent content. Additionally, we ensured that the distribution of the most influential channels in the validation set closely matches that of the test set, and that the distribution of the logarithm of the view count remains close to that of the original dataset. Once the validation set was created, we reserved a small portion of data as our personal final test set, and defined the training set as the remainder. The distributions can be found in Figure 2 and Figure 3.

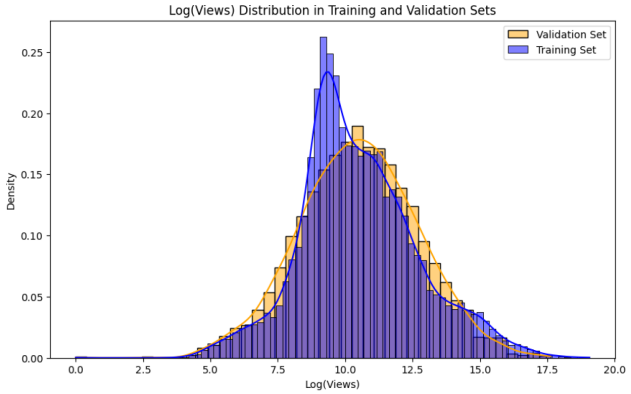

Figure 2. Views' logarithm in Training and validation datasets

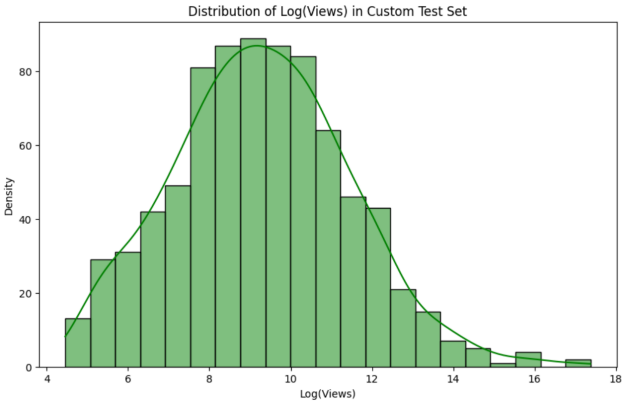

Figure 3. Views' logarithm in custom Test dataset

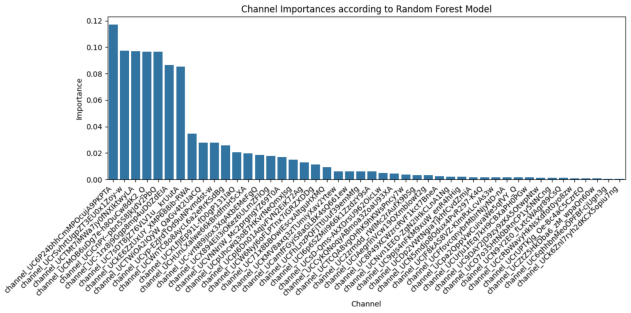

Figure 4. Channels' importance in predicting the target

3. Model design & training

In this section, we present our model and the training process we developed to obtain our final loss of 3.4.

3.1. Explored models (Salah)

Before talking about ResQwen, let us first mention the models that we explored early on.

Our first approach relied on raw features, namely the thumbnail that was passed onto ResNet50 (introduced in [4]) and a multilayer perceptron (MLP) which naturally lead to a loss of 5.0. We then took a multimodal approach using CLIP's [1] image part for the thumbnail, Bert Base multilingual Cased (chosen because it supported Bengali, English and Emojis) for the title, and a couple MLPs which reduced the loss to $\simeq 4.7$. As it turns out, simply exchanging CLIP by ResNet50 coupled with a proper training process lead to a major improvement to a loss of 3.8.

A key phenomenon observed in our experiments is that our models (especially those with deep MLPs) tend to overfit extremely fast (in one or two epochs). Moreover, our best performances were obtained with a low number of epochs ($\simeq 3$ were sufficient). This lead us to explore another alternative. Namely, we replaced our final MLP by an ensembling method of both an XGBoost model and a RandomForest model. However, this did not yield any major improvements.

3.2. ResQwen : Definition of the model (Baptiste)

Our model followed a simple architecture. We started off by encoding the visual and the textual input individually (the title through Qwen3-0.6B[2], the thumbnail through ResNet50). Do note that the encoding of the title is done using Qwen3's antepenultimate layer as it yielded the best results. Moreover, to address memory constraints, we decided to quantize Qwen3 to 8 bits. While Qwen3 is not traditionally designed as a text encoder (being primarily a large language model trained for generation and reasoning) we found it to be effective in our setting. Indeed, it supports Bengali, English and Emojis. Moreover, by extracting representations from its intermediate layers, we leveraged its broad pretraining on diverse internet data, which likely captures textual patterns relevant to YouTube titles. Although this choice may seem unconventional compared to standard encoders like BERT or CLIP, it was experimentally validated in our setup.

Everything is then passed into its own MLP with dropouts. The choice of dropouts here is crucial. Indeed, we saw that our model tends to overfit extremely easily. Therefore, we decided to use a significant amount of dropouts (with a probability of 0.3) in each MLP.

Having done this, we obtain 4 vectors that we concatenate into a unique vector which is then passed through a final MLP with its own dropouts. The entire architecture is detailed in Figure 5.

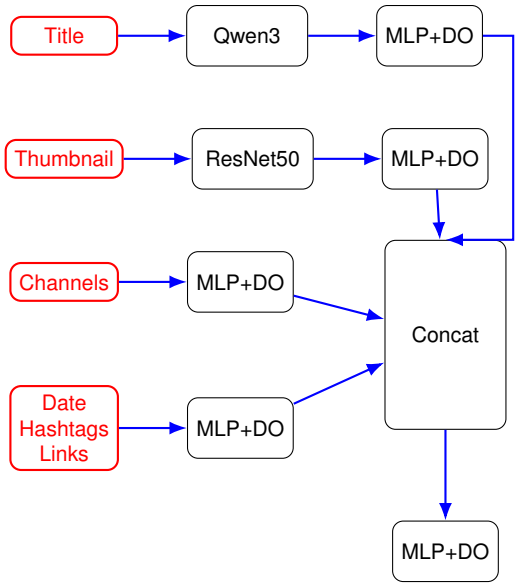

Figure 5. Architecture of ResQwen

We pass each separate input into its own MLP in order to specialize the first set of 4 MLPs. Indeed, each MLP before concatenation of the vectors will be specialized in either thumbnails, title, channel names etc...

Let us now briefly mention the dimensions of each vector prior to the concatenation. The thumbnail's MLP takes a vector of size 2048 given by ResNet50 and yields a vector of size 128. The title's MLP takes a vector of size 768 given by Qwen3 and also yields a vector of size 128. The tabular data (temporal features, year...) is encoded through a vector of size 32, and the channels through a vector of size 64. In doing so, we allocate a larger number of weights to inputs such as the title and the thumbnail which, we believe, require more work by the model to explain the number of views.

3.3. ResQwen: training phase (Salah)

The training phase is also a key part of the model. Indeed, as we have claimed earlier, it is very easy for our MLPs to overfit, hence the importance of this phase.

We divided the training into 2 main phases with an extra optional one.

- **Phase 1:** General training
  - Freeze ResNet50 and Qwen3
  - Only train the MLPs
  - Use a learning rate of 10^{-3}

- **Phase 2:** Finetuning phase
  - Unfreeze the last layer of ResNet50 for finetuning
  - Train the MLPs
  - Use a learning rate of 10^{-5}
- **Phase 3 (optional):** Train the model on the training and the validation sets before making a submission.

Hence the first phase serves a global training purpose whereas the 2nd phase is used to finetune our model. We used multiple optimizers, namely Adam and SGD (with Nesterov momentum) and noticed that Adam yielded the best performances.

During the training, we also noticed using Weights&Biases that certain layers were not properly learning (low gradient norm) and that convergence was relatively slow. To address this, we explored different solutions and made several adjustments to the architecture. The outcome of these changes, and their impact on learning, are discussed in more detail in the following results section.

4. Results

In this section, we present the results of our model for different experiments.

4.1. Robustness of the model (Salah)

At first, our model's performance was very erratic, often jumping from a loss of $\simeq 5$ to $\simeq 3.9$. Our experiments lead us to understand that our predictions on the outliers of the dataset were one of the causes of this problem. This can be seen in the following figure:

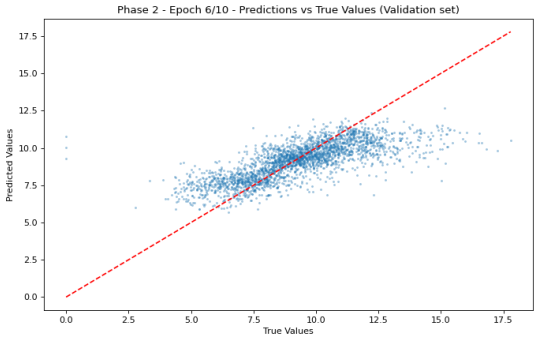

Figure 6. Model underperforming on outliers (Predicted values vs True values, log scale)

As we can see, our model overshoots on videos with low views and undershoots on videos with high amounts of views. We attempted to fix this problem by a couple of methods. Those included **data augmentation on outliers** (which yielded a new training dataset prior to the training phase), **upsampling of outliers** and **penalizing the loss on outliers**. More precisely, we chose the following loss that penalizes videos with a logarithm of views value larger than 10:

$$\underbrace{\frac{1}{n}\left(3\sum_{1\leq i\leq n, y_i>10}(y_i-\hat{y}_i)^2+\sum_{1\leq i\leq n, y_i\leq 10}(y_i-\hat{y}_i)^2\right)}_{\mathcal{L}(y,\hat{y})}$$

This yielded somewhat better results but does not entirely fix the issue of outliers. Indeed, outliers in this challenge are particularly hard to predict with our data. It also seemed that our model had trouble generalizing and often relied on the mean value of the training dataset. In order to deal with this issue, we implemented data augmentation on all thumbnails **during the training phase** by applying random rotations or horizontal flips, and by changing the thumbnails' colorimetry.

The results of our attempt to fix these issues were mixed. We did see an improvement in the variance of the values yielded by our model and a somewhat better performance on outliers, but it was not as drastic as one could have hoped. This is a lead for improvement.

4.2. Ablation studies and results (Baptiste)

To study the influence of each feature, we conducted ablation studies by training the full model for 5 epochs on the training set, and then evaluating it on a validation set where one feature at a time was replaced by a tensor of zeros. This allowed us to measure the contribution of each feature to the model's predictive performance at inference time. The results are summarized in Table 4.2.

Feature Masked	**Validation MSLE**	**Gap**
None (baseline)	2.35	0.000
Title (text)	2.39	+0.004
Thumbnail (image)	2.50	+0.015
Channel ID	2.56	+0.021
Publication Date	2.40	+0.005

Table 1. Ablation study: validation performance when masking individual features.

As expected, the thumbnail is an important feature for predicting the number of views. The channel ID also proves to be highly influential, as previously discussed; while the publication date does not prove to be as crucial. Surprisingly, the title appears to be even less important. This is intriguing, and may be partly explained by the fact that Qwen is not specifically designed for textual encoding: the model may fail to fully capture the sentiment or semantic nuances conveyed in video titles.

4.3. Improved convergence with better initialization (Baptiste)

During training, we observed that certain layers (particularly those within the MLP blocks) were not learning properly, as indicated by stagnant gradients and unchanging weights (see Figure 7). This issue was attributed to the use of ReLU activation functions, which can suffer from the "dying ReLU" problem: when a neuron's input becomes negative, ReLU outputs zero and the gradient vanishes, effectively halting any further learning for that neuron. To mitigate this, we replaced ReLU with LeakyReLU(0.01), which allows a small, non-zero gradient to flow even when the input is negative. This simple modification ensured that all neurons remained active throughout training and significantly improved the learning dynamics of the network. When combined with Kaiming initialization, which is well-suited for ReLU-based activations, the use of LeakyReLU helped stabilize training and improved overall model convergence.

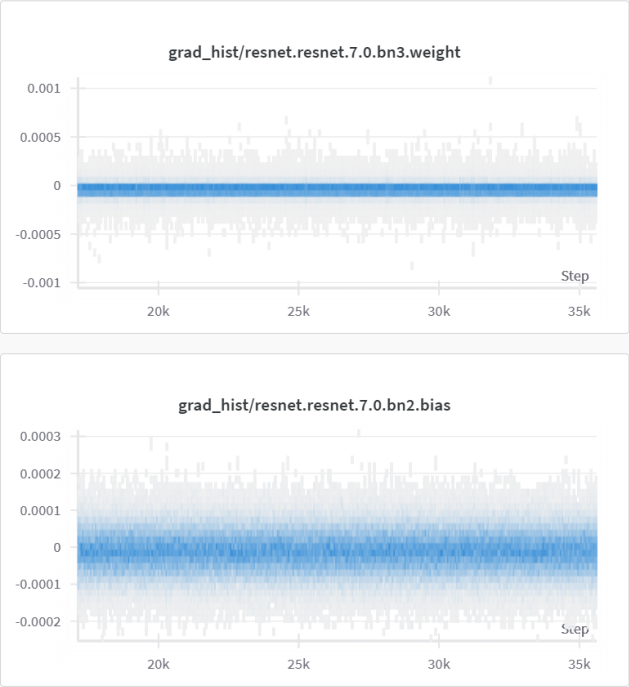

Figure 7. Comparison between a "healthy" neuron (below) and a dead neuron (above)

4.4. Final results on Kaggle

Our final model ResQwen coupled with the methods presented in this report (data augmentation, training phases...) have yielded a final result on the public test set (30% of the test data) with a loss of 3.8.

Passing onto the private set (100% of the data) has improved our loss to a final value of 3.4. One can explain this by noticing that the main problem our model encountered were the outliers that were increasing our loss by a vast margin. Supposing that the outliers are less frequent in the final dataset, our model performs better on average as it is faced with more videos near the center of the distribution

5. Conclusion and perspectives

Predicting the number of views a YouTube video will receive is a challenging task due to the multiple modalities of input and the highly erratic nature of the target variable. In this work, we introduced ResQwen, a simple yet effective multimodal model that leverages both visual and textual signals alongside structured metadata. By carefully designing both our dataset and our model architecture, ResQwen achieves a final MSLE loss of 3.4, demonstrating strong predictive capabilities on real-world data.

This project taught us the importance of simple models and feature engineering. It also highlighted several leads for further improvement. Those include but are not limited to :

- Rather than trying to precisely explain outliers (which appeared to be a difficult task given our data), approaching the problem as a classification task (e.g. categorizing videos into popularity tiers) could help form a better first rough estimate of the number of views.
- Exploring Cross-Attention models in order to jointly analyze the thumbnail and the title (it is intuitive that these two inputs are linked in some way) in order to capture more subtle signals.
- Enhanced feature engineering such as extracting more relevant information from the description like genre, actors, or licenses..
- Explore more ideas and concepts that can be extracted from the title and the thumbnail such as the general sentiment or the presence of faces and their expressions (this is often referred to as clickbait, widely used to attract views on YouTube). This idea has been explored in [3].

References

[1] Alec Radford et al. Learning transferable visual models from natural language supervision, 2021. 3

[2] An Yang et al. Qwen3 technical report, 2025. 3

[3] Geng Cui et al. Clicks for money: Predicting video views through a sentiment analysis of titles and thumbnails. *Journal of Business Research*, 183:114849, 2024. 5

[4] Kaiming He et al. Deep residual learning for image recognition, 2015. 3