CS230

Predicting Virality: Augmenting Creators with Predictions on Content Performance

Brian Wu, Kevin K Yang, Aryan Chaudhary
Department of Computer Science
Stanford University
brianwu@stanford.edu, kevvyang@stanford.edu, achaud@stanford.edu
Code: CS230 Final Project Github

1 Introduction

There has been no better time to become a creator. The COVID-19 Pandemic has enabled society to transform many of its functions to a digital medium - among them, one of the most prolific is the Creator Economy, a collection of software running on the internet that allows users to earn money from their creations. Examples of platforms within the Creator Economy include YouTube and TikTok (for Video content), Fiverr (for services that can be transmitted over the internet), and Spotify (for content that can be distributed in an audio format). Today, creators are constantly experimenting with various ways to grow and scale their platforms from the content that they create, particularly due to the fact that these platforms can only generate revenue if they are able to scale.

However, the main challenge that exists in the creator economy is that it is impossible to predict the engagements a specific piece of content will receive exactly, which means that there is a signfiicantly level of uncertainty as to how much creators are able to profit off of their work. Furthermore, sometimes the number of engagements a piece of video content gets is directly correlated to how appealing the cover image is, regardless of the actual content itself, which can directly influence how a creator is able to produce.

In this study, we propose a Convolutional Neural Network-based architecture that has been trained on thumbnails of YouTube videos from varying categories and creators of varying virality that will be able to predict the amount of views that a certain video can hypothetically receive, taken into context within its category. To ensure that the model is best able to learn the individual features that exist within a diverse array of YouTube thumbnails, we'll explore transfer learning on State-of-the-Art (SOTA) image recognition Convolutional Neural Networks by applying our dataset to the pre-trained weights from these models. This process explored the application of YouTube image thumbnails to SimCLR, a contrastive learning framework for visual representations that is baselined on a ResNet-50.

2 Dataset and Features

The Dataset for this study is adapted from Kaggle's (Youtube Thumbnail Dataset), supplemented by additional YouTube images that we have manually scraped from YouTube and hand-labeled. The YouTube thumbnail dataset contains top performing video thumbnails from 91 of the most popular YouTube channels (spanning 10 different categories that we will classify each thumbnail into), totaling 2515 examples. The dimensions for these images are 1280 x 720 - the standard resolution for high-definition 1080P images and video. These images were prepossessed by reshaping them into 180x180 images so that they can better be trained using the SimCLR architecture that we have selected. This data could not be augmented in the traditional sense (because YouTube thumbnails typically are presented in only one orientation), but additional data augmentation was done by scraping YouTube thumbnails at the same resolution manually and then hand labeling/preprocessing them. Due to the smaller amount of examples we selected, we elected to split the dataset into a train/validation distribution of 80%/20%.

We will be using the thumbnails of each video combined with the number of views it received and its genre to train and test our model. This dataset does not explicitly provide us with the number of views that a given video received; rather, it gives the unique ID that is appended to the URL. Therefore, we have implemented a scraper that can search the webpage for the number of views which we will then use in our neural network.

3 Model Architecture

Given the small amount of examples that we currently have, an optimal strategy would be to use transfer learning - taking advantage of an existing CNN architecture. The architecture that we have selected for this project is the SimCLR architecture, which is a simple framework for contrastive learning of visual representations. SimCLR's advantages are in that it's relatively lightweight network given its capabilities, having used the ResNet-50 architecture as a foundation. Furthermore, the authors of the original SimCLR paper have reported that, after fine-tuning on only 1% of the labels, they were able to achieve a top-5 accuracy of 85.8%, which outperforms the AlexNet architecture with over 100x less models (Chen et al. 2020) We'll be using transfer learning in this research by directly applying SimCLR's weights to our data, and by training over a timestep of 1000 epochs on a NVIDIA Tesla v100 GPU provided through Google Cloud Platform. Our loss function is similar to that of SimCLR:

$$L_{i,j} = \frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} indicator_{k \neq i} exp(sim(z_i, z_j)/\tau)}$$

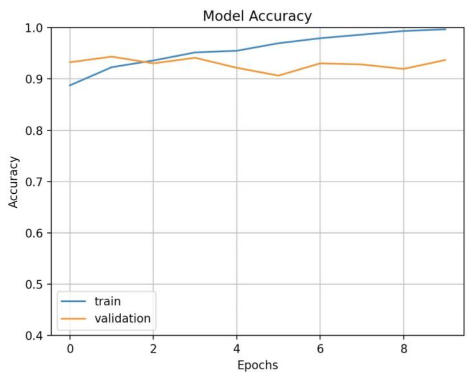
where N is the number of examples in the minibatch, τ is the temperature parameter, $sim(a, b)$ represents the cosine similarity of two vector representations of image examples, and the indicator function is a function that evaluates to 1 if k is not equal to i. The SimCLR architecture uses contrastive learning between a pair of image examples at a time - hence the use of the cosine similarity of two image embeddings in the loss function.

Our model progression began with a simple implementation of ResNet-50 where we used a binary classifier to learn whether a given thumbnail had over or under 1 million views. In order to output the correct data, we added a few output Keras layers like Flatten and Dense which reshaped the data into the desired shape. We also played around with the learning rate and found that 0.01 worked better for this task than the predesigned 0.001. The accuracy of this model can be seen in the first graph under Results. Our next step was to update the number of classes to create ranges for views: we used a log function to determine the number of digits in the view count and split the data set up into categories of 5, 6, 7, and 8 digits. With this new dataset split, we ran ResNet-50 again using Categorical Crossentropy loss with a softmax output, which can be seen in the second graph in Results.

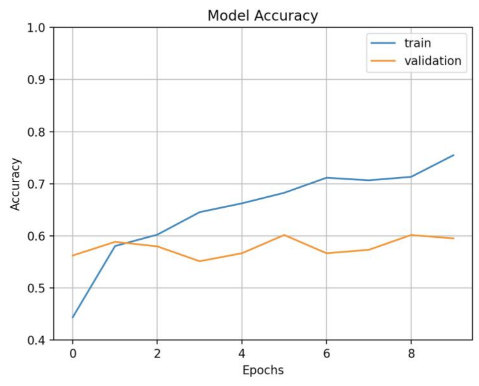
After implementing ResNet-50, we attempted SimCLR as the primary model for thumbnail classification. In the process of doing this, we came across another pretrained model called NNCLR (Nearest-Neighbor Contrastive Learning of Visual Representations) which was built off of SimCLR and was proven to have increased performance on certain types of datasets. However, we chose to stick with SimCLR because NNCLR was a relatively recent release that we were not sure would fully adhere to our intended task.

According to the original SimCLR paper, the ideal hyperparameters for the model were a LARS optimizer, batch size 4096, learning rate of (0.3 * batch size / 256), and 100 epochs. However, after running multiple tests we learned that the ResNet-50 hyperparameter setup was much more optimal for the thumbnail classification task: Adam optimizer, batch size 32, learning rate of 0.001, and 10 epochs. We ran SimCLR on the binary classification task followed by the upgrade to the multi-class classification task. The results for these runs are visualized in the following section.

4 Results

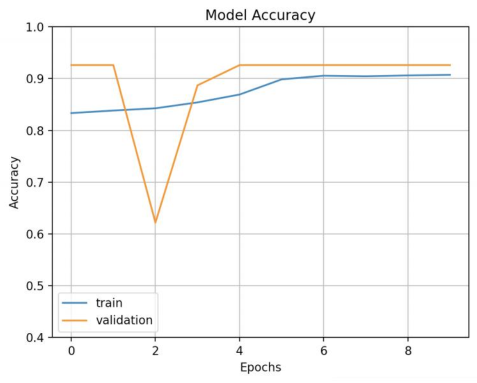


(a) Binary Classifier

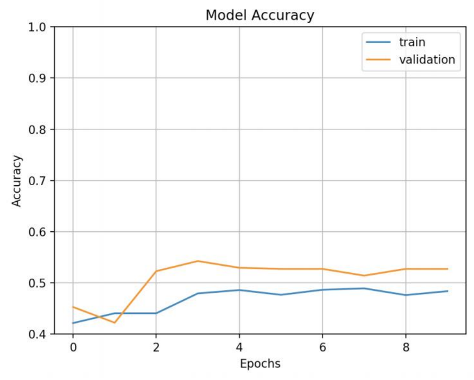


(b) Multi-class Classifier

Figure 1: ResNet-50 implementation



(a) Binary Classifier



(b) Multi-class Classifier

Figure 2: SimCLR implementation

5 Analysis

Looking at these accuracy graphs, it becomes evident that the SimCLR did not perform as well on this task as the original ResNet-50. For the basic binary classification task, the ResNet-50 model achieved nearly 100% accuracy on the training set and around 95% on the validation set. On the other hand, the SimCLR implementation had a noticeable dip in validation accuracy before plateauing around 92%, outperforming the train accuracy which plateaued around 90%. Similarly for multi-class classification, ResNet-50 reached 75% and 60% accuracy on the train and validation sets respectively while SimCLR reached 48% and 54% on the train and validation sets respectively.

This data indicates that the model did not really learn any new representations past 4/5 epochs and was outputting the same predictions. After experimenting with a greater number of epochs and smaller/larger learning rates, we discovered that this repeated occurrence may have been the result of our limited data set size. With a set batch size of 32 and a thumbnail dataset with 2515 images, it was apparent that more data was necessary to learn better mappings between images and view counts. The pretrained weights from SimCLR did not have a significant impact on the model's performance; in fact, ResNet-50 ended up being more useful for this task.

Looking at the disparity between the train and validation accuracy for SimCLR, we can also observe that the model underfit rather than overfit on the training data. To remedy this, we tried various configurations like increasing model complexity by adding more Dense layers in the output, increasing training time by adding epochs, and experimenting with Dropout layers in different locations. In the end, our current model and hyperparameter configurations turned out to be the most successful at the thumbnail classification task.

6 Insights and Discussions

Our algorithm actually performed much better than we expected on this thumbnail classification task. Considering the fact that SimCLR accuracy peaks at around 70-75% and how limited our dataset was, achieving a model accuracy of 54% is quite admirable. Analyzing the results led us to believe that there was simply not enough data for the model to learn better representations after the first few epochs, which makes sense because our dataset consisted of only 2515 examples. We actually identified this limitation early on and attempted to manually scrape more data from YouTube to expand our dataset, but unfortunately encountered numerous challenges involving Google guidelines and scraping the thumbnails themselves. The model tended to converge quite rapidly due to the size of the dataset which made training on larger epochs obsolete.

Ultimately, we could have benefited from trying out more types of models to figure out which performed the best on the thumbnail classification task. Since SimCLR used the basic architecture of ResNet-50, the low-level structures were essentially the same with some implementation differences. SimCLR was supposed to perform better on image datasets but from our experience it did not generalize too well to our task.

YouTube thumbnail classification/ view count prediction is definitely a unique area to explore but we believe that our results indicate a definite correlation between certain types of thumbnails and how popular the video ends up becoming. Obviously, it is difficult to factor in inherent creator popularity (e.g. a creator with more followers will end up getting more views regardless) and personalization techniques that YouTube implements for its users, but excluding these factors our work shows noticeable progress in the fields of deep learning and computer vision.

7 Contributions

Aryan:
Wrote the Model Architecture, Results, and Analysis sections
Created the filemover.py and scraper.py and contributed to both the ResNet-50 and SimCLR models.

Brian:
Wrote the Introduction, Dataset and Features, and Model Architecture
Created poster and recorded video for the team. Debugged Dataset construction and compilation issues with SimCLR.

Kevin:
Wrote Analysis, Insights and Discussions, Contributions, and References
Contributed to both the ResNet-50 and SimCLR models.

References

[1] Bayless, J. (2020) Analyzing YouTube Thumbnails: What Makes The Perfect Thumbnail. In *Best SEO Companies*. https://www.bestseocompanies.com/blog/youtube-thumbnails/

[2] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020) A Simple Framework for Contrastive Learning of Visual Representations. *ICML*. https://arxiv.org/pdf/2002.05709v3.pdf

[3] Mukhopadhyay, P. (2022) YouTube Thumbnail Dataset: Top performing video thumbnails from over 90 YouTube channels. *Kaggle*. https://www.kaggle.com/datasets/praneshmukhopadhyay/youtube-thumbnail-dataset/

[4] Pretorious, K. & Pillay, N. (2020) TA Comparative Study of Classifiers for Thumbnail Selection. *ArXiv*. https://doi.org/10.1109/IJCNN48605.2020.9206951

[5] Yu, Z., & Shi, E. (2020) A Multi-modal Deep Learning Model for Video Thumbnail Selection *ArXiv* https://arxiv.org/pdf/2101.00073.pdf

[6] Abadi, Martx27;in, Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., $\ldots$ others. (2016). Tensorflow: A system for large-scale machine learning. In 12th $USENIX$ Symposium on Operating Systems Design and Implementation ($OSDI$ 16) (pp. 265–283).

[7] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2. (Publisher link).

[8]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science Engineering, vol. 9, no. 3, pp. 90-95, 2007.