Mane 4962 Final Report

Using Convolutional Neural Network Regression to Estimate the Social Performance of Instagram Content

Greta Perez-Haiek

perezg3@rpi.edu

Github: https://github.com/mythicane/Social-Performance-Estimator

April 24th, 2024

Executive Summary

Being a content creator in this ever-changing world of trends is starting to become a game of chance rather than a game of strategy. While one post (for example, on Instagram) might succeed in giving a satisfying number of likes, other posts might fail to become popular for seemingly no reason. To the untrained eye, this might just be a consequence of the Instagram algorithm limiting exposure or reach randomly, but to the eyes of a ML Engineer, there are some patterns that appear in the behavior of followers. By mathematically modeling a function that predicts in advance how many likes an image would get, content creators would be able to make more informed choices on what (and whatnot) to upload to their profile.

The problem that I will be investigating involves putting a pin on the behavior of my personal followers on instagram: Every post that I upload involves my face in some way, so my assumption is that all of my posts would receive a consistent amount of likes and comments by the same individuals. However, despite all 826 of my followers having access to my account at all times, some pictures naturally receive more likes and camaraderie than others. I am not an instagram influencer: however, I can see why others (especially those financially depending on their instagram performance) would want to track these patterns in accordance to their audiences.

Machine Learning is a useful approach in mathematically consolidating these behaviors and putting trends to application. Assuming all 185 of my private instagram images as inputs and the number of likes as features, a Convolutional Neural Network Regression approach will analyze the images (in the form of this feature vector), then predict post popularity (in the form of numerical integer "likes") for each image. This algorithm, called the "Social Performance Estimator", will learn from its mistakes and tune itself to minimize the "Mean Square Error" between the predicted amount of likes and the target amount of likes for each image. This neural network model will also undergo various architecture design iterations to better ensure accuracy, including modifying the number of epochs, the types of layers (and the order at which they appear), and the dataset to align towards a single theme (selfies/pictures of myself).

A training and validation graph plotting the number of training iterations and the "Mean Square Error Loss" was made which ensured that the model was properly learning and performing towards expected standards. A demonstration was also conducted on an image never before seen by the model, which showcased a percentage difference of 8% between the predicted number of likes and the true number of likes that the image has received.

While this model reached expectations for now, there are still imperfections in the approach taken. For example, the dataset is slightly unbalanced. From approximately 10 likes to 200 likes, a healthy number of images can be assumed spanning the entire range. Above 200 likes, however, the density of images became sparse. Because of this gap in the dataset, it is to be assumed that the model will have a slight bias to rate images between 10 and 200 likes as opposed to ranges above 200. Other ways that the dataset could be improved is increasing the dataset count by including more images that would allow the model to learn from more, or increasing the number of features that's associated with each image's feature vector, potentially through more advanced feature vector extraction.

Taking into account these changes would maximize the potential of the model to estimate the social performance of images on social media. Hopefully, in the future, this project will be improved along these factors experimentally.

Introduction

Being a content creator in this ever-changing world of trends is starting to become a game of chance rather than a game of strategy. While one Instagram post might succeed in giving a satisfying number of likes, other posts might fail to become popular for seemingly no reason. To the untrained eye, this might just be a consequence of the Instagram algorithm limiting exposure or reach randomly, but to the eyes of a ML Engineer, there are some patterns that appear in the behavior of followers interacting with the content in question. By mathematically modeling a function that predicts in advance how many likes an image would get, content creators would be able to make more informed choices on what and what not to upload to their profile. That way, creators can maximize the enjoyment they provide to their followers as well.

This project, as part of the Machine Learning Engineering Class (Spring 2024) at Rensselaer Polytechnic Institute, will instigate a supervised regression ML model to predict the popularity (in the amount of "likes," a type of feature that is commonly used in instagram popularity analysis[1]) that a proposed image will be generated given a precedent of images (validated through feature vector extraction) and their likes as a target. This model is designed so that, given a private account, an accurate interpolation between likes and images can be achieved given the dataset from that account.

There are plenty of Machine Learning options to predict the popularity of images over time on social media: for example, a team in the University of Bridgeport conducted an analysis of over 69,000 instagram images using a Gaussian Naive Bayes Classifier in 2015[2]. However, this analysis is too broad as it requires a larger dataset than the one that we have at hand. A more recent approach, attempted by a group of Princeton Students, used a Convolutional Neural Network Regressor[3] to predict the popularity of instagram images on the public sphere (the same goal as the previous study). With over 3,411 images, the Regressor was able to estimate trends on a dataset magnitudes smaller than the Gaussian Naive Bayes Classifier. While both of these approaches demonstrate the effectiveness of machine learning in predicting the social performance of instagram posts, the CNN ended up being a less costly option to reverse engineer the instagram algorithm based upon the general audience of instagram. This project, on the contrary to the state of the art, will narrow the scope significantly using the same Convolutional Regression technique to predict the success of instagram posts on a much smaller dataset (and a much smaller audience).

Because of the literature review, a regression style[4] approach would result in the most optimal results. What would a regressive approach entail, especially since the scope of the project would be greatly narrowed? Regression[5] is the act of scaling one feature against the correlation of another feature for the sake of prediction. There are multiple types of Regression: there is linear regression, or logistical regression. Logistic regression is for classification problems, while linear regression is for problems that are more intuitive and flexible than categories. Since the standing dataset is not being sorted into any explicit categories, a linear regressive approach is the most optimal to regress certain image features against the correlation of number of likes. Knowing the design of the problem and the cost effectiveness of various machine learning approaches, a CNN regressor will be used to solve the problem at hand.

[1] Reference [5]
[2] Reference [1]
[3] Reference [6]
[4] Reference [4]
[5] Reference [3]

Problem Definition

The problem that I will be investigating involves putting a pin on the behavior of my personal followers on instagram: Every post that I upload involves my face in some way, so my assumption is that all of my posts would receive a consistent amount of likes and comments by the same individuals. However, despite all 826 of my followers having access to my account at all times, some pictures naturally receive more likes and camaraderie than others. I am not an instagram influencer: however, I can see why others (especially those financially depending on their instagram performance) would want to track these patterns in accordance to their audiences.

What makes this problem interesting is that, despite the inherent demand, there is not really an open source (or github project) that tackles this problem on a personal, individual dataset. Most "instagram likes predictor" projects online mention web scrapers to build massive swathes of instagram engagement datasets, but those datasets defeat the purpose of predicting the behavior of followers within small, closed off, private accounts like my own. In other words, these projects make attempts to reverse engineer the instagram algorithm to predict what type of content encourages maximum content-consumer interactions on a constantly changing public "for you page". When it comes to smaller, more intimate, private social media profiles, the user is not competing for likes or comments or exposure with others, so their generated "likes" data more reflects the dedicated interests of a loyal control group instead of an ever changing flow of individuals who happen to stumble upon said content.

The purpose of this project is to figure out a way to interpolate the correlation between my likes and my image-data to best predict how popular a post would be given my dedicated friends/family as a control group. Machine learning, as a field, is an excellent way to do this interpolation as it uses mathematics to enumerate data and draw patterns between common trends. Predictions based upon the patterns found can, hence, be made from the machine learning model created. My project will focus directly on the behaviors of my personal followers and their respective likes/dislikes, and hence, data from my account will be used in training and fitting my Machine Learning model.

Methods and Procedure

Machine Learning is a useful approach in mathematically consolidating the behaviors of my followers and putting trends to application. Assuming all 185 of my images as inputs and the number of likes/comments as features, a Linear Regression Convolution Neural Network approach will be taken to predict post popularity (in the form of numerical integer "likes") against a feature vector generated from a dataset of processed images.

The convolutional neural network algorithm that can analyze images and predict numbers from them are pre-existing to an extent. However, the challenge in this project is to collect the training data and testing data. I will consolidate my instagram images into 185 feature vectors containing the pixel strength of the image, the RGB (red, green, blue) color value of each pixel, and the respective likes of each image For the predictions, I will compare the results from my machine learning model against a post that the model has never seen before in it's training or testing, then compare the performance of this post to the prediction made by my model. I will then calculate the mean square error and the losses of the respective likes, and a training/validation graph will be made to track regression metrics. The model will be tuned according to the goal of minimizing the mean square error between the predicted number of likes and the actual number of likes.

Tensorflow Keras, an open source Deep Learning Framework, will be used to create the Convolutional Neural Network. Convolutional Neural Networks (CNN), specifically, a regression type, are commonly used in computer vision applications and are an excellent way to make predictions based on image data alone. Regression mathematical models are also known to scale testing data against training data and then estimate where the testing data occurs on the training data trends. When a feature location is estimated, so would their associated estimated target value. Specifically using CNNs for regression type imaging is a powerful and effective way to confront the problem at hand! Because CNN architecture is designed for computer vision applications, they could more easily process images and regress them. Other regressors, such as K-Means, would suffer from the curse of dimensionality when applied to the same challenge, making CNN the most effective type of machine learning model.

The specific CNN Architecture we will use is (very loosely) inspired by Google's Inception-v3 AI[6]. This AI is the cutting edge of image processors, and is commonly used in image regression analysis (for example, a common task that Inceptionv3 is famed for is predicting housing prices based upon a few pictures of a home). Of course, implementing the entire AI model architecture would be costly and excessive for the task at hand. Nevertheless, the inspiration of layering a unique (self-designed) set of "Convolutional Layers,"Max pooling" layers, "Dense", "Flatten", and "Drop out layers" comes from InceptionV3's complex (but elegant) architecture that's commonly found in other advanced CNNs. The use of these specific layers will be used in my own architecture in my own simplified model, though I did not copy the order (or the included hyperparameters) at which these layers appear. Take note of the following architecture:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)              Output Shape             Param #
=================================================================
==
 conv2d (Conv2D)           (None, 62, 62, 32)       896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0


 conv2d_1 (Conv2D)         (None, 29, 29, 64)       18496

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0


 flatten (Flatten)         (None, 12544)            0

 dense (Dense)             (None, 128)              1605760

 dropout (Dropout)         (None, 128)              0

 dense_1 (Dense)           (None, 1)                129
```

[6]Reference [2]

```
==================================================================
==
Total params: 1,625,281
Trainable params: 1,625,281
Non-trainable params: 0
__________________________________________________________________
```

This model will be compiled using ADAM (learning rate = 0.01) with a batch number of 32 and an epoch number of 50 to optimize the model and confront the loss approach of minimizing Mean Squared Error. Mean Squared Error is a way to measure the error between the truth value of a target versus the predicted result generated from the Model. As the problem is a regression-type, the goal of the model is to tune itself so that the MSE generated between its predictions and the validation data would be minimized over time.

Dataset and Visualization

In the Github[7] of the "Social Performance Estimator" project, the "Images (jpg)" folder under the "Data" folder contains 185 .jpg images that will be extracted into feature vectors: if one would like to use their own personal images, they are free to replace the images in the file with their own as long as they follow the numerical naming conjunction of "1.jpg, 2.jpg, 3.jpg ... "n".jpg" where "n" is the number of total images that will be used. The purpose of this section of the code is to extract the feature vectors from the .jpg images in "Images (jpg)" by reducing the original varying pixel dimensions of each image down to a 256 unit length 1-Dimensional Array that corresponds to a 64 x 64 pixel image. It is also noted that if one were to replace the images with their own data, they should also replace the data written in the "y_likes_data.txt" file with the number of likes corresponding to each image with the following conjunction: 1: [1.jpg's likes] 2: [2.jpg's likes] ... "n": ["n".jpg's likes] where "n" is the number of total images that will be used. This way, the model could effectively process the images in these files without naming convention errors.

In my specific dataset, all of the images in this dataset are pictures of me in one way or another, may it be selfies or close-ups. By limiting the picture dataset to strictly a single theme, the model will be more able to draw correlations between different poses/looks/themes and the associated likes received. This method utilizes image segmentation to highlight important features in an image (for example, in the last image pictured in the example images below, my glasses are lit up in bright orange as an example). The color value and pixel count are important features that the model will take into account when analyzing each image.

A sample of the images is shown below: note how all of the images are reduced to a 64 by 64 pixel shape. Using Computer Vision, each video is extracted to a 265 length feature vector and then written to the "X_feature_vector_data.txt" file. This is done so that the model has the capabilities of pulling from the datafile without loading all 185 files and processing them everytime that the model is run. Because of the many design iterations that the model underwent (for example, the epochs were modified from 30 to 50 and the layers were changed to maximize learning during an iteration), this is done to minimize the cost of fine-tuning the model.

[7] See **Data and Code**

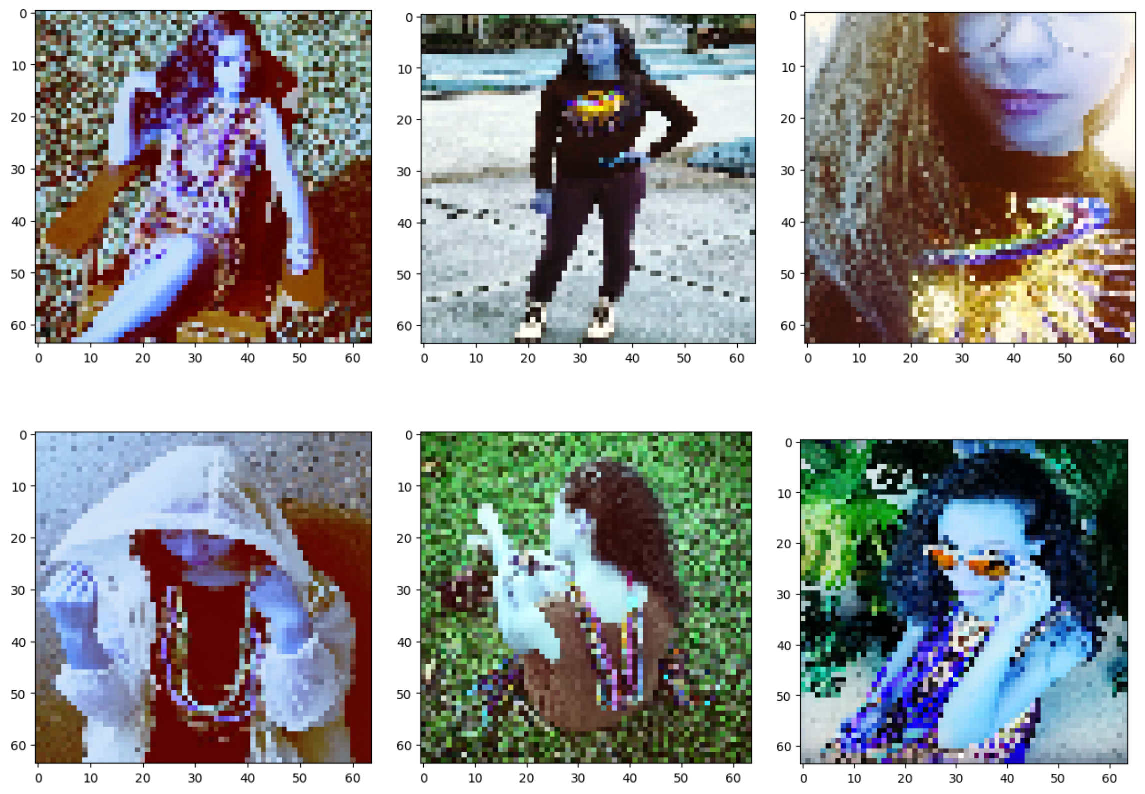
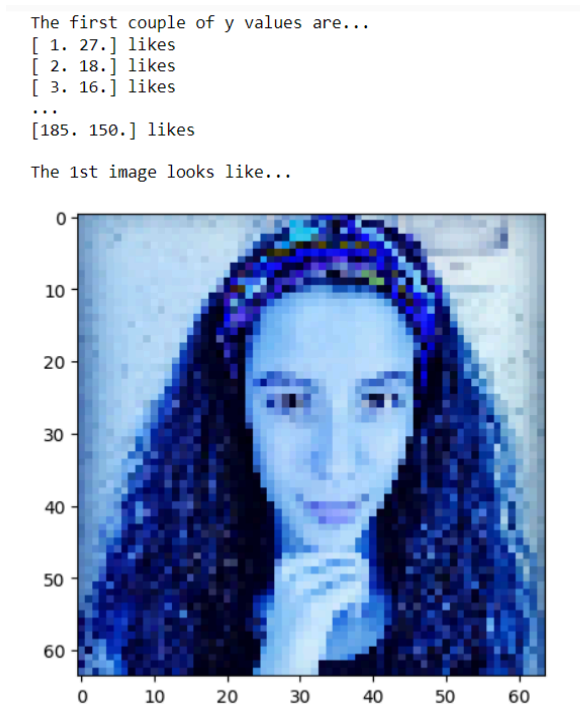


Figure 1) A sample of the training/testing images



```
The first couple of y values are...
[ 1. 27.] likes
[ 2. 18.] likes
[ 3. 16.] likes
...
[185. 150.] likes

The 1st image looks like...
```

Figure 2) Visualizing the "Y" dataset

```
The feature vector array of this image is...
[114 152 194 ...  62 113 177]

Lets visualize how balanced our dataset is...

While the dataset could be more balanced, it encasulates an 'ok' variety...
```

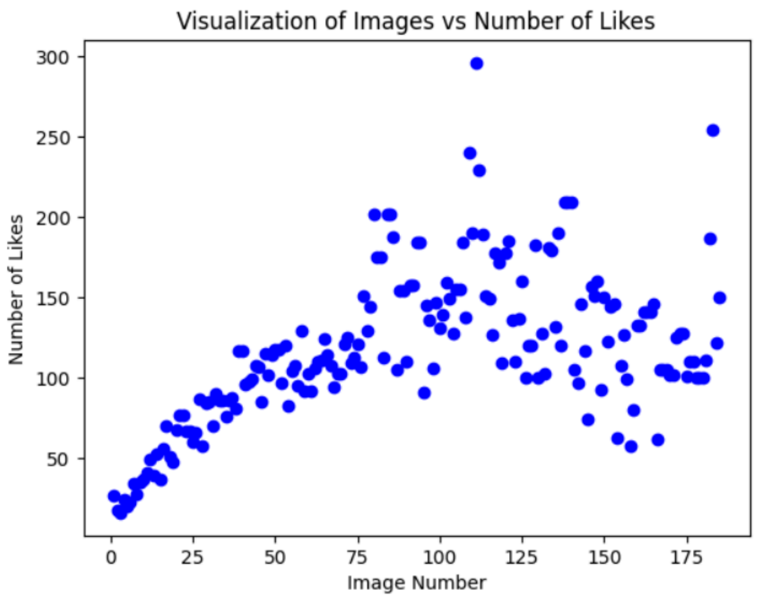


Figure 3) Visualization of the Dataset Scope

It is important to note that a machine learning model is only as good as the data that it processes, bringing into question whether the dataset used for this problem is fair, balanced, and encapsulates the full scope of possibilities. In Figure 3, a visualization of the images versus the number of Likes can be seen. From approximately 10 likes to 200 likes, a healthy number of images can be assumed spanning the entire range. Above 200 likes, however, the density of images became sparse. Because of this gap in the dataset, it is to be assumed that the model will have a slight bias to rate images between 10 and 200 likes as opposed to ranges above 200. This showcases a slightly unbalanced dataset which could be corrected if more images above 200 likes would be included.

Results and Discussion

The CNN regression architecture will be trained upon the data wrangled from the "X_feature_vector_data.txt" and "y_likes_data.txt" files. First, data stored in the text files will be imported as X and y variables. For this model, we are using a 80/20 validation split to train and test the data. This means that approximately 80% of the data will be used for training purposes, and 20% of the data will be used for validation purposes. Each image data is segmented into two feature vectors: the pixel strength, and the RGB color value associated with each pixel. Given these two features, the associated likes will be predicted from this abstract image data.

Validation of a model is important to showcase its effectiveness. For certain types of models, certain types of validation tests are needed. In categorical type problems, common validation practices include generating a confusion matrix, a precision-recall (PR) curve, or a Receiving Operating Characteristic (ROC) curve. Unfortunately, none of these examples would work for this specific CNN, as it's a regressor, not a categorical implementation. The best way to demonstrate this model's capabilities is to plot the Mean Squared Error over epochs of both training and testing validation data. Plotting a loss function over the training and validation trials is the preferred verification method of choice for tracking regressive learning. For this specific instant, minimizing Mean Squared Error would increase the accuracy of the model. As demonstrated below in the figure, as epochs advance, the model (intuitively) gets better at predicting against its own training data since the Mean Squared Error plummets. Similarly, the validation data's loss is higher than the training loss by an approximate factor of two, which is to be expected considering that the validation data is data that the model has never seen before. This is a very good sign as it demonstrates efficient learning.

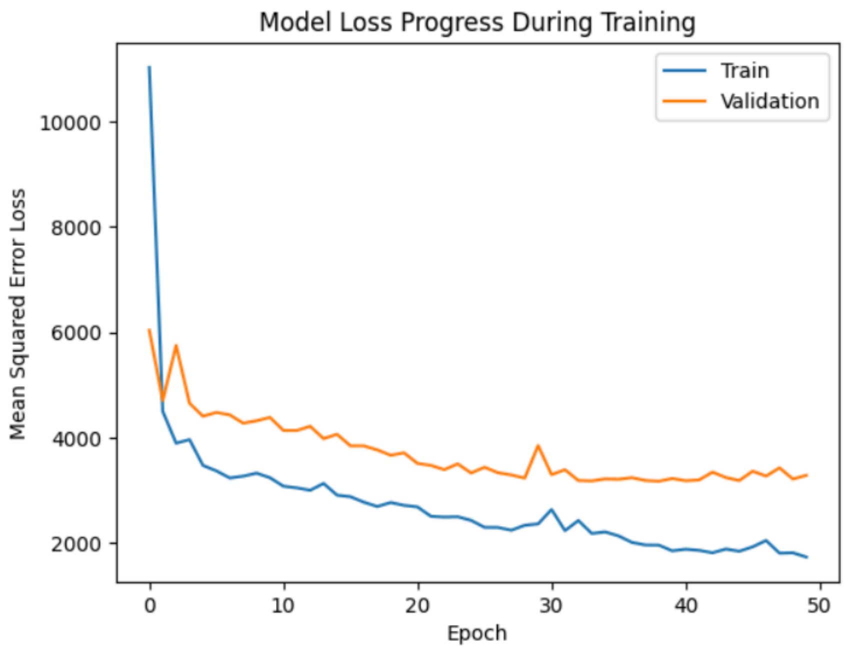
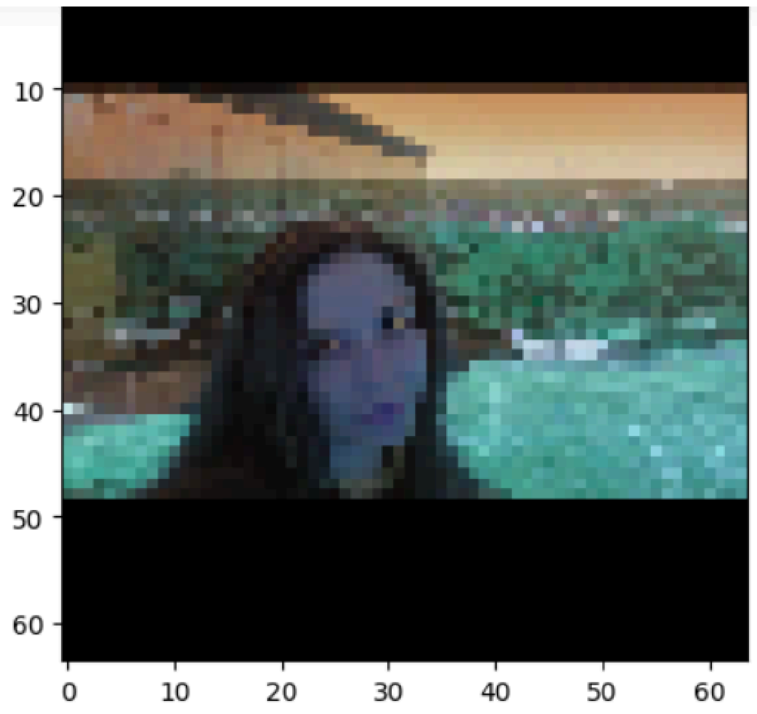


Figure 4) Model Loss during Training and Validation

What does this look like in a practical application? Figure 5 showcases a demonstration on a photograph that is not contained in the standing dataset, and hence, not been seen before by the model during testing or validation. This photograph is akin to the same theme (selfies or pictures of myself in a variety of environments) that occurs in my stand-alone dataset. This picture was dimensionalized to a 256 unit array (that correlates to the 64 by 64 pixels shown in the demonstration), then passed through the CNN architecture to result in a prediction of 103 likes. The original like count of this photograph is 96, which is an approximate 8% difference in error. While there could always be improvements in accuracy, this percentage difference demonstrates the effectiveness of the model fit. Going forward, I would personally use this model in predicting my future likes as long as the number of my followers (826) do not vary, as the model is predicted upon the likes generated from a control group of 826 followers. If I were to gain more followers, the model should be retrained every so often to ensure maximum accuracy in aligning towards what the audience truly enjoys.



```
A Selfie of me infront of Rensselaer Polytechnic Institute's EMPAC building...
This photo has 96 likes on my private instagram account as of April 16th, 2024.

1/1 [==============================] - 0s 93ms/step

The model's prediction of this is... [[103.68022]]
The percentage difference between the two numbers is... 8.00023078918457 %
```

Figure 5) Demonstration on an Image not contained in the Dataset

There are plenty of ways to improve the model. To start, focusing on improving the quality of the data could increase the performance of this model by giving the model more information to learn and tune from. Increasing the dataset count by including more images would give the model more to learn from, but since my current instagram picture collection is small, it will take some creativity to increase the data generated from it. One method is to feed the model a replica of the same exact pictures, but turned 90 degrees to the right. While to us it is the same image visually, but to the model, it is a completely new image to take into account with a different set of pixel strengths relative to the X and Y axis. Another method is to increase the number of features that's associated with each image, potentially through more advanced feature vector extraction. For example, measuring the symmetry, the type of image (selfie, pose, etc.), or whether an image has a certain background (nature backdrop, restaurant backdrop, city backdrop, animals in the backdrop, etc.) would allow the model to learn from much more data than the pixel strengths / colors alone. Humans are naturally pleased with symmetry, animals, and other scenic wonders, so it makes sense to take note when estimating the social performance of these images on social media.

Another way to improve model accuracy, aside from improving the dataset, is to fine-tune the model architecture through hyperparameter tuning. Experimenting with diverse CNN architectures (such as a different combination of layer types) and keeping note on what and what doesn't work would guide future iterations towards a more robust and sensible model. Modifying the epoch count, learning rate, and other hyperparameters are other ways to edit the model without touching the layer type. All of these modifications could change the way that the model fits itself towards the standing dataset, for the better (or for the worst, if done incorrectly)!

Taking into account these changes would maximize the potential of the model to estimate the social performance of images on social media. Hopefully, in the future, this project will be improved along these factors experimentally.

Conclusion

The ultimate goal of the Social Performance Estimator Project is to create a reliable mathematical model (using Machine Learning) that would guide my decision-making in posting on my private instagram profile. To do so, a Regressive Convolutional Neural Network was created using pre-existing literature as an inspiration, and my own personal dataset of images/likes from my instagram was then passed through to properly train, validate, and tune the model. In the end, the model that I created with Convolutional Neural Networks was able to do so with an accuracy error of 8%, which is impressive considering the limited dataset. Going forward, expanding the dataset by including more images and more features within each image would allow the model to learn more effectively and create better predictions in the future. This project could be used, not only by myself, but by other instagram content creators who want to estimate the popularity of an image before posting.

Data and Code

This project is published in Github under the handle "Social Performance-Estimator".
https://github.com/mythicane/Social-Performance-Estimator

8.References

1) Almgren, Khaled & Lee, Jeongkyu & Kim, Minkyu. (2016). Prediction of image popularity over time on social media networks. 1-6. 10.1109/CT-IETA.2016.7868253. https://www.researchgate.net/publication/314194570_Prediction_of_image_popularity_over_time_on_social_media_networks

2) Christian Szegedy and Vincent Vanhoucke and Sergey Ioffe and Jonathon Shlens and Zbigniew Wojna (2015) Rethinking the Inception Architecture for Computer Vision. https://arxiv.org/abs/1512.00567

3) Haidara Saleh and Jamil Antone Layous. (2022) Machine Learning - Regression. https://www.researchgate.net/profile/Jamil-Layous/publication/357992043_Machine_Learning_-Regression/links/61ea67b8dafcdb25fd3e2239/Machine-Learning-Regression.pdf

4) Shen Rong and Zhang Bao-wen. (2018) The Research of Regression Model in Machine Learning Field. MATEC Web of Conferences 176, 01033. https://www.matec-conferences.org/articles/matecconf/pdf/2018/35/matecconf_ifid2018_01033.pdf

5) Kristo Radion Purba and David Asirvatham and Raja Kumar Murugesan.Instagram post popularity trend analysis and prediction using hashtag, image assessment, and user history features.(2020) Int. Arab J. Inf. Technol. Vol.18, pg.85-9. https://api.semanticscholar.org/CorpusID:231858591

6) Qian et al. Instagram Popularity Prediction via Neural Networks and Regression Analysis. https://cjqian.github.io/docs/instagram_paper.pdf