DSA4212 Optimisation for Large-Scale Data-Driven Inference

Academic Year 2021/22 Semester 2

Group 3 Assignment 1 Report

Gender Classification of Images with Logistic Regression and Various Optimization Algorithms

Group members:

Name	Matric No.
Li Zhuoran	A0194510L
Pan Yuting	A0201267N
Zhang Simian	A0204709H
Zhong Zhaoping	A0194519U

Introduction

Image recognition is one of the hottest areas in machine learning. There are numerous image recognition algorithms, many of which can achieve satisfactory recognition accuracy on specific image datasets. Different from the mainstream image recognition based on deep learning, our main objective is to use variants of logistic regression for image recognition, and explore the impact of different optimization algorithms on model performance.

Our group built a logistic regression model that classifies the gender of a person in an image. Our group experimented with logistic regression models based on the stochastic gradient descent (SGD) algorithm and its variants, as well as a second-order algorithm Broyden–Fletcher–Goldfarb–Shanno (BFGS). The dataset we use consists of 20,000 coloured photos of faces, each with a resolution of 178x218 pixels. The logistic regression model with basic SGD serves as the benchmark, which is compared with the performance of other SGD variants and BFGS. Our best model achieved a test accuracy of 93% and a test AUC-ROC score of 0.97. To further test the practicability of our model, we also attempted to test the generality of our model with a small number of training images. These further tests also produced promising accuracy and AUC-ROC score of 77% and 0.86 respectively, indicating a good performance and scalability of our model.

Data Processing

1. Pre-processing

Image pre-processing is the steps taken to format images before they are used by model training and inference. This includes, but is not limited to, resizing, orienting, and colour corrections. In our project, we are going to see the effect of resizing, grayscale and contrast adjustment.

1.1 Resize

The resize in our project only refers to downsampling (e.g. The figure below displays a downsampling from a 128*128 pixel image to a 64*64 pixel image), since upsampling will introduce extra noise to the image data. A widely used technique for image downsampling is (max/average) pooling. A pooling operation is achieved by traversing a non-overlapping patch on the image, which calculates the value inside it. The max pooling calculates the maximum value for each patch of the feature map, while the average pooling calculates the average value for each patch on the feature map. In our project, average pooling is used as the downsample method.

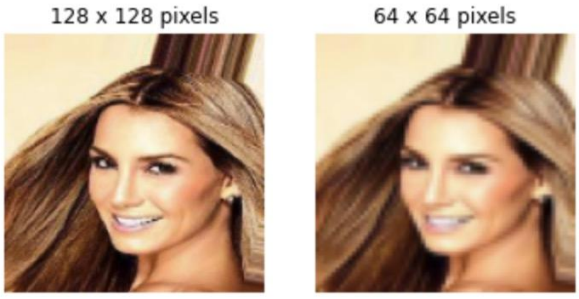

Figure 1

There are two major reasons for trying downsampling resize in our project. The first reason is to avoid overfitting. An image only with major information will increase the flexibility of our model. Secondly, running an optimization algorithm on a high resolution image costs a lot of computational time. Thus, a lower resolution dataset will be prefered if it could yield the same predictive ability as the higher resolution dataset.

1.2 Grey-scale

Converting the image to grey-scale is a common image pre-processing technique used in computer vision. An RGB image contains three channels, while a grey-scale image only has one. Thus, a grayscale image can reduce the complexity of the model at the cost of losing information in the image. In this project, our group tried to train with both RGB and grey-scale images and compared their performance.

1.3 Contrast Adjustment

Contrast is defined as the difference in brightness between light and dark areas of an image. Increasing the contrast will make the light areas become more distinguishable. For a portrait image, the person usually is in lighter areas and the background is in the darker areas. In this project, our group increased the contrast by multiplying a constant to each pixel of the image, which helps separate the person from the background.

2. Feature Extraction

By exploring the given attributes, we found that there is a strong correlation between gender and some facial features, such as moustache, receding hairline, etc. A potential solution is to extract facial features and fit a logistic regression model based on the features, which are strongly correlated with the gender.

2.1 Feature Detection

Our first attempt was to use Haar Cascade and OpenCV to detect and extract the facial features for each image. Haar Cascade is an algorithm that uses edge and line detection features. The algorithm is pre-trained by giving a lot of labelled images and records rectangular filters of different sizes which have dark areas and light areas. (shown in the figure below)

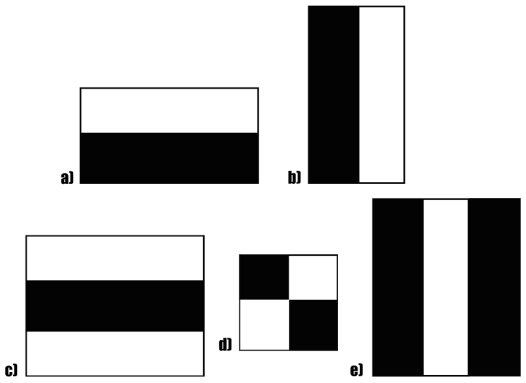

Figure 2

To pose the filter onto the image, we multiply each pixel on the image by the respective number in the filter. We can detect an edge by checking if there is a sudden change in the intensity of the pixels. The filters continuously traverse from the top left of the image to the bottom right to search for the edges of the particular feature. (Viola & Jones, 2001)

With the help of OpenCV, we successfully detected 89.4% of the faces among all images. Most of the remaining undetectable images are side faces. Knowing that all the images are aligned in the centre, our group took the average of all vertices from detectable images, and cropped the same areas for each image. Our group fitted the extracted features (e.g. face without background, nose, mouth, eyes) with logistic regression separately.

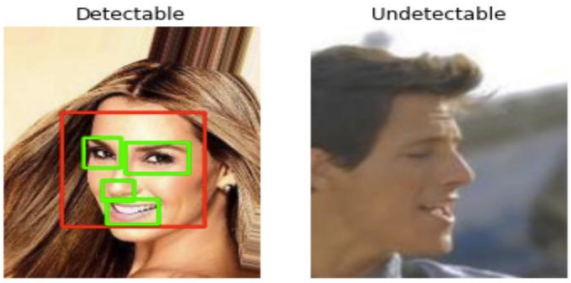

Figure 3

2.2 Ensemble Model

Our group tried to ensemble the logistic regression models trained by different facial features since each extracted feature alone might not be indicative enough for all images. Note that the detected face area (red rectangle) was not be used for the ensemble model because of information overlapping with the smaller facial features (green rectangles).

Our group tried two approaches for ensemble models. The most straightforward approach is to make predictions by majority vote. Assume there are k extracted features in total. we will predict the image to be a male if more than k/2 features predict the image to be a male. (Here we will make k as an odd number to avoid ties.)

A less trivial method is to fit another logistic regression to the probabilities given by each model trained with a single extracted feature. After fitting each of k features by logistic regression, we will have $p_1, p_2, \dots, p_k$ for each image, representing the probability given by each extracted feature. Our group attempted to fit another logistic regression to these probability features of all images, which gives us the ensemble model prediction.

Methods

1. Basic Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an iterative method for optimizing a differentiable objective function. It is a stochastic approximation of gradient descent optimization as it replaces the actual gradient by an estimate from a minibatch of data, which satisfies $E[\widehat{\nabla F(x)}] = \nabla F(x)$. The algorithm for updating the parameter vector x is

$x_{n+1} = x_n - \eta_n * \widehat{\nabla F(x)}$. This helps to achieve faster computations at the cost of slower convergence per epoch as compared to Gradient Descent (GD).

The learning rate η is fixed in the Basic SGD algorithm, which does not provide the optimal training performance. With a high learning rate, the parameter vector bounces around chaotically and is unable to settle down; with a low learning rate, the convergence is slow and computations become wasteful. Therefore, a few methods of choosing the learning rate are discussed below and employed in our experiment.

2. SGD with Step Decay

SGD with Step Decay reduces the learning rate η by some factor every few epochs. In our experiment, the learning rate was reduced by half every 5 epochs. Step decay is proven to provide performance that is competitive with other more complex techniques such as the polynomial decay scheme (Ge et al., 2019).

3. SGD with Heuristic Decay

In Heuristic Decay, a small proportion of the training data is reserved as the validation set and not involved in calculating the gradient and updating the parameter. The validation error is monitored during each epoch of the training process. When the validation error stops improving, the learning rate is reduced by half. This technique is simple to implement and popular in open-source packages such as Keras (Brownlee, 2019).

4. SGD with Scheduled Decay

SGD with Scheduled Decay adopts a decay schedule of the learning rate as $\eta_t = \frac{\eta_0}{1+\lambda \times t}$, where η_0 is the initial learning rate, λ is the decay constant with default value 0.5, and t is the number of the current epoch. The learning rate is scheduled with the epochs number to decay in a reciprocal way.

5. Method of Momentum (MOM)

SGD with momentum has been widely applied in many machine learning tasks and helps to accelerate the gradient vectors, resulting in faster convergence. It has been shown that SGD with Momentum converges as fast as SGD for smooth objectives under both strongly convex and nonconvex settings. (Liu et al., 2020) Specifically, momentum is referring to a vector in the direction of the very last update of the learning rate. The SGD with momentum used can be represented as $x_{n+1} = x_n - \eta\widehat{\nabla F(x_n)} + \beta(x_n - x_{n-1})$, where $\beta(x_n - x_{n-1})$ is the momentum term which can be decompose into the momentum parameter β with a typical value of 0.9 and the momentum $(x_n - x_{n-1})$.

6. Broyden–Fletcher–Goldfarb–Shanno (BFGS)

BFGS as a Quasi-Newton's method does not require matrix inversion, which helps to make Newton's method applicable for many large-scale datasets. The update of its curvature matrix B_{k+1}^{-1} requires only $O(n^2)$, whereas Newton's method needs $O(n^3)$ to compute the inverse of the Hessian matrix. Instead of using the Scipy package of L-BFGS directly, our group built a BFGS algorithm from scratch to show our deep understanding and effort on this algorithm. The pseudo-code is shown as follows.

From an initial guess x_0 and an approximate of the inverse of the Hessian matrix B_0^{-1} (we use the identity matrix here), the following steps are repeated as x_k converges to the solution:

1. Obtain a direction p_k by solving $p_k = -B_k^{-1} \nabla f(x_k)$.
2. Perform a one-dimensional line search to find an acceptable step size α_k satisfying Wolfe Conditions.
3. Set $s_k = \alpha_k p_k$ and update $x_{k+1} = x_k + s_k$.
4. Set $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
5. Update $B_{k+1}^{-1} = B_k^{-1} + \frac{(s_k^T y_k + y_k^T B_k^{-1} y_k)(s_k s_k^T)}{(s_k^T y_k)^2} - \frac{B_k^{-1} y_k s_k^T + s_k y_k^T B_k^{-1}}{s_k^T y_k}$.

Moreover, our self-implemented BFGS achieved a decent test accuracy and a good AUC score, which is shown in the Results part later.

Results

Our group adopted a linear workflow to find the best data preprocessing technique and best training algorithms.

In particular, we first investigated the impact of the size of the training set on test accuracy, followed by resolution of the original pictures. The effect of grey-scale and contrast/brightness was subsequently tested, and specific regions of the face were cropped to be used to test the model performance. All the data processing techniques above were tested with basic SGD to ensure fairness in technique selection.

After the best combination of processing techniques was fixed, different optimization algorithms were used to evaluate the test accuracy. Finally, our group used the best processing technique and optimization algorithm on only the first 200 images to check its performance on test data.

1. Best Data Processing Techniques

1.1 Sample Size of the Training Set

Our group explored different sample sizes of the training set from small to large in order to find the impact of the size of the training set on test accuracy. Specifically, we used training sample sizes of 1000, 3000, 5000, 7000, 9000, 11000, 13000 and the full training set of 15000. The result is shown in the figure below.

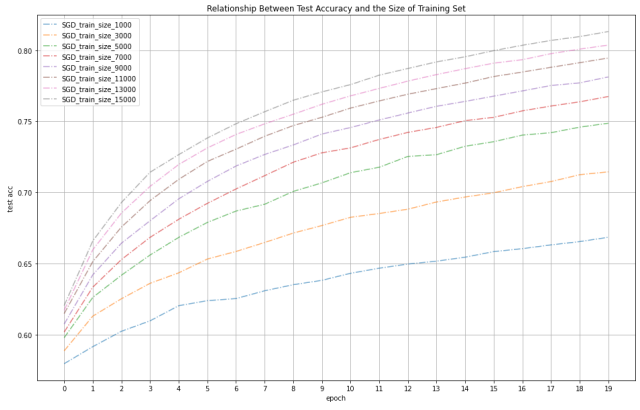

Figure 4

We observed that the test accuracy increases as the size of the training set grows up. However, the improvement of test accuracy was not significant when the size of the training set went beyond ⅗ of the full training set, i.e. 9000 out of 15000. Additionally, when our group set the size of the training set to be 11000, the test accuracy was very close to that of using the full training set. While the result shows that using only ⅔ of the full training set would achieve a decent test accuracy, our group decided to use the full training set since it provides the best performance and does not put unreasonable computational constraints on our end.

1.2 Resolution

Our group trained the model by the same dataset with resolutions of 14 * 14, 28 * 28, 56 * 56 and 112 * 112. The result is shown in the figure below. The performance of the logistic regression decreased as the resolution was downsampled. Although a lower resolution dataset saves computational time, the large decrease in the predictive ability is not acceptable. Hence, our group decided to use full size resolution as our training dataset.

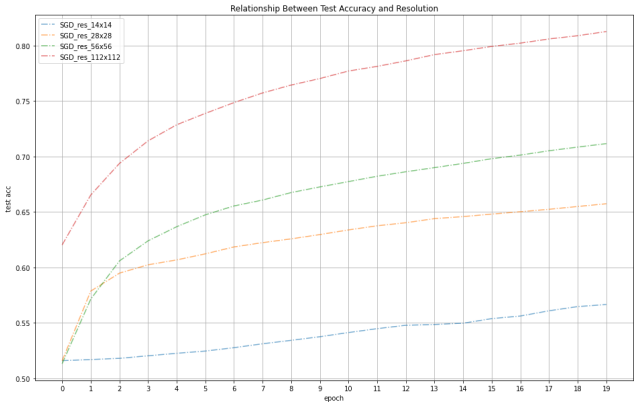

Figure 5

1.3 Colour Adjustment

Our group compared the effect of different colour adjustment combinations (i.e. RGB, Grey-scale, RGB with contrast increment, Grey-scale with contrast increment).

The result below shows that RGB outputs a better performance than the Grey-scale and the training dataset with contrast increment gives better performance than the one without contrast increment. Hence, our group proceeded to use RGB with contrast increase.

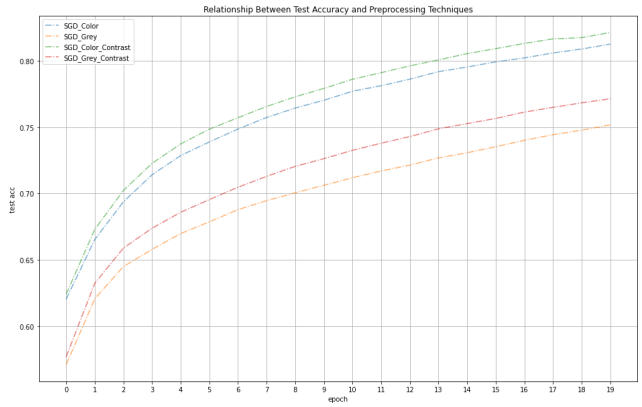

Figure 6

1.4 Feature Extraction

Our group extracted the face, nose, mouth, and eyes area for each image and fitted them with logistic regression separately. As shown in the figure below, the model fitted with faces produced the best accuracy at 85.1%, which outperformed the model trained by the full images. This result was expected because extracting out the face area removes background noises in the images.

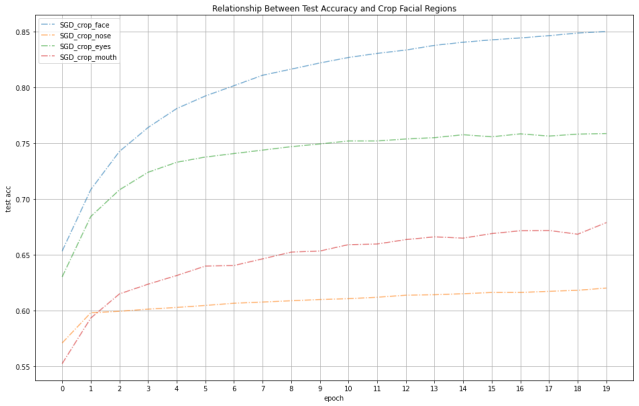

Figure 7

Our group used the technique discussed earlier to ensemble the other three logistic regression models (Nose, mouth and eyes). The "Majority Vote" yielded a test accuracy at 72.6%, while "ensemble by logistic regression" only yielded a test accuracy at 65%. The possible explanation for this unexpected bad result would be that the feature extraction through averaging cropping does not work well for small areas such as the nose and eyes.

2. Best Optimization Algorithm

In the aforementioned parts, the best image input settings are as follows: 15000 images of full size resolution, in RGB with contrast increment, in the form of extracted face parts. Under these optimal input settings, our group tested the SGD, its variants and the BFGS algorithm respectively, in order to find the best optimization algorithm.

Our group adopted the basic SGD model as the baseline model and it produced a considerably high test accuracy of 85% after 20 epochs. From Figure 8, some SGD variants such as SGD with Step Decay and SGD with Heuristic Decay perform very close to the basic SGD model, with a test accuracy of 82% and 83% after 20 epochs respectively. Other variants such as SGD with 1/t Scheduled Decay and SGD with Momentum exhibited better performances with higher test accuracy reaching 91% and 89% respectively. The BFGS model outperformed the basic SGD and some of its variants, achieving a test accuracy of 88% after 20 epochs.

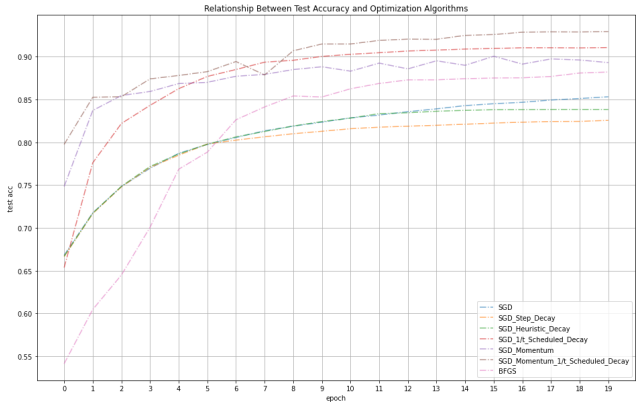

Figure 8

Our group discovered that SGD with 1/t Scheduled Decay and SGD with Momentum have a very promising performance and a much faster training time as compared to BFGS. To benefit from both scheduling and momentum, our group designed a new variant called SGD with Momentum and 1/t Scheduled Decay, which is a superposition of these two variants, using 1/t scheduled decay to change the learning rate and momentum to accelerate convergence simultaneously. It turned out that SGD with Momentum and 1/t Scheduled Decay can yield an even higher test accuracy at 92.5% after 20 epochs. Hence, we adopted the SGD with Momentum and 1/t Scheduled Decay as our best model.

The test accuracy and the test AUC-ROC score after 30 epochs of SGD with Momentum and 1/t Scheduled Decay are 93% and 0.97 respectively, which is very satisfying. The test accuracy history and the training loss history are shown in Figures 9&10.

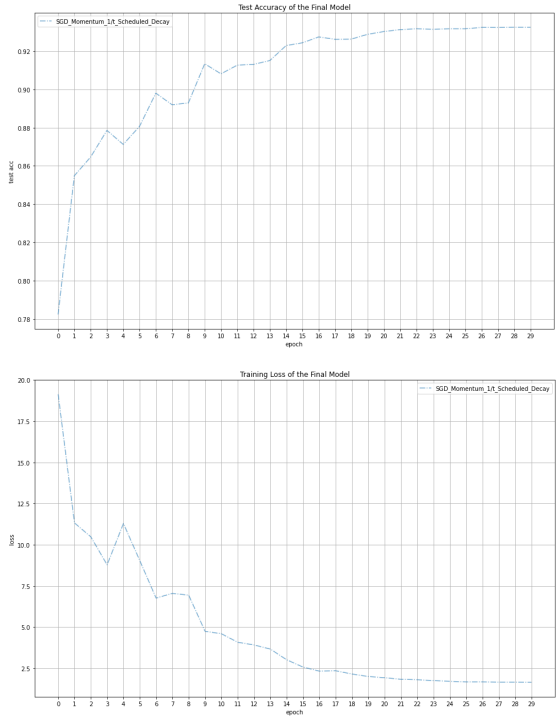

Figure 9&10

3. Performance on Small Data

Our group applied the best data processing techniques and best optimization algorithm selected above on only the first 200 images. Due to the small size of training data, our group attempted adding regularisation to the loss function to prevent over-fitting, namely LASSO and Ridge. While MOM with scheduled decay worked the best for 15000 images, the scheduled decay caused a worse performance in the small data setting (early convergence to 72%). Therefore, the final model for small data is MOM without any learning rate decay. The test accuracy was surprisingly promising as shown in Figure 11, and adding L1-regularisation (LASSO) provided the best performance on test data. The test accuracy and AUC-ROC score are 81% and 0.87 respectively.

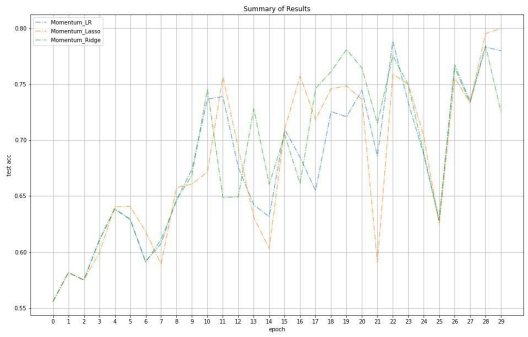

Figure 11

Reflections and Conclusion

One of the main challenges our group encountered in this assignment was the management of random access memory (RAM). The code was carefully reviewed and designed, especially the data loading functions, to ensure that the code does not cause the problem of overflowing RAM.

Our group experimented with how different image input settings would change the results, including the number of input images (the number of training sets), the resolution of the input images, the colour scheme, and whether to crop according to the facial features. These different settings did affect our results. In summary, the setting that yielded the best accuracy is 15000 images of full size resolution, in RGB with contrast increment, in the form of extracted face parts.

It can be concluded from our experiments that logistic regression can also be competent and produce satisfactory results when dealing with simple image classification tasks. Our group implemented the stochastic gradient descent method and its variants and BFGS algorithm from scratch. The most basic SGD model also yielded a relatively good accuracy, and variants such as SGD with 1/t Scheduled Decay and SGD with Momentum further improved the performance of the model. Our group finally combined the advantages of the two best-performing variants to design a new variant called SGD with Momentum and 1/t Scheduled Decay, which achieved the best performance. Therefore, we consider SGD with Momentum and 1/t Scheduled Decay as our final best model.

References

Brownlee, J. (2019, January 25). *Understand the Impact of Learning Rate on Neural Network Performance*. Machine Learning Mastery. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Ge, R., Kakade, S., Kidambi, R., & Netrapalli, P. (2019). The Step Decay Schedule: A Near Optimal, Geometrically Decaying Learning Rate Procedure For Least Squares. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. https://proceedings.neurips.cc/paper/2019/file/2f4059ce1227f021edc5d9c6f0f17dc1-Paper.pdf

Liu, Y., Gao, Y., & Yin, W. (2020). An Improved Analysis of Stochastic Gradient Descent with Momentum. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. https://proceedings.neurips.cc/paper/2020/file/d3f5d4de09ea19461dab00590df91e4f-Paper.pdf

Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *CVPR*.

Appendix

Github Repository:
https://github.com/simonCodeZzz/DSA4212_Assignment1/tree/main

Code Style & Management:
To make the code reproducible and friendly for team cooperation, we fix the code style at the beginning of the project. Most codes are embedded as functions that can be easily called, with the following categories: Common Utility Functions, Utility Functions for Logistic Regression, Utility Functions for Training Model and Backcalling, Utility Functions for Plotting Graphs. The functions are also being properly commented and documented to ensure every team member is on the same page. This allows us to experiment and make adjustments without much hassle.

The function reference manual can be found here:
https://docs.google.com/document/d/138Ml8ZWBoT6oFxWKEzxtZ3JEhaPtY9P32WX_4CRFmJo/edit?usp=sharing