Computer Vision Project

Revisiting Background Segmentation, Gaussian Mixture Models & Deep Learning

1st Oluwatimileyin Obagbuwa (2134111)
Computer Science
University of the Witwatersrand
2134111@students.wits.ac.za

2nd Suraksha Motilal (2108903)
Computer Science
University of the Witwatersrand
2108903@students.wits.ac.za

3rd Byron Kruger (2090048)
Computer Science
University of the Witwatersrand
2090048@students.wits.ac.za

4th Derrin Naidoo (2127039)
Computer Science
University of the Witwatersrand
2127039@students.wits.ac.za

***Abstract*—We present the implementation of Gaussian Mixture Models (GMMs) and U-Nets as viable solutions to Tino's puzzle-solving problem. Through various experiments and analysis of the results, we find that learning features proves to be more useful for generalisation and produces a more robust model. However, we observe that hand-crafted features still produce favourable results when there is known information over the domain. The analysis of both models provides important considerations when selecting a solution.**

I. Introduction

Tino's puzzle-solving problem presents a novel AI domain in which many fundamental Computer Vision techniques are used to present solutions. We present implementations of Gaussian Mixture Models (GMMs) and U-Nets as potential solutions to the problem. For both models, We follow a common pipeline of data processing, feature extraction, training and prediction (testing). Additionally, 6-fold cross-validation will be performed for each model.

We explore the implementation of GMMs using various feature sets, and U-Nets are implemented with varying data augmentation techniques. After evaluating the results from both models, a final comparative analysis will be presented in the form of a discussion.

II. Data Preparation

A. Discussion of the data

SKLearn's train_test_split function was used on the images and masks with a random state of 42 for consistent splitting results per run. The training images comprised of 70% of the data-set, with the testing and validation images consisting of an equal split of the remaining 30% of the data-set.

To increase speed and efficiency, the images were scaled to 25% of their original size.

III. Gaussian Mixture Model (GMM)

A. Implementation and findings

We implemented GMM using reusable functions and not typically with a class.

The GMM function returns the cluster weights, means, covariance matrices and the number of iterations run in total.

We investigated the effect of the feature set used on the accuracy of predictions. The features sets are: RGB, HSV, RGB+doG and RGB_(RGB+doG)_HSV (all feature sets combined to make a vector with 9 features).

After running GMM multiple times with each feature set, we generally found HSV and RGB_(RGB+doG)_HSV to have the best average accuracies. However HSV is smaller than RGB_(RGB+doG)_HSV and could be considered as the optimal feature set. This is likely due to the fact that unlike RGB, HSV separates the image intensity, from the color information thus making it robust to lighting changes.

When we were testing the accuracy of the feature set RGB+doG, we found that the greater the difference of gaussians(greater difference in blurriness between 2 images), the more accurate the prediction. This could be because a greater difference of gaussians would better highlight certain high-frequency components in the image.

Training times decreased significantly after the images were scaled down. Prior to the re-scaling, training times ranged from 14-30 minutes, whereas after the re-scaling, training took less than 10 minutes.

B. Results

Please note that the run times in the tables below are represented by python time units. They have not been converted to minutes or seconds.

TABLE I
GMM RESULTS BY ALTERING CLUSTER (K) SIZES

Features used	Run Time	K Value	Accuracy
Set1 [a]: no 6-Fold	5.349/ 4.42	2/ 3	92.96/ 92.55%
Set1 [a]: 6-Fold	45.4/ 101.199	2/ 3	93.66/ 92.586%
Set3 [c]: no 6-Fold	28.37/ 37.91	2/ 3	96.815/ 96.25%
Set3 [c]: 6-Fold	211.55/ 325.52	2/ 3	96.22/ 95.906 %
Set1 [a], Set2 [b], Set3 [c]: no 6-Fold	61.24/ 83.67	2/ 3	96.828%/ 95.786%
Set1 [a], Set2 [b], Set3 [c]: 6-Fold	384.55/ 608.51	2/ 3	96.31%/ 95.798%

[a] *Set1= [Red, Green, Blue],* [b] *Set2= [Red+Difference of Gaussians, Green+Difference of Gaussians, Blue+Difference of Gaussians],* [c] *Set3= [Hue, Saturation, Value]*

TABLE II
GMM RESULTS BY ALTERING GAUSSIAN DIFFERENCE

Features used	Time taken	Gaussian Difference	Accuracy
Set1 [a], Set2 [b]: no 6-Fold	9.475	10	93.12%
Set1 [a], Set2 [b]: 6-Fold	72.83	10	93.87%
Set1 [a], Set2 [b]: no 6-Fold	15.908	20	93.54%
Set1 [a], Set2 [b]: 6-Fold	74.43	20	94.01%
Set1 [a], Set2 [b]: no 6-Fold	17.601	30	93.82%
Set1 [a], Set2 [b]: 6-Fold	56.62	30	93.87%

[a] *Set1= [Red, Green, Blue],* [b] *Set2= [Red+Difference of Gaussians, Green+Difference of Gaussians, Blue+Difference of Gaussians]*

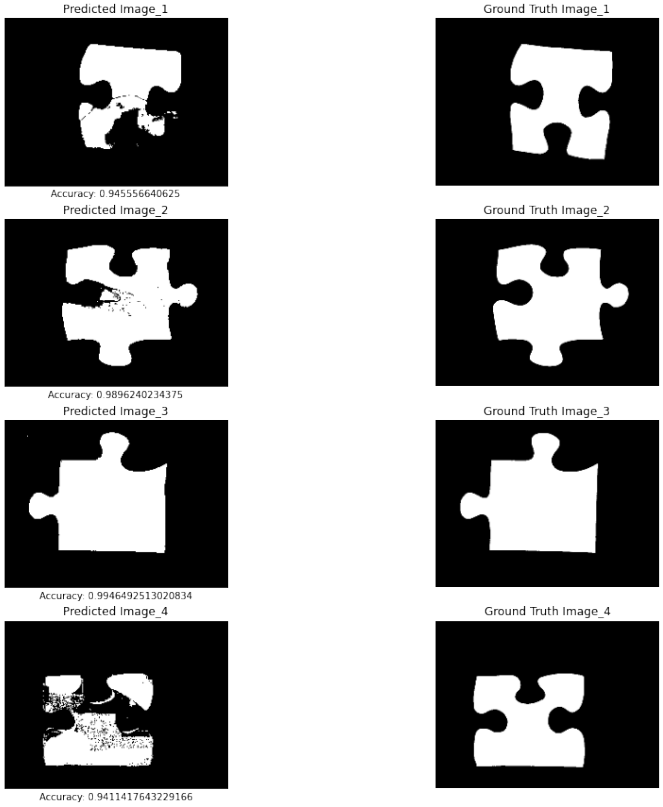

Fig. 1. Segmentation results of the Set1, Set2 and Set3 feature set combinations (no 6-fold validation)

C. Issues and Resolutions

In order to obtain the optimal number of clusters, a silhouette score [1] can be used to calculate the distances between each sample in the cluster and their points in the nearest cluster/ class. This was implemented, however, due to time constraints and the vast number distances to be checked per datapoint, the running of the algorithm took too long to be able to display results (1.5 hours for almost 2 cluster number checks, where k=2 returned a silhouette score of 0.3377221). Regardless, the silhouette score should be a valid identifier in determining this value as the higher the silhouette score is to 1, the more optimal that cluster number k is.

Taking time constraints into consideration, some of the models may not have ran until convergence but rather to the maximum number of iterations, which was 30. The GMM has the capability of convergence and might thus produce better results.

D. Conclusion

Gaussian Mixture Model has proven to be an effective solution to Tino's puzzle-solving problem. The ability of a GMM to segment high-dimensional data based on certain phenomena in the data makes it an option of choice for image segmentation in general. Additionally, the ability to use multiple gaussians for fitting enables GMMs to overcome problems of unimodality.

IV. U-NET

A. Implementation

We have used the Tensorflow Keras VGG16 model in our implementation. The implementation consists of an input layer, the pre-trained VGG16 Encoder, a bridge, a Decoder and an output. The encoder is made up of four skip-connection layers. They are 512×512, 256×256, 128×128 and 64×64 respectively. The bridge between the encoder and decoder consists of a 32×32 block containing pre-trained VGG16 features. Finally, the output of the decoder is passed through a 1×1 convolution layer using a Sigmoid activation function.

The model was trained with and without data augmentation. The data augmentation, at a high-level, consisted of position augmentation and colour augmentation. The position augmentation techniques included a random flip (horizontal and/or vertical), random rotation and random zoom, while the colour augmentation included random contrast and random brightness. The NUM_IMAGE_AUGMENTATIONS parameter was used to determine how many augmented images would be created from an image in the dataset. The parameter was given a value of 5 which resulted in an augmented training set of 204 images. Notably, because the keepOriginal remained had a value of True, the augmented

training set also included the original images without augmentations.

We used 200 epochs with batch sizes of 6 to train the model on the original dataset, both normally as well as using 6-fold cross-validation.

B. Results

TABLE III
U-NET-VGG16 KERAS TESTING RESULTS

Model Type	Train Time	Loss	Accuracy
No Aug*, no 6-Fold	2.5 Min	357.66	32.1%
No Aug*, 6-Fold	13 Min	345.38	94.8%
Position Aug*, no 6-Fold	12 Min	337.28	94.6%
Position Aug*, 6-Fold	32 Min	370.75	18.8%
Colour Aug*, no 6-Fold	12 Min	355.57	47.8%%
Colour Aug*, 6-Fold	32 Min	266.61	24.1%
Colour & Position Aug*, no 6-Fold	12 Min	349.39	59%
Colour & Position Aug*, 6-Fold	32 Min	507.61	14.1%

Aug = Augmentation*

TABLE IV
U-NET-VGG16 BEST-FIT TESTING RESULTS

Model Type	Train Time	Best-fit Loss	Best-fit Accuracy
No Aug*, no 6-Fold	2.5 Min	354.1	50.3%
Position Aug*, no 6-Fold	12 Min	344.2	78.2%
Colour Aug*, no 6-Fold	12 Min	347.49	67.8%
Colour & Position Aug*, no 6-Fold	12 Min	337.42	94.6%

Aug = Augmentation*

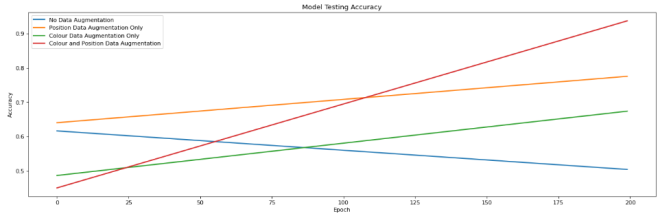

Fig. 2. Best-fit Testing Accuracy

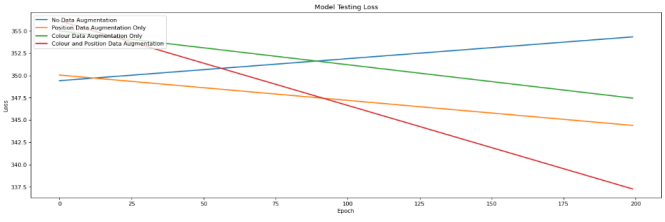

Fig. 3. Best-fit Testing Loss

C. Analysis

In order to effectively train the U-Net, one might consider training the model for far greater than 200 epochs which would yield a more robust model. However, due to resource and time constraints, 200 epochs seems to yield favourable results which is useful to perform analysis of the U-Net model.

The results obtained from the various experiments present expected trends. Upon first evaluation of the results, the accuracy scores reflected in table III are misleading. However, these accuracy scores are obtained using the Keras library. That means that the scores were calculated by calculating the metric for each batch and then returning the mean value across all batches. This is flawed because it is possible that multiple batches may yield low accuracy scores, however, the model may accurately predict a a notable amount of the data within batches well. Therefore, we've calculated a line of best fit for the accuracy and loss functions to produce trends with best-fit metrics, as seen in table IV. Furthermore, we observe expected trends from the various tested models as shown in figure 2 and figure 3 .

Six-fold cross-validation was used in every step of the experiments with keras metrics presented in table III. By implementing six-fold cross-validation, we avoid overfitting the model, which is useful given the small training set, while producing good keras metrics compared to no six-fold cross-validation.

As expected, training the U-Net without any data augmentation produces the lowest best-fit testing accuracy and the highest best-fit testing loss among all the experiments. This is due to the lack of generalisation in the model and the inability of the model to effectively learn using the smaller test data. As a result, the model will take longer to converge to an optima and generalisation will still be limited.

Every data augmentation technique produced better test results over no data augmentation. Notably, we observe that colour augmentation proves to be less effective than position augmentation. This makes intuitive sense since the colour augmentation techniques being applied, random contrast and random brightness, does not produce the same level of quality variation compared to position augmentation. Quality variation refers to variation that one would expect to appear in the real-world or testing data. Since all the images from the original dataset were taken at the same angle with the same lighting conditions, colour variation is not very likely in the testing data. It's more likely, and quite apparent, than puzzle pieces may be shaped differently across photos which infers that puzzle pieces will be positioned differently in our dataset. Therefore, applying position augmentation allows the model to learn the features that better correspond to puzzle pieces in different positions.

Combining both position and colour augmentation results in a larger training set with greater variation which allows the model to train longer, to hopefully reach a point closer to convergence, and to generalise well. We observe that this model with both augmentation techniques applied produces

the best best-fit testing accuracy along with the lowest best-fit loss score.

D. Issues and Resolutions

- Colour Augmentation produced unusable masks when random contrast factors and random brightness factors were at extreme values. These factors were therefore constrained to random values within certain acceptable ranges
- Position Augmentation produced meaningless training data when the random zoom factor was too large. Therefore, this factor was constrained to random values within 0 and 0.3 so images only ever zoomed to a maximum of 70% of the original image size.
- To solve the issue of compute resources being exhausted during training, allocated memory was freed at every point where possible.

E. Conclusion

The U-Net presents an effective solution to the puzzle-solving problem as the deep learning model is able to learn features from the training data rather than manually selecting features. The benefit of doing so allows the model to potentially extract more distinguishable features to discern between background and puzzle pixels. While the model is effectively able to solve the puzzle-solving problem, U-Nets require great compute power and training is very time consuming. One must consider the trade-offs of using U-Nets when choosing a solution for the puzzle-solving problem.

V. Comparative Analysis

A. Discussion

It is observed that the GMM produces higher accuracy than the U-Net overall. This is unexpected but may be accounted for as deep neural networks require effective training in order to reach an optima. The number of epochs for U-Net training may be too small for the U-Net to reach converge. While U-Nets do not generally require a large amount of training data, it's very likely that the distribution of data into training, testing and validation sets results in a training set that does not have good variation of the original dataset. Therefore, this results in a model which is biased and may attribute to the lower accuracy of the U-Net.

While the testing accuracy of the GMMs prove to be higher than the U-Nets, the U-Net produces a more robust model as features are learned rather than hand-crafted.

The differences in accuracy of these two models compared to the Probability Density Function thresholding method is quite substantial, as the PDF method returned an accuracy of approximately 85%. Despite some masks which are generated by the GMM having a relatively noisy appearance, many masks have near perfect segmentation, as seen in Figure 1. The same can be said for the K-Means clustering algorithm, which performs worse than the GMM due to its use of circular radii to classify points. The GMM is more flexible with regard to its classsifications, making use of ellipses when necessary to represent the spread of data.

References

[1] Dobilas, S. (2021, May 23). GMM: Gaussian mixture models — how to successfully use it to cluster ... Retrieved November 2, 2022, from https://towardsdatascience.com/gmm-gaussian-mixture-models-how-to-successfully-use-it-to-cluster-your-data-891dc8ac058f

APPENDIX

A. Group contributions

The Gaussian Mixture Model section was allocated to Oluwatimileyin Favour Obagbuwa and Suraksha Motilal. The U-Net section was allocated to Derrin Naidoo and Byron Kruger.