Semantic Segmentation of Tino Puzzle Using GMM and UNet - an Analysis

Nilesh Jain
School of Computational and Applied Mathematics
University of Witwatersrand
Johannesburg, SA
Email: 2615122@students.wits.ac.za

Abdel Njupoun
School of Computational and Applied Mathematics
University of Witwatersrand
Johannesburg, SA
Email: 2631227@students.wits.ac.za

Bathandwa
School of Computational and Applied Mathematics
University of Witwatersrand
Johannesburg, SA
Email: 1717212@students.wits.ac.za

***Abstract*—This report summarizes our attempt to semantically segment puzzle pieces using Baseline, Gaussian Mixture Models and U-net and the underlying problems encountered and found U-net works best for the semantic segmentation task with an accuracy of 89.9 while GMM produced an accuracy of 84.5 surpassing other techniques. We performed K-cross validation where K=6 and compared models using the ROC AUC metric and found the ROC AUC accuracy to be higher in U-net compared to GMM, Baseline, etc.**

1. Introduction

The goal of image segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation can be used for: • Object recognition: Identifying different objects in an image. For example, a self-driving car needs to be able to identify other cars on the road. • Edge detection: Finding boundaries in images. This can be useful for object detection as well as creating special effects like vignetting or blurring certain parts of an image We solve this problem using a convolutional neural network which is built on top of vgg16 pretrained imagenet weights and a U-net autoencoder and compare different approaches to solving the problem and analysing which is more effective.

2. Data Preprocessing and Cleaning

2.1. Data augmentation

Image data augmentation is a powerful tool that can be used to improve the performance of machine learning models. By artificially creating new images from existing ones, we can increase the amount of training data available to our models and help them learn better.

There are a number of different techniques that can be used for image data augmentation, such as cropping, flipping, rotation, and adding noise. In this blog post, we'll take a look at each of these methods and how they can be used to improve your machine learning models.

Cropping is one of the simplest forms of image data augmentation. By randomly cropping images during training, we force our model to learn how to deal with incomplete information and make predictions based on partial inputs. This helps it generalize better to real-world scenarios where not all information may be available.

Flipping is another common method for augmenting images. This technique involves horizontally or vertically flipping an image (or both) so that it's mirror image is created. This provides additional training examples for our model without having to actually create new ones from scratch - all we need is the original image plus its flipped version(s). Flipping also forces our model to learn about symmetry and invariance which are important concepts in many real-world applications..

Due to time constraints, we only did the following but in the future hope to create more data.
1. Width shift of 5
2. Height shift of 5
3. Sheer stretch of 10
4. Horizontal flipping
5. Vertical flipping

2.2. Train, test split

Divided the puzzle images into training (70%) training and analysis in the tasks below. This works out to 34 training images, 7 validation and 7 test images. Since Deep learning is data hungry, we fed the 34 training images into a data augmentation pipeline to enlarge the size of our training data to 200 images, keeping the validation and test size same.

3. Vgg16

This model was originally trained on the ImageNet dataset, and has since been adapted to many other tasks. The Vgg16pretrained imagenet weights are a great starting point for many applications.

This architecture is well suited to problems where speed is important, such as in real-time object detection. It's also relatively easy to fine-tune this model to other datasets, which makes it a good choice for transfer learning.

4. U-net

Semantic Segmentation is an important part of any image-related tasks in computer vision. Semantic segmentation is the process of classifying each pixel in an image into one of a set of predefined classes. In the context of puzzle pieces, semantic segmentation can be used to automatically identify and classify different types of puzzle pieces.

One approach to semantic segmentation is using a U-net autoencoder. U-net autoencoders are neural networks that are specifically designed for image segmentation tasks. They have an encoder network that extracts features from an input image, and a decoder network that maps the features back to pixels in the output image. The advantage of using a U-net autoencoder for semantic segmentation is that it can learn complex feature representations from images, which makes it well suited for identifying small objects like puzzle pieces.

To train a U-net autoencoder for semantic segmentation, we first need to annotate our training data with labels indicating the class membership of each pixel (e.g., piece type A, piece type B, etc.). We then train our network on this labeled data so that it can learn to map input images to corresponding output images containing only pixels belonging to the desired class (e.g. pixels belonging to foreground or background class). Finally, we can use our trained network on new unlabeled images to automatically predict labels for all pixels in those images.

5. GMM

Gaussian mixture models are a powerful tool for performing semantic segmentation. They allow us to model the distribution of pixels in each category, and then use that information to classify new pixels.

Using a Gaussian mixture model, we can accurately identify the boundaries between puzzle pieces, even when there is significant overlap between them. This makes it much easier to solve puzzles, as we can simply move pieces around until they fit together perfectly."

GMMs are a type of probabilistic model that assumes each data point is generated from a mixture of Gaussian distributions with unknown parameters. This allows them to flexibly model complex datasets with multiple clusters. GMMs are often used in conjunction with other methods, such as k-means clustering, to improve the accuracy of results.

One advantage of using GMMs is that they can account for uncertainty in the data points by assigning probabilities to each point belonging to each cluster. This makes them robust against outliers and noise in the dataset. Additionally, since GMMs generate continuous probability densities rather than discrete clusters, they can provide more accurate results when dealing with high-dimensional datasets where traditional methods may fail.

6. Baseline

We looked at RGB, DoG(Difference of Gaussian) and MR8 features from the previous labs and trained one GMM for the background and one for the foreground respectively.

7. ROC, AUC Metrics

The AUC has several advantages over other evaluation metrics such as accuracy or precision. First, the AUC is not sensitive to class imbalance which is common in semantic segmentation datasets. Second, the AUC can be used even when there are multiple classes present in an image (multi-class). Finally, the AUC provides a single value that represents overall performance on an entire dataset which makes it easier to compared different models or different methods for training a model.

Despite its advantages, there are some limitations to using the AUC for evaluation. One limitation is that it does not consider localisation information such as whether or not a predicted bounding box contains an object of interest within it. Another limitation is that because it summarises performance across an entire dataset it can be difficult to identify where errors are being made and what needs to be improved

8. Maximum likelihood

Our posterior had the following form:

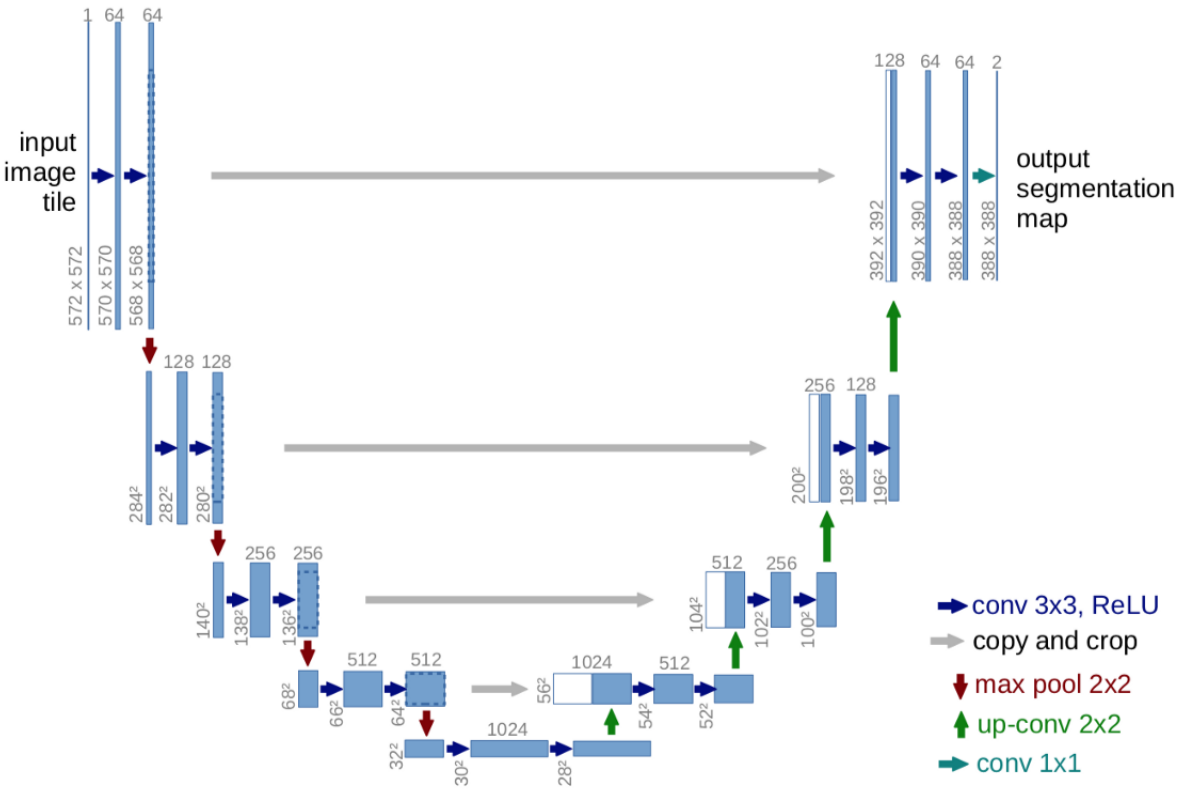
$$\begin{aligned} P(l=f|\vec{x}) &= \frac{P(\vec{x}|l=f)P(l=f)}{P(\vec{x}|l=f)P(l=f)+P(\vec{x}|l=b)P(l=b)} \\ &= \frac{\lambda \mathrm{Norm}_{\vec{x}}[\vec{\mu_f}, \mathbf{\Sigma}_f]}{\lambda \mathrm{Norm}_{\vec{x}}[\vec{\mu_f}, \mathbf{\Sigma}_f] + (1-\lambda)\mathrm{Norm}_{\vec{x}}[\vec{\mu_b}, \mathbf{\Sigma}_b]}, \end{aligned}$$

9. Training Phase

Use of cross entropy loss Use of adam optimizer Use of vgg16 pretrained imagenet weights Use of roc, auc as well as iou score and found IOU works better. number of epochs = 200 Batch Size = 4

9.1. 6-Fold cross validation

Cross validation is a technique for assessing the accuracy of a model. The idea is to split the data into k subsets, train the model on k-1 subsets, and test it on the remaining subset. This process is repeated k times, such that each subset serves



as both a training set and a testing set. The average accuracy across all k runs is then used to assess the model.

There are several benefits of using cross validation over traditional methods like holdout or splitting the data into training and testing sets. First, it helps reduce bias by using all of the data for both training and testing (rather than just one split). Second, it helps reduce variance by averaging out errors across multiple runs. Finally, it provides more reliable estimates of generalization error since each point in the dataset is used as both a training point and a test point at some point during cross validation.

The six fold cross validation method simply splits the data into six equal parts rather than randomly sampling fromthe dataset like other methods do . This can be beneficial if thereis some sortof order or structure tothe dataset that you want to preserve (for example, time series data). However, six fold cross validation does not necessarily provide better results than other methods; it just offers another option for those who wish to use it

We used from sklearn.model_selection import KFold

9.2. Batch size, parameters

9.3. Hardware Used

Nvidia Geforce GTX 1660 Ti GPU AMD Ryzen 7 3750H

10. Results

For baseline - We found that ROC AUC score for RGB features is better than DOG and MR8 features we did in the lab.

For GMM:- ROC AUC score for RGB features was around 0.84

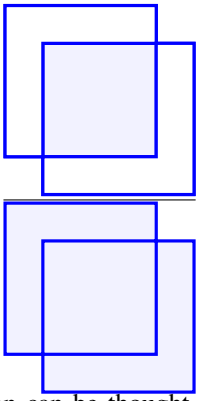
For Unet :- ROC AUC score for RGB features is the highest at 0.89

10.1. Discussion

10.1.1. Intersection Over Union.

Where the labels are the same in all the masks, keep it in the ground truth, and then for where they are not the same, use things like edge detection and morphological image processing operators to get close to the edges in the original image, and take labels after processing all the masks, repeating for where all the pixel values do not match in the masks.

In mathematical analysis, the intersection over union (IoU) of two sets A and B, usually denoted by IoU(A,B), is a measure of the overlap between them. More precisely, it is defined as follows:

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} =$$ 

Intersection over union can be thought of as a generalization of the concept of percent overlap to arbitrary sets. In particular, when A and B are both subsets of some larger set U (not necessarily equal), then their intersection over union can be interpreted as the percentage of U that is shared by A and B.

10.1.2. Assumptions & Problems.

We assume that all the data given is accurate but it is biased and inter-rate reliability is poor. Problems faced are we cannot exactly describe if two images are similar and we have our own biases. Correcting and identifying label bias is an important step in machine learning.

Intersection over Union (IoU) calculation is used to evaluate an image segmentation model. It allows us to evaluate how similar a predicted image is to the ground truth label

11. Conclusion

The results show that U-net outperforms both GMM and Baseline by a significant margin. This is likely due to the fact that U-net is specifically designed for image segmentation tasks. U-net works well with augmented data but would not work well with less images and might fail to generalize for satellite or medical images and possibly problems with multiple small objects in images.

References

Cieślik, Jakub. "How to Do Data Exploration for Image Segmentation and Object Detection (Things I Had to Learn the Hard Way) — Neptune Blog." Neptune.Ai, 28 May 2020, https://neptune.ai/blog/data-exploration-for-image-segmentation-and-object-detection.

"Label Bias and How to Prevent It Explained by Peltarion." Peltarion, https://peltarion.com/knowledge-center/ai-concepts/bias-in-ai/data-bias/label-bias. Accessed 3 Oct. 2022.

12. Appendix

Team work -

1. Nilesh Jain - GMM, Unet and Report Writing, Github
2. Abdel Njupoun - Baseline, GMM, Github
3. Bathandwa - GMM, Github