UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
SCHOOL OF COMPUTER SCIENCE & APPLIED MATHEMATICS

COMS4036A & COMS7050A
Computer Vision

Devon Jarvis, Nathan Michlo
Dr Richard Klein

Project: Revisiting Background Segmentation
Gaussian Mixture Models & Deep Learning

Introduction

In previous labs, we have been helping Quarantino solve his puzzle. In Lab 0, we worked with Tino to create several ground truth masks which indicated which pixels belonged to the background and which ones belonged to the puzzle piece. We then trained a number of different binary classifiers to automate this process in the future. Among others, our classifiers relied on RGB, HSV, Haar-like features, Local Binary Patterns, Textons, Difference of Gaussians, Laplacian of Gaussian, Histograms of Oriented Gradients, MR8 features to extract information that we believed would help discriminate between the different types of pixels. We then built a generative model where we used multivariate normal distributions $(\vec{\mu}_b, \mathbf{\Sigma}_b, \vec{\mu}_f, \mathbf{\Sigma}_f)$ to represent the class conditional probabilities and a Bernoulli prior (λ) over the pixel labels. We used Bayes' rule to calculate the posterior distribution, telling us the probability that a specific pixel was part of the puzzle piece or the background.

Our posterior had the following form:

$$P(l = f|\vec{x}) = \frac{P(\vec{x}|l = f)P(l = f)}{P(\vec{x}|l = f)P(l = f) + P(\vec{x}|l = b)P(l = b)} \tag{1}$$

$$= \frac{\lambda \text{Norm}_{\vec{x}}[\vec{\mu_f}, \mathbf{\Sigma}_f]}{\lambda \text{Norm}_{\vec{x}}[\vec{\mu_f}, \mathbf{\Sigma}_f] + (1 - \lambda)\text{Norm}_{\vec{x}}[\vec{\mu_b}, \mathbf{\Sigma}_b]}, \tag{2}$$

where $\vec{\mu}_b, \mathbf{\Sigma}_b, \vec{\mu}_f, \mathbf{\Sigma}_f$, and λ are the parameters of the class conditional likelihoods and the Bernoulli prior respectively. We were usually able to find a threshold on this posterior to make a binary decision and we considered how to do this by looking at ROC and AUC curves. However, in most cases, our model failed to generalise well to unseen puzzle pieces.

We made a few important assumptions which did not always fit with our data. We assumed (a) that the extracted features were normally distributed within each class and (b) that the hand-crafted features that we were extracting would contain the right discriminative information for the model to learn and generalise adequately.

In this project, you will need to build two fundamentally different types of model and you should investigate what impact they have on the accuracy and generalisation of the background extraction process – while still considering the data constraints that we have.

To address the assumption of unimodal normal distributions over our features, we are going to consider a Gaussian Mixture Model as a way to model more complex probability densities. To avoid having to handcraft and analyse the efficacy of different features, we are also going to use a Deep Learning approach which is usually applied to semantic segmentation problems.

You will need to implement these models as described below, and both train and analyse them using our labelled dataset. You will then need to write a 4 – 6 page report on your findings using the IEEE proceedings double-column LaTeX template. You will also have to write a 1 page reflection discussing your experience of the puzzle-solving process.

1 Dataset

You should use the cleaned masks and images from Lab 6 provided in `puzzle_corners_1024x768__v2.zip` on Moodle. COMS4036A students will only need the masks and images, COMS7050A students will need corner locations as well.

Divide the puzzle images into training (70%), validation (15%) and test (15%) sets, which you will use for training and analysis in the tasks below. This works out to 34 training images, 7 validation and 7 test images. Consider the implications of a dataset this size for the methods below and explain whether or not it will be a problem for each.

Explain how you split your data into these sets.

Grading Note: The last 5% of each task will be awarded for submissions that perform, discuss, and analyse 6-fold cross-validation as well.

2 Gaussian Mixture Models

Recall from Chapter 7 of the textbook that a Gaussian Mixture Model is a marginalisation of the joint probability distribution $P(\vec{x}, h)$ where $\vec{x}$ represents our feature vector and h is a hidden categorical variable representing from which distribution in the mixture a point originates.

$$
\begin{aligned}
P(\vec{x_i}) &= \sum_k P(\vec{x}_i|h_i = k)P(h_i = k) && (3)\\
&= \sum_k \mathrm{Norm}_{\vec{x_i}}[\vec{\mu}_k, \mathbf{\Sigma}_k]\mathrm{Cat}_{h_i}[\vec{\lambda}] && (4)\\
&= \sum_k \lambda_k \mathrm{Norm}_{\vec{x}_i}[\vec{\mu}_k, \mathbf{\Sigma}_k] && (5)
\end{aligned}
$$

This formulation allows us to fit the model with the Expectation Maximisation algorithm, and in this case, we arrive at closed-form solutions for all of the update steps.

In the expectation step, we assign points to distributions in the mixture by calculating the responsibility that

distribution k takes for datum i, i.e. $r_{ik} = P(h_i = k|\vec{x_i})$.

$$r_{ik} = \frac{\lambda_k \text{Norm}_{\vec{x_i}[\vec{\mu_k}, \mathbf{\Sigma}_k]}}{\sum_{j=1}^{K} \lambda_j \text{Norm}_{\vec{x_j}[\vec{\mu_j}, \mathbf{\Sigma}_j]}} \tag{6}$$

In the maximisation step, we assume that the responsibilities are fixed and update the parameters of the distribution using, for example, maximum likelihood. This yields the following closed form update rules:

$$\lambda_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K} \sum_{i=1}^{I} r_{ij}} \tag{7}$$

$$\vec{\mu}_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik} \vec{x}_i}{\sum_{i=1}^{I} r_{ik}} \tag{8}$$

$$\mathbf{\Sigma}_k^{[t+1]} = \frac{\sum_{i=1}^{I} r_{ik} (\vec{x}_i - \vec{\mu}_k^{[t+1]})(\vec{x}_i - \vec{\mu}_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}} \tag{9}$$

You iterate these steps until you reach convergence in your parameters.

Note: There is often a problem when a distribution locks onto a single point in the space. This causes it to have singular covariance and an infinite likelihood. To combat this, you could use a Maximum a Posteriori training regime where the prior ensures that the covariance matrices are never singular.

Note: During random initialisation of the parameters, it is important to remember that the covariance matrix should always be positive definite. To do this, first create a random $n \times n$ matrix: `A = np.random.random(n,n)`, then transform it so that it is symmetric: `A *= A.T` or `A += A.T`, and then add some positive multiple of the identity matrix: `A += n*np.eye(n)`. Matrix `A` is not positive definite. It may be wise to multiply the identity matrix by another large positive number so that the initialisation gives the distributions a large covariance to start with.

Train one GMM for the foreground and another for the background, then replace the normal class conditional likelihood terms in Equation 2 with the relevant mixture models. Do not confuse the λ class prior in Equation 2 with the λ_k categorical weights inside the GMM. These are different quantities.

You should consider the following feature sets:

1. RGB pixel value
2. RGB+Difference of Gaussian
3. One other feature set from the previous labs that performed well.

Evaluate this new model using your validation dataset and explore different numbers of components in the two mixtures and vary the other hyperparameters you have in your features. Once you have chosen the optimal hyperparameters using your validation set, perform a final evaluation of your model using your test set.

In your report, present your results and compare this model with the previous models. How do they compare in terms of training time, inference time, number of parameters, accuracy etc.?

MSc Only: Analyse the components of your GMM and try to assess what aspect of the image each component is responding to. Are the components learning specific subclasses of the data within the foreground/background classes?

Submission Note: You may use `scipi.stats`'s `multivariate_normal` as well as general `numpy` functions but you should implement your actual GMMs from scratch. You may even use Tensorflow/PyTorch to parallelise your implementations, but the actual GMM mathematics cannot come from a library. You should

definitely vectorise your code, note that for-loops in python will introduce a severe computational cost in this model.

Grading Note: You should structure your code sensibly and reuse functions/classes where-ever possible. I suggest creating a GMM class which you can reuse when you need to build a separate background and foreground models.

3 Deep Learning Techniques: UNet & DeepLabV3

A visual problem that is closely related to our current task is Semantic Segmentation. In semantic segmentation, we want to assign a class to each pixel in the image that represents that type of object we see in the image at that location.

For example, Figure 1, shows an image from the Cityscapes Dataset. For example, blue pixels represent cars, red pixels represent people, and pink represents the sidewalk. Fully Convolutional Neural Networks are a popular technique that is used for this task and there are a number of famous state of the art models that work in different ways and focus on slightly different aspects of the problem – e.g. instance segmentation.

Figure 1: Cityscapes Semantic Segmentation [https://www.cityscapes-dataset.com/]

We can re-imagine our background segmentation task to be the simplest form of semantic segmentation where we only have two classes that can be assigned to each pixel: puzzle or background. In this part of the project, you should use Tensorflow, Keras, or PyTorch. While there are various newer networks that we could consider, such as DeepLab v3 or MaskRCNN, for simplicity, we will use a UNet model with pre-trained weights from VGG16.

Setup the model using your favourite deep learning framework and load in pre-trained weights based on VGG16. You should then fine-tune your network using the training dataset that we have already constructed. Use your validation dataset to monitor generalisation performance and over-fitting during training.

Note that the small size of our dataset (i.e. only 48 images), means that Deep Learning may be too data-hungry, and we should be very aware of over-fitting. To try to increase the size of our dataset, perform image augmentations as a pre-processing step during training. Consider which augmentations may be valid or invalid in our context. You should run experiments with and without data augmentation and compare the test set accuracy to understand whether this is helpful.

You should include a review of the UNet architecture and how it works, how it is trained etc.

Present your results and compare the performance of UNet with the GMM and original segmentation models that you built in the lab. Perform any other analysis that you think might be useful and present your findings.

MSc Only: Previously, we also struggled to accurately localise and extract the corner points of the puzzle pieces. Use the corner locations in the dataset to create a third class in your mask. Let pixels that sit on the four corners of the puzzle piece belong in this third class. You should then retrain your model on this new dataset

and analyse whether the UNet model is able to reliably extract the positions of corners. What is the best way to incorporate this information into your label mask? How big should these corner regions be to be reliably detected by the model?

4 Final Reflection [Individual Submission]

Write a 1 page analysis explaining your findings regarding Tino's puzzle-solving problem. To deploy a production system what techniques/features/models would you recommend and what are the benefits/drawbacks of your chosen approach? Provide a full pipeline from start to finish for a system to solve the problem. Identify any missing components that we haven't yet considered and explain how they might be designed and trained.

5 Grading & Submission

- You may work in pairs for all but the reflection in Section 4.
- Your report *must* be typeset in LaTeX using the IEEE double-column conference format.
- You should submit your report as a group on Moodle, and you should submit your reflections independently.
- Your report should have an appendix that outlines how the work was distributed between you. I may adjust grades where there is a large discrepancy in the contribution of group members.
- You may include extra graphs etc. in the appendix.
- The appendix and any references you may have do not contribute to your page count.
- You must submit your code with a readme explaining where everything is and how it should be run. We will not necessarily look at the code in detail unless we need to check the details of your implementation.

Table 1: Grading

Item	Details	Weight
Document	Look and feel of the document, writing, language, correct use of LaTeX	8%
Code Style	Code style, correctness, comments, formatting etc.	8%
Data Prep	Discussion of the data, correct splits.	8%
GMM	Implementation and Analysis, correct results & conclusions, shows insight, analysis of features, analysis of number of components, discussion of issues and resolutions such as singular cov, comparison to previous models and feature sets	25%
	+ 6-fold cross validation and analysis	5%
	MSc Only: Analysis of the individual components. Part of 25% above.	
UNet	Implementation and Analysis, correct results & conclusions, shows insight, explanation of UNet arch/loss etc., comparison with GMM model, discussion on learned vs handcrafted features	25%
	+ 6-fold cross validation and analysis	5%
	MSc Only: Implementation, Analysis & Discussion of corner detection using UNet. Part of 25% above.	
Reflection	Presentation of a sensible pipeline, identifies what did and did not work up to this point, presents a sensible design given this information, discussion on methods/techniques/data/compute requirements, suggested extensions for improvement overall, critique of the current approach and limitations of the suggest approach.	16%