**Using Gaussian Mixture Models with Deep Neural Network features for Image Segmentation.**

**Victor Megir**

**Diploma Thesis**

Supervisor: Prof. Christophoros Nikou

Ioannina, August 2020

**Τμήμα Μηχ. Η/Υ & Πληροφορικής**

**Πανεπιστήμιο Ιωαννίνων**

**Department of Computer Science & Engineering**

**University of Ioannina**

**Acknowledgments**

George Sfikas

<Date>

<Author>

**Abstract**

Semantic Image Segmentation is a challenging problem in computer vision. Getting good results requires image features that hold semantic, real world information. Recent developments in computer vision however, especially the use of Deep Convolutional Neural Networks have allowed for the extraction of high quality features. These features are the result of state of the art neural networks that have trained on massive datasets. Leveraging these high quality features can allow for quality image segmentation, using simpler techniques and significantly smaller datasets. Statistical modeling and clustering techniques, such as Gaussian Mixture Models can be applied on the feature data, while feature engineering and dimensionality reduction can come a long way in improving the segmentation results.

**Keywords:** Deep Features, Gaussian Mixture Model, Dimensionality Reduction.

**Περίληψη (extended abstract in Greek)**

Η κατάτμηση εικόνας αποτελεί ένα από τα πλέον απαιτητικά προβλήματα στον τομέα της υπολογιστικής όρασης, όμως ένα με ενδιαφέρουσες εφαρμογές και μεγάλο ερευνητικό ενδιαφέρον. Η σημαντική πρόοδος που έχει επιτευχθεί τα τελευταία χρόνια οφείλεται στη χρήση μοντέρνων μεθόδων αναγνώρησης προτύπων και ιδιαίτερα τα βαθιά συνελικτικά νευρωνικά δίκτυα. Χάρη στις αρχιτεκτονικές τους, αυτά τα νευρωνικά δίκτυα έχουν τη δυνατότητανα να εξάγουν χαρακτηριστικά από εικόνες, με τα οποία επιτυγχάνουν ασύγκριτα καλύτερα αποτελέσματα σε σχέση με χαρακτηριστικά δομημένα από ανθρώπους.

Η ποιότητα των χαρακτηριστικών που χρησιμοποιούνται, αποτελούν κρίσιμο παράγοντα για την ποιότητα της κατάτμησης εικόνας. Διαφορετικά χαρακτηριστικά αποτυπώνουν διαφορετικές πτυχές των δεδομένων μιας εικόνας και μπορεί ορισμένα χαρακτηριστικά να αποτυπώνουν πιο βασικές πληροφορίες απ΄ότι άλλα και επομένως να μην είναι όσο χρήσιμα για συγκεκριμένες εφαρμογές. Στα βαθιά νευρωνικά δίκτυα τα οποία αποτελούνται από πολλά επίπεδα συνελικτικών νευρώνων, τα πιο ειδικευμένα ως προς την εφαρμογή χαρακτηριστικά βρίσκονται στα πιο βαθιά συνελικτικά επίπεδα.

Τα πλέον επιτυχημένα βαθιά νευρωνικά δίκτυα είναι δίκτυα τα οποία έχουν δημιουργηθεί από μεγάλους ερευνητικούς οργανισμούς και ερευνητές με πολλές γνώσεις, ενώ έχουν εκπαιδευτεί για να αντιμετωπίζουν δύσκολα ερευντικά προβλήματα αξιοποιώντας τεράστιους όγκους δεδομένων.

Ωστόσο μπορούν να χρησιμοποιηθούν για διαφορετικές εφαρμογές από αυτές για τις οποίες δημιουργήθηκαν, απλά αξιοποιώντας τα βαθιά χαρακτηριστικά που εξάγουν από τις εικόνες ενός μικρότερου συνόλου.

Χρησιμοποιώντας αυτά τα βαθιά χαρακτηριστικά σε συνδυασμό με απλούστερες τεχνικές και αγλοριθμους, μπορούμε να πετύχουμε καλά αποτελέσματα στην κατάτμηση εικόνας με μικρότερο σύνολο δεδομένων και κατα συνέπεια μικρό χρόνο εκτέλεσης. Τεχνικές στατισκής μοντελοποιήσης, όπως Γκαουσινά Μοντέλα Μίξης (Gaussian Mixture Models) μπορούν να πετύχουν καλά αποτελέσματα προσεγγίζοντας την κατανομή των δεδομένων στα βαθιά χαρακτηριστικά.

Ταυτόχρονα, τεχνικές μείωσης διάστασης μπορούν να βοηθήσουν ώστε να συμπτυχθεί η χρήσιμη πληροφορία των χαρακτηριστικών ώστε να αποφευχθεί η *Κατάρα της Διάστασης*. Επιπλέον , απλά φίλτρα εφαρμοσμένα στο αποτέλεσμα της κατάτμησης μπορούν να βελτιώσουν το αποτελέσμα.

Σε αυτή την εργασία προτείνουμε τη χρήση βαθιών χαρακτηριστικών από νευρωνικά δίκτυα σε συνδυασμό με Γκαουσιανά Μοντέλα Μίξης και Μείωση Διάστασης για κατάτμηση εικόνας.

**Λέξεις Κλειδιά:** Βαθιά Χαρακτηριστικά, Γκαουσιανά Μοντέλα Μίξης, Μείωση Διάστασης.

**Table of Contents**

[**1. Introduction 6**](draft2.docx#Introduction)

[**2. Related Work 7**](draft2.docx#Related_Work)

[**3. Methods 8**](draft2.docx#Methods)

[Gaussian Mixture Model 8](draft2.docx#Gaussian)

[EM Algorithm *9*](draft2.docx#EM)

[Soft Clustering 11](draft2.docx#Soft_Clustering)

[DeepLabv3+ 11](draft2.docx#DeepLab)

[Deep Features 12](draft2.docx#Deep_Features)

[The Curse of Dimensionality 13](draft2.docx#Curse)

[Principal Component Analysis 14](draft2.docx#PCA)

[Filtering 17](draft2.docx#Filtering)

[**4. Experiments**](draft2.docx#Experiments) **17**

[**5. Conclusions**](draft2.docx#Conclusions) **17**

[**References 18**](draft2.docx#References)

**1. Introduction**

The use of Deep Convolutional Neural Networks (DCNN) in recent years has pushed the progress of computer vision to an unprecedented level. Image classification and object detection have been among the biggest successes in computer vision, as state of the art classifiers are used daily for a variety of complicated tasks such as medical image recognition, face detection, product classification among many. This progress has been possible thanks to the modern DCNN architectures, which are able to extract image features of superior quality than any hand crafted features thus achieving very good results.

Image segmentation is too one of the computer vision problems that has seen great progress with the use of DCNNs, as new state of the art neural networks progressively push the accuracy and performance on some of the hardest problems.

These state of the art DCNNs can be repurposed for tasks that they were not originally built for, allowing us to leverage the information they have obtained from the massive image datasets that they have been trained on. Their feature extraction architecture can be used to extract features from other datasets. These high quality features can then be used in combination with other techniques to achieve a good result in different tasks.

The techniques that can be used can vary depending on the task, as well as the quality and quantity of the extracted features. Real world data is messy and unpredictable, creating the need for pattern recognition techniques with the ability to model complicated data.

In this work we propose the use of a Gaussian Mixture Model to cluster the feature data, extracted from a set of real world images, for the purposes of image segmentation. To extract these features, we will be using the Deeplabv3+ Deep Convolutional Neural Network and we will be extracting the features from the deeper Convolutional layers to achieve better results.

**2. Related Work**

Most successful image segmentation systems of the previous decade relied on hand crafted features such as Gabor features, or features created with Sobel, Prewitt, Roberts or Gaussian filters. These features were then used in combination with a classifier such as a Support Vector Machine or a Random Forest classifier. Even though these systems were robust, they were limited by the quality of the hand crafted features.

In recent year, the rise of Deep Convolutional Neural Networks (DCNN) has improved significantly the feature quality, with state of the art neural networks such as ResNet, DenseNet, Inception, GoogleNet and VGGNet among others, for image classification and Mask R-CNN, U-Net and DeepLab for image segmentation.

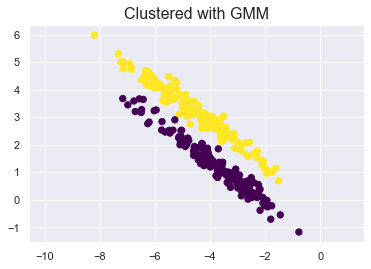
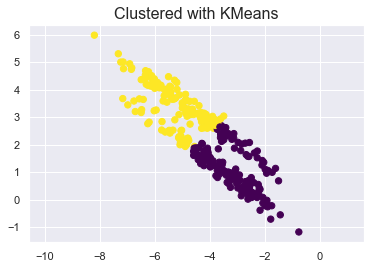
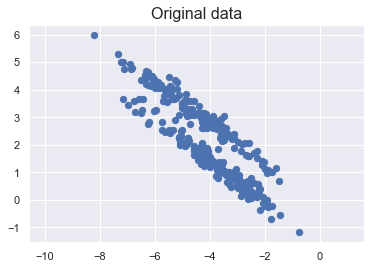
These state of the art neural networks have introduced some innovative architectures for feature extraction, as well as some interesting variations that can be implemented to achieve the best possible result depending on the task and the available data. Most of these neural networks use several fully connected convolutional layers, with features extracted from lower levels of the network being fed forward deeper into the network. The convolutional layers are usually followed by a pooling layer, as is the case with VGGNet, ResNet and others. After the pooling layer several fully connected convolutional layers follow and the process continues. This process allows for the extraction of progressively more expressive features which are ultimately fed into a linear classifier and a softmax layer. A particularly interesting variation of this architecture is introduced by DeepLab which uses *Atrous Convolution* to improve on the feature density of the extracted features.

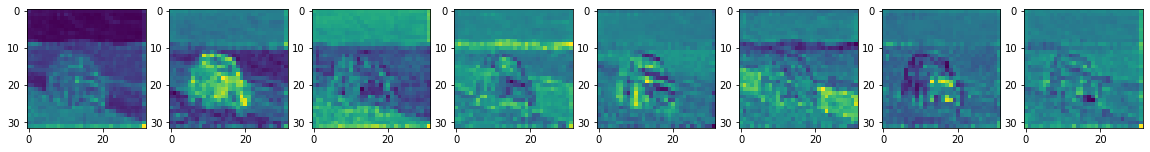
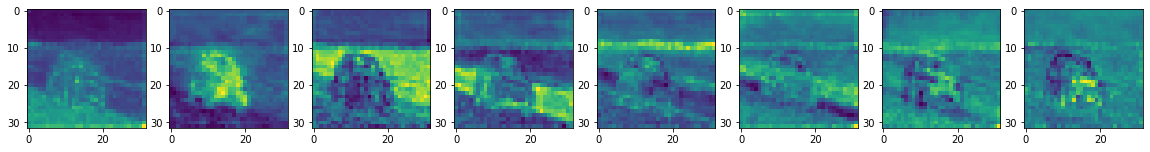
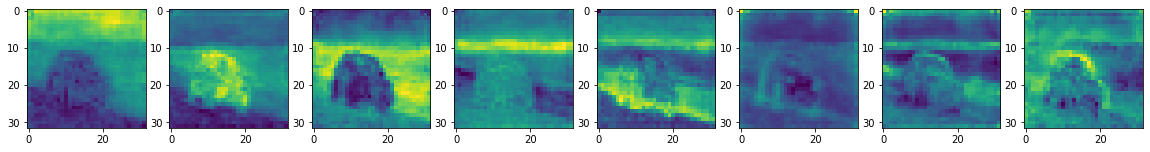
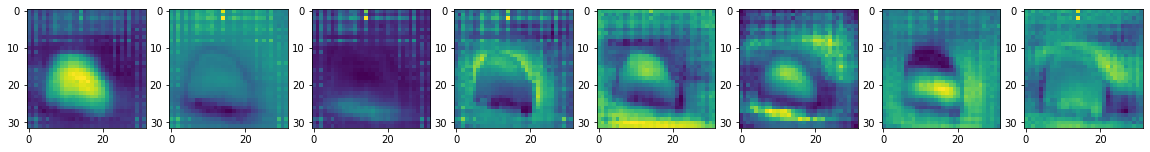
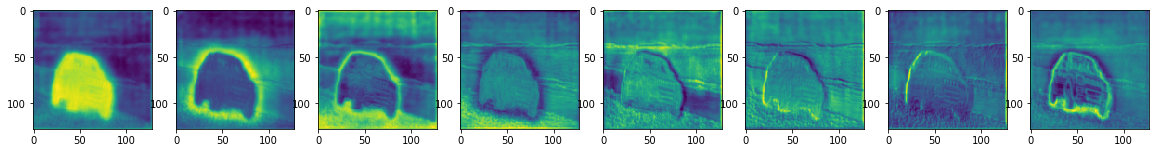
Clustering algorithms have also been used to perform image segmentation on image data in cases where the task does not specify a certain number of classes, in which case a classifier would be ideal. The K-Means algorithm as well as the Estimation Maximization algorithm has been used on raw image data with relative success, especially in segmentation of medical images.

Using features extracted from deep neural networks for a different task is not a new idea. In fact, extract features from the fully connected convolutional layers of the state of the art neural networks, in combination with a custom linear classifier has wielded great results in tasks where the available data was limited. Features extracted from neural networks have also been used for clustering image data in combination with popular clustering algorithms.

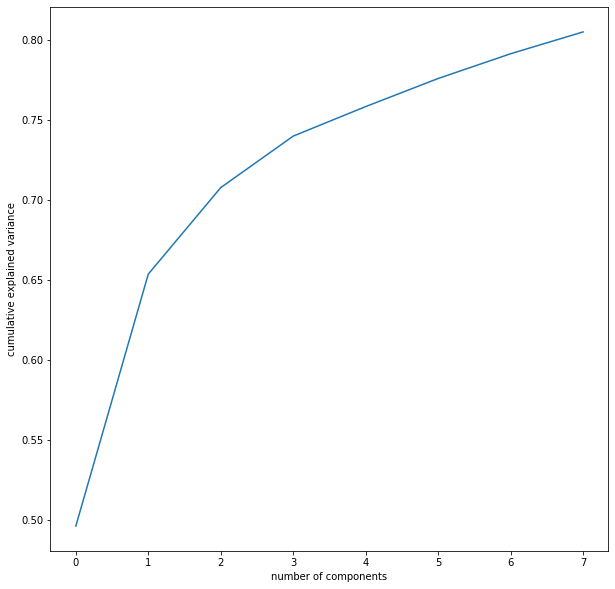
**3. Methods**

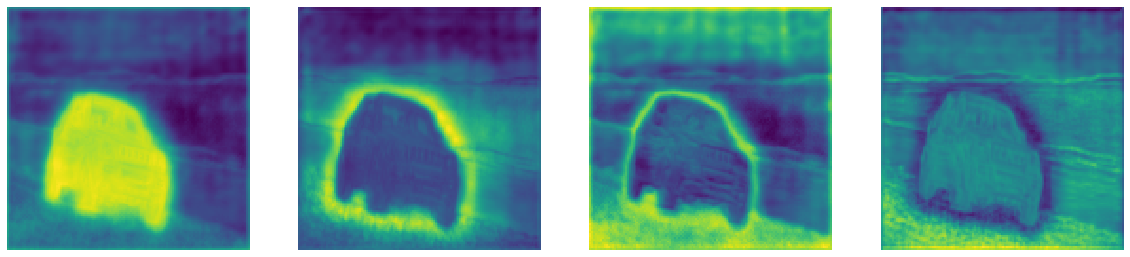
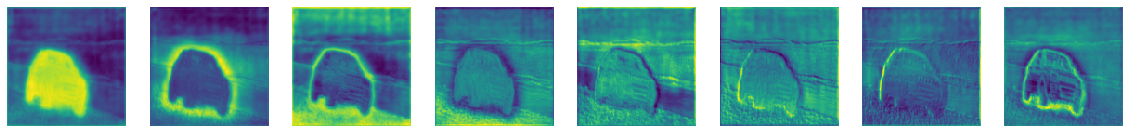
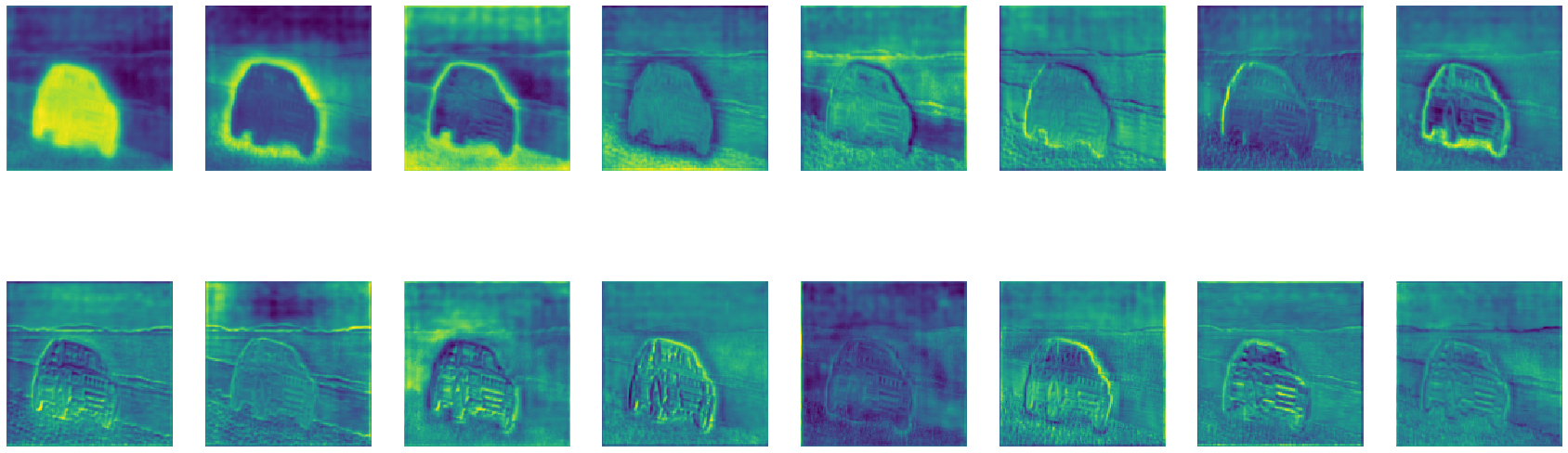
**Gaussian** **Mixture Model**

In statistics a mixture model is a probability model that describes a set of data populations as a singular population. A mixture model is therefore defined by a single probability distribution that incorporates the probability distributions that define all its sub-populations. In essence a mixture model is a probability model that expresses the properties of data with different probability distributions.  
That allows a mixture model to describe data with more complex distributions and which makes it good for modeling real world data. A mixture model is defined by a set of K probability distributions which are described by the set of parameters that uniquely define each of the K probability distributions. A mixture model can optionally include a set of K weights, the sum of which is 1, such that each corresponds to a distribution and defines its contribution to the mixture model’s distribution. Any random variable that is described by a mixture model of K distributions is required to have K components.   
Mixture models are usually used to do deduce properties of the K distributions by using random variables that are described by the mixture model’s distribution. In the context of pattern recognition, mixture models can be used for clustering and unsupervised learning.  
The simplest mixture model is the Gaussian Mixture Model which is defined by a set of K Gaussian or Normal distributions that may be univariate or multivariate and a set of K weights. A Gaussian Mixture Model requires a mean for each of its K distributions but in order to convey the relations between the distributions it also requires the covariance between the variables of each distribution. All the covariances are described as a square K x K positive defined matrix called covariance matrix. A Gaussian Mixture Model is therefore uniquely defined by its K means, covariance matrices and weights.   
  
To determine the optimal values for the model we need to find the maximum likelihood of the model. We can find the likelihood of the model by calculating the joint probability of all the N data points, which is defined as:   
Where X represents the N data points of a given dataset, π represents the set of K weights, μ represents the means of the K distributions, Σ represents the covariance matrices for the K distributions and the equation (1.3) represents the probability density function of a multivariate Gaussian distribution.   
In order to determine the optimal values for the parameters of the model given the equation (1.1) we need to use an iterative method for optimization called Estimation Maximization also known as the EM algorithm.   
  
**The EM** **algorithm**The EM algorithm is an iterative method that is widely used to estimate the parameters of a statistical model in order to maximize the likelihood function of the model. It is essentially the alternation of two operations, *Expectation* and *Maximization*, for several iterations.   
Initially the parameters of the statistical model need to be initialized. Usually the initial initialization of the parameters is random; occasionally however the parameters are initialized as the results of another iterative method such as KMeans. A good initialization is important because it means fewer iterations until convergence. In our case let the parameters of a Gaussian Mixture Model be:   
The *Expectation* step calculates the likelihood function for the current values of the parameters of the model. In essence the likelihood calculated during the *Expectation* step defines the responsibility of each of the N data points with respect to the K distributions of the model. In other words the *Expectation* step calculates how likely it is for any data point to belong in a specific distribution.   
  
  
The calculation made during the Expectation step, for the Gaussian Mixture Model for a given iteration *t* is the one that follows:  
The *Maximization* step updates the parameters of the K distributions such that the likelihood function of the model is maximized. In order to do so the *Maximization* step uses the responsibilities calculated during the *Estimation* step for each of the N data points. The parameters that get updated for the Gaussian Mixture Model are the K means, covariance matrices and weights. The updated values for these parameters for a given iteration *t* are defined as follows:  
Where is the responsibility of data point *n* for the distribution *k* during iteration *t*. The means of the K distributions *μ* are updated as defined by the equation (2.3), the covariance matrices *Σ* as defined by (2.4) and the weights *π* as defined by (2.5).  
The alternation of the *Estimation* and *Maximization* steps is repeated until some convergence criterion is met. For instance, when the changes that happen to the means of the distributions, after an update, are smaller than a specified tolerance level. The EM algorithm may not converge on the global maximum of the likelihood function but instead a local maximum, that is why a convergence criterion needs to be specified.  
After the EM algorithm has converged, the parameters of the K distributions of the Gaussian Mixture Model will be updated optimally, so that the likelihood function of the model is maximized. At this point the Gaussian Mixture Model may be used for inference on new data.   
We can do inference on new data points using the Gaussian Mixture Model by performing a single *Expectation* step on any new data point. This will give us the responsibility of the new data point with respect to each of the K distributions of the model. In other words we can use the *Expectation* step to calculate the probability that the new data point belongs in each of the K distributions. This means that we can use a Gaussian Mixture Model for soft clustering.  
**Soft Cl****ustering**  
Soft clustering or fuzzy clustering techniques are such techniques that allow for each data point to belong in multiple clusters. Essentially, instead of assigning a data point to a specific cluster like how the KMeans algorithm does, soft clustering gives each data point a probability that it belongs to every cluster of the clustering model.   
The benefit of soft clustering compared to hard clustering is the ability to cluster overlapping data or data with some shared features. This gives a soft clustering technique the ability to model complex data, such as real world images as well as features extracted from images.  
  
Figure : Example where soft clustering (Gaussian Mixture Model) performs better than hard clustering (KMeans).

The reason that we used a Gaussian Mixture Model for the purposes of image segmentation is that we needed a clustering technique with the capability of modeling the complex feature data extracted from a neural network. The neural network we chose to extract features from is the DeepLabV3+ neural network.  
  
**Deep****LabV3+**This neural network belongs in Tensor Flow’s Model Garden. It has been trained for semantic image segmentation on the *PASCAL VOC-2012* dataset with 21 classes for segmentation. It is a *Deep Convolutional Neural Network* (DCNN) and it uses several high level methods to achieve the *state of the art* solution at the *PASCAL VOC-2012* problem at 2017, reaching 79.7% mean *Intersection over Union* (mIoU). One of the ways that the DeepLab network is special is that it uses *Atrous Convolution* or Dilated Convolution, a variation of the standard convolution that introduces a spacing between the elements of the convolution kernel, effectively increasing the field of view of the convolution without increasing the number of parameters in the convolution kernel. The operation for the *Atrous Convolution* for a 1D input array *x* is defined as:  
Where *w* is the convolution kernel, *K* is the convolution kernel’s length and *r* is the rate of dilation or the spacing between the kernel’s elements. *Atrous Convolution* allows for the DeepLabV3+ network to retain high spatial resolution in its features. This means that the feature maps generated by the convolutional layers of the network are dense, allowing for more accurate segmentation.  
   
**Deep Feat****ures**  
The idea of extracting features from a state of the art neural network and using them in combination with other techniques is an extremely powerful idea. Instead of engineering features from scratch we can extract as many features as we need from neural networks trained on large datasets and use them to get good results on our own task using a smaller dataset. Some of the state of the art neural networks have been trained on huge and challenging datasets and by repurposing the features that they generate, we are able to exploit the knowledge that these neural networks were able to extract from these datasets.   
However not all features are as useful; the features extracted from lower layers of the network are more general and less task specific compared to features from deeper layers of the network. In the context of semantic segmentation this means that features from deeper layers of the network have more significant semantic information. Therefore it makes sense to use features extracted from deeper layers, the more similar our task is with the original. Similarly if our task is entirely different from the task of the original neural network, then the information of features extracted from the deeper layers is going to be too high level and task specific and will not wield the best results.  
To achieve the best results we need to extract features from intermediate layers of the neural network, deep enough so that the features contain as much useful information as possible, but not so deep that they are too task specific.  
 Figure 2: Example image for feature extraction.  
  
Figure 3: n = 8 *Principal Components* for features extracted from layer 67.  
  
Figure 4: n = 8 *Principal Components* for features extracted from layer 131.  
  
Figure 5: n = 8 *Principal Components* for features extracted from layer 259.  
  
Figure 6: n = 8 *Principal Components* for features extracted from layer 355.  
  
Figure 7: n = 8 *Principal Components* for features extracted from layer 405.  
  
One of the most significant attributes of the extracted features is the high dimensionality. If we were to use these high dimension features to fit the parameters of a Gaussian Mixture Model, the result would certainly be affected by the *Curse of Dimensionality*.  
 **The Curse o****f Dimensionality**

The curse of dimensionality is an important problem that stands to limit the performance and accuracy of our model. It is an effect that always becomes a concern when we do pattern recognition on some data in high dimensional spaces. When we input high dimension data to a machine learning model, this data defines the space in which the model will search for the optimal solution for the problem. So, if the data is high in dimension that means that the space which the model has to explore to find the solution is going to be really big. That makes it so the model requires more and more data to find a good solution. In fact, as the dimension of the data increases, the amount of data required increases exponentially and eventually it reaches the point of diminishing returns where inputting the model more data does not increase its accuracy and performance.

The reason for this effect happening is that when the dimension of the data increases, the data becomes sparse and it becomes difficult for pattern recognition algorithms to detect the similarities in the data. Therefore, more data is required but this only makes the problem worse. Also, more data and high dimension data means longer training times and higher memory requirements. It is therefore necessary to combat this effect by reducing the data dimension. A common technique for this is *Principal Component Analysis*.  
  
**Principal Comp****onent Analysis (PCA)**Principal Component Analysis is a process that allows us to transform a set of data such that the variance of each feature of the data is maximized. In other words *Principal Component Analysis* allows us to “reframe” the data in a way that its features are better separated.   
A benefit of this is that we can order the features of the data by their variance, essentially how important each feature is to the entirety of the data. With that in mind we can reduce the dimension of the data by discarding the features with the lowest variance and keeping as many of the features with the highest variance as we need.   
In order for PCA to achieve this, first the data needs to be normalized. This is an essential step to the process because if there are large differences between the ranges of the values of the data, those values will dominate over the other values leading to biased results. Normalizing the data is a simple transformation:  
where *x* is the value of feature of the data *μ* is the mean of the value distribution for that feature of the data and *σ* is the standard deviation of the value distribution.   
The next step is to calculate the covariance matrix for the features of the data. The covariance matrix is an N x N symmetric matrix, where N is the number of dimensions, and it contains the variance of each feature of the data with respect to all the other features. The covariance matrix can be defined as follows:  
Where are the features of the data. Positive covariance between two features means they are correlated and negative covariance means they are inversely correlated. Since the covariance of a variable with itself is equal to its own variance ( ), that means that all the elements of the diagonal are the variances for each feature of the data. On addition to that the covariance of two variables is commutative ( ), which means that the covariance matrix is symmetric. The covariances are calculated as follows:  
Where X, Y are the discrete random variables that represent the features of the data, and are their respective means and and for *i* = 1,…,*n* are the values they can take.  
PCA uses the covariance matrix to create new features that are linear combinations of the original data features. These new features called *Principal Components* are orthogonal which essentially makes them uncorrelated. Each *Principal Component* is basically a line that maximizes the variance of the features of the data. A line with higher variance means that more data points can be described by that line which means that it holds more information about the data. PCA maximizes the variance of each *Principal Component* by selecting the line that maximizes the average of the squared distances from any data point to the line. This line is the first *Principal Component*; PCA calculates the second *Principal Component* by doing the same process using the first *Principal Component* and its orthogonal line as axes. The same process happens until the total of *Principal Components* is equal to the number of the original features of the data.   
It makes sense that not all *Principal Components* have the same variance, the first one has the greatest variance, and the second has the second greatest and so on.   
In order to perform dimensionality reduction we can simply discard those *Principal Components* that have small variance and project our data in the space of the first few *Principal Components* which will allow us to keep most of the useful information of our data but with significantly smaller dimension. We are sacrificing a small amount of information in exchange for low dimension data that is easier to work with.   
Dimensionality reduction is important to visualize high dimension data, detect latent features in the data, and reduce noise but most importantly to combat the Curse of Dimensionality and to increase the effectiveness of the Gaussian Mixture Model.  
  
Figure 8: Variance ratio for each Principal Component.

We use *n* = 8 *Principal Components* to reduce the feature data that we extracted from the DeepLabV3+ neural network to a manageable dimension. This not only helps the fitting times to be drastically reduced but also the *Principal Components* contain superior semantic information.  
  
  
Figure 9: *n* = 4 *Principal Components.*   
  
Figure 10: *n* = 8 *Principal Components.*  
Figure 11: *n* = 16 *Principal Components.*

**Filter****ing**  
Using simple image filters can go a long way in improving the result of the image segmentation. Filters can be used for a variety of image processing tasks such as denoising, sharpening and smoothing among others. In our case, the goal is to make the segmentation boarders of a segmented image clearer, as well as to remove noise from created by the imperfections of segmentation.

  
Figure 12: Original image and the resulting segmented image with a Gaussian Mixture Model of n=2 clusters. The segmented image has some noise due to imperfections of the segmentation.

In order to achieve our goal we decide to use a max filter. Max filter uses a moving window that gets iterated over the image’s pixels and the pixel value in the center of the window is defined as the maximum of all the pixel values in the window. The following is an example of a max filter with window size 3:

Usually a window size of 3 or 5 is used to remove noise from an image but the window size can change depending on how big of an effect we need the filter to have on the image. The same kind of process can be applied for other operations such as minimum and median.



Figure 13: Max filtering using different window sizes.



Figure 14: Min filtering using different window sizes.



Figure 15: Median filtering using different window sizes.

**4. Expe****riments**

We initialized the DeepLabv3+ neural network using weights pretrained on the PASCAL VOC 2012 semantic segmentation dataset with the xception backbone for 21 classes. We used a collection of 8 images of objects that the DeepLabv3+ network can identify and we extracted the features that the network produces at the 405th layer for each of these images. Next we used PCA for n=8 Principal Components to reduce the dimension of the feature data. W derived with 8 Principal Component features for each image creating a tensor of 8x256x256x8 which we used to fit a Gaussian Mixture Model.

**5. Con****clusions**

Present some conclusions.

**Examples of references**

In [JJQV98], the authors describe an architecture …

/\* ATTENTION: we never say “they describe”, but “the authors describe” \*/

Bernstein et al., [BBC+99] introduce a model for …

The theoretical analysis of the model in [Orr98a] demonstrates that …

**Refer****ences**

|  |  |
| --- | --- |
| [BBC+99] | P.A. Bernstein, T. Bergstraesser, J. Carlson, S. Pal, P. Sanders, D. Shutt. *Microsoft Repository Version 2 and the Open Information Model*. Information Systems 24(2), pp. 71-98, 1999. |
| [BCR94] | V.R. Basili, G.Caldiera, H.D. Rombach. *The Goal Question Metric Approach.* Encyclopedia of Software Engineering, pp. 528-532, John Wiley & Sons, Inc, 1994. Also available at <http://www.cs.umd.edu/users/basili/papers.html> |
| [D97] | E.B. Dean. *Quality Functional Deployment from the Perspective of Competitive Advantage.* Available at <http://mijuno.larc.nasa.gov/dfc/qfd.html> |
| [JJQV98] | M. Jarke, M.A. Jeusfeld, C. Quix, P. Vassiliadis. *Architecture and quality in data warehouses*. In Proc. 10th Conference on Advanced Information Systems Engineering (CAiSE '98), pp. 93-113, Pisa, Italy, June 1998. |
| [JV97] | M. Jarke, Y. Vassiliou. *Foundations of data warehouse quality – a review of the DWQ project*. In Proc. 2nd Intl. Conference Information Quality (IQ-97), pp. 299-313, Cambridge, Mass., USA, June 1997. |
| [O98a] | K. Orr. *Data quality and systems theory*. In Communications of the ACM, 41(2), pp. 54-57, Feb. 1998. |
| [O98b] | K. Orr. *Another paper in the same year*. In Proc International Conference on Computer Science, 41(2), pp. 54-57, 1998. |