Invariant Information Clustering for Unsupervised Image Classification and Segmentation

Kirti Biharie - 4722922, Nishad Tahur - 4681517
{K.S.Biharie, I.H.N.Tahur} @student.tudelft.nl

April 16, 2021

Segmentation

Image segmentation is the process of segmenting the contents of the image based on the objects that are present. The benefit of this segmentation is that the image becomes easier to analyse. A common application where image segmentation is used is in the medical world. Image segmentation can be used to segment different tissues in pictures taken from patients for example. These segmentations can then in turn be used again to render the different tissues separately. With unsupervised image segmentation, this is done by feeding a neural network images such that it learns the characteristics of an image and is able to segment based on the different categories in the content of the image. An example of image segmentation can be seen in figure 1.

Figure 1: On the left there is a giraffe that is drinking water from the ground. On the right three parts of the image are classified. The sky is depicted as blue, plants/vegetation is depicted as green, the sand is depicted as grey and the rest are not classified and remain to be black pixels.

Introduction

In this blog post we will discuss our reproduction of the model used in [1]. In particular, we will try to reproduce the results found in table 4 for the COCO-Stuff-3 dataset. As explained in the paper itself, the COCO-Stuff-3 dataset is the same dataset as the COCO-Stuff dataset but only consists of 3 classes, sky, ground and plant. Images that do not contain either of those three classes are removed from the dataset. Another prerequisite of the image is that it contains at least 75% “stuff pixels”. ”Stuff pixels” can be pixels representing anything in the wide range of buildings to bodies of water. The last criterion is that the image can be shrunk by 2/3 such that later on a crop of 128 by 128 pixels can still be performed.

Method

The method used in the paper is a novel clustering objective. The method is called Invariant Information Clustering (IIC). It is a generic clustering algorithm that makes use of unlabeled images to train a randomly initialised neural network. First we will explain how IIC is used for clustering and afterwards how it is applied for image segmentation.

As input, IIC takes unlabeled training images. The goal is to assign class probabilities to every training image. For training, an image pair is generated for every image by applying a random transformation. The random transformations consist of color jitter, flips, affine transformations and random crops. Both images are fed to the network and are assigned

class probabilites. The objective is to assign the same class probabilities to two images from the same image pair. If the network is capable of doing so, then it means the network is invariant to the random transformations.

For image segmentation, we look at individual pixels instead of the full image. Every single pixel is assigned class probabilities. The number of classes is a hyperparameter, which is 3 classes for the COCO-Stuff-3 dataset. Again, image pairs are generated for training. The image pairs are generated in the same manner as for clustering, so a transformation is applied to the whole image. However, the objective now becomes to assign the same class probabilities to corresponding pixels instead of corresponding images. The loss is described as maximizing mutual information between the corresponding patches and can be calculated as:

$$max(\frac{1}{n|G||\Omega|}\sum_{i=1}^{n}\sum_{g\in G}\sum_{u\in\Omega}\Phi_u(x_i)\cdot[g^{-1}\Phi(gx_i)]_{u+t}^{T})$$

n denotes the number of images. G describes the perturbations in the image. Since we only consider image pairs containing the original image and a perturbation, $|G| = 1$. Ω describes all image patches in the image. $\Phi(x)$ calculates the class probabilities for input image x. gx denotes the perturbation of x, while g^{-1} describes the inverse perturbation.

If an image x is perturbed by flipping which results into gx, a pixel x_{ij} does not correspond to the pixel $(gx)_{ij}$. To find the corresponding pixel, an inverse perturbation, $g^{-1}x$, is used. For flipping, that is simply flipping the image back. Thus x_{ij} corresponds to $(g^{-1}gx)_{ij}$ and we can calculate the mutual information by comparing $(\Phi(x))_{ij}$ and $(g^{-1}\Phi(gx))_{ij}$.

The architecture can be seen in figure 2. It consists of two convolutional blocks, followed by a max pooling, followed by 4 convolutional blocks. After that there are two heads which decide the number of output classes. The first head outputs 3 classes representing the sky, ground and plants. To increase the final accuracy, overclustering is used. The second head outputs 15 classes instead of just 3. The idea is that if the network works better with 15 classes, it will also work better with 3 classes. Every epoch, both heads are trained with all of the data.

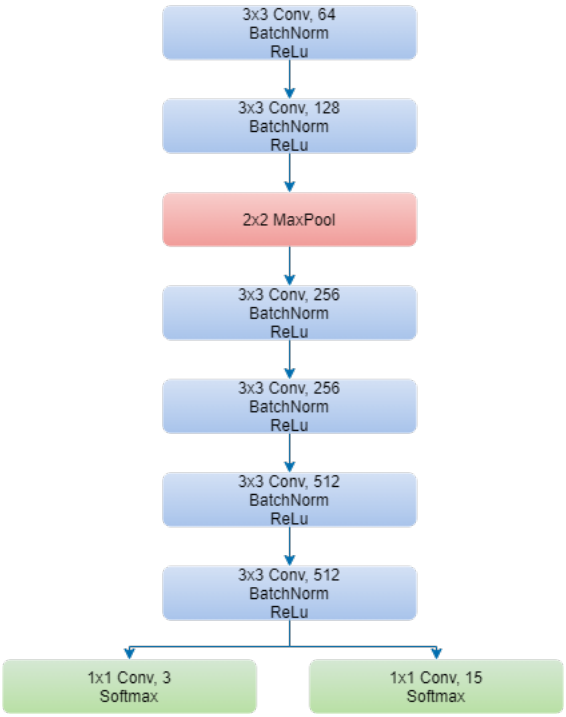

Figure 2: Architecture of the neural network

Reproduction

The resulting accuracy IIC achieved on COCO-Stuff-3 is 72.3%. The previous highest state of the art score was 54%. IIC thus managed to get a significant increase in the accuracy score, a difference of 18.3%. We tried to reproduce this result. For the dataset,

we performed the exact same operations as the authors to filter the dataset. We ended up with $\sim$ 40 thousand images while the authors had $\sim$ 37 thousand images. There are several explanations which include either a different way of calculating how many "stuff pixel" remain or the criterion that it can be shrunk by 2/3. There was, however, no further explanation in the paper on how to exactly filter the COCO-Stuff dataset. Since the provided code also did not provide much input (very unreadable), we decided to continue with $\sim$ 40 thousand images since a difference of $\sim$ 3 thousand images is not that much on such a scale.

Once we acquired the data, we continued by reproducing the model. This model was straightforward and can be seen in figure 2. There was not much trouble setting up the same model and we did that fairly quickly. The training loop consisted of taking a batch of image pairs. An image pair consisted of an original image and a transformed image. These transformations were horizontal flipping and random colour changes in hue, saturation and brightness. We could not, however, find in the paper how exactly to execute these transformations, i.e. at what percentage to do a horizontal flip. Since we wanted to reproduce the results as accurately as possible, we looked into the code and used the same numbers for the transformations. We also used the same loss function as the authors and the backwards step from PyTorch [2] was used.

For the training, we were not able to train it for 4800 epochs due to time constraints. Instead, we trained it for 24 hours. In those 24 hours, we were able to train for 34 epochs. The model was trained on Google Cloud Platform with one GPU, a NVIDIA Tesla T4.

Results and Discussion

The final accuracy we were able to achieve was 61%. The authors gave the images in figure 3 as results.

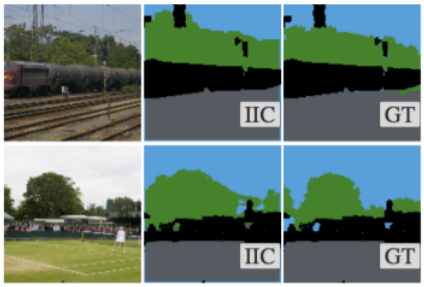

Figure 3: Results shown in the paper (original image, output of their model, ground truth).

When given just the middle column of images in figure 3, we can easily see and understand what the different classes are. Blue is the sky, grey is the ground and green is the vegetation (black pixels are everything else). Given that the accuracy is "only" 72.3%, it is hard to believe that all results look as good as these. Figure 4 shows some really good results from our model.

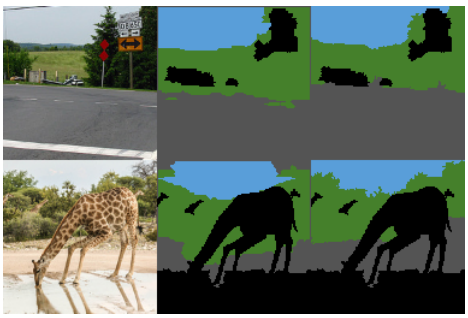

Figure 4: Good results from our model (original image, output of our model, ground truth).

We can see that our model almost perfectly separates the sky, ground and plants just like in the paper. We can also easily understand the categories of each color in the second column of images in figure 4. When comparing it to the ground truth in the third column, they look fairly similar. However, as one might suspect, it is impossible to have these results

but have an accuracy of merely 61%. That is correct, figure 5 shows some really bad results from the same model.

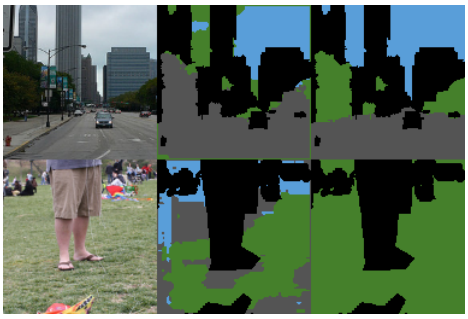

Figure 5: Bad results from our model (original image, output of our model, ground truth).

In the first row of figure 5, the original image contains all three categories, sky, ground and plants. We would also expect the classification to give a clear picture of the segmentations. However, this is not the case. The ground and plants seem to have been taken together and the whole part has been classified as ground, while the sky is classified as both sky and plants. In the second row we just see the legs of a person and the ground where it stands on. The second column should have thus been all grey (with the exception of the legs). Instead, the model predicted that the ground was a combination of sky, ground and plants. The third column contains the ground truth of the images.

There are several reason why there is a difference in accuracy. One of the reason is the difference in our network. We did not train with overclustering for the 24 hours due to time constraints. We did implement overclustering in the end, but only trained the model for two epochs. The first epoch had an accuracy of 62%, while the second epoch resulted in an accuracy of 56%. After training for one epoch, it is thus already better than training without overclustering for 30+ epochs. However, the combination of two heads and more output classes causes overclustering to take at least three times longer per epoch.

Despite this difference in accuracy, we believe that the reproduction itself was a success. We do believe that we would not be able to reproduce the experiment without help of the code. There were many aspects which were not discussed in the paper or the supplementary material, which include preprocessing steps, filtering the data, network architecture, etc. Many of these aspects were only discussed briefly and the exact details on how it should be set up were missing.

The last point of discussion for us is the dataset. The COCO-Stuff-3 dataset was rich of pictures, our filtered set contained $\sim$ 40 thousand images which is a lot. However, the dataset does not fit the segmentation goal in our opinion. We scanned through many of the images and we noticed two things. The first is that many of the images had a low percentage of pixels which were actually sky, ground or plant. With low we mean that approximately 0-15% of the pixels would be classified as either of those three classes. The second is that many of the images only contain either one or two of the classes. For those images, we noticed that the image is still separated in three classes which do not have a link to the semantic classes. For example, the bottom image of figure 5 only contains grass. Our model still segments it into three different classes. We suspect that if most of the images contained all three ground truth classes, the model would be better at segmenting for those three classes.

All code can be found at https://github.com/kbiharie/Deep-Learning-IIC.

Future Work

One of the points future work could work upon is where the same model is trained on a different dataset. A dataset that is specifically tuned such that it contains 3 different classes. Additionaly it could contain masks for all other classes in the images. Most of the images in the dataset should then contain 3 of the classes.

The second point is mainly focused towards reproducing this paper and the results. That is to train the model together with overclustering. We got good results when training the model with overclustering for one epoch so it would be interesting to train the model fully with overclustering. Training it for 4800 epochs seems over the top but training it until

the loss function plateaus seems sufficient. Another interesting point to see whether the accuracy improves is to try to learn the model with a different base. Currently the model's architecture is shown in figure 2, but there could take some research place on whether that model is optimal or not.

References

[1] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019.

[2] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.