COMP4106: Flower Segmentation using Convolutional Neural Networks

Timothy Murphy
University of Nottingham - Computer Science
psxtm6@nottingham.ac.uk

***Abstract*—This paper presents a dual approach to the task of flower segmentation: implementing a pre-trained existing U-Net model, and designing a custom convolutional neural network (CNN) from scratch. Flower segmentation is a challenging problem in the field of computer vision, with applications in botanical research. The study uses the 17 class variation of the Oxford Flower Dataset, consisting of 1360 images of flowers from 17 classes, with 846 corresponding segmentation labels. The pre-trained U-Net model is fine-tuned using transfer leraning, while the custom model is designed to closely resemble the U-Net architecture. The models are then evaluated using accuracy, mean Intersection over Union (IoU), and mean Boundary F1 score metrics. The results show the effectiveness of both approaches, with the custom model outperforming the other in most aspects. This study highlights the potential of CNNs to flower segmentation and similar tasks, and suggests a well-designed CNN tailored to the dataset can outperform a pre-trained existing model.**

***Index Terms*—Flower segmentation, Convolutional Neural Networks, U-Net, Oxford Flower Dataset**

I. Introduction

Image segmentation is the task of separating the pixels of an image into segments, each of which correspond to a specific object or region of the image [1], for example, separating the foreground from the background or, in this case, separating a flower from its background. This task is important in the field of computer vision due to the multitude of applications it can be used for, such as autonomous driving, object detection and tracking, understanding scenes, and so on [2]. However, this task comes with a variety of challenges, involving complicated backgrounds, differing colours, textures, and shapes, and variations in lighting [3].

The emergence of deep learning and Convolutional Neural Networks (CNNs) has rapidly improved progress in this area of research, outperforming traditional methods in accuracy and, in some cases, computational efficiency [4]. CNNs are typically composed of layers, for example, convolutional layers that use filters to capture features, Rectified Linear Unit (ReLU) layers to introduce non-linearity, and pooling layers to reduce the spatial dimensions. These help CNNs achieve cutting-edge performance in segmentation tasks [5].

Flower segmentation is a much narrower area of the general problem of image segmentation. The goal here is to correctly separate the pixels of an image that belong to a flower from the pixels of the background, producing two distinct segments or regions. This is important for applications such as botanical research, monitoring agriculture, and has even been implemented into smartphone apps for flower recognition [6].

This paper focuses on the problem of flower segmentation, using a variation of the Oxford Flower Dataset [7]. Two CNN-based approaches will be explored: (1) fine-tuning a pre-trained U-Net model [8], and (2) developing a custom CNN architecture specifically designed for this task.

The rest of the paper is organised as follows. Section II describes the methodology, which includes the dataset preparation, the existing U-Net model, and the custom CNN architecture. Section III presents the evaluation results and Section IV presents a discussion of the findings alongside directions for future work. Finally, Section V concludes the paper.

II. Method

A. Dataset Preparation

The dataset used in this task is the 17 category version of the Oxford Flower Dataset [7]. This dataset contains 80 images for each class of flower, all of which are common British flowers, resulting in 1360 images. The dataset contains 846 segmentation ground truth labels, so there is an inconsistency between the number of flower images and corresponding segmentation maps. To address this issue, the intersect function in MATLAB was used to find all matching image and label file pairs, resulting in 846 images-label pairs. This ensured that the dataset only contained complete data.

Minimal pre-processing was applied to the images and labels in the dataset. All images were of the same size (256x256 pixels). Normalisation and data augmentation, such as random rotation, scaling, and flipping, were not applied.

The matched image-label pairs were split into training, validation, and test sets using the ratio of 80/10/10, respectively. This ratio was chosen to maximise the training data for the model given that the dataset is relatively small. This split was performed before converting the dataset into datastores to prevent difficulties in splitting the data once it had been converted to a datastore format.

Datastores in MATLAB were chosen due to being an efficient way to manage and use large datasets, such as collections of images and their corresponding labels, without loading all the data into memory. This task required imageDatastore (IMDS) and pixelLabelDatastore (PXDS) datastore types for handling flower images and segmentation ground truth labels respectively. Therefore, the indices of the image-label pairs were shuffled, then split into the three sets, then an IMDS and PXDS were created for each set. The IMDS and PXDS for each set were then combined to create a single datastore for training, validation, and test data.

B. Existing CNN Model: U-Net

Choosing an appropriate pre-trained model to fine-tune on the Oxford Flower Dataset required careful consideration. Several CNN architectures, such as ResNet [9], SegNet [10], and RCNN [11] have been able to successfully perform image classification and segmentation, and would be valid candidates for this task in particular. After evaluating multiple options, U-Net [8] was chosen.

U-Net was originally developed by Ronneberger et al. for biomedical image segmentation and it has achieved impressive performance on various medical image segmentation tasks. The architecture consists of an encoder-decoder structure with skip connections, allowing it to capture both high-level contextual information and low-level spatial details. The encoder consists of convolutional layers for feature extraction and downsampling, and the decoder is responsible for upsampling and precise localisation. The skip connections concatenate features from encoder to decoder at each corresponding level. This makes U-Net a suitable choice for precise segmentation of objects with irregular shapes and sizes, such as flowers.

Furthermore, U-Net has been shown to perform strongly in tasks with limited training data [12], which is also relevant to this task, as the Oxford Flower Dataset contains less than 1000 usable images.

The final reason for choosing U-Net was its successful application to plant phenotyping [13] and seedling root segmentation [14]. These examples show how U-Net can effectively segment plant structures and handle variations in appearance and background, all of which are relevant to the flower segmentation task.

For this task, U-Net was implemented in MATLAB, using the MATLAB Deep Learning Toolbox. To adapt U-Net for flower segmentation, the encoder depth was set to 2. This choice was made due to the need to find a balance between model complexity and the risk of overfitting, given the limited size of the training dataset. A deeper encoder network would have increased the model's capacity to learn complex features, however, this increased complexity also makes the model more likely to overfit. Overfitting occurs when a model learns to fit the training data too closely, capturing noise and specific patterns that do not generalise well to unseen data [15]. Reducing the encoder depth to 2 means that the number of learnable parameters in the model is reduced, but the model is still able to capture important features for segmentation.

C. Custom CNN Model

Designing a custom CNN architecture from scratch would remove the need for transfer learning, as the model would not have been trained on any data previously. This eliminates disadvantages specific to transfer learning, such as feature misalignment, but also means that the advantages of faster convergence, less data required, and efficiency are eliminated too. Despite this, building a CNN from scratch allows for complete customisation of the architecture to suit the specific features of the Oxford Flower Dataset, although at the risk of poor convergence and the increased need for more data and computational efficiency.

In an attempt to find a balance between the advantages of the U-Net architecture and the disadvantages of using a pre-trained U-Net model, the custom CNN architecture was designed to resemble the U-Net architecture. This was achieved in MATLAB, also using the Deep Learning Toolbox.

The architecture mainly consisted of an encoder path and decoder path. The encoder path's purpose was to downsample the feature maps and extract hierarchical features from the images. This included convolutional layers with increasing number of filters, starting with 64, then 128, 256, and finally 512. ReLU activation functions were included after each convolutional layer. Max pooling layers with a pool size of 2 and a stride of 2 were included for downsampling. Finally, a dropout layer with a rate of 0.5 was included at the end of the encoder path to prevent overfitting.

The decoder path's purpose was to upsample the feature maps and recover spatial resolution. The layers for this included transposed convolutional layers for upsampling, with decreasing number of filters, starting with 512, then 256, and finally 128. Convolutional layers were also included with corresponding number of filters (256, 128, 64). ReLU activation functions were included after each convolutional layer, same as in the encoder path.

The final layers consisted of a 1x1 convolutional layer with two filters to equal the number of classes, a softmax activation function for per-pixel classification, and a pixel classification layer for outputting the segmentation mask.

This symmetric encoder-decoder structure was inspired by U-Net architecture, where the increasing number of filters in the encoder path capture more complex features, and the decreasing number of filters in the decoder path recover spatial resolution. It was decided that skip connections and concatenation operations, as commonly seen in U-Net and similar architectures, would not be implemented in this CNN. This decision was made in an attempt to prevent the model architecture from becoming too complex, therefore reducing the risk of overfitting and resulting in better generalisation. Furthermore, it was found that this simpler design yielded better results than a more complicated design. This aligns with a recent study that found skip connections to show little improvements in low and medium complexity tasks [16].

D. Training Options

The training options for both the existing U-Net model and the custom CNN model were carefully selected to find a balance between performance and efficiency, taking into account the limited computational resources available. Training options for both tasks were kept the same.

The Adam optimiser [17] was chosen over other optimisers due to its adaptive learning rate and ability to handle sparse gradients effectively. The Adam optimiser combines the benefits of AdaGrad [18], which adapts the learning rate for each parameter based on its historical gradients, and RSMProp [19], which uses a moving average of squared gradients to normalise the learning rate. This makes Adam an appropriate choice for tasks with noisy or sparse gradients, such as image segmentation.

The initial learning rate was set to 0.0001, which is a good balance of convergence speed and stability. The maximum number of epochs was set to 5, considering the limited computational resources and small dataset. The mini-batch size was set to 64, which provided a trade-off between memory and training speed. To reduce overfitting, the traning data was shuffled at every epoch. This helps the model learn more robust features and prevents it from memorising the order of training examples [20]. The validation data was used to evaluate the model's performance during training, validating every 5 iterations. Early stopping with a patience of 3 iterations was used to prevent overfitting and conserve resources.

III. Evaluation

To evaluate both models, two main metrics were used: accuarcy and mean IoU. Accuracy measures the ratio of correctly classified pixels to the total number of pixels, and mean IoU measures the overlap between the predicted and ground truth segmentations, taking into account both correct predictions and their spatial relevance. Mean IoU is considered a more robust metric. Mean Boundary F1 scores were also used for per-class evaluation, which measure the accuracy of predicted boundaries compared to ground truth boundaries.

A. Existing U-Net Model

The pre-trained U-Net model achieved an overall accuracy of 0.838, meaning that the model correctly predicts 83.8% of all pixels. The overall mean IoU was 0.718, which shows a strong overlap between the predicted segmentation and the true segmentation for both classes. This tells us that the model can effectively locate the relevant areas in the task.

For the flower class, the model achieved an accuracy of 0.799 and a mean IoU of 0.638. These are lower than the overall values, suggesting that the model can capture the general area of the flower but might struggle with precise delineation. This is supported by the low mean Boundary F1 score of 0.359, showing poor performance in precisely marking flower boundaries.

For the background class, the model achieved an accuracy of 0.877 and a mean IoU of 0.799. This shows that the model performs better at identifying background pixels, suggesting less variability in the backgrounds. The mean Boundary F1 score was slightly higher at 0.518, although it is still moderately low, suggesting difficulties with accurately outlining the background boundaries.

B. Custom CNN Model

The custom CNN model achieved an overall accuracy of 0.911, showing that the model correctly classified 91.1% of pixels. The overall mean IoU was 0.818, showing a very strong overlap between predicted and true segmentation masks for both classes.

For the flower class, the model achieved an accuracy of 0.916, which is slightly higher than the overall accuracy. This suggests that the model is very effective at identifying flower pixels. The mean IoU for the flower class was 0.767, which is lower than the overall value but is still relatively high. The mean Boundary F1 score was 0.72, meaning that the model is effective at accurately outlining the flower boundaries.

For the background class, it achieved an accuracy of 0.905, showing that the model can very accurately distinguish background areas. The mean IoU was 0.87, which is higher than the overall value. The mean Boundary F1 score for this class was 0.842, showing that the model can accurately outline the background.

C. IoU Histograms and Confusion Matrices

Figure 3 shows the histogram of IoU values for both models. For the pre-trained model, we can see that majority of values are above 0.5 - the typical threshold for satisfactory performance [21]. The peak in the range of 0.7 to 0.9 suggests a good level of performance, but that there is still room for improvement. For the custom model, we can see a significant peak of IoU values between 0.9 to 1.0, suggesting that this model achieves close to perfect segmentation a lot of the time. Both models have a presence of lower IoU scores, below 0.5, show that there are still instances where the models do not perform well.

Figures 4 and 5 show the confusion matrices with column and row normalisation for the pre-trained and custom CNN models respectively. The U-Net model correctly predicts 90.0% of background pixels and 75.9% of flower pixels (column-normalised), and correctly identifies 87.7% of background pixels and 79.9% of flower pixels (row-normalised). The custom model outperforms the U-Net model, correctly predicting 95.7% of background pixels and 82.5% of flower pixels (column-normalised), and correctly identifying 90.5% of background pixels and 91.6% of flower pixels (row-normalised). This suggests the custom model is better at both predicting and identifying pixels for both classes.

IV. Discussion

The main advantage of the pre-trained model was that it made use of transfer learning, meaning it converged faster and required less training data. This is particularly relevant to this task as the training data was relatively small and the model no doubt benefited from transfer learning in this regard. However, this does not explain the poor evaluation results when compared to the custom model.

One potential reason for this could be that the existing U-Net model is much more complex than the custom one, with deeper architecture and more parameters, therefore making it much more prone to overfitting. It might be attempting to memorise the training examples. In addition, there might not be enough training examples of each flower class for the model to learn robust and generalisable features. The custom model might be performing better because it only learns the essential patterns, not the fine-grained details.

Furthermore, the U-Net model was trained on a different dataset before, which consisted of medical images, and was then fine-tuned on this flower dataset. This transfer learning approach does not always work, especially if the original task of the model is different from the current one.

Figure 1 shows the predicted mask from the pre-trained model for one example. We can see that the model performs well at defining the flower boundaries, yet the low mean Boundary F1 score constrasts this, perhaps due to how it struggles with the finer details, often labelling specific parts of the flower as background. The predicted mask looks too detailed and reflects the actual image too much, suggesting the model has overfitted.

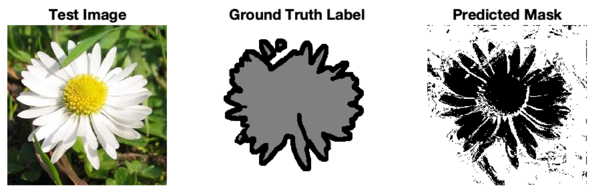

Fig. 1. Pre-trained U-Net model segmentation example.

Compare this with the performance of the custom model on a different flower image in Figure 2. We can see that the predicted mask is much more simple and does not capture fine-grain details like the pre-trained model. Instead, the custom model produces a mask that closer resembles the ground truth. However, this model fails to perfectly capture the small details of the flower boundary, evidenced by its mean Boundary F1 score.

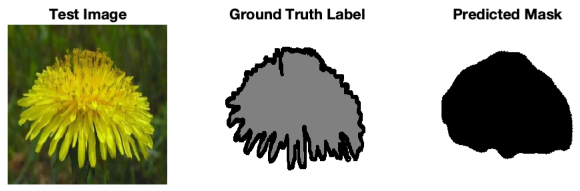

Fig. 2. Custom CNN model segmentation example.

The most significant weaknesses of the custom model were its lack of complexity and the computational resources required. Considering how the custom model outperformed the pre-trained one in terms of overall accuracy, mean IoU, and mean Boundary F1 scores for both classes with far less complex architecture, it suggests that a more complicated model with careful consideration to hyperparameters could yield far more impressive results. For example, skip connections and concatenation could be added to closer resemble U-Net architecture, which may improve performance. This approach was considered, however, due to computational constraints, this was not possible in the scope of the project.

Furthermore, both models might have benefited from additional data augmentation to improve robustness and generalisation to unseen data. This could significantly improve performance of both models by increasing the diversity and variation of the dataset.

Overall, both CNN models have shown to be effective tools for flower segmentation. Transfer learning with a pre-trained model may be a good starting point, but a custom architecture tailored to the specific needs of the task is more likely to perform better. Choosing between pre-trained or custom made depends on factors such as available data, computational resources, and the specific task and desired outcome. The results from this project suggest that designing a custom CNN model that closely resembles an existing pre-trained model can result in a model with combined strengths from both, although a careful balance between complexity, training data, and hyperparameters is necessary.

Future work in this area should explore how data augmentation and normalisation impacts model performance and generalisation, and should investigate how other pre-trained models, such as ResNet or Mask R-CNN, might perform in flower segmentation. It is necessary for further experimentation with parameters and architecture, for example possibly adding skip connections, to reach an optimal balance between complexity and performance. Applying these models to other similar tasks or other plant datasets would allow for assessment on the robustness of the models.

V. Conclusion

To conclude, this project aimed to implement both a pre-trained existing and custom CNN model to the task of flower segmentation, and to investigate the differences between the two approaches. A comparative analysis of both approaches was conducted using accuracy, mean IoU, and mean Boundary F1 score metrics.

Both the pre-trained U-Net model and the custom model demonstrated effectiveness in the segmentation task, with the custom model outperforming the pre-trained model in terms of overall accuracy, mean IoU, and mean Boundary F1 scores. The pre-trained model benefited from transfer learning, resulting in faster convergence, less required training data, and less computational resources, but struggled with precise flower boundary delineation. The custom model, while requiring more computational resources and training time, showed better performance in accurately segmenting both flower and background regions.

This study shows the potential of CNNs for flower segmentation and similar applications. The results suggest that a well-designed custom CNN architecture tailored to the dataset can outperform a pre-trained model, even with limited data and resources.

Appendix

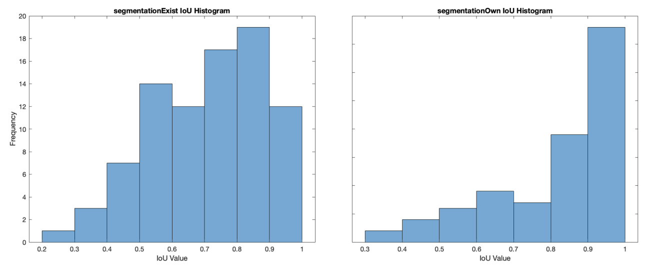

Fig. 3. IoU Histogram for pre-existing (left) and custom (right) models.

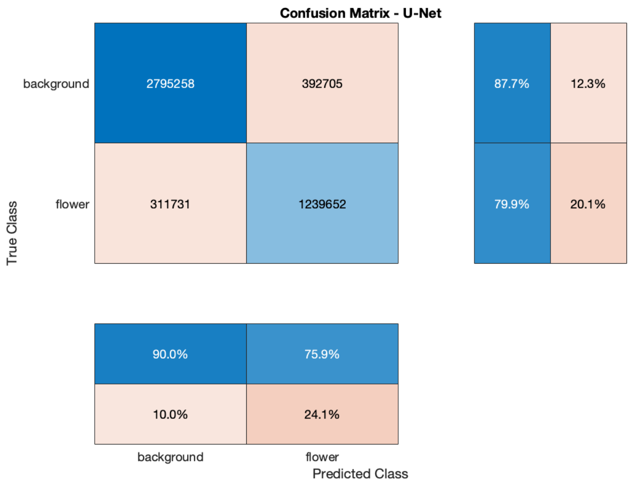

Fig. 4. Confusion Matrix for pre-trained U-Net model.

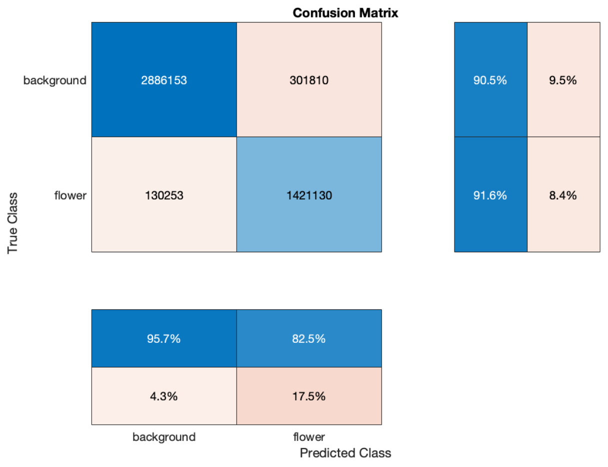

Fig. 5. Confusion Matrix for custom CNN model.

References

[1] S. Abdulateef and M. Salman, "A comprehensive review of image segmentation techniques," *Iraqi Journal for Electrical and Electronic Engineering*, vol. 17, pp. 166–175, 09 2021.

[2] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv (Cornell University)*, 04 2023.

[3] M. A. Hashmani, M. M. Memon, and K. Raza, "Semantic segmentation for visually adverse images – a critical review," *2020 International Conference on Computational Intelligence (ICCI)*, 10 2020.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7298965

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, 05 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[6] I. Gogul and V. S. Kumar, "Flower species recognition system using convolution neural networks and transfer learning," *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, 03 2017.

[7] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," IEEE Xplore, p. 1447–1454, 06 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1640927

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science*, vol. 9351, pp. 234–241, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 06 2016.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, 12 2017.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 06 2014. [Online]. Available: https://dl.acm.org/citation.cfm?id=2679851

[12] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 12 2017.

[13] Y. Li, Y. Huang, M. Wang, and Y. Zhao, "An improved u-net-based in situ root system phenotype segmentation method for plants," *Frontiers in plant science*, vol. 14, 03 2023.

[14] X. Xu, J. Qiu, W. Zhang, Z. Zhou, and Y. Kang, "Soybean seedling root segmentation using improved u-net network," *Sensors*, vol. 22, pp. 8904–8904, 11 2022.

[15] D. M. Hawkins, "The problem of overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, pp. 1–12, 01 2004.

[16] A. Kamath, J. Willmann, N. Andratschke, and M. Reyes, "Do we really need that skip-connection? understanding its interplay with task complexity," *Lecture notes in computer science*, pp. 302–311, 01 2023.

[17] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014. [Online]. Available: https://xueshu.baidu.com/usercenter/paper/show?paperid=37a73866f09edd03830b234716447e4f&site=xueshu_se

[18] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 02 2011.

[19] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6a overview of mini-batch gradient descent," *Coursera: Neural Networks for Machine Learning*, 2012.

[20] A. Koloskova, N. Doikov, S. U. Stich, and M. Jaggi, "Shuffle sgd is always better than sgd: Improved analysis of sgd with arbitrary data orders," *arXiv (Cornell University)*, 05 2023.

[21] J. Hosang, R. Benenson, P. Dollar, and B. Schiele, "What makes for effective detection proposals?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 814–830, 04 2016. [Online]. Available: https://arxiv.org/abs/1502.05082