SegDIP: The Unreasonable Effectiveness of Randomly-Initialized CNNs for Interactive Segmentation

Anagh Malik[1], **Shuaifeng Zhi**[1], **Marwan Taher**[1], **Ronald Clark**[2], **Andrew Davison**[1]

[1]Dyson Robotics Lab at Imperial College London
[2]Department of Computing, Imperial College London
{anagh.malik18, s.zhi17, m.taher, ronald.clark, a.davison}@imperial.ac.uk

Abstract

We present a simple lightweight framework for interactive real time semantic segmentation without prior data based on the Deep Image Prior [21]. For each image we train a single convolutional encoder-decoder network mapping from input coordinates to RGB values and semantic classes. We find the paradigm of simultaneous reconstruction and segmentation to be very powerful, the proposed model can interpolate from just a few user-provided annotations, which we analyze utilizing ideas from the Neural Tangent Kernel literature. Our key contribution is a Temporal Consistency Loss (TCL), which applies a dense loss at each iteration of the training procedure guided by the last one. We implement many types of user interactions such as clicks, bounding boxes and strokes. We further show some promising experiments for real time tracking, by switching between the subsequent RGB frames from a video. With only sparse annotations on the first frame of a video, we are able to track objects through the whole video.

1 Introduction

Semantic segmentation is one of the fundamental problems in the field of computer vision and machine learning. Assigning a label to each pixel of an image is one of the important steps in building complex robotic systems such as driverless cars, human-friendly robots, robot-assisted surgery and intelligent military systems.

The simplest problem definition for semantic segmentation is pixel-wise labelling. However this problem is fundamentally ill-defined, since these labels are non-unique e.g. we could label pixels of a brick in a building as both "brick" and "building". None of these answers are inherently wrong and the preferred answer undoubtedly depends on the application.

This is why typically for semantic segmentation we train deep neural network on large datasets, which pre-define the different types of classes. However, not only is there a high cost to creating these training datasets, but at test time we are limited to the classes we have previously seen. Instead, we explore an approach that allows for high-quality segmentation with minimal human interaction because the user monitors the semantic map as it updates in real-time and clicks only as needed to correct it. The smoothness properties of the CNN mean that regions and objects are coherently represented, and can frequently be labelled with only a few clicks. Sometimes, not even a single click is required as the correct properties will be transferred from already labelled parts of the scene.

Our contributions are as follows:

- We propose an architecture for real-time interactive semantic segmentation without prior data, a setup which has not been explored before.
- We demonstrate the powerful ability of the DIP architecture to predict semantics in an image by interpolating from just a few human provided clicks, without any explicit prior to do so. This is analyzed under the lens of the Neural Tangent Kernel.

- We propose a Temporal Consistency Loss, which takes a big step in helping our setup become a tool for large scale segmentation.
- We present the further capabilities of our setup to perform tracking in a video.

2 Related Work

2.1 Semantic Segmentation

Most standard semantic segmentation systems are based on training on densly labelled datasets without any further interaction during test time, for example U-Net [17] or DeepLabV3 [6]. Such setups firstly require heavy resources for the training phase, but in addition they are inflexible to the addition of new unseen classes or refinement during test time.

There have been methods aiming to alleviate this issue. ScribbleSup [13] proposes training a network on a dataset labelled only with scribbles. However this method again suffers from the issue of being inflexible to the addition of unseen classes at test time, due to the reliance on a training dataset. This is an issue which is seemingly overcome by both Polygon-RNN [5] and Polygon-RNN++ [1], which are based on recurrent architectures with a human-in-the-loop who can adjust the segmentation at each step. However this method is again based on expensive pretraining on a dataset. We argue that one should be able to do real time semantic segmentation without extensive pretraining just by analyzing the self similarity in a single image.

2.2 Single Image Based Learning

Recently, several deep internal learning methods have been proposed and achieve remarkable performance that is comparable to that of external methods trained on large-scale datasets. Methods have been developed to solve superresolution [19] [21] [4], restoration [19] [14] reflection removal [22] or deblurring [16].

All these methods are predated by classical internal learning methods, which have proved internal approaches can be successfully applied to several image manipulation tasks such as super-resolution [9] [10], dehazing [3] or texture synthesis [7].

2.3 Single Image Based Segmentation

There has also been some work proposed for segmentation. Double-DIP [8] solves the issue of foreground/background segmentation using two Deep Image Prior architectures and a saliency initialization, we qualitatively compare against this work. However it's important to mention that despite requiring some interaction our setup allows for flexible multi class segmentation as opposed to being limited to two fixed classes. Furthermore we require no saliency initialization either.

There have also been classical internal methods for segmentation. Bagon et al. [2] proposed an information theory method to define a good image segment and based on that segment an image. GrabCut [18] proposed a foreground/background segmentation method based on graph cuts by using a user provided bounding box, specifying the region the foreground is in. We again use this work as a qualitative comparison baseline, however we again stress our setup allows for flexible multi class segmentation as opposed to being limited to two fixed classes.

3 Method

3.1 Deep Image Prior

Let's first revisit Deep Image Prior. As a general framework the paper treats a neural network f_θ with parameters θ as a parametrization $x = f_\theta(z)$ by mapping a fixed code $z \in \mathbb{R}^{H \times W \times L}$ to the image $x \in \mathbb{R}^{H \times W \times 3}$, where H and W are the height and width of the image.

With this general idea they are able to solve tasks such as denoising by taking the output of their network after utilizing early stopping to avoid overfitting and minimizing the objective:

$$\min_\theta ||x^* - x||^2$$

where $x^* = f_\theta$. A similar setup is used for inpainting, by applying a binary mask $M \in \{0,1\}^{H \times W}$ representing the filled region and solving the objective:

$$\min_\theta ||x^* \times M - x \times M||^2$$

where $x^* = f_\theta$. The output of the network often predicts coherent textures on masked out areas.

The function approximation f in both these cases is taken to be a convolutional neural network with the parameters θ representing the weights and bias in the filers of the network. The code z varies across tasks, but they are either meshgrid i.e. a grid of xy coordinates or random noise. For meshgrid we have $L = 2$, whereas for noise L can be arbitrary.

3.2 Basic Setup

We augment this setup by first assuming we have sparse semantic supervision for the image. More specifically we have both a mask $M \in \{0,1\}^{H\times W}$, which masks out the pixels that don't have a ground truth annotation and a target segmentation $s \in \{0,1\}^{H\times W\times N}$, which contains the ground truth labels for the appropriate pixels contained in the mask M, where N represents the maximum number of classes we want to segment the image into. We then make the network map from the code $z \in \mathbb{R}^{H\times W\times L}$ to both the image $x \in \mathbb{R}^{H\times W\times 3}$ and the segmentation $s \in \mathbb{R}^{H\times W\times N}$ only for the annotated pixels according to the mask M i.e. we minimize the objective:

$$\min_{\theta} ||x^* - x||^2 + ||s^* \times M - s \times M||^2$$

where $x^*, s^* = f_\theta(z)$. This setup can be used for interpolating a few pre-defined annotations or for label denoising. However we further augment the system by real-time training and supplying clicks in the loop. In this case our objective at training timestep t is:

$$\min_{\theta} ||x^* - x||^2 + ||s^* \times M_t - s_t \times M_t||^2$$

where s_t represents the target with the ground-truth annotations supplied till timestep t and M_t is the binary mask containing only those pixels.

With this setup we can perform real-time semantic segmentation. We, however, find that this setup can sometimes degrade the segmentation of different areas with the addition of more annotations. This is why we introduce a novel Temporal Consistency Loss.

3.3 Temporal Consistency Loss

Let $Q = \{q_m\}_m$ be the set of pixels in the image given labels $\{y_m\}_m$ (probability vectors) respectively through an interaction (click, bounding box or stroke). Let S^k be the segmentation output of the network on the k-th training iteration of dimensions $H \times W \times N$, where H is the height of the image, W is the width of the image and N is the number of classes (each S^k_{ijl} is a probability). We define an augmented $H \times W \times N$ dimension matrix T^k as:

$$T^k_{ij} = \begin{cases} y_m, & \text{if } (i,j) = q_m \\ S^k_{ij}, & \text{if } (i,j) \notin Q \end{cases}$$

This is the current model predictions combined with the ground truth provided by the interactions. Furthermore let M^{k+1} be a matrix of dimensions $H \times W$, which has values between 0 and 1 of per pixel weighting.

Then at iteration (k+1), we define:

$$L_{k+1} = \sum_{i,j} \text{BCE}(S^{k+1}_{ij}, T^k_{ij}) * M^{k+1}_{ij}$$

where BCE is the binary cross entropy loss. Where we set:

$${M^{k+1}}_{ij} = \begin{cases} 1, & \text{if } (i,j) \in Q \\ \lambda, & \text{if } (i,j) \notin Q \end{cases}$$

where $\lambda << 1$. We then at iteration $(k+1)$ jointly minimize:

$$\min_{\theta} ||x^* - x||^2 + L_{k+1}$$

only if there are some annotations in the image, otherwise, we only minimize the reconstruction loss, given by:

$$\min_{\theta} ||x^* - x||^2$$

With this setup we can segment many difficult images.

3.4 Tracking

Let $\{x_t\}_t$ be a set of frames from a video, where $x_t \in \mathbb{R}^{H\times W\times 3}$. For tracking in a video on the first frame we train the exact same setup as with the Temporal Consistency Loss, allowing a user to annotate the frame in real time, i.e. if there are annotations in the frame we are minimizing:

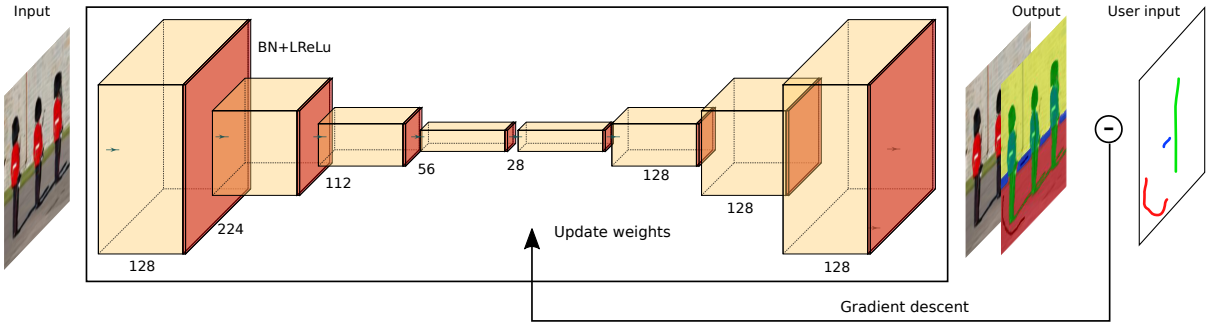

Figure 1: SegDIP architecture.

$$\min_{\theta} ||x^* - x_1||^2 + L_{k+1}$$

However now we allow the user to switch the frame of the video to the next one, retraining the reconstruction loss towards the new image target. In this way if the user switched the loss towards the frame t before timestep $(k+1)$ then we are minimizing:

$$\min_{\theta} ||x^* - x_0||^2 + L_{k+1}$$

however it's important to note that we reinstantiate the aforementioned annotations Q for each frame, hence we have:

$$L_{k+1} = \sum_{i,j} \text{BCE}(S_{ij}^{k+1}, T_{ij}^{k}) * M_{ij}^{k+1}$$

where ${M^{k+1}}_{ij} = \lambda$. This forces also a temporal consistency across different frames on the video. We note that this setup let's us track the objects we annotated only in the first frame across the whole video.

3.5 Implementation

We train the image reconstruction network for each image from scratch. In all our experiments we use an encoder-decoder network. The encoder has 4 layers, then we have a 3 layer decoder with 2 convolutional heads on top of that, one of which maps to the image and the other to the segmentation. Each convolutional layer has 128 filters and LeakyRELU activations. Between each of these layers we perform Batch Normalization [11], which we find to speed up the training process. For the input code z we use meshgrid i.e. xy coordinates, we find that this applies a smooth prior on the segmentation. Replacing meshgrid with random noise increases the capacity of the network to the point that it overfits particular annotations. To allow for a smooth training process and avoid jittering with the addition of new clicks we clip the gradients. The whole setup is trained with AMSGrad [15]. We downsample through strides, but upsample with nearest-neighbour method.

For the setup with temporal consistency we use $\lambda = 0.001$. We train the network real-time with a Nvidia GeForce RTX 3080 GPU, which takes around 1 min depending on the image. We usually downscale images to have both dimensions in the size to be less than 300 pixels.

3.6 Training and annotating procedure

Before starting to annotate we usually wait for the image reconstruction to converge to the point that various objects in the image are recognizable. to start annotating a user has to first specify the class the annotation will be from, then he can use one of 3 types of annotations:

- Clicks - with which one can supervise one pixel
- Stroke - with which a user can more easily supervise pixels on one single line
- Bounding Box - a user can draw a bounding dox, with which he supervises all the pixels inside it

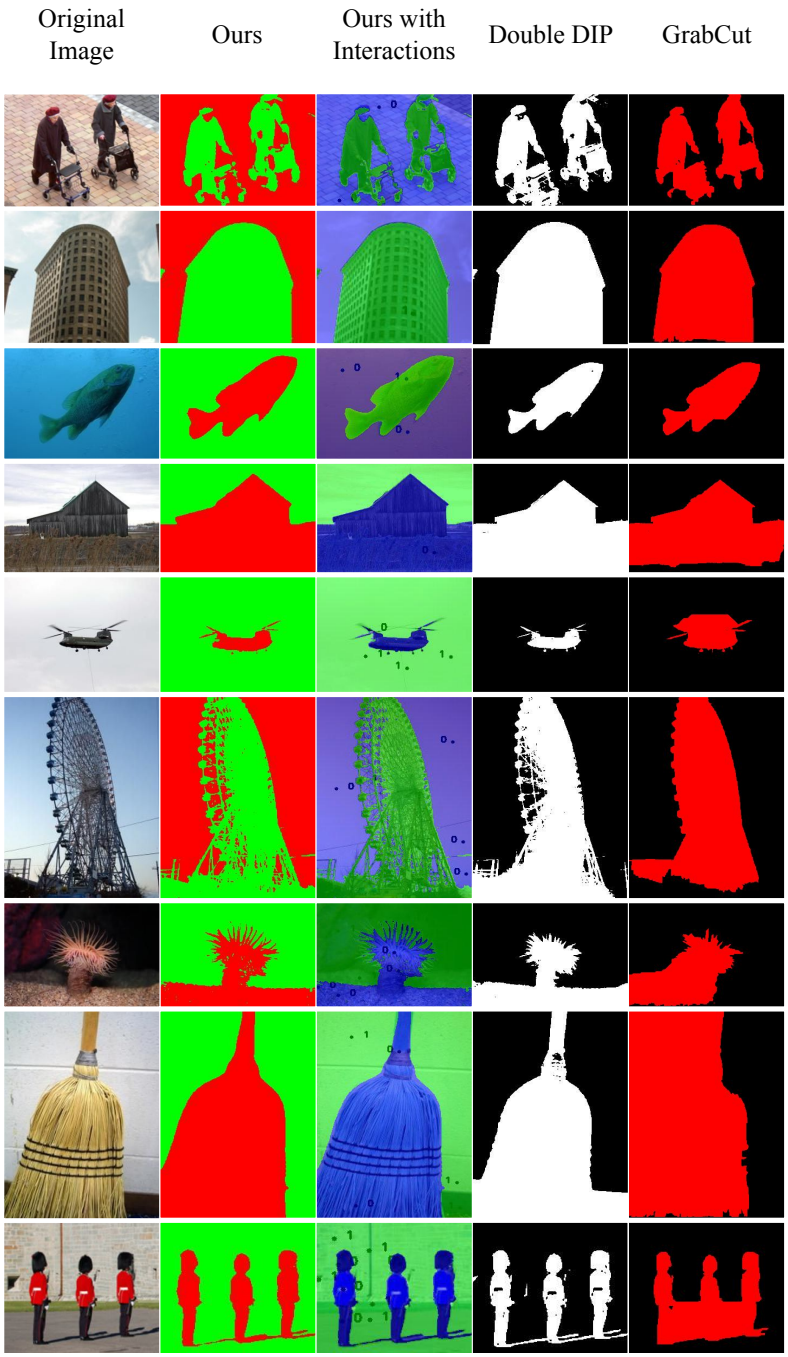

Figure 2: Comparison against Double-DIP [8] and Grabcut [18]. System without temporal consistency and with only click annotations. The number of clicks needed for us for the images is a) 2, b) 1, c) 3, d) 1, e) 4, f) 5, g) 6, h) 4, i) 7.

4 Results

4.1 Binary and Multi-Class Segmentation Without Temporal Consistency

We can first perform experiments without temporal consistency loss for 2 class segmentation on simple images with just clicking annotations (each click contains a single pixel). Even though this is not what we think is the key advantage of our system, it shows how well our architecture can interpolate from extremely sparse annotations. In Figure 2 we can see examples of this, where for each image we use at most 8 clicks, but still manage to produce a reasonable segmentation.

We can further see that on the qualitative examples our system outperforms GrabCut. We presume this is because of the very high sensitivity of GrabCut to the bounding box, for images with large foreground areas (with objects of interests) we see the performance is significantly worse.

Moreover we can see comparable results to Double-DIP despite not using any explicit priors for segmentation such as saliency. However we also exceed their performance on images with unclear foreground/background segmentation like the broom or on high frequency images such as the ferris wheel. This is due to the flexibility of providing supervision in our system and because the segmentation is being performed from the same representation as the reconstruction, thus our system is able to see smaller details in images and recognize tiny patches.

Whats especially sets our system apart is the capability to perform multi class segmentation. This is also something both Double-DIP and Grabcut aren't able to perform. Results for this setup can be seen in Figure 3, where we perform multi-class segmentation from just a few user-provided clicks. We can again notice that for each of these image we need less than 5 clicks per class to produce reasonable segmentation.

However the images we have shown so far are limited to simple or smooth textures. We have found for images with more complicated textures the segmentation quality sometimes degrades over-time, when a previously correctly predicted pixel is attributed the wrong class after the addition of more annotations. This in turns means we need a lot more annotations to keep correcting previously correctly labelled pixels. This effect can also be seen in Figure 4. Where despite there being multiple annotations on the treeline the network gets confused by the highly textured leafs. As we can see on Figure 4 the Temporal Consistency Loss mitigates this issue.

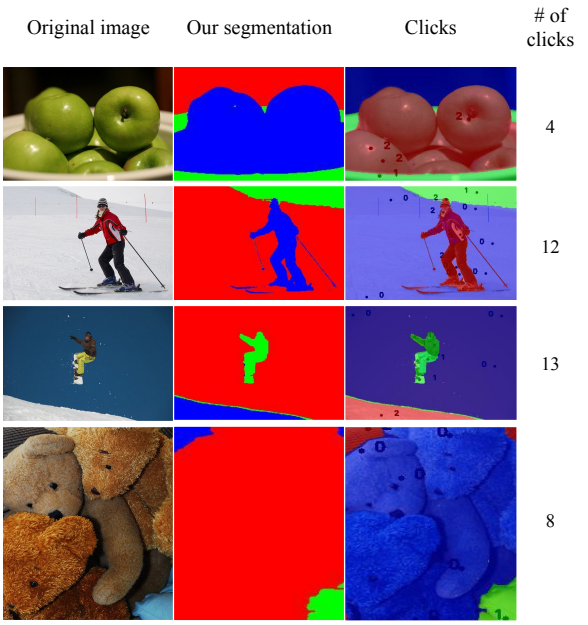

Figure 3: Segmentation from SegDIP on simple multi class images. All segmentation done in real-time with only clicking annotations, where click size is one pixel.

4.2 Multi-Class Segmentation With Temporal Consistency

We can see further results on Figure 3, where we can segment complicated images with as little as 18 human provided strokes. However even more important is that for all but one of these images we are supervising less than 1% of the pixels in the image. Especially impressive are the results for highly textured images as the image with the car or the image with the elephant. Thus we can reach the conclusion that our system is very close to being a very efficient labelling tool. With this performance one could imagine the application of our setup for dataset creation. However it's also important to mention that the TCL suppresses label propagation slightly, hence why for simple images

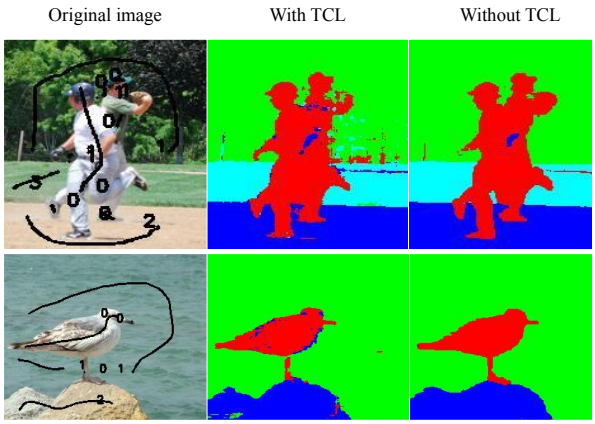

Figure 4: SegDIP for 3 class segmentation with and without the Temporal Consistency Loss (TCL). The number of user annotations for these images is a) 13, b) 6. Both these images were not segmented interactively, but run with pre-recorded clicks.

we might need more annotations than without it, however this issue is negligible in many real-world segmentation tasks.

4.3 Tracking

We can also see perform video tracking experiments on two simple scenes. We switch the frames every some iterations (as described in the Methods section). For each video we only provide annotations in the first frame. From Figure 6 we can see that SegDIP performs well on these simple tracking problems, it doesn't get pixel perfect results, however it is able to generally track the shape as it moves through the scene.

Our setup also allows fixing errors during the training in further frames, however we concentrate on only showing tracking abilities in these experiments.

5 Analysis

One can ask, why does this even work? We can further analyze the reason for the performance of the model, by studying the kernel function imposed by the network itself. In this section, we thus consider only our network trained with a reconstruction loss, without any interaction. We can thus analyze what

Figure 5: SegDIP with Temporal Consistency Loss for multi-class segmentation trained in real-time with a user. Provided are the number of scribbles needed to segment the image in the particular examples and the percentage of pixels annotated by these scribbles. All but one of these images require less than 1% of pixel annotations.

kind of a prior (kernel) is imposed by our network and how this effects the segmentation.

For Gradient Descent and with the infinite depth assumption for each layer the kernel is constant. However this changes when we have a rather small

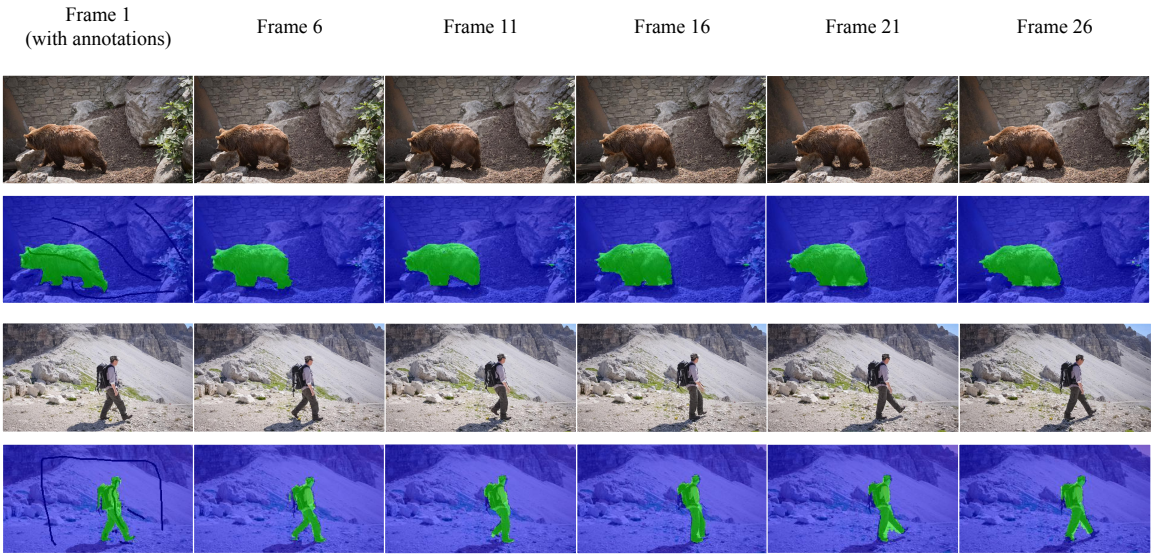

Figure 6: Tracking results for with SegDIP. We can see that annotations given on the first frame and no other annotations on the subsequent ones. Even with videos longer than 20 frames we can see the the inter-frame consistency remains.

depth at each layer and when we use Adam optimizer [12]. At training iteration t, Adam optimizer updates the weights according to:

$$\theta^{t+1} = \theta^t - \tilde{\eta} H^t \frac{\delta \mathcal{L}}{\delta \theta}\left(\theta^t\right) + \beta_1 \left(\theta^t - \theta^{t-1}\right)$$

where θ are the weights of the network β_1 is a hyperparameter controlling the momentum and learning rate, $\tilde{\eta} = \eta\,(1-\beta_1)$ and H^t is a diagonal matrix containing the inverse of a running average of the squared value of the gradients, computed using the hyperparameter β_2. The network outputs are then updated according to:

$$z^{t+1} = z^t + \tilde{\eta}\tilde{\Theta}_L^t \left(y - z^t\right) + \beta_1 \left(z^t - z^{t-1}\right)$$

where for simplicity we wrote $z = x^*$ and where $\tilde{\Theta}^t$ is the resulting gram matrix, which is adapted at each iteration, according to the rule [20]:

$$\tilde{\Theta}^t = \frac{\delta x^*}{\delta \theta} H^t \left(\frac{\delta x^*}{\delta \theta}\right)^\top \Bigg|_{\theta=\theta^t}$$

where $x^* = f_\theta(z)$. The matrix H^k imposes a metric in the weight space which differs from the standard Euclidean metric of Gradient Descent.

The calculation for the whole gram matrix is very expensive and scales quaternarily with the number of pixels. Thus we will instead pick a single pixel and find the affinities with other pixels as per the aforementioned rule. It's important to note that the kernel is not stationary and changes over the training.

In Figure 7 we can see how the gram matrix for a given pixel changes as the training progresses. We can see that at the start of the training high affinity areas are vaguely clustered around the pixel in question. However these high affinity areas become more semantically meaningful as the training continues. We can also see how the affinity map becomes more concentrated around specific pixels with more iterations, this is visible in the bird example and one of the living room examples. This in turn also means a higher intensity on some small specific areas.

The kernel map intuitively predicts how the labels will propagate for a single click in that area, by

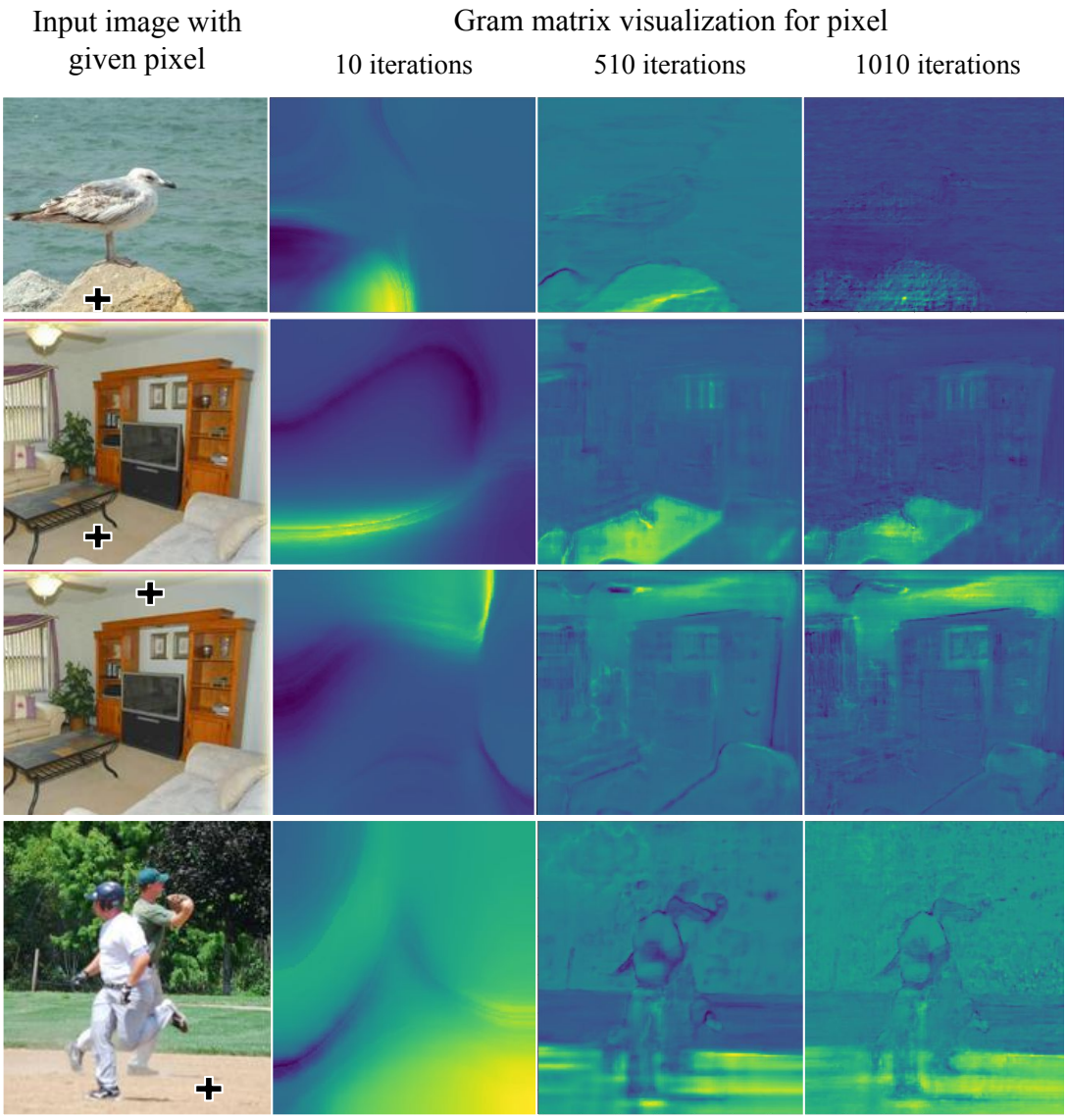

Figure 7: Gram matrix visualized for specific pixels in an image based on the Neural Tangent Kernel at different iterations of the network. The reference pixel given in the first columns. Brighter regions have higher affinity.

denoting the similarity between the pixels. Hence it also makes sense that we would want to start clicking before the network overfits to the image (iteration 1010).

It's interesting to notice that the kernel seems to be doing more than just simply colour based intensity mapping. This is especially visible on the living room picture, where we visualize the similarities for a pixel on the wall. There are parts of wall separated by the furniture that isn't attributed a high affinity despite having a very similar colour to the highlighted pixel. We can thus conclude that our network imposes a similarity based not only on colours, but also the geometry of the image itself.

6 Conclusions

We have shown that online, image-specific training of a compact CNN model which jointly encodes appearance and semantics allows ultra-sparse interactive labelling to produce accurate dense semantic segmentation. Despite promising results, our system's label propagation mechanism works well mainly for proximal regions and/or those sharing similar colour. However it must be mentioned that the setup we are exploring is extremely constrained and apart from colour and proximity, there is not much more information that can be found through a single image. Nonetheless we expect a deeper understanding and study of network architecture and it's biases would give us a lot more insight into engineering further segmentation priors into the network itself e.g. through architectural changes.

We are especially astonished by the video tracking results and even though they are still far from pixel perfect tracking, we believe the results have a lot of potential for improvement. Some further areas of exploration could include incorporating optical flow information into the network or helping the network converge quicker to the new frame.

7 Acknowledgements

Research presented here has been supported by Dyson Technology Ltd. We thank Kentaro Wada, Edgar Sucar, Andre Mouton and Shikun Liu for fruitful discussions.

References

[1] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. 2018.

[2] S. Bagon, O. Boiman, and M. Irani. What is a good image segment? a unified approach to segment extraction. In *ECCV*, 2008.

[3] Y. Bahat and M. Irani. Blind dehazing using internal patch recurrence. In *2016 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9, 2016.

[4] S. Bell-Kligler, A. Shocher, and M. Irani. Blind super-resolution kernel estimation using an internal-gan, 2020.

[5] L. Castrejón. Polyrnn : Polygon-based instance segmentation with recurrent neural networks. 2016.

[6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[7] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2, 1999.

[8] Y. Gandelsman, A. Shocher, and M. Irani. "double-dip": Unsupervised image decomposition via coupled deep-image-priors, 2018.

[9] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. pages 349 – 356, 11 2009.

[10] K. He and J. Sun. Statistics of patch offsets for image completion. pages 16–29, 10 2012.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[13] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation, 2016.

[14] I. D. Mastan and S. Raman. Dcil: Deep contextual internal learning for image restoration and image retargeting, 2019.

[15] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond, 2019.

[16] D. Ren, K. Zhang, Q. Wang, Q. Hu, and W. Zuo. Neural blind deconvolution using deep priors, 2020.

[17] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, aug 2004.

[19] A. Shocher, N. Cohen, and M. Irani. "zero-shot" super-resolution using deep internal learning, 2017.

[20] J. Tachella, J. Tang, and M. Davies. The neural tangent link between cnn denoisers and non-local filters, 2020.

[21] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *International Journal of Computer Vision*, 128(7):1867–1888, Mar 2020.

[22] Y. Yin, Q. Fan, D. Chen, Y. Wang, A. Aviles-Rivero, R. Li, C.-B. Schnlieb, D. Lischinski, and B. Chen. Deep reflection prior, 2020.