Straight to Shapes: Real-time Detection of Encoded Shapes
SUPPLEMENTARY MATERIAL

Saumya Jetley * Michael Sapienza *† Stuart Golodetz Philip H.S. Torr
Department of Engineering Science
University of Oxford
{sjetley,smg,phst}@robots.ox.ac.uk, m.sapienza@samsung.com

1. Training the Regressor

The regression network architecture was based on the YOLO design [7], which has 24 convolutional layers, followed by 2 fully-connected layers; the convolutional layers were pre-trained on the ImageNet dataset [8]. The relative weights between the loss components were set to $\lambda_{shape} = 0.1$, $\lambda_{box} = 5$, $\lambda_{obj} = 1$, $\lambda_{noobj} = 0.5$ and $\lambda_{class} = 1$.

We trained the network for 500 epochs on the training set of the PASCAL VOC SBD splits [3][1], which took about 3 days on one Titan X. We used a batch size of 64, a momentum of 0.9 and a weight decay of 0.0005. The learning rates for the various batches were set as follows:

Batch Numbers	Learning Rate
0 – 200	0.001
201 – 400	0.0025
401 – 20,000	0.005
20,001 – 30,000	0.0025
30,001 – 40,000	0.00125
40,001+	6.25×10^{-4}

To mitigate the effects of a relatively small dataset ($\sim 5k$ training images), we used data augmentation with parameters generated uniformly within the following ranges: rotation ($\pm 4°$), translation ($\pm$ 20% of image size), scaling ($\pm$ 3% of image size), random flipping, intensity scaling ($\alpha I + \beta$, with α in the range ± 2 and β in the range ± 10).

In the main paper, we compared our results qualitatively to the work of Arnab *et al*. [1], who kindly tested the algorithm in [1] on the images that we selected from YouTube videos. Note that we did not compare to [1] quantitatively, as different training and test data was used in [1] (including data from the standard PASCAL VOC dataset [2], the SBD dataset [3], and MS-COCO [5]).

*Authors contributed equally

†M. Sapienza performed this research at the University of Oxford, and is currently with Samsung Research America, Mountain View CA.

[1] https://github.com/bharath272/sds_eccv2014/blob/master/train.txt

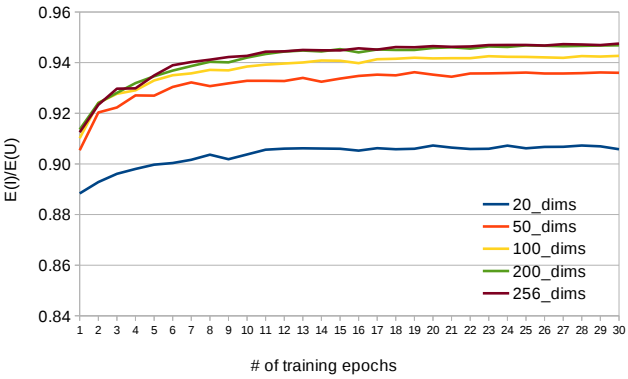


Figure 1: Learning curves for the denoising auto-encoder architectures with different sizes of the embedding space.

2. Training the Autoencoder

We trained the auto-encoder architecture described in §5.3 of the main paper for various dimensionalities of the embedding space, namely 20, 50, 100, 200 and 256. For both training and testing purposes, we cropped each image to the boundary of its inset binary mask and resized it to 64×64. We used a batch size of 128 and an initial learning rate of 0.001, and trained the networks for a total of 300 epochs, with the learning rate being reduced by a factor of 2 every 60 epochs. The network weights were tuned using back-propagation and stochastic gradient descent (in particular, the Adam optimization method [4]).

We adjusted the network weights to minimise the binary cross-entropy loss ℓ_{ce} between the input binary mask m and the reconstructed binary mask m^{recon} as follows:

$$\ell_{ce} = -\sum_{i=1}^{n} m_i \cdot \log(m_i^{recon}) + (1 - m_i) \cdot \log(1 - m_i^{recon}) \quad (1)$$

In this, n is the total number of pixels in the input mask.

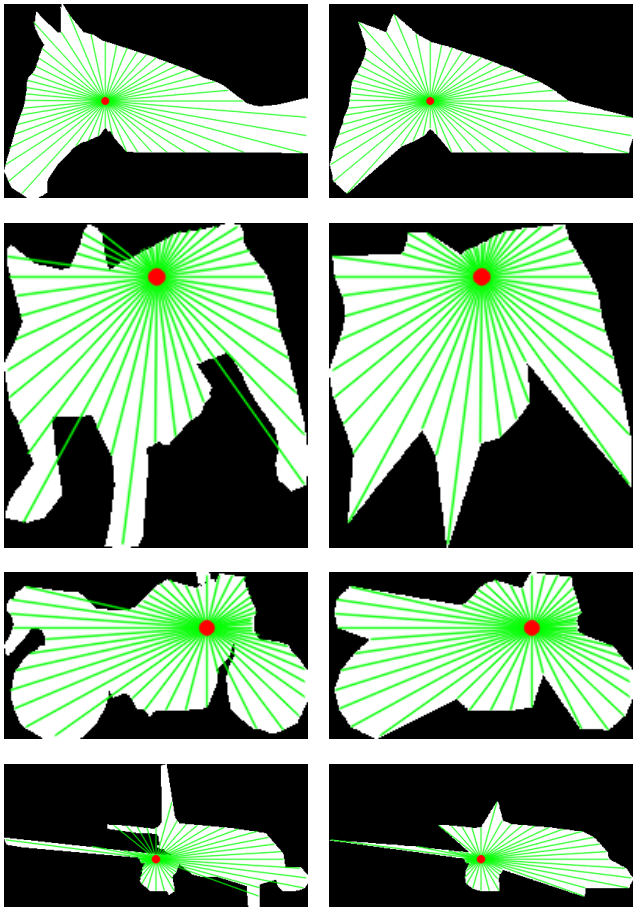


Figure 2: Some example radial descriptors of size 50, superimposed on the original masks (left column) and the reconstructed masks (right column). Each chosen centre is shown with a red dot, and we draw green rays between the centre and each point on the represented shape contour. By choosing the centre and the way in which we cast rays appropriately in each case, we achieve much better reconstruction results than we could using the standard centre of the mask and casting rays outwards; however, for more complicated shapes such as the dog (second row) and the aeroplane (fourth row), our radial descriptors fail to capture important details such as the shapes of the ears or the tailplane.

The convergence plots for various sizes of the embedding space are shown in Fig. 1. It is interesting to note that with the reduction in the size of the embedding space, the performance degrades gracefully and minimally. Noticeably, with a 10 times reduction in the dimensionality, the expected IoU (Expected(I) / Expected(U)) falls by only around 0.04 points.

3. Radial Descriptor Computation

To compute a radial descriptor for a binary shape mask, we adopt the following scheme. Given a fixed descriptor size d, we allocate the first 2 elements of the descriptor to contain the (x, y) coordinates of the centre point we choose within the mask, normalised to $[0, 1] \times [0, 1]$ (where $(0, 0)$ denotes the top-left of the mask and $(1, 1)$ denotes the bottom-right). We allocate the remaining $d - 2$ elements to contain distances between the chosen centre point and the boundary at evenly-spaced angles in $[0, 2\pi)$, each appropriately scaled to fall in the $[0, 1]$ range. In particular, element $i+2$ of the descriptor stores the scaled distance between the centre point and the boundary at an angle of $2\pi i/(d-2)$.

In practice, to achieve better IoU scores for reconstruction, we construct several radial descriptors for each shape and pick one with maximal IoU. In particular, we try 25 possible centre points, evenly spaced on a 5×5 grid superimposed over the $w \times h$ mask at locations

$$\left(\frac{jw}{6}, \frac{ih}{6}\right) \; : \; i, j \in [1, 5].$$

We also try conceptually casting rays both away from and towards the centre point (in practice, casting rays towards the centre point is implemented by casting rays away from the centre point and only stopping when the outer boundary of the shape is reached). This scheme significantly improves the IoU scores we can achieve for reconstruction with respect to always using the centre of the mask and casting rays outwards.

Figure 2 shows some of the size 50 radial descriptors we calculate, superimposed on both the original and reconstructed masks so as to illustrate what aspects of shape the descriptors can and cannot capture. The chosen centre in each case is shown with a red dot, and we draw green rays between the centre and each point on the represented shape contour.

4. Software Implementation

We used the Darknet framework [6] (written in C) from June 2016 for training our shape prediction model. We developed our own C++ software to deal with dataset loading, manipulation, transformations and preparation for the neural network. We also developed C++ code to evaluate our system on instance segmentation; our code is several times faster than the standard MATLAB version of Hariharan et al. [3]. For ease of development, we used the Torch framework for learning the shape embedding, and interfaced the C++ and Lua code using the LuaBridge library[2]. This interface includes delays that could be avoided by implementing an optimised version of our Lua code in C++.

Real-Time Demo. We also developed some code to demonstrate that our system generalises to scenes captured by a web-camera, as illustrated in Fig. 3.

[2] https://github.com/vinniefalco/LuaBridge

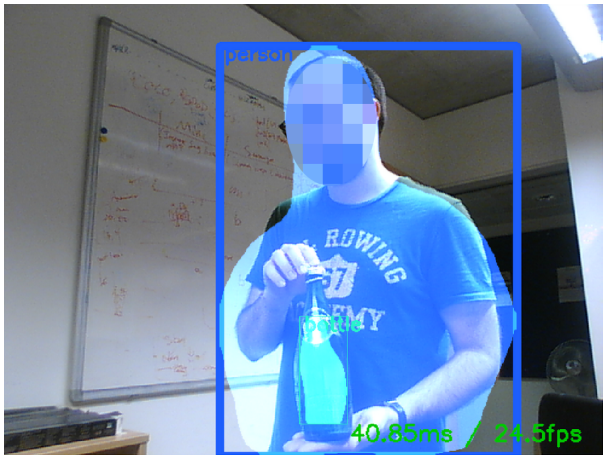


Figure 3: Testing direct regression to shape from a live webcam image stream.

Reproducible Results. We will make the code and models used to obtain our results available upon publication.

5. Further Qualitative Results

Visualisation of the embedding spaces. In Figs. 4, 5, and 6, we visualise the embedding spaces produced by the binary masks, radial vectors and learned embedding respectively. We fix 20 anchor points in each of the embedding spaces around the main diagonal of the 2D space. For each of the anchor points, we sample the 20 nearest neighbours to observe how the shapes are organised in the space. At a macroscopic level, as we move along the anchor points (across the rows of the tiled images), we notice that the shapes change between being bulky and rounded to shapes with multiple protrusions. For the learned embedding space, this pattern is more pronounced (see Fig. 6). In the radial space, both bulky and thin shapes are present along a single row (see Fig. 5 rows 3 and 7), thus in places showing an inferior organisation.

Both the binary mask space and the learned embedding space are well-organised for shape similarity. It is interesting to note that the learned embedding space achieves a similar shape organisation to the downsampled binary mask space, but with an order of magnitude reduction in size.

Shape prediction. In Figs. 7 and 8, we show additional qualitative results of our shape prediction system on the PASCAL VOC [2] (SBD) validation set.

References

[1] A. Arnab and P. H. S. Torr. Bottom-up instance segmentation using deep higher-order crfs. In *Proc. British Machine Vision Conference*, 2016. 1

[2] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. Pascal visual object classes challenge results. *Available from www.pascal-network.org*, 2005. 1, 3

[3] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Proc. European Conf. Computer Vision*, 2014. 1, 2

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014. 1

[5] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *Proc. European Conf. Computer Vision*, 2014. 1

[6] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. 2

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2016. 1

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *Int. Journal of Computer Vision*, 2015. 1

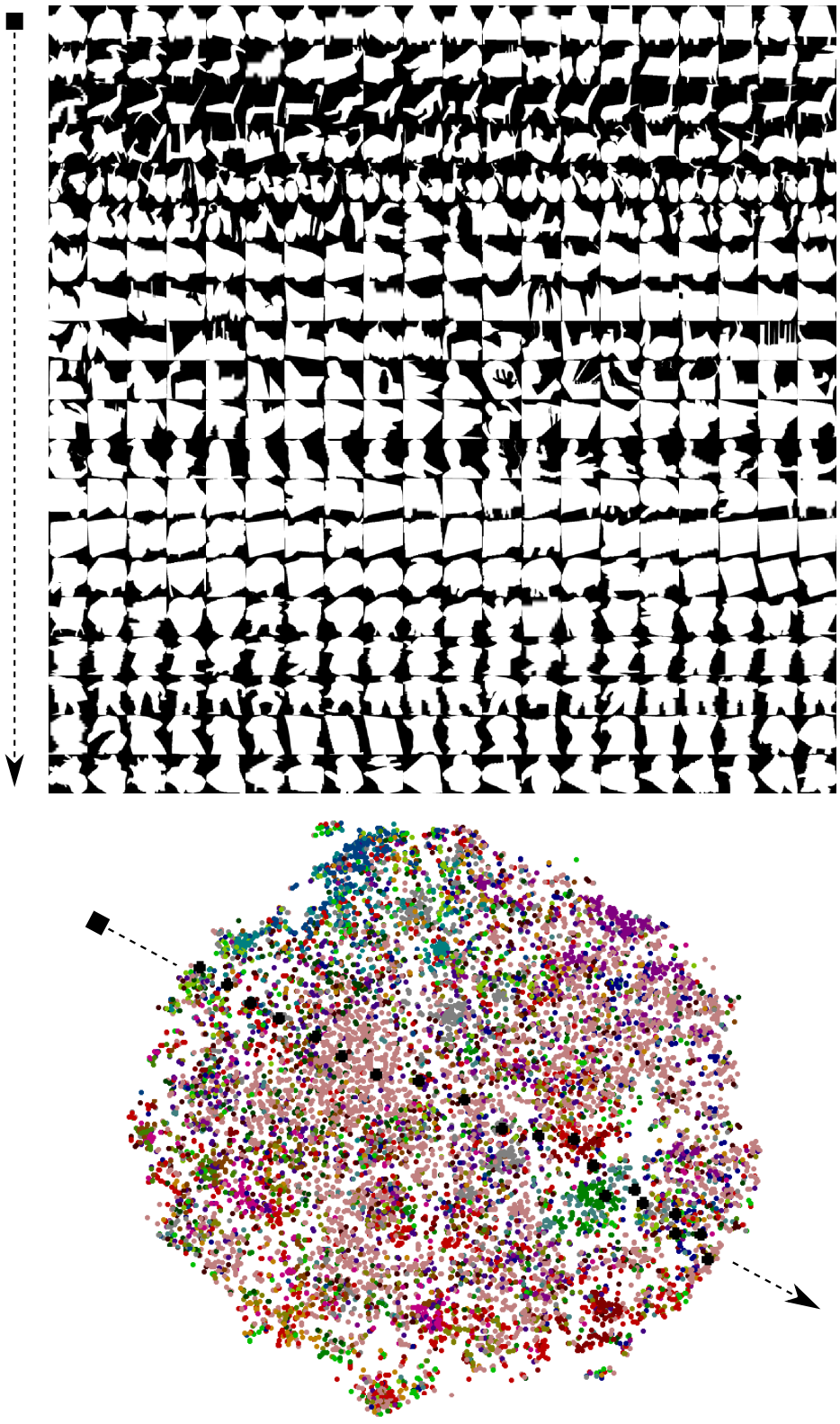


Figure 4: Visualisation of the 16×16 binary mask embedding space. (top) Ground truth binary masks. Rows (in top) represent the 20 neighbouring binary masks for each of the 20 anchor points in the embedding space (bottom).

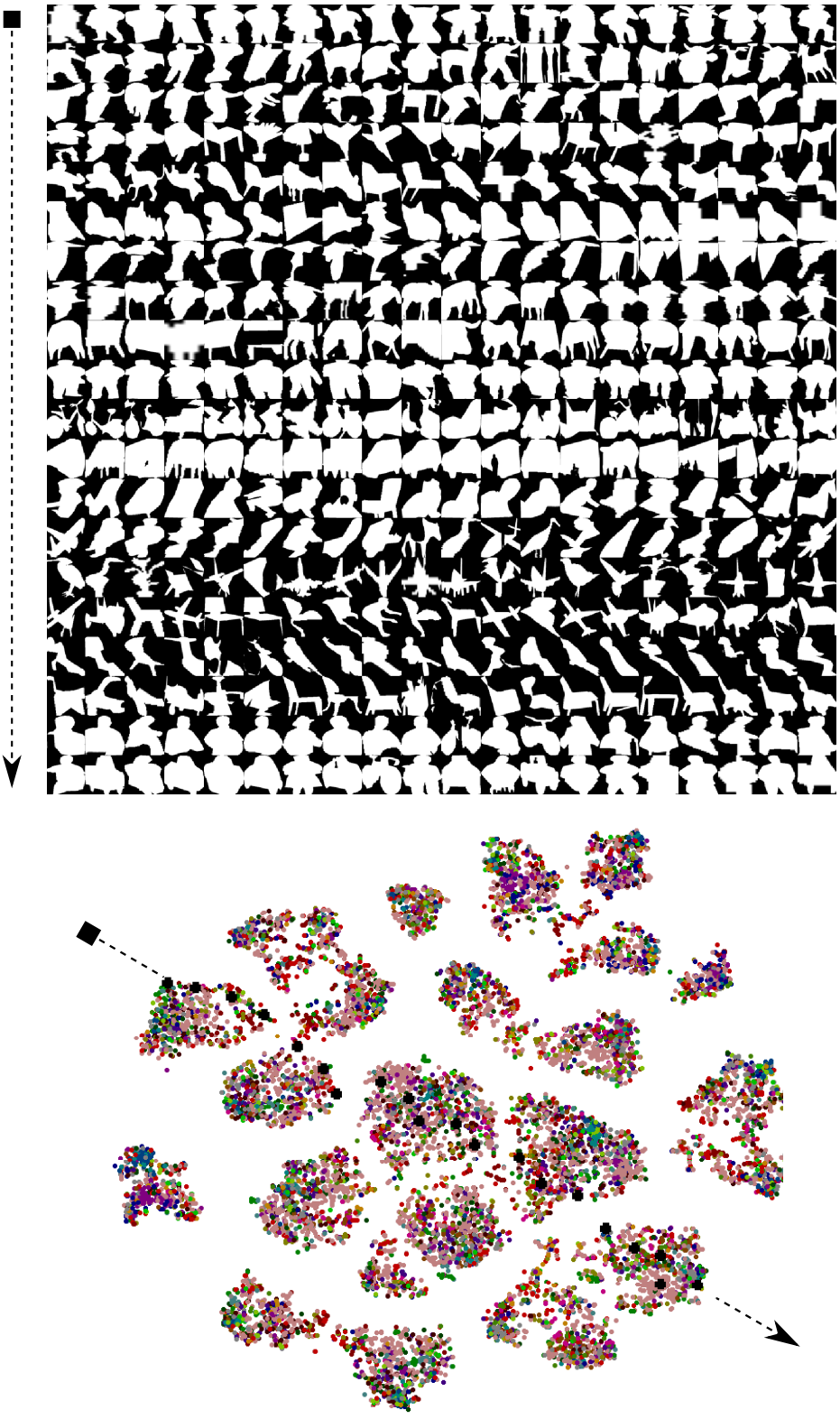


Figure 5: Visualisation of the 256-D radial vector embedding space. (top) Ground truth binary masks. Rows (in top) represent the 20 neighbouring binary masks for each of the 20 anchor points in the embedding space (bottom).

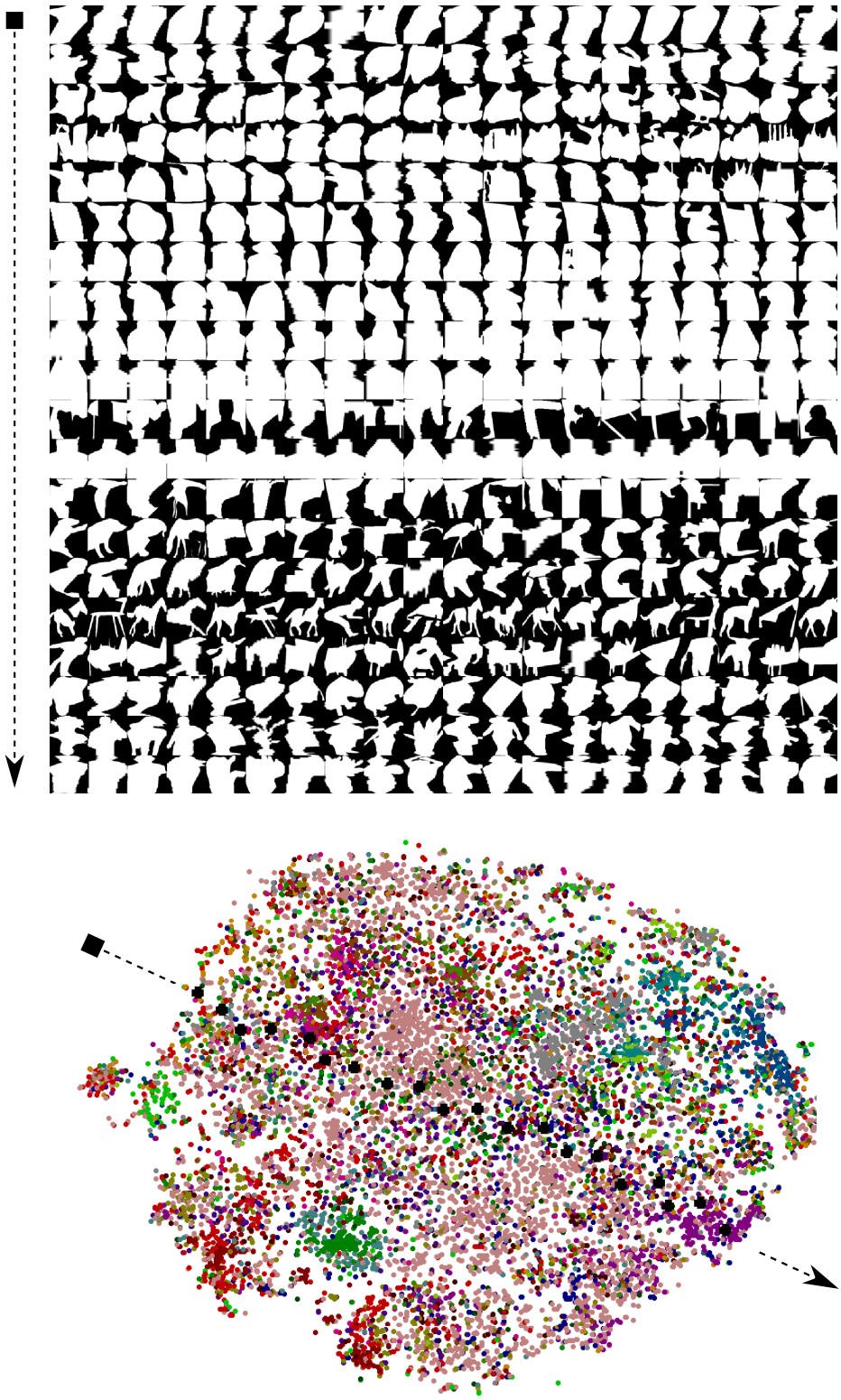
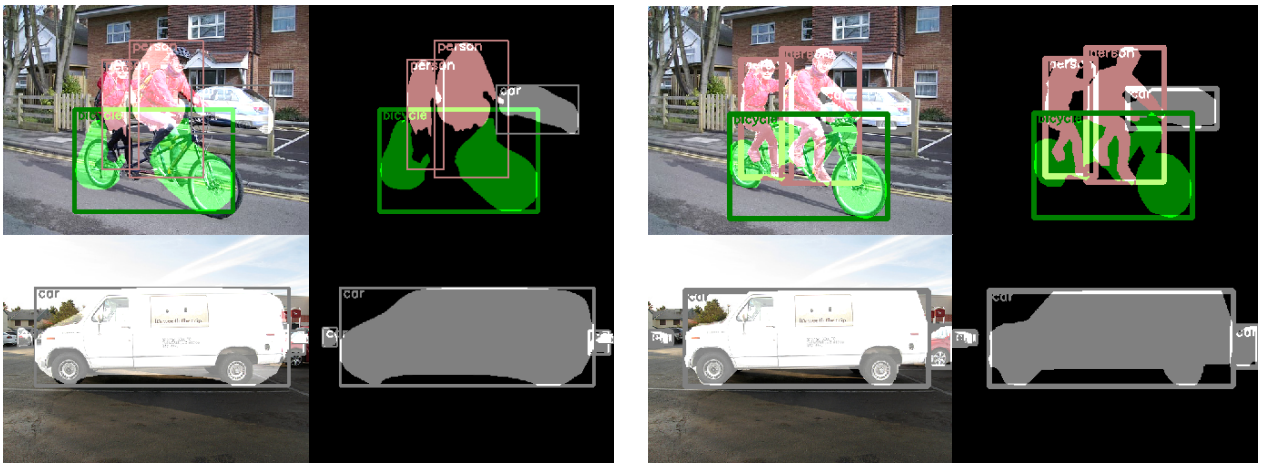


Figure 6: Visualisation of the 20-D learned embedding space. (top) Ground truth binary masks. Rows (in top) represent the 20 neighbouring binary masks for each of the 20 anchor points in the embedding space (bottom).



(a) (b)

Figure 7: A more detailed look at the results. **(a)** The results from our shape prediction network next to **(b)** the ground truth. **(a)** (top) Two people ride a tandem bike. Note that our system predicts shapes and bounding boxes (learnt jointly) for localisation. As such, any errors in localisation will affect the location at which the shape gets superimposed onto the image (the box does not hug the bike tires tightly). In **(b)** (top), note the complexity of the shapes that need to be predicted. In the bottom row, multiple cars are detected. The predicted shape contains information on the direction that the car is facing, information that is not available from bounding boxes alone. It is noteworthy that even thought the output shape mask is not pixel-accurate, it still contains a lot of information about the object's orientation and shape, and its similarity to other shapes in the shape embedding.

Figure 8: Additional qualitative results from the PASCAL VOC (SBD) validation set. Note the huge variety in shapes, even within a single category.