Atom cloud detection and segmentation using a deep neural network

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 045008 https://doi.org/10.1088/2632-2153/abf5ee

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
25 November 2020

REVISED
25 March 2021

ACCEPTED FOR PUBLICATION
8 April 2021

PUBLISHED
15 July 2021

PAPER

Atom cloud detection and segmentation using a deep neural network

Lucas R Hofer[1], Milan Krstajić[1,2], Péter Juhász[1], Anna L Marchant[1,3] and Robert P Smith[1,*]

[1] Clarendon Laboratory, University of Oxford, Parks Road, Oxford OX1 3PU, United Kingdom
[2] Cavendish Laboratory, University of Cambridge, J. J. Thomson Avenue, Cambridge CB3 0HE, United Kingdom
[3] Current address: RAL Space, Science and Technology Facilities Council, Rutherford Appleton Laboratory, Harwell, Didcot, Oxfordshire OX11 0QX, United Kingdom
[*] Author to whom any correspondence should be addressed.

E-mail: robert.smith@physics.ox.ac.uk

Keywords: ultracold quantum matter, machine learning, deep neural networks, Bayesian optimization, object detection, instance segmentation, image processing

Abstract

We use a deep neural network (NN) to detect and place region-of-interest (ROI) boxes around ultracold atom clouds in absorption and fluorescence images—with the ability to identify and bound multiple clouds within a single image. The NN also outputs segmentation masks that identify the size, shape and orientation of each cloud from which we extract the clouds' Gaussian parameters. This allows 2D Gaussian fits to be reliably seeded thereby enabling fully automatic image processing. The method developed performs significantly better than a more conventional method based on a standardized image analysis library (Scikit-image) both for identifying ROI and extracting Gaussian parameters.

1. Introduction

Deep neural networks (NNs) have revolutionized data analysis and led to automation of tasks that previously required human supervision. Image analysis has particularly benefited through the use of convolutional neural networks (CNNs) [1] and their derivatives which have allowed for image classification [2, 3], object detection [4, 5] and instance segmentation [6]. Although many of these NNs were developed for tasks such as facial recognition by social media networks [7, 8], they have also been used to identify laser modes [9], classify phases in condensed-matter systems [10, 11], reduce measurement errors for trapped-ion qubits [12] and process images from cold atom experiments [13–15]. In this work, we use an instance segmentation NN (see figure 1) to analyze experimental images containing atom clouds in magneto-optical traps (MOTs) and optical dipole traps (ODTs).

Neural networks consist of an input layer and an output layer with a number of intermediate hidden layers which are connected to one another via 'weights'. Rather than employing hard-coded algorithms, NNs learn to emulate data they encounter through training cycles, in which data is iteratively passed through the NN and the output compared to the 'ground truth'. The difference between the two is then used to update the weights between the NNs layers, thereby improving its accuracy. When employing supervised training, this requires a dataset which includes both input data and their associated ground truth values. For object detection NNs, these ground truth values are rectangular bounding boxes for each object, as well as labels classifying the object types in the bounding boxes. Instance segmentation NNs build upon object detection NNs by also requiring pixel-to-pixel segmentation masks in which image pixels comprising the object have mask values of one, whereas all other pixels have mask values of zero.

Our dataset consists of images of cold atom clouds in a MOT [16] and an ODT [17] (see figures 2(a)–(c)). Atom clouds in these traps form approximately Gaussian density distributions [18]. Fitting a cloud allows the parameters describing the distribution (Gaussian parameters) to be extracted and used to ascertain information such as the cloud's size and density. Furthermore, by using time-of-flight measurements [19] the temperature of the atoms can be determined.

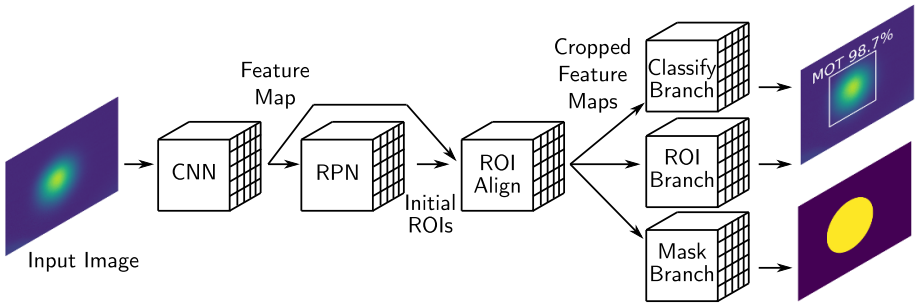


Figure 1. Overview of the neural network (NN). An experimental image is initially fed into a convolutional neural network (CNN) which produces a 'feature map' of the image. The region-proposal network (RPN) uses this feature map to generate regions-of-interest (ROI) where atom clouds are likely contained. In the ROI alignment stage, the CNNs feature map is cropped and resized for each cloud's ROI before being fed into three parallel branches. The first branch (ROI) generates refined ROIs, whereas the second branch (classify) classifies the type of atom cloud in the ROI. Finally, the third branch (mask) generates a segmentation mask corresponding to the $1/e^2$ contour of the atom cloud.

A region-of-interest (ROI) [20] centered on the atom cloud is used during fitting as objects in the image other than the atom cloud can cause an inaccurate fit (e.g. an atom cloud in another trap or extraneous noise). Additionally, decreasing the fit area can significantly decrease the fit time when using two-dimensional fitting. Manually determining the ROI is time-consuming when analyzing a large number of images and an algorithmic procedure is often employed, such as taking the 'center of mass' and then iteratively expanding the ROI around this point until the fractional enclosed 'power' exceeds some threshold. Another method involves performing connected component analysis on a binarized version of the image and then measuring the resulting regions using common image processing libraries [21]. However, if the image is noisy (e.g. contains fringing), the proposed ROIs will be inaccurate for these types of methods (section 6).

Here we propose a deep NN based approach to ROI determination in which a NN finds the ROI for each atom cloud in an image (see figure 1). Furthermore, the NN differentiates between clouds (classification) in a MOT and those in an ODT and also outputs a segmentation mask for each cloud from which Gaussian parameters are directly extracted. The classification feature is particularly useful when the cloud type is not *a priori* known from the experimental sequence (e.g. images containing both MOT and ODT clouds during the ODT loading). Although features such as the position or aspect ratio can be used for classification with additional manual or experimental input, the NN needs the image alone (beyond training on an appropriate dataset) to determine cloud types and so is easily adaptable to other cold atom experiments with different numbers of clouds or cloud types.

The rest of the article is arranged as follows: section 2 describes the experimental dataset used for NN training and validation, section 3 describes the training process, section 4 discusses Bayesian optimization (BO) [22, 23] of the NNs hyperparameters [23] and section 5 examines how the Gaussian parameters are calculated from the segmentation mask. In section 6 we compare our proposed NN method to a more conventional method of determining both ROIs and Gaussian parameters.

2. Experiment and dataset

To produce the ultracold atom clouds, erbium atoms are initially trapped and cooled to $\sim$20 μK in a narrow-line MOT [24] before being loaded into an ODT formed from a 30 W, 1030 nm laser beam focused down to a $\sim$40 μm waist (see figure 2). Optimization of the trap loading involves maximizing the atom number while minimizing the cloud temperature. The atom number is found by fitting the atom cloud in either a fluorescence or absorption image[4] with a two-dimensional (2D) Gaussian (see equation (1)) and then integrating under the curve. The cloud temperature can be determined from how the cloud width evolves during time-of-flight expansion.

The experimental dataset consists of 260 fluorescence and absorption images (1936 $\times$ 1216 pixels) along with their ROIs, labels and segmentation masks [25]. Of these, 130 images contain clouds released from the MOT with no ODT present (see figure 2(b)) and 130 images contain either just atoms released from the ODT (see figure 2(c)) or images where atoms released from the ODT and the MOT are both present (see

[4] Experimental images shown in the paper are processed from two to three raw images. Fluorescence images require both an image with atoms and a background image without atoms. Absorption images additionally require a probe image in which the probe beam is turned on, but no atoms are present.

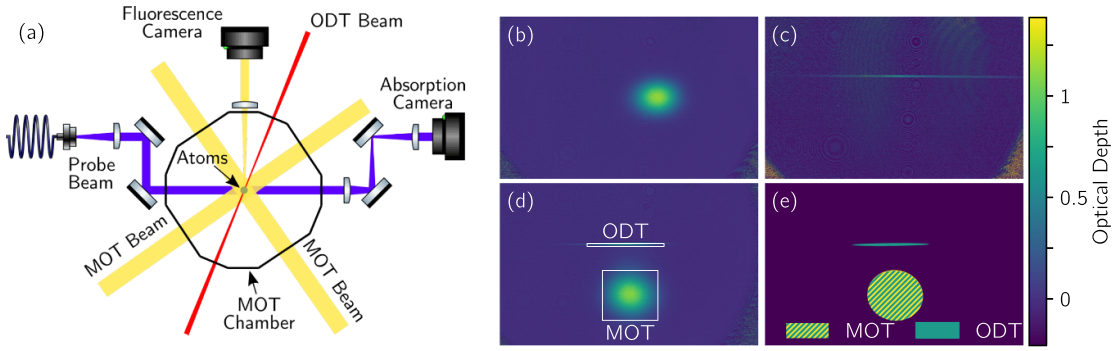


Figure 2. Construction of labelled experimental dataset. (a) Experimental setup. The MOT is formed from six 583 nm laser beams in conjunction with a quadrupole magnetic field. Atoms trapped in the MOT can be loaded into an ODT formed from a single 1030 nm laser beam. The fluorescence camera captures the light scattered by the atoms from the MOT beams. During absorption imaging, a 401 nm probe beam passes through the atom cloud which absorbs a portion of the light creating a shadow imaged by the camera. (b) Image of an atom cloud released from the MOT. (c) Image of an atom cloud released from the ODT. The optical depth is multiplied by six to increase ODT visibility. (d) Image with both MOT and ODT atom clouds. Each cloud is labelled and a region-of-interest (ROI) box is drawn around it (white lines). (e) Segmentation masks for the MOT (hatched yellow) and ODT (solid green) clouds in (d).

figure 2(d)). We manually label the atom clouds in the images as 'MOT' or 'ODT' and draw a ROI box at the clouds' edges which we define as the $1/e^2$ radii along the x and y axes (see figure 2(d)). This definition prevents the ROI boxes from overlapping when the MOT and ODT are both present; however, the ROI boxes are also easily expandable when analysis requires the wings of the distribution.

The manually drawn ROIs were expanded by a factor of two—excepting where the expanded ROIs would overlap—and the atom clouds inside fit with a 2D Gaussian:

$$I(x,y) = I_\mathrm{b} + I_0 e^{-2\left(\frac{[(x-x_0)\cos\theta+(y-y_0)\sin\theta]^2}{w_x^2}+\frac{[(y-y_0)\cos\theta-(x-x_0)\sin\theta]^2}{w_y^2}\right)}, \quad (1)$$

where $I(x,y)$ is the image intensity, I_b is the background intensity, I_0 is the peak intensity, x_0 and y_0 are the center coordinates, w_x and w_y are the $1/e^2$ radii along the major and minor axes and θ is the angular orientation of the distribution. To increase the accuracy of the ROIs used for training, the fit parameters were used to calculate the $1/e^2$ radii along the image axes [9] (previously estimated by eye) and the ROI boxes redrawn using these values. The process of fitting and redrawing the ROI boxes from the fit parameters was then completed a second time with subsequent iterations neglected due to an insignificant increase in accuracy.

A segmentation mask was generated for each atom cloud (see figure 2(e)) with the mask borders placed at the $1/e^2$ contour of the cloud—calculated from the final fit parameters; pixels within the $1/e^2$ contour were set to one, whereas pixels outside were set to zero. Finally, the dataset was randomly split into a training dataset with 200 images and a validation dataset with 60 images.

3. Neural network and training

We use the NN Mask R-CNN [26] to detect and bound the atom clouds, as well as to provide segmentation masks for each cloud (see figures 3(a, b)). The NN (see figure 1) begins with the experimental image being fed into a CNN base (ResNet-50 [27]). The CNNs outputted feature map [28] is then passed into a region-proposal network (RPN) which generates ROIs where objects are likely located. Next, these ROIs are cropped from the CNNs feature map in a ROI alignment stage and resized to uniform dimensions. The cropped feature maps are then fed into three parallel branches. The first applies a classifier to determine the object type and give the confidence of its prediction—which is helpful in determining whether to use the ROI in post-NN analysis. The second branch gives a more accurate ROI box prediction (see figure 3(a)) and finally the third outputs a segmentation mask for the object inside the ROI (see figure 3(b)). Since all three branches share the same base, computation speed is significantly increased [29] for both training and evaluation.

During training, the NN output is compared to the ground truth (i.e. the expected output from the training dataset) via a loss function; the loss is then back-propagated [30] through the NN to adjust the weights between layers and refine the NN model (see figure 4(a)). The loss function for the RPN stage is L1 loss [31], for the ROI branch it is Smooth L1 loss [32], the classifier uses categorical cross entropy loss [33] and lastly the mask branch utilizes binary cross entropy loss [34]. Although the loss can be separately

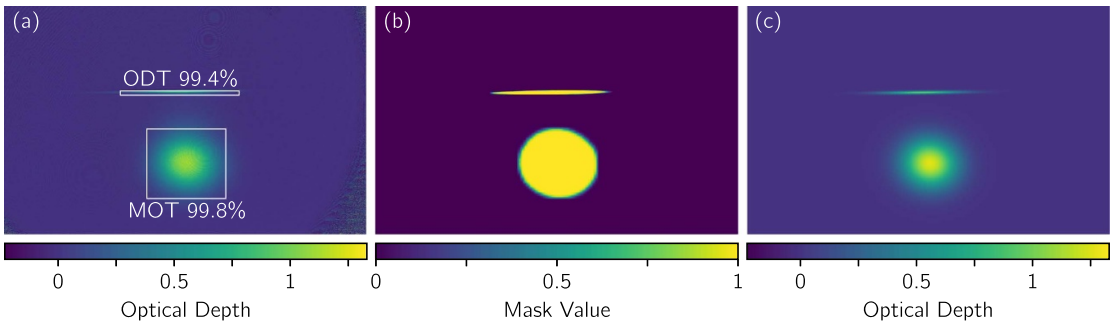


Figure 3. Neural network output. (a) Sample image from the validation dataset with the regions-of-interest (solid white lines), labels and confidence scores predicted by the NN. (b) The segmentation masks returned by the NN for (a) which, due to the internal structure of the NN, have values between zero and unity. (c) Reconstruction of (a) using equation (1) and the extracted 2D Gaussian parameters from (b).

back-propagated for each branch, a simpler approach is taken here in which the losses are summed together and then back-propagated to update the weights of the NN [26].

An epoch denotes a single cycle of training in which every image in the training dataset is passed through the NN, the loss calculated and the model weights updated. Increasing the number of training epochs can increase the NN's final accuracy so long as the NN does not overfit on the training data. However, due to finite computational resources, we restrict the training of an individual NN to 15 epochs—determined to be sufficient, see below—and instead use BO to increase the accuracy of the final trained model (see section 4).

Five hyperparameters, which tune the learning process, are set before the NN's training phase. The first is the *learning rate* which determines the size of the step the NN takes during stochastic gradient descent [35]. If the learning rate is too low, the NN will take too long to converge to the minimum of the loss function, whereas if the learning rate is too high, the NN will not be able to descend to the minimum, but will oscillate around it or diverge. Since larger learning rates are useful at the beginning of training and smaller learning rates are useful towards the end, a learning rate scheduler [36] is employed which decreases the learning rate by some scalar (*decay*, the second hyperparameter) after a fixed number of epochs (*step size*, the third hyperparameter). The fourth hyperparameter is the *momentum* which prevents the NN from getting stuck in a local minimum during training [37]. The last hyperparameter is the *batch size*—the number of images simultaneously passed through the NN—and, unlike the other hyperparameters which are tuned with BO, is fixed to four for all NNs.

The accuracy of the NN is evaluated (see figure 4(b)) using the Common Objects in Context (COCO) evaluation metric which entails calculating the intersection-over-union [38]:

$$\mathrm{IoU} = \frac{\mathrm{area}\left(\mathrm{ROI_p} \cap \mathrm{ROI_{gt}}\right)}{\mathrm{area}\left(\mathrm{ROI_p} \cup \mathrm{ROI_{gt}}\right)}, \tag{2}$$

where $\mathrm{ROI_p}$ is the ROI prediction from the NN and $\mathrm{ROI_{gt}}$ is the ground truth ROI. Values below a predetermined IoU threshold (e.g. $\mathrm{IoU} = 0.5$) are considered a false prediction—mislabelling the object class within the ROI is also a false prediction—whereas values above are considered a true prediction. A precision recall-curve [39] is then constructed and integrated to give the average precision (AP) for the given threshold (e.g. $\mathrm{AP_{50}}$ for the $\mathrm{IoU} = 0.5$ threshold). Finally, the AP is calculated for ten IoU thresholds (0.50–0.95 with a step size of 0.05) and averaged to give the mean average precision (mAP)—which is the metric generally reported for object detection. The mask AP values and mAP can be similarly calculated [40].

The NNs are trained in a Google Colab notebook [41] utilizing a GPU backend and implemented in PyTorch [42] using the pre-built Mask-RCNN model in the Torchvision package. Rather than training the NN from scratch, the model weights are loaded from a network pre-trained on the COCO train2017 dataset [43]—which significantly reduces the time required to train the network and increases the model's final accuracy [44]. Additionally, this transfer learning, complemented by data augmentation (in which images are randomly flipped horizontally during training), allows us to use a relatively small training dataset. After each training epoch, the updated NN is evaluated on the validation dataset which yields the mAP for both the object detection and the segmentation mask branches. These two values are averaged together to give the accuracy of the training epoch (average mAP) and the epoch with the highest average mAP determines the overall accuracy of the NN model. After using BO to determine the hyperparameters which give the highest accuracy, we retrain the NN with these parameters for 30 epochs (see figure 4(c)) and verify that the NNs accuracy becomes asymptotic at the 15 epoch mark—with no overfitting seen thereafter.

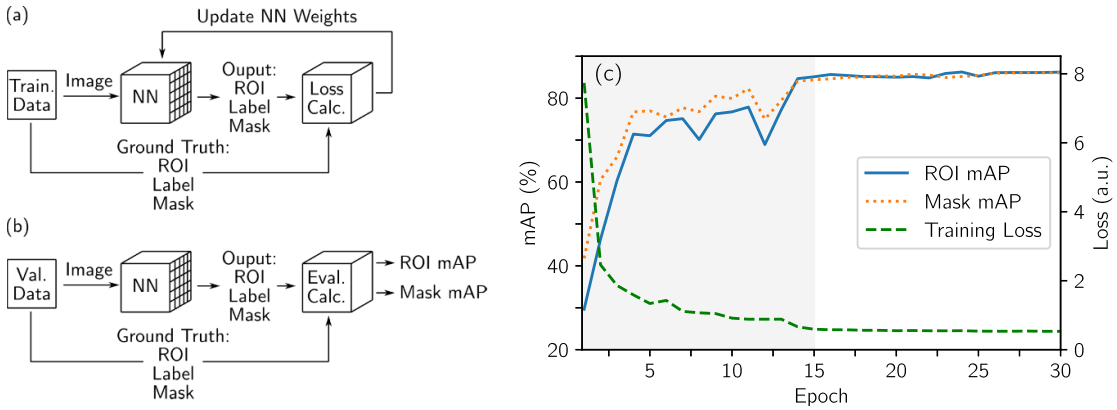


Figure 4. Training and validating the neural network. (a) During training of the NN, images from the training dataset are fed into the NN which outputs the regions-of-interest, labels and masks. These outputs are compared to the ground truth values and the loss calculated. The loss is then back-propagated through the NN and used to update the weights between the NNs layers. (b) When evaluating the NN, the validation dataset is used in a process similar to (a); however, the mean average precision (mAP) for both the object detection branch (ROI) and segmentation mask branch are calculated rather than the loss. (c) The mAP of the ROI branch (solid line), the mAP of the segmentation mask branch (dotted line) and the training loss (dashed line) as a function of the training epoch for the NN trained with the best set of hyperparameters (see table 1). The shaded region represents the number of epochs used during Bayesian optimization.

Table 1. The hyperparameter search space used during Bayesian optimization along with the hyperparameter values used to train the NN to the highest average mAP (see figures 4(a) and 5).

Parameters	Lower bound	Upper bound	Log scale	Best value
Learning rate	0.0001	0.009	Yes	0.0033
Momentum	0.7	0.92	No	0.86
LRS step size	3	15	No	13
LRS decay	0.001	1	Yes	0.049

4. Bayesian optimization of hyperparameters

The accuracy of the trained NN is sensitive to the hyperparameters used during training. In the past, the hyperparameters were tuned through grid search, random search [45] or by hand; however, in recent years, BO has been successfully employed to find the best set of hyperparameters [23, 46]. BO is particularly useful when trying to find the minimum (or maximum) of a function which is noisy and expensive to evaluate—such as NN training—thereby making a grid search of the parameter space impractical [22].

Bayesian optimization takes the function value (cost) at previously evaluated points and uses a Gaussian process to model the cost as a function of the parameter space [47]. The model also determines the confidence of its predictions in a given region of the parameter space: perfect certainty at evaluated points, low uncertainty near evaluated points and high uncertainty far from evaluated points. The BO loop then determines where to evaluate the function next by weighing the benefits of evaluating the function near the model's predicted minimum (or maximum) or evaluating the function in an unexplored region of the parameter space [48].

When optimizing our NN training with BO, the average mAP of the NN is evaluated as a function of the hyperparameter space which consists of the learning rate, momentum, and the learning rate scheduler (LRS) step size and decay (see table 1). As a warm start, the NN is initially trained and evaluated at five quasi-randomly (Sobol generated [49]) distributed points (see black squares in figures 5(a)–(f)). Further evaluation points (red circles in figures 5(a)–(f)) are iteratively determined by the BO loop. The Ax Python package is used for the BO loop as it provides a high level interface to the BoTorch [50] BO package.

With an increasing number of BO evaluation trials the best achieved average mAP rises and converges (see figure 5(g)). The best set of hyperparameters (see table 1) gives a mAP of 86.3% for the object detection branch—locating 88 of the 89 clouds—and a mAP of 85.8% for the mask branch. These mAP values are higher than those for similar NNs trained on the COCO validation dataset [26, 51] which is likely due to our NN only needing to classify two object types rather than the eighty in the COCO validation dataset [38], as well as the relatively simple features of the MOT and ODT clouds. The NN also performs well for clouds with a low signal-to-noise ratio (SNR) which we define as the cloud's peak amplitude divided by the standard

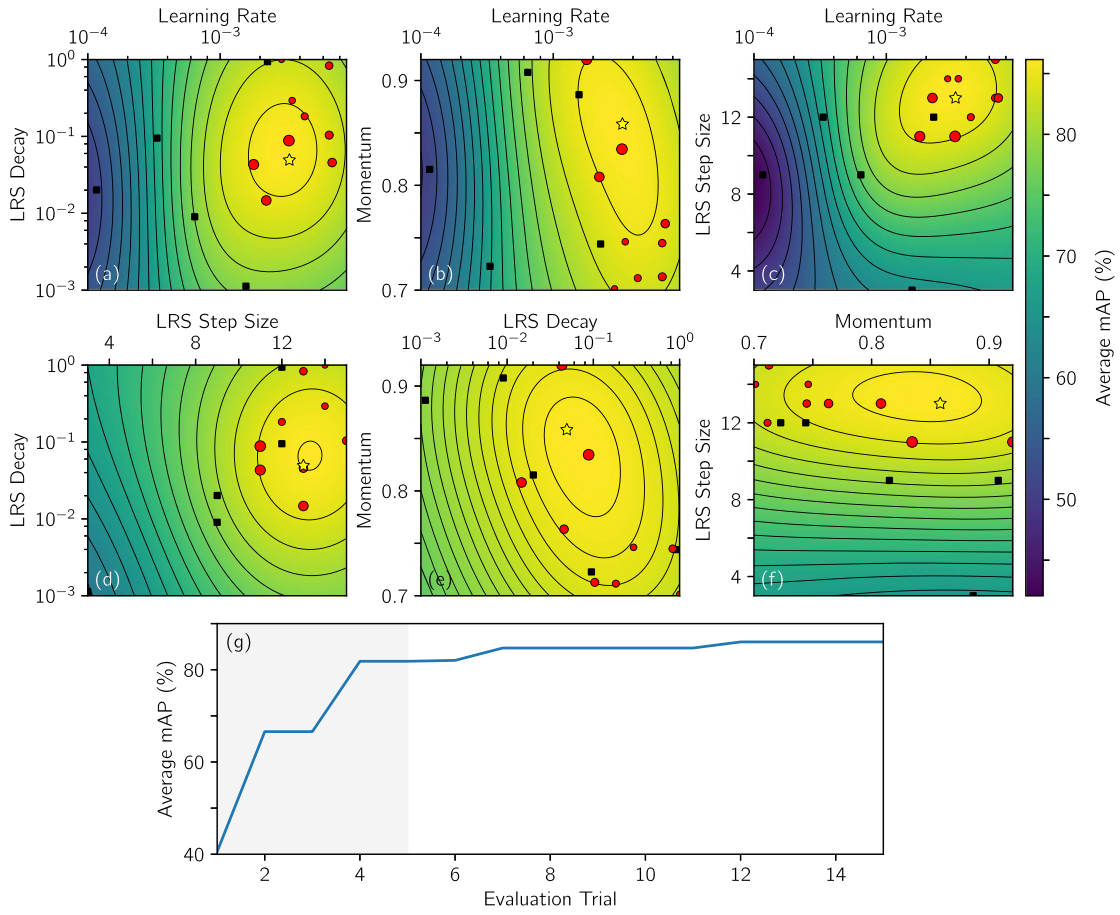


Figure 5. Bayesian optimization of hyperparameters. (a)–(f) The average mAP of the NN predicted by the Bayesian optimization (BO) model as a function of the hyperparameters. The plots are 2D slices through the 4D parameter space taken at the best position (yellow star) for those parameters not displayed. The NN is initially trained and evaluated at five quasi-random hyperparameter positions (black squares) after which the BO loop iteratively determines the evaluation points (red dots, size increases with trial). (g) The best achieved average mAP as a function of the evaluation trial; the quasi-random trial region is shaded in grey.

deviation of the image's background intensity; the validation dataset's nosiest cloud has an SNR of only 2.1 but returns an ROI IoU score of 0.86.

5. Gaussian parameter analysis

We can extract the parameters characterizing the cloud's 2D Gaussian distribution (see equation (1)) directly from the NNs segmentation mask output (see figure 6(a)). Taking the 1st moments of the mask yields the center coordinates $\{x_0, y_0\}$ of the atom cloud, whereas the $1/e^2$ radii $\{w_x, w_y\}$ and the angular orientation θ can be determined by calculating the 2nd moments [52] of the mask. The background intensity I_b is calculated by generating a histogram of the experimental image's pixel values (where there is no cloud) and then taking the peak position (see figure 6(c)). The amplitude of the 2D Gaussian I_0 is calculated by summing the image's intensity inside the $1/e^2$ contour to find the power (P) and then applying:

$$I_0 = \frac{2}{(1-e^{-2})}\left(\frac{P}{\pi w_x w_y} - I_\mathrm{b}\right). \tag{3}$$

This method of extracting clouds' Gaussian parameters directly from the segmentation mask was applied to the validation dataset (see figure 3(c)) and the results compared to the Gaussian parameters extracted via a 2D fit of the experimental image (see figure 7, blue bars). Apart from θ, we normalized the differences to obtain a relative error. The cloud center $\{x_0, y_0\}$ and the cloud radii $\{w_x, w_y\}$ were divided by the fitted radii, whereas the peak amplitude I_0 and the background intensity I_b were both divided by the fitted peak amplitude.

The Gaussian parameters extracted from the NNs segmentation mask can either be used directly or as seed parameters for a conventional 2D fit—increasing the likelihood of fit convergence and reducing the

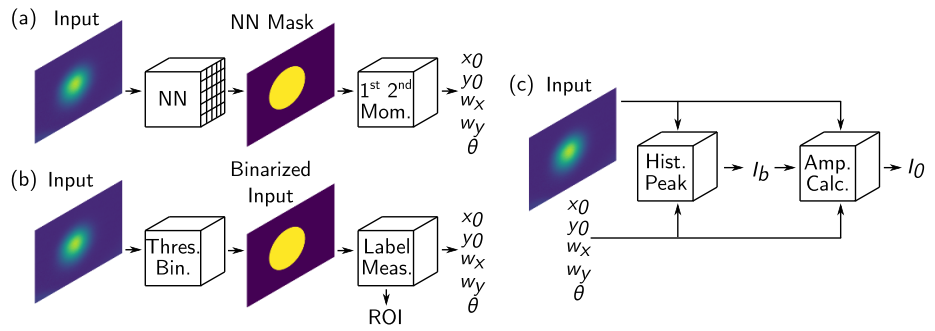
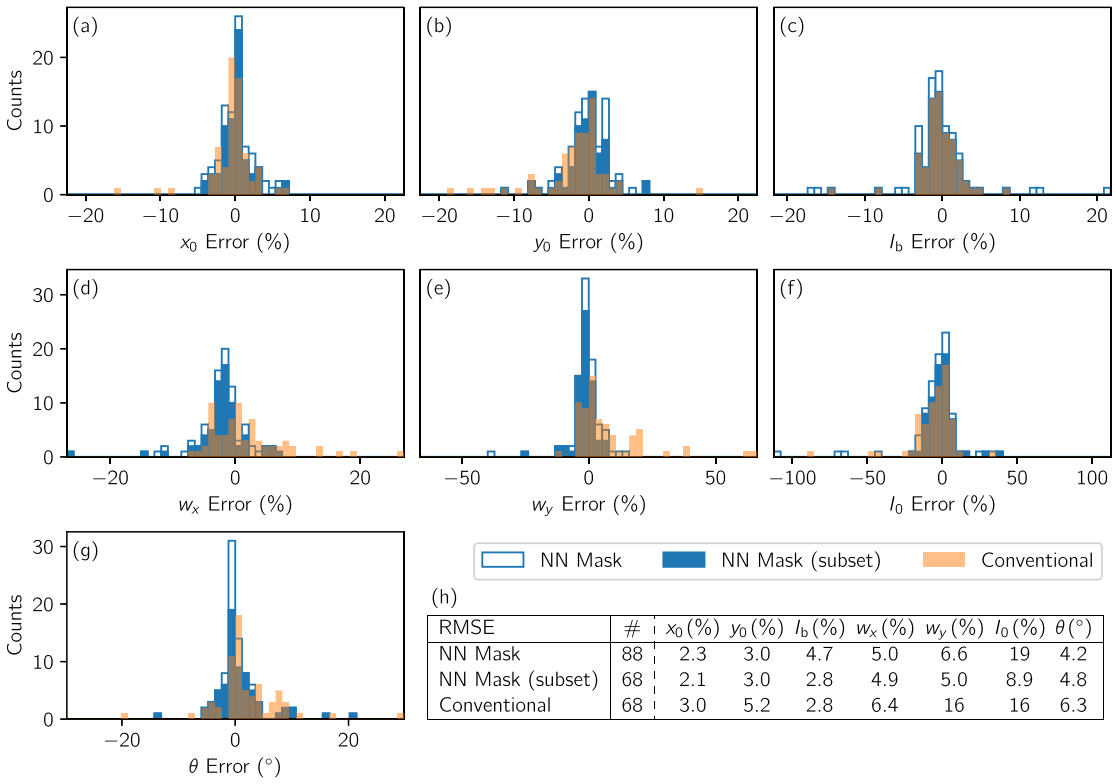


Figure 6. Extraction of Gaussian parameters. (a) NN extraction of the parameters is performed by first inputting the experimental image into the NN which returns a segmentation mask for each atom cloud. Applying the first and second moments (Mom.) directly to the segmentation mask yields the center coordinates of the atom cloud $\{x_0, y_0\}$, its $1/e^2$ radii $\{w_x, w_y\}$ and its angular orientation θ. (b) For conventional extraction the experimental image is binarized using a thresholding method (Thres. Bin.) and the binarized image regions labelled and measured (Label Meas.) using Scikit-image's regionprops function to yield the ROI as well as $\{x_0, y_0, w_x, w_y, \theta\}$. (c) Extracting the intensity parameters. The intensity offset I_b is determined by taking the intensity level corresponding to the peak of a histogram of the experimental image pixel values (hist. peak). The peak intensity I_0 is then calculated (see equation (3)) using the power inside the $1/e^2$ contour and the previously extracted parameters $\{x_0, y_0, w_x, w_y, \theta, I_b\}$.



(h)

RMSE	#	x_0 (%)	y_0 (%)	I_b (%)	w_x (%)	w_y (%)	I_0 (%)	θ (°)
NN Mask	88	2.3	3.0	4.7	5.0	6.6	19	4.2
NN Mask (subset)	68	2.1	3.0	2.8	4.9	5.0	8.9	4.8
Conventional	68	3.0	5.2	2.8	6.4	16	16	6.3

Figure 7. Gaussian parameter errors. (a)–(g) Histograms of the extracted Gaussian parameter errors—calculated by comparison to the 2D fit values—for the atom clouds in the validation dataset. Both the relative errors for the neural network segmentation mask method (subset, filled blue) and the conventional method (orange) are shown for the 68 clouds which the conventional method locates. The relative errors for all 88 clouds the NN method locates are also shown (outlined blue). Cloud center coordinates $\{x_0, y_0\}$ and cloud radii $\{w_x, w_y\}$ are normalized by the fitted cloud radii. The peak amplitude and the background intensity $\{I_0, I_b\}$ are both normalized by the fitted peak amplitude. (h) The root-mean-square error (RMSE) of the Gaussian parameters and the number of clouds analyzed (#).

fitting time. Without a seed, the fit time for a full image (1936 × 1216 pixels) is approximately 15 s on an Intel Xeon 2.2 GHz processor core. For the validation dataset, this is reduced to an average of 3 s when using the ROIs proposed by the NN—expanded by a factor of two except where the expanded ROIs would overlap—which substantially increases the fit speed despite the NNs processing time of 0.17 s per image using a Nvidia Tesla V100 GPU. Extracting the Gaussian parameters takes an average of 0.18 s, but results in an average fit speed up of 2 s per region when used as a seed. Thus, our method of determining ROIs and seed parameters using a NN offers a significant speedup in conjunction with fitting.

6. Comparison with conventional analysis

For comparison with the NN, a conventional method is used to find the ROIs and Gaussian parameters of the clouds (see figure 6(b)). We utilize standard methods based on Python's Scikit-image (Skimage) package [21] to pre-process the images, determine ROIs for the clouds, and extract the Gaussian parameters.

During pre-processing, a histogram is taken of the image's pixel intensities. As the cloud areas are much smaller than that of the overall image an approximately Gaussian peak corresponding to the image's background level and noise is seen. The mean μ and standard deviation σ of a Gaussian fit to this peak are used to define a threshold value $I_{\text{thresh}} = \mu + 3\sigma$ [53] and the image is binarized by setting pixels with $I > I_{\text{thresh}}$ to one and the remainder to zero.

After pre-processing, connected component analysis [54] is applied to the binarized image which groups pixels of the same value together into 'regions' (e.g. an atomic cloud). These regions are fed into Skimage's regionprops method which returns both the ROI coordinates and the geometric parameters $\{x_0, y_0, w_x, w_y, \theta\}$ for each region. ROIs with an area $> \frac{1}{2}$ or $< \frac{1}{800}$ the image size (i.e. much larger or smaller than the size of the clouds under measurement) are removed and the two largest remaining ROIs—if there are more than one—are taken as the MOT and ODT ROIs since smaller regions generally coincide with remaining high noise regions.

Applying this method to the validation set locates 68 of the 89 clouds with an IoU $> 50\%$; this is significantly worse than the NN which locates 88 of the 89 clouds. The mAP is also calculated, but since the conventional method does not differentiate between MOT and ODT clouds, all the validation dataset regions are relabelled as 'cloud'. Even with this simplification, an mAP of only 11.1% is achieved—much lower than the NN's object detection mAP of 86.3%. On closer inspection, the conventional method fails on images with low SNR such as images with significant fringing.

To determine the Gaussian parameters of the 68 successfully bounded clouds we apply a numerical scaling factor (to account for our thresholding method [52]) to the $\{w_x, w_y\}$ returned from regionprops and then generate a binary mask from the resulting $\{x_0, y_0, w_x, w_y, \theta\}$ which is used to find the cloud's background intensity I_b and amplitude I_0 (see figure 6(c)) as in section 5. The parameters are normalized and compared to those from the 2D fit (see figure 7, orange bars). Since the conventional method only recognizes a subset of the clouds, we quantitatively compare against the NN segmentation method for the same cloud subset (shown as filled blue bars in figure 7); a direct comparison of the root mean squared errors is shown in figure 7(h).

When considering both object detection and Gaussian parameter extraction, the NN method significantly outperforms the conventional method. Additionally, the conventional method's efficacy depends on the SNR of the data, whereas the NN is more robust against high noise levels and fringing since it learns higher level features of the atom clouds. Furthermore, the conventional method requires manual determination of the best pre-processing procedures and therefore potentially needs further tuning for new data; however, when faced with new data the NN simply needs to be retrained with more labelled data, thus making it effective in a laboratory setting.

7. Conclusion

An instance segmentation NN (Mask R-CNN) was trained to identify ultracold atom clouds in MOTs and ODTs. The NN generates both a ROI and a segmentation mask for each cloud—corresponding to the cloud's $1/e^2$ radii—with a mAP of 86.3% and 85.8% on the ROI and the segmentation mask branches, respectively. We show that the Gaussian parameters describing the atom clouds' distributions can also be extracted directly from the segmentation masks. Both ROI determination and Gaussian parameter extraction via the NN are significantly more accurate than a conventional method based on Python's Scikit-image library.

With an appropriate training dataset these techniques could be applied to ultracold atom clouds in other traps such as optical lattices [55, 56] and box potentials [57]; they are also directly applicable to laser beam profiling and other machine vision applications which require analysis of one or more 2D Gaussian distributions.

In the future a custom NN could be created by adding a branch after the ROI alignment stage which would output the cloud parameters directly. This would enable the characterization of non-Gaussian density profiles, useful, for example, in the detection, identification and parameterization of the bimodal clouds seen when a Bose–Einstein condensate [58, 59] is present.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.5287/bodleian:Y56kydgmj. Data will be available from 7 January 2021.

Acknowledgments

We thank Elliot Bentine, Shu Ishida and Jirka Kučera for helpful discussions and comments on the manuscript. This work was supported by EPSRC Grant No. EP/P009565/1, the John Fell Oxford University Press (OUP) Research Fund and the Royal Society. M K acknowledges funding from Trinity College, University of Cambridge.

Conflict of interest

The authors declare no conflicts of interest.

ORCID iDs

Lucas R Hofer https://orcid.org/0000-0002-5526-587X
Péter Juhász https://orcid.org/0000-0002-5187-730X
Anna L Marchant https://orcid.org/0000-0002-6350-4842

References

[1] Lecun Y, Bottou L, Bengio Y and Haffner P 1998 *Proc. IEEE* **86** 2278–324
[2] Krizhevsky A, Sutskever I and Hinton G E 2012 *Advances in Neural Information Processing Systems* pp 1097–105
[3] Simonyan K and Zisserman A 2015 (arXiv:1409.1556)
[4] Girshick R, Donahue J, Darrell T and Malik J 2014 *IEEE Conf. Computer Vision and Pattern Recognition* pp 580–7
[5] Redmon J, Divvala S, Girshick R and Farhadi A 2016 *IEEE Conf. Computer Vision and Pattern Recognition* pp 779–88
[6] Long J, Shelhamer E and Darrell T 2015 *IEEE Conf. Computer Vision and Pattern Recognition* pp 3431–40
[7] Taigman Y, Yang M, Ranzato M and Wolf L 2014 *IEEE Conf. Computer Vision and Pattern Recognition* pp 1701–8
[8] Schroff F, Kalenichenko D and Philbin J 2015 *IEEE Conf. Computer Vision and Pattern Recognition* pp 815–23
[9] Hofer L R, Jones L W, Goedert J L and Dragone R V 2019 *J. Opt. Soc. Am.* A **36** 936–43
[10] Ch'Ng K, Carrasquilla J, Melko R G and Khatami E 2017 *Phys. Rev.* X **7** 031038
[11] Carrasquilla J and Melko R G 2017 *Nat. Phys.* **13** 431–4
[12] Seif A, Landsman K A, Linke N M, Figgatt C, Monroe C and Hafezi M 2018 *J. Phys. B: At. Mol. Opt. Phys.* **51** 174006
[13] Picard L R, Mark M J, Ferlaino F and van Bijnen R 2019 *Meas. Sci. Technol.* **31** 025201
[14] Ness G, Vainbaum A, Shkedrov C, Florshaim Y and Sagi Y 2020 *Phys. Rev. Appl.* **14** 014011
[15] Miles C, Bohrdt A, Wu R, Chiu C, Xu M, Ji G, Greiner M, Weinberger K Q, Demler E and Kim E A 2020 (arXiv:2011.03474)
[16] Raab E, Prentiss M, Cable A, Chu S and Pritchard D E 1987 *Phys. Rev. Lett.* **59** 2631
[17] Chu S, Bjorkholm J, Ashkin A and Cable A 1986 *Phys. Rev. Lett.* **57** 314
[18] Pethick C J and Smith H 2008 *Bose–Einstein Condensation in Dilute Gases* (Cambridge: Cambridge University Press)
[19] Lett P D, Watts R N, Westbrook C I, Phillips W D, Gould P L and Metcalf H J 1988 *Phys. Rev. Lett.* **61** 169
[20] Muessel W, Strobel H, Joos M, Nicklas E, Stroescu I, Tomkovič J, Hume D B and Oberthaler M K 2013 *Appl. Phys.* B **113** 69–73
[21] van der Walt S, Schönberger J L, Nunez-Iglesias J, Boulogne F, Warner J D, Yager N, Gouillart E, Yu T and (The Scikit-image Contributors) 2014 *PeerJ* **2** e453
[22] Shahriari B, Swersky K, Wang Z, Adams R P and de Freitas N 2016 *Proc. IEEE* **104** 148–75
[23] Snoek J, Larochelle H and Adams R P 2012 *Advances in Neural Information Processing Systems* vol 25 pp 2951–9
[24] Frisch A, Aikawa K, Mark M, Rietzler A, Schindler J, Zupanič E, Grimm R and Ferlaino F 2012 *Phys. Rev.* A **85** 051401
[25] Hofer L R, Krstajić M, Juhász P, Marchant A L and Smith R P 2021 *Atom Cloud Detection and Segmentation Using a Deep Neural Network (Dataset) Version 1* (https://doi.org/10.5287/bodleian:Y56kydgmj)
[26] He K, Gkioxari G, Dollár P and Girshick R 2017 *IEEE Int. Conf. Computer Vision* pp 2980–8
[27] He K, Zhang X, Ren S and Sun J 2016 *IEEE Conf. Computer Vision and Pattern Recognition* pp 770–8
[28] Zeiler M D and Fergus R 2014 *Computer Vision—ECCV* pp 818–33
[29] Ren S, He K, Girshick R and Sun J 2015 *Advances in Neural Information Processing Systems* vol 28 pp 91–9
[30] Hecht-Nielsen R 1989 *Int. 1989 Joint Conf. Neural Networks* vol 1 pp 593–605
[31] Zhao H, Gallo O, Frosio I and Kautz J 2017 *IEEE Trans. Comput. Imaging* **3** 47–57
[32] Feng Z, Kittler J, Awais M, Huber P and Wu X 2018 *IEEE Conf. Computer Vision and Pattern Recognition* pp 2235–45
[33] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[34] Keren G, Sabato S and Schuller B 2018 *IEEE Int. Conf. Data Mining* pp 227–36
[35] Bottou L 1999 *On-line Learning and Stochastic Approximations* (Cambridge: Cambridge University Press) pp 9–42
[36] Jastrzebski S, Kenton Z, Arpit D, Ballas N, Fischer A, Bengio Y and Storkey A 2018 (arXiv:1711.04623)
[37] Bengio Y 2012 *Practical Recommendations for Gradient-Based Training of Deep Architectures* (Berlin: Springer) pp 437–78
[38] Everingham M, Van Gool L, Williams C K, Winn J and Zisserman A 2010 *Int. J. Comput. Vis.* **88** 303–38
[39] Boyd K, Eng K H and Page C D 2013 *Joint European Conf. Machine Learning and Knowledge Discovery in Databases* pp 451–66
[40] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S and Schiele B 2016 *IEEE Conf. Computer Vision and Pattern Recognition* pp 3213–23

[41] Bisong E 2019 *Google Collaboratory* (Berlin: Springer) pp 59–64
[42] Paszke A *et al* 2017 *31st Conf. Neural Information Processing Systems*
[43] Lin T Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P and Zitnick C L 2014 *Computer Vision—ECCV* pp 740–55
[44] Yosinski J, Clune J, Bengio Y and Lipson H 2014 *Advances in Neural Information Processing Systems* vol 27 pp 3320–8
[45] Bergstra J and Bengio Y 2012 *J. Mach. Learn. Res.* **13** 281–305
[46] Eggensperger K, Feurer M, Hutter F, Bergstra J, Snoek J, Hoos H and Leyton-Brown K 2013 *NIPS Workshop on Bayesian Optimization in Theory and Practice* vol 10 p 3
[47] Frazier P I 2018 (arXiv:1807.02811)
[48] Brochu E, Cora V M and de Freitas N 2010 (arXiv:1012.2599)
[49] Sobol' I M 1967 *Zh. Vychisl. Mat. Mat. Fiz.* **7** 784–802
[50] Balandat M, Karrer B, Jiang D R, Daulton S, Letham B, Wilson A G and Bakshy E 2020 (arXiv:1910.06403)
[51] Redmon J and Farhadi A 2018 (arXiv:1804.02767)
[52] Hofer L R, Dragone R V and MacGregor A D 2017 *Opt. Eng., Bellingham* **56** 043110
[53] ISO 11146-3 2004 *Lasers and Laser-Related Equipment—Test Methods for Laser Beam Widths, Divergence Angles and Beam Propagation Ratios* (International Organization for Standardization)
[54] Fiorio C and Gustedt J 1996 *Theor. Comput. Sci.* **154** 165–81
[55] Bloch I 2005 *Nat. Phys.* **1** 23–30
[56] Viebahn K, Sbroscia M, Carter E, Yu J C and Schneider U 2019 *Phys. Rev. Lett.* **122** 110404
[57] Gaunt A L, Schmidutz T F, Gotlibovych I, Smith R P and Hadzibabic Z 2013 *Phys. Rev. Lett.* **110** 200406
[58] Anderson M H, Ensher J R, Matthews M R, Wieman C E and Cornell E A 1995 *Science* **269** 198–201
[59] Davis K B, Mewes M O, Andrews M R, van Druten N J, Durfee D S, Kurn D and Ketterle W 1995 *Phys. Rev. Lett.* **75** 3969