Scaling digital screen reading with one-shot learning and re-identification

James Charles, Stefano Bucciarelli, Roberto Cipolla
University of Cambridge, U.K
http://www.jjcvision.com/, s.bucciarelli@gmail.com, cipolla@eng.cam.ac.uk

Abstract

Using only a mobile phone app, our objective is to cheaply retro-fit digital meters (e.g blood pressure, blood glucose or industrial gauges) with 'smart' data transfer capabilities. Using the mobile phone camera we build an app to securely and accurately transcribe information from digital meter screens. Only a single labelled training image of a target meter is required to build a custom screen reading module. Here we show how this can scale to potentially hundreds of different meters by learning to recognising the meter type so that the reading module can be automatically selected. This makes the system very easy for a user who would need to scan multiple different meter types. To this end, we build a CNN based system which runs in real-time on mobile device with very high read accuracy and meter recognition. Our contributions include (i) a method of one-shot training by synthesis through domain shift reduction, (ii) a deep embedding network for scale, translation and rotation invariant re-identification of digital meters, (iii) a highly accurate and efficient mobile phone app for recognising and parsing digital meter screens and (iv) release of a new digital meter re-identification dataset.

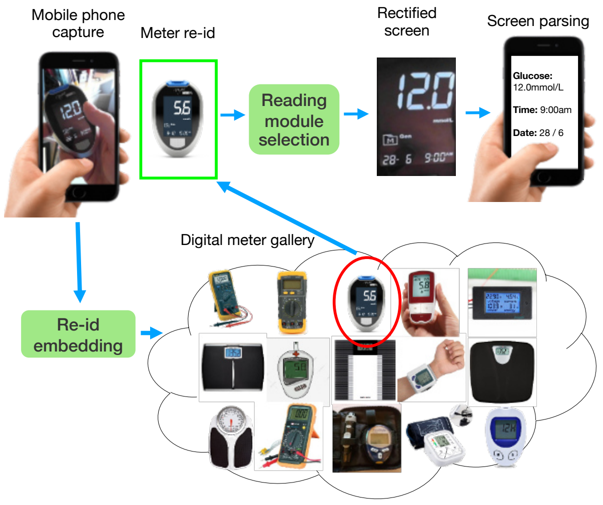


Figure 1. **Recognition and screen reading on mobile phone.** Our system can both recognise and parse the screen of a digital meter at near 100% read accuracy. The system is trainable using a single human labelled image consisting of a binary mask indicating screen and background pixels, and the four corner coordinates of the screen, see Figure 2. For meter recognition the system matches a deep re-id embedding to the best matching meter in a gallery of collected digital meters.

1. Introduction

A surprisingly high number of different digital meters are actively used by any one person on a day to day basis. Whether this be a scale to measure body weight or a thermometer to check temperature. In fact a large majority of these types of meters are for personal health monitoring *e.g.* blood pressure and blood glucose, *etc*. In clinical and industrial settings the number of digital meters escalates *e.g.* oximeter, spirometer and machine status monitors. With all this data, collection and analysis is valuable, but record keeping is still normally done manually. Not only is this time consuming, it is also prone to human entry error.

To combat this problem, modern meters are gradually becoming 'smart', meaning they wirelessly transmit data for remote analysis. This is typically done by pairing the meter with a moblie phone using Bluetooth. However, upgrading equipment to 'smart' capabilities is very costly, particularly in industrial or hospital settings. Furthermore, Bluetooth pairing is a rather slow and cumbersome operation.

We address failings of 'smart' meters and manual data entry and build a vision based system to reliably recognise meters and then read their screens using a mobile phone. Focus is on precise automatic reading, especially essential when handling medical data. To this end, we first recognise and detect the exact model of meter screen and precisely localise screen position. Only then is the display parsed. Tuning our recognition system to the exact target meter as in [3] means very strict validation schemes can be applied based on display type and screen digit positions, see Figure 1.

Unlike [3], we also show how to recognise the meter

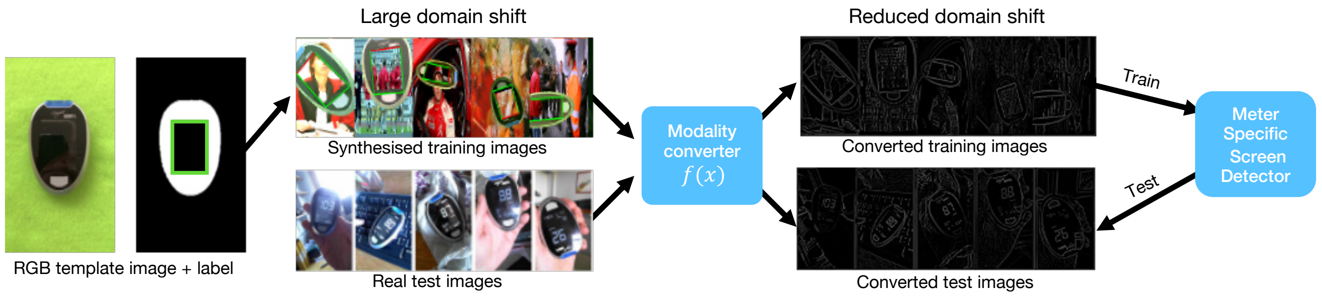

Large domain shift
Reduced domain shift
Synthesised training images
Real test images
RGB template image + label
Modality converter $f(x)$
Converted training images
Converted test images
Train
Test
Meter Specific Screen Detector

Figure 2. **Screen detection training with one-shot learning.** A single template image and training labels: binary mask indicating screen and background pixels, and the four corner coordinates of the screen (green rectangle), is used to train a meter specific screen detection network. The modality converter is applied to synthetic images at train time and real images at test time. Altering the function of the modality converter adjusts the domain shift between real and synthetic data.

make itself allowing the app to read multiple screen types without user guidance on the make/model of meter. We also show how to improve validation performance by using temporal information to check reading consistency between frames. This increases read precision to 100% on all meters in our dataset. The system is designed to be lightweight and runs in real-time at over 50fps on an iPhone 11, an improvement of 20fps over [3].

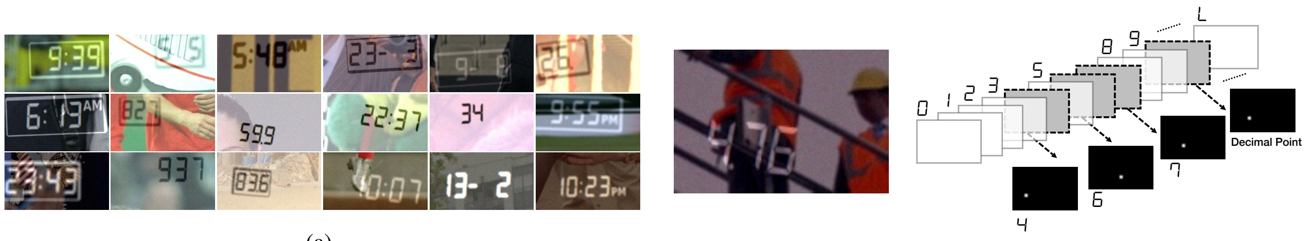
For each meter a reading module is trained to detect and parse the screen. Selection of the correct module is automatic and built from matching deep embeddings of digital meters. The embedding is scale, translation and rotation invariant which means a single training image is all that is necessary for recognition in the wild.

The reading module has to obtain precise screen coordinates of a particular model of meter. This is a challenging task for 4 main reasons: 1) screens are typically highly reflective and in some cases mirror like, 2) screens change appearance, 3) hands occlude many types of meters and 4) manually collecting enough ground truth data of screen corner positions to cover all variations in lighting, position, camera angle, screen appearance, and backgrounds is very restrictive. Because of this, a one-shot learning approach (*i.e.* only one real template image is labelled) is used to synthesise the training data by pasting a single labelled image of the target meter (under various transformations) on multiple different backgrounds. A simple image generation scheme such as this may seem unsophisticated when compared with modern simulation methods using 3D model rendering or using generative adversarial networks. Furthermore, it is often true that such training data does not produce models which generalise well to the real-world. However, using a 'modality converter' [3] one can use this coarse type of synthesis to train deep networks which now generalise to the real-world. Thus leveraging a very simple technique to generate enormous amounts of training data efficiently.

Using synthetic data for object detection is not new, similar methods to ours [7, 6] use in the order of hundreds of labelled images per object instance and also typically require some help from real training data. Here though, we show how only one manually labelled training image is necessary and how synthesis can be used without overfitting. An alternative approach to synthesising data is one-shot-learning methods [21, 13]. However, to date, these methods only provide image axis aligned bounding box detections [5] and do not natively support variation in object size, rotation and perspective transformation (essential for our task). Similarly tracking based approaches such as correlation filtering [2] or long term object trackers [14] (which can be initialised from a single image) do not consider rotation or perspective distortion of the bounding box. Keypoint matching to a single template image theoretically handles these constraints but in our experimental section this is shown to not work in practice.

Recognising text or numbers in generic scenes using neural networks has been addressed by a number of prior works using a two step approach of localisation then recognition [11, 8]. Recently some success has been found by incorporating both localisation and recognition into the same network [17]. In all cases however, localisation does not consider perspective distortion, resulting in systems which either a) fail due to oblique camera angles, or b) require larger networks to cope with larger variation of text.

Here domain shift from synthetic to real data is reduced using a modality converter, see Figure 2. Once trained, our models can be applied to real data for screen detection and perspective distortion correction. As such, a much simpler network for text recognition can be used, this too is trainable from completely synthetic data. Our screen detector assumes a CNN model which the modality converter is plugged into. The modality converter is agnostic to architecture and we evaluate the performance using MobileNet [10] and MaskRCNN [9]. We further show how re-identification can be used to classify meters in the wild, trained purely from the single template image.



(a) (b)

Figure 3. (a) **Example synthetic digit training data** for training the digit recognizer. (b) **Digit heatmaps.** Example synthetic digit image with corresponding goundtruth class heatmaps.

2. Method overview

Our system takes as input an image of the target meter, efficiently identifies the meter and then interprets the values contained on screen. It consists of three parts, (1) meter re-identification, (2) a meter specific screen detector which recovers the size, location and orientation of the meter screen, and (3) a digit recognizer which is applied to an image of the extracted and rectified screen at specific locations, see Figure 1 for an overview. Parts (2) and (3) are what we call the screen reading module. These three parts are processed using three separate deep convolutional neural networks (CNNs).

Meter re-identification. A scale, translation and rotation invariant embeddinng of the target meter is formed to quickly search and retrieve the meter type from an image gallery of meters for which there exist trained screen reading modules. Once a module is identified, the meter specific screen detector and digit recogniser can be applied to the input image.

Screen detection. The meter specific screen detector recovers the precise four coordinates of the digital meter screen corners, forming a quadrangle. Thus, assuming a planar surface of the screen, perspective correction is applied prior to digit recognition. The screen detector consists of a standard CNN backbone architecture, such as VGG16, ResNet or Inception, but with an additional block of layers at the input, which we call the *modality converter*. In order to explain the function of the modality converter we must first briefly describe how the screen detector is trained.

One-shot training. Training of the screen detector is accomplished using a single hand labelled template image. Labels consist of a foreground/background mask as well as locations of meter screen corners. This template is fed into our meter synthesizer which generates other examples of this meter under various homographic transformations, settings, lighting and reflections, please see Figure 2 for examples of blood glucose meter synthesis.

Modality converter. As the screen detector is trained from generated data there is a strong risk of overfitting or learning artefacts which do not transfer to real world images. The goal of the modality converter is to convert synthetic and real data into a common space where they are indistinguishable from one another. Other works have explored using adversarial training to learn how to transfer to a common feature space [24, 16] or adapt synthetic images so they look as real as possible [23, 19]. These approaches require large amounts of unsupervised data to capture the real data distributions. However, there are many real world applications where obtaining this amount of unlabelled data is impossible. For example, capturing the appearance space of a blood glucose meter under the full range of on screen values (which is part of meter appearance) would not be possible *i.e.* all times and dates and ranges of glucose would require an insurmountable number of blood tests. To overcome the limitations of adversarial training we propose to explore low level and efficient functions for the modality converter, such as edge based filters, colour removal, blurring and using pre-trained filters from a convolutional network. We show experimentally that by choosing the right function one can perform very well on our dataset of meters and surpass state-of-the-art object detectors trained without the added modality converter.

Digit recognition. The screen area is rectified and fed to the digit recognizer (another CNN) which extracts strings of characters and labels them according to type *e.g.* date or time. Validation of the strings is also performed. The digit recognition is customised to the meter based on the spatial location it is applied e.g. bottom left is date, bottom right is time.

3. Re-identification and Screen detection

Template image. The template image contains the target meter in a frontal facing pose with the screen surface normal pointing at the camera. Ground truth labels consist of the four coordinates of the meter screen in the training image, as well as the foreground/background mask. The label for

.

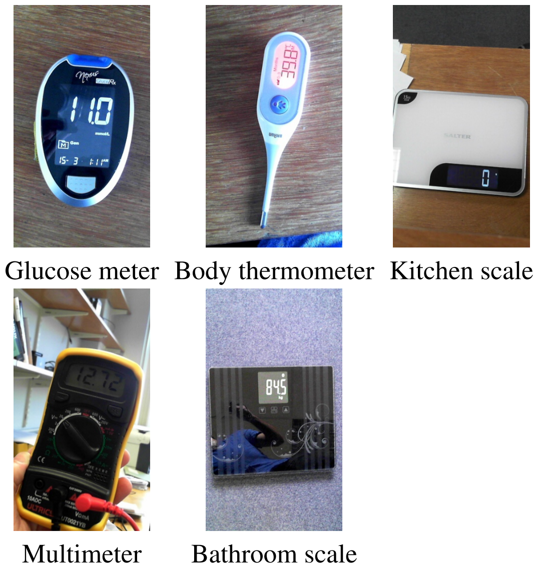


Figure 4. **Digital meters** used in our *Screen-detection* dataset. Screen reflection on these glossy displays severely inhibits screen detection and digit recognition.

background should also include the device screen, see Figure 2, so that during application the screen detector learns to ignore screen content.

Meter synthesis. The template image is placed on randomly chosen background images under various different rotations, scales and perspective distortions using alpha matting with the mask. Background images are taken from a dataset of images containing people. This is also appropriate for transfer to real world as person reflection on the meter screens is typical.

Re-identification. MobileNet [10] is trained to embed images of meters into a 128 length code (by replacing the last fully convolution layers and using linear activations). The network is trained from a set of 10,000 digital meter images downloaded from a google image using keywords such as "digital meter", "home health meters" etc. The network is trained in a self-supervised fashion by forcing it to match input images to themselves but under various affine transformations e.g. scaled, rotated and cropped. The network is trained using a batch size of 128, with the ADAM optimiser and learning rate of 0.0001 and a triplet loss. Triplets are mined during training by identifying "semi-hard" triplets. These are triplets where the negative is not closer to the anchor than the positive, but which still have positive loss [22]. During inference, the cosine similarity is applied to the embeddings to recall the first nearest neighbour from a gallery, see Figure 1.

Screen detection. For screen detection a backbone CNN connects to the modality converter (which is agnostic to the CNN backbone architecture) and performs the screen regression task. We use MobileNet [10] and replace the classification layer with a regression layer to output 8 values representing the four (x,y) coordinates of the screen and train using an L2 loss. During evaluation it is also shown how MaskRCNN [9] or ResNet50 can also be used as a backbone. We obtain a significant speed increase by replacing Relu6 activations in MobileNet with Relu, improving runtime performance from 30fps to 50fps on mobile phone.

4. Digit recognition

Preset regions of the rectified screens are associated with the type of text they contain *e.g.* date, time or weight. As the majority of meters have fixed areas for the type of digits displayed, they can initially be defined by hand on the template image. Characters within these regions are then detected individually using a small CNN which we aptly call LeDigit due to similarities with the LeNet5 [15] network from which it is based.

LeDigit. The network ingests a grayscale image and produces a heatmap for the location of each class of possible characters (19 classes comprising of: the digits 0-9, symbols ':', '.', '-', 'am', 'pm' and the letters 'L', 'o', 'H' and 'i' (used for certain meters in our dataset). LeDigit is derived from LeNet5 by converting the network into a fully convolutional version and changing the loss to L2 for heatmap regression. The receptive field of the network is 50x25 pixels, which is important to know when scaling the training data appropriately.

Training. The training data for LeDigit is completely synthetic. Various combinations of characters are generated, including dates and times, these are augmented in various ways (size, rotation, blurring, erosion/dilation, brightness inversion), then alpha blended with a random background image (see Figure 3 (a) for examples). Associated ground truth heatmaps (see Figure 3 (b)) per character type are artificially created by positioning a Gaussian kernel at the corresponding character centre. Digits are sized so as to be contained within the receptive field of the network.

Character string recognition and validation. Each output channel of LeDigit is thresholded to recover modes of high confidence and scored based on their sum. Modes below a threshold are removed. A line of best fit through remaining modes is found using RANSAC. All modes a set standard deviation from the line are removed. A string of characters are formed by sorting modes according to their class from left to right. These strings are validated using (regular expressions) based on their position in the screen

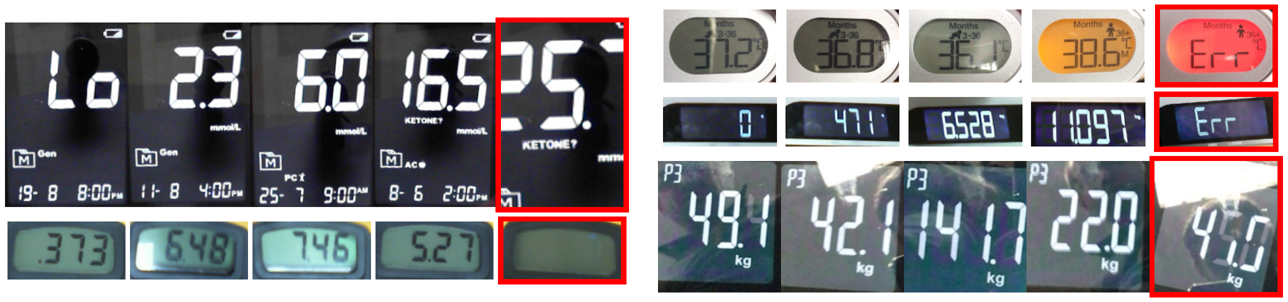


Figure 5. **Digit-recognition dataset.** 5 example screen snapshots from each of our digital meters is shown. First four columns are *valid* examples with the last column an *invalid* example (marked with red border). Notice the difficulty in recognizing the digits due to lighting and reflections.

and expected data type *e.g*. date, time or floating point numbers. Appropriate ranges are also checked.

5. Experiments

Two datasets are collected to separately evaluate the performance of the two aspects of our method (screen detection and digit recognition).

5.1. Datasets

Digital meters. Our dataset is sourced using 5 diverse digital meters consisting of a blood glucose monitor, kitchen scale, bathroom scale, digital multimeter and a digital body thermometer. Figure 4 illustrates each meter and their display.

Screen-detection dataset. This dataset consists of 30 images for each of the digital meters. All images were captured with a mobile phone aspect ratio of 9:16 at a size of 270 by 480 pixels. They contain varying background clutter, rotations, lighting, translations, scale and viewpoint and object occlusions. Please see supplementary material for example images. Screen corner coordinates are manually labelled.

Digit-recognition dataset. For of each of the 5 meters, rectified screen images are semi-automatically recorded by running a trained screen detector live on web-cam video and recording screen snapshots. Snapshots are recorded under varying conditions *e.g*. natural/artificial light from different angles and/or backlight turned on/off, meter rotation, scale and numerical readings, see Figure 5 for examples. Snapshots were manually labelled as *valid* if a reading is present, *i.e*. no null values or blank screens, and if all digits were visible. All other images were labelled as *invalid*, these include screens displaying nothing or null values, or images with out of shot digits. *Valid* images were manually tagged with their corresponding numerical readings. For the blood glucose meter, multiple tags are given *i.e*. glucose, time and date, for other meters one tag is sufficient. For the glucose meter 556 screen snapshots were recorded, and for all other meters 50 images per device were collected. Dataset details can be found in Table 2.

5.2. Screen detection evaluation protocol

We propose two measures for evaluating screen location against factors important for subsequent digit recognition, which is reliant on very precise localisation. These measures are *vertex difference* and *vertex alignment* which consider these factors by measuring the quality in detecting the corners of the screen.

Vertex difference. The predicted screen quadrangle and ground-truth (GT) quadrangle are first normalized by isotropically scaling them so the longest edge of the GT quadrangle is 100 pixels. Average pixel distances between corresponding vertices of the GT and predicted quadrangle is measured.

Vertex accuracy. A predicted quadrangle vertex is considered correctly detected if its *vertex distance* is below a set distance threshold d. Vertex accuracy is recorded as the percentage of correctly detected vertices overall all vertices in the test set.

Baseline methods. We compare against a keypoint matching system. Although a non deep learning based approach, we argue it is a challenging competitor. Unlike the state-of-the-art deep learning based object detectors, this system only needs one labelled training image, does not require any data augmentation and is robust to occlusions and screen changes. Keypoints (a mix of AKAZE [1], ORB [20] and Hessian affine [18]) are extracted from within an area of the template image containing only the meter (using the mask) and matched to keypoints in the test image. The homography is used to determine screen coordinates using LO-RANSAC [4]. We also compare against a ResNet50

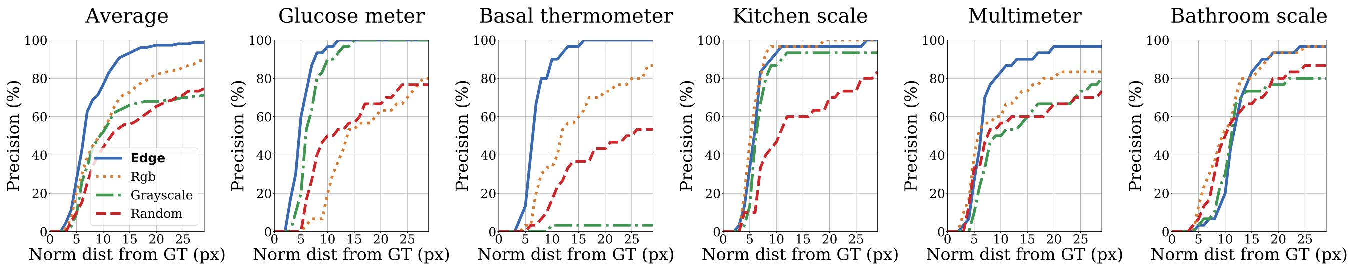


Figure 6. **Modality comparisons.** Precision curves (higher is better) of screen detection show percentage of all predicted screen corner points within a set distance from ground truth. Using an edge modality proves consistently good for all devices.

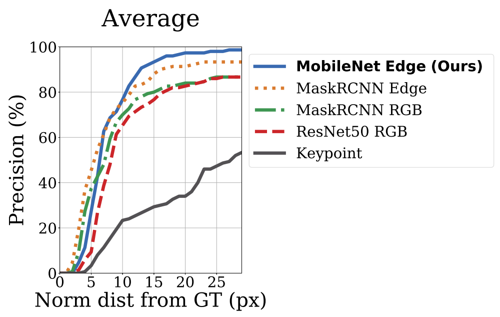


Figure 7. **Quantitative comparisons.** Screen detection precision curves (higher is better) is shown as an average over all meters (right). Our method performs best over all baselines.

network trained to regress screen coordinates and a MaskR-CNN network (with and without the modality converter) which is adapted to produce keypoints at screen corners. All CNN models are pre-trained on ImageNet.

5.3. Digit recognition evaluation protocol

If a recovered string passes validation (as described in section 4) and precisely matches the ground truth we count this as a successful read. Two types of evaluation are conducted:

Read accuracy. The percentage of successful readings from all those which were correctly validated. Because readings are only ever returned to the user when classed as valid, read accuracy can also be viewed as the probability of the system being correct in its returned reading.

Validation performance. Because a string which passes validation is more likely to be correct, one could artificially inflate the read accuracy (at the expense of a lower successful read return rate) by being very strict with validation. Therefore, we also measure the validation performance by quantifying the recognizer's ability in detecting if a presented meter reading is *valid* or *invalid*. Two metrics are used: *validation sensitivity* which measures the proportion of detected *valid* readings, and *validation specificity* which measures the proportion of detected *invalid* readings. Both of these metrics should be high for best performance.

5.4. Re-identification.

For each image in the screen detection dataset (across all 5 meters) we test the top 1 retrieval accuracy. The gallery consists of the 10,000 training images downloaded from a google image search mixed with out dataset. Our system achieved 96% accuracy. Most failures are due to a strong occlusion of the meter by e.g. a hand.

5.5. Screen detection

Image modality experiment. The performance of screen detection is evaluated under four settings of the modality converter using a MobileNet backbone CNN: (1) *RGB*: retains the RGB input completely, (2) *edge*: modality converter layers are constructed to produce the gradient magnitude image, (3) *grayscale*: an averaging 1x1 convolution is applied to the RGB channels and (4) *random*: modality converter consists of 3 layers of randomly initialised convolutional filters of size 3x3 and linear activations, which upscale and downscale the channels from 3 to 5 then back to 3. The random mode has the effect of average blurring the input. All models are trained with batch size 25, a learning rate of 0.005, 300,000 iterations and use the Adam optimizer. Figure 6 shows vertex accuracy for each meter plotted against various threshold levels d. Notice how using an *edge* modality produces consistently good performance across all meters. With an *RGB* modality more likely to overfit to the training data. *Grayscale* produces worse performance and *random* is the least best all round even though it helps blur edge artefacts left over through synthesis. Please see supplementary video illustrating the quality of the *edge* based model for screen detection in practice - these models never saw a single real image.

Baseline comparisons We test our mobile phone based model (MobileNet backbone with *edge* modality) against our baseline methods. Figure 7 shows average performance curves across all meters. The MaskRCNN detector is tested using an *RGB* and *edge* modality. The MaskRCNN network uses a ResNet50 backbone, which we test separately using only *RGB* modality. The ResNet50 backbone pro-

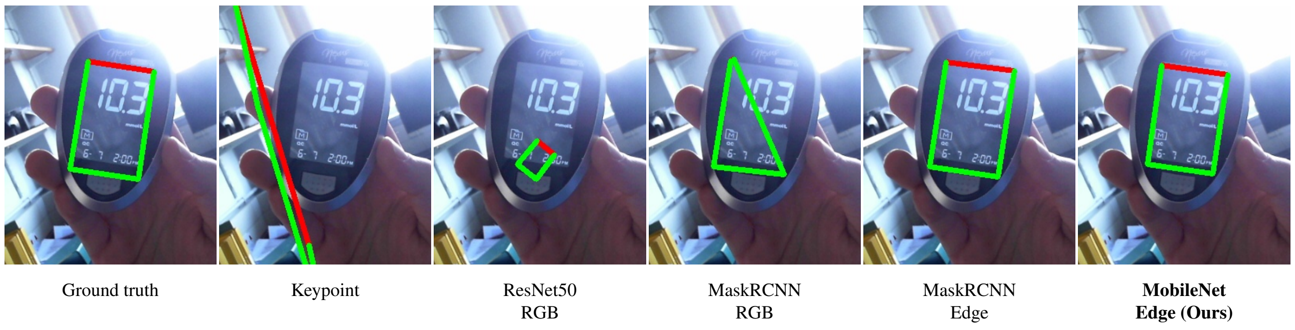


Figure 8. **Qualitative comparisons.** Examples of screen detections for blood glucose meter, quadrangles are shown as green polygons with the red edge indicating screen top. Comparison shown against baselines.

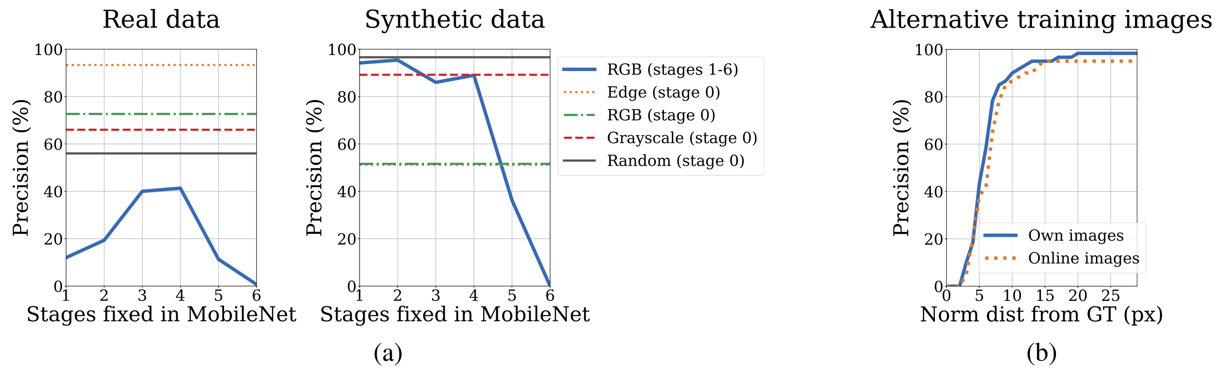


Figure 9. (a) **Staged training of MobileNet**, trained by fixing various stages of the ImageNet pre-trained network. RGB modality when fixing stages 1 to 6 shown as a solid blue line. Performance for having no fixed stages (stage 0) for the RGB and other modalities are shown as a dotted line. (b) **Alternative *vs* own training images.** Internet shopping sites images and our own images are compared when used as a template during training.

duces similar results to the MobileNet-RGB backbone (see Figure 6), indicating over-training occurs in both models. MaskRCNN-RGB produces mildly better results to these, due to its multi-modal output and use of non-maximum-suppression. MaskRCNN-Edge works better but still not as good as our network, this is because our network produces a structured quadrangle output, whereas the MaskRCNN approach independently predicts screen corner coordinates. The keypoint based method is least successful due to reflectance and textureless surfaces of the meters. Table 1 shows vertex accuracy at $d = 15$px from GT. MobileNet-Edge performs best with 93% accuracy and an average vertex difference of only 9px. A qualitative comparison against baselines is show in Figure 8 using the glucose meter and multimeter.

Fixed stage training. As an edge based modality works best, we explore the idea of using pre-trained filters in MobileNet (trained on ImageNet) as modality converters. Pre-trained filters close to the input are known to produce edge like features, hence we experiment with fixing various stages of MobileNet, up to the final global max pooling layer. Fixing up to six stages are tested in total. Each stage corresponds to a downsampling of the spatial dimension. Figure 9(a) shows the vertex accuracy at $d = 15$px *vs* the number of stages fixed. When tested on real data we unexpectedly observe fixing up to mid-way through the network to produce best results. However these results are consistently worse than all other tested modalities. When testing on synthetic data we observe a decrease in accuracy as stages are fixed. *Edge* and *RGB* modalities produce the worst results. This indicates that fixing pre-trained filters attenuates the domain shift early in the network causing over-training. Whereas an *edge* or *RGB* modality allows the network to regularise and reduces the domain shift, with the *edge* modality being best overall.

Alternative training images. In this experiment we explore training our screen detector (MobileNet -Edge) using out of domain template images of meters sourced from internet shopping sites. These images are normally taken under controlled lighting conditions and from a different camera to our own. Figure 9(b) demonstrates our method can be used to train from images scrapped from the internet with

Meter type	Keypoint	ResNet50 RGB	MaskRCNN RGB	MaskRCNN Edge	MobileNet Edge
Blood glucose	90%	50%	70%	100%	100%
Body thermometer	17%	97%	67%	90%	97%
Kitchen scale	17%	97%	100%	93%	97%
Multimeter	80%	60%	90%	97%	90%
Bathroom scale	33%	80%	73%	60%	83%
Avg. vertex accuracy	29%	77%	80%	88%	**93%**
Avg. vertex difference	171px	18px	14px	19px	**9px**

Table 1. **Screen detection performance**, measured using vertex accuracy at $d = 15$px from GT, per device. Average vertex accuracy along with average vertex difference is shown at the bottom of the table.

Meter type	# Images	# Valid	Range	Units	Validation sensitivity	Validation specificity	Read accuracy
Blood glucose - glucose	556	534	1.1—29.1	mmol/l	99.6%	92.9%	99.8%
Blood glucose - time	556	534	-	am/pm	99.6%	100%	99.8%
Blood glucose - date	556	534	-	NA	99.6%	100%	100%
Body thermometer	50	44	32.2—42.5	Celsius	93.6%	100%	95.5%
Kitchen scale	50	47	0—15.65	grams	93.6%	100%	100%
Multimeter	50	49	0—28.6	volts	100%	100%	98.0%
Bathroom Scale	50	48	18-95	kg	100%	100%	100%

Table 2. **Digit-recognition**. Dataset information and recognition results are shown.

very little loss in accuracy, showing the potential of rapidly training many meter readers.

5.6. Digit recognition

Static image performance. Analysis of our digit recognizer was conducted on the *digit-recognition* dataset. Results across all of the digital meters are shown in Table 2. The read accuracy is extremely good across all meters, with all of the meters having greater than 95% accuracy and very high validation sensitivity and specificity close to 100% for all meters. A comparison is formed against two state-of-the-art methods of [12], for reading text in the wild and [8] for reading Google street view door numbers. When applied to our test data they give very poor performance and are computationally slower in comparison to our LeDigit and validation system. The method of [12] executes in 30ms but does not give reasonable digit output which we believe is due to the original training set used. [8] executes in 80ms and ours is less than 5ms (our time includes detection,recognition and validation). The method of [8] requires a tight bounding box around the digits of interest. For four of our meters (all but the glucose meter) we manually annotate 200 tight bounding boxes for testing. [8] has an average read accuracy of 8% far lower than our network (we ignore missing symbols in evaluation as this method cannot detect them). [8] uses 7M parameters and ours only 150K.

Video based performance. Using the app opperating on a video stream, we set it to only validated a reading if after 10 consecutive frames a consensus of at least 8 read frames agree. Under these conditions, by pointing the phone camera at test images under various viewing angels the read accuracy improved to 100% on all images with 100% validation specificity and sensitivity.

5.7. Computational performance

Per video frame processing time when running the system on an iPhone 11 is: 0.6ms for the digit recognizer and 4ms for the screen detector. The overhead due to manipulating the video frame in memory (cropping, resizing, perspective transform) and rendering results in a processing speed of approximately 50fps. Example tracking by detection and recognition can be seen in the supplementary video.

6. Conclusion

Using a single labelled template image we show how one can build a mobile phone app to efficiently recognise and parse the screen of a digital meter. Custom meter screen reading modules give rise to very high validation and read accuracy performance, close to 100%. We demonstrate converting the augmented template image during training and real images at run time to edge images remarkable reduces domain shift, producing very good results. It was also shown how one can scale this approach so that multiple meters can be read without manual specification of the appropriate module. This employed a re-identification system for meter recognition which was shown to have high top 1 retrieval accuracy out of a dataset of 10,000 digital meters. The mobile phone framework is now also employed in a commercial setting by GlucoRx Limited for scanning glucose meter readings and helping with diabetes management of their over 250k patients.

References

[1] P.F. Alcantarilla and T. Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *PAMI*, 2011.

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

[3] J. Charles, S. Bucciarelli, and R. Cipolla. Real-time screen reading: reducing domain shift for one-shot learning. In *BMVC*, 2020.

[4] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, 2003.

[5] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[6] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, 2017.

[7] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. *arXiv preprint arXiv:1702.07836*, 2017.

[8] I. Goodfellow, et. al. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

[9] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[10] A. G Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 2016.

[12] M. Jaderberg, et. al. Deep features for text spotting. In *ECCV*, 2014.

[13] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015.

[14] M Kristan, et. al. The sixth visual object tracking vot2018 challenge results. In *VOT2018 workshop ECCV*, 2018.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[16] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NIPS*, 2016.

[17] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. Fots: Fast oriented text spotting with a unified network. In *CVPR*, 2018.

[18] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, 2002.

[19] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *CVPR*, 2018.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011.

[21] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[23] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[24] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.